

EpiData Software for Operations Research in Tuberculosis Control

A course developed by the EpiData Association



Authors:

Hans L **Rieder**, Kirchlintach, Switzerland

Jens M **Lauritsen**, Odense, Denmark

The input with major contributions to current content and recent revisions is greatly acknowledged:

Nguyen Binh **Hoa**, Hanoi, Viet Nam

Ajay M V **Kumar**, Bengaluru, India

Zaw Myo Tun, Singapore

Date last revised: April 26, 2018

Note on versions of EpiData software: Always use the most recent release version of EpiData software which can be obtained freely from the EpiData website <http://www.epidata.dk>. The course is updated at least whenever a new version requires adaptation or following an in-class course, whichever comes earlier.

Course content

Part A: Quality-assured data capture with EpiData Manager and EpiData EntryClient

- Exercise 1 A data documentation sheet for a simple questionnaire
- Exercise 2 Create a basic data entry form
- Exercise 3 Create a value-label pair from external data
- Exercise 4 Create a composite identifier
- Exercise 5 Data entry and validation
- Exercise 6 Upgrading an EpiData 3.1 REC/CHK file pair to an EPX file
- Exercise 7 Relational database

Part B: EpiData Analysis

- Exercise 1 An introduction to EpiData Analysis
- Exercise 2 More on EpiData Analysis
- Exercise 3 Aggregating data and saving the summary data in a file
- Exercise 4 From a spreadsheet to an EpiData file

Part C: Operations research

- Exercise 1: Creating a working dataset
- Exercise 2: Variability in serial smears
- Exercise 3: Incremental yield from serial smears
- Exercise 4: Confirmatory results in serial smears

Part D: More on EpiData software

- Exercise 1: Relational database and aggregating vs from “Long-to-wide”
- Exercise 2: A statistical process control chart
- Exercise 3: A simplified survival analysis
- Exercise 4: Creating a menu for standard reports
- Exercise 5: Formatting standardized analysis output in a spreadsheet

Part E: Beyond EpiData Analysis using R

- Exercise 1: Introduction to R software: basics
- Exercise 2: Introduction to R software: data bases and functions
- Exercise 3: Multivariable analysis in R part 1: A logistic regression
- Exercise 4: Multivariable analysis in R part 2: A Cox proportional hazard model

Background, objective, and course history

Background

In the 1980ies, the United States Centers for Disease Control and Prevention (CDC) developed public health software for its Epidemic Intelligence Service. The software package was called Epi Info and, later supported and promoted by the World Health Organization, experienced an unprecedented spread throughout the world's public health community.

The initiative to make EpiData was taken by Jens M Lauritsen from Denmark. Initially it was conceived of as part of the "Initiative for Accident Prevention" at Funen County. Why was it necessary to develop a new data entry program, if Epi Info version 6 had all that was needed in terms of control of data entry and simplicity? With the development of the Windows[®] operating system the majority of current computer users found it increasingly harder to cope with the DOS[®] operating system of working in Epi Info. Furthermore, with the change in the operating system of computers to newer versions of the Windows[®] operating system, the Epi Info 6.04d based on the DOS[®] operating system was eventually becoming obsolete (Windows 7[®] is no more reading it). Nevertheless, the principles underlying Epi Info are fundamentally sound and needed to be preserved for the new generation of public health practitioners grown up with the Windows[®] operating system.

On the Epi Info discussion list there were some discussions on strategies around 1997-1998, when the Epi Info team at CDC in USA decided to make an updated Epi Info version 2000. The updated Epi Info applies a different strategy in using a completely new way of working and basing it on the Microsoft Access[®] database format instead of simple text files.

Commercially available proprietary software is not focused on documentation, simplicity of use and validation of double-entered data. EpiData Entry is a program with a focus on data entry. The ambition of EpiData Entry was to create a simple to use independent application, which would not interfere with or require any special database system drivers (dll-based routines) shared with or interfering with other applications.

EpiData software consisted of three modules, EpiData Manager, EpiData EntryClient, and EpiData Analysis. The predecessor of EpiData Manager and EntryClient, EpiData Entry 3.1., followed the principles of Epi Info 6 with the so-called QES, REC, and CHK triplet, where the *.qes file defined the data entry form structure with variable names, labels, type, and length. This information was then passed on to the actual data base structure that would be used for data entry, i.e. the *.rec file. The *.chk file controlled data entry and contained all metadata.

By the 2010s it had become clear that an increasingly larger proportion of computer users had become visually oriented and that systems requiring some comprehension of underlying DOS principles became increasingly difficult to grasp for the regular user. In that vain, a system based on three file types started to overtax some users. It was thus decided to revamp entirely the system for data entry. There would be two software components, one reserved for the person designing and controlling the data entry form, the EpiData Manager software, and the other independent for the data entry person, called the EpiData EntryClient software.

For the EpiData Manager and EntryClient software, a single file is used and, as before, it is a text file but written in a specially adapted language (XML) belonging to the family of languages used for the internet. The language allows field names to be of any type, i.e. instead of English characters only, they can also be e.g. in Chinese or Arabic characters. The same goes for the values of text fields. The software comes in six flavors for the three

operating systems Windows, Mac, and Linux, and each for computers with a 32- and 64-bit architecture.

The development towards the replacement of EpiData Entry 3.1 with EpiData Manager and EntryClient had gone on for about three years of intensive work until it reached a new level of maturity in spring 2015 with release version 2. While some aspects that had been available in its predecessor are still in the pipeline, the new software is considered a full and equivalent replacement of the older one.

In this course you will be learning the use of both EpiData Manager and EpiData EntryClient (Part A) and EpiData Analysis (Part B), and to apply them to operations research (Part C), starting with its very basic functionality to increasing sophistication, and finally extend your programming skills in EpiData Analysis (Part D). The final module concludes with an introduction to R software with the examples of two specific multivariate tasks that exceed the boundaries of EpiData Analysis.

The course is designed and build for PC computers, i.e. for machines using the Windows® operating system. As the principles are the same, most material should be usable and adaptable for the Linux® or Mac OS X® systems. While EpiData Manager and EpiData EntryClient are available for all three operating systems, EpiData Analysis is, for the time being, a Windows® software program only.

Course Objective:

Acquire the skills to capture research study data of high quality, supported by thorough documentation of every procedure, conduct basic analyses, and apply them to operations research.

History of the course

The data for the original course were collected in a study simultaneously conducted in Benin, Malawi, Nicaragua, and Senegal.

The persons who collaborated in study design, data collection, analysis, and writing of a report were Séverin **Anagonou** (Benin), Thuridur **Arnadottir** (The Union), Fatoumata **Ba** (Senegal), Awa Héléne **Diop** (Senegal), Donald A **Enarson** (The Union), Martin **Gninafon** (Benin), A C **Kasalika** (Malawi), Hans L **Rieder** (The Union), Tone **Ringdal** (The Union), Felix L M **Salaniponi** (Malawi), Alejandro A **Tardencilla Gutierrez** (Nicaragua) and Arnaud **Trébucq** (The Union).

Utilizing the data generated in this collaborative research project, **three courses** of two weeks duration each **from 1997 to 1999** with a total of 18 participants were conducted in Paris by The Union. Subsidies to defray costs of these three courses were borne by the United States Centers for Disease Control and Prevention, the Coopération Française, the Norwegian Heart and Lung Association, the International Organization for Migration, and the Korean Institute of Tuberculosis.

The **first course** was held in **April 1997** in Paris, France. The participants were Francis **Adatu-Engwau** (Uganda), Nora **Bonso-Bruce** (Ghana), Awa Héléne **Diop** (Senegal), Asma **Elsony** (Sudan), and Amina **Jindani** (The Union). The facilitators were Thuridur **Arnadottir**, Eric **Brenner** (USA), Lawrence J **Geiter** (The Union), Hans L **Rieder** (The Union), and Arnaud **Trébucq** (The Union).

The **second course** was held in **April 1998** in Paris, France. The participants were Mohammed **Akhtar** (Pakistan), Maurice **Andriamiandrisoa** (Madagascar), Manfred **Danilovits** (Estonia), Lew Woo Jin (Korea, Republic of), Nguyen Phuong **Hoa** (Vietnam), Shanta Bahadur **Pande** (Nepal), and Alejandro A **Tardencilla Gutierrez** (Nicaragua). The facilitators were Thuridur **Arnadottir** (The Union), Eric **Brenner** (USA), Lawrence J **Geiter** (The Union), Hans L **Rieder** (The Union), and Arnaud **Trébucq** (The Union).

The **third course** was held in **April 1999** in Paris, France. The participants were Ademir **de Albuquerque Gomes** (Brazil), Fatoumata **Ba** (Senegal), Ferdinand **Kassa** (Benin), Pushpa **Malla** (Nepal), Tieng Sivanna (Cambodia), and Wang Jie-Siu (China), and. The facilitators were Thuridur **Arnadottir** (The Union), Eric **Brenner** (USA), Lawrence J **Geiter** (The Union), Hans L **Rieder** (The Union), and Arnaud **Trébucq** (The Union).

Because of the large costs associated with conducting the course, the course content and format was tested through interactive distance learning during the year 2000 by email. We are particularly indebted to Panganai **Dhliwayo** (Zimbabwe) and Robert **Makombe** (Zimbabwe) who went through this course in their free time. This helped in clarifying ambiguities and removing errors and uploading the course in September 2000 to the Internet (<http://www.tbrieder.org>).

The **fourth course** was held in Addis Ababa, Ethiopia, in **April 2001**. The participants were Ahmed **Abdurehman** (Ethiopia), Jemal **Aliy** (Ethiopia), Mekdes Gebeyehu **Ayicheh** (Ethiopia), Abebe **Habte** (Ethiopia), Yohannes **Mengistu** (Ethiopia), Moustapha **Ndir** (Senegal), Serkalem **Tadesse** (Ethiopia), Betru **Tekle** (Ethiopia), Mohammed Ahmed **Yassin** (Ethiopia), and Getachew Eyob **Yitelelu** (Ethiopia). The facilitators were Nils E **Billo** (The Union), Panganai **Dhliwayo** (Zimbabwe), and Hans L **Rieder** (The Union).

In 2002, the course material was once more revised to replace Epi Info 6 with EpiData Entry for questionnaire design, data checks, data entry, and data validation. For data analysis the 6.04d DOS version of Epi Info was retained.

The **fifth course** was held in Paris, France, in **January 2003**. The participants were Nadia **Aït-Khaled** (The Union), Kya Jai Maug **Aung** (Bangladesh), **Chiang** Chen-Yuan (Taiwan), Paula I **Fujiwara** (The Union), Achilles **Katamba** (Uganda), **Kim** HeeJin (Korea, Republic of), Dumitru **Laticevschi** (Moldova), Jones **Michongwe** (Malawi) and Jotam G **Pasipanodya** (Zimbabwe). The facilitators were Panganai **Dhliwayo** (Zimbabwe), Robert **Makombe** (Zimbabwe), and Hans L **Rieder** (The Union).

The **sixth course** was held in Yangon, Myanmar, in **December 2003**. The participants were Nyein Nyein **Aye** (Myanmar), May Yee **Chan** (Myanmar), Tin Mi Mi **Khaing** (Myanmar), Thandar **Lwin** (Myanmar), Htar Htar **Oo** (Myanmar), Saw **Thein** (Myanmar), Ti **Ti** (Myanmar), Myo **Zaw** (Myanmar), and Thin Thin **Yee** (Myanmar). The facilitators were **Chiang** Chen-Yuan (The Union), Hans L **Rieder** (The Union), and I D **Rusen** (Canada). In all previous courses the hypothesis was to test whether The Union's assumption that ten suspects needed to be examined to identify one case of tuberculosis was applicable to the study area. During the course it emerged, however, that even refutation of the hypothesis had relatively little programmatic implication. It was thus decided to integrate the previously supplementary exercises (dealing with variability of positive findings, patterns of recorded results, and potential incremental yield) into one single hypothesis addressing the critical value of the number of smear examinations that may be justified to identify one additional case or failure on the third diagnostic or the second follow-up smear examination, respectively.

The **seventh course** was held in Paris, France, in **January 2004**. The participants were Wafaa Hassan **Ali Taha** (Sudan), Goar **Balasanants** (Russia), Nulda **Beyers** (South Africa), Kathy **Lawrence** (South Africa), Biggie **Mabaera** (Zimbabwe), Nymadawaa **Naranbat** (Mongolia), Mahshid **Nasehi** (Iran), Mauro **Occhi** (Mozambique), Yevgeniy **Shubin** (Russia), and Abigail **Wright** (World Health Organization). The facilitators were Panganai **Dhliwayo** (Zimbabwe), Jens M **Lauritsen** (EpiData Association, Denmark), Robert **Makombe** (Zimbabwe), and Hans L **Rieder** (The Union).

The **eighth course** was held in Berne, Switzerland, in **July 2004**. The participants were Sondhja **Bitter** (Switzerland), Eva **Blozik** (Switzerland), Lorenz **Borer** (Switzerland), Michael **Endrich** (Switzerland), Karin **Imoberdorf-Baumgartner** (Switzerland), Caroline **Keller** (Switzerland), Hansjörg **Lüthy** (Switzerland), Christoph **Pammer** (Austria), Sabine **Recker** (Switzerland), Kurt **Schmidlin** (Switzerland), Jan **Wagner** (Switzerland), Sabine **Walser** (Austria), and Mark **Witschi** (Benin). The facilitators were Hans L **Rieder** (The Union) and Marcel **Zwahlen** (University of Berne, Switzerland). The curriculum of this course was changed in two important aspects from the previous courses. First, a core curriculum was developed to allow the conduct of the course within five working days without loss of any relevant course aspects. Refinements of specific aspects were offered in addenda independent of each other. This allowed course completion within one week (core curriculum) or eight days (core and extended curriculum). Second, the Epi Info Analysis component was replaced by a beta test version of EpiData Analysis which proved to be working smoothly without any glitches.

The **ninth course** was held in Paris, France, in **January 2005**. The participants were Anneke **Hesseling** (South Africa), **Hu** Dongmei (China), Zanele **Hwalima** (Zimbabwe), **Liu** Zhentian (China), Henri **Luwaga** (Uganda), Nguyen Thien **Huong** (Vietnam), John **Osho** (Nigeria), Tran Ngoc **Buu** (Vietnam), Nevin **Wilson** (The Union), and Yao Hongyan (China). The facilitators were Panganai **Dhliwayo** (The Union), Jens M **Lauritsen** (EpiData Association, Denmark), Robert **Makombe** (Zimbabwe), and Hans L **Rieder** (The Union). In this course, only free public domain software was used. Data management and analysis were done with

EpiData and EpiData Analysis, supplemented where needed with spreadsheet functions of OpenOffice.

The **tenth course** was held in Berne, Switzerland, in **July 2005**. The participants were Thomas **Bart** (Switzerland), Lisanne **Christen** (Switzerland), Stephanie **Christensen** (Switzerland), Michael **Flück** (Switzerland), Yvonne **Jansen** (Switzerland), Irène **Marty** (Switzerland), Jeanne **Moser** (Switzerland), Christoph **Napierala** (Switzerland), Barbara **Prokup** (Germany), Martin **Raab** (Switzerland), Franziska **Rabenschlag-Trösch** (Switzerland), Claudia **Sauerborn** (Switzerland), and Yasemin **Yüksel** (Switzerland). The facilitators were Hans L **Rieder** (The Union) and Marcel **Zwahlen** (University of Berne, Switzerland). This course used the same software as the January 2005 course with the latest pre-release version of EpiData Analysis (Version 0.9, Release 5, Build 26). Input from the participants and faculty led to further streamlining of some exercises.

The **eleventh course** was conducted in Paris, in **January 2006**. The participants were Chay Sokun (Cambodia), Andrew D R C **Dimba** (Malawi), Elhafiz **Hussein Ibrahim** (Sudan), Akramul **Islam** (Bangladesh), Suksont **Jittimane** (Thailand), Le Van **Duc** (Vietnam), Nguyen Binh **Hoa** (Vietnam), Helmuth **Reuter** (South Africa), Ezra **Shimeles** (The Union), and Xu Min (China). The facilitators were Achilles **Katamba** (Uganda), Jens M **Lauritsen** (EpiData Association, Denmark), and Hans L **Rieder** (The Union). In September 2005, the EpiData Association had released the stable EpiData Analysis Version 1.0. Between release and course commencement, the EpiData Association released upgrade Version 1.1 which added the last functionality (aggregate command) that had been possible previously only in Epi Info DOS Version 6.

During 2005 and 2006 the structure of the course was changed into three parts (EpiData Entry, EpiData Analysis, and Operations Research,). Parts A (EpiData Entry) and B (EpiData Analysis) were stripped of the operations research component for the benefit of those who do not have sufficient time available or who wish to familiarize themselves solely with the software. Conversely, Part C (Operations research) was stripped of components now introduced in Parts A and B, concentrating on the application of the latter to the example research project.

In **July 2006**, the **twelfth course** was conducted in Berne, Switzerland, for Master of Public Health students and other interested participants. These were Martin **Adam** (Switzerland), Edith **Betschart** (Switzerland), Tobias **Eckert** (Switzerland), Denise **Felber Dietrich** (Switzerland), T John **Kessler-Teuscher** (Switzerland), Esther **Kolb** (Switzerland), Brigitte **Kuenzle** (Switzerland), Sonia **Menéndez-González** (Switzerland), Stefan **Neuner-Jehle** (Switzerland), Christoph Paul **Röder** (Switzerland), and Hildebrand **Schwab** (Switzerland). The facilitators were Hans L **Rieder** (The Union) and Marcel **Zwahlen** (University of Berne, Switzerland).

In **August 2006**, the **thirteenth course** was given in Changsha, Hunan Province, China. All participants were from China. They were 范月玲 (**Fan Yueling**), 贾忠彬 (**Jia Zhongbin**), 阚晓宏 (**Kan Xiaohong**), 李晓凤 (**Li Xiaofeng**), 林岩 (**Lin Yan**), 邱柏红 (**Qiu Baihong**), 谭振 (**Tan Zhen**), 王铂 (**Wang Bo**), 汪清雅 (**Wang Qingya**), 张恩溥 (**Zhang Enpu**), 张宏伟 (**Zhang Hongwei**), 张会民 (**Zhang Huimin**), and 赵丁源 (**Zhao Dingyuan**). Facilitators were Chiang Chen-Yuan (The Union) and Hans L **Rieder** (The Union).

In **December 2006**, the **fourteenth** and **fifteenth courses** were given sequentially in Beijing, China. All participants were from China. In the thirteenth course the participants were 陈慧娟 (**Chen Huijuan**), 陈静 (**Chen Jing**), 陈伟 (**Chen Wei**), 房宏霞 (**Fang Hongxia**), 胡嘉 (**Hu Jia**), 梁路 (**Liang Lu**), 刘二勇 (**Liu Eryong**), 马斌忠 (**Ma Binzhong**), 王丹霞 (**Wang Danxia**), 吴顶峰 (**Wu**

Dingfeng), 于宝柱 (**Yu Baozhu**), and 张慧 (**Zhang Hui**). In the fourteenth course, the participants were 陈广华 (**Chen Guanghua**), 孔霞 (**Kong Xia**), 李曙光 (**Li Shuguang**), 罗丹 (**Luo Dan**), 马艳 (**Ma Yan**), 庞学文 (**Pang Xuewen**), 司马雅云 (**Sima Yayun**), 徐吉英 (**Xu Jiyin**), 依帕尔 (**Yi Pa'er**), 张广恩 (**Zhang Guang'en**), 赵津 (**Zhao Jin**), and 周扬 (**Zhou Yang**). Facilitators were Jens M **Lauritsen** (EpiData Association, Denmark), Biggie **Mabaera** (University of Zimbabwe, Zimbabwe), and Hans L **Rieder** (The Union), with the assistance of 胡冬梅 (**Hu Dongmei**), 徐敏 (**Xu Min**), and 张慧 (**Zhang Hui**).

In **June 2007**, the **sixteenth course** was given in Khartoum, Sudan. Two major changes were made to the course curriculum. The first change was that the database for Part C was changed to the utilization of the dataset collected as a course project of the January 2003 and 2004 courses (data courtesy: Dumitru **Laticevschi**, Moldova; Nymadawaa **Naranbat**, Mongolia; Achilles **Katamba**, Uganda; Biggie **Mabaera**, Zimbabwe). The second change was to use the test version 2.0 of EpiData Analysis. All participants were from Sudan. The participants were خديجة ادم محمد (Khadiga **Adam** Mohammed), اسرار محمد عبدالسلام (Asrar M A/Salam **Elegail**), معاذ سرالختم اسماعيل (Maaz Sier Elkhatim Ismail), حباب خالد الخير (Habab Khalid **Elkheir** Omer), أمل السمانى اسماعيل (Amel Elsammani Mohamed Ahmed **Elmuozamil**), مناضل حسن محمد علي (Monadil **Hassan** Mohamed Ali), سيد محمد همت (Sayed Mohammed Shareef **Himat**), ثويبة عمر علي محقر (Thoeiba Omer Ali **Muhagger**), عبدالمجيد عثمان موسى (Abdelmageed Osman **Musa**), علا محمود رحمة الله (Olla Mahmoud **Rahamtalla**) and عزمي عبدالرحمن عمارة (Azmi **Omara**). The coordinator was لوران علي زين العابدين (Louran **Zein Abdin Ali**, National Tuberculosis Programme Sudan), and facilitators were الحافظ حسين ابراهيم (Elhafiz Hussein **Ibrahim**, Epi-Lab, Sudan), and Hans L **Rieder** (The Union).

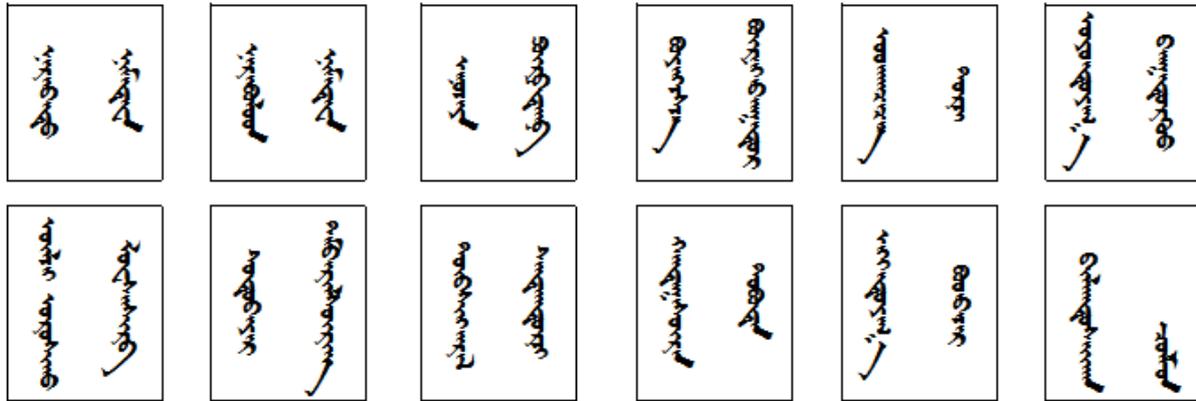
In **July 2007**, the **seventeenth course** was given in Berne, Switzerland, for Master of Public Health students and other interested participants. These were Nicole **Bender-Oser** (Switzerland), Bettina **Bringolf-Isler** (Switzerland), Sabina **Büttner** (Switzerland), Christian **Frei** (Switzerland), Florian **Gutzwiller** (Switzerland), Peter **Heri** (Switzerland), Kerstin **Hug** (Switzerland), André B **Kind** (Switzerland), Dimitri **Korol** (Switzerland), Cornelia **Marti** (Switzerland), Anne **Spaar** (Switzerland), and Françoise **Teuscher** (Switzerland). The facilitators were Hans L **Rieder** (The Union) and Marcel **Zwahlen** (University of Berne, Switzerland).

In **November 2007**, the course material was updated to make minor changes to Part A and to accommodate the changes introduced with the release Version 2.0 of EpiData Analysis.

In **April 2008**, the **eighteenth course** was given in Chisinau, Moldova. EpiData Version 3.1 (Build 270108) and EpiData Analysis Version 2.0 (Pre-release 2.1 Test Build 132) was utilized. The participants were Əlixanova Natəvan (**Alikhanova** Natavan), Azerbaijan; uCa nanava (Ucha **Nanava**), Georgia; maia qavTaraZe (Maia **Kavtaradze**), Georgia; Otilia **Scutelnicu**, Moldova; Angela **Capcelea**, Moldova; Rita **Seicas**, Moldova; Viorel **Soltan** Moldova; Ştefan **Savin**, Moldova; nino lomTaZe (Nino **Lomtadze**), Georgia; Constantin Dan **Nicolaiciu**, Romania; Iuliana **Husar**, Romania; Richard **Oleko** (Sudan). Facilitators were Jens M **Lauritsen** (EpiData Association, Denmark), Biggie **Mabaera** (Temporary Consultant to The Union, Lesotho), and Hans L **Rieder** (The Union).

In **June 2008**, the **nineteenth course** was given in Ulaanbaatar, Mongolia. EpiData Version 3.1 (Build 270108) and EpiData Analysis Version 2.0 (Pre-release 2.1 Test Build 132) were utilized. The participants were Пүрэвдагва Анузаяа (**Anuzaya** Purevdagva), Цолмон Билэгтсайхан (**Bilegtsaikhan** Tsolmon), Бүрнээбаатар Буянхишиг (Burneebaatar **Buyankhishig**), Бүдбазар Энхтуяа (Budbazar **Enkhtuya**), Довдон Хандаасүрэн (Dovdon **Khandaasuren**), N **Naranbold** (Н. Наранболд), Baatarxuu **Oyuntuya** (Баатархүү Оюунтуяа), Dorj **Otgontsetseg** (Дорж Отгонцэцэг), Demberelsuren **Sodbayar**

(Дэмбэрэлсүрэн Содбаяр), Sandagdorj **Tuvshingerel** (Сандагдорж Түвшингэрэл), Luvsansharav **Ulzii-Orshikh** (Лувсаншарав Өлзий-Орших). Hans L Rieder (The Union) was the facilitator. Following are the names of Nymadawa **Naranbat**, who organized the course, and the eleven participants written in the traditional Mongolian alphabet.



In **June / July 2008**, the **twentieth** course was given in Berne, Switzerland, for Master of Public Health students and other interested participants. EpiData Version 3.1 (Build 270108) and EpiData Analysis Version 2.0 (Pre-release 2.1 Test Build 132) were utilized. All participants were from Switzerland. They were Caroline **Bähler-Baumgartner**, Daniela **Dyntar**, Martin **Egger**, Carola A **Huber**, Ursula **Kälin-Keller**, Annette **Koller Doser**, Andrea **Merkel-Hoek**, Eric **Odenheimer**, Helen **Prytherch**, Anna **Späth**, and Melinda **Spießhofer**. The facilitators were Hans L **Rieder** (The Union) and Marcel **Zwahlen** (University of Berne, Switzerland).

In **October 2008**, the **twenty-first** course was given in Kampala, Uganda. Course I was given during this five-day course. It was specifically designed to fit this 5-day course, also incorporating the previously existing brief overview of EpiData software. EpiData Entry Version 3.1 (Build 270108) and EpiData Analysis Version 2.0 (Pre-release 2.1 Test Build 141) were utilized. The participants were Raymond **Asiimwe** (Uganda), Bernard Ssentalo **Bagaya** (Uganda), Freddy **Bwanga** (Uganda), Henry **Byabajungu** (Uganda), Basra Esmail **Doulla** (Tanzania), Nicholas **Ezati** (Uganda), Fred **Kangave** (Uganda), George **Lukyamuzi** (Uganda), Diana **Nadunga** (Uganda), and Raymond **Shirima** (Tanzania). Francis **Adatu-Engwau** (Uganda) was a guest participant, and formerly a participant in the first course. The facilitator was Hans L **Rieder** (The Union), supported in part by Achilles **Katamba** (Makerere University, Uganda, and The Union Country Office, Uganda).

In **February 2009**, the **twenty-second** course was given in Kampala, Uganda. Course II, Part B (EpiData Analysis) and Part C (Operations research) were given during this five-day course, although some exercises had to be skipped due to the brief duration of the course. EpiData Analysis Version 2.1 (Test Build 159) was utilized. The participants were Francis **Adatu-Engwau** (Uganda), Bernard Ssentalo **Bagaya** (Uganda), Freddy **Bwanga** (Uganda), Basra Esmail **Doulla** (Tanzania), Fred **Kangave** (Uganda), Diana **Nadunga** (Uganda), and Raymond **Shirima** (Tanzania). The facilitator was Hans L **Rieder** (The Union), supported in part by Achilles **Katamba** (Makerere University, Uganda, and The Union Country Office, Uganda).

In **July 2009**, the **twenty-third** course was given in Berne, Switzerland, for Master of Public Health students and other interested participants. EpiData Version 3.1 (Build 270108) and EpiData Analysis Version 2.2 (Build 169) were utilized. All but one participant from the principality of Liechtenstein were residents of Switzerland. They were Aline **Barbir**, Rita

Born, Mazda Farshad, Oliver Fuchs, Juliette Gerber, Gerti Kitting Gaillard, Meltem Kutlar Joss, Teresa Leisebach Minder, Elisabeth Oberfeld, Anna Plym, Corinna Rüegg, Dino Schlamp, Eliane Siegenthaler, Federico Soldati, and Esther Walser (Principality of Liechtenstein). The facilitators were Hans L **Rieder** (The Union) and Marcel **Zwahlen** (University of Berne, Switzerland).

In **October 2009**, the **twenty-fourth** course was given in Paris, France, for fellows and participants in the training module 2 offered by the Centre for Operational Research of The Union. EpiData Entry Version 3.1 (Build 270108) and EpiData Analysis Version 2.2.1.171 were utilized. The participants were Dawit **Assefa Lemma** (Ethiopia), Walter Kizito **Kibango** (Kenya), Proscovia Namuwenge **Mukonzo** (Uganda), Mweete Debra **Nglazi** (South Africa), **Nguyen Binh Hoa** (Viet Nam), Mahfuza **Rifat** (Bangladesh), Srinath **Satyanarayana** (India), Zaw Myo **Tun** (Myanmar), Hannock Mukoma **Tweya** (Malawi) and Susanna Sophia **Van Wyk** (South Africa). The facilitators were Hans L **Rieder** (The Union) and Anthony D **Harries** (The Union), assisted by Nguyen Binh **Hoa** (Fellow, Viet Nam).

In **March 2010**, the **twenty-fifth** course was given in Paris, France, for three selected fellows and participants in the training module 2 offered by the Centre for Operational Research of The Union. EpiData Entry Version 3.1 (Build 270108) and EpiData Analysis Version 2.2.1.171 were utilized. The course focused on Parts C (Operations Research) and D (More on EpiData software). The participants Nguyen Binh **Hoa** (Viet Nam), Zaw Myo **Tun** (Myanmar), and Hannock Mukoma **Tweya** (Malawi). The facilitators were Hans L **Rieder** (The Union) and Jens M **Laurtisen** (EpiData Association).

In **July 2010**, the **twenty-sixth** course was given in Berne, Switzerland, for Master of Public Health students and other interested participants. EpiData Version 3.1 (Build 270108) and EpiData Analysis Version 2.2 (Build 171) were utilized. All participants were residents of Switzerland. They were Brigitte **Brunner**, Adrian **Businger**, Elisabeth **Maurer Schild**, Rebecca **Osterwalder**, Alexandra **Rauch**, Charles **Senessie**, Ulf **Tölle**, and Roco **Umbescheidt**. The facilitators were Hans L **Rieder** (The Union) and Marcel **Zwahlen** (University of Berne, Switzerland).

In **October 2010**, the **twenty-seventh** course was given in Paris, France, for fellows and participants in the training module 2 offered by the Centre for Operational Research of The Union. EpiData Entry Version 3.1 (Build 270108) and EpiData Analysis Version 2.2.1.171 were utilized. The participants were Felix **Afutu** (Ghana), Sarabjit Singh **Chadha** (India), Karen **Du Preez** (South Africa), Razia **Fatima** (Pakistan), Oliver **Gadabu** (Malawi), Lucy **Guluka-Gawa** (Malawi), Mohammed **Kogali** (Sudan), Rose Jepchumba **Kosgei** (Kenya), Ajay M V **Kumar** (India), Tonderayi Clive **Murimwa** (Zimbabwe), Mauro **Niskier Sanchez** (Brazil), and Kudakwashe Collin **Takarinda** (Zimbabwe). The facilitators were Hans L **Rieder** (The Union), Nguyen Binh **Hoa** (Viet Nam), Zaw Myo **Tun** (Myanmar), and Sven Gudmund **Hinderaker** (The Union).

In **May 2011**, the **twenty-eighth** course was given in Paris, France, for two selected fellows from the training module 2 offered by the Centre for Operational Research of The Union. EpiData Entry Version 3.1 (Build 270108) and EpiData Analysis Version 2.2.1.171 were utilized. The course focused on Parts C (Operations Research) and D (More on EpiData software). The participants Karen **Du Preez** (South Africa) and Ajay M V **Kumar** (India). The facilitators were Hans L **Rieder** (The Union), Jens M **Laurtisen** (EpiData Association), and Hannock Mukoma **Tweya** (Malawi).

In **November 2011**, the **twenty-ninth** course was given in Paris, France, for fellows and participants in the training module 2 offered by the Centre for Operational Research of The

Union. EpiData Entry Version 3.1 (Build 270108) and EpiData Analysis Version 2.2.1.171 were utilized. The participants were Henry Shadreck **Kanyerere** (Malawi), Nicholas **Kirui** (Kenya), Pranay **Lal** (India), Sharath Bugurina **Nagaraja** (India), Sharan **Ram** (Fiji), Carlos Alberto **Mendoza-Ticono** (Peru), Emmanuel **Singogo** (Malawi), Nelda **van Soelen** (South Africa), Kerry **Viney** (New Caledonia), and Aung Naing **Win** (Myanmar). The facilitators were Hans L **Rieder** (The Union), Sarabjit **Chadha** (The Union), and Ajay M V **Kumar** (India).

In **February 2012**, the **thirtieth** course was given in Kathmandu, Nepal, for fellows and participants in the training module 1b offered by the Centre for Operational Research of South East Asia office of The Union. EpiData Entry Version 3.1 (Build 270108) and EpiData Analysis Version 2.2.1.178 were utilized. The participants were Tshering **Jamtsho** (Bhutan), Tashi **Denup** (Bhutan), Shyla **Islam** (Bangladesh), Wasantha **Jayakodi** (Sri Lanka), Swati **Srivastava** (India), Suresh **Shastri** (India), Ramya **Ananthakrishna** (India), Rajendra **Basnet** (Nepal), Caetano **Gusmao** (Timor Leste), Sokhan **Khann** (Cambodia), Sarwat **Shah** (Pakistan), Iwayan **Artwan** (Indonesia). The facilitators were Ajay M V **Kumar** (The Union, India) and Srinath **Satyanarayana** (The Union, India).

In **March 2012**, the **thirty-first** course was given in Bengaluru, India, for participants of the operational research course conducted by the Centre for Operational Research of South East Asia office of The Union in association with the National Tuberculosis Programme, United States Centre for Disease Control and Prevention, and the National Tuberculosis Institute. EpiData Entry Version 3.1 (Build 270108) was utilized. The participants (all residing in India and working for or in close association with the national tuberculosis programme) were Gurpeet **Singh**, Priyanka **Agrawal**, **Subhakar**, Raju **Chepuri**, Solomon **Muller Ankala**, Chetan **Purad**, D J **Deka**, Rajeev **Pathak**, Pankaj **Nimavat**, Amar **Shah**, Sunil **Kumar**, Karthickeyan **DSA**, Rajesh **Deshmukh**, Tapas **Patra**, Parija **Debudatta**, Asha **Fedrick**, Shivramakrishnan, Rajabhau D **Yeole**, Suparna **Chatopadhaya**, Shiv Kumar **Singh**, Vrinda **Sahasrabhojane**, **Anand** and **Tripathi**. The facilitators were Ajay M V **Kumar** (The Union, India) and Srinath **Satyanarayana** (The Union, India).

In **April 2012**, the **thirty-second** course was given in Paris, France, for three selected fellows from the training module 2 offered by the Centre for Operational Research of The Union. EpiData Entry Version 3.1 (Build 270108) and EpiData Analysis Version 2.2.1.178 were utilized. The course focused on Parts C (Operations Research) and D (More on EpiData software). The participants were Sarabjit Singh **Chadha** (India), Sharath **Burugina Nagaraja** (India), and Nicholas **Kirui** (Kenya). The facilitators were Hans L **Rieder** (The Union), Ajay M V **Kumar** (The Union, India), and Jens M **Laurtisen** (EpiData Association).

In **May 2012**, the **thirty-third** course was conducted in Goa, India, for about 60 or more WHO-hired consultants working for the national tuberculosis programme in India (plus a couple of observers from the WHO-India and WHO-SEARO office) to build their capacity in data management; they were in turn expected to train all the 650 and more data entry operators working for the national tuberculosis programme. EpiData Entry Version 3.1 (Build 270108) and EpiData Analysis Version 2.2.1.178 were utilized. The course duration was three days and focused on Parts A (Data Entry) and B (Data Analysis). The facilitators were Ajay M V **Kumar** (The Union, India), Sharath **Burugina Nagaraja** (India), Kiran **Rade** (India) and Srinath **Satyanarayana** (The Union, India).

In **May-June 2012**, the **thirty-fourth** course was given in Nairobi, Kenya, as a component of a course on data management for personnel working in tuberculosis reference laboratories, largely in Africa. EpiData Entry Version 3.1 (Build 270108) and EpiData Analysis Version

2.2.1.178 were utilized. The course focused on Parts A (Data Entry) and B (Data Analysis). The participants were Pamela Juma **Akinyi** (Kenya), Vignon Charles Frank **Faihun** (Bénin), Frederick **Kangave** (Uganda), Thuwein Yusuf **Makamba** (Tanzania), Faisal **Masood** (Pakistan), Margaret **Ndisha** (Kenya), Diana **Nadunga** (Uganda), Collins Tanaka **Sakubani** (Zimbabwe), Tigist Habtamu **Shiferaw** (Ethiopia), Elizabeth Ndaafetwa **Shipiki** (Namibia), Raymond Philip **Shirima** (Tanzania), and Mukururi **Sibanda** (Botswana). The facilitators were Hans L **Rieder** (The Union) and Armand **Van Deun** (The Union). Subsequent to the course, the structure of Part A (EpiData Entry) was adjusted to improve the flow into more coherent blocks.

In **July 2012**, the **thirty-fifth** course was given in Berne, Switzerland, for Master of Public Health students and other interested participants. EpiData Entry Version 3.1 (Build 270108) and EpiData Analysis Version 2.2 (Build 178) were utilized. The participants (all residing in Switzerland) were Lukas **Fenner**, Micòl **Gianinazzi**, Lotte **Habermann-Horstmeier**, Eva-Maria **Hau-Grosch**, Marianne **Jost**, Tanja **Löhri**, Thomas **Plattner**, Regina **Scharf**, Laura **Wengenroth**, Alfred **Wiesbauer**, and Natascha M **Wyss**. The facilitators were Hans L **Rieder** (The Union) and Marcel **Zwahlen** (University of Berne, Switzerland).

In **July 2012**, the **thirty-sixth** course was given in Paris, France, for fellows and participants in a training module offered by the Centre for Operational Research of The Union. EpiData Entry Version 3.1 (Build 270108) and EpiData Analysis Version 2.2.1.178 were utilized. The participants were Serge **Ade** (Benin), Fengling **Mi** (China), Emmanuel **Kanike** (Malawi), Bodrun **Siddiquea** (Bangladesh), Kang Yang **Lim** (Singapore), Prashant **Bhat** (India), Ganzaya **Sukhbaatar** (Mongolia), Hilary **Gunguwo** (Zimbabwe), Philip **Owiti** (Kenya), Eunice **Wahome** (Kenya), Christine **Njuguna** (South Africa) and Kesete **Yirdaw** (Ethiopia). The facilitators were Ajay M V **Kumar** (The Union, India), Nguyen Binh **Hoa** (Viet Nam) and Srinath **Satyanarayana** (The Union, India).

In **September 2012**, the **thirty-seventh** course was given in Chennai, India, for participants of the operational research course conducted by the South East Asia office of The Union in association with the National Institute for Research in Tuberculosis (erstwhile Tuberculosis Research Centre). EpiData Entry Version 3.1 (Build 270108) and EpiData Analysis Version 2.2.1.178 were utilized. The participants (all residing in and working for tuberculosis control in India) were Kovid **Sharma**, Jaya Prasad **Tripathy**, Jaiswal **Narendra**, Jigneswar **Patel**, Nanda **Kumar**, Sanat Kumar **Tripathi**, Kamal Chand **Naik**, Lord Wasim **Reza**, Priyakanta **Nayak**, Ramesh **Kumar**, **Palanivel C** and **Deepa D**. The facilitators were Ajay MV **Kumar** (The Union, India), Ramya **Ananthkrishnan** (REACH, India) and Amar **Shah** (WHO, India).

In **December 2012**, the **thirty-eighth** course was given in Nadi, Fiji Islands, for participants of the Pacific Operational Research Course conducted by the Centre for Operational Research of The Union and The Secretariat of the Pacific Community. EpiData Entry Version 3.1 (Build 270108) and EpiData Analysis Version 2.2.1.178 were utilized. The participants, from different countries of the Pacific region were Edwin Henry **Daiwo** (Solomon Islands), Rupihner **Defang** (Federated States of Micronesia), Saen **Fanai** (Vanuatu), Louise **Fonua** (Tonga), Natalie **Girin** (New Caledonia), Mareta **Hauma** (Republic of Marshall Islands), Noel **Itogo** (Solomon Islands), Tom **Jack** (Republic of Marshall Islands), Veisia **Matoto** (Tonga), Joaquin **Nasa** (Republic of Marshall Islands), Markleen **Tagaro** (Vanuatu) and Karen **Tairea** (Cook Islands). The facilitators were Ajay M V **Kumar** (The Union, South East Asia office and also module chair), Karen **Bissell** (The Union), Shakti **Gounder** (Fiji Ministry of Health), Bridget **Kool** (The University of Auckland, New Zealand), Sharan **Ram**

(Fiji National University), Christine **Roseveare** (EpiData Trainer, New Zealand) and Kerri Viney (Secretariat of the Pacific Community).

In **January 2013**, the **thirty-ninth** course on “Efficient, Quality assured Data capture using EpiData” was given in Bengaluru, India, for medical college professionals working in close collaboration with the Revised National TB Control Programme in Karnataka. This course was conducted by the South-East Asia office of The Union in association with the Bangalore Medical College and Research Institute. This course, the first of its kind in India, utilized EpiData Entry Version 3.1 (Build 270108). The participants were **Madhusudan M**, Mohammed **Imran**, Amit Kumar **Rao**, Arshiya **Kouser S**, Navya **CJ**, **Vinay M**, Nagaraja **Goud B**, **Ranganath TS**, **Ravish KS**, **Kishore SG**, **Sarsawathi S**, **Thilak SA**, **Shivaraj BM**, Riyaz **Basha S**, Nimmy **Dominic**, Ashok Kumar **Gudagunti**, Deepak **Tamang**, Surbhi **Joshi**, Karuna D **Sagili**, Archana **Trivedi**, Swaroop Kumar **Sahu**, Josephine **Priya K**, Suresh **Kumbhar**, **Anushka T**, Lilian **Dsouza**, Nevin **Wilson** and Sunita **Prasad**. The facilitators were Ajay M V **Kumar** (The Union, South East Asia office and also module chair), Balaji **Naik** (WHO-RNTCP, Bengaluru), **Deepak KG** (WHO-RNTCP, Tumkur), Prashant **Bhat** (WHO-RNTCP, Hubli) and **Palanivel** (IGMCRI, Puducherry).

In **February-March 2013**, the **fortieth** course was given in Kathmandu, Nepal, for participants of the Asian Operational Research Course conducted by the Centre for Operational Research of South-East Asia office of International Union Against Tuberculosis and Lung Disease (The Union). EpiData Entry Version 3.1 (Build 270108) and EpiData Analysis Version 2.2.2.180 were utilized. The participants were Mangal **Bahadur Tharu** (Nepal), Xia **Yinyin** (China), Yaping **Chang** (China), Lekey **Khandu** (Bhutan), Jakir Hossain **Bhuiyan Masud** (Bangladesh), Sumudu Chandana **Abeygunawardena** (Srilanka), Olga **Denisiuk** (Ukraine), Gabeena **Mamoon** (Pakistan), Satyavani (India), Mrinalini **Das** (India), Shankar **D** (India), Ravinder **Kumar** (India). The facilitators were Ajay M V **Kumar** (The Union, India), Swati **Srivastava** (PHFI, India), Balaji **Naik** (WHO-RNTCP, India), Karuna D **Sagili** (The Union, India) and Deepak **Tamang** (The Union, India).

In **March 2013**, the **forty-first** course on “Efficient, Quality-assured Data capture using EpiData” was given for medical college professionals and scientists from the Vector Control Research Centre working in Puducherry, India. This course was conducted by the South-East Asia office of The Union in association with the medical college Jawaharlal Institute of Postgraduate Medical Education and Research (JIPMER). This course utilized EpiData Entry Version 3.1 (Build 270108). The participants were **Murugan V**, Prasad **Dikale**, Kavita **Vasudevan**, Hemant **Shewade**, Johnson **Cherian**, Mahalakshmi, **Subitha L**, Anindo **Majumdar**, **Kalaiselvi**, **Vedapriya**, Anuj **Mittal**, Ramesh **Chauhan**, Madhan **Raj**, S **Srikanth**, E Susiganesh **Kumar**, R **Kanagarethinem**, R **Moorthy**, Yogesh A **Bahurupi**, Om Prakash **Bera**, Surendar **Rangaswamy**, Manoj Kumar **Panigrahi**, R **Srinivasan**, C **Sadanandane** and **Ganesh**. The facilitators were Ajay M V **Kumar** (The Union, South-East Asia office and also module chair), **Palanivel** (IGMCRI, Puducherry), Swaroop Kumar **Sahu** (JIPMER, Puducherry), Karuna D **Sagili** (The Union, India) and Deepak **Tamang** (The Union, India).

In **March 2013**, the **forty-second** course was given in Bengaluru, India, for participants of the operational research course conducted by the Centre for Operational Research of South-East Asia office of The Union in association with the India National Tuberculosis Programme, United States Centers for Disease Control and Prevention (CDC Atlanta), and the National Tuberculosis Institute, Bangalore. EpiData Analysis Version 2.2.2.180 was utilized. The participants (all residing in India and working for or in close association with the national tuberculosis programme) were Gurpeet **Singh**, Priyanka **Agrawal**, Raju **Chepuri**, D J **Deka**,

Rajeev **Pathak**, Amar **Shah**, Sunil **Kumar**, **Karthickeyan** DSA, Rajesh **Deshmukh**, Parija **Debudatta**, Asha **Fedrick**, **Shivramakrishnan**, Rajabhau D **Yeole**, **Anand** and **Tripathi**. The facilitators were Ajay M V **Kumar** (The Union, India), Srinath **Satyanarayana** (The Union, India), Balaji **Naik** (WHO-RNTCP, India), Sreenivas **Nair** (WHO-India), Vineet K **Chadha** (NTI, India), Patrick **Moonan** (CDC Atlanta), Smitha **Ghosh** (CDC Atlanta) and John **Oeltmann** (CDC Atlanta).

In **March 2013**, the **forty-third** course on “Efficient, Quality-assured Data Analysis using EpiData” was given in Bengaluru, India, for medical college professionals working in and around Bengaluru, Karnataka. This course was organized by Bangalore Medical College and Research Institute in collaboration with the South-East Asia office of The Union. This course utilized EpiData Analysis Version 2.2.2.180. The participants were Amit Kumar **Rao**, **Ashwini**, Navya CJ, Nagaraja **Goud B**, **Ranganath TS**, **Ravish KS**, **Sarsawathi S**, **Thilak SA**, **Shivaraj BM**, Riyaz **Basha S**, Nimmy **Dominic**, Swaroop Kumar **Sahu**, Josephine **Priya K**, Lilian **Dsouza**, **Hamsa L**, Puneeth **Kumar** and Subhash **Babu P**. The facilitators were Ajay M V **Kumar** (The Union, South-East Asia office and also module chair), Karuna D **Sagili** (The Union, India) and Deepak **Tamang** (The Union, India).

In **April 2013**, the **forty-fourth** course on “Advanced use of EpiData software for operations research” was conducted in Delhi, India, with an emphasis on Parts B, C, and D, using EpiData Entry Version 3.1 (270108) and EpiData Analysis Version 2.2.2.180. The participants, all from India, were Jaya Prasad **Tripathy**, **Palanivel C**, Amar N **Shah**, Balaji **Naik**, Priyakanta **Nayak**, Debadutta **Parija**, Mrinalini **Das**, Deepak **Tamang**, Karuna Devi **Sagili**, and Riyaz **Basha Sardar**. The facilitators were Ajay M V **Kumar** (The Union, South-East Asia office) and Hans L **Rieder** (The Union, Paris).

In **May 2013**, the **forty-fifth** course was conducted in Yaoundé, Cameroun, using EpiData Entry Version 3.1 (270108) and EpiData Analysis Version 2.2.2.181. The participants were Wilfried **Békou Kossi** (Bénin), Frédéric **Békang Angui** (Cameroun), André **Nana Yakam** (Cameroun), Yvon Martial **Ngana** (Centrafrique), Jules **Toloko Risasi** (Congo, RD), Valéri **Oulaï Ibodé** (Côte d’Ivoire). The facilitator was Hans L **Rieder** (The Union, Paris).

In **May-June 2013**, the **forty-sixth** course was conducted in Paris, France, using EpiData Entry Version 3.1 (270108) and EpiData Analysis Version 2.2.2.182. The course title was “Advanced use of EpiData software for operations research” and was given to four participants selected from the Union’s Centre for Operational Research 2012 cohort who had already passed “Module 1b”. The course focused on Parts B (Introduction to EpiData Analysis), C (Operations Research) and D (More on EpiData software). The participants were Kesete **Yirdaw** (Ethiopia), Prashant **Bhat** (India), Philip **Owiti** (Kenya), and **Lim Kang Yang Leo** (Singapore). The facilitators were Hans L **Rieder** (The Union), Ajay M V **Kumar** (The Union, India), and Jens M **Laurtisen** (EpiData Association).

In **July 2013**, the **forty-seventh** course was conducted in Berne, Switzerland, using EpiData Entry Version 3.1 (270108) and EpiData Analysis Version 2.2.2.182. The course title was “From paper to computer records – ensuring data quality for analysis”. The course was provided for participants enrolled in the Swiss Public Health training curriculum. The participants were Lars P **Andersen**, Anka **Baltensperger**, Nicole **Bartlomé**, Katrin **Crameri**, Corina **Ebnöther**, Nicole **Gysin**, Isabelle **Keel**, Gablu **Kilcher**, Jörg **Lützner**, Miriam **Pfiffner**, Anne **Schmidt**, and Johannes **Wacker**. The facilitators were Hans L **Rieder** (The Union) and Marcel **Zwahlen** (University of Berne, Switzerland).

In **September-October 2013**, the **forty-eighth** course was conducted in Dhaka, Bangladesh, using EpiData Entry Version 3.1 (270108) and EpiData Analysis Version 2.2.2.182. In

addition, an introduction was made to EpiData Manager public release version v1.4.2 and EpiData Entry Client v1.4.2. The course title was “EpiData software for quality-assured data entry and analysis”. The course was provided for personnel working in the Damien Foundation projects in Bangladesh and selected personnel from the national tuberculosis control program. The participants were Muhammad Ameer **Ali**, Panna Lal **Goswani**, Md Anwar **Hossain**, Md Shamin **Hossain**, Aung **Kya Jai Maug**, Priojit Kumar **Nandi**, and Binoy **Tudu**. The facilitator was Hans L **Rieder** (The Union).

In **October 2013**, the **forty-ninth** course was given in Chennai, India, for participants of the National Operational Research Course (2013-14) conducted by the South-East Asia Regional Office of The Union in association with the National Institute for Research in Tuberculosis, India. EpiData Entry Version 3.1 (Build 270108) and EpiData Analysis Version 2.2.2.182 were utilized. The course was titled “Efficient, Quality-assured Data Capture and Analysis using EpiData”. The participants (all residing in and working for tuberculosis control in India) were Kalpita **Shringarpure**, Deepti **Nirwal**, Sairu **Philip**, Shankar **Dapkekar**, Kiran Kumar **Reddy**, Kumaravel **Ilangovan**, Syed Imran **Farooq**, Hemant D **Shewade**, Umesh Chandra **Tripathi**, Bhavin **Vadera**, Anshul **Avijit**, and Poorana **Ganga Devi**. The facilitators were Ajay MV **Kumar**, Karuna **Sagili**, Deepak **Tamang** (The Union, India), Palanivel **C**, Swaroop **Sahu** (JIPMER, Puducherry), and Mrinalini **Das** (MSF, India).

In **March 2014**, the **fiftieth** course was given in Kathmandu, Nepal, for participants of the Asian Operational Research Course conducted by the Centre for Operational Research of South-East Asia office of International Union Against Tuberculosis and Lung Disease (The Union). EpiData Entry Version 3.1 (Build 270108) and EpiData Analysis Version 2.2.2.182 were utilized. The participants were Raman **Mahajan** (India), Wajira **Rajapakshe** (Sri Lanka), Sarder Tanzir **Hossain** (Bangladesh), Nay **Thiha** (Myanmar), Nang Thu Thu **Kyaw** (Myanmar), Basant **Joshi** (Nepal), Kimcheng **Choun** (Cambodia), Srinivas **Bairy** (India), Kinley **Zam** (Bhutan), Sai Ko Ko **Zaw** (Myanmar), Aashifa **Yaqoob** (Pakistan) and Vishal **Dogra** (India). The facilitators were Karuna D **Sagili** (The Union, India), Mrinalini **Das** (MSF, India), Prashant **Bhat** (WHO-RNTCP, India), Hemant **Shewade** (Indira Gandhi Medical College and Research Institute, India) and Jay Prasad **Tripathy** (PGIMER, India). Ajay M V **Kumar** (The Union, India) participated as observer.

In **July 2014**, the **fifty-first** course was given in Paris, France, for participants of the fifth Paris Operational Research Course conducted by the Centre for Operational Research of International Union Against Tuberculosis and Lung Disease (The Union). EpiData Entry Version 3.1 (Build 270108) and EpiData Analysis Version 2.2.2.182 were utilized. The participants were Riitta **Dlodlo** (Zimbabwe), Thi Mai Phuong **Nguyen** (Vietnam), Janet **Dzangare** (Zimbabwe), Daphne **Lagrou** (Afghanistan/Belgium), Shuisen **Zhou** (China), Mar **Verlade** (Switzerland), Charity **Kanyoro** (Kenya), Mbazi **Senkoro** (Tanzania), Wonda Teshome **Amenu** (Ethiopia), Michel **Sawadogo** (Burundi), Justine **Mirembe** (Uganda) and Josephine **Namboze** (Zimbabwe). The facilitators were Ajay M V **Kumar** (The Union, India), Rafael **van den Bergh** (Luxembourg), Kuda **Takarinda** (Zimbabwe), Philip **Owiti** (Kenya) and C **Palanivel** (India).

In **March 2015**, the **fifty-second** course was given in Kathmandu, Nepal, for participants of the fourth Asia Operational Research Course conducted by the Department of Research, International Union Against Tuberculosis and Lung Disease (The Union), South-East Asia Regional Office, New Delhi, India. EpiData Entry Version 3.1 (Build 270108) and EpiData Analysis Version 2.2.2.182 were utilized. The participants were Sudeepa **Khanal** (Nepal), Mongal Singh **Gurung** (Bhutan), Hamayoun **Hemat** (Afghanistan), Mahboob **Ul Haq** (Pakistan), Aye **Myat Thi** (Myanmar), Thi Thanh **Huyen Truong** (Vietnam), Sujit Kumar

Sah (Nepal), Zin Min **Thet Lwin** (Myanmar), Kathirvel **Soundappan** (India), Chandor **Wangdi** (Bhutan), and Srinivas **Reddy Edla** (India). The facilitators were C **Palanivel** (India), Vivek **Gupta** (The Union, India), Hemant **Shewade** (The Union, India), Philip **Owiti** (Kenya), Mrinalini **Das** (MSF, India) and Jay Prasad **Tripathy** (India).

In **March 2015**, the **fifty-third** course was given in Cotonou, Bénin, for collaborators from 9 francophone countries in Africa in the observational study of the “Bangladesh 9-month regimen” for multidrug-resistant tuberculosis. The course was given in French but the English course material was used and it focused on the analysis of study data, using as introduction material from Parts A and B. EpiData Entry Version 3.1 (Build 270108) and EpiData Analysis Version 2.2.2.183 were utilized. The participants were Gisèle **Badoum** (Burkina Faso), Wilfried **Békou** (Bénin) , François **Ciza** (Burundi), Valentin **Fikuma** (République Centrafricaine), Yves **Habimana-Mucyo** (Rwanda), Charlotte **Kamgué Tchieche** (Cameroun), Ferdinand **Kassa** (Bénin), Lucien Koffi **Kouakou** (Côte d’Ivoire), Olivia **Mbitikon** (République Centrafricaine), Marie-Léopoldine **Mbulula** (République Démocratique du Congo), Yvon **Ngana** (République Centrafricaine), Jürgen **Noeske** (Cameroun), Ibodé Valéri **Oulai** (Côte d’Ivoire), Alberto **Piubello** (Niger), Tandaogo **Saouadogo** (Burkina Faso), Bassirou **Souleymane** (Niger), Jules **Toloko** (République Démocratique du Congo). The facilitators were Ghislain Kobto **Koura** (The Union), Hans L **Rieder** (The Union), Valérie **Schwoebel** (The Union), and Arnaud **Trébucq** (The Union).

In **April 2015**, the **fifty-fourth** course on “Efficient, quality-assured data capture using EpiData” was given for medical college faculty and public health professionals of South India at Puducherry, India. This course was conducted by the South-East Asia Regional office of The Union in association with the medical college Jawaharlal Institute of Postgraduate Medical Education and Research (JIPMER). This course utilized EpiData Entry Version 3.1 (Build 270108). The participants were Sheethal **MP**, Gudadappa **Kasabi**, Prasanna **Thirunavukkarasu**, Vaman **Kulkarni**, Giriyantha **Gowda**, Vinayagamorthy **V**, Mohan **Kumar R**, Jeyashree **Kathiresan**, Sagar A **Borker**, Rachana **AR**, Suguna **E**, Sudhir **Prabhu H**, Asha **B**, Shankar **Reddy Dudala**, Sathish **Chandra MR**, Sowmya N **Malamardi**, Iswarya **S**, Achyutananda **Das Mohapatra**, Rakesh K **Nayak** and Prakash **M**. The facilitators were **Palanivel C** (JIPMER, Puducherry), Swaroop Kumar **Sahu** (JIPMER, Puducherry), **Kalaiselvi S**(JIPMER, Puducherry), **Subitha L** (JIPMER, Puducherry), Vivek **Gupta** (The Union, India), Hemant **Shewade** (The Union, India) and Ajay M V **Kumar** (The Union, India).

In **May 2015**, the **fifty-fifth** course with EpiData software was given for staff at the Leibniz-Center for Medicine and Biosciences, Research Center Borstel, Germany. Preceding the course, Part A was revised and the exercises on data form design and data entry were switched from utilization of EpiData 3.1 to EpiData Manager Version 2.0.4.43 and EpiData EntryClient Version 2.0.3.15. For the analysis component, EpiData Analysis Version 2.2.2.183 was used. The participants were Patricia **Sánchez Carballo**, Jan **Heyckendorf**, Barbara **Kalsdorf**, Ioana **Olaru**, Nelleke **Smitsman**, and Nasstasja **Wassilew**. The facilitator was Hans L Rieder (Tuberculosis Consultant Services, Switzerland).

In **July 2015**, the **fifty-sixth** course was conducted in Berne, Switzerland, using EpiData Manager Version 2.0.5.51 and EpiData EntryClient Version 2.0.3.15. For the analysis component, EpiData Analysis Version 2.2.2.183 was used. The course title was “From paper to computer records – ensuring data quality for analysis”. The course was provided for participants enrolled in the Swiss Public Health training curriculum and students and staff from the Institute of Social and Preventive Medicine of the University of Berne.. The participants were Marcel **Bruggisser**, Beat **Brüngger**, Carine **Graf Keer Rendon**, Rahel

Kasteler, Kathrin Pachlatko, Dominik Straumann, Nerina Vischer, Annette Weiss, and Kathrin Zürcher. The facilitators were Marcel **Zwahlen** (University of Berne) and Hans L **Rieder** (Tuberculosis Consultant Services, Switzerland).

In **September 2015**, the **fifty-seventh** course was conducted in Chennai, India, for participants of the operational research course conducted by the South East Asia office of The Union in association with the National Institute for Research in Tuberculosis (erstwhile Tuberculosis Research Centre). EpiData Entry Version 3.1 (Build 270108) and EpiData Analysis Version 2.2.2.183 were utilized. The participants (all residing in India and working in public health) were Himanshu **Gupte**, Himbala **Verma**, **Thirumugam M**, Yasobant **Sandul**, Dina **Nair**, Sendhilkumar **Muthappan**, Shobha **Ekka**, Ashish Kumar **Pandey**, Akash Ranjan **Singh**, Valan Adimai **Siromany**, Kalaiselvi **Selvaraj** and Sudhir **Chawla**. The facilitators were Hemant D **Shewade** (The Union, India), Jaya Prasad **Tripathy** (The Union, India), Karuna **Sagili** (The Union, India), **Palanivel C** (JIPMER, India), Prashant **Bhat** (CHAI, India) and Kalpita **Shringarpure** (Baroda Medical College, India).

In **October 2015**, the **fifty-eighth** course on “Efficient, Quality-Assured Data Analysis Using EpiData” was conducted for medical college faculty and public health professionals of South India at Puducherry, India. This course was conducted by the South-East Asia Regional office of The Union in association with Jawaharlal Institute of Postgraduate Medical Education and Research (JIPMER), a premiere medical college in India. This course utilized EpiData Analysis V2.2.2.183. The participants were **Sheethal MP**, Prasanna **Thirunavukkarasu**, Giriyantha **Gowda**, **Vinayagamorthy V**, **Mohan Kumar R**, Jeyashree **Kathiresan**, **Rachana AR**, Sudhir **Prabhu H**, Sathish **Chandra MR**, Sowmya N **Malamardi**, Achyutananda Das **Mohapatra**, Rakesh K **Nayak**, Yogesh **Bahurupi**, **Mahalakshmi**, Kavita **Vasudevan**, Prasad **Dhikale**, Ramesh Chand **Chouhan**, Bijaya Nanda **Naik** and **Prakash M**. The facilitators were **Palanivel C** (JIPMER, Puducherry), Swaroop Kumar **Sahu** (JIPMER, Puducherry), **Kalaiselvi S** (JIPMER, Puducherry), Jaya Prasad **Tripathy** (The Union, India), Hemant D **Shewade** (The Union, India) and Ajay MV **Kumar** (The Union, India).

In **November 2016**, the **fifty-ninth** course on “EpiData software for efficient, quality-assured data entry and analysis, Part 2 of the course: EpiData Analysis module” was conducted at the Forschungszentrum Borstel (Germany) for staff and invited guests. This course used EpiData Manager v2.0.13.64 and EpiData EntryClient v2.0.10.26 (for one exercise only) and EpiData Analysis V2.2.2.185. The participants were Serghei **Covanțev** (Chisinau, Moldova), Barbara **Kalsdorf** (Borstel, Germany), Niklas **Köhler** (Lübeck and Borstel, Germany), Ioana **Olaru** (Borstel, Germany), Patricia M **Sánchez Carballo**, Borsel (Germany), and Dagmar **Schaub** (Borstel, Germany). The facilitator was Hans L **Rieder** (Tuberculosis Consultant Services, Switzerland).

In **April 2018**, the **sixtieth** course on “EpiData software for efficient, quality-assured data entry and analysis, Part 2 of the course: EpiData Analysis module” was conducted at the Institute of Tropical Medicine in Antwerp, Belgium. This course used EpiData Manager v4.2.0.0 and EpiData EntryClient v4.2.0.0 (for one exercise only) and EpiData Analysis V2.2.2.187. The participants were Tom **Decroo**, Mourad **Gumusboga**, and Armand **Van Deun**, all from the Institute of Tropical Institute. The facilitator was Hans L **Rieder** (Tuberculosis Consultant Services, Switzerland).

Preparatory steps before you begin

If the course comes on a CD-ROM / DVD-ROM

If your course is on a CD-ROM, it might be easiest to copy its entire content, i.e., the folder containing the course, to your computer hard disk. One level below the folder, you will see a file:

index.html

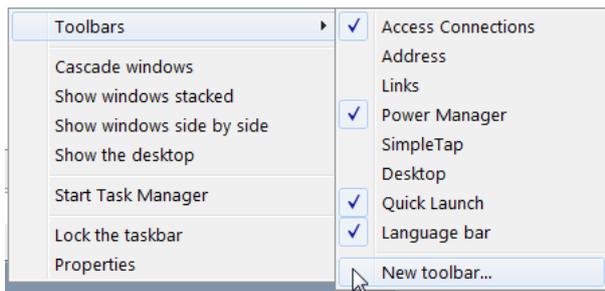
This is the name commonly given by convention to the opening page (the “home page”) of a web site. Indeed, the course material is organized like a web site. If you double-click on this file (INDEX.HTML), the web opens in your default browser. For fastest access to the web you might wish to make a short-cut to this opening page on your desktop and then drag it down to the Quick Launch taskbar (Windows XP® and Windows Vista®). To create a desktop shortcut, right-click on this file, go to Send To and Desktop (create shortcut) and click the latter. You can drag this desktop shortcut down to the Quick Launch taskbar. This will always give you visible access with a single click. Windows 7® has hidden the Quick Launch taskbar and has replaced it with an option to pin programs to the Taskbar. It is possible to restore the old Quick Launch taskbar if you wish (see next paragraph).

Restoring the Quick Launch bar in Windows 7®

Click anywhere on an empty part of the taskbar, right-click and ensure that “Lock the taskbar” is unticked:

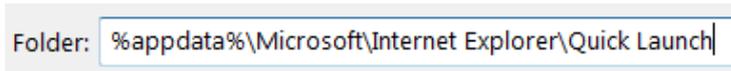


then right-click again and choose “Toolbars” | “New toolbar”:



and type into the location bar:

%appdata%\Microsoft\Internet Explorer\Quick Launch



and press Enter. Ensure that the path is displayed at the top, and click Select Folder. The Quick Launch bar will appear in the Taskbar. Drag the dots to pull it out, then right-click it and untick “Show text” and “Show title”. You can now drag short-cuts from the Desktop into the Quick Launch bar (and then delete them from the Desktop).

Creating working directories

We will require three folders, all in the root of the drive to which you have administrative access, commonly this will be the C:\ drive:

```
\epidata  
\epidata_course  
\temp
```

The three components of EpiData software, i.e. EpiData Manager, EpiData EntryClient, and EpiData Analysis, will be installed in the \epidata folder.

In the \epidata_course folder we will keep all our data files that we use and create during this course.

You may already have a \temp folder as it is common, very useful and good practice to have a temporary “storage” folder, i.e. a folder where we temporarily store files that we may download from the Internet or indeed in this case from our disk-based course web before we move them to the final destination on our PC. The content of the \temp folder should be such as not ever to inflict harm or loss if its entire content is erased (else it wouldn’t really be “temporary”).

Download EpiData Manager, EpiData EntryClient, and EpiData Analysis

EpiData Manager and EpiData EntryClient are very small executable files. Although the version we use is Windows®-based, the setup does not interfere with your Windows® set-up: all files are placed in a single directory (“folder” in current Windows® terminology) without leaving any trace of it anywhere else on your system (no *.dll files).

While EpiData Manager and EpiData EntryClient are available for all three operating systems, EpiData Analysis is, for the time being, a Windows® software program only, but with emulators the latter can be run under Linux and Mac operating systems.

The numbers in the file names indicate version. As development continues these change, but names of Manager and Entryclient will not change.

There are different ways to install / set up the software, and we propose the most simple one which does not even require any installation procedure: there is thus no requirement for administrator’s right, but instead you need to manipulate Windows® a little bit to get the programs accessible through your Windows® Start menu.

If you open the EpiData web site at <http://www.epiddata.dk> and click the link to the download page, you see:

New versions released !!

The new xml file format based system has been released, see [the specific mail](#) with information and get the software from [the download page](#)

You will note a whole list of options, Installation packages and further down Zipped executables (see next page).

For the setup here, we are choosing the Zipped Executable for the Windows® operating system. You need to know the architecture of your machine, whether it is 32-bit or 64-bit.

Installation packages		
OS:	Manager v2.0.2.28	EntryClient v2.0.2.13
Linux:	Standard 32 bit 10.Mar 2015 (4.8 Mb) [224]	Standard 32 bit 06.Mar 2015 (1.8 Mb) [216]
	Linux 64 bit 10.Mar 2015 (5.1 Mb) [210]	Linux 64 bit 06.Mar 2015 (2 Mb) [217]
Mac OS X:	Standard Intel 10.Mar 2015 (8.9 Mb) [560]	Standard Intel 06.Mar 2015 (4.6 Mb) [478]
	Power PC 10.Mar 2015 (8.9 Mb) [259]	Power PC 06.Mar 2015 (4.6 Mb) [247]
Windows:	Standard 32 bit 10.Mar 2015 (4.2 Mb) [1245]	Standard 32 bit 06.Mar 2015 (1.6 Mb) [954]
	Windows 64 bit 10.Mar 2015 (4.3 Mb) [1477]	Windows 64 bit 06.Mar 2015 (1.7 Mb) [1090]

Zipped executables

Zipped executables		
OS:	Manager v2.0.2.28	EntryClient v2.0.2.13
Linux:	Standard 32 bit 10.Mar 2015 (4.8 Mb) [172]	Standard 32 bit 06.Mar 2015 (1.8 Mb) [185]
	Linux 64 bit 10.Mar 2015 (5.2 Mb) [163]	Linux 64 bit 06.Mar 2015 (2 Mb) [188]
Mac OS X:	Standard Intel 10.Mar 2015 (4.8 Mb) [250]	Standard Intel 06.Mar 2015 (1.8 Mb) [241]
	Power PC 10.Mar 2015 (4.8 Mb) [184]	Power PC 06.Mar 2015 (1.8 Mb) [216]
Windows:	Standard 32 bit 10.Mar 2015 (4.6 Mb) [531]	Standard 32 bit 06.Mar 2015 (1.6 Mb) [405]
	Windows 64 bit 10.Mar 2015 (4.7 Mb) [538]	Windows 64 bit 06.Mar 2015 (1.7 Mb) [455]

If you do not know your machine's architecture, then type:

msinfo32

into the Windows® Search bar:



[Enter], and you get the information:

Item	Value
OS Name	Microsoft Windows 7 Professional
Version	6.1.7601 Service Pack 1 Build 7601
Other OS Description	Not Available
OS Manufacturer	Microsoft Corporation
System Name	WS20110930-1
System Manufacturer	LENOVO
System Model	4174P4G
System Type	x64-based PC

Alternatively, you can right-click on My Computer and click on Properties. You will be able to find the system type and operating system (here shown for a Windows 7® PC with a 64-bit architecture):

Processor: Intel(R) Core(TM) i7-3687U CPU @ 2.10GHz 2.60 GHz
 Installed memory (RAM): 12.0 GB (11.7 GB usable)
 System type: 64-bit Operating System
 Pen and Touch: No Pen or Touch Input is available for this Display

From among the Zipped executables, save the pair that is appropriate for your computer to your C:\TEMP folder. As a result you should get two of the following four zipped files:

[epidataentryclient.2.0.2.13.win.32.zip](#)
[epidatamanager.2.0.2.28.win.32.zip](#)

[epidataentryclient.2.0.2.13.win.64.zip](#)
[epidatamanager.2.0.2.28.win.64.zip](#)

The numbers in the file names indicate version. As development continues these change, but names of Manager and Entryclient will not change. As we are in the process of downloading from the EpiData web site, please move further down to EpiData Analysis:

EpiData Analysis

Download either one complete setup file as exe, zip or ALL the smaller parts. Setup file includes program, introduction and core documentation.
 First (V1.0) released Sept. 2005. V2.2 may 2013 (with epx support) A build is a minor revision, see [list of changes](#)

English V2.2 Rel. 2 (Build 183)	list of changes Command reference	7 page Intro 22.Oct 2008 (123 Kb) [32400]	<div style="border: 2px solid red; padding: 2px;"> Setup (Exe) 15.Jan 2015 (2.9 Mb) [2480] </div>	Setup (ZIP) 16.Jan 2015 (2.6 Mb) [917]
		Flowchart 14.Nov 2005 (12 Kb) [27834]		

Pick here please the executable setup file (circled in red above) and save it to your C:\TEMP folder.

setupepidatastat_2.2.2.183.exe

If you are offline but have the course CD-ROM / DVD-ROM, all files are available on it. “Download” them from the drive to your C:\TEMP folder and from there the following steps will be identical.

A setup method to circumvent the requirement for administrator’s rights to install software on your computer

Some users have a PC belonging or linked to a university (or other institution) that explicitly limits access rights to a computer. This reduces the risk of a user installing malicious or other unwanted software / code (broadly also known as “malware”) to a network-linked computer. Different methods exist to prevent users from doing that, e.g. blocking a drive or the entire computer altogether that will block installation. Usually, however, it will be possible to create folders on any drive and to copy downloaded files into such a folder. This is the approach we are employing here, i.e. escaping the usual installation routine for EpiData Manager and EntryClient (but excepting here EpiData Analysis, but see later).

You have already created a folder C:\EPIDATA which will serve as your software folder. If you have unzipping software such as the free open-source 7-zip installed and double-click for instance:

epidataentryclient.2.0.2.13.win.64.zip

you should see its content with two folders and one file:

Name	Size	Packed	Type	Modified
..			File folder	
docs			File folder	06-Mar-15 14:37
samples			File folder	06-Mar-15 14:37
epidataentryclient.exe	3,230,720	1,283,290	Application	06-Mar-15 14:37

Copy / unzip all three (from the correct architecture compatible with your computer) into your C:\EPIDATA folder. From the EpiData Manager zip file, we copy only the *.exe file:

Name	Size	Packed	Type	Modified	CRC32
..			File folder		
docs			File folder	10-Mar-15 0...	
samples			File folder	10-Mar-15 0...	
epidatamanager.exe	6,328,320	2,262,079	Application	10-Mar-15 0...	0F70C452

(the folders are identical in the EpiData Manager and EntryClient zip files respectively). If you cannot unzip because you are missing the software, please see below how to proceed.

With this, we already have everything for working with EpiData Manager and EpiData EntryClient. Note again that it is critical to have the correct version (32- or 64-bit respectively) because neither “upward” nor “downward” compatibility is assured between the two versions.

In your file manager (Windows Explorer® or an alternative), find now the EpiData Analysis setup file:

```
c:\temp\setupepidatastat_2.2.2.183.exe
```

and double-click the file. The installation routine suggests the default installation folder:

```
c:\Program files\EpiData
```

or

```
Program Files (x86)\EpiData
```

While this would be perfectly fine, we recommend installing it instead into the same folder in the root in which we already have the Manager / EntryClient software:

```
c:\epidata
```

The files will be extracted in a breeze.

If you look inside the folder and sort by extension, you see the three *.exe files:

```
epidataentryclient.exe
epidatamanager.exe
EpiDataStat.exe
```

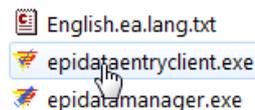
which start respectively any of the three modules that together make up EpiData software.

The approach we chose (circumventing the installation process for Manager and EntryClient) comes at a small cost: access was not written into the Windows® Start menu, but this is easily fixed.

On your Desktop, right-click in an empty space and select New | Shortcut and then Browse for and find the epidataentryclient.exe file:

Type the location of the item:

Browse...



and give it in the Next step an intuitively sensible name:

What would you like to name the shortcut?

Type a name for this shortcut:

And finish to see it on your Desktop:



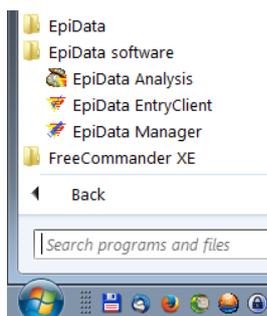
Repeat the procedure for the `epidatamanager.exe` file, and if not yet done during installation, also for the `EpiDataStat.exe` file, so that you have the linking icons to all three on your Desktop:



For fast, single-click access, drag each of the three icons into your Quick Launch bar to get:



Create now on your Desktop a new Folder and name it, say, “EpiData software”, then drag the above three links into that folder. The three icons will “disappear” from the Desktop into the folder. You can now drag the folder into your Start menu (“All programs”) with no need to place it alphabetically correctly as Windows® will take care of that. In this case, there was already a folder “EpiData” for the software on this computer, and the new “EpiData software” was added, showing upon single-click that it has now links to the three software components:



Clicking on any of these will open the respective module.

Getting everything in one zipped package

We have prepared a package named “`epidata_course.zip`” which can be found on the CD-ROM / DVD-ROM or on the web site. If you unzip it into your `C:\` drive, you get the two folders `C:\EPIDATA` containing the software with paths and defaults set, and `C:\EPIDATA_COURSE` containing the folder for the course material. All you need to do is

to make the links to the Desktop, the Quick launch bar and the Start menu as explained above and you are set to go.

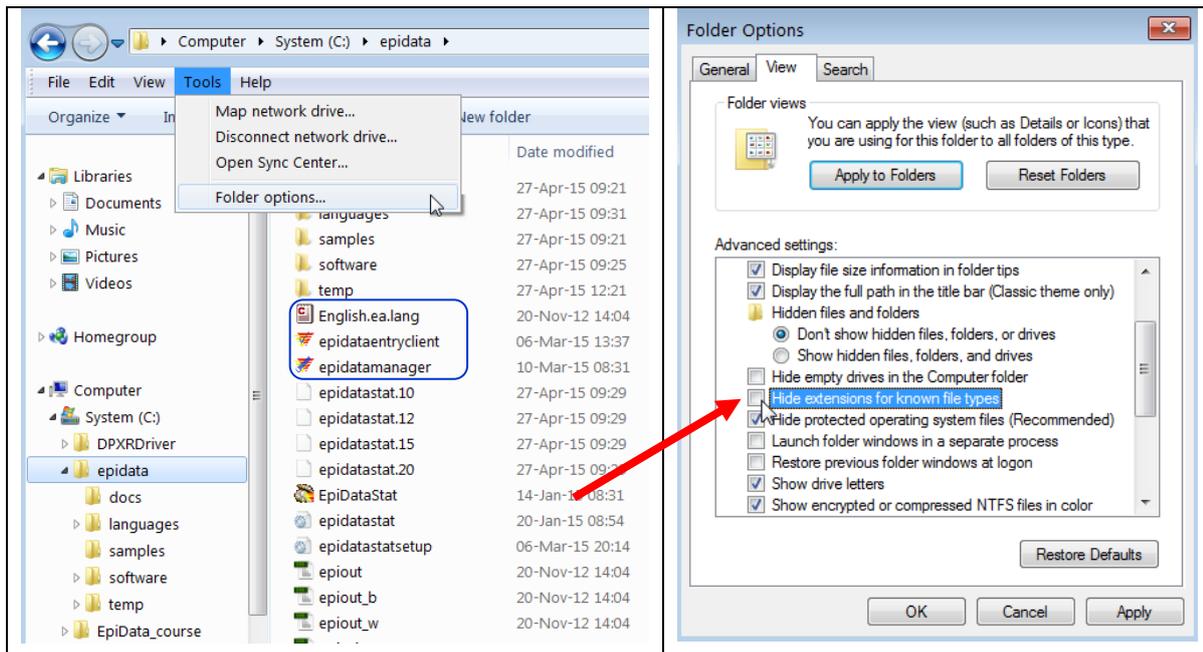
Not able to unzip?

If you are unable to unzip a zip file, it is likely that your computer has no software which can zip and unzip the files. We have provided freely available, high-quality open-source software (7-zip) in the course folder under the navigation panel item Software. Please download and install it. Please ask your facilitator and learn this important skill as you will need it quite often during the course.

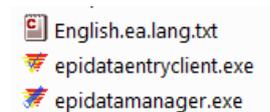
Not able to view the file extensions?

By default, the Windows® setting hides file extensions. However, you will quickly realize that it is essential to know them, thus change the default and make file extensions visible. Let us quickly do this.

Look at the screen shots below. Open any folder on your computer. Click on Organize and choose Folder and search options. A new window opens. Click on the View tab and look for the check box with the description “Hide extensions for the known file types”. By default, the check box is ticked. Click on it to un-tick the checkbox. Click on Apply and then OK to come out of the window.



To visualize the extensions:



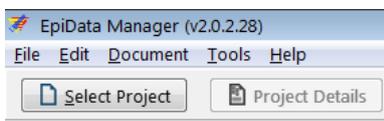
Setting up EpiData Manager

It is important to know the version of the software that you are working with. This serves two important purposes – first, it will let us know whether we are using the latest version. Second, it will be of immense help when you have some problem and are seeking troubleshooting assistance from your colleague/mentor. Letting your mentor know the version of the software you are using will help your mentor understand the problem and help you better. This will be in our next paragraph.

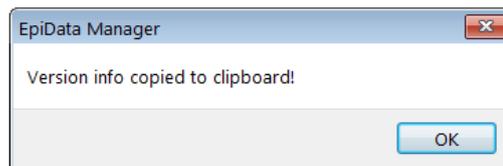
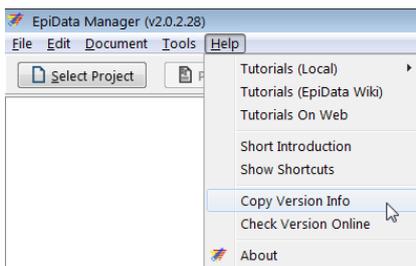
Access EpiData Manager and you note the opening screen:



On the top line we see the software version:



An even greater and more detailed option is to go to Help | Copy Version Info in Manager (identical access also in the EpiData Client):



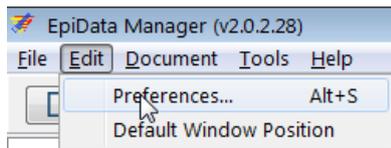
It will be written (and confirmed as such) to the Clipboard and you get it pasted with **CTRL+V** and on this computer the information is:

```
EpiData Manager
Program Version: 2.0.2.28 r1115
Core version: 2.0.0.0 r1126
FPC Version: 2.6.4
Platform: x86_64-Win64
```

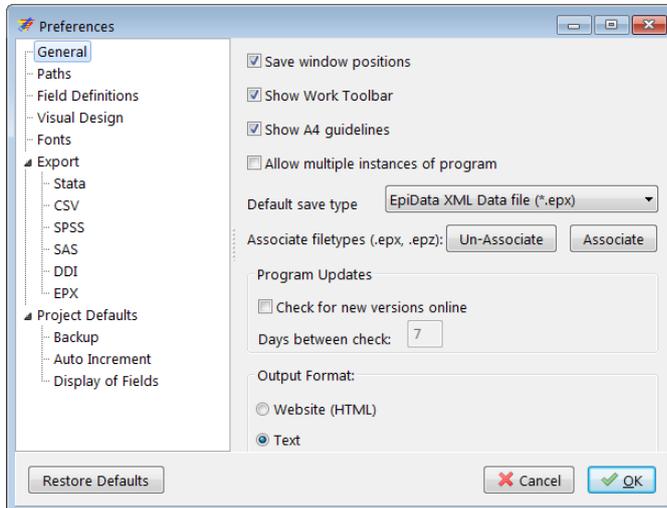
If you post a question to the EpiData list, it would be most useful for those expected to help you to have that information before they need to ask you to supply it! The developers made an effort to keep your life simple with such neat small pieces to aid efficiency! Quite often, users have an outdated buggy version, the problem of which was already solved in an updated version. Rather than forcing the potential assistant to ask back about the version you use, supply it together with your request for help.

Setting Preferences

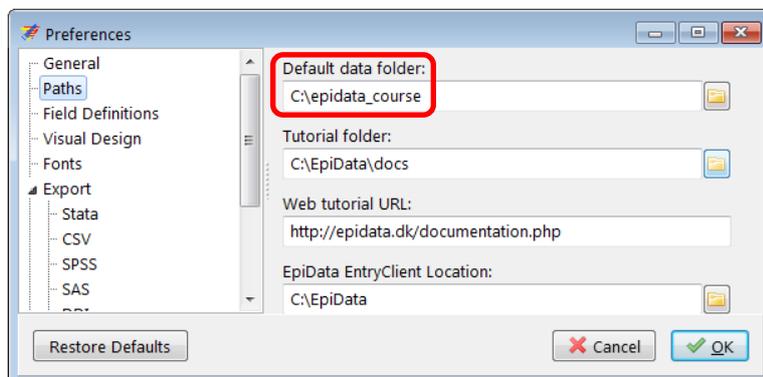
Under Edit we find Preferences, also accessible via shortcut sequence **ALT+E** | **ALT+S**:



and we get an extensive menu:

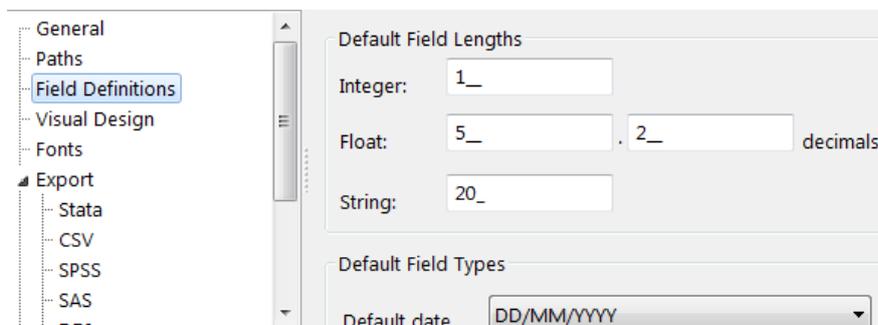


For our initial setup, we will make only a few major and necessary menu settings. We suggest that you study this in detail (every point) on your own sweet time. Of key importance is to set the correct Paths, most notably changing the Default data folder to our course working directory `C:\EPIDATA_COURSE`:

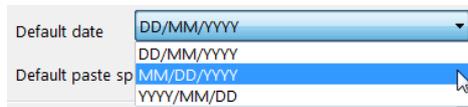


The other three paths remain unaltered as the default already defined by EpiData Manager.

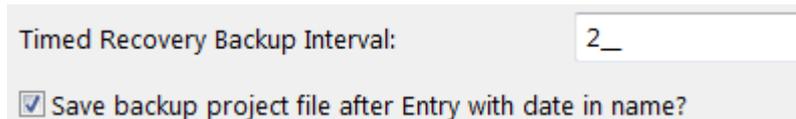
In the Field Definitions, we could set various defaults for the different field types:



For the time being, we allow the default proposed by EpiData. Note perhaps the Default date which is set to European style, but we could set an “American” or a “Chinese” date (critical is to set it to what you are culturally used to reduce the risk of data entry errors):

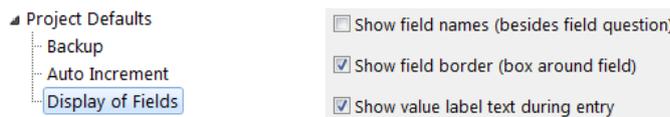


EpiData writes the data you enter first into memory, not immediately to hard disk. It does the latter after intervals you can set under Project Defaults | Backup and we recommend for the beginner to change the default from 10_ to 2__ (minutes):



Note that while shorter times are convenient against loss (due to an OS crash), they come at a cost: writing backups takes time, unnoticeable for small datasets, notable as the database grows. Before you exit a data file, you will always be prompted to save, you cannot leave the data inadvertently unsaved.

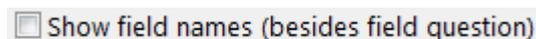
A note perhaps on Display of fields:



While we come back to the variables in the first exercise, it may be noted here that the name of a variable (we call it “Field name”) usually requires a certain format with respect to case, length, and permissible characters, while one is quite free with the so-called “Field label”, the name that explains the field. For instance, we may have a field name `age`, and the field label could be “Age in years at last birthday”. In EpiData 3.1 (and Epi Info 6), one option was the standard to display both next to each other, followed by the field definition like:

Age	Age in years at last birthday	###
-----	-------------------------------	-----

It was possible to suppress the field name by taking recourse to “automatic field naming”, but it required to write the data entry form quite differently and admittedly a bit complex and not so user-friendly. With EpiData Manager, the solution is a simple one for the user: tick or untick the:



Because the field name is of no use for the data entry person (we require working with it in analysis), we keep this unticked (but we see the field name clearly when we create the data entry form).

Preferences in EpiData EntryClient

We open EpiData EntryClient, and also choose Edit | Preferences. We change the data path to `C:\EPIDATA_COURSE` and leave the Colours as they are set by default: when using the

software in the EntryClient module we will learn practically if the colors are convenient, easy for the eye, and allow distinguishing the important components. This is also often individually different.

For the time being, these minor adjustment shall suffice.

At the end of this chapter, you should have done the following:

1. Installed EpiData Manager, EntryClient, and EpiData analysis (look for folder EpiData in C: drive)
2. Created desktop shortcuts or quick launch bar icons for the software components and checked their functionality
3. Created working directories in C: folder (named epidata_course and temp).
4. Have set the options in EpiData Manager and EntryClient software

If you have done the above steps successfully, we are now ready to move ahead!

Part A. EpiData Entry

Part A: Quality-assured data capture with EpiData Manager and EpiData EntryClient

- Exercise 1 A data documentation sheet for a simple questionnaire
- Exercise 2 Create a basic data entry form
- Exercise 3 Create a value-label pair from external data
- Exercise 4 Create a composite identifier
- Exercise 5 Data entry and validation
- Exercise 6 Upgrading an EpiData 3.1 REC/CHK file pair to an EPX file
- Exercise 7 Relational database

Acknowledgments:

We thank Ajay M V Kumar who had made valuable suggestions to improve the structure and flow of argumentation of the preceding version (using EpiData Entry 3.1) of Part A which we partially incorporated into this revised version.

Exercise 1: A data documentation sheet for a simple questionnaire

The first step in the process is to prepare a plan for data entry. This plan is called the **data documentation sheet**. This should not be confused with *data collection form* or *case report form* which is the proforma used for collecting the data from study participants or extracted out of the program records. The data documentation sheet is a codebook containing the details of all the variables (like field names, field labels, field type, field length, possible field values, and field value labels) to be entered.

Note: Please do not worry if you do not yet understand all the technical terms at this point in time. Be assured that you will appreciate these as you go along.

But let us proceed step by step and say that we have the following questionnaire:

Laboratory serial number: ____

Date specimen received (dd/mm/yyyy): ____/____/____

Sex: ____

Age in years: ____

Reason for examination: ____

Result of specimen 1: ____

Result of specimen 2: ____

Result of specimen 3: ____

This might present a typical simple questionnaire as used by an interviewer. Often such questionnaires are first completed on a paper *Case Report Form*. The above is actually an excerpt from the original Tuberculosis Laboratory Register proposed by The Union (note that the current version has been slightly changed):

Tuberculosis Programme Form 2

Tuberculosis laboratory register Year _____

Lab Serial No.	Date specimen received	Name	Sex M/F	Age	Name of referring facility	Address - patient for diagnosis	Reason for examination*		Results of specimen			Only for SS+ for diagnosis: TB Number or treatment centre**	Remarks
							Diagnosis (tick)	Month of follow up	1	2	3		

We will use this register as the basis for this course. For the time being, you plan to write a short and concise electronic data capture form, retaining only variables that are easy to capture and are likely to be useful for the analysis. *Please note this as a first principle in being efficient – capture only those variables which you will use for analysis!*

Each of the questions can be conceived of as a variable and the answer to the question as the value that the variable takes for a particular individual. **Variables** are also referred to as **'Fields'** in EpiData – both refer to the same and will be used interchangeably. We will give each variable a unique name. A completed entered data form for one study subject is called a **'Record'**. A set of such records is called a **'data file'**. The data file thus contains several records and each record contains information about one individual with respect to several variables. We are going

to describe each variable with respect to several attributes in the data documentation sheet. Let us now understand some terminology we are going to use.

- **Field name:** This is the name of the variable and in EpiData, there are certain rules to be followed in arriving at this name. We will come to these rules in a short while.
- **Field Label:** This is the descriptive name for the variable and contains a more detailed description than the variable name / field name can convey.
- **Field Type:** This describes the type of the variable – text, numeric or date being the major types.
- **Field length:** This describes the number of characters that a value can take.
- **Field values:** This describes the possible values that a variable can take.
- **Value labels:** These are descriptive names for the values. For categorical variables which are numerically coded, it is always useful to label them so that it is easier to read and understand what each of the codes mean.

“**Labels**” are also called “**metadata**” or “**data about data**”. They play a key role in data files. We may have entered a value “9” for a given field, but this number remains meaningless without specifying for what this value stands. It is important to get acquainted with these terms and understand them clearly since we will be using them frequently. We will be using several examples later in this chapter to clarify these terms.

Field name: There are some software-specific rules in naming a variable.

First, a Field name has to be **single word** that has **not more than ten characters**. This means that you cannot use a space in the name as a space makes it more than a word. Also, you cannot use any special characters like comma, semicolon, full-stop or underscore, and the first character should not be a number. Thus, do not start the variable name with a number. It cannot be ‘1v’, but it can be ‘v1’.

Note that some other analysis software may accept only a field length of eight characters. If you later plan to export your EpiData files for analysis to such a software package and you had used the full field length of ten, then your field names get truncated.

Second, use a name which is **intuitive** to understand what it means instead of generic field names like v1, v2 and so on.

Third, it may be a good practice to keep the field names in **lower case**. While EpiData is not case-sensitive, some other software is. Important among such are e.g. R and Stata® which are what we call “case-sensitive”. In the latter two, a field name of ‘sex’ (lower case), ‘SEX’ (Upper case), and ‘Sex’ (mixed case) are understood as different variables. If you later plan to export your EpiData files for analysis to such a software package, it may make life unnecessarily complicated if you have been inconsistent without a defined rule. Hence, the recommendation to keep it uniformly, “lower case”.

The following words ‘date’, ‘month’, and ‘year’ are functions in EpiData and are reserved names. Hence they cannot be used as variable names.

Field label: This is the full description of the variable and can be more than a word.

Field Type: There are different types of entry fields for the variables (we will follow the EpiData notation and call them “Fields”):

- **Text fields:** These fields take letters or numbers or a combination of these as possible values, like PETER, KOCH1882, giraffe, 45677 etc. You can type anything on the keyboard into this field. If you enter a number into such a field, it is accepted but you will not be able to make any calculation with it. These fields are also sometimes designated as character or alphanumeric fields, or most simply “**string**” (denoted by **S**) fields as they take any string of characters.
- **Numeric fields:** These are numbers. The numbers might be integers (denoted by **I**) like 885, 33, 1235 or real numbers like 3.4, 6.88, and 66.5 (also called **floats** and denoted by ‘**F**’). You can make calculations with numeric fields.
- **Date fields:** (denoted by ‘**D**’): In different countries, different ways of writing dates are used and this can be confusing for people from another culture. Some write *5 March 2005*, others *March 5 2005*, and again others *2005 March 5*. EpiData lets you choose the type of date you wish to take. We made our choice for European dates in the Preferences as we will be using European dates in this course, i.e. dates of the format *5 March 2005* or symbolized with DD/MM/YYYY.
- One other type of variables is called “**logic**” or “**Boolean**” variables. This is sometimes used in food-borne outbreak investigations. There, answers to questions on food items eaten is limited to “yes” and “no” and “missing”. There is no need for using this field type and we discourage its use as it might pose problems in analysis. The alternative is a numeric field with a label block.

While you are asked to limit the length of the field name, you have much more flexibility with the length of the value a field can take, but we will try to use it as efficiently as possible, that is we will limit the value length to the minimum needed.

Data Documentation Sheet

It is good practice to write what we call a **data documentation sheet** before you make your actual data entry form in EpiData Manager. As mentioned earlier, EpiData refers to this as **Codebook**.

In the past, fields like `sex` were commonly made text field with values “F” or “M” denoting Females and Males. It is efficient as it uses only a length of 1 to remain unambiguous (well, as long as the language is English or French, at least!). Things would get less efficient, if we would have to code treatment outcome with the possible values “cured”, “completed”, “died”, “failed”, “lost from follow-up”, “transferred out”, and “outcome not recorded”. Numeric coding is much simpler as there are up to ten possible values with a field length of just 1:

- 1 Cured
- 2 Treatment completed
- 3 Died of any cause
- 4 Failed bacteriologically
- 5 Lost from follow-up
- 6 Transferred out
- 9 Outcome not recorded

Later, in the analysis, you will also realize that it is very convenient to apply numeric selection criteria when you want to select a subset of data and undertake analysis only on the subset. Of course, a prerequisite is that the link between the numeric value and the text label is unambiguously clear. The role of labels is of enormous importance, also called meta-data or

“data about data” as mentioned above. We are going to make full use of numeric coding of field values and using explicit text as value labels.

Let us now go through a few examples of the variables (from the tuberculosis laboratory register example) and describe the various attributes of the variables in the data documentation sheet. As you go through, you will note that preparation of data documentation sheet requires thinking and knowledge of study data.

This is how we would write such a data documentation sheet:

Field name	Question [Field label]	Field type	Field length	Field values	Value labels	Notes [Comment]
serno	Laboratory serial number *	I	4	1-9000, 9001, 9002,		Serial number starting with 1 each year Enter 9001, 9002,... if serial number is <i>not unique</i> or <i>missing</i> , and write a data entry note
regdate	Registration date	D	10	01/01/2000-31/12/2005		Range of legal registration dates
sex	Examinee's sex	I	1	1 2 9	Female sex Male sex Sex not recorded	

* **Note:** Often, it will be preferable to make the identifier a text field. If it is a number, as in this case here with the laboratory serial number, precautions must be taken to distinguish e.g. “0001” from “1”, requiring that the numeric value is entered into one field. We can make use of the possibility in EpiData Manager to add leading zeros where appropriate.

Task:

- o Complete the data documentation sheet for all fields in the questionnaire. Note that you should always define a value if no answer was provided to a question.*
- o Think of the most efficient ways to code reason for examination and results of microscopic examination*

Solution to Exercise 1: A data documentation sheet for a simple questionnaire

Key Point(s):

- It is good practice to write a data documentation sheet before you make your actual data entry form with EpiData Manager.
- You should always define a value if no answer was provided to a question.
- “Date” is a reserved name in EpiData and cannot be used as a field name.

Task:

- o Complete the data documentation sheet for all fields in the questionnaire. Note that you should always define a value if no answer was provided to a question.*
- o Think of the most efficient ways to code reason for examination and results of microscopic examination.*

Solution:

There are many different solutions, but for the sake of uniformity, we will be using the following (but later revise some components of it) as shown on the next page.

Field name	Question [Field label]	Field type	Field length	Field values	Value labels	Notes [Comment]
serno	Laboratory serial number	I	4	1-9000 9001, 9002,		[Serial number starting with 1 each year] Enter 9001, 9002,... if serial number is <i>not unique</i> or <i>missing</i> , and write a data entry note
regdate	Registration date	D	10	01/01/2000-31/12/2005		
sex	Examinee's sex	I	1	1 2 9	Female sex Male sex Sex not recorded	
age	Examinee's age in years	I	3	0-125, 999		[Range of legal age years] Enter 999 if age not recorded
reason	Examination reason	I	1	0 1 2 3 4 5 6 7 8 9	Diagnosis Follow-up at 1 month Follow-up at 2 months Follow-up at 3 months Follow-up at 4 months Follow-up at 5 months Follow-up at 6 months Follow-up at 7 months or later Follow-up, month not stated Reason not recorded	
res1	Result of specimen 1	I	1	0 1 2 3 4 5 6 9	Negative 1+ positive 2+ positive 3+ positive Positive, not quantified Scanty, not quantified Scanty, quantified Result not recorded	[If the entered value is other than 6, then bypass next field and bypass it]
res1sc	Result of specimen 1 scanty	I	1	1 2 3 4 5 6 7 8 9	1 AFB per 100 OIF 2 AFB per 100 OIF 3 AFB per 100 OIF 4 AFB per 100 OIF 5 AFB per 100 OIF 6 AFB per 100 OIF 7 AFB per 100 OIF 8 AFB per 100 OIF 9 AFB per 100 OIF	
res2	Result of specimen 2	I	1	0 1	Negative 1+ positive	[If the entered value is other than 6, then bypass next field and bypass it]

				2 3 4 5 6 9	2+ positive 3+ positive Positive, not quantified Scanty, not quantified Scanty, quantified Result not recorded	
res2sc	Result of specimen 2 scanty	1	1	1 2 3 4 5 6 7 8 9	1 AFB per 100 OIF 2 AFB per 100 OIF 3 AFB per 100 OIF 4 AFB per 100 OIF 5 AFB per 100 OIF 6 AFB per 100 OIF 7 AFB per 100 OIF 8 AFB per 100 OIF 9 AFB per 100 OIF	
res3	Result of specimen 3	1	1	0 1 2 3 4 5 6 9	Negative 1+ positive 2+ positive 3+ positive Positive, not quantified Scanty, not quantified Scanty, quantified Result not recorded	<i>[If the entered value is other than 6, then bypass next field and bypass it]</i>
res3sc	Result of specimen 3 scanty	1	1	1 2 3 4 5 6 7 8 9	1 AFB per 100 OIF 2 AFB per 100 OIF 3 AFB per 100 OIF 4 AFB per 100 OIF 5 AFB per 100 OIF 6 AFB per 100 OIF 7 AFB per 100 OIF 8 AFB per 100 OIF 9 AFB per 100 OIF	

Note the following here. For an unknown laboratory date (REGDATE), we must enter a legally existing (valid) date and we chose a legal but practically impossible date in the past, i.e. 01/01/1800. EpiData will not accept a date 99/99/9999 nor for that matter 29/02/2001. It is a personal preference of us to usually use 9 or 99 . 9 or the like to define unknown values, be this for text or numeric variables. We also introduced a “legal range” for some variables like REGDATE and AGE. We did this a bit arbitrarily, but still tried to keep it within what might be expected.

We made two fields for each result. There are 17 possibilities for a result, and therefore a length of 2 is the minimum required, but even with that, the values might not be intuitive, but they should be. An alternative version uses a float of length three to get for instance:

```
0.0    Negative
1.0    1+ Positive
2.0    2+ positive
...
0.1    Scanty, 1 AFB per 100 OIF
0.2    Scanty, 2 AFB per 100 AFB
```

Scanty results are relatively rare among positives (perhaps some 10% among diagnostic and some 20% among follow-up examinations in quality-assured high-burden country laboratories), and positives themselves are relatively rare among all (perhaps 10% to 20% among patients coming for diagnostic evaluation). Thus, scanty positive results might be only 1% of all results. We therefore chose to use integer variables and bypass the field scanty, unless the first field defines the result as quantified scanty.

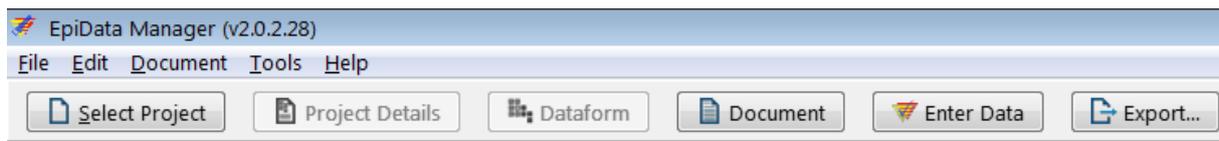
Exercise 2: Create a basic data entry form

At the end of this exercise you should be able to:

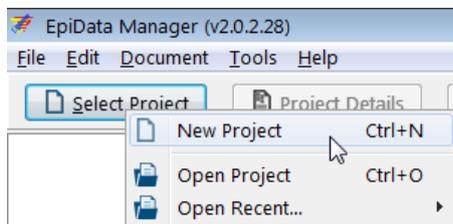
- Create and edit a data entry form with EpiData Manager.
- Include study information and password protection.
- Use headings and sections.
- Create basic label blocks, understand “Must enter”, jumps, ranges, using label blocks for missing / out of range numeric data.
- Align properly horizontally and vertically both for aesthetics and functionality.

You are now ready to start with the design of the questionnaire with EpiData Manager, based on the data documentation sheet prepared in Exercise 1.

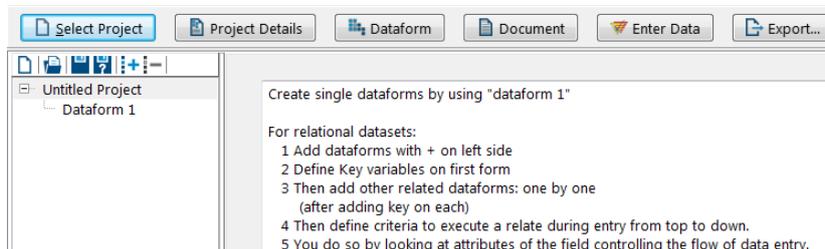
When we open EpiData Manager, we have the interface with three rows, displaying the version in the top row, menu items in the middle row and some specific menu items in the bottom row:



As we start from scratch, we begin with Select Project | New Project (or shortcut **CTRL+N**):



We note the opening of the set up:

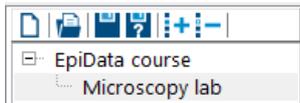


The information on the right side informs about the differences of using a single data entry form (as we will do now) or creating a relational dataset (as we will do in the last exercise of Part A). Hovering over the small icons:

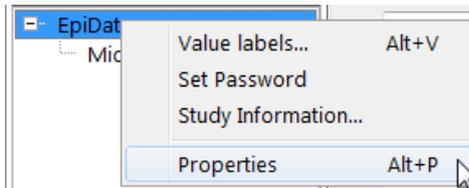


shows what they accomplish.

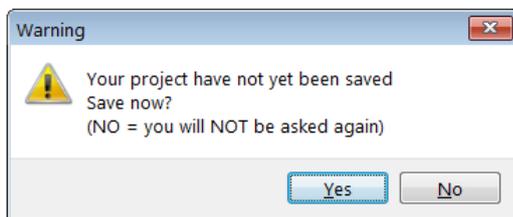
Clicking on Untitled Project and pressing function key F2 allows us changing the name to, say, “EpiData course”, and clicking on the Dataform 1 and pressing function key F2 allows us to change the name of the form to “Microscopy lab”, so that we will get:



Right-clicking on the Project name gives us options:

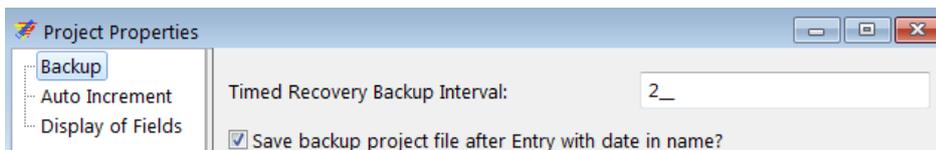


As we play around, at some point in time, EpiData Manager will prompt us to save the project:

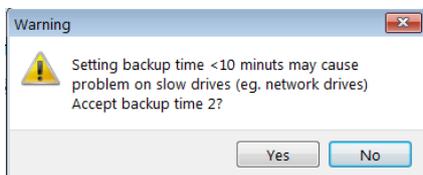


And we will do so promptly, saving the project as **a_ex02.epx** (the extension is supplied and the suggested folder is our chosen data folder C:\EPIDATA_COURSE).

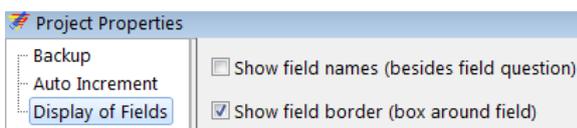
To resume from the above project options, let’s check Properties (shortcut **ALT+P**). You note that we get a window with something we had already defined earlier during set-up in Preferences:



If we click on any of the other two, e.g. Display of Fields, we get a small warning (also mentioned during set-up):



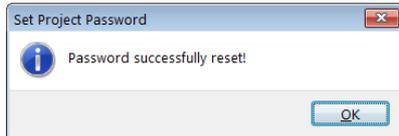
We accept and get what we had already accepted as default in Preferences:



There are thus different possibilities to set our preferences and to change them here at any time without first going back to the initial settings. Thus we OK and check out Set Password with a right-click:



There are different levels of password setting, this is for the entire project, but a password can also be set for sections within a project. For the time being, we do not wish to set a password for the project and we clear with the Enter key and get:



Do you note the difference between when we click on the project name or the data form name respectively? Clicking on the project name “EpiData course” you get:

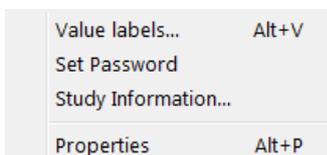


At the very right (circled red) you see that this is the first of the several vertically aligned tabs, further including Coverage, Description, Ownership, Funding, and Version Details. In the current tab we could add an Abstract, clicking on the second tab, Coverage, we can add information on Geographical location, Language, Data time coverage, Population, and Unit of observation. The third tab, Description, allows entering Keywords, Purpose, Citations, and Design. The fourth tab, Ownership, allows entering Organisation/Institute, Agency, Authors/Contributors, and Rights. A fifth tab, Funding, is a free text field, and the final tab, Version, permits entering information about the study and the study form. Short, this menu option allows thorough documentation of the study. It is highly recommended that this is actually used to its fullest for a real study. For the purpose of this course, we will leave it unfilled.

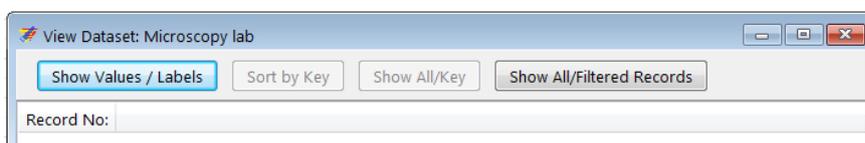
Contrasting right-clicking the Project name and the Dataform name we get:

Project name: EpiData course

Dataform: Microscopy lab



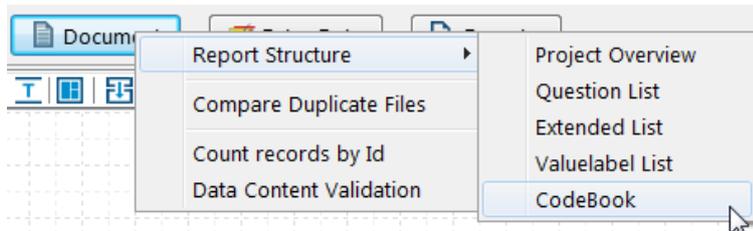
We have already checked out some of these options, while others will be used in what is coming once we start to actually make the data entry form, while further ones like Browse Data will only become available once we have data, while for now, as expected, an empty window opens:



The two sub-menus above can also be accessed any time from the Project Details and Dataform menus respectively:



The third menu point above, Document, gives a sub-menu and the sub-menu item Report Structure has itself a sub-menu:

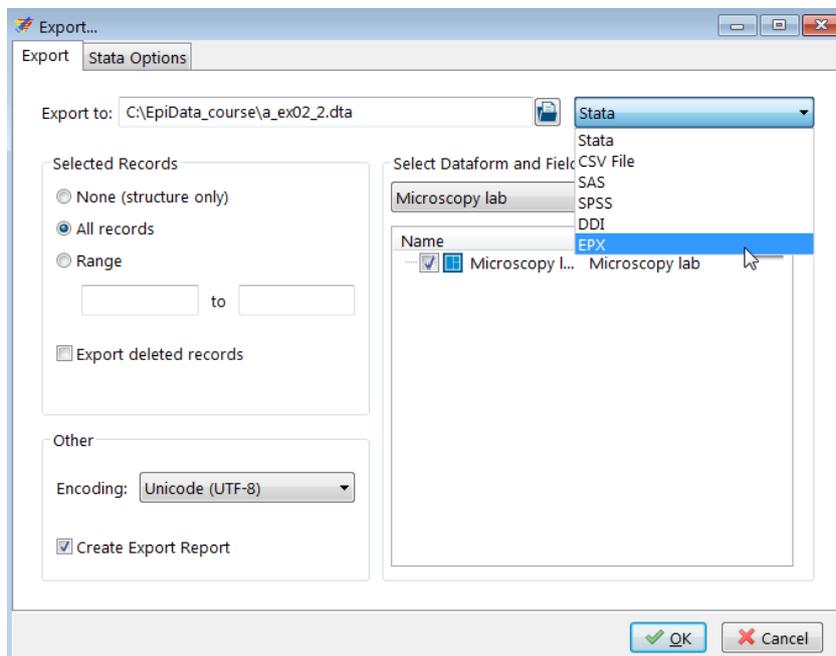


some of which we will be using at a later point of the course.

If the EpiData EntryClient software module is located in the same folder as the EpiData Manager as is the case in our set-up (and it is recommended not to keep them separately), we have direct access to it from the second-to-last menu item Enter Data:



The last menu item, Export, gives multiple options to export the structure or the data to various formats:



We will be making use of this menu item at a later point in time.

Designing the data entry form

Before starting with the actual design of our data entry form, let's clarify the terminology and where what is going to be placed. There are four elements that go with a field:

Field name	Field label	Field value	Comment / Value label
age	Person's age in years	23	Enter 999 if not recorded
sex	Person's sex	9	Not recorded

We call the variable the **Field name** (see Exercise 1). As shown in Exercise 1, the Field name may or may not be displayed. In setting up Preferences, we opted for not displaying it. To have more explicit information than what can be conveyed by the length 10, single-word Field name, we supplement the Field name with a more explanatory **Field label**. The third element on the line is the Field definition which receives the actual **Field value** during data entry. The Field value is the actual datum (singular of “data”) for that variable for that observation. For *continuous variables*, we might add an explanatory **Comment** to be displayed in a Note during data entry, here for instance what to do if the primary source has no information on this variable for this observation. For *categorical variables*, the comment is not necessary: a label block will comprehensively provide information on what is entered. As we use numeric coding, it would, however, be a nice touch if after picking a value from a pick list (label block), the **Value label** would appear to the right of the field, as a kind of visual confirmation that what had been intended was actually picked.

If we click in the Dataform tab “Microscopy lab”, we look now at the data form canvas:



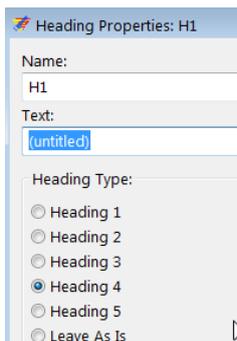
Above the grid we see various icons, most importantly right now those that symbolize different variable types which we can identify by hovering with the cursor over each one:



From the excerpt here of these icons, from left to right, these are:

- New Integer Field
- New Float Field
- New String Field
- New DMY Field
- New Time Field
- Other Field Types...
- New Heading
- New Section

We start with the second-to-last on the left, the New Heading, click on it and get the menu:



We overwrite the Text with:

Tuberculosis Microscopy Laboratory

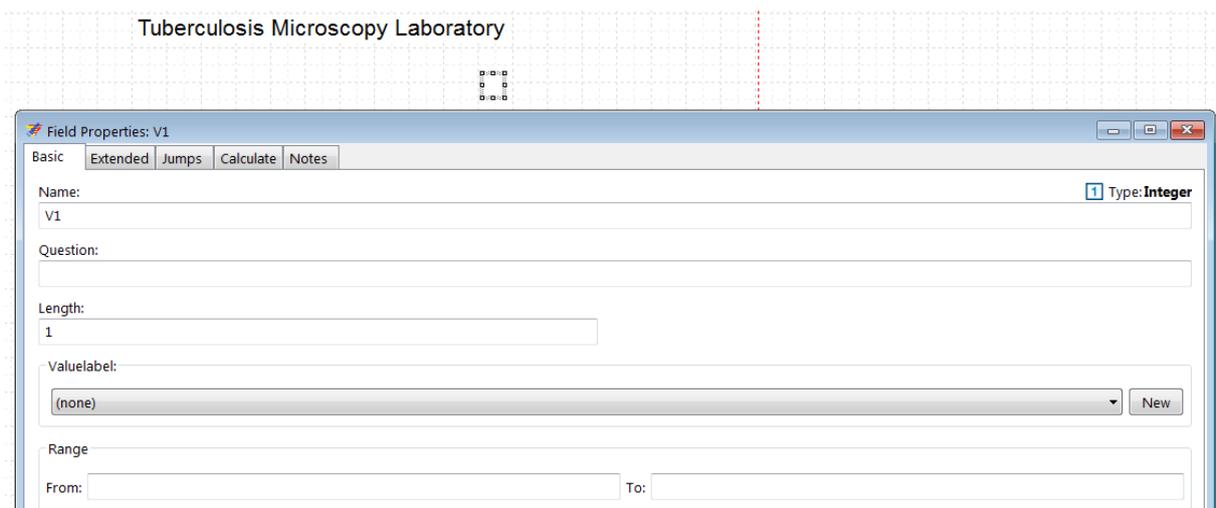
Those with some notion of HTML (the language of the internet) may recognize that text size is expressed relatively, where Heading 1 is the relatively largest and Heading 5 the relatively smallest. Let's choose (arbitrarily) Heading 2 and click Apply to get our heading:



You can drag this text box around to your preferred position. By right-clicking it, you can Edit it at any time to change the wording or the size if you wish to do so.

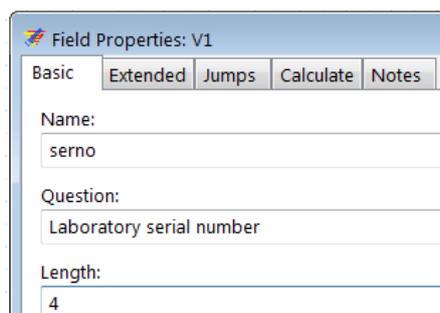
In some larger data entry forms, we then may next choose to make sections to which we could give different attributes, but for the time being, we will simply start with our fields.

Consulting the Data Documentation Sheet, we see the first field to be `serno`, the Laboratory serial number, an integer field of length 4. Clicking on the icon for the integer field, then clicking into a suitable place on the canvas, we get the Field Properties menu:

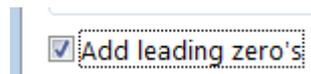


This is our main working tool for the design of the data entry form. We will therefore complete each page where applicable and study carefully the options on each tab.

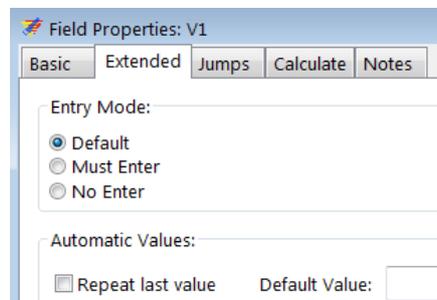
The field Name is given the default name V1 and we change this to `serno` as defined in the Data documentation sheet. The Question is for the Field label which we defined as Laboratory serial number, and its Length is 4. It has no Valuelabel (it is not a categorical variable) and we have no Range to assign, thus completed we get:



At the bottom of the Basic tab we tick to add leading zero's:



We switch to the tab Extended showing (at the top):



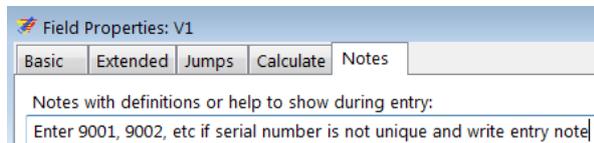
The Entry Mode is of critical importance and there are three possibilities. The Default is “you do as you wish”, i.e. the data entry person is allowed to enter or not to enter a value into the field. The second option Must Enter signifies that a value has to be entered else the data entry person cannot continue to work (or if bypassing the field by using the mouse, will be stopped latest when trying to save the record). This is our preferred method because “forgetting to enter” is not the same as “there is no information” and by forcing Must Enter a value must be entered. Therefore, whenever a value for a variable is missing, we must tell the data entry person what has to be entered in such a situation. We will thus make every field a Must Enter field with two exceptions. The first exception is for calculated fields, something we will use in a later exercise when we create a composite identifier. The second exception is for fields that can be bypassed with Jumps as we will see shortly. For the Laboratory serial number we thus tick Must Enter. There is a possibility for Automatic values, such as Repeat last value. If we tick the latter, the next new record will inherit the value from the current one for this field. This is convenient for instance if a long sequential series of records always has the same date before it changes. Whenever we change the inherited value by overwriting it in a new record, the new value will be passed on as a repeat value to the next record. Obviously, this is not what we need for the Laboratory serial number. After having ticked Must Enter, we thus move to the tab Jumps:



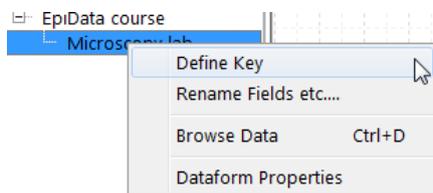
Jumps are used to define whether based on a value in the current field the next field is to be bypassed or not. If the field asks for smoking status and the response is non-smoker then the next field that would quantify the number of cigarettes per day is bypassed while it is not if the response in the current field is smoker. We will use this with the results fields.

The Calculate tab allows creating the value for a new variable from the values of one or more existing variables. For instance one could calculate the value of a date field from three date components. We generally discourage the use of calculations during data entry, they belong to the analysis, with one important exception though. Sometimes, a unique identifier for a record can only be defined by constructing from more than one variable. We will show this in a later exercise.

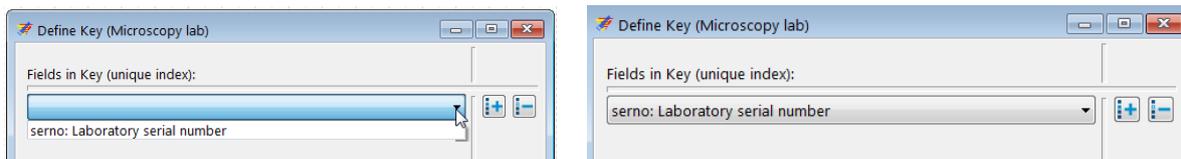
What we write into the Notes tab will briefly pop up if the cursor is in the field. For the Laboratory serial number we actually foresee such a situation, and this is to instruct to enter “artificial” serial numbers 9001, 9002, etc. if we happen to stumble upon a non-unique serial number. We thus enter a Note as for instance:



As an interlude here, we need to clarify key issues about the identifier. No variable is of more importance in a record than the unique identifier. We can even state that a record is useless without a unique identifier. For instance, if we have no value for age in a record, we can always check whether it is truly missing from the original case report form if we can identify that case report form by the unique identifier. Without it, we cannot possibly learn the truth. It is impossible to validate our data with double-entry if we do not have unique identifiers. For persons not familiar with the tuberculosis microscopy laboratory register, this is what you need to know: each examinee who presents to the laboratory for a microscopic examination is given a serial number which starts with 1 each calendar year. An examinee might thus provide a series of up to 3 specimens and the examinee (not the examination) is the unit of observation and “numbered” as such. The Laboratory serial number is unique for an examinee presenting him- or herself newly for a serial set of examinations usually over a one- to three-day period during the course of 1 year in 1 laboratory. We have not yet instructed EpiData Manager to treat the Laboratory serial number as a unique identifier. This is not done here in the Field Properties, it has to be done for the file (you may remember the options by right-clicking the data form name Microscopy lab):



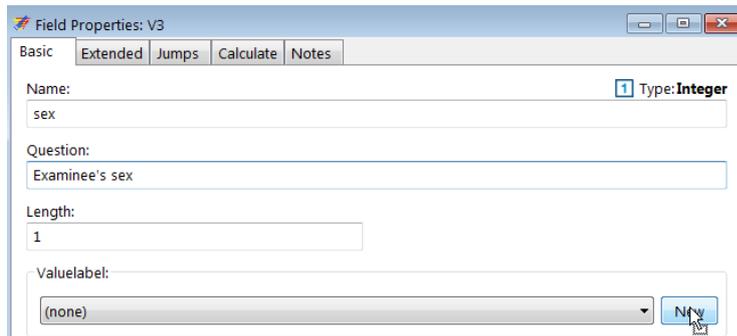
It is of critical importance to Define Key, thus when done, we close the Field Properties dialog menu and go to the data form Microscopy lab, click on Define Key and select it:



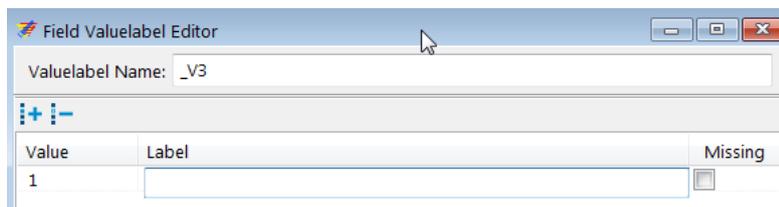
What this will do is that whenever you enter a Laboratory serial number in a new record, EpiData will check the entire data set whether or not this identifier has already been used. If not, you can continue your work. If it had been used in another record, the EpiData EntryClient will tell you in which record and offer to go to it to change it if it had been entered erroneously or to enter an alternate in the current one (as suggested here in Notes). In any case, you cannot continue working unless the identifier is assuredly unique for the file.

Our next field is the Registration date with the field name regdate. We thus pick the icon for New DMY Field and start to complete the Basic tab of Field Properties. We did define a legal range. Enter it and make the field Must Enter.

The third field is `sex` which we defined as integer for length 1 and we need to create here a new Valuelabel:



The sub-menu for the Valuelabel:



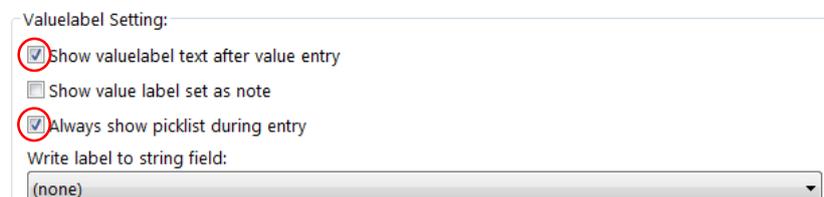
It is quickly completed – note that we give a sensible, easily identifiable Valuelabel Name. Though we are quite free how we do that, taking the default is not very informative, thus:



We could tick the value for Not recorded as Missing, it might or might not come handy later in the analysis, but we leave it unticked for the time being. After accepting, the Valuelabel Name will get inserted automatically as Labelvalue into the foreseen box in the Basic tab:



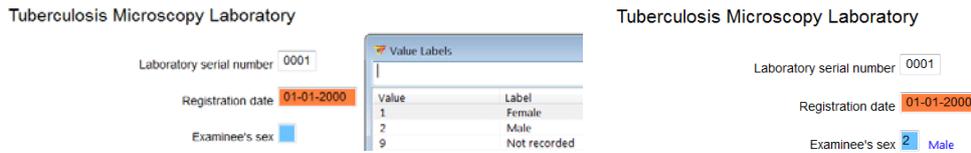
else we can pick it from the drop-down list (red circled here). In the Extended tab we make the field Must Enter and tick the two boxes applicable for the Valuelabel Setting:



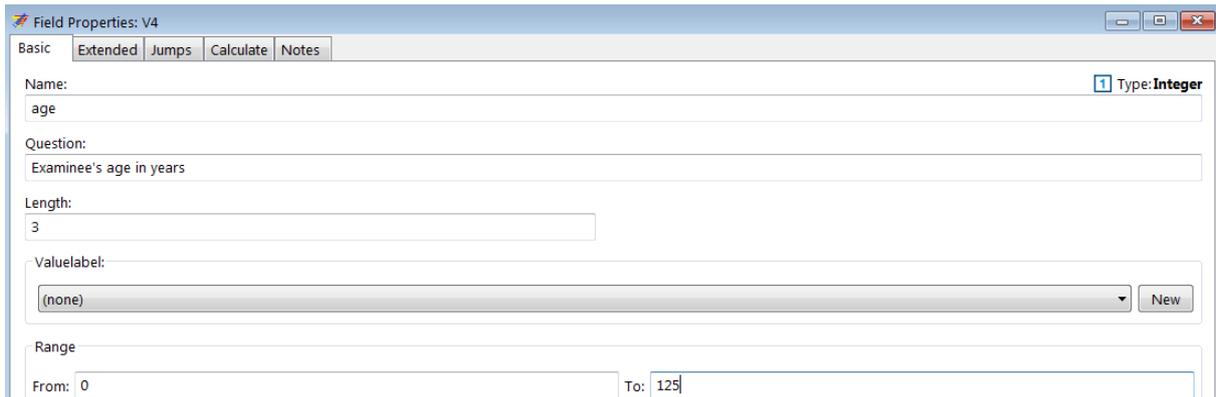
The first (Show valuelabel ...) is ticked by default, but the Always show picklist ... must be set. What the two accomplish is that the value goes into the field box and the Valuelabel is written to the right of the field box (for visual verification), but before this is done, the Picklist with the Value-Valuelabel pairs is popping up ready for selection as soon as the cursor gets into the field. In EpiData Manager, it gets indicated in green that we have a given label block:



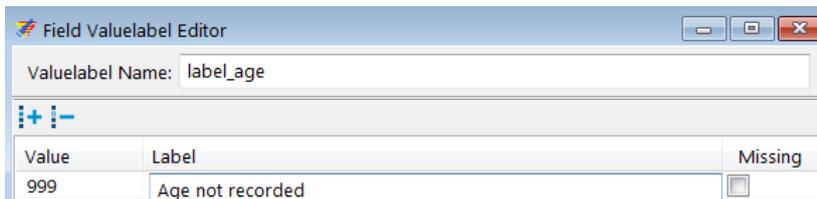
In the EntryClient, ticking both boxes will give the pop-up (left) and the writing of the value label (right) after the selection of the value:



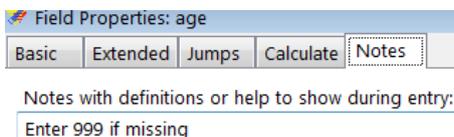
The fourth variable is age, the Examinee's age in years, with a field length of 3, and for which we allow a range from 0 to 125, and a legal value of 999 for missing age:



The out-of-range legal value is put into a Valuelabel, thus pick New and add as follows:



Then set to Must Enter in the Extended tab, but set / keep all boxes in Valuelabel Setting unticked: the value-label pair is just to “legalize” the value 999, not to have a rudimentary pop-up window. Instead, we instruct the data entry person in the Notes tab what to do if the value for age is missing:



The fifth variable is reason, the Reason for examination, an integer of length 1. The principle is the same as for sex and nothing is new here above what was learned before.

The sixth variable is res1, the Result of specimen 1, an integer of length 1. Again, the principle is the same as for the other categorical variables with label blocks, but we do have something new. We will insert a Jumps command for every value, except if it is 6, Scanty quantified. Because we need to tell EpiData where to go, if the field has or has not the value 6, we propose that everything else but the Jumps command for fields res1, res1sc, and res2 is completed first, and then we return to res1 to formulate the Jumps command (not

critical here as will be seen). We don't have to, but let's arrange the fields for `res1` and `res1c` (and the other "pairs") horizontally:

Note also that we do not need to write a label block for each result and for each scanty result, it suffices to write a generic label block for results as all three results have the same ValueLabel blocks and we may name it `label_result` and name the ValueLabel block for scanty results something like `label_scanty`. After we have it for the first, we can pick it from the drop-down list for the ValueLabel:

Once we are done with these, we return to the field `res1` and open the Jumps tab to enter what we require. Because the field to skip is the next field (`res1sc` here), we do not actually need to state to which field to go. We just add all values except 6 and the instruction (Skip Next Field) under the Go To Field. Click the + sign to the right to add a new line and enter the next value:

Jump Value:	Go To Field:	Reset Value:
0	(Skip Next Field)	Leave as is
1	(Skip Next Field)	Leave as is
2	(Skip Next Field)	Leave as is
3	(Skip Next Field)	Leave as is
4	(Skip Next Field)	Leave as is
5	(Skip Next Field)	Leave as is
9	(Skip Next Field)	Leave as is

Tasks:

- o Finalize the data entry form for the remaining results variables. Note that the fields for scanty results cannot be **Must Enter**. Note further the options in the Drop down menu for what is the default "Skip Next Field" to pick the best option for the last result.*
- o Align the field correctly using the Alignment icon. Note that correct vertical alignment is critical if two variables are on the same horizontal pane (EpiData Manager gives nice blue and red guiding lines). If two variables on the same horizontal pane are vertically mismatched, the sequence of data entry will go wrong!*
- o Tell EpiData Manager to create the Codebook and save the output file as a text file "a_ex02_codebook.txt".*

Solution to Exercise 2: Create a basic data entry form

Key Point(s):

- The identifier is the most important variable in a record
- EpiData is not case-sensitive, but some other software is. It is therefore advisable to use a simple rule such as consistently using lower-case for field names.

Tasks:

- o *Finalize the data entry form for the remaining results variables. Note that the fields for scanty results cannot be **Must Enter**. Note further the options in the Drop down menu for what is the default “Skip Next Field” to pick the best option for the last result.*
- o *Align the field correctly using the Alignment icon. Note that correct vertical alignment is critical if two variables are on the same horizontal pane (EpiData Manager gives nice blue and red guiding lines). If two variables on the same horizontal pane are vertically mismatched, the sequence of data entry will go wrong!*
- o *Tell EpiData Manager to create the Codebook and save the output file as a text file “a_ex02_codebook.txt”.*

Solution:

The data entry form may look as follows:

The screenshot shows a data entry form titled "Tuberculosis Microscopy Laboratory" on a grid background. The form contains the following fields:

- Laboratory serial number
- Registration date
- Examinee's sex label_sex
- Examinee's age in years
- Reason for examination label_reason
- Result of specimen 1 label_result
- Result of specimen 1 scanty label_scanty
- Result of specimen 2 label_result
- Result of specimen 2 scanty label_scanty
- Result of specimen 3 label_result
- Result of specimen 3 scanty label_scanty

Once a data entry form has been prepared, it is best to test it right away in the EntryClient with some fake data (without saving) to identify quickly problems.

You can leave the data entry form open in EpiData Manager and access the EpiData EntryClient via the menu and after prompting the Manager will close.

Opening in the EntryClient:

Tuberculosis Microscopy Laboratory

Laboratory serial number

Registration date

Examinee's sex

Examinee's age in years

Reason for examination

Result of specimen 1 Result of specimen 1 scanty

Result of specimen 2 Result of specimen 2 scanty

Result of specimen 3 Result of specimen 3 scanty

we see that all fields which should be Must Enter actually are (orange-brownish color) and that the fields which can be bypassed (for quantified scanty results) are not.

Entering fake data, we get:

Tuberculosis Microscopy Laboratory

Laboratory serial number

Registration date

Examinee's sex Female

Examinee's age in years

Reason for examination Diagnosis

Result of specimen 1 Negative Result of specimen 1 scanty

Result of specimen 2 Scanty, quantified Result of specimen 2 scanty 8 AFB per 100 OIF

Result of specimen 3 Result not recorded Result of specimen 3 scanty

Confirmation

? Save Record?

It all looks as it should, neat and nicely. Do not save the record and exit without saving.

We saved the CodeBook as text file and can look at it in a text editor. It must reflect what we defined in the Data documentation sheet but is now much more detailed and a superb document that can be shared with others who later collaborate in the analysis:

```

=====
Report: CodeBook
Created: 28-04-2015 18:33:16
=====

-----
File 1: C:\EpiData_course\a_ex02.epx
-----

File 1: C:\EpiData_course\a_ex02.epx
-----
Title      EpiData course
Created    28-04-2015 09:50:32
Last Edited 28-04-2015 18:06:40
Version    1
Cycle      13
-----
Backup on shutdown: yes
Encrypted data: no

Dataforms:
-----

```

Field: res3c: Result of specimen 3 scanty

Type Integer
Length 1
Show Value Label true
Show Picklist true

Value label: label_scanty [I]: (Integer)

Value Label Missing (M), set: label_scanty

1 1 AFB per 100 OIF
2 2 AFB per 100 OIF
3 3 AFB per 100 OIF
4 4 AFB per 100 OIF
5 5 AFB per 100 OIF
6 6 AFB per 100 OIF
7 7 AFB per 100 OIF
8 8 AFB per 100 OIF
9 9 AFB per 100 OIF

Exercise 3: Create a value-label pair from external data

At the end of this exercise you should be able to:

- a. Create an EpiData two-variable dataset from a text file
- b. Use an external two-variable dataset to create a label block internally or externally

In the previous Exercise we worked with small label blocks of not more than ten category levels. These are quickly created directly in the data form. Often we encounter the need for larger standard label blocks. This might be a list of administrative units in a country and it may have hundreds of levels. Such lists might be publicly available on the internet from the government and they can come in different formats such as text files or spreadsheets. In order to standardize as much as possible for our data collection, we should take recourse to official lists whenever possible, rather than inventing our own parallel system. If we can get such a list into a formatted file for EpiData, it might also be used in more than a single project.

In this Exercise, we provide a spreadsheet which contains two variables, the name of a given tuberculosis laboratory paired with its code, not dissimilar to a list of communities paired with their zip codes.

Formatting the external file

The first thing to know is the file type, and into which format it should be brought. We have a spreadsheet `a_ex03_namecode.xls` with two columns:

	A	B
1	Awuna	ML_J
2	Beitbridge	MS_D
3	Bindura	MC_A
4	Binga	MN_G
5	Birchenough	ML_M
6	Bonda	ML_I
7	Brunapeg	MS_G
8	Chasit...	MM...

You remember how the label block for the field `sex` was set up:

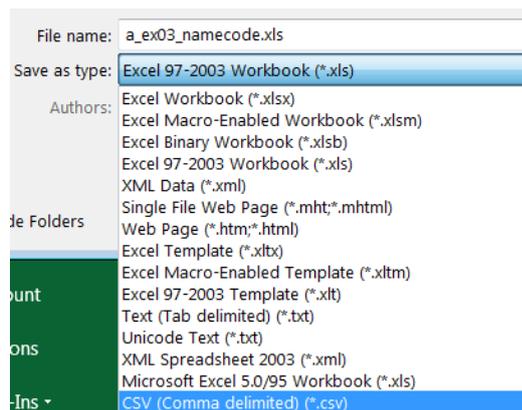
Value	Label
1	Female
2	Male
9	Not recorded

On the left is the Value that is written into the field and on the right the Label that provides the meta-data. The setup is thus the Value presenting a code on the left and the Label on the right. With short lists there is no issue because our eye captures quickly both Value and Label, but how about a very long list? How do we search a list of communities and their zip codes? People tend to know the name of a community, but not their zip codes. We are also used to search alphabetically by name not by zip code, yet we wish the code to be the value to be written into the field, not the name. EpiData makes it easy for us: we can search substrings, and it will find them in either column! If we start typing `birch...` in EpiData EntryClient, the software will

go to the first appearance of these letters be they in the left or the right column. Therefore, what we might do preferably, is to put the code of the laboratory into the left column and the name of the laboratory into the right and sort alphabetically by the name of the laboratory (not its code). This way the laboratory code is the value that is written into the field and the laboratory name is its label. Therefore invert to get:

	A	B
1	ML_J	Awuna
2	MS_D	Beitbridge
3	MC_A	Bindura
4	MN_G	Binga
5	ML_M	Birchenough
6	ML_I	Bonda
7	MS_G	Brunapeg
8	MW_J	Chegututu
9	MV_I	Chikombedzi
10	MC_H	Chimhanda

EpiData cannot read a spreadsheet file with an *.xls extension, but it reads text files with a *.txt extension or the common text standard of comma-delimited *.csv files. Conversely, most software, including proprietary Excel® or the non-proprietary LibreOffice Calc spreadsheets can save a file as a *.csv file. Use thus Save as (in Excel®) in search for the appropriate file type:



A word on delimiters: we do need a delimiter. A delimiter is an element that separates two variables. In normal text, a space is the delimiter separating two words. For our purpose, neither spaces nor tabs are good separators because they could be part of a variable. Indeed, the name for one laboratory is Collin Saunders. If we had a space as separator, these two components would erroneously be interpreted as two variables. A comma as in *.csv is better, but it is not necessarily fail-safe, best would be semi-colon delimited: it is easy to see and it is rarely used. In any case, in our 95 records (please visualize them in your text editor):

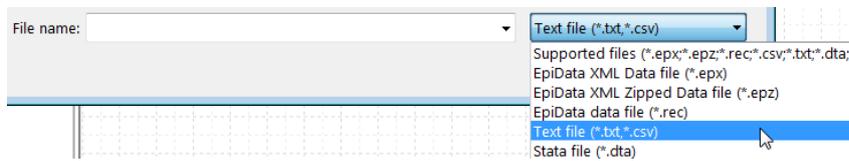
```
ML_J, Awuna
MS_D, Beitbridge
MC_A, Bindura
MN_G, Binga
ML_M, Birchenough
ML_I, Bonda
MS_G, Brunapeg
MW_J, Chegututu
MV_I, Chikombedzi
MC_H, Chimhanda
```

there are no commas apart from the desired ones to separate the two variables. We almost have now the format we require to make an EpiData *.epx file out of the *.csv file. **Important Note:** EpiData Manager interprets the first line as header but because the file does not have headers, the first line will erroneously be cut off. Therefore, we add a header to the *.csv file as follows:

```
labcode,labname
ML_J,Awuna
MS_D,Beitbridge
MC_A,Bindura
MN_G,Binga
```

Creating an EpiData *.epx file from a *.csv file

Open EpiData Manager and File | Import file ..., and select the correct file type:



Pick the a_ex03_namecode.csv file. This will open the dialog window:



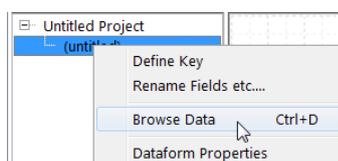
Note here the helpful options should you expect duplicates in Field Name – Field Value Labels. We can leave the defaults as they are and click OK. This gives us an Untitled Project with an (untitled) data form:



At the bottom of the screen we see that we have correctly 95 records:



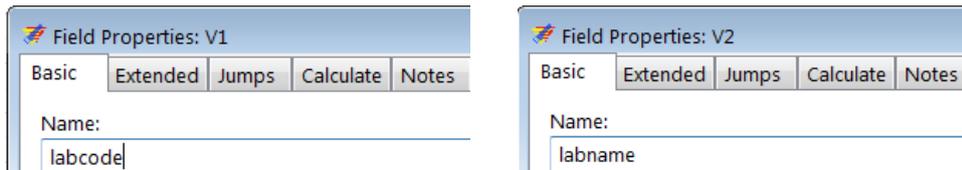
Whether or not prompted to save, we save the file as a_ex03_namecode.epx. Right-clicking the data form (or with CTRL+D):



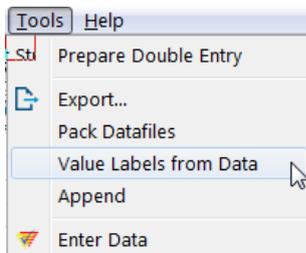
we browse to view the data set:

Record No:	V1	V2
1	ML_J	Awuna
2	MS_D	Beitbridge
3	MC_A	Bindura
4	MN_G	Binga

Closing and clicking on the first, and then on the second field respectively we can rename the two field names to labcode and labname to have a bit more meaningful information:



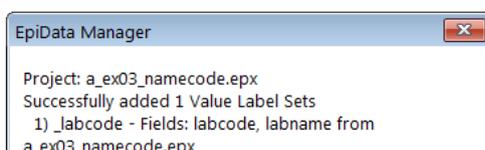
To make a label block set of the file go to Tools | Value Labels from Data:



and pick the file itself, ensuring that Include is ticked and both Value Field and Label Field are correctly selected:

Order	Filename	Include	Value Field	Label Field	Missing Field	Dataforms	Created	Last edited	Version	Cycle
1	a_ex03_namecode.epx	<input checked="" type="checkbox"/>	labcode	labname		1	29-04-2015	29-04-2015	1	1

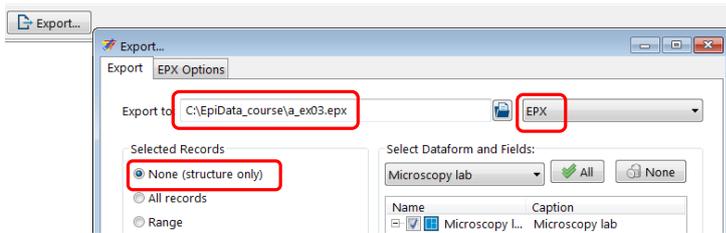
We get confirmation that the Value Label Set was created:



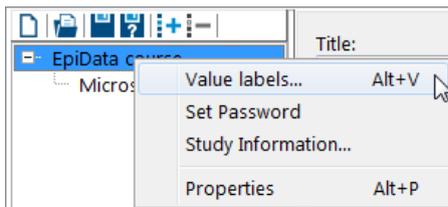
Save and close.

Updating the EpiData data form

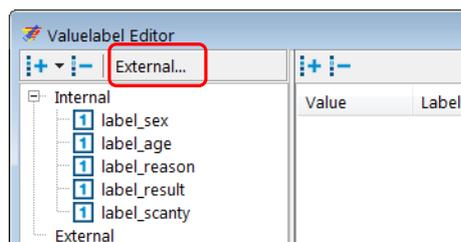
The first thing to do is to open the file a_ex02 .epx from Exercise 2 and Export the structure only to an EPX file we name a_ex03 .epx:



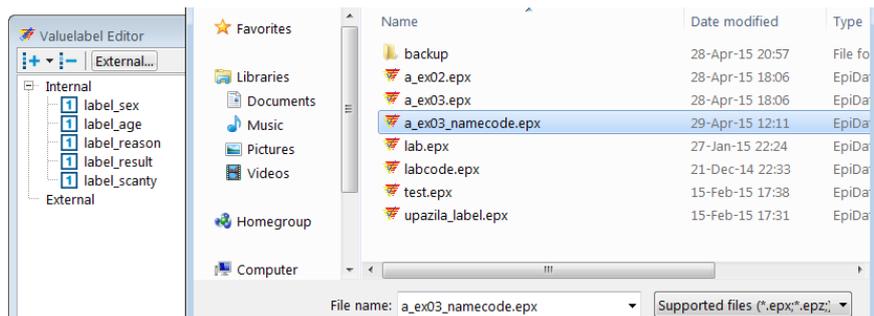
We then open the thus newly created `a_ex03.epx`. If we right-click on the project name EpiData Course we get a menu line Value labels (**ALT+V**):



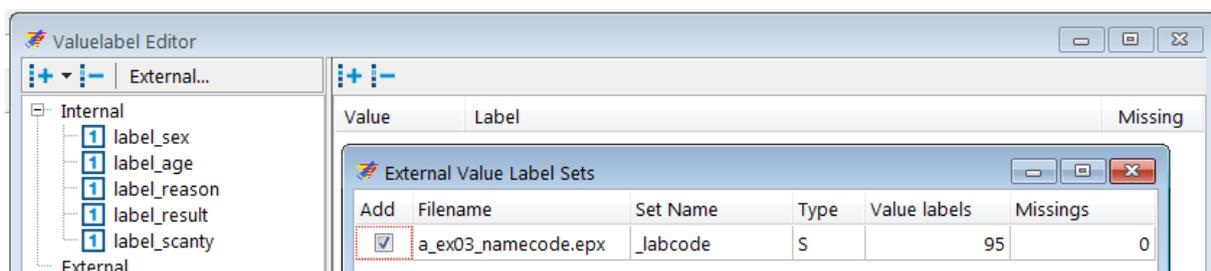
This gives us the Valuelabel Editor:



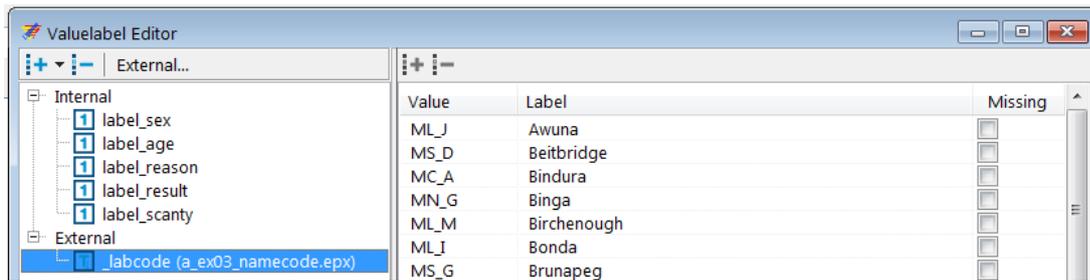
which allows the use of an external file as a label block. Using this approach (you will also look for an alternative to incorporate it), we can keep the label block externally. As long as it is in the same folder, the Manager / EntryClient will access it. Click and select the file:



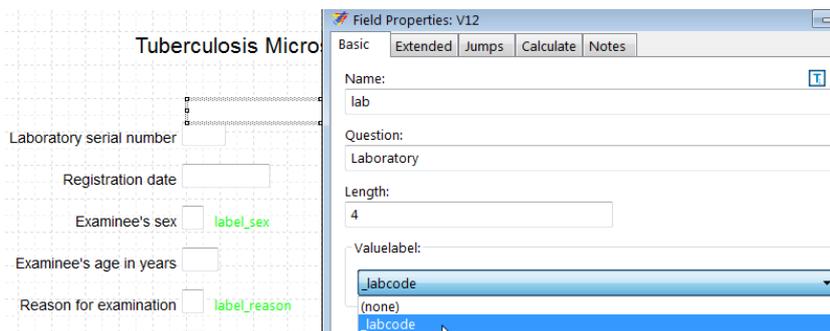
We are shown the name of the file, what the Set Name for the label block is (`_labcode`), that it is a string field with 95 Value labels:



We have now an external label block available and its content is visualized by clicking on it:



On the data entry form, we add a new string variable lab of length 4 with the field label Laboratory and add the now available Valuelabel:



Define the Extended as usual, save and test in EpiData EntryClient. Note particularly in EntryClient that for searching you start typing the name of a Field label and the cursor jumps to it.

Tasks:

- o Remove the external label block.*
- o Use the a_ex03_namecode.epx file and use it internally as a label block.*
- o Discuss advantages and disadvantages of External versus Internal label blocks.*

Solution to Exercise 3: Create a value-label pair from external data

Key Point(s):

- If possible, use an official list of, for instance, administrative units, and create a file format that can be imported into EpiData (such as *.csv)
- Create an EpiData *.epx file and make a label block of it
- You can use that label block internally or externally in your data collection form

Tasks:

- o *Remove the external label block.*
- o *Use the a_ex03_namecode .epx file and use it internally as a label block.*
- o *Discuss advantages and disadvantages of External versus Internal label blocks.*

Solution:

This is the modified data entry form with a fake record:

Tuberculosis Microscopy Laboratory

Laboratory **MV_J** Collin Saunders

Laboratory serial number **0001**

Registration date **01-01-2001**

Examinee's sex **1** Female

Examinee's age in years **45**

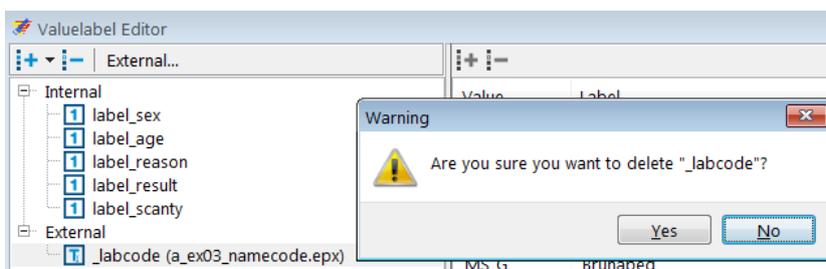
Reason for examination **1** Follow-up at 1 month

Result of specimen 1 **0** Negative Result of specimen 1 scanty

Result of specimen 2 **6** Scanty, quantified Result of specimen 2 scanty **5** 5 AFB per 100 OIF

Result of specimen 3 **9** Result not recorded Result of specimen 3 scanty

We access the Valuelabel Editor with **ALT+V**, click on it and press the Delete key:



and it's gone after accepting. To use the file internally, we close the Valuelabel Editor and use instead Tools | Value labels from data, select the file, and insure that all is ticked and selected correctly:

Order	Filename	Include	Value Field	Label Field
1	a_ex03_namecode.epx	<input checked="" type="checkbox"/>	labcode	labname

Importantly, the deletion of the external also removed the link to the Valuelabel for the field:

Name: Type: **String**

Question:

Length:

Valuelabel:

and this must then be set again. Conveniently, this internal label block can also be removed by accessing the Valuelabel Editor with ALT+V and proceeding analogously.

Advantages and disadvantages

The advantage of an internal label block is to have everything in a single file. Unless the label block is very large (how to define that?) it will thus mostly be advantages to do precisely that. If a label block is very large (say the ICD 10 codes), then it might be advantages to actually keep it externally as larger file names will slow down processing time. Furthermore, the current EpiData Analysis version cannot yet make the connection between the main file and an external label block.

Exercise 4: Create a composite identifier

At the end of this exercise you should be able to:

- a. Recognize the efficiency of using date components instead of dates
- b. Creating a unique identifier from several variables
- c. Automatically capturing data entry time

Sometimes we have time information but it is incomplete, such as when a patient may remember the month but not the day of disease onset. In such a case a date field cannot be used sensibly because a date needs all three date components. It becomes thus useful to utilize instead three variables for day, month, and year respectively and then reserve it for the analysis to set rules what approximation might be used if one or more date components are missing. For data entry, this can also have the advantage that some components (like the year, or even the month) are set to Repeat if records are entered serially along a time axis and several sequential records have the identical year and perhaps also month.

A date component might also be used to construct a more complex identifier if the simple one does not suffice. In these exercises using the tuberculosis microscopy laboratory we defined the unique identifier as the laboratory serial number. However, the serial number starts with 1 every calendar year and is thus unique only for a single year. By combining the year plus the serial number, it becomes unique for the laboratory at any time, and adding the laboratory identifier, we can make a unique identifier for any examinee for any year, in any laboratory.

To learn about the workload of data entry, there is a possibility to instruct EpiData to record automatically the computer time at the time of opening a new record and to add another time stamp at the completion of the record.

We will introduce here these three concepts and integrate the computer-calculated fields into a special section.

Capturing time with date components instead of with dates

Export the structure of the `a_ex03.epx` file to a new `a_ex04.epx` file and open the new file. We then start with deleting the field `regdate`, dragging down the block of variables below it, and inserting the three new integer variables `regdd`, `regmm`, and `regyy` of lengths 2, 2, and 4 respectively, for `Registration day`, `Registration month`, and `Registration year`. We make each field `Must Enter` and provide a legal range and put the respective three values 99, 99, and 9999 for not recorded into two different `ValueLabels`. Remember from Exercise 2 that we need to untick the `Show valuelabel text after ...` in the `Extended` tab. We can make the fields for day and month set to `Repeat` as these two remain identical in most laboratories for quite a few sequential records.

Making an identifier composed of 3 existing variables

On the top we section off a part that will accommodate three calculated fields: the new identifier, the start time and the ending time of data entry for the record. Drag thus all fields down on the

canvass and insert a section above them that we call `Calculated fields`. The identifier `id`, labeled `Unique identifier` is a string field of length 14 as we will give it the form `YYYY-labcode-serno`, i.e. the 4-digit year, a hyphen, the 4-digit laboratory code, a hyphen, and the 4-digit laboratory serial number. It is placed inside the section and it has to be a `No Enter` field as we want to prevent a data entry person being able to ever touch a computer-calculated field. The top of our revised data entry form may thus look now as follows:

How do we go about making the value for this identifier? It can only be composed once all three required components have been entered, i.e. after entering the `Registration year`. We thus go to that field and go to the tab `Calculate`, where we choose the bottom option and fill by selection from the drop-down fields and inserting a hyphen in between, that's all there is to it:

A little more sophisticated is the approach to make `id` a unique identifier. Intuitively, we might think that we just make this field `id` the key field, but that is not how it's done: it is each of the components that is set to be the key and EpiData will then correctly interpret the resulting combined field required to be unique. It may appear a bit counter-intuitive, but that's how it's done. Remember where to `Define Key`? Right-click the data form name or go to the menu point `Dataform`:

Tasks:

- o Add two fields for *Starting time* and *Ending time* for data entry in the section *Calculated fields*.
- o Test the data entry form in *EpiData EntryClient*

Solution to Exercise 4: Create a composite identifier

Key Point(s):

- If one variable is insufficient to create a unique identifier, one can construct a composite identifier from more than one variable.
- Each of the contributing fields, but not the composite resulting variable is set to be Key.
- EpiData provides automatically calculated time fields that can be used to capture data entry time.

Tasks:

- o Add two fields for Starting time and Ending time for data entry in the section Calculated fields.
- o Test the data entry form in EpiData EntryClient

Solution:

A complete record may look as follows:

Tuberculosis Microscopy Laboratory

Calculated fields

Unique identifier

Starting time Ending time

Laboratory Collin Saunders

Laboratory serial number

Registration day

Registration month

Registration year

Examinee's sex Female

Examinee's age in years

Reason for examination Diagnosis

Result of specimen 1 Negative Result of specimen 1 scanty

Result of specimen 2 Scanty, quantified Result of specimen 2 scanty 5 AFB per 100 OIF

Result of specimen 3 Result not recorded Result of specimen 3 scanty

Exercise 5: Data entry and validation

At the end of this exercise you should be able to:

- Know the three ways of reducing data entry errors
- Copy the structure of a EPX file
- Export data from EpiData files
- Validate duplicate data files

You have a line listing of 15 records on the page following the task description. These data should be entered in this exercise. But before you start working, a few considerations are in place.

Ensuring quality data entry

The motto for this course is:

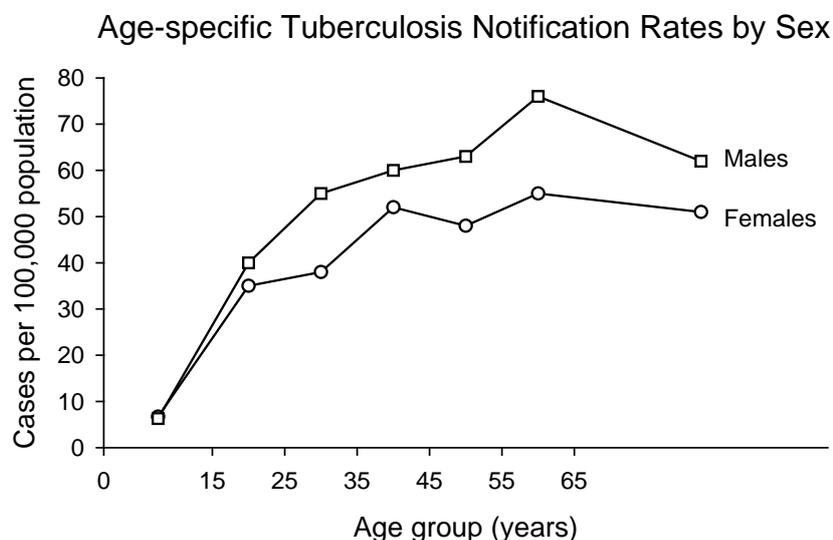
“You wish never to find yourself in a position to defend the quality of your data”

Michael B Gregg, formerly MMWR Editor, deceased

See also: Gregg M B. Field epidemiology (2nd ed). Oxford University Press. 2002: p 414

You might be challenged about the interpretation of your data, that is part of the scientific process, but your data should be of impeccable quality.

What do you think about the following graph?



It looks nice and we could talk about the differences between males and females and this and that. But we will keep it short: it is nonsense. The data underlying this graph have no basis, they were made up. Of course, if we were to present these data for real, it would be outright scientific fraud. Few people commit that (but it exists). *Nevertheless, often no assurance can be given that the computerized data are a true reflection of the original data source.* People may have in all honesty done “their best” and assume that they made no errors or so few that it really doesn’t matter. However, this is not good enough for science in general and public health and epidemiology in particular.

There are three ways how we reduce and ultimately eliminate data entry errors:

- o Defining well thought-through data entry controls in the EpiData Manager
- o Working together
- o Duplicate data entry and validation

Defining well thought-through data entry controls in the EpiData Manager

We have already a few inbuilt conditions that limit data entry errors by creating the `a_ex04.epx` file. For instance, a Must Enter field will prevent a data entry person to skip an actually recorded value, as one cannot continue without having entered a value for that field. For the field `sex`, we allowed only 1, 2, and 9 as legal values. It is thus not possible to enter “3” into this field. Combined with the pop-up menu during entry, no confusion can arise. The controls set in the EpiData Manager are an extremely powerful tool to control how data entry can be controlled through restrictions.

Working together

Entering data alone requires continuously shifting attention between the paper record and the computer screen. This will almost by necessity result in numerous errors, be it that a record is skipped or that it is forgotten what we just read. It should be routine that two persons work on data entry: one person reads aloud the Field value, the other repeats it aloud and enters the value.

Duplicate data entry and validation

Even with both of the above precautionary measures, data entry errors will still occur, and worse, to an unknown extent. ***The only way, and the only acceptable one, is to enter the data twice into two different files, and then to compare the two files for discordances.*** Any discordance uncovered will then be corrected the entry with the original paper record.

The rationale behind this process is: *the probability of committing the same error in the same field twice when data entry is done independently by two persons is very small.* Hence, if we list all the discrepancies by comparing the two databases and correct all of them, then we can be reassured that the remaining frequency of data entry errors is miniscule.

EpiData provides this powerful tool and we need a unique identifier to do this. We have made a provision that we have such an identifier (see previous exercises). Sometimes an identifier must be constructed from more than one variable as we have shown.

If a duplicate key is revealed (because there is a perhaps a problem with a component contributor), then a data entry note should be written, best as a text file that is kept open during data entry and amended as one proceeds. In this note, you must specify exactly with what identifier you have replaced the duplicate key, so that this note can be passed on to those who

enter the data the second time, enabling them to use the same alternative key when the necessarily stumble over the same problem.

Before you get to actually enter the data, you find here some assistance to make your data entry work more efficient.

Make duplicate EPX files

As we are entering the same data twice, we need two sets of the *.epx files, one for the first, the other for the second entry.

Task:

- o Download the solution of Exercise 4 and save the file as a_ex05_a.epx and a_ex05_b.epx files*

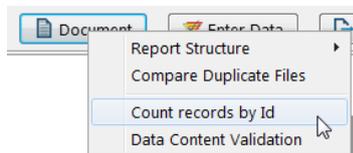
Double-entry

Enter the 15 records into the a_ex05_a.epx, then repeat data entry with the a_ex05_b.epx file.

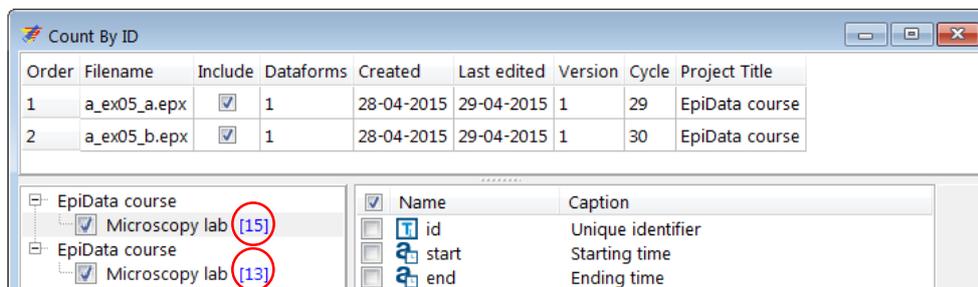
Data validation

After completing double-entry, the two data files are compared, a procedure termed “**data validation**”.

The first thing to verify is that we have the same number of records in both sets. If that is not the case, then this must be fixed first. Here for instance, we chose to count the records:

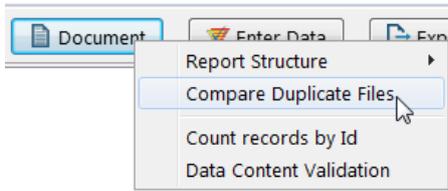


and found them to be unequal:

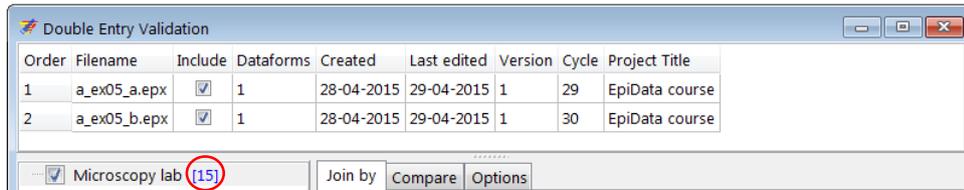


While we know from this small dataset that there actually are 15 records, and thus that the first set has the correct number and the second is missing 2, knowing the expected number of records is the exception in real life. It may also very well be that both sets have the same number of records but both are short of records because by chance each set is missing the same number of

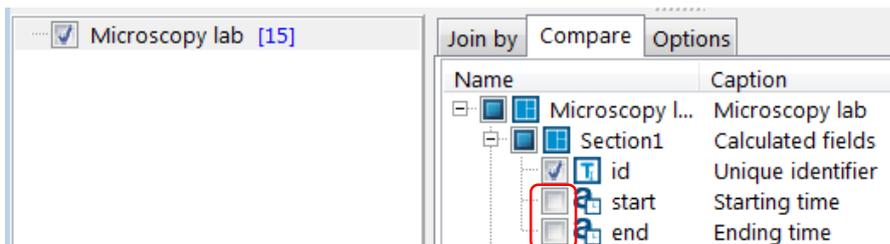
records. We thus have to check which records are missing and we do this by a validation, actually comparing the two sets:



The first of the two files is the “referent” and it is the number of records in this set that is displayed:



In the tab Compare, we untick the computer-provided time fields because they are likely to vary for virtually every record between the two files and would undesirably be listed as discordances:



In the report, we move to the Overview and find the confirmation that the duplicate file (the *_b.epx file) has 2 records missing:

```
-----
Result of Validation:
-----

Overview
-----
Test                                     Result
Records missing in main file             0
Records missing in duplicate file        2
Non-unique records in main file          0
Non-unique records in duplicate file     0
Number of fields checked                  12
Common records                           13
Records with errors                       1
Field entries with errors                 1
Error percentage (#records)               7.69
Error percentage (#fields)                0.64
-----
```

Moving further down, we find the serial numbers missing from the second file and thus know which two records must be added first before a proper validation is possible:

```
Record no: 14
Key Fields:
lab = ML J
serno = 3310
regyy = 2003   Record not found
-----
Record no: 15
Key Fields:
lab = ML J
serno = 3311
regyy = 2003   Record not found
-----
```

The Validation report

We add these two records and then start again the validation process. This time we get the same number of records in the two files and find one discordance in serno 3302:

```
-----
Result of Validation:
-----

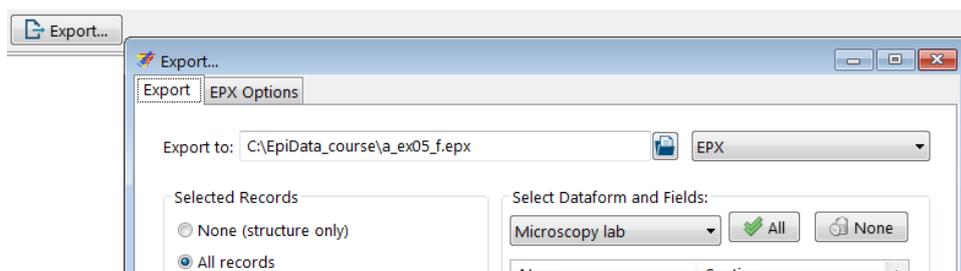
Overview
-----
Test                                     Result
-----
Records missing in main file             0
Records missing in duplicate file        0
Non-unique records in main file          0
Non-unique records in duplicate file     0
Number of fields checked                 12
Common records                           15
Records with errors                       1
Field entries with errors                 1
Error percentage (#records)              6.67
Error percentage (#fields)               0.56
-----

Datasets comparison:
-----
Main Dataset:      Duplicate dataset:
-----
Record no: 5      Record no: 5
Key Fields:
  lab = ML_J
  serno = 3302
  regyy = 2003
Compared Fields:
  sex = 2          sex = 1
-----
```

It is essential that this validation report is saved to ensure having a permanent record of the validation process. We propose to save it as a text file `a_ex05_validation.txt`.

Creating a final dataset

One might be tempted to make corrections of any errors that might be identified through discordances in either the `*_a.epx` or the `*_b.epx` file. Doing so would, however, break the “chain of evidence”: you could never repeat the validation process and get the same result, but data quality-assurance requires that the validation process is actually exactly reproducible. Therefore, the corrections must be made in a third file. To this end, we export the data from one of the source files to another EpiData file that we will call the `a_ex05_f.epx` file. To standardize as many things as possible, we always export the `a_ex05_a.epx` file to the `a_ex05_f.epx` file (even if in fact it is irrelevant whether we use the `a_ex05_a.epx` or the `a_ex05_b.epx` file, but consistency is good policy). We thus select from Export the `a_ex05_a.epx` file and define in the menu the name and type and All records:



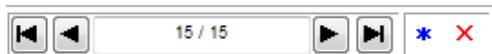
We get a report that export was successful:



Back from Manager to the EntryClient we open a_ex05_f .epx and search the record with serno 3302.

How to navigate through an * .epx file?

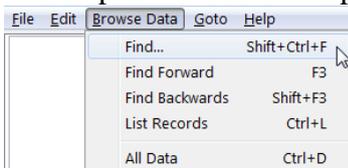
To navigate between the records of the * .epx file use the navigation panel on the left bottom end of the data entry screen which can be used to navigate through the records.



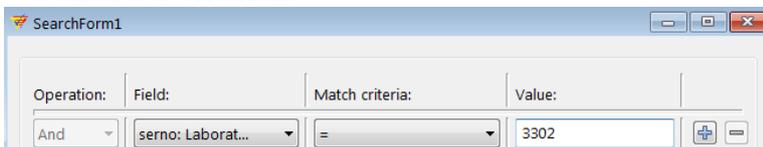
Shown vertically one by one:

- Go to first record
- Go to previous record
- 11 / 15 This is record 11 out of a total of 15 records
- Go to next record
- Go to last record
- Insert a new record
- Mark current record for deletion

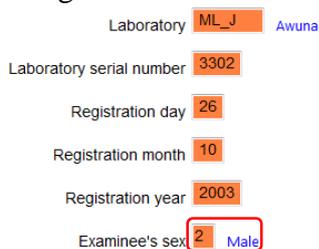
This is useful for a quick forth and back, mainly during data entry, but for the current task to find a specific record in a potentially large file, we use Browse Data | Find:



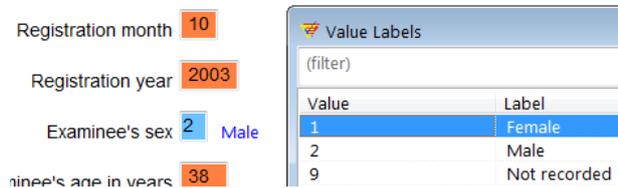
We enter our criterion:



and get this to the record:



Comparing with the original paper record, we see that the true Examinee's sex should be female. Moving the cursor into the field and pressing **F9**, we can now pick the correcting value:



As this is the only discordance we save the revised record and exit. We now have a validated file with all discordances resolved.

How to delete a record?

Deleting a record consists of two steps – first, marking a record for deletion; second, permanently deleting it. This is just a safety feature in EpiData to ensure the deletion of record does not happen by chance.

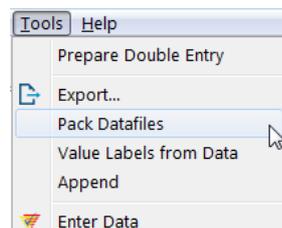
Steps in marking a record for deletion (Look at the screenshot below)

1. Open the *.epx file and go to the record you want to delete.
2. Click on the red 'cross' mark next to the navigation panel at the left bottom of the data entry screen. The word DEL appears at the side of the red 'cross' mark.
3. Click the arrow in the navigation panel to go to the next record. This will prompt you to save the record. Click 'Yes' and this successfully marks the record for deletion.
4. Note that the record is not yet permanently deleted from the database. If you realize that this record was not to be deleted, you can undo the action by clicking on the same button and saving the record. DEL will disappear now: the red "cross" is a toggle key:



How to permanently delete a record? (Pack Data Files)

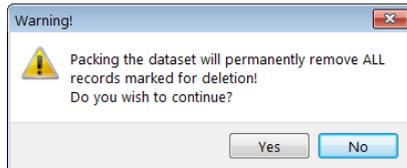
For data protection, it is not foreseen to permanently delete a record in the EpiData EntryClient. This must be done in the EpiData Manager. Close thus the file (if open) in EpiData EntryClient and go to the Manager to Tools | Pack Datafiles:



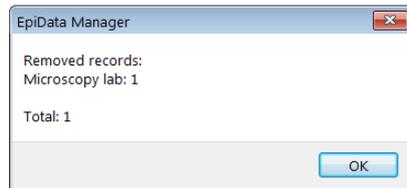
Choose the data file and tick the data form (it could be a relational data base with several forms and you got to choose which one):



Because this is going to be permanent, you receive a final Warning:



After accepting, you get confirmation that it is done:



No backup file is written, this is permanent.

Tasks:

- o Download the solution of Exercise 4 and save your a_ex04.epx file as a_ex05_a.epx and a_ex05_b.epx.*
- o Enter the 15 records using the a_ex05_a.epx file. After completing data entry, enter the same data again into the a_ex05_b.epx file.*
- o After you have completed the two files, proceed to validation as explained here.*
- o After ensuring that no record is missing in either file, export the a_ex05_a.epx file to a a_ex05_f.epx file, check out the discordances if any and correct them. This is your final dataset.*

On the next page you find the dataset with 15 records

Laboratory: Awuna

Tuberculosis laboratory register

Year: 2003

Lab Serial No.	Date specimen received	Name	Sex M/F	Age	Name of referring facility	Address - patient for diagnosis	Reason for examination*		Results of specimen			Only for SS+ for diagnosis: TB Number or BMU**	Remarks
							Diagnosis (tick)	Month of follow up	1	2	3		
3298	26 Oct	Mary	F	35	Bindura	Beijingstr. 6		5	neg	neg			
3299	26 Oct	John	M	20	Awuna	Tokyo Ave 5	√		neg	neg	neg		
3300	26 Oct	Petra	F	30	Birchenough	Bangkok Rd 108		5	neg	neg			
3301	26 Oct	Charles	M	24	Bindura	Hanoi Street 7a		2	neg	neg			
3302	26 Oct	Tiffany	F	38	Bindura	Hongkong Ave 8	√		neg	neg	neg		
3303	26 Oct	George	M	60	Bindura	Zurich Rd 923	√		neg	neg	neg		
3304	26 Oct	Luke	M	78	Awuna	Paris Street 18a	√		neg	neg	neg		
3304	26 Oct	Virginia	F	28	Birchenough	London Rd 24	√		neg	neg	neg		
3305	27 Oct	David	M	50	Awuna	Baltimore Str 1		6	neg	neg			
3306	27 Oct	Hans	M	50	Ganda Chivua	Bern Str 12	√		1+	1+	1+	Ganda Chivua No 342	
3307	27 Oct	Bill	M	68	Bindura	Berlin Ave 88	√		neg	neg	neg		
3308	27 Oct	Susan	F	29	Birchenough	Amsterdam Rd 3		5	neg	neg			
3309	27 Oct	Marc	M	36	Bindura	Vienna Str 76		2	neg	neg			
3310	27 Oct	Eve	F	15	Awuna	Rome Ave 4		5	neg	neg			
3311	27 Oct	Anthony	M	37	Birchenough	Antwerp Str 26c		6	neg	neg			

* Check the appropriate category from the Request for Sputum Examination

**TB register number

Solution to Exercise 5: Data entry and validation

Key Point(s):

- It should be routine that two persons work on data entry, and never one.
- The only and acceptable way to minimize data entry errors is to enter the data twice into two different files, and then compare the two files for discordances.
- Avoid using the mouse to move around fields during data entry, because the Check file cannot be applied to fields you skip by moving the mouse from one field to another.

Tasks:

- o *Download the solution of Exercise 4 and save your a_ex04.epx file as a_ex05_a.epx and a_ex05_b.epx.*
- o *Enter the 15 records using the a_ex05_a.epx file. After completing data entry, enter the same data again into the a_ex05_b.epx file.*
- o *After you have completed the two files, proceed to validation as explained here.*
- o *After ensuring that no record is missing in either file, export the a_ex05_a.epx file to a a_ex05_f.epx file, check out the discordances if any and correct them. This is your final dataset.*

Solution

Depending on the errors you made, you will get an output like the following:

```
=====
Report: Double Entry Validation Report.
Created: 29-04-2015 22:03:30
=====
```

```
-----
File 1: C:\EpiData_course\a_ex05_a.epx
File 2: C:\EpiData_course\a_ex05_b.epx
-----
```

```
File 1: C:\EpiData_course\a_ex05_a.epx
-----
```

```
Title      EpiData course
Created    28-04-2015 09:50:32
Last Edited 29-04-2015 20:33:53
Version    1
Cycle      29
-----
```

```
Backup on shutdown: yes
Encrypted data: no
```

```
Dataforms:
```

```
-----
Caption      Created      Structure Edited  Data Edited  Sections Fields
Records Deleted
-----
```


Caption Fields in key

Microscopy lab (serno:Laboratory serial number) + (regyy:Registration year) + (lab:Laboratory)

=====
DataForm: Microscopy lab
=====

Selections for validation:

Options:

Option Selected

Ignore deleted records No
Ignore missing records No
Add result to field No
Case sensitive text No

Key Fields:

lab serno regyy

Compared Fields:

id: Unique identifier
regdd: Registration day
regmm: Registration month
sex: Examinee's sex
age: Examinee's age in years
reason: Reason for examination
res1: Result of specimen 1
res1sc: Result of specimen 1 scanty
res2: Result of specimen 2
res2sc: Result of specimen 2 scanty
res3: Result of specimen 3
res3c: Result of specimen 3 scanty

Result of Validation:

Overview

Test Result

Records missing in main file 0
Records missing in duplicate file 0
Non-unique records in main file 0
Non-unique records in duplicate file 0
Number of fields checked 12
Common records 15
Records with errors 1
Field entries with errors 1
Error percentage (#records) 6.67
Error percentage (#fields) 0.56

Datasets comparison:

Main Dataset: Duplicate dataset:

Record no: 5 Record no: 5

Key Fields:
lab = ML_J
serno = 3302

```
regyy = 2003
Compared Fields:
sex = 2          sex = 1
-----
```

After making correction in the “F” file, your data should be correct, or are they not? While your final data file should be correct, there is still a slim chance that it has errors. How is this possible? If by chance the same error was entered in both files (which can happen particularly if the same person enters the data in both files), you will not be able to identify the error. For uniformity, you should overwrite your existing file with the `a_ex05_f.epx` file that is provided with the solution.

Exercise 6: Upgrading an EpiData 3.1 REC/CHK file pair to an EPX file

At the end of this exercise you should be able to:

- a. Understand how you import an EpiData Entry 3.1 REC / CHK file pair into EpiData Manager
- b. How to edit the new file and add the link to the metadata from the old CHK file to the new EPX file.

Importing an EpiData REC / CHK file containing data

In the required zip folder you find 4 data sets, a, b, c, and d, complete with all 16 files, i.e. for the “a” files a.qes, a.rec, a.chk, and a.eix, containing 75 records, and analogously for the “b”, “c”, and “d” sets. It contains microscopy laboratory data from four laboratories in Yangon (Myanmar), courtesy Dr Ti Ti. If you look at the A.QES file in EpiData Entry 3.1, you see the visual format:

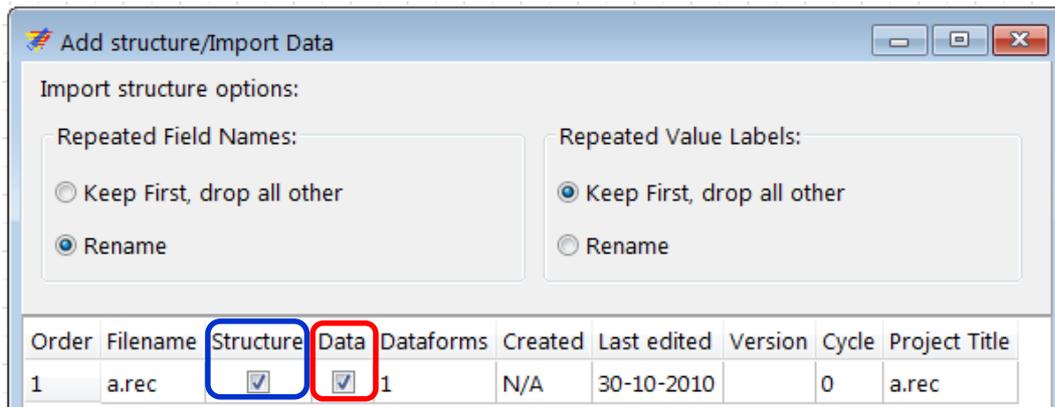
```
id          Unique laboratory identifier  _____

serno       Laboratory serial number  ####
labcode     Laboratory code           _
regdate     Registration date          <dd/mm/yyyy>
age         Examinee's age in years      ##
sex         Examinee's sex          #
reason      Examination reason          #
res1        Result of specimen 1        #.#
res2        Result of specimen 2        #.#
res3        Result of specimen 3        #.#
```

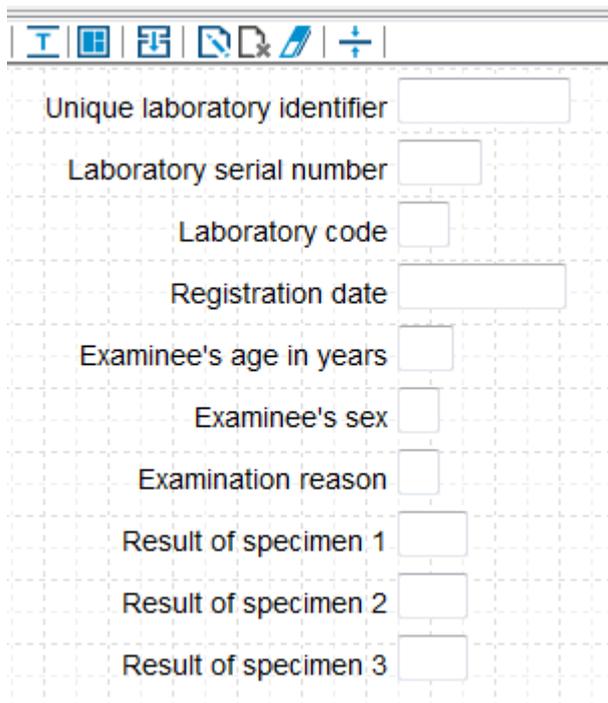
We note that the results were defined as floats with the following values (the definition of which we can best see by opening the A.CHK file (e.g. in a text editor):

```
LABEL label_result
0.0 Negative
0.1 "Scanty, 1 AFB per 100 fields"
0.2 "Scanty, 2 AFB per 100 fields"
0.3 "Scanty, 3 AFB per 100 fields"
0.4 "Scanty, 4 AFB per 100 fields"
0.5 "Scanty, 5 AFB per 100 fields"
0.6 "Scanty, 6 AFB per 100 fields"
0.7 "Scanty, 7 AFB per 100 fields"
0.8 "Scanty, 8 AFB per 100 fields"
0.9 "Scanty, 9 AFB per 100 fields"
1.0 "1+ positive"
2.0 "2+ positive"
3.0 "3+ positive"
4.0 "Positive, not quantified"
5.0 "Scanty, not quantified"
9.0 "No result recorded"
END
```

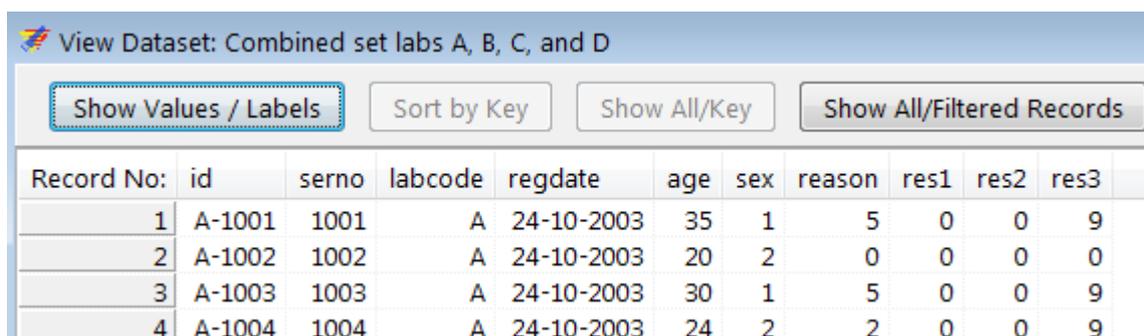
In EpiData Manager, go to File | Import File... and pick the a.rec to see it then in an editable box:



You note that we encountered this box in an earlier exercise when we imported a value-label set. Note that we can import the structure only or both the structure and the data, as we plan to do here. Once confirmed with OK we get:



and see at the bottom left that there are indeed 75 records which we can browse in **Dataform | Browse Data** (shortcut **CTRL+D**) (first lines only shown):



Note that the button Show Values / Labels is non-functional because the labels are not yet available, we thus see only the values at this point in time. Save the file when prompted as a_ex06a.epx.

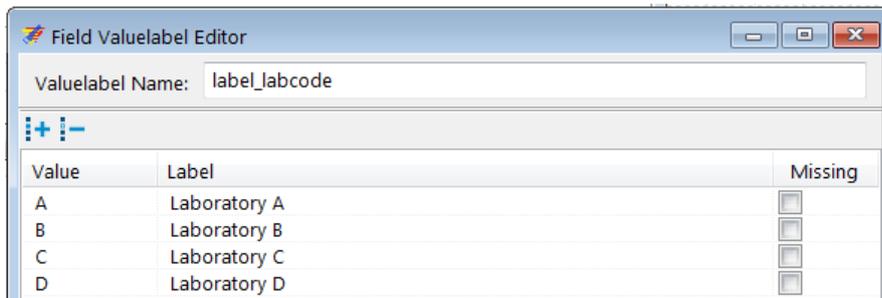
Make the metadata from the CHK file available to the EPX file

We work our way through the entry form from top to bottom.

The Unique laboratory identifier (field name `id`) is a calculated value from the Laboratory code and the Laboratory serial number. It is thus a field set to No Enter and calculated after both components that make it up are available.

The Laboratory serial number (field name `serno`) is Must Enter and we Add leading zeros.

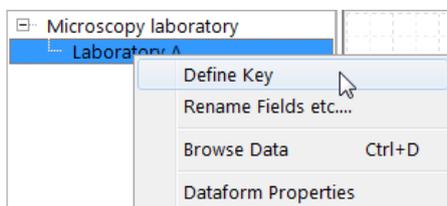
The Laboratory code (field name `labcode`) is Must Enter. We noted above in browsing that the value was “A”. The field does also not have a Label Block (the Laboratory code was added only after data entry in the analysis module). As we will have also laboratories B, C, and D, we will add here a Label block with string values to later accommodate the three additional laboratories. Thus create a New Value Label as:



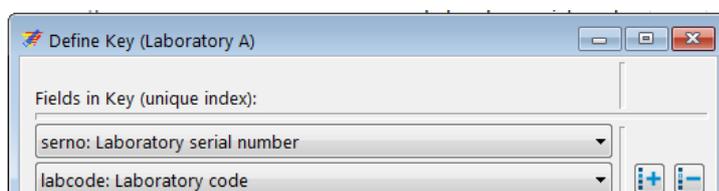
As we do with variables with label blocks, we tick Always show picklist during entry. As we have now all the necessary information we Combine Fields in the Calculate tab:



Let's also make both `labcode` and `serno` Key fields (right-click the data form [which we have here already properly relabeled]):

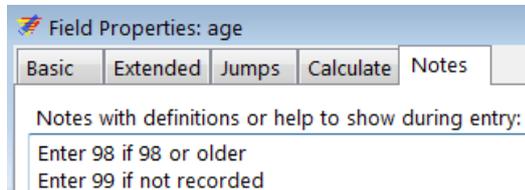


and make `serno` and `labcode` Key fields:



The Registration date (field name `regdate`) is Must Enter and as all laboratory results are from the year 2003 we might as well add a Range from 01/01/2003 to 31/12/2003.

The Examinee's age in years (field name `age`) is Must Enter and as a continuous variable of length 2 we may add Notes:



Field Properties: age

Basic Extended Jumps Calculate Notes

Notes with definitions or help to show during entry:

Enter 98 if 98 or older
Enter 99 if not recorded

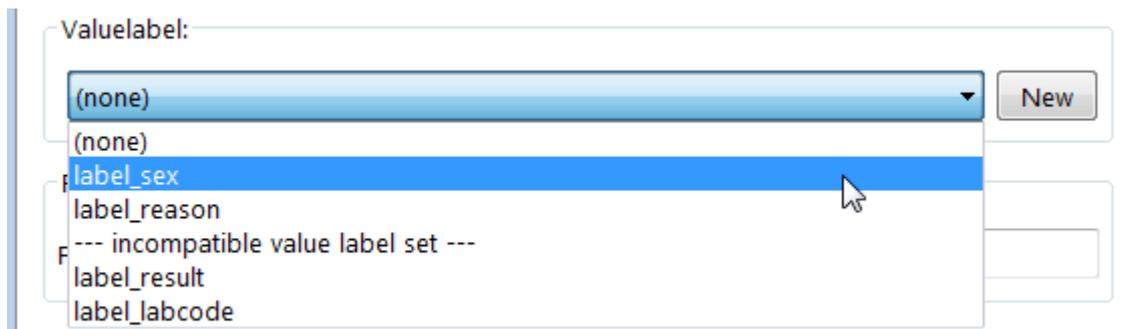
The Examinee's sex (field name `sex`) has no Valuelabel ticked:



Valuelabel:

(none) [dropdown arrow] New

However, if you pick the drop-down list, you see that it actually exists (it is in the CHK file) and all we need to do is to pick it:

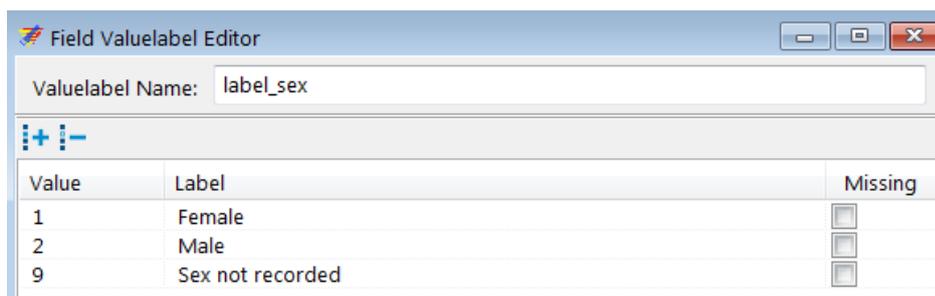


Valuelabel:

(none) [dropdown arrow] New

- (none)
- label_sex
- label_reason
- incompatible value label set ---
- label_result
- label_labcode

and then we can view it with Edit that replaces New once picked:



Field Valuelabel Editor

Valuelabel Name: label_sex

Value	Label	Missing
1	Female	<input type="checkbox"/>
2	Male	<input type="checkbox"/>
9	Sex not recorded	<input type="checkbox"/>

The list of Value labels also shows that we have `label_reason` and `label_result` which we can use for the last four variables to complete updating the new EPX file with the metadata from the old EpiData Entry CHK file.

Finally, once the above is done, we may edit it nicely and check functionality in the EpiData EntryClient (without saving the record!!):

Tuberculosis Microscopy Laboratory Laboratory A

Unique laboratory identifier

Laboratory serial number

Laboratory code Laboratory A

Registration date

Examinee's age in years

Examinee's sex Female

Examination reason Diagnosis

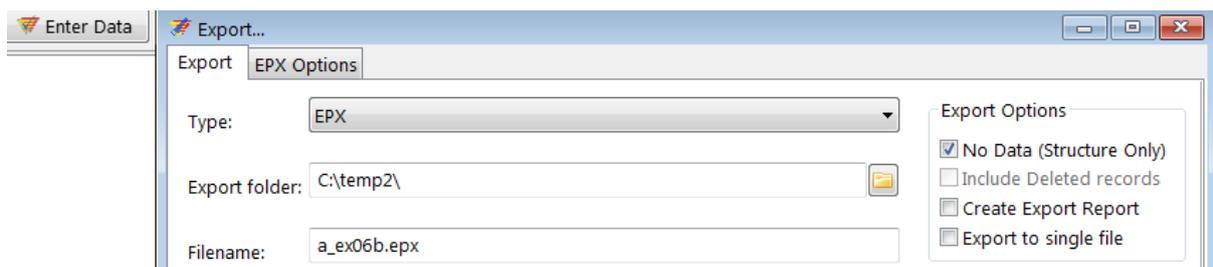
Result of specimen 1 1+ positive

Result of specimen 2 2+ positive

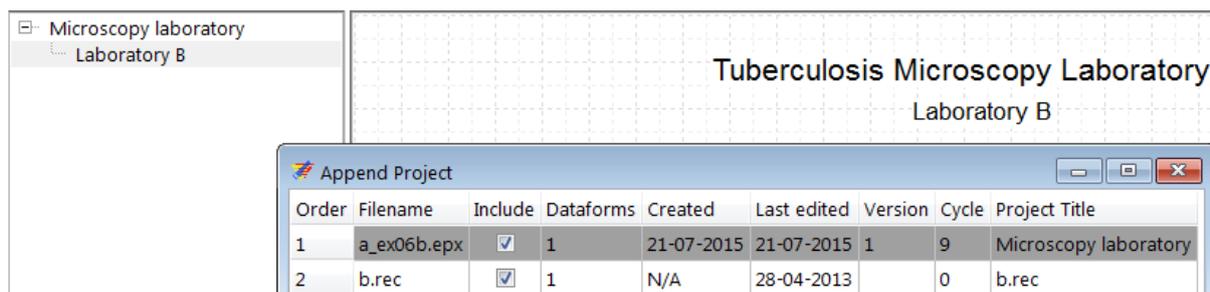
Result of specimen 3 No result recorded

Exporting the structure of an EPX file and importing REC file data into the new structure

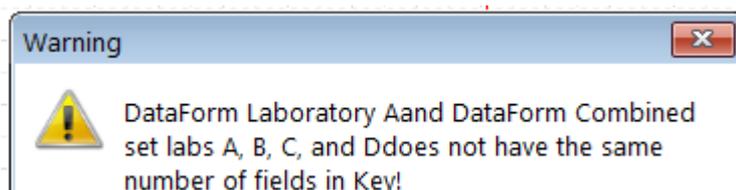
The data form structure of the now created a_ex06a.epx file can be exported without the data using Export and ticking the appropriate box. We export it to a new file a_ex06b.epx:



We then open this a_ex06b.epx file, edit it (changing the name of the data form and the header to become “Laboratory B”) and use Tools | Append and pick the b.rec file:



However, doing so, we get an error message:



The reason is that in the EpiData Entry REC file serno is the key, and in this EpiData Manager EPX file serno and labcode are the keys. The best approach is to remove the b.chk file

and to remove both keys in the EPX file and subsequently add them again after successful import which is prompt if these actions are done:



If we wanted to have all four laboratories in one single a_ex06abcd .epx file, we could have done that simply by adding all four REC files (after removing keys in the EPX file and removing the CHK files for the four REC files after successfully using their metadata). However, it cannot be done in one sweep, each must be added on its own.

Task:

- o Create the entire set of four EPX files, so that in the end we have a_ex06a.epx, a_ex06b.epx, a_ex06c.epx, and a_ex06d.epx*

Solution to Exercise 6: Upgrading an EpiData 3.1 REC/CHK file pair to an EPX file

At the end of this exercise you should be able to:

- Understand how you import an EpiData Entry 3.1 REC / CHK file pair into EpiData Manager
- How to edit the new file and add the link to the metadata from the old CHK file to the new EPX file.

Task:

- Create the entire set of four EPX files, so that in the end we have *a_ex06a.epx*, *a_ex06b.epx*, *a_ex06c.epx*, and *a_ex06d.epx*

Solution:

We have four files with an identical structure (the Headers differ, but these are just labels) with different records (but each file has 75 records):

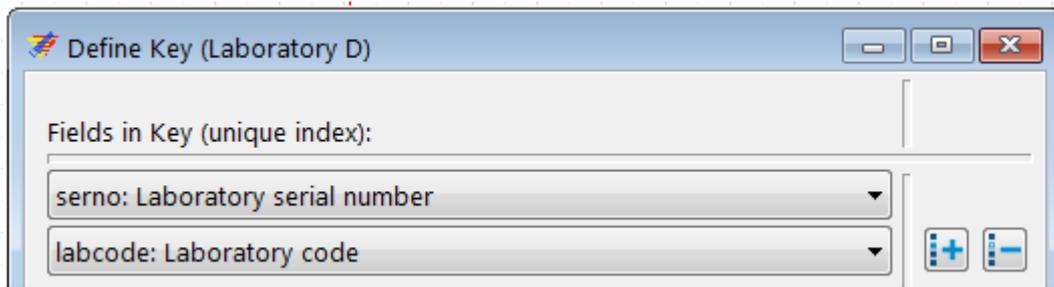
a_ex06a.epx
a_ex06b.epx
a_ex06c.epx
a_ex06d.epx

The final data form for a file looks like this:

The screenshot displays the EpiData Manager interface. On the left, a tree view shows the project structure: 'Microscopy laboratory' containing 'Laboratory D'. The main area shows a data form titled 'Tuberculosis Microscopy Laboratory Laboratory D'. The form is set against a grid background and contains the following fields:

- Unique laboratory identifier
- Laboratory serial number
- Laboratory code label_labcode
- Registration date
- Examinee's age in years
- Examinee's sex label_sex
- Examination reason label_reason
- Result of specimen 1 label_result
- Result of specimen 2 label_result
- Result of specimen 3 label_result

You have ensured that the two keys are added again after import in each of the four files:



While we are not going to enter new records into any of these files and therefore this adding of the keys is not essential, it would be if new records were added to ensure that each resulting combined identifier is unique.

Exercise 7: A relational database

At the end of this exercise you should be able to:

- a. Understand when a single data entry form and when a relational database is the preferred data collection instrument
- b. Create a relational database for a varying number of observations.

Not all laboratories keep their registers as The Union and WHO recommend for the Tuberculosis Laboratory Register, where 1 line corresponds to 1 examinee rather than to 1 examination. In fact, many laboratories enter the results for each examination on one line. If such an approach is chosen, we may find a register as follows:

Patient	Date of exam	Sex	Marital status	Blood sugar	Sputum	Result
A	24-Mar-2007	Male	Married	6.3	Mucoid	1+
B	24-Mar-2007	Male	Divorced	4.9	Muco-purulent	Neg
C	24-Mar-2007	Female	Single	5.2	Purulent	Neg
D	24-Mar-2007	Female	Widowed	7.3	Blood-tinged	2 per 100
A	25-Mar-2007	Male		7.3	Salivary	Neg
D	25-Mar-2007	Female		7.4	Mucoid	2+
A	26-Mar-2007			7.2	Purulent	1+
C	26-Mar-2007	Female		4.8	Muco-purulent	Neg
E	27-Mar-2007	Male	Married	8.2		1+
F	27-Mar-2007	Female	Annulled	7.4	Purulent	Neg
G	27-Mar-2007	Male	Cohabiting	6.9	Salivary	Neg
G	28-Mar-2007	Male		7.2	Mucoid	2+
E	28-Mar-2007	Male		7.9	Purulent	2+
F	31-Mar-2007	Female		7.2	Muco-purulent	3+
H	31-Mar-2007		Married	6.6	Mucoid	Neg
I	31-Mar-2007	Male	Separated	8.3	Salivary	Neg
H	1-Apr-2007	Female		6.9	Muco-purulent	1+
F	1-Apr-2007	Female	Engaged	7.7	Purulent	2+
I	1-Apr-2007	Male	Single	8.0	Mucoid	8 per 100
G	1-Apr-2007	Male		7.6	Muco-purulent	1+
K	2-Apr-2007	Female	Married	4.5	Purulent	Neg
I	2-Apr-2007	Male		8.2	Muco-purulent	
H	2-Apr-2007	Female		6.6	Mucoid	1+
I	3-Apr-2007	Male		8.1	Mucoid	1+

This type of a register requires a different approach to data entry than we used before. Two important things need to be considered:

- 1) The same patient may appear again and again on sequential dates
- 2) Not every patient has the same number of visits

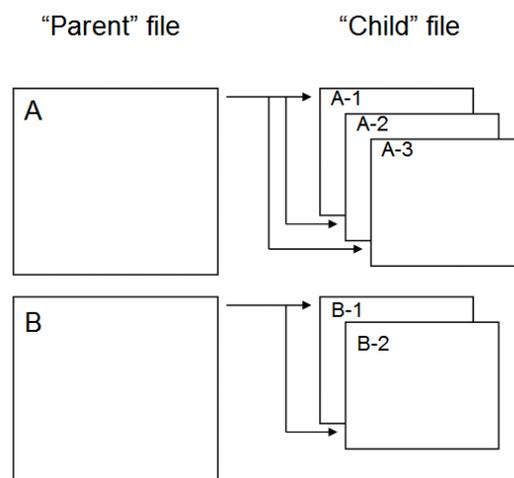
Some patient characteristics do not change over time such as, in this example, the identity of the patient, sex, and marital status (well, perhaps not during these short intervals). Others do change, such as the date of examination, blood sugar, the aspect of the sputum and the sputum smear examination result.

To capture such information in a single data entry form as was done in the previous exercises would be very inconvenient: 1) one would have to anticipate the maximum of allowable visits, and 2) if one patient has a single visit, one would still have to complete all fields with the codes for missing values up to the maximum allotted if we insist that all fields must usually be Must Enter fields.

Rule: *If an individual has a fixed number of observations for each variable, then a single EpiData form is the best solution. If an individual has a variable number of observations for each variable, the choice is a **relational data base**.*

Building a relational database requires deciding which information is static for an individual (during the observation period) and which information changes over repeated observations. We will illustrate a relational database with a very limited set of variables.

What is the structure of such a relational database? The figure below outlines a relational database with two levels.



At the Level 1 (the "Parent" file), we may have patients, denoted here with A and B. At Level 2 (the "Child" file), denoted here as A-1, A-2, and A-3, we may have different visits to a clinic where each time the date of the visit is recorded, and blood pressure blood sugar are measured. Each individual may have at least one, but up to an unlimited number of such visits, and the number of visits varies for each individual.

Note: *Because of the simplicity, it is always preferable to have a single data entry form. However, if the data available concern an individual with fixed information (e.g. age, sex, etc) on one hand, and variable information (e.g. serial examinations) and there are fixed sets of variables in serial examinations (e.g. repeated measures of blood sugar and blood pressure) and each examinee has a varying number of examinations, then it may become more efficient to use a relational database.*

In the following, we will call the records at Level 1 the parent records and those at Level 2 child records.

To build the relational database, we collect different information at each level and link the two levels, so that at some point during entering parent information will trigger the opening of the

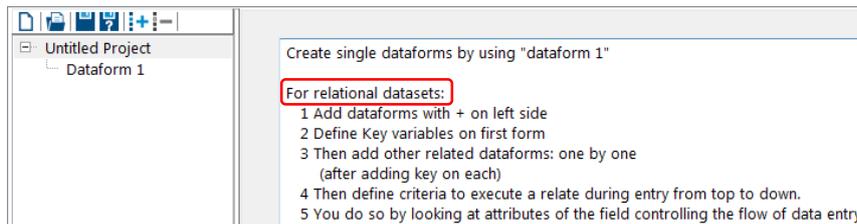
child form to enter one or more child records. One or more child records are added until one wishes to return back to the upper level of parent records.

Important components for the structuring of the relation between the two data forms are:

- The two forms have a common identifier field. This identifier might be called `idparent`.
- The child form must have its own unique identifier, which might be called `idchild`.

Illustrating it practically in EpiData Manager

We start a New Project. We note the brief information on the relational database in the opening window:



Change the name of the Untitled Project to Exercise 7 and Dataform 1 to `1_parent` and save the project as `a_ex07.epx`.

We add as first field the string variable `idparent`, the Unique parent identifier of length 2 as a Must Enter field. We add only two fields: `age`, an integer of length 2 for Patient's age in years and `sex` for Patient's sex, an integer of length 1. Both are Must Enter fields. For `age` as a continuous variable, we might write in Notes:

Enter 0 to 97 for exact age
 Enter 98 if 98 or older
 Enter 99 if not recorded

Finally, we define `idparent` as Key field. Next we make the `2_child` Dataform (rename it using **F2**) by clicking on the:



And we note that EpiData opens the new data form with the `idparent` identifier already written onto the canvass:



If you click on it, you see that in the Field Properties menu everything is grayed out and in its Extended tab we also see the grayed out setting to No Enter, all courtesy of EpiData Manager!

While we are in the `2_child` data form, we add the four variables `visit`, `syst`, `diast`, and `bs`, for Visit date, Systolic blood pressure, Diastolic blood pressure, and Plasma glucose in mMol respectively, a date, two integer and one float field, all Must Enter (and define some range, but assume no missing values). Preceding

Visit date, add idchild, the Unique child identifier, a No Enter string field of length 13, resulting from the combination of the values of idparent and visit.

This is all there is to it: the flow of the related forms is such that once the 1_parent is completed, the flow leads right over to the 2_child data form, where you can complete from one to many records before you move back up to a new 1_parent data form.

Other options

You can add more related forms. If there is more than one data form, you can choose under Properties of the 1_parent data form (right-click its name) in the After Record tab the order of flow which is to be followed after the parent record:



For a single related form as in our example, there is only one possibility, and this is pre-recorded as default.

You can also set conditions in a field of the 1_parent form, i.e. which value in the field should trigger access to the related field. As of now, there is still some bug that should be fixed shortly that prevents this extended functionality to work as intended.

Task:

- o Complete the related data set and check it for functionality in the EpiData EntryClient*

Solution to Exercise 7: A relational database

Key Point(s):

- A relational database is appropriate when a dataset contains fixed person information (such as age, sex, hair color, and the like) and variable information like examination of weight and blood sugar over a variable number of examination visits.
- A unique identifier in the parent file serves as the key connecting the parent file with the child file

Task:

- o Complete the related data set and check it for functionality in the EpiData EntryClient

Solution:

A completed parent and child record set may look as this:

Relational database: Parent file

Unique parent identifier
Patient's age in years
Patient's sex Female

Relational database: Child file

Unique parent identifier
Unique child identifier
Visit date
Systolic blood pressure
Diastolic blood pressure
Plasma glucose in mMol

[Press F10 or click on the 1_parent form to return to it](#)

Part B. EpiData Analysis

Part B: EpiData Analysis

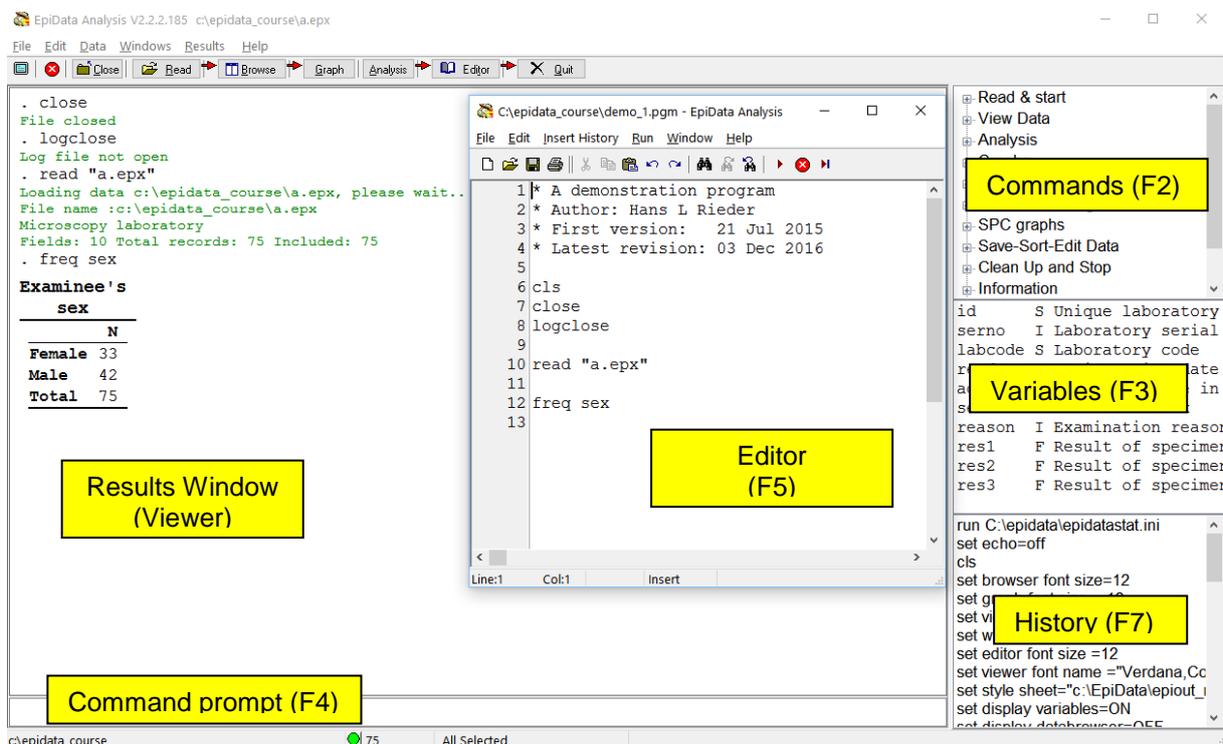
- Exercise 1 An introduction to EpiData Analysis
- Exercise 2 More on EpiData Analysis
- Exercise 3 Aggregating data and saving the summary data in a file
- Exercise 4 From a spreadsheet to an EpiData file

Exercise 1: An introduction to EpiData Analysis

At the end of this exercise you should be able to:

- Do some analyses using the commands on the menu bar.
- Use the 'Editor' to write commands into a program that can be saved to make a permanent record.
- Do some calculations.
- Create new variables.

The image below shows the EpiData Analysis interface with all components open:



The screen shot shows the various windows:

On the right:

- With **F2** (toggle key) you access various Commands
- With **F3** (toggle key) you see information on field names, types, and labels
- With **F7** (toggle key) you display the History of commands in the current session (which you can paste into the Editor which you access with **F5** (center)).

In the center:

- With **F5** you access the Editor in which we will be writing our program (script) which can be run as a batch and will produce output in the Results window (the Viewer)

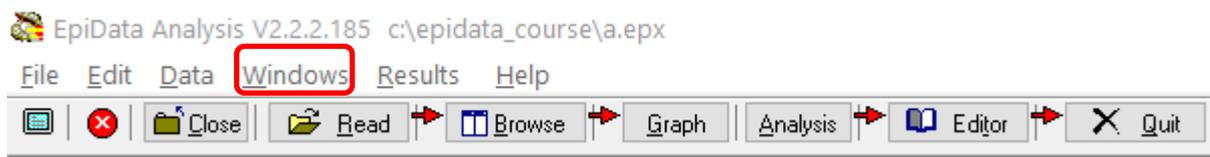
On the left:

- The Viewer or the Results window, here after running the program shown in the Editor. On the top you see in black the commands interspersed with green information on what is being done / has been done and finally the requested output table of a frequency for the variable `sex`.

At the bottom:

- Accessible with mouse-click or **F4** is the Command line into which you can type any command (we could have written `FREQ sex` here instead of into the Editor, provided that the file was open). Below the Command line, from left to right, you see the current folder including the path to it, a green bullet indicating readiness, the number of records (75) and that all of them from the dataset `a.epx` have been selected. That the open data file is the `a.epx` can be seen at the very top of the screen.

If we look at more details, we find on top of the screen:

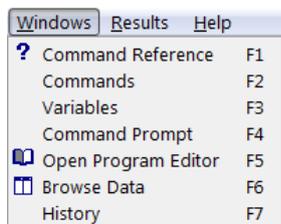


Top row: EpiData version (important to specify when reporting bugs) and path and currently open file name.

Center row: Menu bar

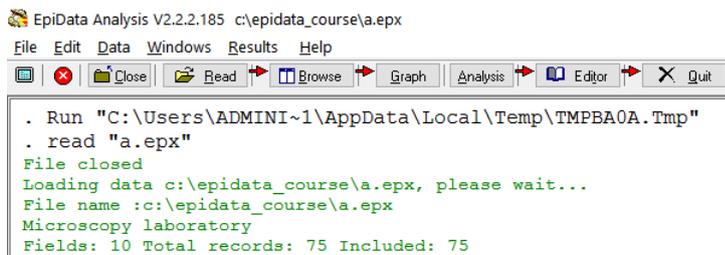
Bottom row: Work tool bar

If you select in the Menu bar Windows you get a drop-down menu:



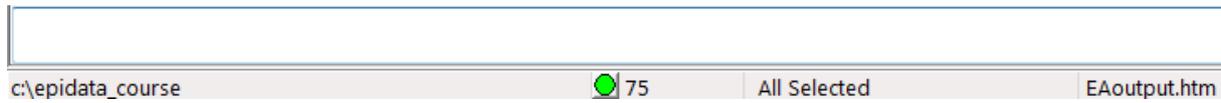
This gives you spelled out the names of the various command and windows and the shortcuts **F1** ,... , **F7** that can be used to access them.

If we open a file, the Viewer (Results window) gives information (written in green) on File name, File label, number of Fields and number of Records:



At the bottom of the screen (below the command line) we find the location (path) on the very left, in the center the green bullet indicating readiness, and right to it the number of records

(75 here) available, further to the right that the entire dataset is available (All selected), and at the very right that EpiData Analysis has opened an output file EAoutput .htm:



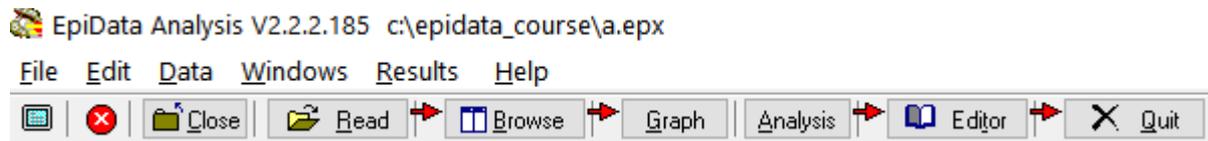
We can double-click on the path (c:\epidata_course here) if we wish to change it and get a pop-up box to Browse for the desired data Folder (see more on that below).

We can only summarize some basic essentials in the approach to the use of EpiData Analysis and we will split it into the following components:

- Using the menu interactively
- Writing a script (program) to execute a series of commands
- Analysis of continuous variables
- Analysis of categorical variables

Using the menu interactively

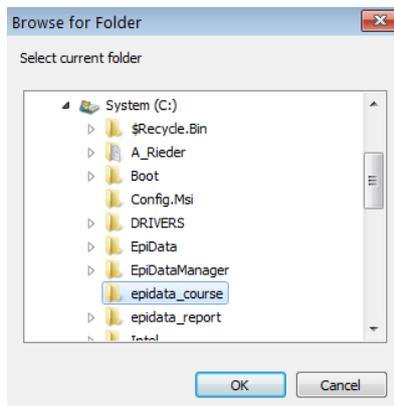
When you open EpiData Analysis, you see information on Version and currently open dataset in the top row, the Menu bar in the second and the Work tool bar in the third row:



For beginners, it is best to use the Menu and Work tool bars. Before anything else, we need to tell EpiData where our data files are located. This can be done in different ways. The simplest way is probably to double-click at the bottom left on the path that is there:

c:\epidata_course

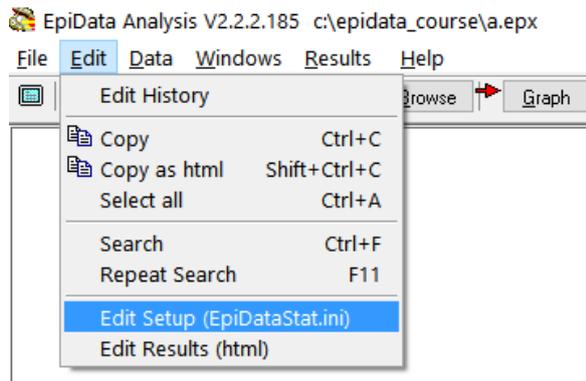
This will open an Explorer-like window:



allowing searching for the desired folder. This folder selection will persist during the current session (unless you change it during the session), but once you exit and re-enter EpiData Analysis, the path will again be the default path. Therefore, it is desirable to get a bit more control over the default path and folder that is opened when you start EpiData Analysis. Commonly, one works in project folders like we do here where the:

..\epidata_course

is our project folder. On the top you have Edit with a pop-up menu:



If you open Edit Setup (EpiDataStat.ini), you get the file opened in the Editor allowing us to customizing it to our needs:

```
C:\EpiData\epidatastat.ini - EpiData Analysis
File Edit Insert History Run Window Help
[Icons]
1 set echo=off
2 cls
3
4 set browser font size=12
5 set graph font size =12
6 set viewer font size =12
7 set window font size =12
8 set editor font size =12
9 set viewer font name ="Verdana,Courier"
10 set style sheet="c:\epidata\epiout_monospace.css"
11 set output folder="C:\EpiData\TestData"
12 set language=english
13
14 set display variables=ON
15 set display databrowser=OFF
16 set output open=ON
17
18 * Working folders
19 cd c:\temp
20 cd c:\epidata_course
21
22 set echo=ON
```

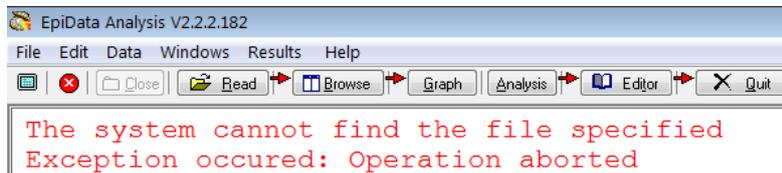
In lines 4 to 16 above we set font sizes, font types and other things (that can be changed according to your needs). It is lines 18 to 20 that are of interest here for the data folder. We start in line 19 with:

[drive name]:\folder in the PC root\subfolder\sub-sub-folder

You note specifically in line 20 `cd c:\epidata_course`. It is this line which directs EpiData Analysis to open at start-up in that folder. If we had not customized the starting folder in this `EpiDataStat.ini` file, EpiData Analysis would start in its program file folder.

In both lines 19 and 20 above you find the command “`cd`”. “`CD`” denotes “Change Directory” (a term formerly used to denote for what Microsoft denotes now as “folder”). In line 19, EpiData is instructed to go to a folder `temp` (and will do so if it exists, but stops if it doesn’t). In line 20, the previous instruction is overridden and it will end up (the last instruction sets the

final destination) in the `epidata_course` folder. We can thus list our different working folders and just rearrange them so that the currently desired is the last one. Note though that each of the listed folders must exist, else you get an error message at the start of EpiData Analysis and it will stop at the last “legal” folder found in the list of folders you may have:



Opening an EpiData data set

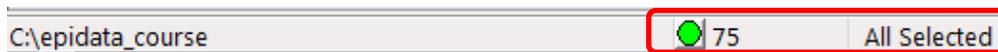
The command to open an EpiData EPX or REC file is Read which can be accessed by clicking on the respective icon in the menu (or with the short-cut **ALT+R**). Make sure (the first time) that you are in the right folder “`c:\epidata_course`” to choose the file:

a.epx

from the `[C:] \epidata_course` folder. In the Viewer window you should then get:

```
. read
File closed
Loading data C:\epidata_course\a.epx, please wait...
File name :C:\epidata_course\a.epx
Microscopy laboratory
Fields: 10 Total records: 75 Included: 75
```

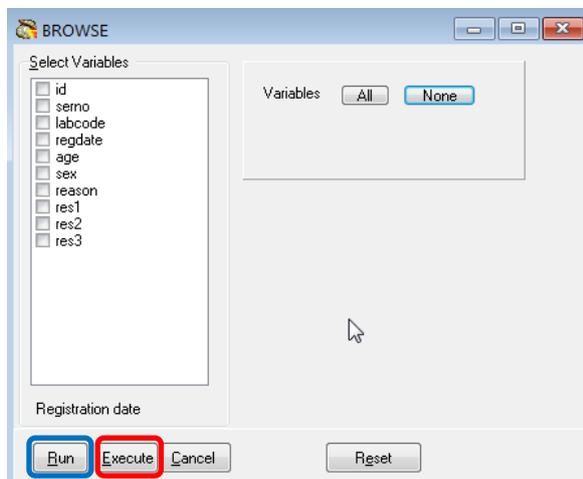
Below the command line you note:



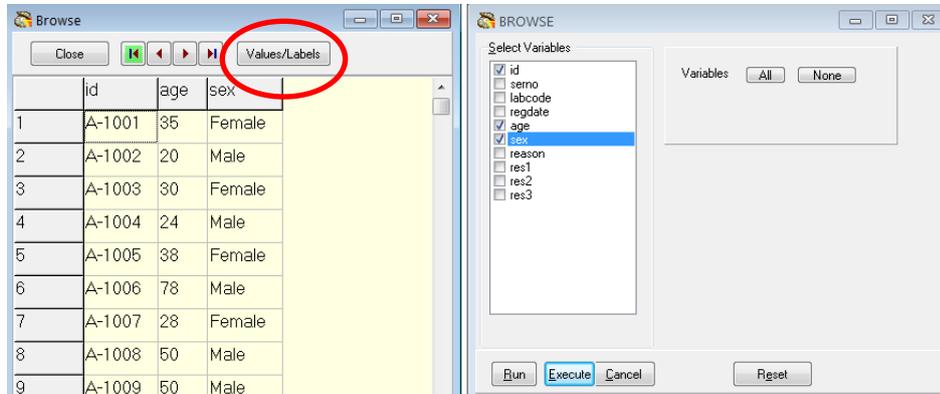
that all of the 75 records are selected. You will see later that any selections you make are listed here and the remaining number of records is shown.

Browsing the EpiData data set

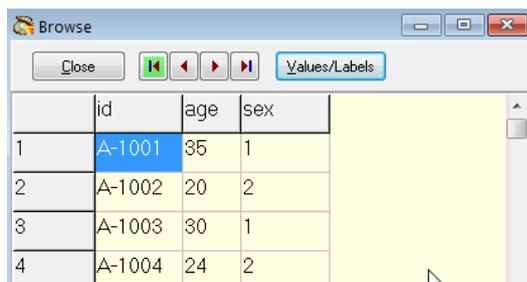
Clicking on Browse (or short-cut **ALT+B**) gives a menu box:



where you can choose All or ticking which ones you need specifically. Note at the bottom the possibilities Run and Execute (circled above in blue and red respectively). The difference is that Run shows the results and simultaneously closes this selection box. If you change your mind, you have to reopen it and start again from scratch. In contrast, with Execute this window with all selections stays open and you can modify easily if you dislike what you got:

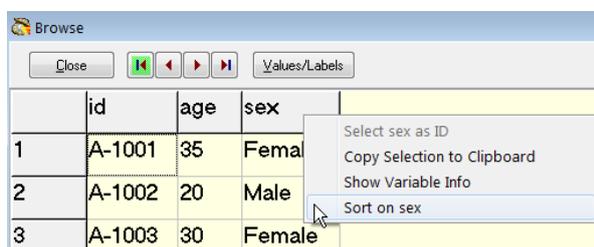


Note the “switch” (toggle key) box Values/Labels. If you click on it you get:



If the toggle key is on the Values, the information in the older EpiData file format comes from the *.REC file, if on the Labels, it comes from the *.CHK file, in the *.EPX file format, there is only one file. Thus, if the *.CHK was not in the folder, the toggle key would not work and only the Values would be seen. You can thus BROWSE either the Field values or the Field labels.

If you right-click on the field name (e.g., *sex*):



You can Copy the column (you can also copy one or more cells or rows) to Clipboard, or you can Sort the data set on the chosen field.

If you go to the command line (**F4**), and use the up-arrow key, you get the last command:

```
BROWSE id age sex
```

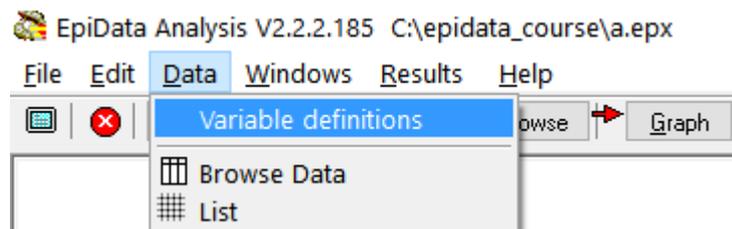
Now you know that the actual syntax to browse is the command `BROWSE` followed by the variable(s). Note that as in EpiData Entry, the capitalizing is just the way that EpiData shows the commands, it is not something you are required to use.

The use of the interactive approach with clicking can thus be used to see the actual commands that are behind the clicking which will come in handy when we start writing a program. All what we do is also written into the history which can be accessed from Windows or with the short-cut key **F7**, where the last lines read:

```
Read
Read "C:\epidata_course\a.epx"
BROWSE id serno labcode regdate age sex reason res1 res2 res3
BROWSE id age sex
```

Again, to make life easier for the beginner, the history can be invoked and pasted into the Editor, and then edited for the program.

Browsing allows seeing variables and their values for each record, but one would also like to know all possible values and labels for a given variable. In the Menu bar, choose Variable definitions from the drop-down menu of Data:



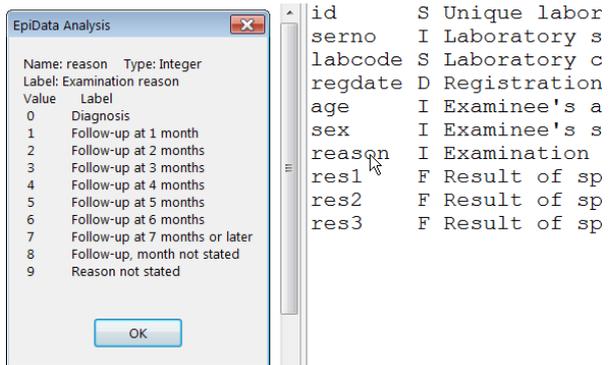
and in the Viewer you get (beginning only shown):

```
. var
File: C:\epidata_course\a.epx
Microscopy Laboratory
```

Name	Type	Length	Decimals	Label	Value label	Missing
id	String	6	0	unique laboratory identifier		
serno	Integer	4	0	laboratory serial number		
labcode	String	1	0	laboratory code	A = Laboratory A B = Laboratory B C = Laboratory C D = Laboratory D	
regdate	Date DMY	10	0	registration date		
age	Integer	2	0	examinee's age in years		
sex	Integer	1	0	examinee's sex	1 = Female 2 = Male 9 = Sex not recorded	

Note at the very beginning `var` which is the command equivalent to the selection from the Menu and which you can type into the command line to clicking on the menu.

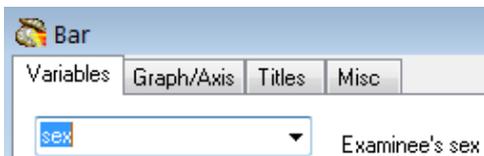
If you are only interested in one specific variable, for instant the variable `reason`, you can right-click on it in the variables window:



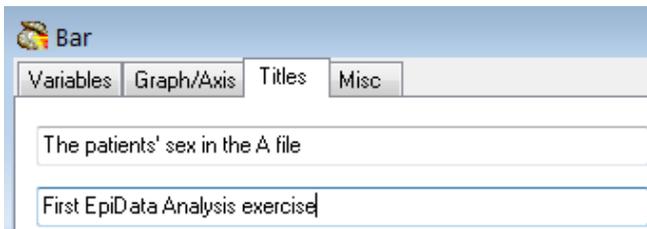
and you get to see what the label block looked like during data entry.

Making a graph

To make for instance a bar graph (or any other), open the graph dialogue by clicking on Graph (**ALT+G**) and pick for instance the field `sex`:

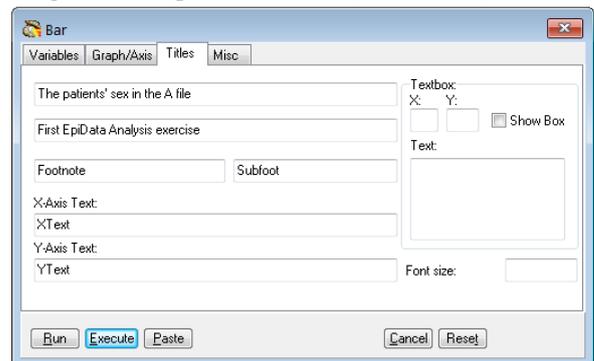
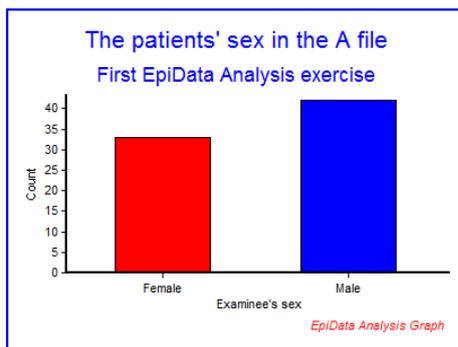


and modify the lines for Title and Sub-Title if you wish:



then click Execute get:

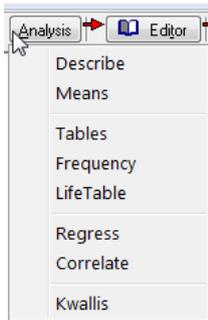
```
. BAR sex /Ti="The patients' sex in the A file" /Sub="First EpiData Analysis exercise" /SizeX=400 /SizeY=300
```



Note that you not only get the graph you asked for but above it in this instance also the syntax required to get it which you can copy and paste into your program.

Interactive analysis

Clicking on Analysis (**ALT+A**) give a drop-down menu:



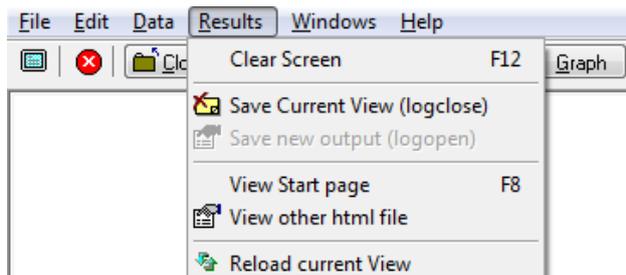
Let's use the first one, Describe to describe the content of a numeric field such as age and get:

```
. DESCRIBE age
```

Variable	N=75	Sum	Mean	(95% cfi)	Min	p5	p10	p25	Median	p75	p90	p95	Max
age	75	2997.00	39.96	35.35 44.57	14.00	16.60	19.00	24.00	35.00	50.00	72.20	81.00	99.00

We will come back later to the error of including here examinees with an age coded as missing (99) and how to fix that.

In the Results menu, you can clear the Viewer output (and memory) with Clear screen (short cut function key **F12**):



Writing a script (program) to execute a series of commands

This is all good and fine. It is what might be called “interactive analysis”. The advantage is that you can very quickly “fish” in your dataset and familiarize yourself with it. The disadvantage of interactive analysis is that everything is lost once you leave EpiData Analysis. It defies thus the underwritten purpose that everything you do in research must be thoroughly documented.

The solution of EpiData Analysis to documentation is to use:



(Function key **F5** also opens the Editor)

It opens a new window which is simply a text editor for EpiData Analysis programs that take the extension *.PGM. There are many different ways to do it, but things become more quickly automated if done always in the same way. We propose that the first few command lines have the following content:

```
1 * Part B, Exercise 1
2 * Written by: Hans L Rieder
3 * First version: 23 Apr 2013
4 * Last revision: 21 Jul 2015
5
6 cls
7 close
8 logclose
```

Save the program immediately as “b_ex01.pgm” (the extension is supplied automatically) in our working folder and verify at the top that this is so:

```
C:\epidata_course\b_ex01.pgm - EpiData Analysis
File Edit Insert History Run Window Help
```

It is always a good idea to make comments on what you do. Comments in EpiData Analysis are written by putting an asterisk * as the first thing on a line like in:

```
* The title of my program
```

If you want a command to be executed but would like to write a comment, you may do this with a double forward slash (“//”) after the command:

```
cls // This will clear the screen and free memory
```

It is also a good idea to write the date of the first version, lesser so of subsequent versions, because the original date will disappear from the file information once you update and save while one is often interested when it was started. It is also helpful to state who wrote the first draft, and add names if it is a collaborative work. We cannot overemphasize the importance of thorough annotations.

In lines 6 to 8 we start with a clean slate:

```
cls // CLS clears the screen (identical to the interactive F12)
close // CLOSE closes any open data (*.REC) file
logclose // LOGCLOSE closes any open log file
```

While `cls` and `close` are usually easily understood, the concept of “`logclose`” is perhaps intuitively not equally clear. When we opened EpiData Analysis and read the “a.epx”, EpiData Analysis gave the following information below the command line:

```
c:\epidata_course 75 All Selected EAoutput.htm
```

On the very right, the `EAoutput.htm` is the log file that EpiData Analysis opens and in which it records our doings and at which we can look (after quitting EpiData Analysis). We recognize some of the things we just did if we open it in our browser (the extension `.htm` identifies it as a file to be examined in a browser):

```
. read
Loading data C:\epidata_course\a.epx, please wait...
File name :C:\epidata_course\a.epx
Microscopy laboratory
Fields: 10 Total records: 75 Included: 75
. BROWSE id
. BROWSE id age sex
```

With our command “logclose”, this file is closed (and its name disappears from the line below the command line) and we are ready to write another log file (into which we perhaps wish to write specifically a table output or record anything else). We close it because like with data files only one log file can be open at a time. We could of course close it just before opening another one, but we propose to start with a clean slate where everything is closed, the screen is empty and the memory free.

We propose the above to be a standard approach to start any program: identifying, labeling and dating, followed by making a clean slate.

As we can work (and have open) only one data file at a time and we have closed any other that might be open with the above command, we can now open the file to analyze, with the command:

```
read "a.epx"
```

This does the same thing as we did above interactively when pressing the button Read: it opens an EpiData data file. The next three lines:

```
append /file="b.epx"
append /file="c.epx"
append /file="d.epx"
```

do something that is often required: we have several files with the same data structure and we wish to combine them vertically (“append”) to have a single data set for analysis. In this case, the four *.EPX files a, b, c, and d contain each 75 records from four different tuberculosis sputum smear microscopy laboratories (these are real data from Yangon, data courtesy Dr Ti Ti). Finally, we save the data in a new file, and thus have now:

```
read "a.epx"
append /file="b.epx"
append /file="c.epx"
append /file="d.epx"
savedata "abcd.rec" /replace
```

Important Note: The current version of EpiData Analysis (Build 2.2.2.183) can read *.EPX files, but it cannot yet save files in this format: any file you save will have the EpiData Entry 3.1 format with a *.REC file extension. Simultaneously, a paired *.CHK file containing the metadata will be saved.

Without the /replace, this would run once, but give an error the second time:

```
DataFile and/or Chk file exists.
Add /REPLACE or erase
```

```
c:\epidata_course\abcd.rec
c:\epidata_course\abcd.chk
Exception occurred: Operation aborted
Failed to save data to c:\epidata_course\abcd.rec
```

Creating a new *.REC file (as here) with “savedata” also creates an accompanying *.CHK file as we see from the error message.

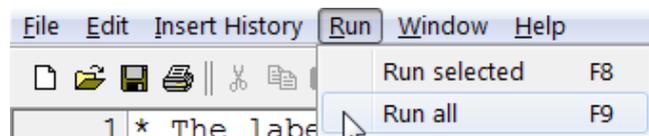
This requirement is a precaution to prevent the inadvertent overwriting of an existing file. The option /replace gives the explicit permission to overwrite the file in case it exists.

Finally, we close the file (a.epx) which is still open (the file that was read last is always the file that is open) and we need to close it first before we can open the newly created abcd.rec:

```
close
read "abcd.rec"
```

Now we are ready to analyze the data.

First run what we have. At the top of the Editor you see Run with two options:



F9 runs the entire program and **F8** whatever command line(s) you selected. To run a single command line, there is no need to mark it, just place the cursor anywhere onto it and press **F8**.

Analysis of continuous variables

We have one continuous variable in this data set, age in years (at last birthday). What you cannot know without a data documentation sheet is that examinees with unknown age were given a value 99 for age.

It is of course possible to make a frequency of age but this could give us up to 100 different values for a 2-digit field like age. Frequencies are better used for categorical variables while for continuous variables we use other methods to describe them. As is a common standard in any analysis package, we give a command followed by the name of the variable(s) on which the command should be executed. Type (on two lines):

```
describe age
means age
```

We get:

```
. describe age
```

Variable	N=300	Sum	Mean	(95%	cfi)	Min	p5	p10	p25	Median	p75	p90	p95	Max
age	300	11656.0	38.85	36.82	40.88	5.00	17.00	19.00	26.25	35.00	49.00	65.00	74.95	99.00

. means age

Examinee's age in years								
Obs.	Sum	Mean	Variance	Std Dev	(95% CI mean)		Std Err	
300	11656.0	38.85	319.26	17.87	36.82	40.88	1.03	
Minimum	p5	p10	p25	Median	p75	p90	p95	Max
5.00	17.00	19.00	26.25	35.00	49.00	65.00	74.95	99.00

The first output is from “describe”, the second from “means”. Comparison shows that means is more informative than describe. We also note that the maximum is 99.00, meaning that at least one person had a unknown age coded as 99. Thus, our calculations are incorrect: we need to exclude the unknowns before calculating this kind of statistics:

```
select age<>99
```

The command to select a subset is `select` followed by the variable `var01` in question, followed by an operator and the value. You may need these operators:

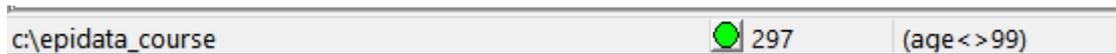
```
= Equal to
< Less than
> Greater than
>= Greater or equal to
<= Less or equal to
<> Unequal
```

There are more operators which you may look up in the Help file (F1).

Repeat `means age` after the appropriate selection and get:

Examinee's age in years								
Obs.	Sum	Mean	Variance	Std Dev	(95% CI mean)		Std Err	
297	11359.0	38.25	285.46	16.90	36.32	40.18	0.98	
Minimum	p5	p10	p25	Median	p75	p90	p95	Max
5.00	17.00	19.00	26.00	35.00	48.00	65.00	74.00	86.00

Note at the bottom of the EpiData Analysis screen:



Apparently, our selection led to the exclusion of 3 examinees from the set of 300. Often we wish to compare a continuous variable in subgroups, for instance sex. The command is:

```
means age /by=sex
```

which gives us:

Examinee's age in years									
sex	Obs.	Sum	Mean	Variance	Std Dev	(95% CI mean)		Std Err	
Female	107	4065.00	37.99	301.93	17.38	34.66	41.32	1.68	
Male	190	7294.0	38.39	277.68	16.66	36.00	40.77	1.21	
sex	Minimum	p5	p10	p25	Median	p75	p90	p95	Max
Female	13.00	16.00	18.00	26.00	34.00	50.00	61.40	74.60	86.00
Male	5.00	17.00	19.10	27.00	35.00	45.25	65.00	73.45	86.00

Please note that the stratifying variable (`sex`, in this case) has to be a numeric variable for this command to work, another reason why we prefer numeric variables with value labels to text variables. The output shows that there is a small difference in the mean age between females and males. To know if this difference was just by chance or is statistically significant,

we may wish to do an unpaired t test. We then simply add the option /T (capitalization not required):

```
means age /by=sex /T
```

which gives us (only the lower part shown here):

Source	SS	df	MS	F	p Value
Between	10.89	1	10.89	0.04	0.846
Within	84486.17	295	286.39		
Total	84497.06	296	285.46		

Bartlett's test for homogeneity of variance
Chi²= 0.239 df(1) p= 0.625

The p value of 0.846 (being more than 0.05, the level used conventionally to assess statistical significance) here thus tells that the difference between mean ages of males and females is not statistically significant. Please note that this also gives the results of Bartlett's test of homogeneity of variance at the bottom of the output, one of the assumptions to be satisfied before using the t test. If the p value of Bartlett's test is <0.05, then the assumption of homogeneity of variance is violated and it suggests to use one of the non-parametric tests for assessing statistical significance. (Read about a non-parametric test called Kruskal-Wallis test and the command `kwallis` in the help file)

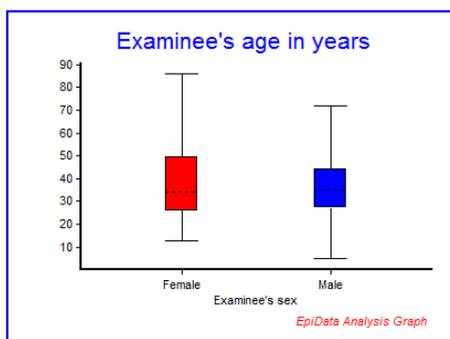
Note:

- If there are paired observations for the same individuals, then the paired t test is recommended. To calculate this in EpiData, calculate the difference in paired values in a new variable (let us say "diff") and run the command line `means diff /t`.
- If there are more than two means to be compared, then we use ANOVA (Analysis of Variance), though the EpiData command to achieve it will be the same `/t`.

In addition to the numeric output, one might like to see it graphically represented with a boxplot:

```
boxplot age /by=sex
```

Box Plot								
	N	Min	P ₁₀	P ₂₅	Median	P ₇₅	P ₉₀	Max
Examinee's age in years								
Female	107	13	18	26	34	50	61	86
Male	190	5	19	27	35	45	65	86



The command to de-select is simply `SELECT`. You thus select (one or more selections), execute a program and then `deselect` to return to the full data set:

```
select age<>99
boxplot age /by=sex
```

```
select
```

Sometimes, we wish to categorize a continuous variable. For age, a common categorization is “WHO age groups”. This is how we would write the commands:

```
cls
select
define agegrp #
if age>=00 and age<15 then agegrp=1
if age>=15 and age<25 then agegrp=2
if age>=25 and age<35 then agegrp=3
if age>=35 and age<45 then agegrp=4
if age>=45 and age<55 then agegrp=5
if age>=55 and age<65 then agegrp=6
if age>=65 and age<99 then agegrp=7
if age =99 then agegrp=9
label agegrp "WHO standard age groups"
labelvalue agegrp /1="0- to 14-yr-old"
labelvalue agegrp /2="15- to 24-yr-old"
labelvalue agegrp /3="25- to 34-yr-old"
labelvalue agegrp /4="35- to 44-yr-old"
labelvalue agegrp /5="45- to 54-yr-old"
labelvalue agegrp /6="55- to 64-yr-old"
labelvalue agegrp /7="65-yr-old and older"
labelvalue agegrp /9="Unknown age"
```

After ensuring that we have the full data set (“SELECT”), we define a new integer variable `agegrp` and categorize the patients’ age into the seven WHO age groups plus one for those with no age recorded. After categorization, we give a Field label and then Value labels. Looking at how to use this new variable leads us over to the analysis of categorical variables.

Because “CLS” does more than just clearing the screen – it also frees memory – we tend to write frequently a `cls` into the program. You can also manually, while a program runs, press **F12**, and it will do the same thing, speeding up the running of the program.

We could have defined a text variable:

```
define agegrptxt1 _____
define agegrptxt2 <AAAAAAAAAA>
```

Analysis of categorical variables

The most frequent command we use for a categorical variable is, well, frequencies:

```
cls
freq agegrp
```

WHO standard age groups	N
0- to 14-yr-old	6
15- to 24-yr-old	57
25- to 34-yr-old	84
35- to 44-yr-old	67
45- to 54-yr-old	28
55- to 64-yr-old	25
65-yr-old and older	30
Unknown age	3
Total	300

The output displays thus the Value labels we wrote, not the values, but we can have both with the option /vl:

```
freq agegrp /vl
```

WHO standard age groups	
	N
1 0- to 14-yr-old	6
2 15- to 24-yr-old	57
3 25- to 34-yr-old	84
4 35- to 44-yr-old	67
5 45- to 54-yr-old	28
6 55- to 64-yr-old	25
7 65-yr-old and older	30
9 Unknown age	3
Total	300

This is important to know if we want to make a selection like for instance including those in the category “Unknown age” by writing “select agegrp<>9”. With frequencies we often want to know the percentage distribution:

```
freq agegrp /c
```

WHO standard age groups		
	N	%
0- to 14-yr-old	6	2.0
15- to 24-yr-old	57	19.0
25- to 34-yr-old	84	28.0
35- to 44-yr-old	67	22.3
45- to 54-yr-old	28	9.3
55- to 64-yr-old	25	8.3
65-yr-old and older	30	10.0
Unknown age	3	1.0
Total	300	100.0

The option “/c” is thus for column percent, while an option “/r” would be for row percent (not applicable here but in tables). With frequencies, one can also get 95% confidence intervals (not very sensible here, just shown for the principle):

```
freq agegrp /c /ci
```

WHO standard age groups			
	N	%	(95% CI)
0- to 14-yr-old	6	2.0	(0.9-4.3)
15- to 24-yr-old	57	19.0	(15.0-23.8)
25- to 34-yr-old	84	28.0	(23.2-33.3)
35- to 44-yr-old	67	22.3	(18.0-27.4)
45- to 54-yr-old	28	9.3	(6.5-13.2)
55- to 64-yr-old	25	8.3	(5.7-12.0)
65-yr-old and older	30	10.0	(7.1-13.9)
Unknown age	3	1.0	(0.3-2.9)
Total	300	100.0	

If you use more than one variable:

```
freq sex reason
```

you get frequencies for each of the (as many as you like) variables:

**Examinee's
sex**

	N
Female	109
Male	191
Total	300

Examination reason

	N
Diagnosis	134
Follow-up at 2 months	40
Follow-up at 3 months	12
Follow-up at 4 months	3
Follow-up at 5 months	35
Follow-up at 6 months	31
Follow-up at 7 months or later	16
Follow-up, month not stated	27
Reason not stated	2
Total	300

Let's make one more new variable for a case definition. The dataset is, as mentioned above, from laboratory registers. Each examinee can have up to three sequential examinations. Let's define that an examinee is a "case" if any of the three possible examinations has at least 1 acid-fast bacillus (a proxy for tubercle bacilli) in it. If we make a frequency of the first result and also ask for the display of the values (not just the value labels):

```
freq res1 /vl
```

Result of specimen 1

	N
0.4 Scanty, 4 AFB per 100 fields	1
0.0 Negative	272
1.0 1+ positive	18
2.0 2+ positive	5
3.0 3+ positive	4
Total	300

We get something, but not quite what we need. What we need is to know what values were possible, not just which values were actually present. We can obtain that information by typing `var` (as explained above) into the command line and we get the coding for the entire dataset. Alternatively, for just one categorical variable, we can right-click the variable in the variable (**F3**, if not open) window (see above).

We can repeat this for each of the three result variables and we see that all use the same label block. Based on this information, we can now safely define a case:

```
define case #
    let case=0
if res1>0 and res1<9 then case=1
if res2>0 and res2<9 then case=1
if res3>0 and res3<9 then case=1
label case "Smear microscopy defined case"
labelvalue case /0="Non-case"
labelvalue case /1="Case"
```

and try:

```
tables case sex /r /c /tp /pct
```

Smear microscopy defined case									
Examinee's sex	Non-case	%	%	% Case	%	%	% Total	%	%
Female	94 (86.2)	{35.9}	[31.3]	15 (13.8)	{39.5}	[5.0]	109 (100.0)	{36.3}	[36.3]
Male	168 (88.0)	{64.1}	[56.0]	23 (12.0)	{60.5}	[7.7]	191 (100.0)	{63.7}	[63.7]
Total	262 (87.3)	{100.0}	[87.3]	38 (12.7)	{100.0}	[12.7]	300		

Percents: (Row) {Col} [Total]

This gives us row (/r), column (/c), and total (/tp) percentages. The /pct aligns them properly in separate columns.

Note that in the command, whichever variable you write first after the command tables is the column variable (outcome variable in epidemiologic terms) and the second variable is the row variable (exposure variable).

Meanwhile you have noted that EpiData Analysis gives just the raw numbers as default output in a frequency or table, but you can add multiple options, and combine them as needed. Options are preceded by the forward slash (/).

tables case sex /r /t

Smear microscopy defined case				
Examinee's sex	Non-case	% Case	% Total	%
Female	94 (86.2)	15 (13.8)	109 (100.0)	
Male	168 (88.0)	23 (12.0)	191 (100.0)	
Total	262 (87.3)	38 (12.7)	300	

Percents: (Row)

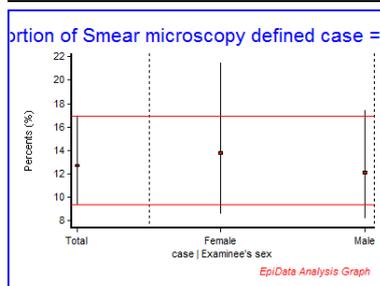
Chi²= 0.185 df(1) p= 0.6667

shows us with the option “/t” (for performing the Chi-square test) that there is no significant difference between males and females in being a case. In some instances, when the expected cell value of a 2-by-2 table is less than 5, it is recommended to use Fischer’s exact test instead of the Chi-square test. For Fischer’s exact test, use the option “/ex”. This is indicated by EpiData automatically. We could make 95% confidence intervals with frequencies, but we cannot use the “/ci” option for tables. We need a little trick to get that, and that is with the confidence interval plot:

ciplot case sex

Crude: Proportion of Smear microscopy defined case = Case among all.					
variable	stratum	Total N	n _{Smear microscopy defined case}	Case %	(95% CI)
Smear microscopy defined case	Total	300	38	12.7	(9.4-16.9)
Examinee's sex	Female	109	15	13.8	(8.5-21.5)
	Male	191	23	12.0	(8.2-17.4)

Crude: Proportion of Smear microscopy defined case = Case among all.



And if we don’t want to see the graph, only the table we use the option “/ng” (“no graph”)

ciplot case sex /ng

By default, the higher value of the variable is considered as outcome. In case you want to change that, use the option “/O=value” (O for outcome).

Tasks:

- o Determine the year of birth (new variable created from age and date of registration), then make groups of examinees (another variable) born respectively before 1930, from 1930 to including 1949, 1950 and later, and those without known year of birth.*
- o Use two approaches, one with text field coding and the other with numeric coding and value labels.*

Solution to Exercise 1: An introduction to EpiData Analysis

Key Point(s):

- It is important to make a comment first in the program. This preferably is what you want to do in that program. Comments are preceded by an asterisk and are bypassed by the analysis program.
- The F9 key runs the whole program whilst the F8 key runs only the selected part of the program.

Tasks:

- o Determine the year of birth (new variable created from age and date of registration), then make groups of examinees (another variable) born respectively before 1930, from 1930 to including 1949, 1950 and later, and those without known year of birth.
- o Use two approaches, one with text field coding and the other with numeric coding and value labels.

Solution:

The output solution is as follows:

```
. tables sex birthgrp1
```

Examinee's sex			
Exercise birth years	Female	Male	Total
1929 and before	6	9	15
1930-1949	14	25	39
1950 and later	83	152	235
Unknown	6	5	11
Total	109	191	300

```
. tables sex birthgrp2
```

Examinee's sex			
Exercise birth years	Female	Male	Total
Born before 1930	6	9	15
Born 1930 to 1949	14	25	39
Born after 1950	83	152	235
Unknown birth year	6	5	11
Total	109	191	300

The "Task" part of program B_EX01.PGM might look as follows:

```
*****
* Part B, Exercise 1
* Task
*****
```

```
cls
close
logclose
```

```

read "a.epx"
append /file="b.epx"
append /file="c.epx"
append /file="d.epx"
savedata "abcd.rec" /replace

close
read "abcd.rec"

* Define the year of birth
define birthyr ####
birthyr=year(regdate)-age
if age=99 or year(regdate)=1900 then birthyr=9999

* Using text variables
* define groupings for birth years
define birthgrp1 _____
                                let birthgrp1="other"
if birthyr <1930 then birthgrp1="1929 and before"
if birthyr>1929 and birthyr<1950 then birthgrp1="1930-1949"
if birthyr>1949 then birthgrp1="1950 and later"
if birthyr=9999 then birthgrp1="Unknown"
label birthgrp1 "Exercise birth years text coding"

* Using numeric variables
* define groupings for birth years
define birthgrp2 #
                                let birthgrp2=8
if birthyr <1930 then birthgrp2=1
if birthyr>1929 and birthyr<1950 then birthgrp2=2
if birthyr>1949 then birthgrp2=3
if birthyr=9999 then birthgrp2=9
label birthgrp2 "Exercise birth years numeric coding"
labelvalue birthgrp2 /1="Born before 1930"
labelvalue birthgrp2 /2="Born 1930 to 1949"
labelvalue birthgrp2 /3="Born after 1950"
labelvalue birthgrp2 /8="Unaccounted for"
labelvalue birthgrp2 /9="Unknown birth year"

cls
tables sex birthgrp1
tables sex birthgrp2

```

Exercise 2: More on EpiData Analysis

At the end of this exercise you should be able to:

- Know more about data set and variable manipulation
- Know more about tables
- Know more about graphs

More about data set manipulation

Merge and append

In the first exercise of Part B you learned to `append` files. It was said then that you append files if the files have the same structure. That is not exactly necessary. You can append files with different structures as long as you ensure 1) that the file that is read contains all the same variables as the file(s) you plan to append to it, and 2) that the variables have the same definitions. Let's show an example of two data sets, Set A and Set B, each with three records, where `id` is the unique identifier in each set:

Set A

	id	sex	age
1	a	m	23
2	b	f	30
3	c	m	50

Set B

	id	bs	ht
1	d	5.6	186
2	e	6.2	170
3	f	4.8	168

Note, that the identifiers in Set A are all different from the identifiers in Set B, the information in the two sets belongs thus to a total of six different individuals. We can append the two files if we first create the same variables in the file we read (let's say Set A) before we append Set B:

```
cls
close
read "b_ex02_01.rec"
define bs #.#
define ht ###
append /file="b_ex02_02.rec"
```

We get correctly a file with six records:

	id	sex	age	bs	ht
1	a	m	23	.	.
2	b	f	30	.	.
3	c	m	50	.	.
4	d	.	.	5.6	186
5	e	.	.	6.2	170
6	f	.	.	4.8	168

We could have gone the way with `merge`. `merge` requires the identifier to be unique within a set which is the case for `id` within Set A and within Set B. If we use `merge`, there is no need to first ensure that all variables exist in the set that is read:

```
cls
close
read "b_ex02_01.rec"
merge id /file="b_ex02_02.rec"
```

We get:

	id	sex	age	bs	ht	MergeVar
1	a	m	23	.	.	Only in memory (Original)
2	b	f	30	.	.	Only in memory (Original)
3	c	m	50	.	.	Only in memory (Original)
4	d	.	.	5.6	186	Only in external file
5	e	.	.	6.2	170	Only in external file
6	f	.	.	4.8	168	Only in external file

In other words, we get exactly the same thing, except that EpiData Analysis added a variable `MergeVar` which can take 3 values:

```
Only in memory (Original)
Only in external file
In both [not existing here]
```

If the two data sets thus contain different individuals, we can choose either `append` or `merge` as long as we ensure that (if we choose `append`) the read file has all the variables that the appended file has. If one or more individuals are identical in Set A and set B, then using `append` becomes wrong because we would get two records from the same individual. In such a case, we must use `merge`. To exemplify this, we saved the above data Set B as `b_ex02_03.rec`, after changing the value for `id=d` to `id=c`. In other words, person “c” is in both data sets. If we juxtapose the results from appending and merging:

Appending						Merging						
	id	sex	age	bs	ht		id	sex	age	bs	ht	MergeVar
1	a	m	23	.	.	1	a	m	23	.	.	Only in memory (Original)
2	b	f	30	.	.	2	b	f	30	.	.	Only in memory (Original)
3	c	m	50	.	.	3	c	m	50	5.6	186	In both
4	c	.	.	5.6	186	4	e	.	.	6.2	170	Only in external file
5	e	.	.	6.2	170	5	f	.	.	4.8	168	Only in external file
6	f	.	.	4.8	168							

The file resulting from appending has 6 records, two of which belong to individual “c”, the result is thus wrong. Conversely, the file resulting from merging has only 5 records corresponding to the total of 5 individuals and is thus correct. EpiData Analysis gives us with the automatically created field `MergeVar` also the relevant information about the source of the data.

In a later exercise you will learn about important options (look-up table) that you may need when merging data files.

Select

We used `select` as follows:

```
read "abcd.rec"
select age<>99
means age /by=sex
select
freq sex
```

We selected a subset, then carried out the command, and finally used `select` again to obtain back the full data set. Thus, before we wrote `select` again, we kept the reduced data set. Sometimes it is desirable to make a selection only for a specific command as in the above example. A powerful possibility in EpiData Analysis is to do precisely that by writing the condition for which a command is to be executed onto the same line as the command. Instead of the above, we thus write:

```
read "abcd.rec"
means age /by=sex if age<>99
freq sex
```

Drop and Keep

`Select` reduced the number of records in a data set. Sometimes it is useful to reduce the number of variables, not the number of records in a data set. This is where `drop` and `keep` come in. The two are complementary. We have identifier field `id` and `serno` in the data set `abcd.rec`. If we don't want to keep them, we would write:

```
cls
close
read "abcd.rec"
drop id serno
savedata "abcd_temp_01.rec" /replace
```

This is more efficient than the complementary approach with `keep`:

```
cls
close
read "abcd.rec"
keep labcode regdate age sex reason \
    res1 res2 res3
savedata "abcd_temp_02.rec" /replace
```

Note here also in passing that you can use the backslash `\` to continue a command on the following line (this was not possible in the `CHK` file).

Sort

Sorting a dataset on one or more variables is a key component in programming. For instance, if you would like to sort our data set `abcd.rec` on the laboratory, you would write:

```
cls
close
read "abcd.rec"
sort labcode
```

and within each laboratory by registration date:

```
sort labcode regdate
```

8	A	24/10/2003
9	A	27/10/2003

...

87	B	08/09/2003
88	B	09/09/2003

The last sorting command overrides any previous sorting command. Note that saving a data set after sorting will save it in the last sort order.

Define and gen

We used so far `define` to create a variable in the memory exactly as we did in the CHK file for temporary variables. For instance, if we create a new field `case` to define a sputum-smear positive case:

```
define case #
case=0
if res1>0 and res1<9 then case=1
if res2>0 and res2<9 then case=1
if res3>0 and res3<9 then case=1
label case "Microscopy case definition"
labelvalue case /0="Non-case"
labelvalue case /1="Case"
```

There is an alternative in EpiData Analysis, `gen` for generating a new variable. Be careful though to use “`gen`” and not “`generate`” because the latter is also a legitimate command, but with an entirely different meaning (it creates empty records). We would thus write instead of the above:

```
cls
gen i case2=0
if res1>0 and res1<9 then case2=1
if res2>0 and res2<9 then case2=1
if res3>0 and res3<9 then case2=1
label case2 "Microscopy case definition"
labelvalue case2 /0="Non-case"
labelvalue case2 /1="Case"
```

We don't have to assign it a value (as we do here with “`=0`”). If we don't, the value is set to missing with a period. With `define` we defined at the same time with the field name also both its type and length. An integer variable created with `gen` will automatically get a field length of 9.

There are other field types we can create this way:

```
gen d copydate=regdate
gen f agedays=age*365.25
gen s(2) labsex=labcode+sex
```

If you create a **float field**, it will have the length 12, including 4 decimal points. This could be inconvenient, if we create a label block for instance (like we had for the results) as we must pay attention that there is full compatibility. There is no option to change this default.

Wonderfully, we can change the default length (of 20) for **string fields**, in the non-sensical example here defined as having a length of 2.

Note that “gen” is a bit faster to code, “define” runs faster. This becomes important in large datasets. For date and string fields “gen” is virtually always preferable, for integer fields “define” is perhaps preferable if you know the expected length of the field, and for float fields, “define” is probably almost always preferable.

Recode

We had used DEFINE and / or GEN go make new variables. If we have a continuous variable from which we wish to make a categorical one, like converting the variable AGE to age groups, EpiData offers RECODE, as shown in this example:

```
define agegrp2 #
recode age to agegrp2 00-14=1 15-24=2 25-34=3 35-44=4 45-54=5 \
                    55-64=6 65-98=7 99=9
freq agegrp2
```

gives:

```
agegrp2
  N
00-14  6
15-24 57
25-34 84
35-44 67
45-54 28
55-64 25
65-98 30
.99   3
Total 300
```

Thus, the groupings become the labels. Of course, one can override these default value labels.

String fields and substrings

Let’s assume we wish to make a string field for each microscopy result, which can take on the three values “N” (for negative), “P” (for any positive), and “9” (for not available). We thus write:

```
cls
close
read "abcd.rec"

      gen s(1) result1="P"
if res1=0 then result1="N"
if res1=9 then result1="9"
cls
      gen s(1) result2="P"
if res2=0 then result2="N"
if res2=9 then result2="9"
cls
      gen s(1) result3="P"
if res3=0 then result3="N"
if res3=9 then result3="9"
```

We then wish to combine the three results into one single string to obtain the pattern:

```
cls
gen s(3) pattern=result1+result2+result3
label pattern "Pattern of 3 serial smears"
freq pattern
```

and get:

Pattern of 3 serial smears	
	N
NN9	157
NNN	105
NP9	1
NPP	9
PP9	11
PPN	2
PPP	15
Total	300

Finally, we can extract a subset of a string field:

```
cls
gen s(1) firstres=substr(pattern,1,1)
label firstres "Result of 1st smear"
freq firstres /c /ci
```

and get:

Result of 1st smear			
	N	%	(95% CI)
N	272	90.7	(86.8-93.5)
P	28	9.3	(6.5-13.2)
Total	300	100.0	

Admittedly, not the most efficient approach to something that could have been obtained directly from the original field `res1`, but the point here is to show the way how to extract a substring from a text field:

SUBSTR (fieldname, start position, number of characters including start position)

Thus:

```
cls
gen s(2) res12=substr(pattern,1,2)
gen s(2) res23=substr(pattern,2,2)
freq res12 res23
```

gives:

res12		res23	
	N		N
NN	262	N9	157
NP	10	NN	105
PP	28	P9	12
Total	300	PN	2
		PP	24
		Total	300

Date fields

We know that date fields are a hassle because they don't fit our decimal system and people use different ways to write dates. To further complicate matters, EpiData Analysis deals

somewhat differently with date definitions than we learned in the CHK file. There are different ways of doing it, but it might be best to learn one and learn to master it. Let's assume, we wish to calculate the number of years elapsed since the registration of the examinee in our data set and 1 January 2013. The syntax to accomplish this is:

```
cls
close
read "abcd.rec"
gen f intyrs=(dmy(01,01,2013)-regdate)/365.25
```

We thus tell EpiData Analysis 1) that it is a date and 2) the format of our date with `dmy` followed by the values for the three date components in parenthesis, separated by commas.

Results variables

If we execute certain commands, EpiData Analysis will produce variables in memory that keep temporarily certain values which we can visualize with the command `result`. For example:

```
cls
close
read "abcd.rec"
```

```
means age if (age<>99) and (sex=1)
result
```

produces:

```
System Variables Value
SYSTEMDATE          27/04/2013
SYSTEMTIME          15:55:43
SYSDIR              C:\EpiData\
CURRENTDIR          C:\epidata_course
LANG                EN

Temporary Variables Value
$VARIABLE1          age
$LEVEL1
$OBS1               107
$SUM1               4065
$MEAN1              37.9906542056075
$VAR1               301.933874096279
$SD1                17.3762445337386
$CFIL1              34.6602378330551
$CFIH1              41.3210705781599
$STDERR1            1.67982496333453
$MIN1               13
$P051               16
$P101               18
$P251               26
$P501               34
$P751               50
$P901               61.4
$P951               74.6
$MAX1               86
```

Our interest here is in the Temporary Variables Value. We can use them and save the values of any of them into another variable. For instance:

```
means age if (age<>99) and (sex=1)
result
gen meanfem=$mean1
```

```
means age if (age<>99) and (sex=2)
result
gen meanmal=$mean1
```

We don't even need to write the line "result" if we know how EpiData Analysis defines the variable we require.

Visualizing with BROWSE:

meanfem	meanmal
37.9907	38.3895
37.9907	38.3895
37.9907	38.3895

It is not of particular value here (every record has the same values), but this is a potentially very powerful tool to further process data.

Writing output into a file

You learned perhaps a bit mechanically to start every program with a command "logclose" to close any log file that might be open, but you haven't quite tested the opposite, the opening of a log file. You can write any output into three types of files, text, HTML, or Excel. We strongly discourage Excel, because for proprietary reasons EpiData is forced to use an outdated format. If you need your output in a spreadsheet program, it is way preferable to save the output into a text file and then open the text file in your spreadsheet application software.

The commands are straight forward:

```
logopen "my_output.txt" /replace
* Some EpiData Analysis commands
logclose
```

Specifically, if we write:

```
cls
close
read "abcd.rec"
```

```
logopen "output_01.txt" /replace
tables sex reason
logclose
```

If we look at the output_01.txt file in our text editor, we see:

```
Logopen output_01.txt
EpiData Analysis V2.2.2.180 27-Apr-13 17:45
. tables sex reason
Examination reason      Female      Male      Total
Diagnosis      52      82      134
Follow-up at 2 months  18      22      40
Follow-up at 3 months   4       8       12
Follow-up at 4 months   0       3       3
Follow-up at 5 months  13      22      35
Follow-up at 6 months  11      20      31
Follow-up at 7 months or later      3      13      16
Follow-up, month not stated  8      19      27
Reason not stated  0       2       2
Total 109      191      300
```

```
. logclose
```

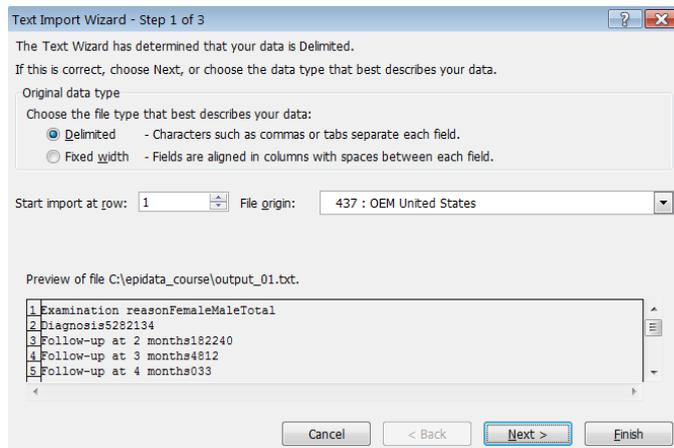
Apart from the “ragged” appearance, there is also too much superfluous output there, if we only want the table. We can greatly improve it by setting the echo first off before opening the log file and then setting it on again after it has been closed:

```
set echo=off
logopen "output_01.txt" /replace
tables sex reason
logclose
set echo=on
```

This way, we get it trimmed down:

```
Examination reason      Female      Male      Total
Diagnosis      52      82      134
Follow-up at 2 months      18      22      40
Follow-up at 3 months      4      8      12
Follow-up at 4 months      0      3      3
Follow-up at 5 months      13      22      35
Follow-up at 6 months      11      20      31
Follow-up at 7 months or later      3      13      16
Follow-up, month not stated      8      19      27
Reason not stated      0      2      2
Total 109      191      300
```

If we go now to our spreadsheet program and open it as a delimited text file:



it will come out nicely as intended:

	A	B	C	D
1	Examination reason	Female	Male	Total
2	Diagnosis	52	82	134
3	Follow-up at 2 months	18	22	40
4	Follow-up at 3 months	4	8	12
5	Follow-up at 4 months	0	3	3
6	Follow-up at 5 months	13	22	35
7	Follow-up at 6 months	11	20	31
8	Follow-up at 7 months or later	3	13	16
9	Follow-up, month not stated	8	19	27
10	Reason not stated	0	2	2
11	Total	109	191	300

More about tables

Let's first copy our case definition from above:

```
cls
close
read "abcd.rec"

define case #
case=0
if res1>0 and res1<9 then case=1
if res2>0 and res2<9 then case=1
if res3>0 and res3<9 then case=1
label case "Microscopy case definition"
labelvalue case /0="Non-case"
labelvalue case /1="Case"
```

and then make another categorical variable from the categorical variable reason to get only three levels and verify that we got what we intended to get, making a cross-table between the old variable (reason) and the new one (reason2) derived from it:

```
cls
gen i reason2=2
if reason=0 then reason2=1
if reason=9 then reason2=9
label reason2 "Reason for examination"
labelvalue reason2 /1="Diagnosis"
labelvalue reason2 /2="Follow-up"
labelvalue reason2 /9="Reason unknown"
tables reason2 reason
```

Reason for examination				
Examination reason	Diagnosis	Follow-up	Reason unknown	Total
Diagnosis	134	0	0	134
Follow-up at 2 months	0	40	0	40
Follow-up at 3 months	0	12	0	12
Follow-up at 4 months	0	3	0	3
Follow-up at 5 months	0	35	0	35
Follow-up at 6 months	0	31	0	31
Follow-up at 7 months or later	0	16	0	16
Follow-up, month not stated	0	27	0	27
Reason not stated	0	0	2	2
Total	134	164	2	300

Previously when we made a table also showing the values with:

```
tables case sex /v1
```

we got:

Microscopy case definition			
Examinee's sex	0 Non-case	1 Case	Total
1 Female	94	15	109
2 Male	168	23	191
Total	262	38	300

If we look at the sorting sequence, we see that the sequence is ascending by value (not ascending by label alphabet, see labels for case definition). We can invert the sequence with:

```
tables case sex /v1 /sd
```

Microscopy case definition			
Examinee's sex	1 Case	0 Non-case	Total
2 Male	23	168	191
1 Female	15	94	109
Total	38	262	300

There are multiple sorting options but one should take note that some more complex sorting options do not deliver what we expect, thus always check carefully. For simple sorting like ascending on values (/sa) or descending on values (/sd) as we just did, we usually get what we want.

Important to note is what happens if we like to get an odds ratio, compare the two outputs:

```
cls
tables case sex
tables case sex /o
```

“Plain”

Microscopy case definition			
Examinee's sex	Non-case	Case	Total
Female	94	15	109
Male	168	23	191
Total	262	38	300

“Epidemiologic”

Outcome:Microscopy case definition			
Examinee's sex	Case	Non-case	Total
Male	23	168	191
Female	15	94	109
Total	38	262	300

Exposure: Examinee's sex = Male
Outcome: Microscopy case definition = Case

Odds Ratio = 0.86 (95% CI: 0.43-1.72) (Robins,Greenland,Breslow CI)

This is a reflection of the fact that EpiData Analysis is truly an epidemiologist’s tool: most commonly in outbreak investigations or case-control studies we show the cases to the left and the non-cases to the right of the column, and show the exposure on top and the non-exposure at the bottom of the row.

If we now want to switch these, then we have to check carefully whether we get what we want, but one or the other options will give us what we need. Let’s try then with our logic from above telling us that we should probably sort in ascending order:

```
tables case sex /o /sa
```

Outcome:Microscopy case definition			
Examinee's sex	Non-case	Case	Total
Female	94	15	109
Male	168	23	191
Total	262	38	300

Exposure: Examinee's sex = Female
Outcome: Microscopy case definition = Non-case

Odds Ratio = 0.86 (95% CI: 0.43-1.72) (Robins,Greenland,Breslow CI)

Indeed, it did invert it. If we wish to invert only one of the two, say keep the “epidemiologic” presentation for cases and non-cases, but change it for sex, then we might need to do some recoding:

```
cls
gen i sexinvert=1
if sex=1 then sexinvert=2
label sexinvert "Sex of person"
labelvalue sexinvert /1="Male"
labelvalue sexinvert /2="Female"
tables case sexinvert /o
```

and we get:

Outcome:Microscopy case definition			
Sex of person	Case	Non-case	Total
Female	15	94	109
Male	23	168	191
Total	38	262	300

Exposure: Sex of person = Female
Outcome: Microscopy case definition = Case

Odds Ratio = 1.17 (95% CI: 0.58-2.34) (Robins,Greenland,Breslow CI)

Again, be sure to always check that what you get is what you need and learn creatively to approach things as you need them to be.

Stratification

So far, we have dealt with two-by-two tables, but one of the common requirements is a stratified analysis. EpiData Analysis makes it easy for us in that we just list the variables in a `tables` command, though we must pay attention to the sequence of the variables. If we need to look at cases versus non-cases by sex, stratified by the (summary) reason, we write:

```
tables case sex reason2 if reason2<>9
```

and get:

Microscopy case definition			
Examinee's sex	Non-case	Case	Total
Female	94	15	109
Male	167	22	189
Total	261	37	298

Unstratified table

Reason for examination: Diagnosis Microscopy case definition			
Examinee's sex	Non-case	Case	Total
Female	40	12	52
Male	64	18	82
Total	104	30	134

Reason for examination: Follow-up Microscopy case definition			
Examinee's sex	Non-case	Case	Total
Female	54	3	57
Male	103	4	107
Total	157	7	164

The first table is the crude, unstratified table, the following table(s) are the results of case by sex in the different strata. Of course, the interest here is commonly a summary odds ratio calculated by the Mantel-Haenszel procedure:

```
tables case sex reason2 /o if reason2<>9
```

We get more output here: first the “epidemiologic” table with crude odds ratios and 95% confidence intervals followed by these measures of association in the strata:

Outcome: Microscopy case definition				
Examinee's sex	Case	Non-case	Total	
Male	22	167	189	
Female	15	94	109	
Total	37	261	298	

Unstratified table
Exposure: Examinee's sex = Male
Outcome: Microscopy case definition = Case

Odds Ratio = 0.83 (95% CI: 0.41-1.67) (Robins, Greenland, Breslow CI)

Outcome: Reason for examination: Diagnosis Microscopy case definition				
Examinee's sex	Case	Non-case	Total	
Male	18	64	82	
Female	12	40	52	
Total	30	104	134	

Exposure: Examinee's sex = Male
Outcome: Microscopy case definition = Case

Odds Ratio = 0.94 (95% CI: 0.41-2.15) (Robins, Greenland, Breslow CI)

Outcome: Reason for examination: Follow-up Microscopy case definition				
Examinee's sex	Case	Non-case	Total	
Male	4	103	107	
Female	3	54	57	
Total	7	157	164	

Exposure: Examinee's sex = Male
Outcome: Microscopy case definition = Case

Odds Ratio = 0.70 (95% CI: 0.15-3.24) (Robins, Greenland, Breslow CI)

After these tables, we get the summary of the adjusted analysis:

Microscopy case definition by Examinee's sex adjusted for Reason for examination				
	N = 298	N	OR	(95% CI)
Crude		298	0.83	(0.41-1.67)
Adjusted		298	0.88	(0.42-1.82)
Reason for examination: Diagnosis	134	0.94		(0.41-2.15)
Reason for examination: Follow-up	164	0.70		(0.15-3.24)

Summary Estimates
 Total 2 strata, 2 informative & 0 non-informative.
 Exposure: Examinee's sex = Male
 Outcome: Microscopy case definition = Case

Note also the all-important information at the bottom to help us ensuring that we got what we wanted:

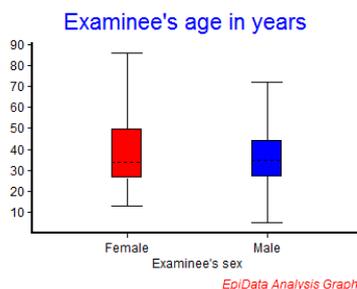
Exposure: Examinee's sex = Male
 Outcome: Microscopy case definition = Case

More about graphs

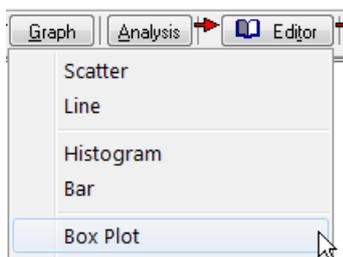
In the previous exercise, we did:

```
select age<>99
boxplot age /by=sex
```

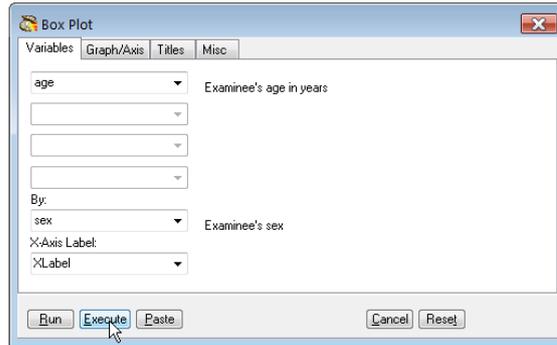
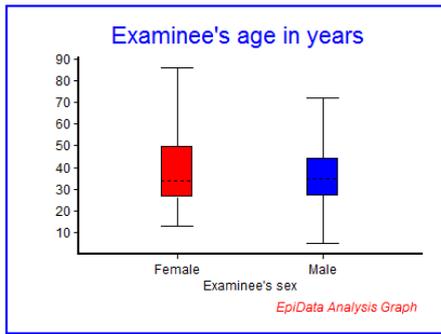
and got this graph:



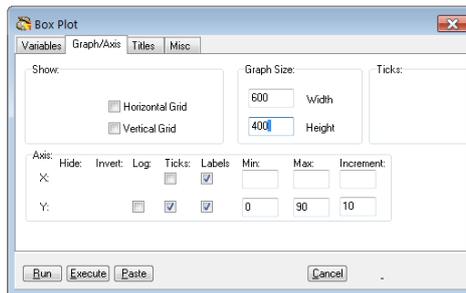
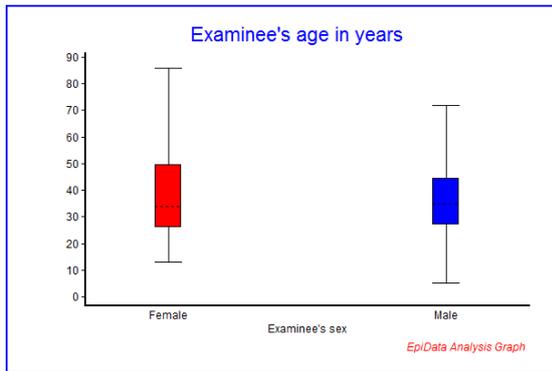
To explore the options a bit more without first taking recourse to the Help file, we will approach it by using the menu interactively:



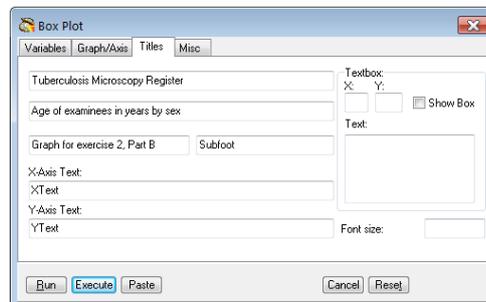
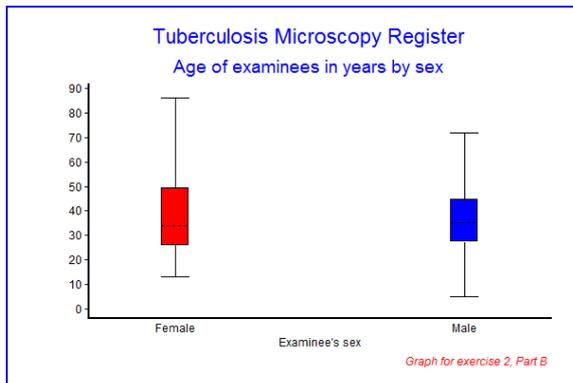
We will use Execute (not Run) to progressively edit the graph to our liking, completing the first tab "Variables":



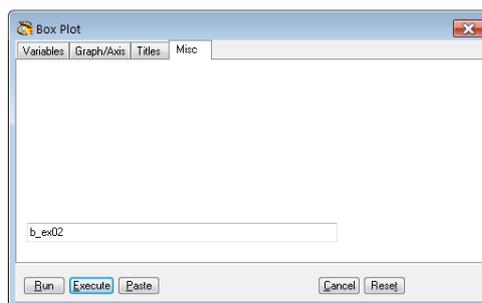
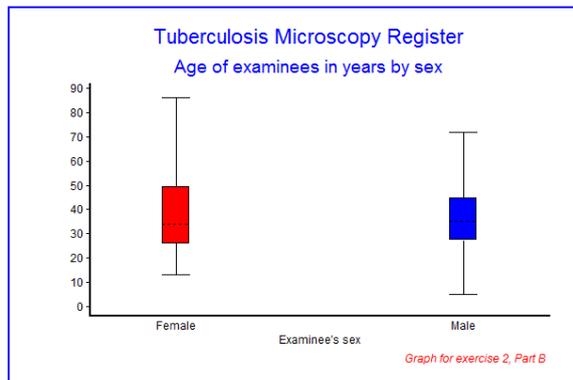
Then we refine the tab “Graph/Axis”:



then “Titles”:



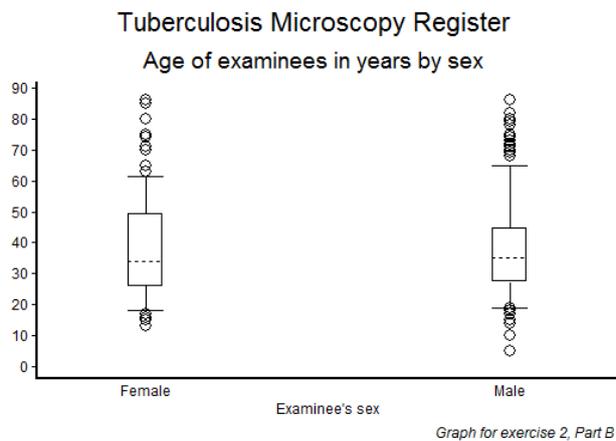
and finally “Misc”:



The most important part comes now in that we go to the command line (**F4**) and use the up cursor to get the entire list of commands, mark it, and paste it into our PGM file, making back slashes where appropriate and then have:

```
cls
BOXPLOT age /By=sex \
           /SizeX=600 /SizeY=400 \
           /Noxtick \
           /Ymin=0 /Ymax=90 /Yinc=10 \
           /Ti="Tuberculosis Microscopy Register" \
           /Sub="Age of examinees in years by sex" \
           /Fn="Graph for exercise 2, Part B" \
           /Save="b_ex02"
```

We do some additional editing, like allowing replacement of the graph (which is necessary if we name the output), and look some additional things up in the Help file, and finally have it refined to:



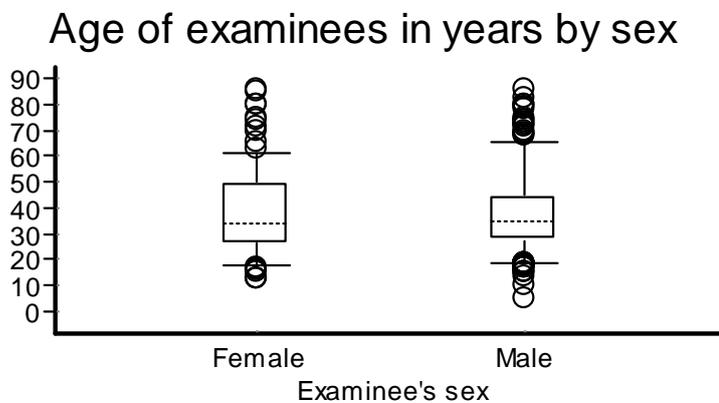
Note the difference in quality if we replace:

```
/Save="b_ex02" /replace \
```

with:

```
/Save="b_ex02.wmf" /replace \
```

Tuberculosis Microscopy Register

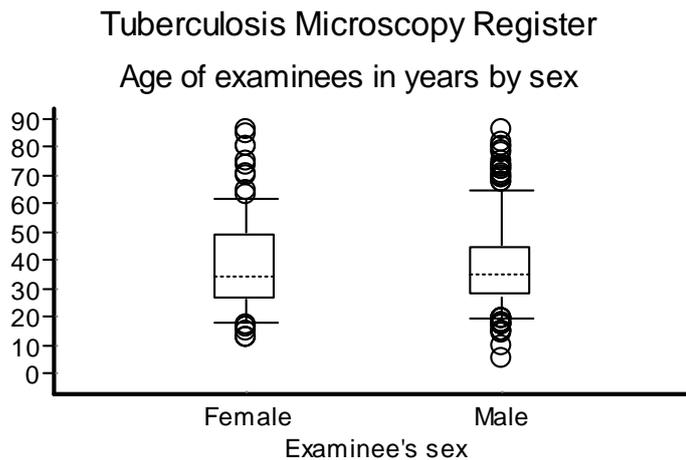


We note:

- If we save the graph as *.wmf (Microsoft metafile, a vector graph), it does not show up in the Results window
- Vector graphs are crisp and sharp and retain this irrespective of changing the graph size
- The result is not a faithful reproduction of the default *.png graph.

Sometimes the title gets too large in the metafile. This can be remediated by adding the SET to change the default font size from 10 to 9:

```
set graph font size=9
```



Graph for exercise 2, Part B

Task:

- o Write a program *B_EX02.PGM* using the data set “*abcd.rec*”. Limit your analysis to patients with a diagnostic sputum smear examination. Create an output that shows the incremental yield of cases from the first, second and third of three serial smears. With incremental yield we mean determining the proportion of examinees who are positive already on the first, patients who are negative on the first, but positive on the second, and examinees who are negative on the first two but positive on the third serial smear examination. The denominator should be those who have had the required number of smears to determine the yield.

Solution to Exercise 2: More on EpiData Analysis

Key point(s):

- EpiData Analysis is a powerful tool to manipulate and restructure data sets
- Tabular outputs can be customized to specific needs
- To evaluate graph options, one may best use first the visual menu-driven interface and then copy and paste the resulting options into a program for permanent reproduction.

Task:

- Write a program *B_EX02.PGM* using the data set “*abcd.rec*”. Limit your analysis to patients with a diagnostic sputum smear examination. Create an output that shows the incremental yield of cases from the first, second and third of three serial smears. With incremental yield we mean determining the proportion of examinees who are positive already on the first, patients who are negative on the first, but positive on the second, and examinees who are negative on the first two but positive on the third serial smear examination. The denominator should be those who have had the required number of smears to determine the yield.

Solution:

The desired output is:

Yield of first smear: Px [exclude none]

Six essential patterns

	N	%	(95% CI)
NNN	104	77.6	(69.8-83.8)
NPx	10	7.5	(4.1-13.2)
Px	20	14.9	(9.9-21.9)
Total	134	100.0	

Yield of second smear: NPx [exclude: N99]

Six essential patterns

	N	%	(95% CI)
NNN	104	77.6	(69.8-83.8)
NPx	10	7.5	(4.1-13.2)
Px	20	14.9	(9.9-21.9)
Total	134	100.0	

Yield of third smear: NNP [exclude: N99, NN9]

Six essential patterns

	N	%	(95% CI)
NNN	104	77.6	(69.8-83.8)
NPx	10	7.5	(4.1-13.2)
Px	20	14.9	(9.9-21.9)
Total	134	100.0	

The part of the program B_EX02.PGM which gave this output:

* Task Exercise 2, Part B

```
cls
close
read "abcd.rec"

define case #
case=0
if res1>0 and res1<9 then case=1
if res2>0 and res2<9 then case=1
if res3>0 and res3<9 then case=1
label case "Microscopy case definition"
labelvalue case /0="Non-case"
labelvalue case /1="Case"

cls
gen s(3) esspattern="NNN"
if res3=9 then esspattern="NN9"
if res2=9 then esspattern="N99"
if res3>0 and res3<9 then esspattern="NNP"
if res2>0 and res2<9 then esspattern="NPx"
if res1>0 and res1<9 then esspattern="Px"
label esspattern "Six essential patterns"

set echo=off
select reason=0
cls
type "Yield of first smear: Px [exclude none]" /h2
freq esspattern /c /ci
type "Yield of second smear: NPx [exclude: N99]" /h2
freq esspattern /c /ci if esspattern<>"N99"
type "Yield of third smear: NNP [exclude: N99, NN9]" /h2
freq esspattern /c /ci if esspattern<>"NN9" and esspattern<>"N99"
select
set echo=on
```

Exercise 3: Aggregating data and saving the summary data in a file

At the end of this exercise you should be able to:

- a. Understand “Aggregate” data and apply it to a specific task

We are quite familiar with:

```
cls
close
logclose

read "abcd.rec"

freq sex
```

getting as result:

Examinee's	
sex	N
Female	109
Male	191
Total	300

If we replace “freq” with “agg” (short for aggregate):

```
agg sex
```

we get essentially the same information (without the marginal, i.e. here the total):

sex	N
Female	109
Male	191

Making a frequency of a variable is nothing other than aggregating the values of that variable to the smallest common denominator and the same thing is done with aggregate. We can do it analogously for a table:

```
tables sex labcode
```

This gives (discounting the marginal) 8 inner cells for the 2 by 4 possibilities:

Examinee's sex			
Laboratory code	Female	Male	Total
A	33	42	75
B	31	44	75
C	24	51	75
D	21	54	75
Total	109	191	300

If we replace tables by aggregate:

```
agg sex labcode
```

we get the 8 inner cells listed in a column sorted by sex then by labcode:

sex	labcode	N
Female	A	33
Female	B	31
Female	C	24
Female	D	21
Male	A	42
Male	B	44
Male	C	51
Male	D	54

As an option we can close the file that was open (the abcd.rec):

```
agg sex labcode /close
```

and we can see what remains in the Variables window:

sex	I	Examinee's sex
labcode	S	Laboratory code
n	I (N)	Total observations used in aggregate

We can browse it as it has been written to an EpiData REC file, aggregated into 8 records:

	sex	labcode	N
1	Female	A	33
2	Female	B	31
3	Female	C	24
4	Female	D	21
5	Male	A	42
6	Male	B	44
7	Male	C	51
8	Male	D	54

As an additional option we can save this aggregation to a REC file (note that the option is “/save”, not “/savedata”):

```
agg sex labcode /close /save="labsex_set.rec" /replace
```

then close everything and re-open either file.

The first property of the aggregate command is thus that it can replace the summary of a tables command and save the cells of the table (made from 1, 2, or more variables) in a REC file.

The second property of the aggregate command is related to the power it provides with options. For example:

```
cls
close
logclose
read "abcd.rec"
agg sex /mean=age if age<>99
```

gives:

sex	N	Nage	MEAage
Female	109	109	39.11
Male	191	191	38.71

We have now the mean age by sex, but we could also get the mean age by sex and laboratory:

```
agg sex labcode /mean=age if age<>99
```

sex	labcode	N	Nage	MEAage
Female	A	33	33	42.97
Female	B	31	31	34.87
Female	C	24	24	37.75
Female	D	21	21	40.86
Male	A	42	42	37.60
Male	B	44	44	35.11
Male	C	51	51	39.24
Male	D	54	54	42.00

and as above, all can be written into a REC file:

```
agg sex labcode /mean=age /close /save="labsex_set_2.rec" /replace \
    if age<>99
cls
close
read "labsex_set_2.rec"
```

	sex	labcode	N	Nage	MEAage
1	Female	A	33	33	42.9696969697
2	Female	B	31	31	34.8709677419
3	Female	C	24	24	37.7500000000
4	Female	D	21	21	40.8571428571
5	Male	A	42	42	37.5952380952
6	Male	B	44	44	35.1136363636
7	Male	C	51	51	39.2352941176
8	Male	D	54	54	42.0000000000

The automatically created variable N denotes the number of records in the original file, Nage the number of records in the original file with non-missing information on age, and MEAage, the mean age in the aggregated stratum.

There are more options of this kind, like:

```
/min
/max
/stat
/sum
```

You can also cumulate different options, and you can use the same option for different variables, e.g. “/sum=var1 /sum=var2 /min=var3”.

It is the option /sum we are particularly interested here to apply. We can sum up all the values of a given variable across all records aggregated within a stratum. For instance, in the following task you will need to count the number of smears each examinee had, and then you can sum up all the smears within each stratum resulting from your aggregation.

Workload in the laboratory

One measure of the workload in laboratories is the number of smears they have to examine per day. Even the busiest laboratories do not work every day, they may close on weekends and public holidays. One may get some approximate estimate of the number of working days, but a much cleaner way is to count the actual working days. The tuberculosis laboratory register gives a possible good approximation with the date of registration. While smears are also examined on other than the registration date (the first specimen defines that date, and a patient gives the first specimen on the spot but brings in an early morning specimen one day later), this is a reasonably good approximation to the number of days work was actually carried out, certainly better than some other approximation without a good data basis.

We provide a data set “b_ex03_workload.rec” from three laboratories in Zimbabwe (data courtesy Dr Biggie Mabaera), each complete but also limited to during one calendar year.

Task:

- o The B_EX03_WORKLOAD.REC has been edited to contain only three laboratories (out of the original 30) and only the year 2002. Nonsensical results (e.g., first examination not recorded, followed by a valid result) have been excluded. Create a program B_EX03.PGM to provide the mean number of smears examined per registration day in each of the three laboratories.*

Solution to Exercise 3: Aggregating data and saving the summary data

Key point(s);

- a) Aggregate is a powerful tool to summarize the analogue to a tabular output in a REC file and to make calculations on the vertical for each stratum thus obtained.

Task:

- o The B_EX03_WORKLOAD.REC has been edited to contain only three laboratories (out of the original 30) and only the year 2002. Nonsensical results (e.g., first examination not recorded, followed by a valid result) have been excluded. Create a program B_EX03.PGM to provide the mean number of smears examined per registration day in each of the three laboratories.*

The result:

(SUM) smears								
laboratory	Obs.	Sum	Mean	Variance	Std Dev	(95% CI mean)		Std Err
BY_A	242	23044.0	95.22	1069.10	32.70	91.08	99.36	2.10
ME_L	135	1211.00	8.97	53.34	7.30	7.73	10.21	0.63
ML_L	241	6328.0	26.26	244.65	15.64	24.27	28.24	1.01

The program B_EX03.PGM is very simple in the end (but admittedly it took us a while to get to this level of efficiency):

```
cls
close
logclose

read "b_ex03_workload.rec"

* Determine the number of smears for each examinee
gen i smears=1
if result2<>9 then smears=2
if result3<>9 then smears=3

* Sum up the number of smears done on each
* working day in each laboratory
aggregate regdate laboratory /sum=smears /close

* Calculate the average number of smears done
* each working day in each laboratory
cls
means sumsmears /by=laboratory
```

Exercise 4: From a spreadsheet to an EpiData file

At the end of this exercise you should be able to:

- a. Master the import of spreadsheet content into an EpiData file
- b. Recode the imported content to an analyzable file

Quite often data are captured in a spreadsheet. When trying to then analyze these data, the substantial limitations to the use of spreadsheet software in this respect become recognized. We are then compelled to import the spreadsheet data into a data file format more amenable to analysis. An EpiData file is an appealing solution as the software also disposes of powerful restructuring tools permitting reshaping the data into a format that is convenient for analysis using EpiData Analysis.

There are two approaches which are the same in principle: 1) copying the spreadsheet content to the clipboard, or 2) exporting the spreadsheet content to a delimited text file. EpiData Analysis is given a respective command to either read the clipboard content or the text file. We will demonstrate both approaches with two simple examples and then provide for the task a real dataset of somewhat more complexity.

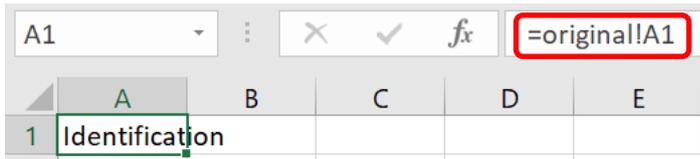
Formatting the spreadsheet correctly

In the required files, you find the workbook “b_ex04_spreadsheet_data_1_original.xlsx” with 8 records (data rows) and 6 variables (column headings):

	A	B	C	D	E	F
1	Identification	sex	date of birth	marital status	Living arrangements	housing surface in sqm
2	A	m		married	apartment lease	
3	B	F	20-02-01	divorced	Living alone	80
4	C	f	15-12-95	widowed	house owner	150
5	D	m	26-07-85	cohabitating	apartment	105
6	E	M	24-03-03	married	own house	180
7	F		04-04-79	married	leasing an apartment	95
8	G	f	17-02-98	not married	apartment	110
9	H	m	12-01-05	unmarried	living with parents	75

The workbook has this single sheet labeled “original”. Under no circumstance do we wish to make any alteration whatsoever to the original. We therefore make as the first thing a copy of this content into a new sheet in the same workbook. We create a new sheet and label it “epidata”. We don’t want to copy and paste but we rather resolve the copying by using a formula.

In the new sheet “epidata” in cell A1 we type the “=” sign (no quotes) and then place the cursor into cell A1 in the “original” sheet and press the Enter key. This gets us back to cell A1 in the “epidata” sheet in which we now find written:



Copy cell A1 in “epidata” to the first two rows, columns A to F and get:

	A	B	C	D	E	F	G
1	Identificat	sex	date of bir	marital sta	Living arra	housing surface in sqm	
2	A	m	0	married	apartmen	0	0
3							

There are 3 rules that must be followed else the transfer of data from spreadsheet to EpiData will either not function or it will contain errors:

- 1) The variable names must be valid EpiData names i.e. a) single word, b) length not exceeding 10 characters, c) not beginning with a number.
- 2) The first row of data cannot contain any empty cell.
- 3) The data for a given variable must all be of the same type.

We thus overwrite the first row containing the variable names with valid EpiData names. Doing this, we remember that EpiData is not case sensitive, but many other analysis software is (like Stata or R). Despite EpiData not being case sensitive for variable names, we will make it always strictly unambiguous with a simple rule: we recommend and use only lower-case letters for variable names. Thus, we might write:

	A	B	C	D	E	F
1	id	sex	dob	marital	living	sqmeter

While a length of up to 10 characters is allowed for EpiData variable names, we prefer to have the names even shorter: our longest names here have 7 characters. We will later make explicit variable labels so that it is unambiguously clear what the variable names refer to.

Next we modify the formula for the cells A1 to F1 in a way to take care of two things at the same time, i.e. providing a value for a given cell in case it is empty and defining the type of variable the column contains. We assume that column A with the identifier always contains a value (else we will discard records without identifier later in EpiData Analysis). Thus, cell A1 is left untouched. Column B is a string (text) field and it has length 1. Accordingly, we modify the current formula from:

`=original!B2`

to:

`=IF(original!B2<>"",original!B2,"x")`

If you are not familiar with the “IF” statements in spreadsheet syntax, you may consult the help file. Basically it is:

`IF(logical test, if true write this, else write that)`

We thus evaluate with the logical test whether the content of the cell in the sheet “original” is not empty (denoted as opening and closing double-quote “ ”), and if it is not empty, then we take that value, else an x is written (denoted as a string by surrounding it with double-quotes “x”).

We copy the thus revised formula of cell B2 to cell C2. “x” is obviously erroneous, because this is a date field, not a string field. It probably depends on the language of your spreadsheet software how a date is written, thus check first in the help file. In our English version it becomes:

```
=IF(original!C2<>"" ,original!C2,DATE(1900,1,1))
```

We need to choose a legal date, but one that obviously cannot be a genuine date of birth in our data set. We chose the earliest possible date in the spreadsheet after the anchor date (which is 31 Dec 1899, in EpiData it is 31 Dec 1799), thus the choice of 1 Jan 1900: it’s legal, it is written as an actual date, and it is obviously not a genuine date of birth in the context of the data set.

Cell D2 is a string analogous to cell B2 which we thus copy to here, but make perhaps 3 x’s as it is a longer field than just length 1:

```
=IF(original!D2<>"" ,original!D2,"xxx")
```

Cell E2 is actually the same as D2, so we copy cell D2 to cell E2. We introduce something more here. While it is actually not relevant for this specific case here, it can become relevant with text fields like comments: in EpiData a text field cannot exceed a length of 80 characters, else an error occurs when attempting to read a field with a larger length. To prevent that the value in such a string field exceeds the permissible length, we tell the spreadsheet that it should cut off anything beyond a certain length. We choose here (arbitrarily) that critical length to be 50 characters after which the rest is cut off and write:

```
=IF(original!E2<>"" ,MID(original!E2,1,50),"xxx")
```

The function in the spreadsheet software is “MID” and it requires three arguments, 1) the cell name, 2) the starting character position (1), and 3) the ending character position (50).

The final variable `sqmeter` (first data cell F2) is an integer field of a length 3, thus the formula becomes a copy of D2, yet the value for missing doesn’t have quotes and must be a number (because the variable is an integer field):

```
=IF(original!F2<>"" ,original!F2,999)
```

As the first two rows are now complete, row 2 can be copied up to and including row 9 and we get:

	A	B	C	D	E	F
1	id	sex	dob	marital	living	sqmeter
2	A	m	01-01-1900	married	apartment lease	999
3	B	F	20-02-2001	divorced	Living alone	80
4	C	f	15-12-1995	widowed	house owner	150
5	D	m	26-07-1985	cohabitating	appartment	105
6	E	M	24-03-2003	married	own house	180
7	F	x	04-04-1979	married	leasing an apartment	95
8	G	f	17-02-1998	not married	appartment	110
9	H	m	12-01-2005	unmarried	living with parents	75

If we mark the data rectangle A1:F9, copy it to clipboard and then paste it into our text editor, we get:

```

1 id sex dob marital living sqmeter
2 A m 01-01-1900 married apartment lease 999
3 B F 20-02-2001 divorced Living alone 80
4 C f 15-12-1995 widowed house owner 150
5 D m 26-07-1985 cohabitating appartment 105
6 E M 24-03-2003 married own house 180
7 F x 04-04-1979 married leasing an apartment 95
8 G f 17-02-1998 not married appartment 110
9 H m 12-01-2005 unmarried living with parents 75

```

This suggests that the clipboard memory content has a Tab-delimited format. You can test this assumption by placing your cursor after the “A” and then move the cursor: it jumps to before “m”. In Notepad++ text editor (available on the course web) you can choose View=>Show symbol=>Show white space and TAB and you actually see spaces as points and TABs as → arrows:

```

1 id→sex→dob→marital→living→sqmeter
2 A→m→01-01-1900→married→apartment·lease→999
3 B→F→20-02-2001→divorced→Living·alone→80
4 C→f→15-12-1995→widowed→house·owner→150
5 D→m→26-07-1985→cohabitating→appartment→105
6 E→M→24-03-2003→married→own·house→180
7 F→x→04-04-1979→married→leasing·an·apartment→95
8 G→f→17-02-1998→not·married→appartment→110
9 H→m→12-01-2005→unmarried→living·with·parents→75

```

EpiData will correctly interpret the difference between a TAB delimiter and one or more space characters.

Reading data from clipboard into EpiData Analysis and save it as an EpiData file

In EpiData Analysis we prepare the basics to read data from clipboard and save them to an EpiData *.REC file:

```

cls
close
logclose

```

```
read /cb
savedata "spreadsheet_data_1.rec" /replace
```

The command is remarkably simple: `read` with the option `/cb` (for clipboard). It makes of course sense to save it right away from memory to a data file with an intermediary name of your choice. Also, save the program as “`b_ex04_task.pgm`”.

Switch to the sheet “`epidata`” and copy the rectangular data `A1:F9` to the clipboard, then switch back to EpiData analysis and run all with `F9`. The output should then read:

```
. read /cb
Loading data from clipboard, please wait...
First line: (clipboard) id sex dob marital living sqmeter
Separator: Tab (tab: 45) (semicolon: 0) (comma: 0) (space: 9) Lines: 9 (incl. field names)
File name :from clipboard
Fields: 6 Total records: 8 Included: 8
. savedata "spreadsheet_data_1.rec" /replace
Saving data to: c:\epidata_course\spreadsheet_data_1.rec
6 Fields 8 Records
```

If we browse, we get:

	id	sex	dob	marital	living	sqmeter
1	A	m	01/01/1900	married	apartment lease	999
2	B	F	20/02/2001	divorced	Living alone	80
3	C	f	15/12/1995	widowed	house owner	150
4	D	m	26/07/1985	cohabitating	apartment	105
5	E	M	24/03/2003	married	own house	180
6	F	x	04/04/1979	married	leasing an apartment	95
7	G	f	17/02/1998	not married	apartment	110
8	H	m	12/01/2005	unmarried	living with parents	75

From the Variables window (press `F3`, if not shown spontaneously), we check whether all the variables are of the type that we wanted them to be, most notably that dates are date variables and numbers are integer or float variables:

```
id      S id
sex     S sex
dob     D dob
marital S marital
living  S living
sqmeter I sqmeter
```

This is the case here, but this should not just be assumed: we may encounter that a date field has become a text field because the spreadsheet column hadn’t been formatted correctly or that one single date among all had a wrong format. It is crucial to ensure that all is in order in respect to the type of variable.

It is immediately apparent in the small dataset here that all records have an identifier. However, we better check it formally as we cannot use records that cannot be checked against the original. We make a binary variable:

```
cls
close
read "spreadsheet_data_1.rec"

cls
gen i hasid=1
if id=. then hasid=0
freq hasid /m
```

and get:

hasid	N
1	8
Total	8

We are thus satisfied that all records have an identifier. That none is a duplicate will be done later again in another form, but this is how we would do that:

```
cls
sort id
gen i seq=1
if id=id[_n-1] then seq=seq[_n-1]+1
freq seq
```

and we get:

seq	N
1	8
Total	8

No identifier occurs more than once and every record has an identifier. What we did here was to sort the records by ID to ensure that any identical identifiers would be next to each other. Then we make a variable SEQ and give it in every record the default value 1. A powerful capability of EpiData Analysis is to look at records before or after the current position of the process. EpiData Analysis proceeds record by record through a data file, from top to bottom. The current record has the position [_n]. There is no need to give the designation for the current record, but in fact id and id[_n] are both valid designations for the identifier in the current position. id[_n-1] designates the identifier of the record immediately preceding that of the current record, while id[_n+5] would be the identifier of the fifth record forward from the current one. Our commands above thus say “if the record before the current one has the same identifier as the current one, then add 1 to the value of seq in the current record to the value that seq has in the record before the current record”. As a result, all seq values must be 1 if there are no multiples of the identifier. We will make extensive use of this positioning when we deal with the analysis of a relational database. For the time being we

can asterisk the frequencies and drop the two variables we just created, and give a label to the variable ID:

```
cls
close
read "spreadsheet_data_1.rec"

cls
gen i hasid=1
if id=. then hasid=0
* freq hasid /m

cls
sort id
gen i seq=1
if id=id[_n-1] then seq=seq[_n-1]+1
* freq seq

drop hasid seq

label id "Unique identifier"
```

It is still common practice to use string (text) fields for the values of categorical variables like sex or housing arrangement, etc, as was also done here in the spreadsheet. For instance, a frequency of the variable SEX gives here:

sex	
	N
f	2
F	1
m	3
M	1
x	1
Total	8

The value “x” was our doing to denote the value in records with no information on the variable SEX. In this spreadsheet, there was no control over the way how the value should be written and we have thus 4 instead of the 2 known sexes. Of course, in EpiData it is easy to exert control to avoid such errors (it’s also possible in a spreadsheet). In any case, there are issues with using text values for categorical variables. There are good reasons why it is nowadays standard to use systematically numeric coding for all variables unless there is a compelling reason to use a string (such as always for identifiers as nobody uses them to count something). The main reason for numeric coding is efficiency. Evaluation of text fields in analysis is “expensive” as text must be evaluated, then converted into numbers internally: computers don’t add up letters, they add up numbers. Text values take processing time in excess of what is required if all calculation can be done directly with numbers. Numeric coding is also efficient for file size and thus another factor for processing speed: to be unambiguous and clear, the required length of a field containing a text value for a categorical field is often more than 1, while with numeric coding it can commonly be just 1 (allowing up to 10 strata).

In the following, we will convert all categorical text fields to fields with numeric coding with unambiguous metadata for the labels.

Starting with the variable `SEX`, we can use the `EpiData` function to convert all text values (m and M, and f and F) to lower case to get more swiftly from the current 5 to the desired 3 required values:

```
cls
freq sex
define sexn #
sexn=9
if lower(sex)="f" then sexn=1
if lower(sex)="m" then sexn=2
tables sexn sex
```

The function is `LOWER(variablename)`. To do this systematically, we start with a frequency of the current variable, and end with a cross-tabulation of the new against the old variable to check that all assignments are correct. We could then drop the two respective commands, but we prefer to leave them with a preceding asterisk denoting it as a comment. This is a comment in the script to assure ourselves that we had checked. Then we drop the old variable and label the new one properly, followed by a final frequency of the new variable to assure ourselves that all is in order:

```
cls
* freq sex
define sexn #
sexn=9
if lower(sex)="f" then sexn=1
if lower(sex)="m" then sexn=2
* tables sexn sex
drop sex
rename sexn to sex
label sex "Patient's sex"
labelvalue sex /1="Female"
labelvalue sex /2="Male"
labelvalue sex /9="Not recorded"
* freq sex
```

The output is now neat and clean without ambiguity:

Patient's	
sex	
	N
Female	3
Male	4
Not recorded	1
Total	8

The variable `DOB` only needs a proper variable label:

```
label dob "Date of birth"
```

The variable `LIVING` has obviously been made free text with the resulting short-comings:

living	
	N
apartment lease	1
apartment	2
house owner	1
leasing an apartment	1
Living alone	1
living with parents	1
own house	1
Total	8

We have 7 values in 8 records. Such coding could be very tedious and is obviously pretty useless for analysis as such. There is no other way than actually going through the actual original values one by one and trying to put them into reasonable categories. We could be forgetting one or the other value as it is so tedious. To have an alert for omissions, we make a default value (choosing here "8") that should not remain if all re-assignments are proper. We also make use of the function SUBSTR(variablename, beginning position, ending position) to avoid copying all of it. The latter is unfortunately often required, but not in the relatively simple case here:

```
cls
* freq living
define livingn #
livingn=8
if living="xxx" then livingn=9
if substr(living,1,4) ="apar" then livingn=1
if substr(living,1,4) ="appa" then livingn=1
if substr(living,12,4)="apar" then livingn=1
if substr(living,1,4) ="hous" then livingn=2
if substr(living,5,4) ="hous" then livingn=2
if substr(living,13,4)="pare" then livingn=3
if substr(living,8,4) ="alon" then livingn=4
* tables livingn living
drop living
rename livingn to living
label living "Housing arrangements"
labelvalue living /1="Apartment"
labelvalue living /2="House"
labelvalue living /3="Parents"
labelvalue living /4="Other"
labelvalue living /9="Not recorded"
```

The recoded field has somewhat more appeal than the original:

Housing arrangements	
	N
Apartment	4
House	2
Parents	1
Other	1
Total	8

We finalize recoding with the field MARITAL analogously and provide the field SQMETER with just a variable label so that in the end we have a neatly recoded dataset:

	id	dob	sqmeter	sex	living	marital
1	A	01/01/1900	999	Male	Apartment	Married
2	B	20/02/2001	80	Female	Other	Divorced
3	C	15/12/1995	150	Female	House	Widowed
4	D	26/07/1985	105	Male	Apartment	Cohabiting
5	E	24/03/2003	180	Male	House	Married
6	F	04/04/1979	95	Not recorded	Apartment	Married
7	G	17/02/1998	110	Female	Apartment	Single
8	H	12/01/2005	75	Male	Parents	Single

Exporting the spreadsheet to a delimited text file and importing that file into EpiData

We mentioned above that copying the data in the spreadsheet to clipboard and exporting the data to a delimited text file is essentially the same approach. We showed that the copying to the clipboard is copying a tab-delimited file to the clipboard that EpiData Analysis understands to be of this format with the option /cb. EpiData Analysis also allows importing text files that are physically on disk and have a name and an extension that identifies them as a text file. Text files have the extension *.txt, but this extension does not define the nature of the delimiter, i.e. the separator between the values for separate variables. In EpiData Entry 3.1 we can import files with the extension *.txt and tell the software which delimiter the file actually has (i.e. Tab, comma, semi-colon). The issue is of importance because the choice of the delimiter must always be something that is not used in the data: for instance, a tab will not be used between words of the value in a text field, as we use a space character to separate words. In the US and many other countries, the decimal separator is the full stop. In many other countries, the decimal separator is a comma. In our Windows® software we make in our set-up the definitions and our spreadsheet software will then adhere to these rules. Not everybody actually makes the effort to properly set up the display rules for the display of numbers. As a result, we must be very careful to check whether what we demand is then actually what we get.

A point in case is the most commonly used delimited text file format, the comma-separated values, with the extension *.csv. Fortunately, it seems, the issue is solved correctly by spreadsheet software such as Excel® or LibreOffice Calc:

- If the Windows set-up is to use a full stop as the decimal separator, then the delimiter between variables will be a comma when exporting to *.csv.
- If the Windows set-up is to use a comma as the decimal separator, then the delimiter between variables will be a semicolon when exporting to *.csv.

EpiData Analysis will be able to correctly identify the delimiter either way (but always check!). We will thus export to *.csv. Once you have exported, examine it in the text editor.

When using a full stop as decimal separator, we get:

```
1 id,sex,DOB,marital,living,scmeter
2 A,m,01-01-1900,married,apartment.lease,999
3 B,F,20-02-2001,divorced,Living.alone,80
4 C,f,15-12-1995,widowed,house.owner,150
5 D,m,26-07-1985,cohabitating,apartment,105
6 E,M,24-03-2003,married,own.house,180
7 F,x,04-04-1979,married,leasing.an.apartment,95
8 G,f,17-02-1998,not.married,apartment,110
9 H,m,12-01-2005,unmarried,living.with.parents,75
```

When using a comma as decimal separator, we get:

```
1 id;sex;DOB;marital;living;scmeter
2 A;m;01-01-1900;married;apartment.lease;999
3 B;F;20-02-2001;divorced;Living.alone;80
4 C;f;15-12-1995;widowed;house.owner;150
5 D;m;26-07-1985;cohabitating;apartment;105
6 E;M;24-03-2003;married;own.house;180
7 F;x;04-04-1979;married;leasing.an.apartment;95
8 G;f;17-02-1998;not.married;apartment;110
9 H;m;12-01-2005;unmarried;living.with.parents;75
```

We note that this works correctly even if – like in our case – there is not even a real number (float variable) in the data set. Important is here to note again that the two approaches are the same, except that one is tab-limited, the other comma (or semicolon) delimited. Neither will work if the content is not properly formatted!

More complexity: two specimens with a discordant result taken at the same time

In the workbook “b_ex04_2_original.xlsx” we have a bit added complexity. The unit of observation here is not one patient but one specimen. Specimens are taken at different visits of individual patients. Furthermore, it is possible that more than one specimen is taken at one given visit. We added the case where two different specimens yield a different result at one given visit. What we wish to do is to 1) import the data from the spreadsheet and then 2) manipulate the EpiData file so that each observation time per patient has a single hierarchically dominant result and that the way to go about it would not be limited to just two specimens as in our example.

We proceed as before by creating a second sheet “epidata” in the workbook that contains the EpiData-compatible reformatted data from the “original” sheet. It is easiest to open the “b_ex04_1.pgm” program in EpiData Analysis and save it right away as “b_ex04_2.pgm”. The rectangular file is then copied to clipboard and read and saved in EpiData Analysis as before. All the variable manipulations that exist are retained with perhaps minor modifications, and the additional ones are added.

Special consideration is given to the recoding of the specimen result. We have more than one specimen in a given month per patient and they may be discordant such as in:

PATID	MM	SPECSEQ	RESULT
A	1	1	negative
A	1	2	positive
A	1	3	no result

What we want to have is a single RESULT per patient-month. We also want this to be hierarchical, i.e. that any result overrides no result, and a positive result overrides a negative result, and that at the end of our procedure the hierarchically highest result is written into the first of the sequences, i.e. that the above becomes:

PATID	MM	SPECSEQ	RESULTN
A	1	1	positive
A	1	2	positive
A	1	3	no result

Then we can retain only the first record in the sequence, i.e. SPECSEQ=1 to have only one result per patient-month. There are of course different approaches to solving this, but we propose hierarchical numeric recoding, i.e.:

Result	Value
No result	0
Negative	1
Positive	2

We make first a variable that combines patient identifier and month to a new identifier (e.g. PATMMID):

```
gen s(3) patmmid=patid+"-"+mm
label patmmid "Patient-month identifier"
```

We then sort by this new identifier and within it by month, and number the sequence similar as we did before, i.e.:

```
sort patmmid mm
gen i seq=1
if patmmid[_n-1]=patmmid then seq=seq[_n-1]+1
```

Thus, we should get something of the type:

PATMMID	SEQ	MM	SPECID	RESULT
A-1	1	1	X123	negative
A-1	2	1	Y321	positive
A-1	3	1	A412	no result
A-2	1	2	B123	negative
B-0	1	0	C123	positive

The sequence of the RESULT is irrelevant as the hierarchical coding suffices to get things right in the next step. In this next step we reshuffle the hierarchically highest to the first in the sequence:

```
define resultn #
resultn=result
if result[_n+1]>result then resultn=result[_n+1]
drop result
rename resultn to result
label result "DST result"
labelvalue result /0="No result"
labelvalue result /1="Susceptible"
labelvalue result /2="Resistant"
```

and we will get something of the type:

PATMMID	SEQ	MM	SPECID	RESULT
A-1	1	1	X123	positive
A-1	2	1	Y321	positive
A-1	3	1	A412	no result
A-2	1	2	B123	negative
B-0	1	0	C123	positive

After this the specimen identifier is incorrectly paired with the result and can be dropped together with the sequence counter and we have a data set in which PATMMID is a unique identifier for patient-month (which we must check).

Tasks:

- o The B_EX04_TASK.XLSX is a real data set from a study on laboratory drug susceptibility test results from several countries / jurisdictions. Create a sheet satisfying EpiData format requirements. Copy the rectangular selection to the clipboard and read it into EpiData Analysis and save it to an EpiData *.REC file. Note that it might be tricky to deal properly with some variables that may prevent correct reading of the rectangular file!*
- o The data set has a patient identifier, IDCODE. This identifier is unique for the patient in a given jurisdiction, but it is not unique for the data set. Specimens might be taken at the start of treatment or at any month on treatment. One patient in the data set thus may have several results. Make a variable IDCODE2 by adding the month to IDCODE. This will make it unique for a given patient in a given month. Actually – not quite: there may be more than one specimen in a given month, but only one result should count. The IDCODE2 could also come from two different patients if they are from a different jurisdiction. Make a third identifier so that you get in total three: 1) taking jurisdiction, patient and month of treatment, 2) taking jurisdiction and patient and 3) taking patient into account.*
- o Sort the data set so that all specimens for a given patient-month are next to each other, then recode all variable pertaining to isoniazid in a hierarchical manner, so that it becomes easy to assign the value that will be retained to the first specimen for that patient-month. Hierarchy: high-level resistance>low-level resistance>susceptible>no result. Then select to retain only the first specimen per patient-month.*
- o Recode the various variables for isoniazid drug susceptibility test results (phenotypic and genotypic results). Recode to a single variable for the isoniazid result. Discuss the hierarchy that you wish to assign to this end.*

Solution to Exercise 4: From a spreadsheet to an EpiData file

At the end of this exercise you should be able to:

- a. Master the import of spreadsheet content into an EpiData file
- b. Recode the imported content to an analyzable file

Task:

- o The B_EX04_TASK.XLSX is a real data set from a study on laboratory drug susceptibility test results from several countries / jurisdictions. Create a sheet satisfying EpiData format requirements. Copy the rectangular selection to the clipboard and read it into EpiData Analysis and save it to an EpiData *.REC file. Note that it might be tricky to deal properly with some variables that may prevent correct reading of the rectangular file!*

Solution:

The tricky part was perhaps to deal with the original sheet variable IDCODE. If the values were left unchanged, EpiData may misread it as a date variable with erroneous dates that did not exist and in consequence the import failed. We solved it by unambiguously making it to a text field. The value:

2014-01-006

was made to be:

x-2014-01-006

by the following code:

```
=IF(original!G2<>"",CONCATENATE("x-", original!G2),"xxx")
```

The preceding “x-” identified the value unambiguously to a string field for EpiData Analysis. The extraneous “x-” was then easily stripped in EpiData Analysis with:

```
gen s(11) idcode0=substr(idcode2,3,11)
```

Task:

- o The data set has a patient identifier, IDCODE. This identifier is unique for the patient in a given jurisdiction, but it is not unique for the data set. Specimens might be taken at the start of treatment or at any month on treatment. One patient in the data set thus may have several results. Make a variable IDCODE2 by adding the month to IDCODE. This will make it unique for a given patient in a given month. Actually – not quite: there may be more than one specimen in a given month, but only one result should count.*

The IDCODE2 could also come from two different patients if they are from a different jurisdiction. Make a third identifier so that you get in total three: 1) taking jurisdiction, patient and month of treatment, 2) taking jurisdiction and patient and 3) taking patient into account.

Solution:

This is relatively simple to accomplish. We chose this approach:

```
cls
* freq country
define countryyn #
if country="A" then countryyn=1
if country="B" then countryyn=2
if country="C" then countryyn=3
if country="D" then countryyn=4
if country="E" then countryyn=5
if country="F" then countryyn=6
if country="G" then countryyn=7
if country="H" then countryyn=8
drop country
rename countryyn to country
label country "Name of country / jurisdiction"
labelvalue country /1="Jurisdiction A"
labelvalue country /2="Jurisdiction B"
labelvalue country /3="Jurisdiction C"
labelvalue country /4="Jurisdiction D"
labelvalue country /5="Jurisdiction E"
labelvalue country /6="Jurisdiction F"
labelvalue country /7="Jurisdiction G"
labelvalue country /8="Jurisdiction H"

cls
* Create derived identifiers
gen s(16) idcoden=country+substr(idcode,2,12)+"-"+mmtxt
drop idcode2 idcode
rename idcoden to idcode2
label idcode2 "Country-patient-month identifier"
cls
gen s(13) idcode=substr(idcode2,1,13)
label idcode "Country-patient identifier"
cls
gen s(11) idcode0=substr(idcode2,3,11)
label idcode0 "Patient identifier"
```

Task:

- o Sort the data set so that all specimens for a given patient-month are next to each other, then recode all variable pertaining to isoniazid in a hierarchical manner, so that it becomes easy to assign the value that will be retained to the first specimen for that patient-month. Hierarchy: high-level resistance>low-level resistance>susceptible>no result. Then select to retain only the first specimen per patient-month.***

Solution:

First, we determined the frequency of the number of specimens per patient-month, using a standard approach which you will be using very often as this question arises very frequently:

```
cls
* Identify multiple specimens per patient-month
sort idcode2 mm
gen i specseq=1
if idcode2=idcode2[_n-1] then specseq=specseq[_n-1]+1
* freq specseq
* =>
*      N
* 1    582
* 2    26
* 3     5
* 4     4
* 5     2
* 6     2
* Total 621
* => Up to 6 specimens for 1 patient in 1 month
```

It is not necessary to know in this specific situation that at most 6 specimens are available per patient months, but it will be of key importance to know it for the next step. We show this here for isoniazid phenotypic result at 0.2 mg/L, i.e. the variable H02. Currently, these are the original (spreadsheet-coded) results:

```
NT    490
R     126
S      5
Total 621
```

As we recommended a hierarchical numerical coding, we create a new numeric variable H02N and then “reshuffle the values” hierarchically to the first record within IDCODE2:

```
cls
define h02n #
h02n=0
if h02="S" then h02n=1
if h02="R" then h02n=2
if idcode2[_n+1]=idcode2 and h02n[_n+1]>h02n then h02n=h02n[_n+1]
if idcode2[_n+2]=idcode2 and h02n[_n+2]>h02n then h02n=h02n[_n+2]
if idcode2[_n+3]=idcode2 and h02n[_n+3]>h02n then h02n=h02n[_n+3]
if idcode2[_n+4]=idcode2 and h02n[_n+4]>h02n then h02n=h02n[_n+4]
if idcode2[_n+5]=idcode2 and h02n[_n+5]>h02n then h02n=h02n[_n+5]
drop h02
rename h02n to h02
label h02 "INH result at 0.2 mg/L"
```

Before we drop h02 we browse to see all relevant variables, picking the case with 6 specimens in a given patient-month:

idcode2	mm	specseq	h02	h02n
1-2014-01-011-00	0	1	NT	2
1-2014-01-011-00	0	2	NT	2
1-2014-01-011-00	0	3	R	2
1-2014-01-011-00	0	4	R	2
1-2014-01-011-00	0	5	R	2
1-2014-01-011-00	0	6	NT	0

Note that the importance is that the first line of data (that is the record with SPECSEQ=1) must be correct in that it takes the hierarchically highest among all results from the total of specimens for the given patient-month. The highest value for H02 is R which translates numerically to 2 and this is correctly appearing in the first record of the sequence. It is thereby irrelevant that the last value is 0 (it is logical that it retains the default because there is no other record afterwards with the same IDCODE2) because after completing this approach for each variable (on isoniazid, then other drugs), all that is retained are the records in which SPECSEQ=1:

```
select specseq=1
```

Task:

- o **Recode the various variables for isoniazid drug susceptibility test results (phenotypic and genotypic results). Recode to a single variable for the isoniazid result. Discuss the hierarchy that you wish to assign to this end.**

Solution:

This might be a somewhat arbitrary classification and will surely required expert input. We chose here the following hierarchy (this is specific for isoniazid, but must be defined for each individual drug, and for other drugs this may differ):

Phenotypic result > genotypic result

In the absence of a phenotypic result, the genotypic result counts

If there are genotypically mutations in both the *katG* and *inhA* promoter gene, then the resistance is high-level, whatever the result of phenotypic testing may be. Thus, we coded:

```
define inh #
inh=9
if inhlevel=0 then inh=inhmolsum
if inhlevel=1 then inh=1
if inhlevel=2 then inh=2
if inhlevel=3 then inh=2
if inhlevel=4 then inh=3
if inhmolsum=3 then inh=3
if inha=2 and katg=2 then inh=3 // already done, but make sure
label inh "Resistance to INH"
```

```
labelvalue inh /0="No result"  
labelvalue inh /1="Susceptible"  
labelvalue inh /2="Low-level resistance"  
labelvalue inh /3="High-level resistance"
```

The entire b_ex04_solution.pgm reads as:

```
* Part B, Exercise 4
* Import spreadsheet data and create identifiers
* EpiData course
* Author: Hans L Rieder
* First version: 05 Feb 2018
* Current version: 02 Mar 2018
```

```
*****
```

```
* 1) Read data from spreadsheet and save to EpiData file
cls
close
logclose
```

```
* read /cb
* savedata "b_ex04_task_in.rec" /replace
```

```
*****
```

```
* 2) Read EpiData file, make basic checks and create new
* identifiers:
* idcode : idcode [original, derived]
* idcode0: country-idcode [original extended]
* idcode2: country-idcode-month [original extended]
cls
close
logclose
```

```
read "b_ex04_task_in.rec"
```

```
cls
* Manually inspect for typing errors and correct
if substr(idcode,1,4)="x- 2" then \
  idcode=substr(idcode,1,2)+substr(idcode,4,11)
if substr(idcode,7,1)<>"-" then \
  idcode=substr(idcode,1,6)+"-"+substr(idcode,7,6)
```

```
cls
* Exclude records without valid identifier
* <= they cannot be used
gen i hasid=1
if lower(substr(idcode, 1,3))="xxx" then hasid=0
if lower(substr(idcode,11,3))="xxx" then hasid=0
select hasid=1
drop hasid
```

```
cls
* freq mm /m
define mmn ##
mmn=99
mmn=integer(mm) if mm<>"Unknown"
drop mm
rename mmn to mm
label mm "Month of treatment"
```

```

cls
* Create text month with leading zero
* to allow correct alphabetical sorting
gen s(2) mmtxt=mm
if mm<10 then mmtxt="0"+mm
select mm<>99

cls
* freq country
define countryyn #
if country="A" then countryyn=1
if country="B" then countryyn=2
if country="C" then countryyn=3
if country="D" then countryyn=4
if country="E" then countryyn=5
if country="F" then countryyn=6
if country="G" then countryyn=7
if country="H" then countryyn=8
drop country
rename countryyn to country
label country "Name of country / jurisdiction"
labelvalue country /1="Jurisdiction A"
labelvalue country /2="Jurisdiction B"
labelvalue country /3="Jurisdiction C"
labelvalue country /4="Jurisdiction D"
labelvalue country /5="Jurisdiction E"
labelvalue country /6="Jurisdiction F"
labelvalue country /7="Jurisdiction G"
labelvalue country /8="Jurisdiction H"

cls
* Create derived identifiers
gen s(16) idcoden=country+substr(idcode,2,12)+"-"+mmtxt
drop idcode2 idcode
rename idcoden to idcode2
label idcode2 "Country-patient-month identifier"
cls
gen s(13) idcode=substr(idcode2,1,13)
label idcode "Country-patient identifier"
cls
gen s(11) idcode0=substr(idcode2,3,11)
label idcode0 "Patient identifier"

label spnr "Specimen number"

cls
* Reduce dataset size for faster processing
keep idcode0 idcode idcode2 country mm spnr \
    h02 h10 h50 inhlevel katg inha inhmol inhmolsum

cls
* Identify multiple specimens per patient-month
sort idcode2 mm
gen i specseq=1
if idcode2=idcode2[_n-1] then specseq=specseq[_n-1]+1

```

```

* freq specseq
* =>
*      N
* 1    582
* 2    26
* 3     5
* 4     4
* 5     2
* 6     2
* Total 621
* => Up to 6 specimens for 1 patient in 1 month

```

```

savedata "temp_01.rec" /replace

```

```

*****

```

```

* 3) Recode to priority on first of multiple specimens
* => Isoniazid

```

```

* Hierarchy (code numerically from 0 to highest):
* High level resistance > low level resistance
* Resistant > Susceptible
* Susceptible > No result

```

```

cls
close
logclose

```

```

read "temp_01.rec"

```

```

cls
define h02n #
h02n=0
if h02="S" then h02n=1
if h02="R" then h02n=2
if idcode2[_n+1]=idcode2 and h02n[_n+1]>h02n then h02n=h02n[_n+1]
if idcode2[_n+2]=idcode2 and h02n[_n+2]>h02n then h02n=h02n[_n+2]
if idcode2[_n+3]=idcode2 and h02n[_n+3]>h02n then h02n=h02n[_n+3]
if idcode2[_n+4]=idcode2 and h02n[_n+4]>h02n then h02n=h02n[_n+4]
if idcode2[_n+5]=idcode2 and h02n[_n+5]>h02n then h02n=h02n[_n+5]
drop h02
rename h02n to h02
label h02 "INH result at 0.2 mg/L"

```

```

cls
define h10n #
h10n=0
if h10="S" then h10n=1
if h10="R" then h10n=2
if idcode2[_n+1]=idcode2 and h10n[_n+1]>h10n then h10n=h10n[_n+1]
if idcode2[_n+2]=idcode2 and h10n[_n+2]>h10n then h10n=h10n[_n+2]
if idcode2[_n+3]=idcode2 and h10n[_n+3]>h10n then h10n=h10n[_n+3]
if idcode2[_n+4]=idcode2 and h10n[_n+4]>h10n then h10n=h10n[_n+4]
if idcode2[_n+5]=idcode2 and h10n[_n+5]>h10n then h10n=h10n[_n+5]
drop h10
rename h10n to h10

```

```

label h10 "INH result at 1.0 mg/L"

cls
define h50n #
h50n=0
if h50="S" then h50n=1
if h50="R" then h50n=2
if idcode2[_n+1]=idcode2 and h50n[_n+1]>h50n then h50n=h50n[_n+1]
if idcode2[_n+2]=idcode2 and h50n[_n+2]>h50n then h50n=h50n[_n+2]
if idcode2[_n+3]=idcode2 and h50n[_n+3]>h50n then h50n=h50n[_n+3]
if idcode2[_n+4]=idcode2 and h50n[_n+4]>h50n then h50n=h50n[_n+4]
if idcode2[_n+5]=idcode2 and h50n[_n+5]>h50n then h50n=h50n[_n+5]
drop h50
rename h50n to h50
label h50 "INH result at 5.0 mg/L"

labelvalue h02-h50 /0="No result"
labelvalue h02-h50 /1="Susceptible"
labelvalue h02-h50 /2="Resistant"

cls
define inhleveln #
inhleveln=0
if inhlevel="-" then inhleveln=1 // Susceptible!
if inhlevel="S" then inhleveln=1 // Susceptible!
if inhlevel="H0,2" then inhleveln=2
if inhlevel="H1" then inhleveln=3
if inhlevel="H5" then inhleveln=4
if idcode2[_n+1]=idcode2 and inhleveln[_n+1]>inhleveln then
inhleveln=inhleveln[_n+1]
if idcode2[_n+2]=idcode2 and inhleveln[_n+2]>inhleveln then
inhleveln=inhleveln[_n+2]
if idcode2[_n+3]=idcode2 and inhleveln[_n+3]>inhleveln then
inhleveln=inhleveln[_n+3]
if idcode2[_n+4]=idcode2 and inhleveln[_n+4]>inhleveln then
inhleveln=inhleveln[_n+4]
if idcode2[_n+5]=idcode2 and inhleveln[_n+5]>inhleveln then
inhleveln=inhleveln[_n+5]
drop inhlevel
rename inhleveln to inhlevel
label inhlevel "INH result summary level"
labelvalue inhlevel /0="No result"
labelvalue inhlevel /1="Susceptible"
labelvalue inhlevel /2="Resistant at 0.2 mg/L"
labelvalue inhlevel /3="Resistant at 1.0 mg/L"
labelvalue inhlevel /4="Resistant at 5.0 mg/L"

cls
define katgn #
katgn=0
if katg="-" then katgn=1
if lower(substr(katg,1,4))="wild" then katgn=1
if lower(substr(katg,1,3))="del" then katgn=2
if (substr(katg,1,3))="315" then katgn=2
if lower(substr(katg,1,3))="mix" then katgn=2
if lower(substr(katg,1,3))="mut" then katgn=2

```

```

if idcode2[_n+1]=idcode2 and katgn[_n+1]>katgn then katgn=katgn[_n+1]
if idcode2[_n+2]=idcode2 and katgn[_n+2]>katgn then katgn=katgn[_n+2]
if idcode2[_n+3]=idcode2 and katgn[_n+3]>katgn then katgn=katgn[_n+3]
if idcode2[_n+4]=idcode2 and katgn[_n+4]>katgn then katgn=katgn[_n+4]
if idcode2[_n+5]=idcode2 and katgn[_n+5]>katgn then katgn=katgn[_n+5]
drop katg
rename katgn to katg
label katg "INH katG result"
labelvalue katg /0="No result"
labelvalue katg /1="No mutation"
labelvalue katg /2="Mutation"

cls
define inhan #
inhan=0
if lower(substr(inha,11,4))="wild" then inhan=1
if lower(substr(inha,15,4))="wild" then inhan=1
if lower(substr(inha, 1,1))="c" then inhan=2
if lower(substr(inha, 1,1))="g" then inhan=2
if lower(substr(inha, 1,3))="del" then inhan=2
if lower(substr(inha, 1,3))="mut" then inhan=2
if lower(substr(inha,11,3))="mut" then inhan=2
if idcode2[_n+1]=idcode2 and inhan[_n+1]>inhan then inhan=inhan[_n+1]
if idcode2[_n+2]=idcode2 and inhan[_n+2]>inhan then inhan=inhan[_n+2]
if idcode2[_n+3]=idcode2 and inhan[_n+3]>inhan then inhan=inhan[_n+3]
if idcode2[_n+4]=idcode2 and inhan[_n+4]>inhan then inhan=inhan[_n+4]
if idcode2[_n+5]=idcode2 and inhan[_n+5]>inhan then inhan=inhan[_n+5]
drop inha
rename inhan to inha
label inha "INH inha result"
labelvalue inha /0="No result"
labelvalue inha /1="No mutation"
labelvalue inha /2="Mutation"

cls
define inhmoln #
inhmoln=0
if lower(substr(inhmol,1,1))="s" then inhmoln=1
if lower(substr(inhmol,1,1))="s?" then inhmoln=0
if lower(substr(inhmol,1,2))="rb" then inhmoln=2
if lower(substr(inhmol,1,2))="rh" then inhmoln=3
if idcode2[_n+1]=idcode2 and inhmoln[_n+1]>inhmoln then
inhmoln=inhmoln[_n+1]
if idcode2[_n+2]=idcode2 and inhmoln[_n+2]>inhmoln then
inhmoln=inhmoln[_n+2]
if idcode2[_n+3]=idcode2 and inhmoln[_n+3]>inhmoln then
inhmoln=inhmoln[_n+3]
if idcode2[_n+4]=idcode2 and inhmoln[_n+4]>inhmoln then
inhmoln=inhmoln[_n+4]
if idcode2[_n+5]=idcode2 and inhmoln[_n+5]>inhmoln then
inhmoln=inhmoln[_n+5]
drop inhmol
rename inhmoln to inhmol
label inhmol "INH inhmol result"
labelvalue inhmol /0="No result"
labelvalue inhmol /1="Susceptible"

```

```

labelvalue inhmol /2="Low level resistance"
labelvalue inhmol /3="High level resistance"

cls
define inhmolsumn #
inhmolsumn=0
if lower(substr(inhmolsum,1,1))="-" then inhmolsumn=1
if lower(substr(inhmolsum,1,2))="hb" then inhmolsumn=2
if lower(substr(inhmolsum,1,2))="hh" then inhmolsumn=3
if idcode2[_n+1]=idcode2 and inhmolsumn[_n+1]>inhmolsumn then
inhmolsumn=inhmolsumn[_n+1]
if idcode2[_n+2]=idcode2 and inhmolsumn[_n+2]>inhmolsumn then
inhmolsumn=inhmolsumn[_n+2]
if idcode2[_n+3]=idcode2 and inhmolsumn[_n+3]>inhmolsumn then
inhmolsumn=inhmolsumn[_n+3]
if idcode2[_n+4]=idcode2 and inhmolsumn[_n+4]>inhmolsumn then
inhmolsumn=inhmolsumn[_n+4]
if idcode2[_n+5]=idcode2 and inhmolsumn[_n+5]>inhmolsumn then
inhmolsumn=inhmolsumn[_n+5]
drop inhmolsum
rename inhmolsumn to inhmolsum
label inhmolsum "INH inhmolsum result"
labelvalue inhmolsum /0="No result"
labelvalue inhmolsum /1="Susceptible"
labelvalue inhmolsum /2="Low level resistance"
labelvalue inhmolsum /3="High level resistance"

savedata "temp_02.rec" /replace

*****
* 4) Recode to priority on first of multiple specimens
* => Next drug

cls
close
logclose
read "temp_02.rec"

* Do this for each drug
* At the end of the process, the hierarchically
* highest value will be in the sequentially
* first SPNR if there are multiple SPNRs
* => keeping only the first SPNR suffices:

select specseq=1

* Create a single result for isoniazid

define inh #
inh=9
if inhlevel=0 then inh=inhmolsum
if inhlevel=1 then inh=1
if inhlevel=2 then inh=2
if inhlevel=3 then inh=2
if inhlevel=4 then inh=3

```

```
if inhmolsum=3 then inh=3
if inha=2 and katg=2 then inh=3 // already done, but make sure
label inh "Resistance to INH"
labelvalue inh /0="No result"
labelvalue inh /1="Susceptible"
labelvalue inh /2="Low-level resistance"
labelvalue inh /3="High-level resistance"
```

```
savedata "temp_03.rec" /replace
```

```
*****
```

```
* 5) Any other recoding
```

```
cls
close
logclose
read "temp_03.rec"
```

```
* xxx
```

```
drop spnr specseq idcode0 idcode2
drop h02 h10 h50 inhlevel
drop katg inha inhmol inhmolsum
```

```
sort country idcode mm
```

```
savedata "b_ex04_task_out.rec" /replace
```

```
*****
```

```
* Clean up and erase temporary session files
```

```
set echo=off
close
define yesno # global
yesno=?Delete all temporary files: 1=yes 0=no?
imif yesno=1 then
cls
type "Be patient ... you will be alerted upon completion" /h2
erase "temp_01.chk"
erase "temp_01.rec"
erase "temp_02.chk"
erase "temp_02.rec"
erase "temp_03.chk"
erase "temp_03.rec"
cls
type "All temporary files erased" /h2
else
type "All temporary files retained" /h2
endif
set echo=on
```

```
*****
```

```
cls
```

```
close  
read "b_ex04_task_out.rec"
```

Part C. Operations Research

Part C: Operations research

Exercise 1: Creating a working dataset

Exercise 2: Variability in serial smears

Exercise 3: Incremental yield from serial smears

Exercise 4: Confirmatory results in serial smears

Introductory note

In this Part C three operationally relevant research questions will be answered:

- Does the dataset tell us something about how diligent the work was performed in the tuberculosis microscopy laboratory?
- Is the third serial smear examination associated with an excessive amount of work for little gain?
- Is it necessary to confirm a positive smear result?

These and related questions were asked by graduates from The Union's operations research courses in fulfillment of the field component of the course. The data were collected in Moldova (Dr Dumitru Laticevschi, fifth course, Paris, 2003), Mongolia (Dr Nymadawaa Naranbat, seventh course, Paris, 2004), Uganda (Dr Achilles Katamba, fifth course, Paris, 2003), and Zimbabwe (Dr Biggie Mabaera, seventh course, Paris, 2004). Six publications have resulted from this study:

Mabaera B, Naranbat N, Dhliwayo P, Rieder H L. Efficiency of serial smear examinations in excluding sputum smear-positive tuberculosis. *Int J Tuberc Lung Dis* 2006;10:1030-5.

Katamba A, Laticevschi D, Rieder H L. Efficiency of a third serial sputum smear examination in the diagnosis of tuberculosis in Moldova and Uganda. *Int J Tuberc Lung Dis* 2007;11:659-64.

Mabaera B, Lauritsen J M, Katamba A, Laticevschi D, Naranbat N, Rieder H L. Sputum smear-positive tuberculosis: empiric evidence challenges the need for confirmatory smears. *Int J Tuberc Lung Dis* 2007;11:959-64.

Mabaera B, Lauritsen J M, Katamba A, Laticevschi D, Naranbat N, Rieder H L. Making pragmatic sense of data in the tuberculosis laboratory register. *Int J Tuberc Lung Dis* 2008;12:294-300.

Mabaera B, Naranbat N, Katamba A, Laticevschi D, Lauritsen J M, Rieder H L. Seasonal variation among tuberculosis suspects in four countries. *International Health* 2009;1:53-60.

Rieder H L, Lauritsen J M, Naranbat N, Katamba A, Laticevschi D, Mabaera B. Quantitative differences in sputum smear microscopy results for acid-fast bacilli by age and sex in four countries. *Int J Tuberc Lung Dis* 2009;13:1393-8.

With permission of the investigators, the datasets have been made publicly accessible for use in this course exactly as they have been collected.

Exercise 1: Creating a working dataset

At the end of this exercise you should be able to:

- a. Combine different datasets into one combined dataset
- b. Recode 'text variables' to 'numeric variables'
- c. Remove 'undesirable' records from a dataset
- d. Correct obvious gross errors from the datasets
- e. Create a 'cleaned' final working dataset from available datasets

Moldova and Uganda worked together using the same data entry forms. You obtained MOL_25.ZIP and UGA_30.ZIP. These two files contain respectively the data files obtained from the 25 laboratories in Moldova and the data files obtained from the 30 laboratories in Uganda. In addition, each of the zip files contains the base pair of QES and CHK files (which are identical for both countries, except for the field name for the laboratory).

Mongolia and Zimbabwe worked together using the same data entry forms. You obtained MON_31.ZIP and ZIM_23.ZIP. These two files contain respectively the data files obtained from the 31 laboratories in Mongolia and the data files obtained from the 23 laboratories in Zimbabwe. In addition, each of the zip files contains the base pair of QES and CHK files (which are identical for both countries).

The two pairs of countries collected exactly the same information from the laboratory register, but their data collection forms (the QES files, and thus REC files) and CHK files had small differences. You can find these by inspecting the respective files. However, as you come in here as an outsider, we summarize these in the following table, and also give you the field names that the final data set combining all files should have.

Field label	Field name Moldova / Uganda	Field name Mongolia / Zimbabwe	Final Field name
Study country	--	--	country
Laboratory code	labcode / labno	laboratory	laboratory
Laboratory serial number	serno	serno	--
Registration date	labdate	regdate	regdate
Year of registration	--	--	regyear
Created unique identifier	unique	id	--
Sex of examinee	sex	sex	sex
Age (in years) of examinee	age	age	age
Reason for examination	reason	reason	reason
Result of first examination	res1	res1	result1
Result of second examination	res2	res2	result2
Result of third examination	res3	res3	result3

Omissions and commissions

In contrast to what you learned in Part A, the data entry form used only field names but had no field labels.

In both studies SEX and REASON were coded as text fields rather than numerically (but label blocks were used). The fields RES1, RES2, and RES3 also differed slightly: a value of 4.0 did not exist in Moldova / Uganda, but denoted “Positive, not quantified” in Mongolia / Zimbabwe, while “Positive, not quantified” was coded as 8.0 in the latter but did not exist in the former. “Scanty, not quantified” was coded as 5.0 in Mongolia / Zimbabwe, but was forgotten as a possible value in Moldova / Uganda.

You could obtain the information from the CHK files, but the summary of the coding for the fields of relevance with the differences is as follows:

Field name	Field value Moldova / Uganda	Field value Mongolia / Zimbabwe	Value label
Sex	F M 9	F M 9	Female Male Unknown sex
Reason	D F 9 -- -- -- -- -- -- -- --	D F 9 1 2 3 4 5 6 7 8	Diagnosis Follow-up, month not stated Reason not stated Follow-up at 1 month Follow-up at 2 months Follow-up at 3 months Follow-up at 4 months Follow-up at 5 months Follow-up at 6 months Follow-up at 7 months Follow-up at 8 months or later
res1 (also res2, res3)	0.0 0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8 0.9 1.0 2.0 3.0 -- -- 8.0 9.0	0.0 0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8 0.9 1.0 2.0 3.0 4.0 5.0 -- 9.0	Negative Scanty, 1 AFB / 100 fields Scanty, 2 AFB / 100 fields Scanty, 3 AFB / 100 fields Scanty, 4 AFB / 100 fields Scanty, 5 AFB / 100 fields Scanty, 6 AFB / 100 fields Scanty, 7 AFB / 100 fields Scanty, 8 AFB / 100 fields Scanty, 9 AFB / 100 fields 1+ positive 2+ positive 3+ positive Positive, not quantified Scanty, not quantified Positive, not quantified No result recorded

Tasks:

- o Create a combined dataset C_EX01_COMBINE.REC from all 107 files with a program C_EX01_COMBINE.PGM.*

Notes to the first task:

From the dataset from Moldova, drop the data for the laboratory “BND” (containing data from only 1 week) and remove one empty record.

From the dataset from Mongolia, remove the empty records

In Zimbabwe, one record has no laboratory value, but it has an ID (this is most likely attributable to some manipulation with the mouse after ID creation). You can retain this record by giving the laboratory the correct code that we know from the ID.

If you have removed all empty records (plus the one laboratory from Moldova) and you make a frequency of COUNTRY you should get:

country		
	N	%
MOL	17865	13.7
MON	22588	17.3
UGA	55114	42.3
ZIM	34744	26.7
Total	130311	100.0

- o Create a “cleaned” final working dataset C_EX01.REC with a program C_EX01_2_RESTRUCTURE.PGM which excludes non-sensically coded result sequences, and with all fields codes numerically (including COUNTRY and LABORATORY).*

Notes to the second task:

For the numeric coding of the COUNTRY follow the alphabet: 1 for Moldova, 2 for Mongolia, ..., 4 for Zimbabwe.

For the numeric coding of the laboratories, make a frequency for each country, and then code numerically following the country notation:

Moldova laboratories:

```
if laboratory="ANR" then lab0=101
if laboratory="BLM" then lab0=102
if laboratory="BRL" then lab0=103
if laboratory="BSR" then lab0=104
if laboratory="CCE" then lab0=105
...etc
```

Mongolia laboratories:

```
if laboratory="AR_B" then lab0=201
if laboratory="BG_B" then lab0=202
if laboratory="BN_B" then lab0=203
...etc
```

Uganda laboratories:

```
if trim(laboratory)="1" then lab0=301
if trim(laboratory)="2" then lab0=302
if trim(laboratory)="3" then lab0=303
...etc
```

Zimbabwe laboratories:

```
if laboratory="BY_A" then lab0=401
if laboratory="MC_A" then lab0=402
if laboratory="MC_B" then lab0=403
if laboratory="MC_C" then lab0=404
...etc
```

We also propose to correct some obvious gross errors (which are obvious from the sequence in recording what they should have been) in the registration date. In order to get a common ground, we point these out here and provide the program file commands for these (note that we made a date variable just for this manipulation here):

```

define regyear0 ####
regyear0=year(regdate)
define regyear ####
regyear=regyear0
* correct errors in year of recording
if regyear0=1990 and laboratory=301 then regyear=1999
if regyear0=1990 and laboratory=306 then regyear=1999
if regyear0=1990 and laboratory=319 then regyear=1999
if regyear0=1990 and laboratory=320 then regyear=2000
if regyear0=1990 and laboratory=410 then regyear=2002

if regyear0=2000 and laboratory=408 then regyear=2002
if regyear0=2000 and laboratory=416 then regyear=2002
if regyear0=2000 and laboratory=419 then regyear=2002

if regyear0=2004 and laboratory=211 then regyear=2003
if regyear0=2004 and laboratory=223 then regyear=2003
if regyear0=2004 and laboratory=401 then regyear=2002
if regyear0=2004 and laboratory=408 then regyear=2002
if regyear0=2004 and laboratory=412 then regyear=2003
if regyear0=2004 and laboratory=413 then regyear=2002

if regyear0=2005 and laboratory=207 then regyear=2003
if regyear0=2033 and laboratory=207 then regyear=2003

```

If you have cleaned the dataset and you make a table of COUNTRY by REGYEAR you should get:

	Study country				
Year of registration	Moldova	Mongolia	Uganda	Zimbabwe	Total
1999	0	0	17308	0	17308
2000	0	0	18655	0	18655
2001	0	0	18087	1213	19300
2002	0	149	0	29307	29456
2003	17725	22406	0	3958	44089
Total	17725	22555	54050	34478	128808

Note the following on the CHK and QES files:

If you start with a REC file that is accompanied by its CHK file and then create new variables with Field values and Value labels using the LABELVALUE, EpiData Analysis takes the original CHK file and appends it with the new Field values and their Value labels when you create a new REC file. You can also define a Field label (command LABEL newvar "X").

Solution to Exercise 1: Creating a working dataset

Key Learning Points

- You should clean the final dataset so as to remove 'undesirable records' and correct obvious gross errors. Records removed from the dataset should be documented as well as the reason.
- The 'cleaned' working dataset will then be used for data analysis.

Task:

Create a combined dataset *C_EX01_COMBINE.REC* from all 107 files with a program *C_EX01_1_COMBINE.PGM*.

Solution

This is the dataset by country and year that should result from your program:

	Country				
Year of registration	Moldova	Mongolia	Uganda	Zimbabwe	Total
1999	0	0	17308	0	17308
2000	0	0	18655	0	18655
2001	0	0	18087	1213	19300
2002	0	149	0	29307	29456
2003	17725	22406	0	3958	44089
Total	17725	22555	54050	34478	128808

A possible solution is the following *C_EX01_1_COMBINE.PGM*:

```
* Part C, Exercise 1, first program

* Produce combined dataset for
* Moldova, Mongolia, Uganda, Zimbabwe
* and remove empty records

* Data courtesy:
* Moldova: Dumitru Latticevski, OR Paris 2003
* Mongolia: Nymadawa Naranbat, OR Paris 2004
* Uganda: Achilles Katamba, OR Paris 2003
* Zimbabwe: Biggie Mabaera, OR Paris 2004

* Written by: Hans L Rieder
* First version: 17 Jan 2010
* Last revision: 03 Jun 2013

cls
close
logclose

*****
* Combine original final Moldova datasets
* Create mol_1.rec

cls
logclose
```

```

close

read "mol_01.rec"
append /file="mol_02.rec"
append /file="mol_03.rec"
append /file="mol_04.rec"
append /file="mol_05.rec"
append /file="mol_06.rec"
append /file="mol_07.rec"
append /file="mol_08.rec"
append /file="mol_09.rec"
append /file="mol_10.rec"
append /file="mol_11.rec"
append /file="mol_12.rec"
append /file="mol_13.rec"
append /file="mol_14.rec"
append /file="mol_15.rec"
append /file="mol_16.rec"
append /file="mol_17.rec"
append /file="mol_18.rec"
append /file="mol_19.rec"
append /file="mol_20.rec"
append /file="mol_21.rec"
append /file="mol_22.rec"
append /file="mol_23.rec"
append /file="mol_24.rec"
append /file="mol_25.rec"
savedata "mol_0.rec" /replace

cls
logclose
close
read "mol_0.rec"
define country #
country=1
label country "Study country"
* Exclude laboratory BND with 13 records
* collected during 1 week only
select labcode<>"BND"
* remove 1 empty record
select serno<>.
var drop unique serno
savedata "mol_1.rec" /replace

close
*****
* Combine original final Mongolia datasets
* Create mon_1.rec

cls
logclose
close

read "mon_01.rec"
append /file="mon_02.rec"
append /file="mon_03.rec"
append /file="mon_04.rec"
append /file="mon_05.rec"
append /file="mon_06.rec"
append /file="mon_07.rec"
append /file="mon_08.rec"
* Note: 1 record in MON_09.REC had a corrupted
* date which prevented appending. This record
* was manually changed in EpiData from "203" to "2003"
append /file="mon_09.rec"
append /file="mon_10.rec"
append /file="mon_11.rec"
append /file="mon_12.rec"

```

```

append /file="mon_13.rec"
append /file="mon_14.rec"
append /file="mon_15.rec"
append /file="mon_16.rec"
append /file="mon_17.rec"
append /file="mon_18.rec"
append /file="mon_19.rec"
append /file="mon_20.rec"
append /file="mon_21.rec"
append /file="mon_22.rec"
append /file="mon_23.rec"
append /file="mon_24.rec"
append /file="mon_25.rec"
append /file="mon_26.rec"
append /file="mon_27.rec"
append /file="mon_28.rec"
append /file="mon_29.rec"
append /file="mon_30.rec"
append /file="mon_31.rec"
savedata "mon_0.rec" /replace
close

read "mon_0.rec"
define country #
country=2
label country "Study country"
* The following removes 10 empty records
select serno<>.
savedata "mon_1.rec" /replace

close
*****
* Combine original final Uganda datasets
* Create uga_1.rec

cls
close
logclose

read "uga_01.rec"
append /file="uga_02.rec"
append /file="uga_03.rec"
append /file="uga_04.rec"
append /file="uga_05.rec"
append /file="uga_06.rec"
append /file="uga_07.rec"
append /file="uga_08.rec"
append /file="uga_09.rec"
append /file="uga_10.rec"
append /file="uga_11.rec"
append /file="uga_12.rec"
append /file="uga_13.rec"
append /file="uga_14.rec"
append /file="uga_15.rec"
append /file="uga_16.rec"
append /file="uga_17.rec"
append /file="uga_18.rec"
append /file="uga_19.rec"
append /file="uga_20.rec"
append /file="uga_21.rec"
append /file="uga_22.rec"
append /file="uga_23.rec"
append /file="uga_24.rec"
append /file="uga_25.rec"
append /file="uga_26.rec"
append /file="uga_27.rec"
append /file="uga_28.rec"
append /file="uga_29.rec"

```

```

append /file="uga_30.rec"
savedata "uga_0.rec" /replace

cls
logclose
close
read "uga_0.rec"
define country #
let country=3
label country "Study country"
define labcode _____
let labcode=labno
var drop labno serno
savedata "uga_1.rec" /replace

close
*****
* Combine original final Zimbabwe datasets
* Create zim_1.rec

cls
logclose
close

read "zim_01.rec"
append /file="zim_02.rec"
append /file="zim_03.rec"
append /file="zim_04.rec"
append /file="zim_05.rec"
append /file="zim_06.rec"
append /file="zim_07.rec"
append /file="zim_08.rec"
append /file="zim_09.rec"
append /file="zim_10.rec"
append /file="zim_11.rec"
append /file="zim_12.rec"
append /file="zim_13.rec"
append /file="zim_14.rec"
append /file="zim_15.rec"
append /file="zim_16.rec"
append /file="zim_17.rec"
append /file="zim_18.rec"
append /file="zim_19.rec"
append /file="zim_20.rec"
append /file="zim_21.rec"
append /file="zim_22.rec"
append /file="zim_23.rec"
savedata "zim_0.rec" /replace
close

read "zim_0.rec"
define country #
country=4
label country "Study country"
* Note: if you freq on laboratory then
* you have a lab without a code. When you sort
* on laboratory, then you see it on the top with
* 4 dots. Curiously, an ID was created nevertheless
* it is laboratory "MW_L"
* Thus, from the following recoding, we get
* an appropriate laboratory and can retain the record
if ID="MW_L-2002-554" then laboratory="MW_L"
* Laboratory coded as "G867" is actually "ML_L"
* Thus, from the following recoding, we get
* an appropriate laboratory and can retain the record
if laboratory="G867" then laboratory="ML_L"
savedata "zim_1.rec" /replace

```

```

close
*****
* Combine 4 country sets

cls
close
logclose

cls
read "mon_1.rec"
drop serno id result pattern
savedata "montemp.rec" /replace
close

read "mol_1.rec"
define laboratory ____
laboratory=labcode
define regdate <dd/mm/yyyy>
regdate=labdate
drop labcode labdate
savedata "moltemp.rec" /replace
close

read "uga_1.rec"
define laboratory ____
laboratory=labcode
define regdate <dd/mm/yyyy>
regdate=labdate
drop labcode labdate
savedata "ugatemp.rec" /replace
close

cls
read "zim_1.rec"
drop serno id result pattern
savedata "zimtemp.rec" /replace
close

read "moltemp.rec"
append /file="montemp.rec"
append /file="ugatemp.rec"
append /file="zimtemp.rec"
labelvalue country /1="Moldova"
labelvalue country /2="Mongolia"
labelvalue country /3="Uganda"
labelvalue country /4="Zimbabwe"
savedata "c_ex01_combine.rec" /replace
close

read "c_ex01_combine.rec"
freq country

*****
* Clean up and erase temporary session files

set echo=off
close
define yesno # global
yesno=?Delete all temporary files: 1=yes 0=no?
imif yesno=1 then
  cls
  type "Be patient, you will be alerted to completion" /h2
  erase "mon_0.chk "
  erase "mon_0.rec "
  erase "mon_1.chk "
  erase "mon_1.rec "
  erase "mol_0.chk "
  erase "mol_0.rec "

```

```

erase "mol_1.chk "
erase "mol_1.rec "
erase "uga_0.chk "
erase "uga_0.rec "
erase "uga_1.chk "
erase "uga_1.rec "
erase "zim_0.chk "
erase "zim_0.rec "
erase "zim_1.chk "
erase "zim_1.rec "
erase "moltemp.chk"
erase "moltemp.rec"
erase "montemp.chk"
erase "montemp.rec"
erase "ugatemp.chk"
erase "ugatemp.rec"
erase "zimtemp.chk"
erase "zimtemp.rec"
cls
type "All temporary files erased" /h2
else
type "All temporary files retained" /h2
endif
set echo=on

```

Task:

- o Create a “cleaned” final working dataset C_EX01.REC with a program C_EX01_2_RESTRUCTURE.PGM which excludes non-sensically coded result sequences, and with all fields codes numerically (including COUNTRY and LABORATORY).***

Solution

A possible solution is the following C_EX01_2_RESTRUCTURE.PGM:

```

* Part C, Exercise 1, second program

* Produce cleaned dataset for
* Moldova, Mongolia, Uganda, Zimbabwe
* Removing results with nonsensical sequence

* Data courtesy:
* Moldova: Dumitru Laticevschi, OR Paris 2003
* Mongolia: Nymadawa Naranbat, OR Paris 2004
* Uganda: Achilles Katamba, OR Paris 2003
* Zimbabwe: Biggie Mabaera, OR Paris 2004

* Written by: Hans L Rieder
* First version: 17 Jan 2010
* Last revision: 03 Jun 2013

cls
close
logclose

read "c_ex01_combine.rec"

define res1b _
if res1=0 then res1b="N"
if res1>0 and res1<9 then res1b="P"
if res1=9 then res1b="9"

```

```

        define res2b _
            if res2=0 then res2b="N"
if res2>0 and res2<9 then res2b="P"
            if res2=9 then res2b="9"

        define res3b _
            if res3=0 then res3b="N"
if res3>0 and res3<9 then res3b="P"
            if res3=9 then res3b="9"

define sequence _____
label sequence "Sequence of serial results"
let sequence=res1b+"-"+res2b+"-"+res3b

* The following removes records with an impossible
* sequence of results
cls
select sequence<>"9-9-9"
select sequence<>"9-9-N"
select sequence<>"9-9-P"
select sequence<>"9-N-9"
select sequence<>"9-N-N"
select sequence<>"9-P-P"
select sequence<>"N-9-N"
select sequence<>"N-9-P"
select sequence<>"P-9-P"
select sequence<>"9-P-9"
select sequence<>"9-P-N"
select sequence<>"P-9-N"
select sequence<>"9-N-P"

cls
define result1 #.#
label result1 "Result of 1st examination"
let result1=res1
if res1=8.0 then result1=4.0

define result2 #.#
label result2 "Result of 2nd examination"
let result2=res2
if res2=8.0 then result2=4.0

define result3 #.#
label result3 "Result of 3rd examination"
let result3=res3
if res3=8.0 then result3=4.0

cls
define reason0 ##
if reason="D" then reason0=00
if reason="F" then reason0=10
if reason="9" then reason0=99
if reason="1" then reason0=01
if reason="2" then reason0=02
if reason="3" then reason0=03
if reason="4" then reason0=04
if reason="5" then reason0=05
if reason="6" then reason0=06
if reason="7" then reason0=07
if reason="8" then reason0=08

cls
define sex0 #
if sex="F" then sex0=1
if sex="M" then sex0=2
if sex="9" then sex0=9

```

```

cls
define lab0 ###
* Moldova laboratories
if laboratory="ANR" then lab0=101
if laboratory="BLM" then lab0=102
if laboratory="BRL" then lab0=103
if laboratory="BSR" then lab0=104
if laboratory="CCE" then lab0=105
if laboratory="CDR" then lab0=106
if laboratory="CHR" then lab0=107
if laboratory="CLR" then lab0=108
if laboratory="CMN" then lab0=109
if laboratory="CMR" then lab0=110
if laboratory="CNR" then lab0=111
if laboratory="CRR" then lab0=112
if laboratory="CTR" then lab0=113
if laboratory="DNR" then lab0=114
if laboratory="EDR" then lab0=115
if laboratory="FLR" then lab0=116
if laboratory="FRR" then lab0=117
if laboratory="HNR" then lab0=118
if laboratory="LVR" then lab0=119
if laboratory="PRB" then lab0=120
if laboratory="RZR" then lab0=121
if laboratory="SRR" then lab0=122
if laboratory="STR" then lab0=123
if laboratory="VLR" then lab0=124

```

```

cls
* Mongolia laboratories
if laboratory="AR_B" then lab0=201
if laboratory="BG_B" then lab0=202
if laboratory="BN_B" then lab0=203
if laboratory="BU_B" then lab0=204
if laboratory="BZ_B" then lab0=205
if laboratory="CH_B" then lab0=206
if laboratory="DA_B" then lab0=207
if laboratory="DD_B" then lab0=208
if laboratory="DG_B" then lab0=209
if laboratory="DU_B" then lab0=210
if laboratory="GA_B" then lab0=211
if laboratory="GS_B" then lab0=212
if laboratory="KE_B" then lab0=213
if laboratory="KH_B" then lab0=214
if laboratory="KO_B" then lab0=215
if laboratory="KU_B" then lab0=216
if laboratory="NA_B" then lab0=217
if laboratory="OR_B" then lab0=218
if laboratory="PR_B" then lab0=219
if laboratory="RE_B" then lab0=220
if laboratory="SB_B" then lab0=221
if laboratory="SK_B" then lab0=222
if laboratory="SU_B" then lab0=223
if laboratory="TU_B" then lab0=224
if laboratory="UM_B" then lab0=225
if laboratory="US_B" then lab0=226
if laboratory="UV_B" then lab0=227
if laboratory="ZA_B" then lab0=228
if laboratory="SE_B" then lab0=229
if laboratory="BK_B" then lab0=230
if laboratory="B-UB" then lab0=231

```

```

cls
* Uganda laboratories
if trim(laboratory)="1" then lab0=301
if trim(laboratory)="2" then lab0=302
if trim(laboratory)="3" then lab0=303
if trim(laboratory)="4" then lab0=304

```

```

if trim(laboratory)="5" then lab0=305
if trim(laboratory)="6" then lab0=306
if trim(laboratory)="7" then lab0=307
if trim(laboratory)="8" then lab0=308
if trim(laboratory)="9" then lab0=309
if trim(laboratory)="10" then lab0=310
if trim(laboratory)="11" then lab0=311
if trim(laboratory)="12" then lab0=312
if trim(laboratory)="13" then lab0=313
if trim(laboratory)="14" then lab0=314
if trim(laboratory)="15" then lab0=315
if trim(laboratory)="16" then lab0=316
if trim(laboratory)="17" then lab0=317
if trim(laboratory)="18" then lab0=318
if trim(laboratory)="19" then lab0=319
if trim(laboratory)="20" then lab0=320
if trim(laboratory)="21" then lab0=321
if trim(laboratory)="22" then lab0=322
if trim(laboratory)="23" then lab0=323
if trim(laboratory)="24" then lab0=324
if trim(laboratory)="25" then lab0=325
if trim(laboratory)="26" then lab0=326
if trim(laboratory)="27" then lab0=327
if trim(laboratory)="28" then lab0=328
if trim(laboratory)="29" then lab0=329
if trim(laboratory)="30" then lab0=330

```

```

cls
* Zimbabwe laboratories
if laboratory="BY_A" then lab0=401
if laboratory="MC_A" then lab0=402
if laboratory="MC_B" then lab0=403
if laboratory="MC_C" then lab0=404
if laboratory="MC_G" then lab0=405
if laboratory="MC_I" then lab0=406
if laboratory="MC_J" then lab0=407
if laboratory="MD_G" then lab0=408
if laboratory="ME_A" then lab0=409
if laboratory="ME_C" then lab0=410
if laboratory="ME_L" then lab0=411
if laboratory="ME_O" then lab0=412
if laboratory="ML_E" then lab0=413
if laboratory="ML_G" then lab0=414
if laboratory="ML_I" then lab0=415
if laboratory="ML_L" then lab0=416
if laboratory="MN_G" then lab0=417
if laboratory="MV_A" then lab0=418
if laboratory="MV_C" then lab0=419
if laboratory="MV_E" then lab0=420
if laboratory="MW_B" then lab0=421
if laboratory="MW_E" then lab0=422
if laboratory="MW_L" then lab0=423

```

```

drop sequence
drop res1 res2 res3
drop reason
drop sex
drop laboratory

```

```

rename reason0 to reason
rename sex0 to sex
rename lab0 to laboratory

```

```

savedata "temp0.rec" /replace

```

```

*****
cls
close

```

```

read "temp0.rec"

define regyear0 ####
regyear0=year(regdate)

define regyear ####
regyear=regyear0

* correct errors in year of recording
if regyear0=1990 and laboratory=301 then regyear=1999
if regyear0=1990 and laboratory=306 then regyear=1999
if regyear0=1990 and laboratory=319 then regyear=1999
if regyear0=1990 and laboratory=320 then regyear=2000
if regyear0=1990 and laboratory=410 then regyear=2002

if regyear0=2000 and laboratory=408 then regyear=2002
if regyear0=2000 and laboratory=416 then regyear=2002
if regyear0=2000 and laboratory=419 then regyear=2002

if regyear0=2004 and laboratory=211 then regyear=2003
if regyear0=2004 and laboratory=223 then regyear=2003
if regyear0=2004 and laboratory=401 then regyear=2002
if regyear0=2004 and laboratory=408 then regyear=2002
if regyear0=2004 and laboratory=412 then regyear=2003
if regyear0=2004 and laboratory=413 then regyear=2002

if regyear0=2005 and laboratory=207 then regyear=2003
if regyear0=2033 and laboratory=207 then regyear=2003

label regyear "Year of registration"
labelvalue sex /1="Female" /2="Male" /9="Missing"
label sex "Sex of examinee"
labelvalue reason /0="Diagnosis"
labelvalue reason /1="Follow-up at 1 month"
labelvalue reason /2="Follow-up at 2 months"
labelvalue reason /3="Follow-up at 3 months"
labelvalue reason /4="Follow-up at 4 months"
labelvalue reason /5="Follow-up at 5 months"
labelvalue reason /6="Follow-up at 6 months"
labelvalue reason /7="Follow-up at 7 months"
labelvalue reason /8="Follow-up at 8 months or later"
labelvalue reason /10="Follow-up, month not stated"
labelvalue reason /99="Reason not stated"
label reason "Reason for examination"

labelvalue result1 /0.0="Negative"
labelvalue result1 /4.0="Positive, not quantified"
labelvalue result1 /5.0="Scanty, not quantified"
labelvalue result1 /0.1="Scanty, 1 AFB per 100 fields"
labelvalue result1 /0.2="Scanty, 2 AFB per 100 fields"
labelvalue result1 /0.3="Scanty, 3 AFB per 100 fields"
labelvalue result1 /0.4="Scanty, 4 AFB per 100 fields"
labelvalue result1 /0.5="Scanty, 5 AFB per 100 fields"
labelvalue result1 /0.6="Scanty, 6 AFB per 100 fields"
labelvalue result1 /0.7="Scanty, 7 AFB per 100 fields"
labelvalue result1 /0.8="Scanty, 8 AFB per 100 fields"
labelvalue result1 /0.9="Scanty, 9 AFB per 100 fields"
labelvalue result1 /1.0="1+ positive"
labelvalue result1 /2.0="2+ positive"
labelvalue result1 /3.0="3+ positive"
labelvalue result1 /9.0="No result recorded"
label result1 "Result of 1st examination"

labelvalue result2 /0.0="Negative"
labelvalue result2 /4.0="Positive, not quantified"
labelvalue result2 /5.0="Scanty, not quantified"
labelvalue result2 /0.1="Scanty, 1 AFB per 100 fields"

```

```

labelvalue result2 /0.2="Scanty, 2 AFB per 100 fields"
labelvalue result2 /0.3="Scanty, 3 AFB per 100 fields"
labelvalue result2 /0.4="Scanty, 4 AFB per 100 fields"
labelvalue result2 /0.5="Scanty, 5 AFB per 100 fields"
labelvalue result2 /0.6="Scanty, 6 AFB per 100 fields"
labelvalue result2 /0.7="Scanty, 7 AFB per 100 fields"
labelvalue result2 /0.8="Scanty, 8 AFB per 100 fields"
labelvalue result2 /0.9="Scanty, 9 AFB per 100 fields"
labelvalue result2 /1.0="1+ positive"
labelvalue result2 /2.0="2+ positive"
labelvalue result2 /3.0="3+ positive"
labelvalue result2 /9.0="No result recorded"
label result2 "Result of 2nd examination"

labelvalue result3 /0.0="Negative"
labelvalue result3 /4.0="Positive, not quantified"
labelvalue result3 /5.0="Scanty, not quantified"
labelvalue result3 /0.1="Scanty, 1 AFB per 100 fields"
labelvalue result3 /0.2="Scanty, 2 AFB per 100 fields"
labelvalue result3 /0.3="Scanty, 3 AFB per 100 fields"
labelvalue result3 /0.4="Scanty, 4 AFB per 100 fields"
labelvalue result3 /0.5="Scanty, 5 AFB per 100 fields"
labelvalue result3 /0.6="Scanty, 6 AFB per 100 fields"
labelvalue result3 /0.7="Scanty, 7 AFB per 100 fields"
labelvalue result3 /0.8="Scanty, 8 AFB per 100 fields"
labelvalue result3 /0.9="Scanty, 9 AFB per 100 fields"
labelvalue result3 /1.0="1+ positive"
labelvalue result3 /2.0="2+ positive"
labelvalue result3 /3.0="3+ positive"
labelvalue result3 /9.0="No result recorded"
label result3 "Result of 3rd examination"
label regdate "Date of registration"
label laboratory "Laboratory code"
keep country laboratory regdate regyear age sex reason result1 result2 result3
savedata "c_ex01.rec" /replace

```

```

*****
* Test labels, sorting, and count

```

```

cls
close
read "c_ex01.rec"

```

```

tables country result1 /SLA /v1
tables country regyear

```

```

*****
* Clean up and erase temporary session files

```

```

set echo=off
close
define yesno # global
yesno=?Delete all temporary files: 1=yes 0=no?
imif yesno=1 then
    cls
    erase "temp0.rec"
    erase "temp0.chk"
    cls
    type "All temporary files erased" /h2
else
    type "All temporary files retained" /h2
endif
set echo=on

```

Exercise 2: Variability in serial smear results

At the end of this exercise you should be able to:

- Create a subset of 'suspects' from the working dataset
- Create a string variable that combines the three results for each examinee
- Test the given hypothesis on variation in the serial pattern of the results
- Reject or accept a study hypothesis for each country

The diligence of technicians may suffer if they are over-burdened with work. Decreasing diligence in sputum smear examinations may result in copying a first result.

This exercise examines a bit more closely the variability of serial smear grading among those with at least one positive result (it cannot be ascertained among the majority without any positive result).

In a given laboratory A we might find among suspects the following patterns:

Laboratory A Register

Examinee	Other variables	Res 1	Res 2	Res 3
Examinee 1		1+	1+	1+
Examinee 2		neg	neg	neg
Examinee 3		2+	2+	
Examinee 4		neg	neg	neg
Examinee 5		2+	2+	
Examinee 6		neg	1+	1+
Examinee 7		3+	3+	
Examinee 8		neg	neg	neg
Etc				

In a given laboratory B we might find among suspects the following patterns:

Laboratory B Register

Examinee	Other variables	Res 1	Res 2	Res 3
Examinee 1		1+	neg	1+
Examinee 2		neg		
Examinee 3		2+	1+	
Examinee 4		neg	neg	neg
Examinee 5		2+	3+	
Examinee 6		neg	1+	1+
Examinee 7		3+	1+	2+
Examinee 8		neg		
Etc				

If we compare the patterns found in laboratory A with those in laboratory B, we notice that there is much more variation in laboratory B than in laboratory A. In fact, there is virtually no variation in laboratory A for the series of smears for a given suspect.

The amount of tubercle bacilli is, however, not constant in a series of specimens. Most conspicuously, we see this phenomenon when we compare the number of bacilli found in an early morning specimen with an on-the-spot specimen from the same patient. But even if we took a series of 5 on-the-spot specimens from a patient, e.g., in two-hour intervals (as done in “front-loading”), it is likely that the grading of each of the smears made from these specimens will vary to some extent. This may be because the number of bacilli in the secretions varies and / or because the quality of the produced specimen varies and / or the laboratory technician takes by chance particles that differ in content: fresh sputum is not homogenous.

It is thus highly unlikely that all the results from a given examinee recorded in laboratory A reflect the true content of the series of smears. One becomes suspicious that once the technician in laboratory A found a slide to be positive with grade 2+, the subsequent specimen was not properly examined or perhaps even not examined at all, and the result of the first positive specimen was simply copied into the next column. Such observations can be made in seriously overworked laboratories which are forced to examine three smears until they can declare an examinee not to be a case, and if one specimen is positive, to examine additional specimens until the first positive is confirmed by a second positive smear.

By definition, we cannot examine variation among suspects with a series of three negative smears, which is regrettable because this is precisely the group in which this type of problem is most likely to occur. To assess the quality of examination among negative slides, a system of external quality assessment is required. Nevertheless, the results among suspects with at least one positive result may show the extent of variability between such results that might nevertheless be a useful indicator.

We do not know how much variation there must be to make the results look credible (and even if they vary, the technician could in fact have recorded a fictitious variation). What we can do, however, is to compare the extent of variation between laboratories, or in the data set available here, between the four countries, but we can only assess variations among suspects who are cases in the definition of this course.

In other words, the differences in variation are a crude tool to identify laboratories which pay more and which pay less attention to careful and recommended procedures for the examination of serial smears. This exercise should accomplish this.

Tasks:

Exercise hypothesis:

H₀: In each study country, at least 60% of cases found among suspects with a complete diagnostic series show a variation in the serial pattern

- ***Determine with a program C_EX02.PGM the proportions of smears with and without variation in serial smears by country***
- ***Interpret the findings***

Solution to Exercise 2: Variability in serial smear results

Key Learning Points

When you have a hypothesis to test, remember that it may be logical to:

- Create and use a subset of the working dataset
- Create new variable(s)

Tasks:

Exercise hypothesis:

H₀: In each study country, at least 60% of cases found among suspects with a complete diagnostic series show a variation in the serial pattern

- Determine with a program C_EX02.PGM the proportions of smears with and without variation in serial smears by country*
- Interpret the findings*

Solution

Determine with a program C_EX02.PGM the proportions of smears with and without variation in serial smears by country

The following summary output was created:

Crude: Proportion of Grading variation = With variation among all.						
variable	stratum	Total N	n _{Grading variation=With variation}	%	(95% CI)	
Grading variation	Total	7900	3466	43.9	(42.8-45.0)	
Study country	Moldova	870	552	63.4	(60.2-66.6)	
	Mongolia	1499	542	36.2	(33.8-38.6)	
	Uganda	3465	1608	46.4	(44.8-48.1)	
	Zimbabwe	2066	764	37.0	(34.9-39.1)	

Crude: Proportion of Grading variation = With variation among all.

Interpret the findings

Conclusion: Except for Moldova, the hypothesis has to be refuted for each country. Of course, there is no accepted standard what constitutes an “acceptable” minimum level of variation that should be found. Nevertheless, it would appear that the level of variation particularly in Mongolia and Zimbabwe is unexpectedly low, that is the serial results raise some questions on the diligence of reading and reporting sputum smear examination results.

The program C_EX02.PGM that produced the above output is the following:

* Part C, Exercise 2

* Identifying patterns of serial smear results with identical individual results

```

* Objective of the exercise
* Identify series of identical result patterns in the four countries
* The reason for this exercise is that we hypothesize
*   that too regular patterns indicate that the laboratory
*   simply copies a positive result once found to (a) subsequent
*   result(s) rather than properly examining the individual smear
* Thus, this analysis may be an indirect quality assurance program

* First decision: denominator:
* Define the denominator with the choice of the appropriate dataset
*   Data set must be suspects
*   Assessing variability among persons with only negative results
*   is biased as the proportion of these varies widely, thus excluding
*   such examinees
*   Assessing variability among patients with only two results provides
*   too little insight in variability, selecting thus those with three
*   results of which at least one is positive
*   Furthermore, those with unquantified positive results will also
*   bias the result

* Data courtesy:
* Moldova: Dumitru Laticevschi, OR Paris 2003
* Mongolia: Nymadawa Naranbat, OR Paris 2004
* Uganda: Achilles Katamba, OR Paris 2003
* Zimbabwe: Biggie Mabaera, OR Paris 2004

* Written by: Hans L Rieder
* First version: 17 Jan 2010
* Last revision: 28 Apr 2013

```

```

cls
close
logclose

```

```

*****

```

```

* Selection process

```

```

cls
close

```

```

read "c_ex01.rec"

```

```

* All records in dataset:
* 128,808 records

```

```

* Include only suspects for analysis
select reason=0
* 89,362 records retained

```

```

* Select only examinees with three quantified smear results
select result1<4
select result2<4
select result3<4
* 61,064 records retained

```

```

define include #
      include=0
if result1>0 then include=1
if result2>0 then include=1
if result3>0 then include=1
select include=1
* 7,900 records retained

```

```

savedata "temp_01.rec" /replace

```

```

*****

```

```

* Variable definition for analysis

```

```

cls
close

read "temp_01.rec"

define variation #
variation=1
if (result1=result2) and (result1=result3) then variation=0
label variation "Grading variation"
labelvalue variation /0="No variation"
labelvalue variation /1="With variation"

*****
* Analysis: Hypothesis testing that
*   at least 60% have variation

cls
ciplot variation country /ng

*****
* Clean up

define yesno # global
yesno=?Delete all temporary files: 1=yes 0=no?
imif yesno=1 then
  cls
  close
  erasepng /all /noconfirm
  erase "temp_01.chk"
  erase "temp_01.rec"
  cls
  type "All temporary files erased" /h2
else
  type "All temporary files retained" /h2
endif
set echo=on

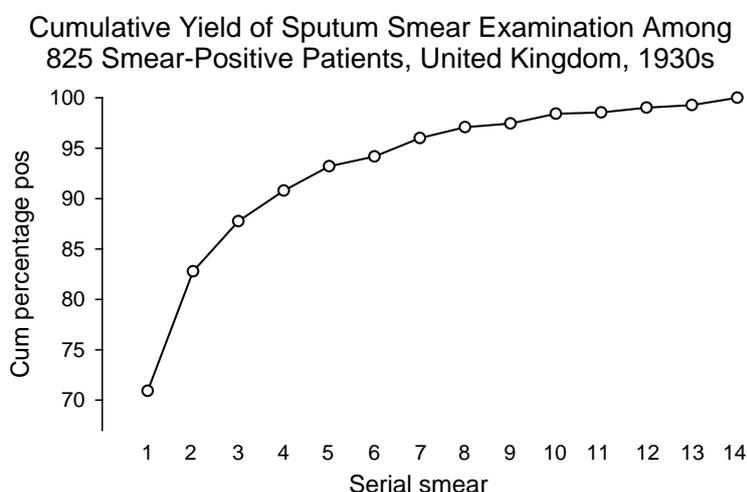
```

Exercise 3: Incremental yield from serial smears

At the end of this exercise you should be able to:

- Create a subset of 'suspects' from the working dataset
- Create a string variable that combines the three results for each examinee
- Make calculations using a spreadsheet
- Test the given hypothesis on the incremental yield from the third smear
- Reject or accept a study hypothesis for each country

The diminishing return of serial smears is known from studies that have examined multiple serial specimens, as for example the following study from the 1930s:



Hunter RA. *Tubercle* 1940;21:341-59

This study suggests that each serial smear adds an additional increment in case yield, but the incremental yield gets smaller with each additional examination. Program managers must thus arrive at some optimum that requires the least amount of work (number of smear examinations) to yield a large proportion of cases. The “three smear policy” is such a compromise that has been reached internationally and became reflected in the above mentioned guidelines.

The Union and WHO recommended in the past that each suspect should have three sputum smear examinations before being declared to be “sputum smear-negative”. Some countries recommended only two examinations. The reason for this difference is that The Union and the WHO thought that making a third examination after two smears are negative would offer a sufficiently rewarding incremental yield (how much is rewarding – has anybody ever defined it?) from this third smear as to justify the additional work load for laboratories. Some microscopy laboratories are, however, so burdened with work (particularly in Africa) that a reduction in the required number of examinations would come as a great relief. It is also very possible that over-burdened laboratories may become less meticulous in the examination of a third smear after a first and second smear have been negative, which may reduce the potential incremental gain. Most of the studies determining the incremental yield from the third examination were done under relatively controlled conditions, but there was not much information around on the yield under routine conditions in low- and middle-income

countries. The primary hypothesis for the operations research study of the course cohorts of 2003 and 2004 was precisely to test the effectiveness under routine conditions from a representative sample of laboratories in four countries. As these were the only studies of this extent and representativeness, the published findings of these studies greatly contributed to the change in policy of WHO in June 2008 to recommend that routine screening of tuberculosis suspects should be limited to two serial examinations to exclude sputum smear-positive tuberculosis. This demonstrates how powerful a relatively simple study design can be in public health policy shaping, if carried out in a representative manner with diligent adherence to quality assurance.

In this exercise, the approach to this issue will be reproduced. We will analyze the data in two parts:

Part 1: incremental yield: we ask what proportion of all cases (being a case in any of up to three serial examinations) is found already on the first smear, which proportion is found on the second smear if the first is negative, and which proportion is found on the third smear after the first two have been negative.

Part 2: program efficiency of the third smear: in addition to the fraction of all ultimately positives, we also take the proportion of cases among all diagnostic examinees into account. The product of these two fractions is the proportion of cases found only with a third examination out of the total number of examinees with a diagnostic examination. We can express the efficiency by taking the reciprocal of this product and obtain the number of smears that need to be examined with a third smear to find one additional case among diagnostic examinees solely when doing a third serial smear (negative on the first two smears).

For *Part 1* we look only at cases and just determine frequencies of “essential patterns” if at least one of the three smears is positive. EpiData provides the possibility to display the point estimate and the 95% confidence interval around the three proportions.

For *Part 2* we go a bit more into detail because we also need to know the total of all examinees who came for a diagnostic examination. We do thus not only need the three essential patterns among those positive on at least one of the three, but all six essential patterns. With our case definition that a suspect becomes a case once acid-fast bacilli are found implies that an additional third examination has to be done only if the two preceding examinations have been negative:

$$\text{NNP} / (\text{NNP} + \text{NNN})$$

This is a fraction, but the hypothesis was about the number of smears. In analogy, you may consider the situation where you know that 20 out of 100 people have a characteristic and you now ask how many you have to examine to find the characteristic.

Confidence intervals

Our fraction might be very small despite the large number of suspects in the database. As the implication of refuting the hypothesis has serious programmatic consequences it is advisable to calculate confidence intervals around the number of smears and decide only to refute if the lower interval is in excess of the hypothesis number X.

The classic approach to estimating 95% confidence intervals is used when the population from which the cases arise is defined (observable) and a subset of this population is examined.

We define **P** as the proportion of cases found on the third smear only among those with three examinations:

$$P = (NNP)/(NNN+NNP)$$

The standard error of P [(SE(P))] is calculated from the square root of a function derived from P:

$$SE(P) = \text{SQRT}(P*(1-P)/(NNN+NNP))$$

And the 95% confidence intervals are:

$$95\%_{\text{low}} = P - 1.96*SE(P)$$

$$95\%_{\text{upper}} = P + 1.96*SE(P)$$

However, for the number of slides we will need the reciprocals of these values.

Tasks:

- *Determine with a program C_EX03.PGM the incremental yield of cases with serial smears with the denominator being all cases (Part 1)*
- *Determine in the same program for Part 2 the number of suspects with the six essential patterns listed in the table below. Use the following hypothesis to guide you in this part:*

Exercise hypothesis:
 Ho: Not more than 125 third smear examinations have to be made to find one additional case of tuberculosis in each of the four study countries

- *Create a table in spreadsheet by country as follows:*

	Moldova	Mongolia	Uganda	Zimbabwe	Total
Total					
Pattern					
N99					
NN9					
NNN					
NNP					
Npx					
Px					
Prop positive					
Yield					
First					
Second					
Third					

X

P

SE(P)

95% low

95% high

Smears

95% low

95% high

Hypothesis:

- *Interpret the findings*

Solution to Exercise 3: Incremental yield from serial smears

Key Learning Points

When you have a hypothesis to test, remember that it may be logical to:

- a. Create and use a subset of the working dataset
- b. Create new variable(s)

Tasks:

- *Determine with a program C_EX03.PGM the incremental yield of cases with serial smears with the denominator being all cases (Part 1)*
- *Determine in the same program for Part 2 the number of suspects with the six essential patterns listed in the table below. Use the following hypothesis to guide you in this part:*

Exercise hypothesis:

H₀: Not more than 125 third smear examinations have to be made to find one additional case of tuberculosis in each of the four study countries

- *Create a table in spreadsheet by country as follows:*

	Moldova	Mongolia	Uganda	Zimbabwe	Total
Total					
Pattern					
N99					
NN9					
NNN					
NNP					
Npx					
Px					
Prop positive					
Yield					
First					
Second					
Third					
X					
P					
SE(P)					
95% low					

95% high

Smears

95% low

95% high

Hypothesis:

- *Interpret the findings*

Solution:

Part 1: After creating the essential patterns and selecting only the cases among examinees with a diagnostic examination, we created the following output:

Incremental yield

Moldova

Essential patterns			
	N	%	(95% CI)
NNP	34	3.0	(2.2-4.2)
NPx	84	7.4	(6.0-9.1)
Px	1013	89.6	(87.6-91.2)
Total	1131	100.0	

Mongolia

Essential patterns			
	N	%	(95% CI)
NNP	12	0.7	(0.4-1.2)
NPx	42	2.4	(1.8-3.3)
Px	1663	96.9	(95.9-97.6)
Total	1717	100.0	

Uganda

Essential patterns			
	N	%	(95% CI)
NNP	107	1.5	(1.2-1.8)
NPx	487	6.7	(6.1-7.3)
Px	6686	91.8	(91.2-92.4)
Total	7280	100.0	

Zimbabwe

Essential patterns			
	N	%	(95% CI)
NNP	155	4.5	(3.9–5.2)
NPx	325	9.4	(8.5–10.4)
Px	2969	86.1	(84.9–87.2)
Total	3449	100.0	

In Part 2, the following output was created in EpiData Analysis:

	country	nnp	np _x	p _x	nnn	nn ₉	tot	sm95low	smpoint	sm95high	hypothesis
1	Moldova	34	84	1013	8424	1579	12713	186.3	248.8	374.3	Refute
2	Mongolia	12	42	1663	12264	708	15397	653.5	1023.0	2354.0	Refute
3	Uganda	107	487	6686	14736	3325	28666	116.7	138.7	171.0	Accept
4	Zimbabwe	155	325	2969	17740	2706	26601	99.8	115.5	136.9	Accept

Interpretation:

The recorded results show that the number of smears that need to be examined to find one additional case on a third serial smear examination that had not been found already on the first two exceeded 125 (one week's work) in Moldova and Mongolia, indicating the inefficiency of the requirement for three smears before declaring a patient smear-negative at least in these two countries.

We used the following program C_EX03.PGM to get these outputs:

```
* Part C, Exercise 3
* This is b_ex03 EpiData Analysis program
* to determine the incremental yield from serial smears

* Data courtesy:
* Moldova: Dumitru Laticevschi, OR Paris 2003
* Mongolia: Nymadawa Naranbat, OR Paris 2004
* Uganda: Achilles Katamba, OR Paris 2003
* Zimbabwe: Biggie Mabaera, OR Paris 2004

* Written by: Hans L Rieder
* First version: 12 Feb 2009
* Last revision: 26 Apr 2018

cls
close
logclose

*****
* Prepare data set

cls
close
logclose

read "c_ex01.rec"

* Definition positive: any AFB in any of three results
* Values: "P" (positive) or "N" (negative)
* or "9" (unknown)

cls
```

```

* Case definition
gen i case=0
if result1>0 and result1<9 then case=1
if result2>0 and result2<9 then case=1
if result3>0 and result3<9 then case=1
label case "Microscopy-defined case"

cls
* Define essential patterns from
* all possible patterns
gen s(3) pattern="NNN"
if result3=9 then pattern="NN9"
if result2=9 then pattern="N99"
if result3>0 and result3<9 then pattern="NNP"
if result2>0 and result2<9 then pattern="NPx"
if result1>0 and result1<9 then pattern="Px"
label pattern "Essential patterns"

keep pattern reason case country
savedata "temp_01.rec" /replace

*****
* Analysis part 1
* Look for yield from 1st, 2nd, 3rd smear by country

cls
close
logclose

read "temp_01.rec"

cls
set echo=off
select reason=0 and case=1
cls
type "Incremental yield" /h2
type "Moldova" /h2
freq pattern /c /ci if country=1
type "Mongolia" /h2
freq pattern /c /ci if country=2
type "Uganda" /h2
freq pattern /c /ci if country=3
type "Zimbabwe" /h2
freq pattern /c /ci if country=4
select
set echo=on

*****
* Analysis part 2
* Efficiency of the third smear in finding
* an additional case: number of smears that
* need to be examined among all diagnostic
* examinees to find one additional case

cls
close
logclose

read "temp_01.rec"

aggregate pattern country /save="yield.rec" /replace /close

* Note: a more efficient way to do the following will be shown \
* in Part D
cls
close
read "yield.rec"
select pattern="NNP"
gen i nnp=n
savedata "nnp.rec" /replace

cls
close
read "yield.rec"

```

```

select pattern="NPx"
gen i npx=n
savedata "npx.rec" /replace

cls
close
read "yield.rec"
select pattern="Px"
gen i px=n
savedata "px.rec" /replace

cls
close
read "yield.rec"
select pattern="NNN"
gen i nnn=n
savedata "nnn.rec" /replace

cls
close
read "yield.rec"
select pattern="NN9"
gen i nn9=n
savedata "nn9.rec" /replace

cls
close
read "yield.rec"
select pattern="N99"
gen i n99=n
savedata "n99.rec" /replace

cls
close
read "nnp.rec"
merge country /file="npx.rec"
merge country /file="px.rec"
merge country /file="nnn.rec"
merge country /file="nn9.rec"
merge country /file="n99.rec"

define tot #####
tot=nnp+npx++px+nnn+nn9+n99

define totpos #####
totpos=nnp+npx++px

drop n pattern mergevar
savedata "pattern.rec" /replace

cls
close
read "pattern.rec"

cls
define p #.#####
p=nnp/(nnp+nnn)

define sep #.#####
sep=sqrt(p*(1-p)/(nnp+nnn))

define cilow #.#####
cilow=p-1.96*sep

define cihigh #.#####
cihigh=p+1.96*sep

define smpoint ###.#
smpoint=1/p

define sm95low ###.#
sm95low=1/cihigh

define sm95high ###.#
sm95high=1/cilow

```

```

cls
        define hypothesis _____
                hypothesis="Accept"
if sm95low>125 then hypothesis="Refute"

cls
set display databrowser=on
browse country nnp npx px nnn nn9 n99 tot sm95low smpoint sm95high hypothesis
set display databrowser=off

*****
* Clean up and erase temporary session files

set echo=off
define yesno # global
yesno=?Delete all temporary files: 1=yes 0=no?
imif yesno=1 then
    close
    cls
    type "Be patient, you will be alerted to completion" /h2
    erase "n99.chk"      "
    erase "n99.rec"     "
    erase "nn9.chk"     "
    erase "nn9.rec"     "
    erase "nnn.chk"     "
    erase "nnn.rec"     "
    erase "nnp.chk"     "
    erase "nnp.rec"     "
    erase "npx.chk"     "
    erase "npx.rec"     "
    erase "pattern.chk"
    erase "pattern.rec"
    erase "px.chk"      "
    erase "px.rec"      "
    erase "temp_01.chk"
    erase "temp_01.rec"
    erase "yield.chk"   "
    erase "yield.rec"   "
    cls
    type "All temporary files erased" /h2
else
    type "All temporary files retained" /h2
endif
set echo=on

```

Exercise 4: Confirmatory results in serial smears

At the end of this exercise you should be able to:

- a. Create a subset of 'suspects' from the working dataset, with the required number of examinations to test the hypotheses
- b. Make a distinction between scanty and positive smear results
- c. Create string variables that combines the three results for each examinee
- d. Recode some string variables to numeric variables
- e. Make calculations using a spreadsheet
- f. Test the given hypotheses on confirmatory results in serial smears
- g. Reject or accept a study hypothesis for each country
- h. Interpret your findings

The bacteriological definition by microscopy of a sputum smear-positive tuberculosis case following WHO required that a positive smear examination had to be confirmed by a second positive result.

This study:

Mabaera B, Lauritsen J M, Katamba A, Laticevschi D, Naranbat N, Rieder H L. Sputum smear-positive tuberculosis: empiric evidence challenges the need for confirmatory smears. *Int J Tuberc Lung Dis* 2007;11:959-64.

contributed to a policy change in WHO recommendations that were decided in June 2007 following the publication of these findings.

In this exercise, the approach to the problem is reproduced.

The dataset provided here allows the determination of how frequent a scanty positive or a positive smear result is actually confirmed in daily practice in these four countries. It allows further to determine how frequent such a confirmation can be made among suspects who actually had a complete set of examinations.

Exercise hypotheses

- H₀₁: At least 80 per cent of suspects with at least one scanty or positive smear result have a confirmatory scanty or positive result
- H₀₂: At least 90 per cent of suspects with three serial examination among which there is at least one scanty or positive smear result have a confirmatory scanty or positive result in another examination

Tasks:

- *Write a program C_EX04.PGM that determines the proportion of suspects who have a confirmatory examination, making a distinction between scanty and positive smears. Produce a table by country.*
- *Produce a second table in the same program to determine the proportion of suspects who have a confirmatory examination and who had a complete series of smears, making a distinction between scanty and positive smears.*
- *Interpret the findings.*

Solution to Exercise 4: Confirmatory results in serial smears

Key Learning Points

When you have a hypothesis to test, remember that it may be logical to:

- a. Create and use a subset of the working dataset
- b. Create new variable(s)
- c. Produce multiple frequencies of results with different selection criteria
- d. Make use of other software applications e.g. a spreadsheet to make calculations.

Exercise hypotheses

H₀₁: At least 80 per cent of suspects with at least one scanty or positive smear result have a confirmatory scanty or positive result

H₀₂: At least 90 per cent of suspects with three serial examination among which there is at least one scanty or positive smear result have a confirmatory scanty or positive result in another examination

Tasks:

- *Write a program C_EX04.PGM that determines the proportion of suspects who have a confirmatory examination, making a distinction between scanty and positive smears. Produce a table by country.*
- *Produce a second table in the same program to determine the proportion of suspects who have a confirmatory examination and who had a complete series of smears, making a distinction between scanty and positive smears.*
- *Interpret the findings.*

Solution

Producing the required results requires multiple frequencies with different selection criteria. The program C_EX04.PGM producing these is shown afterwards, followed by a summary table that is best made in a spreadsheet C_EX04.XLS.

Interpretation:

Moldova had the highest frequency of confirmatory results, in fact more than 95 per cent. As suggested in previous exercises, there might be considerable copying of results, thus it is doubtful to what extent the recorded confirmations correspond to actual results. The opposite is the case in Uganda, where fewer than 65 per cent had a confirmatory result (Table 1).

As shown in table 2, the absence of confirmatory results is simply attributable to the fact that once a smear is positive (or scanty), no further examination is being made. If such an examination is being made, then a confirmation was obtained in 90 per cent of more, with the exception of Zimbabwe, where it was just slightly below the critical proportion.

In summary, this exercise showed that confirmatory smears can generally be made, but in some countries, they are simply not sought. The more general question then is whether it is sensible to require such confirmatory smears, particular in the light that the treatment decision is not greatly affected by it, only the surveillance definition.

The program C_EX04.PGM:

```

* Part C, Exercise 4
* This is b_ex04 EpiData Analysis program
* to determine the frequency of confirmatory smears

* Moldova, Mongolia, Uganda, Zimbabwe
* Data courtesy:
* Moldova: Dumitru Laticevschi, OR Paris 2003
* Mongolia: Nymadawa Naranbat, OR Paris 2004
* Uganda: Achilles Katamba, OR Paris 2003
* Zimbabwe: Biggie Mabaera, OR Paris 2004

* Written by: Hans L Rieder
* First version: 17 Jan 2010
* Last revision: 29 Apr 2013

*****
* Selection
cls
close
logclose

read "c_ex01.rec"

* Selection criteria:
* - At least 1 smear must be positive
* - Diagnostic examinees only

* Definitions:
* - Case: at least 1 AFB in at least 1 smear
* - Scanty: 1-9 AFB or "Scanty not quantified"
* - Positive: any non-scanty positive result
* - Scanty series: at least 1 smear is scanty
* - Positive series: at least 1 is positive, none is scanty

gen i case=0
if result1>0 and result1<9 then case=1
if result2>0 and result2<9 then case=1
if result3>0 and result3<9 then case=1

select case=1
select reason=0

keep country result1 result2 result3
savedata "temp_01.rec" /replace

*****
* New variable definition
cls
close
logclose

read "temp_01.rec"

* code for scanty results in series
define scanty1 <A>
if result1=0 then scanty1="N"
if result1>0 and result1<1 then scanty1="S"
if result1>=1 and result1<5 then scanty1="P"

```

```

if result1=5          then scanty1="S"
if result1=4          then scanty1="P"
if result1=9          then scanty1="9"

                        define scanty2 <A>
if result2=0          then scanty2="N"
if result2>0 and result2<1 then scanty2="S"
if result2>=1 and result2<5 then scanty2="P"
if result2=5          then scanty2="S"
if result2=4          then scanty2="P"
if result2=9          then scanty2="9"

                        define scanty3 <A>
if result3=0          then scanty3="N"
if result3>0 and result3<1 then scanty3="S"
if result3>=1 and result3<5 then scanty3="P"
if result3=5          then scanty3="S"
if result3=4          then scanty3="P"
if result3=9          then scanty3="9"

define scanty ____
scanty=scanty1+scanty2+scanty3

cls
gen i confirm=0
if substr(scanty,1,1)="N" and substr(scanty,2,1)="N" and substr(scanty,3,1)="P" then confirm=1
if substr(scanty,1,1)="N" and substr(scanty,2,1)="N" and substr(scanty,3,1)="S" then confirm=3
if substr(scanty,1,1)="N" and substr(scanty,2,1)="P" and substr(scanty,3,1)="9" then confirm=1
if substr(scanty,1,1)="N" and substr(scanty,2,1)="P" and substr(scanty,3,1)="N" then confirm=1
if substr(scanty,1,1)="N" and substr(scanty,2,1)="P" and substr(scanty,3,1)="P" then confirm=2
if substr(scanty,1,1)="N" and substr(scanty,2,1)="P" and substr(scanty,3,1)="S" then confirm=4
if substr(scanty,1,1)="N" and substr(scanty,2,1)="P" and substr(scanty,3,1)="9" then confirm=3
if substr(scanty,1,1)="N" and substr(scanty,2,1)="S" and substr(scanty,3,1)="N" then confirm=3
if substr(scanty,1,1)="N" and substr(scanty,2,1)="S" and substr(scanty,3,1)="P" then confirm=4
if substr(scanty,1,1)="N" and substr(scanty,2,1)="S" and substr(scanty,3,1)="S" then confirm=4

cls
if substr(scanty,1,1)="P" and substr(scanty,2,1)="N" and substr(scanty,3,1)="9" then confirm=1
if substr(scanty,1,1)="P" and substr(scanty,2,1)="N" and substr(scanty,3,1)="N" then confirm=1
if substr(scanty,1,1)="P" and substr(scanty,2,1)="N" and substr(scanty,3,1)="P" then confirm=2
if substr(scanty,1,1)="P" and substr(scanty,2,1)="N" and substr(scanty,3,1)="S" then confirm=4
if substr(scanty,1,1)="P" and substr(scanty,2,1)="P" and substr(scanty,3,1)="9" then confirm=2
if substr(scanty,1,1)="P" and substr(scanty,2,1)="P" and substr(scanty,3,1)="N" then confirm=2
if substr(scanty,1,1)="P" and substr(scanty,2,1)="P" and substr(scanty,3,1)="P" then confirm=2
if substr(scanty,1,1)="P" and substr(scanty,2,1)="P" and substr(scanty,3,1)="S" then confirm=4
if substr(scanty,1,1)="P" and substr(scanty,2,1)="S" and substr(scanty,3,1)="9" then confirm=4
if substr(scanty,1,1)="P" and substr(scanty,2,1)="S" and substr(scanty,3,1)="N" then confirm=4
if substr(scanty,1,1)="P" and substr(scanty,2,1)="S" and substr(scanty,3,1)="P" then confirm=4
if substr(scanty,1,1)="P" and substr(scanty,2,1)="S" and substr(scanty,3,1)="S" then confirm=4
if substr(scanty,1,1)="P" and substr(scanty,2,1)="9" and substr(scanty,3,1)="9" then confirm=1

cls
if substr(scanty,1,1)="S" and substr(scanty,2,1)="N" and substr(scanty,3,1)="9" then confirm=3
if substr(scanty,1,1)="S" and substr(scanty,2,1)="N" and substr(scanty,3,1)="N" then confirm=3
if substr(scanty,1,1)="S" and substr(scanty,2,1)="N" and substr(scanty,3,1)="P" then confirm=4
if substr(scanty,1,1)="S" and substr(scanty,2,1)="N" and substr(scanty,3,1)="S" then confirm=4
if substr(scanty,1,1)="S" and substr(scanty,2,1)="P" and substr(scanty,3,1)="9" then confirm=4
if substr(scanty,1,1)="S" and substr(scanty,2,1)="P" and substr(scanty,3,1)="N" then confirm=4
if substr(scanty,1,1)="S" and substr(scanty,2,1)="P" and substr(scanty,3,1)="P" then confirm=4
if substr(scanty,1,1)="S" and substr(scanty,2,1)="P" and substr(scanty,3,1)="S" then confirm=4
if substr(scanty,1,1)="S" and substr(scanty,2,1)="S" and substr(scanty,3,1)="9" then confirm=4
if substr(scanty,1,1)="S" and substr(scanty,2,1)="S" and substr(scanty,3,1)="N" then confirm=4
if substr(scanty,1,1)="S" and substr(scanty,2,1)="S" and substr(scanty,3,1)="P" then confirm=4
if substr(scanty,1,1)="S" and substr(scanty,2,1)="S" and substr(scanty,3,1)="S" then confirm=4
if substr(scanty,1,1)="S" and substr(scanty,2,1)="9" and substr(scanty,3,1)="9" then confirm=3

cls
label confirm "Confirmed by another positive"
labelvalue confirm /0="All negative"
labelvalue confirm /1="Pos not confirmed"
labelvalue confirm /2="Pos confirmed"
labelvalue confirm /3="Scanty not confirmed"
labelvalue confirm /4="Scanty confirmed"

cls
gen i scantpos=0

```

```

* Scanty, not confirmed
if scanty="NNS" then scantpos=1
if scanty="NS9" then scantpos=1
if scanty="NSN" then scantpos=1
if scanty="S99" then scantpos=1
if scanty="SN9" then scantpos=1
if scanty="SNN" then scantpos=1

cls
* Positive not confirmed
if scanty="NNP" then scantpos=2
if scanty="NP9" then scantpos=2
if scanty="NPN" then scantpos=2
if scanty="P99" then scantpos=2
if scanty="PN9" then scantpos=2
if scanty="PNN" then scantpos=2

cls
* Positive, confirmed, no Scanty in series
if scanty="NPP" then scantpos=3
if scanty="PNP" then scantpos=3
if scanty="PP9" then scantpos=3
if scanty="PPN" then scantpos=3
if scanty="PPP" then scantpos=3

cls
* Scanty, confirmed, no Positive in series
if scanty="NSS" then scantpos=4
if scanty="SNS" then scantpos=4
if scanty="SSN" then scantpos=4
if scanty="SS9" then scantpos=4
if scanty="SSS" then scantpos=4

cls
* Scanty-Positive, mixed scanty and positive in series
if scanty="NPS" then scantpos=5
if scanty="NSP" then scantpos=5
if scanty="PNS" then scantpos=5
if scanty="PPS" then scantpos=5
if scanty="PS9" then scantpos=5
if scanty="PSN" then scantpos=5
if scanty="PSP" then scantpos=5
if scanty="PSS" then scantpos=5
if scanty="SNP" then scantpos=5
if scanty="SP9" then scantpos=5
if scanty="SPN" then scantpos=5
if scanty="SPP" then scantpos=5
if scanty="SPS" then scantpos=5
if scanty="SSP" then scantpos=5

cls
label scantpos "Confirmation of smears"
labelvalue scantpos /1="Single Scanty"
labelvalue scantpos /2="Single Positive"
labelvalue scantpos /3="Positive confirmed by Positive"
labelvalue scantpos /4="Scanty confirmed by Scanty"
labelvalue scantpos /5="Scanty confirmed by Positive"

cls
define confres #
if confirm=1 or confirm=3 then confres=0
if confirm=2 or confirm=4 then confres=1
label confres "Confirmed by another positive"
labelvalue confres /0="Not confirmed"
labelvalue confres /1="Confirmed"

savedata "temp_02.rec" /replace

*****
* Output for C_EX04

cls
close

read "temp_02.rec"

```

```

* Table 1. Confirmatory smears among all cases
cls
logclose
set echo=off
logopen "c_ex04_1.txt" /replace
ciplot confres country /ng if confirm<>0
ciplot scantpos country /ng if confirm<>0

title "Confirmation in all countries"
freq confres /c /ci if confirm<>0
freq scantpos /c /ci if confirm<>0

title "Confirmation in Moldova"
freq confres /c /ci if confirm<>0 and country=1
freq scantpos /c /ci if confirm<>0 and country=1

title "Confirmation in Mongolia"
freq confres /c /ci if confirm<>0 and country=2
freq scantpos /c /ci if confirm<>0 and country=2

title "Confirmation in Uganda"
freq confres /c /ci if confirm<>0 and country=3
freq scantpos /c /ci if confirm<>0 and country=3

title "Confirmation in Zimbabwe"
freq confres /c /ci if confirm<>0 and country=4
freq scantpos /c /ci if confirm<>0 and country=4
logclose
set echo=on

* Table 2. Confirmatory smears among all cases with three examinations
cls
logclose
logopen "c_ex04_2.txt" /replace
select
select confirm<>0
select substr(scanty,2,1)<>"9"
select substr(scanty,3,1)<>"9"

set echo=off
cls

title "Confirmation in all countries"
freq confres /c /ci
freq scantpos /c /ci

title "Confirmation in Moldova"
freq confres /c /ci if country=1
freq scantpos /c /ci if country=1

title "Confirmation in Mongolia"
freq confres /c /ci if country=2
freq scantpos /c /ci if country=2

title "Confirmation in Uganda"
freq confres /c /ci if country=3
freq scantpos /c /ci if country=3

title "Confirmation in Zimbabwe"
freq confres /c /ci if country=4
freq scantpos /c /ci if country=4
logclose

```

Exercise 4. Table 1. Confirmatory smears among all cases

	Moldova			Mongolia			Uganda			Zimbabwe			Total		
	Number	%	(95% CI)	Number	%	(95% CI)	Number	%	(95% CI)	Number	%	(95% CI)	Number	%	(95% CI)
Total	1,131			1,717			7,280			3,449			13,577		
Not confirmed	151	13.4	(11.5-15.5)	89	5.2	(4.2-6.3)	2,804	38.5	(37.4-39.6)	672	19.5	(18.2-20.8)	3,716	27.4	(26.6-28.1)
Single scanty	27	2.4	(1.6-3.5)	24	1.4	(0.9-2.1)	43	0.6	(0.4-0.8)	98	2.8	(2.3-3.5)	192	1.4	(1.2-1.6)
Single positive	124	11.0	(9.3-12.9)	65	3.8	(3.0-4.8)	2,761	37.9	(36.8-39.0)	574	16.6	(15.4-17.9)	3,524	26.0	(25.2-26.7)
Confirmed	980	86.6	(84.5-88.5)	1,628	94.8	(93.7-95.8)	4,476	61.5	(60.4-62.6)	2,777	80.5	(79.2-81.8)	9,861	72.6	(71.9-73.4)
Positive+positive	843	74.5	(71.9-77.0)	1,502	87.5	(85.8-89.0)	4,342	59.6	(58.5-60.8)	2,563	74.3	(72.8-75.7)	9,250	68.1	(67.3-68.9)
Scanty+scanty	24	2.1	(1.4-3.1)	30	1.7	(1.2-2.5)	23	0.3	(0.2-0.5)	107	3.1	(2.6-3.7)	184	1.4	(1.2-1.6)
Scanty+positive	113	10.0	(8.4-11.9)	96	5.6	(4.6-6.8)	111	1.5	(1.3-1.8)	107	3.1	(2.6-3.7)	427	3.1	(2.9-3.5)

Exercise 4. Table 2. Confirmatory smears among all cases with three examinations

	Moldova			Mongolia			Uganda			Zimbabwe			Total		
	Number	%	(95% CI)	Number	%	(95% CI)	Number	%	(95% CI)	Number	%	(95% CI)	Number	%	(95% CI)
Total	904			1,503			3,778			2,829			9,014		
Not confirmed	92	10.2	(8.4-12.3)	55	3.7	(2.8-4.7)	184	4.9	(4.2-5.6)	358	12.7	(11.5-13.9)	689	7.6	(7.1-8.2)
Single scanty	19	2.1	(1.3-3.3)	19	1.3	(0.8-2.0)	17	0.4	(0.3-0.7)	44	1.6	(1.2-2.1)	99	1.1	(0.9-1.3)
Single positive	73	8.1	(6.5-10.0)	36	2.4	(1.7-3.3)	167	4.4	(3.8-5.1)	314	11.1	(10.0-12.3)	590	6.5	(6.1-7.1)
Confirmed	812	89.8	(87.7-91.6)	1,448	96.3	(95.3-97.2)	3,594	95.1	(94.4-95.8)	2,471	87.3	(86.1-88.5)	8,325	92.4	(91.8-92.9)
Positive+positive	688	76.1	(73.2-78.8)	1,337	89.0	(87.3-90.4)	3,470	91.8	(90.9-92.7)	2,308	81.6	(80.1-83.0)	7,803	86.6	(85.8-87.3)
Scanty+scanty	20	2.2	(1.4-3.4)	21	1.4	(0.9-2.1)	22	0.6	(0.4-0.9)	76	2.7	(2.2-3.3)	139	1.5	(1.3-1.8)
Scanty+positive	104	11.5	(9.6-13.7)	90	6.0	(4.9-7.3)	102	2.7	(2.2-3.3)	87	3.1	(2.5-3.8)	383	4.2	(3.9-4.7)

Part D. More on EpiData software

Part D: More on EpiData software

Exercise 1: Relational database and aggregating vs from “Long-to-wide”

Exercise 2: A statistical process control chart

Exercise 3: A simplified survival analysis

Exercise 4: Creating a menu for standard reports

Exercise 5: Formatting standardized analysis output in a spreadsheet

Introductory note

Part D will address operationally relevant concepts in data collection and data analysis:

- How do we deal with a situation, where each individual has a varying number of observations?
- How do we determine statistically relevant deviations from a proportion over an observation period when the denominator varies with each time unit over the observation period?
- What is survival analysis and how to deal with it in EpiData Analysis?
- How to make a menu-driven, HTML-based EpiData Analysis interface to run standard reports?

Exercise 1: A relational database and “Aggregating” vs from “long-to-wide”

At the end of this exercise you should be able to:

- a. Create a relational database for a varying number of observations
- b. Merge a child file to the parent file
- c. Recognizing when to use “Aggregate” and when to transpose data
- d. Transpose multiple observations (columns) into a single record (row)

Not all laboratories keep their registers as The Union and WHO recommend for the Tuberculosis Laboratory Register, where 1 line corresponds to 1 examinee rather than to 1 examination. In fact, many laboratories enter the results for each examination on one line. If such an approach is chosen, we may find a register as follows:

Patient	Date of exam	Sex	Marital status	Blood sugar	Sputum	Result
A	24-Mar-2007	Male	Married	6.3	Mucoid	1+
B	24-Mar-2007	Male	Divorced	4.9	Muco-purulent	Neg
C	24-Mar-2007	Female	Single	5.2	Purulent	Neg
D	24-Mar-2007	Female	Widowed	7.3	Blood-tinged	2 per 100
A	25-Mar-2007	Male		7.3	Salivary	Neg
D	25-Mar-2007	Female		7.4	Mucoid	2+
A	26-Mar-2007			7.2	Purulent	1+
C	26-Mar-2007	Female		4.8	Muco-purulent	Neg
E	27-Mar-2007	Male	Married	8.2		1+
F	27-Mar-2007	Female	Annulled	7.4	Purulent	Neg
G	27-Mar-2007	Male	Cohabiting	6.9	Salivary	Neg
G	28-Mar-2007	Male		7.2	Mucoid	2+
E	28-Mar-2007	Male		7.9	Purulent	2+
F	31-Mar-2007	Female		7.2	Muco-purulent	3+
H	31-Mar-2007		Married	6.6	Mucoid	Neg
I	31-Mar-2007	Male	Separated	8.3	Salivary	Neg
H	1-Apr-2007	Female		6.9	Muco-purulent	1+
F	1-Apr-2007	Female	Engaged	7.7	Purulent	2+
I	1-Apr-2007	Male	Single	8.0	Mucoid	8 per 100
G	1-Apr-2007	Male		7.6	Muco-purulent	1+
K	2-Apr-2007	Female	Married	4.5	Purulent	Neg
I	2-Apr-2007	Male		8.2	Muco-purulent	
H	2-Apr-2007	Female		6.6	Mucoid	1+
I	3-Apr-2007	Male		8.1	Mucoid	1+

This type of a register requires a different approach to both data entry and data analysis than we used before. Two important things need to be considered:

- 1) The same patient may appear again and again on sequential dates
- 2) Not every patient has the same number of visits

Some patient characteristics do not change over time such as, in this example, the identity of the patient, sex, and marital status (well, perhaps not during these short intervals). Others do change, such as the date of examination, blood sugar, the aspect of the sputum and the sputum smear examination result.

To capture such information in a single data entry form would be very inconvenient: 1) one would have to anticipate the maximum of allowable visits, and 2) if one patient has a single visit, one would still have to complete all fields with the codes for missing values up to the maximum allotted if we insist that all fields must be MUSTENTER fields.

Rule: *If an individual has a single observation for each variable or a fixed number of observations for each variable, then a single EpiData entry form is the best solution. If an individual has a variable number of observations for each variable, the choice is a relational database.*

Building a relational database requires determining which information is static for an individual and which changes over repeated observations.

EpiData Manager and EntryClient

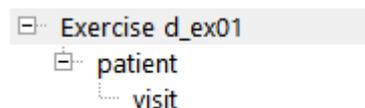
You may refer to Exercise 7 “Relational database” in Part A on the principles of and how to create a relational database.

The working example

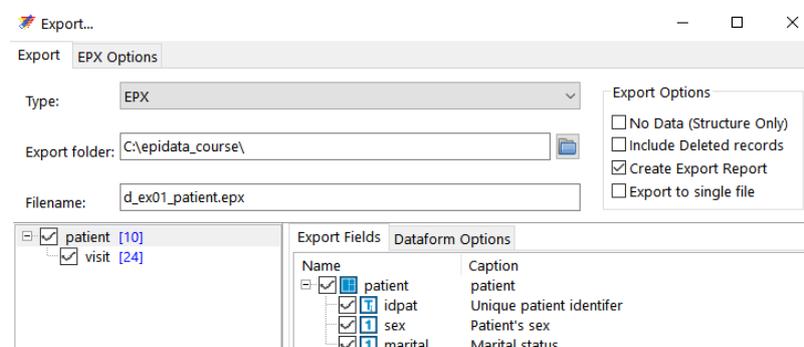
You will create an EpiData file:

d_ex01.epx

To get uniform naming during design and entry, we propose the following:



The project is saved as “d_ex01.epx” and the two databases it consists of are the parent dataset named “patient” and the child set “visit”. When exporting:



EpiData will automatically create the appropriate two files:

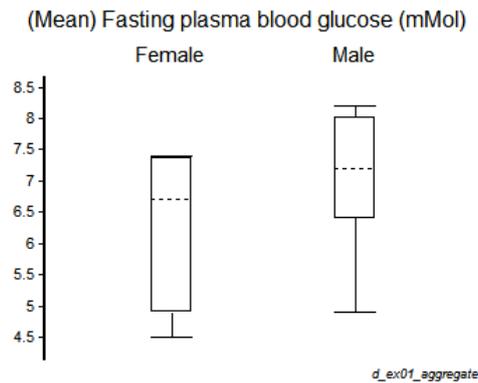
d_ex01_patient.epx
d_ex01_visit.epx

These files will be required in EpiData Analysis for merging.

EpiData Analysis

The final result of the analysis will be to obtain the following EpiData Analysis output:

Part 1 with AGGREGATE



Part 2 with transposing values

Overall microscopy result			
of 4 serial smears	Negative	Positive	Total
N-	2	0	2
NN-	1	0	1
NN9P	0	1	1
NP-	0	1	1
NPP-	0	3	3
PNP-	0	1	1
PP-	0	1	1
Total	3	7	10

Patient's sex				
Incremental yield of first 3 smears	Female	% Male	% Total	%
NNP	0 (0.0)	1 (25.0)	1 (14.3)	
NPx	3 (100.0)	1 (25.0)	4 (57.1)	
Px	0 (0.0)	2 (50.0)	2 (28.6)	
Total	3 (100.0)	4 (100.0)	7	

Percents: (Col)

It doesn't really look like much. Nevertheless, quite a few steps are needed to get from the source files D_EX01_PATIENT.EPX and D_EX01_VISIT.EPX to this point. We will elaborate on some considerations you have to make and offer hints for the sequential components in EpiData Analysis.

Merging the files

In EpiData Manager you made a relation through a unique identifier from the parent to the child file. In EpiData Analysis you must now merge these to files to get the following dataset.

We start by reading the child file:

```
read "childfile.epx"
merge parentidentifier /file="parentfile.epx" /table
```

To each record from the child file, information from the parent file is added repetitively for each observation for the same individual. In other words, you start from the other way around than in EpiData EntryClient, starting in EpiData Analysis with the child file and using the parent as a lookup table.

As a result we get (first 13 records only shown):

	idpat	visitid	visitdate	bs	sputum	micres	sex	marital	MergeVar
1	A	A-25-03-2007	25/03/2007	7.3	Salivary	Negative	Male	Married	In both
2	A	A-26-03-2007	26/03/2007	7.2	Purulent	1+ positive	Male	Married	In both
3	A	A-24-03-2007	24/03/2007	6.3	Mucoid	1+ positive	Male	Married	In both
4	B	B-24-03-2007	24/03/2007	4.9	Muco-purulent	Negative	Male	Divorced	In both
5	C	C-24-03-2007	24/03/2007	5.2	Purulent	Negative	Female	Single	In both
6	C	C-26-03-2007	26/03/2007	4.8	Muco-purulent	Negative	Female	Single	In both
7	D	D-25-03-2007	25/03/2007	7.4	Mucoid	2+ positive	Female	Widowed	In both
8	D	D-24-03-2007	24/03/2007	7.3	Blood-tinged	Scanty positive	Female	Widowed	In both
9	E	E-28-03-2007	28/03/2007	7.9	Purulent	2+ positive	Male	Married	In both
10	E	E-27-03-2007	27/03/2007	8.2	Not recorded	1+ positive	Male	Married	In both
11	F	F-01-04-2007	01/04/2007	7.7	Purulent	2+ positive	Female	Annulled	In both
12	F	F-31-03-2007	31/03/2007	7.2	Muco-purulent	3+ positive	Female	Annulled	In both

A variable MERGEVAR has been created by EpiData Analysis. It can take 3 values:

- 1 Only in memory (original)
- 2 Only in external file
- 3 In both

A frequency helps to identify quickly whether there are for instance any “orphans”, that is child records which do not find the same identifier in a parent record and are thus useless.

We note in the above that the visit dates are not temporally sequential within each parent identifier, we need thus sorting, first by parent identifier, then within them by date, thus:

```
sort idpat visitdate
```

However, this may not be correct in all circumstances.

Note: We commonly suggested to code unknown dates as “01/01/1800”. With sorting, the unknown date will thus come first (sorting is ascending by default) and this might not be desirable. In this dataset we don’t have unknown dates. But only because the small dataset allows us to see that, you would have to anticipate that possibility with a larger dataset and thus first make a new date variable where the unknowns take a value for a date in the future, so they appear with sorting at the end of the information on an individual.

Following the sorting, it might be advantageous to number the visits in order to be able to make a frequency on them to see what the maximum number of visits is (here we can see that it is a maximum of 4 visits, but in a large database, it would be more difficult).

To create a new variable EXAM and set its default initially to 1 (each record takes first the value 1), we have so far used the following grammar:

```
define exam #
exam=1
```

or alternatively the one-line alternative approach which accomplishes exactly the same thing:

```
gen i exam=1
```

Note: the command GEN will produce integer of length of 9. If you need a shorter and fixed field length integer, you must utilize DEFINE.

The command GEN replaces DEFINE and “i” stands for an integer field.

There are other similar commands (see an earlier Exercise):

```
gen f doorheight=1.85
gen d birthdate=dmy(31,12,1899)
gen s firstname="john"
```

for date fields (d) float (real number) fields (f), and string (text) fields (s). Look it up in the help file (type `gen+F1` in the command line).

Now that you have a new variable EXAM, how can you tell EpiData that it should look at the person (identified by an ID) and number each visit, starting with 1 until the next individual comes, when it must start again with 1. The command is:

```
if id=id[_n-1] then visit=visit[_n-1]+1
```

This looks admittedly complex. Let’s thus take it apart. `[_n]` identifies the current record and accordingly `[_n-1]` the immediately preceding record. Let’s say, EpiData Analysis has proceeded to record 547 and looks at the ID of this record. It looks whether record 546 had the same ID as record number 547: `if id=id[_n-1]`. If that is the case, then it should take the VISIT number of record 546 and add 1 to it for record 547: `visit=visit[_n-1]+1`. If it is not the case, then the default stays (which we defined as 1) and it moves on to the next record.

As a result, you should get:

	idpat	visitdate	visit
1	A	24/03/2007	1
2	A	25/03/2007	2
3	A	26/03/2007	3
4	B	24/03/2007	1
5	C	24/03/2007	1
6	C	26/03/2007	2
7	D	24/03/2007	1
8	D	25/03/2007	2
9	E	27/03/2007	1
10	E	28/03/2007	2
11	F	27/03/2007	1
12	F	31/03/2007	2
13	F	01/04/2007	3

the first variable column showing the identifier, the second the date of the examination, and the third the number of the examination for that individual.

Aggregating data

If we look at the sex (given the field name SEX), date of visit (given the field name VISITDATE), and blood sugar (given the field name BS):

	idpat	sex	visitdate	bs
1	A	Male	24/03/2007	6.3
2	A	Male	25/03/2007	7.3
3	A	Male	26/03/2007	7.2
4	B	Male	24/03/2007	4.9
5	C	Female	24/03/2007	5.2
6	C	Female	26/03/2007	4.8

SEX remains obviously the same (and is thus from the parent file), and is a categorical variable unique to the examined person, while BS varies by examination date and is a continuous variable. If we want to examine blood sugar by sex, there is no need to transpose the blood sugar values from the vertical to the horizontal as EpiData Analysis has inbuilt a tool to aggregate the data. For the individual and its sex we would write:

```
aggregate idpat sex /close
```

and get with BROWSE the ten individuals and their SEX:

	idpat	sex	N
1	A	Male	3
2	B	Male	1
3	C	Female	2
4	D	Female	2
5	E	Male	2
6	F	Female	3
7	G	Male	3
8	H	Female	3
9	I	Male	4
10	K	Female	1

We can expand this command using an option to calculate the MEAN of blood sugar for each individual:

```
aggregate idpat sex /mean="bs" /close
```

and get:

	idpat	sex	N	Nbs	MEAbs
1	A	Male	3	3	6.9333333333
2	B	Male	1	1	4.9000000000
3	C	Female	2	2	5.0000000000
4	D	Female	2	2	7.3500000000
5	E	Male	2	2	8.0500000000
6	F	Female	3	3	7.4333333333
7	G	Male	3	3	7.2333333333
8	H	Female	3	3	6.7000000000
9	I	Male	4	4	8.1500000000
10	K	Female	1	1	4.5000000000

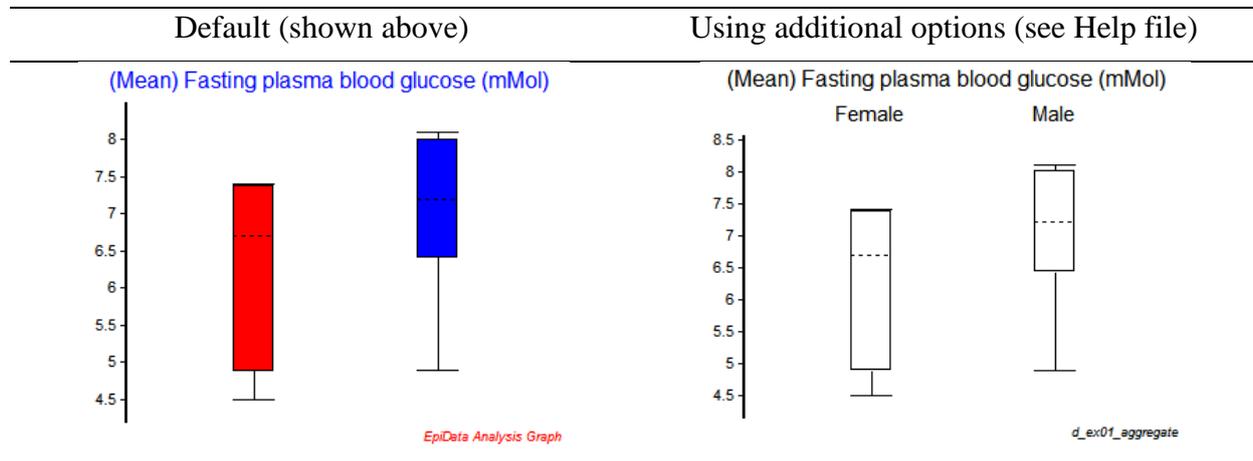
where MEAbs is the calculated mean value of blood sugar for an individual from all the individual's measurements. To save the aggregate data in a file, we add another option:

```
aggregate patid sex /mean="bs" /close /save="d_ex01_aggregate.rec" /replace
```

Having accomplished this, it is now straight forward to write:

```
cls
close
read "d_ex01_aggregate.rec"
boxplot meabs /by=sex
```

and get:



From long-to-wide

For showing means, there is thus no need to transpose data from the vertical to the horizontal. But it is different for the macroscopic aspect of sputum the examination results. We do not want (nor would we get a sensible result) aggregate sputum smear results. We wish to know each result from each individual.

The first thing before starting copying results from the vertical to the horizontal, we need to know the maximum number of examinations an individual had in the data set. We can get this with a frequency on the VISIT:

```
freq visit
```

Number of visit	
	N
1	10
2	8
3	5
4	1
Total	24

This shows that the maximum number of examinations an individual had in this dataset was 4. We need thus to prepare 4 new variables for each field that are in the sequence of the examination in the vertical but should also become part of each record.

Let's say we have a variable VAR1 with different values for each visit:

ID	VISIT	VAR1
B1	1	1
B1	2	3
B1	3	2
B1	4	2
C1	1	3
D1	1	2

What we need is:

ID	VISIT	VAR1	VAR11	VAR12	VAR13	VAR14
B1	1	1				
B1	2	3				
B1	3	6				
B1	4	2				
C1	1	3				
D1	1	2				

First, we make these 4 variables and give them all the default value of -1 (the minus one is a good way to see missing data and helps later in the selection):

```
gen i var11=-1
gen i var12=-1
gen i var13=-1
gen i var14=-1
```

After these four command lines, our above dataset becomes:

ID	VISIT	VAR1	VAR11	VAR12	VAR13	VAR14
B1	1	1	-1	-1	-1	-1
B1	2	3	-1	-1	-1	-1
B1	3	6	-1	-1	-1	-1
B1	4	2	-1	-1	-1	-1
C1	1	3	-1	-1	-1	-1
D1	1	2	-1	-1	-1	-1

and we are ready to copy the values from the vertical to the horizontal by respecting that the value of VAR1 from VISIT 1 goes to VAR11, the value from VISIT 2 to VAR12, etc.

For VISIT 1, the value for VAR11 is equal to the value of VAR1 and we make it thus the default:

```
VAR11=VAR1
```

Then we use the same approach as above to identify the record:

```
if id[_n]=id[_n+1] then var12=var1[_n+1]
```

This means that if the current record [_n] has the same ID as the next record [_n+1], then VAR12 in the current record should take the value of VAR1 from the next record [_n+1].

Now we do this for all possible 4 records (the maximum of VISITS):

```
if id[_n]=id[_n+2] then var13=var1[_n+2]
if id[_n]=id[_n+3] then var14=var1[_n+3]
```

All the lines to be written for this original field VAR1 are thus:

```
gen i var11=-1
gen i var12=-1
gen i var13=-1
gen i var14=-1
VAR11=VAR1
if id[_n]=id[_n+1] then var12=var1[_n+1]
if id[_n]=id[_n+2] then var13=var1[_n+2]
if id[_n]=id[_n+3] then var14=var1[_n+3]
```

and we get (assuming that the last patient D1 had only 1 VISIT):

ID	VISIT	VAR1	VAR11	VAR12	VAR13	VAR14
B1	1	1	1	3	6	2
B1	2	3	1	3	6	-1
B1	3	6	1	3	-1	-1
B1	4	2	1	-1	-1	-1
C1	1	3	3	-1	-1	-1
D1	1	2	2	-1	-1	-1

You may note that only for VISIT 1 for patient B1 with four visits all new 4 variables have all respective 4 values from the 4 VISITS.

Now we can safely get rid of the records of VISITS 2, 3, and 4 and we end up just with individuals who have all information from each VISIT:

```
select visit=1
```

and we get:

ID	VISIT	VAR1	VAR11	VAR12	VAR13	VAR14
B1	1	1	1	3	6	2
C1	1	3	3	-1	-1	-1
D1	1	2	2	-1	-1	-1

The conversion from “Long-to-wide” is thus successfully completed, well, for this variable. Of course, before you make this selection, you have to repeat the same approach for each field, so that in the end you get something like:

	idpat	sex	marital	visitdate1	visitdate2	visitdate3	visitdate4	sputum1	sputum2	sputum3	sputum4	micros1	micros2	micros3	micros4	pattern	case	yield
1	A	Male	Married	24/03/2007	25/03/2007	26/03/2007		Mucoid	Salivary	Muco-purulent		1+ positive	Negative	1+ positive		PNP	Positive	Px
2	B	Male	Divorced	24/03/2007				Purulent				Negative				N	Negative	
3	C	Female	Single	24/03/2007	26/03/2007			Muco-purulent	Purulent			Negative	Negative			NN	Negative	
4	D	Female	Widowed	24/03/2007	26/03/2007			Blood-tinged	Mucoid			Scanty positive	1+ positive			PP	Positive	Px
5	E	Male	Married	27/03/2007	28/03/2007			Not recorded	Muco-purulent			1+ positive	1+ positive			PP	Positive	Px
6	F	Female	Annulled	27/03/2007	31/03/2007	01/04/2007		Muco-purulent	Purulent	Muco-purulent		Negative	1+ positive	1+ positive		NPP	Positive	NPx
7	G	Male	Cohabiting	27/03/2007	28/03/2007	01/04/2007		Salivary	Mucoid	Purulent		Negative	1+ positive	1+ positive		NPP	Positive	NPx
8	H	Female	Married	31/03/2007	01/04/2007	02/04/2007		Mucoid	Purulent	Mucoid		Negative	1+ positive	1+ positive		NPP	Positive	NPx
9	I	Male	Separated	31/03/2007	01/04/2007	02/04/2007	03/04/2007	Salivary	Mucoid	Purulent	Mucoid	Negative	Scanty posi	Not recorded	1+ positive	NFP	Positive	NPx
10	K	Female	Married	02/04/2007				Muco-purulent				Negative				N	Negative	

You have now ten patients and you are at the point where you can continue to work in the same way as you used to work before. While it is much more complex to get to here from 1 line per examination than from 1 line per examinee, it is also obvious that in the end this is much more informative.

For each examination we have the date and for each specimen we have the quality of the sputum. In the Union / WHO approach you have only one date (the date of collection of the first specimen) for a series of three, and the quality of sputum in the Tuberculosis Laboratory is not that informative as it is very possible that every day the quality of the specimen is different, but there is no space allocated to write 3 different ones.

Tasks:

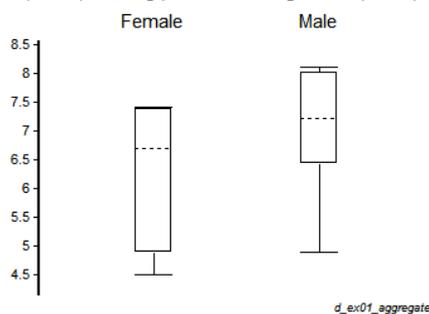
- *Prepare a data documentation sheet*
- *Prepare the EpiData Manager form for the relational database*
- *Enter the data from the following sample data set:*

Patient	Date of exam	Sex	Marital status	Blood sugar	Sputum	Result
A	24-Mar-2007	Male	Married	6.3	Mucoid	1+
B	24-Mar-2007	Male	Divorced	4.9	Muco-purulent	Neg
C	24-Mar-2007	Female	Single	5.2	Purulent	Neg
D	24-Mar-2007	Female	Widowed	7.3	Blood-tinged	2 per 100
A	25-Mar-2007	Male		7.3	Salivary	Neg
D	25-Mar-2007	Female		7.4	Mucoid	2+
A	26-Mar-2007			7.2	Purulent	1+
C	26-Mar-2007	Female		4.8	Muco-purulent	Neg
E	27-Mar-2007	Male	Married	8.2		1+
F	27-Mar-2007	Female	Annulled	7.4	Purulent	Neg
G	27-Mar-2007	Male	Cohabiting	6.9	Salivary	Neg
G	28-Mar-2007	Male		7.2	Mucoid	2+
E	28-Mar-2007	Male		7.9	Purulent	2+
F	31-Mar-2007	Female		7.2	Muco-purulent	3+
H	31-Mar-2007		Married	6.6	Mucoid	Neg
I	31-Mar-2007	Male	Separated	8.3	Salivary	Neg
H	1-Apr-2007	Female		6.9	Muco-purulent	1+
F	1-Apr-2007	Female	Engaged	7.7	Purulent	2+
I	1-Apr-2007	Male	Single	8.0	Mucoid	8 per 100
G	1-Apr-2007	Male		7.6	Muco-purulent	1+
K	2-Apr-2007	Female	Married	4.5	Purulent	Neg
I	2-Apr-2007	Male		8.2	Muco-purulent	
H	2-Apr-2007	Female		6.6	Mucoid	1+
I	3-Apr-2007	Male		8.1	Mucoid	1+

- Write a program *D_EX01.PGM* that merges the two files, then prepare sets for the aggregated data and for the “long-to-wide” transformation to produce the following output respectively:

From aggregating the data:

(Mean) Fasting plasma blood glucose (mMol)



From transformation “long-to-wide”:

"Table 1. Pattern of serial smear results"

Definition of case by microscopy			
of 4 serial smears	Negative	Positive	Total
N---	2	0	2
NN--	1	0	1
NF9F	0	1	1
NFP-	0	3	3
FNF-	0	1	1
FP--	0	2	2
Total	3	7	10

"Table 2. Incremental yield among positive results"

Incremental yield of	Patient's sex		
	Female	% Male	% Total
first 3 smears			
NFX	2 (66.7)	2 (50.0)	4 (57.1)
Fx	1 (33.3)	2 (50.0)	3 (42.9)
Total	3 (100.0)	4 (100.0)	7

Percentages: (Col)

Solution to Exercise 1: A relational database and “Aggregating” vs from “long-to-wide”

Key points:

- A relational database is the solution to a varying number of observations per individual
- The child file is merged with the parent file to give a dataset of all observations
- To obtain means for an individual from continuous variables, aggregating the data is the strategy of choice
- To reduce the dataset to individuals with information on each examination, one must copy the information from observations in the vertical to newly created fields in the first observation of each individual before selecting that record from the individual (“long-to-wide”)

Task

- *Prepare a data documentation sheet*

The documentation sheet is shown on the next page.

Task

- *Prepare the EpiData Manager form for the relational database*

The data entry forms may be made as follows:

D_EX01_PATIENT.EPX	D_EX01_VISIT.EPX
<p>Entry form for the patient</p> <p>Unique patient identifier <input type="text"/></p> <p>Patient's sex <input type="text"/> label_sex</p> <p>Marital status <input type="text"/> label_marital</p>	<p>Entry form for the visit</p> <p>Unique patient identifier <input type="text"/></p> <p>Unique visit identifier <input type="text"/></p> <p>Date of visit <input type="text"/></p> <p>Plasma glucose in mMol/L <input type="text"/></p> <p>Quality aspect of sputum <input type="text"/> label_sputum</p> <p>Microscopy result <input type="text"/> label_micres</p>

The data documentation sheet:

Data documentation sheet

	Field name	Field label	Field type	Field length	Field values	Value label	Field comment
Patient file	idpat	Unique patient identifier	U	1	A,...,Z		Any given unique ID
	sex	Patient's sex	I	1	1 Female 2 Male 3 Unknown		
	marital	Patient's marital status	I	1	1 Single 2 Married 3 Cohabiting 4 Annulled 5 Divorced 6 Widowed 7 Separated 8 Engaged 9 Not recorded		

NOTE: If SEX is given during an earlier examination and left empty in a subsequent examination, update the information to known
 If SEX is different in different examinations, record as UNKNOWN
 If MARITAL is given during an earlier examination and left empty in a subsequent examination, keep the initial information from earlier
 If MARITAL is different in different examinations, update to most recent information

Examination file	idvisit	Unique examination identifier	S	12	A-2007-01-31,...		Automatically calculated
	visitdate	Date of visit	dd/mm/yyyy	10	01/01/2007,...,31/12/2007 01/01/1800		Legal visit date recordings Enter if visit date is missing
	bs	Fasting plasma blood glucose (mMol)	F	4	2.5,...,19.9 99.9		Legal valid value Enter if blood sugar is missing
	sputum	Macroscopic sputum aspect	I	1	1 Mucoid 2 Purulent		



micres

Microscopy result

|

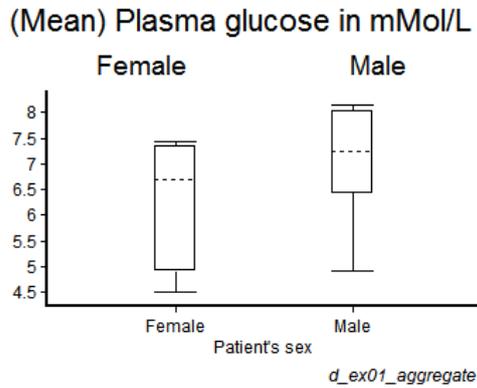
1

- 3 Muco-purulent
- 4 Blood-tinged
- 5 Salivary
- 9 Not recorded
- 0 Negative
- 1 "1+ positive"
- 2 "2+ positive"
- 3 "3+ positive"
- 4 "Scanty positive"
- 9 "Not recorded"

Task:

- Write a program *D_EX01.PGM* that merges the two files, then prepare sets for the aggregated data and for the “long-to-wide” transformation to produce the following output respectively:

From aggregating the data:



From transformation “long-to-wide”:

"Table 1. Pattern of serial smear results"

Definition of case by microscopy			
of 4 serial smears	Negative	Positive	Total
N---	2	0	2
NN--	1	0	1
NP9P	0	1	1
NPF-	0	3	3
PNF-	0	1	1
PP--	0	2	2
Total	3	7	10

"Table 2. Incremental yield among positive results"

Patient's sex			
Incremental yield of Female first 3 smears	% Female	% Male	% Total
NPx	2 {66.7}	2 {50.0}	4 {57.1}
Px	1 {33.3}	2 {50.0}	3 {42.9}
Total	3 {100.0}	4 {100.0}	7

Percents: {Col}

The *D_EX01.PGM* reads:

```

* Part D, Exercise 1
* Merging files and aggregating files
* Copying and transposing data from "Long-to-wide"

* Written by:    Hans L Rieder
* First version: 17 Jan 2010
* Last revision: 11 Nov 2016

cls
close
logclose

*****
* Merge child and parent files

cls
close

read "d_ex01_visit.epx"
merge idpat /file="d_ex01_patient.epx" /table

sort idpat visitdate
gen i visit=1
if idpat=idpat[_n-1] then visit=visit[_n-1]+1
label visit "Visit number"

savedata "temp_01.rec" /replace

*****
* Create an aggregate data set
* to determine means

cls
close
read "temp_01.rec"

```

```

aggregate idpat sex /mean="bs" /close /save="d_ex01_aggregate.rec" /replace

cls
close
read "d_ex01_aggregate.rec"

* set display databrowser=on
* browse
* tables sex meabs //Testing here only, will be done in Analysis
* boxplot meabs /by=sex //Testing here only, will be done in Analysis

*****
* Transpose , copy "long-to-wide"

cls
close
read "temp_01.rec"

* freq exam
* => Maximum is 4 visits

cls
gen d visitdate1
gen d visitdate2
gen d visitdate3
gen d visitdate4

visitdate1=visitdate
if (idpat[_n])=(idpat[_n+1]) then visitdate2=visitdate[_n+1]
if (idpat[_n])=(idpat[_n+2]) then visitdate3=visitdate[_n+2]
if (idpat[_n])=(idpat[_n+3]) then visitdate4=visitdate[_n+3]

cls
define bs1 ##.#
define bs2 ##.#
define bs3 ##.#
define bs4 ##.#

bs1=bs
if (idpat[_n])=(idpat[_n+1]) then bs2=bs[_n+1]
if (idpat[_n])=(idpat[_n+2]) then bs3=bs[_n+2]
if (idpat[_n])=(idpat[_n+3]) then bs4=bs[_n+3]

cls
gen i sputum1
gen i sputum2
gen i sputum3
gen i sputum4

sputum1=sputum
if (idpat[_n])=(idpat[_n+1]) then sputum2=sputum[_n+1]
if (idpat[_n])=(idpat[_n+2]) then sputum3=sputum[_n+2]
if (idpat[_n])=(idpat[_n+3]) then sputum4=sputum[_n+3]

cls
gen i micres1
gen i micres2
gen i micres3
gen i micres4

micres1=micres
if (idpat[_n])=(idpat[_n+1]) then micres2=micres[_n+1]
if (idpat[_n])=(idpat[_n+2]) then micres3=micres[_n+2]
if (idpat[_n])=(idpat[_n+3]) then micres4=micres[_n+3]

cls
label visitdate1 "Date of 1st visit"
label visitdate2 "Date of 2nd visit"
label visitdate3 "Date of 3rd visit"

```

```

label visitdate4 "Date of 4th visit"
cls
label bs1 "Plasma glucose at 1st visit"
label bs2 "Plasma glucose at 2nd visit"
label bs3 "Plasma glucose at 3rd visit"
label bs4 "Plasma glucose at 4th visit"
cls
label sputum1 "Macroscopic sputum of 1st visit"
label sputum2 "Macroscopic sputum of 2nd visit"
label sputum3 "Macroscopic sputum of 3rd visit"
label sputum4 "Macroscopic sputum of 4th visit"
labelvalue sputum1-sputum4 /1="Mucoid"
labelvalue sputum1-sputum4 /2="Purulent"
labelvalue sputum1-sputum4 /3="Muco-purulent"
labelvalue sputum1-sputum4 /4="Blood-tinged"
labelvalue sputum1-sputum4 /5="Salivary"
labelvalue sputum1-sputum4 /9="Not recorded"
cls
label micres1 "Microscopy result of 1st visit"
label micres2 "Microscopy result of 2nd visit"
label micres3 "Microscopy result of 3rd visit"
label micres4 "Microscopy result of 4th visit"
labelvalue micres1-micres4 /0="Negative"
labelvalue micres1-micres4 /1="1+ positive"
labelvalue micres1-micres4 /2="1+ positive"
labelvalue micres1-micres4 /3="1+ positive"
labelvalue micres1-micres4 /4="Scanty positive"
labelvalue micres1-micres4 /9="Not recorded"
cls
label marital "Civil status"

select visit=1
drop visitid mergevar visit

savedata "temp_02.rec" /replace

cls
close
read "temp_02.rec"

define restxt1 _
define restxt2 _
define restxt3 _
define restxt4 _

cls
restxt1="-"
if micres1>0 and micres1<9 then restxt1="P"
if micres1=0 then restxt1="N"
if micres1=9 then restxt1="9"
cls
restxt2="-"
if micres2>0 and micres2<9 then restxt2="P"
if micres2=0 then restxt2="N"
if micres2=9 then restxt2="9"
cls
restxt3="-"
if micres3>0 and micres3<9 then restxt3="P"
if micres3=0 then restxt3="N"
if micres3=9 then restxt3="9"
cls
restxt4="-"
if micres4>0 and micres4<9 then restxt4="P"
if micres4=0 then restxt4="N"
if micres4=9 then restxt4="9"

```

```

cls
define pattern ____
pattern=restxt1+restxt2+restxt3+restxt4
label pattern "Pattern of 4 serial smears"

* freq pattern

cls
define case #
                                case=0
if substr(pattern,1,1)="P" then case=1
if substr(pattern,2,1)="P" then case=1
if substr(pattern,3,1)="P" then case=1
if substr(pattern,4,1)="P" then case=1

label case "Definition of case by microscopy"
labelvalue case /0="Negative"
labelvalue case /1="Positive"

define yield ____
if substr(pattern,1,3)="N--" then yield="N99"
if substr(pattern,1,3)="NN-" then yield="NN9"
if substr(pattern,1,3)="NP9" then yield="NPx"
if substr(pattern,1,3)="NPP" then yield="NPx"
if substr(pattern,1,3)="PNP" then yield="Px "
if substr(pattern,1,3)="PP-" then yield="Px"
label yield "Incremental yield of first 3 smears"

keep idpat sex marital \
    visitdate1 visitdate2 visitdate3 visitdate4 \
    sputum1 sputum2 sputum3 sputum4 \
    micres1 micres2 micres3 micres4 \
    case pattern yield
savedata "d_ex01.rec" /replace

*****
* Produce tables on smear pattern and incremental yield

cls
close
read "d_ex01_aggregate.rec"

set option graph /sizex=400
set graph footnote="d_ex01_aggregate"

set echo=off
cls
boxplot meabs /by=sex /bw /sub="    Female                                Male"

cls
close
read "d_ex01.rec"

title "Table 1. Pattern of serial smear results"
tables case pattern
select case=1
title "Table 2. Incremental yield among positive results"
tables sex yield /c /PCT
set echo=on

*****
* Clean up

set echo=off
define yesno # global
cls

```

```
yesno=?Delete temporary files: 1=yes 0=no?
imif yesno=1 then
  erasepng /all /noconfirm
  erase "temp_01.rec"
  erase "temp_01.chk"
  erase "temp_02.rec"
  erase "temp_02.chk"
  erase "d_ex01_aggregate.chk"
  erase "d_ex01_aggregate.rec"
  select
  cls
  type "All temporary files erased" /h2
else
  select
  type "File D_EX01_EXAMINEE.REC remains open" /h2
endif
set echo=on
```

Exercise 2: A statistical process control chart

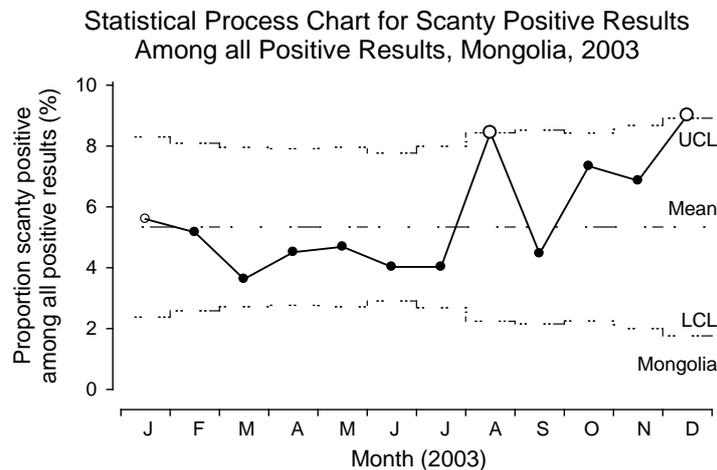
At the end of this exercise you should be able to:

- Aggregate data into the format required for a binomial outcome
- Create a statistical process control chart for a proportion

EpiData Analysis offers a variety of statistical process control (SPC) graphs. In this exercise we will deal with the determination of a proportion that changes over a period of time, and to what extent this variation deviates significantly from the expectation.

Let's assume that you have 1,200 observations during one year in a laboratory. Among these, 10 per cent (120) have positive result. We could determine the standard deviation and a confidence interval around the positive result or go one step further and take the average we expect for one month (12 of 100) with some measurement of uncertainty around this monthly estimate and then chart the actually observed monthly proportion. This would be a correct procedure if the expected monthly denominator is exactly one twelfth of the annual observation. However, this is rarely the case if ever and more likely is the scenario that the denominator varies in each month.

An SPC graph takes these fluctuations in the denominator into account and calculates the uncertainty as a function of the denominator in the element that is of interest (in this example the month). While there are some discussions on what is best to use, it has become customary to use 3 standard deviations as the upper and lower, so-called control limits. In the chart below, the proportion of scanty positive sputum smear microscopy results among all positive results is shown for Mongolia from the large laboratory register study over a one-year period:



Because the denominator differs in every month, the upper and lower control limits also differ from month to month.

We will be looking at examinations (not at examinees) and determine five different proportions (see below).

The dataset for the exercise

The dataset to be used in this exercise is the cleaned dataset of the four-country laboratory study which was provided in the solution to Part C, Exercise 1, dataset `C_EX01.REC` which is included as a “supplementary required file” with the current exercise with the name `MMUZ.REC`. `MMUZ.REC` is slightly different from `C_EX01.REC` in that the coding for the registration date has been corrected in a few records with an apparent error and the field `REGYEAR` has been removed.

To simplify the task (see specifics at the end), you will limit the analysis to the laboratories in Uganda and to tuberculosis suspects presenting for a diagnostic examination.

The time periods

The Uganda dataset contains information on three years and the unit of measurement of time will be the month. Because each month will thus appear three times (in each year), a new variable must be created that gives a sequential number for each month over the three-year period.

The outcome

The outcome is the monthly proportion of some type of positive results among either all smears or among positive smears.

There are five different proportions we might be interested in: 1) positive smears of any grade among all smears, 2) scanty positive smears among all smears, 3) low-positive smears (defined as either scanty positive or 1+ positive) among all smears, 4) scanty positive smears among all positive smears, and 5) low-positive smears among all positive smears.

Aggregating the data into the correct format

What is thus needed are counts of examinations with the characteristic (a scanty, a low-positive, or any positive result) among all examinations or among all positive examinations. To this end, we aggregate the data as detailed in Part B, Exercise 3.

Making a statistical process control (SPC) chart for proportions over time

The above aggregate file is all we need to get an SPC chart, the general command for which is:

```
pchart numerator denominator [time unit]
```

Proposed procedure in writing the program

It is proposed to split up the program into five distinct sequential procedures:

- 1) Make the basic dataset
- 2) Count all smears, all positive, all low-positive smears, and all scanty positive smears
- 3) Aggregate the months and sum up the smears in question
- 4) Make the SPC charts

1) Make the basic dataset

In the basic dataset we need to create the years and the months as separate variables and make the appropriate selections:

Month and year of recording: the registers were collected reporting laboratory results during one year up to three years between January 1999 and December 2003. When making a cross-tabulation of country versus registration year (create a field for registration year from REGDATE), we see that:

Year of registration	Country				Total
	Moldova	Mongolia	Uganda	Zimbabwe	
1999	0	0	17300	0	17300
2000	0	0	18662	0	18662
2001	0	0	18088	1213	19301
2002	0	149	0	29307	29456
2003	17725	22406	0	3958	44089
Total	17725	22555	54050	34478	128808

It is thus best to sequentially number all the 60 months from beginning to the end, even if at the end we will require only the three years covered by Uganda.

Select for diagnostic examinations, country, and range of months.

Save the dataset with a new name.

2) Count all smears, all positive, all low-positive smears, and all scanty positive smears

Create four new variables that count for each record respectively all smears, all positive smears, all low-positive smears and all scanty positive smears

3) Aggregate the months and sum up the smears in question and save them to four different files

The element to be aggregated is the month and for each month one must have the relevant smears (all, all positive, all scanty positive, all low-positive).

4) Make the SPC charts

The appropriate chart type for this binomial outcome is a PChart which has the format:

```
pchart numerator denominator time
```

Task

- Produce five PCharts to display the proportion of 1) positive smears among all smears, 2) low-positive smears among all smears, 3) scanty positive smears among all smears, 4) low-positive smears among all positive smears, and 5) scanty positive smears among all positive smears.

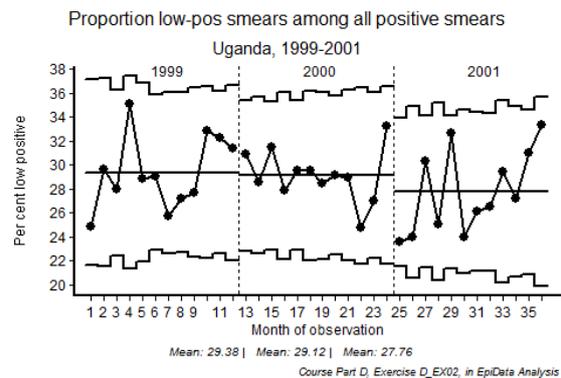
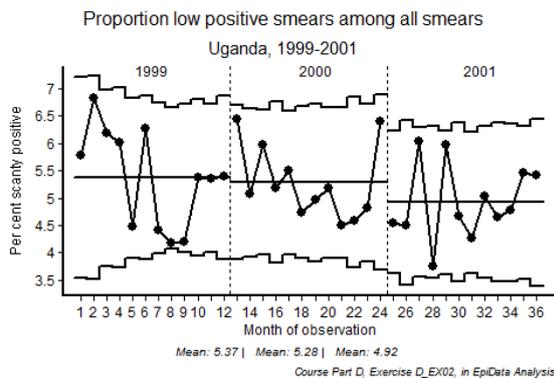
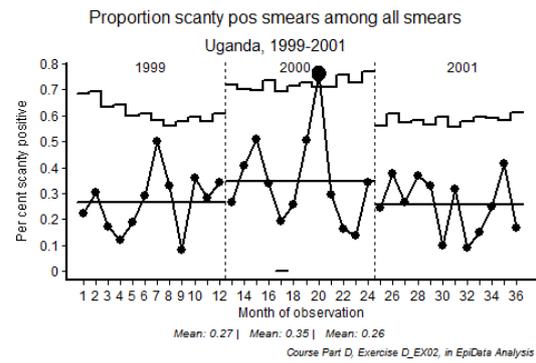
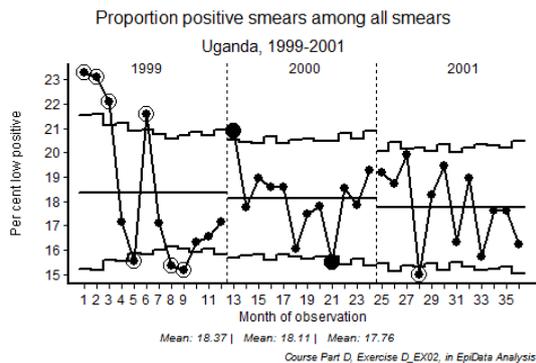
Solution to Exercise 2: A statistical process control chart

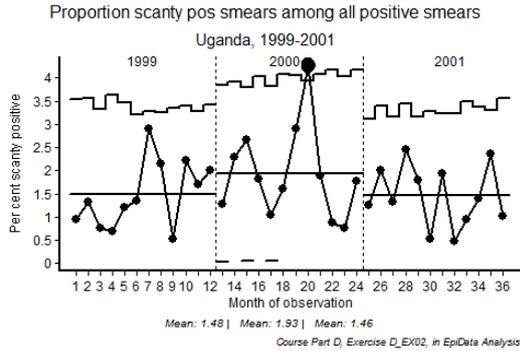
- Key points:**
- You must determine on how to aggregate data to obtain the numerator and denominator and, where necessary, the time components over the observation period
 - For a binomial outcome, a PChart is the appropriate SPC chart

Task

- Produce five PCharts to display the proportion of 1) positive smears among all smears, 2) low-positive smears among all smears, 3) scanty positive smears among all smears, 4) low-positive smears among all positive smears, and 5) scanty positive smears among all positive smears.

These are the five PCharts:





This is the program d_ex02.pgm that produced them:

```
* Part D, Exercise 2

* 1) Determine the proportion of positive
* smears among all smears
* 2) Determine the proportion of low-positive
* smears among all positive smears
* 3) Determine the proportion of scanty positive
* smears among all positive smears
* Definition of positive: any quantified positive
* or any scanty (quantified scanty or unquantified scanty)
* Definition low-positive: any smear which is 1+ positive or scanty positive

* Written by: Hans L Rieder
* First version: 26 Jun 2011
* Last revision: 29 Apr 2013

cls
close
logclose

*****
* Procedural steps
* 1) Make basic dataset
* 2) Start selection process
* 3) Aggregate data
* 4) Make SPC charts

*****
* 1) Prepare basic dataset

cls
close

read "mmuz.rec"

define regyear ####
regyear=year(regdate)
label regyear "Registration year"

cls
tables country regyear

gen i mmseq=0
if year(regdate)=1999 then mmseq=month(regdate)
if year(regdate)=2000 then mmseq=month(regdate)+12
if year(regdate)=2001 then mmseq=month(regdate)+24
if year(regdate)=2002 then mmseq=month(regdate)+36
if year(regdate)=2003 then mmseq=month(regdate)+48
label mmseq "Sequential month"

select reason=0
select country=3
select mmseq>0 and mmseq<37
* The selection above is not necessary for Uganda alone
* as there are no examinees in 2003

keep result1 result2 result3 regyear mmseq
savedata "temp_01.rec" /replace
```

```

*****
* 2) Count all smears, all positive, all scanty positive,
* all low positive smears

cls
close
read "temp_01.rec"

cls
* Count all smears
* Note: non-sensical sequences removed, thus simply:
gen i allsmears=1
if result2<>9 then allsmears=2
if result3<>9 then allsmears=3
label allsmears "Number of smears"

cls
* Count all positive quantified smears
* (include scanty not quantified)
gen i allpos1=0
if result1>0 and result1<4 then allpos1=1
if result1=5 then allpos1=1
gen i allpos2=0
if result2>0 and result2<4 then allpos2=1
if result2=5 then allpos2=1
gen i allpos3=0
if result3>0 and result3<4 then allpos3=1
if result3=5 then allpos3=1
gen i allpos=allpos1+allpos2+allpos3
label allpos "Number of positive smears"

cls
* Count all scanty smears
* (include scanty not quantified)
gen i scantpos1=0
* (include scanty not quantified)
if result1>0 and result1<1 then scantpos1=1
if result1=5 then scantpos1=1
gen i scantpos2=0
if result2>0 and result2<1 then scantpos2=1
if result2=5 then scantpos2=1
gen i scantpos3=0
if result3>0 and result3<1 then scantpos3=1
if result3=5 then scantpos3=1
gen i scantypos=scantpos1+scantpos2+scantpos3
label scantypos "Number of scanty positive smears"

cls
* Count all low positive smears
* (include scanty not quantified)
gen i lowpos1=0
if result1>0 and result1<2 then lowpos1=1
if result1=5 then lowpos1=1
gen i lowpos2=0
if result2>0 and result2<2 then lowpos2=1
if result2=5 then lowpos2=1
gen i lowpos3=0
if result3>0 and result3<2 then lowpos3=1
if result3=5 then lowpos3=1
gen i lowpos=lowpos1+lowpos2+lowpos3
label lowpos "Number of low positive smears"

keep regyear mmseq allsmears allpos scantypos lowpos
savedata "temp_02.rec" /replace
*****
* 3) Aggregate data

cls
close
read "temp_02.rec"

agg mmseq /sum=allsmears /sum=allpos /sum=scantypos /sum=lowpos /close
drop n nallsm1 nallpos nscantpos nlowpos
rename sumallsm1 to allsmears
rename sumallpos to allpos
rename sumscant1 to scantpos
rename sumlowpos to lowpos

```

```

drop n nallsm1 nallpos nscant1 nlowpos
savedata "temp_03.rec" /replace
*****
* 4) Make SPC charts

set option spc= /sizex=500 /sizey=350
set graph font size=9

cls
close
logclose

read "temp_03.rec"

set echo=off
pchart allpos allsmears mmseq /xtext="Month of observation" /bw \
  /ytext="Per cent low positive" \
  /ti="Proportion positive smears among all smears" \
  /sub="Uganda, 1999-2001" \
  /fn="Course Part D, Exercise D_EX02, in EpiData Analysis" \
  /b=12 /b=24 \
  /xlined=12.5 /xlined=24.5 \
  /t1 \
  /text="120,60,1999,0" \
  /text="260,60,2000,0" \
  /text="400,60,2001,0"

pchart scantpos allsmears mmseq /xtext="Month of observation" /bw \
  /ytext="Per cent scanty positive" \
  /ti="Proportion scanty pos smears among all smears" \
  /sub="Uganda, 1999-2001" \
  /fn="Course Part D, Exercise D_EX02, in EpiData Analysis" \
  /b=12 /b=24 \
  /xlined=12.5 /xlined=24.5 \
  /t1 \
  /text="120,60,1999,0" \
  /text="250,60,2000,0" \
  /text="400,60,2001,0"

pchart lowpos allsmears mmseq /xtext="Month of observation" /bw \
  /ytext="Per cent scanty positive" \
  /ti="Proportion low positive smears among all smears" \
  /sub="Uganda, 1999-2001" \
  /fn="Course Part D, Exercise D_EX02, in EpiData Analysis" \
  /b=12 /b=24 \
  /xlined=12.5 /xlined=24.5 \
  /t1 \
  /text="120,60,1999,0" \
  /text="260,60,2000,0" \
  /text="400,60,2001,0"

pchart lowpos allpos mmseq /xtext="Month of observation" /bw \
  /ytext="Per cent low positive" \
  /ti="Proportion low-pos smears among all positive smears" \
  /sub="Uganda, 1999-2001" \
  /fn="Course Part D, Exercise D_EX02, in EpiData Analysis" \
  /b=12 /b=24 \
  /xlined=12.5 /xlined=24.5 \
  /t1 \
  /text="130,60,1999,0" \
  /text="260,60,2000,0" \
  /text="400,60,2001,0"

pchart scantpos allpos mmseq /xtext="Month of observation" /bw \
  /ytext="Per cent scanty positive" \
  /ti="Proportion scanty pos smears among all positive smears" \
  /sub="Uganda, 1999-2001" \
  /fn="Course Part D, Exercise D_EX02, in EpiData Analysis" \
  /b=12 /b=24 \
  /xlined=12.5 /xlined=24.5 \
  /t1 \
  /text="120,60,1999,0" \
  /text="250,60,2000,0" \
  /text="400,60,2001,0"

set echo=on

```

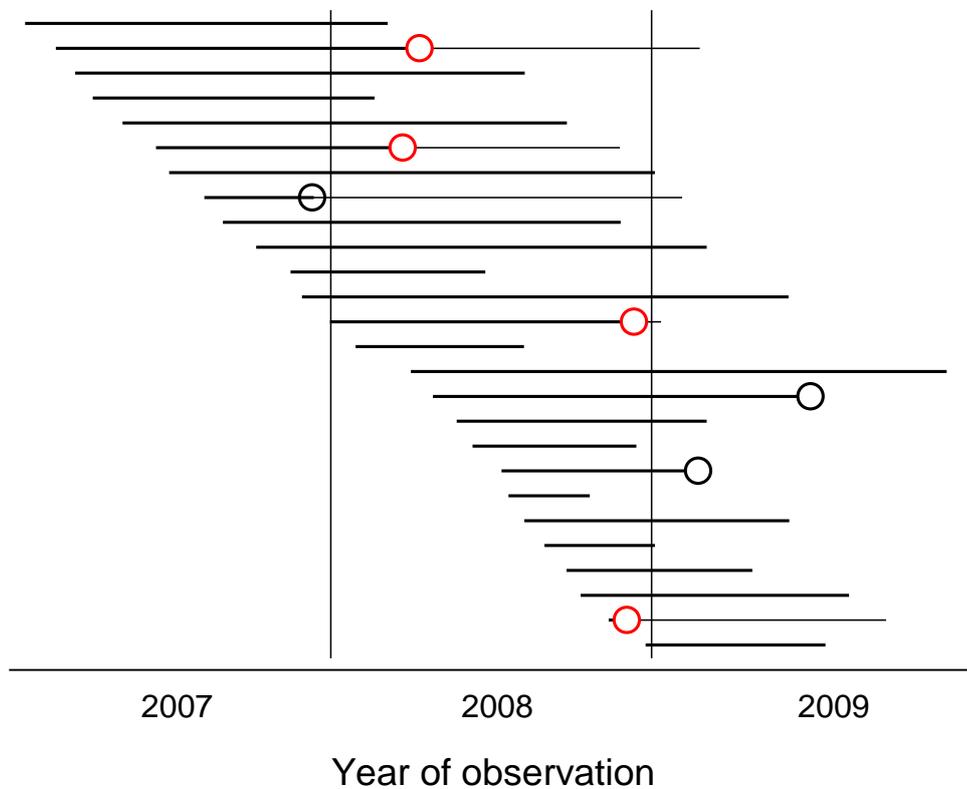
Exercise 3: Survival analysis

At the end of this exercise you should be able to:

- a. Understand the indications for survival analysis
- b. Be able to do a simple survival analysis in EpiData Analysis

An example of a survival analysis using the Kaplan-Meier method

We observe people over time, starting in 2007 into 2009 and note whether or not they develop an event (whatever it may be) during the observation time as shown for 26 individuals in the following graph:



We can think of this setting like of an institution, such as a prison, where inmates enter the prison and are discharged at some time. During incarceration some may develop tuberculosis. If our interest is focused on the year 2008, we have two measures of the magnitude of the problem.

We might calculate the incidence rate in the year 2008. The numerator is 4 cases. The denominator is person-time of observation. As the following table shows, we know the date of entry and exit from the institution, and the date the event occurs among those who had and event:

ID	Entry date	Exit date	Event	Event date	Obs start	Obs end	Obs days
A	20-01-2007	04-03-2008	No		01-01-2008	04-03-2008	63
B	24-02-2007	23-02-2009	Yes	11-04-2008	01-01-2008	11-04-2008	101
C	18-03-2007	07-08-2008	No		01-01-2008	07-08-2008	219

D	07-04-2007	18-02-2008	No		01-01-2008	18-02-2008	48
E	11-05-2007	24-09-2008	No		01-01-2008	24-09-2008	267
F	18-06-2007	24-11-2008	Yes	23-03-2008	--	--	
G	03-07-2007	02-01-2009	No		01-01-2008	31-12-2008	365
H	12-08-2007	03-02-2009	Yes	11-12-2007	01-01-2008	31-12-2008	365
I	02-09-2007	24-11-2008	No		01-01-2008	24-11-2008	328
J	10-10-2007	02-03-2009	No		01-01-2008	31-12-2008	365
K	18-11-2007	23-06-2008	No		01-01-2008	23-06-2008	174
L	01-12-2007	03-06-2009	No		01-01-2008	31-12-2008	365
M	02-01-2008	10-01-2009	Yes	11-12-2008	02-01-2008	11-12-2008	344
N	31-01-2008	06-08-2008	No		31-01-2008	06-08-2008	188
O	03-04-2008	30-11-2009	No		03-04-2008	31-12-2008	272
P	28-04-2008	05-07-2009	Yes	30-06-2009	28-04-2008	31-12-2008	247
Q	25-05-2008	02-03-2009	No		25-05-2008	31-12-2008	220
R	12-06-2008	12-12-2008	No		12-06-2008	12-12-2008	183
S	15-07-2008	02-03-2009	Yes	22-02-2009	15-07-2008	31-12-2008	169
T	23-07-2008	20-10-2008	No		23-07-2008	20-10-2008	89
U	10-08-2008	04-06-2009	No		10-08-2008	31-12-2008	143
V	02-09-2008	02-01-2009	No		02-09-2008	31-12-2008	120
W	27-09-2008	23-04-2009	No		27-09-2008	31-12-2008	95
X	13-10-2008	11-08-2009	No		13-10-2008	31-12-2008	79
Y	14-11-2008	23-09-2009	Yes	03-12-2008	14-11-2008	03-12-2008	19
Z	26-12-2008	15-07-2009	No		26-12-2008	31-12-2008	5

Days	4833
Cases	4
Cases/1000 person-days	0.828
Cases/100 person-years	30.2

Each person contributes person-time of observation, starting earliest from the beginning of the year 2008 or later if entry into the system was later. Person-time is contributed until exit from the institution or up to the point of the event if either happened in the year 2008. If the event happens later, observation time ends at the right-censoring point of 31 December 2008. Individual F developed the event in 2007 and although still in the system in 2008 does not contribute any person-time of observation in 2008. Individuals P and S developed the event only in 2009 and are thus not counted in 2008 and they contribute to person-time of observation through the end of the year 2008. Thus summed up, the 25 of the 26 inmates contributed 4,833 days of observation time which gives a case rate of 0.8 per 1,000 person-days of observation, or annualized, 30.2 cases per 100 observation years.

Survival analysis

The second way to look at the size of the problem is to ask what the probability for an individual is to “survive” the year 2008 without developing the event.

We exclude the person who had the event already in 2007 and sort the persons by the time of event or censoring (discharge or latest end of 2008). We note the number at risk of the

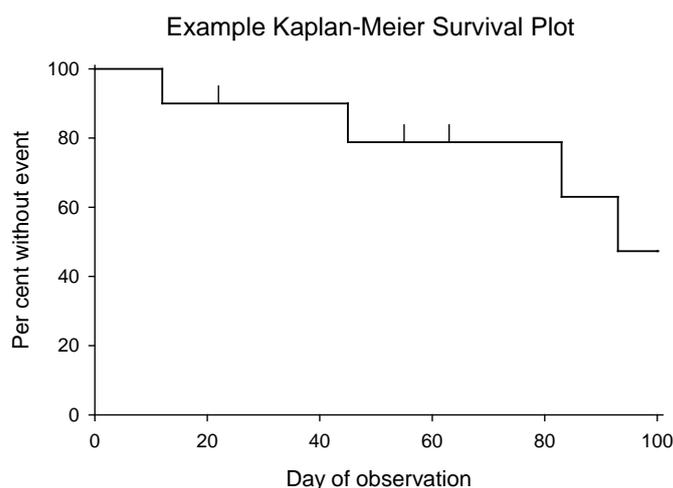
beginning of the interval, then the number who got censored, then how many were left after censoring, and finally how many get an event during the interval:

At risk at beginning	Censored	At risk at end	Event at end	Proportion surviving	Survival
25	1	24	0		
24	0	24	1	23/24	0.958
23	1	22	0		0.958
22	1	21	0		0.958
21	1	20	0		0.958
20	0	20	1	(19/20)*0.958	0.910
19	1	18	0		0.910
18	1	17	0		0.910
17	0	17	1	(16/17)*0.910	0.857
16	1	15	0		0.857
15	1	14	0		0.857
14	1	13	0		0.857
13	1	12	0		0.857
12	1	11	0		0.857
11	1	10	0		0.857
10	1	9	0		0.857
9	1	8	0		0.857
8	1	7	0		0.857
7	1	6	0		0.857
6	1	5	0		0.857
5	1	4	0		0.857
4	0	4	1	(3/4)*0.857	0.643
3	3	0	0		0.643

At each point where an event (not censoring) happens, we calculate the survival probability by dividing the number “surviving” after the event by the number at risk at the beginning of the interval when the event occurred.

You may note that censoring during an interval is assumed not to affect survival probability during that interval, but censored individuals are also subtracted from those at risk for the next interval.¹⁻³ This assumption is a simplification because censoring may, under certain circumstances, indeed be affecting the survival probability during the remaining interval. In any case, however, taking both the occurrence of the event and censoring into account for each interval following an event is a much more appropriate way to calculate survival than the proportion with an event among all who entered the cohort or by removing those censored from the cohort.

Graphically, we summarize the Kaplan-Meier survival probability as a step graph, sometimes showing the points when an individual was censored:



1. Kaplan E L, Meier P. Nonparametric estimation from incomplete observations. J Am Stat Ass 1958;53:457-81.
2. Bland J M, Altman D G. Survival probabilities (the Kaplan-Meier method). (Statistics notes). BMJ 1998;317:1572.
3. Fink S A, Brown R S, Jr. Survival analysis. Gastroenterol Hepatol 2006;2:380-3.

Survival analysis in EpiData Analysis

The command for a Kaplan-Meier survival analysis in EpiData Analysis is:

```
lifetable outcome interval
```

or

```
lifetable outcome startdate enddate
```

There are a multitude of options available both lifetable-specific options and general graph options. Use the Help file to test out various options until you have the survival plot that suits your needs.

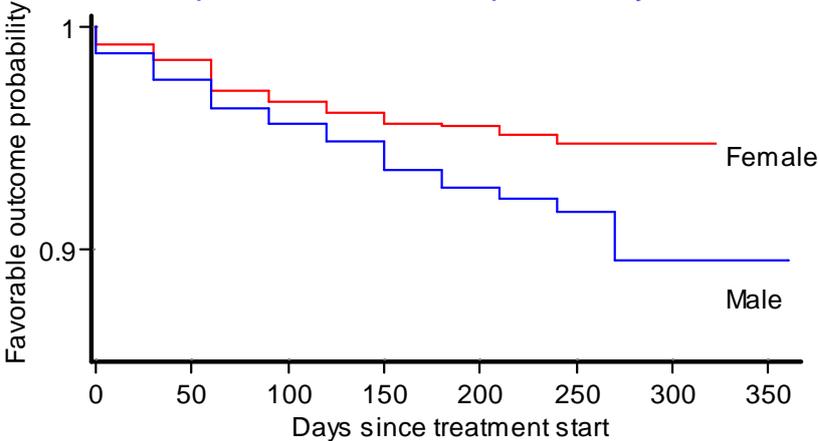
Tasks

The purpose of the exercise is to demonstrate quantitatively the probability of remaining without an event.

We use to this end a real dataset from a tuberculosis program, although we have removed any identifier and most variables, and retained to simplify your work only records of patients who have an exact date of treatment start and an exact date of treatment end. The dataset is provided as a supplementary file `d_ex03_required.rec`.

- 1) *Define a binomial outcome, where “favorable” is cured or treatment completed and all other outcomes are “unfavorable”*
- 2) *Do the lifetable analysis only for new sputum smear-positive cases and show the survival probability stratified by sex*

Kaplan-Meier survival probability



EpiData course: Exercise D_EX03

Solution to Exercise 3: Survival analysis

Key points:

- In this simplified example, there was no censoring, people either had the event or they didn't
- Survival analysis requires two variables, the time when the event occurred or the endpoint of observation, and whether there was an event or not

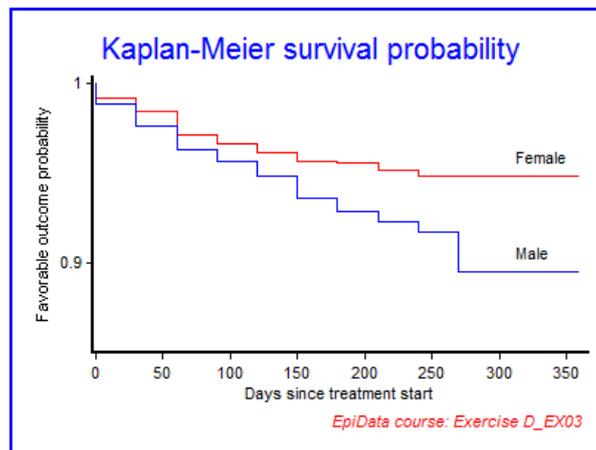
Tasks

The purpose of the exercise is to demonstrate quantitatively the probability of remaining without an event.

We use to this end a real dataset from a tuberculosis program, although we have removed any identifier and most variables, and retained to simplify your work only records of patients who have an exact date of treatment start and an exact date of treatment end. The dataset is provided as a supplementary file `d_ex03.rec_required`.

- 1) *Define a binomial outcome, where “favorable” is cured or treatment completed and all other outcomes are “unfavorable”*
- 2) *Do the lifetable analysis only for new sputum smear-positive cases and show the survival probability stratified by sex*

The graphic output produced by the program `D_EX03.PGM`:



The program `D_EX03.PGM` to produce the above graphic output:

- * Part D, Exercise 3
- * Survival analysis
- * Tuberculosis case register:
- * Treatment outcome

- * Written by: Hans L Rieder
- * First version: 30 Oct 2010
- * Last revision: 29 Apr 2013

```

cls
close
logclose

*****
* Create dataset

cls
close
read "d_ex03_required.rec"

define advout #
            advout=1 // adverse outcome
if outcome<=2 then advout=0 // favorable outcome
label advout "Treatment outcome"
labelvalue advout /0="Favorable"
labelvalue advout /1="Adverse"

gen i case=0
if sm00>0 and sm00<9 then case=1
select case=1
select sex<>9
select category=1

keep sex advout interval
savedata "temp_01.rec" /replace

*****
* Analysis

cls
close

read "temp_01.rec"

cls
set option graph=/sizex=600 /sizey=350
set graph font size=9
lifetable advout interval \
    /by=sex \
    /i=b30 /adj \
    /noc1 \
    /ymin=0.85 \
    /ymax=1 \
    /t \
    /ti="Kaplan-Meier survival probability" \
    /fn="EpiData course: Exercise D_EX03" \
    /xtext="Days since treatment start" \
    /ytext="Favorable outcome probability" \
    /text="335,90,Female,0" \
    /text="335,155,Male,0"

```

Exercise 4: Creating a menu for standard reports

At the end of this exercise you should be able to:

- a. Write an HTML-based interface for a menu
- b. Writing programs with interactive prompting

This exercise involves working with HTML to make a user-friendly menu-based interface allowing data entry and running EpiData Analysis programs on clicking which produce standard reports with interactive prompting for choices.

The end product interface will appear as follows:



Quarterly reports on case finding and treatment outcome



Enter data



Access EpiData Entry

Create reports



A quarterly report on case-finding



A quarterly report on treatment outcome

Terminate session



Erase temporary session files



Exit the report menu

We will structure the exercise as follows:

Preparatory work

- Installing EpiData Entry and EpiData Analysis

- Delete an obsolete sub-folder and create new sub-folders

- Unzip the required files from the course website into the relevant sub-folder

Background how EpiData Analysis starts and how to shape its looks when opening

Making the interface in HTML

- Using an HTML editor to make the skeleton of the interface

- Editing the HTML file

Making the EpiData Analysis programs

- The program to produce the quarterly report on case finding

- Make the basic dataset

- Make the interactive selection process

- Make the charts side by side

- The program to produce the quarterly report on treatment outcome

- Make the basic dataset

- Make the interactive selection process

Preparatory work

Installing EpiData Entry and EpiData Analysis

First make sure your “Normal EpiData Analysis” is updated to the newest version. The recommended location for both EpiData Entry and EpiData Analysis is C:\EPIDATA.

In addition, we will install EpiData Entry and EpiData Analysis into a separate folder (make it user-defined to keep control over what’s going to happen). You could also just copy the files and sub-folders in your C:\EPIDATA to C:\EPIDATA_REPORT. We simulate here what happens if you start from scratch. When prompted for the path, put it into:

```
c:\epidata_report
```

You will get in this folder three sub-folders and 25 files:

```
languages\  
samples\  
temp\  
English.ea.lang.txt  
Epdintro.pdf  
EPIDATA.CNT  
EpiData.exe  
EPIDATA.HLP  
epidata.ini  
EpiData.lbl  
epidatastat.10  
epidatastat.12  
epidatastat.15  
epidatastat.20  
EpiDataStat.exe  
epidatastat.ini  
epiout.css  
epiout_b.css  
epiout_w.css  
epiprint.css  
Francais.ea.lang.txt  
license.txt  
preventdouble.ea  
readme.rtf  
unins000.dat  
unins000.exe  
unins001.dat  
unins001.exe
```

Delete unnecessary sub-folders and create new sub-folders

The `samples` sub-folder is superfluous for this exercise and can be deleted. Instead create **four new** sub-folders, so that you have a total of **six** sub-folders:

```
images\  
languages\  
originals  
pgm\  
required\  
temp\  

```

```
English.ea.lang.txt
```

```
...
```

In the sub-folder “languages” delete:

~~en\~~
en\
~~fr\~~
images\

In the sub-folder “temp” you have (and can delete both):

~~docs\~~
~~examples\~~

Of the 25 files in the root you can delete all that are marked below:

English.ea.lang.txt
~~epidintro.pdf~~
EPIDATA.CNT
EpiData.exe
EPIDATA.HLP
epidata.ini
EpiData.lbl
epidatastat.10
epidatastat.12
epidatastat.15
epidatastat.20
EpiDataStat.exe
epidatastat.ini
epiout.css
epiout_b.css
epiout_w.css
epiprint.css
Francais.ea.lang.txt
license.txt
preventdouble.ea
readme.rtf
~~unins000.dat~~
~~unins000.exe~~
~~unins001.dat~~
~~unins001.exe~~

It is of course not necessary to delete all these files but it reduces package size (notably the uninstall files).

Unzip the required files from the course website into the relevant sub-folder

The course website contains a zip file with 23 required files:

epidata.ini
epidata.png
epidata_wikiL.png
epidatastat.png
eraser.png
next.gif
quit.jpg
sample_2004.chk
sample_2004.eix
sample_2004.qes
sample_2004.rec

```
sample_2005.chk
sample_2005.eix
sample_2005.qes
sample_2005.rec
sample_2006.chk
sample_2006.qes
sample_2006.rec
side_by_side_graphs.html
start_template.htm
uganda.chk
uganda.rec
union.jpg
```

Unzip all these 23 files into the EPIDATA_REPORT\REQUIRED sub-folder. Then move and over-write if necessary as follows:

Move the 7 image files from the EPIDATA_REPORT\REQUIRED sub-folder to the EPIDATA_REPORT\IMAGES sub-folder:

```
epidata.png
epidata_wikiL.png
epidatastat.png
eraser.png
next.gif
quit.jpg
union.jpg
```

Move the 14 EpiData files (plus an *.HTML file) from the EPIDATA_REPORT\REQUIRED sub-folder to the EPIDATA_REPORT\ORIGINALS sub-folder:

```
sample_2004.chk
sample_2004.eix
sample_2004.qes
sample_2004.rec
sample_2005.chk
sample_2005.eix
sample_2005.qes
sample_2005.rec
sample_2006.chk
sample_2006.qes
sample_2006.rec
side_by_side_graphs.html
uganda.chk
uganda.rec
```

Move from the EPIDATA_REPORT\REQUIRED sub-folder to the EPIDATA_REPORT\LANGUAGES\EN sub-folder the file:

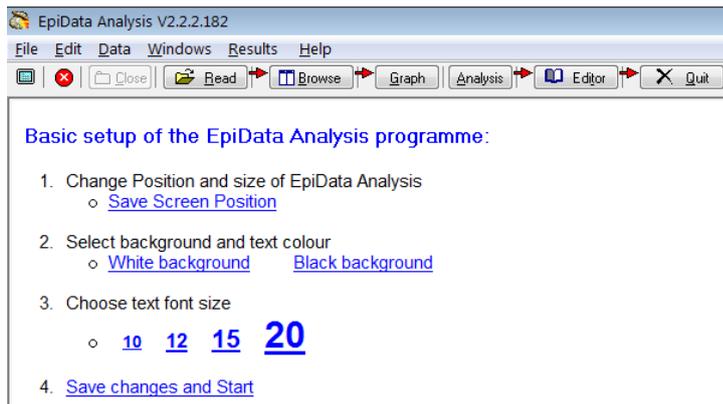
```
start_template.htm
```

Finally, move (and overwrite the existing) the:

```
epidata.ini
```

file to the root of the C:\EPIDATA_REPORT. And with that all “required” files should have been moved. Your “REQUIRED” folder should be empty and you can delete it.

Open EpiData Analysis by double-clicking its EpiDataStat.exe executable file and adjust and save font sizes when you see:



so that you end up with an empty screen, save the Windows position and exit EpiData Analysis.

Background how EpiData Analysis starts and how to shape its looks when opening

In any software you may use, there is an executable file that starts the process of opening the program and displaying the standard interface. In EpiData Analysis, this file is in the root of the folder in which you installed the program and its name is:

`EpiDataStat.exe`

This file contains all the essential code to direct EpiData Analysis to do what it is expected to do. Publishing this code will make EpiData Analysis what we call “open-source” software. While the designers and programmers of EpiData software are working on preparing this source code with sufficiently detailed documentation to ultimately make it open-source, the time is not yet quite mature to do so because the documentation must be un-ambiguous and clear for any other developer to derive usefulness for further development from it. As we are not software developer ourselves, we do not have any need to know the source code, the only thing we need to know is some very basic things that this executable file does, and how we can override certain things to meet our needs.

Among other files, the executable file looks first for an HTML file that is located in the `EPIDATA_REPORT\LANGUAGES\EN` sub-folder:

`start.htm`

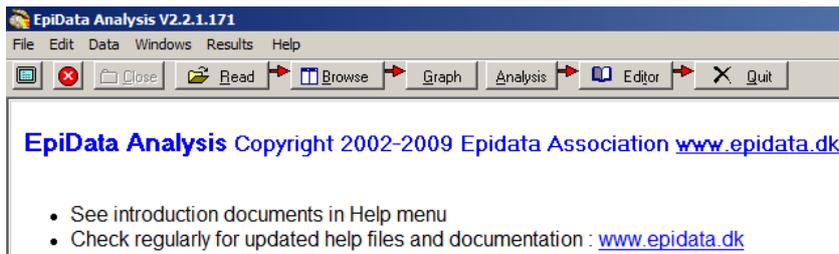
If we double-click this file it opens in our default browser and this is the display we get:

EpiData Analysis Copyright 2002-2009 Epidata Association www.epidata.dk

- See introduction documents in Help menu
- Check regularly for updated help files and documentation : www.epidata.dk

We can change this visualized part to our liking in the `start.htm` file which will be one of our tasks.

When we open EpiData Analysis, we see in addition the version display, the menu bar, and the process bar:

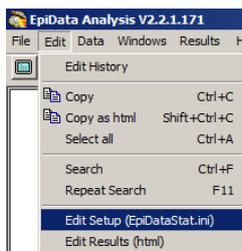


The upper component:



is part of the default defined in the executable file. It is, however, possible to suppress part or all of it by modifying the `EpiDataStat.ini` file.

With the opening of EpiData Analysis, a small program called `EpiDataStat.ini` is run. You can access it from the Edit menu:



As we have exited EpiData Analysis, we will now look for this `EpiDataStat.ini` file in the root of our folder and open it in our preferred text editor. Its initial default script is:

```
* EpiData Analysis default settings file
* Edit the next lines to change size or font
set echo=off

* Viewer font and size: (plus editor and help windows)
set browser font size =12
set graph font size =12
set viewer font size =12
set window font size =12
set editor font size =12
set viewer font name ="Verdana,Courier"

* uncomment next two lines to display Chinese Characters
*SET viewer font charset = "gb2312"
*set viewer font name="Arial Unicode MS"

*****
* Set options defined during installation:
* To see other set: issue "set" command or look in help file (F1)
*****
set display variables=OFF
set display databrowser=OFF
set output open=OFF
* Default folder defined as:
cd C:\EpiData_report\temp
```

```
set output folder="C:\EpiData_report\temp"
set language=english
set echo=ON
```

If we strip it of all comments, what remains is:

```
set echo=off

set browser font size =12
set graph font size =12
set viewer font size =12
set window font size =12
set editor font size =12
set viewer font name ="Verdana,Courier"

set display variables=OFF
set display databrowser=OFF
set output open=OFF
cd C:\EpiData_report\temp
set output folder="C:\EpiData_report\temp"
set language=english

set echo=ON
```

a series of SET commands that tells EpiData Analysis some defaults that are operative until we change. We asked you before to write over this file and that replacing file differs from the above (comments that remain are not shown) is shown in red font:

```
1 set echo=off
2 cd temp
3 set display mainmenu      =off
4 set display command prompt=off
5 set display worktoolbar  =off
6 set viewer font size =12
7 set window font size =12
8 set editor font size =12
9 set viewer font name ="Verdana,Courier"
10 set display variables=OFF
11 set display databrowser=OFF
12 set output open=OFF
13 set output folder="..\temp"
14 set language=english
15 set echo=on
16 set start page    ="..\languages\en\start_tb.htm"
```

If we rearrange a bit to put similar things together, we may summarize as:

```
set echo=off

cd temp

set start page    ="..\languages\en\start_tb.htm"
set output folder="..\temp"

set viewer font size =12
set window font size =12
set editor font size =12
set viewer font name ="Verdana,Courier"
set language=english
```

```
set display variables      =off
set display databrowser   =off
set display mainmenu      =off
set display worktoolbar   =off
set display command prompt=off
set output open           =off
```

```
set echo=on
```

We turn the echo off and on respectively at the beginning and at the end, so that we don't notice that it is run.

We then direct EpiData Analysis to go to the (above created) sub-folder EPIDATA_REPORT\TEMP. EpiData Analysis will be in this place when you start with an analysis and if you need to access data files or a program file or any other file, you will need to tell it where these are located *relative to this location*.

Relative locations

This might be an opportune time to introduce the concept of relative and absolute path. The path:

```
C:\epidata_report\languages\en
```

is an **absolute** path. It defines the drive ("C:\") and where within this drive one finds the folder and its named sub-folders. If we were to write this into the EpiDataStat.ini file, it would be alright as long as both is true, the drive and the path. If we would give the "package" with the "epidata_report" to somebody else it would therefore only work if that person were to copy it also into the root of his or her PC and if that main drive actually had the name "c:\". It wouldn't work from a USB drive for instance. However, if we would give the "package" to a colleague and inside the EpiDataStat.ini file all file locations were given relative to the "container" "epidata_report", then it would work from any location on the PC or indeed from an external drive with another drive designation than "c:\". We could go one step further, and say that we don't even want to name "epidata_report" as "epidata_report", so that a user is entirely free to give any name to this container and it would still be working.

What we need is a folder separator, and this is the backslash ("\") and the replacement indicator for the parent folder (directory), which is the double period (".."). If we thus write:

```
"..\temp\sometext.txt"
```

we refer to a file "sometext.txt" which is located (absolute path) in a folder temp which in itself is located in another folder that is not named. If we are in that other folder (whatever its name might be) than we refer relatively to it here and have no need to name it. This is precisely what we are doing here with this:

```
set start page = "..\languages\en\start_tb.htm"
```

We are in EpiData Analysis which is in the folder epidata_report. In this folder we have a sub-folder "languages", directly one level below the "epidata_report" and we can thus replace the parent directory "epidata_report" with ".." and insert the "\" as the folder separator followed by the designation of the "languages" sub-folder. This approach allows us to give the parent directory any name we wish: all that must be guaranteed

is that the sub-folder has a fixed position relative to the parent folder. Let's evaluate how we can move around within our "container" epidata_report. When we start EpiData Analysis from the EpiDataStat.exe file, we have our 6 folders at the same level, one of them being the temp folder. We get first into this folder with:

```
cd temp
```

To get back from within this folder to the root (epidata_report), we would type:

```
cd ..
```

indicating that with the two periods that we want to go to the parent folder. Being in the parent folder now, we wish to go to the sub-folder en of the languages folder:

```
cd languages\en
```

Being now in the en sub-folder and wishing to go to the sub-folder originals of the "container", we can do it in 3 steps:

```
cd ..  
cd ..  
cd originals
```

The first line gets us to the parent of en, i.e. to languages, the second gets us to the parent of languages, i.e. epidata_report, and the third line directs us to the sub-folder originals of epidata_report. Simpler, we can write this in a single line:

```
cd ..\..\originals
```

We have entered a path to a non-existing file:

```
set start page = "..\languages\en\start_tb.htm"
```

While it is now clear why we use this designation for the path, we also note that the file "start_tb.htm" does not exist in that folder. In the folder en we currently have:

```
start.htm  
start_template.htm  
start0.htm  
startfont.htm
```

It is then in a next step that we will create the "start_tb.htm" file. EpiData Analysis will open despite this error, simply by using the default file "start.htm" which is in this folder and give us:



At the bottom, you see don't see the command line anymore, and only the sub-folder in which EpiData Analysis is after executing the first four lines:



The default style sheet that EpiData Analysis uses is the `output.css` file, a cascading style sheet. You could make the style in the `start.htm` file, but the recommendation of the World Wide Web Consortium (W3C) is to refer in the main file to another specific (`*.CSS`) file that defines the style. This recommendation is for good reasons: you can always change the style sheet without changing the main HTML file.

You can make your own style sheets but this will require learning a bit more on how to make one, and this not subject of this exercise.

EpiData Analysis writes log files and other stuff that are useful to review when something goes wrong. These files are not usually used when all goes as expected and to get them out of the way, the command line:

```
set output folder="..\temp"
```

redirects the output to be stored in the sub-folder TEMP.

With this brief and rather simplified introduction of how EpiData Analysis starts to work, you should now have an understanding on what the `EpiDataStat.ini` file does and how you can always access it and change it on the fly. In your “regular” EpiData Analysis program, it will often prove useful to direct it with this file to a specific project folder on which you are currently working with even a series of CD commands as long as the folders exist and you know that the last CD command in a sequence overwrites all previous ones.

After we are now done with the role of the `EpiDataStat.ini` file, we have to deal with the HTML file `start_tb.htm`, which gets us into learning some basics about the HTML language.

Making the interface in HTML

Using an HTML editor to make the skeleton of the interface

HTML is the language of the Internet which is interpreted by a browser such as the proprietary Internet Explorer™ or the open-source and free Firefox® to make it visually comprehensible and appealing for the user. A simple text that looks like:

This is a simple text to show the browser interpretation of HTML text and the actual underlying HTML.



We add above an icon for display.

has the following underlying HTML code in the browser:

```
1 <!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01//EN" "http://www.w3.org/TR/html4/strict.dtd">
2
3 <html>
4
5 <head>
6 <meta content="text/html; charset=ISO-8859-1" http-equiv="content-type"><title>temp</title>
7 </head>
8
9 <body>
10 <big><span style="font-family: Arial;">This is a simple text to show the browser</span>
11 <br style="font-family: Arial;"><span style="font-family: Arial;">interpretation of HTML text and the actual</span>
12 <br style="font-family: Arial;"><span style="font-family: Arial;">underlying HTML.</span></big><br>
13 <br>
14 <big><span style="font-family: Arial;">We add above an icon for display.</span></big>
15 </body>
16
17 </html>
```

This is complex at first look, but when we look at it carefully then we realize that it is a highly logical language and sequence of instructions to the browser.

In Line 1, information is provided that the language conforms with W3C standards and the version of HTML it is using and that you can find this confirmed at the W3C website.

Every component in an HTML document has the principle to define the beginning and the end of the component and the system is the same for all. In the above example we have:

Begin	End
<HTML>	</HTML>
<head>	</head>
<body>	</body>

The structure indicates that everything between the opening and ending HTML tags is in fact HTML. Embedded are two parts, the head and the body, each indicating where it starts and where it ends. Every HTML page is build around these key parts, and within these you may have other sub-components imbedded that follow the same principles such as here within the body:

Begin	End
	

For the time being, we will leave it at that but will come back later to these principles when we work specifically on the `start_tb.htm` file.

Because of the complexity for the beginner, a multitude of software has been developed allowing the user to write normally as in a word processor to see what one actually wants to get. The software translates it into HTML and the page becomes interpretable by the browser. The advantages of such software are obvious but the downside is that it is often very expensive (hundreds of Euros perhaps), and not all is adhering strictly to W3C standards which will make it difficult for some browsers that require strict adherence to interpret the language properly.

You may have heard about the Mozilla Foundation which produces free and excellent open-source software, such as the browser Firefox, the email client Thunderbird, the calendar Sunbird and yes, the HTML editor Nvu. Nvu is still in its infancy and has some problems, some of which have been resolved with the HTML editor KompoZer which you find on this course web in the software section and that we will be using.

You do not need to install KompoZer, it is just a zip file. Unzip it into the root of your hard drive and you get a folder:

```
KompoZer 0.7.10\
```

You may make a shortcut to its executable file:

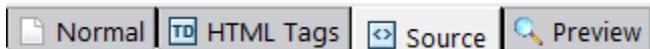
```
KompoZer.exe
```

to your desktop and if desired to the quick launch bar to have it accessible at your fingertip with one click away or simply look for it and double-click the `KompoZer.exe` file.

Access KompoZer and find the `start_template.htm` file in the `EPIDATA_REPORT\LANGUAGES\EN` sub-folder of the project. It is an empty page with a title:



That it is not quite empty becomes clear if you tick at the bottom to “Source”:



In fact, we have taken the normal `start.htm` file that comes with the installation of EpiData Analysis, have stripped it down and added a few essentials to prepare it for an interactive menu (template courtesy, Jens M Lauritsen, April 2008) and saved it under this `start_template.htm` file name.

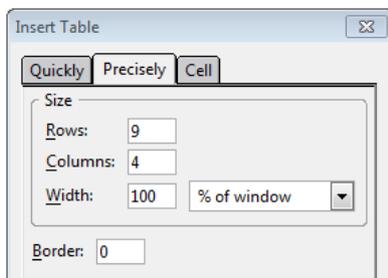
The first thing is to save it as:

`start_tb.htm`

Remember that we direct the `EpiDataStat.ini` file to go to this file when initiating:

```
set start page="..\languages\en\start_tb.htm"
```

You will be working in the “Normal” tab and the first thing we do is to insert a table with 4 columns and 9 rows (you can count them in the screenshot of the final interface shown at the beginning). Choose “Precisely”:

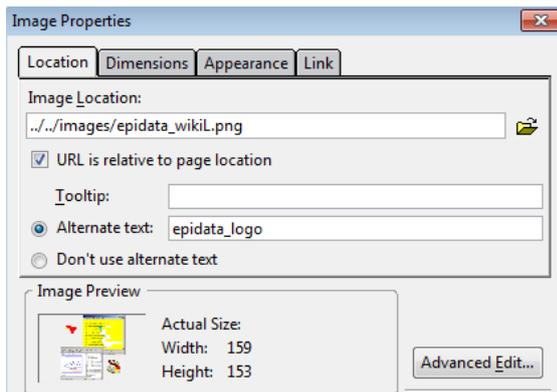


and set the Border to “0”. The default for “Border” is 1: a line around each cell is displayed in the browser. Setting it to “0” allows entering the information into cells instead of using the Tab key (which we cannot do properly in an HTML page), but the user does not see any line and remains unaware that we used a table.

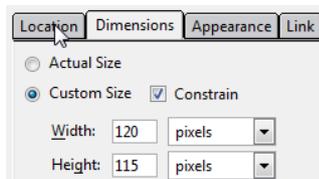
If the table cell background is not white, you may have to fiddle with the options for them to render them white (non-transparent). The best way is to change the background as follows:

1. Put the cursor into the top left cell
2. Right-click, choose “Table select” | “All cells”
3. Right-click again, choose “Table or Cell Background Color”
4. Pick the color (in our case “white”)

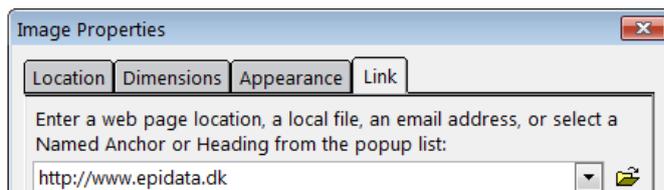
In the most upper left cell we insert one of the provided EpiData logos (the `epidata_wikiL.png` file from the `\IMAGES` sub-folder) using the “Insert” menu and also supply a (required) alternate text for the name:



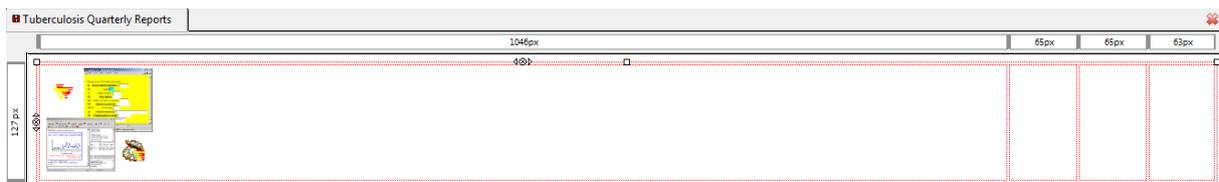
Then go to the tab “Dimensions” and decrease the current size of the width to 120 pixels:



In the “Link” tab add the URL of the EpiData website:



Then accept and you get:



Never mind for now the distortion of the column width. Add the Union logo into the most upper right cell and make a link to the Union website (<http://www.theunion.org>) in the same manner.

Join the central two cells of the first row, add the text and format it (see beginning of this Exercise) to get:



Continue adding text, formatting it, and add the appropriate icons into the appropriate places until you have the complete lay-out:

	<h1>Quarterly Reports on Case Finding and Treatment Results</h1>	
	Enter Data	
	Access EpiData Entry	
	Create Reports	
	A quarterly report on case finding	
	A quarterly report on treatment results	
	Terminate session	
	Erase temporary session files	
	Exit the menu	

Make sure to save the file. Perhaps you want to look at the source code and see that a lot has been added. We will not further edit the source code here, we will do this in a text editor. Thus exit KompoZer, this is all we are going to do with it.

If you now click the `EpiDataStat.exe` file, you will get:



Of course, there is no functionality yet (except the EpiData and Union web site icons if you are on the internet). In the next step we add functionality to the other icons.

Editing the HTML file

Windows has an inbuilt text editor, NotePad™. There are several free text editors available that are more flexible and powerful than this one. We have selected Crimson Editor® as our choice for this course. This free text editor that has several powerful features such as showing HTML code in colors and providing the very useful feature for rectangular selections.

Open now in this text editor the `start_tb.htm` file. What you see may seem a bit overwhelming and we will thus take it apart into manageable pieces to understand what we find there and to edit it where appropriate.

You have seen before the first few lines:

```

1 <!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN" "http://www.w3.org/TR/html4/strict.dtd">
2 <html><head>
3 <meta content="text/html; charset=ISO-8859-1" http-equiv="Content-Type">
4 <meta name="copyright" content="EpiData Association">
5 <meta name="description" content="Introduction to EpiData reports"><title>Tuberculosis Quarterly Reports</title>

```

They inform us that we deal with an HTML file and that the head starts. If we collapse (and hide from view) lines 9 to 28, we see up to line 28:

```

1 <!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN" "http://www.w3.org/TR/html4/strict.dtd">
2 <html><head>
3 <meta content="text/html; charset=ISO-8859-1" http-equiv="Content-Type">
4 <meta name="copyright" content="EpiData Association">
5 <meta name="description" content="Introduction to EpiData reports"><title>Tuberculosis Quarterly Reports</title>
6
7 <style>
8 <!--
28 </style><!--<link href="epiout.css" rel="stylesheet" type="text/css" media="screen">--></head>

```

You note here the beginning and end tags for HEAD and those for STYLE embedded inside these:

```
<head><style>...</style></head>
```

Make two hard carriage returns after the closing `</head>` tag, so we see a bit better what we have next. Make another hard carriage return after the end of:

```
<body class="bodywhite">
```

so that we separate for visibility the beginning of the table definition:

```
33 <table style="text-align: left; width: 100%; border="0" cellpadding=...
```

Can you see where our first row begins and where it ends?

```

<tr><td style="background-color: white;"><a href="http://www.epidata.dk"></a></td><td colspan="2" rowspan="1" style="background-
color: white; text-align: center; width: 957px;"><big><big><big><big>Quarterly Reports on Case
Finding and Treatment Results</big></big></big></big></td><td style="background-color:
white;"><a href="http://www.theunion.org"></a></td></tr>

```

The row begins with the opening tag `<tr>` and its ending tag `</tr>`. Each cell embedded in the row has its beginning and end tags `<td ...></td>`:

	Begin tag	End tag
Row	<code><tr></code>	<code></tr></code>
Cell	<code><td></code>	<code></td></code>

If we isolate the first cell with the logo of EpiData, we have thus:

```

<tr><td style="background-color: white;"><a href="http://www.epidata.dk"></a></td>

```

```
<a href ... </a>
```

is the opening about everything dealing with this logo: first the link to the web site, then information about the cell style, the dimensions of the logo, the alternate text, and finally the reference to the name of the image and the place where it is relative to the current position:

```
src="../../../images/epidata_wikiL.png"
```

Thus, in summary:

The definition of the image is defined as:

```
src=
```

and the image itself is given with its name relative to the location it had when you inserted it:

```
"../../../images/epidata.png"
```

You may note here the use of forward slashes (“/”) instead of the backward slashes (“\”) that we have go used to on the PC. HTML as a web language uses forward slashes to indicate path.

However, in everything we are going to write in this `start_tb.htm` file, we can do with the backward slash that we are accustomed to as there are situations where both are valid. We have here no reason to change what we know that it works. On the other hand, we are also not going to change any forward slash to a backward slash if a forward slash had been inserted into the document in KompoZer or had already been present in the initial template.

We have no links to any of the programs that should be run when the user clicks on the EpiData Entry image defined as:

```
src="../../../images/epidata.png"
```

because there is not yet any instruction on what to do here. We are going to do that in the next step.

Opening EpiData Entry from within EpiData Analysis

The task is to open EpiData Entry from within EpiData Analysis. To this end, we write HTML code associated with the icon for EpiData Entry:



and add the code in the correct place. Locate the HTML code that gives information about the embedding of this icon:

```

```

(You may not have `border="0"`, this removes the blue lines around the icon that designates it as a hyperlink).

Actually, the above information about the icon is located in one single cell, within the tags

```
<td> ... </td>:
```

```
<td style="background-color: white;"><td style="background-color: white; width: 42px;"></td>
```

The instruction to access EpiData Entry and, once there, to open a specific REC file is placed immediately before the definition of the icon . The following text is placed there:

```
<a href="../../../epidata.exe ../originals/sample_2006.rec">
```

The `href` is a hyperlink to a reference, here to “`epidata.exe`”, the executable file that opens EpiData Entry. Why the “`../../../`”? The `start_tb.htm` is located two levels down (`\languages\en`), therefore the instruction must be to go two levels up to be in the root of the “`epidata_report`” as the `epidata.exe` file is in the root. Note that after the `\epidata.exe` there is a space before `../originals`. The reason is that the first part refers to starting the software, while the second is to open a specific file that is located in `epidata_report\originals` and to keep it independent of the name of the container name we replace “`epidata_report\`” with “`..\`”. The file we are requesting to be opened, `sample_2006.rec`, is located one level down in the `EPIDATA_REPORT\originals` folder.

After we insert the above code, we must also place an ending `` tag to close the opening `<a href` before the cell closes with `</td>`:

```
<a href="../../../epidata.exe ../originals/sample_2006.rec"><img src=
"../../images/epidata.png" alt="" border="0" style="width: 32px; height:
32px;"></a></td>
```

Test functionality after saving.

Writing HTML code to invoke an EpiData Analysis program when clicking on the icon

Quite similar to the above, we locate now the HTML code for the EpiData Analysis icon:

```

```

Again as above, we need to insert HTML code immediately preceding the `<img src=...`. What is different is that we are already in EpiData Analysis, so it is not necessary to execute it (it would actually give an error if the default is set to run only one copy of EpiData Analysis at a time). All we need to do is to make a hyperlink to an EpiData Analysis program, specifically here to the program that should produce the first quarterly report on case finding. We will call the program “`quarterly_report_1.pgm`” and it will be located in the folder `PGM`. The command to run a program is “`run`”. We would thus think that:

```
href=run "../pgm\quarterly_report_1.pgm"
```

should do it. It is, however, not quite sufficient as the interplay between HTML and EpiData Analysis requires two additional things:

- 1) an opening command “`epi:`” to give the indicate that EpiData Analysis instructions will follow
- 2) everything related to EpiData Analysis must be placed between single quotes.

Accordingly, the above becomes:

```
href='epi: run "..\pgm\quarterly_report_1.pgm"'
```

We will add more EpiData Analysis commands, each separated by a semi-colon, like:

```
href='epi: CLS; set echo=off; run "..\pgm\quarterly_report_1.pgm"'
```

Once the program has run and produced the output, the user should get an instruction to press F8 which refreshes the start_tb.htm file (thus gets the user back to the main menu interface):

```
href='epi: CLS; set echo=off; run "..\pgm\quarterly_report_1.pgm" type "F8: Go back to menu"'
```

Including the opening <a href and the closing tag and showing the entire content of the cell, we then get:

```
<td style="background-color: white;"></td></tr><tr><td style="background-color: white;"></td><td style="background-color: white; width: 42px;"><a href='epi:CLS; set echo=off; run "..\pgm\quarterly_report_1.pgm"; type "F8: Go back to menu"'></a></td>
```

Summarizing some of the specific EpiData Analysis commands that we may use in HTML:

Action	EpiData Analysis command
Clear the screen and memory	epi:CLS
Show only output, not the running of the program	set echo=off
Do not display the main menu	set display mainmenu=off
Do not show the process bar	set display worktoolbar=off
Do not show the command line	set display command prompt=off
Run the program	run "..\pgm\run_entry.pgm"
Tell the user how to return to the main menu after the program has completed	type "F8: go back to the main menu"

This summarizes the principles we use to connect (making a link to) the execution of an EpiData Analysis program to an icon. While the HTML code is now correct, nothing will happen yet because the program file does not yet exist.

Preparing the EpiData Analysis program

We cannot work with the version of EpiData Analysis that we have in our epidata_report because we have turned off everything that is essential to work, the menu and the command line. What we do instead is to simulate the structure of the

epidata_report in our “normal” version of EpiData Analysis. We need the following folders in addition to what we have:

```
originals
pgm
temp
```

Copy all files from the epidata_report\originals folder into the originals folder you just made in your “normal” version of your EpiData software folder.

Change also the EpiDataStat.ini file in your “normal” version so that it ends up in the sub-folder \temp.

This way everything is set up in a way that will allow us once we have written all the program files to simply copy them over to the \epidata_report\pgm folder and everything will work out.

Reading the data file

We remember that the starting default folder is the \epidata_report\temp folder. After the standard closing of any open file, the path must thus be changed to the folder in which the data files are located, the \epidata_report\originals folder:

```
cls
close
logclose

cd ..
cd originals

read "x.rec"
```

This is how we would usually do it. We propose here an alternative and that is to copy all EpiData files that are currently in the \originals sub-folder into the \temp folder and to actually work in the \temp folder. While this is not essential here (nor anywhere else usually), it is a way that allows us manipulating our datasets in a temporary folder and leaving the original files untouched. The commands are then as follows (assuming that we are in the \temp sub-folder):

```
copyfile "..\originals\sample_2004.rec" "sample_2004.rec" /replace
copyfile "..\originals\sample_2004.chk" "sample_2004.chk" /replace
copyfile "..\originals\sample_2005.rec" "sample_2005.rec" /replace
copyfile "..\originals\sample_2005.chk" "sample_2005.chk" /replace
copyfile "..\originals\sample_2006.rec" "sample_2006.rec" /replace
copyfile "..\originals\sample_2006.chk" "sample_2006.chk" /replace

copyfile "..\originals\uganda.rec" "uganda.rec" /replace
copyfile "..\originals\uganda.chk" "uganda.chk" /replace
```

Note 1) that the path to the sub-folder \originals takes into account the current position in the sub-folder \temp and 2) that there is no connecting “to” between the two file designations, and 3) that we need to replace the files in the \temp sub-folder in case they exist.

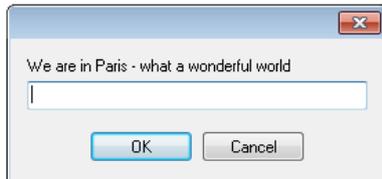
In addition, copy manually (not here in the program file) the file “side_by_side_graphs.html” into the \temp folder.

Interactive prompting

During the running of a program in a menu, there should be options to choose from. Depending on the selection the user makes, the program will then continue one or the other way(s). The program is thus to stop at a certain point and prompt the user for input. This is accomplished by typing something between two question marks. The following:

```
?We are in Paris - what a wonderful world?
```

gives you a pop-up box:



There is no option here, obviously, it is just a statement. To expand interactive prompting to include an option, we make use of a constant (global “variable”). To illustrate it, we are using the Uganda laboratory dataset (`uganda.rec`, included in the `originals` folder), collected over three years with a variable `REGDATE` denoting the registration date. We want to give the user the option to make a frequency of the field `SEX` for a selected year:

```
cls
close

read "uganda.rec"

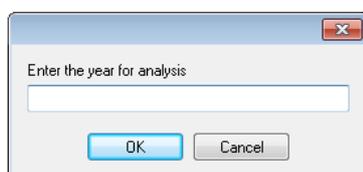
gen i regyy=year(regdate)
define yr ##### global
cls

freq regyy
  yr=?Enter the year for analysis?
set echo=off
  select if regyy=@yr
  cls
  freq sex
```

We first extract the registration year `REGYY` from the registration date. Then we create a constant `YR`. Then we display the frequency (`FREQ REGYY`) of the registration year and get:

<u>regyy</u>	
	<u>N</u>
1999	17300
2000	18662
2001	18088
Total	54050

The user is then immediately prompted to enter the year:



If, say, the year 2000 is entered, then the output is:

Sex of examinee	N
Female	7745
Male	9326
Missing	1591
Total	18662

Note the:

```
select if regyy=@yr
```

where we make use of the constant YR.

We can use the same principles, but expand it to provide an option to do either one thing or another:

```
define yesno # global

yesno=?Do this or that: 1=Do this - 2=Do that?
  imif yesno=1 then
    (do something)
  else
    (do something else)
  endif
```

The user then enters a "1" or a "2" into the box.

We can use a nested sequence of events. If we take the Uganda dataset again and we want to give the choice to first select the year, and then have the option of analyzing the entire selected year or only a quarter of it, we write (this sample program `imif_example.pgm` is available in your PGM folder):

```
set echo=off
gen i regyy=year(regdate)
label regyy "Registration year"
gen i quarter=month(regdate)
label quarter "Quarter of the year"
recode quarter 1-3=1 4-6=2 7-9=3 10-12=4
labelvalue quarter /1="Jan-Mar"
labelvalue quarter /2="Apr-Jun"
labelvalue quarter /3="Jul-Sep"
labelvalue quarter /4="Oct-Dec"

define yr #### global
define yesno # global
define q # global
cls

set echo=on
freq regyy
  yr=?Enter year for analysis?
set echo=off
  select if regyy=@yr
  cls
```

```

yesno=?Analyze: 1: Whole year; 2: One quarter only?
imif yesno=1 then
  cls
  type "You selected: Registration year @yr" /h2
  freq sex
else
  freq quarter /v1
  q=?Select quarter?
  select if quarter=@q
  cls
  type "You selected: Registration year @yr and Quarter @q" /h2
  freq sex
endif
set echo=off

```

In the first block, we extract registration year and registration month and recode the latter into quarters.

In the second block, we define the three constants.

In the third (last) block, we allow selection of the registration year:

```

freq regyy
  yr=?Enter year for analysis?
  select if regyy=@yr

```

Then based on the selected year, we ask to choose the quarter

```

cls
yesno=?Analyze: 1: Whole year; 2: One quarter only?

```

After choosing between “Whole year” and “One quarter only”, we make use of the `imif` command which is closed with an `endif` command:

```

imif yesno=1 then
  cls
  type "You selected: Registration year @yr" /h2
  freq sex
else
  freq quarter /v1
  q=?Select quarter?
  select if quarter=@q
  cls
  type "You selected: Registration year @yr and Quarter @q" /h2
  freq sex
endif

```

IMIF is used to divert course in a `pgm` file depending on parameters which can, as done here, be acquired by "? ... ?".

We use the `type` command to inform the user of the selection made. The `/h2` is HTML language for header size.

We will store all EpiData Analysis programs in the `\epidata_report\pgm` sub-folder.

Making the EpiData Analysis program to produce the quarterly report on case finding

You will make a total of three programs:

```

quarterly_report_1.pgm
quarterly_report_2.pgm

```

cleanup.pgm

We have two actual datasets of a random selection of tuberculosis case registers from the years 2004 and 2005 from Viet Nam. They were provided by courtesy of Dr Nguyen Binh Hoa from the National Tuberculosis Program Viet Nam. They were slightly edited from the original case registers in that the original identifier was removed and replaced by an artificial one. The names of the 30 units were also changed to have non-reality units. Several variables that were collected were also removed and some other minor modifications were made. However, overall, these data are real.

There are three data files:

```
sample_2004.rec
sample_2005.rec
sample_2006.rec
```

The last file (sample_2006.rec) contains no records and is used for data entry only.

The task is to produce a quarterly report on case finding. We follow here the guidelines of The Union for the quarterly report. It requires that all notifiable cases in the quarter are reported. We will get back to this shortly.

Because we set `echo=off` in the `start_tb.htm` file, the screen will stay blank during the running of the program. As this may confuse the reader, it will be useful to insert after every CLS command a line that informs the user that despite the blank screen something is happening, like:

```
cls
type "Be patient...files are identified and prepared" /h2
```

The following two parts are required for the quarterly report as recommended by The Union:

ALL CASES REGISTERED IN THE QUARTER

SMEAR-POSITIVE				SMEAR-NEGATIVE		EXTRA-PULMONARY	TOTAL
New cases	Relapses	Treatment after failure	Treatment after default	< 15 yrs	15 + yrs		

NEW SMEAR-POSITIVE CASES ONLY

Age group (years)														TOTAL		
0-14		15-24		25-34		35-44		45-54		55-64		65 +		Male	Female	Total
M	F	M	F	M	F	M	F	M	F	M	F	M	F			

A note on the notifiable cases: Patient categories “Transfer in” and “Other” must not be reported.

When you define your variables, note that not all registered cases may always have the required information. For instance, you need three variables to determine “SMEAR-NEGATIVE, <15 yrs”: The case must be pulmonary, you must know the case is smear-negative, and you must know the age. Take this into account when deciding what you are going to present the analysis.

If you want to produce a graphics output of two graphs and show them side by side, you must first save the graphs, e.g.:

```

bar repdef2 \
    /ti="All cases" \
    /save="all_cases_1.png" /replace
cls

select repdef2<>0
select repdef2<>9
bar repdef2 \
    /save="all_cases_2.png" /replace
cls

```

Among the required files was the HTML file “side_by_side_graphs.HTML”:

```

echo <table><tr><td colspan=2 border align=center >
<font color=black face="Arial" size="4">Some title that crosses over both
graphs</font></td>
<tr><td></td><td>
</td></table>

```

Replace the example names with the actual names of your graph files and then the simple command in EpiData Analysis:

```
show "side_by_side_graphs.html"
```

will show them side-by-side.

Other programs

The process is analogous for the quarterly report on treatment results. You can use the same basic program up to the point where you produce the actual output. You may consider two outputs, one for all cases, and one for sputum smear-positive cases only.

It might be useful to have a separate program to erase all files created during the session, so that only the files remain that are needed to start anew.

Command to exit the program

To exit the program, you do not need a special program. The EpiData Analysis command for the HTML file to quit the program without asking for conformation is:

```
<a href='epi:exit'> ... </a>
```

Opening the menu

To make things as simple as possible for the user, not requiring search for the epidatastat.exe file, you may add a batch file (that might be named start.bat, the *.bat extension being mandatory) at the same level as the \EPIDATA_REPORT folder. This batch file follows simple conventions (see internet for more on writing batch files):

```
cd epidata_report
start epidatastat.exe
```

Zip both the \EPIDATA_REPORT folder and the start.bat file into one single zip file (any name like anyname.zip will do of course), send it by email to the user with the instructions:

- 1) Unzip the anyname.zip file to any place on your computer
- 2) Double-click the start.bat file to open the menu
- 3) Start working

Task

Produce a single folder containing the EpiData Entry and Analysis programs, including the database of with an interface that permits by simple clicking to enter data or to run two standard reports, giving the user the option of choosing which country, year, laboratory, etc to analyze and is transferable to any drive or folder, with a total zipped file size of just 3.81 MB.

A user will be able to unzip this file onto any drive or folder, double-click the start.bat file, and be in the menu interface. Test it out by zipping it and then unzipping it onto a flash USB stick.

Solution to Exercise 4: Creating a menu for standard reports

Key points:

- The EpiData Analysis interface is largely based on HTML which gives the user great flexibility in customizing it to the needs
- A menu can be written for a user that requires not skills in EpiData Analysis and allows producing standard summary reports by simply clicking like on any web site

Task

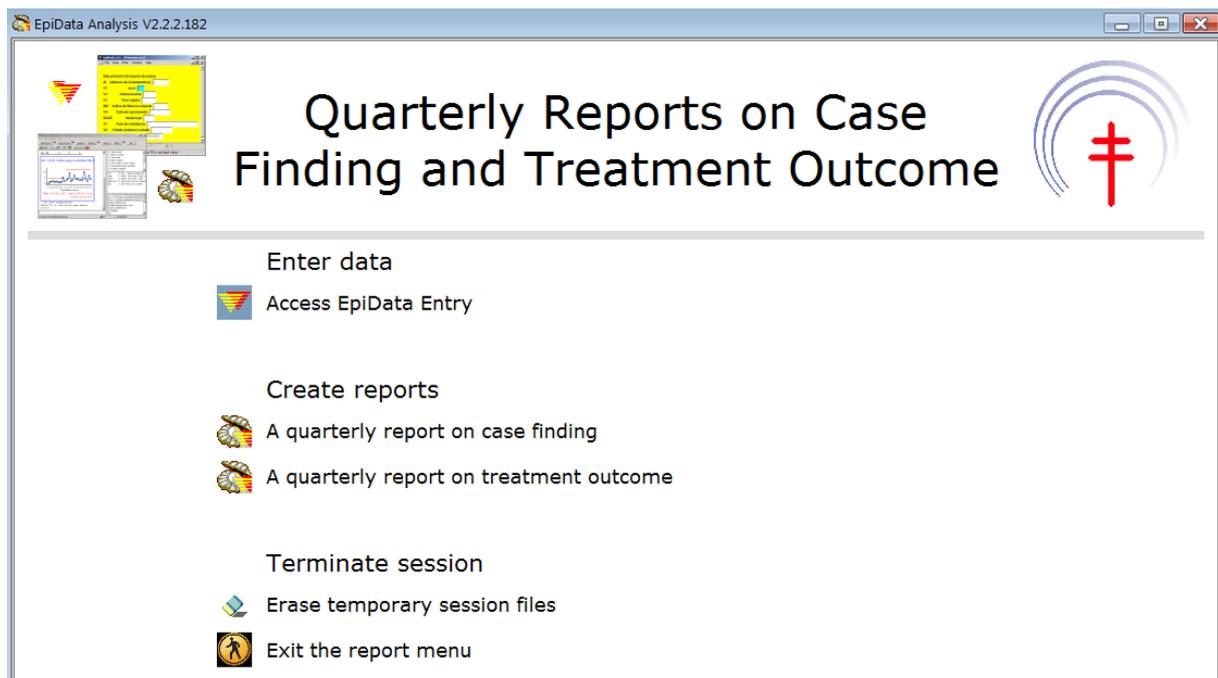
Produce a single folder containing the EpiData Entry and Analysis programs, including the database of with an interface that permits by simple clicking to enter data or to run two standard reports, giving the user the option of choosing which country, year, laboratory, etc to analyze and is transferable to any drive or folder, with a total zipped file size of just 3.81 MB.

A user will be able to unzip this file onto any drive or folder, double-click the `start.bat` file, and be in the menu interface. Test it out by zipping it and then unzipping it onto a flash USB stick.

Solution

The entire folder is available on the course web site as a zip file.

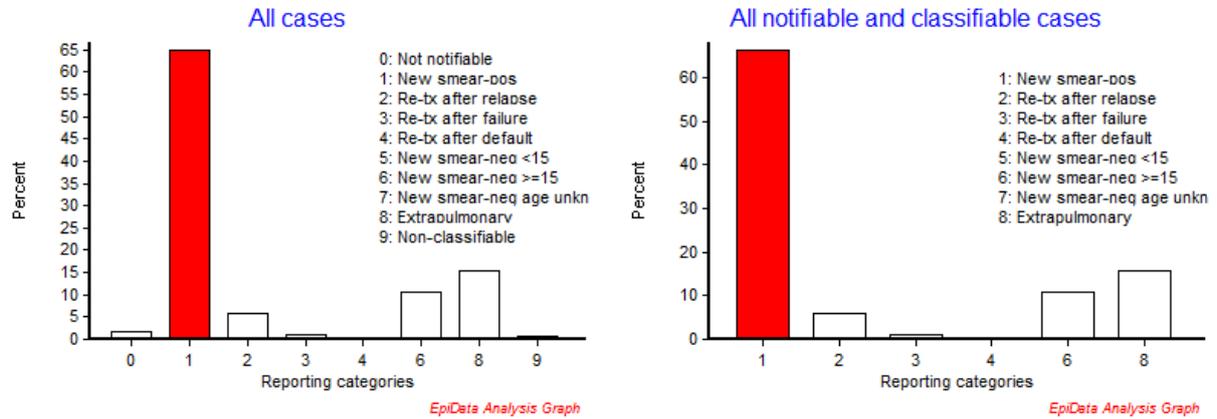
The interface the user sees:



In the following just select output graphs are shown.

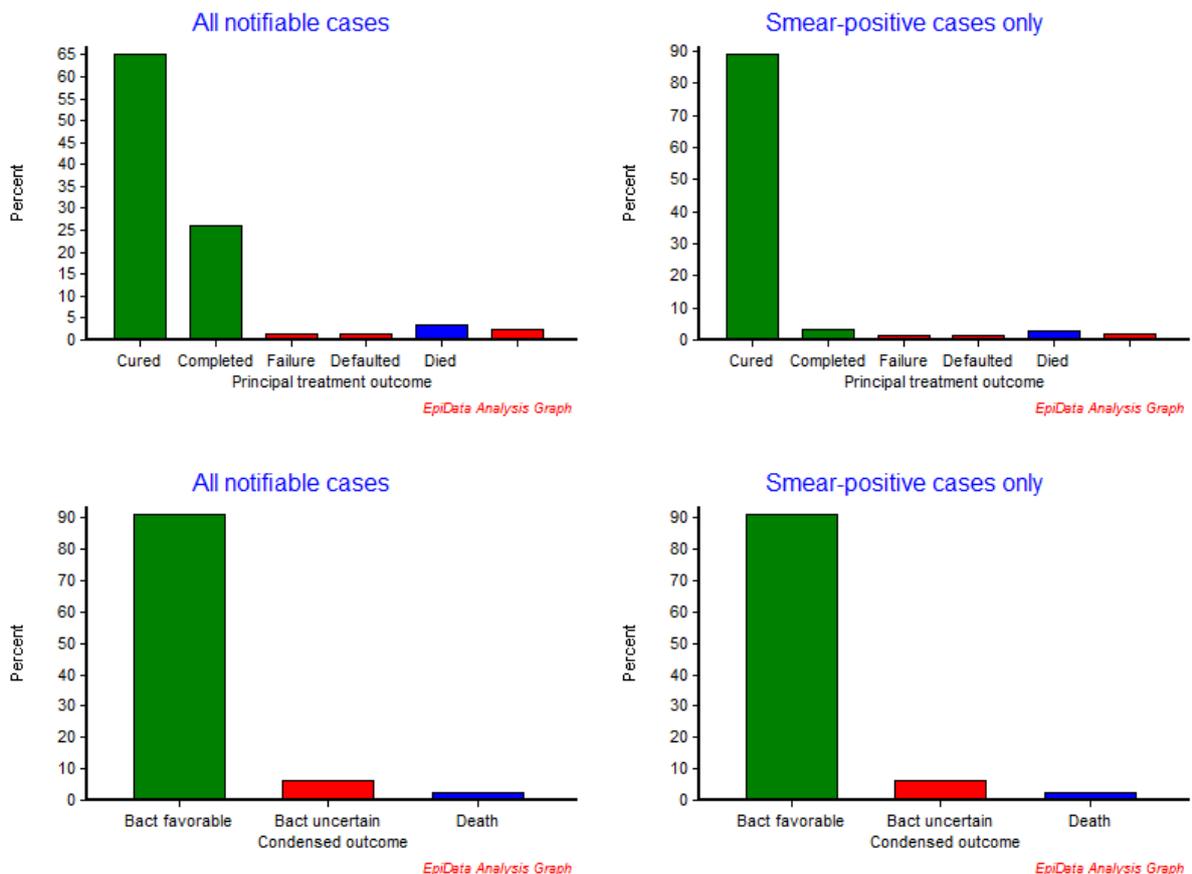
The output graphs from the first program:

Quarterly report on case finding



The output graph from the second program:

Quarterly report on treatment outcome



Exercise 5: Formatting standardized analysis output in a spreadsheet

At the end of this exercise you should be able to:

- a. Use a dummy file to produce a standardized aggregated EpiData file
- b. Producing formatted EpiData output in a spreadsheet

The following is a report describing tuberculosis patients by microscopic case definition and four variables (age, sex, disease site, and patient category), summarized in one single nicely formatted table:

Report from the National Tuberculosis Program

Characteristics of microscopy confirmed and other cases

Characteristic	Confirmed cases		Other cases	Total	Col %
	N	Row %			
Total	13	92.9	1	14	100.0
Age quartiles					
Quartile 1	3	100.0	0	3	21.4
Quartile 2	3	100.0	0	3	21.4
Quartile 3	1	50.0	1	2	14.3
Quartile 4	6	100.0	0	6	42.9
Age missing	0	NA	0	0	0.0
Sex					
Female	4	100.0	0	4	28.6
Male	9	90.0	1	10	71.4
Sex missing	0	NA	0	0	0.0
Disease site					
Pulmonary	13	92.9	1	14	100.0
Extrapulmonary	0	NA	0	0	0.0
Site missing	0	NA	0	0	0.0
Patient category					
New	12	92.3	1	13	92.9
Relapse	1	100.0	0	1	7.1
After failure	0	NA	0	0	0.0
After default	0	NA	0	0	0.0
Transfer in	0	NA	0	0	0.0
Other	0	NA	0	0	0.0
Category missing	0	NA	0	0	0.0

We note that the dataset is so small (14 records) that for every variable some strata have no cases. For instance, if we make a table in EpiData Analysis for case by category:

tables case category

we get:

Smear-based case definition			
Treatment category	Case	Non-case	Total
New	12	1	13
Relapse	1	0	1
Total	13	1	14

In other words, the EpiData output shows only strata containing at least one case. This is not a particularity of EpiData, this is quite generally so with any analysis program. In other words, depending on the selection criteria, we get visualization of varying column-row combinations, for instance:

Smear-based case definition			
Treatment category	Case	Non-case	Total
New	6	1	7
Relapse	1	0	1
No category recorded	0	1	1
Total	7	2	9

It is of course also not possible to have several tables condensed into a single one. For instance, if we produce two cross-tabulations, we formulate it as:

```
tables case sex
tables case category
```

and we get two separate tables:

Smear-based case definition			
Patient's sex	Case	Non-case	Total
Female	0	1	1
Male	7	1	8
Total	7	2	9

Smear-based case definition			
Treatment category	Case	Non-case	Total
New	6	1	7
Relapse	1	0	1
No category recorded	0	1	1
Total	7	2	9

In this exercise, we will automate a procedure to get EpiData output from multiple tables into one single nicely formatted table by using 1) a work-around to get all strata displayed even if their case count is zero, and 2) formatting the output in an esthetically appealing way by taking recourse to a spreadsheet template.

Modifying the basic menu

You have a required zip file “course_d_ex05_required.zip”. This is essentially the solution to Exercise 4, but with a slightly changed package name (“epidata_report2\”) to avoid confusion. Ensure first that your temporary folder C:\TEMP is empty, then unzip this required zip file into that folder and you get a folder and its start.bat:



Test it by double-clicking the start.bat file and then exit the menu again.

We need four more sub-folders in the epidata_report2 folder:

```
dummyset\
LibreOfficePortable\
templates\
workfolder\
```

After you created them, make sure that the epidata_course folder is empty, then copy the following five folders into the epidata_course folder:

```
dummyset\
originals\
pgm\
temp\
workfolder\
```

These will be used as simulation folders: we do everything in our “regular” EpiData Analysis, simulating what will happen in the menu. Once we assured functionality, we copy our

simulation folders into the menu in the `c:\temp` folder, start the menu and then run the program from there.

We will not yet deal with the `LibreOfficePortable` folder nor now change the `start_tb.htm` file.

The dummy files

We mentioned above that a prerequisite for a standard output is that we always need the same output style or, in other words, a display of all possible cells in a table. If we have a column variable `COLVAR` with 3 levels and a row variable `ROWVAR` with 4 levels we have $3 \times 4 = 12$ cells (not counting the marginals):

ROWVAR	COLVAR		
	1	2	3
1	c1-r1	c2-r1	c3-r1
2	c1-r2	c2-r2	c3-r2
3	c1-r3	c2-r3	c3-r3
4	c1-r4	c2-r4	c3-r4

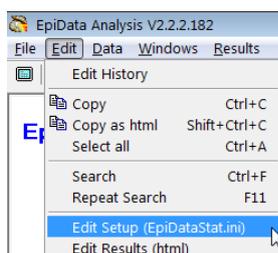
We need `c` times `r` records to have one record for every possible combination. To make this generically applicable, we will create an EpiData `*.rec` file with two variables:

```
colvar ##
rowvar ##
```

and pretend that we have 12 columns and only 1 row. We thus complete a total of 12 records with column values 1, 2, ..., 12 and row value 1, starting with making a `dummy_00.qes` file, then from it its `dummy_00.rec` file. We save both files in our `dummyset` sub-folder:

```
epidata_course\dummyset\
```

When now opening our “regular” or “standard” (not menu) EpiData Analysis, we first edit the `EpiDataStat.ini` file in `Edit`:



so that the last CD command directs EpiData Analysis to start in the `temp` sub-folder of the `epidata_course` folder:

```
cd c:\epidata_course\temp
```

As you perhaps remember, the `EpiDataStat.ini` file of the menu is also set to start in that folder. This allows moving one level up, then one level down into any sub-folder of the package.

Our intention will be to create a table consisting of four sub-tables, resulting from cross-tabulation of microscopy case status (smear-positive or other than smear-positive) *versus*

quartiles of age, sex, site of tuberculosis, and category of tuberculosis patient. For each of the four sub-tables we create a dummy set with 1 record for each possible cell. We start with creating one such dataset for case status versus age quartiles. Case status (the column variable) has 2 levels (case, non-case) and quartiles (the row variable) has 5 levels (the 4 quartiles, and missing age).

We start by opening a new *.pgm file in the EpiData Analysis editor, directing it from the current temp sub-folder to the dummyset sub-folder and reading the dummy_00.rec file:

```
* Create dummy set
```

```
cls  
close
```

```
cd ..  
cd dummyset
```

```
read "dummy_00.rec"
```

and save it as dummy_case_demogr_01.pgm in the dummyset folder. The dataset has 12 levels for the column variable rather than only the 2 we need. This will be addressed at the end of the program with a selection. The row value it has is 1, but as we will do some copying later with modification, we will overwrite it with, well "1" (one).

```
read "dummy_00.rec"  
rowvar=1
```

We need to create a unique identifier from the combination of the column and the row value. Later, we will create such an identifier in the same manner from the actual data set and then the dummy file created here can be merged to the actual dataset on this identifier. As we want to make this generically valid, we will consider how we actually designed the entry form, i.e. with two-digit fields for both column and row, thus we expand:

```
cls  
close  
read "dummy_00.rec"  
rowvar=1  
gen s(2) tempcol=string(colvar)  
gen s(2) temprow=string(rowvar)  
if colvar<10 then tempcol="0"+string(colvar)  
if rowvar<10 then temprow="0"+string(rowvar)  
gen s(5) id=tempcol+"-"+temprow  
drop tempcol temprow  
savedata "dummy_01.rec" /replace
```

Now, we copy the whole block and change only what is shown in red:

```
cls  
close  
read "dummy_00.rec"  
rowvar=2  
gen s(2) tempcol=string(colvar)  
gen s(2) temprow=string(rowvar)  
if colvar<10 then tempcol="0"+string(colvar)  
if rowvar<10 then temprow="0"+string(rowvar)  
gen s(5) id=tempcol+"-"+temprow  
drop tempcol temprow  
savedata "dummy_02.rec" /replace
```

and repeat it until we have a total of 5 dummy sets with row values of 1, 2, 3, 4, and 5 respectively, then append the 5 files:

```

cls
close
read "dummy_01.rec"
append /file="dummy_02.rec"
append /file="dummy_03.rec"
append /file="dummy_04.rec"
append /file="dummy_05.rec"
select colvar<=2
savedata "dummy_case_demogr_01.rec" /replace

```

Note the selection (in red) for the column levels in the second to last line. We must end up with 10 records, shown in the browser window as:

	colvar	rowvar	id
1	1	1	01-01
2	2	1	02-01
3	1	2	01-02
4	2	2	02-02
5	1	3	01-03
6	2	3	02-03
7	1	4	01-04
8	2	4	02-04
9	1	5	01-05
10	2	5	02-05

At the end, we erase all intermediary files:

```

erase "dummy_01.rec"
erase "dummy_02.rec"
erase "dummy_03.rec"
erase "dummy_04.rec"
erase "dummy_05.rec"

```

While we are at it, we should make dummy sets for the other three variables:

Row variable	Levels (strata)	Dummy set file name	Records
age quartile	5 (Quartiles 1, 2, 3, 4, and missing age)	dummy_case_demogr_01.rec	10
sex	3 (female, male, missing)	dummy_case_demogr_02.rec	6
site	3 (pulmonary, extrapulmonary, missing)	dummy_case_demogr_03.rec	6
category	7 (new, relapse, failure, default, transfer, other, missing)	dummy_case_demogr_04.rec	14

The principle is the same, and it is recommended to save each *.pgm file under a separate corresponding name to facilitate corrections if errors had been made and are found out only later.

The main analysis file

Let's open a new file in the EpiData Analysis editor and save it right away as case_demogr.pgm in the \pgm subfolder.

Similar as in the previous exercise, we copy the required files into the newly created working folder which we then access:

```

cd ..
cd originals
copyfile "sample_2004.chk" "..\workfolder\sample_2004.chk" /replace
copyfile "sample_2004.rec" "..\workfolder\sample_2004.rec" /replace
copyfile "sample_2005.chk" "..\workfolder\sample_2005.chk" /replace
copyfile "sample_2005.rec" "..\workfolder\sample_2005.rec" /replace

cd ..
cd dummyset
copyfile "dummy_case_demogr_01.rec" "..\workfolder\dummy_case_demogr_01.rec" /replace
copyfile "dummy_case_demogr_02.rec" "..\workfolder\dummy_case_demogr_02.rec" /replace
copyfile "dummy_case_demogr_03.rec" "..\workfolder\dummy_case_demogr_03.rec" /replace
copyfile "dummy_case_demogr_04.rec" "..\workfolder\dummy_case_demogr_04.rec" /replace

cd ..
cd workfolder

```

We then append the two years of data and make our case definition for the column variable:

```

read "sample_2004.rec"
append /file="sample_2005.rec"

define case #
case=2
if sm00>0 and sm00<9 then case=1
label case "Smear-based case definition"
labelvalue case /1="Case"
labelvalue case /2="Non-case"

```

Note that “Non-cases” are not smear-negative, they are not proven smear-positive, i.e. include all cases without a positive smear.

If you run the above, we see that the dataset contains a variable `regdate`, the registration date. In the previous exercise we selected cases for quarters. We want to make this more flexible here and allow the person running the program to choose interactively any time interval. We need thus six variables for day, month, and year to begin and to end and two additional variables to construct the two corresponding dates:

```

cls
type "Be patient...data compilation in progress" /h2
define regbegdd ##    global
define regbegmm ##    global
define regbegyy #### global
define regenddd ##    global
define regendmm ##    global
define regendyy #### global
define regbegdate <dd/mm/yyyy>
define regenddate <dd/mm/yyyy>

```

We then allow interactive selection for the beginning of the interval:

```

cls
type "Enter beginning of registration period, day, month, year" /h2
type "Only a date between 1 Jan 2004 and 31 Dec 2005 is possible" /h2
regbegdd=?Enter begin day of registration?
regbegmm=?Enter begin month of registration?
regbegyy=?Enter begin year (4-digit) of registration?
regbegdate=dmy(regbegdd, regbegmm, regbegyy)

```

and selection for the ending of the interval:

```

cls
type "Your chosen start date is @regbegdd - @regbegmm - @regbegyy" /h2
type "Enter now your choice of the end of registration period, day, month, year" /h2
type "Only a date between 1 Jan 2004 and 31 Dec 2005 is possible" /h2
regenddd=?Enter end day of registration?
regendmm=?Enter end month of registration?

```

```
regendyy=?Enter end year (4-digit) of registration?
regenddate=dmy(regendddd, regendmmm, regendyy)
```

Note the blue line which allows displaying the chosen date on the output screen.

Finally, we make the selection of the interval, and retain just the variables that we need for further work to speed up processing time and save the reduced data set:

```
cls
type "Be patient ... analysis in progress" /h2
select regdate>=regbegdate and regdate<=regenddate
keep case age sex site category
savedata "temp_01.rec" /replace
```

Run this part and for demonstration purposes, use the time interval including 1 Jan 2004 through 4 Jan 2004 to get a small data set that will assure that not all strata have cases (you should get 15 records).

So far, we had the preparatory steps. Now comes the construction of the first table, which is case status *versus* quartiles of age. We read the dataset and define a variable `colvar` to have identity with the variable name used in the dummy dataset and assign it the values of the variable `case`:

```
*****
* case_demogr_01: Age quartiles
```

```
cls
type "Be patient...data compilation in progress" /h2
close
logclose
read "temp_01.rec"
```

```
define colvar ##
colvar=case
```

The row variable should take the values of the age quartiles. In Part B, Exercise 2, we introduced how we can use the values EpiData Analysis stores in the variable `result` if we let it execute a command such as `means`. We utilize this here for the variable `age`:

```
means age if age<>99
define quart #
if age <$p251 then quart=1
if age>=$p251 and age<$p501 then quart=2
if age>=$p501 and age<$p751 then quart=3
if age>=$p751 then quart=4
if age =99 then quart=5
```

Then, in analogy to the column variable, we write for the row variable:

```
define rowvar ##
rowvar=quart
```

This way we have the two variables required to make our cross-table:

```
tables colvar rowvar
```

In Part B, Exercise 3, you learned that by replacing the `tables` command with the `aggregate` command we can get the result written into a file and then manipulate that file like creating a unique identifier in the same manner as we did in the dummy file:

```
aggregate rowvar colvar /close
cls
type "Be patient ... analysis in progress" /h2
```

```

gen s(2) id01=string(colvar)
gen s(2) id02=string(rowvar)
if colvar<10 then id01="0"+string(colvar)
if rowvar<10 then id02="0"+string(rowvar)
gen s(5) id=id01+"-"+id02
drop id01 id02
savedata "temp_02.rec" /replace

```

If we run this (with our above selection of 15 records) and browse this file, we get:

	rowvar	colvar	N	id
1	1	1	3	01-01
2	2	1	3	01-02
3	2	2	1	02-02
4	3	1	1	01-03
5	3	2	1	02-03
6	4	1	6	01-04

Manually adding up the field N we get the 15 case we expect. We have 6 records. As the column variable has 2 levels and the row variable 5, we should have $2*5=10$ records if each possible cell had cases. Thus, 4 cells have no cases. Our dummy file for age quartiles `dummy_case_demogr_01.rec` has all 10 records. We thus read in the dummy file and merge it on the id to this `temp_02.rec` file:

```

cls
type "Be patient ... analysis in progress" /h2
close
read "dummy_case_demogr_01.rec"
merge id /file="temp_02.rec"
sort rowvar colvar

```

Browsing shows which cells are missing in the actual data set and are contributed by the dummy set (Only in memory):

	colvar	rowvar	id	N	MergeVar
1	1	1	01-01	3	In both
2	2	1	02-01	.	Only in memory (Original)
3	1	2	01-02	3	In both
4	2	2	02-02	1	In both
5	1	3	01-03	1	In both
6	2	3	02-03	1	In both
7	1	4	01-04	6	In both
8	2	4	02-04	.	Only in memory (Original)
9	1	5	01-05	.	Only in memory (Original)
10	2	5	02-05	.	Only in memory (Original)

It is now simple to replace the period (for missing) with a count of zero and do a bit book keeping:

```

gen i cases=n
if n=. then cases=0
drop n mergevar
savedata "temp_03.rec" /replace

```

to get with browsing:

	colvar	rowvar	id	cases
1	1	1	01-01	3
2	2	1	02-01	0
3	1	2	01-02	3
4	2	2	02-02	1
5	1	3	01-03	1
6	2	3	02-03	1
7	1	4	01-04	6
8	2	4	02-04	0
9	1	5	01-05	0
10	2	5	02-05	0

We add a string variable to allow easy identification of values of the field rowvar:

```
gen s(35) rowvartxt=.
if rowvar=1 then rowvartxt="01_age_Q1"
if rowvar=2 then rowvartxt="01_age_Q2"
if rowvar=3 then rowvartxt="01_age_Q3"
if rowvar=4 then rowvartxt="01_age_Q4"
if rowvar=5 then rowvartxt="01_age_missing"
```

As shown in Part D, Exercise 1, we reshuffle from long to wide, select only the remaining necessary row, and drop superfluous variables:

```
cls
type "Be patient ... analysis in progress" /h2
gen i case=.
gen i noncase=.

cls
type "Be patient ... analysis in progress" /h2
case=cases
if colvar[_n+1]=2 then noncase=cases[_n+1]

select colvar=1
drop colvar rowvar id cases

savedata "case_demogr_01.rec" /replace
```

and we get a compact rectangular file in browsing

	rowvartxt	case	noncase
1	01_age_Q1	3	0
2	01_age_Q2	3	1
3	01_age_Q3	1	1
4	01_age_Q4	6	0
5	01_age_missing	0	0

in which each cell has a value, in contrast to the tables command which omits the missing:

colvar			
rowvar	1	2	Total
1	3	0	3
2	3	1	4
3	1	1	2
4	6	0	6
Total	13	2	15

The next sub-table deals with case status versus sex. Because of our setup, we can copy the entire block, and then change the assignment to `rowvar`, the dummy file set used for merging, change the values for `rowvartxt`, and finally save the file as `case_demogr_02.rec`. When done and then browsing you should get:

	rowvartxt	case	noncase
1	02_sex_female	4	0
2	02_sex_male	9	2
3	02_sex_missing	0	0

The two files we created, `case_demogr_01.rec` and `case_demogr_02.rec` have the same columns and can thus be appended:

```
cls
type "Be patient ... analysis in progress" /h2
close
read "case_demogr_01.rec"
append /file="case_demogr_02.rec"
```

In the menu, we want that at the end of the program, the browser window opens and the user is instructed what to do with the content of the browser window:

```
set display databrowser=on
browse
cls
type "Data compilation complete" /h2
type "Copy content of the browser window to clipboard and paste into spreadsheet" /h2
```

This gives us, labeled and clear, but with some esthetic deficiencies for the general user:

	rowvartxt	case	noncase
1	01_age_Q1	3	0
2	01_age_Q2	3	1
3	01_age_Q3	1	1
4	01_age_Q4	6	0
5	01_age_missing	0	0
6	02_sex_female	4	0
7	02_sex_male	9	2
8	02_sex_missing	0	0

With that we are done (run the entire program to test that it works). We will now turn to the next part, the menu. The finalization of the script with all four variables rather than just these two will be part of the task.

For the time being we are done with the work in the “regular” EpiData Analysis and what we did will work smoothly in the EpiData Analysis menu. Thus, copy all sub-folders from the `epidata_course` to the `epidata_report2` folder, overwriting everything if prompted.

Installing the LibreOfficePortable suite

The challenge at hand is to format the above `*.rec` file into something useful and esthetically pleasing that can easily be shared with others who do not have EpiData installed on their computer. The most appealing approach is the use of non-proprietary software that is part of the menu package and can thus be fully controlled and easily used through a few clicks.

This is most efficiently accomplished with a pre-formatted spreadsheet template. Pre-formatting is possible because the number of columns and rows is fixed in the EpiData Analysis output: it was the very purpose of merging a pre-defined dummy set with a dataset sub-selection. Using proprietary Excel™ cannot as smoothly solve the problem. It notably requires that the user has purchased the software. It is also not that trivial to access Excel™ from the EpiData `start_tb.htm` file. A straight forward way is to use the portable edition of non-proprietary, open-source LibreOffice (formerly known as OpenOffice). It is a true suite, that is, it has a core functionality that holds all individual sub-components (seven in total) together. There is no point in trying to install only the spreadsheet sub-component, it is making things complicated while not at all being a space-saver. In the software group you find the install file (or preferably download it from the Internet, but note its large file size). Install it into the sub-folder LibreOfficePortable in the `epidata_report2` folder we have already created.

Editing the `start_tb.htm` file

As you remember, the `start_tb.htm` file is the file called when you start the EpiData Analysis executable file `epidatastat.exe`. It defines the menu interface. It is located in the `\languages\en` sub-folder. We open it in our text editor (while possible of course, don't do it in NotePad™, it is not a quality text editor. Use instead free Crimson Editor you find in the software section, the coloring is just one of its many advantageous aspects).

Just after the opening table tag `<tbody>` you find (on line 34 if you use Crimson):

```
33 <tbody>
34 <tr><td style="background-color: white; width: 73px;"><a href="http://www.epidata.dk"></a></td><td colspan="2" rowspan="1" style="background-color:
  white; width: 41px; text-align: center;"><big><big><big><big>Quarterly Reports on Case
  Finding and Treatment Outcome</big></big></big></big></td><td style="background-color:
  white; width: 121px;"><a href="http://www.theunion.org"></a></td></tr>
```

All cells in a given the table row (shown with the beginning tag `</tr>` and the ending tag `</tr>` circled red above) are a bit crammed up. We want to isolate the individual cells for clarity's sake, take the row apart with hard carriage returns so that each cell starting with an opening `<td>` and a closing `</td>` tag gets its own paragraph as follows:

```

34 <tr>
35 <td style="background-color: white; width: 73px;"><a href="http://www.epidata.dk"></a></td>
36 <td colspan="2" rowspan="1" style="background-color: white; width: 41px; text-align: center;
   "><big><big><big><big>Quarterly Reports on Case Finding and Treatment Outcome</big></big>
   </td>
37 <td style="background-color: white; width: 121px;"><a href="http://www.theunion.org"></a></td>
38 </tr>

```

This allows counting the cells: here we have 4 (two cells in the middle are merged with colspan="2"). However, we want to have 6 cells per row:

We now have:

EpiData icon				Union icon
	Analysis icon	A quarterly report on case finding		

We needed thus only 4 cells, but now we want to change this to:

EpiData icon					Union icon
	Analysis icon	Run EpiData analysis program	Spreadsheet icon	Paste output into spreadsheet	

This requires 6 cells per row. In the top row above, we thus change this line:

```

36 <td colspan="2" rowspan="1" style="background-color: white; width: 41px; text-align: center;
   "><big><big><big><big>Quarterly Reports on Case Finding and Treatment Outcome</big></big>
   </td>

```

to expand the middle colspan="2" to colspan="4":

```

36 <td colspan="4" rowspan="1" style="background-color: white; width: 41px; text-align: center;
   "><big><big><big><big>Quarterly Reports on Case Finding and Treatment Outcome</big></big>
   </td>

```

The subsequent empty (no text, no icons) row is changed from:

```

40 <tr>
41 <td colspan="4" rowspan="1"></td>
42 </tr>

```

to

```

40 <tr>
41 <td colspan="6" rowspan="1"></td>
42 </tr>

```

We thus change (tedious perhaps but also educational) every row to always have a total of 6 cells per row, irrespective of whether the row is empty or contains icons and / or instructions. Attention must be paid to make the merging (colspan) in the correct places, and noting that

sometimes we need `colspan="2"` and sometimes `colspan="3"`. If you are all done with each row in this way and you refresh the menu interface (F8), there should not be any visually apparent change, but you have changed your menu table from 4 to 6 columns.

Next we duplicate by copying the row with “A quarterly report on treatment outcome” and change the text to “Demographic and other characteristics of patients” which should give us:

-  A quarterly report on treatment outcome
-  Demographic and other characteristics of patients

If we remove the `colspan="3"` in what is below line 86:

```
83 <tr>
84 <td style="background-color: white; width: 73px;"></td>
85 <td style="background-color: white; width: 41px;"><a href='epi: set echo=off; CLS; run "..\pgm\quarterly_report_2.pgm"; type "F8: Go back to Menu"'></td>
86 <td colspan="3" style="background-color: white; width: 898px;">Demographic and other characteristics of patients</td>
87 <td style="background-color: white; width: 121px;"></td></tr>
```

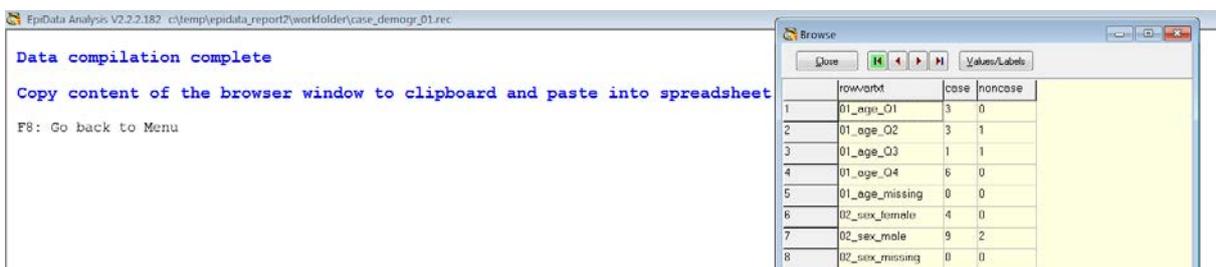
and duplicate lines 85 and 86, then we should get:

```
83 <tr>
84 <td style="background-color: white; width: 73px;"></td>
85 <td style="background-color: white; width: 41px;"><a href='epi: set echo=off; CLS; run "..\pgm\quarterly_report_2.pgm"; type "F8: Go back to Menu"'></td>
86 <td style="background-color: white; width: 898px;">Demographic and other characteristics of patients</td>
87 <td style="background-color: white; width: 41px;"><a href='epi: set echo=off; CLS; run "..\pgm\quarterly_report_2.pgm"; type "F8: Go back to Menu"'></td>
88 <td style="background-color: white; width: 898px;">Demographic and other characteristics of patients</td>
89 <td style="background-color: white; width: 121px;"></td></tr>
```

displayed in the browser as:

-  A quarterly report on case finding
-  A quarterly report on treatment outcome
-  Demographic and other characteristics of patients
-  Demographic and other characteristics of patients

In line 85 (here) we change the name of the program file “quarterly_report_2.pgm” to the program file we wrote before, i.e. “case_demogr.pgm”. After saving and then refreshing the menu in the browser, we test it and should get (perhaps with different numbers, depending on our interval selection):



The screenshot shows the EpiData Analyze interface. The main window displays "Data compilation complete" and a button to "Copy content of the browser window to clipboard and paste into spreadsheet". Below this, it says "F8: Go back to Menu". An embedded browser window shows a table with the following data:

	rowwrtxt	case	noncase
1	01_age_01	3	0
2	01_age_02	3	1
3	01_age_03	1	1
4	01_age_04	6	0
5	01_age_missing	0	0
6	02_sex_female	4	0
7	02_sex_male	9	2
8	02_sex_missing	0	0

What’s above in lines 86 and 87 describes the contents for cells 4 and 5 of the total 6 cells in the row. It must become something quite different. What’s between the single quotes shown below in blue:

```
<a href='epi: set echo=off; CLS; run "..\pgm\case_demogr.pgm"; type "F8: Go back to Menu"'>
```

are all EpiData Analysis commands. Instead we want to call the LibreOffice spreadsheet and within a template (which we have not yet written), which will get the name “case_demogr.ots” (LibreOffice spreadsheet templates have the extension *.ots). Thus, all between the single quotes must be deleted and be replaced with the path to the *.exe file of the LibreOffice spreadsheet (the “LibreOfficeCalcPortable.exe” file in the sub-folder LibreOfficePortable). The template “case_demogr.ots” will reside in the sub-folder templates. Hierarchically at the same location of images the path to the executable for the LibreOffice spreadsheet needs analogously a preceding double “../..”, i.e. the reference becomes:

```
<a href= "../..../LibreOfficePortable/LibreOfficeCalcPortable.exe ..\templates\case_demogr.ots">
```

The reference to the image:

```
alt="epidata_analysis" src="../../../images/epidatastat.png"
```

needs to be changed, using the logo of LibreOffice spreadsheet provided in the images folder:

```
alt="LibreOffice" src="../../../images/logo_libreoffice_calc.png"
```

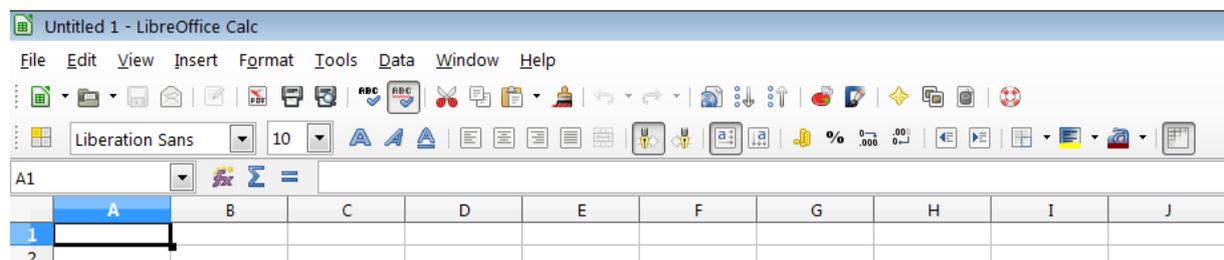
Finally, the label: “Demographic and other characteristics of patients” could be changed to something like “Access the spreadsheet to format EpiData output”. Displayed in the browser window after refreshing it shows as:



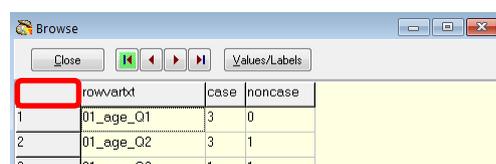
Access is of course not functional as the template is not yet available. This is the next thing we are making.

Creating a template in the LibreOfficeCalcPortable spreadsheet

Open a new spreadsheet by double-clicking the executable “LibreOfficeCalcPortable.exe” and you get an interface similar to the Microsoft Excel™ spreadsheet:



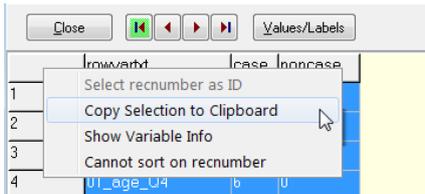
If you return to our menu, you can still call browsing with F6 as you would in the “regular” EpiData Analysis, the file created is still there and you get:



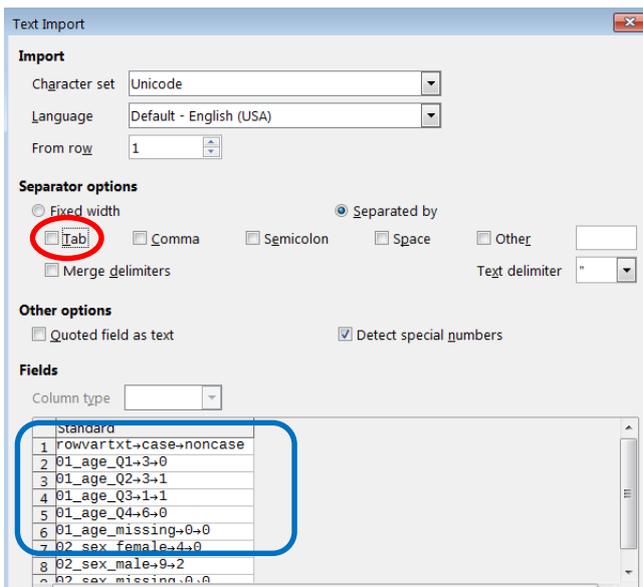
If you click on the intersection between rows and columns (shown red-circled above), the entire file gets marked (as in a spreadsheet):

	rowvartxt	case	noncase
1	01_age_Q1	3	0
2	01_age_Q2	3	1
3	01_age_Q3	1	1
4	01_age_Q4	6	0
5	01_age_missing	0	0
6	02_sex_female	4	0
7	02_sex_male	9	2
8	02_sex_missing	0	0

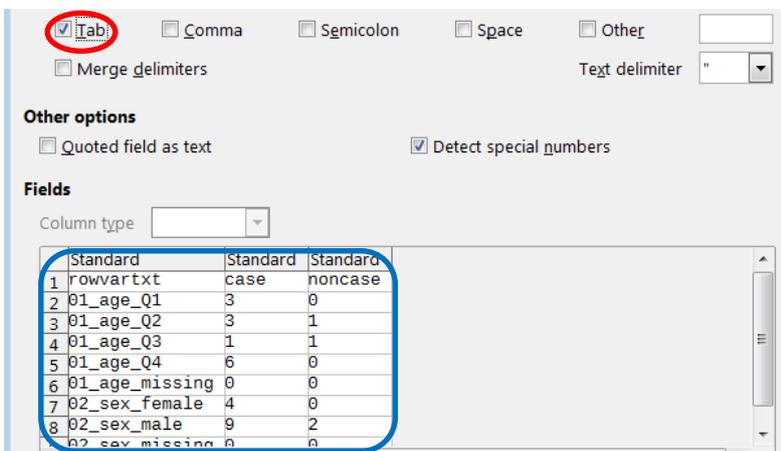
Right-clicking gives the option to copy to Clipboard (**CTRL+C** will also do):



Copy and Close and return to the spreadsheet and paste (**CTRL+V**) to get prompted:



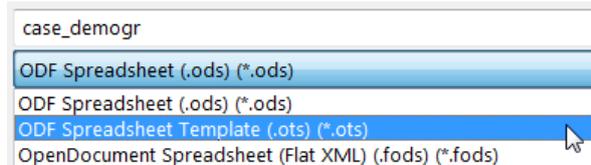
If **Tab** is not ticked, you have to, to get the input into columns rather than into one cell per row:



After OK you get the browser output copied into the spreadsheet:

	A	B	C
1	rowvarbt	case	noncase
2	01_age_Q1	3	0
3	01_age_Q2	3	1
4	01_age_Q3	1	1
5	01_age_Q4	6	0
6	01_age_miss	0	0
7	02_sex_fem	4	0
8	02_sex_male	9	2
9	02_sex_miss	0	0

Save this right away as an ODF Spreadsheet Template (.ots) (*.ots):



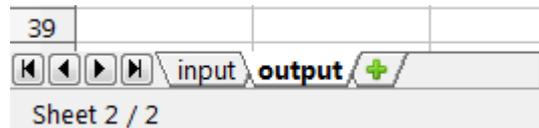
in the `\templates` sub-folder. As long as we don't quit LibreOffice, saving is always to the template (not a regular spreadsheet file, which would have the extension `*.ods`).

We will have two sheets in this template, one for input, and a second for output, and we label them at the bottom, changing:

From default label of one sheet ...



... to customized label with two sheets



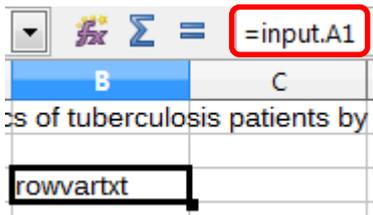
In the output sheet, we start with writing some more or less sensible title:

	A	B	C	D	E
1	Characteristics of tuberculosis patients by microscopy-defined case status				
2					

You may know from Microsoft Excel™ that you can copy content from one sheet's cell to any cell in a second sheet (here output), if you type an equal sign "=" into the destination cell in the output sheet then switch to the source cell in the first sheet (here input) and press there the hard carriage return. Specifically here, move your cursor into cell B3 of the output sheet, type the equal sign, then go to cell A1 in the input sheet and press the hard carriage return key. You will automatically be returned to the output sheet and now have the value written into cell B3:

	A	B	C	D	E
1	Characteristics of tuberculosis patients by microscopy-defined case status				
2					
3		rowvarbt			
4					

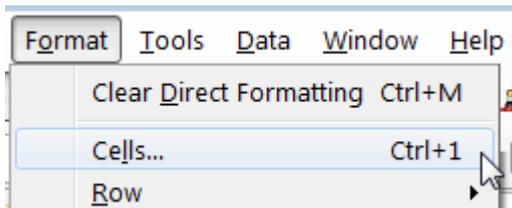
If you inspect the B3 cell content you see the formula:



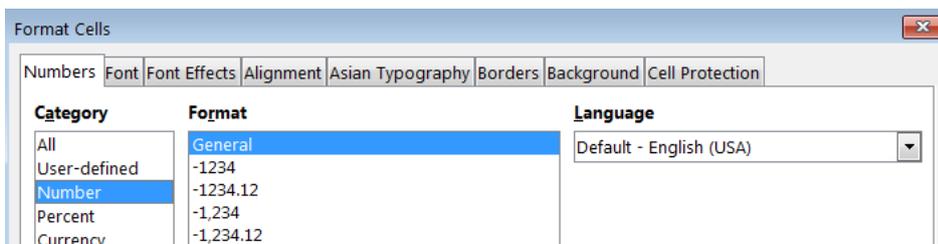
Copy this cell down and then copy the column to the right to get the entire content of the input sheet:

	A	B	C	D
1	Characteristics of tuberculosis patients by microscopy-def			
2				
3		rowvartxt	case	noncase
4		01_age_Q1	3	0
5		01_age_Q2	3	1
6		01_age_Q3	1	1
7		01_age_Q4	6	0
8		01_age_miss	0	0
9		02_sex_fem	4	0
10		02_sex_male	9	2
11		02_sex_miss	0	0

We refine formatting further, inserting rows and making more appealing labels (after all our row names were just for unambiguous identification). In addition to the standards of inserting rows and columns and so on, make use of Format | Cells:



It has an extensive menu for all sorts of things on formatting cells:



To make sums you may use `=sum(a1:a4)` to get the sum of cells $a_1+a_2+a_3+a_4$. Do you know how to avoid the error message from a division by zero:

Female	4	100.0	0	4	26.7
Male	9	81.8	2	11	73.3
Sex missing	0	#DIV/0!	0	0	0.0

If you type:

`=IF(F11<>0, C11/F11*100, "NA")`

you get:

Female	4	100.0	0	4	26.7
Male	9	81.8	2	11	73.3
Sex missing	0	NA	0	0	0.0

Note that the default of LibreOffice are American measures. For instance, the default page size is Letter size rather than A4. It is easy to change if you need to do so (consider printing).

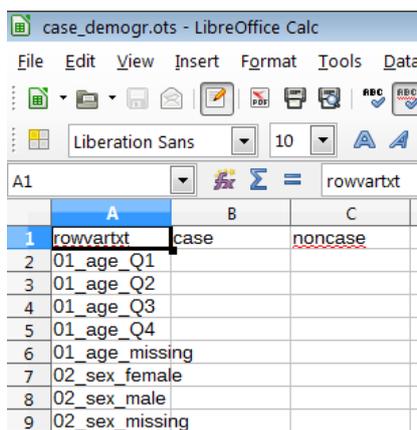
After you have exhausted your creativity, you may get something like:

Report from the National Tuberculosis Program

Characteristics of tuberculosis patients by microscopy-defined case status

Characteristic	Smear-positive cases		Other cases	Total	Col %
	n	Row %			
Total	13	86.7	2	15	100.0
Age quartiles					
Quartile 1	3	100.0	0	3	20.0
Quartile 2	3	75.0	1	4	26.7
Quartile 3	1	50.0	1	2	13.3
Quartile 4	6	100.0	0	6	40.0
Age missing	0	NA	0	0	0.0
Sex					
Female	4	100.0	0	4	26.7
Male	9	81.8	2	11	73.3
Sex missing	0	NA	0	0	0.0

After being happy with your formatting (don't overdo it now, you will have more opportunity in the upcoming task), we return to the input sheet, delete all numbers (leave the labels) and put the cursor into cell A1:



Make sure you save it (it will still be saved as the *.ots template file), and then close LibreOffice Calc (**ALT+F4** closes the program).

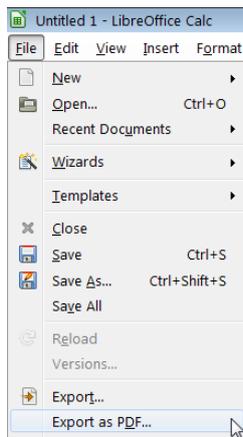
To test, run the program again, make some other interval selection, and then click the LibreOffice icon to open the template.

Note: LibreOffice is a slow opener, much slower to open than Microsoft Office. Be patient, do not click the icon more than once: either it works with one click or it will not work with 100 clicks, but multiply clicking can cause conflicts and will surely slow it down even further!

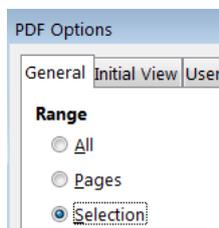
The last thing to show, is the inbuilt possibility to make a PDF file. If you have run your program, and pasted the EpiData Analysis browser content into the input file, switch to the output file and mark all fields:

	A	B	C	D	E	F	G
1	Characteristics of tuberculosis patients by microscopy-defined case status						
2							
3	Characteristic	Smear-positive cases		Other cases	Total	Col %	
4		n	Row %				
5	Total	5,058	72.5	1,921	6,979	100.0	
6	Age quartiles						
7	Quartile 1	1,014	61.8	627	1,641	23.5	
8	Quartile 2	1,365	74.6	465	1,830	26.2	
9	Quartile 3	1,367	77.6	395	1,762	25.2	
10	Quartile 4	1,312	75.2	432	1,744	25.0	
11	Age missing	0	0.0	2	2	0.0	
12	Sex						
13	Female	1,408	65.9	728	2,136	30.6	
14	Male	3,650	75.4	1,192	4,842	69.4	
15	Sex missing	0	0.0	1	1	0.0	

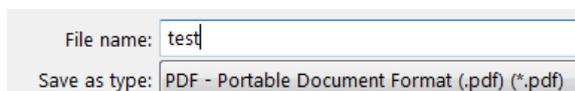
In File you have Export as PDF:



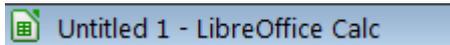
Clicking it, you need to choose Selection under the General tab:



and save the selection in some folder with your favorite name:



Then go there to view it. Because the access was to a template file, this spreadsheet is still untitled:



You can save it (as a LibreOffice *.ods spreadsheet file or also as a Microsoft Office Excel™ file).

Task:

- o Complete the EpiData Analysis program file “case_demogr.pgm” to make an output for all four variables i.e. adding site of disease and patient category. Then upgrade the LibreOffice template file “case_demogr.ots” and create the output as a PDF file.*
- o Write a “cleaning” program that ensures that all temporary session files are erased.*

Solution to Exercise 5: Formatting standardized analysis output in a spreadsheet

Key points:

- Cross-tables in analysis may have varying dimensions, depending on how many strata have counts
- A work-around to obtain fixed dimensions is the use of merging the aggregated data with a dummy set in which there is one record for each possible column-row combination
- Output from multiple tables can be stacked in a *.rec file after aggregation and a and pasted into a preformatted spreadsheet for a formatted output

Tasks:

- o Complete the EpiData Analysis program file “case_demogr.pgm” to make an output for all four variables i.e. adding site of disease and patient category. Then upgrade the LibreOffice template file “case_demogr.ots” and create the output as a PDF file.*
- o Write a “cleaning” program that ensures that all temporary session files are erased.*

The solution is the entire menu package. The PDF exported spreadsheet output is shown on the following page.

Report from the National Tuberculosis Program

Characteristics of microscopy confirmed and other cases

Characteristic	Confirmed cases		Other cases	Total	Col %
	N	Row %			
Total	5,058	72.5	1,921	6,979	100.0
Age quartiles					
Quartile 1	1,014	61.8	627	1,641	23.5
Quartile 2	1,365	74.6	465	1,830	26.2
Quartile 3	1,367	77.6	395	1,762	25.2
Quartile 4	1,312	75.2	432	1,744	25.0
Age missing	0	0.0	2	2	0.0
Sex					
Female	1,408	65.9	728	2,136	30.6
Male	3,650	75.4	1,192	4,842	69.4
Sex missing	0	0.0	1	1	0.0
Disease site					
Pulmonary	5,030	85.6	848	5,878	84.2
Extrapulmonary	23	2.1	1,072	1,095	15.7
Site missing	5	83.3	1	6	0.1
Patient category					
New	4,366	74.5	1,491	5,857	83.9
Relapse	502	97.9	11	513	7.4
After failure	64	100.0	0	64	0.9
After default	22	91.7	2	24	0.3
Transfer in	76	52.4	69	145	2.1
Other	8	14.3	48	56	0.8
Category missing	20	6.3	300	320	4.6

Part E. Beyond EpiData Analysis using R

Part E: Beyond EpiData Analysis using R

Exercise 1: Introduction to R software: basics

Exercise 2: Introduction to R software: data bases and functions

Exercise 3: Multivariable analysis in R part 1: A logistic regression

Exercise 4: Multivariable analysis in R part 2: A Cox proportional hazard model

Introductory note

Part E addresses how to conduct a multivariable analysis for two situations for which EpiData Analysis is not designed to be operational. With our philosophy that there should preference be given to outstanding freely available software, we chose to introduce the open-source package R rather than any of the proprietary software solution packages.

The first exercise introduces the bare-bone basics of R software with a few simple exercises to get a feel for its working. Recommended texts for self-study are (available on the course CD-ROM):

Aragón T J. Applied epidemiology using R. University of California, Berkeley School of Public Health, and the San Francisco Department of Public Health. Version 14 October 2013, available from <http://medepi.com/courses/applied-epi-using-r/>.

Lumley T and the R Core Development Team and UW Department of Biostatistics. R fundamentals and programming techniques. Version Birmingham, 2006-2-27/28, available from <http://faculty.washington.edu/tlumley/Rcourse/R-fundamentals.pdf>.

Venables W N, Smith D M and the R Core Team. An introduction to R. Notes on R: a programming environment for data analysis and graphics, Version 3.1.0 (2014-04-10), available from: <http://cran.r-project.org/doc/manuals/R-intro.pdf>.

The second exercise extends on the basics by introducing data bases and functions. The recommended text for self-study is (available on the course CD-ROM):

Myatt M. Open source solutions – R. Nordic School of Public Health and University of Tartu. Computer software in epidemiology / statistical practice in epidemiology. Version 16 May 2005, available from <http://www.brixtonhealth.com>.

The third exercise gives the first application, the use of logistic regression to determine predictors that include continuous variables and variables with more than two levels. The recommended text for self-study is (available in the course material):

Manning C. Logistic regression (with R). Version 4 November 2007, available from <http://nlp.stanford.edu/~manning/courses/ling289/logistic.pdf>.

The fourth exercise gives the second application, survival analysis where stratification with the Kaplan-Meier approach does no more suffice and adjustments are preferred by using a Cox

proportional hazard model. The recommended texts for self-study are (available in the course material):

Tableman M. Survival analysis using S/R. Unterlagen für den Weiterbildungsgang in angewandter Statistik an der ETH Zürich. [Note: despite the German subtitle, the document is entirely in English, except for the subtitle which refers to the course at the Swiss Federal Institute of Technology in Zurich]. Version August-September 2012. It is an excerpt from the book *Survival Analysis Using S: Analysis of Time-to-Event Data* by Mara Tableman and Jong Sung Kim, published by Chapman & Hall/CRC, Boca Raton, 2004.

Therneau T M. Package 'survival'. Version 2.37-7, July 2, 2014, available from <http://cran.r-project.org/web/packages/survival/index.html>.

Acknowledgments:

We used the following data in the exercises:

In **Exercise 1**, we use an example given by Altman in:

Altman D G, Machin D, Bryant T N, Gardner M J. Statistics with confidence. 2nd edition. 2 ed. Bristol: BMJ Group; 2000.

In **Exercises 2, 3, and 4** we use data from:

Aung K J M, Van Deun A, Declercq E, Sarker M R, Das P K, Hossain M A, Rieder H L. Successful “9-month Bangladesh regimen” for multidrug-resistant tuberculosis among over 500 consecutive patients. *Int J Tuberc Lung Dis* 2014;18: in press.

We are grateful to Drs Aung, Declercq, and Van Deun to grant permission to use an excerpt of individual data from the original dataset.

Nguyen Binh Hoa, Ajay M V Kumar, and Zaw Myo Tun went through the first draft of the exercises and gave valuable input into the development of Part E.

Exercise 1: Introduction to R software

At the end of this exercise you should be able to:

- Have a basic understanding of how R works and to expand on this basis in a self-learning process
- Be informed about resources on R software

R is a language and environment for statistical computing and graphing. It provides a suite of numerous packages for a virtually unlimited number of applications, including the most powerful tools that an epidemiologist might need. It has a large community of collaborators and contributors, it is entirely open to everybody who wants to use it and who wishes to contribute.



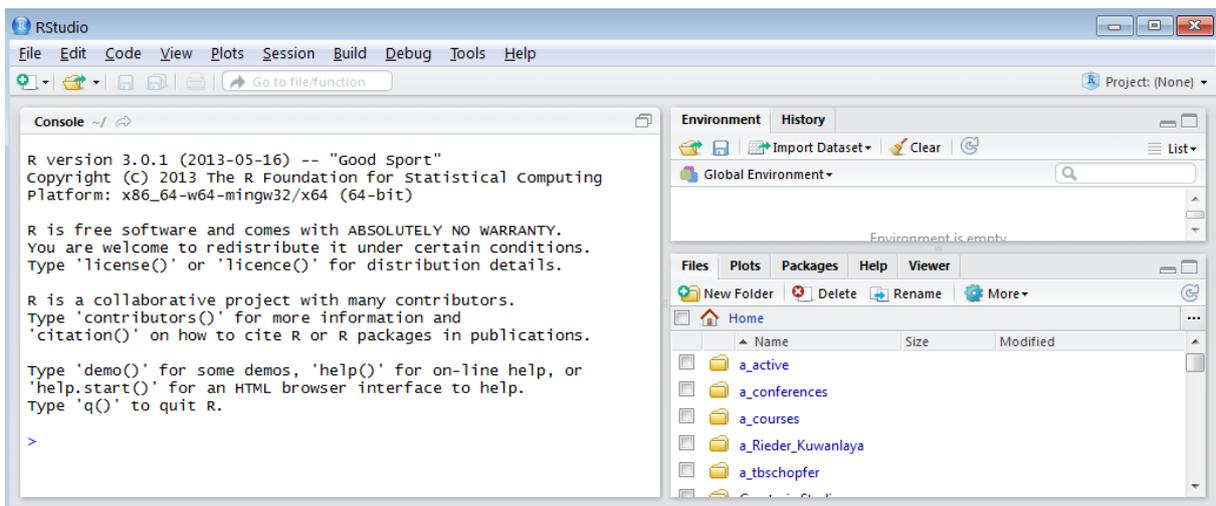
The starting point on the internet is the “The R Project for Statistical Computing” at its web site: <http://www.r-project.org/>. This is from where you access one if its mirror sites to download the software.



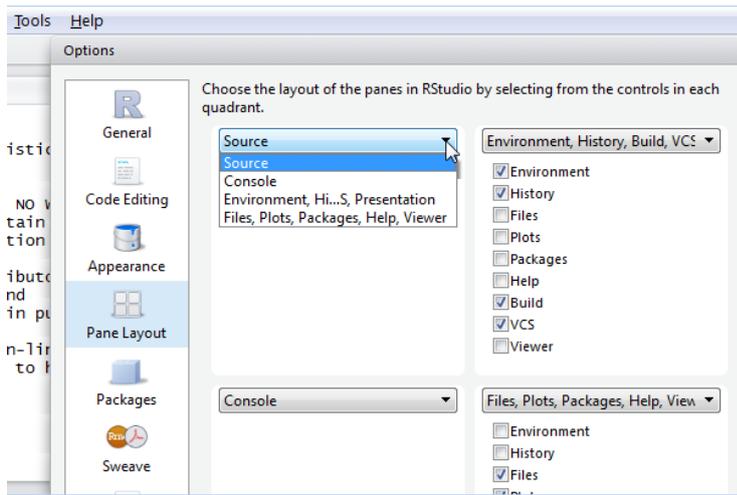
In tandem with R, we will also use RStudio, also open-source and free, a powerful and productive user interface for R. It can be found for download at its web site: <http://www.rstudio.com/>.

If your course comes on a CD-ROM, you find the two software packages on your CD-ROM, but as always with software, we strongly recommend that you obtain it from its original source on the internet.

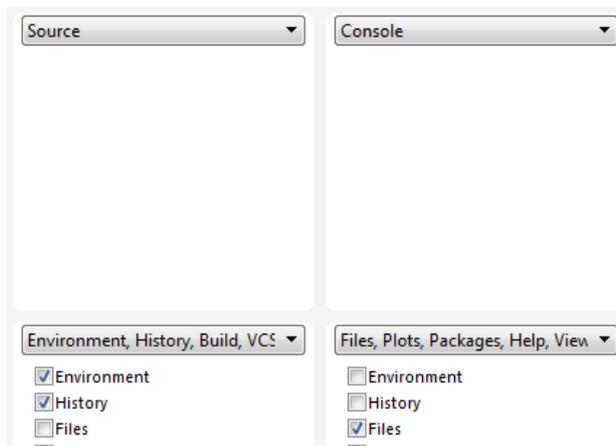
Install R on your computer and then install RStudio. If you open RStudio, you see that it is divided into a left-hand-sided panel, and a split right-handed panel:



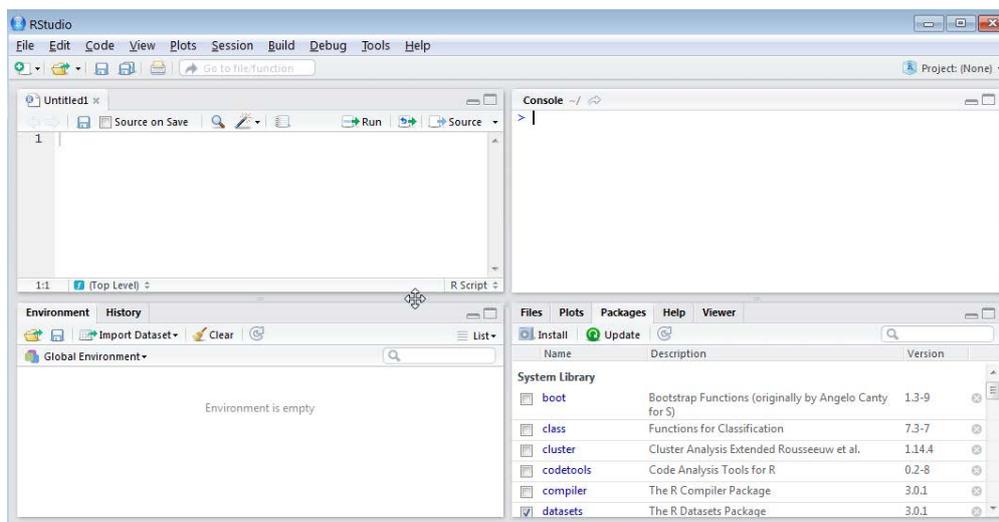
In Tools | Options we can adjust to the preferred layout:



We chose here (somewhat arbitrarily) to have the Source on the top left, the Console on the top right, the Environment, History, ... on the bottom left, and the Files, Plots, Packages, Help on the bottom left. and get altogether four quadrants:



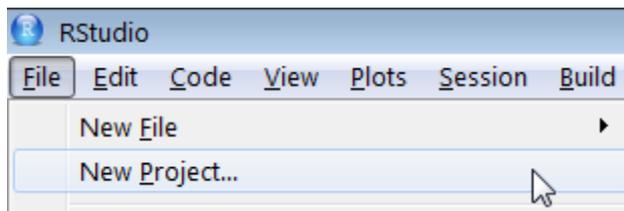
So that after these adjustments we get:



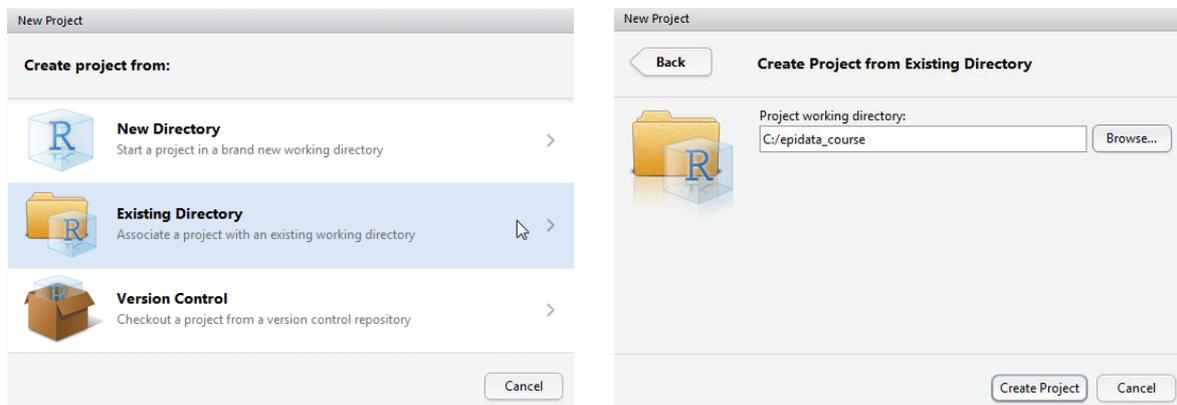
We cleared the screen of the Console by using the shortcut **CTRL+L**. To switch between the Console and the Source editor quadrants use **CTRL+1** to get to the Source editor, and **CTRL+2** to get to the Console.

Of course, you are free to arrange it at your liking, but for the time being in this course, we will refer to this arrangement.

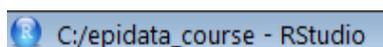
First we Create project from an Existing directory:



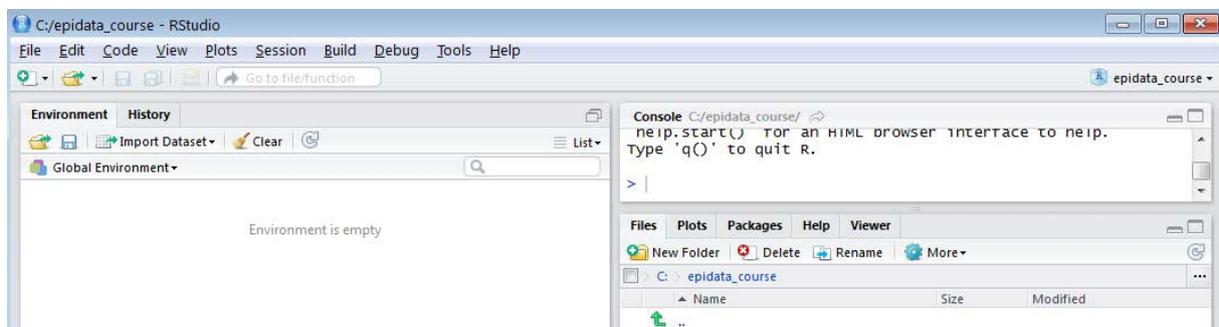
and browse for the EPIDATA_COURSE project folder:



You note on the left top:



and note that the left side has been collapsed to show the environment only and on the right side we are in Files:



If you click on the toggle icon above Source:



You see the split screen again.

Note also that RStudio and R use forward slashes “C:/epidata_course/” in the Console. RStudio actually created a sub-folder “.Rproj.user\” which is not public and by default hidden from view.

We are therefore all set to go. To introduce the workings of R we will create a two-by-two table as in a case-control study and calculate the odds ratio with a Woolf confidence interval.

A case-control study example

Our reference text for statistical calculations like for EpiData Analysis is, whenever possible, the text by Altman and collaborators:

Altman D G, Machin D, Bryant T N, Gardner M J. Statistics with confidence. 2nd edition. Bristol: BMJ Group, 2000: pp 61-62.

The odds ratio is defined as the odds of exposure among cases divided by the odds of exposure among controls. Woolf’s logit method uses normal approximation to the distribution of the logarithm of the odds ratio in which the standard error is

$$SE(\log_e OR) = \text{SQRT}(1/a + 1/b + 1/c + 1/d)$$

To obtain the 95% confidence interval, we calculate the quantities:

$$95\% \text{ CI} = \log_e OR \pm 1.96 * SE(\log_e OR)$$

By exponentiating, we get the 95% confidence interval.

To enable verification of our numbers, we are going to use the same example as Altman uses in table 7.4 for the worked example:

ABO non-secretor state	Study group		Total
	Cases	Controls	
Yes	54	89	143
No	60	245	305
Total	114	334	448

We will be writing our commands into the upper left quadrant (Source editor). When we run it, it will show in the right upper quadrant (the Console). Objects that we create will be found in the left lower quadrant.

In the Source editor, you should now have the empty workspace like any text editor:



Type:

```
a <- 54
```

“a” is what we call an “object” to which we assign (using “<-”) the value 54. Some people say “Everything is an object in R”. There is (incomplete, though) truth to this. Anything and everything can be assigned to an object. Here we assign a number to an object “a”. We can name our objects whatever we want them to be called:

```
A  
a  
a.plus.b
```

but note importantly, they are case-sensitive: “A” is not the same as “a” in R. Another important note to make is that the value of an object can be overwritten without any warning.

You probably type the lesser sign “<”, followed by the minus sign “-”. RStudio allows you to use **ALT+** alone (the ALT key plus the minus sign) and you get the “assign” symbol combination including the spaces that we often like to make in R.

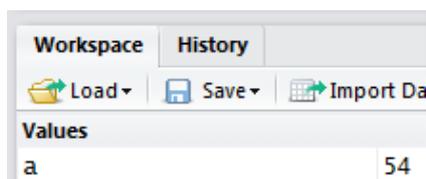
Nothing happened when you hit the carriage return key and this is how it is supposed to be in the Source editor. You only type here your stuff. To execute it, you need to tell it to do it. It is like writing commands in an EpiData Analysis program, and like in the latter you have:



a Run command: anywhere on a command line, it runs the current line, marking whatever you wish, it runs the marked section, nothing new here for the seasoned EpiData user. If we run this line, two things happen: in the right upper quadrant, the Console, you get:



This is confirmation that the command Run was executed. In the left lower quadrant, the Values, you get:



As we are going to use the Run command often, it is useful to know that the shortcut **CTRL+Enter** accomplishes the same thing as Run. **CTRL+Shift+Enter** runs the entire script.

All objects at your disposal will be listed here. In the Console, the value of object “a” is not visible. Some things you do will give you the result immediately, others will not, this assignment obviously belongs to the latter category. But type into the second line below the assignment:

```
365/7
```

and Run it. What you get now in your console is:

```
Console C:/epidata_course/ ↗
> a <- 54
> 365/7
[1] 52.14286
>
```

both the command you wrote and the result from the calculation. If you write into your working space:

a

and Run it you get:

```
Console C:/epidata_course/ ↗
> a <- 54
> 365/7
[1] 52.14286
> a
[1] 54
```

CTRL+L clears the Console.

Let's do the other assignments, writing each on a separate line:

```
b <- 89
c <- 60
d <- 245
```

then Run after marking. You should now have all four objects visible in the lower left quadrant:

Workspace		History
Load	Save	Import Dataset
Values		
a	54	
b	89	
c	60	
d	245	

While we have our four objects that make up the core of the 2-by-2 table, we would like to have the complete table including its marginals, and all assigned to one object.

To this end we will first create a **vector**, that is a one-dimensional object containing the values of a, b, and their sum. To keep with intuition, we will call this object "ab". Type:

```
ab <- c(a, b, a+b)
```

The "c" stands for "concatenate", that is putting a string consisting of sub-strings together. The elements of the vector are in parenthesis, separated by commas. They are, in the order in which we want to have them, our created two objects a and b, and the third element the sum of the two.

To see the result, type

```
ab
```

on a new line. Alternatively, you could also put a series of commands on one single line, separated by semi-colons, like in:

```
ab <- c(a, b, a+b); ab
```

and you should get:

```
Console C:/epidata_course/ ↵
> ab <- c(a, b, a+b); ab
[1] 54 89 143
```

Note, that in the left lower quadrant, you get now the object `ab` and a definition of what it is:

```
Values
a          54
ab        numeric[3]
```

Analogously, we create the second vector for the non-exposed:

```
cd <- c(c, d, c+d)
```

and get accordingly:

```
> cd <- c(c, d, c+d); cd
[1] 60 245 305
```

We have thus two vectors, one for the exposed and another for the non-exposed. Now we want them to be together in one single object that we will call `abcd`. The command to “bind” two rows together is:

```
abcd <- rbind(ab, cd); abcd
```

and we get:

```
> abcd <- rbind(ab, cd); abcd
  [,1] [,2] [,3]
ab  54  89 143
cd  60 245 305
```

The “`rbind`” is for “row binding”. Of course, if there is “`rbind`”, there must also be a “`cbind`” for column binding. Note also on the top `[,1]`, `[,2]` and `[,3]`: R labels (missing an actual label) things in a Cartesian way as `[row, col]`. R writes the indices to designate position in square brackets. As the column headers don’t have a row designation, it is thus the comma followed by the column number. What might be the internal designation for the location of the number 54? Yes, it is `[1 , 1]` for first row, first column, while 305 is in position `[2 , 3]`.

We have combined two **vectors** (one-dimensional objects) and the result is a **matrix** (a two-dimensional object). We could have created a matrix directly from the four objects `a`, `b`, `c`, and `d`. However, we have to pay attention to the correct the sequence:

```
abcd_m <- matrix(c(a, c, b, d, a+b, c+d), nrow=2, ncol=3); abcd_m
```

noting that the `matrix` function reads the data column-wise (!), we get:

```
> abcd_m <- matrix(c(a, c, b, d, a+b, c+d), nrow=2, ncol=3); abcd_m
  [,1] [,2] [,3]
[1,]  54  89 143
[2,]  60 245 305
```

We don’t need to write “`nrow=2, ncol=3`” to tell R that these 6 objects must be allocated to 2 rows and 3 columns, it suffices to stated:

```
abcd_m <- matrix(c(a, c, b, d, a+b, c+d), 2, 3); abcd_m
```

as R understands the first assignment (“2”) to be for the number of rows, and the second (“3”) for the number of columns. If we don’t pay attention to the default sequence row-column:

```
abcd_m <- matrix(c(a, c, b, d, a+b, c+d), 3, 2); abcd_m
```

we get a solution, but not the intended one:

```
> abcd_m <- matrix(c(a, c, b, d, a+b, c+d), 3, 2); abcd_m
      [,1] [,2]
[1,]   54  245
[2,]   60  143
[3,]   89  305
```

In contrast, if we specify:

```
abcd_m <- matrix(c(a, c, b, d, a+b, c+d), ncol=3, nrow=2); abcd_m
```

we will get what we intend:

```
> abcd_m <- matrix(c(a, c, b, d, a+b, c+d), ncol=3, nrow=2); abcd_m
      [,1] [,2] [,3]
[1,]   54   89  143
[2,]   60  245  305
```

Thus, as beginners, not yet fully conscious of default sequences, it is safer to be overly explicit.

To get back to the matrix: A matrix is a “two-dimensional table of like elements” (Aragón). Remove the object `abcd_m` as we are not going down this way for the time being:

```
rm(abcd_m)
```

where `rm` is the command to remove an object which is specified within the following paranthesis.

To complete the table, we need to make a row that is the sum of the first and the second row. We will call it `rtot` and it must be:

```
rtot <- abcd[1,] + abcd[2,]; rtot
```

because we are adding values of the respective column in row 1 `[1,]` to the values in row 2 `[2,]`. The result:

```
> rtot <- abcd[1,] + abcd[2,]; rtot
[1] 114 334 448
```

The last thing we have to do to get our complete table with all the marginals is row binding to an object that we will call `tab`:

```
tab <- rbind(abcd, rtot); tab
```

giving:

```
> tab <- rbind(abcd, rtot); tab
      [,1] [,2] [,3]
ab     54   89  143
cd     60  245  305
rtot  114  334  448
```

Before we lose our hard work, let's save what we typed so far in our editing window (the Source as we named the upper left quadrant initially) as “`e_ex01_or.r`” (by going through File | Save or by clicking the diskette icon). To our knowledge, R is not very particular what extension we give things, but we keep anyhow to what is quite widely common usage, and that is to give the extension `*.r`. You note how the file name replaces the Untitled:



While we have everything we need, we could make it a bit more appealing with appropriate column and row labeling which is done as:

```
colnames(tab) <- c("Case", "Control", "Total")
rownames(tab) <- c("Exp+", "Exp-", "Total")
```

and check to verify with:

```
tab
```

	Case	Control	Total
Exp+	54	89	143
Exp-	60	245	305
Total	114	334	448

“colnames” and “rownames” are the respective commands to give labels to columns and rows, while what follows in the parenthesis (tab) is the object to which the command applies.

We can also label the dimensions:

```
names(dimnames(tab)) <- c("Exposure", "Status"); tab
```

and get:

	Status		
Exposure	Case	Control	Total
Exp+	54	89	143
Exp-	60	245	305
Total	114	334	448

So far so good, but all we have is input, but what we really need are calculations. The odds ratio should be simple enough, we need to define the location by row and column of a, b, c, and d to then calculate $(a/c) / (b/d)$, thus:

```
or <- (tab[1,1]/tab[2,1])/(tab[1,2]/tab[2,2]); or
```

Note again here the brackets (not parentheses): whenever something needs to be indexed in R, brackets are used, here for the Cartesian positioning [row, col] but also if one wishes to describe the position of a record.

We specify the object tab and the position of its numeric elements to get the correct calculation, and the result should be:

```
2.477528
```

For the calculation of the 95% confidence interval (CI) we do this preferably in more than one step, first calculating the standard error se:

```
se <- sqrt(1/a+1/b+1/c+1/d)
```

We take here directly our four input objects as this calculation is independent of the position in the table. Next, for the lower and upper 95% CI:

```
or.ci.lower <- exp(log(or)-1.96*se)
or.ci.upper <- exp(log(or)+1.96*se)
```

Let's remove the objects that we don't need, the command to remove objects is rm:

```
rm(ab, cd, abcd, rtot, se)
```

Finally, let's get an output that summarizes it all together:

```
print(tab)
cat("\nOR:", round(or, digits=3), "\n95% CI:", round(or.ci.lower,
  digits=3), "to ", round(or.ci.upper, digits=3))
```

With the function “cat” we avoid quotes where we don't need them, and then take control by writing them where we need them:

```
x <- c("Ajay", "Hoa", "Zaw")
x
```

gives:

```
"Ajay" "Hoa" "Zaw"
```

but

```
cat(x)
```

gives:

```
Ajay Hoa Zaw
```

The function “\n” is equivalent to a hard carriage return.

The function round is followed by the object, then a comma, then the number of digits to which it is rounded, all in parentheses as is usual with functions.

Note that it is best if you write everything of the second item (starting with “cat”) on a single line. We should get:

```
      Status
Exposure Case Control Total
Exp+      54         89   143
Exp-      60        245   305
Total    114        334   448
```

```
OR: 2.478
```

```
95% CI: 1.595 to 3.849
```

Verifying by comparison with the results in Altman's shows that what we obtained is correct.

To make this more generically useful, we remove now all non-essential things, first of all the input objects a, b, c, and d. While not necessary (any object can be overwritten without warning), it is intuitively more appealing if we just keep the rest and then enter new values for a, b, c, and d and then run our “stripped” e_ex01_or.r. Thus, the stripped e_ex01_or.r may look like (note the hash “#” symbol at the beginning of the line correspondes to the asterisk “*” in EpiData Analysis denoting a non-executed comment):

```
# Complete 2 by 2 table from
# a, b, c, d and calculate odds ratio with Wolf CI
# Assign below values for a, b, c, and d as in:
# a <- 100

ab <- c(a, b, a+b)
cd <- c(c, d, c+d)
abcd <- rbind(ab, cd)
```

```

rtot <- abcd[1,]+abcd[2,]
tab <- rbind(abcd, rtot)
colnames(tab) <- cbind("Case", "Control", "Total")
rownames(tab) <- rbind("Exp+", "Exp-", "Total")
names(dimnames(tab)) <- c("Exposure", "Status"); tab
or <- (tab[1,1]/tab[2,1])/(tab[1,2]/tab[2,2])
se <- 1.96*sqrt(1/a+1/b+1/c+1/d)
or.ci.lower <- exp(log(or)-se)
or.ci.upper <- exp(log(or)+se)

print(tab)
cat("\nOR:", round(or, digits=3), "\n95% CI:", round(or.ci.lower,
digits=3), round(or.ci.upper, digits=3), "\n")

```

You would now assign any value to your input objects a, b, c, and d, for instance directly in the console and then open this script and Run it, and there it will be.

To quit R, type:

```
q()
```

You will be prompted whether you wish to save the workspace image:

```
> q()
Save workspace image to C:/epidata_course/.RData? [y/n/c]:
```

to which you can safely answer with “n”: we have saved our scripts and do not need to save in addition the log file.

Task

*In analogy to the calculation of the odds ratio, write an R script `e_ex01_rr.r` that calculates the relative risk from two proportions (thus not a ratio of incidence rates, but a ratio of two prevalence proportions) for a study used in Altman’s textbook. It has the isolation of *Helicobacter pylori* as the outcome and the history of an ulcer in the mother as the exposure. We use the following set-up of notations:*

Exposure	Outcome		Total
	Ill	Healthy	
Yes	A	B	A+B
No	C	D	C+D
Total	A+B	B+D	A+B+C+D

The data provided by Altman in table 7.2 (page 59) are as follows:

Mother with a history of ulcer	<i>H pylori</i> isolated		Total
	Yes	No	

Yes	6	16	22
No	112	729	841
Total	118	745	863

The relative risk is calculated by:

$$R = [A/(A+B)]/[C/(C+D)]$$

The standard error of the log_eR is:

$$se = \text{SQRT}(1/A-1/(A+B)+1/C-1/(C+D))$$

The 95% CI is thus:

$$95\%CI = \exp(\log_e R \pm 1.96*se)$$

Solution to Exercise 1: Introduction to R software: basics

Key points:

- R is both a language and an environment of computing
- Anything and everything can be assigned to an object
- Object names are case-sensitive
- A vector is a one-dimensional object of like elements
- A matrix is a two-dimensional table (another object) of like elements
- Vectors can be bound to matrices

Task

*In analogy to the calculation of the odds ratio, write an R script `e_ex01_rr.r` that calculates the relative risk from two proportions (thus not ratio of incidence rates, but ratio of two prevalence proportions) for a study used in Altman's textbook. It has the isolation of *Helicobacter pylori* as the outcome and the history of an ulcer in the mother as the exposure. We use the following set-up of notations:*

Exposure	Outcome		Total
	Ill	Healthy	
Yes	A	B	A+B
No	C	D	C+D
Total	A+B	B+D	A+B+C+D

The data provided by Altman in table 7.2 (page 59) are as follows:

Mother with a history of ulcer	<i>H pylori</i> isolated		Total
	Yes	No	
Yes	6	16	22
No	112	729	841
Total	118	745	863

The relative risk is calculated by:

$$R = (A / (A+B)) / (C / (C+D))$$

The standard error of the $\log_e R$ is:

$$se = \sqrt{1/A - 1/(A+B) + 1/C - 1/(C+D)}$$

The 95% CI is thus:

$$95\%CI = \exp(\log_e R \pm 1.96 * se)$$

Solution:

The solution for the `e_ex01_rr.r` is a straight forward modification of the `e_ex01_or.r`:

```
# Complete 2 by 2 table from
# A, B, C, D and calculate relative risk with Wolf CI

A <- 54
B <- 89
C <- 60
D <- 245

AB <- c(A, B, A+B)
CD <- c(C, D, C+D)
ABCD <- rbind(AB, CD)
rtot <- abcd[1,]+abcd[2,]
tab <- rbind(ABCD, rtot)
colnames(tab) <- cbind("Ill", "Healthy", "Total")
rownames(tab) <- rbind("Exp+", "Exp-", "Total")
names(dimnames(tab)) <- c("Exposure", "Status")
se <- sqrt(1/A-1/(A+B)+1/C-1/(C+D))
rr <- (tab[1,1]/tab[1,3])/(tab[2,1]/tab[2,3]); rr
rr.ci.lower <- exp(log(rr)-1.96*se)
rr.ci.upper <- exp(log(rr)+1.96*se)

print(tab)
cat("\nRR:", round(rr, digits=3), "\n95% CI:", round(rr.ci.lower,
digits=3), round(rr.ci.upper, digits=3), "\n")
```

The result is:

	Ill	Healthy	Total
Exp+	6	16	22
Exp-	112	729	841
Total	118	745	863

RR: 2.048
95% CI: 1.013 4.14

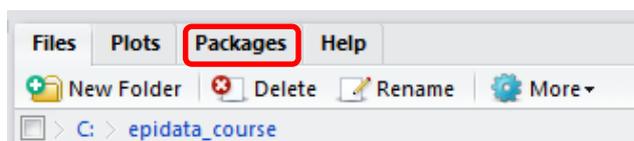
Exercise 2: Introduction to R software: data bases and functions

At the end of this exercise you should be able to:

- Know how to import a data base from another format
- Create a function and apply it to an analytic problem

What we did in Exercise 1 is unusual in analysis: we entered aggregate data and analyzed them, while the usual currency of analysis is appropriate aggregation of individual observations. Commonly, datasets have been entered in other software as R is not a data entry tool. In the context of this course, data will come as an EpiData dataset with a *.REC and a *.CHK file. R accepts a multitude of data formats for import. In fact, a package has been developed that expands on the number of formats of datasets which can be imported into R, including for instance Epi Info / EpiData *.REC files and Stata *.dta files.

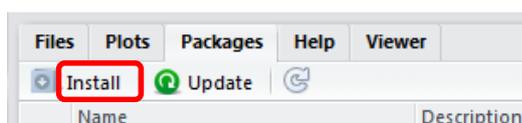
An important part of the versatility of R lies with the system of “packages”. What we have been using is the basic package, but if we want to utilize the capability of importing EpiData *.REC files or Stata *.dta files, the package “**Foreign**” should be added to our existing setup. In the lower right quadrant in RStudio, you see:



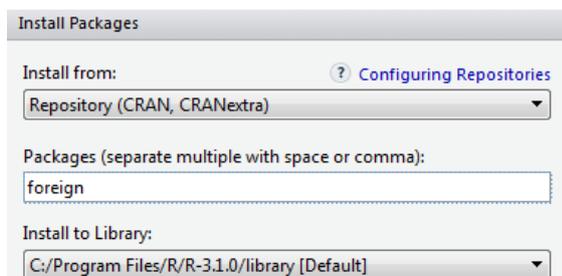
Click on Packages and you get a list of packages that are available with those loaded already ticked as for e.g. a basic statistical package:



The package we need is called foreign. You have to tick Install to get a menu:



You shouldn't need to do anything here except typing `foreign` into the line Packages and then Install as RStudio knows by definition where you have installed your copy of R. You need to be on the Internet though to get to the repository:



Alternatively, you can use the included Zip package in the course material, but then you need to change the Install from and get the downloaded zip file. But as always, to ensure obtaining

the most up-to-date version, get it whenever possible directly from the internet (<http://cran.r-project.org/web/packages/foreign/index.html>).

Having installed the package does not automatically put it at your disposal, you will have to call it. We start a new `*.r` script that we will name `e_ex02.r`.

In the required files on the course site you find the EpiData REC/CHK pair `e_ex02.rec` and `e_ex02.chk`. This is an abbreviated set of treatment results among patients with laboratory-confirmed multidrug resistance in Bangladesh.

The package `foreign` allows importing `*.REC` files with the command:

```
read.epiinfo("c:/epidata_course/e_ex02.rec")
```

However, this approach disregards the metadata (the `*.CHK` file) and only imports the field values from the `*.REC` file. We can write value labels in R, but we would first need to examine the `*.CHK` file and be careful about errors: tedious and error-prone. More efficient and safer is to use EpiData Entry 3.1 (or the EpiData Manager) to export the `e_ex02.*` file pair to a Stata `*.dta` file that contains both values and labels and then read the Stata file into R. Once you have exported the EpiData file pair to a Stata `e_ex02.dta` file, we are ready for import into R.

Being sticklers in labeling things, we start with a title, followed by actually invoking the now installed package **foreign**:

```
# Import an EpiData REC file

library(foreign)
e_ex02.dat <- read.dta("c:/epidata_course/e_ex02.dta")
```

Because we have made the “`epidata_course`” our project folder, it suffices to type:

```
e_ex02.dat <- read.dta("e_ex02.dta")
```

We read the Stata file by putting file name (and file path) in quotation marks inside a parenthesis (note the use of forward slashes) as in EpiData. We assign the imported file to an object that we will call `e_ex02.dat`. The `*.dat` is not required, it is just an object after all. We could call the object “`a`” if we wished to do so. The period separating the elements “`e_ex02`” and “`dat`” is not designating an extension, it is just one of the many ways R allows giving names to objects.

In a second step, we write the data to disk:

```
library(foreign)
e_ex02.dat <- read.dta("c:/epidata_course/e_ex02.dta")
write.table(e_ex02.dat, file="e_ex02.dat", row.names=TRUE)
```

Perhaps this is as good as any occasion to introduce the Help functions in R. If you know already “`write.table`”, then it is easy, type:

```
help(write.table)
```

and you get in the lower right-hand corner box quite extensive information (try it out!).

To see what the import did, type:

```
e_ex02.dat[1:5,]
```

This gives the first 5 records:

```

age      fq04      sex  totobstime  agequart      agemed  outcome07  outcome02
1  28 susceptible Male      999 quartile 2  Below median  Cure      Success
2  45 susceptible Male      482 quartile 4  Median or larger  Cure      Success
3  22 susceptible Female    190 quartile 1  Below median  Default   Failure
4  26 susceptible Female      72 quartile 2  Below median  Default   Failure
5  20 susceptible Female    1000 quartile 1  Below median  Cure      Success

      pza02      kmy02      pth02      cxr02
1 PZA not known resistant KM not known resistant PTH not known resistant  Bilateral
2 PZA not known resistant KM not known resistant PTH not known resistant  Bilateral
3 PZA not known resistant KM not known resistant PTH not known resistant  Bilateral
4 PZA not known resistant KM not known resistant PTH resistant Not known bilateral
5 PZA not known resistant KM not known resistant PTH not known resistant  Bilateral

```

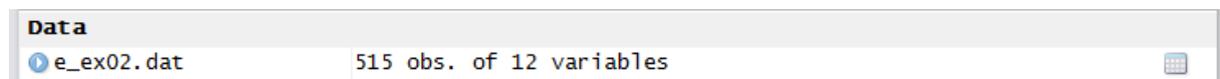
To see the variable names only, type:

```
# Get the list of variable names only
names(e_ex02.dat)
```

and get:

```
[1] "age"      "fq04"      "sex"      "totobstime" "agequart"  "agemed"    "outcome07"
[8] "outcome02" "pza02"     "kmy02"     "pth02"      "cxr02"
```

R has a special way to deal with missing information. It uses NA to denote any missing value, be it numeric or text. If NA is the assigned value, then a record in an analysis using the variable with such a value will be excluded. We will use different approaches to the analysis. In some analyses, we will include all 515 records – how many records we have, can be seen in the Environment lower quadrant of the left panel:



In another analysis, we will exclude missing observations. We will create three datasets:

- e_ex02_01.dat is the unaltered full set with missing values designated by NA, R's way to define missing values.
- e_ex02_02.dat is the dataset containing only records with information on initial fluoroquinolone resistance.
- e_ex02_03.dat is the subset of patients with initial ofloxacin resistance with either a bacteriologically unsuccessful outcome (failure or relapse) or a bacteriologically successful outcome (relapse-free cure or treatment completion).

The dataset

To get information on the structure of the dataset:

```
str(e_ex02.dat)
```

and we get (top only shown):

```
'data.frame': 515 obs. of 12 variables:
 $ age      : int  28 45 22 26 20 31 35 24 60 20 ...
 $ fq04     : Factor w/ 4 levels "Susceptible",...: 1 1 1 1 1 1 1 1 1 1 ...
 $ sex      : Factor w/ 2 levels "Male","Female": 1 1 2 2 2 1 2 1 1 1 ...
 $ tobstime: int  999 482 190 72 1000 1010 37 250 832 791 ...
 $ agequart : Factor w/ 4 levels "Quartile 1","Quartile 2",...: 2 4 1 2 1 3 3 2 4 1 ...
 $ agemed   : Factor w/ 2 levels "Below median",...: 1 2 1 1 1 2 2 1 2 1 ...
 $ outcome07 : Factor w/ 7 levels "Cure","Completion",...: 1 1 5 5 1 1 5 4 1 1 ...
 $ outcome02 : Factor w/ 2 levels "Success","Failure": 1 1 2 2 1 1 2 2 1 1 ...
 $ pza02    : Factor w/ 2 levels "PZA not known resistant",...: 1 1 1 1 1 1 1 1 1 1 ...
 $ kmy02    : Factor w/ 2 levels "KM not known resistant",...: 1 1 1 1 1 1 1 1 1 1 ...
 $ pth02    : Factor w/ 2 levels "PTH not known resistant",...: 1 1 1 2 1 1 1 1 1 1 ...
 $ cxr02    : Factor w/ 2 levels "Not known bilateral",...: 2 2 2 1 2 2 2 2 2 2 ...
```

The full dataset thus contains 515 records with 12 variables (rearranged):

AGE: Age in years as an integer.

AGEQUART: Age as a categorical variable in quartiles.

AGEMED: Age as a binary categorical variable (below the median, and median and larger).

SEX: Patient's sex as categorical variable (Female, Male).

OUTCOME07: 7-level treatment outcome (Cure, Completion, Failure, Death, Default, Relapse, Reinfection).

OUTCOME02: Outcome as a binary categorical variable, i.e. successful (relapse-free cure or completion) vs all other outcomes (Success, Failure).

TOTOBSTIME: Time of observation, an integer in days from treatment start until an event occurred (see later) or the observation time ended because of follow-up completion.

FQ04: Drug susceptibility test result for ofloxacin. If resistant, the minimum inhibitory concentration to gatifloxacin was determined. This categorical variable has four levels (Susceptible, Low-level resistance, High-level resistance, Missing).

PZA02: Drug susceptibility test result of molecular (*pncA*) for pyrazinamide as a categorical binary variable, resistant vs not known to be resistant (PZA not known resistant, PZA resistant).

KMY02: Drug susceptibility test result of phenotypic testing for kanamycin as a categorical binary variable, resistant vs not known to be resistant (KM not known resistant, KM resistant).

PTH02: Drug susceptibility test result of phenotypic testing for prothionamide as a categorical binary variable, resistant vs not known to be resistant (PTH not known resistant, PTH resistant).

CXR02: Radiographic disease extent as a categorical binary variable, bilateral vs not known to be bilateral (Not known bilateral, Bilateral).

The information for some variables was complete (age and sex), for some very few had missing information (fluoroquinolone drug susceptibility test result, missing assigned to a defined category). Some variables had quite a few missing (known *pncA* sequencing result for pyrazinamide), others were quite complete (like radiographic disease extent). One could argue to deal differently with the missing than what was done for this exercise in this simplified categorization as "not known resistant". In any case, we chose here certain simplifications that are unlikely to importantly bias the planned analysis in the wrong direction. For some key questions, a "purist's" approach is chosen to deal with missing values.

The easiest to create is the full set: the only manipulation required is an assignment of the only variable with a missing value, the variable FQ04 for ofloxacin drug susceptibility test result:

```
e_ex02.dat[e_ex02.dat$fq04=="Missing", "fluoroquinolone"] <- NA
```

Several new and important things are seen in this command line. Starting with the outermost right assignment NA, we note how R deals with missing data. In the data entry exercises with EpiData software we had taught (a bit dogmatically) that we wish to have a value (commonly designated as 9, 99, 9.99 etc) for all records with a missing value in a field. But this is by no means a universal standard. Rather more commonly such a field may remain empty or might have some value for missing or might be a mixture of both. Dealing correctly with missing values remains one of the most challenging tasks in any analysis. When we read the dataset and look at a frequency as follows:

```
table(e_ex02.dat$fq04)
```

we get (note the use of “table” rather than “tables” as in EpiData) before and after the NA assignment above:

Before:

```
Susceptible  Low-level resistance High-level resistance      Missing
      439                33                29                14
```

After:

```
Susceptible  Low-level resistance High-level resistance      Missing
      439                33                29                0
```

We could deal with the 14 missing as a category and leave it just as it is with these 4 category levels. In our, we will, however, pay special interest to this variable. As 14 of 515 is just 2.7% of all observations, there is no important bias if we later simply exclude these 14 cases. There are different ways to sub-setting, but we assign the R designation for missing, which is NA.

To the very left we have the name of the dataset e_ex02.dat. Inside the brackets we identify the variable within the dataset as in:

```
setname$varname
```

The “==” is new and differs what we learned in EpiData. The following are the logical operators used in R:

Operator	Description
<	less than
<=	less than or equal to
>	greater than
>=	greater than or equal to
==	exactly equal to
!=	not equal to
!x	Not x
x y	x OR y
x & y	x AND y
isTRUE(x)	test if X is TRUE

This is from the Quick-R website:

<http://www.statmethods.net/>

an extraordinarily useful website to get – as it says – quick answers to questions on R, highly recommendable! R (and Stata) uses the double == sign for equality rather than the single equal sign because the latter (=) can be used as “assign” instead of the <- we now got used to (and will stick to).

Thus, to get back:

```
e_ex02.dat$fq04=="Missing"
```

This identifies the variable `fq04` within the dataset `e_ex02.dat` and identifies records in which the category value is “Missing” (note that the value is not the “value” from the *.REC file, but the value is the label obtained from the Stata file for easy identification). After the comma, we state that we retain the variable name `fq04`:

```
e_ex02.dat[e_ex02.dat$fq04=="Missing", "fq04"] <- NA
```

we could check whether we have now records with NAs with the important command:

```
na.is(x)
```

which is used to find out which elements of `x` are recorded as missing (NA), i.e. here:

```
is.na(e_ex02.dat$fq04)
```

and we get (an excerpt from the list of all 515 records):

```
[289] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE TRUE FALSE FALSE FALSE FALSE FALSE FALSE TRUE
[305] FALSE FALSE
[321] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE TRUE FALSE FALSE FALSE FALSE FALSE FALSE
```

To assign this revised set to the new object with the desired name and to ensure that it is a “data frame”, we write:

```
e_ex02_01.dat <- data.frame(e_ex02.dat)
```

What is called a “data frame” in R is just what we EpiData users call a dataset. “It is a list of vectors and / or factors of the same length that are related “across” such that data in the same position come from the same experimental unit” (formulation copied from Peter Dalgaard, Introductory statistics with R, 2008). The distinction is made because R can deal with other types of data collection.

Sub-setting

The next dataset is trickier to create. An excellent source on the internet to find quick answers is Quick-R at <http://www.statmethods.net/index.html>. If we enter “subsets” into the Search box, we see among several option, option Nr 1:

```
1. Subsetting Data (36%)
   Description: Keeping or Deleting Variables, Observations, Random Samples ...
   URL: http://www.statmethods.net/management/subset.html
```

Clicking on the link we get all we need to know for starters. To select observations, the full explicit is:

```
newdata <- mydata[which(mydata$gender=='F' & mydata$age > 65),]
```

There is a short-cut allowed:

```
newdata <- mydata[which(gender=='F' & age > 65),]
```

Note here the single quotes to enclose text, but we can also use double quotes. In any case, we need to know the criteria. We mentioned above that we want to do a sub-analysis on patients with initial ofloxacin resistance. If we just write some table command, we might have the problem of ambiguity on which of the currently existing two datasets the command needs to be executed.

Our dataset to create is to be named “e_ex02_02.dat”. It will only contain records without missing data for the variable fq04. We are thus making a selection in EpiData language or sub-setting in R language. If we start from the originally read dataset, we write:

```
e_ex02_02.dat <- e_ex02.dat[which(e_ex02.dat$fq04 != "Missing"), ]
```

Alternatively, and perhaps simpler, we can use the subset function:

```
e_ex02_02.dat <- subset(e_ex02.dat, fq04 != "Missing")
```

Note (and see above) the “!=” denoting “unequal”.

To make the even more complex third dataset, the newcomer might best first stick to the fully explicit. We thus write three lines, making intermediary datasets, each line being self-explanatory:

We start with the dataset e_ex02_02.dat which has 501 records. Excluding 26 deaths defined in the variable outcome07, we get 475 observations in the new dataset e_ex02_02_03a.dat:

```
e_ex02_03a.dat <- subset(e_ex02_02.dat, outcome07 != "Death")
```

Excluding then 40 defaulters defined in the variable outcome07, we get 435 observations in the new dataset e_ex02_02_03b.dat:

```
e_ex02_03b.dat <- subset(e_ex02_03a.dat, outcome07 != "Default")
```

Excluding finally 382 patients with initial fluoroquinolone susceptibility defined in the variable fq04, we get 53 observations in the final desired dataset e_ex02_02_03.dat:

```
e_ex02_03.dat <- subset(e_ex02_03b.dat, fq04 != "Susceptible")
```

Of course, this can all be written into a single command line:

```
e_ex02_03.dat <- subset(e_ex02_02.dat, outcome07 != "Death" & outcome07 !=  
  "Default" & fq04 != "Susceptible")
```

and one gets the same result of 53 observations.

Finally, to save the three datasets to disk, we do as above:

```
write.table(e_ex02_01.dat, file="e_ex02_01.dat", row.names=TRUE)  
write.table(e_ex02_02.dat, file="e_ex02_02.dat", row.names=TRUE)  
write.table(e_ex02_03.dat, file="e_ex02_03.dat", row.names=TRUE)
```

A generically applicable function for the analysis of a 2-by-2 table

If we had a variable `fq02` to denote a binary outcome for initial fluoroquinolone (FQ) resistance (aggregating low and high resistance) and excluding those with missing fluoroquinolone test result, an analysis by the binary outcome in EpiData Analysis is as follows:

```
select fq02<>9 // exclude records with missing FQ result
tables outcome02 fq02 /o
```

which gives:

Outcome: Binomial outcome			
Binomial FQ resistance	Failure	Success	Total
Resistant	18	44	62
Susceptible	59	380	439
Total	77	424	501

Exposure: Binomial FQ resistance = Resistant
Outcome: Binomial outcome = Failure

Odds Ratio = 2.63 (95% CI: 1.43-4.86) (Robins, Greenland, Breslow CI)

Second, we make a stratified analysis to look at the influence of SEX:

```
tables outcome02 fq02 sex /o
```

which gives a summary:

Binomial outcome by Binomial FQ resistance adjusted for Patient's sex			
	N = 501	N	OR (95% CI)
Crude		501	2.63 (1.43-4.86)
Adjusted		501	2.65 (1.43-4.93)
Patient's sex: Male	355		1.91 (0.85-4.29)
Patient's sex: Female	146		4.54 (1.65-12.54)

Summary Estimates
Total 2 strata. 2 informative & 0 non-informative.
Exposure: Binomial FQ resistance = Resistant
Outcome: Binomial outcome = Failure

by the Mantel-Haenszel procedure.

The first table in R requires that we make a new variable for a binary result of fluoroquinolone resistance and then create the table:

```
e_ex02_02.dat[e_ex02_02.dat$fq04=="Susceptible", "fq02"] <- "1-Susceptible"
e_ex02_02.dat[e_ex02_02.dat$fq04=="Low-level resistance", "fq02"] <- "2-Resistant"
e_ex02_02.dat[e_ex02_02.dat$fq04=="High-level resistance", "fq02"] <- "2-Resistant"
e_ex02_fq02.dat <- data.frame(e_ex02_02.dat)
```

At this point let's introduce `attach`. Followed in parenthesis by the dataset name:

```
attach(e_ex02_02_fq02.dat)
```

it puts the dataset `e_ex02_02_fq02.dat` into the search path and we do thus not to need to repeat the dataset name all the time, it suffices to write the variable names. It is equally important not to forget to detach a dataset to get it out of the path. Thus, we type here:

```
detach(e_ex02_02.dat)
```

If the data set is in the path, what would otherwise be:

```
table(e_ex02_fq02.dat$fq02, e_ex02_fq02.dat$outcome02)
```

becomes simplified:

```
table(fq02, outcome02)
```

This gives just the bare bone core of the correct output table:

	Success	Failure
1-Susceptible	380	59
2-Resistant	44	18

Note here that for the R command `table` the sequence is different from EpiData: the first variable gives the row and the second variable the column. We noted earlier that the default sequence in R is row, then column. Note also that we numbered “1-Susceptible” and “2-Resistant” to get these into our preferred alphabetical sequence (the default would be the inverse).

Could we somehow use what we did in Exercise 1, edit it a bit and get it functional for this situation? Open the `e_ex01_or.r` in the text editor and remove everything, except the following lines:

```
or <- (tab[1,1]/tab[2,1])/(tab[1,2]/tab[2,2])
se <- sqrt(1/a+1/b+1/c+1/d)
or.ci.lower <- exp(log(or)-1.96*se)
or.ci.upper <- exp(log(or)+1.96*se)
print(tab)
cat("\nOR:", round(or, digits=3), "\n95% CI:", round(or.ci.lower, digits=3),
    "to ", round(or.ci.upper, digits=3))
```

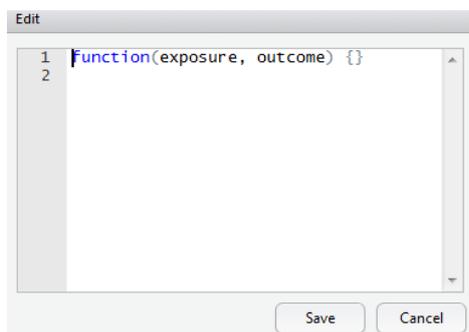
We are going to create a **Function** that can be applied in the future to the same setting. We have already used functions, `TABLE` is a function. To make our own function, type:

```
tab2by2 <- function(exposure, outcome) {}
```

This is the framework only, and it doesn’t yet do anything. To put it to useful work, type:

```
fix(tab2by2)
```

and a function editor window opens:



Our commands will go between the curly braces `{ }`. It is a custom (but no real need, but we will faithfully follow customs of those with more experience, they usually have their reasons) to put the opening brace `{` on a new line and then the commands after that on new lines, and have the closing brace `}` again at the end alone on a line. After we paste from the text editor what we have, we should thus get:

```

Edit
1 function(exposure, outcome)
2   {
3     or <- (tab[1,1]/tab[2,1])/(tab[1,
4     se <- sqrt(1/a+1/b+1/c+1/d)
5     or.ci.lower <- exp(log(or)-1.96*s
6     or.ci.upper <- exp(log(or)+1.96*s
7     print(tab)
8     cat("\nOR:", round(or, digits=3),
9   }

```

It is not yet quite correct, because there is no object `tab`. Thus type on the line before the “or”:

```
tab <- table(exposure, outcome)
```

That is, we assign the output of any two-by-two table of this format to an object `tab`. You may note that the RStudio short-cut key **ALT+** to get the assign combination “<-” is non-functional in functions, you have to type it out. As we are already using `a`, `b`, `c`, and `d`, we might as well keep these, and then make in the “or” line assignments to these, so that in the revision we get:

```

function(exposure, outcome)
{
  tab <- table(exposure, outcome)
  a <- tab[1,1]; b <- tab[2,1]; c <- tab[1,2]; d <- tab[2,2]
  or <- (a/c)/(b/d)
  se <- sqrt(1/a+1/b+1/c+1/d)
  or.ci.lower <- exp(log(or)-1.96*se)
  or.ci.upper <- exp(log(or)+1.96*se)
  print(tab)
  cat("\nOR:", round(or, digits=3), "\n95% CI:", round(or.ci.lower,
  digits=3), "to ", round(or.ci.upper, digits=3))
}

```

Save it. We could save it later when quitting in the workspace, but it is better to also save in in a file and doing that now:

```
save(tab2by2, file = "tab2by2.r")
```

Conversely, when we open RStudio anew, it can be loaded whenever we need it with:

```
load("tab2by2.r")
```

Note that you actually need the full path, thus in our case:

```
load("C:/epidata_course/tab2by2.r")
```

Also note that this approach results in an `*.r` file that is not anymore a straight text file (you see gibberish in the text editor). It seems to be the price that has to be paid for a function we write by ourselves.

Before we apply it, we need to reopen and Run our `e_ex02.r` script which should read now:

```

# Exercise 2: Introduction to R software: data bases and functions

#####
# 1) Import a Stata file

library(foreign)
e_ex02.dat <- read.dta("e_ex02.dta")
# e_ex02.dat <- read.dta("c:/epidata_course/e_ex02.dta")
write.table(e_ex02.dat, file="e_ex02.dat", row.names=TRUE)

```

```

# See the first 5 records
e_ex02.dat[1:5,]

# Get the list of variable names only
names(e_ex02.dat)

table(e_ex02.dat$fq04)

#####
# 2) Create full unaltered set (515 records)
e_ex02.dat[e_ex02.dat$fq04=="Missing", "fq04"] <- NA
is.na(e_ex02.dat$fq04)
e_ex02_01.dat <- data.frame(e_ex02.dat)

#####
# 3) Create a subset with records with known FQ DST result (401 records)
e_ex02_02.dat <- e_ex02.dat[which(e_ex02.dat$fq04 != "Missing"), ]
# or alternatively with the SUBSET function:
e_ex02_02.dat <- subset(e_ex02.dat, fq04 != "Missing")

#####
# 4) Create a subset with records with known FQ resistance and
# bacteriological success or failure (53 records)
# Commands on three lines
# e_ex02_03a.dat <- subset(e_ex02_02.dat, outcome07 != "Death")
# e_ex02_03b.dat <- subset(e_ex02_03a.dat, outcome07 != "Default")
# e_ex02_03.dat <- subset(e_ex02_03b.dat, fq04 != "Susceptible")
# Alternatively, commands on single line
e_ex02_03.dat <- subset(e_ex02_02.dat, outcome07 != "Death" & outcome07 != "Default"
  & fq04 != "Susceptible")

# Table of outcome by FQ resistance
# Create first a new variable FQ02

e_ex02_02.dat[e_ex02_02.dat$fq04=="Susceptible", "fq02"] <- "1-Susceptible"
e_ex02_02.dat[e_ex02_02.dat$fq04=="Low-level resistance", "fq02"] <- "2-Resistant"
e_ex02_02.dat[e_ex02_02.dat$fq04=="High-level resistance", "fq02"] <- "2-Resistant"
e_ex02_fq02.dat <- data.frame(e_ex02_02.dat)
table(e_ex02_fq02.dat$fq02, e_ex02_fq02.dat$outcome02)

write.table(e_ex02_01.dat, file="e_ex02_01.dat", row.names=TRUE)
write.table(e_ex02_02.dat, file="e_ex02_02.dat", row.names=TRUE)
write.table(e_ex02_03.dat, file="e_ex02_03.dat", row.names=TRUE)

#####
# 5) Make functions: 2-by-2 table
# Prepare making a function
# tab2by2 <- function(exposure, outcome) {}
# fix(tab2by2)
# save(tab2by2, file = "tab2by2.r")

attach(e_ex02_fq02.dat)

```

As the last command is ATTACH the proper file, let's try to apply it. Type:

```

load("C:/epidata_course/tab2by2.r")
tab2by2(fq02, outcome02)

```

and you should get:

```

      outcome
exposure Success Failure
1-Susceptible 380    59
2-Resistant   44    18

```

```

OR: 2.635
95% CI: 1.427 to 4.865

```

identical to the EpiData Analysis output (see above). Note that the R `load` function is necessary only after exiting R and reopening the script. Similarly, the three lines:

```

# tab2by2 <- function(exposure, outcome) {}
# fix(tab2by2)
# save(tab2by2, file = "tab2by2.r")

```

have been put as comments as they do not need repeat execution once done.

We can use this function for any other EXPOSURE-OUTCOME pair, such as:

```
tab2by2(sex, outcome02)
```

```

      outcome
exposure Success Failure
Male      308    47
Female    116    30

```

```

OR: 1.695
95% CI: 1.022 to 2.809

```

In EpiData Analysis:

```
tables outcome02 sex /o
```

Outcome: Binomial outcome			
Patient's sex	Failure	Success	Total
Female	30	116	146
Male	47	308	355
Total	77	424	501

Exposure: Patient's sex = Female
Outcome: Binomial outcome = Failure

Odds Ratio = 1.69 (95% CI: 1.02-2.81) (Robins, Greenland, Breslow CI)

We surely have to watch out for the variable `SEX`. You know from earlier that EpiData Analysis inverts (by design) the sequence of values for the exposure and outcome variables, while the net effect on the odds ratio thus remains unaffected.

To summarize: `TABLE` is an inbuilt function in R which cross tabulates the two variables. Replacing `TABLE` by our function ‘`tab2by2`’ runs our code creating the table and with the odds ratio and the 95% confidence interval, displayed it in the desired format. There are many functions that have been created already by R users and collaborators which we can make use of. Thus, a skill we have to master is to search for a function which suits our purpose and make correct and careful use of it. This example illustrates the basic mode of using and applying functions in R.

A generically applicable function for the analysis of a 2-by-2-by-2 table

In the previous paragraph we dealt with a **matrix** (a two-dimensional object of like elements). It was the special case of a 2-by-2 matrix. In this part, we move now forward to deal with **arrays**. An array is an n-dimensional table of like objects. In particular, we will concern

ourselves with the special case of a 2-by-2-by-2 table, i.e. a 3-dimensional table. We can construct such an object in R with the command:

```
z <- array(1:8, c(2, 2, 2)); z
```

and get:

```
, , 1
      [,1] [,2]
[1,]    1    3
[2,]    2    4

, , 2
      [,1] [,2]
[1,]    5    7
[2,]    6    8
```

What is the location of “7”? Try:

```
u <- z[1, 2, 2]; u
```

The third dimension (row is the first, the column the second, the added one the third) is also indicated by the the two leading commas:

```
, , 2
```

We mentioned before that the location [indexing] in R is defined within brackets []. The default sequence is by dimension, separated by commas, where the first dimension is always the row, the second the column and here the third is what we epidemiologists call the stratifying dimension. If we apply this concept of a multidimensional array to epidemiology, we thus have the concept of stratification.

The generic grammar for a table command for this basic stratification is:

```
table(exposure, outcome, stratvar)
```

or

```
table(rowvar, columnvar, stratvar)
```

where `stratvar` is the name of the stratifying variable. In our case, specifically, we try:

```
table(fq02, outcome02, sex)
```

to get, if the last `attach` command was `attach(e_ex02_fq02.dat)`:

```
, , sex = Male

      outcome02
fq02   Success Failure
1-Susceptible    274    38
2-Resistant      34     9

, , sex = Female

      outcome02
fq02   Success Failure
1-Susceptible    106    21
2-Resistant      10     9
```

In 1959, Mantel and Haenszel proposed an efficient method for estimating a summary odds ratio from a series of 2 by 2 tables. It is easy to apply and does not require iterative calculations. In the following introduction, we follow Schlesselman's explanations and use of example (Schlesselman J J. Case-control studies. Design, conduct, analysis. New York: Oxford University Press, 1982).

The Mantel-Haenszel estimate of the odds ratio (OR_{mh}), adjusted for the effects of the stratification variables is calculated as:

$$\text{SUM}(a_i d_i / n_i) / \text{SUM}(b_i c_i / n_i)$$

An estimate of the variance of the OR_{mh} has been proposed. If we define:

$$w_i = b_i c_i / n_i$$

and

$$v_i = (a_i + c_i) / a_i c_i + (b_i + d_i) / b_i d_i$$

then the variance of the $\log_e OR_{mh}$ is given by:

$$\text{var}(\ln OR_{mh}) \approx \text{SUM} w_i^2 v_i / (\text{SUM} w_i)^2$$

The approximate 95% confidence interval is thus given by:

$$\ln OR_{mh} \pm 1.96 * \text{SQRT}[\text{var}(\ln OR_{mh})]$$

If we apply this to our data:

	Female		Male	
	Failure	Success	Failure	Success
Resistant	9	10	9	34
Susceptible	21	106	38	274
OR	4.543		1.909	
n_i	146		355	
W_i	1.438		3.639	
v_i	0.268		0.170	
$a_i d_i / n_i$	6.534		6.946	
$b_i c_i / n_i$	1.438		3.639	
$\text{SUM}(a_i d_i / n_i)$	13.481			
$\text{SUM}(b_i c_i / n_i)$	5.078			
OR_{mh}	2.655			
$w_i^2 * v_i$	0.555		2.258	
$\text{SUM}(w_i^2 * v_i)$	2.813			
$(\text{SUM} w_i)^2$	25.784			
$\text{SUM}(w_i^2 * v_i) / (\text{SUM} w_i)^2$	0.109			
$\text{var}(\ln OR_{mh})$	0.109			
OR _{mh} -lower	1.390			
OR _{mh} -upper	5.072			

EpiData Analysis gives:

$$OR_{mh}: 2.65 \quad (1.43-4.93)$$

The point estimate is the same, but the 95% CI interval is slightly different because EpiData Analysis uses consistently the Robins, Greenland, Breslow confidence intervals, while we followed here Schlesselman's example using Hauck's interval. As the exercise is about the

principle of learning on how to go about when creating a function, the choice of the confidence interval does not really matter here.

We will keep to this most simple example with only 8 cells total among three variables: each variable is binary.

Let's best start from scratch and make a new function:

```
ormh <- function(exposure, outcome, stratvar) {}
```

We create a function ORMH (short for Mantel-Heanszel odds ratio) that takes three parameters, in that given sequence.

We propose to draft the content that goes between the curly braces `{ }` in the text editor before we put into it with `fix`. We can modify what we had before, just slightly so:

```
tab <- table(exposure, outcome, stratvar)
a1 <- tab[1,1,1]; b1 <- tab[2,1,1]; c1 <- tab[1,2,1]; d1 <- tab[2,2,1]
a2 <- tab[1,1,2]; b2 <- tab[2,1,2]; c2 <- tab[1,2,2]; d2 <- tab[2,2,2]
```

This takes into account that we work now with a 3-dimensional array and have `a1`, `a2`, and `b1`, `b2`, etc.

We have to create the $a_i d_i / n_i$ and the $b_i c_i / n_i$ for each of the two tables:

```
adn1 <- (a1*d1)/n1
adn2 <- (a2*d2)/n2
bcn1 <- (b1*c1)/n1
bcn2 <- (b2*c2)/n2
```

and from these we calculate the object MHOR:

```
mhor <- (adn1+adn2)/(bcn1+bcn2)
```

Note that we chose the object name to be `mhor` to distinguish in from the function name `ormh`. Before we get too much carried away in our excitement, we should check whether we are on the right track:

```
fix(ormh)
```

and insert what we currently have:

```
function(exposure, outcome, stratvar)
{
  tab <- table(exposure, outcome, stratvar)
  a1 <- tab[1,1,1]; b1 <- tab[2,1,1]; c1 <- tab[1,2,1]; d1 <- tab[2,2,1]
  a2 <- tab[1,1,2]; b2 <- tab[2,1,2]; c2 <- tab[1,2,2]; d2 <- tab[2,2,2]
  n1 <- a1+b1+c1+d1
  n2 <- a2+b2+c2+d2
  adn1 <- (a1*d1)/n1
  adn2 <- (a2*d2)/n2
  bcn1 <- (b1*c1)/n1
  bcn2 <- (b2*c2)/n2
  mhor <- (adn1+adn2)/(bcn1+bcn2)
  print(tab)
  cat("\nOR:", round(mhor, digits=3))
}
```

and test it with the command:

```
ormh(fq02, outcome02, sex)
```

We get:

```
, , stratvar = Male

      outcome
exposure  Success Failure
1-Susceptible  274     38
2-Resistant    34      9

, , stratvar = Female

      outcome
exposure  Success Failure
1-Susceptible  106     21
2-Resistant    10      9
```

OR: 2.655

We are definitely on the right track! Therefore, we might continue requiring next the two components w and v that we calculate as intermediate steps for calculation of the variance and then the variance itself:

```
w1 <- (b1*c1)/n1
w2 <- (b2*c2)/n2
v1 <- ((a1+c1)/(a1*c1))+((b1+d1)/(b1*d1))
v2 <- ((a2+c2)/(a2*c2))+((b2+d2)/(b2*d2))
varor <- ((w1^2*v1)+(w2^2*v2))/((w1+w2)^2)
```

Then we calculate the confidence interval:

```
se <-sqrt(var.mhor)
mhor.lower <-exp(log(mhor)-1.96*se)
mhor.upper <-exp(log(mhor)+1.96*se)
```

Finally, by adding what is to appear on the screen, we get the full function as:

```
function(exposure, outcome, stratvar)
{
  tab <- table(exposure, outcome, stratvar)
  a1 <- tab[1,1,1]; b1 <- tab[2,1,1]; c1 <- tab[1,2,1]; d1 <- tab[2,2,1]
  a2 <- tab[1,1,2]; b2 <- tab[2,1,2]; c2 <- tab[1,2,2]; d2 <- tab[2,2,2]
  a <- a1+a2; b <- b1+b2; c <-c1+c2; d <- d1+d2
  or <- (a/c)/(b/d)
  se <- sqrt(1/a+1/b+1/c+1/d)
  or.ci.lower <- exp(log(or)-1.96*se)
  or.ci.upper <- exp(log(or)+1.96*se)
  n1 <- a1+b1+c1+d1
  n2 <- a2+b2+c2+d2
  adn1 <- (a1*d1)/n1
  adn2 <- (a2*d2)/n2
  bcn1 <- (b1*c1)/n1
  bcn2 <- (b2*c2)/n2
  mhor <- (adn1+adn2)/(bcn1+bcn2)
  w1 <- (b1*c1)/n1
  w2 <- (b2*c2)/n2
  v1 <- ((a1+c1)/(a1*c1))+((b1+d1)/(b1*d1))
  v2 <- ((a2+c2)/(a2*c2))+((b2+d2)/(b2*d2))
  var.mhor <- ((w1^2*v1)+(w2^2*v2))/((w1+w2)^2)
```

```

se <-sqrt(var.mhor)
mhor.lower <-exp(log(mhor)-1.96*se)
mhor.upper <-exp(log(mhor)+1.96*se)
print(tab)
cat("\nOR adj:", round(mhor, digits=3), "\n95% CI:", round(mhor.lower,
  digits=3), "-", round(mhor.upper, digits=3))
}

```

If we test with:

```
ormh(oflres, outcome02, sex)
```

we get:

```

, , stratvar = Male

      outcome
exposure  Success Failure
1-Susceptible    274     38
2-Resistant      34      9

, , stratvar = Female

      outcome
exposure  Success Failure
1-Susceptible    106     21
2-Resistant      10      9

```

```

OR: 2.655
95% CI: 1.39 5.072

```

We can use this generically for other variables, such as:

```
ormh(fq02, outcome02, cxr02)
```

and get:

```

, , stratvar = Not known bilateral

      outcome
exposure  Success Failure
1-Susceptible    77     13
2-Resistant      7      3

, , stratvar = Bilateral

      outcome
exposure  Success Failure
1-Susceptible    303     46
2-Resistant      37     15

```

```

OR: 2.647
95% CI: 1.432 4.892

```

Do not forget to save the function to a file:

```
save(ormh, file = "ormh.r")
```

Admittedly, what we have done here is still quite amateurish. Fortunately, more professional contributors to R have written a function to calculate the Mantel-Haenszel estimate of the odds ratio. If you just type the function:

```
mantelhaen.test
```

you see the full function. We can use it for our example:

```
mantelhaen.test(table(fq02, outcome02, sex))
```

and we get:

```
Mantel-Haenszel chi-squared test with continuity correction

data: table(fq02, outcome02, sex)
Mantel-Haenszel X-squared = 8.8339, df = 1, p-value = 0.002957
alternative hypothesis: true common odds ratio is not equal to 1
95 percent confidence interval:
 1.430554 4.926883
sample estimates:
common odds ratio
 2.65484
```

To look what the author of that function actually wrote, you can type in analogy of what we did above:

```
fix(mantelhaen.test)
```

We note thereby that the producers of this function that is part of the basic R package has the same approach to the calculation of the confidence interval as was chosen for EpiData Analysis, which gave as a summary:

Binomial outcome by Binomial FQ resistance adjusted for Patient's sex				
	N = 501	N	OR	(95% CI)
Crude		501	2.63	(1.43-4.86)
Adjusted		501	2.65	(1.43-4.93)
Patient's sex: Male		355	1.91	(0.85-4.29)
Patient's sex: Female		146	4.54	(1.65-12.54)

In summary, we have demonstrated that we can create any function in R which has a generalizable applicability. Fortunately, others have already done most of what is probably needed and it is thus not usually necessary to write our own functions. This is good news, of course, as it proves rather tedious and definitely requires sometimes basic, and sometimes more sophisticated knowledge of statistics. It also requires a thorough knowledge of the S programming language. Functions abound for R, but one has to look out for them to know how they are used properly. How did we find that this one exists? At the R prompt, type:

```
??mantel
```

and you get in the right lower quadrant of RStudio:

Search Results 

The search string was "mantel" 

Help pages:

- [base::all.equal](#) Test if Two Objects are (Nearly) Equal
- [stats::Box.test](#) Box-Pierce and Ljung-Box Tests
- [stats::mantelhaen.test](#) Cochran-Mantel-Haenszel Chi-Squared Test for Count Data
- [survival::rats](#) Rat treatment data from Mantel et al

where you then click on the link for detailed information.

Tasks:

- o Append the `ormh.r` script to calculate and show also the stratum-specific and crude (unstratified) odds ratioa with 95% confidence intervals (you may use the same type of confidence interval as we used in the `tab2by2.r` script)*

Solution to Exercise 2: Introduction to R software: data bases and functions

Key points:

- The package “foreign” allows importing a variety of data base formats including EpiData REC files. It will not import any labels
- “Everything in R is either an object or a function”: in this exercise you learned creating a generally applicable function for a simple 2-by-2 table, and for a stratified table to calculate a Mantel-Haenszel estimate of an adjusted odds ratio with a confidence interval
- Numerous functions are available already in the base package, you will have to learn to use the Help file to get the fullest out of these functions

Tasks:

- Append the *ormh.r* script to calculate and show also the stratum-specific and crude (unstratified) odds ratio with 95% confidence intervals (you may use the same type of confidence interval as we used in the *tab2by2.r* script)

Solution:

The script *e_ex02.r*:

```
# Exercise 2: Introduction to R software: data bases and functions

#####
# 1) Import a Stata file

library(foreign)
e_ex02.dat <- read.dta("e_ex02.dta")
# e_ex02.dat <- read.dta("c:/epidata_course/e_ex02.dta")
write.table(e_ex02.dat, file="e_ex02.dat", row.names=TRUE)

# See the first 5 records
e_ex02.dat[1:5,]

# Get the list of variable names only
names(e_ex02.dat)

table(e_ex02.dat$fq04)

#####
# 2) Create full unaltered set (515 records)
e_ex02.dat[e_ex02.dat$fq04=="Missing", "fq04"] <- NA
is.na(e_ex02.dat$fq04)
e_ex02_01.dat <- data.frame(e_ex02.dat)

#####
# 3) Create a subset with records with known FQ DST result (401 records)
e_ex02_02.dat <- e_ex02.dat[which(e_ex02.dat$fq04 != "Missing"), ]
```

```

# or alternatively with the SUBSET function:
e_ex02_02.dat <- subset(e_ex02.dat, fq04 != "Missing")

#####
# 4) Create a subset with records with known FQ resistance and
# bacteriological success or failure (53 records)
attach(e_ex02_02.dat)
detach(e_ex02_02.dat)
# Commands on three lines
# e_ex02_03a.dat <- subset(e_ex02_02.dat, outcome07 != "Death")
# e_ex02_03b.dat <- subset(e_ex02_03a.dat, outcome07 != "Default")
# e_ex02_03.dat <- subset(e_ex02_03b.dat, fq04 != "Susceptible")
# Alternatively, commands on single line
e_ex02_03.dat <- subset(e_ex02_02.dat, outcome07 != "Death" & outcome07 != "Default"
  & fq04 != "Susceptible")

# Table of outcome by FQ resistance
# Create first a new variable FQ02

e_ex02_02.dat[e_ex02_02.dat$fq04=="Susceptible", "fq02"] <- "1-Susceptible"
e_ex02_02.dat[e_ex02_02.dat$fq04=="Low-level resistance", "fq02"] <- "2-Resistant"
e_ex02_02.dat[e_ex02_02.dat$fq04=="High-level resistance", "fq02"] <- "2-Resistant"
e_ex02_fq02.dat <- data.frame(e_ex02_02.dat)
table(e_ex02_fq02.dat$fq02, e_ex02_fq02.dat$outcome02)

write.table(e_ex02_01.dat, file="e_ex02_01.dat", row.names=TRUE)
write.table(e_ex02_02.dat, file="e_ex02_02.dat", row.names=TRUE)
write.table(e_ex02_03.dat, file="e_ex02_03.dat", row.names=TRUE)

#####
# 5) Make functions: 2-by-2 table
# Prepare making a function
# tab2by2 <- function(exposure, outcome) {}
# fix(tab2by2)
# save(tab2by2, file = "tab2by2.r")

attach(e_ex02_fq02.dat)

load("C:/epidata_course/tab2by2.r")
tab2by2(fq02, outcome02)
tab2by2(sex, outcome02)

#####
# 6) Make functions: 2-by-2-by-2 table / Mantel-Heanszel
# Arrays for a stratification

table(fq02, outcome02, sex)

# ormh <- function(exposure, outcome, startvar) {}
# fix(ormh)
# save(ormh, file = "ormh.r")
ormh(fq02, outcome02, sex)
ormh(fq02, outcome02, cxr02)

#####
# 7) On restart use LOAD to get the functions
load("C:/epidata_course/ormh.r")
tab2by2(fq02, outcome02)
load("C:/epidata_course/tab2by2.r")
table(fq02, outcome02, sex)
ormh(fq02, outcome02, sex)

mantelhaen.test(table(fq02, outcome02, sex))

```

The script ormh.r:

```
function(exposure, outcome, stratvar)
{
  tab <- table(exposure, outcome, stratvar)
  a1 <- tab[1,1,1]; b1 <- tab[2,1,1]; c1 <- tab[1,2,1]; d1 <- tab[2,2,1]
  a2 <- tab[1,1,2]; b2 <- tab[2,1,2]; c2 <- tab[1,2,2]; d2 <- tab[2,2,2]
  a <- a1+a2; b <- b1+b2; c <-c1+c2; d <- d1+d2
  or1 <- (a1/c1)/(b1/d1)
  se1 <- sqrt(1/a1+1/b1+1/c1+1/d1)
  or1.ci.lower <- exp(log(or1)-1.96*se1)
  or1.ci.upper <- exp(log(or1)+1.96*se1)
  or2 <- (a2/c2)/(b2/d2)
  se2 <- sqrt(1/a2+1/b2+1/c2+1/d2)
  or2.ci.lower <- exp(log(or2)-1.96*se2)
  or2.ci.upper <- exp(log(or2)+1.96*se2)
  or <- (a/c)/(b/d)
  se <- sqrt(1/a+1/b+1/c+1/d)
  or.ci.lower <- exp(log(or)-1.96*se)
  or.ci.upper <- exp(log(or)+1.96*se)
  n1 <- a1+b1+c1+d1
  n2 <- a2+b2+c2+d2
  adn1 <- (a1*d1)/n1
  adn2 <- (a2*d2)/n2
  bcn1 <- (b1*c1)/n1
  bcn2 <- (b2*c2)/n2
  mhor <- (adn1+adn2)/(bcn1+bcn2)
  w1 <- (b1*c1)/n1
  w2 <- (b2*c2)/n2
  v1 <- ((a1+c1)/(a1*c1))+((b1+d1)/(b1*d1))
  v2 <- ((a2+c2)/(a2*c2))+((b2+d2)/(b2*d2))
  var.mhor <- ((w1^2*v1)+(w2^2*v2))/((w1+w2)^2)
  se <-sqrt(var.mhor)
  mhor.lower <-exp(log(mhor)-1.96*se)
  mhor.upper <-exp(log(mhor)+1.96*se)
  print(tab)
  cat("\nOR stratum 1:", round(or1, digits=3), "\n95% CI:", round(or1.ci.lower,
digits=3), "-", round(or1.ci.upper, digits=3),
      "\n\nOR stratum 2:", round(or2, digits=3), "\n95% CI:", round(or2.ci.lower,
digits=3), "-", round(or2.ci.upper, digits=3),
      "\n\nOR crude:", round(or, digits=3), "\n95% CI:", round(or.ci.lower,
digits=3), "-", round(or.ci.upper, digits=3),
      "\n\nOR adjusted:", round(mhor, digits=3), "\n95% CI:", round(mhor.lower,
digits=3), "-", round(mhor.upper, digits=3))
}
```

The output:

, , stratvar = Male

exposure	outcome	
	Success	Failure
1-Susceptible	274	38
2-Resistant	34	9

, , stratvar = Female

exposure	outcome	
	Success	Failure
1-Susceptible	106	21
2-Resistant	10	9

OR stratum 1: 1.909
95% CI: 0.85 - 4.287

OR stratum 2: 4.543
95% CI: 1.646 - 12.535

OR crude: 2.635
95% CI: 1.427 - 4.865

OR adjusted: 2.655
95% CI: 1.39 - 5.072

Exercise 3: Multivariable analysis in R part 1: Logistic regression

At the end of this exercise you should be able to:

- a. Know how to use logistic regression in R
- b. Know how to properly remove factors for which most likely adjustment is not required

In Exercise 1, you learned some basic principles about the R language. As you go along, it will prove worthwhile to familiarize yourself more and more with the workings of the R / S language, and we provided you with links to some particularly useful texts that help in becoming more proficient.

In Exercise 2, you learned how to import a dataset, to assign the special values for missing data that R recognizes as such, to create data subsets for analysis. Next you learned to write your own basic function.

For none of the things you learned in Exercises 1 and 2 will we have a need for R. EpiData delivers all that, and in a very simple and intuitive way. We will take recourse to R only if we cannot solve a problem analytically with EpiData Analysis. One such application is the logistic regression analysis which is the subject of this exercise.

Before we get started with the actual work, open a new script page and save it as “e_ex03.r”. We will use the dataset `e_ex02_02.dat` as our starting point, that is, the set with 501 cases with known fluoroquinolone drug susceptibility test result. The simplest way to make a dataset available in R is with the command `read.table`:

```
e_ex03 <- read.table("e_ex02_02.dat")
attach(e_ex03)
```

We assign its content to an object `e_ex03` which we attach, so that it is available in the path.

The indication for a logistic regression

A major challenge in the analysis of epidemiologic data is to avoid falsely inferring causality between some factor and an outcome. As a simplified example we might find in a given population that cases of lung cancer occur far more frequently among males than among females. One would not conclude that being male is a risk factor for lung cancer before first examining whether smoking might also be a virtually male addiction in that population. For categorical variables we might use the Mantel-Haenszel stratification approach to adjust for such confounding. The method has two major limitations. First, the more variables we need to adjust for, the larger the uncertainty of our estimates and the more likely some strata are left without counts and making a summary measure becomes impossible. Second, the method is limited to categorical variables but often we do not wish to categorize a naturally continuous variable (like age) just to accommodate the method. From various possibilities, one favored method is logistic regression analysis that overcomes these two major limitations of stratified

analysis. If carefully done, factors that independently predict a given outcome can be isolated and thus get the investigator closer to inference of causality.

Logistic regression using R

Logistic regression is part of `glm` which is used to fit **generalized linear models**. GLM is part of the R base package. The basic formulation of the model is simple:

```
output <- glm(formula = outcome ~ factor(var01) + factor (var02) + var03,
  data=datasetname, family=binomial)
```

where `output` is the object to which the model results are assigned to, and `glm` is the actual function. In parenthesis, one opens with the formula statement and the name of the outcome variable following the `=` sign, followed by a tilde `~` and then all the variables in the model. Specification with `factor` is required for categorical variables, followed by the variable name of the variable in parenthesis as with any function (`factor` being the function here). In the example `var01` and `var02` are categorical variables, while `var03` is treated as a continuous variable. All the variables entering the equation are connected by `+` signs. A comma designates the end of the variable list and is followed by `data=` and the name of the dataset. Finally, `family=binomial` defines that the logit member of the `glm` family is to be used.

Why don't we just enter here for starters an example from our dataset? Please add the following line (all written onto a single line) at the end of our `e_ex03.r`:

```
mylogit <- glm(formula = outcome02 ~ factor(fq02) + factor(sex) + age, data=
  e_ex03, family=binomial)
```

Into the next line type:

```
summary(mylogit)
```

and you get:

```
Call:
glm(formula = outcome02 ~ factor(fq02) + factor(sex) + age, family = binomial,
  data = e_ex03)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-2.3770  0.3750  0.5025  0.6043  1.3678

Coefficients:
            Estimate Std. Error z value Pr(>|z|)
(Intercept)      2.54862    0.37757   6.750 1.48e-11 ***
factor(fq02)2-Resistant -1.10402    0.32517  -3.395 0.000686 ***
factor(sex)Male      0.90295    0.29063   3.107 0.001891 **
age                -0.03619    0.01032  -3.506 0.000455 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 429.92  on 500  degrees of freedom
Residual deviance: 404.93  on 497  degrees of freedom
AIC: 412.93

Number of Fisher scoring iterations: 5
```

Let's examine the output a bit, notably the part that begins with the **Coefficients**. We have `factor(fq02)2-Resistant` and `factor(sex)Male`. The referent for a variable is the lowest value. The values for `fq02` are 1-Susceptible and 2-Resistant. We

therefore see `factor(fq02)2-Resistant` in the line. `age` is a continuous variable, and as such it does not have a referent and is shown as it is.

For the numeric output we have the `Estimate` and `Std. Error`, and in the last column a probability $Pr(>|z|)$ for the `Estimate`. The latter shows that all three variables are highly significant predictors, but otherwise it is difficult to gauge what these numbers mean. Only by making this more explicit it becomes more meaningful. The estimate is a logarithm, thus we do some exponentiation and column binding (`cbind`):

```
lreg.or <- exp(cbind(OR = coef(mylogit), confint(mylogit)))
round(lreg.or, digits=4)
```

and then get more meaningfully:

	OR	2.5 %	97.5 %
(Intercept)	12.7894	6.2062	27.3682
factor(fq02)2-Resistant	0.3315	0.1768	0.6367
factor(sex)Male	2.4669	1.3943	4.3737
age	0.9645	0.9450	0.9841

The two functions `coef` and `confint` take the estimate and calculate the confidence interval from the standard error respectively. `cbind` “glues” (Dalgaard) the vectors together. Exponentiating the logarithmic terms gives the odds ratios (labeled here as OR) with the 95% confidence interval that is more amenable to interpretation. Thus, we need both the summary and the conversion of the estimates. The former is required to determine how to proceed in the stepwise exclusion of variables from the model. Intuitively, we correctly use `age` as a continuous variable, but it is not certain whether this is actually also analytically correct. We should at least check first whether the influence of `age` has continuity as logistic regression will force a fixed slope.

How should we categorize `age`? We could follow the standard classification of WHO, but perhaps a bit less arbitrary is to have the data themselves dictating the categories. In our dataset we have the variables `agequart` and `aged`. These are indeed data-driven quartiles of `age` and the variable `age` split into a binary variable defined by the median respectively. These variables were made based on the entire set of 515 records. As we are using a dataset that has 14 records removed, the values are now conceptually if not actually incorrect. We will thus first remove the two obsolete variables and then create a new variable for quartiles of `age` based on the actual 501 records. We used:

```
names(e_ex03)
```

before. This gives us the names of the variables and their position in the dataset:

```
[1] "age"      "fq04"     "sex"      "totobstime" "agequart"  "aged"     "outcome07"
[8] "outcome02" "pza02"   "kmy02"    "pth02"     "cxr02"    "fq02"
```

We can see / count that `agequart` and `aged` take positions 5 and 6 respectively in the sequence of variables. In other words, we need only the variables in positions 1 through 4 and 7 through 13, which we write as:

```
e_ex03b <- e_ex03[c(1:4, 7:13)]
detach(e_ex03)
attach(e_ex03b)
names(e_ex03b)
```

where the last line `names(e_ex03b)` is only to check that we got what we intended to get and we did:

```
[1] "age"      "fq04"      "sex"      "totobstime" "outcome07" "outcome02" "pza02"
[8] "kmy02"    "pth02"     "cxr02"    "fq02"
```

Thus here the principle how to drop variables in R. Next we want to create a new variable from the existing variable age. As the values for the new variable are driven by the data, we need to know first more about the distribution of age. R has a powerful way of showing quantiles (look up the Help file by typing `?quantiles`). To obtain quartiles, we type:

```
quantile(age, probs = c(25, 50, 75)/100)
```

and we get:

```
25% 50% 75%
 23  31  42
```

To create a new variable `agecat` from the existing variable `age`, we ensure (no problem if it is duplicated from above) that the correct file is in the path and then make the assignments followed by again detaching the file:

```
attach(e_ex03b)
e_ex03b$agecat[age >= 00 & age < 23] <- "Q1"
e_ex03b$agecat[age >= 23 & age < 31] <- "Q2"
e_ex03b$agecat[age >= 31 & age < 42] <- "Q3"
e_ex03b$agecat[age >= 42]           <- "Q4"
detach(e_ex03b)
```

We also note that our outcome variable “`outcome02`” is alphabetically listed as “`Failure`” first, then “`Success`”. Intuitively, we might be more interested in risk factors for “`Failure`” rather than in risk factors for “`Success`”. We thus make a rearrangement to get the order sequentially such that the odds calculated is `Failure / Success`, taking into account that R wishes the outcome to be 0 and 1 (zero and one) and write in full:

```
attach(e_ex03b)
e_ex03b$agecat[age >= 00 & age < 23] <- "Q1"
e_ex03b$agecat[age >= 23 & age < 31] <- "Q2"
e_ex03b$agecat[age >= 31 & age < 42] <- "Q3"
e_ex03b$agecat[age >= 42]           <- "Q4"
e_ex03b$out02[outcome02 == "Success"] <- 0
e_ex03b$out02[outcome02 == "Failure"] <- 1
detach(e_ex03b)
```

Then we give both new variables new names, write the data to disk, attach the new file, check the names to verify (not necessary, but it is always a good idea in the beginning to verify), and then write the output:

```
e_ex03c <- data.frame(e_ex03b)
write.table(e_ex03c, file="e_ex03c", row.names=TRUE)
attach(e_ex03c)
names(e_ex03c)
table(agecat, out02)
```

and get:

```
      out02
agecat 0  1
  Q1 103 10
  Q2 107 23
  Q3 109 20
  Q4 105 24
```

This ensures that for the age quartiles Q1 is the referent and for treatment outcome 1–Success is the referent.

Then we repeat the regression model with the categorized agecat:

```
# Logistic regression with AGE as categorical variable
mylogit2 <- glm(formula = out02 ~ factor(fq02) + factor(sex) + factor(agecat),
  data= e_ex03c, family=binomial)
summary(mylogit2)
lreg.or <-exp(cbind(OR = coef(mylogit2), confint(mylogit2)))
round(lreg.or, digits=4)
```

and get:

	OR	2.5 %	97.5 %
(Intercept)	0.0841	0.0322	0.1807
factor(fq02)2-Resistant	0.7392	0.1711	2.2200
factor(sex)Male	0.4388	0.2357	0.8244
factor(agecat)Q2	3.1949	1.2311	9.3846
factor(agecat)Q3	3.4681	1.3332	10.2368
factor(agecat)Q4	5.6083	2.1556	16.7452

The referent is the youngest quartile set to unity. Relative to the referent, the risk of failure increases in the third and fourth quartile to similar levels and is then highest in the last quartile. This observation seems to support the treating of age as a continuous variable, or at least not plainly contradicting it.

This, so far has been “sampling” only. What we really wanted to evaluate is the influence of several factors that might modify the treatment outcome of multidrug-resistant tuberculosis. We will be starting with a full model inputting all variables, using instead of the binary fq02 the more detailed fq04, slightly modified. The variable fq04 had originally 4 levels, but in the source dataset of 501 records we are using in this exercise, the 14 records with a missing result have been dropped. We will first re-categorize the remaining 3 levels so as to ensure that the referent is Susceptible. As shown earlier, we do this by adding a sequential number to the desired sequence to get it alphabetically correct:

```
attach(e_ex03c)
e_ex03c$fq03[fq04 == "Susceptible"] <- "1-Susceptible"
e_ex03c$fq03[fq04 == "Low-level resistance"] <- "2-Low-level resistance"
e_ex03c$fq03[fq04 == "High-level resistance"] <- "3-High-level resistance"
detach(e_ex03c)
```

Not that it is essential but for getting things tidy, save the new as a data frame and read it in:

```
e_ex03d <- data.frame(e_ex03c)
write.table(e_ex03d, file="e_ex03d", row.names=TRUE)
e_ex03e <- read.table("e_ex03d")
```

and then we are ready for the full model:

```
mylogit3 <- glm(formula = out02 ~ factor(fq03) + factor(sex) + age +
  factor(pza02) + factor(kmy02) + factor(pth02) + factor(cxr02),
  data=e_ex03e, family=binomial)
summary(mylogit3)
```

and we get:

```

                Estimate Std. Error z value Pr(>|z|)
(Intercept)      -2.54930    0.39982  -6.376 1.82e-10 ***
factor(fq03)2-Low-level resistance -0.16501    0.63817  -0.259 0.795972
factor(fq03)3-High-level resistance  2.21715    0.46176   4.802 1.57e-06 ***
factor(sex)Male  -0.85679    0.30327  -2.825 0.004726 **
age               0.03765    0.01054   3.571 0.000356 ***
factor(pza02)PZA resistant  -0.37179    0.37015  -1.004 0.315169
factor(kmy02)KM resistant    0.18024    1.49296   0.121 0.903908
factor(pth02)PTH resistant  -0.27951    0.37987  -0.736 0.461845
factor(cxr02)Not known bilateral  0.06088    0.32561   0.187 0.851678

```

Note also the “AIC” at the end to which will pay particular attention in the following:

AIC: 407.95

AIC stands for **Akaike’s Information Criterion** and is a weighted criterion of goodness of fit. The smaller the value, the better the fit. As we are going with stepwise elimination as one of the common procedures (Crawley M J. The R book. Second edition, Wiley, Chichester, UK, 2013, 1051 pp), we are checking how the criterion changes. Gauging from the probabilities above (last column), kanamycin resistance is the first factor to be removed, thus the model is reduced to:

```

mylogit3 <- glm(formula = out02 ~ factor(fq03) + factor(sex) + age +
  factor(pza02) + factor(pth02) + factor(cxr02), data=e_ex03e,
  family=binomial)
summary(mylogit3)

```

Improving the AIC to 405.96 and putting removal of the chest radiography result as the next. This removal improves the AIC to 403.99 and suggest removal of prothionamide resistance next. This removal improves the AIC to 402.54 and suggests removal of pyrazinamide resistance next. This removal improves the AIC to 401.55. The model is now:

```

mylogit3 <- glm(formula = out02 ~ factor(fq03) + factor(sex) + age,
  data=e_ex03e, family=binomial)
summary(mylogit3)

```

and gives:

```

                Estimate Std. Error z value Pr(>|z|)
(Intercept)      -2.65259    0.38786  -6.839 7.97e-12 ***
factor(fq03)2-Low-level resistance -0.28525    0.62974  -0.453 0.650577
factor(fq03)3-High-level resistance  2.07719    0.41969   4.949 7.45e-07 ***
factor(sex)Male  -0.80689    0.29781  -2.709 0.006741 **
age               0.03733    0.01047   3.564 0.000366 ***

```

While low-level resistance to the fluoroquinolone is not a predictor, high-level resistance is a strong predictor, and both *sex* and *age* are predictors. Following the rules of being strict, none of the remaining factors should thus be removed.

If you watched it through the process of elimination, you found:

Modeling step	Resulting AIC value
Full model	407.95
Remove <i>kmy02</i>	405.96
Remove <i>cxr02</i>	403.99
Remove <i>pth02</i>	402.54
Remove <i>pza02</i>	401.55

What we haven't done yet, but are very much obliged to do is to test whether there is heterogeneity in the data and we therefore need one or more interaction terms. *Woolf's test for interaction* (also known as *Woolf's test for the homogeneity of odds ratios*) provides a formal test for Interaction. According to Mark Myatt (see text recommended at the beginning of Part E), R does not provide Woolf's test specifically, but the grammar of the function can be found in the help section on `mantelhaen.test` which we have used earlier. We use here the slightly modified script by Myatt (giving the same result), but is slightly more explicit. We type (perhaps best in the console, so that it does not interfere with our script `e_ex03.r`):

```
woolf.test <- function(x) {}
fix(woolf.test)
```

Then in the editor box:

```
function(x)
{
  x <- x + 0.5
  k <- dim(x)[3]
  or <- apply(x, 3, function(x)
  {(x[1, 1] / x[1, 2]) / (x[2, 1] / x[2, 2])})
  w <- apply(x, 3, function(x) {1 / sum(1 / x)})
  chi.sq <- sum(w * (log(or) - weighted.mean(log(or), w))^2)
  p <- pchisq(chi.sq, df = k - 1, lower.tail = FALSE)
  cat("\nWoolf's X2 :", chi.sq, "\np-value :", p, "\n")
}
```

Finally we save:

```
save(woolf.test, file = "woolf.test.r")
```

Back to our `e_ex03.r` script (not necessary, but no harm either for this time, as it is loaded, but for the next run after we close, and also that we note where the credit goes to):

```
# Test for interaction, using the script for
# Woolf's test provided by Mark Myatt
load("c:/epidata_course/woolf.test.r")
```

The function requires one parameter, denoted here as `x`. If we are interested in the interaction between outcome, ofloxacin resistance, and sex, we could make one object:

```
tabof1 <- table(fq03, out02, sex)
```

and then:

```
woolf.test(tabof1)
```

and we get:

```
woolf's X2 : 0.2271536
p-value : 0.6336425
```

Thus, not to worry about heterogeneity. For the sake of exercise, and also to look what happens, we will nevertheless put in an interaction term:

```
mylogit3 <- glm(formula = out02 ~ factor(fq03) + factor(sex) + age +
  factor(sex):factor(fq03), data=e_ex03e, family=binomial)
summary(mylogit3)
```

and we get a poorer AIC of 405.45 and insignificant interaction terms:

	Estimate	Std. Error	z value	Pr(> z)
(Intercept)	-2.666178	0.391316	-6.813	9.53e-12 ***
factor(fq03)2-Low-level resistance	0.008907	1.132799	0.008	0.993727
factor(fq03)3-High-level resistance	2.146511	0.626505	3.426	0.000612 ***
factor(sex)Male	-0.772537	0.326321	-2.367	0.017913 *
age	0.037092	0.010518	3.526	0.000421 ***
factor(fq03)2-Low-level resistance:factor(sex)Male	-0.413234	1.365405	-0.303	0.762160
factor(fq03)3-High-level resistance:factor(sex)Male	-0.123552	0.837512	-0.148	0.882719

If we find a significant interaction and perhaps even the AIC improves, then the interaction term must be retained. As you note above, it is simple to write the interaction term which is writing the two variables connected by a colon as in `factor(var1) : factor(var2)`.

We are therefore pretty much assured that this is our final model:

```
# Final model:
mylogit3 <- glm(formula = out02 ~ factor(fq03) + factor(sex) + age,
  data=e_ex03e, family=binomial)
summary(mylogit3)
lreg.or <- exp(cbind(OR = coef(mylogit3), confint(mylogit3)))
round(lreg.or, digits=4)
```

With the following odds ratios and 95% confidence intervals:

	OR	2.5 %	97.5 %
(Intercept)	0.0705	0.0322	0.1478
factor(fq03)2-Low-level resistance	0.7518	0.1744	2.2454
factor(fq03)3-High-level resistance	7.9820	3.5104	18.4214
factor(sex)Male	0.4462	0.2485	0.8021
age	1.0380	1.0170	1.0598

Task:

o Examine whether there are interactions between age and sex and fluoroquinolone resistant and age, so that all possibilities have been checked before we accept the model.

Solution to Exercise 3: Multivariable analysis in R part 1: Logistic regression

Key points:

- Logistic regression is the standard and most commonly used approach to multivariable analysis if the questions cannot be answered by adjusting in a stratified analysis using Mantel-Haenszel estimation of the odds ratio
- Use stepwise exclusion of variables to improve the model fit, by observing p values and the AIC summary for quality of model fit
- Beware of heterogeneity, test formally for heterogeneity using Woolf's test

Task:

o Examine whether there are interactions between age and sex and fluoroquinolone resistant and age, so that all possibilities have been checked before we accept the model.

Solution:

The e_ex03.r script:

```
# Exercise 3: Multivariable analysis in R part 1: Logistic regression

rm(list=ls())

e_ex03 <- read.table("e_ex02_02.dat")
attach(e_ex03)

# Logistic regression with AGE as continuous variable
mylogit <- glm(formula = outcome02 ~ factor(fq02) + factor(sex) + age, data=e_ex03,
               family=binomial)
summary(mylogit)
lreg.or <- exp(cbind(OR = coef(mylogit), confint(mylogit)))
round(lreg.or, digits=4)

# Drop variables
names(e_ex03)
e_ex03b <- e_ex03[c(1:4, 7:13)]
detach(e_ex03)
attach(e_ex03b)
names(e_ex03b)

# Create new variable for Quartiles of age
# Age as categorical variables in quartiles
quantile(age, probs = c(25, 50, 75)/100)
## 25% 50% 75%
## 23 31 42

attach(e_ex03b)
e_ex03b$agecat[age >= 00 & age < 23] <- "Q1"
e_ex03b$agecat[age >= 23 & age < 31] <- "Q2"
e_ex03b$agecat[age >= 31 & age < 42] <- "Q3"
e_ex03b$agecat[age >= 42] <- "Q4"
e_ex03b$out02[outcome02 == "Success"] <- 0
e_ex03b$out02[outcome02 == "Failure"] <- 1
detach(e_ex03b)
```

```

e_ex03c <- data.frame(e_ex03b)
write.table(e_ex03c, file="e_ex03c", row.names=TRUE)
attach(e_ex03c)
names(e_ex03c)
table(agecat, out02)

# Logistic regression with AGE as categorical variable
mylogit2 <- glm(formula = out02 ~ factor(fq02) + factor(sex) + factor(agecat), data=
  e_ex03c, family=binomial)
summary(mylogit2)
lreg.or <- exp(cbind(OR = coef(mylogit2), confint(mylogit2)))
round(lreg.or, digits=4)

# Full model logistic regression
attach(e_ex03c)
e_ex03c$fq03[fq04 == "Susceptible"] <- "1-Susceptible"
e_ex03c$fq03[fq04 == "Low-level resistance"] <- "2-Low-level resistance"
e_ex03c$fq03[fq04 == "High-level resistance"] <- "3-High-level resistance"

detach(e_ex03c)
e_ex03d <- data.frame(e_ex03c)
write.table(e_ex03d, file="e_ex03d", row.names=TRUE)
e_ex03e <- read.table("e_ex03d")

# Full model
mylogit3 <- glm(formula = out02 ~ factor(fq03) + factor(sex) + age + factor(pza02) +
  factor(kmy02) + factor(pth02) + factor(cxr02), data=e_ex03e, family=binomial)
summary(mylogit3)
# AIC 407.95

# Remove kmy02
mylogit3 <- glm(formula = out02 ~ factor(fq03) + factor(sex) + age + factor(pza02) +
  factor(pth02) + factor(cxr02), data=e_ex03e, family=binomial)
summary(mylogit3)
# AIC 405.96

# Remove cxr02
mylogit3 <- glm(formula = out02 ~ factor(fq03) + factor(sex) + age + factor(pza02) +
  factor(pth02), data=e_ex03e, family=binomial)
summary(mylogit3)
# AIC 403.99

# Remove pth02
mylogit3 <- glm(formula = out02 ~ factor(fq03) + factor(sex) + age + factor(pza02),
  data=e_ex03e, family=binomial)
summary(mylogit3)
# AIC 402.54

# Remove pza02
mylogit3 <- glm(formula = out02 ~ factor(fq03) + factor(sex) + age, data=e_ex03e,
  family=binomial)
summary(mylogit3)
# AIC 401.55

# Test for interaction, using the script for
# Woolf's test provided by Mark Myatt
load("c:/epidata_course/woolf.test.r")
attach(e_ex03e)
tabof1 <- table(fq03, out02, sex)
woolf.test(tabof1)
# p-value : 0.6336425
tabof1 <- table(fq03, out02, agecat)
woolf.test(tabof1)
# p-value : 0.9898
tabof1 <- table(sex, out02, agecat)

```

```

woolf.test(tabofl)
# p-value : 0.5075

# Put in the interaction term anyhow
mylogit3 <- glm(formula = out02 ~ factor(fq03) + factor(sex) + age +
  factor(sex):factor(fq03), data=e_ex03e, family=binomial)
summary(mylogit3)
# AIC 405.45
# => AIC worsens

# Final model:
mylogit3 <- glm(formula = out02 ~ factor(fq03) + factor(sex) + age, data=e_ex03e,
  family=binomial)
summary(mylogit3)

```

The Woolf's test for heterogeneity:

```

tabofl <- table(fq03, out02, sex)
woolf.test(tabofl)
tabofl <- table(fq03, out02, agecat)
woolf.test(tabofl)
tabofl <- table(sex, out02, agecat)
woolf.test(tabofl)

```

```

tabofl <- table(fq03, out02, sex)
Woolf's X2 : 0.2271536
p-value : 0.6336425

```

```

tabofl <- table(fq03, out02, agecat)
Woolf's X2 : 0.115918
p-value : 0.9898611

```

```

tabofl <- table(sex, out02, agecat)
Woolf's X2 : 2.326063
p-value : 0.5075462

```

Conclusion:

No interaction between fluoroquinolone resistance and sex, fluoroquinolone resistance and age, nor sex and age. Therefore the final model does not need an interaction term.

Exercise 4: Multivariable analysis in R part 2: Cox proportional hazard model

At the end of this exercise you should be able to:

- a. Use the Cox proportional hazard model
- b. Test the assumption for proportionality and if violated, carry out a stratified analysis

Often the results of the logistic regression are the culminating final summary of your analysis. In our specific setting, the logistic regression was an intermediate step to an adjusted form of a survival analysis. You have learned the principle of using a Kaplan-Meier survival analysis in an earlier Exercise. We extend on this approach and look at factors that determine the binary outcome in the treatment of multidrug-resistant tuberculosis. A Kaplan-Meier analysis would get us quite far, but we would face the problem of adjustment and the problem of dealing with continuous predictor variables such as age. While categorizing age is an option, it might be arbitrary and not the most satisfactory solution. We could end up with multiple small groups if values from more than one variable are made strata of a combining new variable. Almost inevitably, differences might become meaningless with the loss of power.

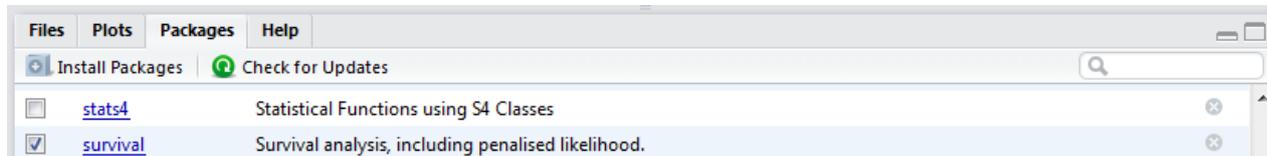
There are several techniques of multivariate analysis. Principally all do the same thing – simultaneous adjustment for multiple variables and providing independent effect estimates for each exposure variable in the model on the outcome. Depending on the type of outcome variable, different techniques are used.

If the outcome is a continuous variable, we use linear regression. If the outcome is categorical, we use logistic regression. If the outcome is ‘time to event’, we use a Cox proportional hazard model. If the outcome is ‘number of events’ (discrete numeric), then we use Poisson regression. Here, the outcome measure is ‘time to event’, and hence we use Kaplan-Meier analysis for univariate analysis and the Cox proportional hazard model for multivariate analysis.

In our approach to the analysis of the dataset on multidrug-resistant tuberculosis we combine the two techniques of logistic regression modeling and the Cox proportional hazard model in a way that is quite common: logistic regression is used first to evaluate and determine which variables have to be considered. There are alternative approaches, including determining the factors within the Cox model itself. In our case, we had isolated three factors, initial fluoroquinolone resistance, age, and sex. After fitting the Cox proportional hazard model including these three variables, we test whether any of the variables grossly violates the assumption of proportionality of hazards (which must be met). If there is such a variable, it must be removed from the adjustment and instead stratification by this variable is indicated. Then each of the strata are adjusted for the remaining variables and it is checked again whether anything remains that violates the proportionality assumption, and so on, until the final model emerges. This is the procedure we are going to apply.

The R survival package

The base package of R does not include survival analysis, and the package “survival” must thus be installed (see lower right quadrant in RStudio):



The “survival” package was written by Terry Therneau from the Mayo Clinic. The procedure is the same as we used before for the “foreign” package. Open a new file in the Source editor and save it as e_ex04.r file. Analogous to calling “foreign” from the library, we also need to be calling “survival” from the library. We are going to use the e_ex02_02.dat as our starting dataset. Thus, make sure that it is attached before we start doing anything.

* Exercise 4: Multivariable analysis in R part 2: Cox proportional hazard model

```
library(survival)
rm(list=ls())
e_ex04 <- read.table("e_ex02_02.dat")
names(e_ex04a)
```

The last line gives us the variables:

```
[1] "age"          "fq04"         "sex"          "totobstime"  "agequart"    "aged"        "outcome07"
[8] "outcome02"   "pza02"       "kmy02"       "pth02"       "cxr02"       "fq02"
```

We need only AGE, FQ04, SEX, TOTOBSTIME, and OUTCOME02, thus:

```
e_ex04b <- e_ex04a[c(1:4, 8)]
```

(It is not just to repeat what was learnt before, there is also some reason to reduce datasets: like EpiData Analysis, R works in the memory which makes it a relatively “slow processor”). We then make the same modification we discussed and did in Exercise 3, attach the file and check the names:

```
attach(e_ex04b)
e_ex04b$fq03[fq04 == "Susceptible"] <- "1-Susceptible"
e_ex04b$fq03[fq04 == "Low-level resistance"] <- "2-Low-level resistance"
e_ex04b$fq03[fq04 == "High-level resistance"] <- "3-High-level resistance"
e_ex04b$out02[outcome02 == "Success"] <- 0
e_ex04b$out02[outcome02 == "Failure"] <- 1
detach(e_ex04b)
names(e_ex04b)
e_ex04c <- e_ex04b[c(1, 3:4, 6:7)]
attach(e_ex04c)
names(e_ex04c)
```

The last line shows:

```
[1] "age"          "sex"          "totobstime"  "fq03"        "out02"
```

We have thus reduced our dataset to the bare essential minimum with which we can work.

Kaplan-Meier survival analysis

Let's first compare notes, i.e. check whether we get identical survival probabilities using the Kaplan-Meier method in EpiData Analysis and in R. In EpiData Analysis, we would use:

```
lifetable out02 totobstime /by=fq03 /i=b30 /adj \  
  /ymin=0.30 /ymax=1 /NG /e3 \  
  /ti="KM successful outcome probability" \  
  /sub="by intitial fluoroquinolone susceptibility" \  
  /t
```

We get:

```
Survival probability for susceptible:          0.865  
Survival probability for low-level resistance: 0.909  
Survival probability for high-level resistance: 0.480
```

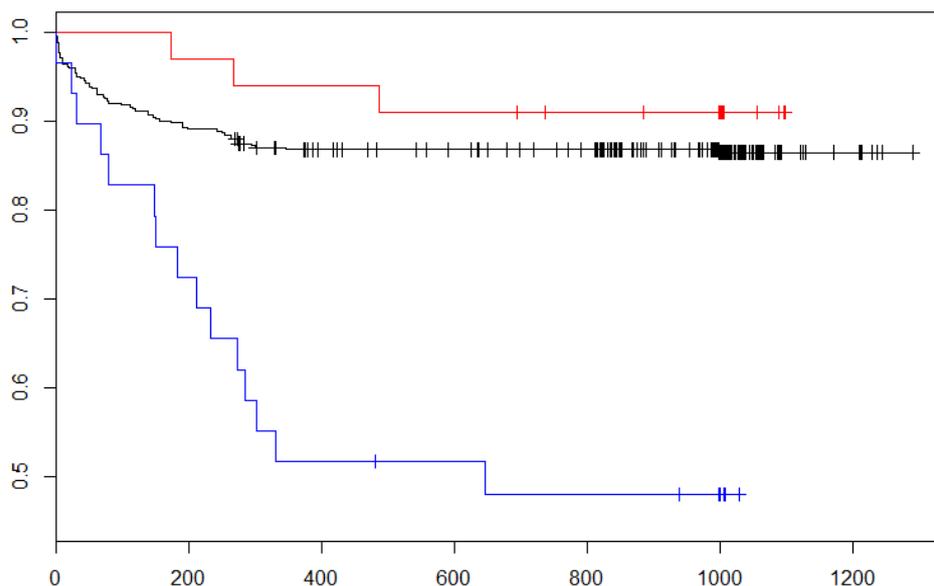
The model commands in R:

```
fit1 <- survfit(formula=Surv(totobstime, out02) ~ fq03, data=e_ex04c)
```

We want to add a simple plot:

```
plot(fit1, col = c(1, 2, 4), ymin=0.45)  
# colors: 1=black; 2=red, 4=blue
```

which gives us the plot in the lower-right quadrant:



Remember that initially we defined R as a “language and environment for statistical computing and graphing”. The graphics capabilities of R are enormous but it will take time to learn and acquire them (see more information in the text by Thomas Lumley). As it is not our primary purpose to evaluate the graphics capabilities of R, we leave that for the time being. We must see the statistical output, however, so we add a third command line:

```
fit1 <- survfit(formula=Surv(totobstime, out02) ~ fq03, data=e_ex04c)  
plot(fit1, col = c(1, 2, 4), ymin=0.45)  
# colors: 1=black; 2=red, 4=blue  
summary(fit1, times = seq(0, 2000, 30))
```

We get exactly the same survival probabilities as with the `lifetable` in EpiData Analysis. This should strengthen our confidence in both software and our skills. We note, however, that the confidence intervals are different. In EpiData Analysis, events change the survival probability but censored observations do not. This is identical in R and it is what we expect. In EpiData analysis, the 95% confidence interval, however, continues to widen as observations with the passage of time become censored, while this is not the case in R. EpiData Analysis uses the philosophy that smaller numbers lead to larger uncertainty, while R focuses on the importance of uncertainty at the point of the last event. Comparison shows that Stata and R get the same results. EpiData Analysis has adapted the approach proposed by Altman, and this will be reviewed during the re-writing of the EpiData Analysis module. In any case, we can reproduce the survival probability in the Kaplan-Meier approach.

The Cox proportional hazard model

The proportional hazards model allows the analysis of survival data by regression modeling. Linearity is assumed on the log scale of the hazard. Relative to a referent, say the rate of death among a control group, the rate of death among the experimental group might be half that of the control group and the hazard ratio is thus 0.5. In contrast to relative risks which are cumulative over observation time, hazard ratios reflect an instantaneous risk over the study period or a subset of the period. Hazard ratios suffer therefore somewhat less from possible selection bias introduced by endpoints. Under the Cox proportional hazard model, the hazard ratio is constant. The Cox model thus assumes an underlying hazard function with a corresponding survival curve. In a stratified analysis, there will be one such curve for each stratum.

The command lines for the Cox model are:

```
mdrcox <- coxph(Surv(totobstime, out02) ~ factor(fq03) + factor(sex) + age,
  data=e_ex04c)
summary(mdrcox)
```

where `totobstime` is the variable for the total observation time from treatment start until the event occurs or the observation time is censored (be it e.g. to loss from follow-up after treatment cessation or reaching the end of the observation time). Note that “Surv” is capitalized (R is case-sensitive). The second line gives the output:

```
Call:
coxph(formula = Surv(totobstime, out02) ~ factor(fq03) + factor(sex) +
  age, data = e_ex04c)

n= 501, number of events= 77

              coef exp(coef)  se(coef)      z Pr(>|z|)
factor(fq03)2-Low-level resistance -0.329337  0.719400  0.593274 -0.555 0.578814
factor(fq03)3-High-level resistance  1.506031  4.508801  0.293728  5.127 2.94e-07 ***
factor(sex)Male                    -0.643091  0.525665  0.256959 -2.503 0.012325 *
age                                 0.031887  1.032400  0.008783  3.630 0.000283 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

	exp(coef)	exp(-coef)	lower .95	upper .95
factor(fq03)2-Low-level resistance	0.7194	1.3900	0.2249	2.3013
factor(fq03)3-High-level resistance	4.5088	0.2218	2.5354	8.0183
factor(sex)Male	0.5257	1.9024	0.3177	0.8698
age	1.0324	0.9686	1.0148	1.0503

```

Concordance= 0.687 (se = 0.033 )
Rsquare= 0.068 (max possible= 0.848 )
Likelihood ratio test= 35.06 on 4 df, p=4.521e-07
wald test = 43.95 on 4 df, p=6.565e-09
Score (logrank) test = 49.31 on 4 df, p=5.026e-10

```

As it should be, the three variables are significantly associated. The $\exp(\text{coef})$, i.e. the exponent of the coefficient is the hazard ratio.

As mentioned above, the Cox proportional hazard model requires that the assumption of proportionality is met, that is the survival function for different factors are required to change proportionately and do not, for instance cross each other.

The test diagnostic to evaluate whether the assumption of proportionality is met is:

```
cox.zph(mdrcox)
```

gives:

	rho	chisq	p
factor(fq03)2-Low-level resistance	0.248	4.72	0.029826
factor(fq03)3-High-level resistance	0.257	5.16	0.023055
factor(sex)Male	-0.105	0.82	0.365224
age	-0.241	4.18	0.040909
GLOBAL	NA	18.51	0.000982

The global chi-square test is highly significant, that is the assumption of proportionality is violated. The most important culprit is seemingly the variable fq03. AGE is also significant but not that grossly and sex not at all. The way to resolve it is stratification, that is fq03 must be taken out and the model must be stratified by fq03. R makes it easy for us to do that with the following modification:

```

mdrcox <- coxph(Surv(totobstime, out02) ~ strata(fq03) + factor(sex) + age,
  data=e_ex04c)
summary(mdrcox)

```

This gives:

	coef	exp(coef)	se(coef)	z	Pr(> z)
factor(sex)Male	-0.643315	0.525547	0.256857	-2.505	0.012260 *
age	0.031399	1.031898	0.008758	3.585	0.000337 ***

Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

	exp(coef)	exp(-coef)	lower .95	upper .95
factor(sex)Male	0.5255	1.9028	0.3177	0.8695
age	1.0319	0.9691	1.0143	1.0498

```

Concordance= 0.647 (se = 0.039 )
Rsquare= 0.028 (max possible= 0.806 )
Likelihood ratio test= 14.26 on 2 df, p=0.0008024
wald test = 14.38 on 2 df, p=0.0007543
Score (logrank) test = 14.73 on 2 df, p=0.0006345

```

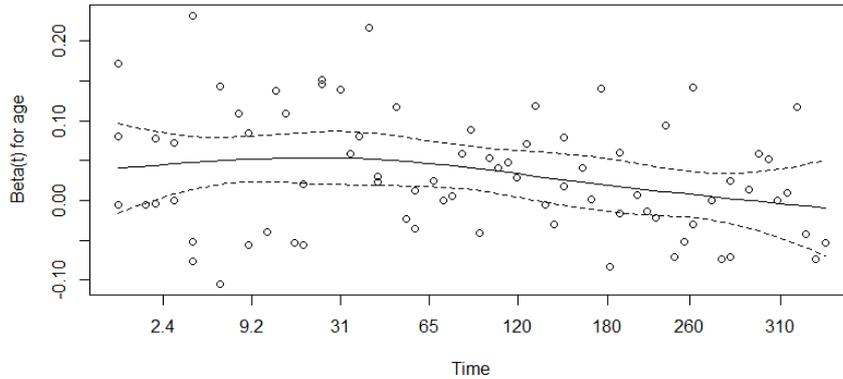
We have now the hazard of sex and age but not anymore of fluoroquinolone resistance because the latter was not a constant hazard and was thus taken out by stratification.

Testing the assumption for proportionality and plotting the test result:

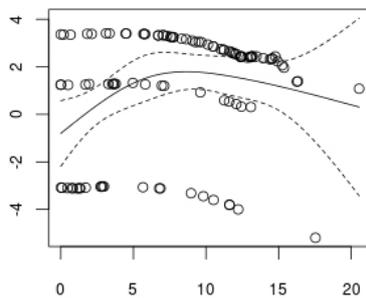
```
cox.zph(mdrcox)
plot(cox.zph(mdrcox))
```

	rho	chisq	p
factor(sex)Male	-0.112	0.926	0.3358
age	-0.245	4.288	0.0384
GLOBAL	NA	7.864	0.0196

and



This indicates that the model is not ideal for the variable age, but with a bit lenience and the more or less regular graphical test we will allow it to pass without further more complex stratification. The interpretation of the graph in its most simplest way is how curved it is: if it is fairly flat (as we think it is here), the assumption of proportionality is not (much) violated. If it is decidedly different from flat, then the assumption is violated as in this example:



Copied from:

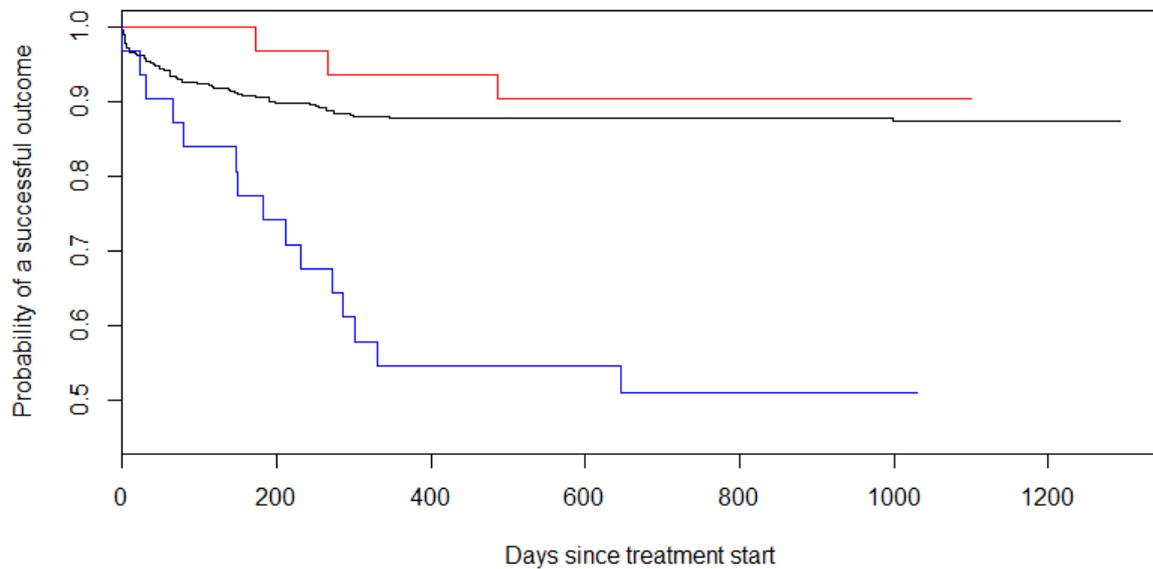
<http://stats.stackexchange.com/questions/15114/how-to-understand-the-plotting-of-the-cox-zph-function-in-r>

```
help(plot.survfit)
```

gives us information on how to plot it. It can get elaborate, of course, but here just a bare-bone sequence of commands:

```
p <- plot(survfit(mdrcox), ylim=c(.45, 1), xlab="Days since treatment start",
mark.time=F, ylab="Probability of a successful outcome", col=c(1, 2, 4),
main="Cox proportional hazard model by initial fluoroquinolone
resistance")
```

Cox proportional hazard model by initial fluoroquinolone resistance



Our main interest is in the numeric output quantification of survival probability:

```
print(p)
$х
[1] 1292 1099 1030
$у
[1] 0.8744596 0.9045559 0.5096336
```

Thus, our main result, the survival probabilities among patients with initial fluoroquinolone susceptibility, low-, and high-level resistance, adjusted for age and sex are:

```
Survival probability susceptible:          0.874
Survival probability low-level resistant:  0.905
Survival probability high-level resistant: 0.510
```

To obtain the detailed data:

```
mdrcox2 <- survfit(mdrcox)
summary(mdrcox2, times = seq(0, 3000, 30))
```

gives (just the first 150 days for susceptible, low-level resistant, and high-level resistant):

fq03=1-susceptible							
time	n.risk	n.event	survival	std.err	lower	95% CI upper	95% CI
0	439	2	0.996	0.00292		0.990	1.000
30	419	19	0.956	0.00954		0.938	0.975
60	411	7	0.942	0.01103		0.920	0.963
90	404	7	0.927	0.01235		0.903	0.951
120	400	4	0.918	0.01304		0.893	0.944
150	397	4	0.910	0.01369		0.883	0.937

fq03=2-Low-level resistance								
time	n.risk	n.event	survival	std.err	lower	95% CI	upper	95% CI
0	33	0	1.000	0.0000		1.000		1
30	33	0	1.000	0.0000		1.000		1
60	33	0	1.000	0.0000		1.000		1
90	33	0	1.000	0.0000		1.000		1
120	33	0	1.000	0.0000		1.000		1
150	33	0	1.000	0.0000		1.000		1

fq03=3-High-level resistance								
time	n.risk	n.event	survival	std.err	lower	95% CI	upper	95% CI
0	29	1	0.969	0.0309		0.910		1.000
30	27	2	0.905	0.0526		0.807		1.000
60	26	0	0.905	0.0526		0.807		1.000
90	24	2	0.839	0.0667		0.718		0.981
120	24	0	0.839	0.0667		0.718		0.981
150	23	2	0.775	0.0764		0.638		0.940

Here, we show it for 30-day intervals, out for up to 3000 days but we could also just get it for the first 5 days, but **daily**:

```
summary(mdrcox2, times = seq(0, 5, 1))
```

fq03=1-Susceptible								
time	n.risk	n.event	survival	std.err	lower	95% CI	upper	95% CI
0	439	2	0.996	0.00292		0.990		1.000
1	437	1	0.994	0.00358		0.987		1.000
2	436	2	0.990	0.00463		0.981		0.999
3	434	2	0.986	0.00548		0.975		0.996
4	432	3	0.979	0.00656		0.967		0.992
5	429	2	0.975	0.00719		0.961		0.989

fq03=2-Low-level resistance								
time	n.risk	n.event	survival	std.err	lower	95% CI	upper	95% CI
0	33	0	1	0		1		1
1	33	0	1	0		1		1
2	33	0	1	0		1		1
3	33	0	1	0		1		1
4	33	0	1	0		1		1
5	33	0	1	0		1		1

fq03=3-High-level resistance								
time	n.risk	n.event	survival	std.err	lower	95% CI	upper	95% CI
0	29	1	0.969	0.0309		0.91		1
1	28	0	0.969	0.0309		0.91		1
2	28	0	0.969	0.0309		0.91		1
3	28	0	0.969	0.0309		0.91		1
4	28	0	0.969	0.0309		0.91		1
5	28	0	0.969	0.0309		0.91		1

R is indeed powerful through flexibility.

Task:

This analysis shows that low-level fluoroquinolone resistance has seemingly no influence on the outcome while in contrast high-level fluoroquinolone resistance is a powerful predictor for an adverse outcome. Nevertheless, even with high-level fluoroquinolone resistance, a remarkable 51% still had a successful outcome. Unsuccessful here also included death and default. What is of key interest with resistance to the core drug fluoroquinolone is whether the outcome is bacteriologically favorable or unfavorable.

- o The first task is to identify factors that are predictors for an unsuccessful bacteriological outcome (failure or relapse) versus a bacteriologically favorable outcome (completion and relapse-free cure) among cases with any type of fluoroquinolone resistance (low-level or high-level) and who did neither die nor default.*
- o Summarize your analysis in a table showing numbers, odds ratios and 95% confidence intervals both univariate and adjusted multivariate for the factors identified in your regression analysis.*

Solution to Exercise 4: Multivariable analysis in R part 2: Cox proportional hazard model

At the end of this exercise you should be able to:

- Use stepwise logistic regression to determine risk factors for a bacteriologically adverse outcome among patients with initial fluoroquinolone resistance who neither defaulted nor died
- Summarize your findings in a table

Solution:

The `e_ex04.r` script from Exercise 4:

```
# Exercise 4: Multivariable analysis in R part 2: Cox proportional hazard
model

library(survival)
rm(list=ls())
e_ex04a <- read.table("e_ex02_02.dat")
names(e_ex04a)

e_ex04b <- e_ex04a[c(1:4, 8)]

attach(e_ex04b)
e_ex04b$fq03[fq04 == "Susceptible"] <- "1-Susceptible"
e_ex04b$fq03[fq04 == "Low-level resistance"] <- "2-Low-level resistance"
e_ex04b$fq03[fq04 == "High-level resistance"] <- "3-High-level resistance"
e_ex04b$out02[outcome02 == "Success"] <- 0
e_ex04b$out02[outcome02 == "Failure"] <- 1
detach(e_ex04b)
names(e_ex04b)
e_ex04c <- e_ex04b[c(1, 3:4, 6:7)]
attach(e_ex04c)
names(e_ex04c)

# Reproducing the Kaplan-Meier lifetable analysis from EpiData Analysis
fit1 <- survfit(formula=Surv(totobstime, out02) ~ fq03, data=e_ex04c)
plot(fit1, col = c(1, 2, 4), ymin=0.45)
# colors: 1=black; 2=red, 4=blue
summary(fit1, times = seq(0, 2000, 30))

# Cox proportional hazard model
mdrcox <- coxph(Surv(totobstime, out02) ~ factor(fq03) + factor(sex) + age,
  data=e_ex04c)
summary(mdrcox)
cox.zph(mdrcox)

mdrcox <- coxph(Surv(totobstime, out02) ~ strata(fq03) + factor(sex) + age,
  data=e_ex04c)
summary(mdrcox)
cox.zph(mdrcox)
plot(cox.zph(mdrcox))
```

```

p <- plot(survfit(mdrcox), ylim=c(.45, 1), xlab="Days since treatment start",
  mark.time=F, ylab="Proportion without adverse event", col=c(1, 2, 4),
  main="Cox proportional hazard model by initial fluoroquinolone
  resistance")
print(p)

mdrcox2 <- survfit(mdrcox)
summary(mdrcox2, times = seq(0, 3000, 30))
summary(mdrcox2, times = seq(0, 5, 1))

```

The `e_ex04_solution.r` script for the task:

```

# Exercise 4 Solution: Logistic regression to identify risk factors for
  failure

library(survival)
rm(list=ls())
e_ex04a_task <- read.table("e_ex02_03.dat")
names(e_ex04a_task)

e_ex04b_task <- e_ex04a_task[c(1:3, 8:11)]

attach(e_ex04b_task)
e_ex04b_task$fq03[fq04 == "Low-level resistance"] <- "1-Low-level
  resistance"
e_ex04b_task$fq03[fq04 == "High-level resistance"] <- "2-High-level
  resistance"
e_ex04b_task$out02[outcome02 == "Success"] <- 0
e_ex04b_task$out02[outcome02 == "Failure"] <- 1
detach(e_ex04b_task)
names(e_ex04b_task)
e_ex04c_task <- e_ex04b_task[c(1, 3, 5:9)]
attach(e_ex04c_task)
names(e_ex04c_task)

# Logistic regression
mylogit<- glm(formula = out02 ~ factor(fq03) + factor(sex) + factor(pza02) +
  factor(kmy02) + factor(pth02) + age, data=e_ex04c_task, family=binomial)
summary(mylogit)

# Remove KMY02
mylogit<- glm(formula = out02 ~ factor(fq03) + factor(sex) + factor(pza02) +
  factor(pth02) + age, data=e_ex04c_task, family=binomial)
summary(mylogit)

# Remove PTH02
mylogit<- glm(formula = out02 ~ factor(fq03) + factor(sex) + factor(pza02) +
  age, data=e_ex04c_task, family=binomial)
summary(mylogit)

# Remove SEX
mylogit<- glm(formula = out02 ~ factor(fq03) + factor(pza02) + age,
  data=e_ex04c_task, family=binomial)
summary(mylogit)

# Remove AGE

```

```

mylogit<- glm(formula = out02 ~ factor(fq03) + factor(pza02),
  data=e_ex04c_task, family=binomial)
summary(mylogit)

# => PZA just not significant with p=0.5468, leave it in
# Final model
mylogit<- glm(formula = out02 ~ factor(fq03) + factor(pza02),
  data=e_ex04c_task, family=binomial)
summary(mylogit)
lreg.or <-exp(cbind(OR = coef(mylogit), confint(mylogit)))
round(lreg.or, digits=4)

load("c:/epidata_course/woolf.test.r")
tab <- table(fq03, out02, pza02)
woolf.test(tab)
# P=0.936 => no interaction

# Univariate analysis
load("c:/epidata_course/tab2by2.r")
tab2by2(fq03, out02)
tab2by2(pza02, out02)

```

	Successful		Not successful		Total	Crude odds ratio			Adjusted odds ratio*		
	n	n	%	Point		CI low	CI high	Point	CI low	CI high	P value
Total	44	9	17.0	53							
Fluoroquinolone resistance											
Low level	30	1	3.2	31	Ref						
High level	14	8	36.4	22	17.1	2.0	150.7	13.0	1.9	262.1	0.025
Pyrazinamide resistance											
Not known *	27	1	3.6	28	Ref						
Resistant	17	8	32.0	25	12.7	1.5	110.8	9.2	1.3	187.0	0.055

CI 95% confidence interval
* By logistic regression
** Susceptible or not tested