

Part B. EpiData Analysis

Part B: EpiData Analysis

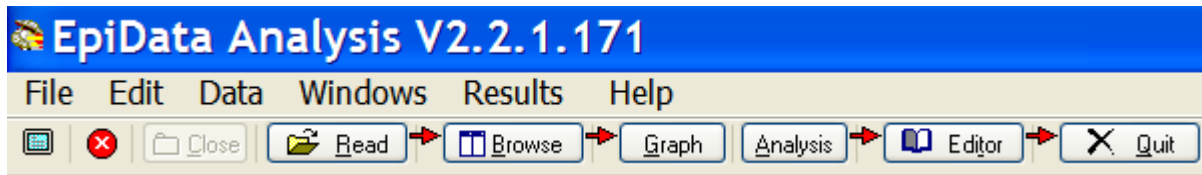
- Exercise 1 A basic program in EpiData Analysis
- Exercise 2 Appending and making new REC files in EpiData Analysis
- Exercise 3 Creating a string variable in EpiData Analysis
- Exercise 4 Aggregating data and saving the summary data in a file

Exercise 1: A basic program in EpiData Analysis

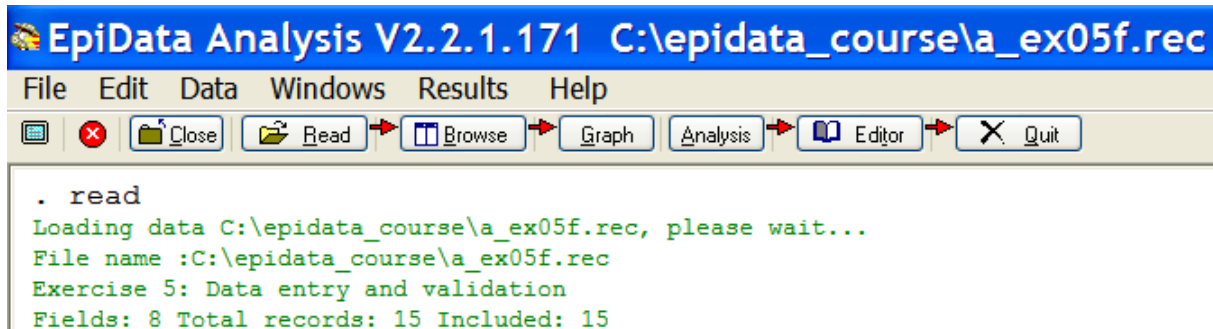
At the end of this exercise you should be able to:

- 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.

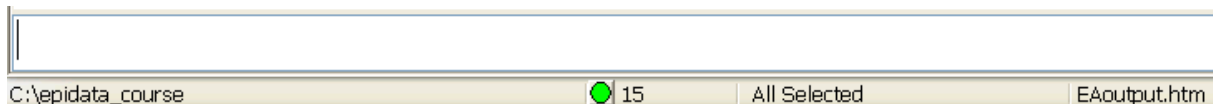
If you open EpiData Analysis, you see the flow chart-like sequence of boxes you are familiar with from EpiData Entry:



First, data must be READ before one can look at them. If you click on Read data (shortcut **ALT+R**), find the path `c:\epidata_course`, open the file `A_EX05F.REC`. On the top of the screen you see the file and its path and in the output window that it has 15 records:



At the bottom of the screen (below the command line), you get some additional information:



The green circle indicates "readiness".

Browse Data shows the eight variables. Marking the variables `SEX` and `REASON` and then clicking the OK sign will give us the list of all records with just these two Variables. You can

also see in the output window `BROWSE sex reason`. Close the window and type on the command line:

```
Browse sex reason res1
```

and we get the browsing window with these three variables:

Note on the top right the box:



What you see when you open is the default is set to show the labels which come from the `CHK` file. To visualize the actual values in the `REC` file, click on this switch:

Browsing the value labels

	sex	reason	res1
1	Femal	Follow-up at 5 months	Negative
2	Male	Diagnosis	Negative
3	Femal	Follow-up at 5 months	Negative
4	Male	Follow-up at 2 months	Negative
5	Femal	Diagnosis	Negative
6	Male	Diagnosis	Negative
7	Male	Diagnosis	Negative
8	Femal	Diagnosis	Negative
9	Male	Follow-up at 6 months	Negative
10	Male	Diagnosis	1+ positive
11	Male	Diagnosis	Negative
12	Femal	Follow-up at 5 months	Negative
13	Male	Follow-up at 2 months	Negative
14	Femal	Follow-up at 5 months	Negative
15	Male	Follow-up at 6 months	Negative

Browsing the values

	sex	reason	res1
1	1	5	0.0
2	2	0	0.0
3	1	5	0.0
4	2	2	0.0
5	1	0	0.0
6	2	0	0.0
7	2	0	0.0
8	1	0	0.0
9	2	6	0.0
10	2	0	1.0
11	2	0	0.0
12	1	5	0.0
13	2	2	0.0
14	1	5	0.0
15	2	6	0.0

`BROWSE` without a variable following will give all variables in the browsing window:

	serno	regdate	sex	age	reason	res1	res2	res3
1	3298	26/10/2003	Femal	35	Follow-up at 5 months	Negative	Negative	No result recorded
2	3299	26/10/2003	Male	20	Diagnosis	Negative	Negative	Negative
3	3300	26/10/2003	Femal	30	Follow-up at 5 months	Negative	Negative	No result recorded
4	3301	26/10/2003	Male	24	Follow-up at 2 months	Negative	Negative	No result recorded
5	3302	26/10/2003	Femal	38	Diagnosis	Negative	Negative	Negative
6	3303	26/10/2003	Male	60	Diagnosis	Negative	Negative	Negative
7	3304	26/10/2003	Male	78	Diagnosis	Negative	Negative	Negative
8	9001	26/10/2003	Femal	28	Diagnosis	Negative	Negative	Negative
9	3305	27/10/2003	Male	50	Follow-up at 6 months	Negative	Negative	No result recorded
10	3306	27/10/2003	Male	50	Diagnosis	1+ positive	1+ positive	1+ positive
11	3307	27/10/2003	Male	68	Diagnosis	Negative	Negative	Negative
12	3308	27/10/2003	Femal	29	Follow-up at 5 months	Negative	Negative	No result recorded
13	3309	27/10/2003	Male	36	Follow-up at 2 months	Negative	Negative	No result recorded
14	3310	27/10/2003	Femal	15	Follow-up at 5 months	Negative	Negative	No result recorded
15	3311	27/10/2003	Male	37	Follow-up at 6 months	Negative	Negative	No result recorded

There are thus several ways to get the same results. Beginners will probably start with the drop down sequence of menus, the more advanced will use it interactively on the command line. If we type BROWSE on the command line but don't know the variables, we simply type:

```
browse F3
```

and F3 will show us the variable list from which we can pick the one we need – don't forget the space between the command and the variable(s).

Using the Editor

We will not be making much use of the drop down menus, but go right away to what is the preferred method in our opinion, and that is to make a permanent record of all we do by writing the commands into programs.

We can access the program Editor in four ways, 1) by using the mouse on the Editor box, 2) by using the ALT+T short-cut key, and 3) by using the mouse by clicking on the symbol with the crossed hammer and wrench, and 4) simply with F5. Either way opens the program editor window.

Preparatory steps in a program

Make it a habit to begin every program with the following:

Write a title. This will help you to remember in the future what this program was all about. You can write anywhere in the program comments. Comments are not executed by EpiData and are recognized as such if the line starts with an asterisk:

```
* This is the program b_ex01.pgm, my first program
```

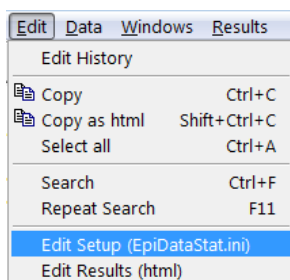
Afterward clear the screen, close any open data file and any open log file:

```
cls  
close  
logclose
```

Tell EpiData, in which folder the work will be done and where the folder is:

```
cd c:\epidata_course
```

Note: you can set the default path in the epidatastat.ini file that should open the next time when you open EpiData Analysis:



Because all of the analysis is going to be done in the c:\epidata_course folder, it might be very useful to make the necessary modification:

```

* 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 viewer font size =12
set window font size =12
set editor font size =12
set viewer font name ="Verdana,Courier"
set style sheet ="c:\Epidata\epiout_monospace.css"

set display variables=ON
set display databrowser=OFF
set output open=ON

cd c:\epidata_course
set echo=ON

```

Making this change will obviate the need to type the path as it will be dealt with as a relative path (rather than absolute), that is assuming the files are located in this folder. You can run it right away by pressing **F9** to ensure that it works properly.

Tell EpiData what file should opened (read) for analysis:

```
read "a_ex05f.rec"
```

We have thus as the beginning:

```

* This is the program b_ex01.pgm, my first program

cls
close
logclose

cd c:\epidata_course
read "a_ex05f.rec"

```

The first line is a comment. Comments are preceded by an asterisk and are bypassed by the analysis program.

CLS stands for “**clear screen**”. You don’t have to write this command, but it helps often as you will later see to speed up the programming process if a lot has been written into the memory.

CLOSE: EpiData Analysis will work only with one REC file open at any time. If you try to open a second file while the first is still open, you will get an error message. With the command CLOSE, any file that might be open will be closed.

LOGCLOSE: EpiData Analysis writes a log file about what it does. In case it did already something else before, we want to start with a new log file, so any log file that might be open should be closed first.

CD C:\EPIDATA_COURSE: This is a DOS command (CD = **Change Directory**) that tells EpiData Analysis in which directory or folder it has to look for files and where our work will be done. This line is not necessary if you set the path in the `epidatastat.ini` file.

READ "A_EX05F.REC": this reads the file. Note that the file name is in quotation marks. File names should be placed in quotation marks (except this is not necessary in the command line (F4)). As we specified the path with the CD command, EpiData Analysis will look for this file in that folder.

These commands above should always be in the beginning of a program. Before you proceed, save the program as B_EX01.PGM (the *.PGM extension will be supplied automatically).

To have a little bit larger dataset (75 records) we will preferably be using:

```
read "a.rec"
```

(rather than the a_ex05f.rec file) for the following examples.

The two most frequently used commands

FREQ

TABLES

FREQ gives frequencies on one or more variables. Try:

```
freq sex
freq sex reason
freq *
```

TABLES gives cross-tabulations by two (or more) variables. Try:

```
tables sex reason
tables reason res1
```

You may note that all the output we get is showing the Value labels, not the values. This makes it much easier to understand than if the Values were shown, particularly if numeric coding had been chosen.

Options in EpiData Analysis

You noted above with frequencies and tables that EpiData Analysis gives you just the numbers without any frills in contrast to many other software that gives you all sorts of things in which you are not interested but not necessarily those in which you are. You can get all sorts of additional things from EpiData Analysis, but you have to ask for it. You do this with options.

Options begin with a forward slash (/) at the end of a line, followed by whatever you need and is available for that particular command. There are options that apply for a multitude of commands, and some that are specific for certain commands.

Most frequently, you may wish to have percentages with frequencies and tables. Try the following:

```
tables sex reason /c
tables sex reason /r
tables sex reason /c /r
tables sex reason /c /d0
```

The default for the third command line above may give you:

Examinee's sex						
Examination reason	Female		% Male		% Total	%
Diagnosis	16 (43.2)	(48.5)	21 (56.8)	(50.0)	37 (100.0)	(49.3)
Follow-up at 2 months	5 (38.5)	(15.2)	8 (61.5)	(19.0)	13 (100.0)	(17.3)
Follow-up at 4 months	0 (0.0)	(0.0)	1 (100.0)	(2.4)	1 (100.0)	(1.3)
Follow-up at 5 months	7 (58.3)	(21.2)	5 (41.7)	(11.9)	12 (100.0)	(16.0)
Follow-up at 6 months	4 (50.0)	(12.1)	4 (50.0)	(9.5)	8 (100.0)	(10.7)
Follow-up at 7 months or later	0 (0.0)	(0.0)	2 (100.0)	(4.8)	2 (100.0)	(2.7)
Follow-up, month not stated	1 (100.0)	(3.0)	0 (0.0)	(0.0)	1 (100.0)	(1.3)
Reason not stated	0 (0.0)	(0.0)	1 (100.0)	(2.4)	1 (100.0)	(1.3)
Total	33 (44.0)	(100.0)	42 (56.0)	(100.0)	75	

Percents: (Row) (Col)

This is a bit inconvenient as the type of parentheses for both column and row percentages are the same and you have first to figure out which one is which.

You can adjust this to your preference using one of the dozens available SET commands, for instance:

```
set table percent format col="P1[ ]"
```

and repeat the command and you would get:

Examinee's sex						
Examination reason	Female		% Male		% Total	%
Diagnosis	16 (43.2)	[48.5]	21 (56.8)	[50.0]	37 (100.0)	[49.3]
Follow-up at 2 months	5 (38.5)	[15.2]	8 (61.5)	[19.0]	13 (100.0)	[17.3]
Follow-up at 4 months	0 (0.0)	[0.0]	1 (100.0)	[2.4]	1 (100.0)	[1.3]
Follow-up at 5 months	7 (58.3)	[21.2]	5 (41.7)	[11.9]	12 (100.0)	[16.0]
Follow-up at 6 months	4 (50.0)	[12.1]	4 (50.0)	[9.5]	8 (100.0)	[10.7]
Follow-up at 7 months or later	0 (0.0)	[0.0]	2 (100.0)	[4.8]	2 (100.0)	[2.7]
Follow-up, month not stated	1 (100.0)	[3.0]	0 (0.0)	[0.0]	1 (100.0)	[1.3]
Reason not stated	0 (0.0)	[0.0]	1 (100.0)	[2.4]	1 (100.0)	[1.3]
Total	33 (44.0)	[100.0]	42 (56.0)	[100.0]	75	

Percents: (Row) [Col]

With the little footnote being quite explicit on what is what.

If you add another option, you can get the percentages into separate columns:

```
tables sex reason /c /r /pct
```

and get an even clearer lay-out:

Examinee's sex								
Examination reason	Female	%	% Male	%	% Total	%	%	%
Diagnosis	16 (43.2)	[48.5]	21 (56.8)	[50.0]	37 (100.0)	[49.3]		
Follow-up at 2 months	5 (38.5)	[15.2]	8 (61.5)	[19.0]	13 (100.0)	[17.3]		
Follow-up at 4 months	0 (0.0)	[0.0]	1 (100.0)	[2.4]	1 (100.0)	[1.3]		
Follow-up at 5 months	7 (58.3)	[21.2]	5 (41.7)	[11.9]	12 (100.0)	[16.0]		
Follow-up at 6 months	4 (50.0)	[12.1]	4 (50.0)	[9.5]	8 (100.0)	[10.7]		
Follow-up at 7 months or later	0 (0.0)	[0.0]	2 (100.0)	[4.8]	2 (100.0)	[2.7]		
Follow-up, month not stated	1 (100.0)	[3.0]	0 (0.0)	[0.0]	1 (100.0)	[1.3]		
Reason not stated	0 (0.0)	[0.0]	1 (100.0)	[2.4]	1 (100.0)	[1.3]		
Total	33 (44.0)	[100.0]	42 (56.0)	[100.0]	75			

Percents: (Row) [Col]

As you see, you can add a multitude of options and combinations thereof depending on what you need.

Note: Options apply only for a command in the given program while SET retains the settings for all commands to which it applies until you change it.

One critical option is to get not only the Value labels displayed, but also the values of a field as you will see in a moment. Try:

```
freq sex
freq sex /v1
freq sex /v
freq sex /vn
freq sex /vn1
```

Particularly useful of the above is:

```
freq sex /v1
```

as you will see in the next paragraph.

Selecting records meeting some criteria

Commonly, one wishes to look only at a subset of the dataset, making a selection to include or exclude observations meeting a certain criterion.

Selections must be based on the value, not the value label. As a regular frequency shows only value labels, the value for a given value is first determined to learn how to make a selection.

Try:

```
cls
freq reason /v1
select reason=0
tables sex res1
select
tables sex res1
```

You note that SELECT without anything following restores the full dataset in the memory.

You may use other operators:

```
<    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)

Continuing with a larger dataset

To get a little beyond the 15 records you had entered to obtain A_EX05F.REC, we will be using henceforth the supplementary file A.REC. You can do this in the same program after you close the currently open data file, then reading in the new one:

```
cls
close
read "a.rec"
```

Define new variables in memory

In EpiData you learned to make temporary variables in the Check file (Exercises 8 and 9). These did not become part of the database. Similarly, in EpiData Analysis, you define new variables from existing ones that are kept in memory and do not change the original database (if you want to keep them you will have to save them in a new database, see later).

You can define numeric, text or date variables as you would do in a Check file, e.g.

```

define newvar1 ###
define newvar2 _____
define newvar3 <dd/mm/yyyy>

```

After definition of a new variable you have to assign it the values it should take based on an existing variable. For instance, you may wish to create standard WHO age groups, defining the new variable as a text variable of length 5:

```

* define WHO age groups
      define agegrp1 _____
      agegrp1="other"
if age>=00 and age<15 then agegrp1="00-14"
if age>=15 and age<25 then agegrp1="15-24"
if age>=25 and age<35 then agegrp1="25-34"
if age>=35 and age<45 then agegrp1="35-44"
if age>=45 and age<55 then agegrp1="45-54"
if age>=55 and age<65 then agegrp1="55-64"
if age>=65 and age<99 then agegrp1="65 + "
if age>=99          then agegrp1="unkn"
tables sex agegrp1
select age<>99
tables sex agegrp1
select

```

and you get:

Examinee's sex			
agegrp1	Female	Male	Total
00-14	0	1	1
15-24	5	14	19
25-34	8	8	16
35-44	6	6	12
45-54	6	6	12
55-64	3	2	5
65 +	4	4	8
unkn	1	1	2
Total	33	42	75

```

. select age<>99
. tables sex agegrp1
Current select - (age<>99)

```

Examinee's sex			
agegrp1	Female	Male	Total
00-14	0	1	1
15-24	5	14	19
25-34	8	8	16
35-44	6	6	12
45-54	6	6	12
55-64	3	2	5
65 +	4	4	8
Total	32	41	73

Alternatively, you may decide to use numeric coding and make a LABEL for the new variable and give LABELVALUES to the values:

```

* define WHO age groups
      define agegrp2 #
      agegrp2=8
if age>=00 and age<15 then agegrp2=1
if age>=15 and age<25 then agegrp2=2
if age>=25 and age<35 then agegrp2=3
if age>=35 and age<45 then agegrp2=4
if age>=45 and age<55 then agegrp2=5

```

```

if age>=55 and age<65 then agegrp2=6
if age>=65 and age<99 then agegrp2=7
if age>=99 then agegrp2=9
label agegrp2 "WHO age groups"
labelvalue agegrp2 /1="0 to 14 years"
labelvalue agegrp2 /2="15 to 24 years"
labelvalue agegrp2 /3="25 to 34 years"
labelvalue agegrp2 /4="35 to 44 years"
labelvalue agegrp2 /5="45 to 54 years"
labelvalue agegrp2 /6="55 to 64 years"
labelvalue agegrp2 /7="65 years and older"
labelvalue agegrp2 /8="Other" //Note this should not exist - control
labelvalue agegrp2 /9="Unknown age"

```

```

cls
tables sex agegrp2
select age<>99
tables sex agegrp2
select

```

and you would get:

Examinee's sex			
WHO age groups	Female	Male	Total
0 to 14 years	0	1	1
15 to 24 years	5	14	19
25 to 34 years	8	8	16
35 to 44 years	6	6	12
45 to 54 years	6	6	12
55 to 64 years	3	2	5
65 years and older	4	4	8
Unknown age	1	1	2
Total	33	42	75

```

. select age<>99
. tables sex agegrp2
Current select - (age<>99)

```

Examinee's sex			
WHO age groups	Female	Male	Total
0 to 14 years	0	1	1
15 to 24 years	5	14	19
25 to 34 years	8	8	16
35 to 44 years	6	6	12
45 to 54 years	6	6	12
55 to 64 years	3	2	5
65 years and older	4	4	8
Total	32	41	73

Note that we let everybody first be assigned the value OTHER. As EpiData Analysis proceeds line by line, everybody first gets assigned the value OTHER. In the next line, all those meeting the condition of inclusions (not outside) are taken out and assigned that new value, and so on, line by line. At the end, only those who did not meet that condition will have the value "OTHER" assigned. This is an excellent tool to control whether everyone in the file has been assigned as needed.

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 between 1900 and 1929, 1930 and 1949, 1950 and 1999, 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: A basic program in 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 between 1900 and 1929, 1930 and 1949, 1950 and 1999, 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:

```
Examinee's sex
Exercise birth years Female Male Total
text coding
1900-1929                3    2    5
1930-1949                4    5    9
1950-1999               25   34   59
Unknown                  1    1    2
Total                   33   42   75
. tables sex birthgrp2
Examinee's sex
Exercise birth years Female Male Total
numeric coding
Born 1900 to 1929        3    2    5
Born 1930 to 1949        4    5    9
Born 1950 to 1999       25   34   59
Unknown birth year      1    1    2
Total                   33   42   75
```

The program B_EX01.PGM might look as follows (exercises and tasks):

```
* This is the program b_ex01.pgm, my first program

*****
* Preparatory steps 1-3

* 1: Start with a clean slate
cls
logclose
close

* 2: Tell EpiData where the work is done
```

```
* This is not necessary if you specified the
* project path / folder in the EpiDataStat.ini file
* cd c:\epidata_course
```

```
* 3: Read the file you wish to analyze
read "a.rec"
```

```
*****
```

```
* Examples with the two most frequently used commands:
```

```
*  FREQ
*  TABLES
```

```
*  FREQ
freq sex
freq sex reason
freq *
```

```
*  TABLES
tables sex reason
tables reason res1
```

```
*****
```

```
* Options in EpiData Analysis:
```

```
* Without options, you get just the numbers
* Options allow obtaining more information
* Options start with a forward slash at the end of a line: /
* After the forward slash you enter the option(s) you need
```

```
* Options to get different percentages:
```

```
tables sex reason /c
tables sex reason /r
tables sex reason /c /r
tables sex reason /c /r
set table percent format total="P1[]"
tables sex reason /c /r /tp
tables sex reason /c /r /pct
tables sex reason /c /d0
```

```
* Options to see values and value labels
```

```
freq sex
freq sex /vl
freq sex /v
freq sex /vn
freq sex /vnl
```

```
*****
```

```
* Selecting records meeting some criteria:
```

```
*  SELECT
*   followed by the field name,
*   followed by an operator,
*   followed by a value of that field
*   gives you the desired selection
*  SELECT
*   without anything restores the full dataset
```

```

cls
freq reason /vl
select reason=0
tables sex res1
select
tables sex res1

```

* For operators see Help (F1)

* Define new variables in memory

* DEFINE

* followed by a new variable name

* followed by the variable definition and variable length

* gives a new variable

* Define a text variable of length 5

* Then give values to the new variable based

* on an existing variable, e.g., WHO age groups:

* define WHO age groups

```

                define agegrp1 _____
                agegrp1="other"
if age>=00 and age<15 then agegrp1="00-14"
if age>=15 and age<25 then agegrp1="15-24"
if age>=25 and age<35 then agegrp1="25-34"
if age>=35 and age<45 then agegrp1="35-44"
if age>=45 and age<55 then agegrp1="45-54"
if age>=55 and age<65 then agegrp1="55-64"
if age>=65 and age<99 then agegrp1="65 + "
if age>=99                then agegrp1="unkn"

```

```

cls
tables sex agegrp1
select age<>99
tables sex agegrp1
select

```

* Define a numeric variable of length 1

* Then give values to the new variable based

* on an existing variable, e.g., WHO age groups

* Then give a field LABEL to the new variable

* and a LABELVALUE (field values):

* define WHO age groups

```

                define agegrp2 #
                agegrp2=8
if age>=00 and age<15 then agegrp2=1
if age>=15 and age<25 then agegrp2=2
if age>=25 and age<35 then agegrp2=3
if age>=35 and age<45 then agegrp2=4
if age>=45 and age<55 then agegrp2=5
if age>=55 and age<65 then agegrp2=6
if age>=65 and age<99 then agegrp2=7
if age>=99                then agegrp2=9
label agegrp2 "WHO age groups"
labelvalue agegrp2 /1="0 to 14 years"
labelvalue agegrp2 /2="15 to 24 years"
labelvalue agegrp2 /3="25 to 34 years"
labelvalue agegrp2 /4="35 to 44 years"
labelvalue agegrp2 /5="45 to 54 years"

```

```

labelvalue agegrp2 /6="55 to 64 years"
labelvalue agegrp2 /7="65 years and older"
labelvalue agegrp2 /8="Other" //Note this should not exist - control
labelvalue agegrp2 /9="Unknown age"

```

```

cls
tables sex agegrp2
select age<>99
tables sex agegrp2
select

```

```

*****
*****
* Solution of task

```

```

* define the year of birth
define birthyr ####
birthyr=year(regdate)-age

```

```

* Using text variables
* define groupings for birth years
define birthgrp1 _____
                                let birthgrp1="other"
if birthyr>1899 and birthyr<1930 then birthgrp1="1900-1929"
if birthyr>1929 and birthyr<1950 then birthgrp1="1930-1949"
if birthyr>1949 and birthyr<2000 then birthgrp1="1950-1999"
if age=99                                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>1899 and birthyr<1930 then birthgrp2=1
if birthyr>1929 and birthyr<1950 then birthgrp2=2
if birthyr>1949 and birthyr<2000 then birthgrp2=3
if age=99                                then birthgrp2=9
label birthgrp2 "Exercise birth years numeric coding"
labelvalue birthgrp2 /1="Born 1900 to 1929"
labelvalue birthgrp2 /2="Born 1930 to 1949"
labelvalue birthgrp2 /3="Born 1950 to 1999"
labelvalue birthgrp2 /8="Unaccounted for"
labelvalue birthgrp2 /9="Unknown birth year"

```

```

cls
tables sex birthyr
tables sex birthgrp1
tables sex birthgrp2

```

Exercise 2: Appending and making new REC files EpiData Analysis

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

- a. Differentiate between merging and appending data files.
- b. Append different data files into a single data file
- c. Drop some fields from the data set
- d. Creating value labels for new numeric fields

Sometimes, it is useful to *merge* several data files or to *append* some data files to an existing one. For the definitions of MERGE and APPEND and how to proceed you may refer to EpiData Entry. The following graphic outline summarizes the difference between the procedures.

It is common that different data files exist and a researcher wishes to combine them into a single set. If we take two data sets, set A and set B, we can imagine different possibilities.

Set A and set B have the same variables:

Set A			
ID	VAR1	VAR2	VAR3
A	33	A	2
B	21	B	3
C	24	X	7
D	44	Y	8
E	56	C	2
Set B			
F	74	T	1
G	67	A	3
H	34	B	2
I	2	J	1

If we want to combine the data set A and data set B, we will have to **append** set B to set A.

Set B may contain the same variables as set A plus some additional variables:

Set A (and Set B)				Set B only			
ID	VAR1	VAR2	VAR3	ID	VAR4	VAR5	VAR6
A	33	A	2				
B	21	B	3				

C	24	X	7				
D	44	Y	8				
E	56	C	2				
F	74	T	1	F	ER	DFG	6
G	67	A	3	G	YT	DFG	7
H	34	B	2	H	CX	ERT	8
I	2	J	1	I	EW	CVB	1

It is of course also possible that set B contains only variables that are not contained in set A:

Set A				Set B			
ID	VAR1	VAR2	VAR3	ID	VAR4	VAR5	VAR6
A	33	A	2				
B	21	B	3				
C	24	X	7				
D	44	Y	8				
E	56	C	2				
				F	ER	DFG	6
				G	YT	DFG	7
				H	CX	ERT	8
				I	EW	CVB	1

or any combination with some variables contained in both sets:

Set A (and Set B)				Set B (and set A)			
ID	VAR1	VAR2	VAR3	ID	VAR4	VAR5	VAR6
A	33	A	2				
B	21	B	3				
C	24	X	7				
D	44	Y	8				
E	56	C	2				
F			1	F	ER	DFG	6
G			3	G	YT	DFG	7
H			2	H	CX	ERT	8
I			1	I	EW	CVB	1

In all these cases, we will have to **merge** the two data sets.

In brief, we might summarize four rules on when to use APPEND and when to use MERGE:

- Append – add records (rows) when data structure is identical;
- Merge – add variables (columns) based on unique identifier;
- Adding records and variables – use merge;
- Merge function can do what append can do and more

Thus, APPENDING is the procedure where several files with exactly the same fields and field definitions are combined. You may have prepared a single QES file and created from that one questionnaire file several identical REC and CHK files for data entry in various locations.

With the previous exercise B_EX01 you obtained four supplementary EpiData REC files:

A.REC
B.REC
C.REC
D.REC

The data for these files were obtained from four different Tuberculosis laboratory registers and they contain the following fields:

Field name	Field label	Field type	Field length	Field values	Value labels	Comment
Id	Unique laboratory identifier	T	6	Any		Any unique identifier as a combination of LABCODE and SERNO
Serno	Laboratory serial number	I	4	1001,,1075 2001,,2075 3001,,3075 4001,,4075		
labcode	Laboratory code	T	1	A B C D	Laboratory A Laboratory B Laboratory C Laboratory D	
regdate	Registration date	D	10	01/01/1980, ...,31/12/200 2, 01/01/1900		Date of registration known Unknown date of registration
sex	Examinee's sex	I	1	1 2 9	Female Male Sex not recorded	
age	Examinee's age in years	I	2	0, ... ,97 98 99		Known age in years 98 years or older Unknown age
reason	Examination reason	I	1	0 8 9 1 2 3 4 5 6 7	Diagnostic examination Follow-up, month not stated No reason recorded 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 m or later	
res1		I	3	0.0 0.1,,0.9 1.0 2.0 3.0 4.0 8.0 9.0	No bacilli found 1 to 9 AFB per 100 fields 1+ positive 2+ positive 3+ positive Scanty, no quantification Positive, no quantification No result recorded	
res2		F	3	0.0 0.1,,0.9 1.0 2.0 3.0 4.0 8.0 9.0	No bacilli found 1 to 9 AFB per 100 fields 1+ positive 2+ positive 3+ positive Scanty, no quantification Positive, no quantification No result recorded	
res3		F	3	0.0 0.1,,0.9 1.0 2.0 3.0 4.0 8.0 9.0	No bacilli found 1 to 9 AFB per 100 fields 1+ positive 2+ positive 3+ positive Scanty, no quantification Positive, no quantification No result recorded	

Appending files and making a new data file

The beginning of the program looks familiar, but has added two new components shown in bold:

- * This B_EX02.PGM appends three files
- * to a first file and creates a new
- * REC file

```

cls
logclose
close

read "a.rec"

```

```
append /file="b.rec"  
append /file="c.rec"  
append /file="d.rec"  
savedata "abcd.rec"
```

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

The command APPEND adds another file with the same structure to the file that was read in first (A.REC) and at the end of appending all three files (giving a total of four files), the command SAVEDATA followed by the name of the new REC file saves that data file.

This program will run fine if we run it only once. However, if we run it a second time, we will get an error message that the ABCD.REC file cannot be saved because it exists already. We must therefore ensure that the ABCD.REC file does not exist when the program starts to run. To this end, we add another command line before we even read the first file:

```
* This B_EX02.PGM appends three files  
* to a first file and creates a new  
* REC file
```

```
cls  
logclose  
close
```

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

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

This will work even if we run the program for the first time: the first time, there is simply no file to be erased and EpiData Analysis will report so and continue to execute the program.

Let's now say that we do not want to keep the identifiers (fields ID and SERNO). We can tell EpiData Analysis to drop these two fields:

```
* This B_EX02.PGM appends three files  
* to a first file and creates a new  
* REC file
```

```
cls  
logclose  
close
```

```
read "a.rec"  
append /file="b.rec"  
append /file="c.rec"  
append /file="d.rec"  
var drop id serno  
savedata "abcd.rec" /replace
```

* Note: instead of "var drop" you may use more simply just "drop"

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

If you create a new variable, this variable can also be made with numeric coding with a numeric field value and a text as value label, as you learned in EpiData Entry. For instance, you wish to have only two values for the first result (RES1), positive or negative, then you could define:

```
define result1 #
result1=0
if res1>0 and res1<9 then result1=1
labelvalue result1 /0="Negative" /1="Positive"
label result1 "Result of first examination"
```

Use this approach in solving the following task for the case definition.

Task:

- o Start with the program B_EX01.PGM, save it as B_EX02.PGM, and edit it as needed to make the new dataset ABCD.REC.*
- o In the same program, create a new dataset B_EX02.REC file that has a new variable CASE which can be either positive or negative and one for WHO age groups. Define as positive any examinee who has at least one bacillus in at least one result and make a tabulation by case (column) and age group (row), stratified by a new variable that groups reason into diagnosis, follow-up (irrespective of month) and unknown reason. Calculate row percentages.*

Solution to Exercise 2: Appending and making new REC files EpiData Analysis

Key Point(s)

- You can append several files with exactly the same fields and field definitions. The several identical REC and CHK files should have been created from one QES or respectively CHK file.

Task:

- Start with the program *B_EX01.PGM*, save it as *B_EX02.PGM*, and edit it as needed to make the new dataset *ABCD.REC*.
- In the same program, create a new dataset *B_EX02.REC* file that has a new variable *CASE* which can be either positive or negative and one for WHO age groups. Define as positive any examinee who has at least one bacillus in at least one result and make a tabulation by case (column) and age group (row), stratified by a new variable that groups reason into diagnosis, follow-up (irrespective of month) and unknown reason. Calculate row percentages.

Solution:

The solution output table:

The first table is the unstratified table:

Bacteriologic case definition				
WHO age groups	Non-case	% Case	% Total	%
0 to 14 years	5 (83.3)	1 (16.7)	6 (100.0)	
15 to 24 years	50 (87.7)	7 (12.3)	57 (100.0)	
25 to 34 years	73 (86.9)	11 (13.1)	84 (100.0)	
35 to 44 years	59 (88.1)	8 (11.9)	67 (100.0)	
45 to 54 years	24 (85.7)	4 (14.3)	28 (100.0)	
55 to 64 years	22 (88.0)	3 (12.0)	25 (100.0)	
65 years and older	26 (86.7)	4 (13.3)	30 (100.0)	
Unknown age	3 (100.0)	0 (0.0)	3 (100.0)	
Total	262 (87.3)	38 (12.7)	300	
Unstratified table				

Then follow the stratified tables:

Reason grouped: Diagnosis
Bacteriologic case definition

WHO age groups	Non-case	%	Case	%	Total	%
0 to 14 years	1	(50.0)	1	(50.0)	2	(100.0)
15 to 24 years	20	(76.9)	6	(23.1)	26	(100.0)
25 to 34 years	29	(76.3)	9	(23.7)	38	(100.0)
35 to 44 years	17	(70.8)	7	(29.2)	24	(100.0)
45 to 54 years	9	(69.2)	4	(30.8)	13	(100.0)
55 to 64 years	12	(100.0)	0	(0.0)	12	(100.0)
65 years and older	14	(82.4)	3	(17.6)	17	(100.0)
Unknown age	2	(100.0)	0	(0.0)	2	(100.0)
Total	104	(77.6)	30	(22.4)	134	

Percents: (Row)

Reason grouped: Follow-up
Bacteriologic case definition

WHO age groups	Non-case	%	Case	%	Total	%
0 to 14 years	4	(100.0)	0	(0.0)	4	(100.0)
15 to 24 years	30	(96.8)	1	(3.2)	31	(100.0)
25 to 34 years	44	(97.8)	1	(2.2)	45	(100.0)
35 to 44 years	42	(97.7)	1	(2.3)	43	(100.0)
45 to 54 years	15	(100.0)	0	(0.0)	15	(100.0)
55 to 64 years	9	(75.0)	3	(25.0)	12	(100.0)
65 years and older	12	(92.3)	1	(7.7)	13	(100.0)
Unknown age	1	(100.0)	0	(0.0)	1	(100.0)
Total	157	(95.7)	7	(4.3)	164	

Percents: (Row)

Reason grouped: Unknown
Bacteriologic case definition

WHO age groups	Non-case	%	Case	%	Total	%
0 to 14 years	0	(0.0)	0	(0.0)	0	(100.0)
15 to 24 years	0	(0.0)	0	(0.0)	0	(100.0)
25 to 34 years	0	(0.0)	1	(100.0)	1	(100.0)
35 to 44 years	0	(0.0)	0	(0.0)	0	(100.0)
45 to 54 years	0	(0.0)	0	(0.0)	0	(100.0)
55 to 64 years	1	(100.0)	0	(0.0)	1	(100.0)
65 years and older	0	(0.0)	0	(0.0)	0	(100.0)
Unknown age	0	(0.0)	0	(0.0)	0	(100.0)
Total	1	(50.0)	1	(50.0)	2	

Percents: (Row)

The program B_EX02.PGM might look as follows:

```
* This is b_ex02 EpiData Analysis program

cls
close

read "a.rec"
append /file="b.rec"
append /file="c.rec"
append /file="d.rec"
drop serno id
savedata "abcd.rec" /replace

close
read "abcd.rec"

* Definition positive: any AFB in any of three cases
* Values: "pos" or "neg" or "unk"

cls
                                define case #
                                let case=0
if (res1>0.0) and (res1<9) then case=1
if (res2>0.0) and (res2<9) then case=1
if (res3>0.0) and (res3<9) then case=1

labelvalue case /0="Non-case" /1="Case"
label case "Bacteriologic case definition"

* define WHO age groups
                                define agegrp #
                                agegrp=8
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 age groups"
labelvalue agegrp /1="0 to 14 years"
labelvalue agegrp /2="15 to 24 years"
labelvalue agegrp /3="25 to 34 years"
labelvalue agegrp /4="35 to 44 years"
labelvalue agegrp /5="45 to 54 years"
labelvalue agegrp /6="55 to 64 years"
labelvalue agegrp /7="65 years and older"
labelvalue agegrp /8="Other" //Note this should not exist - control
labelvalue agegrp /9="Unknown age"

define reason2 #
reason2=2
if reason=0 then reason2=1
if reason=9 then reason2=9
label reason2 "Reason grouped"
labelvalue reason2 /1="Diagnosis"
labelvalue reason2 /2="Follow-up"
labelvalue reason2 /9="Unknown"
```

```
savedata "b_ex02.rec" /replace
```

```
cls
```

```
close
```

```
read "b_ex02.rec"
```

```
cls
```

```
tables case agegrp reason2 /r
```

Exercise 3: Creating a string variable in EpiData Analysis

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

- a. Create string (character) variables from character or numeric variables.
- b. Save the output of the program in 'html' or text format.

Sometimes it can be useful to create string (character) variables from character or numeric variables. If we use the B_EX02.REC created in the previous exercise, we note that the fields RES1, RES2, and RES3 are numeric variables where the values:

- 0 indicates a negative result
- 0.1, ..., 0.9, 1, 2, 3, 4, 8 indicate positive results
- 9 indicates an examination not done

Let us define that it matters what the sequential results are until there is a first one with any positive result, then we can define six essential patterns:

- NNN Three negative results
- NN9 Two negative results with the third examination not done
- N99 One negative result with the second and third examination not done
- NNP Two negative results followed by a positive result
- NPx The first negative result followed by a positive result
- Px The first result already positive

We would now like to create a new variable PATTERN that takes any of the above six values, created from the three results of sputum smear examinations.

To get the above patterns, we need to create a text variable from numeric variables (RES1, RES2, and RES3), because if we add:

```
RES1+RES2+RES3
```

We get a numeric result (the sum of the real numbers in these three fields), which is not what is required.

The grammar is as follows:

```
define textvar1 _
define textvar2 _
define textvar3 _

if numvar1= X then textvar1=Y
if numvar2 ...
...

define newstring ____
newstring=textvar1+textvar2+textvar3
```

where NUMVAR is an existing numeric variable.

The above is not new, you learned this in EpiData Entry when creating a unique identifier from three existing variables.

However, we will add one more thing and that is to write our output into a file. To this end, we modify the program as follows:

```
logopen "b_ex03.txt" /replace  
freq newstring  
logclose
```

We chose here B_EX03.TXT, a text file as output. EpiData Analysis can also create HTML output, such as B_EX03.HTML.

Tasks:

- o Write the program B_EX03.PGM to obtain patterns.*
- o Limit your analysis to patients with a diagnostic sputum smear examination. Create an output file with three tables. The first table shows the result (positive or negative) by all patterns, the second the result only by essential patterns (defined above). The third determines which proportion among cases with any positive result is positive on the first, which is negative on the first but positive on the second, and which proportion is positive for the first time on the third examination only.*
- o Write the output into a B_EX03.TXT output file.*

Solution to Exercise 3: Creating a string variable in EpiData Analysis

Tasks:

- o Write the program B_EX03.PGM to obtain patterns.*
- o Limit your analysis to patients with a diagnostic sputum smear examination. Create an output file with three tables. The first table shows the result (positive or negative) by all patterns, the second the result only by essential patterns (defined above). The third determines which proportion among cases with any positive result is positive on the first, which is negative on the first but positive on the second, and which proportion is positive for the first time on the third examination only.*
- o Write the output into a B_EX03.TXT output file.*

Solution:

The output is:

Bacteriologic case definition				
pattern	Non-case	% Case	% Total	%
NNN	104 {100.0}	0 {0.0}	104 {77.6}	
NP9	0 {0.0}	1 {3.3}	1 {0.7}	
NPP	0 {0.0}	9 {30.0}	9 {6.7}	
PP9	0 {0.0}	5 {16.7}	5 {3.7}	
FPN	0 {0.0}	2 {6.7}	2 {1.5}	
PPP	0 {0.0}	13 {43.3}	13 {9.7}	
Total	104 {100.0}	30 {100.0}	134	

Percents: {Col}

Bacteriologic case definition				
Essential patterns	Non-case	% Case	% Total	%
NNN	0 {0.0}	20 {66.7}	20 {14.9}	
NN9	0 {0.0}	10 {33.3}	10 {7.5}	
Ex	104 {100.0}	0 {0.0}	104 {77.6}	
Total	104 {100.0}	30 {100.0}	134	

Percents: {Col}

Bacteriologic case definition				
Essential patterns	Case	% Total	%	
NNN	20 {66.7}	20 {66.7}		
NN9	10 {33.3}	10 {33.3}		
Total	30 {100.0}	30		

Percents: {Col}

The second table answers the question on the incremental gain from serial smears in this small dataset: the third smear examination did not add anything, three quarters of all cases (as defined here) were positive already on the first, and the final quarter was added with the second examination.

The program B_EX03.PGM might look as follows:

```
* This is b_ex03 EpiData Analysis program
* to determine the incremental yield from serial smears
```

```
cls
close
logclose
```

```
cd c:\epidata_course
```

```

read "b_ex02.rec"

* Definition positive: any AFB in any of three results
* Values: "pos" or "neg" or "unk"

define result1 _
if res1=0          then result1="N"
if res1=9          then result1="9"
if res1>0 and res1<9 then result1="P"

define result2 _
if res2=0          then result2="N"
if res2=9          then result2="9"
if res2>0 and res2<9 then result2="P"

define result3 _
if res3=0          then result3="N"
if res3=9          then result3="9"
if res3>0 and res3<9 then result3="P"

define pattern ____
pattern=result1+result2+result3

savedata "b_ex03.rec" /replace

cls
close
logclose
read "b_ex03.rec"

define esspatt #
                                esspatt=9
if pattern="PPP"                then esspatt=1
if pattern="PPN" or pattern="PP9" then esspatt=1
if pattern="PNN" or pattern="PN9" or pattern="P99" then esspatt=1
if pattern="PNP"                then esspatt=1
if pattern="NPP" or pattern="NP9" then esspatt=2
if pattern="NNP"                then esspatt=3
if pattern="NNN"                then esspatt=4
if pattern="NN9"                then esspatt=5
if pattern="N99"                then esspatt=6
label esspatt "Essential patterns"
labelvalue esspatt /1="NNN"
labelvalue esspatt /2="NN9"
labelvalue esspatt /3="N99"
labelvalue esspatt /4="Px"
labelvalue esspatt /5="NPx"
labelvalue esspatt /6="NNP"

cls
set echo=off
logopen "b_ex03.txt" /replace
select reason=0
tables case pattern /c
tables case esspatt /c
select esspatt<4
tables case esspatt /c
logclose
select
set echo=on

```

Exercise 4: Aggregating data and saving the summary data in a file

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

- a. Aggregate data using two different approaches, and save summary data from the other approach
- b. Do some manipulations and calculations in a spreadsheet

If aggregate data are collected, there is no way to get back to the individual records. Conversely, it is easy to aggregate data from individual records. This was done in various ways in other exercises (making age groups from age, eg). One can easily find a bit more complex task.

For this exercise you must download the supplementary file `B_EX04_WORKLOAD.REC`, which is an extract from a laboratory dataset (personal communication and data courtesy Biggie Mabaera, Zimbabwe, January 24, 2005).

First approach

You will have to create a new variable that counts the number of smears done on each examinee. We could then try to make a table of the registration date and the number of smears for each examinee, stratified by laboratory. This would give a very large table (many days in one year). If you try this, you will get an error message that there are too many categories. You will thus have to work with tables where possible and frequencies where needed, making the appropriate selection for the laboratory. This approach does not require anything new, but a somewhat innovative approach to what was learned earlier. The final manipulations are done in a spreadsheet.

Second approach

You can use the `AGGREGATE` command of EpiData Analysis. This creates the underlying basis for a table which can be written to a new data file.

To explain what `AGGREGATE` does, we include the dataset `banana.rec` which contains just 7 observations with three variables (person's sex, person's country, and the number of bananas each person has). `AGGREGATE` has many things in common with the `FREQ` and `TABLES` commands, but it extends on these in important ways. Like `FREQ` and `TABLES`, `AGGREGATE` does calculations vertically in a dataset. Using the `banana.rec`:

```
freq sex
```

gives:

```
Female    2
Male      5
Total     7
```

and similarly:

```
aggregate sex
```

gives:

```
Female  2
Male    5
```

Note that the result is the same, but the Total is not given with AGGREGATE. If we have here the similarity of AGGREGATE with FREQ, we can look at its similarity with TABLES:

```
TABLES sex country
```

we get:

```
Participant's country  Female  Male  Total
Tanzania                1      1     2
Uganda                  1      4     5
Total                   2      5     7
```

while with:

```
AGGREGATE sex country
```

we get:

```
sex    country  N
Female Tanzania  1
Female Uganda   1
Male   Tanzania  1
Male   Uganda   4
```

The similarity is not particularly surprising because all what analysis is about (using FREQ and TABLES) is to actually aggregate observations in some meaningful way.

Where the power of AGGREGATE gets in and leaves FREQ and TABLES behind is when we want to make calculations on the variables of interest. For instance, we could ask “How many bananas do all the women and how many do all the men have together?” To calculate this we use options. For this example, we would write:

```
aggregate sex /sum=banana
```

and get:

```
sex    N    Nbanana  SUMbanana
Female 2    2          10
Male   5    5          11
```

This informs that the 2 women had 10 bananas and the 5 men had 11. Because sex is very unbalanced in the data set, we might wish to know that mean number of bananas that were possessed by each sex and while at it also get the confidence interval for the mean:

```
aggregate sex /mci=banana
```

and get:

```
sex    N    Nbanana  MEAbanana  MEAbananaLOCI  MEAbananaHICI
Female 2    2          5.00        -7.71           17.71
Male   5    5          2.20         1.16            3.24
```

We could aggregate as above by sex and country and sum up the bananas for each of the four strata:

```
aggregate sex country /sum=banana
```

and get:

sex	country	N	Nbanana	SUMbanana
Female	Tanzania	1	1	4
Female	Uganda	1	1	6
Male	Tanzania	1	1	3
Male	Uganda	4	4	8

The option `/close` closes the original `*.REC` file and opens the file with the aggregated data. To write the aggregated file into a new file you add the option:

```
aggregate var1 var2 /close /save="newfile1"
```

For our task of investigating the workload in the various laboratories, we probably want to sort on `VAR1` and `VAR2` and write the resulting output into another file that will then be the working file:

```
aggregate var1 var2 /sum=var3 /save="newfile1"
sort var1 var2
savedata "newfile2.rec" /replace
```

Finally, we might want the `NEWFILE2.REC` as a text file. This might be accomplished by going through the export function in EpiData Entry or if the aggregated file is not too large through browsing and copying the content to the clipboard which can be imported or pasted into a spreadsheet for final manipulations and calculations.

Tasks:

- o The `B_EX04_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_EX04.PGM` to provide you with the necessary information to determine the number of smears performed on average on each day on which at least one examinee was examined for each of the three laboratories.*
- o Use your spreadsheet program to do the necessary final calculations and save it as `B_EX04.XLS`. Summarize the result in a simple table.*

Solution to Exercise 4: Aggregating data and saving the summary data

Key Point(s):

- If aggregate data are collected, there is no way to get back to the individual records. Conversely, it is easy to aggregate data from individual records.

Task:

- The B_EX04_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_EX04.PGM to provide you with the necessary information to determine the number of smears performed on average on each day on which at least one examinee was examined for each of the three laboratories.*

Solution

There are two ways to solve the problem, both done here in the same program:

- * Program B_EX04.PGM aggregating the data to
- * determine the workload in three laboratories
- * Data courtesy: Biggie Mabaera, University of Zimbabwe

- * The following is the first method

```
cls
close
logclose

cd c:\epidata_course
read "b_ex04_workload.rec"

cls
                                define smears #
                                let smears=9
if result1<9 and result2<9 and result3<9 then smears=3
if result1<9 and result2<9 and result3=9 then smears=2
if result1<9 and result2=9 and result3=9 then smears=1
if result1=9 and result2=9 and result3=9 then smears=0

cls
erase "b_ex04a.txt"
logopen "b_ex04a.txt"
tables smears laboratory
select laboratory=401
freq regdate
select

cls
select laboratory=411
freq regdate
select
```

```

cls
select laboratory=416
freq regdate
select

logclose
close

* The following is the second method
* Alternative to the above program
* using the aggregate function

cls
close
logclose

cd c:\epidata_course

read "b_ex04_workload.rec"

                                define smears ###
                                let smears=9
if result1<9 and result2<9 and result3<9 then smears=3
if result1<9 and result2<9 and result3=9 then smears=2
if result1<9 and result2=9 and result3=9 then smears=1
if result1=9 and result2=9 and result3=9 then smears=0

erase "b_ex04b.rec"

aggregate regdate laboratory /sum=smears /save="b_ex04b.rec" /replace

cls
close
read "b_ex04.rec"
sort laboratory regdate

keep regdate laboratory n sumsmears
savedata "b_ex04.rec" /replace

close
erase "b_ex04b.rec"
logclose

```

The first method uses what you learned in an earlier exercise using simple frequencies to aggregate the data. You may have noted that it is not possible to use the TABLES command as there are too many categories. Note that it is possible to write the data directly into an Excel™ *.XLS file (in addition to the earlier shown possibilities of *.TXT and *.HTML files).

The second approach is to use the AGGREGATE command of EpiData Analysis. Any table a user produces in EpiData Analysis is built on this AGGREGATE command, but the user never gets to see it. It is, however, possible to use it as a function and to write the aggregated data directly into a file. This file (named here B_EX04.REC) contains only the aggregated 618 records produced from the 12,087 records of the original B_EX04_WORKLOAD.REC file.

Task:

- o Use your spreadsheet program to do the necessary final calculations and save it as B_EX04.XLS. Summarize the result in a simple table.*

If you used the first approach, you created an output text file (named here B_EX04A.TXT) which could be imported into a spreadsheet, and with some cutting and pasting getting the table as required.

If you used the second approach, EpiData Analysis produced the summary aggregate file (named here B_EX04.REC). In EpiData Entry this file could then be exported to a text file, exporting only the necessary variables LABORATORY to identify the laboratory, NSMEARS, the number of examinees on each day, and SUMSMEARS, the number of smears done on each day.

Both approaches must provide the same result:

Method 1: Workload by laboratory

Laboratory	Days	Smears	Smears/day
401	242	23,044	95.2
411	135	1,211	9.0
416	241	6,328	26.3

Method 2: Workload by laboratory

Laboratory	Days	Smears	Smears/day
401	242	23,044	95.2
411	135	1,211	9.0
416	241	6,328	26.3