

Introduction to Statistics with R

Session R00: Kickoff and Setup

Marvin Schmitt

Welcome!

Kickoff



- Where do you feel **at home**?
- What is your **background**?
- What is your **motivation** for participating in this course?
- How would you retrospectively notice that the course was good?

Syllabus

Let's take a look at the course website:

<https://marvinschmitt.github.io/IntroStatisticsR/>

Setup

Installing R

Any problems?

Please execute in the console:

```
R.version
```

Installing RStudio

Any problems?

Find out which version of RStudio you have:

```
rstudioapi::versionInfo()$version
```

Installing the packages

Any problems?

Find out which package versions you have:

```
packageVersion("rmarkdown")  
packageVersion("tinytex")  
packageVersion("ggplot2")  
packageVersion("tidyverse")
```


Testfile

Any problems?

R Fundamentals

Basic usage

R is an **interpreted script language**, so you can just write your code into a text file and execute it line by line:

```
1+2
```

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

Let's try it out!

- Open RStudio
- Create a new file: File -> New File -> R Script
- Type some expression and execute each line with Ctrl + Enter (Windows) or Cmd + Enter (Mac)

Basic calculations

R supports basic arithmetic:

```
1+2  
3-4  
5*6  
7/8  
sqrt(9)  
10**2  
sin(pi/2)  
log(55)
```

Comments

If you type #, you start a **comment** for the rest of the line.
Commented text is not evaluated!

```
# This is a program that adds my two favorite numbers, namely 2 and 8.  
# I thank all my sponsors and my mom for the invaluable support  
# while writing this program.
```

```
2+8    # this line is where I do all the addition magic!
```

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

```
# End of the program  
# Thank you!
```

Use comments where ever you can. Your future self will thank you!

The help function

If you come across a command that you don't know (yet), you can use R's internal help. Just move the cursor to the function and press F1. Alternatively, you can call `help()` and type the command you want to help for in the parentheses:

```
help(log)      # Read the help for the log() function
```

Variables

Just like in math, we often want to store numbers in variables:

```
a = 5  
b = 10  
a+b  
## [1] 15
```

Arrays

Arrays are vectors with homogeneous entries (e.g. only numbers or only text). Arrays are constructed using the command `c()`. We can also store vectors in variables.

```
age = c(27, 28, 32, 30, 27)
institute = c('IDF', 'Slavistik', 'Germanistik', 'IDF', 'IUED')
ID = 1:5      # shortcut for c(1, 2, 3, 4, 5)
```

```
age
```

```
## [1] 27 28 32 30 27
```

```
institute
```

```
## [1] "IDF"          "Slavistik"    "Germanistik" "IDF"          "IUED"
```

```
ID
```

```
## [1] 1 2 3 4 5
```


We can do **calculations** on arrays:

```
age * 365
```

```
## [1] 9855 10220 11680 10950 9855
```

```
ID ^ 2 + (ID-20)^2 - 200
```

```
## [1] 162 128 98 72 50
```

```
age + 10 * ID
```

```
## [1] 37 48 62 70 77
```

We can **access** array elements by their index with `v[index]`. The index can also be an array if we want to access multiple entries:

```
age[1]
```

```
## [1] 27
```

```
institute[c(1, 2)]
```

```
## [1] "IDF"      "Slavistik"
```

```
age[2:4]
```

```
## [1] 28 32 30
```

To **append** items to an array, use this trick: Just create a **new array** with the original array as the first argument:

```
age = c(age, 31)
institute = c(institute, 'Germanistik')
ID = c(ID, 6)
```

Dataframes

Dataframes are the fundamental data structure for data sets in R.

```
df = data.frame(ID, age, institute)
df
```

```
##   ID age  institute
## 1  1  27         IDF
## 2  2  28   Slavistik
## 3  3  32 Germanistik
## 4  4  30         IDF
## 5  5  27        IUED
## 6  6  31 Germanistik
```

Add an additional column `matnr` that contains the matricial number:

```
matnr = c(3324567, 4356475, 4335263, 4231526, 3324153, 3657687)
df = data.frame(df, matnr)
df
```

```
##   ID age  institute  matnr
## 1  1  27         IDF 3324567
## 2  2  28   Slavistik 4356475
## 3  3  32 Germanistik 4335263
## 4  4  30         IDF 4231526
## 5  5  27        IUED 3324153
## 6  6  31 Germanistik 3657687
```

Access element in a data frame with index `df[row, column]`:

```
df[1, 3]      # row 1, column 3
```

```
## [1] "IDF"
```

```
df[2, 1]      # row 2, column 1
```

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

```
df[1:4, 1:3]  # rows 1-4, columns 1-3
```

```
##   ID age  institute
## 1  1  27      IDF
## 2  2  28 Slavistik
## 3  3  32 Germanistik
## 4  4  30      IDF
```

Access column by name `df$columnname`:

```
df$age        # column 'age'
```

```
## [1] 27 28 32 30 27 31
```

```
df$institute  # column 'institute'
```

```
## [1] "IDF"      "Slavistik" "Germanistik" "IDF"      "IUED"
## [6] "Germanistik"
```

`df$columnname` returns a vector, so we can handle it like one:

```
df$age
```

```
## [1] 27 28 32 30 27 31
```

```
df$age * 365
```

```
## [1] 9855 10220 11680 10950 9855 11315
```

```
df$age[2:4]
```

```
## [1] 28 32 30
```

```
df$ID + df$age
```

```
## [1] 28 30 35 34 32 37
```

Reading and writing files

You often find yourself with a data file (e.g. .csv or Excel) that you want to analyze in R. The following slides will cover:

- Setting your **working directory**
- **Writing** .csv files
- **Reading** .csv files

You need to set your **working directory**. That is the base path from where all your references originate. You can set it using the `setwd()` command:

```
setwd("C:\\Users\\Marvin\\IntroStatisticsR\\material\\R00_setup") # Windows
setwd("~/IntroStatisticsR/material/R00_setup")                 # Mac
```

Print the current working directory using `getwd()`:

```
getwd()
## [1] "/Users/marvin/IntroStatisticsR/material/R00_setup"
```

Use this trick to set the working directory to where your current file is located:

```
setwd(dirname(rstudioapi::getSourceEditorContext()$path))
```

Write a dataframe into a file using the `write.csv` command:

```
write.csv(df, file='R00_student_data.csv', row.names=FALSE)
```

- `df` is the data frame that you want to save.
- `file='R00_student_data.csv'` specifies the file name relative to the working directory.
- `row.names=FALSE` controls that we do not want to save row names. That's optional.

Read a data set using the `read.csv()` command:

```
df_new = read.csv(file="R00_student_data.csv")  
df_new
```

```
##   ID age  institute  matnr  
## 1  1  27         IDF 3324567  
## 2  2  28   Slavistik 4356475  
## 3  3  32 Germanistik 4335263  
## 4  4  30         IDF 4231526  
## 5  5  27         IUED 3324153  
## 6  6  31 Germanistik 3657687
```

- `file="R00_student_data.csv"` provides the file name relative to the working directory.
- We save the result of the `read.csv()` command into the object `df_new`.

RMarkdown

RMarkdown is a very simple way to create *beautiful* output files for your program code. Let's head right into RStudio and see how we can use RMarkdown.

Task: Translate all the previous code into a `.Rmd` notebook and write some wrapper text. Use the following header:

```
---  
title: "Introduction to Statistics with R -- R00: Setup"  
subtitle: "RMarkdown Exercise"  
author: "<YOUR NAME>"  
output: pdf_document  
---
```