Applied Statistics with R

Topic 02: Data import and export in R

Kaloyan Ganev

2022/2023

Lecture contents

Manual data entry

Manual data entry

- Convenient for small amounts of data, otherwise forget about it
- Demonstrated here just for illustration purposes
- To enter a single variable containing a few numbers (characters, etc.), use the already familiar `c()` function, for example:

```
x <- c(1,2,3,4,5,6,7,8,9)
```

- If the variable elements are a little bit more, you can use the `scan()`:

```
x <- scan()
```

  Entry stops when you enter a blank line
- Can read in all kinds of data, only tell it what the data type is, e.g.:

```
scan(what = character())
```

Manual data entry (2)

- The `scan()` function accepts also values from the clipboard
- For example, you can select a column of data in MS Excel, press `Ctrl + V` and paste it in R after you have issued the `scan()` command
- Before we conclude with manual entry we consider a simple example of creating a data frame from vectors in R

A simple data frame

- It will have five rows and five columns, so we need five vectors
- Let us have:

```
Age <- c(19,22,21,23,20)
Major <- c("Ec", "Ec", "BA", "BA", "Ec")
Degree <- c("BSc","MSc","BSc","MSc","BSc")
FName <- c("John","Mary","Maria","Juan","Jackie")
LName <- c("Hopkins","Jane","Curie","Carlos","Chan")
```

- Combining the vectors in a data frame (call it `Students`) is done through:

```
Students <- data.frame(FName,LName,Age,Major,Degree)
```

Viewing data frames

- Data frames can be easily visualized in RStudio
- There are several ways to do that:
  - Either type

```
View(Students)
```

  - Type just

```
Students
```

    or
  - Double click the object in the browser on the right-hand side
- Note that there is a button to expand the view after opening the frame

Editing data frames

- Unfortunately, RStudio does not have an editor but the R editor is available
- To edit, just type:

```
Students <- edit(Students)
```

or, equivalently

```
Students <- fix(Students)
```

- Finally, save your data:

```
save(Students, file = "students.RData")
```

- Later it can be loaded using

```
load("students.RData")
```

Foreign data file import

Importing `.txt` files

- The `read.table()` function is used
- Text files can have various extensions but this is non-essential
- What matters is that data have to be delimited in a specified way in them
- Let's use for example COVID-19 data for all the countries in the world
- We will directly use the web source

```
covid_data <- read.table("https://covid.ourworldindata.org/data/
    owid-covid-data.csv",
  sep = ",",
  header = T)
```

- To be punctual, check data type of variables...
- ... and change that of the `Date` one

```
covid_data$date <- as.Date(covid_data$date)
```

Importing `.csv` files

- There are some ready-made functions that call `read.table()` with pre-entered options
- `read.csv()` – read comma-separated data, decimal separator is "."
- `read.csv2()` – read semi-column-separated data, decimal separator is ","
- `read.delim()` – read tab-separated data, decimal separator is "."
- `read.delim2()`– read tab-separated data, decimal separator is ","

Importing .csv files (2)

- Take another example: download gold price data from GitHub

```
download.file("https://raw.githubusercontent.com/datasets/gold-
    prices/master/data/monthly.csv",
  destfile = "gold_monthly.csv",
  method = "libcurl")
```

- Then read into R

```
gold_price <- read.csv("gold_monthly.csv")
```

- . . . and then format the date data appropriately

```
gold_price$Date <- paste0(gold_price$Date,"-01") # Day needed
    for Date format
gold_price$Date <- as.Date(gold_price$Date)
```

Importing `.xls` and `.xlsx` files

- No built-in functionality in R
- However, several contributed packages are available such as **xlsx**, **readxl**, and **openxlsx**
- **xlsx** is good but requires Java so we will skip it as an option
- Let's pick **readxl** (part of the **tidyverse** ecosystem)
- We will read in some data on employees in Bulgaria contained in the `Labour_2.1.1_EN.xls` file

```
library(readxl)
employees_bg <- read_excel("Labour_2.1.1_EN.xls",
  sheet = "2020NaceRev2",
  skip = 4, na = "x")
```

- Pay attention to the options that are used

Reading from and writing to other software data formats

- Using the **foreign** package
- `read.dta(), write.dta()` – Stata files
- `read.octave()` – GNU Octave
- `read.spss()` – SPSS
- some other less familiar...
- The **haven** package
- The **hexView** package

Example: read sav file

- This is the default data storage format of SPSS
- Read the data using the **foreign** package:

```
library(foreign)
spss_data <- read.spss("survey.sav", to.data.frame = T)
```

- Read the data using the **haven** package:

```
library(haven)
spss_data2 <- read_spss("survey.sav")
```

Exporting data to text files

- `write.table()`
- `write.csv()`
- `write.csv2()`
- A useful tip: you can append exported data to an existing file using the option:

```
append = TRUE
```

- Example:

```
write.csv(spss_data2, "spss_data.csv")
```

Write `xlsx` files

- Using the **writexl** package:

```
library(writexl)
write_xlsx(spss_data2, "excel1.xlsx")
```

- Advantage of **readxl**: can select a cell range from an Excel sheet
- Using the **openxlsx** package:

```
library(openxlsx)
write.xlsx(spss_data2, "excel2.xlsx")
```

- Advantage of **openxlsx**: can create multi-sheet Excel files