Introduction to R

Statistics for Geosciences

Julian Rodemann[1]

October 20, 2021

[1] Some slides were provided by Alexander Bauer, Andreas Bender and André Klima. Thanks for sharing them!

Outline

Basics I

Basics – What is R?

R is...

- a software for Statistics/ data analysis
- a programming language
- available on all operating systems
- extendable by loading different `packages`
- state-of-the-art in scientific research
- open source
- powered by a big community of users out of all (scientific) areas

Basics – Why R, not Python?

R

+ has way more implemented statistical methods
+ is **the** language for scientific research
+ has great packages for data visualization
+ has a concave learning curve, i.e. beginners can perform data analysis within minutes (in this course)

Python

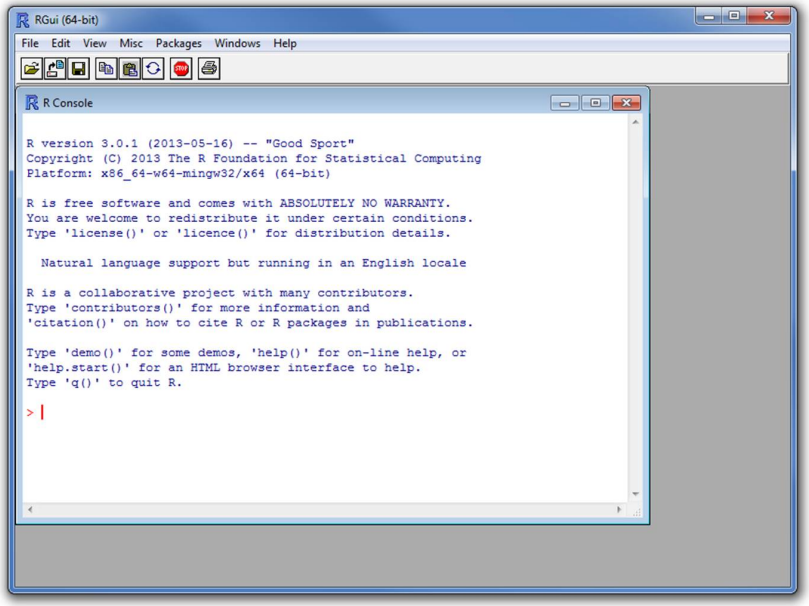
+ is a multi-purpose language
+ machine learning (deep learning) a large scale
+ has a learning curve that's linear and smooth, thanks to its easy-to-read syntax

Basics – Why R, not Python?

Basics – Installing R

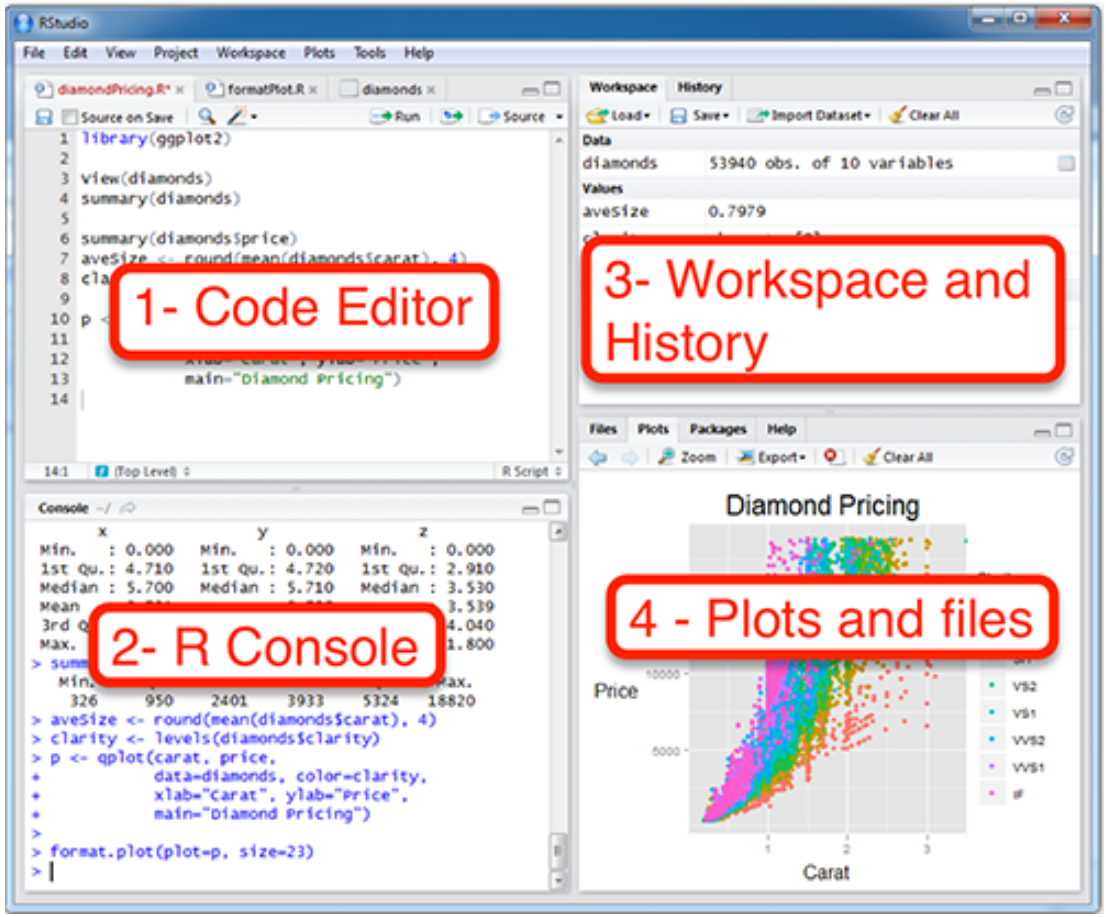
Installation:

1. Download R from https://cran.r-project.org/



2. Tip: Working with this R software is tedious, use another editor instead, e.g. RStudio: http://www.rstudio.com/ide/

Basics – RStudio



Source:

Basics: Object classes

Basics – Object classes

Each single object in R has a specific class. Examples:

```
> class(17)
[1] "numeric"
> class("O'zapft is!")
[1] "character"
```

In a **vector**, all elements have the same class:

```
> a <- c("I'm", "a vector of", 4, "elements")
> class(a[3])
[1] "character"
```

In a **list**, each element has its own class:

```
> b <- list("I'm", "a list of", 4, "elements")
> class(b[[3]])
[1] "numeric"
```

Basics – Type conversion

Arithmetic operations are only valid for some object types:

```
> 2 + 2
[1] 4
> 2 + "2"
Error in 2 + "2": non-numeric argument
```

Use **type conversion** as a solution:

```
> 2 + as.numeric("2")
[1] 4
# Also possible: as.character, as.factor, as.Date, ...
```

Basics – Object classes II

Now let's look at tables:

In a **matrix**, all elements have the same class:

```
> m <- matrix(c("word",1,"word",9), nrow = 2, byrow = TRUE)
> m
     [,1]   [,2]
[1,] "word" "1"
[2,] "word" "9"
> class(m[,2]) # second column
[1] "character"
```

In a **data frame**, each column has its own class:

```
> d <- data.frame("Var1" = c("word","word"), "Var2" = c(2,4)
    )
> d
  Var1 Var2
1 word    2
2 word    4
> class(d[,2]) # second column
```

Using functions and packages

Basics – Working with functions

Working with functions

- First things first: Use the help pages

```
> ?mean
```

- How to call a function:

```
> sample <- c(2,5,3,17)
> mean(sample)
[1] 6.75
```

- How to define your own function:

```
> do_something <- function(a, b) {
+   result <- a + b
+   return(result)
+ }
> do_something(2,4)
[1] 6
```

Basics – Using packages

Working with packages

1. Find a package that does what you need, e.g. using Google
2. Install the package: *necessary once on a PC*

```
> install.packages("paketXY")
```

3. Load the package: *necessary each time you restarted R*

```
> library("paketXY")
```

Working with data

Working with data

Reading in data:

```
> dat <- read.table(
+   file = "Exercises/01/Daten/wdi_data.csv",
+   header = TRUE,
+   sep = ",",
+   dec = ".")
```

Working with variables:

```
> mean(dat$Area)
[1] 491912.6
> dat$new_variable <- dat$CO2emission / dat$Area
```

Working with data

Get to know your data using `str()`:

```
> str(dat)
'data.frame': 315 obs. of  8 variables:
 $ country    : Factor w/ 45 levels "Albania","Andorra",..:
    2 2 2 2 2 2 2 1 1 1 ...
 $ region     : Factor w/ 4 levels "Eastern Europe",..: 3 3
    3 3 3 3 3 3 3 3 ...
 $ year       : int  2005 2006 2007 2008 2009 2010 2011 2005
     2006 2007 ...
 $ CO2emission: num  576 546 539 539 517 ...
 $ Population : int  81223 83373 84878 85616 85474 84419
    82326 3011487 2992547 2970017 ...
 $ Area       : int  470 470 470 470 470 470 470 27400 27400
     27400 ...
 $ BevDichte  : num  173 177 181 182 182 ...
 $ LandJahr   : Factor w/ 315 levels "Albania_2005",..: 8 9
    10 11 12 13 14 1 2 3 ...
```

Working with data - Gotta Plot 'Em All!

Plotting helps you (and others) understanding your data better.

Here are some examples using the ggplot2 package:

```
> library(ggplot2)
> theme_set(theme_bw()) # Set the general plot theme

> ggplot(dat, aes(x = region)) +
+   geom_bar()
> ggplot(dat, aes(x = year, y = CO2emission)) +
+   geom_point()
> ggplot(dat, aes(x = year, y = CO2emission, color = country
    )) +
+   geom_line()
```

Looking for some inspiration? Look no further.

Working with data - Best Practices

Workflow for an analysis:

1. Create a project folder, where you put the data
2. Work with an `.R` file and not in the console
3. First step: Set the working directory using `setwd()`

   NOTE: In Windows you have to turn around the slashes in the path!
4. First step after reading in the data:
   Make sure that every variable has the correct type.
   If not, use `as.numeric()`, `as.factor()` etc.
5. Perform data preparation in `R` (it's reproducible!) and not in Excel
6. Use comments in your code to describe what the code does!

Best Practices

Source: `https://twitter.com/Dale_Masch/status/1138564316594970624/photo/1`

Style and Naming

Style and Naming

"Programs must be written for people to read, and only incidentally for machines to execute." - Harold Abelson

Style and Naming

Good:

```
fit-models.R
day_one
average <- mean(feet / 12 + inches, na.rm = TRUE)
```

Bad:

```
stuff.r
DayOne
average<-mean(feet/12+inches,na.rm=TRUE)
```

What's the matter here?

Style and Naming

Good:

```
fit-models.R # file
day_one
average <- mean(feet / 12 + inches, na.rm = TRUE)
```

Bad:

```
stuff.r # file
DayOne
average<-mean(feet/12+inches,na.rm=TRUE)
```

Cf. Hadley Wickham's style guide

Style and Naming

Good:

```
fit-models.R # file
day_one # object
average <- mean(feet / 12 + inches, na.rm = TRUE)
```

Bad:

```
stuff.r # file
DayOne # object
average<-mean(feet/12+inches,na.rm=TRUE)
```

Cf. Hadley Wickham's style guide

Style and Naming

Good:

```
fit-models.R # file
day_one # object
average <- mean(feet / 12 + inches, na.rm = TRUE) # spacing
```

Bad:

```
stuff.r # file
DayOne # object
average<-mean(feet/12+inches,na.rm=TRUE) # spacing
```

Cf. Hadley Wickham's style guide

Style and Namings

"There are only two hard things in Computer Science: cache invalidation and naming things." - Phil Karlton

Coding demo

Summing things up

Brief coding demo

- use R as calculator
- define objects, vectors
- working with pre-loaded data in R packages

Your turn: set up R and RStudio on your machine and read in the old faithful data set from {`datasets`}

Summing things up

Summing things up - Useful functions

Command	Functionality
`<-` / `=`	Assignment in R
`c()`	Put multiple elements in one vector
`seq()`	Generate numberic sequences
`rep()`	Generate general sequances
`length()`	Length of an object
`getwd()`	Get current path (**w**orking **d**irectory)
`setwd()`	Set current path (**w**orking **d**irectory)
`ls()`	Shows all objects in the workspace
`rm(list=ls())`	Delete all objects from the workspace
`?log`	Call help page for function `log()`
`citation()`	Citation info for `R` itself or specific packages

Summing things up - Useful functions II

Command	Functionality
`summary()`	Useful information regarding the object
`str()`	Even more useful information
`which()`	Get indices, at which a vector is `TRUE`
`paste()`	Paste two character objects together
`head()`	Show only the first elements of an object
`dim()`	Get the dimensions of a dataset
`names()`	Get the names of a dataset
`cbind()`	Merge two datasets regarding the columns
`rbind()`	Merge two datasets regarding the rows
`load()`	Load an object (as `.RData`)
`save()`	Save an object (as `.RData`)

Summing things up

Other questions?

See this cheatsheet, this introduction,

... or ask Stackoverflow/Google

Summing things up

Source: `https://twitter.com/MaartenvSmeden/status/1394374703846989830`

Basic workflow

Basic workflow: Working with data

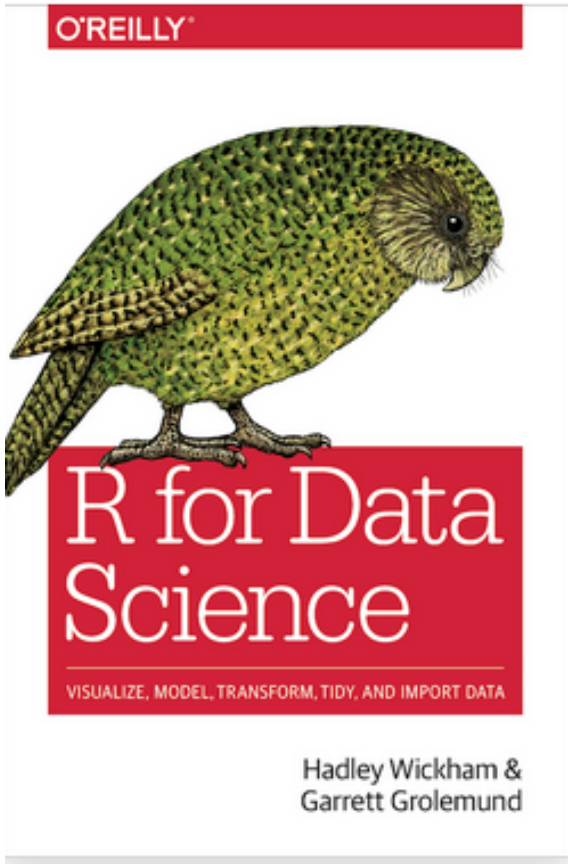
1. Save the csv-file locally
2. Read in the file as dataframe to the R workspace
3. Name the dataframe
4. Inspect the data
5. Manipulate/analyze it
6. Save edited dataframe as csv-file

Your turn: Download the file `construction-components.csv` from moodle and perform the above steps

Literature

Literature

Wickham, Grolemund: R for Data Science



Let's dive right in!

Make yourself familiar with `R`, completing the following tasks:

a. Set up a project folder for this first tutorial, where you save the downloaded dataset `wdi_data.csv` from Moodle as well as a new (and empty) `R` script created with RStudio.

b. Read in the dataset and play around with the functions in `R` that let you get a better image of the data. Create frequency tables with `table()`, plot things with `plot()` (or the `ggplot2` package).