Introduction to R

Maximilian Kasy

Fall 2019

Agenda

- Comparison of R to its alternatives
- Ressources for learning R
- Installing R
- An introductory R session

Why R?

- Most popular environment in statistics and machine learning communities.
- Open source, fast growing ecosystem.
- Packages for almost everything:
  - Data processing and cleaning
  - Data visualization
  - Interactive web-apps
  - Typesetting, writing articles and slides
  - The newest machine learning routines
  - . . .
- Accomplishes the things you might be used to do doing in Stata (data processing, fitting standard models) and those you might be used to doing in Matlab (numerical programming).
- High level language that (mostly) avoids having to deal with technicalities.

Alternatives to R

- **Stata** (proprietary): Most popular statistical software in economics, easy to use for standard methods, not a good programming language.
- **Matlab** (proprietary): Numerical programming environment, matrix based. Programming in (base) R is quite similar to Matlab.
- **Python** (open): General purpose programming language, standard in industry, not targeted toward data analysis and statistics, but lots of development for machine learning. More overhead to write relative to R.
- **Julia** (open): New language for numerical programming, fast, increasingly popular in macro / for solving complicated structural models, not geared toward data analysis.

Installing R, RStudio, and tidyverse

- **Install R**:
  https://cran.rstudio.com/
- **Install RStudio**:
  https://www.rstudio.com/products/rstudio/download/
- **Install tidyverse** packages: Type in RStudio terminal

```
install.packages("tidyverse")
```

- You will often install other packages using this command.

Ressources for learning R

- **An Introduction to R**
  Complete introduction to base R. My recommended place to get started.
  https://cran.r-project.org/doc/manuals/r-release/R-intro.pdf
- **R for Data Science**
  Introduction to data analysis using R, focused on the tidyverse packages. If your goal is to find a substitute for Stata, start here.
  http://r4ds.had.co.nz/
- **Advanced R**
  In-depth discussion of programming in R. Read later, if you want to become a good R programmer.
  https://adv-r.hadley.nz/

Ressources for data visualization in R

- **Data Visualization - A Practical Introduction**
  Textbook on data visualization, using ggplot2. http://socviz.co/
- **ggplot2 - Elegant Graphics for Data Analysis**
  In depth discussion of R-package for data vizualization.
  http://moderngraphics11.pbworks.com/f/ggplot2-Book09hWickham.pdf
- **An Economist's Guide to Visualizing Data**
  Guidelines for good visualizations (not R-specific).
  https://pubs.aeaweb.org/doi/pdfplus/10.1257/jep.28.1.209
- **A Layered Grammar of Graphics**
  The theory behind ggplot2.
  https://byrneslab.net/classes/biol607/readings/wickham_layered-grammar.pdf

Ressources for learning extensions to R

- **Programming interactive R-apps using Shiny**
  Useful if you want to make your methods easy to use for people not familiar with R, or want to include interactive visualizations in web-pages.
  https://shiny.rstudio.com/articles/
- **Markdown**
  A lightweight markup language.
  https://www.markdownguide.org/
- **R markdown** Integrate code and output into typeset documents and slides. These slides are written in R markdown. https://rmarkdown.rstudio.com/lesson-1.html
- **RStudio Cheat Sheets**
  Cheatsheets for numerous packages.
  https://www.rstudio.com/resources/cheatsheets/

A sample session in R

- Please type the commands on the following slides in your RStudio terminal.
- This session is based on https://en.wikibooks.org/wiki/R_Programming/Sample_Session
- R can be used as a simple calculator and we can perform any simple computation.

```r
# Sample Session
# This is a comment
2 # print a number

2+3 # perform a simple calculation

log(2) # natural log
```

A sample session in R

- R can be used as a simple calculator and we can perform any simple computation.

```
# Sample Session
# This is a comment
2 # print a number
```

```
## [1] 2
```

```
2+3 # perform a simple calculation
```

```
## [1] 5
```

```
log(2) # natural log
```

```
## [1] 0.6931472
```

Numeric and string objects.

```
x = 2 # store an object
x # print this object

(x = 3) # store and print an object


x = "Hello" # store a string object
x
```

Numeric and string objects.

```r
x = 2 # store an object
x # print this object
```

```
## [1] 2
```

```r
(x = 3) # store and print an object
```

```
## [1] 3
```

```r
x = "Hello" # store a string object
x
```

```
## [1] "Hello"
```

Vectors.

```r
 #store a vector
Height =
  c(168, 177, 177, 177, 178, 172, 165, 171, 178, 170)
Height[2] # Print the second component

# Print the second, the 3rd, the 4th and 5th component
Height[2:5]

(obs = 1:10) # Define a vector as a sequence (1 to 10)
```

Vectors.

```
 #store a vector
Height =
  c(168, 177, 177, 177, 178, 172, 165, 171, 178, 170)
Height[2] # Print the second component
```

```
## [1] 177
```

```
# Print the second, the 3rd, the 4th and 5th component
Height[2:5]
```

```
## [1] 177 177 177 178
```

```
(obs = 1:10) # Define a vector as a sequence (1 to 10)
```

```
##  [1]  1  2  3  4  5  6  7  8  9 10
```

Vectors 2

```r
Weight = c(88, 72, 85, 52, 71, 69, 61, 61, 51, 75)

# Performs a simple calculation using vectors
BMI = Weight/((Height/100)^2)
BMI
```

Vectors 2

```
Weight = c(88, 72, 85, 52, 71, 69, 61, 61, 51, 75)

# Performs a simple calculation using vectors
BMI = Weight/((Height/100)^2)
BMI
```

```
##  [1] 31.17914 22.98190 27.13141 16.59804 22.40879 23.32342 22.40588
##  [8] 20.86112 16.09645 25.95156
```

Vectors 3

- We can also describe the vector with **length()**, **mean()** and **var()**.

```r
length(Height)

mean(Height) # Compute the sample mean

var(Height)
```

Vectors 3

- We can also describe the vector with **length()**, **mean()** and **var()**.

```r
length(Height)
```

```
## [1] 10
```

```r
mean(Height) # Compute the sample mean
```

```
## [1] 173.3
```

```r
var(Height)
```

```
## [1] 22.23333
```

Matrices.

```
M = cbind(obs,Height,Weight,BMI) # Create a matrix
typeof(M) # Give the type of the matrix

class(M)  # Give the class of an object

is.matrix(M) # Check if   M is a matrix

dim(M)    # Dimensions of a matrix
```

Matrices.

```
M = cbind(obs,Height,Weight,BMI) # Create a matrix
typeof(M) # Give the type of the matrix
```

```
## [1] "double"
```

```
class(M)  # Give the class of an object
```

```
## [1] "matrix"
```

```
is.matrix(M) # Check if   M is a matrix
```

```
## [1] TRUE
```

```
dim(M)    # Dimensions of a matrix
```

```
## [1] 10  4
```

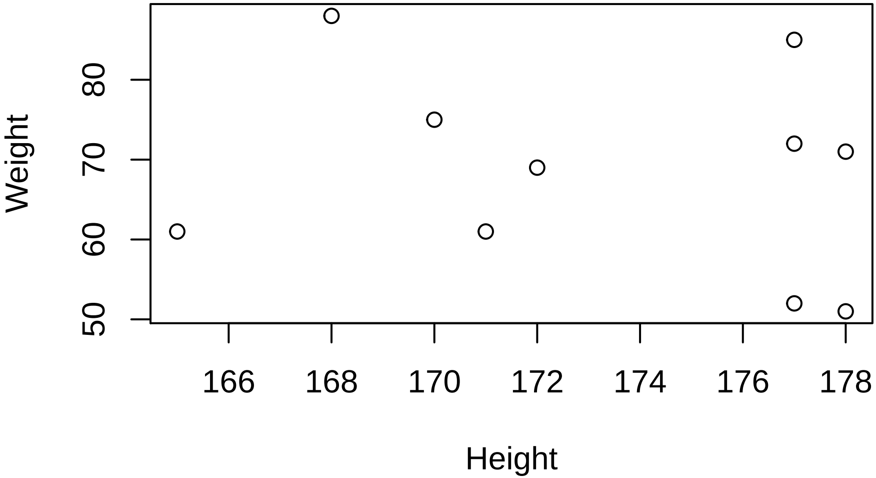
Simple plotting

- For “quick and dirty” plots, use **plot**.
- For more advanced and attractive data visualizations, use **ggplot**.

```r
plot(Height,Weight,ylab="Weight",xlab="Height")
```

Simple plotting

```
plot(Height,Weight,ylab="Weight",xlab="Height")
```



Dataframes (tibbles)

- **tibbles** are modernized versions of **dataframes**.
- Technically: Lists of vectors (with names).
- Can have different datatypes in different vectors.

```
library(tibble) # Load the tidyverse tibble package
mydat = as_tibble(M) # Creates a dataframe
names(mydat) # Give the names of each variable

summary(mydat)  # Descriptive Statistics
```

Dataframes

```
library(tibble) # Load the tidyverse tibble package
mydat = as_tibble(M) # Creates a tibble
names(mydat) # Give the names of each variable
```

```
## [1] "obs"    "Height" "Weight" "BMI"
```

```
summary(mydat)  # Descriptive Statistics
```

```
##       obs            Height          Weight           BMI
##   Min.   : 1.00   Min.   :165.0   Min.   :51.00   Min.   :16.10
##   1st Qu.: 3.25   1st Qu.:170.2   1st Qu.:61.00   1st Qu.:21.25
##   Median : 5.50   Median :174.5   Median :70.00   Median :22.70
##   Mean   : 5.50   Mean   :173.3   Mean   :68.50   Mean   :22.89
##   3rd Qu.: 7.75   3rd Qu.:177.0   3rd Qu.:74.25   3rd Qu.:25.29
##   Max.   :10.00   Max.   :178.0   Max.   :88.00   Max.   :31.18
```

Reading and writing data

- There are many routines for reading and writing files.
- Tidyverse versions are in the readr package.

```
library(readr) #load the tidyverse readr package
write_csv(mydat, "my_data.csv")
mydat2=read_csv("my_data.csv")
mydat2
```

Reading and writing data

```r
library(readr) #load the tidyverse readr package
write_csv(mydat, "my_data.csv")
mydat2=read_csv("my_data.csv")
```

```
## Parsed with column specification:
## cols(
##   obs = col_double(),
##   Height = col_double(),
##   Weight = col_double(),
##   BMI = col_double()
## )
```

Reading and writing data

```
mydat2
```

```
## # A tibble: 10 x 4
##      obs Height Weight   BMI
##    <dbl>  <dbl>  <dbl> <dbl>
##  1     1    168     88  31.2
##  2     2    177     72  23.0
##  3     3    177     85  27.1
##  4     4    177     52  16.6
##  5     5    178     71  22.4
##  6     6    172     69  23.3
##  7     7    165     61  22.4
##  8     8    171     61  20.9
##  9     9    178     51  16.1
## 10    10    170     75  26.0
```

Special characters in R

- **NA**: Not Available (i.e. missing values)
- **NaN**: Not a Number (e.g. 0/0)
- **Inf**: Infinity
- **-Inf**: Minus Infinity. For instance 0 divided by 0 gives a **NaN**, but 1 divided by 0 gives **Inf**.

```
0/0


1/0
```

Special characters in R

- **NA**: Not Available (i.e. missing values)
- **NaN**: Not a Number (e.g. 0/0)
- **Inf**: Infinity
- **-Inf**: Minus Infinity. For instance 0 divided by 0 gives a **NaN**, but 1 divided by 0 gives **Inf**.

```r
0/0
```

```
## [1] NaN
```

```r
1/0
```

```
## [1] Inf
```

Working directory

We can define a working directory. Note for Windows users : R uses slash (“/”) in the directory instead of backslash ("").

```r
setwd("~/Desktop") # Sets working directory
getwd() # Returns current working directory

dir() # Lists the content of the working directory
```

Defining functions

- Whenever you program something more involved, you should use functions.
- R makes it easy to provide default arguments.

```r
example_function = function(a, b=2) {
  r=a/b
  return(r)
}

example_function(3)

example_function(3,4)
```

Defining functions

```
example_function = function(a, b=2) {
  r=a/b
  return(r)
}

example_function(3)
```

```
## [1] 1.5
```

```
example_function(3,4)
```

```
## [1] 0.75
```

Linear regressions

- R makes it easy to fit linear regressions and other models
- The objects returned contain coefficients, residuals, fitted values, etc.

```r
example_regression = lm(Height ~ Weight + BMI, mydat)

summary(example_regression)
```

Linear regressions

```r
example_regression = lm(Height ~ Weight + BMI, mydat)
summary(example_regression)
```

```
## 
## Call:
## lm(formula = Height ~ Weight + BMI, data = mydat)
## 
## Residuals:
##     Min      1Q  Median      3Q     Max 
## -1.0168 -0.5849 -0.1534  0.4682  1.4380 
## 
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)    
## (Intercept) 174.24291    1.68433  103.45 2.08e-12 ***
## Weight        1.20911    0.08745   13.83 2.45e-06 ***
## BMI          -3.65895    0.23993  -15.25 1.26e-06 ***
## ---
```

Some further important commands

- Look up the help files for the following commands:

```
map()
ggplot()
```