Epidemiologic data analysis using R

Practicals 1

Faculty of Social Sciences, University of Tampere

–

Janne Pitkäniemi

10.02.2020

# Topics of practical 1

Learning objectives of this practical

* get familiar with R scripting and running R commands
* utilize R-studio in your daily work
* learn to use R packages
* data input from an external SPSS file,
* basic data manipulation tasks,
* tabulation using function stat.table() in package Epi.

# 1. Basics with R and R-Studio

Defining working directory and launching R with R-studio

1. Create a special subdirectory within your own user account as a **working directory** to contain the necessary R scripts and data files to be used in this course.
2. For example, my working directory for this course on my laptop is

setwd("C:/Users/janne.pitkaniemi/Documents/GitHub/TRE-EPI-R-2020")

, but yours is probably something else.

1. Open R-studio by clicking on the appropriate icon.
2. Change the default working directory of R by choosing *Session - Set Working Directory - Choose Directory* Use your own directory name here! instead of the default directory offered by R.
3. Check whether there are any objects in the memory from the upper risgt hand corner - Environment

**Comment**: When working with R it is useful to allocate for each project its own directory in which the files pertaining to that project are located, and to which especially the files created during an R session will be saved. When this directory is declared as the working directory in the beginning of a session, it will specify the default directory path for the files to be loaded and saved. However, files can still be loaded from and saved to other directories, but then the whole directory path must be specified in the file names.

**Attention!** Within an R script a slash ‘/’ must be then used instead of backslash ‘’ in the directory paths.

# 2. Working with script files in R-Studio

Writing and saving commands in a R-script file and running them from it in R-studio

1. Open new file in R-studio : *File - New file - R script*.
2. Install an R-package called foreign that will enable you to read several datafiles from other statistical programs like SPSS: *Tools - Install Packages …* . Types foreign and choose Install

Installation of a new package needs to be done only once when start to use a new package (or it’s latest version in your use). 3. Type the following two lines:

x<-2  
print(x)

1. Save the script from Save As and give a name for the script file \*.R
2. Close R-studio

# 3. Reading external data

Data file *breastca.sav* in SPSS-format is found from the Moodle site designated for this course. It contains data on 11 variables from 1207 women with breast cancer. These describe characteristics related to the survival time of the patient. We will read the data set into an R data frame and analyze it in subsequent tasks. First we’ll view the data in SPSS.

1. Write on the editor window the following R command lines, which will load some packages, read in the SPSS data set into a data frame and view its properties. Comments after **#** in each line can be omitted.

library(haven)  
bca <- read\_sav("C:/Users/janne.pitkaniemi/Documents/GitHub/TRE-EPI-R-2020/breastca.sav")  
bca[1:10, ] # listing the first 20 rows of the data frame  
str(bca) # viewing the structure  
attributes(bca)$variable.labels # description of variables  
summary(bca) # summary statistics of the variables

1. Save the script file (*File - Save - etc.*) into your own directory with name *bca.R*.
2. Run the commands written to the script file as follows: click the Run command in the scritping window
3. View the resulting output. Which of the variables are numeric and which are factors? Which values are referring to missing data for each of the variables? % Take your time!
4. Continue with writing each command in the following tasks to the script file, save, and run selected command lines as above.

# 4. Working with variables in R

Categorization of a numeric variable and forming 1-way frequency and percentage tables. 1. Create a factor named *age.gr* to this data frame with levels or age groups (years) 20-49, 50-64, 65-89 from variable *AGE* using function *cut( )*.

Get familiar with the syntax of this function by visiting its *help* page. Specify *right=F* so that each breakpoint is also an exact lower limit for an age group:

bca$age.gr <- cut(bca$AGE, br=c(20, 50, 65, 90), right=F)  
head(bca[1:5,c(2,12)])

1. It is possible to label the age groups to be more of publication style:

levels(bca$age.gr) <- c('20-49', '50-64', '65-89')  
head(bca[1:5,c(2,12)])

You may attach the data frame after this: *attach(bca)*

1. Form a marginal frequency table for *age.gr* using function *table()* and print:

table(bca$age.gr)

Data.table version

bcadt[order(age.gr),.N,by=age.gr]

1. Print a frequency table of variable *HISTGRAD* in the same way as for *age.gr*.

table(bca$HISTGRAD)

Data.table version

bcadt[order(HISTGRAD),.N,by=HISTGRAD]

# 5. Tabulations using stat.table (or data.table)

Do tabulations using the *stat.table()* function in the *Epi* package. First install package *Epi* from R-studio *Tools – install packages* and run following scrpit

library(Epi)

1. Frequencies and percentages of histological grade simultaneously, and marginal totals:

stat.table( HISTGRAD, list(count(), percent(HISTGRAD)),  
 margins=T,  
 data=bca);

Data.table version without missing values

res<-bcadt[order(HISTGRAD) & !is.na(HISTGRAD),.N,by=HISTGRAD] [, prop := 100\*(N/sum(N)), ]  
print(res)

Data.table version with missing values

res<-bcadt[order(HISTGRAD),.N,by=HISTGRAD] [, prop := 100\*(N/sum(N)), ]  
print(res)

1. The columns can be neatly labelled:

stat.table( HISTGRAD,  
 list(Number = count(), 'Per cent' = percent(HISTGRAD)),  
 margins=T,   
 data=bca);

Almost like a publication quality table!

1. Tabulate the mean age of patients as well as the minimum and maximum ages in each grade group:\

stat.table( HISTGRAD,   
 list(Number = count(),   
 'Mean age' = mean(AGE), min(AGE), max(AGE)),   
 margins=T,data=bca);

1. The numerical precision of numbers representing other quantities than counts or percentages is two decimal points by default. For us, two decimals in mean ages is exaggerating, so we wish to cut it into one decimal. There is a special **print method** for **stat.table()**, by which one can tune the number of decimal points. Before that, save the table into an own object.

mage.gr <- stat.table( HISTGRAD,   
list(Number = count(),   
'Mean age' = mean(AGE), min(AGE), max(AGE)),   
 margins=T,data=bca);  
  
print(mage.gr, digits=c(mean=1, min=0, max=0));

# 6. Two-way contingency tables, row & column percentages,and chi-square testing.

1. Form a 2-way contingency table with *age.gr* as the row variable and *HISTGRAD* as the column variable using function *table()*. Assign it to object *grbyage* and print. Take a look at the table. Can you judge anything about the association between age and histological grade from the table?

grbyage<-table(bca$age.gr,bca$HISTGRAD)  
grbyage

1. Using function *stat.table()* compute and print the frequency (counts) and percentage distribution of *HISTGRAD* in the three age groups:

stat.table( index = list(age.gr, HISTGRAD),  
 contents = list(count(), percent(HISTGRAD) ), data=bca );

Can you now say more about, how the grade distribution depends on age?

1. Maybe you wish to add marginal distributions to the table. For later purposes we also add the data frame as a *data* argument indicating that *stat.table()* can also operate on variables that are hidden in unattached data frames:

stat.table( index = list(age.gr, HISTGRAD),   
 contents = list(count(), percent(HISTGRAD) ),   
 margins = T, data = bca)

1. Print a similar table as in (c) that contains the row percentages only. This can be obtained by dropping the *count()* argument (and the comma after it!) from the list of *contents*.

stat.table( index = list(age.gr, HISTGRAD),   
 contents = list(percent(HISTGRAD) ),   
 margins = T, data = bca)

1. Perform a chi-square test for independence between *age.gr* and *HISTGRAD* using function *chisq.test()* with these variables as its main (and only) arguments.

chisq.test(bca$age.gr,bca$HISTGRAD)

1. Assign the value of the chi-square test function in the previous item to an object with name *res*, say, and view its structure usint *str()* function.

res<-chisq.test(bca$age.gr,bca$HISTGRAD)  
str(res)

1. Can you extract the *expected frequencies* from it? If yes, print them.

res$expected

# 7. Two- and three-dimensional tables

1. We are interested to know how does the presenec of lymphatic nodes (*LN\_YESNO*) in breast cancer patients seem to depend on age?

Create and print by *stat.table()* a 2-way frequency table with *age.gr* as the row variable and variable *LN\_YESNO* (presence vs. absence of lymph nodes) as the column variable. Present the row percentages, too.

stat.table( index = list(age.gr, LN\_YESNO),   
 contents = list(percent(LN\_YESNO) ),   
 margins = T, data = bca)

1. How does *LN\_YESNO* seem to depend on *HISTGRAD*?

Create another 2-way frequency and percentage table as in 1. but now with *HISTGRAD* as the row variable.

stat.table( index = list(age.gr, HISTGRAD),   
 contents = list(percent(HISTGRAD) ),   
 margins = T, data = bca)

1. Multidimensional tables are challenching, especially when they need to be interpreted to people. However, suppose we are interested knowing if the association between histological grading (*HISTGRAD*) and lymphatic nodes (*LN\_YESNO*) is the same by age groups (*age.gr*) of patients.

Ignoring the age:

stat.table( index = list(HISTGRAD,LN\_YESNO),   
 contents = list(count(),percent(LN\_YESNO) ),   
 margins = T, data = bca)

Let’s do this with data.table to illustrate the usefullness of it.

bcadt<-bcadt[!is.na(bca$HISTGRAD),] #remove missing observations for histological grading  
freq\_tab<-bcadt[order(age.gr,HISTGRAD,LN\_YESNO), .(.N), by = list(age.gr,HISTGRAD,LN\_YESNO)] # make table for all combinations of stratifying variables in order to verify our calculations  
  
res<-bcadt[order(age.gr,HISTGRAD),   
 .(percentage = round(100\*tabulate(LN\_YESNO)/.N)),  
 by = list(age.gr,HISTGRAD)]  
  
print(res)

# 8. Examining the properties of a table object.

1. Print again the 2-way table *grbyage* created above. Apply the functions *length()* and *sum()* on the table object *grbyage*. It seems like these functions treat the table as if it were a numeric vector …

length(grbyage)  
sum(grbyage)

1. Continue examining the inner structure of the table object by functions *class(), mode(), dim() and str()* What information do these provide?

class(grbyage)  
mode(grbyage)  
dim(grbyage)  
str(grbyage)

1. The cells of any 2-dimensional table are accessed using double indexing A given row (column) is identified by leaving the column number (row number) empty. – Leaning on this instruction, print only the following selected items from the table object *grbyage*:

* the cell frequency in the crossing of the 2nd row ja 2nd column,
* the frequencies of the whole 2nd row; compute and print also their sum,
* the frequencies of the whole 2nd column; compute and print also their sum.

grbyage[2,2]  
print(grbyage[2,])  
print(sum(grbyage[2,]))  
print(grbyage[,2])  
print(sum(grbyage[,2]))

# 9. Additional task

If you managed to do the previous tasks within the time allocated, get familial with data.table functions

<https://cran.r-project.org/web/packages/data.table/vignettes/datatable-intro.html>

Data table package is an alternative for handling (large) datasets and making summary tables

library(data.table)  
library(haven)  
bca <- read\_sav("C:/Users/janne.pitkaniemi/Documents/GitHub/TRE-EPI-R-2020/breastca.sav")  
bcadt<-data.table(bca) # This converts data.frame to data.table object  
bcadt # listing the first 20 rows of the data frame  
str(bcadt) # viewing the structure  
summary(bcadt) # summary statistics of the variables

Data.table version creating grouping variable

bcadt[,age.gr:=cut(bca$AGE, br=c(20, 50, 65, 90), right=F)]  
head(bcadt[1:5,c(2,12)])