

Part B. EpiData Analysis

Part B: EpiData Analysis

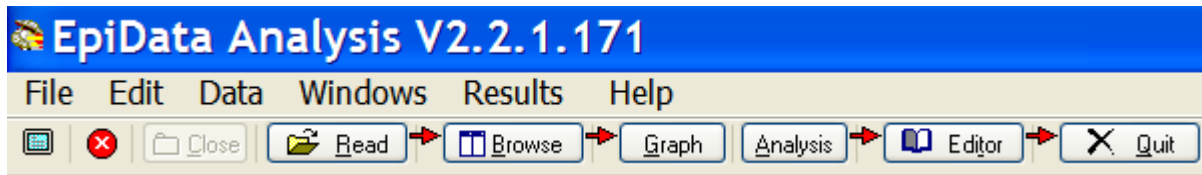
- Exercise 1 A basic program in EpiData Analysis
- Exercise 2 Appending and making new REC files in EpiData Analysis
- Exercise 3 Creating a string variable in EpiData Analysis
- Exercise 4 Aggregating data and saving the summary data in a file

Exercise 1: A basic program in EpiData Analysis

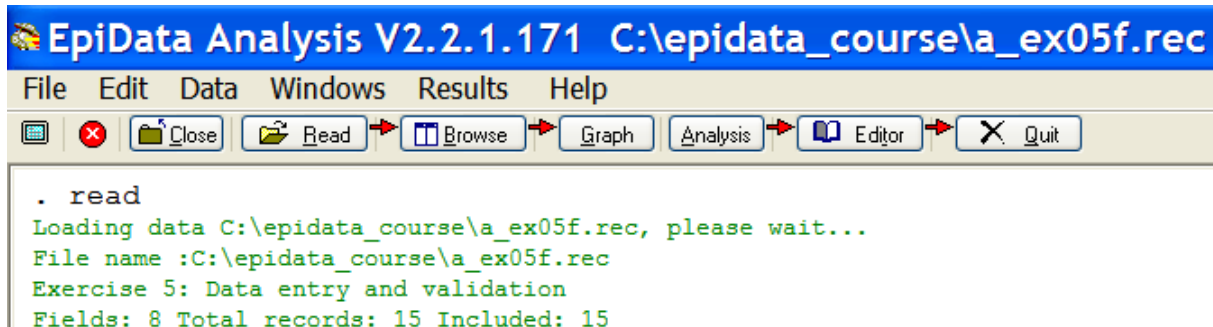
At the end of this exercise you should be able to:

- 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.

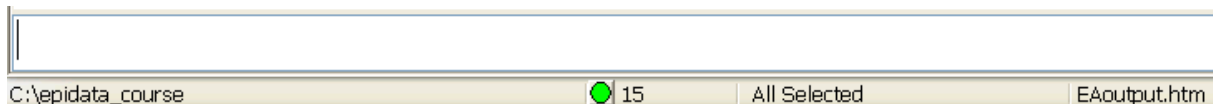
If you open EpiData Analysis, you see the flow chart-like sequence of boxes you are familiar with from EpiData Entry:



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
```

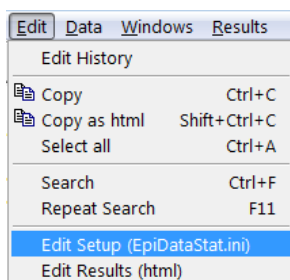
Afterward clear the screen, close any open data file and any open log file:

```
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.*