

Part B. EpiData Analysis

Part B: EpiData Analysis

Exercise 1 An introduction to EpiData Analysis

Exercise 2 More on EpiData Analysis

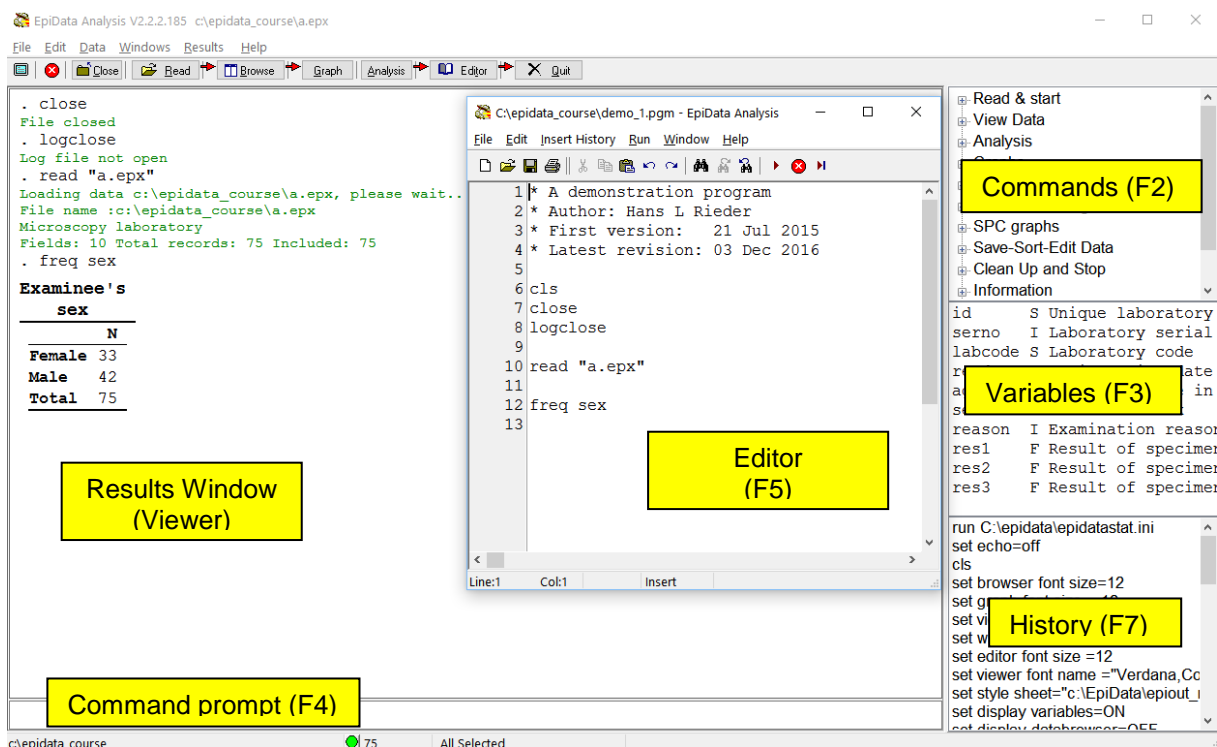
Exercise 3 Aggregating data and saving the summary data in a file

Exercise 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)

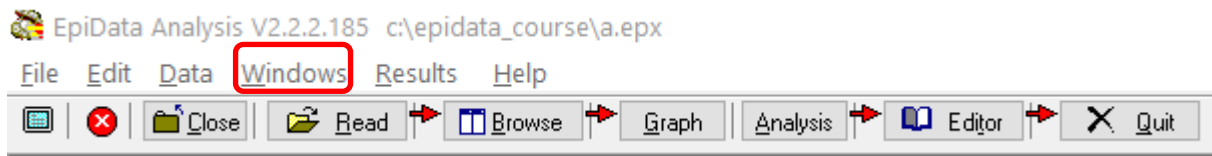
On the left:

- 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:

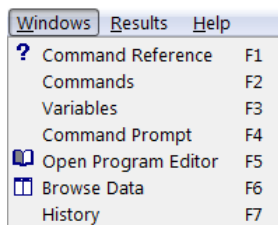


Top row: EpiData version (important to specify when reporting bugs) and path and currently open file name.

Center row: Menu bar

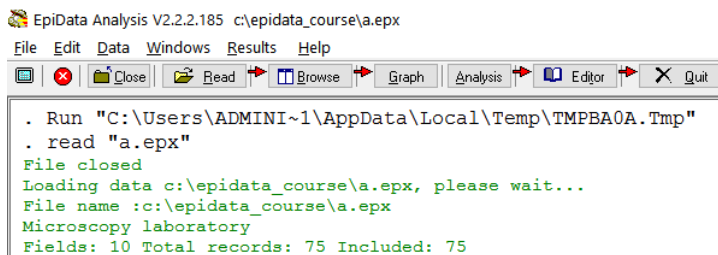
Bottom row: Work tool bar

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



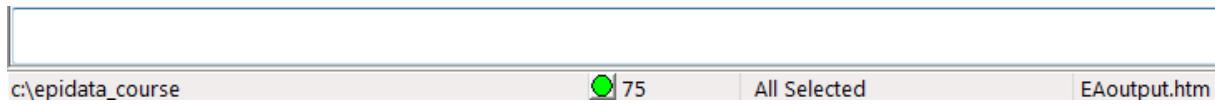
This gives you spelled out the names of the various command and windows and the shortcuts **F1** ,..., **F7** that can be used to access them.

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:



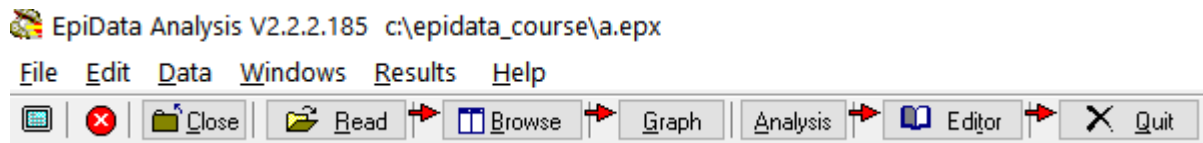
We can double-click on the path (c:\epidata_course here) if we wish to change it and get a pop-up box to Browse for the desired data Folder (see more on that below).

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

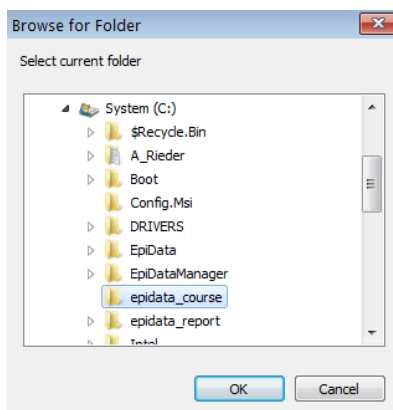
When you open EpiData Analysis, you see information on Version and currently open dataset in the top row, the Menu bar in the second and the Work tool bar in the third row:



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

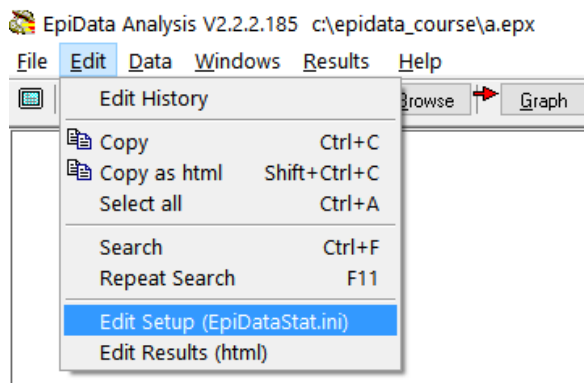
This will open an Explorer-like window:



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
```

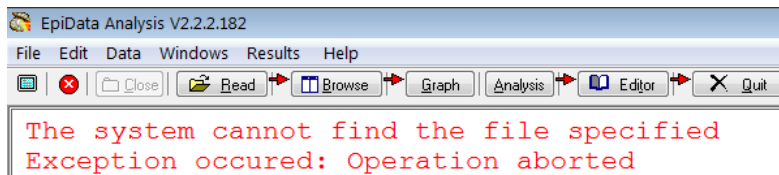
In lines 4 to 16 above we set font sizes, font types and other things (that can be changed according to your needs). It is lines 18 to 20 that are of interest here for the data folder. We start in line 19 with:

[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

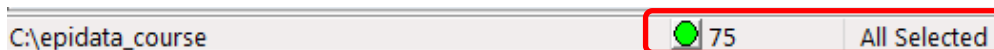
The command to open an EpiData EPX or REC file is Read which can be accessed by clicking on the respective icon in the menu (or with the short-cut **ALT+R**). Make sure (the first time) that you are in the right folder “`c:\epidata_course`” to choose the file:

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
```

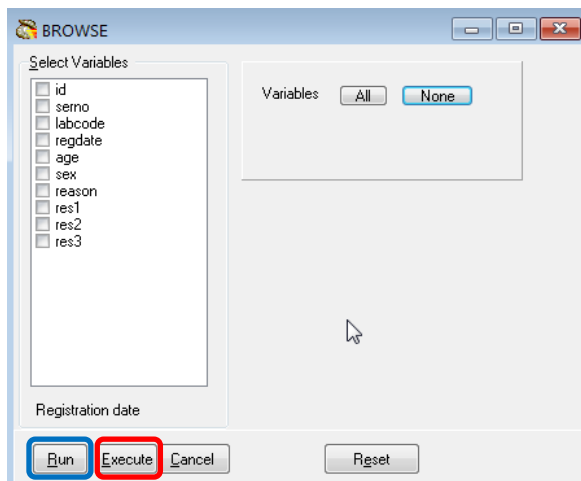
Below the command line you note:



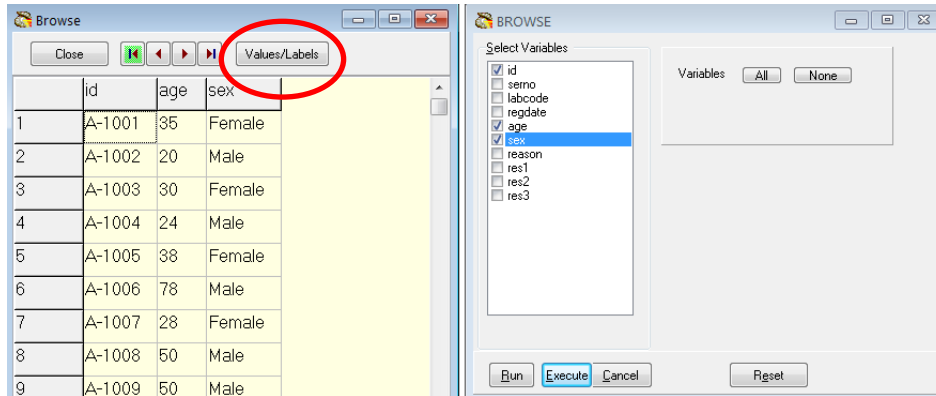
that all of the 75 records are selected. You will see later that any selections you make are listed here and the remaining number of records is shown.

Browsing the EpiData data set

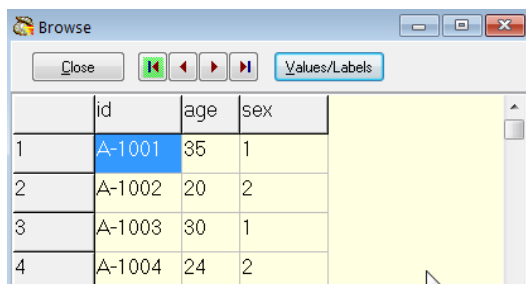
Clicking on Browse (or short-cut **ALT+B**) gives a menu box:



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:

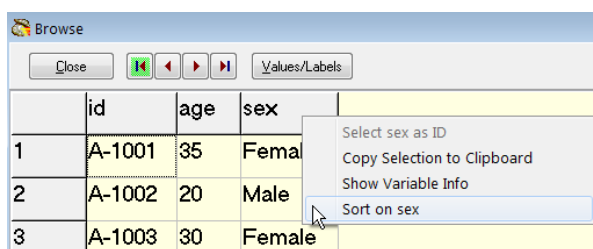


Note the “switch” (toggle key) box Values/Labels. If you click on it you get:



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
```

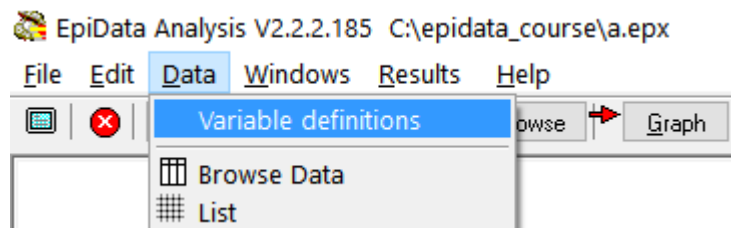
Now you know that the actual syntax to browse is the command BROWSE followed by the variable(s). Note that as in EpiData Entry, the capitalizing is just the way that EpiData shows the commands, it is not something you are required to use.

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:



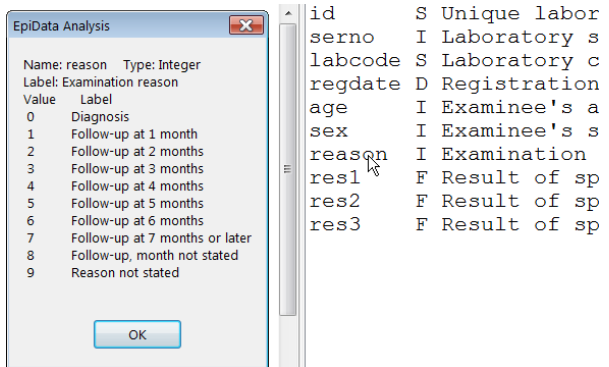
and in the Viewer you get (beginning only shown):

```
. 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.

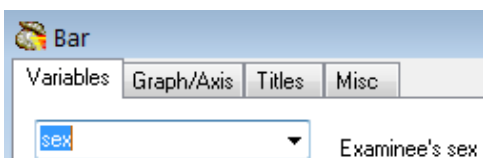
If you are only interested in one specific variable, for instant the variable reason, you can right-click on it in the variables window:



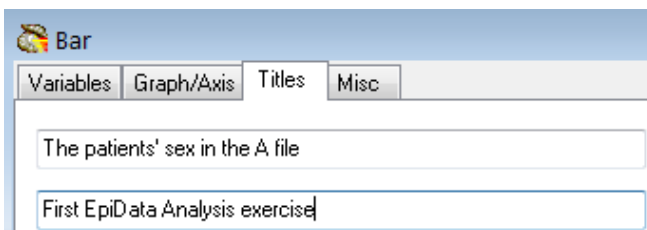
and you get to see what the label block looked like during data entry.

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`:

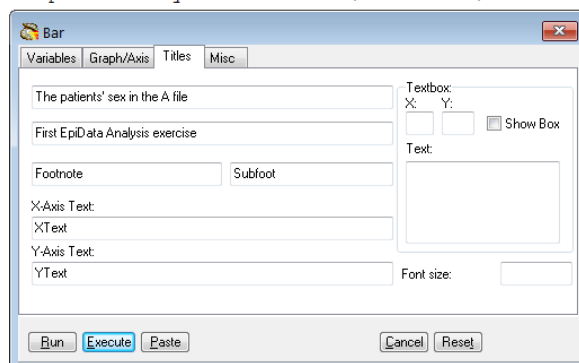
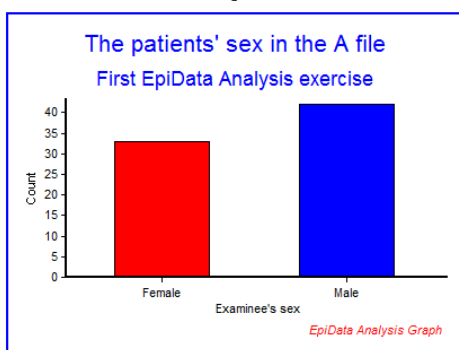


and modify the lines for Title and Sub-Title if you wish:



then click Execute get:

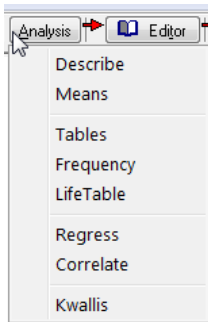
```
. BAR sex /Ti="The patients' sex in the A file" /Sub="First EpiData Analysis exercise" /SizeX=400 /SizeY=300
```



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:



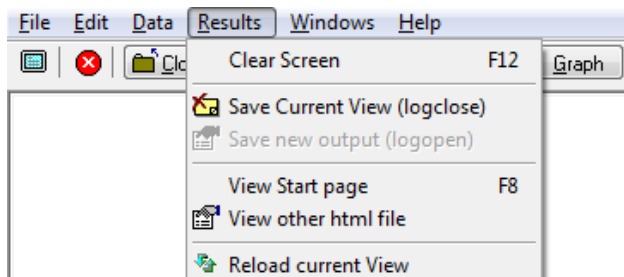
Let's use the first one, Describe to describe the content of a numeric field such as age and get:

```
. 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.

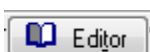
In the Results menu, you can clear the Viewer output (and memory) with Clear screen (short cut function key **F12**):



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

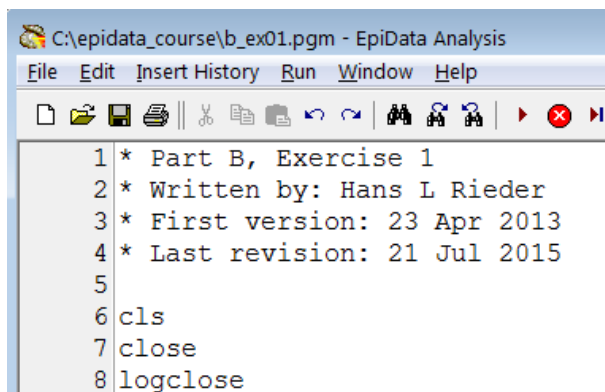
This is all good and fine. It is what might be called “interactive analysis”. The advantage is that you can very quickly “fish” in your dataset and familiarize yourself with it. The disadvantage of interactive analysis is that everything is lost once you leave EpiData Analysis. It defies thus the underwritten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

The solution of EpiData Analysis to documentation is to use:



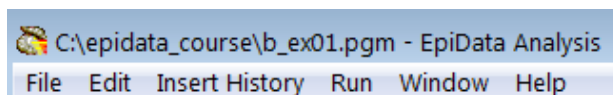
(Function key **F5** also opens the Editor)

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:



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

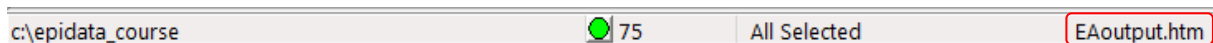
```
cls // This will clear the screen and free memory
```

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.

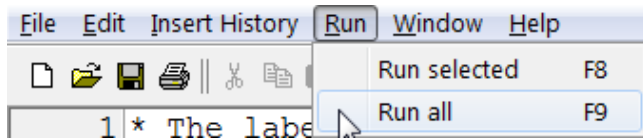
This requirement is a precaution to prevent the inadvertent overwriting of an existing file. The option /replace gives the explicit permission to overwrite the file in case it exists.

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:

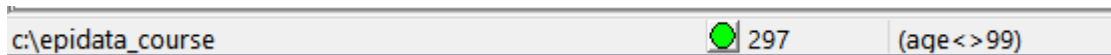
```
= Equal to
< Less than
> Greater than
>= Greater or equal to
<= Less or equal to
<> Unequal
```

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)

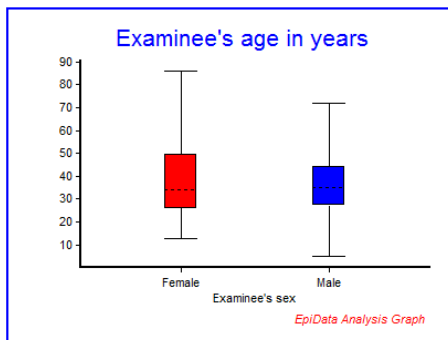
Note:

- 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
```

	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)
Male	168	(88.0)	23	(12.0)
Total	262	(87.3)	38	(12.7)

Percents: (Row)

Chi²= 0.185 df(1) p= 0.6667

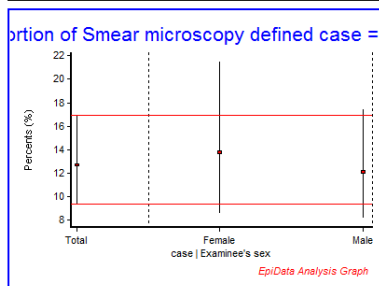
shows us with the option “/t” (for performing the Chi-square test) that there is no significant difference between males and females in being a case. In some instances, when the expected cell value of a 2-by-2 table is less than 5, it is recommended to use Fischer’s exact test instead of the Chi-square 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.*