

## **Part D. More on EpiData software**

### **Part D: More on EpiData software**

Exercise 1: Relational database and aggregating vs from “Long-to-wide”

Exercise 2: A statistical process control chart

Exercise 3: A simplified survival analysis

Exercise 4: Creating a menu for standard reports

Exercise 5: Formatting standardized analysis output in a spreadsheet

### **Introductory note**

Part D will address operationally relevant concepts in data collection and data analysis:

- How do we deal with a situation, where each individual has a varying number of observations?
- How do we determine statistically relevant deviations from a proportion over an observation period when the denominator varies with each time unit over the observation period?
- What is survival analysis and how to deal with it in EpiData Analysis?
- How to make a menu-driven, HTML-based EpiData Analysis interface to run standard reports?

## Exercise 1: A relational database and “Aggregating” vs from “long-to-wide”

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

- a. Create a relational database for a varying number of observations
- b. Merge a child file to the parent file
- c. Recognizing when to use “Aggregate” and when to transpose data
- d. Transpose multiple observations (columns) into a single record (row)

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

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

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

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

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

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

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

Building a relational database requires determining which information is static for an individual and which changes over repeated observations.

## EpiData Manager and EntryClient

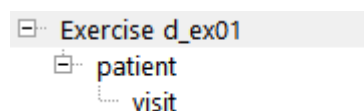
You may refer to Exercise 7 “Relational database” in Part A on the principles of and how to create a relational database.

### The working example

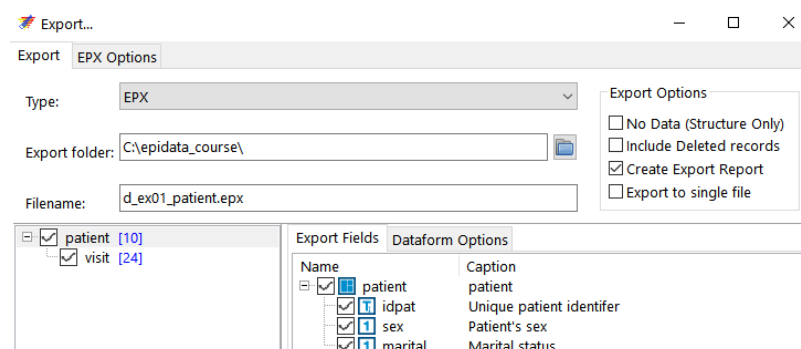
You will create an EpiData file:

d\_ex01.epx

To get uniform naming during design and entry, we propose the following:



The project is saved as “d\_ex01.epx” and the two databases it consists of are the parent dataset named “patient” and the child set “visit”. When exporting:



EpiData will automatically create the appropriate two files:

```
d_ex01_patient.epx  
d_ex01_visit.epx
```

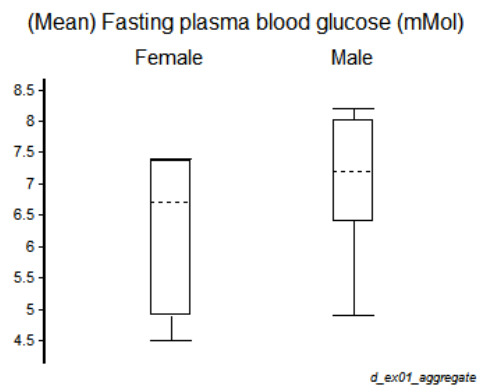
These files will be required in EpiData Analysis for merging.

## EpiData Analysis

The final result of the analysis will be to obtain the following EpiData Analysis output:

---

### Part 1 with AGGREGATE



---

### Part 2 with transposing values

Overall microscopy result			
of 4 serial smears	Negative	Positive	Total
N—	2	0	2
NN—	1	0	1
NN9P	0	1	1
NP—	0	1	1
NPP—	0	3	3
PNP—	0	1	1
PP—	0	1	1
Total	3	7	10

Patient's sex				
Incremental yield of first 3 smears	Female	%	Male	%
NNP	0 (0.0)		1 (25.0)	1 (14.3)
NPx	3 (100.0)		1 (25.0)	4 (57.1)
Px	0 (0.0)		2 (50.0)	2 (28.6)
Total	3 (100.0)		4 (100.0)	7

Percents: (Col)

It doesn't really look like much. Nevertheless, quite a few steps are needed to get from the source files D\_EX01\_PATIENT.EPX and D\_EX01\_VISIT.EPX to this point. We will elaborate on some considerations you have to make and offer hints for the sequential components in EpiData Analysis.

## Merging the files

In EpiData Manager you made a relation through a unique identifier from the parent to the child file. In EpiData Analysis you must now merge these to files to get the following dataset.

We start by reading the child file:

```
read "childfile.epx"  
merge parentidentifier /file="parentfile.epx" /table
```

To each record from the child file, information from the parent file is added repetitively for each observation for the same individual. In other words, you start from the other way around than in EpiData EntryClient, starting in EpiData Analysis with the child file and using the parent as a lookup table.

As a result we get (first 13 records only shown):

	idpat	visitid	visitdate	bs	sputum	micres	sex	marital	MergeVar
1	A	A-25-03-2007	25/03/2007	7.3	Salivary	Negative	Male	Married	In both
2	A	A-26-03-2007	26/03/2007	7.2	Purulent	1+ positive	Male	Married	In both
3	A	A-24-03-2007	24/03/2007	6.3	Mucoid	1+ positive	Male	Married	In both
4	B	B-24-03-2007	24/03/2007	4.9	Muco-purulent	Negative	Male	Divorced	In both
5	C	C-24-03-2007	24/03/2007	5.2	Purulent	Negative	Female	Single	In both
6	C	C-26-03-2007	26/03/2007	4.8	Muco-purulent	Negative	Female	Single	In both
7	D	D-25-03-2007	25/03/2007	7.4	Mucoid	2+ positive	Female	Widowed	In both
8	D	D-24-03-2007	24/03/2007	7.3	Blood-tinged	Scanty positive	Female	Widowed	In both
9	E	E-28-03-2007	28/03/2007	7.9	Purulent	2+ positive	Male	Married	In both
10	E	E-27-03-2007	27/03/2007	8.2	Not recorded	1+ positive	Male	Married	In both
11	F	F-01-04-2007	01/04/2007	7.7	Purulent	2+ positive	Female	Annulled	In both
12	F	F-31-03-2007	31/03/2007	7.2	Muco-purulent	3+ positive	Female	Annulled	In both

A variable MERGEVAR has been created by EpiData Analysis. It can take 3 values:

- 1 Only in memory (original)
- 2 Only in external file
- 3 In both

A frequency helps to identify quickly whether there are for instance any “orphans”, that is child records which do not find the same identifier in a parent record and are thus useless.

We note in the above that the visit dates are not temporally sequential within each parent identifier, we need thus sorting, first by parent identifier, then within them by date, thus:

```
sort idpat visitdate
```

However, this may not be correct in all circumstances.

**Note:** We commonly suggested to code unknown dates as “01/01/1800”. With sorting, the unknown date will thus come first (sorting is ascending by default) and this might not be desirable. In this dataset we don’t have unknown dates. But only because the small dataset allows us to see that, you would have to anticipate that possibility with a larger dataset and thus first make a new date variable where the unknowns take a value for a date in the future, so they appear with sorting at the end of the information on an individual.

Following the sorting, it might be advantageous to number the visits in order to be able to make a frequency on them to see what the maximum number of visits is (here we can see that it is a maximum of 4 visits, but in a large database, it would be more difficult).

To create a new variable EXAM and set its default initially to 1 (each record takes first the value 1), we have so far used the following grammar:

```
define exam #
exam=1
```

or alternatively the one-line alternative approach which accomplishes exactly the same thing:

```
gen i exam=1
```

**Note:** the command *GEN* will produce integer of length of 9. If you need a shorter and fixed field length integer, you must utilize *DEFINE*.

The command *GEN* replaces *DEFINE* and “i” stands for an integer field.

There are other similar commands (see an earlier Exercise):

```
gen f doorheight=1.85
gen d birthdate=dmy(31,12,1899)
gen s firstname="john"
```

for date fields (*d*) float (real number) fields (*f*), and string (text) fields (*s*). Look it up in the help file (type *gen+F1* in the command line).

Now that you have a new variable *EXAM*, how can you tell EpiData that it should look at the person (identified by an *ID*) and number each visit, starting with 1 until the next individual comes, when it must start again with 1. The command is:

```
if id=id[_n-1] then visit=visit[_n-1]+1
```

This looks admittedly complex. Let’s thus take it apart. *[\_n]* identifies the current record and accordingly *[\_n-1]* the immediately preceding record. Let’s say, EpiData Analysis has proceeded to record 547 and looks at the *ID* of this record. It looks whether record 546 had the same *ID* as record number 547: *if id=id[\_n-1]*. If that is the case, then it should take the *VISIT* number of record 546 and add 1 to it for record 547: *visit=visit[\_n-1]+1*. If it is not the case, then the default stays (which we defined as 1) and it moves on to the next record.

As a result, you should get:

	idpat	visitdate	visit
1	A	24/03/2007	1
2	A	25/03/2007	2
3	A	26/03/2007	3
4	B	24/03/2007	1
5	C	24/03/2007	1
6	C	26/03/2007	2
7	D	24/03/2007	1
8	D	25/03/2007	2
9	E	27/03/2007	1
10	E	28/03/2007	2
11	F	27/03/2007	1
12	F	31/03/2007	2
13	F	01/04/2007	3

the first variable column showing the identifier, the second the date of the examination, and the third the number of the examination for that individual.

## Aggregating data

If we look at the sex (given the field name SEX), date of visit (given the field name VISITDATE), and blood sugar (given the field name BS):

	idpat	sex	visitdate	bs
1	A	Male	24/03/2007	6.3
2	A	Male	25/03/2007	7.3
3	A	Male	26/03/2007	7.2
4	B	Male	24/03/2007	4.9
5	C	Female	24/03/2007	5.2
6	C	Female	26/03/2007	4.8

SEX remains obviously the same (and is thus from the parent file), and is a categorical variable unique to the examined person, while BS varies by examination date and is a continuous variable. If we want to examine blood sugar by sex, there is no need to transpose the blood sugar values from the vertical to the horizontal as EpiData Analysis has inbuilt a tool to aggregate the data. For the individual and its sex we would write:

```
aggregate idpat sex /close
```

and get with BROWSE the ten individuals and their SEX:

	idpat	sex	N
1	A	Male	3
2	B	Male	1
3	C	Female	2
4	D	Female	2
5	E	Male	2
6	F	Female	3
7	G	Male	3
8	H	Female	3
9	I	Male	4
10	K	Female	1

We can expand this command using an option to calculate the MEAN of blood sugar for each individual:

```
aggregate idpat sex /mean="bs" /close
```

and get:

	idpat	sex	N	Nbs	MEAbs
1	A	Male	3	3	6.9333333333
2	B	Male	1	1	4.9000000000
3	C	Female	2	2	5.0000000000
4	D	Female	2	2	7.3500000000
5	E	Male	2	2	8.0500000000
6	F	Female	3	3	7.4333333333
7	G	Male	3	3	7.2333333333
8	H	Female	3	3	6.7000000000
9	I	Male	4	4	8.1500000000
10	K	Female	1	1	4.5000000000

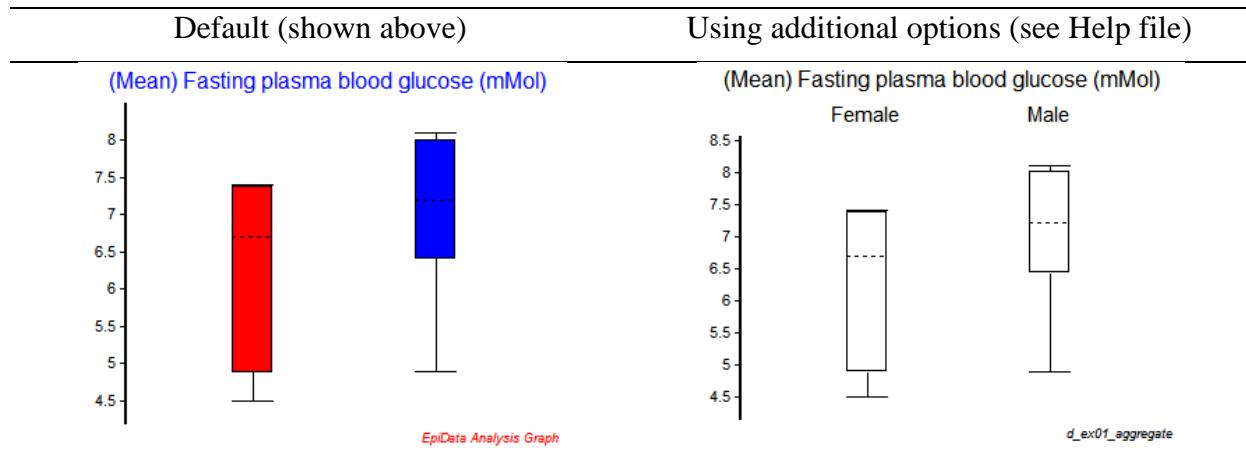
where MEAbs is the calculated mean value of blood sugar for an individual from all the individual's measurements. To save the aggregate data in a file, we add another option:

```
aggregate patid sex /mean="bs" /close /save="d_ex01_aggregate.rec" /replace
```

Having accomplished this, it is now straight forward to write:

```
cls
close
read "d_ex01_aggregate.rec"
boxplot meabs /by=sex
```

and get:



### *From long-to-wide*

For showing means, there is thus no need to transpose data from the vertical to the horizontal. But it is different for the macroscopic aspect of sputum the examination results. We do not want (nor would we get a sensible result) aggregate sputum smear results. We wish to know each result from each individual.

The first thing before starting copying results from the vertical to the horizontal, we need to know the maximum number of examinations an individual had in the data set. We can get this with a frequency on the VISIT:

```
freq visit
```

Number of visit	
	N
1	10
2	8
3	5
4	1
Total	24

This shows that the maximum number of examinations an individual had in this dataset was 4. We need thus to prepare 4 new variables for each field that are in the sequence of the examination in the vertical but should also become part of each record.

Let's say we have a variable VAR1 with different values for each visit:

ID	VISIT	VAR1
B1	1	1
B1	2	3
B1	3	2
B1	4	2
C1	1	3
D1	1	2



What we need is:

ID	VISIT	VAR1	VAR11	VAR12	VAR13	VAR14
B1	1	1				
B1	2	3				
B1	3	6				
B1	4	2				
C1	1	3				
D1	1	2				

First, we make these 4 variables and give them all the default value of -1 (the minus one is a good way to see missing data and helps later in the selection):

```
gen i var11=-1
gen i var12=-1
gen i var13=-1
gen i var14=-1
```

After these four command lines, our above dataset becomes:

ID	VISIT	VAR1	VAR11	VAR12	VAR13	VAR14
B1	1	1	-1	-1	-1	-1
B1	2	3	-1	-1	-1	-1
B1	3	6	-1	-1	-1	-1
B1	4	2	-1	-1	-1	-1
C1	1	3	-1	-1	-1	-1
D1	1	2	-1	-1	-1	-1

and we are ready to copy the values from the vertical to the horizontal by respecting that the value of VAR1 from VISIT 1 goes to VAR11, the value from VISIT 2 to VAR12, etc.

For VISIT 1, the value for VAR11 is equal to the value of VAR1 and we make it thus the default:

```
VAR11=VAR1
```

Then we use the same approach as above to identify the record:

```
if id[_n]=id[_n+1] then var12=var1[_n+1]
```

This means that if the current record [\_n] has the same ID as the next record [\_n+1], then VAR12 in the current record should take the value of VAR1 from the next record [\_n+1].

Now we do this for all possible 4 records (the maximum of VISITs):

```
if id[_n]=id[_n+2] then var13=var1[_n+2]
if id[_n]=id[_n+3] then var14=var1[_n+3]
```

All the lines to be written for this original field VAR1 are thus:

```
gen i var11=-1
gen i var12=-1
gen i var13=-1
gen i var14=-1
VAR11=VAR1
if id[_n]=id[_n+1] then var12=var1[_n+1]
if id[_n]=id[_n+2] then var13=var1[_n+2]
if id[_n]=id[_n+3] then var14=var1[_n+3]
```

and we get (assuming that the last patient D1 had only 1 VISIT):

ID	VISIT	VAR1	VAR11	VAR12	VAR13	VAR14
B1	1	1	1	3	6	2
B1	2	3	1	3	6	-1
B1	3	6	1	3	-1	-1
B1	4	2	1	-1	-1	-1
C1	1	3	3	-1	-1	-1
D1	1	2	2	-1	-1	-1

You may note that only for VISIT 1 for patient B1 with four visits all new 4 variables have all respective 4 values from the 4 VISITS.

Now we can safely get rid of the records of VISITS 2, 3, and 4 and we end up just with individuals who have all information from each VISIT:

```
select visit=1
```

and we get:

ID	VISIT	VAR1	VAR11	VAR12	VAR13	VAR14
B1	1	1	1	3	6	2
C1	1	3	3	-1	-1	-1
D1	1	2	2	-1	-1	-1

The conversion from “Long-to-wide” is thus successfully completed, well, for this variable. Of course, before you make this selection, you have to repeat the same approach for each field, so that in the end you get something like:

	idpat	sex	marital	visitdate1	visitdate2	visitdate3	visitdate4	sputum1	sputum2	sputum3	sputum4	micros1	micros2	micros3	micros4	pattern	case	yield
1	A	Male	Married	24/03/2007	25/03/2007	26/03/2007		Mucoid	Salivary	Muco-purulent		1+ positive	Negative	1+ positive		PNP	Positive	Px
2	B	Male	Divorced	24/03/2007				Purulent				Negative				N	Negative	
3	C	Female	Single	24/03/2007	26/03/2007			Muco-purulent	Purulent			Negative	Negative			NN	Negative	
4	D	Female	Widowed	24/03/2007	26/03/2007			Blood-tinged	Mucoid			Scanty positive	1+ positive			PP	Positive	Px
5	E	Male	Married	27/03/2007	28/03/2007			Not recorded	Muco-purulent			1+ positive	1+ positive			PP	Positive	Px
6	F	Female	Annulled	27/03/2007	31/03/2007	01/04/2007		Muco-purulent	Purulent	Muco-purulent		Negative	1+ positive	1+ positive		NPP	Positive	NPx
7	G	Male	Cohabiting	27/03/2007	28/03/2007	01/04/2007		Salivary	Mucoid	Purulent		Negative	1+ positive	1+ positive		NPP	Positive	NPx
8	H	Female	Married	31/03/2007	01/04/2007	02/04/2007		Mucoid	Purulent	Mucoid		Negative	1+ positive	1+ positive		NPP	Positive	NPx
9	I	Male	Separated	31/03/2007	01/04/2007	02/04/2007	03/04/2007	Salivary	Mucoid	Purulent	Mucoid	Negative	Scanty posi	Not recorded	1+ positive	NP9P	Positive	NPx
10	K	Female	Married	02/04/2007				Muco-purulent				Negative				N	Negative	

You have now ten patients and you are at the point where you can continue to work in the same way as you used to work before. While it is much more complex to get to here from 1 line per examination than from 1 line per examinee, it is also obvious that in the end this is much more informative.

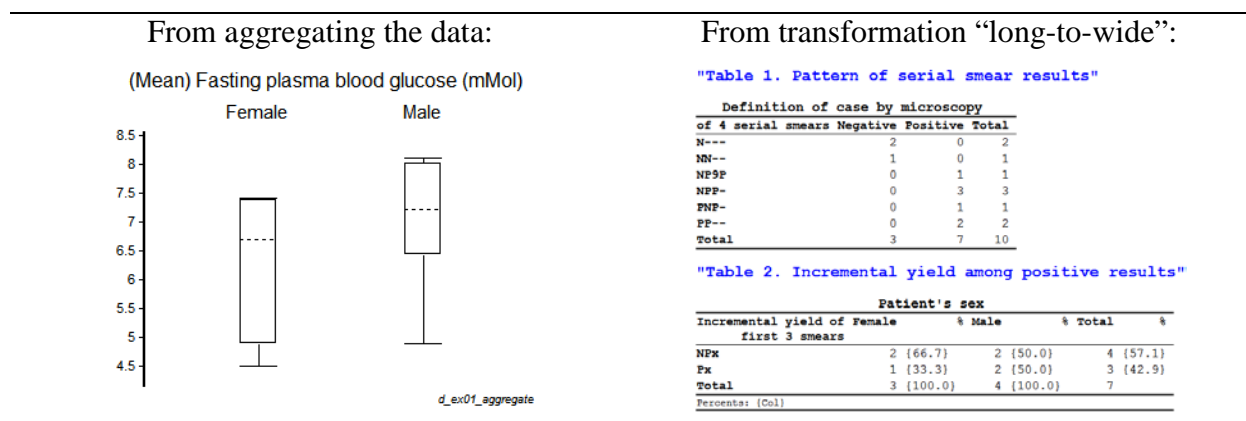
For each examination we have the date and for each specimen we have the quality of the sputum. In the Union / WHO approach you have only one date (the date of collection of the first specimen) for a series of three, and the quality of sputum in the Tuberculosis Laboratory is not that informative as it is very possible that every day the quality of the specimen is different, but there is no space allocated to write 3 different ones.

### Tasks:

- *Prepare a data documentation sheet*
- *Prepare the EpiData Manager form for the relational database*
- *Enter the data from the following sample data set:*

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

- Write a program *D\_EX01.PGM* that merges the two files, then prepare sets for the aggregated data and for the “long-to-wide” transformation to produce the following output respectively:



## Solution to Exercise 1: A relational database and “Aggregating” vs from “long-to-wide”

### Key points:

- A relational database is the solution to a varying number of observations per individual
- The child file is merged with the parent file to give a dataset of all observations
- To obtain means for an individual from continuous variables, aggregating the data is the strategy of choice
- To reduce the dataset to individuals with information on each examination, one must copy the information from observations in the vertical to newly created fields in the first observation of each individual before selecting that record from the individual (“long-to-wide”)

### Task

- *Prepare a data documentation sheet*

The documentation sheet is shown on the next page.

### Task

- *Prepare the EpiData Manager form for the relational database*

The data entry forms may be made as follows:

D_EX01_PATIENT.EPX	D_EX01_VISIT.EPX
<p><b>Entry form for the patient</b></p> <p>Unique patient identifier <input type="text"/></p> <p>Patient's sex <input type="text"/> label_sex</p> <p>Marital status <input type="text"/> label_marital</p>	<p><b>Entry form for the visit</b></p> <p>Unique patient identifier <input type="text"/></p> <p>Unique visit identifier <input type="text"/></p> <p>Date of visit <input type="text"/></p> <p>Plasma glucose in mMol/L <input type="text"/></p> <p>Quality aspect of sputum <input type="text"/> label_sputum</p> <p>Microscopy result <input type="text"/> label_micres</p>

The data documentation sheet:

### Data documentation sheet

	Field name	Field label	Field type	Field length	Field values	Value label	Field comment
Patient file	idpat	Unique patient identifier	U	1	A,...,Z		Any given unique ID
	sex	Patient's sex	I	1	1 Female 2 Male 3 Unknown		
	marital	Patient's marital status	I	1	1 Single 2 Married 3 Cohabiting 4 Annulled 5 Divorced 6 Widowed 7 Separated 8 Engaged 9 Not recorded		

NOTE: If SEX is given during an earlier examination and left empty in a subsequent examination, update the information to known  
 If SEX is different in different examinations, record as UNKNOWN  
 If MARITAL is given during an earlier examination and left empty in a subsequent examination, keep the initial information from earlier  
 If MARITAL is different in different examinations, update to most recent information

Examination file	idvisit	Unique examination identifier	S	12	A-2007-01-31,...		Automatically calculated
	visitdate	Date of visit	dd/mm/yyyy	10	01/01/2007,...,31/12/2007 01/01/1800		Legal visit date recordings Enter if visit date is missing
	bs	Fasting plasma blood glucose (mMol)	F	4	2.5,...,19.9 99.9		Legal valid value Enter if blood sugar is missing
	sputum	Macroscopic sputum aspect	I	1	1 Mucoid 2 Purulent		

micres

Microscopy result

I

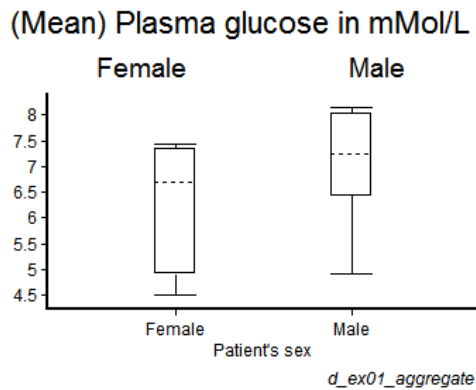
1

- 3 Muco-purulent
- 4 Blood-tinged
- 5 Salivary
- 9 Not recorded
- 0 Negative
- 1 "1+ positive"
- 2 "2+ positive"
- 3 "3+ positive"
- 4 "Scanty positive"
- 9 "Not recorded"

### Task:

- Write a program *D\_EX01.PGM* that merges the two files, then prepare sets for the aggregated data and for the “long-to-wide” transformation to produce the following output respectively:

From aggregating the data:



From transformation “long-to-wide”:

"Table 1. Pattern of serial smear results"

Definition of case by microscopy			
of 4 serial smears	Negative	Positive	Total
N---	2	0	2
NN--	1	0	1
NP9P	0	1	1
NPF-	0	3	3
PNF-	0	1	1
PP--	0	2	2
Total	3	7	10

"Table 2. Incremental yield among positive results"

Patient's sex			
Incremental yield of Female	% Male	% Total	%
first 3 smears			
NPx	2 {66.7}	2 {50.0}	4 {57.1}
Px	1 {33.3}	2 {50.0}	3 {42.9}
Total	3 {100.0}	4 {100.0}	7

Percents: {Col}

The *D\_EX01.PGM* reads:

```
* Part D, Exercise 1
* Merging files and aggregating files
* Copying and transposing data from "Long-to-wide"

* Written by:      Hans L Rieder
* First version: 17 Jan 2010
* Last revision: 11 Nov 2016

cls
close
logclose

*****
* Merge child and parent files

cls
close

read "d_ex01_visit.epx"
merge idpat /file="d_ex01_patient.epx" /table

sort idpat visitdate
gen i visit=1
if idpat=idpat[_n-1] then visit=visit[_n-1]+1
label visit "Visit number"

savedata "temp_01.rec" /replace

*****
* Create an aggregate data set
* to determine means

cls
close
read "temp_01.rec"
```

```

aggregate idpat sex /mean="bs" /close /save="d_ex01_aggregate.rec" /replace

cls
close
read "d_ex01_aggregate.rec"

* set display databrowser=on
* browse
* tables sex meabs          //Testing here only, will be done in Analysis
* boxplot meabs /by=sex     //Testing here only, will be done in Analysis

*****
* Transpose , copy "long-to-wide"

cls
close
read "temp_01.rec"

* freq exam
* => Maximum is 4 visits

cls
gen d visitdate1
gen d visitdate2
gen d visitdate3
gen d visitdate4

                                visitdate1=visitdate
if (idpat[_n])=(idpat[_n+1]) then visitdate2=visitdate[_n+1]
if (idpat[_n])=(idpat[_n+2]) then visitdate3=visitdate[_n+2]
if (idpat[_n])=(idpat[_n+3]) then visitdate4=visitdate[_n+3]

cls
define bs1 ##.#
define bs2 ##.#
define bs3 ##.#
define bs4 ##.#

                                bs1=bs
if (idpat[_n])=(idpat[_n+1]) then bs2=bs[_n+1]
if (idpat[_n])=(idpat[_n+2]) then bs3=bs[_n+2]
if (idpat[_n])=(idpat[_n+3]) then bs4=bs[_n+3]

cls
gen i sputum1
gen i sputum2
gen i sputum3
gen i sputum4

                                sputum1=sputum
if (idpat[_n])=(idpat[_n+1]) then sputum2=sputum[_n+1]
if (idpat[_n])=(idpat[_n+2]) then sputum3=sputum[_n+2]
if (idpat[_n])=(idpat[_n+3]) then sputum4=sputum[_n+3]

cls
gen i micres1
gen i micres2
gen i micres3
gen i micres4

                                micres1=micres
if (idpat[_n])=(idpat[_n+1]) then micres2=micres[_n+1]
if (idpat[_n])=(idpat[_n+2]) then micres3=micres[_n+2]
if (idpat[_n])=(idpat[_n+3]) then micres4=micres[_n+3]

cls
label visitdate1 "Date of 1st visit"
label visitdate2 "Date of 2nd visit"
label visitdate3 "Date of 3rd visit"

```



```

label visitdate4 "Date of 4th visit"
cls
label bs1 "Plasma glucose at 1st visit"
label bs2 "Plasma glucose at 2nd visit"
label bs3 "Plasma glucose at 3rd visit"
label bs4 "Plasma glucose at 4th visit"
cls
label sputum1 "Macroscopic sputum of 1st visit"
label sputum2 "Macroscopic sputum of 2nd visit"
label sputum3 "Macroscopic sputum of 3rd visit"
label sputum4 "Macroscopic sputum of 4th visit"
labelvalue sputum1-sputum4 /1="Mucoid"
labelvalue sputum1-sputum4 /2="Purulent"
labelvalue sputum1-sputum4 /3="Muco-purulent"
labelvalue sputum1-sputum4 /4="Blood-tinged"
labelvalue sputum1-sputum4 /5="Salivary"
labelvalue sputum1-sputum4 /9="Not recorded"
cls
label micres1 "Microscopy result of 1st visit"
label micres2 "Microscopy result of 2nd visit"
label micres3 "Microscopy result of 3rd visit"
label micres4 "Microscopy result of 4th visit"
labelvalue micres1-micres4 /0="Negative"
labelvalue micres1-micres4 /1="1+ positive"
labelvalue micres1-micres4 /2="1+ positive"
labelvalue micres1-micres4 /3="1+ positive"
labelvalue micres1-micres4 /4="Scanty positive"
labelvalue micres1-micres4 /9="Not recorded"
cls
label marital "Civil status"

select visit=1
drop visitid mergevar visit

savedata "temp_02.rec" /replace

cls
close
read "temp_02.rec"

define restxt1 _
define restxt2 _
define restxt3 _
define restxt4 _

cls
restxt1="-"
if micres1>0 and micres1<9 then restxt1="P"
if micres1=0 then restxt1="N"
if micres1=9 then restxt1="9"
cls
restxt2="-"
if micres2>0 and micres2<9 then restxt2="P"
if micres2=0 then restxt2="N"
if micres2=9 then restxt2="9"
cls
restxt3="-"
if micres3>0 and micres3<9 then restxt3="P"
if micres3=0 then restxt3="N"
if micres3=9 then restxt3="9"
cls
restxt4="-"
if micres4>0 and micres4<9 then restxt4="P"
if micres4=0 then restxt4="N"
if micres4=9 then restxt4="9"

```

```

cls
define pattern ____
pattern=restxt1+restxt2+restxt3+restxt4
label pattern "Pattern of 4 serial smears"

* freq pattern

cls
define case #
                                case=0
if substr(pattern,1,1)="P" then case=1
if substr(pattern,2,1)="P" then case=1
if substr(pattern,3,1)="P" then case=1
if substr(pattern,4,1)="P" then case=1

label case "Definition of case by microscopy"
labelvalue case /0="Negative"
labelvalue case /1="Positive"

define yield ____
if substr(pattern,1,3)="N--" then yield="N99"
if substr(pattern,1,3)="NN-" then yield="NN9"
if substr(pattern,1,3)="NP9" then yield="NPx"
if substr(pattern,1,3)="NPP" then yield="NPx"
if substr(pattern,1,3)="PNP" then yield="Px "
if substr(pattern,1,3)="PP-" then yield="Px"
label yield "Incremental yield of first 3 smears"

keep idpat sex marital \
    visitdate1 visitdate2 visitdate3 visitdate4 \
    sputum1 sputum2 sputum3 sputum4 \
    micres1 micres2 micres3 micres4 \
    case pattern yield
savedata "d_ex01.rec" /replace

*****
* Produce tables on smear pattern and incremental yield

cls
close
read "d_ex01_aggregate.rec"

set option graph /sizex=400
set graph footnote="d_ex01_aggregate"

set echo=off
cls
boxplot meabs /by=sex /bw /sub="    Female                Male"

cls
close
read "d_ex01.rec"

title "Table 1.  Pattern of serial smear results"
tables case pattern
select case=1
title "Table 2.  Incremental yield among positive results"
tables sex yield /c /PCT
set echo=on

*****
* Clean up

set echo=off
define yesno # global
cls

```

```
yesno=?Delete temporary files: 1=yes 0=no?
imif yesno=1 then
  erasepng /all /noconfirm
  erase "temp_01.rec"
  erase "temp_01.chk"
  erase "temp_02.rec"
  erase "temp_02.chk"
  erase "d_ex01_aggregate.chk"
  erase "d_ex01_aggregate.rec"
  select
  cls
  type "All temporary files erased" /h2
else
  select
  type "File D_EX01_EXAMINEE.REC remains open" /h2
endif
set echo=on
```

## Exercise 2: A statistical process control chart

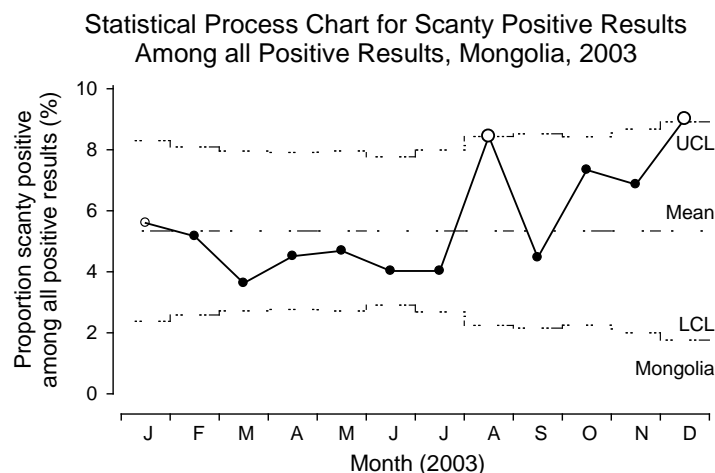
At the end of this exercise you should be able to:

- Aggregate data into the format required for a binomial outcome
- Create a statistical process control chart for a proportion

EpiData Analysis offers a variety of statistical process control (SPC) graphs. In this exercise we will deal with the determination of a proportion that changes over a period of time, and to what extent this variation deviates significantly from the expectation.

Let's assume that you have 1,200 observations during one year in a laboratory. Among these, 10 per cent (120) have positive result. We could determine the standard deviation and a confidence interval around the positive result or go one step further and take the average we expect for one month (12 of 100) with some measurement of uncertainty around this monthly estimate and then chart the actually observed monthly proportion. This would be a correct procedure if the expected monthly denominator is exactly one twelfth of the annual observation. However, this is rarely the case if ever and more likely is the scenario that the denominator varies in each month.

An SPC graph takes these fluctuations in the denominator into account and calculates the uncertainty as a function of the denominator in the element that is of interest (in this example the month). While there are some discussions on what is best to use, it has become customary to use 3 standard deviations as the upper and lower, so-called control limits. In the chart below, the proportion of scanty positive sputum smear microscopy results among all positive results is shown for Mongolia from the large laboratory register study over a one-year period:



Because the denominator differs in every month, the upper and lower control limits also differ from month to month.

We will be looking at examinations (not at examinees) and determine five different proportions (see below).

## **The dataset for the exercise**

The dataset to be used in this exercise is the cleaned dataset of the four-country laboratory study which was provided in the solution to Part C, Exercise 1, dataset C\_EX01.REC which is included as a “supplementary required file” with the current exercise with the name MMUZ.REC. MMUZ.REC is slightly different from C\_EX01.REC in that the coding for the registration date has been corrected in a few records with an apparent error and the field REGYEAR has been removed.

To simplify the task (see specifics at the end), you will limit the analysis to the laboratories in Uganda and to tuberculosis suspects presenting for a diagnostic examination.

## **The time periods**

The Uganda dataset contains information on three years and the unit of measurement of time will be the month. Because each month will thus appear three times (in each year), a new variable must be created that gives a sequential number for each month over the three-year period.

## **The outcome**

The outcome is the monthly proportion of some type of positive results among either all smears or among positive smears.

There are five different proportions we might be interested in: 1) positive smears of any grade among all smears, 2) scanty positive smears among all smears, 3) low-positive smears (defined as either scanty positive or 1+ positive) among all smears, 4) scanty positive smears among all positive smears, and 5) low-positive smears among all positive smears.

## **Aggregating the data into the correct format**

What is thus needed are counts of examinations with the characteristic (a scanty, a low-positive, or any positive result) among all examinations or among all positive examinations. To this end, we aggregate the data as detailed in Part B, Exercise 3.

## **Making a statistical process control (SPC) chart for proportions over time**

The above aggregate file is all we need to get an SPC chart, the general command for which is:

```
pchart numerator denominator [time unit]
```

## **Proposed procedure in writing the program**

It is proposed to split up the program into five distinct sequential procedures:

- 1) Make the basic dataset
- 2) Count all smears, all positive, all low-positive smears, and all scanty positive smears
- 3) Aggregate the months and sum up the smears in question
- 4) Make the SPC charts

### 1) Make the basic dataset

In the basic dataset we need to create the years and the months as separate variables and make the appropriate selections:

*Month and year of recording:* the registers were collected reporting laboratory results during one year up to three years between January 1999 and December 2003. When making a cross-tabulation of country versus registration year (create a field for registration year from REGDATE), we see that:

Year of registration	Country				Total
	Moldova	Mongolia	Uganda	Zimbabwe	
1999	0	0	17300	0	17300
2000	0	0	18662	0	18662
2001	0	0	18088	1213	19301
2002	0	149	0	29307	29456
2003	17725	22406	0	3958	44089
Total	17725	22555	54050	34478	128808

It is thus best to sequentially number all the 60 months from beginning to the end, even if at the end we will require only the three years covered by Uganda.

Select for diagnostic examinations, country, and range of months.

Save the dataset with a new name.

### 2) Count all smears, all positive, all low-positive smears, and all scanty positive smears

Create four new variables that count for each record respectively all smears, all positive smears, all low-positive smears and all scanty positive smears

### 3) Aggregate the months and sum up the smears in question and save them to four different files

The element to be aggregated is the month and for each month one must have the relevant smears (all, all positive, all scanty positive, all low-positive).

### 4) Make the SPC charts

The appropriate chart type for this binomial outcome is a PChart which has the format:

```
pchart numerator denominator time
```

### Task

- Produce five PCharts to display the proportion of 1) positive smears among all smears, 2) low-positive smears among all smears, 3) scanty positive smears among all smears, 4) low-positive smears among all positive smears, and 5) scanty positive smears among all positive smears.

## Solution to Exercise 2: A statistical process control chart

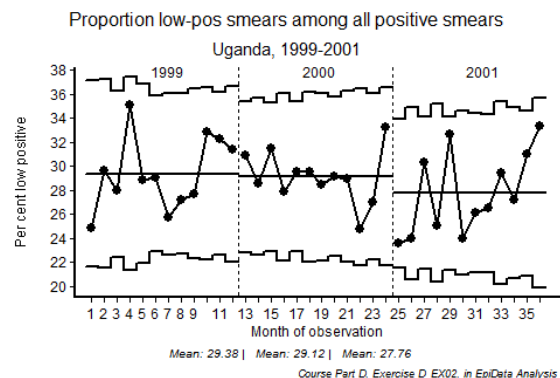
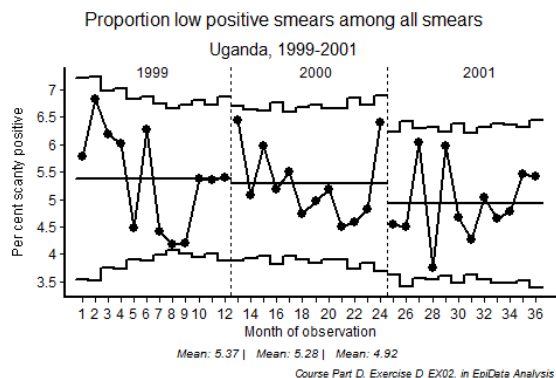
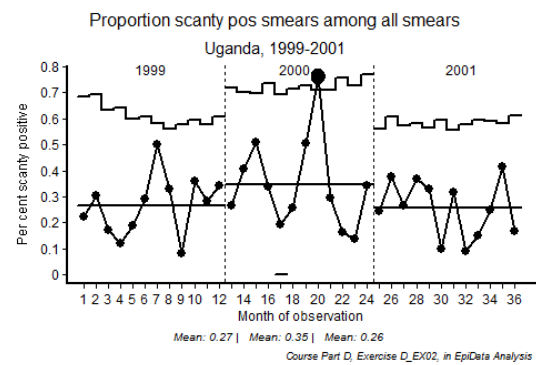
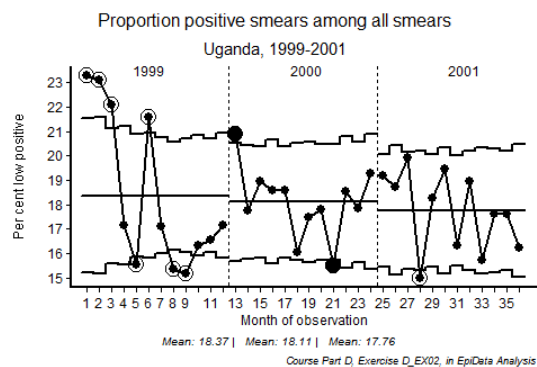
### Key points:

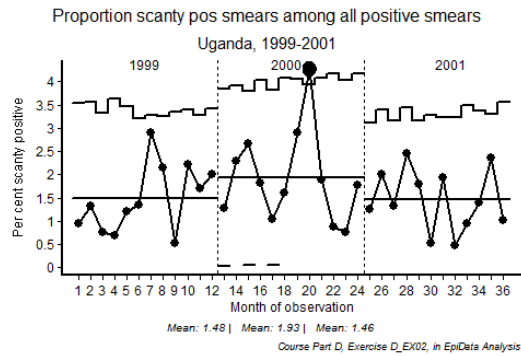
- You must determine on how to aggregate data to obtain the numerator and denominator and, where necessary, the time components over the observation period
- For a binomial outcome, a PChart is the appropriate SPC chart

### Task

- Produce five PCharts to display the proportion of 1) positive smears among all smears, 2) low-positive smears among all smears, 3) scanty positive smears among all smears, 4) low-positive smears among all positive smears, and 5) scanty positive smears among all positive smears.*

These are the five PCharts:





This is the program d\_ex02.pgm that produced them:

```
* Part D, Exercise 2

* 1) Determine the proportion of positive
*   smears among all smears
* 2) Determine the proportion of low-postive
*   smears among all positive smears
* 3) Determine the proportion of scanty positive
*   smears among all positive smears
* Definition of positive: any quantified positive
*   or any scanty (quantified scanty or unquantified scanty)
* Definition low-positive: any smear which is 1+ positive or scanty positive

* Written by:   Hans L Rieder
* First version: 26 Jun 2011
* Last revision: 29 Apr 2013

cls
close
logclose

*****
* Procedural steps
* 1) Make basic dataset
* 2) Start selection process
* 3) Aggregate data
* 4) Make SPC charts

*****
* 1) Prepare basic dataset

cls
close

read "mmuz.rec"

define regyear ####
regyear=year(regdate)
label regyear "Registration year"

cls
tables country regyear

gen i mmseq=0
if year(regdate)=1999 then mmseq=month(regdate)
if year(regdate)=2000 then mmseq=month(regdate)+12
if year(regdate)=2001 then mmseq=month(regdate)+24
if year(regdate)=2002 then mmseq=month(regdate)+36
if year(regdate)=2003 then mmseq=month(regdate)+48
label mmseq "Sequential month"

select reason=0
select country=3
select mmseq>0 and mmseq<37
* The selection above is not necessary for Uganda alone
* as there are no examinees in 2003

keep result1 result2 result3 regyear mmseq
savedata "temp_01.rec" /replace
```



```

*****
* 2) Count all smears, all positive, all scanty positive,
* all low positive smears

cls
close
read "temp_01.rec"

cls
* Count all smears
* Note: non-sensical sequences removed, thus simply:
gen i allsmears=1
if result2<>9 then allsmears=2
if result3<>9 then allsmears=3
label allsmears "Number of smears"

cls
* Count all positive quantified smears
* (include scanty not quantified)
gen i allpos1=0
if result1>0 and result1<4 then allpos1=1
if result1=5 then allpos1=1
gen i allpos2=0
if result2>0 and result2<4 then allpos2=1
if result2=5 then allpos2=1
gen i allpos3=0
if result3>0 and result3<4 then allpos3=1
if result3=5 then allpos3=1
gen i allpos=allpos1+allpos2+allpos3
label allpos "Number of positive smears"

cls
* Count all scanty smears
* (include scanty not quantified)
gen i scantpos1=0
* (include scanty not quantified)
if result1>0 and result1<1 then scantpos1=1
if result1=5 then scantpos1=1
gen i scantpos2=0
if result2>0 and result2<1 then scantpos2=1
if result2=5 then scantpos2=1
gen i scantpos3=0
if result3>0 and result3<1 then scantpos3=1
if result3=5 then scantpos3=1
gen i scantypos=scantpos1+scantpos2+scantpos3
label scantypos "Number of scanty positive smears"

cls
* Count all low positive smears
* (include scanty not quantified)
gen i lowpos1=0
if result1>0 and result1<2 then lowpos1=1
if result1=5 then lowpos1=1
gen i lowpos2=0
if result2>0 and result2<2 then lowpos2=1
if result2=5 then lowpos2=1
gen i lowpos3=0
if result3>0 and result3<2 then lowpos3=1
if result3=5 then lowpos3=1
gen i lowpos=lowpos1+lowpos2+lowpos3
label lowpos "Number of low positive smears"

keep regyear mmseq allsmears allpos scantypos lowpos
savedata "temp_02.rec" /replace
*****
* 3) Aggregate data

cls
close
read "temp_02.rec"

agg mmseq /sum=allsmears /sum=allpos /sum=scantypos /sum=lowpos /close
drop n nallsm1 nallpos nscantpos nlowpos
rename sumallsm1 to allsmears
rename sumallpos to allpos
rename sumscant1 to scantpos
rename sumlowpos to lowpos

```

```

drop n nallsm1 nallpos nscant1 nlowpos
savedata "temp_03.rec" /replace
*****
* 4) Make SPC charts

set option spc= /size=500 /sizey=350
set graph font size=9

cls
close
logclose

read "temp_03.rec"

set echo=off
pchart allpos allsmears mmseq /xtext="Month of observation" /bw \
    /ytext="Per cent low positive" \
    /ti="Proportion positive smears among all smears" \
    /sub="Uganda, 1999-2001" \
    /fn="Course Part D, Exercise D_EX02, in EpiData Analysis" \
    /b=12 /b=24 \
    /xlined=12.5 /xlined=24.5 \
    /t1 \
    /text="120,60,1999,0" \
    /text="260,60,2000,0" \
    /text="400,60,2001,0"

pchart scantpos allsmears mmseq /xtext="Month of observation" /bw \
    /ytext="Per cent scanty positive" \
    /ti="Proportion scanty pos smears among all smears" \
    /sub="Uganda, 1999-2001" \
    /fn="Course Part D, Exercise D_EX02, in EpiData Analysis" \
    /b=12 /b=24 \
    /xlined=12.5 /xlined=24.5 \
    /t1 \
    /text="120,60,1999,0" \
    /text="250,60,2000,0" \
    /text="400,60,2001,0"

pchart lowpos allsmears mmseq /xtext="Month of observation" /bw \
    /ytext="Per cent scanty positive" \
    /ti="Proportion low positive smears among all smears" \
    /sub="Uganda, 1999-2001" \
    /fn="Course Part D, Exercise D_EX02, in EpiData Analysis" \
    /b=12 /b=24 \
    /xlined=12.5 /xlined=24.5 \
    /t1 \
    /text="120,60,1999,0" \
    /text="260,60,2000,0" \
    /text="400,60,2001,0"

pchart lowpos allpos mmseq /xtext="Month of observation" /bw \
    /ytext="Per cent low positive" \
    /ti="Proportion low-pos smears among all positive smears" \
    /sub="Uganda, 1999-2001" \
    /fn="Course Part D, Exercise D_EX02, in EpiData Analysis" \
    /b=12 /b=24 \
    /xlined=12.5 /xlined=24.5 \
    /t1 \
    /text="130,60,1999,0" \
    /text="260,60,2000,0" \
    /text="400,60,2001,0"

pchart scantpos allpos mmseq /xtext="Month of observation" /bw \
    /ytext="Per cent scanty positive" \
    /ti="Proportion scanty pos smears among all positive smears" \
    /sub="Uganda, 1999-2001" \
    /fn="Course Part D, Exercise D_EX02, in EpiData Analysis" \
    /b=12 /b=24 \
    /xlined=12.5 /xlined=24.5 \
    /t1 \
    /text="120,60,1999,0" \
    /text="250,60,2000,0" \
    /text="400,60,2001,0"

set echo=on

```

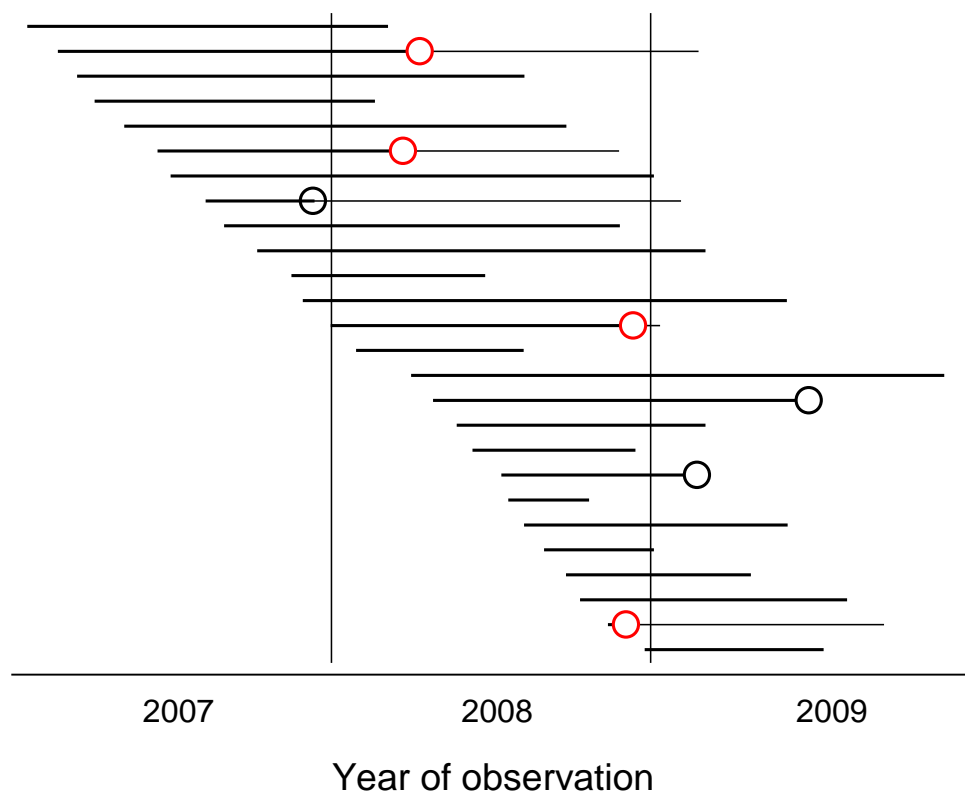
### Exercise 3: Survival analysis

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

- Understand the indications for survival analysis
- Be able to do a simple survival analysis in EpiData Analysis

#### An example of a survival analysis using the Kaplan-Meier method

We observe people over time, starting in 2007 into 2009 and note whether or not they develop an event (whatever it may be) during the observation time as shown for 26 individuals in the following graph:



We can think of this setting like of an institution, such as a prison, where inmates enter the prison and are discharged at some time. During incarceration some may develop tuberculosis. If our interest is focused on the year 2008, we have two measures of the magnitude of the problem.

We might calculate the incidence rate in the year 2008. The numerator is 4 cases. The denominator is person-time of observation. As the following table shows, we know the date of entry and exit from the institution, and the date the event occurs among those who had and event:

ID	Entry date	Exit date	Event	Event date	Obs start	Obs end	Obs days
A	20-01-2007	04-03-2008	No		01-01-2008	04-03-2008	63
B	24-02-2007	23-02-2009	Yes	11-04-2008	01-01-2008	11-04-2008	101
C	18-03-2007	07-08-2008	No		01-01-2008	07-08-2008	219

D	07-04-2007	18-02-2008	No		01-01-2008	18-02-2008	48
E	11-05-2007	24-09-2008	No		01-01-2008	24-09-2008	267
F	18-06-2007	24-11-2008	Yes	23-03-2008	--	--	
G	03-07-2007	02-01-2009	No		01-01-2008	31-12-2008	365
H	12-08-2007	03-02-2009	Yes	11-12-2007	01-01-2008	31-12-2008	365
I	02-09-2007	24-11-2008	No		01-01-2008	24-11-2008	328
J	10-10-2007	02-03-2009	No		01-01-2008	31-12-2008	365
K	18-11-2007	23-06-2008	No		01-01-2008	23-06-2008	174
L	01-12-2007	03-06-2009	No		01-01-2008	31-12-2008	365
M	02-01-2008	10-01-2009	Yes	11-12-2008	02-01-2008	11-12-2008	344
N	31-01-2008	06-08-2008	No		31-01-2008	06-08-2008	188
O	03-04-2008	30-11-2009	No		03-04-2008	31-12-2008	272
P	28-04-2008	05-07-2009	Yes	30-06-2009	28-04-2008	31-12-2008	247
Q	25-05-2008	02-03-2009	No		25-05-2008	31-12-2008	220
R	12-06-2008	12-12-2008	No		12-06-2008	12-12-2008	183
S	15-07-2008	02-03-2009	Yes	22-02-2009	15-07-2008	31-12-2008	169
T	23-07-2008	20-10-2008	No		23-07-2008	20-10-2008	89
U	10-08-2008	04-06-2009	No		10-08-2008	31-12-2008	143
V	02-09-2008	02-01-2009	No		02-09-2008	31-12-2008	120
W	27-09-2008	23-04-2009	No		27-09-2008	31-12-2008	95
X	13-10-2008	11-08-2009	No		13-10-2008	31-12-2008	79
Y	14-11-2008	23-09-2009	Yes	03-12-2008	14-11-2008	03-12-2008	19
Z	26-12-2008	15-07-2009	No		26-12-2008	31-12-2008	5
					Days		4833
					Cases		4
					Cases/1000 person-days		0.828
					Cases/100 person-years		30.2

Each person contributes person-time of observation, starting earliest from the beginning of the year 2008 or later if entry into the system was later. Person-time is contributed until exit from the institution or up to the point of the event if either happened in the year 2008. If the event happens later, observation time ends at the right-censoring point of 31 December 2008. Individual F developed the event in 2007 and although still in the system in 2008 does not contribute any person-time of observation in 2008. Individuals P and S developed the event only in 2009 and are thus not counted in 2008 and they contribute to person-time of observation through the end of the year 2008. Thus summed up, the 25 of the 26 inmates contributed 4,833 days of observation time which gives a case rate of 0.8 per 1,000 person-days of observation, or annualized, 30.2 cases per 100 observation years.

## Survival analysis

The second way to look at the size of the problem is to ask what the probability for an individual is to “survive” the year 2008 without developing the event.

We exclude the person who had the event already in 2007 and sort the persons by the time of event or censoring (discharge or latest end of 2008). We note the number at risk of the

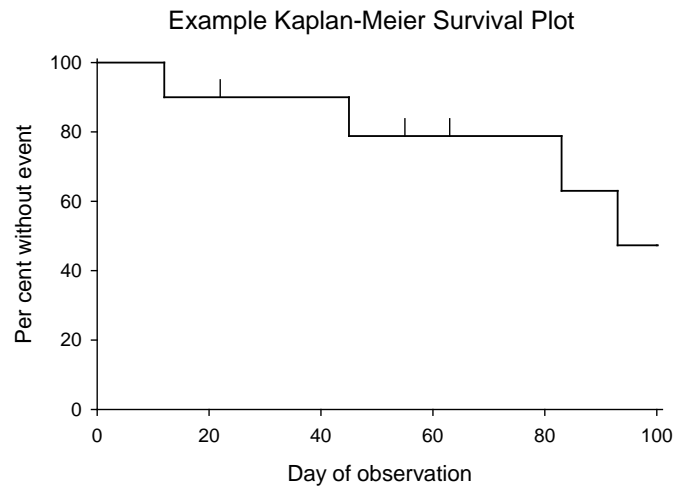
beginning of the interval, then the number who got censored, then how many were left after censoring, and finally how many get an event during the interval:

At risk at beginning	Censored	At risk at end	Event at end	Proportion surviving	Survival
25	1	24	0		
24	0	24	1	23/24	0.958
23	1	22	0		0.958
22	1	21	0		0.958
21	1	20	0		0.958
20	0	20	1	(19/20)*0.958	0.910
19	1	18	0		0.910
18	1	17	0		0.910
17	0	17	1	(16/17)*0.910	0.857
16	1	15	0		0.857
15	1	14	0		0.857
14	1	13	0		0.857
13	1	12	0		0.857
12	1	11	0		0.857
11	1	10	0		0.857
10	1	9	0		0.857
9	1	8	0		0.857
8	1	7	0		0.857
7	1	6	0		0.857
6	1	5	0		0.857
5	1	4	0		0.857
4	0	4	1	(3/4)*0.857	0.643
3	3	0	0		0.643

At each point where an event (not censoring) happens, we calculate the survival probability by dividing the number “surviving” after the event by the number at risk at the beginning of the interval when the event occurred.

You may note that censoring during an interval is assumed not to affect survival probability during that interval, but censored individuals are also subtracted from those at risk for the next interval.<sup>1-3</sup> This assumption is a simplification because censoring may, under certain circumstances, indeed be affecting the survival probability during the remaining interval. In any case, however, taking both the occurrence of the event and censoring into account for each interval following an event is a much more appropriate way to calculate survival than the proportion with an event among all who entered the cohort or by removing those censored from the cohort.

Graphically, we summarize the Kaplan-Meier survival probability as a step graph, sometimes showing the points when an individual was censored:



1. Kaplan E L, Meier P. Nonparametric estimation from incomplete observations. J Am Stat Ass 1958;53:457-81.
2. Bland J M, Altman D G. Survival probabilities (the Kaplan-Meier method). (Statistics notes). BMJ 1998;317:1572.
3. Fink S A, Brown R S, Jr. Survival analysis. Gastroenterol Hepatol 2006;2:380-3.

### Survival analysis in EpiData Analysis

The command for a Kaplan-Meier survival analysis in EpiData Analysis is:

```
lifetable outcome interval
```

or

```
lifetable outcome startdate enddate
```

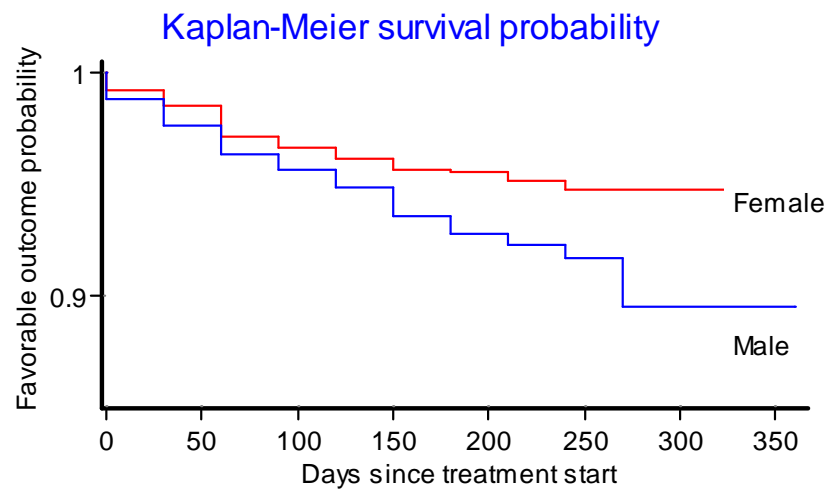
There are a multitude of options available both lifetable-specific options and general graph options. Use the Help file to test out various options until you have the survival plot that suits your needs.

### Tasks

*The purpose of the exercise is to demonstrate quantitatively the probability of remaining without an event.*

*We use to this end a real dataset from a tuberculosis program, although we have removed any identifier and most variables, and retained to simplify your work only records of patients who have an exact date of treatment start and an exact date of treatment end. The dataset is provided as a supplementary file **d\_ex03\_required.rec**.*

- 1) *Define a binomial outcome, where “favorable” is cured or treatment completed and all other outcomes are “unfavorable”*
- 2) *Do the lifetable analysis only for new sputum smear-positive cases and show the survival probability stratified by sex*



*EpiData course: Exercise D\_EX03*

## Solution to Exercise 3: Survival analysis

### Key points:

- In this simplified example, there was no censoring, people either had the event or they didn't
- Survival analysis requires two variables, the time when the event occurred or the endpoint of observation, and whether there was an event or not

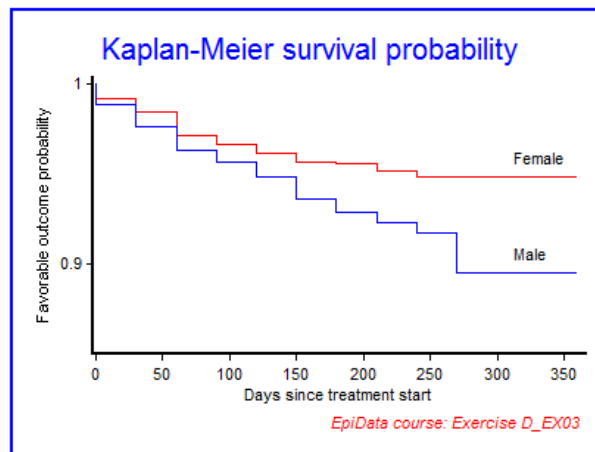
### Tasks

*The purpose of the exercise is to demonstrate quantitatively the probability of remaining without an event.*

*We use to this end a real dataset from a tuberculosis program, although we have removed any identifier and most variables, and retained to simplify your work only records of patients who have an exact date of treatment start and an exact date of treatment end. The dataset is provided as a supplementary file `d_ex03.rec_required`.*

- 1) *Define a binomial outcome, where “favorable” is cured or treatment completed and all other outcomes are “unfavorable”*
- 2) *Do the lifetable analysis only for new sputum smear-positive cases and show the survival probability stratified by sex*

The graphic output produced by the program `D_EX03.PGM`:



The program `D_EX03.PGM` to produce the above graphic output:

```
* Part D, Exercise 3
* Survival analysis
* Tuberculosis case register:
*   Treatment outcome

* Written by:    Hans L Rieder
* First version: 30 Oct 2010
* Last revision: 29 Apr 2013
```



```

cls
close
logclose

*****
* Create dataset

cls
close
read "d_ex03_required.rec"

define advout #
    advout=1 // adverse outcome
if outcome<=2 then advout=0 // favorable outcome
label advout "Treatment outcome"
labelvalue advout /0="Favorable"
labelvalue advout /1="Adverse"

gen i case=0
if sm00>0 and sm00<9 then case=1
select case=1
select sex<>9
select category=1

keep sex advout interval
savedata "temp_01.rec" /replace

*****
* Analysis

cls
close

read "temp_01.rec"

cls
set option graph=/sizex=600 /sizey=350
set graph font size=9
lifetable advout interval \
    /by=sex \
    /i=b30 /adj \
    /noc1 \
    /ymin=0.85 \
    /ymax=1 \
    /t \
    /ti="Kaplan-Meier survival probability" \
    /fn="EpiData course: Exercise D_EX03" \
    /xtext="Days since treatment start" \
    /ytext="Favorable outcome probability" \
    /text="335,90,Female,0" \
    /text="335,155,Male,0"

```

## Exercise 4: Creating a menu for standard reports

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

- a. Write an HTML-based interface for a menu
- b. Writing programs with interactive prompting

This exercise involves working with HTML to make a user-friendly menu-based interface allowing data entry and running EpiData Analysis programs on clicking which produce standard reports with interactive prompting for choices.

The end product interface will appear as follows:



We will structure the exercise as follows:

### Preparatory work

- Installing EpiData Entry and EpiData Analysis
- Delete an obsolete sub-folder and create new sub-folders
- Unzip the required files from the course website into the relevant sub-folder

Background how EpiData Analysis starts and how to shape its looks when opening

### Making the interface in HTML

- Using an HTML editor to make the skeleton of the interface
- Editing the HTML file

### Making the EpiData Analysis programs

- The program to produce the quarterly report on case finding
  - Make the basic dataset
  - Make the interactive selection process
  - Make the charts side by side

The program to produce the quarterly report on treatment outcome

- Make the basic dataset
- Make the interactive selection process

## Preparatory work

### Installing EpiData Entry and EpiData Analysis

First make sure your “Normal EpiData Analysis” is updated to the newest version. The recommended location for both EpiData Entry and EpiData Analysis is C:\EPIDATA.

In addition, we will install EpiData Entry and EpiData Analysis into a separate folder (make it user-defined to keep control over what’s going to happen). You could also just copy the files and sub-folders in your C:\EPIDATA to C:\EPIDATA\_REPORT. We simulate here what happens if you start from scratch. When prompted for the path, put it into:

```
c:\epidata_report
```

You will get in this folder three sub-folders and 25 files:

```
languages\  
samples\  
temp\  
English.ea.lang.txt  
Epdintro.pdf  
EPIDATA.CNT  
EpiData.exe  
EPIDATA.HLP  
epidata.ini  
EpiData.lbl  
epidatastat.10  
epidatastat.12  
epidatastat.15  
epidatastat.20  
EpiDataStat.exe  
epidatastat.ini  
epiout.css  
epiout_b.css  
epiout_w.css  
epiprint.css  
Francais.ea.lang.txt  
license.txt  
preventdouble.ea  
readme.rtf  
unins000.dat  
unins000.exe  
unins001.dat  
unins001.exe
```

### Delete unnecessary sub-folders and create new sub-folders

The samples sub-folder is superfluous for this exercise and can be deleted. Instead create **four new** sub-folders, so that you have a total of **six** sub-folders:

```
images\  
languages\  
originals  
pgm\  
required\  
temp\  

```

```
English.ea.lang.txt
```

```
...
```

In the sub-folder “languages” delete:

~~en\~~  
en\  
~~fr\~~  
images\

In the sub-folder “temp” you have (and can delete both):

~~docs\~~  
~~examples\~~

Of the 25 files in the root you can delete all that are marked below:

English.ea.lang.txt  
~~Epidintro.pdf~~  
EPIDATA.CNT  
EpiData.exe  
EPIDATA.HLP  
epidata.ini  
EpiData.lbl  
epidatastat.10  
epidatastat.12  
epidatastat.15  
epidatastat.20  
EpiDataStat.exe  
epidatastat.ini  
epiout.css  
epiout\_b.css  
epiout\_w.css  
epiprint.css  
Francais.ea.lang.txt  
license.txt  
preventdouble.ea  
readme.rtf  
~~unins000.dat~~  
~~unins000.exe~~  
~~unins001.dat~~  
~~unins001.exe~~

It is of course not necessary to delete all these files but it reduces package size (notably the uninstall files).

### **Unzip the required files from the course website into the relevant sub-folder**

The course website contains a zip file with 23 required files:

epidata.ini  
epidata.png  
epidata\_wikiL.png  
epidatastat.png  
eraser.png  
next.gif  
quit.jpg  
sample\_2004.chk  
sample\_2004.eix  
sample\_2004.qes  
sample\_2004.rec

```
sample_2005.chk
sample_2005.eix
sample_2005.qes
sample_2005.rec
sample_2006.chk
sample_2006.qes
sample_2006.rec
side_by_side_graphs.html
start_template.htm
uganda.chk
uganda.rec
union.jpg
```

Unzip all these 23 files into the EPIDATA\_REPORT\REQUIRED sub-folder. Then move and over-write if necessary as follows:

Move the 7 image files from the EPIDATA\_REPORT\REQUIRED sub-folder to the EPIDATA\_REPORT\IMAGES sub-folder:

```
epidata.png
epidata_wikiL.png
epidatastat.png
eraser.png
next.gif
quit.jpg
union.jpg
```

Move the 14 EpiData files (plus an \*.HTML file) from the EPIDATA\_REPORT\REQUIRED sub-folder to the EPIDATA\_REPORT\ORIGINALS sub-folder:

```
sample_2004.chk
sample_2004.eix
sample_2004.qes
sample_2004.rec
sample_2005.chk
sample_2005.eix
sample_2005.qes
sample_2005.rec
sample_2006.chk
sample_2006.qes
sample_2006.rec
side_by_side_graphs.html
uganda.chk
uganda.rec
```

Move from the EPIDATA\_REPORT\REQUIRED sub-folder to the EPIDATA\_REPORT\LANGUAGES\EN sub-folder the file:

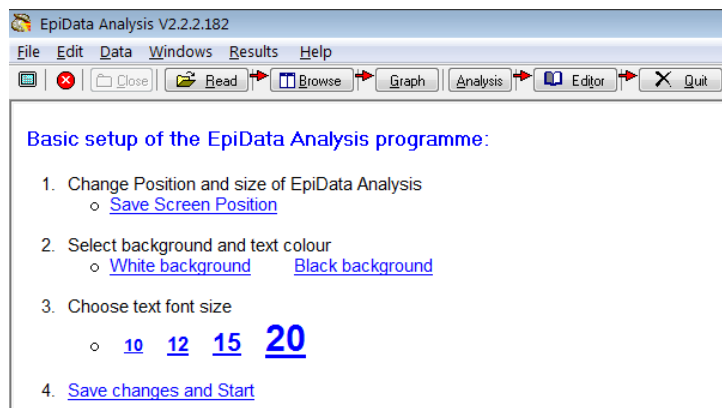
```
start_template.htm
```

Finally, move (and overwrite the existing) the:

```
epidata.ini
```

file to the root of the C:\EPIDATA\_REPORT. And with that all “required” files should have been moved. Your “REQUIRED” folder should be empty and you can delete it.

Open EpiData Analysis by double-clicking its EpiDataStat.exe executable file and adjust and save font sizes when you see:



so that you end up with an empty screen, save the Windows position and exit EpiData Analysis.

### Background how EpiData Analysis starts and how to shape its looks when opening

In any software you may use, there is an executable file that starts the process of opening the program and displaying the standard interface. In EpiData Analysis, this file is in the root of the folder in which you installed the program and its name is:

`EpiDataStat.exe`

This file contains all the essential code to direct EpiData Analysis to do what it is expected to do. Publishing this code will make EpiData Analysis what we call “open-source” software. While the designers and programmers of EpiData software are working on preparing this source code with sufficiently detailed documentation to ultimately make it open-source, the time is not yet quite mature to do so because the documentation must be un-ambiguous and clear for any other developer to derive usefulness for further development from it. As we are not software developer ourselves, we do not have any need to know the source code, the only thing we need to know is some very basic things that this executable file does, and how we can override certain things to meet our needs.

Among other files, the executable file looks first for an HTML file that is located in the `EPIDATA_REPORT\LANGUAGES\EN` sub-folder:

`start.htm`

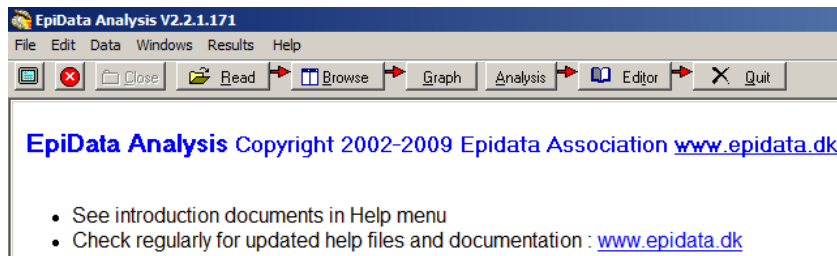
If we double-click this file it opens in our default browser and this is the display we get:

**EpiData Analysis Copyright 2002-2009 Epidata Association [www.epidata.dk](http://www.epidata.dk)**

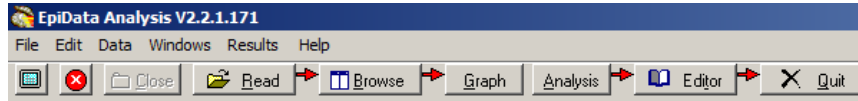
- See introduction documents in Help menu
- Check regularly for updated help files and documentation : [www.epidata.dk](http://www.epidata.dk)

We can change this visualized part to our liking in the `start.htm` file which will be one of our tasks.

When we open EpiData Analysis, we see in addition the version display, the menu bar, and the process bar:

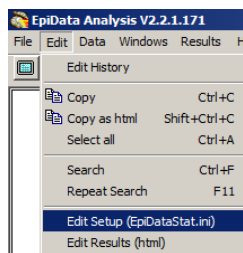


The upper component:



is part of the default defined in the executable file. It is, however, possible to suppress part or all of it by modifying the `EpiDataStat.ini` file.

With the opening of EpiData Analysis, a small program called `EpiDataStat.ini` is run. You can access it from the Edit menu:



As we have exited EpiData Analysis, we will now look for this `EpiDataStat.ini` file in the root of our folder and open it in our preferred text editor. Its initial default script is:

```
* EpiData Analysis default settings file
* Edit the next lines to change size or font
set echo=off

* Viewer font and size: (plus editor and help windows)
set browser font size =12
set graph font size =12
set viewer font size =12
set window font size =12
set editor font size =12
set viewer font name ="Verdana,Courier"

* uncomment next two lines to display Chinese Characters
*SET viewer font charset = "gb2312"
*set viewer font name="Arial Unicode MS"

*****
* Set options defined during installation:
* To see other set: issue "set" command or look in help file (F1)
*****
set display variables=OFF
set display databrowser=OFF
set output open=OFF
* Default folder defined as:
cd C:\EpiData_report\temp
```

```
set output folder="C:\EpiData_report\temp"
set language=english
set echo=ON
```

If we strip it of all comments, what remains is:

```
set echo=off

set browser font size =12
set graph font size =12
set viewer font size =12
set window font size =12
set editor font size =12
set viewer font name ="Verdana,Courier"

set display variables=OFF
set display databrowser=OFF
set output open=OFF
cd C:\EpiData_report\temp
set output folder="C:\EpiData_report\temp"
set language=english

set echo=ON
```

a series of SET commands that tells EpiData Analysis some defaults that are operative until we change. We asked you before to write over this file and that replacing file differs from the above (comments that remain are not shown) is shown in red font:

```
1 set echo=off
2 cd temp
3 set display mainmenu      =off
4 set display command prompt=off
5 set display worktoolbar   =off
6 set viewer font size =12
7 set window font size =12
8 set editor font size =12
9 set viewer font name ="Verdana,Courier"
10 set display variables=OFF
11 set display databrowser=OFF
12 set output open=OFF
13 set output folder="..\temp"
14 set language=english
15 set echo=on
16 set start page    ="..\languages\en\start_tb.htm"
```

If we rearrange a bit to put similar things together, we may summarize as:

```
set echo=off

cd temp

set start page    ="..\languages\en\start_tb.htm"
set output folder="..\temp"

set viewer font size =12
set window font size =12
set editor font size =12
set viewer font name ="Verdana,Courier"
set language=english
```



```
set display variables      =off
set display databrowser   =off
set display mainmenu      =off
set display worktoolbar   =off
set display command prompt=off
set output open           =off
```

```
set echo=on
```

We turn the echo off and on respectively at the beginning and at the end, so that we don't notice that it is run.

We then direct EpiData Analysis to go to the (above created) sub-folder EPIDATA\_REPORT\TEMP. EpiData Analysis will be in this place when you start with an analysis and if you need to access data files or a program file or any other file, you will need to tell it where these are located *relative to this location*.

## Relative locations

This might be an opportune time to introduce the concept of relative and absolute path. The path:

```
C:\epidata_report\languages\en
```

is an **absolute** path. It defines the drive ("C:\") and where within this drive one finds the folder and its named sub-folders. If we were to write this into the EpiDataStat.ini file, it would be alright as long as both is true, the drive and the path. If we would give the "package" with the "epidata\_report" to somebody else it would therefore only work if that person were to copy it also into the root of his or her PC and if that main drive actually had the name "c:\". It wouldn't work from a USB drive for instance. However, if we would give the "package" to a colleague and inside the EpiDataStat.ini file all file locations were given relative to the "container" "epidata\_report", then it would work from any location on the PC or indeed from an external drive with another drive designation than "c:\". We could go one step further, and say that we don't even want to name "epidata\_report" as "epidata\_report", so that a user is entirely free to give any name to this container and it would still be working.

What we need is a folder separator, and this is the backslash ("\") and the replacement indicator for the parent folder (directory), which is the double period (".."). If we thus write:

```
"..\temp\sometext.txt"
```

we refer to a file "sometext.txt" which is located (absolute path) in a folder temp which in itself is located in another folder that is not named. If we are in that other folder (whatever its name might be) then we refer relatively to it here and have no need to name it. This is precisely what we are doing here with this:

```
set start page    ="..\languages\en\start_tb.htm"
```

We are in EpiData Analysis which is in the folder epidata\_report. In this folder we have a sub-folder "languages", directly one level below the "epidata\_report" and we can thus replace the parent directory "epidata\_report" with ".." and insert the "\" as the folder separator followed by the designation of the "languages" sub-folder. This approach allows us to give the parent directory any name we wish: all that must be guaranteed

is that the sub-folder has a fixed position relative to the parent folder. Let's evaluate how we can move around within our "container" `epidata_report`. When we start EpiData Analysis from the `EpiDataStat.exe` file, we have our 6 folders at the same level, one of them being the `temp` folder. We get first into this folder with:

```
cd temp
```

To get back from within this folder to the root (`epidata_report`), we would type:

```
cd ..
```

indicating that with the two periods that we want to go to the parent folder. Being in the parent folder now, we wish to go to the sub-folder `en` of the `languages` folder:

```
cd languages\en
```

Being now in the `en` sub-folder and wishing to go to the sub-folder `originals` of the "container", we can do it in 3 steps:

```
cd ..  
cd ..  
cd originals
```

The first line gets us to the parent of `en`, i.e. to `languages`, the second gets us to the parent of `languages`, i.e. `epidata_report`, and the third line directs us to the sub-folder `originals` of `epidata_report`. Simpler, we can write this in a single line:

```
cd ../../originals
```

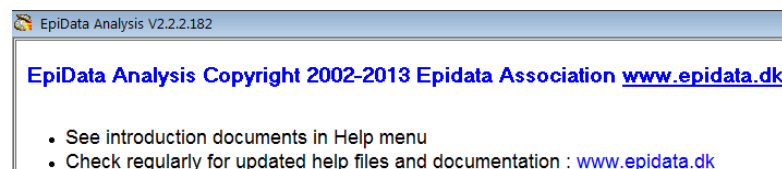
We have entered a path to a non-existing file:

```
set start page = "../../languages/en/start_tb.htm"
```

While it is now clear why we use this designation for the path, we also note that the file "`start_tb.htm`" does not exist in that folder. In the folder `en` we currently have:

```
start.htm  
start_template.htm  
start0.htm  
startfont.htm
```

It is then in a next step that we will create the "`start_tb.htm`" file. EpiData Analysis will open despite this error, simply by using the default file "`start.htm`" which is in this folder and give us:



At the bottom, you see don't see the command line anymore, and only the sub-folder in which EpiData Analysis is after executing the first four lines:



The default style sheet that EpiData Analysis uses is the output .css file, a cascading style sheet. You could make the style in the start.htm file, but the recommendation of the World Wide Web Consortium (W3C) is to refer in the main file to another specific (\* .CSS) file that defines the style. This recommendation is for good reasons: you can always change the style sheet without changing the main HTML file.

You can make your own style sheets but this will require learning a bit more on how to make one, and this not subject of this exercise.

EpiData Analysis writes log files and other stuff that are useful to review when something goes wrong. These files are not usually used when all goes as expected and to get them out of the way, the command line:

```
set output folder="..\temp"
```

redirects the output to be stored in the sub-folder TEMP.

With this brief and rather simplified introduction of how EpiData Analysis starts to work, you should now have an understanding on what the EpiDataStat.ini file does and how you can always access it and change it on the fly. In your “regular” EpiData Analysis program, it will often prove useful to direct it with this file to a specific project folder on which you are currently working with even a series of CD commands as long as the folders exist and you know that the last CD command in a sequence overwrites all previous ones.

After we are now done with the role of the EpiDataStat.ini file, we have to deal with the HTML file start\_tb.htm, which gets us into learning some basics about the HTML language.

## Making the interface in HTML

### Using an HTML editor to make the skeleton of the interface

HTML is the language of the Internet which is interpreted by a browser such as the proprietary Internet Explorer™ or the open-source and free Firefox® to make it visually comprehensible and appealing for the user. A simple text that looks like:

This is a simple text to show the browser  
interpretation of HTML text and the actual  
underlying HTML.



We add above an icon for display.

has the following underlying HTML code in the browser:

```
1 <!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01//EN" "http://www.w3.org/TR/html4/strict.dtd">
2
3 <html>
4
5   <head>
6     <meta content="text/html; charset=ISO-8859-1" http-equiv="content-type"><title>temp</title>
7   </head>
8
9   <body>
10    <big><span style="font-family: Arial;">This is a simple text to show the browser</span>
11    <br style="font-family: Arial;"><span style="font-family: Arial;">interpretation of HTML text and the actual</span>
12    <br style="font-family: Arial;"><span style="font-family: Arial;">underlying HTML.</span></big><br>
13    <br>
14    <big><span style="font-family: Arial;">We add above an icon for display.</span></big>
15  </body>
16
17 </html>
```

This is complex at first look, but when we look at it carefully then we realize that it is a highly logical language and sequence of instructions to the browser.

In Line 1, information is provided that the language conforms with W3C standards and the version of HTML it is using and that you can find this confirmed at the W3C website.

Every component in an HTML document has the principle to define the beginning and the end of the component and the system is the same for all. In the above example we have:

Begin	End
<HTML>	</HTML>
<head>	</head>
<body>	</body>

The structure indicates that everything between the opening and ending HTML tags is in fact HTML. Embedded are two parts, the head and the body, each indicating where it starts and where it ends. Every HTML page is build around these key parts, and within these you may have other sub-components imbedded that follow the same principles such as here within the body:

Begin	End
<span>	</span>

For the time being, we will leave it at that but will come back later to these principles when we work specifically on the `start_tb.htm` file.

Because of the complexity for the beginner, a multitude of software has been developed allowing the user to write normally as in a word processor to see what one actually wants to get. The software translates it into HTML and the page becomes interpretable by the browser. The advantages of such software are obvious but the downside is that it is often very expensive (hundreds of Euros perhaps), and not all is adhering strictly to W3C standards which will make it difficult for some browsers that require strict adherence to interpret the language properly.

You may have heard about the Mozilla Foundation which produces free and excellent open-source software, such as the browser Firefox, the email client Thunderbird, the calendar Sunbird and yes, the HTML editor Nvu. Nvu is still in its infancy and has some problems, some of which have been resolved with the HTML editor KompoZer which you find on this course web in the software section and that we will be using.

You do not need to install KompoZer, it is just a zip file. Unzip it into the root of your hard drive and you get a folder:

KompoZer 0.7.10\

You may make a shortcut to its executable file:

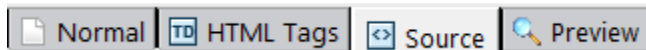
KompoZer.exe

to your desktop and if desired to the quick launch bar to have it accessible at your fingertip with one click away or simply look for it and double-click the KompoZer.exe file.

Access KompoZer and find the `start_template.htm` file in the `EPIDATA_REPORT\LANGUAGES\EN` sub-folder of the project. It is an empty page with a title:



That it is not quite empty becomes clear if you tick at the bottom to “Source”:



In fact, we have taken the normal `start.htm` file that comes with the installation of EpiData Analysis, have stripped it down and added a few essentials to prepare it for an interactive menu (template courtesy, Jens M Lauritsen, April 2008) and saved it under this `start_template.htm` file name.

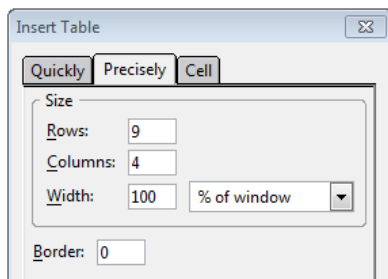
The first thing is to save it as:

`start_tb.htm`

Remember that we direct the `EpiDataStat.ini` file to go to this file when initiating:

```
set start page="..\languages\en\start_tb.htm"
```

You will be working in the “Normal” tab and the first thing we do is to insert a table with 4 columns and 9 rows (you can count them in the screenshot of the final interface shown at the beginning). Choose “Precisely”:

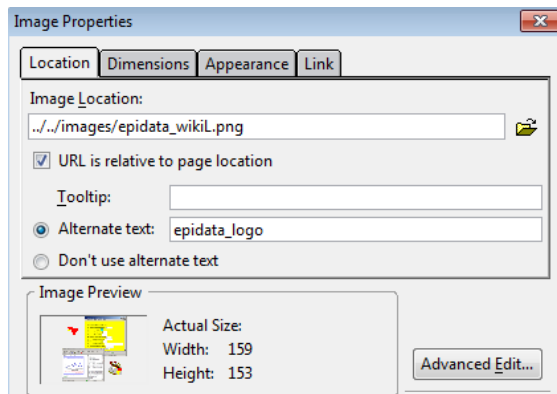


and set the Border to “0”. The default for “Border” is 1: a line around each cell is displayed in the browser. Setting it to “0” allows entering the information into cells instead of using the Tab key (which we cannot do properly in an HTML page), but the user does not see any line and remains unaware that we used a table.

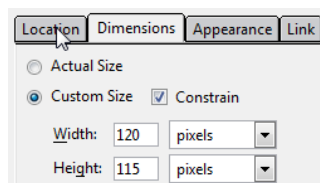
If the table cell background is not white, you may have to fiddle with the options for them to render them white (non-transparent). The best way is to change the background as follows:

1. Put the cursor into the top left cell
2. Right-click, choose “Table select” | “All cells”
3. Right-click again, choose “Table or Cell Background Color”
4. Pick the color (in our case “white”)

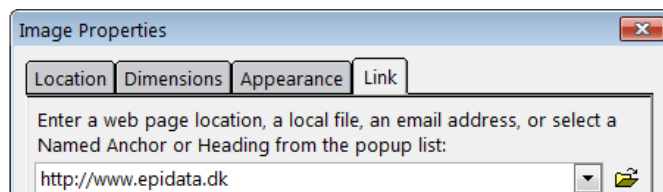
In the most upper left cell we insert one of the provided EpiData logos (the `epidata_wikiL.png` file from the `\IMAGES` sub-folder) using the “Insert” menu and also supply a (required) alternate text for the name:



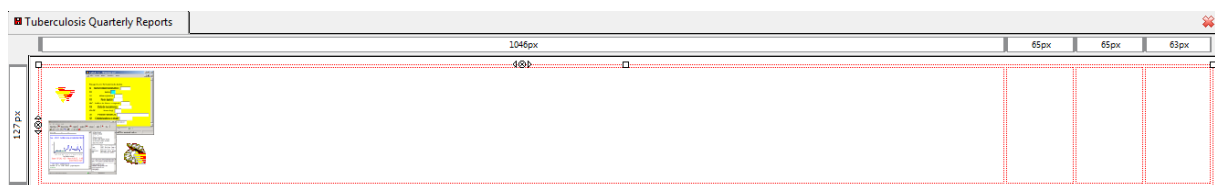
Then go to the tab “Dimensions” and decrease the current size of the width to 120 pixels:



In the “Link” tab add the URL of the EpiData website:



Then accept and you get:










Never mind for now the distortion of the column width. Add the Union logo into the most upper right cell and make a link to the Union website (<http://www.theunion.org>) in the same manner.

Join the central two cells of the first row, add the text and format it (see beginning of this Exercise) to get:

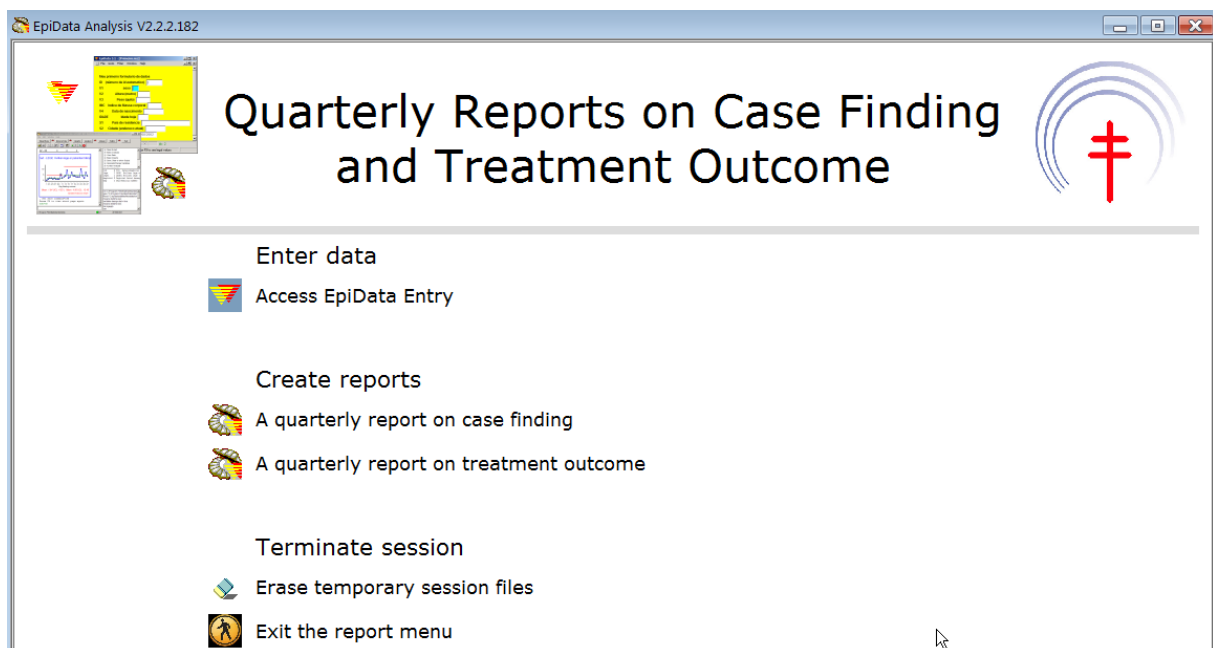


Continue adding text, formatting it, and add the appropriate icons into the appropriate places until you have the complete lay-out:

	<h2>Quarterly Reports on Case Finding and Treatment Results</h2>	
	Enter Data	
	Access EpiData Entry	
	Create Reports	
	A quarterly report on case finding	
	A quarterly report on treatment results	
	Terminate session	
	Erase temporary session files	
	Exit the menu	

Make sure to save the file. Perhaps you want to look at the source code and see that a lot has been added. We will not further edit the source code here, we will do this in a text editor. Thus exit KompoZer, this is all we are going to do with it.

If you now click the `EpiDataStat.exe` file, you will get:



Of course, there is no functionality yet (except the EpiData and Union web site icons if you are on the internet). In the next step we add functionality to the other icons.

### Editing the HTML file

Windows has an inbuilt text editor, NotePad™. There are several free text editors available that are more flexible and powerful than this one. We have selected Crimson Editor® as our choice for this course. This free text editor that has several powerful features such as showing HTML code in colors and providing the very useful feature for rectangular selections.

Open now in this text editor the `start_tb.htm` file. What you see may seem a bit overwhelming and we will thus take it apart into manageable pieces to understand what we find there and to edit it where appropriate.

You have seen before the first few lines:

```
1 <!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN" "http://www.w3.org/TR/html4/strict.dtd">
2 <html><head>
3 <meta content="text/html; charset=ISO-8859-1" http-equiv="Content-Type">
4 <meta name="copyright" content="EpiData Association">
5 <meta name="description" content="Introduction to EpiData reports"><title>Tuberculosis Quarterly Reports</title>
```

They inform us that we deal with an HTML file and that the head starts. If we collapse (and hide from view) lines 9 to 28, we see up to line 28:

```
1 <!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN" "http://www.w3.org/TR/html4/strict.dtd">
2 <html><head>
3 <meta content="text/html; charset=ISO-8859-1" http-equiv="Content-Type">
4 <meta name="copyright" content="EpiData Association">
5 <meta name="description" content="Introduction to EpiData reports"><title>Tuberculosis Quarterly Reports</title>
6
7 <style>
8 <!--
28 </style><!--<link href="epiout.css" rel="stylesheet" type="text/css" media="screen">--></head>
```

You note here the beginning and end tags for HEAD and those for STYLE embedded inside these:

```
<head><style>...</style></head>
```

Make two hard carriage returns after the closing `</head>` tag, so we see a bit better what we have next. Make another hard carriage return after the end of:

```
<body class="bodywhite">
```

so that we separate for visibility the beginning of the table definition:

```
33 <table style="text-align: left; width: 100%; border="0" cellpadding=...
```

Can you see where our first row begins and where it ends?

```
<tr><td style="background-color: white;"><a href="http://www.epidata.dk"></a></td><td colspan="2" rowspan="1" style="background-
color: white; text-align: center; width: 957px;"><big><big><big><big>Quarterly Reports on Case
Finding and Treatment Results</big></big></big></big></td><td style="background-color:
white;"><a href="http://www.theunion.org"></a></td></tr>
```

The row begins with the opening tag `<tr>` and its ending tag `</tr>`. Each cell embedded in the row has its beginning and end tags `<td ...></td>`:

	Begin tag	End tag
Row	<code>&lt;tr&gt;</code>	<code>&lt;/tr&gt;</code>
Cell	<code>&lt;td&gt;</code>	<code>&lt;/td&gt;</code>

If we isolate the first cell with the logo of EpiData, we have thus:

```
<tr><td style="background-color: white;"><a href="http://www.epidata.dk"></a></td>
```

```
<a href ... </a>
```



is the opening about everything dealing with this logo: first the link to the web site, then information about the cell style, the dimensions of the logo, the alternate text, and finally the reference to the name of the image and the place where it is relative to the current position:

```
src="../../../images/epidata_wikiL.png"
```

Thus, in summary:

The definition of the image is defined as:

```
src=
```

and the image itself is given with its name relative to the location it had when you inserted it:

```
"../../../images/epidata.png"
```

You may note here the use of forward slashes ("/") instead of the backward slashes ("\") that we have go used to on the PC. HTML as a web language uses forward slashes to indicate path.

*However, in everything we are going to write in this `start_tb.htm` file, we can do with the backward slash that we are accustomed to as there are situations where both are valid. We have here no reason to change what we know that it works. On the other hand, we are also not going to change any forward slash to a backward slash if a forward slash had been inserted into the document in KompoZer or had already been present in the initial template.*

We have no links to any of the programs that should be run when the user clicks on the EpiData Entry image defined as:

```
src="../../../images/epidata.png"
```

because there is not yet any instruction on what to do here. We are going to do that in the next step.

### Opening EpiData Entry from within EpiData Analysis

The task is to open EpiData Entry from within EpiData Analysis. To this end, we write HTML code associated with the icon for EpiData Entry:



and add the code in the correct place. Locate the HTML code that gives information about the embedding of this icon:

```

```

(You may not have `border="0"`, this removes the blue lines around the icon that designates it as a hyperlink).

Actually, the above information about the icon is located in one single cell, within the tags

```
<td> ... </td>:
```

```
<td style="background-color: white;"><td style="background-color: white; width: 42px;"></td>
```

The instruction to access EpiData Entry and, once there, to open a specific REC file is placed immediately before the definition of the icon <img>. The following text is placed there:

```
<a href="../../../epidata.exe ../originals/sample_2006.rec">
```

The `href` is a hyperlink to a reference, here to “`epidata.exe`”, the executable file that opens EpiData Entry. Why the “`../../../`”? The `start_tb.htm` is located two levels down (`\languages\en`), therefore the instruction must be to go two levels up to be in the root of the “`epidata_report`” as the `epidata.exe` file is in the root. Note that after the `\epidata.exe` there is a space before `../originals`. The reason is that the first part refers to starting the software, while the second is to open a specific file that is located in `epidata_report\originals` and to keep it independent of the name of the container name we replace “`epidata_report\`” with “`..\`”. The file we are requesting to be opened, `sample_2006.rec`, is located one level down in the `EPIDATA_REPORT\originals` folder.

After we insert the above code, we must also place an ending `</a>` tag to close the opening `<a href` before the cell closes with `</td>`:

```
<a href="../../../epidata.exe ../originals/sample_2006.rec"><img src=
"../../../images/epidata.png" alt="" border="0" style="width: 32px; height:
32px;"></a></td>
```

Test functionality after saving.

## Writing HTML code to invoke an EpiData Analysis program when clicking on the icon

Quite similar to the above, we locate now the HTML code for the EpiData Analysis icon:

```

```

Again as above, we need to insert HTML code immediately preceding the `<img src=...`. What is different is that we are already in EpiData Analysis, so it is not necessary to execute it (it would actually give an error if the default is set to run only one copy of EpiData Analysis at a time). All we need to do is to make a hyperlink to an EpiData Analysis program, specifically here to the program that should produce the first quarterly report on case finding. We will call the program “`quarterly_report_1.pgm`” and it will be located in the folder `PGM`. The command to run a program is “`run`”. We would thus think that:

```
href=run "../pgm\quarterly_report_1.pgm"
```

should do it. It is, however, not quite sufficient as the interplay between HTML and EpiData Analysis requires two additional things:

- 1) an opening command “`epi:`” to give the indicate that EpiData Analysis instructions will follow
- 2) everything related to EpiData Analysis must be placed between single quotes.

Accordingly, the above becomes:

```
href='epi: run "..\pgm\quarterly_report_1.pgm"'
```

We will add more EpiData Analysis commands, each separated by a semi-colon, like:

```
href='epi: CLS; set echo=off; run "..\pgm\quarterly_report_1.pgm"'
```

Once the program has run and produced the output, the user should get an instruction to press F8 which refreshes the start\_tb.htm file (thus gets the user back to the main menu interface):

```
href='epi: CLS; set echo=off; run "..\pgm\quarterly_report_1.pgm" type "F8: Go back to menu"'
```

Including the opening <a href and the closing </a> tag and showing the entire content of the cell, we then get:

```
<td style="background-color: white;"></td></tr><tr><td style="background-color: white; width: 42px;"><a href='epi:CLS; set echo=off; run "..\pgm\quarterly_report_1.pgm"; type "F8: Go back to menu"'></a></td>
```

Summarizing some of the specific EpiData Analysis commands that we may use in HTML:

Action	EpiData Analysis command
Clear the screen and memory	epi:CLS
Show only output, not the running of the program	set echo=off
Do not display the main menu	set display mainmenu=off
Do not show the process bar	set display worktoolbar=off
Do not show the command line	set display command prompt=off
Run the program	run "..\pgm\run_entry.pgm"
Tell the user how to return to the main menu after the program has completed	type "F8: go back to the main menu"

This summarizes the principles we use to connect (making a link to) the execution of an EpiData Analysis program to an icon. While the HTML code is now correct, nothing will happen yet because the program file does not yet exist.

## Preparing the EpiData Analysis program

We cannot work with the version of EpiData Analysis that we have in our epidata\_report because we have turned off everything that is essential to work, the menu and the command line. What we do instead is to simulate the structure of the

epidata\_report in our “normal” version of EpiData Analysis. We need the following folders in addition to what we have:

```
originals
pgm
temp
```

Copy all files from the epidata\_report\originals folder into the originals folder you just made in your “normal” version of your EpiData software folder.

Change also the EpiDataStat.ini file in your “normal” version so that it ends up in the sub-folder \temp.

This way everything is set up in a way that will allow us once we have written all the program files to simply copy them over to the \epidata\_report\pgm folder and everything will work out.

### Reading the data file

We remember that the starting default folder is the \epidata\_report\temp folder. After the standard closing of any open file, the path must thus be changed to the folder in which the data files are located, the \epidata\_report\originals folder:

```
cls
close
logclose

cd ..
cd originals

read "x.rec"
```

This is how we would usually do it. We propose here an alternative and that is to copy all EpiData files that are currently in the \originals sub-folder into the \temp folder and to actually work in the \temp folder. While this is not essential here (nor anywhere else usually), it is a way that allows us manipulating our datasets in a temporary folder and leaving the original files untouched. The commands are then as follows (assuming that we are in the \temp sub-folder):

```
copyfile "..\originals\sample_2004.rec" "sample_2004.rec" /replace
copyfile "..\originals\sample_2004.chk" "sample_2004.chk" /replace
copyfile "..\originals\sample_2005.rec" "sample_2005.rec" /replace
copyfile "..\originals\sample_2005.chk" "sample_2005.chk" /replace
copyfile "..\originals\sample_2006.rec" "sample_2006.rec" /replace
copyfile "..\originals\sample_2006.chk" "sample_2006.chk" /replace

copyfile "..\originals\uganda.rec" "uganda.rec" /replace
copyfile "..\originals\uganda.chk" "uganda.chk" /replace
```

Note 1) that the path to the sub-folder \originals takes into account the current position in the sub-folder \temp and 2) that there is no connecting “to” between the two file designations, and 3) that we need to replace the files in the \temp sub-folder in case they exist.

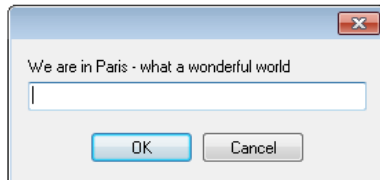
In addition, copy manually (not here in the program file) the file “side\_by\_side\_graphs.html” into the \temp folder.

## Interactive prompting

During the running of a program in a menu, there should be options to choose from. Depending on the selection the user makes, the program will then continue one or the other way(s). The program is thus to stop at a certain point and prompt the user for input. This is accomplished by typing something between two question marks. The following:

`?We are in Paris - what a wonderful world?`

gives you a pop-up box:



There is no option here, obviously, it is just a statement. To expand interactive prompting to include an option, we make use of a constant (global “variable”). To illustrate it, we are using the Uganda laboratory dataset (`uganda.rec`, included in the `originals` folder), collected over three years with a variable `REGDATE` denoting the registration date. We want to give the user the option to make a frequency of the field `SEX` for a selected year:

```
cls
close

read "uganda.rec"

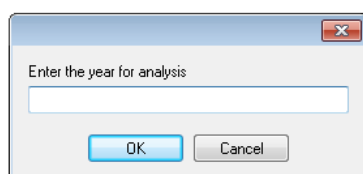
gen i regyy=year(regdate)
define yr #### global
cls

freq regyy
  yr=?Enter the year for analysis?
set echo=off
  select if regyy=@yr
  cls
  freq sex
```

We first extract the registration year `REGYY` from the registration date. Then we create a constant `YR`. Then we display the frequency (`FREQ REGYY`) of the registration year and get:

<u>regyy</u>	
	<u>N</u>
1999	17300
2000	18662
2001	18088
Total	54050

The user is then immediately prompted to enter the year:



If, say, the year 2000 is entered, then the output is:

<b>Sex of examinee</b>	
	<b>N</b>
Female	7745
Male	9326
Missing	1591
Total	18662

Note the:

```
select if regyy=@yr
```

where we make use of the constant YR.

We can use the same principles, but expand it to provide an option to do either one thing or another:

```
define yesno # global

yesno=?Do this or that: 1=Do this - 2=Do that?
  imif yesno=1 then
    (do something)
  else
    (do something else)
  endif
```

The user then enters a “1” or a “2” into the box.

We can use a nested sequence of events. If we take the Uganda dataset again and we want to give the choice to first select the year, and then have the option of analyzing the entire selected year or only a quarter of it, we write (this sample program `imif_example.pgm` is available in your PGM folder):

```
set echo=off
gen i regyy=year(regdate)
label regyy "Registration year"
gen i quarter=month(regdate)
label quarter "Quarter of the year"
recode quarter 1-3=1 4-6=2 7-9=3 10-12=4
labelvalue quarter /1="Jan-Mar"
labelvalue quarter /2="Apr-Jun"
labelvalue quarter /3="Jul-Sep"
labelvalue quarter /4="Oct-Dec"

define yr #### global
define yesno # global
define q # global
cls

set echo=on
freq regyy
  yr=?Enter year for analysis?
set echo=off
  select if regyy=@yr
  cls
```

```

yesno=?Analyze: 1: Whole year; 2: One quarter only?
imif yesno=1 then
  cls
  type "You selected: Registration year @yr" /h2
  freq sex
else
  freq quarter /v1
  q=?Select quarter?
  select if quarter=@q
  cls
  type "You selected: Registration year @yr and Quarter @q" /h2
  freq sex
endif
set echo=off

```

In the first block, we extract registration year and registration month and recode the latter into quarters.

In the second block, we define the three constants.

In the third (last) block, we allow selection of the registration year:

```

freq regyy
yr=?Enter year for analysis?
select if regyy=@yr

```

Then based on the selected year, we ask to choose the quarter

```

cls
yesno=?Analyze: 1: Whole year; 2: One quarter only?

```

After choosing between “Whole year” and “One quarter only”, we make use of the `imif` command which is closed with an `endif` command:

```

imif yesno=1 then
  cls
  type "You selected: Registration year @yr" /h2
  freq sex
else
  freq quarter /v1
  q=?Select quarter?
  select if quarter=@q
  cls
  type "You selected: Registration year @yr and Quarter @q" /h2
  freq sex
endif

```

IMIF is used to divert course in a `pgm` file depending on parameters which can, as done here, be acquired by “? ... ?”.

We use the `type` command to inform the user of the selection made. The `/h2` is HTML language for header size.

We will store all EpiData Analysis programs in the `\epidata_report\pgm` sub-folder.

## Making the EpiData Analysis program to produce the quarterly report on case finding

You will make a total of three programs:

```

quarterly_report_1.pgm
quarterly_report_2.pgm

```

cleanup.pgm

We have two actual datasets of a random selection of tuberculosis case registers from the years 2004 and 2005 from Viet Nam. They were provided by courtesy of Dr Nguyen Binh Hoa from the National Tuberculosis Program Viet Nam. They were slightly edited from the original case registers in that the original identifier was removed and replaced by an artificial one. The names of the 30 units were also changed to have non-reality units. Several variables that were collected were also removed and some other minor modifications were made. However, overall, these data are real.

There are three data files:

```
sample_2004.rec  
sample_2005.rec  
sample_2006.rec
```

The last file (sample\_2006.rec) contains no records and is used for data entry only.

The task is to produce a quarterly report on case finding. We follow here the guidelines of The Union for the quarterly report. It requires that all notifiable cases in the quarter are reported. We will get back to this shortly.

Because we set `echo=off` in the `start_tb.htm` file, the screen will stay blank during the running of the program. As this may confuse the reader, it will be useful to insert after every `CLS` command a line that informs the user that despite the blank screen something is happening, like:

```
cls  
type "Be patient...files are identified and prepared" /h2
```

The following two parts are required for the quarterly report as recommended by The Union:

ALL CASES REGISTERED IN THE QUARTER							
SMEAR-POSITIVE				SMEAR-NEGATIVE		EXTRA-PULMONARY	TOTAL
New cases	Relapses	Treatment after failure	Treatment after default	< 15 yrs	15 + yrs		

NEW SMEAR-POSITIVE CASES ONLY																
Age group (years)														TOTAL		
0-14		15-24		25-34		35-44		45-54		55-64		65 +				
M	F	M	F	M	F	M	F	M	F	M	F	M	F	Male	Female	Total

A note on the notifiable cases: Patient categories “Transfer in” and “Other” must not be reported.

When you define your variables, note that not all registered cases may always have the required information. For instance, you need three variables to determine “SMEAR-NEGATIVE, <15 yrs”: The case must be pulmonary, you must know the case is smear-negative, and you must know the age. Take this into account when deciding what you are going to present the analysis.

If you want to produce a graphics output of two graphs and show them side by side, you must first save the graphs, e.g.:



```

bar repdef2 \
    /ti="All cases" \
    /save="all_cases_1.png" /replace
cls

select repdef2<>0
select repdef2<>9
bar repdef2 \
    /save="all_cases_2.png" /replace
cls

```

Among the required files was the HTML file “side\_by\_side\_graphs.HTML”:

```

echo <table><tr><td colspan=2 border align=center >
<font color=black face="Arial" size="4">Some title that crosses over both
graphs</font></td>
<tr><td></td><td>
</td></table>

```

Replace the example names with the actual names of your graph files and then the simple command in EpiData Analysis:

```
show "side_by_side_graphs.html"
```

will show them side-by-side.

## Other programs

The process is analogous for the quarterly report on treatment results. You can use the same basic program up to the point where you produce the actual output. You may consider two outputs, one for all cases, and one for sputum smear-positive cases only.

It might be useful to have a separate program to erase all files created during the session, so that only the files remain that are needed to start anew.

## Command to exit the program

To exit the program, you do not need a special program. The EpiData Analysis command for the HTML file to quit the program without asking for conformation is:

```
<a href='epi:exit'> ... </a>
```

## Opening the menu

To make things as simple as possible for the user, not requiring search for the epidatastat.exe file, you may add a batch file (that might be named start.bat, the \*.bat extension being mandatory) at the same level as the \EPIDATA\_REPORT folder. This batch file follows simple conventions (see internet for more on writing batch files):

```

cd epidata_report
start epidatastat.exe

```

Zip both the \EPIDATA\_REPORT folder and the start.bat file into one single zip file (any name like anyname.zip will do of course), send it by email to the user with the instructions:

- 1) Unzip the anyname.zip file to any place on your computer
- 2) Double-click the start.bat file to open the menu
- 3) Start working

### **Task**

*Produce a single folder containing the EpiData Entry and Analysis programs, including the database of with an interface that permits by simple clicking to enter data or to run two standard reports, giving the user the option of choosing which country, year, laboratory, etc to analyze and is transferable to any drive or folder, with a total zipped file size of just 3.81 MB.*

*A user will be able to unzip this file onto any drive or folder, double-click the start.bat file, and be in the menu interface. Test it out by zipping it and then unzipping it onto a flash USB stick.*

## Solution to Exercise 4: Creating a menu for standard reports

### Key points:

- The EpiData Analysis interface is largely based on HTML which gives the user great flexibility in customizing it to the needs
- A menu can be written for a user that requires not skills in EpiData Analysis and allows producing standard summary reports by simply clicking like on any web site

### Task

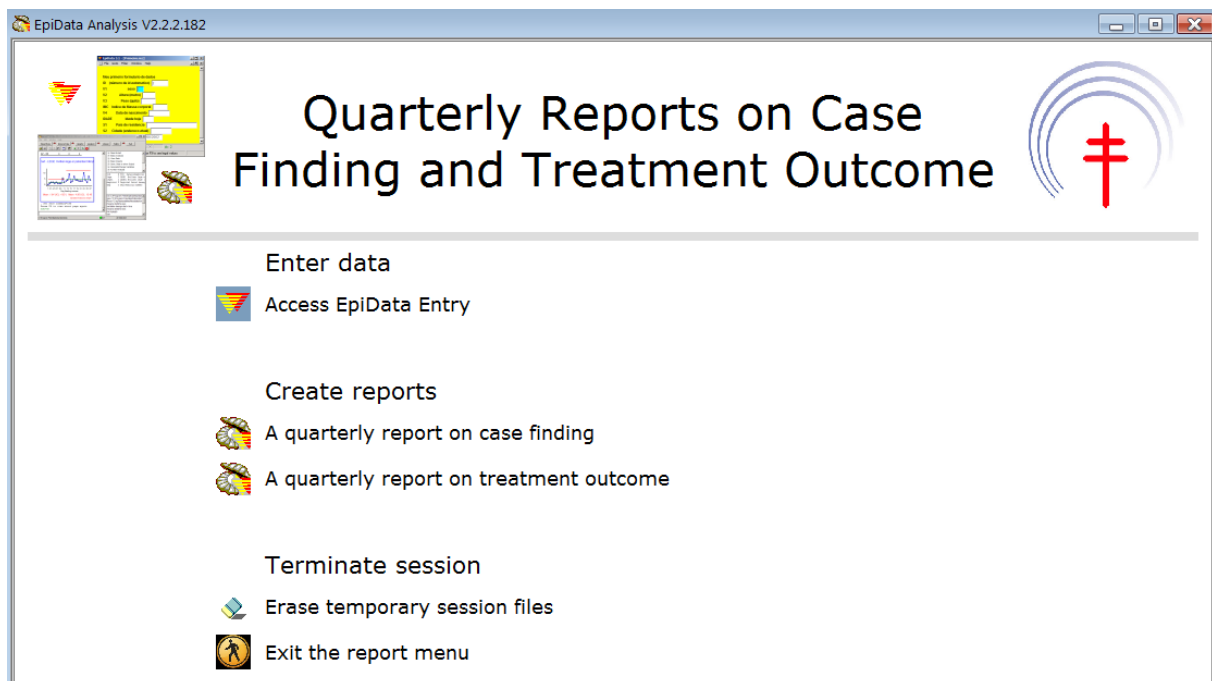
*Produce a single folder containing the EpiData Entry and Analysis programs, including the database of with an interface that permits by simple clicking to enter data or to run two standard reports, giving the user the option of choosing which country, year, laboratory, etc to analyze and is transferable to any drive or folder, with a total zipped file size of just 3.81 MB.*

*A user will be able to unzip this file onto any drive or folder, double-click the `start.bat` file, and be in the menu interface. Test it out by zipping it and then unzipping it onto a flash USB stick.*

### Solution

The entire folder is available on the course web site as a zip file.

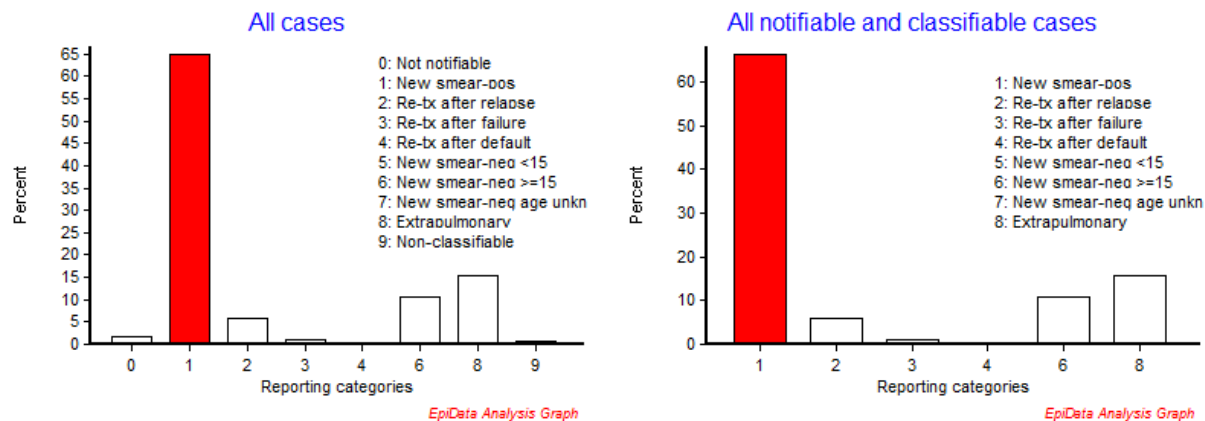
The interface the user sees:



In the following just select output graphs are shown.

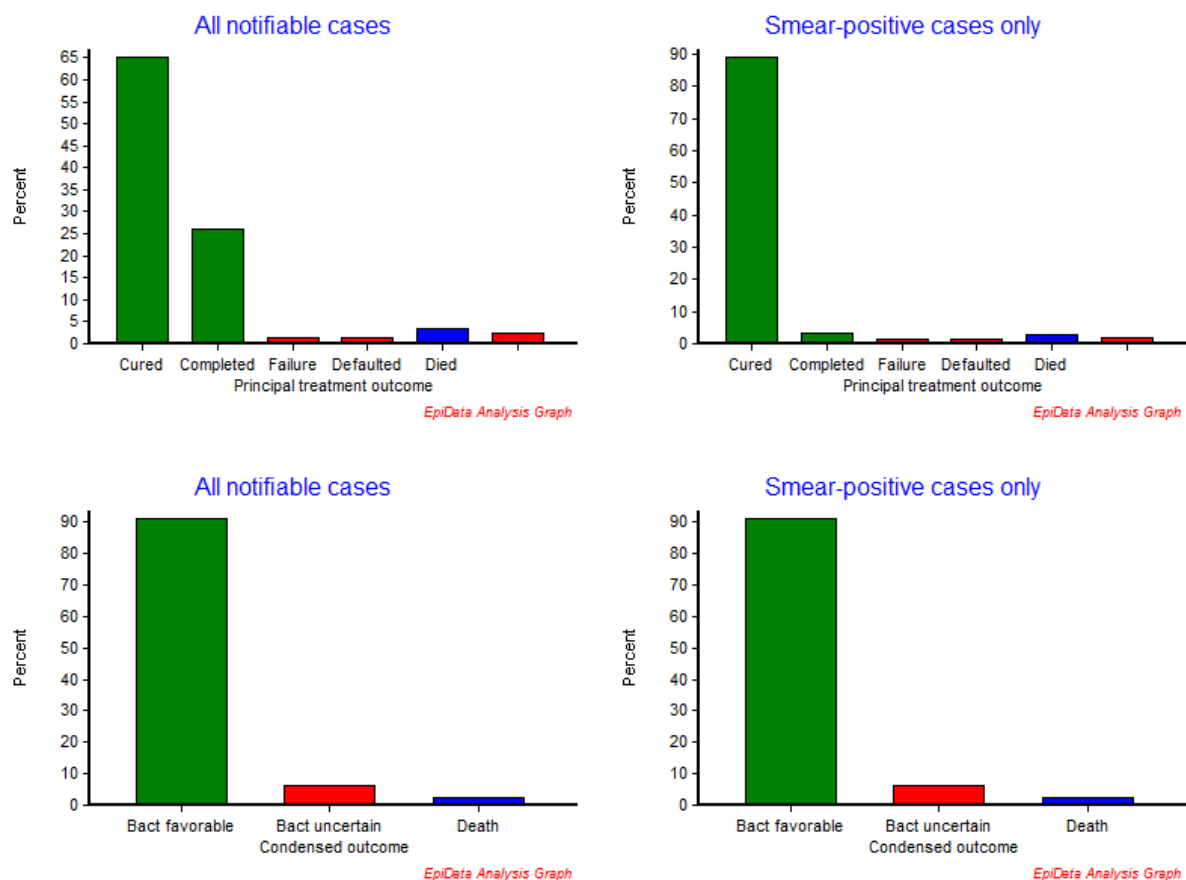
The output graphs from the first program:

### Quarterly report on case finding



The output graph from the second program:

### Quarterly report on treatment outcome



## Exercise 5: Formatting standardized analysis output in a spreadsheet

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

- Use a dummy file to produce a standardized aggregated EpiData file
- Producing formatted EpiData output in a spreadsheet

The following is a report describing tuberculosis patients by microscopic case definition and four variables (age, sex, disease site, and patient category), summarized in one single nicely formatted table:

Report from the National Tuberculosis Program

Characteristics of microscopy confirmed and other cases

Characteristic	Confirmed cases		Other cases	Total	Col %
	N	Row %			
Total	13	92.9	1	14	100.0
Age quartiles					
Quartile 1	3	100.0	0	3	21.4
Quartile 2	3	100.0	0	3	21.4
Quartile 3	1	50.0	1	2	14.3
Quartile 4	6	100.0	0	6	42.9
Age missing	0	NA	0	0	0.0
Sex					
Female	4	100.0	0	4	28.6
Male	9	90.0	1	10	71.4
Sex missing	0	NA	0	0	0.0
Disease site					
Pulmonary	13	92.9	1	14	100.0
Extrapulmonary	0	NA	0	0	0.0
Site missing	0	NA	0	0	0.0
Patient category					
New	12	92.3	1	13	92.9
Relapse	1	100.0	0	1	7.1
After failure	0	NA	0	0	0.0
After default	0	NA	0	0	0.0
Transfer in	0	NA	0	0	0.0
Other	0	NA	0	0	0.0
Category missing	0	NA	0	0	0.0

We note that the dataset is so small (14 records) that for every variable some strata have no cases. For instance, if we make a table in EpiData Analysis for case by category:

tables case category

we get:

Smear-based case definition			
Treatment category	Case	Non-case	Total
New	12	1	13
Relapse	1	0	1
Total	13	1	14

In other words, the EpiData output shows only strata containing at least one case. This is not a particularity of EpiData, this is quite generally so with any analysis program. In other words, depending on the selection criteria, we get visualization of varying column-row combinations, for instance:

Smear-based case definition			
Treatment category	Case	Non-case	Total
New	6	1	7
Relapse	1	0	1
No category recorded	0	1	1
Total	7	2	9

It is of course also not possible to have several tables condensed into a single one. For instance, if we produce two cross-tabulations, we formulate it as:

```
tables case sex
tables case category
```

and we get two separate tables:

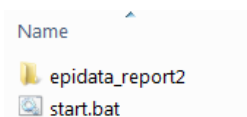
Smear-based case definition			
Patient's sex	Case	Non-case	Total
Female	0	1	1
Male	7	1	8
Total	7	2	9

Smear-based case definition			
Treatment category	Case	Non-case	Total
New	6	1	7
Relapse	1	0	1
No category recorded	0	1	1
Total	7	2	9

In this exercise, we will automate a procedure to get EpiData output from multiple tables into one single nicely formatted table by using 1) a work-around to get all strata displayed even if their case count is zero, and 2) formatting the output in an esthetically appealing way by taking recourse to a spreadsheet template.

## Modifying the basic menu

You have a required zip file “course\_d\_ex05\_required.zip”. This is essentially the solution to Exercise 4, but with a slightly changed package name (“epidata\_report2\”) to avoid confusion. Ensure first that your temporary folder C:\TEMP is empty, then unzip this required zip file into that folder and you get a folder and its start.bat:



Test it by double-clicking the start.bat file and then exit the menu again.

We need four more sub-folders in the epidata\_report2 folder:

```
dummyset\
LibreOfficePortable\
templates\
workfolder\
```

After you created them, make sure that the epidata\_course folder is empty, then copy the following five folders into the epidata\_course folder:

```
dummyset\
originals\
pgm\
temp\
workfolder\
```

These will be used as simulation folders: we do everything in our “regular” EpiData Analysis, simulating what will happen in the menu. Once we assured functionality, we copy our

simulation folders into the menu in the `c:\temp` folder, start the menu and then run the program from there.

We will not yet deal with the `LibreOfficePortable` folder nor now change the `start_tb.htm` file.

## The dummy files

We mentioned above that a prerequisite for a standard output is that we always need the same output style or, in other words, a display of all possible cells in a table. If we have a column variable `COLVAR` with 3 levels and a row variable `ROWVAR` with 4 levels we have  $3 \times 4 = 12$  cells (not counting the marginals):

ROWVAR	COLVAR		
	1	2	3
1	c1-r1	c2-r1	c3-r1
2	c1-r2	c2-r2	c3-r2
3	c1-r3	c2-r3	c3-r3
4	c1-r4	c2-r4	c3-r4

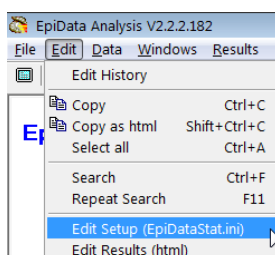
We need  $c$  times  $r$  records to have one record for every possible combination. To make this generically applicable, we will create an EpiData \*.rec file with two variables:

```
colvar ##  
rowvar ##
```

and pretend that we have 12 columns and only 1 row. We thus complete a total of 12 records with column values 1, 2, ..., 12 and row value 1, starting with making a `dummy_00.qes` file, then from it its `dummy_00.rec` file. We save both files in our `dummyset` sub-folder:

```
epidata_course\dummyset\
```

When now opening our “regular” or “standard” (not menu) EpiData Analysis, we first edit the `EpiDataStat.ini` file in Edit:



so that the last CD command directs EpiData Analysis to start in the `temp` sub-folder of the `epidata_course` folder:

```
cd c:\epidata_course\temp
```

As you perhaps remember, the `EpiDataStat.ini` file of the menu is also set to start in that folder. This allows moving one level up, then one level down into any sub-folder of the package.

Our intention will be to create a table consisting of four sub-tables, resulting from cross-tabulation of microscopy case status (smear-positive or other than smear-positive) *versus*

quartiles of age, sex, site of tuberculosis, and category of tuberculosis patient. For each of the four sub-tables we create a dummy set with 1 record for each possible cell. We start with creating one such dataset for case status versus age quartiles. Case status (the column variable) has 2 levels (case, non-case) and quartiles (the row variable) has 5 levels (the 4 quartiles, and missing age).

We start by opening a new \*.pgm file in the EpiData Analysis editor, directing it from the current temp sub-folder to the dummyset sub-folder and reading the dummy\_00.rec file:

```
* Create dummy set

cls
close

cd ..
cd dummyset

read "dummy_00.rec"
```

and save it as dummy\_case\_demogr\_01.pgm in the dummyset folder. The dataset has 12 levels for the column variable rather than only the 2 we need. This will be addressed at the end of the program with a selection. The row value it has is 1, but as we will do some copying later with modification, we will overwrite it with, well "1" (one).

```
read "dummy_00.rec"
rowvar=1
```

We need to create a unique identifier from the combination of the column and the row value. Later, we will create such an identifier in the same manner from the actual data set and then the dummy file created here can be merged to the actual dataset on this identifier. As we want to make this generically valid, we will consider how we actually designed the entry form, i.e. with two-digit fields for both column and row, thus we expand:

```
cls
close
read "dummy_00.rec"
rowvar=1
gen s(2) tempcol=string(colvar)
gen s(2) temprow=string(rowvar)
if colvar<10 then tempcol="0"+string(colvar)
if rowvar<10 then temprow="0"+string(rowvar)
gen s(5) id=tempcol+"-"+temprow
drop tempcol temprow
savedata "dummy_01.rec" /replace
```

Now, we copy the whole block and change only what is shown in red:

```
cls
close
read "dummy_00.rec"
rowvar=2
gen s(2) tempcol=string(colvar)
gen s(2) temprow=string(rowvar)
if colvar<10 then tempcol="0"+string(colvar)
if rowvar<10 then temprow="0"+string(rowvar)
gen s(5) id=tempcol+"-"+temprow
drop tempcol temprow
savedata "dummy_02.rec" /replace
```

and repeat it until we have a total of 5 dummy sets with row values of 1, 2, 3, 4, and 5 respectively, then append the 5 files:



```

cls
close
read "dummy_01.rec"
append /file="dummy_02.rec"
append /file="dummy_03.rec"
append /file="dummy_04.rec"
append /file="dummy_05.rec"
select colvar<=2
savedata "dummy_case_demogr_01.rec" /replace

```

Note the selection (in red) for the column levels in the second to last line. We must end up with 10 records, shown in the browser window as:

	colvar	rowvar	id
1	1	1	01-01
2	2	1	02-01
3	1	2	01-02
4	2	2	02-02
5	1	3	01-03
6	2	3	02-03
7	1	4	01-04
8	2	4	02-04
9	1	5	01-05
10	2	5	02-05

At the end, we erase all intermediary files:

```

erase "dummy_01.rec"
erase "dummy_02.rec"
erase "dummy_03.rec"
erase "dummy_04.rec"
erase "dummy_05.rec"

```

While we are at it, we should make dummy sets for the other three variables:

Row variable	Levels (strata)	Dummy set file name	Records
age quartile	5 (Quartiles 1, 2, 3, 4, and missing age)	dummy_case_demogr_01.rec	10
sex	3 (female, male, missing)	dummy_case_demogr_02.rec	6
site	3 (pulmonary, extrapulmonary, missing)	dummy_case_demogr_03.rec	6
category	7 (new, relapse, failure, default, transfer, other, missing)	dummy_case_demogr_04.rec	14

The principle is the same, and it is recommended to save each \*.pgm file under a separate corresponding name to facilitate corrections if errors had been made and are found out only later.

### The main analysis file

Let's open a new file in the EpiData Analysis editor and save it right away as case\_demogr.pgm in the \pgm subfolder.

Similar as in the previous exercise, we copy the required files into the newly created working folder which we then access:

```

cd ..
cd originals
copyfile "sample_2004.chk" "..\workfolder\sample_2004.chk" /replace
copyfile "sample_2004.rec" "..\workfolder\sample_2004.rec" /replace
copyfile "sample_2005.chk" "..\workfolder\sample_2005.chk" /replace
copyfile "sample_2005.rec" "..\workfolder\sample_2005.rec" /replace

cd ..
cd dummyset
copyfile "dummy_case_demogr_01.rec" "..\workfolder\dummy_case_demogr_01.rec" /replace
copyfile "dummy_case_demogr_02.rec" "..\workfolder\dummy_case_demogr_02.rec" /replace
copyfile "dummy_case_demogr_03.rec" "..\workfolder\dummy_case_demogr_03.rec" /replace
copyfile "dummy_case_demogr_04.rec" "..\workfolder\dummy_case_demogr_04.rec" /replace

cd ..
cd workfolder

```

We then append the two years of data and make our case definition for the column variable:

```

read "sample_2004.rec"
append /file="sample_2005.rec"

define case #
case=2
if sm00>0 and sm00<9 then case=1
label case "Smear-based case definition"
labelvalue case /1="Case"
labelvalue case /2="Non-case"

```

Note that “Non-cases” are not smear-negative, they are not proven smear-positive, i.e. include all cases without a positive smear.

If you run the above, we see that the dataset contains a variable `regdate`, the registration date. In the previous exercise we selected cases for quarters. We want to make this more flexible here and allow the person running the program to choose interactively any time interval. We need thus six variables for day, month, and year to begin and to end and two additional variables to construct the two corresponding dates:

```

cls
type "Be patient...data compilation in progress" /h2
define regbegdd ##    global
define regbegmm ##    global
define regbegyy ####  global
define regenddd ##    global
define regendmm ##    global
define regendyy ####  global
define regbegdate <dd/mm/yyyy>
define regenddate <dd/mm/yyyy>

```

We then allow interactive selection for the beginning of the interval:

```

cls
type "Enter beginning of registration period, day, month, year" /h2
type "Only a date between 1 Jan 2004 and 31 Dec 2005 is possible" /h2
regbegdd=?Enter begin day of registration?
regbegmm=?Enter begin month of registration?
regbegyy=?Enter begin year (4-digit) of registration?
regbegdate=dmy(regbegdd, regbegmm, regbegyy)

```

and selection for the ending of the interval:

```

cls
type "Your chosen start date is @regbegdd - @regbegmm - @regbegyy" /h2
type "Enter now your choice of the end of registration period, day, month, year" /h2
type "Only a date between 1 Jan 2004 and 31 Dec 2005 is possible" /h2
regenddd=?Enter end day of registration?
regendmm=?Enter end month of registration?

```

```
regendyy=?Enter end year (4-digit) of registration?
regenddate=dmy(regenddd, regendmm, regendyy)
```

Note the blue line which allows displaying the chosen date on the output screen.

Finally, we make the selection of the interval, and retain just the variables that we need for further work to speed up processing time and save the reduced data set:

```
cls
type "Be patient ... analysis in progress" /h2
select regdate>=regbegdate and regdate<=regenddate
keep case age sex site category
savedata "temp_01.rec" /replace
```

Run this part and for demonstration purposes, use the time interval including 1 Jan 2004 through 4 Jan 2004 to get a small data set that will assure that not all strata have cases (you should get 15 records).

So far, we had the preparatory steps. Now comes the construction of the first table, which is case status *versus* quartiles of age. We read the dataset and define a variable `colvar` to have identity with the variable name used in the dummy dataset and assign it the values of the variable `case`:

```
*****
* case_demogr_01: Age quartiles
```

```
cls
type "Be patient...data compilation in progress" /h2
close
logclose
read "temp_01.rec"
```

```
define colvar ##
colvar=case
```

The row variable should take the values of the age quartiles. In Part B, Exercise 2, we introduced how we can use the values EpiData Analysis stores in the variable `result` if we let it execute a command such as `means`. We utilize this here for the variable `age`:

```
means age if age<>99
define quart #
if age <$p251 then quart=1
if age>=$p251 and age<$p501 then quart=2
if age>=$p501 and age<$p751 then quart=3
if age>=$p751 then quart=4
if age =99 then quart=5
```

Then, in analogy to the column variable, we write for the row variable:

```
define rowvar ##
rowvar=quart
```

This way we have the two variables required to make our cross-table:

```
tables colvar rowvar
```

In Part B, Exercise 3, you learned that by replacing the `tables` command with the `aggregate` command we can get the result written into a file and then manipulate that file like creating a unique identifier in the same manner as we did in the dummy file:

```
aggregate rowvar colvar /close
cls
type "Be patient ... analysis in progress" /h2
```

```

gen s(2) id01=string(colvar)
gen s(2) id02=string(rowvar)
if colvar<10 then id01="0"+string(colvar)
if rowvar<10 then id02="0"+string(rowvar)
gen s(5) id=id01+"-"+id02
drop id01 id02
savedata "temp_02.rec" /replace

```

If we run this (with our above selection of 15 records) and browse this file, we get:

	rowvar	colvar	N	id
1	1	1	3	01-01
2	2	1	3	01-02
3	2	2	1	02-02
4	3	1	1	01-03
5	3	2	1	02-03
6	4	1	6	01-04

Manually adding up the field N we get the 15 case we expect. We have 6 records. As the column variable has 2 levels and the row variable 5, we should have  $2 \times 5 = 10$  records if each possible cell had cases. Thus, 4 cells have no cases. Our dummy file for age quartiles `dummy_case_demogr_01.rec` has all 10 records. We thus read in the dummy file and merge it on the id to this `temp_02.rec` file:

```

cls
type "Be patient ... analysis in progress" /h2
close
read "dummy_case_demogr_01.rec"
merge id /file="temp_02.rec"
sort rowvar colvar

```

Browsing shows which cells are missing in the actual data set and are contributed by the dummy set (Only in memory):

	colvar	rowvar	id	N	MergeVar
1	1	1	01-01	3	In both
2	2	1	02-01	.	Only in memory (Original)
3	1	2	01-02	3	In both
4	2	2	02-02	1	In both
5	1	3	01-03	1	In both
6	2	3	02-03	1	In both
7	1	4	01-04	6	In both
8	2	4	02-04	.	Only in memory (Original)
9	1	5	01-05	.	Only in memory (Original)
10	2	5	02-05	.	Only in memory (Original)

It is now simple to replace the period (for missing) with a count of zero and do a bit book keeping:

```

gen i cases=n
if n=. then cases=0
drop n mergevar
savedata "temp_03.rec" /replace

```

to get with browsing:

	colvar	rowvar	id	cases
1	1	1	01-01	3
2	2	1	02-01	0
3	1	2	01-02	3
4	2	2	02-02	1
5	1	3	01-03	1
6	2	3	02-03	1
7	1	4	01-04	6
8	2	4	02-04	0
9	1	5	01-05	0
10	2	5	02-05	0

We add a string variable to allow easy identification of values of the field rowvar:

```
gen s(35) rowvartxt=.
if rowvar=1 then rowvartxt="01_age_Q1"
if rowvar=2 then rowvartxt="01_age_Q2"
if rowvar=3 then rowvartxt="01_age_Q3"
if rowvar=4 then rowvartxt="01_age_Q4"
if rowvar=5 then rowvartxt="01_age_missing"
```

As shown in Part D, Exercise 1, we reshuffle from long to wide, select only the remaining necessary row, and drop superfluous variables:

```
cls
type "Be patient ... analysis in progress" /h2
gen i case=.
gen i noncase=.

cls
type "Be patient ... analysis in progress" /h2
case=cases
if colvar[_n+1]=2 then noncase=cases[_n+1]

select colvar=1
drop colvar rowvar id cases

savedata "case_demogr_01.rec" /replace
```

and we get a compact rectangular file in browsing

	rowvartxt	case	noncase
1	01_age_Q1	3	0
2	01_age_Q2	3	1
3	01_age_Q3	1	1
4	01_age_Q4	6	0
5	01_age_missing	0	0

in which each cell has a value, in contrast to the tables command which omits the missing:

colvar			
rowvar	1	2	Total
1	3	0	3
2	3	1	4
3	1	1	2
4	6	0	6
Total	13	2	15

The next sub-table deals with case status versus sex. Because of our setup, we can copy the entire block, and then change the assignment to rowvar, the dummy file set used for merging, change the values for rowvartxt, and finally save the file as case\_demogr\_02.rec. When done and then browsing you should get:

	rowvartxt	case	noncase
1	02_sex_female	4	0
2	02_sex_male	9	2
3	02_sex_missing	0	0

The two files we created, case\_demogr\_01.rec and case\_demogr\_02.rec have the same columns and can thus be appended:

```
cls
type "Be patient ... analysis in progress" /h2
close
read "case_demogr_01.rec"
append /file="case_demogr_02.rec"
```

In the menu, we want that at the end of the program, the browser window opens and the user is instructed what to do with the content of the browser window:

```
set display databrowser=on
browse
cls
type "Data compilation complete" /h2
type "Copy content of the browser window to clipboard and paste into spreadsheet" /h2
```

This gives us, labeled and clear, but with some esthetic deficiencies for the general user:

	rowvartxt	case	noncase
1	01_age_Q1	3	0
2	01_age_Q2	3	1
3	01_age_Q3	1	1
4	01_age_Q4	6	0
5	01_age_missing	0	0
6	02_sex_female	4	0
7	02_sex_male	9	2
8	02_sex_missing	0	0

With that we are done (run the entire program to test that it works). We will now turn to the next part, the menu. The finalization of the script with all four variables rather than just these two will be part of the task.

For the time being we are done with the work in the “regular” EpiData Analysis and what we did will work smoothly in the EpiData Analysis menu. Thus, copy all sub-folders from the `epidata_course` to the `epidata_report2` folder, overwriting everything if prompted.

## Installing the LibreOfficePortable suite

The challenge at hand is to format the above `*.rec` file into something useful and esthetically pleasing that can easily be shared with others who do not have EpiData installed on their computer. The most appealing approach is the use of non-proprietary software that is part of the menu package and can thus be fully controlled and easily used through a few clicks.

This is most efficiently accomplished with a pre-formatted spreadsheet template. Pre-formatting is possible because the number of columns and rows is fixed in the EpiData Analysis output: it was the very purpose of merging a pre-defined dummy set with a dataset sub-selection. Using proprietary Excel™ cannot as smoothly solve the problem. It notably requires that the user has purchased the software. It is also not that trivial to access Excel™ from the EpiData `start_tb.htm` file. A straight forward way is to use the portable edition of non-proprietary, open-source LibreOffice (formerly known as OpenOffice). It is a true suite, that is, it has a core functionality that holds all individual sub-components (seven in total) together. There is no point in trying to install only the spreadsheet sub-component, it is making things complicated while not at all being a space-saver. In the software group you find the install file (or preferably download it from the Internet, but note its large file size). Install it into the sub-folder LibreOfficePortable in the `epidata_report2` folder we have already created.

## Editing the `start_tb.htm` file

As you remember, the `start_tb.htm` file is the file called when you start the EpiData Analysis executable file `epidatastat.exe`. It defines the menu interface. It is located in the `\languages\en` sub-folder. We open it in our text editor (while possible of course, don’t do it in NotePad™, it is not a quality text editor. Use instead free Crimson Editor you find in the software section, the coloring is just one of its many advantageous aspects).

Just after the opening table tag `<tbody>` you find (on line 34 if you use Crimson):

```
33 <tbody>
34 <tr><td style="background-color: white; width: 73px;"><a href="http://www.epidata.dk"></a></td><td colspan="2" rowspan="1" style="background-color:
    white; width: 41px; text-align: center;"><big><big><big><big>Quarterly Reports on Case
    Finding and Treatment Outcome</big></big></big></big></td><td style="background-color:
    white; width: 121px;"><a href="http://www.theunion.org"></a></td></tr>
```

All cells in a given the table row (shown with the beginning tag `</tr>` and the ending tag `</tr>` circled red above) are a bit crammed up. We want to isolate the individual cells for clarity’s sake, take the row apart with hard carriage returns so that each cell starting with an opening `<td>` and a closing `</td>` tag gets its own paragraph as follows:

```

34 <tr>
35 <td style="background-color: white; width: 73px;"><a href="http://www.epidata.dk"></a></td>
36 <td colspan="2" rowspan="1" style="background-color: white; width: 41px; text-align: center;
   "><big><big><big><big>Quarterly Reports on Case Finding and Treatment Outcome</big></big>
   </big></big></td>
37 <td style="background-color: white; width: 121px;"><a href="http://www.theunion.org"></a></td>
38 </tr>

```

This allows counting the cells: here we have 4 (two cells in the middle are merged with colspan="2"). However, we want to have 6 cells per row:

We now have:

EpiData icon				Union icon
	Analysis icon	A quarterly report on case finding		

We needed thus only 4 cells, but now we want to change this to:

EpiData icon					Union icon
	Analysis icon	Run EpiData analysis program	Spread-sheet icon	Paste output into spreadsheet	

This requires 6 cells per row. In the top row above, we thus change this line:

```

36 <td colspan="2" rowspan="1" style="background-color: white; width: 41px; text-align: center;
   "><big><big><big><big>Quarterly Reports on Case Finding and Treatment Outcome</big></big>
   </big></big></td>

```

to expand the middle colspan="2" to colspan="4":

```

36 <td colspan="4" rowspan="1" style="background-color: white; width: 41px; text-align: center;
   "><big><big><big><big>Quarterly Reports on Case Finding and Treatment Outcome</big></big>
   </big></big></td>

```

The subsequent empty (no text, no icons) row is changed from:

```

40 <tr>
41 <td colspan="4" rowspan="1"></td>
42 </tr>

```

to

```

40 <tr>
41 <td colspan="6" rowspan="1"></td>
42 </tr>



```

We thus change (tedious perhaps but also educational) every row to always have a total of 6 cells per row, irrespective of whether the row is empty or contains icons and / or instructions. Attention must be paid to make the merging (colspan) in the correct places, and noting that



sometimes we need `colspan="2"` and sometimes `colspan="3"`. If you are all done with each row in this way and you refresh the menu interface (F8), there should not be any visually apparent change, but you have changed your menu table from 4 to 6 columns.

Next we duplicate by copying the row with “A quarterly report on treatment outcome” and change the text to “Demographic and other characteristics of patients” which should give us:

-  A quarterly report on treatment outcome
-  Demographic and other characteristics of patients


If we remove the `colspan="3"` in what is below line 86:

```
83 <tr>
84 <td style="background-color: white; width: 73px;"></td>
85 <td style="background-color: white; width: 41px;"><a href='epi: set echo=off; CLS; run "..\pgm\quarterly_report_2.pgm"; type "F8: Go back to Menu"'></td>
86 <td colspan="3" style="background-color: white; width: 898px;">Demographic and other characteristics of patients</td>
87 <td style="background-color: white; width: 121px;"></td></tr>
```

and duplicate lines 85 and 86, then we should get:

```
83 <tr>
84 <td style="background-color: white; width: 73px;"></td>
85 <td style="background-color: white; width: 41px;"><a href='epi: set echo=off; CLS; run "..\pgm\quarterly_report_2.pgm"; type "F8: Go back to Menu"'></td>
86 <td style="background-color: white; width: 898px;">Demographic and other characteristics of patients</td>
87 <td style="background-color: white; width: 41px;"><a href='epi: set echo=off; CLS; run "..\pgm\quarterly_report_2.pgm"; type "F8: Go back to Menu"'></td>
88 <td style="background-color: white; width: 898px;">Demographic and other characteristics of patients</td>
89 <td style="background-color: white; width: 121px;"></td></tr>
```

displayed in the browser as:

-  A quarterly report on case finding
-  A quarterly report on treatment outcome
-  Demographic and other characteristics of patients
-  Demographic and other characteristics of patients

In line 85 (here) we change the name of the program file “quarterly\_report\_2.pgm” to the program file we wrote before, i.e. “case\_demogr.pgm”. After saving and then refreshing the menu in the browser, we test it and should get (perhaps with different numbers, depending on our interval selection):

EpiData Analysis: V2.2.2.182 c:\temp\epidata\_report2\workfolder\case\_demogr\_01.rec

Data compilation complete

Copy content of the browser window to clipboard and paste into spreadsheet

F8: Go back to Menu

Browse

	row/col	case	noncase
1	01_age_01	3	0
2	01_age_02	3	1
3	01_age_03	1	1
4	01_age_04	6	0
5	01_age_missing	0	0
6	02_sex_female	4	0
7	02_sex_male	9	2
8	02_sex_missing	0	0

What’s above in lines 86 and 87 describes the contents for cells 4 and 5 of the total 6 cells in the row. It must become something quite different. What’s between the single quotes shown below in blue:

```
<a href='epi: set echo=off; CLS; run "..\pgm\case_demogr.pgm"; type "F8: Go back to Menu"'>
```

are all EpiData Analysis commands. Instead we want to call the LibreOffice spreadsheet and within a template (which we have not yet written), which will get the name “case\_demogr.ots” (LibreOffice spreadsheet templates have the extension \*.ots). Thus, all between the single quotes must be deleted and be replaced with the path to the \*.exe file of the LibreOffice spreadsheet (the “LibreOfficeCalcPortable.exe” file in the sub-folder LibreOfficePortable). The template “case\_demogr.ots” will reside in the sub-folder templates. Hierarchically at the same location of images the path to the executable for the LibreOffice spreadsheet needs analogously a preceding double “../..”, i.e. the reference becomes:

```
<a href= "../..../LibreOfficePortable/LibreOfficeCalcPortable.exe ../templates/case_demogr.ots">
```


The reference to the image:


```
alt="epidata_analysis" src="../../images/epidatastat.png"
```


needs to be changed, using the logo of LibreOffice spreadsheet provided in the images folder:

```
alt="LibreOffice" src="../../images/logo_libreoffice_calc.png"
```

Finally, the label: “Demographic and other characteristics of patients” could be changed to something like “Access the spreadsheet to format EpiData output”. Displayed in the browser window after refreshing it shows as:

 A quarterly report on treatment outcome

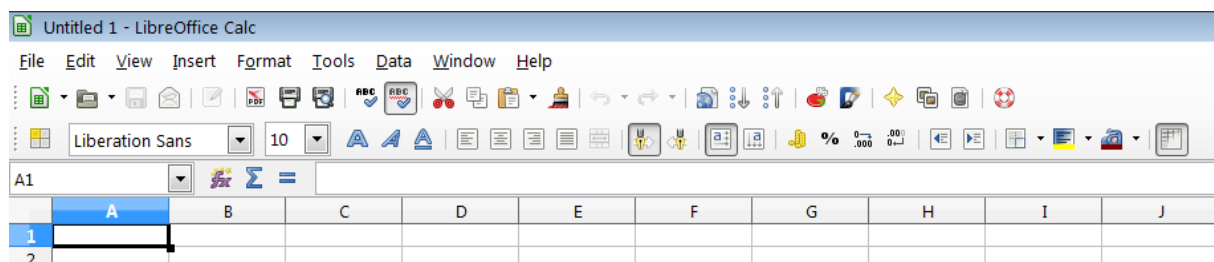
 Demographic and other characteristics of patients

 Access the spreadsheet to format EpiData output

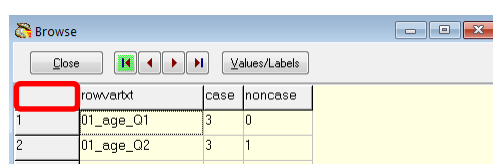
Access is of course not functional as the template is not yet available. This is the next thing we are making.

## Creating a template in the LibreOfficeCalcPortable spreadsheet

Open a new spreadsheet by double-clicking the executable “LibreOfficeCalcPortable.exe” and you get an interface similar to the Microsoft Excel™ spreadsheet:



If you return to our menu, you can still call browsing with F6 as you would in the “regular” EpiData Analysis, the file created is still there and you get:

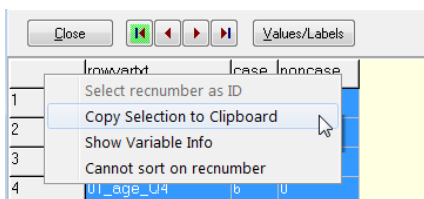


row	case	noncase
1	01_age_Q1	3
2	01_age_Q2	3
3	01_age_Q3	1

If you click on the intersection between rows and columns (shown red-circled above), the entire file gets marked (as in a spreadsheet):

	rowvartxt	case	noncase
1	01_age_Q1	3	0
2	01_age_Q2	3	1
3	01_age_Q3	1	1
4	01_age_Q4	6	0
5	01_age_missing	0	0
6	02_sex_female	4	0
7	02_sex_male	9	2
8	02_sex_missing	0	0

Right-clicking gives the option to copy to Clipboard (**CTRL+C** will also do):



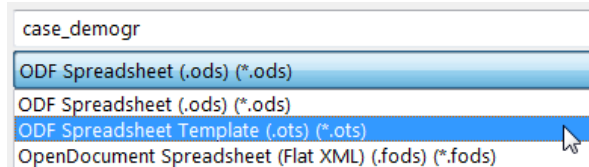
Copy and Close and return to the spreadsheet and paste (**CTRL+V**) to get prompted:

If **Tab** is not ticked, you have to, to get the input into columns rather than into one cell per row:

After OK you get the browser output copied into the spreadsheet:

	A	B	C
1	rowvarbxt	case	noncase
2	01_age_Q1	3	0
3	01_age_Q2	3	1
4	01_age_Q3	1	1
5	01_age_Q4	6	0
6	01_age_miss	0	0
7	02_sex_female	4	0
8	02_sex_male	9	2
9	02_sex_miss	0	0

Save this right away as an ODF Spreadsheet Template (.ots) (\*.ots):



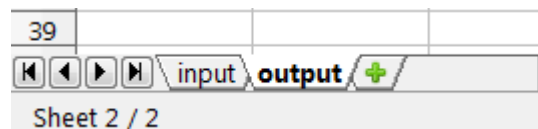
in the \templates sub-folder. As long as we don't quit LibreOffice, saving is always to the template (not a regular spreadsheet file, which would have the extension \*.ods).

We will have two sheets in this template, one for input, and a second for output, and we label them at the bottom, changing:

From default label of one sheet ...



... to customized label with two sheets



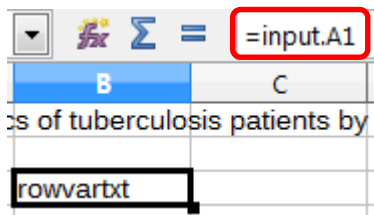
In the output sheet, we start with writing some more or less sensible title:

	A	B	C	D	E
1	Characteristics of tuberculosis patients by microscopy-defined case status				
2					

You may know from Microsoft Excel™ that you can copy content from one sheet's cell to any cell in a second sheet (here output), if you type an equal sign "=" into the destination cell in the output sheet then switch to the source cell in the first sheet (here input) and press there the hard carriage return. Specifically here, move your cursor into cell B3 of the output sheet, type the equal sign, then go to cell A1 in the input sheet and press the hard carriage return key. You will automatically be returned to the output sheet and now have the value written into cell B3:

	A	B	C	D	E
1	Characteristics of tuberculosis patients by microscopy-defined case status				
2					
3		rowvarbxt			
4					

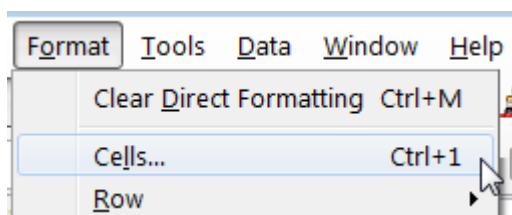
If you inspect the B3 cell content you see the formula:



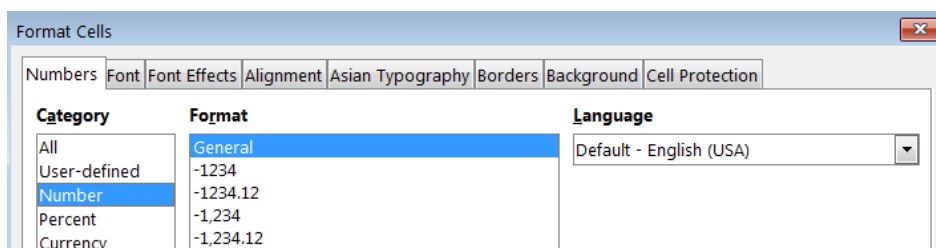
Copy this cell down and then copy the column to the right to get the entire content of the input sheet:

	A	B	C	D
1	Characteristics of tuberculosis patients by microscopy-def			
2				
3		rowvartxt	case	noncase
4		01_age_Q1	3	0
5		01_age_Q2	3	1
6		01_age_Q3	1	1
7		01_age_Q4	6	0
8		01_age_miss	0	0
9		02_sex_fem	4	0
10		02_sex_male	9	2
11		02_sex_miss	0	0

We refine formatting further, inserting rows and making more appealing labels (after all our row names were just for unambiguous identification). In addition to the standards of inserting rows and columns and so on, make use of Format | Cells:



It has an extensive menu for all sorts of things on formatting cells:



To make sums you may use =sum(a1:a4) to get the sum of cells a1+a2+a3+a4. Do you know how to avoid the error message from a division by zero:

Female	4	100.0	0	4	26.7
Male	9	81.8	2	11	73.3
Sex missing	0	#DIV/0!	0	0	0.0

If you type:

=IF(F11<>0,C11/F11\*100,"NA")

you get:

Female	4	100.0	0	4	26.7
Male	9	81.8	2	11	73.3
Sex missing	0	NA	0	0	0.0

Note that the default of LibreOffice are American measures. For instance, the default page size is Letter size rather than A4. It is easy to change if you need to do so (consider printing).

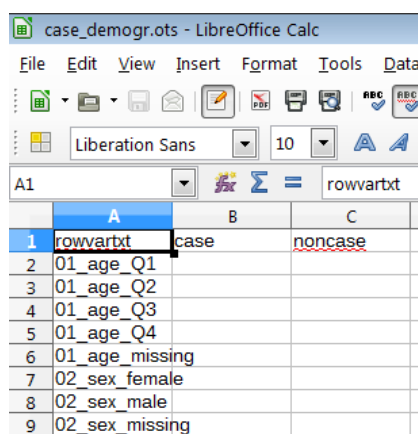
After you have exhausted your creativity, you may get something like:

#### Report from the National Tuberculosis Program

#### Characteristics of tuberculosis patients by microscopy-defined case status

Characteristic	Smear-positive cases		Other cases	Total	Col %
	n	Row %			
Total	13	86.7	2	15	100.0
Age quartiles					
Quartile 1	3	100.0	0	3	20.0
Quartile 2	3	75.0	1	4	26.7
Quartile 3	1	50.0	1	2	13.3
Quartile 4	6	100.0	0	6	40.0
Age missing	0	NA	0	0	0.0
Sex					
Female	4	100.0	0	4	26.7
Male	9	81.8	2	11	73.3
Sex missing	0	NA	0	0	0.0

After being happy with your formatting (don't overdo it now, you will have more opportunity in the upcoming task), we return to the input sheet, delete all numbers (leave the labels) and put the cursor into cell A1:



Make sure you save it (it will still be saved as the \*.ots template file), and then close LibreOffice Calc (**ALT+F4** closes the program).

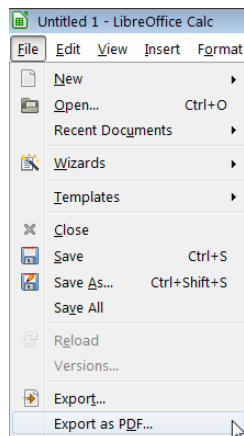
To test, run the program again, make some other interval selection, and then click the LibreOffice icon to open the template.

**Note: LibreOffice is a slow opener, much slower to open than Microsoft Office. Be patient, do not click the icon more than once: either it works with one click or it will not work with 100 clicks, but multiply clicking can cause conflicts and will surely slow it down even further!**

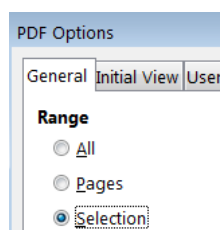
The last thing to show, is the inbuilt possibility to make a PDF file. If you have run your program, and pasted the EpiData Analysis browser content into the input file, switch to the output file and mark all fields:

	A	B	C	D	E	F	G
1	Characteristics of tuberculosis patients by microscopy-defined case status						
2							
3	Characteristic	Smear-positive cases		Other cases	Total	Col %	
4		n	Row %				
5	Total	5,058	72.5	1,921	6,979	100.0	
6	Age quartiles						
7	Quartile 1	1,014	61.8	627	1,641	23.5	
8	Quartile 2	1,365	74.6	465	1,830	26.2	
9	Quartile 3	1,367	77.6	395	1,762	25.2	
10	Quartile 4	1,312	75.2	432	1,744	25.0	
11	Age missing	0	0.0	2	2	0.0	
12	Sex						
13	Female	1,408	65.9	728	2,136	30.6	
14	Male	3,650	75.4	1,192	4,842	69.4	
15	Sex missing	0	0.0	1	1	0.0	

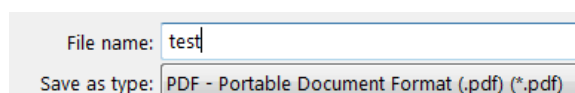
In File you have Export as PDF:



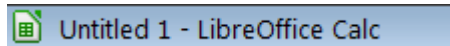
Clicking it, you need to choose Selection under the General tab:



and save the selection in some folder with your favorite name:



Then go there to view it. Because the access was to a template file, this spreadsheet is still untitled:



You can save it (as a LibreOffice \*.ods spreadsheet file or also as a Microsoft Office Excel™ file).

***Task:***

- o Complete the EpiData Analysis program file “case\_demogr.pgm” to make an output for all four variables i.e. adding site of disease and patient category. Then upgrade the LibreOffice template file “case\_demogr.ots” and create the output as a PDF file.*
- o Write a “cleaning” program that ensures that all temporary session files are erased.*



## Solution to Exercise 5: Formatting standardized analysis output in a spreadsheet

### Key points:

- Cross-tables in analysis may have varying dimensions, depending on how many strata have counts
- A work-around to obtain fixed dimensions is the use of merging the aggregated data with a dummy set in which there is one record for each possible column-row combination
- Output from multiple tables can be stacked in a \*.rec file after aggregation and a and pasted into a preformatted spreadsheet for a formatted output

### Tasks:

- o Complete the EpiData Analysis program file “case\_demogr.pgm” to make an output for all four variables i.e. adding site of disease and patient category. Then upgrade the LibreOffice template file “case\_demogr.ots” and create the output as a PDF file.*
- o Write a “cleaning” program that ensures that all temporary session files are erased.*

The solution is the entire menu package. The PDF exported spreadsheet output is shown on the following page.

## Characteristics of microscopy confirmed and other cases

Characteristic	Confirmed cases		Other cases	Total	Col %
	N	Row %			
Total	5,058	72.5	1,921	6,979	100.0
Age quartiles					
Quartile 1	1,014	61.8	627	1,641	23.5
Quartile 2	1,365	74.6	465	1,830	26.2
Quartile 3	1,367	77.6	395	1,762	25.2
Quartile 4	1,312	75.2	432	1,744	25.0
Age missing	0	0.0	2	2	0.0
Sex					
Female	1,408	65.9	728	2,136	30.6
Male	3,650	75.4	1,192	4,842	69.4
Sex missing	0	0.0	1	1	0.0
Disease site					
Pulmonary	5,030	85.6	848	5,878	84.2
Extrapulmonary	23	2.1	1,072	1,095	15.7
Site missing	5	83.3	1	6	0.1
Patient category					
New	4,366	74.5	1,491	5,857	83.9
Relapse	502	97.9	11	513	7.4
After failure	64	100.0	0	64	0.9
After default	22	91.7	2	24	0.3
Transfer in	76	52.4	69	145	2.1
Other	8	14.3	48	56	0.8
Category missing	20	6.3	300	320	4.6