# Part E.  Beyond EpiData Analysis using R

## Part E: Beyond EpiData Analysis using R

## Introductory note

Part E addresses how to conduct a multivariable analysis for two situations for which EpiData Analysis is not designed to be operational.  With our philosophy that there should preference be given to outstanding freely available software, we chose to introduce the open-source package R rather than any of the proprietary software solution packages.

The first exercise introduces the bare-bone basics of R software with a few simple exercises to get a feel for its working.  Recommended texts for self-study are (available on the course CD-ROM):

Aragón T J.  Applied epidemiology using R.  University of California, Berkeley School of Public Health, and the San Francisco Department of Public Health. Version 14 October 2013, available from http://medepi.com/courses/applied-epi-using-r/.

Lumley T and the R Core Development Team and UW Department of Biostatistics.  R fundamentals and programming techniques. Version Birmingham, 2006-2-27/28, available from http://faculty.washington.edu/tlumley/Rcourse/R-fundamentals.pdf.

Venables W N, Smith D M and the R Core Team.  An introduction to R.  Notes on R: a programming environment for data analysis and graphics, Version 3.1.0 (2014-04-10), available from: http://cran.r-project.org/doc/manuals/R-intro.pdf.

The second exercise extends on the basics by introducing data bases and functions.  The recommended text for self-study is (available on the course CD-ROM):

Myatt M.  Open source solutions – R.  Nordic School of Public Health and University of Tartu.  Computer software in epidemiology / statistical practice in epidemiology.  Version 16 May 2005, available from http://www.brixtonhealth.com.

The third exercise gives the first application, the use of logistic regression to determine predictors that include continuous variables and variables with more than two levels.  The recommended text for self-study is (available in the course material):

Manning C.  Logistic regression (with R).  Version 4 November 2007, available from http://nlp.stanford.edu/~manning/courses/ling289/logistic.pdf.

The fourth exercise gives the second application, survival analysis where stratification with the Kaplan-Meier approach does no more suffice and adjustments are preferred by using a Cox proportional hazard model. The recommended texts for self-study are (available in the course material):

Tableman M. Survival analysis using S/R. Unterlagen für den Weiterbildungsgang in angewandter Statistik an der ETH Zürich. [*Note: despite the German subtitle, the document is entirely in English, except for the subtitle which refers to the course at the Swiss Federal Institute of Technology in Zurich*]. Version August-September 2012. It is an excerpt from the book *Survival Analysis Using S: Analysis of Time-to-Event Data* by Mara Tableman and Jong Sung Kim, published by Chapman & Hall/CRC, Boca Raton, 2004.

Therneau T M. Package 'survival'. Version 2.37-7, July 2, 2014, available from http://cran.r-project.org/web/packages/survival/index.html.

**Acknowledgments:**

We used the following data in the exercises:

In **Exercise 1**, we use an example given by Altman in:

Altman D G, Machin D, Bryant T N, Gardner M J. Statistics with confidence. 2nd edition. 2 ed. Bristol: BMJ Group; 2000.

In **Exercises 2, 3, and 4** we use data from:

Aung K J M, Van Deun A, Declercq E, Sarker M R, Das P K, Hossain M A, Rieder H L. Successful "9-month Bangladesh regimen" for multidrug-resistant tuberculosis among over 500 consecutive patients. Int J Tuberc Lung Dis 2014;18: in press.

We are grateful to Drs Aung, Declercq, and Van Deun to grant permission to use an excerpt of individual data from the original dataset.

Nguyen Binh Hoa, Ajay M V Kumar, and Zaw Myo Tun went through the first draft of the exercises and gave valuable input into the development of Part E.

# Exercise 1: Introduction to R software

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

a. Have a basic understanding of how R works and to expand on this basis in a self-learning process

b. Be informed about resources on R software

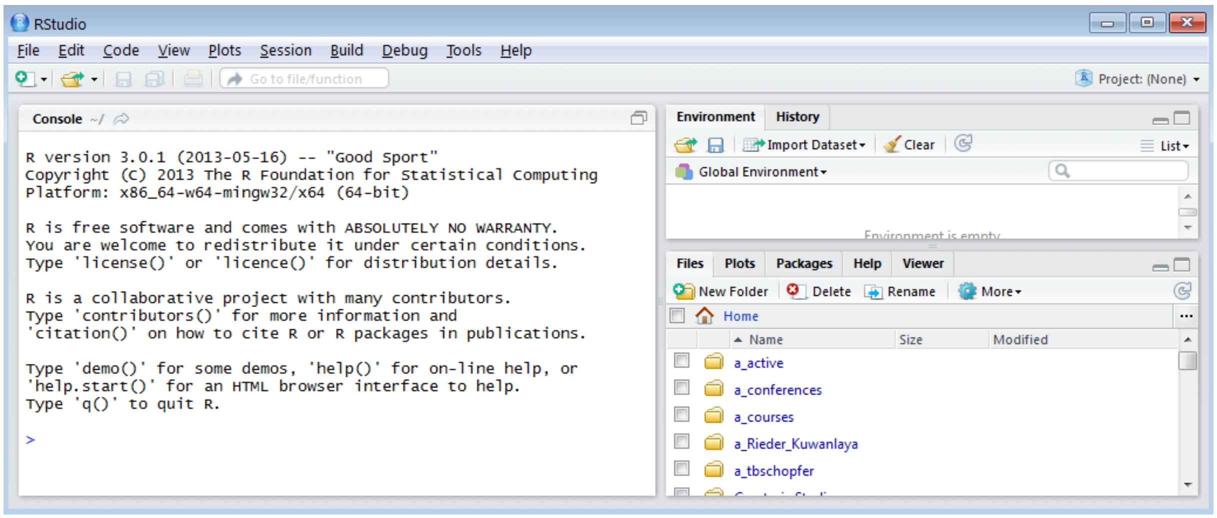
R is a language and environment for statistical computing and graphing. It provides a suite of numerous packages for a virtually unlimited number of applications, including the most powerful tools that an epidemiologist might need. It has a large community of collaborators and contributors, it is entirely open to everybody who wants to use it and who wishes to contribute.

The starting point on the internet is the “The R Project for Statistical Computing” at its web site: `http://www.r-project.org/`. This is from where you access one if its mirror sites to download the software.

In tandem with R, we will also use RStudio, also open-source and free, a powerful and productive user interface for R. It can be found for download at its web site: `http://www.rstudio.com/`.

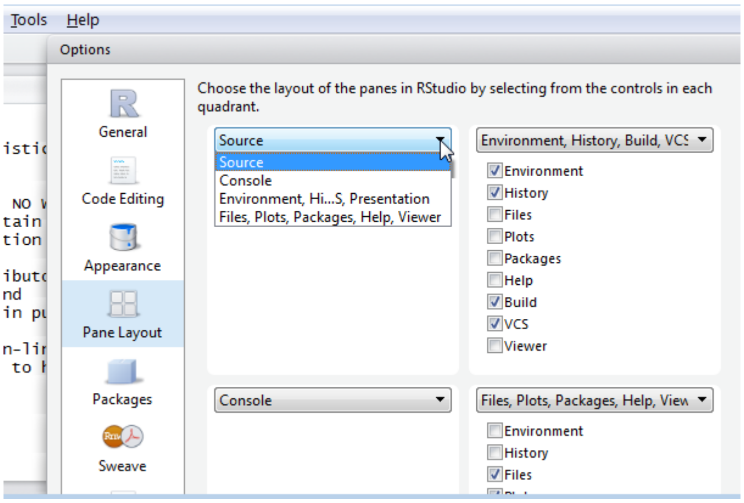
If your course comes on a CD-ROM, you find the two software packages on your CD-ROM, but as always with software, we strongly recommend that you obtain it from its original source on the internet.

Install R on your computer and then install RStudio. If you open RStudio, you see that it is divided into a left-hand-sided panel, and a split right-handed panel:



```
R version 3.0.1 (2013-05-16) -- "Good Sport"
Copyright (C) 2013 The R Foundation for Statistical Computing
Platform: x86_64-w64-mingw32/x64 (64-bit)

R is free software and comes with ABSOLUTELY NO WARRANTY.
You are welcome to redistribute it under certain conditions.
Type 'license()' or 'licence()' for distribution details.

R is a collaborative project with many contributors.
Type 'contributors()' for more information and
'citation()' on how to cite R or R packages in publications.

Type 'demo()' for some demos, 'help()' for on-line help, or
'help.start()' for an HTML browser interface to help.
Type 'q()' to quit R.

>
```

In Tools | Options we can adjust to the preferred layout:



Choose the layout of the panes in RStudio by selecting from the controls in each quadrant.

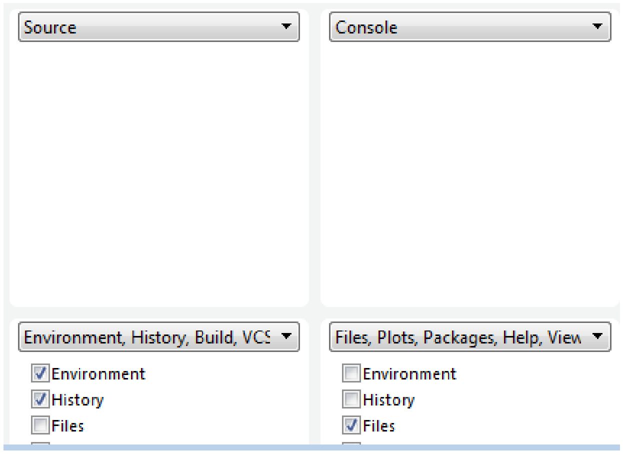
Source
- Source
- Console
- Environment, Hi...S, Presentation
- Files, Plots, Packages, Help, Viewer

Environment, History, Build, VCS
- [✓] Environment
- [✓] History
- [ ] Files
- [ ] Plots
- [ ] Packages
- [ ] Help
- [✓] Build
- [✓] VCS
- [ ] Viewer

Console

Files, Plots, Packages, Help, Viewer
- [ ] Environment
- [ ] History
- [✓] Files

We chose here (somewhat arbitrarily) to have the Source on the top left, the Console on the top right, the Environment, History, ... on the bottom left, and the Files, Plots, Packages, Help on the bottom left. and get altogether four quadrants:



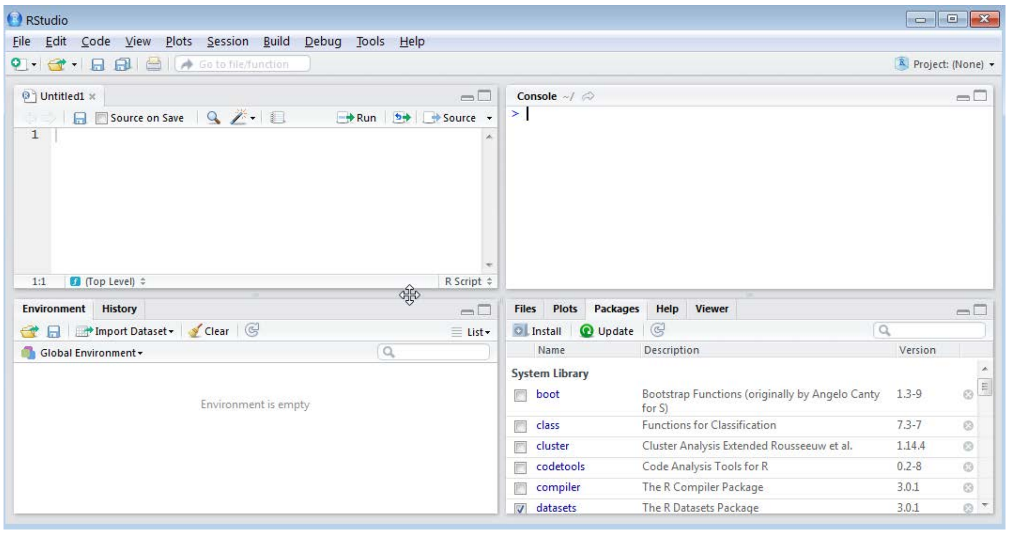
Source

Console

Environment, History, Build, VCS
- [✓] Environment
- [✓] History
- [ ] Files

Files, Plots, Packages, Help, Viewer
- [ ] Environment
- [ ] History
- [✓] Files

So that after these adjustments we get:

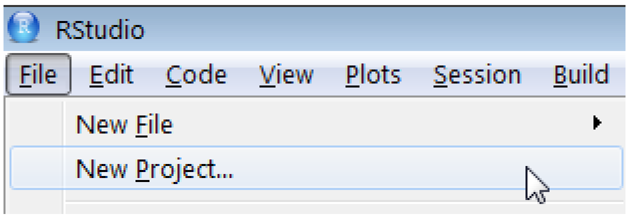


| | Name | Description | Version |
|---|---|---|---|
| | System Library | | |
| [ ] | boot | Bootstrap Functions (originally by Angelo Canty for S) | 1.3-9 |
| [ ] | class | Functions for Classification | 7.3-7 |
| [ ] | cluster | Cluster Analysis Extended Rousseeuw et al. | 1.14.4 |
| [ ] | codetools | Code Analysis Tools for R | 0.2-8 |
| [ ] | compiler | The R Compiler Package | 3.0.1 |
| [✓] | datasets | The R Datasets Package | 3.0.1 |

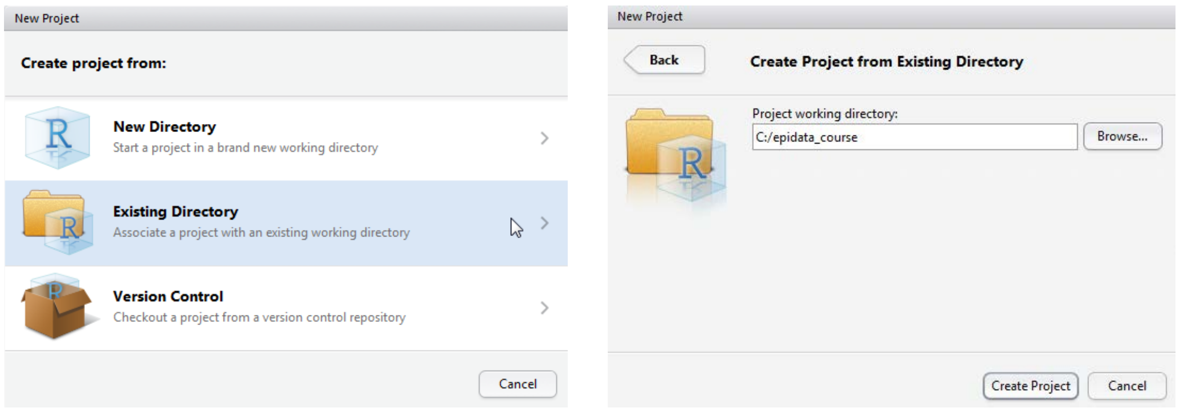
We cleared the screen of the Console by using the shortcut **CTRL+L**. To switch between the Console and the Source editor quadrants use **CTRL+1** to get to the Source editor, and **CTRL+2** to get to the Console.

Of course, you are free to arrange it at your liking, but for the time being in this course, we will refer to this arrangement.

First we Create project from an Existing directory:

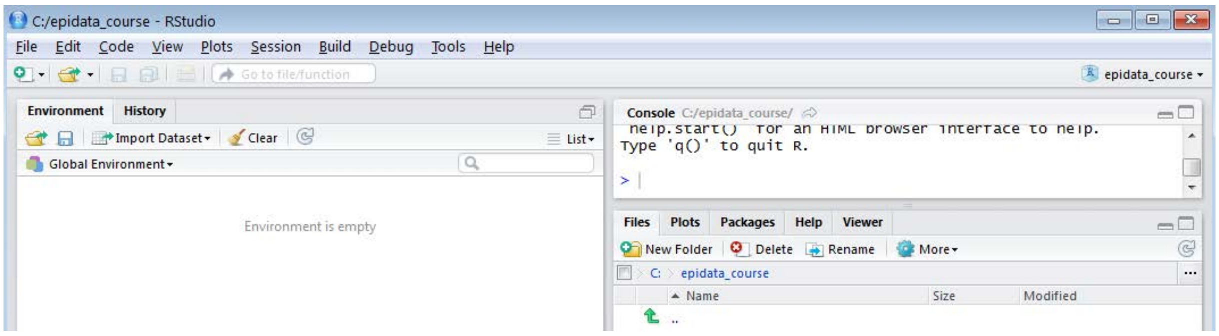


and browse for the `EPIDATA_COURSE` project folder:


New Project
Create project from:
New Directory
Start a project in a brand new working directory
Existing Directory
Associate a project with an existing working directory
Version Control
Checkout a project from a version control repository
Cancel
New Project
Back
Create Project from Existing Directory
Project working directory:
C:/epidata_course
Browse...
Create Project
Cancel

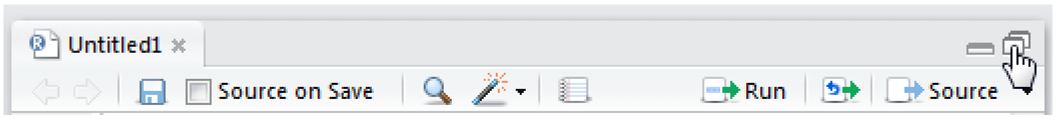
You note on the left top:

and note that the left side has been collapsed to show the environment only and on the right side we are in Files:



```
help.start() for an HTML browser interface to help.
Type 'q()' to quit R.

> |
```

If you click on the toggle icon above Source:



You see the split screen again.

Note also that RStudio and R use forward slashes “C:/epidata_course/” in the Console. RStudio actually created a sub-folder “`.Rproj.user\`” which is not public and by default hidden from view.

We are therefore all set to go. To introduce the workings of R we will create a two-by-two table as in a case-control study and calculate the odds ratio with a Woolf confidence interval.

## A case-control study example

Our reference text for statistical calculations like for EpiData Analysis is, whenever possible, the text by Altman and collaborators:

> Altman D G, Machin D, Bryant T N, Gardner M J. Statistics with confidence. 2nd edition. Bristol: BMJ Group, 2000: pp 61-62.

The odds ratio is defined as the odds of exposure among cases divided by the odds of exposure among controls. Woolf’s logit method uses normal approximation to the distribution of the logarithm of the odds ratio in which the standard error is

$\mathrm{SE}(\log_e OR) = \mathrm{SQRT}(1/a + 1/b + 1/c + 1/d)$

To obtain the 95% confidence interval, we calculate the quantities:

$95\%\ \mathrm{CI} = \log_e OR \pm 1.96 * \mathrm{SE}(\log_e OR)$

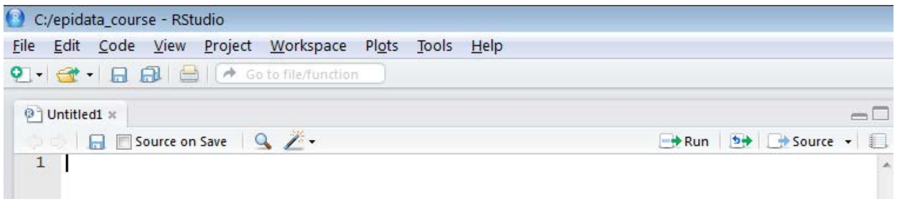
By exponentiating, we get the 95% confidence interval.

To enable verification of our numbers, we are going to use the same example as Altman uses in table 7.4 for the worked example:

| ABO non-secretor state | Study group | | |
|---|---|---|---|
| | Cases | Controls | Total |
| Yes | 54 | 89 | 143 |
| No | 60 | 245 | 305 |
| Total | 114 | 334 | 448 |

We will be writing our commands into the upper left quadrant (Source editor). When we run it, it will show in the right upper quadrant (the Console). Objects that we create will be found in the left lower quadrant.

In the Source editor, you should now have the empty workspace like any text editor:



Type:

```
a <- 54
```

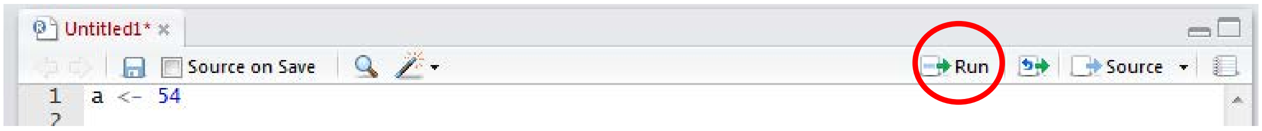
“a” is what we call an “object” to which we assign (using “<-”) the value 54. Some people say “Everthing is an object in R”. There is (incomplete, though) truth to this. Anything and everything can be assigned to an object. Here we assign a number to an object “a”. We can name our objects whatever we want them to be called:

```
A
a
a.plus.b
```

but note importantly, they are case-sensitive: “A” is not the same as “a” in R. Another important note to make is that the value of an object can be overwritten without any warning.

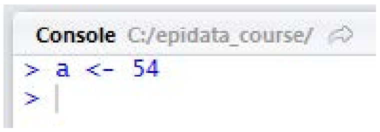
You probably type the lesser sign “<”, followed by the minus sign “-”. RStudio allows you to use **ALT+-** alone (the ALT key plus the minus sign) and you get the “assign” symbol combination including the spaces that we often like to make in R.

Nothing happened when you hit the carriage return key and this is how it is supposed to be in the Source editor. You only type here your stuff. To execute it, you need to tell it to do it. It is like writing commands in an EpiData Analysis program, and like in the latter you have:



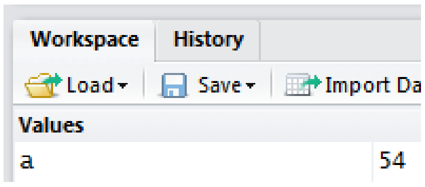
```
a <- 54
```

a Run command: anywhere on a command line, it runs the current line, marking whatever you wish, it runs the marked section, nothing new here for the seasoned EpiData user. If we run this line, two things happen: in the right upper quadrant, the Console, you get:



```
> a <- 54
> 
```

This is confirmation that the command Run was executed. In the left lower quadrant, the Values, you get:



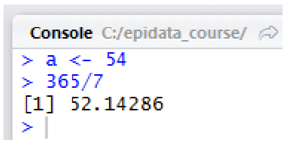
```
Values
a                    54
```

As we are going to use the Run command often, it is useful to know that the shortcut **CTRL+Enter** accomplishes the same thing as Run. **CTRL+Shift+Enter** runs the entire script.

All objects at your disposal will be listed here. In the Console, the value of object “a” is not visible. Some things you do will give you the result immediately, others will not, this assignment obviously belongs to the latter category. But type into the second line below the assignment:

```
365/7
```

and Run it. What you get now in your console is:

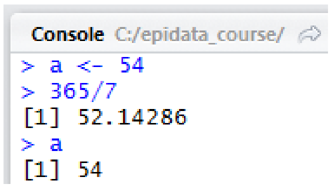


```
> a <- 54
> 365/7
[1] 52.14286
>
```

both the command you wrote and the result from the calculation. If you write into your working space:

a

and Run it you get:



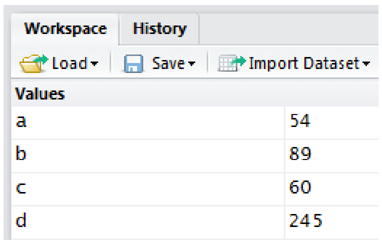
```
> a <- 54
> 365/7
[1] 52.14286
> a
[1] 54
```

**CTRL+L** clears the Console.

Let's do the other assignments, writing each on a separate line:

```
b <- 89
c <- 60
d <- 245
```

then Run after marking. You should now have all four objects visible in the lower left quadrant:



| Values | |
|---|---|
| a | 54 |
| b | 89 |
| c | 60 |
| d | 245 |

While we have our four objects that make up the core of the 2-by-2 table, we would like to have the complete table including its marginals, and all assigned to one object.

To this end we will first create a **vector**, that is a one-dimensional object containing the values of a, b, and their sum. To keep with intuition, we will call this object "ab". Type:

```
ab <- c(a, b, a+b)
```

The "c" stands for "concatenate", that is putting a string consisting of sub-strings together. The elements of the vector are in parenthesis, separated by commas. They are, in the order in which we want to have them, our created two objects a and b, and the third element the sum of the two.

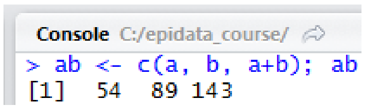
To see the result, type

```
ab
```

on a new line. Alternatively, you could also put a series of commands on one single line, separated by semi-colons, like in:

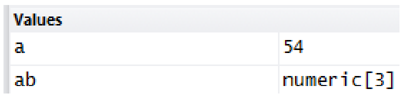
```
ab <- c(a, b, a+b); ab
```

and you should get:



```
Console C:/epidata_course/
> ab <- c(a, b, a+b); ab
[1]  54  89 143
```

Note, that in the left lower quadrant, you get now the object `ab` and a definition of what it is:

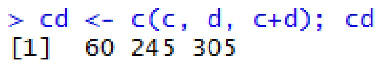


| Values | |
|---|---|
| a | 54 |
| ab | numeric[3] |

Analogously, we create the second vector for the non-exposed:

```
cd <- c(c, d, c+d)
```

and get accordingly:

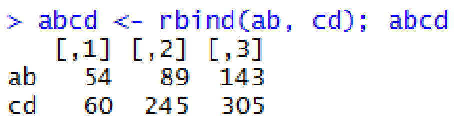


```
> cd <- c(c, d, c+d); cd
[1]  60 245 305
```

We have thus two vectors, one for the exposed and another for the non-exposed. Now we want them to be together in one single object that we will call `abcd`. The command to "bind" two rows together is:

```
abcd <- rbind(ab, cd); abcd
```

and we get:



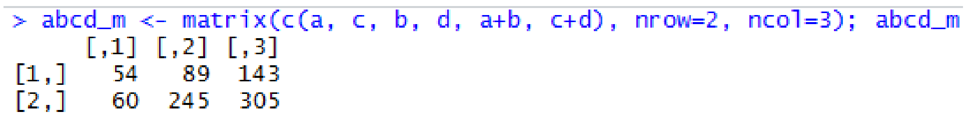
```
> abcd <- rbind(ab, cd); abcd
   [,1] [,2] [,3]
ab   54   89  143
cd   60  245  305
```

The "`rbind`" is for "row binding". Of course, if there is "`rbind`", there must also be a "`cbind`" for column binding. Note also on the top `[,1]`, `[,2]` and `[,3]`: R labels (missing an actual label) things in a Cartesian way as `[row, col]`. R writes the indices to designate position in square brackets. As the column headers don't have a row designation, it is thus the comma followed by the column number. What might be the internal designation for the location of the number `54`? Yes, it is `[1,1]` for first row, first column, while `305` is in position `[2,3]`.

We have combined two **vectors** (one-dimensional objects) and the result is a **matrix** (a two-dimensional object). We could have created a matrix directly from the four objects `a`, `b`, `c`, and `d`. However, we have to pay attention to the correct the sequence:

```
abcd_m <- matrix(c(a, c, b, d, a+b, c+d), nrow=2, ncol=3); abcd_m
```

noting that the `matrix` function reads the data column-wise (!), we get:



```
> abcd_m <- matrix(c(a, c, b, d, a+b, c+d), nrow=2, ncol=3); abcd_m
     [,1] [,2] [,3]
[1,]   54   89  143
[2,]   60  245  305
```

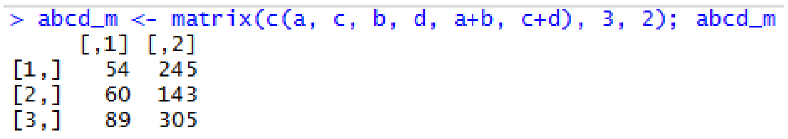
We don't need to write "nrow=2, ncol=3" to tell R that these 6 objects must be allocated to 2 rows and 3 columns, it suffices to stated:

```
abcd_m <- matrix(c(a, c, b, d, a+b, c+d), 2, 3); abcd_m
```

as R understands the first assignment ("`2`") to be for the number of rows, and the second ("`3`") for the number of columns. If we don't pay attention to the default sequence row-column:

```
abcd_m <- matrix(c(a, c, b, d, a+b, c+d), 3, 2); abcd_m
```

we get a solution, but not the intended one:

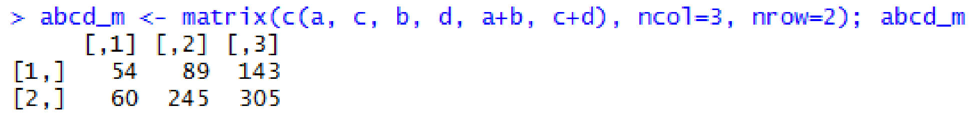


```
> abcd_m <- matrix(c(a, c, b, d, a+b, c+d), 3, 2); abcd_m
     [,1] [,2]
[1,]   54  245
[2,]   60  143
[3,]   89  305
```

In contrast, if we specify:

```
abcd_m <- matrix(c(a, c, b, d, a+b, c+d), ncol=3, nrow=2); abcd_m
```

we will get what we intend:



```
> abcd_m <- matrix(c(a, c, b, d, a+b, c+d), ncol=3, nrow=2); abcd_m
     [,1] [,2] [,3]
[1,]   54   89  143
[2,]   60  245  305
```

Thus, as beginners, not yet fully conscious of default sequences, it is safer to be overly explicit.

To get back to the matrix: A matrix is a “two-dimensional table of like elements” (Aragón). Remove the object `abcd_m` as we are not going down this way for the time being:

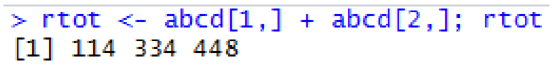
```
rm(abcd_m)
```

where `rm` is the command to remove an object which is specified within the following paranethesis.

To complete the table, we need to make a row that is the sum of the first and the second row. We will call it `rtot` and it must be:

```
rtot <- abcd[1,] + abcd[2,]; rtot
```

because we are adding values of the respective column in row 1 `[1,]` to the values in row 2 `[2,]`. The result:

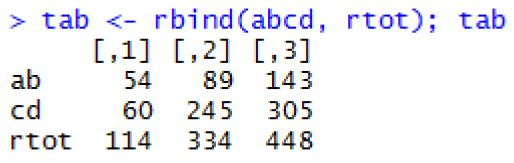


```
> rtot <- abcd[1,] + abcd[2,]; rtot
[1] 114 334 448
```

The last thing we have to do to get our complete table with all the marginals is row binding to an object that we will call `tab`:

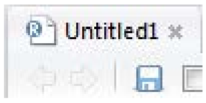
```
tab <-  rbind(abcd, rtot); tab
```

giving:



```
> tab <- rbind(abcd, rtot); tab
     [,1] [,2] [,3]
ab     54   89  143
cd     60  245  305
rtot  114  334  448
```

Before we lose our hard work, let’s save what we typed so far in our editing window (the Source as we named the upper left quadrant initially) as “`e_ex01_or.r`” (by going through File | Save or by clicking the diskette icon). To our knoweldge, R is not very particular what extension we give things, but we keep anyhow to what is quite widely common usage, and that is to give the extension `*.r`. You note how the file name replaces the Untitled:



=>

While we have everything we need, we could make it a bit more appealing with appropriate column an row labeling which is done as:

```
colnames(tab) <- c(“Case”, “Control”, “Total”)
rownames(tab) <- c(“Exp+”, “Exp-”, “Total”)
```

and check to verify with:

```
tab
```

```
      Case Control Total
Exp+    54      89   143
Exp-    60     245   305
Total  114     334   448
```

“`colnames`” and “`rownames`” are the respective commands to give labels to columns and rows, while what follows in the parenthesis `(tab)` is the object to which the command applies.

We can also label the dimensions:

```
names(dimnames(tab)) <- c("Exposure", "Status"); tab
```

and get:

```
        Status
Exposure Case Control Total
   Exp+    54      89   143
   Exp-    60     245   305
   Total  114     334   448
```

So far so good, but all we have is input, but what we really need are calculations. The odds ratio should be simple enough, we need to define the location by row and column of `a`, `b`, `c`, and `d` to then calculate `(a/c)/(b/d)`, thus:

```
or <- (tab[1,1]/tab[2,1])/(tab[1,2]/tab[2,2]); or
```

Note again here the brackets (not parentheses): whenever something needs to be indexed in R, brackets are used, here for the Cartesian positioning `[row, col]` but also if one wishes to describe the position of a record.

We specify the object `tab` and the position of its numeric elements to get the correct calculation, and the result should be:

```
2.477528
```

For the calculation of the 95% confidence interval (CI) we do this preferably in more than one step, first calculating the standard error `se`:

```
se <- sqrt(1/a+1/b+1/c+1/d)
```

We take here directly our four input objects as this calculation is independent of the position in the table. Next, for the lower and upper 95% CI:

```
or.ci.lower <- exp(log(or)-1.96*se)
or.ci.upper <- exp(log(or)+1.96*se)
```

Let’s remove the objects that we don’t need, the command to remove objects is `rm`:

```
rm(ab, cd, abcd, rtot, se)
```

Finally, let's get an output that summarizes it all together:

```
print(tab)
```

```
cat("\nOR:", round(or, digits=3), "\n95% CI:", round(or.ci.lower,
     digits=3), "to ", round(or.ci.upper, digits=3))
```

With the function "`cat`" we avoid quotes where we don't need them, and then take control by writing them where we need them:

```
x <- c("Ajay", "Hoa", "Zaw")
x
```

gives:

```
"Ajay" "Hoa" "Zaw"
```

but

```
cat(x)
```

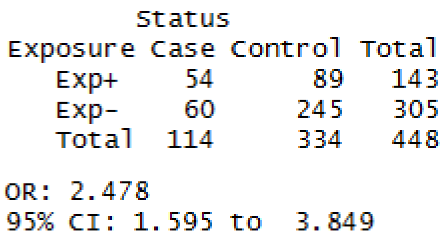
gives:

```
Ajay Hoa Zaw
```

The function "`\n`" is equivalent to a hard carriage return.

The function `round` is followed by the object, then a comma, then the number of digits to which it is rounded, all in parentheses as is usual with functions.

Note that it is best if you write everything of the second item (starting with "`cat`") on a single line. We should get:



```
         Status
Exposure Case Control Total
   Exp+    54      89   143
   Exp-    60     245   305
   Total  114     334   448

OR: 2.478
95% CI: 1.595 to  3.849
```

Verifying by comparison with the results in Altman's shows that what we obtained is correct.

To make this more generically useful, we remove now all non-essential things, first of all the input objects `a`, `b`, `c`, and `d`. While not necessary (any object can be overwritten without warning), it is intuitively more appealing if we just keep the rest and then enter new values for `a`, `b`, `c`, and `d` and then run our "stripped" `e_ex01_or.r`. Thus, the stripped `e_ex01_or.r` may look like (note the hash "#" symbol at the beginning of the line correspondes to the asterisk "`*`" in EpiData Analysis denoting a non-executed comment):

```
# Complete 2 by 2 table from
# a, b, c, d and calculate odds ratio wih Wolf CI
# Assign below values for a, b, c, and d as in:
# a <- 100

ab <- c(a, b, a+b)
cd <- c(c, d, c+d)
abcd <- rbind(ab, cd)
```

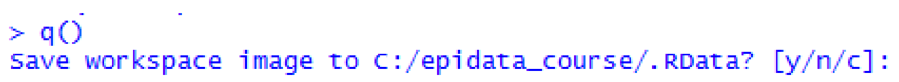
```
rtot <- abcd[1,]+abcd[2,]
tab <- rbind(abcd, rtot)
colnames(tab) <- cbind("Case", "Control", "Total")
rownames(tab) <- rbind("Exp+", "Exp-", "Total")
names(dimnames(tab)) <- c("Exposure", "Status"); tab
or <- (tab[1,1]/tab[2,1])/(tab[1,2]/tab[2,2])
se <- 1.96*sqrt(1/a+1/b+1/c+1/d)
or.ci.lower <- exp(log(or)-se)
or.ci.upper <- exp(log(or)+se)

print(tab)
cat("\nOR:", round(or, digits=3), "\n95% CI:", round(or.ci.lower,
digits=3), round(or.ci.upper, digits=3), "\n")
```

You would now assign any value to your input objects `a`, `b`, `c`, and `d`, for instance directly in the console and then open this script and `Run` it, and there it will be.

To quit R, type:

```
q()
```

You will be prompted whether you wish to save the workspace image:



```
> q()
Save workspace image to C:/epidata_course/.RData? [y/n/c]:
```

to which you can safely answer with “`n`”: we have saved our scripts and do not need to save in addition the log file.

***Task***

***In analogy to the calculation of the odds ratio, write an R script* `e_ex01_rr.r` *that calculates the relative risk from two proportions (thus not a ratio of incidence rates, but a ratio of two prevalence proportions) for a study used in Altman’s textbook. It has the isolation of* Helicobacter pylori *as the outcome and the history of an ulcer in the mother as the exposure. We use the following set-up of notations:***

| Exposure | Outcome | | |
|---|---|---|---|
| | Ill | Healthy | Total |
| Yes | A | B | A+B |
| No | C | D | C+D |
| Total | A+B | B+D | A+B+C+D |

***The data provided by Altman in table 7.2 (page 59) are as follows:***

| Mother with a history of ulcer | *H pylori* isolated | | |
|---|---|---|---|
| | Yes | No | Total |

| | | | |
|---|---|---|---|
| Yes | 6 | 16 | 22 |
| No | 112 | 729 | 841 |
| Total | 118 | 745 | 863 |

***The relative risk is calculated by:***

```
R = [A/(A+B)]/[C/(C+D)]
```

***The standard error of the log$_e$R is:***

```
se = SQRT(1/A-1/(A+B)+1/C-1/(C+D))
```

***The 95% CI is thus:***

```
95%CI = exp(logₑR ± 1.96*se)
```

# Solution to Exercise 1: Introduction to R software: basics

Key points:

a. R is both a language and an environment of computing
b. Anything and everything can be assigned to an object
c. Object names are case-sensitive
d. A vector is a one-dimensional object of like elements
e. A matrix is a two-dimensional table (another object) of like elements
f. Vectors can be bound to matrices

***Task***

***In analogy to the calculation of the odds ratio, write an R script* `e_ex01_rr.r` *that calculates the relative risk from two proportions (thus not ratio of incidence rates, but ratio of two prevalence proportions) for a study used in Altman's textbook. It has the isolation of*** **Helicobacter pylori** ***as the outcome and the history of an ulcer in the mother as the exposure. We use the following set-up of notations:***

| Exposure | Outcome | | |
|---|---|---|---|
| | Ill | Healthy | Total |
| Yes | A | B | A+B |
| No | C | D | C+D |
| Total | A+B | B+D | A+B+C+D |

***The data provided by Altman in table 7.2 (page 59) are as follows:***

| Mother with a history of ulcer | *H pylori* isolated | | |
|---|---|---|---|
| | Yes | No | Total |
| Yes | 6 | 16 | 22 |
| No | 112 | 729 | 841 |
| Total | 118 | 745 | 863 |

***The relative risk is calculated by:***

```
R = (A/(A+B))/(C/(C+D))
```

***The standard error of the log$_e$R is:***

```
se = SQRT(1/A-1/(A+B)+1/C-(1/(C+D))
```

***The 95% CI is thus:***

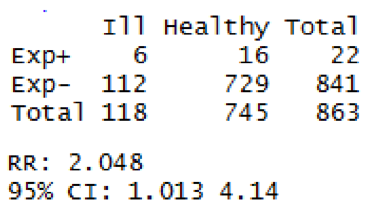
```
95%CI = exp(log_eR ± 1.96*se)
```

***Solution:***

The solution for the `e_ex01_rr.r` is a straight forward modification of the `e_ex01_or.r`:

```
# Complete 2 by 2 table from
# A, B, C, D and calculate relative risk with Wolf CI

A <- 54
B <- 89
C <- 60
D <- 245

AB <- c(A, B, A+B)
CD <- c(C, D, C+D)
ABCD <- rbind(AB, CD)
rtot <- abcd[1,]+abcd[2,]
tab <- rbind(ABCD, rtot)
colnames(tab) <- cbind("Ill", "Healthy", "Total")
rownames(tab) <- rbind("Exp+", "Exp-", "Total")
names(dimnames(tab)) <- c("Exposure", "Status")
se <- sqrt(1/A-1/(A+B)+1/C-1/(C+D))
rr <- (tab[1,1]/tab[1,3])/(tab[2,1]/tab[2,3]); rr
rr.ci.lower <- exp(log(rr)-1.96*se)
rr.ci.upper <- exp(log(rr)+1.96*se)

print(tab)
cat("\nRR:", round(rr, digits=3), "\n95% CI:", round(rr.ci.lower,
digits=3), round(rr.ci.upper, digits=3), "\n")
```

The result is:



```
      Ill Healthy Total
Exp+    6      16    22
Exp-  112     729   841
Total 118     745   863

RR: 2.048
95% CI: 1.013 4.14
```

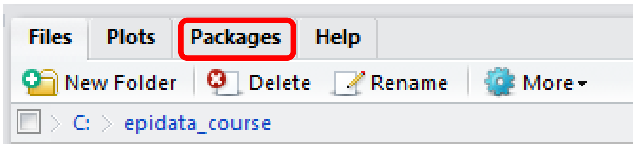
# Exercise 2: Introduction to R software: data bases and functions

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

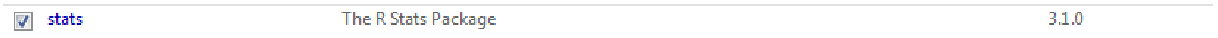
a. Know how to import a data base from another format
b. Create a function and apply it to an analytic problem

What we did in Exercise 1 is unusual in analysis: we entered aggregate data and analyzed them, while the usual currency of analysis is appropriate aggregation of individual observations. Commonly, datasets have been entered in other software as R is not a data entry tool. In the context of this course, data will come as an EpiData dataset with a `*.REC` and a `*.CHK` file. R accepts a multitude of data formats for import. In fact, a package has been developed that expands on the number of formats of datasets which can be imported into R, including for instance Epi Info / EpiData `*.REC` files and Stata `*.dta` files.

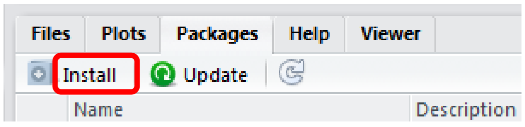
An important part of the versatility of R lies with the system of "packages". What we have been using is the basic package, but if we want to utilize the capability of importing EpiData `*.REC` files or Stata `*.dta` files, the package "**Foreign**" should be added to our existing setup. In the lower right quadrant in RStudio, you see:



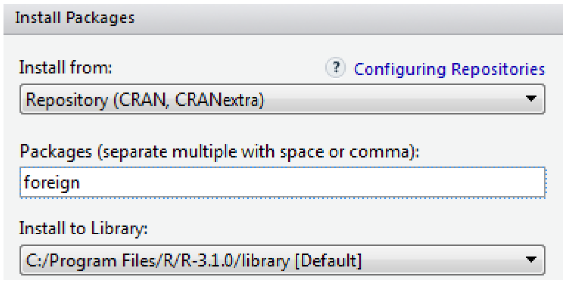
Click on Packages and you get a list of packages that are available with those loaded already ticked as for e.g. a basic statistical package:



The package we need is called foreign. You have to tick Install to get a menu:



You shouldn't need to do anything here except typing `foreign` into the line Packages and then Install as RStudio knows by definition where you have installed your copy of R. You need to be on the Internet though to get to the repository:



Install from: Repository (CRAN, CRANextra)

Packages (separate multiple with space or comma): foreign

Install to Library: C:/Program Files/R/R-3.1.0/library [Default]

Alternatively, you can use the included Zip package in the course material, but then you need to change the Install from and get the downloaded zip file. But as always, to ensure obtaining the most up-to-date version, get it whenever possible directly from the internet (http://cran.r-project.org/web/packages/foreign/index.html).

Having installed the package does not automatically put it at your disposal, you will have to call it. We start a new `*.r` script that we will name `e_ex02.r`.

In the required files on the course site you find the EpiData `REC/CHK` pair `e_ex02.rec` and `e_ex02.chk`. This is an abbreviated set of treatment results among patients with laboratory-confirmed multidrug resistance in Bangladesh.

The package `foreign` allows importing `*.REC` files with the command:

```
read.epiinfo("c:/epidata_course/e_ex02.rec")
```

However, this approach disregards the metadata (the `*.CHK` file) and only imports the field values from the `*.REC` file. We can write value labels in R, but we would first need to examine the `*.CHK` file and be careful about errors: tedious and error-prone. More efficient and safer is to use EpiData Entry 3.1 (or the EpiData Manager) to export the `e_ex02.*` file pair to a Stata `*.dta` file that contains both values and labels and then read the Stata file into R. Once you have exported the EpiData file pair to a Stata `e_ex02.dta` file, we are ready for import into R.

Being sticklers in labeling things, we start with a title, followed by actually invoking the now installed package **foreign**:

```
# Import an EpiData REC file

library(foreign)
e_ex02.dat <- read.dta("c:/epidata_course/e_ex02.dta")
```

Because we have made the "epidata_course" our project folder, it suffices to type:

```
e_ex02.dat <- read.dta("e_ex02.dta")
```

We read the Stata file by putting file name (and file path) in quotation marks inside a parenthesis (note the use of forward slashes) as in EpiData. We assign the imported file to an object that we will call `e_ex02.dat`. The `*.dat` is not required, it is just an object after all. We could call the object "`a`" if we wished to do so. The period separating the elements "`e_ex02`" and "`dat`" is not designating an extension, it is just one of the many ways R allows giving names to objects.

In a second step, we write the data to disk:

```
library(foreign)
e_ex02.dat <- read.dta("c:/epidata_course/e_ex02.dta")
write.table(e_ex02.dat, file="e_ex02.dat", row.names=TRUE)
```

Perhaps this is as good as any occasion to introduce the `Help` functions in R. If you know already "`write.table`", then it is easy, type:

```
help(write.table)
```

and you get in the lower right-hand corner box quite extensive information (try it out!).

To see what the import did, type:

```
e_ex02.dat[1:5,]
```

This gives the first 5 records:

```
  age        fq04    sex totobstime   agequart         agemed outcome07 outcome02
1  28 Susceptible   Male        999 Quartile 2   Below median      Cure   Success
2  45 Susceptible   Male        482 Quartile 4 Median or larger    Cure   Success
3  22 Susceptible Female        190 Quartile 1   Below median   Default   Failure
4  26 Susceptible Female         72 Quartile 2   Below median   Default   Failure
5  20 Susceptible Female       1000 Quartile 1   Below median      Cure   Success
                   pza02                  kmy02                  pth02              cxr02
1 PZA not known resistant KM not known resistant PTH not known resistant          Bilateral
2 PZA not known resistant KM not known resistant PTH not known resistant          Bilateral
3 PZA not known resistant KM not known resistant PTH not known resistant          Bilateral
4 PZA not known resistant KM not known resistant          PTH resistant Not known bilateral
5 PZA not known resistant KM not known resistant PTH not known resistant          Bilateral
```

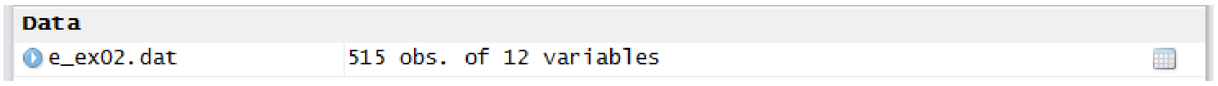
To see the variable names only, type:

```
# Get the list of variable names only
names(e_ex02.dat)
```

and get:

```
[1] "age"        "fq04"       "sex"        "totobstime" "agequart"   "agemed"     "outcome07"
[8] "outcome02"  "pza02"      "kmy02"      "pth02"      "cxr02"
```

R has a special way to deal with missing information. It uses NA to denote any missing value, be it numeric or text. If NA is the assigned value, then a record in an analysis using the variable with such a value will be excluded. We will use different approaches to the analysis. In some analyses, we will include all 515 records – how many records we have, can be seen in the Environment lower quadrant of the left panel:



```
Data
e_ex02.dat                515 obs. of 12 variables
```

In another analysis, we will exclude missing observations. We will create three datasets:

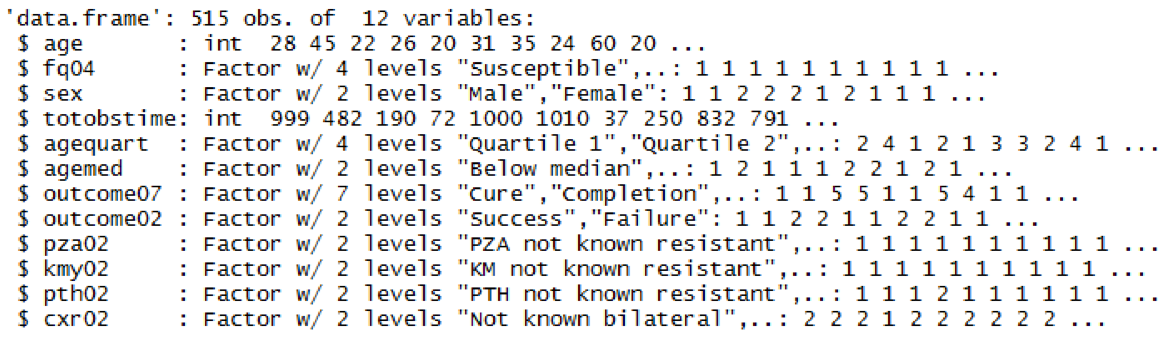
| | |
|---|---|
| `e_ex02_01.dat` | is the unaltered full set with missing values designated by NA, R's way to define missing values. |
| `e_ex02_02.dat` | is the dataset containing only records with information on initial fluoroquinolone resistance. |
| `e_ex02_03.dat` | is the subset of patients with initial ofloxacin resistance with either a bacteriologically unsuccessful outcome (failure or relapse) or a bacteriologically successful outcome (relapse-free cure or treatment completion). |

## The dataset

To get information on the structure of the dataset:

```
str(e_ex02.dat)
```

and we get (top only shown):



```
'data.frame':	515 obs. of  12 variables:
 $ age       : int  28 45 22 26 20 31 35 24 60 20 ...
 $ fq04      : Factor w/ 4 levels "Susceptible",..: 1 1 1 1 1 1 1 1 1 1 ...
 $ sex       : Factor w/ 2 levels "Male","Female": 1 1 2 2 2 1 2 1 1 1 ...
 $ totobstime: int  999 482 190 72 1000 1010 37 250 832 791 ...
 $ agequart  : Factor w/ 4 levels "Quartile 1","Quartile 2",..: 2 4 1 2 1 3 3 2 4 1 ...
 $ agemed    : Factor w/ 2 levels "Below median",..: 1 2 1 1 1 2 2 1 2 1 ...
 $ outcome07 : Factor w/ 7 levels "Cure","Completion",..: 1 1 5 5 1 1 5 4 1 1 ...
 $ outcome02 : Factor w/ 2 levels "Success","Failure": 1 1 2 2 1 1 2 2 1 1 ...
 $ pza02     : Factor w/ 2 levels "PZA not known resistant",..: 1 1 1 1 1 1 1 1 1 1 ...
 $ kmy02     : Factor w/ 2 levels "KM not known resistant",..: 1 1 1 1 1 1 1 1 1 1 ...
 $ pth02     : Factor w/ 2 levels "PTH not known resistant",..: 1 1 1 2 1 1 1 1 1 1 ...
 $ cxr02     : Factor w/ 2 levels "Not known bilateral",..: 2 2 2 1 2 2 2 2 2 2 ...
```

The full dataset thus contains 515 records with 12 variables (rearranged):

`AGE`: Age in years as an integer.

`AGEQUART`: Age as a categorical variable in quartiles.

`AGEMED`: Age as a binary categorical variable (below the median, and median and larger).

`SEX:` Patient's sex as categorical variable (Female, Male).

`OUTCOME07`: 7-level treatment outcome (Cure, Completion, Failure, Death, Default, Relapse, Reinfection).

`OUTCOME02`: Outcome as a binary categorical variable, i.e. successful (relapse-free cure or completion) vs all other outcomes (Success, Failure).

`TOTOBSTIME`: Time of observation, an integer in days from treatment start until an event occurred (see later) or the observation time ended because of follow-up completion.

`FQ04`: Drug susceptibility test result for ofloxacin. If resistant, the minimum inhibitory concentration to gatifloxacin was determined. This categorical variable has four levels (Susceptible, Low-level resistance, High-level resistance, Missing).

`PZA02`: Drug susceptibility test result of molecular (*pncA*) for pyrazinamide as a categorical binary variable, resistant vs not known to be resistant (PZA not known resistant, PZA resistant).

`KMY02`: Drug susceptibility test result of phenotypic testing for kanamycin as a categorical binary variable, resistant vs not known to be resistant (KM not known resistant, KM resistant).

`PTH02`: Drug susceptibility test result of phenotypic testing for prothionamide as a categorical binary variable, resistant vs not known to be resistant (PTH not known resistant, PTH resistant).

`CXR02`: Radiographic disease extent as a categorical binary variable, bilateral vs not known to be bilateral (Not known bilateral, Bilateral).

The information for some variables was complete (age and sex), for some very few had missing information (fluoroquinolone drug susceptibility test result, missing assigned to a defined category). Some variables had quite a few missing (known *pncA* sequencing result for pyrazinamide), others were quite complete (like radiographic disease extent). One could argue to deal differently with the missing than what was done for this exercise in this simplified categorization as "not known resistant". In any case, we chose here certain simplifications that are unlikely to importantly bias the planned analysis in the wrong direction. For some key questions, a "purist's" approach is chosen to deal with missing values.

The easiest to create is the full set: the only manipulation required is an assignment of the only variable with a missing value, the variable `FQ04` for ofloxacin drug susceptibility test result:

```
e_ex02.dat[e_ex02.dat$fq04=="Missing", "fluoroquinolone"] <- NA
```

Several new and important things are seen in this command line. Starting with the outermost right assignment NA, we note how R deals with missing data. In the data entry exercises with EpiData software we had taught (a bit dogmatically) that we wish to have a value (commonly designated as 9, 99, 9.99 etc) for all records with a missing value in a field. But this is by no means a universal standard. Rather more commonly such a field may remain empty or might have some value for missing or might be a mixture of both. Dealing correctly with missing values remains one of the most challenging tasks in any analysis. When we read the dataset and look at a frequency as follows:

```
table(e_ex02.dat$fq04)
```

we get (note the use of "table" rather than "tables" as in EpiData) before and after the NA assignment above:

Before:

```
 Susceptible  Low-level resistance High-level resistance              Missing
         439                    33                    29                   14
```

After:

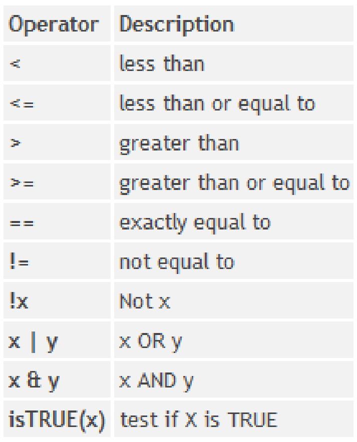
```
 Susceptible  Low-level resistance High-level resistance              Missing
         439                    33                    29                    0
```

We could deal with the 14 missing as a category and leave it just as it is with these 4 category levels. In our, we will, however, pay special interest to this variable. As 14 of 515 is just 2.7% of all observations, there is no important bias if we later simply exclude these 14 cases. There are different ways to sub-setting, but we assign the R designation for missing, which is NA.

To the very left we have the name of the dataset e_ex02.dat. Inside the brackets we identify the variable within the dataset as in:

```
setname$varname
```

The "==" is new and differs what we learned in EpiData. The following are the logical operators used in R:



| Operator | Description |
|---|---|
| < | less than |
| <= | less than or equal to |
| > | greater than |
| >= | greater than or equal to |
| == | exactly equal to |
| != | not equal to |
| !x | Not x |
| x \| y | x OR y |
| x & y | x AND y |
| isTRUE(x) | test if X is TRUE |

This is from the Quick-R website:

http://www.statmethods.net/

an extraordinarily useful website to get – as it says – quick answers to questions on R, highly recommendable! R (and Stata) uses the double `==` sign for equality rather than the single equal sign because the latter (`=`) can be used as "assign" instead of the `<-` we now got used to (and will stick to).

Thus, to get back:

```
e_ex02.dat$fq04=="Missing"
```

This identifies the variable `fq04` within the dataset `e_ex02.dat` and identifies records in which the category value is "`Missing`" (note that the value is not the "value" from the `*.REC` file, but the value is the label obtained from the Stata file for easy identification). After the comma, we state that we retain the variable name `fq04`:

```
e_ex02.dat[e_ex02.dat$fq04=="Missing", "fq04"] <- NA
```

we could check whether we have now records with `NA`s with the important command:

```
na.is(x)
```

which is used to find out which elements of `x` are recorded as missing (`NA`), i.e. here:

```
is.na(e_ex02.dat$fq04)
```

and we get (an excerpt from the list of all 515 records):

```
[289] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE  TRUE FALSE FALSE FALSE FALSE FALSE FALSE  TRUE
[305] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
[321] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE  TRUE FALSE FALSE FALSE FALSE FALSE FALSE
```

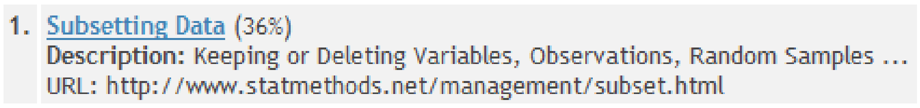
To assign this revised set to the new object with the desired name and to ensure that it is a "data frame", we write:

```
e_ex02_01.dat <- data.frame(e_ex02.dat)
```

What is called a "data frame" in R is just what we EpiData users call a dataset. "It is a list of vectors and / or factors of the same length that are related "across" such that data in the same position come from the same experimental unit" (formulation copied from Peter Dalgaard, Introductory statistics with R, 2008). The distinction is made because R can deal with other types of data collection.

## Sub-setting

The next dataset is trickier to create. An excellent source on the internet to find quick answers is Quick-R at http://www.statmethods.net/index.html. If we enter "subsets" into the Search box, we see among several option, option Nr 1:


1. Subsetting Data (36%)
Description: Keeping or Deleting Variables, Observations, Random Samples ...
URL: http://www.statmethods.net/management/subset.html

Clicking on the link we get all we need to know for starters. To select observations, the full explicit is:

```
newdata <- mydata[which(mydata$gender=='F' & mydata$age > 65),]
```

There is a short-cut allowed:

```
newdata <- mydata[which(gender=='F' & age > 65),]
```

Note here the single quotes to enclose text, but we can also use double quotes. In any case, we need to know the criteria. We mentioned above that we want to do a sub-analysis on patients with initial ofloxacin resistance. If we just write some table command, we might have the problem of ambiguity on which of the currently existing two datasets the command needs to be executed.

Our dataset to create is to be named "`e_ex02_02.dat`". It will only contain records without missing data for the variable `fq04`. We are thus making a selection in EpiData language or sub-setting in R language. If we start from the originally read dataset, we write:

```
e_ex02_02.dat <- e_ex02.dat[which(e_ex02.dat$fq04 != "Missing"), ]
```

Alternatively, and perhaps simpler, we can use the `subset` function:

```
e_ex02_02.dat <- subset(e_ex02.dat, fq04 != "Missing")
```

Note (and see above) the "`!=`" denoting "unequal".

To make the even more complex third dataset, the newcomer might best first stick to the fully explicit. We thus write three lines, making intermediary datasets, each line being self-explanatory:

We start with the dataset `e_ex02_02.dat` which has 501 records. Excluding 26 deaths defined in the variable `outcome07`, we get 475 observations in the new dataset `e_ex02_02_03a.dat`:

```
e_ex02_03a.dat <- subset(e_ex02_02.dat,  outcome07 != "Death")
```

Excluding then 40 defaulters defined in the variable `outcome07`, we get 435 observations in the new dataset `e_ex02_02_03b.dat`:

```
e_ex02_03b.dat <- subset(e_ex02_03a.dat, outcome07 != "Default")
```

Excluding finally 382 patients with initial fluoroquinolone susceptibility defined in the variable `fq04`, we get 53 observations in the final desired dataset `e_ex02_02_03.dat`:

```
e_ex02_03.dat <- subset(e_ex02_03b.dat, fq04 != "Susceptible")
```

Of course, this can all be written into a single command line:

```
e_ex02_03.dat <- subset(e_ex02_02.dat,  outcome07 != "Death" & outcome07 !=
   "Default" & fq04 != "Susceptible")
```

and one gets the same result of 53 observations.

Finally, to save the three datasets to disk, we do as above:

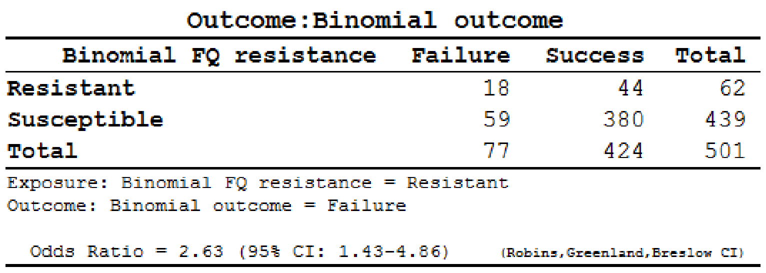
```
write.table(e_ex02_01.dat, file="e_ex02_01.dat", row.names=TRUE)
write.table(e_ex02_02.dat, file="e_ex02_02.dat", row.names=TRUE)
write.table(e_ex02_03.dat, file="e_ex02_03.dat", row.names=TRUE)
```

## A generically applicable function for the analysis of a 2-by-2 table

If we had a variable `fq02` to denote a binary outcome for initial fluoroquinolone (FQ) resistance (aggregating low and high resistance) and excluding those with missing fluoroquinolone test result, an analysis by the binary outcome in EpiData Analysis is as follows:

```
select fq02<>9 // exclude records with missing FQ result
tables outcome02 fq02 /o
```

which gives:

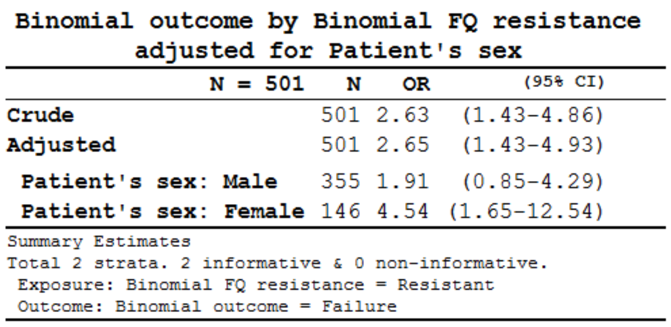


```
                 Outcome:Binomial outcome
 Binomial FQ resistance   Failure   Success   Total
Resistant                      18        44      62
Susceptible                    59       380     439
Total                          77       424     501
Exposure: Binomial FQ resistance = Resistant
Outcome: Binomial outcome = Failure

  Odds Ratio = 2.63 (95% CI: 1.43-4.86)      (Robins,Greenland,Breslow CI)
```

Second, we make a stratified analysis to look at the influence of SEX:

```
tables outcome02 fq02 sex /o
```

which gives a summary:



```
 Binomial outcome by Binomial FQ resistance
         adjusted for Patient's sex
                   N = 501     N    OR        (95% CI)
Crude                        501  2.63   (1.43-4.86)
Adjusted                     501  2.65   (1.43-4.93)
 Patient's sex: Male         355  1.91   (0.85-4.29)
 Patient's sex: Female       146  4.54  (1.65-12.54)
Summary Estimates
Total 2 strata. 2 informative & 0 non-informative.
 Exposure: Binomial FQ resistance = Resistant
 Outcome: Binomial outcome = Failure
```

by the Mantel-Haenszel procedure.

The first table in R requires that we make a new variable for a binary result of fluoroquinolone resistance and then create the table:

```
e_ex02_02.dat[e_ex02_02.dat$fq04=="Susceptible", "fq02"] <- "1-Susceptible"
e_ex02_02.dat[e_ex02_02.dat$fq04=="Low-level resistance", "fq02"] <- "2-Resistant"
e_ex02_02.dat[e_ex02_02.dat$fq04=="High-level resistance", "fq02"] <- "2-Resistant"
e_ex02_fq02.dat <- data.frame(e_ex02_02.dat)
```

At this point let's introduce `attach`. Followed in parenthesis by the dataset name:

```
attach(e_ex02_02_fq02.dat)
```

it puts the dataset `e_ex02_02_fq02.dat` into the search path and we do thus not to need to repeat the dataset name all the time, it suffices to write the variable names. It is equally important not to forget to `detach` a dataset to get it out of the path. Thus, we type here:

```
detach(e_ex02_02.dat)
```

If the data set is in the path, what would otherwise be:

```
table(e_ex02_fq02.dat$fq02, e_ex02_fq02.dat$outcome02)
```

becomes simplified:

```
table(fq02, outcome02)
```

This gives just the bare bone core of the correct output table:

```
               Success Failure
 1-Susceptible     380      59
 2-Resistant        44      18
```

Note here that for the R command `table` the sequence is different from EpiData: the first variable gives the row and the second variable the column. We noted earlier that the default sequence in R is row, then column. Note also that we numbered "`1-Susceptible`" and "`2-Resistant`" to get these into our preferred alphabetical sequence (the default would be the inverse).

Could we somehow use what we did in Exercise 1, edit it a bit and get it functional for this situation? Open the `e_ex01_or.r` in the text editor and remove everything, except the following lines:

```
or <- (tab[1,1]/tab[2,1])/(tab[1,2]/tab[2,2])
se <- sqrt(1/a+1/b+1/c+1/d)
or.ci.lower <- exp(log(or)-1.96*se)
or.ci.upper <- exp(log(or)+1.96*se)
print(tab)
cat("\nOR:", round(or, digits=3), "\n95% CI:", round(or.ci.lower, digits=3),
"to ", round(or.ci.upper, digits=3))
```

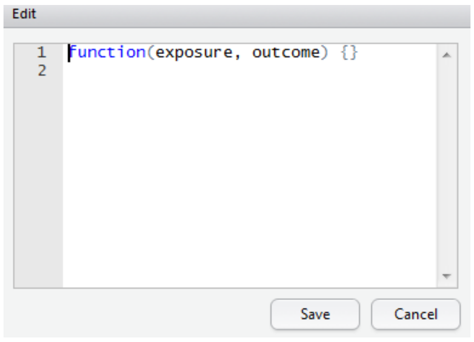
We are going to create a **Function** that can be applied in the future to the same setting. We have already used functions, `TABLE` is a function. To make our own function, type:

```
tab2by2 <- function(exposure, outcome) {}
```

This is the framework only, and it doesn't yet do anything. To put it to useful work, type:

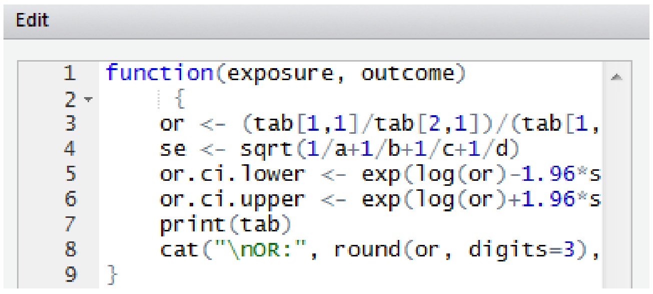
```
fix(tab2by2)
```

and a function editor window opens:



```
function(exposure, outcome) {}
```

Our commands will go between the curly braces `{}`. It is a custom (but no real need, but we will faithfully follow customs of those with more experience, they usually have their reasons) to put the opening brace `{` on a new line and then the commands after that on new lines, and have the closing brace `}` again at the end alone on a line. After we paste from the text editor what we have, we should thus get:



```
function(exposure, outcome)
    {
    or <- (tab[1,1]/tab[2,1])/(tab[1,
    se <- sqrt(1/a+1/b+1/c+1/d)
    or.ci.lower <- exp(log(or)-1.96*s
    or.ci.upper <- exp(log(or)+1.96*s
    print(tab)
    cat("\nOR:", round(or, digits=3),
}
```

It is not yet quite correct, because there is no object `tab`. Thus type on the line before the "`or`":

```
tab <- table(exposure, outcome)
```

That is, we assign the output of any two-by-two table of this format to an object `tab`. You may note that the RStudio short-cut key **ALT+-** to get the assign combination "`<-`" is non-functional in functions, you have to type it out. As we are already using `a`, `b`, `c`, and `d`, we might as well keep these, and then make in the "`or`" line assignments to these, so that in the revision we get:

```
function(exposure, outcome)
    {
    tab <- table(exposure, outcome)
    a <- tab[1,1]; b <- tab[2,1]; c <- tab[1,2]; d <- tab[2,2]
    or <- (a/c)/(b/d)
    se <- sqrt(1/a+1/b+1/c+1/d)
    or.ci.lower <- exp(log(or)-1.96*se)
    or.ci.upper <- exp(log(or)+1.96*se)
    print(tab)
    cat("\nOR:",  round(or,  digits=3),  "\n95%  CI:",  round(or.ci.lower,
     digits=3), "to ", round(or.ci.upper, digits=3))
}
```

Save it. We could save it later when quitting in the workspace, but it is better to also save in in a file and doing that now:

```
save(tab2by2, file = "tab2by2.r")
```

Conversely, when we open RStudio anew, it can be loaded whenever we need it with:

```
load("tab2by2.r")
```

Note that you actually need the full path, thus in our case:

```
load("C:/epidata_course/tab2by2.r")
```

Also note that this approach results in an *.r file that is not anymore a straight text file (you see gibberish in the text editor). It seems to be the price that has to be paid for a function we write by ourselves.

Before we apply it, we need to reopen and Run our `e_ex02.r` script which should read now:

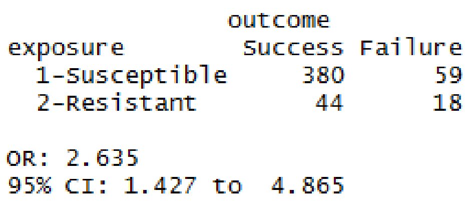
```
# Exercise 2:  Introduction to R software: data bases and functions

####################################
# 1) Import a Stata file

library(foreign)
e_ex02.dat <- read.dta("e_ex02.dta")
# e_ex02.dat <- read.dta("c:/epidata_course/e_ex02.dta")
write.table(e_ex02.dat, file="e_ex02.dat", row.names=TRUE)
```

```
# See the first 5 records
e_ex02.dat[1:5,]

# Get the list of variable names only
names(e_ex02.dat)

table(e_ex02.dat$fq04)

####################################
# 2) Create full unaltered set (515 records)
e_ex02.dat[e_ex02.dat$fq04=="Missing", "fq04"] <- NA
is.na(e_ex02.dat$fq04)
e_ex02_01.dat <- data.frame(e_ex02.dat)

####################################
# 3) Create a subset with records with known FQ DST result (401 records)
e_ex02_02.dat <- e_ex02.dat[which(e_ex02.dat$fq04 != "Missing"), ]
# or alternatively with the SUBSET function:
e_ex02_02.dat <- subset(e_ex02.dat, fq04 != "Missing")

####################################
# 4) Create a subset with records with known FQ resistance and
#  bacteriological success or failure (53 records)
# Commands on three lines
#  e_ex02_03a.dat <- subset(e_ex02_02.dat,  outcome07 != "Death")
#  e_ex02_03b.dat <- subset(e_ex02_03a.dat, outcome07 != "Default")
#  e_ex02_03.dat <- subset(e_ex02_03b.dat, fq04 != "Susceptible")
# Alternatively, commands on single line
e_ex02_03.dat <- subset(e_ex02_02.dat,  outcome07 != "Death" & outcome07 != "Default"
   & fq04 != "Susceptible")

# Table of outcome by FQ resistance
# Create first a new variable FQ02

e_ex02_02.dat[e_ex02_02.dat$fq04=="Susceptible", "fq02"] <- "1-Susceptible"
e_ex02_02.dat[e_ex02_02.dat$fq04=="Low-level resistance", "fq02"] <- "2-Resistant"
e_ex02_02.dat[e_ex02_02.dat$fq04=="High-level resistance", "fq02"] <- "2-Resistant"
e_ex02_fq02.dat <- data.frame(e_ex02_02.dat)
table(e_ex02_fq02.dat$fq02, e_ex02_fq02.dat$outcome02)

write.table(e_ex02_01.dat, file="e_ex02_01.dat", row.names=TRUE)
write.table(e_ex02_02.dat, file="e_ex02_02.dat", row.names=TRUE)
write.table(e_ex02_03.dat, file="e_ex02_03.dat", row.names=TRUE)


####################################
# 5) Make functions: 2-by-2 table
# Prepare making a function
# tab2by2 <- function(exposure, outcome) {}
# fix(tab2by2)
# save(tab2by2, file = "tab2by2.r")

attach(e_ex02_fq02.dat)
```

As the last command is `ATTACH` the proper file, let's try to apply it.  Type:

```
load("C:/epidata_course/tab2by2.r")
tab2by2(fq02, outcome02)
```

and you should get:



```
                  outcome
exposure          Success Failure
  1-Susceptible       380      59
  2-Resistant          44      18

OR: 2.635
95% CI: 1.427 to  4.865
```

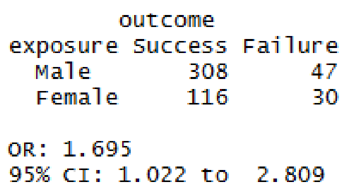
identical to the EpiData Analysis output (see above). Note that the R `load` function is necessary only after exiting R and reopening the script. Similarly, the three lines:

```
# tab2by2 <- function(exposure, outcome) {}
# fix(tab2by2)
# save(tab2by2, file = "tab2by2.r")
```

have been put as comments as they do not need repeat execution once done.

We can use this function for any other `EXPOSURE-OUTCOME` pair, such as:

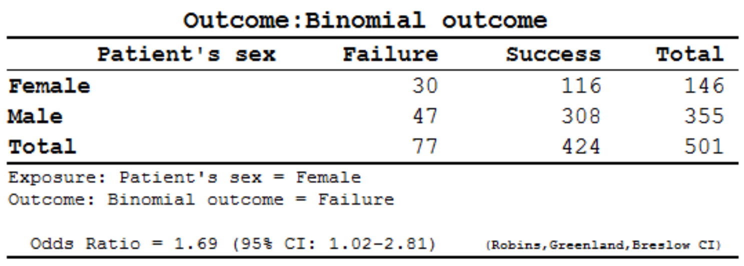
```
tab2by2(sex, outcome02)
```



```
         outcome
exposure Success Failure
  Male       308      47
  Female     116      30

OR: 1.695
95% CI: 1.022 to  2.809
```

In EpiData Analysis:

```
tables outcome02 sex /o
```



**Outcome:Binomial outcome**

| **Patient's sex** | **Failure** | **Success** | **Total** |
|---|---|---|---|
| **Female** | 30 | 116 | 146 |
| **Male** | 47 | 308 | 355 |
| **Total** | 77 | 424 | 501 |

Exposure: Patient's sex = Female
Outcome: Binomial outcome = Failure

Odds Ratio = 1.69 (95% CI: 1.02-2.81) (Robins,Greenland,Breslow CI)

We surely have to watch out for the variable SEX. You know from earlier that EpiData Analysis inverts (by design) the sequence of values for the exposure and outcome variables, while the net effect on the odds ratio thus remains unaffected.

To summarize: TABLE is an inbuilt function in R which cross tabulates the two variables. Replacing TABLE by our function '`tab2by2`' runs our code creating the table and with the odds ratio and the 95% confidence interval, displayed it in the desired format. There are many functions that have been created already by R users and collaborators which we can make use of. Thus, a skill we have to master is to search for a function which suits our purpose and make correct and careful use of it. This example illustrates the basic mode of using and applying functions in R.

## A generically applicable function for the analysis of a 2-by-2-by-2 table

In the previous paragraph we dealt with a **matrix** (a two-dimensional object of like elements). It was the special case of a 2-by-2 matrix. In this part, we move now forward to deal with **arrays**. An array is an n-dimensional table of like objects. In particular, we will concern ourselves with the special case of a 2-by-2-by-2 table, i.e. a 3-dimensional table. We can construct such an object in R with the command:

```
z <- array(1:8, c(2, 2, 2)); z
```

and get:

```
, , 1

     [,1] [,2]
[1,]    1    3
[2,]    2    4

, , 2

     [,1] [,2]
[1,]    5    7
[2,]    6    8
```

What is the location of "7"? Try:

```
u <- z[1, 2, 2]; u
```

The third dimension (row is the first, the column the second, the added one the third) is also indicated by the the two leading commas:

```
, , 2
```

We mentioned before that the location [indexing] in R is defined within brackets `[]`. The default sequence is by dimension, separated by commas, where the first dimension is always the row, the second the column and here the third is what we epidemiologists call the stratifying dimension. If we apply this concept of a multidimensional array to epidemiology, we thus have the concept of stratification.

The generic grammar for a table command for this basic stratification is:

```
table(exposure, outcome, stratvar)
```

or

```
table(rowvar, columnvar, stratvar)
```

where `stratvar` is the name of the straifying variable. In our case, specifically, we try:

```
table(fq02, outcome02, sex)
```

to get, if the last `attach` command was `attach(e_ex02_fq02.dat)`:

```
, , sex = Male

               outcome02
fq02             Success Failure
  1-Susceptible      274      38
  2-Resistant         34       9

, , sex = Female

               outcome02
fq02             Success Failure
  1-Susceptible      106      21
  2-Resistant         10       9
```

In 1959, Mantel and Haenszel proposed an efficient method for estimating a summary odds ratio from a series of 2 by 2 tables. It is easy to apply and does not require iterative calculations. In the following introduction, we follow Schlesselman's explanations and use of example (Schlesselman J J. Case-control studies. Design, conduct, analysis. New York: Oxford University Press, 1982).

The Mantel-Haenszel estimate of the odds ratio ($OR_{mh}$), adjusted for the effects of the stratification variables is calculated as:

$SUM(a_i d_i/n_i)/SUM(b_i c_i/n_i)$

An estimate of the variance of the $OR_{mh}$ has been proposed. If we define:

$w_i = b_i c_i/n_i$

and

$v_i = (a_i + c_i)/a_i c_i + (b_i + d_i)/b_i d_i$

then the variance of the $\log_e OR_{mh}$ is given by:

$var(\ln OR_{mh}) \approx SUM w_i^2 v_i/(SUM w_i)^2$

The approximate 95% confidence interval is thus given by:

$\ln OR_{mh} \pm 1.96*SQRT[var(\ln OR_{mh})]$

If we apply this to our data:

| | Female | | Male | |
|---|---|---|---|---|
| | Failure | Success | Failure | Success |
| Resistant | 9 | 10 | 9 | 34 |
| Susceptible | 21 | 106 | 38 | 274 |
| OR | 4.543 | | 1.909 | |
| $n_i$ | 146 | | 355 | |
| $W_i$ | 1.438 | | 3.639 | |
| $v_i$ | 0.268 | | 0.170 | |
| $a_i d_i/n_i$ | 6.534 | | 6.946 | |
| $b_i c_i/n_i$ | 1.438 | | 3.639 | |
| $SUM(a_i d_i/n_i)$ | 13.481 | | | |
| $SUM(b_i c_i/n_i)$ | 5.078 | | | |
| **ORmh** | **2.655** | | | |
| $w_i$^2*$v_i$ | 0.555 | | 2.258 | |
| SUM($w_i$^2*$v_i$) | 2.813 | | | |
| (SUM$w_i$)^2 | 25.784 | | | |
| SUM($w_i$^2*$v_i$)/(SUM$w_i$)^2 | 0.109 | | | |
| var(lnORmh) | 0.109 | | | |
| ORmh-lower | **1.390** | | | |
| ORmh-upper | **5.072** | | | |

EpiData Analysis gives:

$OR_{mh}$: 2.65 (1.43-4.93)

The point estimate is the same, but the 95% CI interval is slightly different because EpiData Analysis uses consistently the Robins, Greenland, Breslow confidence intervals, while we followed here Schlesselman's example using Hauck's interval. As the exercise is about the principle of learning on how to go about when creating a function, the choice of the confidence interval does not really matter here.

We will keep to this most simple example with only 8 cells total among three variables: each variable is binary.

Let’s best start from scratch and make a new function:

```
ormh <- function(exposure, outcome, stratvar) {}
```

We create a function `ORMH` (short for Mantel-Heanszel odds ratio) that takes three parameters, in that given sequence.

We propose to draft the content that goes between the curly braces `{}` in the text editor before we put into it with `fix`. We can modify what we had before, just slightly so:

```
tab <- table(exposure, outcome, stratvar)
a1 <- tab[1,1,1]; b1 <- tab[2,1,1]; c1 <- tab[1,2,1]; d1 <- tab[2,2,1]
a2 <- tab[1,1,2]; b2 <- tab[2,1,2]; c2 <- tab[1,2,2]; d2 <- tab[2,2,2]
```

This takes into account that we work now with a 3-dimensional array and have `a1`, `a2`, and `b1`, `b2`, etc.

We have to create the $a_i d_i / n_i$ and the $b_i c_i / n_i$ for each of the two tables:

```
adn1 <- (a1*d1)/n1
adn2 <- (a2*d2)/n2
bcn1 <- (b1*c1)/n1
bcn2 <- (b2*c2)/n2
```

and from these we calculate the object `MHOR`:

```
mhor <- (adn1+adn2)/(bcn1+bcn2)
```

Note that we chose the object name to be `mhor` to distinguish in from the function name `ormh`. Before we get too much carried away in our excitement, we should check whether we are on the right track:

```
fix(ormh)
```

and insert what we currently have:

```
function(exposure, outcome, stratvar)
    {
    tab <- table(exposure, outcome, stratvar)
    a1 <- tab[1,1,1]; b1 <- tab[2,1,1]; c1 <- tab[1,2,1]; d1 <- tab[2,2,1]
    a2 <- tab[1,1,2]; b2 <- tab[2,1,2]; c2 <- tab[1,2,2]; d2 <- tab[2,2,2]
    n1 <- a1+b1+c1+d1
    n2 <- a2+b2+c2+d2
    adn1 <- (a1*d1)/n1
    adn2 <- (a2*d2)/n2
    bcn1 <- (b1*c1)/n1
    bcn2 <- (b2*c2)/n2
    mhor <- (adn1+adn2)/(bcn1+bcn2)
    print(tab)
    cat("\nOR:", round(mhor, digits=3))
}
```

and test it with the command:

```
ormh(fq02, outcome02, sex)
```

We get:

```
, , stratvar = Male

                 outcome
exposure          Success Failure
  1-Susceptible       274      38
  2-Resistant          34       9

, , stratvar = Female

                 outcome
exposure          Success Failure
  1-Susceptible       106      21
  2-Resistant          10       9


OR: 2.655
```

We are definitely on the right track! Therefore, we might continue requiring next the two components `w` and `v` that we calculate as intermediate steps for calculation of the variance and then the variance itself:

```
w1 <- (b1*c1)/n1
w2 <- (b2*c2)/n2
v1 <- ((a1+c1)/(a1*c1))+((b1+d1)/(b1*d1))
v2 <- ((a2+c2)/(a2*c2))+((b2+d2)/(b2*d2))
varor <- ((w1^2*v1)+(w2^2*v2))/((w1+w2)^2)
```

Then we calculate the confidence interval:

```
se <-sqrt(var.mhor)
mhor.lower <-exp(log(mhor)-1.96*se)
mhor.upper <-exp(log(mhor)+1.96*se)
```

Finally, by adding what is to appear on the screen, we get the full function as:

```
function(exposure, outcome, stratvar)
{
    tab <- table(exposure, outcome, stratvar)
    a1 <- tab[1,1,1]; b1 <- tab[2,1,1]; c1 <- tab[1,2,1]; d1 <- tab[2,2,1]
    a2 <- tab[1,1,2]; b2 <- tab[2,1,2]; c2 <- tab[1,2,2]; d2 <- tab[2,2,2]
    a <- a1+a2; b <- b1+b2; c <-c1+c2; d <- d1+d2
    or <- (a/c)/(b/d)
    se <- sqrt(1/a+1/b+1/c+1/d)
    or.ci.lower <- exp(log(or)-1.96*se)
    or.ci.upper <- exp(log(or)+1.96*se)
    n1 <- a1+b1+c1+d1
    n2 <- a2+b2+c2+d2
    adn1 <- (a1*d1)/n1
    adn2 <- (a2*d2)/n2
    bcn1 <- (b1*c1)/n1
    bcn2 <- (b2*c2)/n2
    mhor <- (adn1+adn2)/(bcn1+bcn2)
    w1 <- (b1*c1)/n1
    w2 <- (b2*c2)/n2
    v1 <- ((a1+c1)/(a1*c1))+((b1+d1)/(b1*d1))
    v2 <- ((a2+c2)/(a2*c2))+((b2+d2)/(b2*d2))
    var.mhor <- ((w1^2*v1)+(w2^2*v2))/((w1+w2)^2)
    se <-sqrt(var.mhor)
    mhor.lower <-exp(log(mhor)-1.96*se)
    mhor.upper <-exp(log(mhor)+1.96*se)
    print(tab)
    cat("\nOR adj:", round(mhor, digits=3), "\n95% CI:", round(mhor.lower,
      digits=3), "-", round(mhor.upper, digits=3))
}
```

If we test with:

```
ormh(oflres, outcome02, sex)
```

we get:

```
, , stratvar = Male

               outcome
exposure        Success Failure
  1-Susceptible     274      38
  2-Resistant        34       9

, , stratvar = Female

               outcome
exposure        Success Failure
  1-Susceptible     106      21
  2-Resistant        10       9


OR: 2.655
95% CI: 1.39 5.072
```

We can use this generically for other variables, such as:

```
ormh(fq02, outcome02, cxr02)
```

and get:

```
, , stratvar = Not known bilateral

               outcome
exposure        Success Failure
  1-Susceptible      77      13
  2-Resistant         7       3

, , stratvar = Bilateral

               outcome
exposure        Success Failure
  1-Susceptible     303      46
  2-Resistant        37      15


OR: 2.647
95% CI: 1.432 4.892
```

Do not forget to save the function to a file:

```
save(ormh, file = "ormh.r")
```

Admittedly, what we have done here is still quite amateurish.  Fortunately, more professional contributors to R have written a function to calculate the Mantel-Haenszel estimate of the odds ratio.  If you just type the function:

```
mantelhaen.test
```

you see the full function. We can use it for our example:

```
mantelhaen.test(table(fq02, outcome02, sex))
```

and we get:

```
 Mantel-Haenszel chi-squared test with continuity correction

data:  table(fq02, outcome02, sex)
Mantel-Haenszel X-squared = 8.8339, df = 1, p-value = 0.002957
alternative hypothesis: true common odds ratio is not equal to 1
95 percent confidence interval:
 1.430554 4.926883
sample estimates:
common odds ratio
          2.65484
```

To look what the author of that function actually wrote, you can type in analogy of what we did above:

```
fix(mantelhaen.test)
```

We note thereby that the producers of this function that is part of the basic R package has the same approach to the calculation of the confidence interval as was chosen for EpiData Analysis, which gave as a summary:

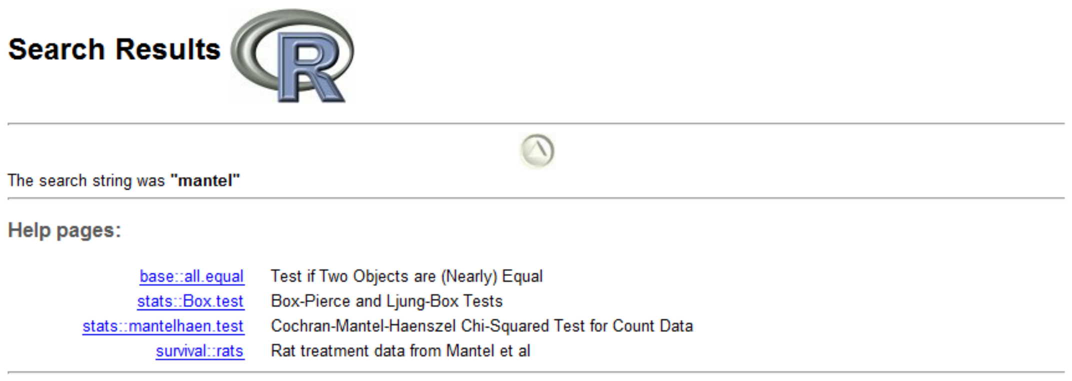
**Binomial outcome by Binomial FQ resistance adjusted for Patient's sex**

| N = 501 | N | OR | (95% CI) |
|---|---|---|---|
| **Crude** | 501 | 2.63 | (1.43-4.86) |
| **Adjusted** | 501 | 2.65 | (1.43-4.93) |
| **Patient's sex: Male** | 355 | 1.91 | (0.85-4.29) |
| **Patient's sex: Female** | 146 | 4.54 | (1.65-12.54) |

In summary, we have demonstrated that we can create any function in R which has a generalizable applicability. Fortunately, others have already done most of what is probably needed and it is thus not usually necessary to write our own functions. This is good news, of course, as it proves rather tedious and definitely requires sometimes basic, and sometimes more sophisticated knowledge of statistics. It also requires a thorough knowledge of the S programming language. Functions abound for R, but one has to look out for them to know how they are used properly. How did we find that this one exists? At the R prompt, type:

```
??mantel
```

and you get in the right lower quadrant of RStudio:



**Search Results**

The search string was **"mantel"**

**Help pages:**

| | |
|---|---|
| base::all.equal | Test if Two Objects are (Nearly) Equal |
| stats::Box.test | Box-Pierce and Ljung-Box Tests |
| stats::mantelhaen.test | Cochran-Mantel-Haenszel Chi-Squared Test for Count Data |
| survival::rats | Rat treatment data from Mantel et al |

where you then click on the link for detailed information.

***Tasks:***

- ***o*** ***Append the `ormh.r script` to calculate and show also the stratum-specific and crude (unstratified) odds ratioa with 95% confidence intervals (you may use the same type of confidence interval as we used in the `tab2by2.r` script)***

# Solution to Exercise 2: Introduction to R software: data bases and functions

Key points:

a. The package “foreign” allows importing a variety of data base formats including EpiData REC files.  It will not import any labels

b. “Everything in R is either an object or a function”: in this exercise you learned creating a generally applicable function for a simple 2-by-2 table, and for a stratified table to calculate a Mantel-Haenszel estimate of an adjusted odds ratio with a confidence interval

c. Numerous functions are available already in the base package, you will have to learn to use the Help file to get the fullest out of these functions

***Tasks:***

- ***Append the `ormh.r script` to calculate and show also the stratum-specific and crude (unstratified) odds ratioa with 95% confidence intervals (you may use the same type of confidence interval as we used in the `tab2by2.r` script)***

***Solution:***

The script `e_ex02.r`:

```
# Exercise 2:  Introduction to R software: data bases and functions

####################################
# 1) Import a Stata file

library(foreign)
e_ex02.dat <- read.dta("e_ex02.dta")
# e_ex02.dat <- read.dta("c:/epidata_course/e_ex02.dta")
write.table(e_ex02.dat, file="e_ex02.dat", row.names=TRUE)

# See the first 5 records
e_ex02.dat[1:5,]

# Get the list of variable names only
names(e_ex02.dat)

table(e_ex02.dat$fq04)

####################################
# 2) Create full unaltered set (515 records)
e_ex02.dat[e_ex02.dat$fq04=="Missing", "fq04"] <- NA
is.na(e_ex02.dat$fq04)
e_ex02_01.dat <- data.frame(e_ex02.dat)

####################################
# 3) Create a subset with records with known FQ DST result (401 records)
e_ex02_02.dat <- e_ex02.dat[which(e_ex02.dat$fq04 != "Missing"), ]
```

```
# or alternatively with the SUBSET function:
e_ex02_02.dat <- subset(e_ex02.dat, fq04 != "Missing")

####################################
# 4) Create a subset with records with known FQ resistance and
#  bacteriological success or failure (53 records)
attach(e_ex02_02.dat)
detach(e_ex02_02.dat)
# Commands on three lines
#  e_ex02_03a.dat <- subset(e_ex02_02.dat,  outcome07 != "Death")
#  e_ex02_03b.dat <- subset(e_ex02_03a.dat, outcome07 != "Default")
#  e_ex02_03.dat <- subset(e_ex02_03b.dat, fq04 != "Susceptible")
# Alternatively, commands on single line
e_ex02_03.dat <- subset(e_ex02_02.dat,  outcome07 != "Death" & outcome07 != "Default"
   & fq04 != "Susceptible")

# Table of outcome by FQ resistance
# Create first a new variable FQ02

e_ex02_02.dat[e_ex02_02.dat$fq04=="Susceptible", "fq02"] <- "1-Susceptible"
e_ex02_02.dat[e_ex02_02.dat$fq04=="Low-level resistance", "fq02"] <- "2-Resistant"
e_ex02_02.dat[e_ex02_02.dat$fq04=="High-level resistance", "fq02"] <- "2-Resistant"
e_ex02_fq02.dat <- data.frame(e_ex02_02.dat)
table(e_ex02_fq02.dat$fq02, e_ex02_fq02.dat$outcome02)

write.table(e_ex02_01.dat, file="e_ex02_01.dat", row.names=TRUE)
write.table(e_ex02_02.dat, file="e_ex02_02.dat", row.names=TRUE)
write.table(e_ex02_03.dat, file="e_ex02_03.dat", row.names=TRUE)


####################################
# 5) Make functions: 2-by-2 table
# Prepare making a function
# tab2by2 <- function(exposure, outcome) {}
# fix(tab2by2)
# save(tab2by2, file = "tab2by2.r")

attach(e_ex02_fq02.dat)

load("C:/epidata_course/tab2by2.r")
tab2by2(fq02, outcome02)
tab2by2(sex, outcome02)

####################################
# 6) Make functions: 2-by-2-by-2 table / Mantel-Heanszel
# Arrays for a stratification

table(fq02, outcome02, sex)

# ormh <- function(exposure, outcome, startvar) {}
# fix(ormh)
# save(ormh, file = "ormh.r")
ormh(fq02, outcome02, sex)
ormh(fq02, outcome02, cxr02)

####################################
# 7) On restart use LOAD to get the funcions
load("C:/epidata_course/ormh.r")
tab2by2(fq02, outcome02)
load("C:/epidata_course/tab2by2.r")
table(fq02, outcome02, sex)
ormh(fq02, outcome02, sex)

mantelhaen.test(table(fq02, outcome02, sex))
```

The script `ormh.r`:

```
function(exposure, outcome, stratvar)
{
    tab <- table(exposure, outcome, stratvar)
    a1 <- tab[1,1,1]; b1 <- tab[2,1,1]; c1 <- tab[1,2,1]; d1 <- tab[2,2,1]
    a2 <- tab[1,1,2]; b2 <- tab[2,1,2]; c2 <- tab[1,2,2]; d2 <- tab[2,2,2]
    a <- a1+a2; b <- b1+b2; c <-c1+c2; d <- d1+d2
    or1 <- (a1/c1)/(b1/d1)
    se1 <- sqrt(1/a1+1/b1+1/c1+1/d1)
    or1.ci.lower <- exp(log(or1)-1.96*se1)
    or1.ci.upper <- exp(log(or1)+1.96*se1)
    or2 <- (a2/c2)/(b2/d2)
    se2 <- sqrt(1/a2+1/b2+1/c2+1/d2)
    or2.ci.lower <- exp(log(or2)-1.96*se2)
    or2.ci.upper <- exp(log(or2)+1.96*se2)
    or <- (a/c)/(b/d)
    se <- sqrt(1/a+1/b+1/c+1/d)
    or.ci.lower <- exp(log(or)-1.96*se)
    or.ci.upper <- exp(log(or)+1.96*se)
    n1 <- a1+b1+c1+d1
    n2 <- a2+b2+c2+d2
    adn1 <- (a1*d1)/n1
    adn2 <- (a2*d2)/n2
    bcn1 <- (b1*c1)/n1
    bcn2 <- (b2*c2)/n2
    mhor <- (adn1+adn2)/(bcn1+bcn2)
    w1 <- (b1*c1)/n1
    w2 <- (b2*c2)/n2
    v1 <- ((a1+c1)/(a1*c1))+((b1+d1)/(b1*d1))
    v2 <- ((a2+c2)/(a2*c2))+((b2+d2)/(b2*d2))
    var.mhor <- ((w1^2*v1)+(w2^2*v2))/((w1+w2)^2)
    se <-sqrt(var.mhor)
    mhor.lower <-exp(log(mhor)-1.96*se)
    mhor.upper <-exp(log(mhor)+1.96*se)
    print(tab)
    cat("\nOR stratum 1:", round(or1, digits=3),  "\n95% CI:", round(or1.ci.lower,
  digits=3), "-", round(or1.ci.upper, digits=3),
        "\n\nOR stratum 2:", round(or2, digits=3),  "\n95% CI:", round(or2.ci.lower,
  digits=3), "-", round(or2.ci.upper, digits=3),
        "\n\nOR crude:",     round(or, digits=3),   "\n95% CI:", round(or.ci.lower,
  digits=3),  "-", round(or.ci.upper, digits=3),
        "\n\nOR adjusted:",  round(mhor, digits=3), "\n95% CI:", round(mhor.lower,
  digits=3),   "-", round(mhor.upper, digits=3))
}
```

The output:

```
, , stratvar = Male

               outcome
exposure        Success Failure
  1-Susceptible     274      38
  2-Resistant        34       9

, , stratvar = Female

               outcome
exposure        Success Failure
  1-Susceptible     106      21
  2-Resistant        10       9


OR stratum 1: 1.909
95% CI: 0.85 - 4.287

OR stratum 2: 4.543
95% CI: 1.646 - 12.535

OR crude: 2.635
95% CI: 1.427 - 4.865

OR adjusted: 2.655
95% CI: 1.39 - 5.072
```

# Exercise 3: Multivariable analysis in R part 1: Logistic regression

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

a. Know how to use logistic regression in R
b. Know how to properly remove factors for which most likely adjustment is not required

In Exercise 1, you learned some basic principles about the R language. As you go along, it will prove worthwhile to familiarize yourself more and more with the workings of the R / S language, and we provided you with links to some particularly useful texts that help in becoming more proficient.

In Exercise 2, you learned how to import a dataset, to assign the special values for missing data that R recognizes as such, to create data subsets for analysis. Next you learned to write your own basic function.

For none of the things you learned in Exercises 1 and 2 will we have a need for R. EpiData delivers all that, and in a very simple and intuitive way. We will take recourse to R only if we cannot solve a problem analytically with EpiData Analysis. One such application is the logistic regression analysis which is the subject of this exercise.

Before we get started with the actual work, open a new script page and save it as "`e_ex03.r`". We will use the dataset `e_ex02_02.dat` as our starting point, that is, the set with 501 cases with known fluoroquinolone drug susceptibility test result. The simplest way to make a dataset available in R is with the command `read.table`:

```
e_ex03 <- read.table("e_ex02_02.dat")
attach(e_ex03)
```

We assign its content to an object `e_ex03` which we attach, so that it is available in the path.

## The indication for a logistic regression

A major challenge in the analysis of epidemiologic data is to avoid falsely inferring causality between some factor and an outcome. As a simplified example we might find in a given population that cases of lung cancer occur far more frequently among males than among females. One would not conclude that being male is a risk factor for lung cancer before first examining whether smoking might also be a virtually male addiction in that population. For categorical variables we might use the Mantel-Haenszel stratification approach to adjust for such confounding. The method has two major limitations. First, the more variables we need to adjust for, the larger the uncertainty of our estimates and the more likely some strata are left without counts and making a summary measure becomes impossible. Second, the method is limited to categorical variables but often we do not wish to categorize a naturally continuous variable (like age) just to accommodate the method. From various possibilities, one favored method is logistic regression analysis that overcomes these two major limitations of stratified analysis. If carefully done, factors that independently predict a given outcome can be isolated and thus get the investigator closer to inference of causality.

## Logistic regression using R

Logistic regression is part of `glm` which is used to fit **g**eneralized **l**inear **m**odels. `GLM` is part of the R base package. The basic formulation of the model is simple:

```
output <- glm(formula = outcome ~ factor(var01) + factor (var02) + var03,
   data=datasetname, family=binomial)
```

where `output` is the object to which the model results are assigned to, and `glm` is the actual function. In parenthesis, one opens with the formula statement and the name of the outcome variable following the = sign, followed by a tilde ~ and then all the variables in the model. Specification with `factor` is required for categorical variables, followed by the variable name of the variable in parenthesis as with any function (`factor` being the function here). In the example `var01` and `var02` are categorical variables, while `var03` is treated as a continuous variable. All the variables entering the equation are connected by + signs. A comma designates the end of the variable list and is followed by `data=` and the name of the dataset. Finally, `family=binomial` defines that the logit member of the `glm` family is to be used.

Why don't we just enter here for starters an example from our dataset? Please add the following line (all written onto a single line) at the end of our `e_ex03.r`:

```
mylogit <- glm(formula = outcome02 ~ factor(fq02) + factor(sex) + age, data=
   e_ex03, family=binomial)
```

Into the next line type:

```
summary(mylogit)
```

and you get:

```
Call:
glm(formula = outcome02 ~ factor(fq02) + factor(sex) + age, family = binomial,
    data = e_ex03)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-2.3770   0.3750   0.5025   0.6043   1.3678

Coefficients:
                        Estimate Std. Error z value Pr(>|z|)
(Intercept)              2.54862    0.37757   6.750 1.48e-11 ***
factor(fq02)2-Resistant -1.10402    0.32517  -3.395 0.000686 ***
factor(sex)Male          0.90295    0.29063   3.107 0.001891 **
age                     -0.03619    0.01032  -3.506 0.000455 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 429.92  on 500  degrees of freedom
Residual deviance: 404.93  on 497  degrees of freedom
AIC: 412.93

Number of Fisher Scoring iterations: 5
```

Let's examine the output a bit, notably the part that begins with the `Coefficients`. We have `factor(fq02)2-Resistant` and `factor(sex)Male`. The referent for a variable is the lowest value. The values for `fq02` are `1-Susceptible` and `2-Resistant`. We therefore see `factor(fq02)2-Resistant` in the line. `age` is a continuous variable, and as such it does not have a referent and is shown as it is.

For the numeric output we have the `Estimate` and `Std. Error`, and in the last column a probability `Pr(>|z|)` for the `Estimate`. The latter shows that all three variables are highly significant predictors, but otherwise it is difficult to gauge what these numbers mean. Only by making this more explicit it becomes more meaningful. The estimate is a logartithm, thus we do some exponentiation and column binding (`cbind`):

```
lreg.or <-exp(cbind(OR = coef(mylogit), confint(mylogit)))
round(lreg.or, digits=4)
```

and then get more meaningfully:

```
                              OR  2.5 %  97.5 %
(Intercept)              12.7894 6.2062 27.3682
factor(fq02)2-Resistant   0.3315 0.1768  0.6367
factor(sex)Male           2.4669 1.3943  4.3737
age                       0.9645 0.9450  0.9841
```

The two functions `coef` and `confint` take the estimate and calculate the confidence interval from the standard error respectively. `cbind` “glues” (Dalgaard) the vectors together. Exponentiating the logarithmic terms gives the odds ratios (labeled here as `OR`) with the 95% confidence interval that is more amenable to interpretation. Thus, we need both the `summary` and the conversion of the estimates. The former is required to determine how to proceed in the stepwise exclusion of variables from the model. Intuitively, we correctly use `age` as a continuous variable, but it is not certain whether this is actually also analytically correct. We should at least check first whether the influence of age has continuity as logistic regression will force a fixed slope.

How should we categorize age? We could follow the standard classification of WHO, but perhaps a bit less arbitrary is to have the data themselves dictating the categories. In our dataset we have the variables `agequart` and `agemed`. These are indeed data-driven quartiles of age and the variable age split into a binary variable defined by the median respectively. These variables were made based on the entire set of 515 records. As we are using a dataset that has 14 records removed, the values are now conceptually if not actually incorrect. We will thus first remove the two obsolete variables and then create a new variable for quartiles of age based on the actual 501 records. We used:

```
names(e_ex03)
```

before. This gives us the names of the variables and their position in the dataset:

```
 [1] "age"        "fq04"       "sex"        "totobstime" "agequart"   "agemed"     "outcome07"
 [8] "outcome02"  "pza02"      "kmy02"      "pth02"      "cxr02"      "fq02"
```

We can see / count that `agequart` and `agemed` take positions 5 and 6 respectively in the sequence of variables. In other words, we need only the variables in positions 1 through 4 and 7 through 13, which we write as:

```
e_ex03b <- e_ex03[c(1:4, 7:13)]
detach(e_ex03)
attach(e_ex03b)
names(e_ex03b)
```

where the last line `names(e_ex03b)` is only to check that we got what we intended to get and we did:

```
[1] "age"         "fq04"        "sex"         "totobstime" "outcome07"   "outcome02"   "pza02"
[8] "kmy02"       "pth02"       "cxr02"       "fq02"
```

Thus here the principle how to drop variables in R. Next we want to create a new variable from the existing variable `age`. As the values for the new variable are driven by the data, we need to know first more about the distribution of `age`. R has a powerful way of showing quantiles (look up the Help file by typing `??quantiles`). To obtain quartiles, we type:

```
quantile(age,  probs = c(25, 50, 75)/100)
```

and we get:

```
25% 50% 75%
 23  31  42
```

To create a new variable `agecat` from the existing variable age, we ensure (no problem if it is duplicated from above) that the correct file is in the path and then make the assignments followed by again detaching the file:

```
attach(e_ex03b)
e_ex03b$agecat[age >= 00 & age < 23] <- "Q1"
e_ex03b$agecat[age >= 23 & age < 31] <- "Q2"
e_ex03b$agecat[age >= 31 & age < 42] <- "Q3"
e_ex03b$agecat[age >= 42]            <- "Q4"
detach(e_ex03b)
```

We also note that our outcome variable "`outcome02`" is alphabetically listed as "`Failure`" first, then "`Success`". Intuitively, we might be more interested in risk factors for "Failure" rather than in risk factors for "Success". We thus make a rearrangement to get the order sequentially such that the odds calculated is Failure / Success, taking into account that R wishes the outcome to be `0` and `1` (zero and one) and write in full:

```
attach(e_ex03b)
e_ex03b$agecat[age >= 00 & age < 23] <- "Q1"
e_ex03b$agecat[age >= 23 & age < 31] <- "Q2"
e_ex03b$agecat[age >= 31 & age < 42] <- "Q3"
e_ex03b$agecat[age >= 42]            <- "Q4"
e_ex03b$out02[outcome02 == "Success"] <- 0
e_ex03b$out02[outcome02 == "Failure"] <- 1
detach(e_ex03b)
```

Then we give both new variables new names, write the data to disk, attach the new file, check the names to verify (not necessary, but it is always a good idea in the beginning to verify), and then write the output:

```
e_ex03c <- data.frame(e_ex03b)
write.table(e_ex03c, file="e_ex03c", row.names=TRUE)
attach(e_ex03c)
names(e_ex03c)
table(agecat, out02)
```

and get:

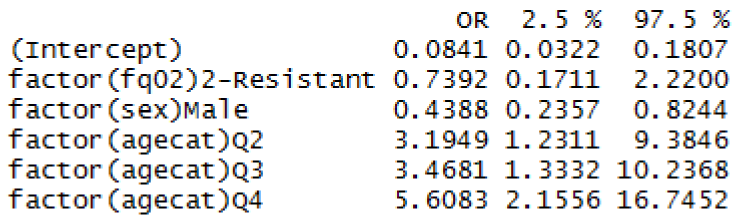
```
      out02
agecat   0   1
    Q1 103  10
    Q2 107  23
    Q3 109  20
    Q4 105  24
```

This ensures that for the age quartiles `Q1` is the referent and for treatment outcome `1-Success` is the referent.

Then we repeat the regression model with the categorized `agecat`:

```
# Logistic regression with AGE as categorical variable
mylogit2 <- glm(formula = out02 ~ factor(fq02) + factor(sex) + factor(agecat),
    data= e_ex03c, family=binomial)
summary(mylogit2)
lreg.or <-exp(cbind(OR = coef(mylogit2), confint(mylogit2)))
round(lreg.or, digits=4)
```

and get:



```
                              OR  2.5 %  97.5 %
(Intercept)               0.0841 0.0322  0.1807
factor(fq02)2-Resistant 0.7392 0.1711  2.2200
factor(sex)Male           0.4388 0.2357  0.8244
factor(agecat)Q2          3.1949 1.2311  9.3846
factor(agecat)Q3          3.4681 1.3332 10.2368
factor(agecat)Q4          5.6083 2.1556 16.7452
```

The referent is the youngest quartile set to unity. Relative to the referent, the risk of failure increases in the third and fourth quartile to similar levels and is then highest in the last quartile. This observation seems to support the treating of age as a continuous variable, or at least not plainly contradicting it.

This, so far has been "sampling" only. What we really wanted to evaluate is the influence of several factors that might modify the treatment outcome of multidrug-resistant tuberculosis. We will be starting with a full model inputting all variables, using instead of the binary `fq02` the more detailed `fq04`, slightly modified. The variable `fq04` had originally 4 levels, but in the source dataset of 501 records we are using in this exercise, the 14 records with a missing result have been dropped. We will first re-categorize the remaining 3 levels so as to ensure that the referent is `Susceptible`. As shown earlier, we do this by adding a sequential number to the desired sequence to get it alphabetically correct:

```
attach(e_ex03c)
e_ex03c$fq03[fq04 == "Susceptible"]           <- "1-Susceptible"
e_ex03c$fq03[fq04 == "Low-level resistance"]  <- "2-Low-level resistance"
e_ex03c$fq03[fq04 == "High-level resistance"] <- "3-High-level resistance"
detach(e_ex03c)
```

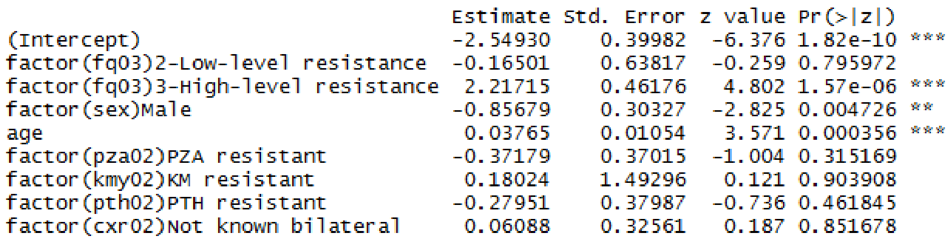
Not that it is essential but for getting things tidy, save the new as a data frame and read it in:

```
e_ex03d <- data.frame(e_ex03c)
write.table(e_ex03d, file="e_ex03d", row.names=TRUE)
e_ex03e <- read.table("e_ex03d")
```

and then we are ready for the full model:

```
mylogit3  <-  glm(formula  =  out02  ~  factor(fq03)  +  factor(sex)  +  age  +
    factor(pza02)   +   factor(kmy02)   +   factor(pth02)   +   factor(cxr02),
    data=e_ex03e, family=binomial)
summary(mylogit3)
```

and we get:


```
                                  Estimate Std. Error z value Pr(>|z|)    
(Intercept)                       -2.54930    0.39982  -6.376 1.82e-10 ***
factor(fq03)2-Low-level resistance -0.16501   0.63817  -0.259 0.795972    
factor(fq03)3-High-level resistance 2.21715   0.46176   4.802 1.57e-06 ***
factor(sex)Male                   -0.85679    0.30327  -2.825 0.004726 ** 
age                                0.03765    0.01054   3.571 0.000356 ***
factor(pza02)PZA resistant        -0.37179    0.37015  -1.004 0.315169    
factor(kmy02)KM resistant          0.18024    1.49296   0.121 0.903908    
factor(pth02)PTH resistant        -0.27951    0.37987  -0.736 0.461845    
factor(cxr02)Not known bilateral   0.06088    0.32561   0.187 0.851678    
```

Note also the “`AIC`” at the end to which will pay particular attention in the following:

```
AIC: 407.95
```

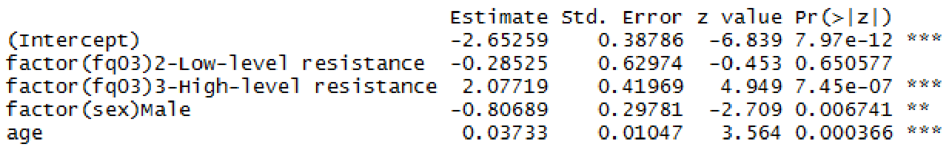
**A**IC stands for **A**kaike’s **I**nformation **C**riterion and is a weighted criterion of goodness of fit. The smaller the value, the better the fit. As we are going with stepwise elimination as one of the common procedures (Crawley M J. The R book. Second edition, Wiley, Chichester, UK, 2013, 1051 pp), we are checking how the criterion changes. Gauging from the probabilities above (last column), kanamycin resistance is the first factor to be removed, thus the model is reduced to:

```
mylogit3 <- glm(formula = out02 ~ factor(fq03) + factor(sex) + age +
    factor(pza02) + factor(pth02) + factor(cxr02), data=e_ex03e,
    family=binomial)
summary(mylogit3)
```

Improving the AIC to 405.96 and putting removal of the chest radiography result as the next. This removal improves the AIC to 403.99 and suggest removal of prothionamide resistance next. This removal improves the AIC to 402.54 and suggests removal of pyrazinamide resistance next. This removal improves the AIC to 401.55. The model is now:

```
mylogit3 <- glm(formula = out02 ~ factor(fq03) + factor(sex) + age,
    data=e_ex03e, family=binomial)
summary(mylogit3)
```

and gives:


```
                                  Estimate Std. Error z value Pr(>|z|)    
(Intercept)                       -2.65259    0.38786  -6.839 7.97e-12 ***
factor(fq03)2-Low-level resistance -0.28525   0.62974  -0.453 0.650577    
factor(fq03)3-High-level resistance 2.07719   0.41969   4.949 7.45e-07 ***
factor(sex)Male                   -0.80689    0.29781  -2.709 0.006741 ** 
age                                0.03733    0.01047   3.564 0.000366 ***
```

While low-level resistance to the fluoroquinolone is not a predictor, high-level resistance is a strong predictor, and both `sex` and `age` are predictors. Following the rules of being strict, none of the remaining factors should thus be removed.

If you watched it through the process of elimination, you found:

| **Modeling step** | **Resulting AIC value** |
|---|---|
| Full model | `407.95` |
| Remove `kmy02` | `405.96` |
| Remove `cxr02` | `403.99` |
| Remove `pth02` | `402.54` |
| Remove `pza02` | `401.55` |

What we haven't done yet, but are very much obliged to do is to test whether there is heterogeneity in the data and we therefore need one or more interaction terms. *Woolf's test for interaction* (also known as *Woolf's test for the homogeneity of odds ratios*) provides a formal test for Interaction. According to Mark Myatt (see text recommended at the beginning of Part E), R does not provide Woolf's test specifically, but the grammar of the function can be found in the help section on `mantelhaen.test` which we have used earlier. We use here the slightly modified script by Myatt (giving the same result), but is slightly more explicit. We type (perhaps best in the console, so that it does not interfere with our script `e_ex03.r`:

```
woolf.test <- function(x) {}
fix(woolf.test)
```

Then in the editor box:

```
function(x)
    {
    x <- x + 0.5
    k <- dim(x)[3]
    or <- apply(x, 3, function(x)
    {(x[1, 1] / x[1, 2]) / (x[2, 1] / x[2, 2])})
    w <- apply(x, 3, function(x) {1 / sum(1 / x)})
    chi.sq <- sum(w * (log(or) - weighted.mean(log(or), w))^2)
    p <- pchisq(chi.sq, df = k - 1, lower.tail = FALSE)
    cat("\nWoolf's X2 :", chi.sq, "\np-value :", p, "\n")
}
```

Finally we save:

```
save(woolf.test, file = "woolf.test.r")
```

Back to our `e_ex03.r` script (not necessary, but no harm either for this time, as it is loaded, but for the next run after we close, and also that we note where the credit goes to):

```
# Test for interaction, using the script for
# Woolf's test provided by Mark Myatt
load("c:/epidata_course/woolf.test.r")
```

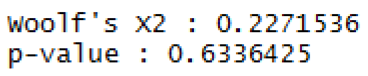
The function requires one parameter, denoted here as `x`. If we are interested in the interaction between outcome, ofloxacin resistance, and sex, we could make one object:

```
tabofl <- table(fq03, out02, sex)
```

and then:

```
woolf.test(tabofl)
```

and we get:

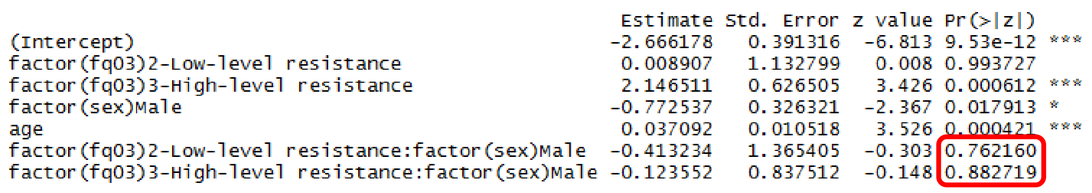


```
Woolf's X2 : 0.2271536
p-value : 0.6336425
```

Thus, not to worry about heterogeneity. For the sake of exercise, and also to look what happens, we will nevertheless put in an interaction term:

```
mylogit3 <- glm(formula = out02 ~ factor(fq03) + factor(sex) + age +
   factor(sex):factor(fq03), data=e_ex03e, family=binomial)
summary(mylogit3)
```

and we get a poorer AIC of 405.45 and insignificant interaction terms:



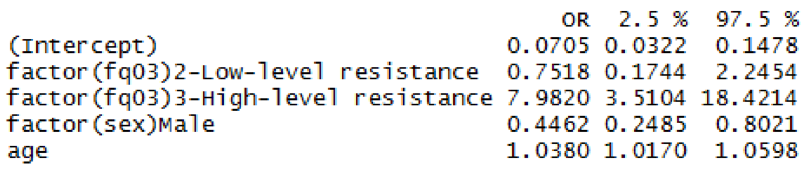
```
                                                   Estimate Std. Error z value Pr(>|z|)    
(Intercept)                                       -2.666178   0.391316  -6.813 9.53e-12 ***
factor(fq03)2-Low-level resistance                 0.008907   1.132799   0.008 0.993727    
factor(fq03)3-High-level resistance                2.146511   0.626505   3.426 0.000612 ***
factor(sex)Male                                   -0.772537   0.326321  -2.367 0.017913 *  
age                                                0.037092   0.010518   3.526 0.000421 ***
factor(fq03)2-Low-level resistance:factor(sex)Male -0.413234   1.365405  -0.303 0.762160    
factor(fq03)3-High-level resistance:factor(sex)Male -0.123552   0.837512  -0.148 0.882719    
```

If we find a significant interaction and perhaps even the AIC improves, then the interaction term must be retained.  As you note above, it is simple to write the interaction term which is writing the two variables connected by a colon as in `factor(var1):factor(var2)`.

We are therefore pretty much assured that this is our final model:

```
# Final model:
mylogit3  <-  glm(formula  =  out02  ~  factor(fq03)  +  factor(sex)  +  age,
  data=e_ex03e, family=binomial)
summary(mylogit3)
lreg.or <-exp(cbind(OR = coef(mylogit3), confint(mylogit3)))
round(lreg.or, digits=4)
```

With the following odds ratios and 95% confidence intervals:



```
                                        OR  2.5 %  97.5 %
(Intercept)                         0.0705 0.0322  0.1478
factor(fq03)2-Low-level resistance  0.7518 0.1744  2.2454
factor(fq03)3-High-level resistance 7.9820 3.5104 18.4214
factor(sex)Male                     0.4462 0.2485  0.8021
age                                 1.0380 1.0170  1.0598
```

***Task:***

- ***Examine whether there are interactions between age and sex and fluoroquinolone resistant and age, so that all possibilities have been checked before we accept the model.***

# Solution to Exercise 3: Multivariable analysis in R part 1: Logistic regression

**Key points:**

a. Logistic regression is the standard and most commonly used approach to multivariable analysis if the questions cannot be answered by adjusting in a stratified analysis using Mantel-Haenszel estimation of the odds ratio

b. Use stepwise exclusion of variables to improve the model fit, by observing p values and the AIC summary for quality of model fit

c. Beware of heterogeneity, test formally for heterogeneity using Woolf's test

***Task:***

- ***Examine whether there are interactions between age and sex and fluoroquinolone resistant and age, so that all possibilities have been checked before we accept the model.***

***Solution:***

The e_ex03.r script:

```
# Exercise 3: Multivariable analysis in R part 1: Logistic regression

rm(list=ls())

e_ex03 <- read.table("e_ex02_02.dat")
attach(e_ex03)

# Logistic regression with AGE as conintuous variable
mylogit <- glm(formula = outcome02 ~ factor(fq02) + factor(sex) + age, data=e_ex03,
     family=binomial)
summary(mylogit)
lreg.or <-exp(cbind(OR = coef(mylogit), confint(mylogit)))
round(lreg.or, digits=4)

# Drop variables
names(e_ex03)
e_ex03b <- e_ex03[c(1:4, 7:13)]
detach(e_ex03)
attach(e_ex03b)
names(e_ex03b)

# Create new variable for Quartiles of age
# Age as categorical variables in quartiles
quantile(age,  probs = c(25, 50, 75)/100)
## 25% 50% 75%
## 23  31  42

attach(e_ex03b)
e_ex03b$agecat[age >= 00 & age < 23] <- "Q1"
e_ex03b$agecat[age >= 23 & age < 31] <- "Q2"
e_ex03b$agecat[age >= 31 & age < 42] <- "Q3"
e_ex03b$agecat[age >= 42]            <- "Q4"
e_ex03b$out02[outcome02 == "Success"] <- 0
e_ex03b$out02[outcome02 == "Failure"] <- 1
detach(e_ex03b)
```

```
e_ex03c <- data.frame(e_ex03b)
write.table(e_ex03c, file="e_ex03c", row.names=TRUE)
attach(e_ex03c)
names(e_ex03c)
table(agecat, out02)

# Logistic regression with AGE as categorical variable
mylogit2 <- glm(formula = out02 ~ factor(fq02) + factor(sex) + factor(agecat), data=
     e_ex03c, family=binomial)
summary(mylogit2)
lreg.or <-exp(cbind(OR = coef(mylogit2), confint(mylogit2)))
round(lreg.or, digits=4)

# Full model logistic regression
attach(e_ex03c)
e_ex03c$fq03[fq04 == "Susceptible"]           <- "1-Susceptible"
e_ex03c$fq03[fq04 == "Low-level resistance"]  <- "2-Low-level resistance"
e_ex03c$fq03[fq04 == "High-level resistance"] <- "3-High-level resistance"

detach(e_ex03c)
e_ex03d <- data.frame(e_ex03c)
write.table(e_ex03d, file="e_ex03d", row.names=TRUE)
e_ex03e <- read.table("e_ex03d")

# Full model
mylogit3 <- glm(formula = out02 ~ factor(fq03) + factor(sex) + age + factor(pza02) +
     factor(kmy02) + factor(pth02) + factor(cxr02), data=e_ex03e, family=binomial)
summary(mylogit3)
# AIC 407.95

# Remove kmy02
mylogit3 <- glm(formula = out02 ~ factor(fq03) + factor(sex) + age + factor(pza02) +
     factor(pth02) + factor(cxr02), data=e_ex03e, family=binomial)
summary(mylogit3)
# AIC 405.96

# Remove cxr02
mylogit3 <- glm(formula = out02 ~ factor(fq03) + factor(sex) + age + factor(pza02) +
     factor(pth02), data=e_ex03e, family=binomial)
summary(mylogit3)
# AIC 403.99

# Remove pth02
mylogit3 <- glm(formula = out02 ~ factor(fq03) + factor(sex) + age + factor(pza02),
     data=e_ex03e, family=binomial)
summary(mylogit3)
# AIC 402.54

# Remove pza02
mylogit3 <- glm(formula = out02 ~ factor(fq03) + factor(sex) + age, data=e_ex03e,
     family=binomial)
summary(mylogit3)
# AIC 401.55

# Test for interaction, using the script for
# Woolf's test provided by Mark Myatt
load("c:/epidata_course/woolf.test.r")
attach(e_ex03e)
tabofl <- table(fq03, out02, sex)
woolf.test(tabofl)
# p-value : 0.6336425
tabofl <- table(fq03, out02, agecat)
woolf.test(tabofl)
# p-value : 0.9898
tabofl <- table(sex, out02, agecat)
```

```
woolf.test(tabofl)
# p-value : 0.5075



# Put in the interaction term anyhow
mylogit3  <-  glm(formula  =  out02  ~  factor(fq03)  +  factor(sex)  +  age  +
     factor(sex):factor(fq03), data=e_ex03e, family=binomial)
summary(mylogit3)
# AIC 405.45
# => AIC worsens

# Final model:
mylogit3 <- glm(formula = out02 ~ factor(fq03) + factor(sex) + age, data=e_ex03e,
     family=binomial)
summary(mylogit3)
```

The Woolf's test for heterogeneity:

```
tabofl <- table(fq03, out02, sex)
woolf.test(tabofl)
tabofl <- table(fq03, out02, agecat)
woolf.test(tabofl)
tabofl <- table(sex, out02, agecat)
woolf.test(tabofl)
```

```
tabofl <- table(fq03, out02, sex)
Woolf's X2 : 0.2271536
p-value : 0.6336425

tabofl <- table(fq03, out02, agecat)
Woolf's X2 : 0.115918
p-value : 0.9898611

tabofl <- table(sex, out02, agecat)
Woolf's X2 : 2.326063
p-value : 0.5075462
```

Conclusion:

No interaction between fluoroquinolone resistance and sex, fluoroquinolone resistance and age, nor sex and age.  Therfore the final model does not need an interaction term.

# Exercise 4: Multivariable analysis in R part 2: Cox proportional hazard model

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

a. Use the Cox proportional hazard model
b. Test the assumption for proportionality and if violated, carry out a stratified analysis

Often the results of the logistic regression are the culminating final summary of your analysis. In our specific setting, the logistic regression was an intermediate step to an adjusted form of a survival analysis. You have learned the principle of using a Kaplan-Meier survival analysis in an earlier Exercise. We extend on this approach and look at factors that determine the binary outcome in the treatment of multidrug-resistant tuberculosis. A Kaplan-Meier analysis would get us quite far, but we would face the problem of adjustment and the problem of dealing with continuous predictor variables such as age. While categorizing age is an option, it might be arbitrary and not the most satisfactory solution. We could end up with multiple small groups if values from more than one variable are made strata of a combining new variable. Almost inevitably, differences might become meaningless with the loss of power.

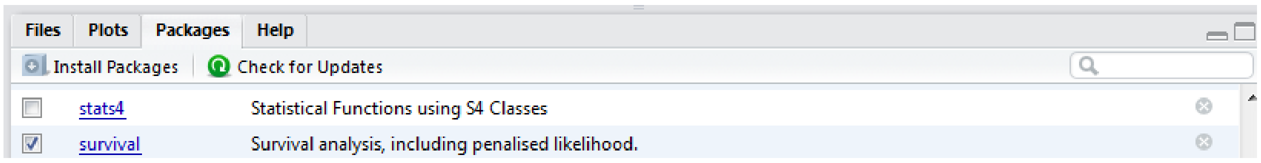
There are several techniques of multivariate analysis. Principally all do the same thing – simultaneous adjustment for multiple variables and providing independent effect estimates for each exposure variable in the model on the outcome. Depending on the type of outcome variable, different techniques are used.

If the outcome is a continuous variable, we use linear regression. If the outcome is categorical, we use logistic regression. If the outcome is ‘time to event’, we use a Cox proportional hazard model. If the outcome is ‘number of events’ (discrete numeric), then we use Poisson regression. Here, the outcome measure is ‘time to event’, and hence we use Kaplan-Meier analysis for univariate analysis and the Cox proportional hazard model for multivariate analysis.

In our approach to the analysis of the dataset on multidrug-resistant tuberculosis we combine the two techniques of logistic regression modeling and the Cox proportional hazard model in a way that is quite common: logistic regression is used first to evaluate and determine which variables have to be considered. There are alternative approaches, including determining the factors within the Cox model itself. In our case, we had isolated three factors, initial fluoroquinolone resistance, age, and sex. After fitting the Cox proportional hazard model including these three variables, we test whether any of the variables grossly violates the assumption of proportionality of hazards (which must be met). If there is such a variable, it must be removed from the adjustment and instead stratification by this variable is indicated. Then each of the strata are adjusted for the remaining variables and it is checked again whether anything remains that violates the proportionality assumption, and so on, until the final model emerges. This is the procedure we are going to apply.

## The R survival package

The base package of R does not include survival analysis, and the package “survival” must thus be installed (see lower right quadrant in RStudio):



| | | |
|---|---|---|
| ☐ | stats4 | Statistical Functions using S4 Classes |
| ☑ | survival | Survival analysis, including penalised likelihood. |

The “survival” package was written by Terry Therneau from the Mayo Clinic. The procedure is the same as we used before for the “foreign” package. Open a new file in the Source editor and save it as `e_ex04.r` file. Analogous to calling “foreign” from the library, we also need to be calling “survival” from the library. We are going to use the `e_ex02_02.dat` as our starting dataset. Thus, make sure that it is attached before we start doing anything.

```
* Exercise 4: Multivariable analysis in R part 2: Cox proportional hazard
  model
library(survival)
rm(list=ls())
e_ex04 <- read.table("e_ex02_02.dat")
names(e_ex04a)
```

The last line gives us the variables:

```
[1] "age"        "fq04"       "sex"        "totobstime" "agequart"   "agemed"     "outcome07"
[8] "outcome02"  "pza02"      "kmy02"      "pth02"      "cxr02"      "fq02"
```

We need only AGE, FQ04, SEX, TOTOBSTIME, and OUTCOME02, thus:

```
e_ex04b <- e_ex04a[c(1:4, 8)]
```

(It is not just to repeat what was learnt before, there is also some reason to reduce datasets: like EpiData Analysis, R works in the memory which makes it a relatively “slow processor”). We then make the same modification we discussed and did in Exercise 3, attach the file and check the names:

```
attach (e_ex04b)
e_ex04b$fq03[fq04 == "Susceptible"]           <- "1-Susceptible"
e_ex04b$fq03[fq04 == "Low-level resistance"]  <- "2-Low-level resistance"
e_ex04b$fq03[fq04 == "High-level resistance"] <- "3-High-level resistance"
e_ex04b$out02[outcome02 == "Success"] <- 0
e_ex04b$out02[outcome02 == "Failure"] <- 1
detach(e_ex04b)
names(e_ex04b)
e_ex04c <- e_ex04b[c(1, 3:4, 6:7)]
attach(e_ex04c)
names(e_ex04c)
```

The last line shows:

```
[1] "age"        "sex"        "totobstime" "fq03"       "out02"
```

We have thus reduced our dataset to the bare essential minimum with which we can work.

## Kaplan-Meier survival analysis

Let's first compare notes, i.e. check whether we get identical survival probabilities using the Kaplan-Meier method in EpiData Analysis and in R. In EpiData Analysis, we would use:

```
lifetable out02 totobstime /by=fq03 /i=b30 /adj \
          /ymin=0.30 /ymax=1 /NG /e3 \
          /ti="KM successful outcome probability" \
          /sub="by intitial fluoroquinolone susceptibility" \
          /t
```

We get:

| | |
|---|---|
| Survival probability for `susceptible`: | `0.865` |
| Survival probability for `low-level resistance`: | `0.909` |
| Survival probability for `high-level resistance`: | `0.480` |

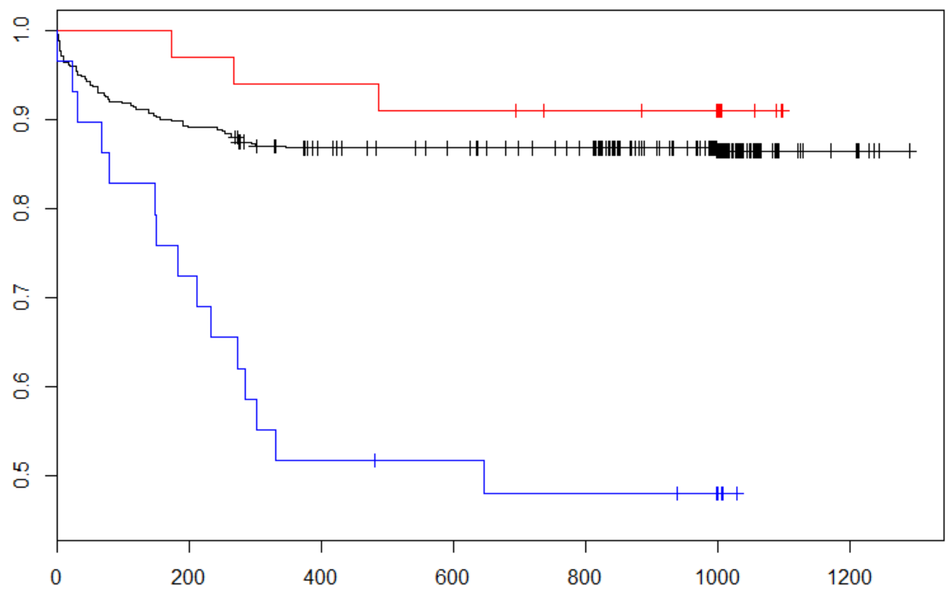
The model commands in R:

```
fit1 <- survfit(formula=Surv(totobstime, out02) ~ fq03, data=e_ex04c)
```

We want to add a simple plot:

```
plot(fit1, col = c(1, 2, 4), ymin=0.45)
# colors: 1=black; 2=red, 4=blue
```

which gives us the plot in the lower-right quadrant:



Remember that initially we defined R as a "language and environment for statistical computing and graphing". The graphics capabilities of R are enormous but it will take time to learn and acquire them (see more information in the text by Thomas Lumley). As it is not our primary purpose to evaluate the graphics capabilities of R, we leave that for the time being. We must see the statistical output, however, so we add a third command line:

```
fit1 <- survfit(formula=Surv(totobstime, out02) ~ fq03, data=e_ex04c)
plot(fit1, col = c(1, 2, 4), ymin=0.45)
# colors: 1=black; 2=red, 4=blue
summary(fit1, times = seq(0, 2000, 30))
```

We get exactly the same survival probabilities as with the `lifetable` in EpiData Analysis. This should strengthen our confidence in both software and our skills. We note, however, that the confidence intervals are different. In EpiData Analysis, events change the survival probability but censored observations do not. This is identical in R and it is what we expect. In EpiData analysis, the 95% confidence interval, however, continues to widen as observations with the passage of time become censored, while this is not the case in R. EpiData Analysis uses the philosophy that smaller numbers lead to larger uncertainty, while R focuses on the importance of uncertainty at the point of the last event. Comparison shows that Stata and R get the same results. EpiData Analysis has adapted the approach proposed by Altman, and this will be reviewed during the re-writing of the EpiData Analysis module. In any case, we can reproduce the survival probability in the Kaplan-Meier approach.

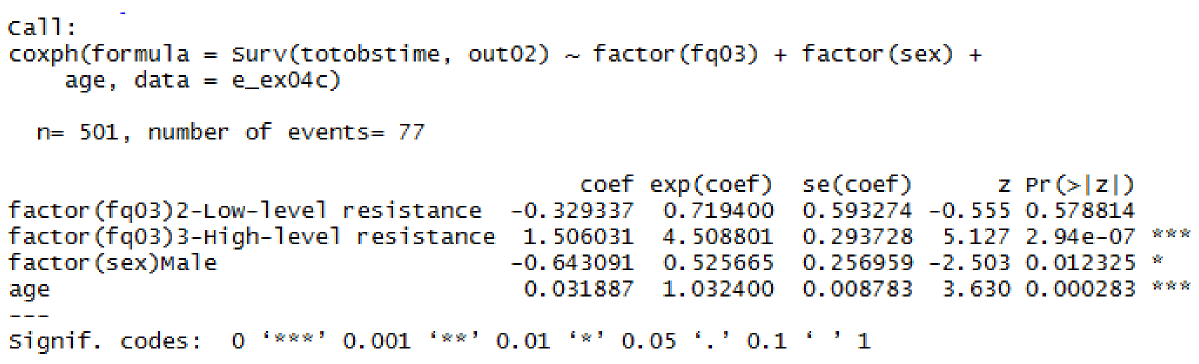
## The Cox proportional hazard model

The proportional hazards model allows the analysis of survival data by regression modeling. Linearity is assumed on the log scale of the hazard. Relative to a referent, say the rate of death among a control group, the rate of death among the experimental group might be half that of the control group and the hazard ratio is thus 0.5. In contrast to relative risks which are cumulative over observation time, hazard ratios reflect an instantaneous risk over the study period or a subset of the period. Hazard ratios suffer therefore somewhat less from possible selection bias introduced by endpoints. Under the Cox proportional hazard model, the hazard ratio is constant. The Cox model thus assumes an underlying hazard function with a corresponding survival curve. In a stratified analysis, there will be one such curve for each stratum.

The command lines for the Cox model are:

```
mdrcox  <- coxph(Surv(totobstime, out02) ~ factor(fq03) + factor(sex) + age,
   data=e_ex04c)
summary(mdrcox)
```

where `totobstime` is the variable for the total observation time from treatment start until the event occurs or the observation time is censored (be it e.g. to loss from follow-up after treatment cessation or reaching the end of the observation time). Note that "`Surv`" is capitalized (R is case-sensitive). The second line gives the output:



```
Call:
coxph(formula = Surv(totobstime, out02) ~ factor(fq03) + factor(sex) +
    age, data = e_ex04c)

  n= 501, number of events= 77

                                     coef exp(coef)  se(coef)      z Pr(>|z|)    
factor(fq03)2-Low-level resistance  -0.329337  0.719400  0.593274 -0.555 0.578814    
factor(fq03)3-High-level resistance  1.506031  4.508801  0.293728  5.127 2.94e-07 ***
factor(sex)Male                     -0.643091  0.525665  0.256959 -2.503 0.012325 *  
age                                  0.031887  1.032400  0.008783  3.630 0.000283 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

```
                                    exp(coef) exp(-coef) lower .95 upper .95
factor(fq03)2-Low-level resistance     0.7194     1.3900    0.2249    2.3013
factor(fq03)3-High-level resistance    4.5088     0.2218    2.5354    8.0183
factor(sex)Male                        0.5257     1.9024    0.3177    0.8698
age                                    1.0324     0.9686    1.0148    1.0503

Concordance= 0.687  (se = 0.033 )
Rsquare= 0.068   (max possible= 0.848 )
Likelihood ratio test= 35.06  on 4 df,   p=4.521e-07
Wald test            = 43.95  on 4 df,   p=6.565e-09
Score (logrank) test = 49.31  on 4 df,   p=5.026e-10
```

As it should be, the three variables are significantly associated. The `exp(coef)`, i.e. the exponent of the coefficient is the hazard ratio.

As mentioned above, the Cox proportional hazard model requires that the assumption of proportionality is met, that is the survival function for different factors are required to change proportionately and do not, for instance cross each other.

The test diagnostic to evaluate whether the assumption of proportionality is met is:

```
cox.zph(mdrcox)
```

gives:

```
                                       rho chisq        p
factor(fq03)2-Low-level resistance   0.248  4.72 0.029826
factor(fq03)3-High-level resistance  0.257  5.16 0.023055
factor(sex)Male                     -0.105  0.82 0.365224
age                                 -0.241  4.18 0.040909
GLOBAL                                  NA 18.51 0.000982
```

The global chi-square test is highly significant, that is the assumption of proportionality is violated. The most important culprit is seemingly the variable `fq03`. `AGE` is also significant but not that grossly and `sex` not at all. The way to resolve it is stratification, that is `fq03` must be taken out and the model must be stratified by `fq03`. R makes it easy for us to do that with the following modification:

```
mdrcox  <- coxph(Surv(totobstime, out02) ~ strata(fq03) + factor(sex) + age,
   data=e_ex04c)
summary(mdrcox)
```

This gives:

```
                     coef exp(coef)  se(coef)      z Pr(>|z|)
factor(sex)Male -0.643315  0.525547  0.256857 -2.505 0.012260 *
age              0.031399  1.031898  0.008758  3.585 0.000337 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

                exp(coef) exp(-coef) lower .95 upper .95
factor(sex)Male    0.5255     1.9028    0.3177    0.8695
age                1.0319     0.9691    1.0143    1.0498

Concordance= 0.647  (se = 0.039 )
Rsquare= 0.028   (max possible= 0.806 )
Likelihood ratio test= 14.26  on 2 df,   p=0.0008024
Wald test            = 14.38  on 2 df,   p=0.0007543
Score (logrank) test = 14.73  on 2 df,   p=0.0006345
```

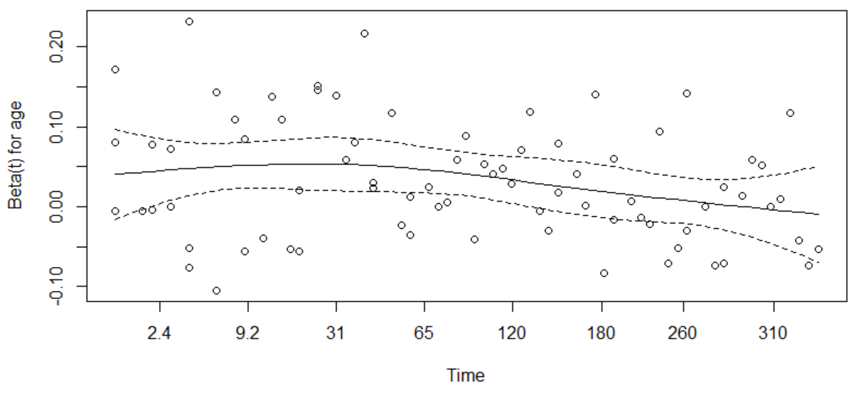
We have now the hazard of `sex` and `age` but not anymore of fluoroquinolone resistance because the latter was not a constant hazard and was thus taken out by stratification.

Testing the assumption for proportionality and plotting the test result:

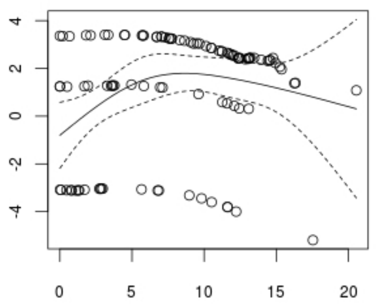
```
cox.zph(mdrcox)
plot(cox.zph(mdrcox))
```

```
                     rho chisq      p
factor(sex)Male -0.112 0.926 0.3358
age             -0.245 4.288 0.0384
GLOBAL              NA 7.864 0.0196
```

and



This indicates that the model is not ideal for the variable age, but with a bit lenience and the more or less regular graphical test we will allow it to pass without further more complex stratification. The interpretation of the graph in its most simplest way is how curved it is: if it is fairly flat (as we think it is here), the assumption of proportionality is not (much) violated. If it is decidedly different from flat, then the assumption is violated as in this example:



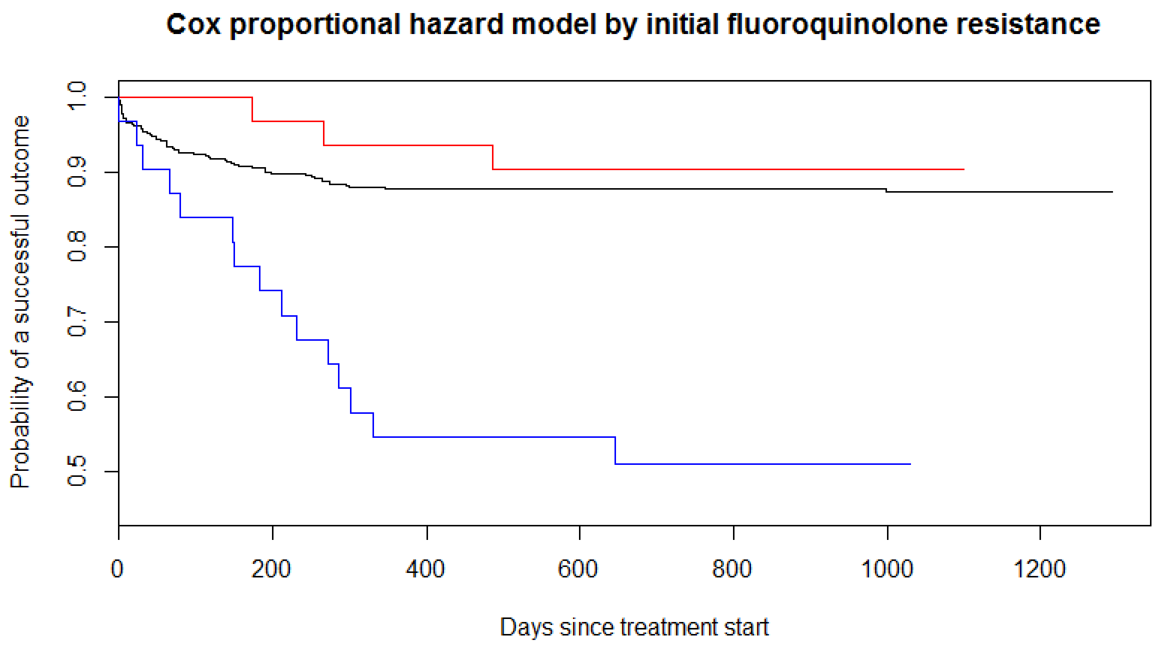
Copied from:

[http://stats.stackexchange.com/questions/15114/how-to-understand-the-plotting-of-the-cox-zph-function-in-r](http://stats.stackexchange.com/questions/15114/how-to-understand-the-plotting-of-the-cox-zph-function-in-r)

```
help(plot.survfit)
```

gives us information on how to plot it. It can get elaborate, of course, but here just a bare-bone sequence of commands:

```
p <- plot(survfit(mdrcox), ylim=c(.45, 1), xlab="Days since treatment start",
   mark.time=F, ylab="Probability of a successful outcome",  col=c(1, 2, 4),
   main="Cox   proportional   hazard   model   by   initial   fluoroquinolone
   resistance")
```



Our main interest is in the numeric output quantification of survival probability:

```
print(p)
```

```
$x
[1] 1292 1099 1030

$y
[1] 0.8744596 0.9045559 0.5096336
```

Thus, our main result, the survival probabilities among patients with initial fluoroquinolone susceptibility, low-, and high-level resistance, adjusted for age and sex are:

| | |
|---|---|
| Survival probability `susceptible`: | `0.874` |
| Survival probability `low-level resistant`: | `0.905` |
| Survival probability `high-level resistant`: | `0.510` |

To obtain the detailed data:

```
mdrcox2 <- survfit(mdrcox)
summary(mdrcox2, times = seq(0, 3000, 30))
```

gives (just the first 150 days for susceptible, low-level resistant, and high-level resistant):

```
                fq03=1-Susceptible
time n.risk n.event survival std.err lower 95% CI upper 95% CI
   0    439       2    0.996 0.00292        0.990        1.000
  30    419      19    0.956 0.00954        0.938        0.975
  60    411       7    0.942 0.01103        0.920        0.963
  90    404       7    0.927 0.01235        0.903        0.951
 120    400       4    0.918 0.01304        0.893        0.944
 150    397       4    0.910 0.01369        0.883        0.937
```

```
                fq03=2-Low-level resistance
 time n.risk n.event survival std.err lower 95% CI upper 95% CI
    0     33       0    1.000  0.0000        1.000            1
   30     33       0    1.000  0.0000        1.000            1
   60     33       0    1.000  0.0000        1.000            1
   90     33       0    1.000  0.0000        1.000            1
  120     33       0    1.000  0.0000        1.000            1
  150     33       0    1.000  0.0000        1.000            1

                fq03=3-High-level resistance
 time n.risk n.event survival std.err lower 95% CI upper 95% CI
    0     29       1    0.969  0.0309        0.910        1.000
   30     27       2    0.905  0.0526        0.807        1.000
   60     26       0    0.905  0.0526        0.807        1.000
   90     24       2    0.839  0.0667        0.718        0.981
  120     24       0    0.839  0.0667        0.718        0.981
  150     23       2    0.775  0.0764        0.638        0.940
```

Here, we show it for 30-day intervals, out for up to 3000 days but we could also just get it for the first 5 days, but daily:

```
summary(mdrcox2, times = seq(0, 5, 1))
```

```
                fq03=1-Susceptible
 time n.risk n.event survival std.err lower 95% CI upper 95% CI
    0    439       2    0.996 0.00292        0.990        1.000
    1    437       1    0.994 0.00358        0.987        1.000
    2    436       2    0.990 0.00463        0.981        0.999
    3    434       2    0.986 0.00548        0.975        0.996
    4    432       3    0.979 0.00656        0.967        0.992
    5    429       2    0.975 0.00719        0.961        0.989

                fq03=2-Low-level resistance
 time n.risk n.event survival std.err lower 95% CI upper 95% CI
    0     33       0        1       0            1            1
    1     33       0        1       0            1            1
    2     33       0        1       0            1            1
    3     33       0        1       0            1            1
    4     33       0        1       0            1            1
    5     33       0        1       0            1            1

                fq03=3-High-level resistance
 time n.risk n.event survival std.err lower 95% CI upper 95% CI
    0     29       1    0.969  0.0309         0.91            1
    1     28       0    0.969  0.0309         0.91            1
    2     28       0    0.969  0.0309         0.91            1
    3     28       0    0.969  0.0309         0.91            1
    4     28       0    0.969  0.0309         0.91            1
    5     28       0    0.969  0.0309         0.91            1
```

R is indeed powerful through flexibility.

***Task:***

***This analysis shows that low-level fluoroquinolone resistance has seemingly no influence on the outcome while in contrast high-level fluoroquinolone resistance is a powerful predictor for an adverse outcome. Nevertheless, even with high-level fluoroquinolone resistance, a remarkable 51% still had a successful outcome. Unsuccessful here also included death and default. What is of key interest with resistance to the core drug fluoroquinolone is whether the outcome is bacteriologically favorable or unfavorable.***

- ***The first task is to identify factors that are predictors for an unsuccessful bacteriological outcome (failure or relapse) versus a bacteriologically favorable outcome (completion and relapse-free cure) among cases with any type of fluoroquinolone resistance (low-level or high-level) and who did neither die nor default.***
- ***Summarize your analysis in a table showing numbers, odds ratios and 95% confidence intervals both univariate and adjusted multivariate for the factors identified in your regression analysis.***

# Solution to Exercise 4: Multivariable analysis in R part 2: Cox proportional hazard model

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

a. Use stepwise logistic regression to determine risk factors for a bacteriologically adverse outcome among patients with initial fluoroquinolone resistance who neither defaulted nor died

b. Summarize you findings in a table

***Solution:***

The `e_ex04.r` script from Exercise 4:

```
# Exercise 4: Multivariable analysis in R part 2: Cox proportional hazard
  model

library(survival)
rm(list=ls())
e_ex04a <- read.table("e_ex02_02.dat")
names(e_ex04a)

e_ex04b <- e_ex04a[c(1:4, 8)]

attach(e_ex04b)
e_ex04b$fq03[fq04 == "Susceptible"]           <- "1-Susceptible"
e_ex04b$fq03[fq04 == "Low-level resistance"]  <- "2-Low-level resistance"
e_ex04b$fq03[fq04 == "High-level resistance"] <- "3-High-level resistance"
e_ex04b$out02[outcome02 == "Success"] <- 0
e_ex04b$out02[outcome02 == "Failure"] <- 1
detach(e_ex04b)
names(e_ex04b)
e_ex04c <- e_ex04b[c(1, 3:4, 6:7)]
attach(e_ex04c)
names(e_ex04c)

# Reproducing the Kaplan-Meier lifetable analysis from EpiData Analysis
fit1 <- survfit(formula=Surv(totobstime, out02) ~ fq03, data=e_ex04c)
plot(fit1, col = c(1, 2, 4), ymin=0.45)
# colors: 1=black; 2=red, 4=blue
summary(fit1, times = seq(0, 2000, 30))

# Cox proportional hazard model
mdrcox  <- coxph(Surv(totobstime, out02) ~ factor(fq03) + factor(sex) + age,
  data=e_ex04c)
summary(mdrcox)
cox.zph(mdrcox)

mdrcox  <- coxph(Surv(totobstime, out02) ~ strata(fq03) + factor(sex) + age,
  data=e_ex04c)
summary(mdrcox)
cox.zph(mdrcox)
plot(cox.zph(mdrcox))
```

```
p <- plot(survfit(mdrcox), ylim=c(.45, 1), xlab="Days since treatment start",
  mark.time=F, ylab="Proportion without adverse event",  col=c(1, 2, 4),
  main="Cox proportional hazard model by initial fluoroquinolone
  resistance")
print(p)

mdrcox2 <- survfit(mdrcox)
summary(mdrcox2, times = seq(0, 3000, 30))
summary(mdrcox2, times = seq(0, 5, 1))
```

The `e_ex04_solution.r` script for the task:

```
# Exercise 4 Solution: Logistic regression to identify risk factors for
  failure

library(survival)
rm(list=ls())
e_ex04a_task <- read.table("e_ex02_03.dat")
names(e_ex04a_task)

e_ex04b_task <- e_ex04a_task[c(1:3, 8:11)]

attach(e_ex04b_task)
e_ex04b_task$fq03[fq04 == "Low-level resistance"] <- "1-Low-level
  resistance"
e_ex04b_task$fq03[fq04 == "High-level resistance"] <- "2-High-level
  resistance"
e_ex04b_task$out02[outcome02 == "Success"] <- 0
e_ex04b_task$out02[outcome02 == "Failure"] <- 1
detach(e_ex04b_task)
names(e_ex04b_task)
e_ex04c_task <- e_ex04b_task[c(1, 3, 5:9)]
attach(e_ex04c_task)
names(e_ex04c_task)

# Logistic regression
mylogit<- glm(formula = out02 ~ factor(fq03) + factor(sex) + factor(pza02) +
  factor(kmy02) + factor(pth02) + age, data=e_ex04c_task, family=binomial)
summary(mylogit)

# Remove KMY02
mylogit<- glm(formula = out02 ~ factor(fq03) + factor(sex) + factor(pza02) +
  factor(pth02) + age, data=e_ex04c_task, family=binomial)
summary(mylogit)

# Remove PTH02
mylogit<- glm(formula = out02 ~ factor(fq03) + factor(sex) + factor(pza02) +
  age, data=e_ex04c_task, family=binomial)
summary(mylogit)

# Remove SEX
mylogit<- glm(formula = out02 ~ factor(fq03) + factor(pza02) + age,
  data=e_ex04c_task, family=binomial)
summary(mylogit)

# Remove AGE
```

```
mylogit<-   glm(formula   =   out02   ~   factor(fq03)   +   factor(pza02),
  data=e_ex04c_task, family=binomial)
summary(mylogit)

# => PZA just not significant with p=0.5468, leave it in
# Final model
mylogit<-   glm(formula   =   out02   ~   factor(fq03)   +   factor(pza02),
  data=e_ex04c_task, family=binomial)
summary(mylogit)
lreg.or <-exp(cbind(OR = coef(mylogit), confint(mylogit)))
round(lreg.or, digits=4)

load("c:/epidata_course/woolf.test.r")
tab <- table(fq03, out02, pza02)
woolf.test(tab)
# P=0.936 => no interaction

# Univariate analysis
load("c:/epidata_course/tab2by2.r")
tab2by2(fq03, out02)
tab2by2(pza02, out02)
```

| | Successful | Not successful | | Total | Crude odds ratio | | | Adjusted odds ratio* | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | | CI | CI | | CI | CI | P |
| | n | n | % | | Point | low | high | Point | low | high | value |
| Total | 44 | 9 | 17.0 | 53 | | | | | | | |
| Fluoroquinolone resistance | | | | | | | | | | | |
| Low level | 30 | 1 | 3.2 | 31 | Ref | | | | | | |
| High level | 14 | 8 | 36.4 | 22 | 17.1 | 2.0 | 150.7 | 13.0 | 1.9 | 262.1 | 0.025 |
| Pyrazinamide resistance | | | | | | | | | | | |
| Not known * | 27 | 1 | 3.6 | 28 | Ref | | | | | | |
| Resistant | 17 | 8 | 32.0 | 25 | 12.7 | 1.5 | 110.8 | 9.2 | 1.3 | 187.0 | 0.055 |

| | |
|---|---|
| CI | 95% confidence interval |
| * | By logistic regression |
| ** | Susceptible or not tested |