Exercise 2: More on EpiData Analysis

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

a. Know more about data set and variable manipulation
b. Know more about tables
c. Know more about graphs

More about data set manipulation

Merge and append

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

Set A

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

Set B

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

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

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

We get correctly a file with six records:

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

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

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

We get:

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

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

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

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

Appending

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

Merging

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

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

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

Select

We used `select` as follows:

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

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

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

Drop and Keep

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

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

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

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

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

Sort

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

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

and within each laboratory by registration date:

```
sort labcode regdate
```

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

…

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

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

Define and gen

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

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

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

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

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

There are other field types we can create this way:

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

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

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

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

Recode

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

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

gives:

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

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

String fields and substrings

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

```
cls
close
read "abcd.rec"

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

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

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

and get:

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

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

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

and get:

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

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

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

Thus:

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

gives:

res12	N
NN	262
NP	10
PP	28
Total	300

res23	N
N9	157
NN	105
P9	12
PN	2
PP	24
Total	300

Date fields

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

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

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

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

Results variables

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

```
cls
close
read "abcd.rec"

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

produces:

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

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

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

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

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

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

Visualizing with `BROWSE`:

meanfem	meanmal
37.9907	38.3895
37.9907	38.3895
37.9907	38.3895

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

Writing output into a file

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

The commands are straight forward:

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

Specifically, if we write:

```
cls
close
read "abcd.rec"

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

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

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

```
. logclose
```

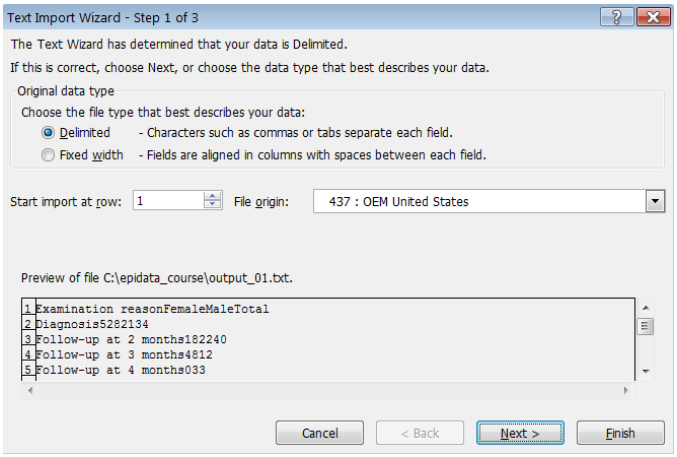
Apart from the "ragged" appearance, there is also too much superfluous output there, if we only want the table. We can greatly improve it by setting the echo first off before opening the log file and then setting it on again after it has been closed:

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

This way, we get it trimmed down:

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

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



it will come out nicely as intended:

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

More about tables

Let's first copy our case definition from above:

```
cls
close
read "abcd.rec"

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

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

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

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

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

```
tables case sex /vl
```

we got:

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

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

```
tables case sex /vl /sd
```

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

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

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

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

"Plain"

Microscopy case definition

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

"Epidemiologic"

Outcome:Microscopy case definition

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

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

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

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

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

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

Outcome:Microscopy case definition

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

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

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

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

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

and we get:

Outcome:Microscopy case definition

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

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

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

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

Stratification

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

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

and get:

Microscopy case definition

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

Unstratified table

Reason for examination: Diagnosis
Microscopy case definition

Examinee's sex	Non-case	Case	Total
Female	40	12	52
Male	64	18	82
Total	104	30	134

Reason for examination: Follow-up
Microscopy case definition

Examinee's sex	Non-case	Case	Total
Female	54	3	57
Male	103	4	107
Total	157	7	164

The first table is the crude, unstratified table, the following table(s) are the results of `case` by `sex` in the different strata. Of course, the interest here is commonly a summary odds ratio calculated by the Mantel-Haenszel procedure:

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

We get more output here: first the "epidemiologic" table with crude odds ratios and 95% confidence intervals followed by these measures of association in the strata:

Outcome:Microscopy case definition

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

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

Odds Ratio = 0.83 (95% CI: 0.41-1.67) (Robins,Greenland,Breslow CI)

Outcome: Reason for examination: Diagnosis
Microscopy case definition

Examinee's sex	Case	Non-case	Total
Male	18	64	82
Female	12	40	52
Total	30	104	134

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

Odds Ratio = 0.94 (95% CI: 0.41-2.15) (Robins,Greenland,Breslow CI)

Outcome: Reason for examination: Follow-up
Microscopy case definition

Examinee's sex	Case	Non-case	Total
Male	4	103	107
Female	3	54	57
Total	7	157	164

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

Odds Ratio = 0.70 (95% CI: 0.15-3.24) (Robins,Greenland,Breslow CI)

After these tables, we get the summary of the adjusted analysis:

Microscopy case definition by Examinee's sex adjusted for Reason for examination

	N = 298	N	OR	(95% CI)
Crude		298	0.83	(0.41-1.67)
Adjusted		298	0.88	(0.42-1.82)
Reason for examination: Diagnosis		134	0.94	(0.41-2.15)
Reason for examination: Follow-up		164	0.70	(0.15-3.24)

Summary Estimates
Total 2 strata. 2 informative & 0 non-informative.
Exposure: Examinee's sex = Male
Outcome: Microscopy case definition = Case

Note also the all-important information at the bottom to help us ensuring that we got what we wanted:

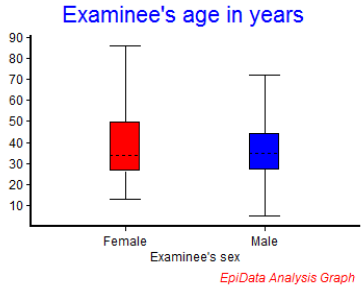
```
Exposure: Examinee's sex = Male
Outcome: Microscopy case definition = Case
```

More about graphs

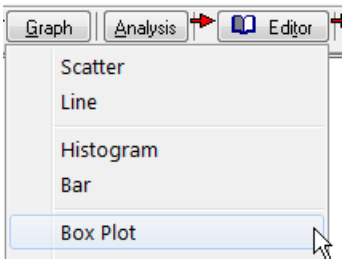
In the previous exercise, we did:

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

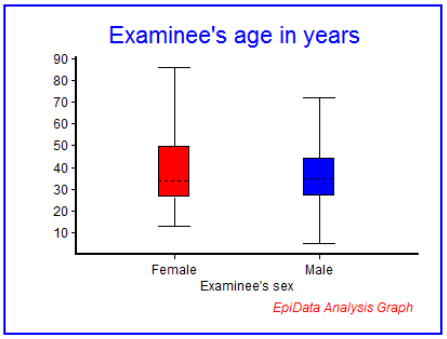
and got this graph:

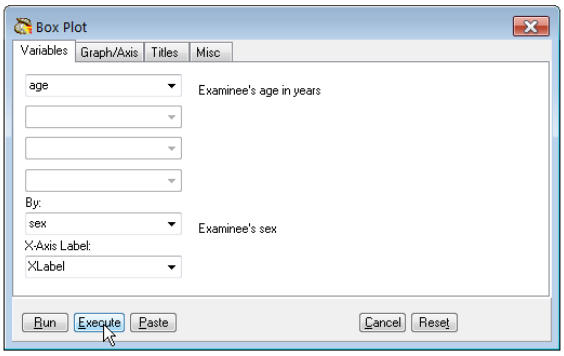


To explore the options a bit more without first taking recourse to the Help file, we will approach it by using the menu interactively:

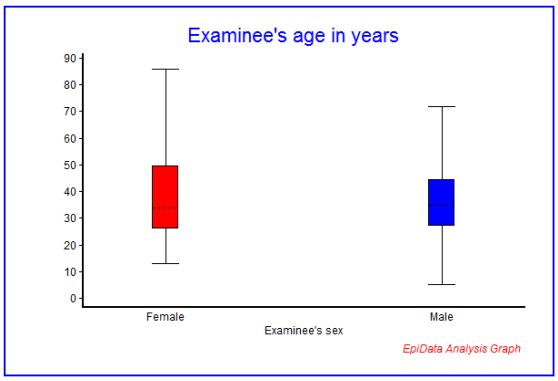


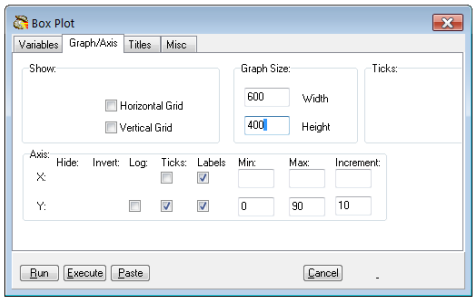
We will use Execute (not Run) to progressively edit the graph to our liking, completing the first tab "Variables":



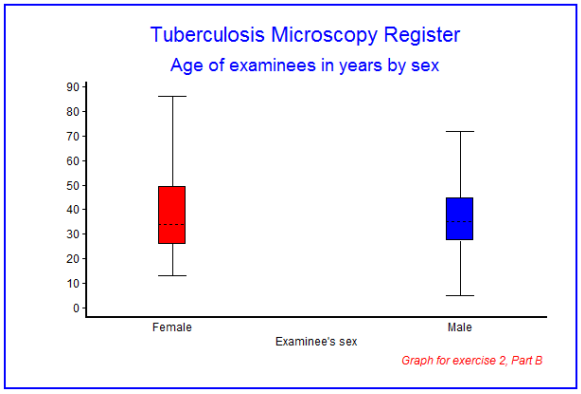


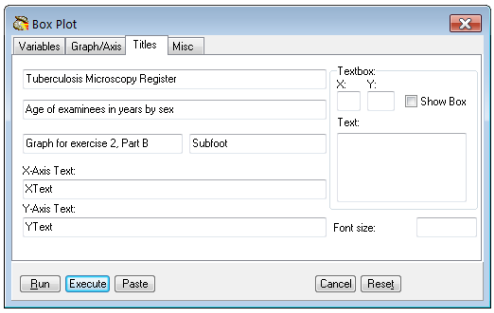
Then we refine the tab “Graph/Axis”:



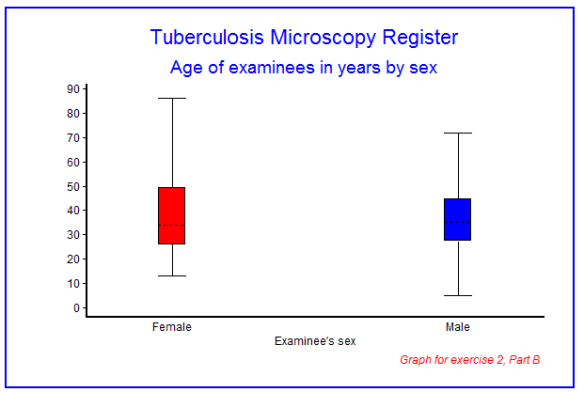


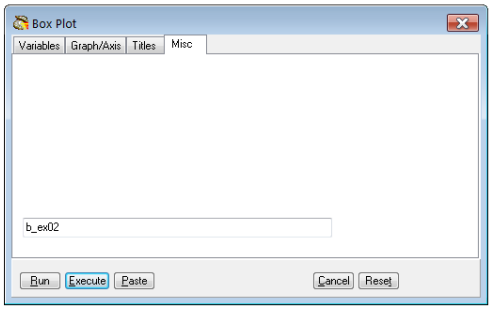
then “Titles”:





and finally “Misc”:

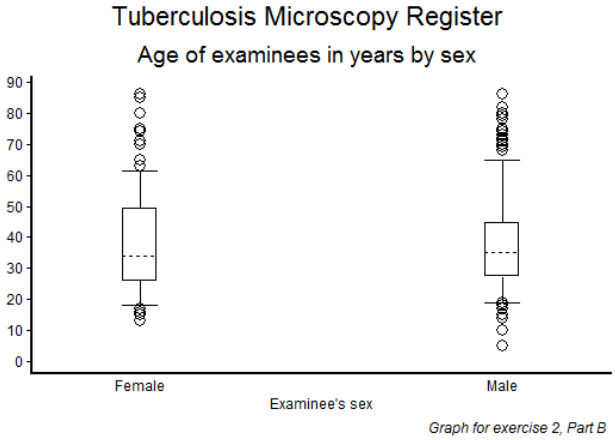




The most important part comes now in that we go to the command line (**F4**) and use the up cursor to get the entire list of commands, mark it, and paste it into our `PGM` file, making back slashes where appropriate and then have:

```
cls
BOXPLOT age /By=sex  \
            /SizeX=600 /SizeY=400 \
            /Noxtick \
            /Ymin=0 /Ymax=90 /Yinc=10 \
            /Ti="Tuberculosis Microscopy Register" \
            /Sub="Age of examinees in years by sex" \
            /Fn="Graph for exercise 2, Part B" \
            /Save="b_ex02"
```

We do some additional editing, like allowing replacement of the graph (which is necessary if we name the output), and look some additional things up in the Help file, and finally have it refined to:



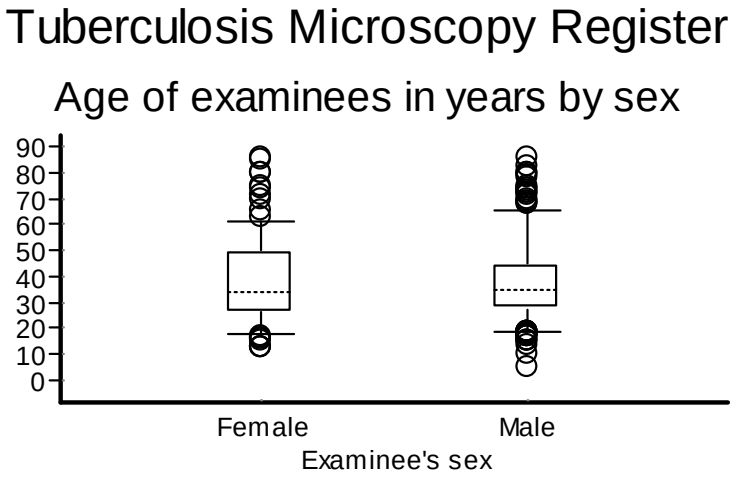
Graph for exercise 2, Part B

Note the difference in quality if we replace:

```
            /Save="b_ex02" /replace \
```

with:

```
            /Save="b_ex02.wmf" /replace \
```



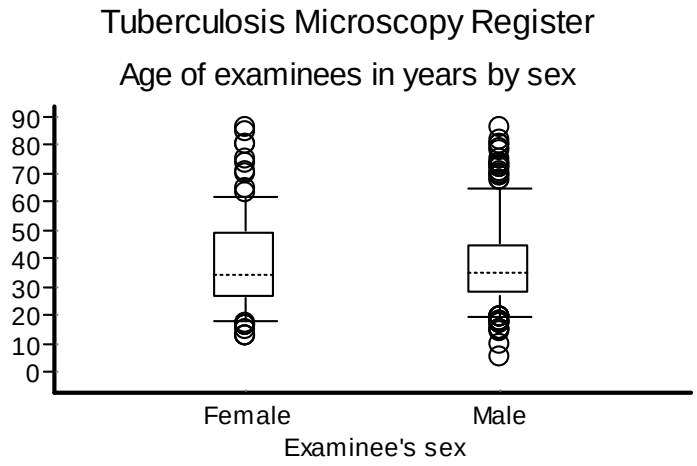
Graph for exercise 2, Part B

We note:

- If we save the graph as `*.wmf` (Microscoft metafile, a vector graph), it does not show up in the Results window
- Vector graphs are crisp and sharp and retain this irrespective of changing the graph size
- The result is not a faithful reproduction of the default `*.png` graph.

Sometimes the title gets too large in the metafile. This can be remediated by adding the SET to change the default font size from 10 to 9:

```
set graph font size=9
```



Graph for exercise 2, Part B

Task:

- ***Write a program B_EX02.PGM using the data set "`abcd.rec`". Limit your analysis to patients with a diagnostic sputum smear examination. Create an output that shows the incremental yield of cases from the first, second and third of three serial smears. With incremental yield we mean determining the proportion of examinees who are positive already on the first, patients who are negative on the first, but positive on the second, and examinees who are negative on the first two but positive on the third serial smear examination. The denominator should be those who have had the required number of smears to determine the yield.***