

# A1 GETTING STARTED WITH R

Jan-Philipp Kolb

21 Februar, 2020

# COURSE OBJECTIVES

- Perform your data analysis in a literate programming environment
- Import and manage structured and unstructured data
- Manipulate, transform, and summarize your data
- Join disparate data sources
- Methodically explore and visualize your data
- Perform iterative functions
- Write your own functions

... all with R!

# INTRODUCTION ROUND

## PLEASE TELL ME SHORTLY...

- Where are you from? What are you studying/working?
- What is your experience level in R/other programming languages?
- What are your expectations of this course?
- Where do you think you can use R in the future?

- Usually we have big differences in knowledge and abilities of the participants - please tell, if it is too fast or slow.
- We have lots of hands-on coding **exercises** - later you can only learn on your own
- We have many **examples** - try them
- If there are questions - always ask
- R is more fun together - ask your neighbor - strong proponent of collaborative work!

# SOURCES OF THIS COURSE

## SOURCES FOR FIGURES, TEXT, EXERCISES ETC:

- If the source is a website, the links are often in the header or in bold somewhere on the slide.
- At the end of a chapter, we often have additional links to read on.
- Please ask us, if something is unclear.

# REASONS FOR USING R...

- ... because it is an **open source language**
- ... outstanding graphs - **graphics, graphics, graphics**
- ... relates to other languages - **R can be used in combination with other programs** - e.g. **data linking**
- ... R can be used **for automation**
- ... Vast Community - **you can use the intelligence of other people ;-)**
- ...

- R can be downloaded for **free**.



[\[Home\]](#)

**Download**

[CRAN](#)

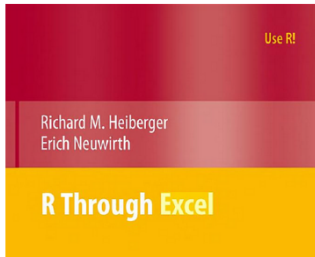
## The R Project for Statistical Computing

### Getting Started

R is a free software environment for statistical computing and graphics. It compiles and runs on a wide variety of UNIX platforms, Windows and MacOS. To [download R](#), please choose your preferred [CRAN mirror](#).

- R is a **scripting language**
- R is becoming more **popular**
- **Good** possibilities for **visualization**

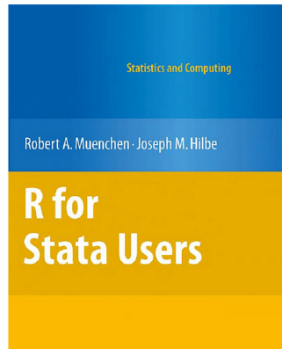
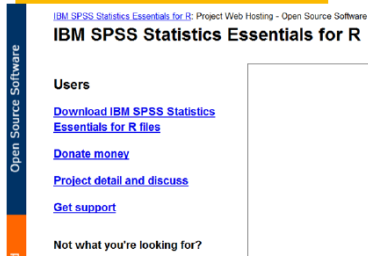
# R CAN BE USED IN COMBINATION...



SASmixed



rPython R package



- Interface to: **Python, Excel, SPSS, SAS, Stata**



# The popularity of R-packages



<http://www.r-project.org/>



*CRAN*

[Mirrors](#)

[What's new?](#)

[Task Views](#)

[Search](#)

*About R*

[R Homepage](#)

[The R Journal](#)

*Software*

[R Sources](#)

[R Binaries](#)

[Packages](#)

[Other](#)

## The Comprehensive R Archive Network

### Download and Install R

Precompiled binary distributions of the base system and contributed packages, **Windows and Mac** users most likely want one of these versions of R:

- [Download R for Linux](#)
- [Download R for \(Mac\) OS X](#)
- [Download R for Windows](#)

R is part of many Linux distributions, you should check with your Linux package management system in addition to the link above.

### Source Code for all Platforms

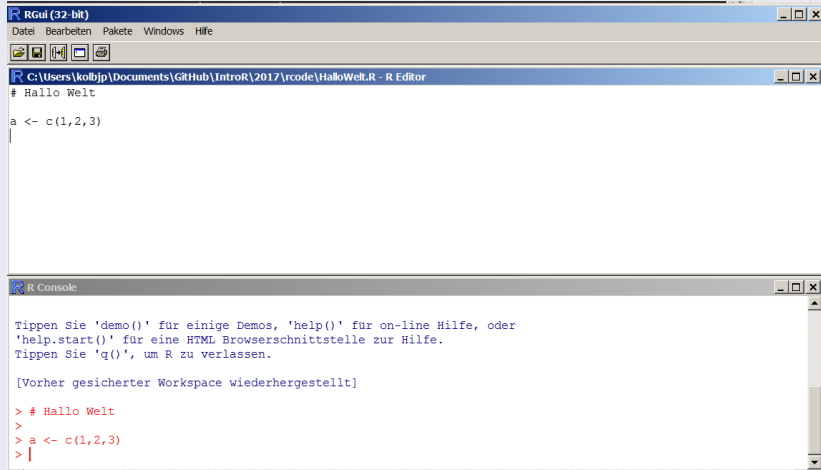
Windows and Mac users most likely want to download the precompiled binaries listed in the upper box, not the source code. The sources have to be compiled before you can use them. If you do not know what this means, you probably do not want to do it!

- The latest release (Friday 2017-04-21, You Stupid Darkness) [R-3.4.0.tar.gz](#), read [what's new](#) in the latest version.

# OPEN SOURCE PROGRAMM R

- R is a free, non-commercial implementation of the S programming language (by AT&T Bell Laboratories)
- Free participation - modular structure

## THIS IS BASE R:



The screenshot displays the R GUI (32-bit) interface. The top window, titled "R GUI (32-bit)", contains a menu bar with "Datei", "Bearbeiten", "Pakete", "Windows", and "Hilfe". Below the menu is a toolbar with icons for file operations. The main editor window shows the file path "C:\Users\kolbjp\Documents\GitHub\IntroR\2017\rcode\HalloWelt.R - R Editor" and the following R code:

```
# Hallo Welt  
  
a <- c(1,2,3)
```

The bottom window, titled "R Console", displays the following text:

```
Tippen Sie 'demo()' für einige Demos, 'help()' für on-line Hilfe, oder  
'help.start()' für eine HTML Browserschnittstelle zur Hilfe.  
Tippen Sie 'q()', um R zu verlassen.  
  
[Vorher gesicherter Workspace wiederhergestellt]  
  
> # Hallo Welt  
>  
> a <- c(1,2,3)  
> |
```

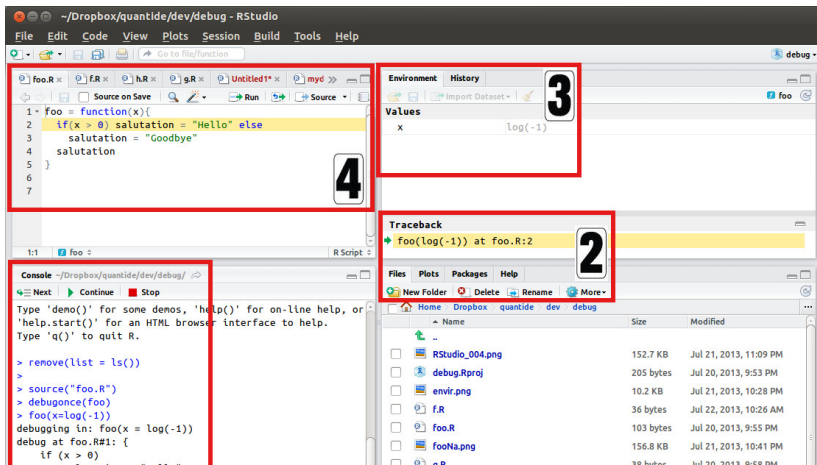
But many people use a graphical user interface (GUI) or a integrated development interface (IDE).

For the following reasons:

- Syntax highlighting
- Auto-completion
- Better overview on graphics, libraries, files, ...

# VARIOUS TEXT EDITORS / IDEs

- **Gedit** with R-specific Add-ons for Linux
- **Emacs** and **ESS** (Emacs speaks statistics)- An extensible, customizable, free/libre text editor — and more.
- I use **Rstudio!**



## Script files

- Saves your script
- Allows code & comments
- Can have multiple files open at a time

## Console/Command line

- Can use as calculator
- Does not save code
- This is where your output is displayed

The screenshot displays the RStudio IDE with the following components:

- Script Editor (Top Left):** Contains R code for a function `block_summary` and its execution. The code defines parameters `m`, `n`, `r`, and `na.rm`, and calculates the sum of `m` through `n`. The execution shows the output of the function call.
- Console (Bottom Left):** Displays the output of the script execution, including the results of the `block_summary` function and the calculation of the sum of `m` through `n`.
- Environment (Top Right):** Shows the current environment with variables `m`, `n`, `r`, and `t` assigned values. It also lists functions available in the environment.
- Help Pane (Bottom Right):** Displays the documentation for the `mean` function, including its description, usage, and arguments.

## Workspace environment

- Holds your objects
- Can review history

## Misc - Displays:

- files in working directory
- plots when produced
- help files/search

# IMPORTANT RSTUDIO BUTTONS



create a new script



open an existing script



run line where cursor is

# R AS A CALCULATOR

```
3 + 2 / 10^2 # Uses PEMDAS convention (order of operations)
```

```
## [1] 3.02
```

```
3 + (2 / 10^2)
```

```
## [1] 3.02
```

```
(3 + 2) / 10^2
```

```
## [1] 0.05
```

```
1 / 19^4 # scientific notation is used for large numbers
```

```
## [1] 7.67336e-06
```

```
1/0 # Undefined calculations
```

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

```
Inf - Inf
```

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



# EXERCISE: PREPARATION

- Check if R is installed on your computer.
- If not, download **R** and install it.
- Check if Rstudio is installed.
- If not - **install** Rstudio.
- Start RStudio. Go to the console (lower left window) and write

```
3+2
```

- If there is not already an editor open in the upper left window, then go to the file menu and open a new script. Check the date with `date()` and the R version with `sessionInfo()`.

```
date()
```

```
sessionInfo()
```

# EXERCISE: SEE WHERE THINGS HAPPEN

- Create a new .R script named `my_first_script.R`
- Write and execute the following code in the .R script and identify where in Rstudio the outputs can be found.

```
mtcars
?sum
hist(mtcars$mpg)
random_numbers <- runif(40)
history()
```

# R IS A OBJECT-ORIENTED LANGUAGE

## VECTORS AND ASSIGNMENTS

- R is a object-oriented language
- `<-` is the assignment operator

```
b <- c(1,2) # create an object with the numbers 1 and 2
```

- A function can be applied to this object:

```
mean(b) # computes the mean
```

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

We can learn something about the properties of the object:

```
length(b) # b has the length 2
```

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

```
sqrt(b) # the square root of b
```

```
## [1] 1.000000 1.414214
```

# FUNCTIONS IN BASE-PACKAGE

Function	Meaning	Example
<code>str()</code>	Object structure	<code>str(b)</code>
<code>max()</code>	Maximum	<code>max(b)</code>
<code>min()</code>	Minimum	<code>min(b)</code>
<code>sd()</code>	Standard deviation	<code>sd(b)</code>
<code>var()</code>	Variance	<code>var(b)</code>
<code>mean()</code>	Mean	<code>mean(b)</code>
<code>median()</code>	Median	<code>median(b)</code>

These functions only need one argument.

# FUNCTIONS WITH MORE ARGUMENTS

## OTHER FUNCTIONS NEED MORE ARGUMENTS:

Argument	Meaning	Example
quantile()	90 % Quantile	quantile(b,.9)
sample()	Draw a sample	sample(b,1)

```
quantile(b,.9)
```

```
## 90%
```

```
## 1.9
```

```
sample(b,1)
```

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

# EXAMPLES - FUNCTIONS WITH MORE THAN ONE ARGUMENT

```
max(b); min(b)
```

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

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

```
sd(b); var(b)
```

```
## [1] 0.7071068
```

```
## [1] 0.5
```

## FUNCTIONS WITH ONE ARGUMENT

```
mean(b)
```

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

```
median(b)
```

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

# EXERCISE: ASSIGNMENTS AND FUNCTIONS

Create a vector `b` with the numbers from 1 to 5 and calculate ...

- 1 the mean
- 2 the variance
- 3 the standard deviation
- 4 the square root from the mean

<http://cran.r-project.org/doc/manuals/R-intro.html>

## An Introduction to R

### Table of Contents

#### [Preface](#)

#### [1 Introduction and preliminaries](#)

##### [1.1 The R environment](#)

##### [1.2 Related software and documentation](#)

##### [1.3 R and statistics](#)

##### [1.4 R and the window system](#)

##### [1.5 Using R interactively](#)

##### [1.6 An introductory session](#)

##### [1.7 Getting help with functions and features](#)

##### [1.8 R commands, case sensitivity, etc.](#)

##### [1.9 Recall and correction of previous commands](#)

##### [1.10 Executing commands from or diverting output to a file](#)

##### [1.11 Data permanency and removing objects](#)



# EXERCISE: ECONOMIC ORDER QUANTITY MODEL

## Economic order quantity

From Wikipedia, the free encyclopedia

In [inventory management](#), **economic order quantity (EOQ)** is the order quantity that minimizes the total [holding costs](#) and [ordering costs](#). It is one of the oldest classical [production scheduling](#) models. The model was developed by [Ford W. Harris](#) in 1913, but R. H. Wilson, a consultant who applied it extensively, and K. Andler are given credit for their in-depth analysis.<sup>[1]</sup>

## ECONOMIC ORDER QUANTITY MODEL

$$Q = \sqrt{\frac{2DK}{h}}$$

## CALCULATE Q WHERE:

- $D = 1000$
- $K = 5$
- $h = 0.25$

# R DATA TYPES

- R supports a few basic data types: integer, numeric, logical, character/string, factor, and complex

## LOGICAL

– binary, two possible values represented by TRUE and FALSE

```
x <- c(3,7, 1, 2)
```

```
x > 2
```

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

```
x == 2
```

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

```
!(x < 3)
```

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

```
which(x > 2)
```

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

# CHARACTER VECTORS

```
y <- c("a","bc","def")  
length(y)
```

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

```
nchar(y)
```

```
## [1] 1 2 3
```

```
y == "a"
```

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

```
y == "b"
```

```
## [1] FALSE FALSE FALSE
```

# OBJECT STRUCTURE

```
str(b) # b is a numeric vector
```

```
## num [1:2] 1 2
```

VARIABLE TYPE CHARACTER

```
a <- letters  
length(letters)
```

```
## [1] 26
```

```
a[1:4]
```

```
## [1] "a" "b" "c" "d"
```

```
str(a)
```

```
## chr [1:26] "a" "b" "c" "d" "e" "f" "g" "h" "i" "j" "k" "l" "
```

# PROBLEMS WITH CHARACTER VECTOR

```
mean(b)
```

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

```
(b1 <- c(b, "a"))
```

```
## [1] "1" "2" "a"
```

```
mean(b1)
```

```
## Warning in mean.default(b1): argument is not numeric or logical
```

```
## [1] NA
```

- All elements in a vector must be of the same type. R coerces the elements to a common type
- `c(1.2,3,TRUE)` – In this case all elements are coerced to numeric.

```
x <- c(TRUE,FALSE,TRUE)
c(1.2,x)
```

```
## [1] 1.2 1.0 0.0 1.0
```

```
y <- c("2","3",".2")
c(1.2,y, x)
```

```
## [1] "1.2" "2" "3" ".2" "TRUE" "FALSE" "TRUE"
```

- Sometimes this coercion occurs to perform an arithmetic operation:

```
1 + x
```

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

# PERFORM THE COERCION

- Other times we need to perform the coercion

```
c(1.2,y)
```

```
## [1] "1.2" "2"   "3"   ".2"
```

```
c(1.2,as.numeric(y))
```

```
## [1] 1.2 2.0 3.0 0.2
```

- Aggregator functions - `sum`, `mean`, `range`, `min`, `max`, `summary`, `table`, `cut`, ...
- `class(x)` – returns the type of an object.
- `is.logical(x)` – tells us whether the object is a logical type. There is also `is.numeric`, `is.character`, `is.integer`
- `is.null` – determines whether an object is empty, i.e. has no content. 'NULL' is used mainly to represent the lists with zero length, and is often returned by expressions and functions whose value is undefined.



# COERCE OBJECTS FROM ONE TO ANOTHER

- `as.numeric(x)` – we use the `as-`type functions to coerce objects from one type (e.g. logical) to another, in this case numeric.
- There are several of these functions, including `as.integer`, `as.character`, `as.logical`

```
x <- c("1",2,"one","1plus","2_and")  
as.numeric(x)
```

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

# HOW TO LEARN AFTER THIS WORKSHOP

*How to actually learn any new programming concept*

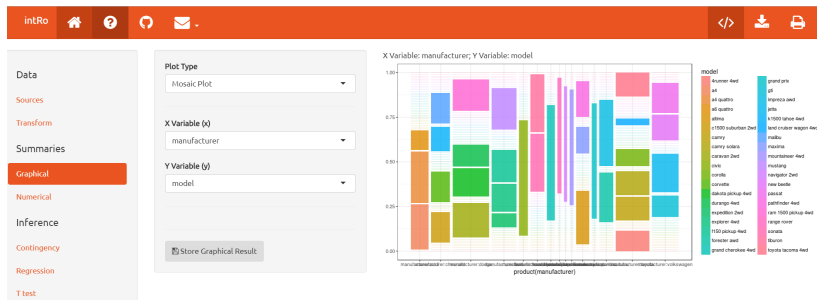


*Essential*

Changing Stuff and  
Seeing What Happens

# SHINY APP - INTRO R

<http://www.intro-stats.com/>



# SOME LINKS TO READ ON

- Six **reasons** to use **Rstudio**.
- **Why you should learn R first for data science**
- **RStudio – Infoworld 2015 Technology of the Year Award Recipient!**
- **Why the R programming language is good for business?**
- **Have a look at R-bloggers**
- **Comparisson between python and R**
- **R and Stata Side-by-side**
- **AWESOME R**
- **1000 R tutorials/Links**
- **Learn R by watching two-minute videos**

## R FOR STATA USERS

- Oscar Torres-Reyna - **Exploring Data and Descriptive Statistics (using R)**