

GETTING STARTED WITH R

Jan-Philipp Kolb

26 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](#)

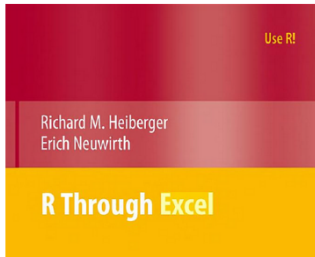
The R Project for Statistical Computing

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**

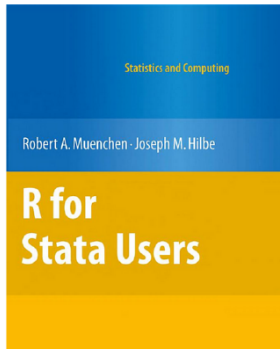
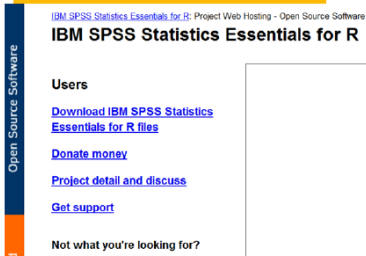
R CAN BE USED IN COMBINATION...



SASmixed

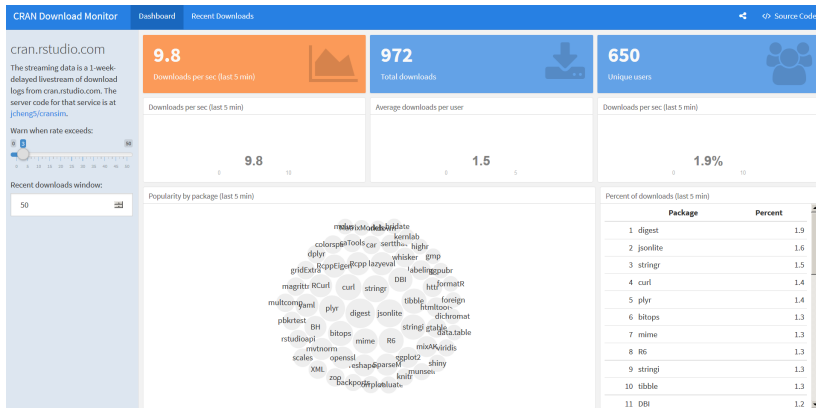


rPython R package



- 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 editing area 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)  
> |
```

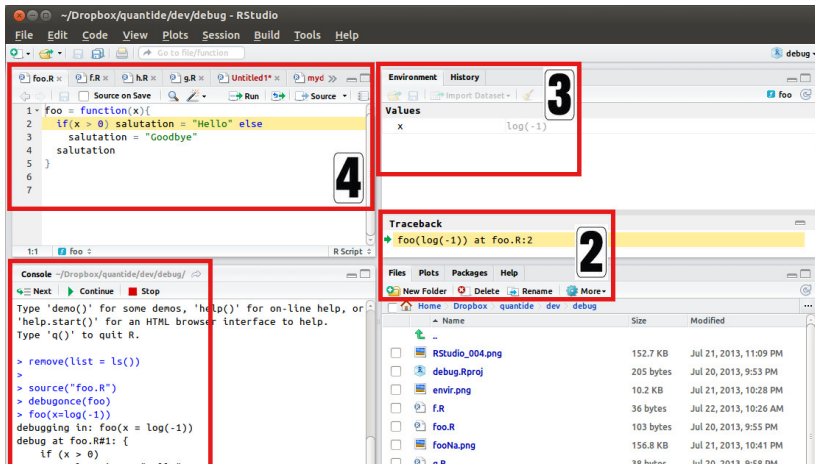
But many people use a graphical user interface (GUI) or a integrated development interface (IDE).

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 interface. The top pane shows a script editor with R code for a function named `block_summary`. The bottom-left pane shows the console output, and the bottom-right pane shows the environment and history.

Script Editor (Source):

```

231 }
232 }
233
234 # Provides the summary for the block containing units m through n, n > m
235 # t = time for firsts unit
236 # m = lower bound unit of production block
237 # n = upper bound unit of production block
238 # r = learning curve rate
239
240 block_summary <- function(t, m, n, r, na.rm = FALSE){
241
242   if(!is.numeric(t) | !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
251   block_summary(t, m, n, r, na.rm)
252 }

```

Console:

```

$'block hours'
[1] 3668.436

$'midpoint unit'
[1] 44.03189

$'midpoint hours'
[1] 40.31249

> 7sum
> m + n
[1] 118
> m + n * 54r
[1] 355.3082
>

```

Environment:

Values
m
n
r
t

Functions:

Function Name	Function Definition
block_summary	function (t, m, n, r, na.rm = FALSE)
cum.approx	function (t, m, n, r, na.rm = FALSE)
cum.exact	function (t, m, n, r, na.rm = FALSE)
lc_rate	function (b, na.rm = FALSE)
midpoint	function (m, n, r, na.rm = FALSE)
natural_slope	function (r, na.rm = FALSE)
unit_curve	function (t, n, r, na.rm = FALSE)

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
<code>quantile()</code>	90 % Quantile	<code>quantile(b,.9)</code>
<code>sample()</code>	Draw a sample	<code>sample(b,1)</code>

```
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.^[1]

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
- 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 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
```

- 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 GET HELP?

- To get help in general:

```
help.start()
```

- Online documentation for most of the functions:

```
help(name)
```

- Use ? to get help.

```
?mean
```

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

```
example(lm)
```

AGAIN, WE GET HELP WITH THE QUESTION MARK

```
?paste
```

Different sections in the help:

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

- 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")
```

<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)

      demo(nlm)
      ---- ~~~

Type <Return> to start :
```

THE FUNCTION APROPOS

- searches everything about the given string

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

```
## [1] ".colMeans"      ".lm.fit"         "colMeans"        "c
## [5] "contr.helmert"   "dummy.coef.lm"   "getAllMethods"   "g
## [9] "glm.control"     "glm.fit"         "KalmanForecast"  "K
## [13] "KalmanRun"       "KalmanSmooth"    "kappa.lm"        "l
## [17] "lm.fit"          "lm.influence"    "lm.wfit"         "m
## [21] "nlm"             "nlminb"          "predict.glm"     "p
## [25] "residuals.glm"   "residuals.lm"    "summary.glm"     "s
```

APROPOS IN COMBINATION WITH REGULAR EXPRESSION

- you can use that in combination with regular expressions

```
? "regular expression"
```

```
help.search("^glm")
```

- ?? is a synonym for help.search

Search engine for the R-Site

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

R Site Search

Query: [\[How to search\]](#)

Display: Description: Sort:

Target:

- ☒ Functions
- ☒ Task views

For problems WITH THIS PAGE (not with R) contact baron@upenn.edu.

Results:

References:

- **views:** [glm: 11]
- **vignettes:** [(can't open the index)]
- **functions:** [glm: 4391]

Total 4402 documents matching your query.

1. [R: Bias reduction in Binomial-response GLMs](#) (score: 299)

Author: *unknown*

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!




DuckDuckGo

R-project + "what I want to know" |



- 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](#) [?](#) [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

JAN-PHILIPP KOLB

GETTING STARTED WITH R

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.

table(x)

See counts of values.

rev(x)

Return x reversed.

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)  
}
```

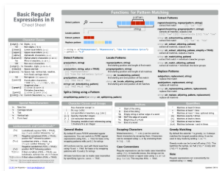
Reading and Writing Data

Also see the **readr** package.

Input	Output	Description
<code>df <- read.table('file.txt')</code>	<code>write.table(df, 'file.txt')</code>	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

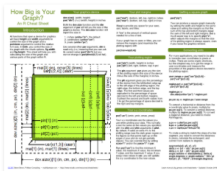
The leaflet package



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.

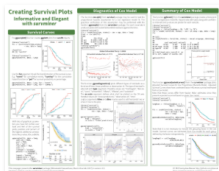
DOWNLOAD

The eurostat package



R tools to access the eurostat database,
by rOpenGov. Updated 03/17.

The survminer package



Elegant survival plots, by Przemyslaw
Biecek. Updated 03/17.

The sjmisc package



dplyr friendly Data and Variable
Transformation, by Daniel Lüdtke.
Updated 08/17.

- 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
```

- **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.

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`.

HOW TO LEARN AFTER THIS WORKSHOP

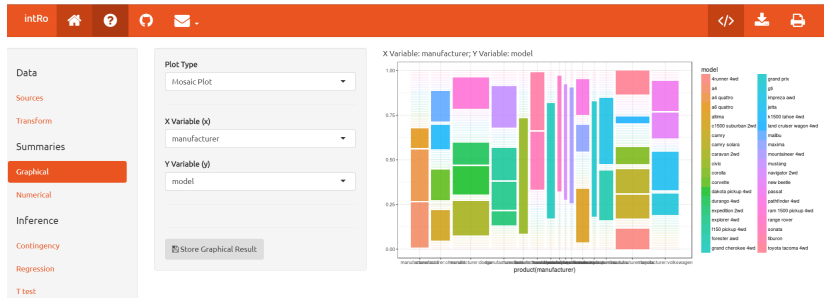
How to actually learn any new programming concept



Essential

Changing Stuff and
Seeing What Happens

<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)**