

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:



Quarterly reports on case finding and treatment outcome



Enter data



Access EpiData Entry

Create reports



A quarterly report on case-finding



A quarterly report on treatment outcome

Terminate session



Erase temporary session files



Exit the report menu

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:

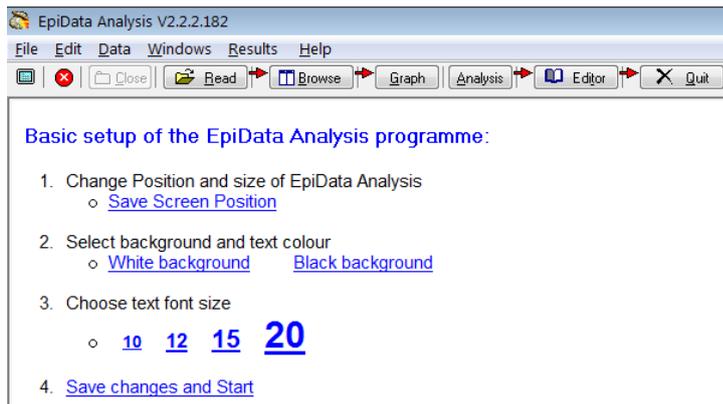
```
start_template.htm
```

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`

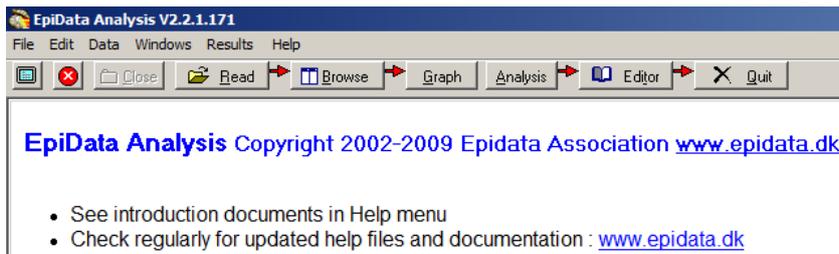
If we double-click this file it opens in our default browser and this is the display we get:

EpiData Analysis Copyright 2002-2009 Epidata Association www.epidata.dk

- See introduction documents in Help menu
- Check regularly for updated help files and documentation : 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:

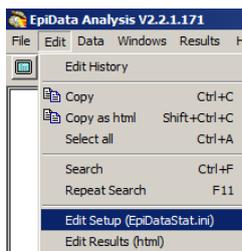


The upper component:



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) than 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
	

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:

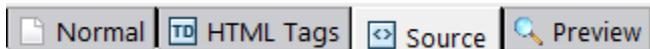
```
KompoZer.exe
```

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.

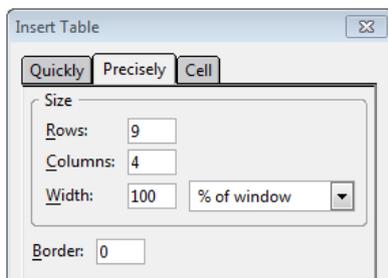
The first thing is to save it as:

`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”:

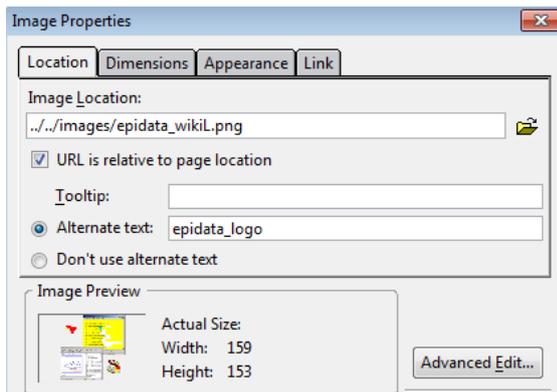


and set the Border to “0”. The default for “Border” is 1: a line around each cell is displayed in the browser. Setting it to “0” allows entering the information into cells instead of using the Tab key (which we cannot do properly in an HTML page), but the user does not see any line and remains unaware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

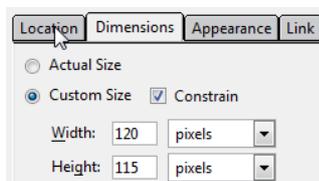
If the table cell background is not white, you may have to fiddle with the options for them to render them white (non-transparent). The best way is to change the background as follows:

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”)

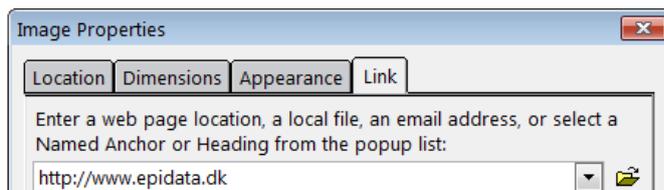
In the most upper left cell we insert one of the provided EpiData logos (the `epidata_wikiL.png` file from the `\IMAGES` sub-folder) using the “Insert” menu and also supply a (required) alternate text for the name:



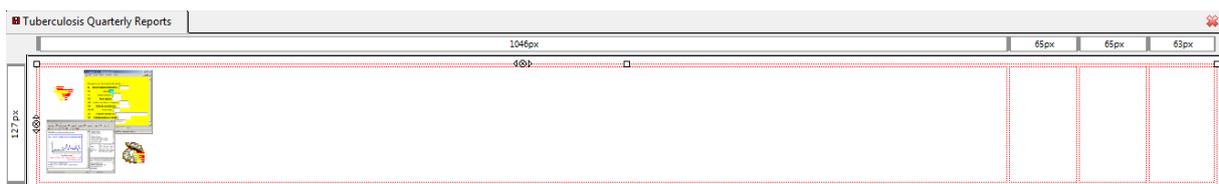
Then go to the tab “Dimensions” and decrease the current size of the width to 120 pixels:



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:

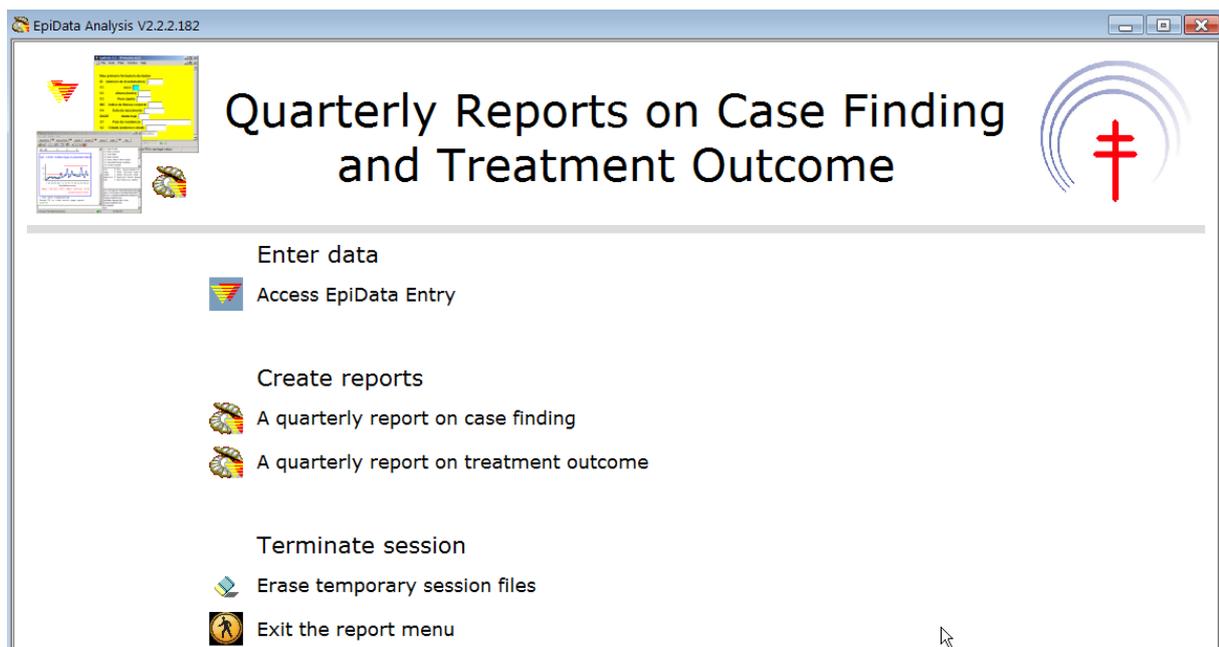


Continue adding text, formatting it, and add the appropriate icons into the appropriate places until you have the complete lay-out:

	<h1>Quarterly Reports on Case Finding and Treatment Results</h1>	
	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><tr></code>	<code></tr></code>
Cell	<code><td></code>	<code></td></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 . 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 `` 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 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;"></td><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.

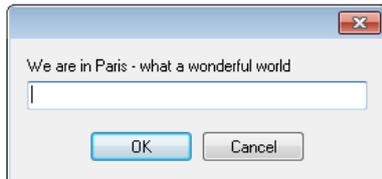
In addition, copy manually (not here in the program file) the file “side_by_side_graphs.html” into the \temp folder.

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"

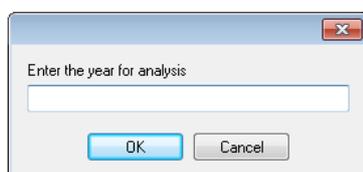
gen i regyy=year(regdate)
define yr ##### global
cls

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:

Sex of examinee	N
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 +		Male	Female	Total
M	F	M	F	M	F	M	F	M	F	M	F	M	F			

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.