Data Analysis in R: An Introduction

Anatoly Arlashin

Boston College

April 7, 2017

The latest version of this tutorial can be found here.
Outline

1. R’s basics
2. Importing data
3. Subsetting data
4. Graphics in R
5. Efficient data handling
Class R-scripts

The list below contains the links to R scripts that I used in our sessions.

- **Feb 10th** — Introduction to R and RStudio, data import
- **Feb 17th** — Data subsetting
- **Mar 3rd** — Visualizing data with ggplot2, part 1
- **Mar 17th** — Visualizing data with ggplot2, part 2
- **Mar 31st** — Introduction to data.table, part 1
The two essential things are R itself and the coding environment for it called RStudio.

- [www.r-project.org](http://www.r-project.org)
- [www.rstudio.com](http://www.rstudio.com)

Installation of both packages is pretty straightforward, the default options will be sufficient for all our needs.
Online resources

A small selection of tutorials:

- **An Introduction to R** — a manual from R Project, the creators of R
- **A (very) short introduction to R** — a 12 page summary of the most basic features of R and RStudio
- **Cookbook for R** — a collection of detailed and colorful tutorials
- **R bloggers** — a blog platform with lots of small tutorials and Q&As
- **Quick-R** — yet another collection

Online courses:

- **Rstudio Webinars** — free video tutorials on R and RStudio
- **Online learning** — a collection of links from RStudio’s authors
UI and coding basics

Tutorials on user interface of Rstudio, and some basic stuff about coding:

- Introduction to R and RStudio — #1
- Introduction to R and RStudio — #2
- Introduction to R and RStudio — #3
- Coding Style guide — useful advice on how to write clean and readable code
The way you import data into R depends on two things:

- the existing format and structure of the data
- the desired format and structure of the data

As with almost any other aspect of R, there exist a variety of packages designed to import specific data types. Some of those packages are minimalistic, while others offer a wider selection of options.

The differences in various packages are mostly about

- data formats that are supported (*.xls, *.dta, *.csv, etc.)
- type of R object that will contain the imported data (matrix, list, dataframe, etc.)
- speed of import (only matters for big data)
As with everything else, there’s plenty of tutorials online on how to import various types of data into R. Below is a small selection of them:

- Quick-R — Importing Data (short)
- R Tutorial — Data Import (short)
- IDRE UCLA — How to Input data into R (short)
- CRAN — R Data Import/Export (long)
It is useful to tell R explicitly what folder/directory on your computer you would like to use as the current location, so that you will know where to put imported data files or where the output will be saved to by default.

```r
# displays the current working directory
gewd()

# sets the working directory to "C:/MSAE/
setwd("C:/MSAE/")
```

Note that the path your put in `setwd("")` command must have forward slashes.
Excel data (*.xls, *.xlsx)

Many ways to import Excel data, we’ll look at the `gd`ata package and the `read.xls()` and `read.xlsx()` functions it provides.

Note: this package requires Perl language interpreter installed on your system. It’s preinstalled on MacOS and Linux system, while for Windows systems one can download it [here](#).

```r
# load package
library(gdata)
# load help documentation
help(read.xls)
# read the contents of the file "mydata.xls"
# into a new R object called "mydata.xls"
mydata.xls <- read.xls("mydata.xls")
```

Both functions return a data frame. The default option is to read the contents of the first sheet in Excel workbook.
Stata data (* . dat)

We’ll use **readstata13** package, it supports importing data from Stata versions 14 and older.

```
# load package
library(readstata13)

help(read.dta13)
# read the contents of the file "mydata.dta"
# into a new R object called "mydata.dta"
mydata.dta <- read.dta13("mydata.dta") # read file
```

The function returns a data frame.
Tabular separated data (*.csv, *.txt)

This is the type of data stored in a raw text format with variables separated by either symbols (comma, semicolon, etc) or spaces.

We'll use the built-in functions `read.csv()` and `read.table()`:

```r
# load help documentation
help(read.table)
# read the contents of the file "mydata.csv"
# into a new R object called "mydata.csv"
mydata.csv <- read.csv("mydata.csv")
```

The function returns a data frame. While technically `read.csv()` is a special case of `read.table()`, the difference between them is that `read.csv()` is able to correctly read various csv files without specifying extra options, while `read.table()` may fail to do so.
Data subsetting tutorials

We will follow the first tutorial in the list below in our class. Others contain shorter and less formal versions of intro to data subsetting in R.

- Advanced R
- Quick R
- Cookbook for R
ggplot2 tutorials

ggplot2 is the most popular package for producing complex graphic visualizations in R. We will follow the first tutorial in the list below in our class, but other are very helpful too.

- Introduction to R Graphics with ggplot2
- Top 50 ggplot2 Visualizations
- ggplot2 Cheat-Sheet from RStudio
- Official website of ggplot2
- Data visualization using ggplot2
- 3 hours to learn ggplot2
- Data Visualization with ggplot2
data.table and dplyr

data.table came as an extension of the popular dplyr way (combination of dplyr package and data.frame object) of handling large datasets in R.
Both of them have low-level routines coded in C++ for highest efficiency, but data.table was designed specifically with the SQL logic behind it, making it faster and more efficient at some of the common data manipulation tasks.

- Getting started with data.table
- Introduction to dplyr
- data.table vs dplyr