

Introduction to R

Getting started with R

Jan-Philipp Kolb

05 März, 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?

Preliminaries

- 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** - at the end 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 pink somewhere on the slide.
- At the end of a chapter, you'll find additional links to read on.
- Please ask, 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 ;-)**
- ...

Advantages of R

- 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...

Use R!

Richard M. Heiberger
Erich Neuwirth

R Through Excel

Stata

SASmixed

Open Source Software

IBM SPSS Statistics Essentials for R: Project Web Hosting - Open Source Software

IBM SPSS Statistics Essentials for R

Users

[Download IBM SPSS Statistics Essentials for R files](#)

[Donate money](#)

[Project detail and discuss](#)

[Get support](#)

Not what you're looking for?

R-Forge

rPython R package

Statistics and Computing

Robert A. Muenchen · Joseph M. Hilbe

R for Stata Users

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

The popularity of R-packages



Download R:

<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:

Graphical user interface

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!**



An IDE that was built just for R

- Syntax highlighting, code completion, and smart indentation
- Execute R code directly from the source editor
- Quickly jump to function definitions



Bring your workflow together

- Integrated R help and documentation
- Easily manage multiple working directories using projects
- Workspace browser and data viewer



Powerful authoring & Debugging

- Interactive debugger to diagnose and fix errors quickly
- Extensive package development tools
- Authoring with Sweave and R Markdown

RStudio

The functionality of the panes in 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 shows the RStudio interface with the following panes:

- Script Editor (Top Left):** Displays R code for a function `block_summary`. The code includes comments and a function definition that checks for numeric inputs and calculates a summary.
- Environment/History (Top Right):** Shows the current workspace environment with variables `m` (10), `n` (100), `r` (0.77), and `t` (5). It also lists functions like `block_summary`, `cum_appx`, `cum_exact`, `lc_rate`, `midpoint`, `natural_slope`, and `unit_curve`.
- Console (Bottom Left):** Shows the output of the `block_summary` function call, displaying the results for 'block hours', 'midpoint unit', and 'midpoint hours'.
- Files/Plots/Packages/Help/Viewer (Bottom Right):** Shows the R 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

Rstudio - script and console

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 shows the RStudio application window. The top toolbar includes icons for file operations and a 'Go to file/function' search bar. The main editor pane displays two open files: 'Initial_functions.R' and 'plot_functions.R'. The 'Initial_functions.R' file is active, showing R code with line numbers 231 to 250. The code defines a function 'block_summary' that takes parameters 't', 'm', 'n', 'r', and 'na.rm'. It includes a check for numeric inputs and a 'stop' message if they are not numeric. The console window at the bottom shows the output of the 'block_summary' function call, displaying three values: 3668.436, 44.03189, and 40.31249.

```

231 }
232 }
233
234
235 # Provides the summary for the block containing units m through n, n > m
236 # t = time for firsts unit
237 # m = lower bound unit of production block
238 # n = upper bound unit of production block
239 # r = learning curve rate
240 block_summary <- function(t, m, n, r, na.rm = FALSE){
241
242   if(!is.numeric(t) | !is.numeric(m) | !is.numeric(n) | !is.numeric(r)){
243     stop('This function only works for numeric inputs!\n',
244         'You have provided objects of the following classes:\n',
245         't: ', class(t), '\n',
246         'm: ', class(m), '\n',
247         'n: ', class(n), '\n',
248         'r: ', class(r))
249   }
250
276:20 block_summary(t, m, n, r, na.rm)
  
```

Console ~ /Desktop/Personal/Academia/Learning Curve R Package/

```

$'block hours'
[1] 3668.436

$'midpoint unit'
[1] 44.03189

$'midpoint hours'
[1] 40.31249
  
```

Rstudio - Environment and help

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
```

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
```

Functions in base-package

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

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] 2
```

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); median(b)
```

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

```
## [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

Overview commands

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

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.^[1]

{width=110%}

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
```

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" "m" "n" "o" "p"
```

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: returning
```

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

Coercion

- All elements in a vector must be of the same type. R coerces the elements to a common type
- In the following 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
```

- To character:

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

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

- The following arithmetic operation works:

```
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
```

Information about Vectors

- 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` and `is.integer`
- `is.null` – determines whether an object is empty.
- `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 get help?

- **To get help in general:**

```
help.start()
```

- Use ? to get help.

```
?mean
```

- `example(lm)` gives an example for a linear regression

```
example(lm)
```

- **Online documentation for most of the functions:**

Again, we get help with the question mark

```
?paste
```

Different sections in the help:

- Description
- Usage
- Arguments
- Value
- Author(s)
- See Also
- Examples

Vignettes

- A vignette is a paper that present the most important functions of a package
- You get many reproducible examples
- Vignettes are a rather new tool, that is why not every package has a vignette

```
browseVignettes()
```

- to get a vignette:

```
vignette("osmdata")
```

An example for a vignette - package `osmdata`

<https://cran.r-project.org/web/packages/osmdata/vignettes/osmdata.html>

1. Introduction

`osmdata` is an R package for downloading and using data from OpenStreetMap ([OSM](#)). OSM is a global open access mapping project, which is free and open under the [ODbL licence](#) [[@OpenStreetMap](#)]. This has many benefits, ensuring transparent data provenance and ownership, enabling real-time evolution of the database and, by allowing anyone to contribute, encouraging democratic decision making and citizen science [[@johnson_models_2017](#)]. See the [OSM wiki](#) to find out how to contribute to the world's open geographical data commons.

Unlike the [openStreetMap](#) package, which facilitates the download of raster tiles, `osmdata` provides access to the vector data underlying OSM.

`osmdata` can be installed from CRAN with

```
install.packages("osmdata")
```

and then loaded in the usual way:

```
library(osmdata)
```

```
## Data (c) openStreetMap contributors, ODbL 1.0. http://www.openstreetmap.org/copyright
```

The development version of `osmdata` can be installed with the `devtools` package using the following command:

```
devtools::install_github('osmdatar/osmdata')
```

Demos

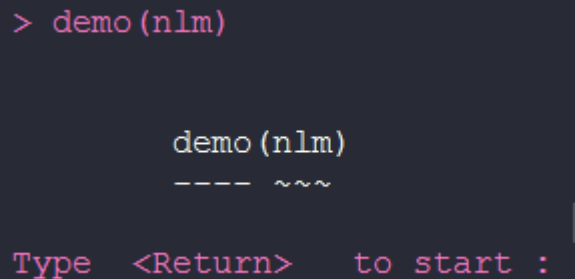
- for some packages you have demos:

```
demo() # shows all available demos
demo(package = "httr") # Show all demos in a package

# Run a specific demo:
demo("oauth1-twitter", package = "httr")
```

- if you run a demo, the code is shown in the console

```
demo(nlm)
```



```
> demo(nlm)

      nlm
      ---- ~~~

Type <Return> to start :
```


The function `apropos`

- searches everything about the given string

```
apropos("lm")
```

```
## [1] ".colMeans"      ".lm.fit"         "colMeans"        "confint.lm"
## [5] "contr.helmert"   "dummy.coef.lm"   "getAllMethods"    "glm"
## [9] "glm.control"     "glm.fit"         "KalmanForecast"   "KalmanLike"
## [13] "KalmanRun"       "KalmanSmooth"    "kappa.lm"        "lm"
## [17] "lm.fit"          "lm.influence"    "lm.wfit"          "model.matrix.l
## [21] "nlm"            "nlminb"          "predict.glm"      "predict.lm"
## [25] "residuals.glm"   "residuals.lm"    "summary.glm"      "summary.lm"
```

Search engine for the R-Site

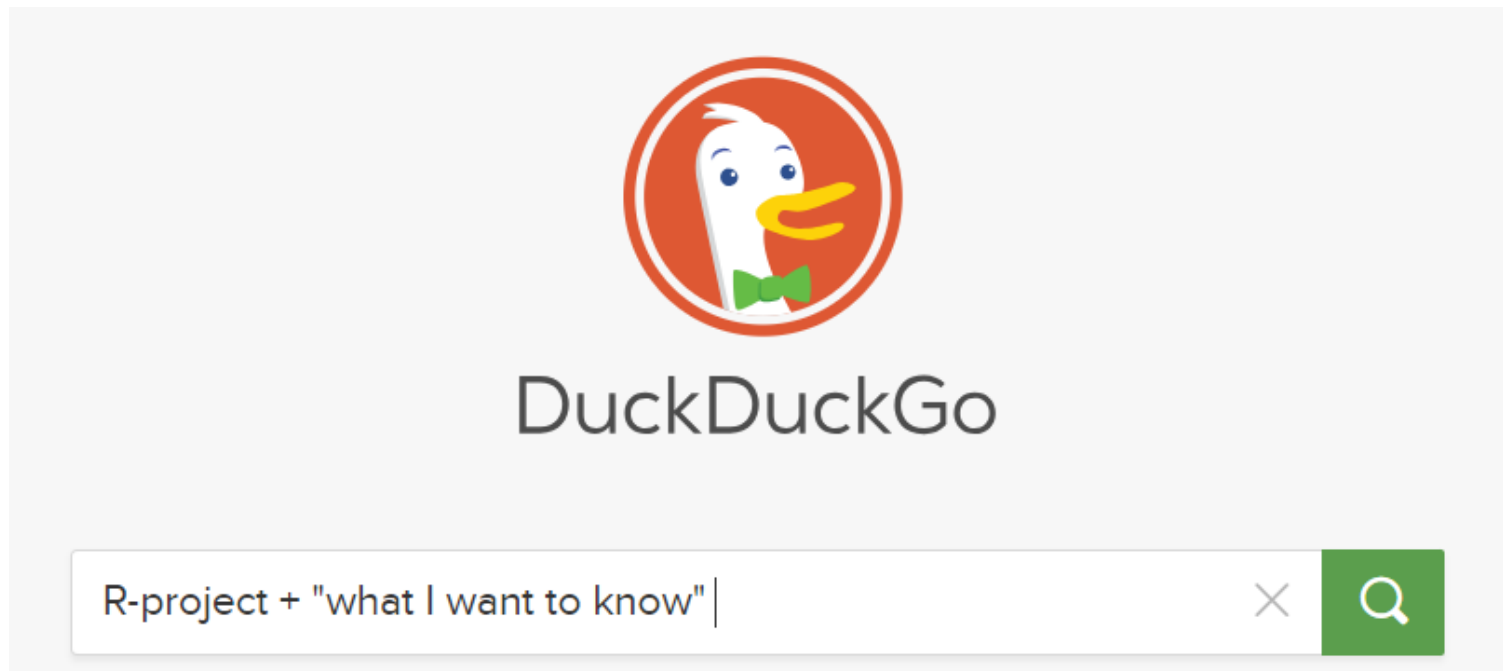
```
RSiteSearch("glm")
```

Usage of search engines

- I use **duckduckgo**:
- just add "with R" at the end of any search. Or:

R-project + "what I want to know"


- this works of course for all search engines!



{ height=80% }

Stackoverflow

- A searchable Q&A site oriented toward programming issues.
- Is not focused on R - but **many discussions on R**
- Very detailed discussions


stackoverflow

[Questions](#)
[Jobs](#)
[Documentation](#)
[Tags](#)
[Users](#)

[?](#)
[Menu](#)
[Log In](#)
[Sign Up](#)

Tagged Questions

[info](#)
[newest](#)
[8 featured](#)
[frequent](#)
[votes](#)
[active](#)
[unanswered](#)

R is a free, open-source programming language and software environment for statistical computing, bioinformatics, and graphics. Please supplement your question with a minimal reproducible example. Use dput() for data and specify all non-base packages with library calls. For statistical questions ...

[learn more...](#) [top users](#) [synonyms \(2\)](#) [r jobs](#)

1776
votes

22
answers

147k
views

How to make a great R reproducible example?


When discussing performance with colleagues, teaching, sending a bug report or searching for guidance on mailing lists and here on SO, a reproducible example is often asked and always helpful. What ...

[r](#)
[r-faq](#)

[community wiki](#)
11 revs, 8 users 54%
[Hack-R](#)

22,187
frequent questions tagged

[r](#)
[about »](#)


R Language
DOCUMENTATION
[Find a request to handle or browse 121 topics.](#)

Related Tags

[ggplot2](#) × 2875

[dataframe](#) × 1351

[plot](#) × 1105

A cheatsheet for base R

<https://www.rstudio.com/resources/cheatsheets/>

Base R

Cheat Sheet

Getting Help

Accessing the help files

?mean
Get help of a particular function.
help.search('weighted mean')
Search the help files for a word or phrase.
help(package = 'dplyr')
Find help for a package.

More about an object

str(iris)
Get a summary of an object's structure.
class(iris)
Find the class an object belongs to.

Using Packages

install.packages('dplyr')
Download and install a package from CRAN.

library(dplyr)
Load the package into the session, making all its functions available to use.

dplyr::select
Use a particular function from a package.

data(iris)
Load a built-in dataset into the environment.

Vectors

Creating Vectors

c(2, 4, 6)	2 4 6	Join elements into a vector
2:6	2 3 4 5 6	An integer sequence
seq(2, 3, by=0.5)	2.0 2.5 3.0	A complex sequence
rep(1:2, times=3)	1 2 1 2 1 2	Repeat a vector
rep(1:2, each=3)	1 1 1 2 2 2	Repeat elements of a vector

Vector Functions

sort(x) Return x sorted.	rev(x) Return x reversed.
table(x) See counts of values.	unique(x) See unique values.

Selecting Vector Elements

By Position

x[4]	The fourth element.
x[-4]	All but the fourth.
x[2:4]	Elements two to four.
x[-(2:4)]	All elements except two to four.
x[c(1, 5)]	Elements one and five.

Programming

For Loop

```
for (variable in sequence){
  Do something
}
```

Example

```
for (i in 1:4){
  j <- i + 10
  print(j)
}
```

While Loop

```
while (condition){
  Do something
}
```

Example

```
while (i < 5){
  print(i)
  i <- i + 1
}
```

If Statements

```
if (condition){
  Do something
} else {
  Do something different
}
```

Example

```
if (i > 3){
  print('Yes')
} else {
  print('No')
}
```

Functions

```
function_name <- function(var){
  Do something
  return(new_variable)
}
```

Example

```
square <- function(x){
  squared <- x*x
  return(squared)
}
```

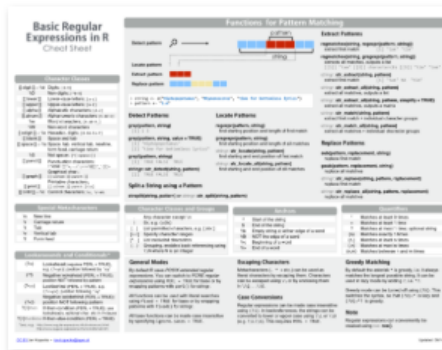
Reading and Writing Data

Also see the **readr** package.

Input	Ouput	Description
df <- read.table('file.txt')	write.table(df, 'file.txt')	Read and write a delimited text file.

More cheatsheets

Regular Expressions



Basics of regular expressions and pattern matching in R by Ian Kopacka. Updated 09/16.

DOWNLOAD

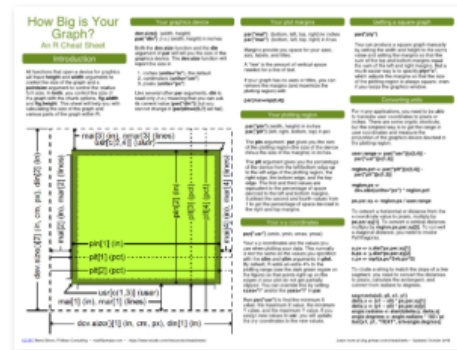
The leaflet package



Interactive maps in R with leaflet, by Kejia Shi. Updated 05/17.

DOWNLOAD

How big is your graph?



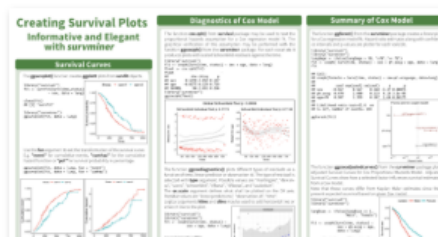
Graph sizing with base R by Stephen Simon. Updated 10/16.

DOWNLOAD

The eurostat package



The survminer package



The sjmisc package



Quick R

- Always a page with examples and help concerning a topic
- Example: **Quick R - Getting Help**



[R Tutorial](#) | [R Interface](#) | [Data Input](#) | [Data Management](#) | [Statistics](#) | [Advanced Statistics](#) | [Graphs](#) | [Advanced Graphs](#)

< R Interface

Getting Help

The Workspace

Input/Output

Packages

Graphic User Interfaces

Customizing Startup

Publication Quality Output

Batch Processing

Reusing Results

Getting Help

Once R is installed, there is a comprehensive built-in help system. At the program's command prompt you can use any of the following:

```
help.start()  # general help
help(foo)     # help about function foo
?foo          # same thing
apropos("foo") # list all functions containing string foo
example(foo)  # show an example of function foo
```

Exercise: Getting help

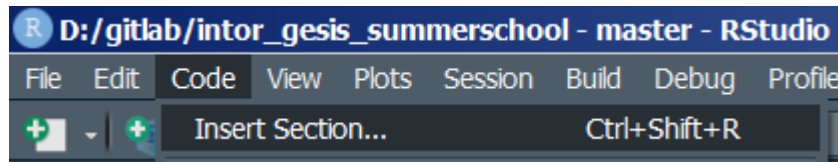
Exercise on help

- Try the command `?which.min` This opens a help page in the lower right window of RStudio. What does the function do?
- You must know the name of the function in order to open the help page as above. Sometimes you do not know the name of the R functions; then a **search engine** can often help you. Try, for example, to search the text `R minimum vector`.

Structure your code

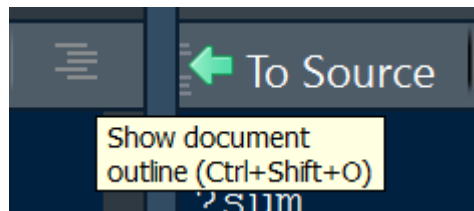
Work with sections

- In a Rscript you can use "Ctrl + Shift + R" to include a section



Outline R-code

- Use as many comments as possible
- Use shortcut "Ctrl + Shift + R" to insert a new section



Save your Work

- When conducting research, keeping all of your code, data, and files in the same place is useful.
- Many journals now require that you make, e.g., your data and code publicly available.
- Now is the time to invest in file structures and versioning programs (e.g., Dropbox and Github).
- Save your script file often to prevent loss of your work.
- Also save your workspace in R in order to save time.
- If you do this, you will be able to load your console in the future as though you had already completed all of the operations that you ran from your script file.

Where to find routines

- Many functions are included in basic R
- Many specific functions are integrated in additional libraries
- R can be modularly extended by so-called packages or libraries
- Most important packages hosted on CRAN (15349 at Do Mrz 05)
- Further packages can be found e.g. at **bioconductor**

Overview R packages

Installation of packages

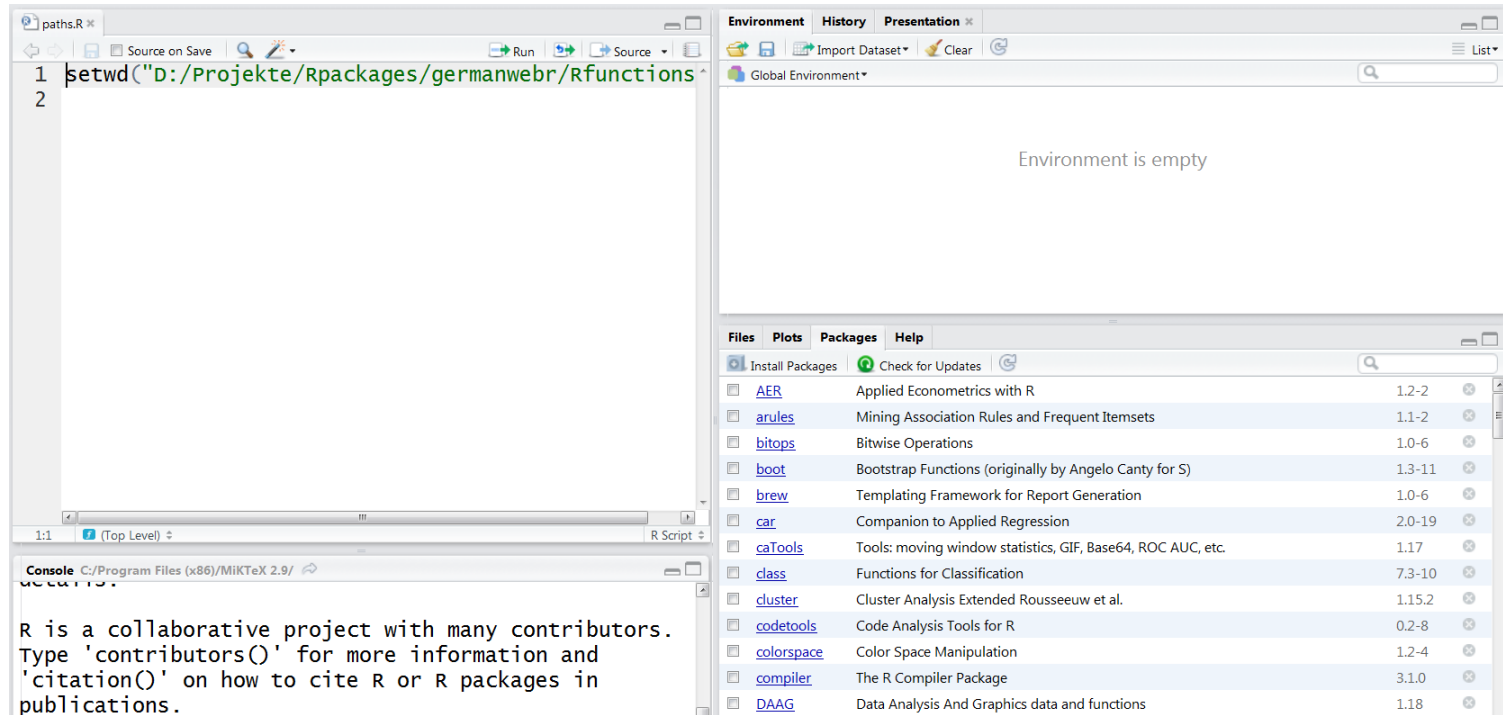
Install and load a package

- The quotes around the package name are necessary for the command `install.packages`.
- They are optional for the command `library`.

```
install.packages("lme4")
```

```
library(lme4)
```

Installation of packages with RStudio



The screenshot shows the RStudio interface with the following components:

- Source Editor:** Contains the R script `setwd("D:/Projekte/Rpackages/germanwehr/Rfunctions")`.
- Environment:** Shows "Global Environment" and "Environment is empty".
- Packages Panel:** Lists installed and available packages.

Package Name	Description	Version	Status
AER	Applied Econometrics with R	1.2-2	Installed
arules	Mining Association Rules and Frequent Itemsets	1.1-2	Installed
bitops	Bitwise Operations	1.0-6	Installed
boot	Bootstrap Functions (originally by Angelo Canty for S)	1.3-11	Installed
brew	Templating Framework for Report Generation	1.0-6	Installed
car	Companion to Applied Regression	2.0-19	Installed
caTools	Tools: moving window statistics, GIF, Base64, ROC AUC, etc.	1.17	Installed
class	Functions for Classification	7.3-10	Installed
cluster	Cluster Analysis Extended Rousseeuw et al.	1.15.2	Installed
codetools	Code Analysis Tools for R	0.2-8	Installed
colorspace	Color Space Manipulation	1.2-4	Installed
compiler	The R Compiler Package	3.1.0	Installed
DAAG	Data Analysis And Graphics data and functions	1.18	Installed
- Console:** Displays the message: "R is a collaborative project with many contributors. Type 'contributors()' for more information and 'citation()' on how to cite R or R packages in publications."

Existing packages and installation

The screenshot shows the RStudio interface with the 'Packages' tab selected. The pane displays a list of installed packages, each with a checkbox, a link to the package page, the package name, a description, the version number, and a close button. The packages listed are AER, arules, bitops, boot, and brew.

Exercise: Download packages

Download and install the following packages from CRAN:

- tidyverse
- nycflights13
- cluster
- ggplot2
- tmap

Have a look at the package documentation. What are these packages for?

Overview of many useful packages:

- Luhmann - **Table with many useful packages**

Other interesting packages:

- Package for Import/Export - **foreign**
- **sampling-package for survey Sampling**
- xtable Package for integrating LaTeX in R (**xtable Galerie**)
- **dummies package for creating dummies**
- **Package mvtnorm for getting a multivariate normal distribution**
- **Package maptools for creating maps**

Install packages from various sources

Install packages from CRAN Server

```
install.packages("lme4")
```

Install packages from Bioconductor Server

```
source("https://bioconductor.org/biocLite.R")  
biocLite(c("GenomicFeatures", "AnnotationDbi"))
```

Install packages from Github

```
install.packages("devtools")  
library(devtools)  
  
install_github("hadley/ggplot2")
```

Packages

```
# load the package to use in the current R session  
library(tidyverse)  
  
# use a particular function within a package  
# without loading the package  
stringr::str_replace()
```

Getting help on packages

```
# provides details regarding contents of a package  
help(package = "tidyr")  
# list vignettes available for a specific package  
vignette(package="tidyr")  
# view specific vignette  
vignette("tidy-data")
```

How do I get an overview

- **Discover packages recently uploaded to CRAN**
- Look at the Shiny web app that shows the **packages recently downloaded from CRAN**
- Have a look at a **quick-list of useful packages**,...
- ..., or at a list with the **best packages for data processing and analysis**,...
- ..., or at **the 50 most used packages**

CRAN Task Views

- For some topics all possibilities are arranged in R. (**Overview of Task Views**)
- Currently there are 35 task views.
- All packages of a task view can be installed with the following **command**:

```
install.packages("ctv")
library("ctv")
install.views("Bayesian")
```

CRAN Task Views

Bayesian	Bayesian Inference
ChemPhys	Chemometrics and Computational Physics
ClinicalTrials	Clinical Trial Design, Monitoring, and Analysis
Cluster	Cluster Analysis & Finite Mixture Models
DifferentialEquations	Differential Equations
Distributions	Probability Distributions
Econometrics	Econometrics
Environmetrics	Analysis of Ecological and Environmental Data
ExperimentalDesign	Design of Experiments (DoE) & Analysis of Experimental Data
ExtremeValue	Extreme Value Analysis
Finance	Empirical Finance

Exercise: additional packages

Go for example to:

<https://cran.r-project.org/>

<https://awesome-r.com/>

or search for

most interesting r packages

and search for packages ...

- for descriptive data analysis.
- with functions to work with date-times and time-spans.
- to use an interface to python.
- to import foreign data (e.g. SPSS data).
- to handle large amounts of data

How to learn after this workshop

How to actually learn any new programming concept



Essential

Changing Stuff and

The `swirl` package

Learn R, in R

```
install.packages("swirl")
```

```
library(swirl)
```

```
# type the following into the console:
```

```
swirl()
```

The tutorial functionality in Rstudio

- You need RStudio v1.3.869-1 Preview for this:
-

```
install.packages("learnr")
```


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** <!--
- **Intro R**
- **Intro R II** -->
- **Comparisson between python and R**
- **R and Stata Side-by-side**
- **AWESOME R**
- **1000 R tutorials/Links**
- **Learn R by watching two-minute videos**

Further Links

- **Overview - how to get help in R**



[\[Home\]](#)

Download

[CRAN](#)

Getting Help with R

Helping Yourself

Before asking others for help, it's generally a good idea for you to try to help yourself. R includes extensive facilities for accessing documentation and searching for help. There are also specialized search engines for accessing information about R on the internet, and general internet search engines can also prove useful ([see below](#)).

- **A list with HowTo`s**
- **A list with the most important R-commands**
- **R-bloggers:** a central hub of content from over 500 bloggers who provide news and tutorials about R.

Shiny App - Intro R

www.intro-stats.com

