R for Statistics and Graphics

Session 1

Basics

Mehmet Tevfik DORAK, MD PhD

School of Life Sciences, Pharmacy & Chemistry

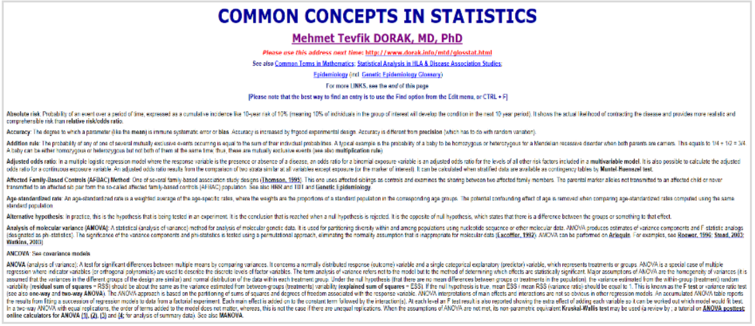
Kingston University London

Istanbul University, Capa Faculty of Medicine

19 April 2019

Introduction to the Crash Course

WEB-BASED SOURCES


COMMON CONCEPTS IN STATISTICS

Mehmet Tevfik DORAK, MD, PhD

Please use this address next time: http://www.dorak.info/mtd/glosstat.html

See also Common Terms in Mathematics; Statistical Analysis in HLA & Disease Association Studies; Epidemiology (incl. Genetic Epidemiology Glossary)

For more LINKS, see the end of this page

[Please note that the best way to find an entry is to use the Find option from the Edit menu, or CTRL + F]

Absolute risk: Probability of an event over a period of time, expressed as a cumulative incidence like 10-year risk of 10% (meaning 10% of individuals in the group of interest will develop the condition in the next 10 year period). It shows the actual likelihood of contracting the disease and provides more realistic and comprehensible risk than **relative risk/odds ratio**.

Accuracy: The degree to which a parameter (like the **mean**) is immune systematic error or **bias**. Accuracy is increased by f=good experimental design. Accuracy is different from **precision** (which has to do with random variation).

Addition rule: The probability of any of one of several mutually exclusive events occurring is equal to the sum of their individual probabilities. A typical example is the probability of a baby to be homozygous or heterozygous for a Mendelian recessive disorder when both parents are carriers. This equals to 1/4 + 1/2 = 3/4. A baby can be either homozygous or heterozygous but not both of them at the same time; thus, these are mutually exclusive events (see also **multiplication rule**).

Adjusted odds ratio: In a multiple logistic regression model where the response variable is the presence or absence of a disease, an odds ratio for a binomial exposure variable is an adjusted odds ratio for the levels of all other risk factors included in a **multivariable model**. It is also possible to calculate the adjusted odds ratio for a continuous exposure variable. An adjusted odds ratio results from the comparison of two strata similar at all variables except exposure (or the marker of interest). It can be calculated when stratified data are available as contingency tables by **Mantel-Haenszel test**.

Affected Family-Based Controls (AFBAC) Method: One of several family-based association study designs (Thomson, 1995). This one uses affected siblings as controls and examines the sharing between two affected family members. The parental marker alleles not transmitted to an affected child or never transmitted to an affected sib pair form the so-called affected family-based controls (AFBAC) population. See also HRR and TDT and **Genetic Epidemiology**.

Age-standardized rate: An age-standardized rate is a weighted average of the age-specific rates, where the weights are the proportions of a standard population in the corresponding age groups. The potential confounding effect of age is removed when comparing age-standardized rates computed using the same standard population.

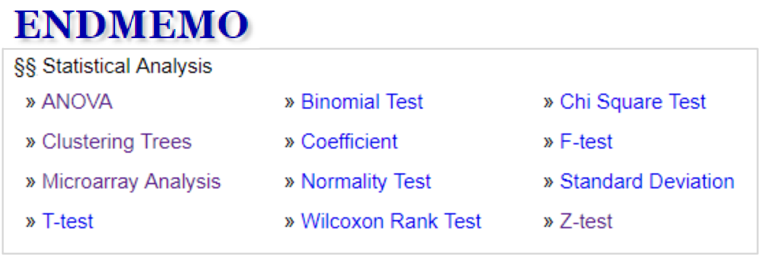
Alternative hypothesis: In practice, this is the hypothesis that is being tested in an experiment. It is the conclusion that is reached when a null hypothesis is rejected. It is the opposite of null hypothesis, which states that there is a difference between the groups or something to that effect.

Analysis of molecular variance (AMOVA): A statistical (analysis of variance) method for analysis of molecular genetic data. It is used for partitioning diversity within and among populations using nucleotide sequence or other molecular data. AMOVA produces estimates of variance components and F-statistic analogs (designated as phi-statistics). The significance of the variance components and phi-statistics is tested using a permutational approach, eliminating the normality assumption that is inappropriate for molecular data (Excoffier, 1992). AMOVA can be performed on Arlequin. For examples, see Roewer, 1996; Stead, 2003; Watkins, 2003)

ANCOVA: See covariance models

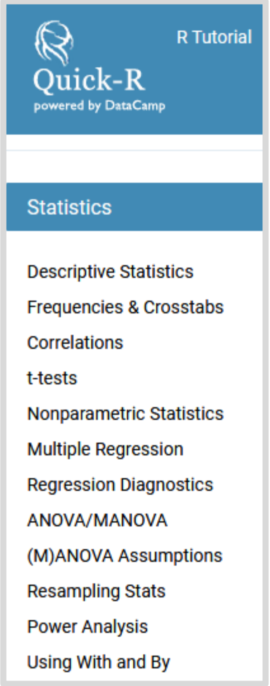
ANOVA (analysis of variance): A test for significant differences between multiple means by comparing variances. It concerns a normally distributed response (outcome) variable and a single categorical explanatory (predictor) variable, which represents treatments or groups. ANOVA is a special case of multiple regression where indicator variables (or orthogonal polynomials) are used to describe the discrete levels of factor variables. The term analysis of variance refers not to the model but to the method of determining which effects are statistically significant. Major assumptions of ANOVA are the homogeneity of variances (it is assumed that the variances in the different groups of the design are similar) and normal distribution of the data within each treatment group. Under the null hypothesis (that there are no mean differences between groups or treatments in the population), the variance estimated from the within-group (treatment) random variability (**residual sum of squares** = RSS) should be about the same as the variance estimated from between-groups (treatments) variability (**explained sum of squares** = ESS). If the null hypothesis is true, mean ESS / mean RSS (variance ratio) should be equal to 1. This is known as the **F test** or variance ratio test (see also **one-way** and **two-way ANOVA**). The ANOVA approach is based on the partitioning of sums of squares and degrees of freedom associated with the response variable. ANOVA interpretations of main effects and interactions are not so obvious in other regression models. An accumulated ANOVA table reports the results from fitting a succession of regression models to data from a factorial experiment. Each main effect is added on to the constant term followed by the interaction(s). At each level an F test result is also reported showing the extra effect of adding each variable so it can be worked out which model would fit best. In a two-way ANOVA with equal replications, the order of terms added to the model does not matter, whereas, this is not the case if there are unequal replications. When the assumptions of ANOVA are not met, its non-parametric equivalent **Kruskal-Wallis test** may be used (a review by ; a tutorial on **ANOVA posttest**; **online calculators for ANOVA** (1), (2), (3) and (4; for analysis of summary data). See also **MANOVA**.

http://www.dorak.info/mtd/glosstat.html


ENDMEMO

§§ Statistical Analysis

» ANOVA	» Binomial Test	» Chi Square Test
» Clustering Trees	» Coefficient	» F-test
» Microarray Analysis	» Normality Test	» Standard Deviation
» T-test	» Wilcoxon Rank Test	» Z-test


Quick-R powered by DataCamp

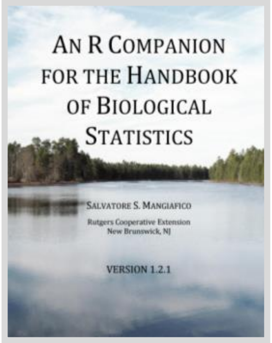
R Tutorial

Statistics

- Descriptive Statistics
- Frequencies & Crosstabs
- Correlations
- t-tests
- Nonparametric Statistics
- Multiple Regression
- Regression Diagnostics
- ANOVA/MANOVA
- (M)ANOVA Assumptions
- Resampling Stats
- Power Analysis
- Using With and By

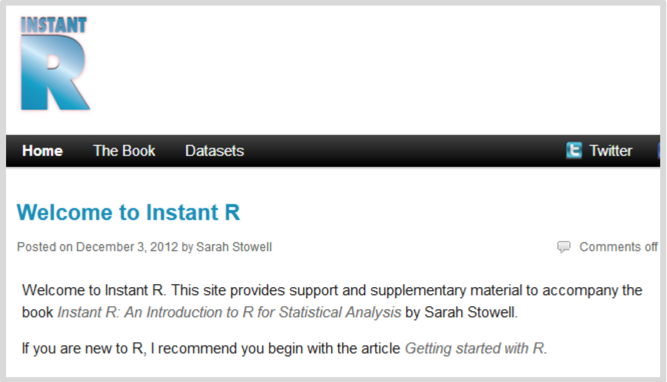
R Tutorial

An R Introduction to Statistics


AN R COMPANION FOR THE HANDBOOK OF BIOLOGICAL STATISTICS

SALVATORE S. MANGIAFICO
Rutgers Cooperative Extension
New Brunswick, NJ

VERSION 1.2.1


INSTANT R

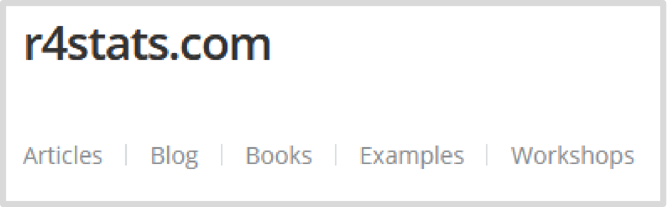
Home | The Book | Datasets | Twitter

Welcome to Instant R

Posted on December 3, 2012 by Sarah Stowell — Comments off

Welcome to Instant R. This site provides support and supplementary material to accompany the book *Instant R: An Introduction to R for Statistical Analysis* by Sarah Stowell.

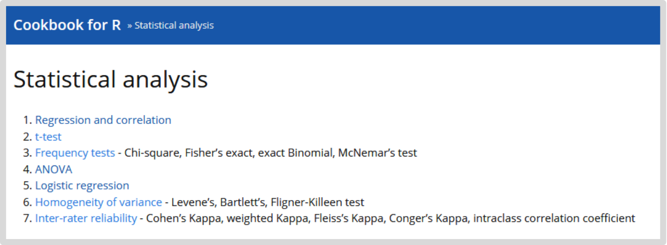
If you are new to R, I recommend you begin with the article *Getting started with R.*


r4stats.com

Articles | Blog | Books | Examples | Workshops

R Tutorials @ ListenData:
https://www.listendata.com/p/r-programming-tutorials.html

How to in R


Cookbook for R » Statistical analysis

Statistical analysis

1. Regression and correlation
2. t-test
3. Frequency tests - Chi-square, Fisher's exact, exact Binomial, McNemar's test
4. ANOVA
5. Logistic regression
6. Homogeneity of variance - Levene's, Bartlett's, Fligner-Killeen test
7. Inter-rater reliability - Cohen's Kappa, weighted Kappa, Fleiss's Kappa, Conger's Kappa, intraclass correlation coefficient

r-statistics.co by Selva Prabhakaran

BU Boston University School of Public Health

Kingston University London

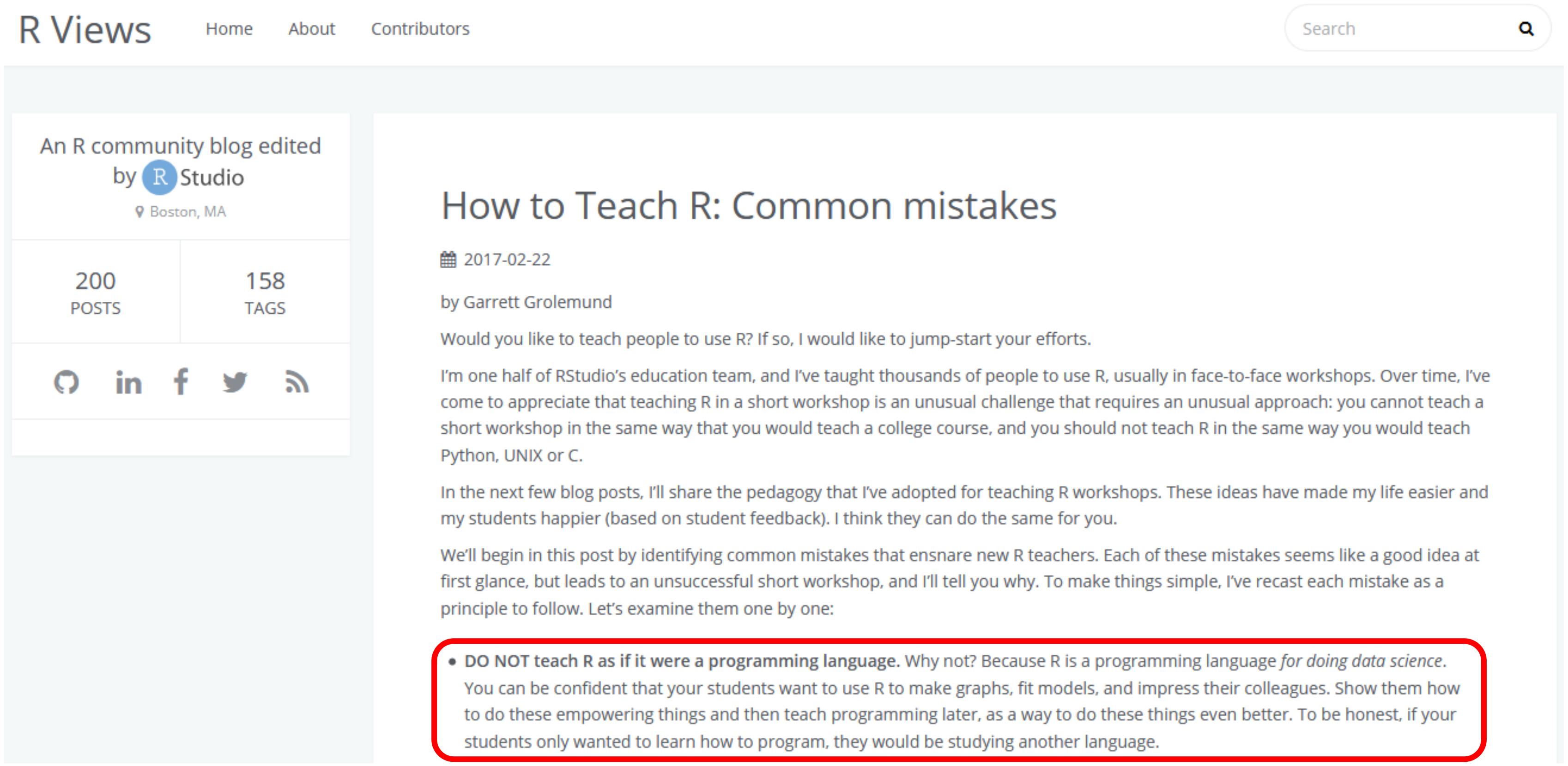
Basic Statistical Analysis Using the R Statistical Package

Outline of the Crash Course

- R installation and a quick demo
- R syntax and a few simple rules
- Basic and beyond basic statistics
- Statistical power, survival analysis, meta-analysis, permutation test, diagnostic test assessment incl. ROC analysis, cluster analysis
- R Studio and R Commander

Kingston University London

Teaching Philosophy of the Crash Course



R Views Home About Contributors Search

An R community blog edited by R Studio

Boston, MA

200 POSTS

158 TAGS

How to Teach R: Common mistakes

2017-02-22

by Garrett Grolemund

Would you like to teach people to use R? If so, I would like to jump-start your efforts.

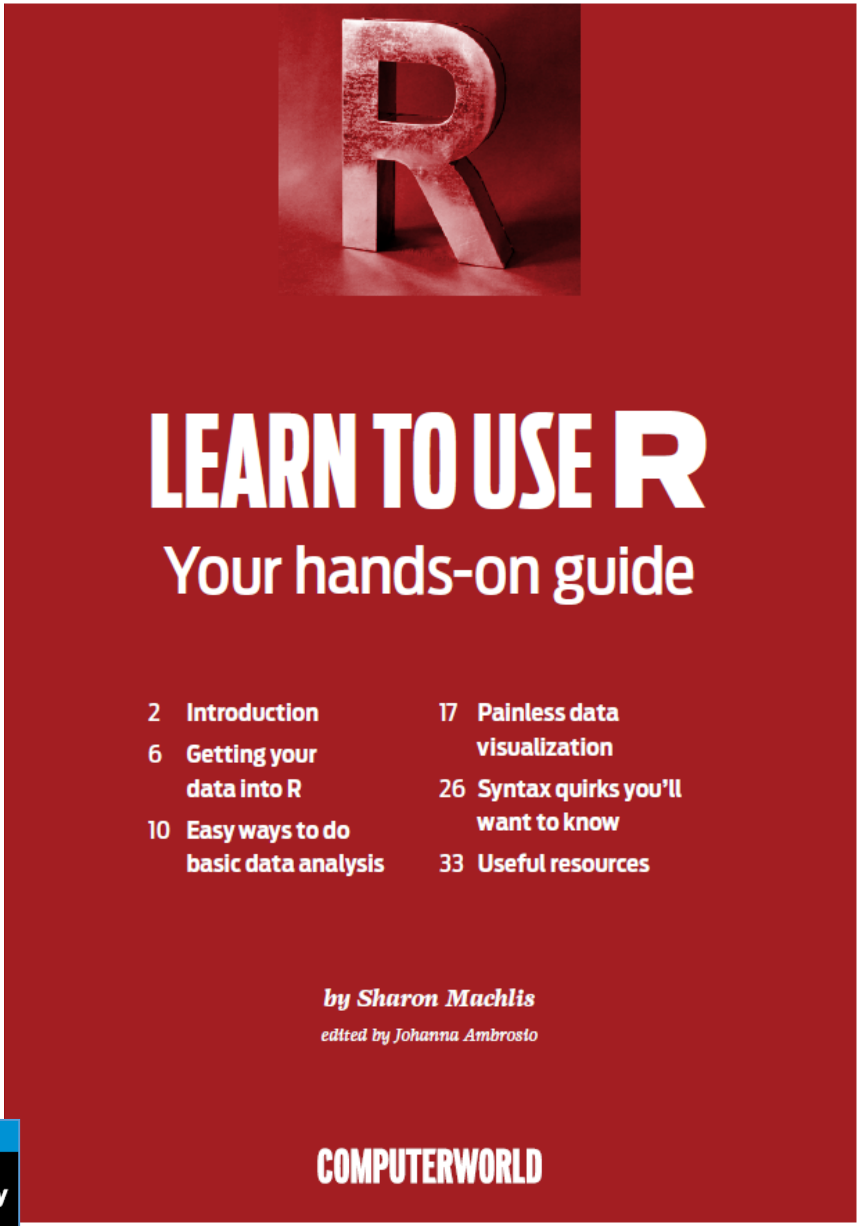
I'm one half of RStudio's education team, and I've taught thousands of people to use R, usually in face-to-face workshops. Over time, I've come to appreciate that teaching R in a short workshop is an unusual challenge that requires an unusual approach: you cannot teach a short workshop in the same way that you would teach a college course, and you should not teach R in the same way you would teach Python, UNIX or C.

In the next few blog posts, I'll share the pedagogy that I've adopted for teaching R workshops. These ideas have made my life easier and my students happier (based on student feedback). I think they can do the same for you.

We'll begin in this post by identifying common mistakes that ensnare new R teachers. Each of these mistakes seems like a good idea at first glance, but leads to an unsuccessful short workshop, and I'll tell you why. To make things simple, I've recast each mistake as a principle to follow. Let's examine them one by one:

- **DO NOT teach R as if it were a programming language.** Why not? Because R is a programming language *for doing data science*. You can be confident that your students want to use R to make graphs, fit models, and impress their colleagues. Show them how to do these empowering things and then teach programming later, as a way to do these things even better. To be honest, if your students only wanted to learn how to program, they would be studying another language.

The Best Single Printed Source for Beginners



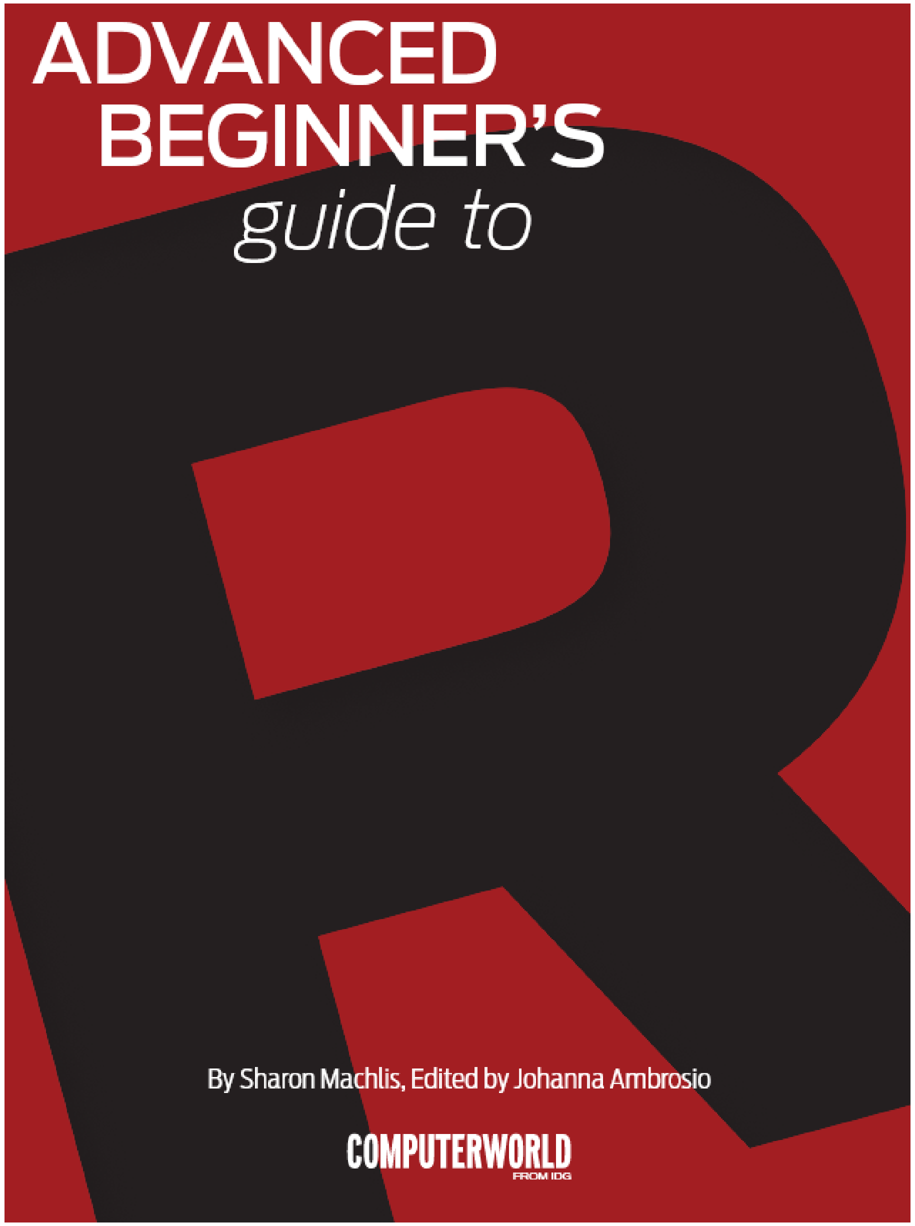
LEARN TO USE R

Your hands-on guide

by Sharon Machlis

edited by Johanna Ambrosio

COMPUTERWORLD



ADVANCED BEGINNER'S *guide to* R

By Sharon Machlis, Edited by Johanna Ambrosio

COMPUTERWORLD FROM IDG

Kingston University London

R Installation I

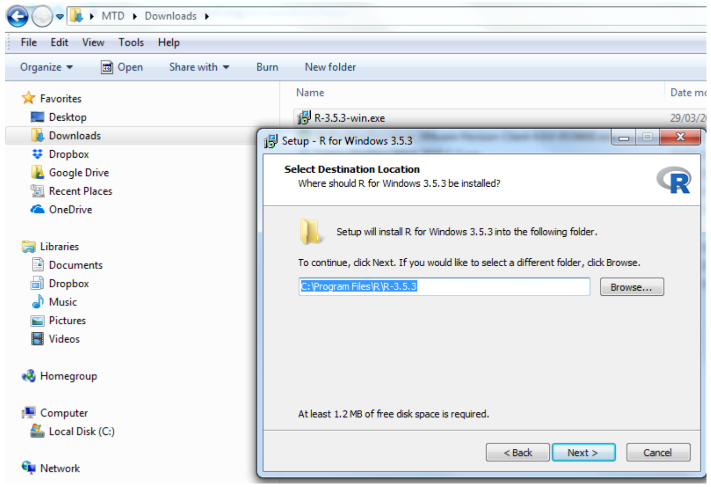
- Go to https://cran.r-project.org/bin/windows/base
- Click on Download R 4.0.3 for Windows * *(as of December 2020)*
- Download R-4.0.3-win.exe
- Locate the exe file and double click on it
- Click RUN (accept defaults or see the next slide)

* If you have R version <4.0.0 already installed, please update your R to version >4.0.0)

Kingston University London

R Installation II

- When reached "Select Destination Location", change it to "C:\R" (optional)



Select Destination Location

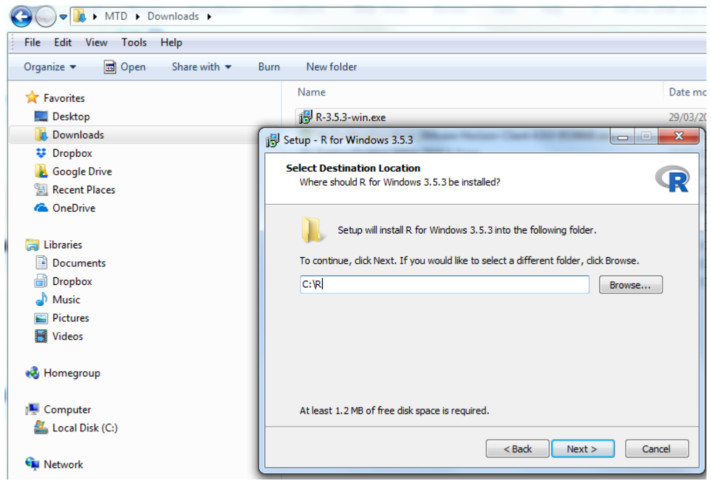
Where should R for Windows 3.5.3 be installed?

Setup will install R for Windows 3.5.3 into the following folder.

To continue, click Next. If you would like to select a different folder, click Browse.

C:\Program Files\R\R-3.5.3

At least 1.2 MB of free disk space is required.



Select Destination Location

Where should R for Windows 3.5.3 be installed?

Setup will install R for Windows 3.5.3 into the following folder.

To continue, click Next. If you would like to select a different folder, click Browse.

C:\R

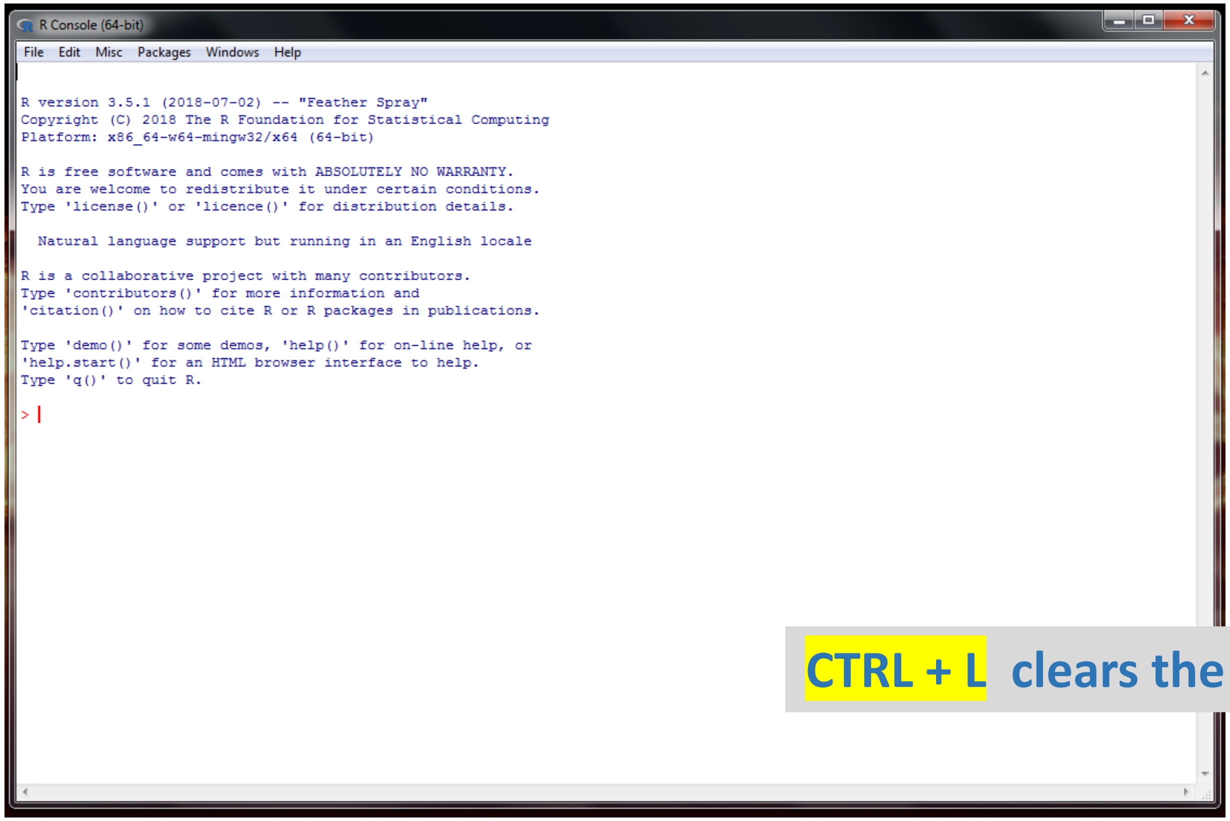
At least 1.2 MB of free disk space is required.

R Installation III

- Continue with installation accepting default options
- In a few minutes, R is installed and there should be R icons in the start up menu and on the desktop
- Start R!

What Does R Look Like?

- R looks nothing like you expect!
- After some messages, just a blank page



```
R version 3.5.1 (2018-07-02) -- "Feather Spray"
Copyright (C) 2018 The R Foundation for Statistical Computing
Platform: x86_64-w64-mingw32/x64 (64-bit)

R is free software and comes with ABSOLUTELY NO WARRANTY.
You are welcome to redistribute it under certain conditions.
Type 'license()' or 'licence()' for distribution details.

  Natural language support but running in an English locale

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.

Type 'demo()' for some demos, 'help()' for on-line help, or
'help.start()' for an HTML browser interface to help.
Type 'q()' to quit R.

> 
```

CTRL + L clears the screen

R Installation IV

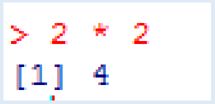
- If the necessary R packages are not yet installed for you on the computer you are using, or if you are using your own computer, please use the script "`installation1.R`" to install them (see slide 30)

  OR:

- Select "`Set CRAN mirror..`" in the Packages menu on top of the R console. A pop-up window will open
- Select the nearest mirror site for downloading the packages
- Open the script file with Notepad or any text editor. You can also open it with R (`File > Open script..`)
- Then type each line of the text in the script file and press ENTER. Keep doing so until the end of the script file
- Wait until R stops working, and you have installed all necessary packages. Disregard any messages from R

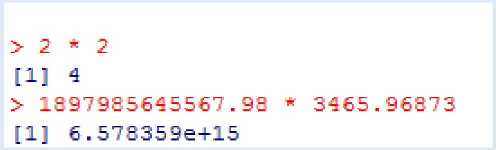
What To Do With R?

- You need to know how to talk to R
- Download and open "`s1.R`" as described in slide 30
- Try typing `2*2` and press ENTER



```
> 2 * 2
[1] 4
```

- Now try typing something like `1897985645567.98 * 3465.96873` and press ENTER



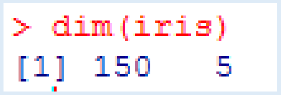
```
> 2 * 2
[1] 4
> 1897985645567.98 * 3465.96873
[1] 6.578359e+15
```

- You can use R as at least a calculator even for very complex calculations
- Let's do something more serious using a dataset that comes with R (called `iris`)
- Loading a built-in dataset (there are many of them) is easy; to see the complete list, type: `data()`
- Type: `data(iris)` and press ENTER
- Nothing happens!

Script: `s1.R`

What To Do With R?

- Dataset `iris`, however, is loaded in the memory and you can use it for analysis or for manipulation
- Just type: `iris` and press ENTER to see the whole dataset on the screen (which is not a good idea!)
- In case the dataset is too large, you want to be careful. First, check its dimensions (row and column numbers) by typing: `dim(iris)`
- It has 150 rows and 5 columns



```
> dim(iris)
[1] 150   5
```

- Now type: `head(iris)`
- Now type: `tail(iris)`
- The top (head) or bottom (tail) 5 rows of the dataset will be printed on the screen
- You can learn more about the structure of the dataset by typing: `str(iris)` {also try: `Str(iris)`}

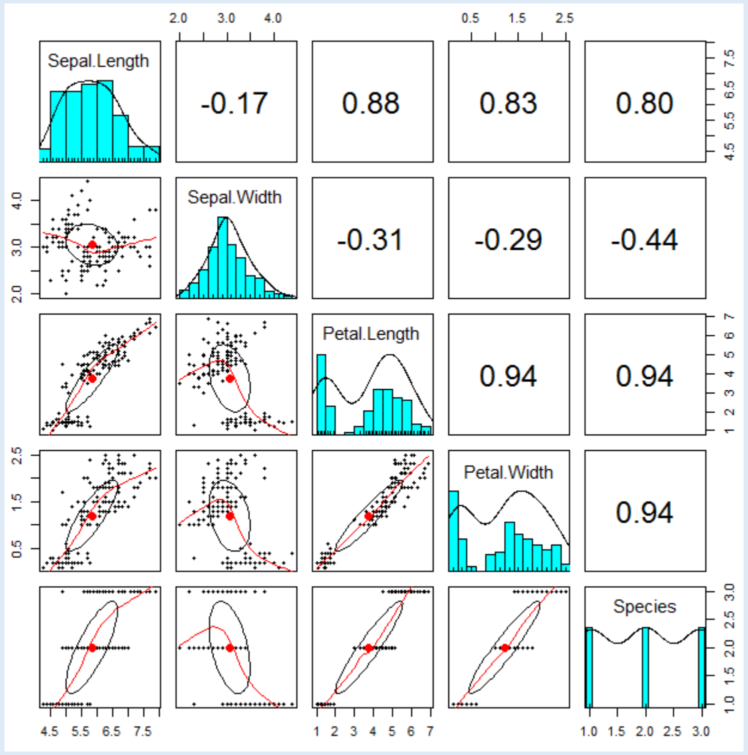
What To Do With R?

- Time to do some real analysis of this dataset. Type (or copy/paste) each of the following lines one by one followed by ENTER
- `summary(iris)`
- `boxplot(iris)`
- `boxplot(iris[-5])`
- `boxplot(iris)`
- `boxplot(iris[-5], col = "red") #adding color`
- `abline(h = 3.5, col = "red") #adding a horizontal line`
- `boxplot(iris$Sepal.Length ~ iris$Species)`
- `hist(iris$Sepal.Length)`
- `pairs(iris[-5])`
- `my_cols <- c("red", "blue", "green") #carry on to the next line`
- `pairs(iris[1:4], pch = 19, cex = 0.5, col = my_cols[iris$Species], lower.panel=NULL)`
- Have you said Wow! yet?

What To Do With R?

- If you haven't said Wow! yet, try this:

```
library("psych"); pairs.panels(iris, method = "spearman")
```



- Wow!

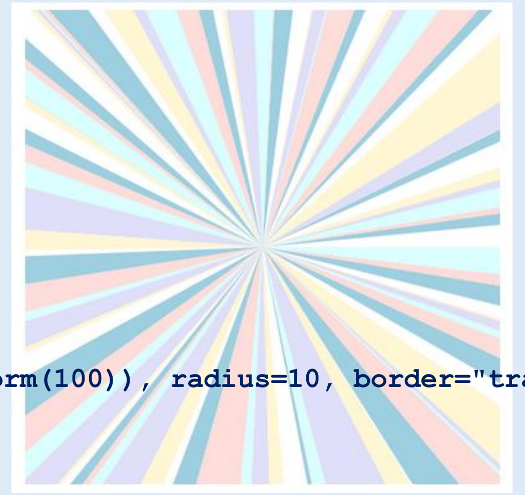
What To Do With R?

Now, run the script "demo1.R"

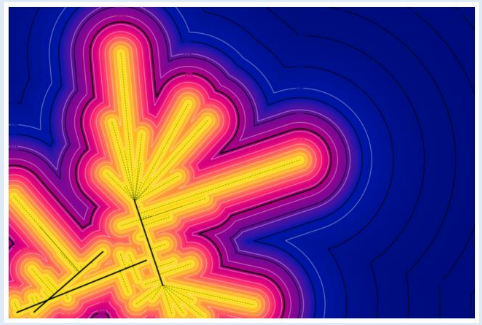
(You can do aRt with R!)

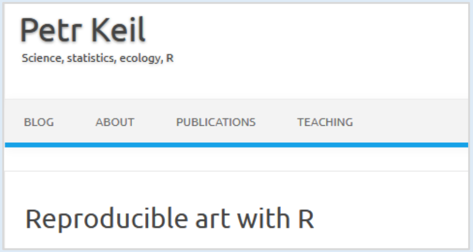
THE R GRAPH GALLERY

Inspiration and Help with R Graphics



```
pie(abs(rnorm(100)), radius=10, border="transparent")
```




Petr Keil

Science, statistics, ecology, R

Reproducible art with R

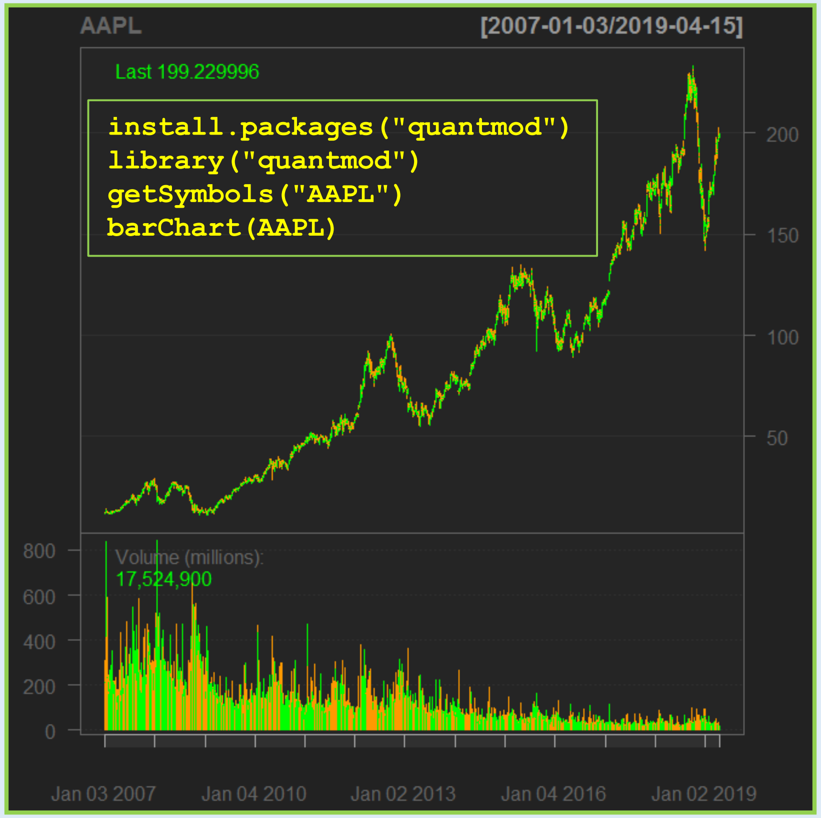
wow!

Script: demo1.R

What To Do With R?

Now, run the script "quantmod.R"

(You can even do real-time financial explorations with R!)



```
install.packages("quantmod")
library("quantmod")
getSymbols("AAPL")
barChart(AAPL)
```

WOW!!!

Script: quantmod.R

R Syntax

Here are some simple rules we have already used:

- **R language is case-sensitive (`Str(iris)` did not work!)**
- **Every R function is followed by brackets even if they contain nothing like:** `data()` `quit()` `ls()` `windows()` `file.choose()` `date()` `colors()`
- **Every R function has a number of arguments/options to be placed within brackets (most of them have default values):**

```
boxplot(iris[-5], col = "red")
```

  Here, color is an option and defined as red. You need to check the default options.

- **In R language, non-numerical objects are written between quotes (as above) and single or double quotes are acceptable**
- **R is quite tolerant to spacing: both `quit()` and `quit ()` will work**
- **Anything beginning with " # " is ignored by R (comment line)**
- **In R, a dataset is called a dataframe (and a matrix is different)**
- **In R, the convention for writing a PATH is always a single forward slash**

R Syntax

- To denote a column in a dataset, use the `$` sign:

  `hist(iris$Sepal.Length)`

  Here, we made a histogram of the data in the `Sepal.Length` column of `iris` dataset

  Alternatively, use the `attach(dataframe)` function, and you do not need to define the dataframe (and no $ sign needed)

- To install a package/library, use: `install.packages()` function; type: `install.packages("psych")`

- To load a library, use `library()` function; type: `library("psych")`

- R uses an aRrow `<-` as an assignment operator:

```
x <- 5   # x is assigned a value of 5
x * 6    # x is multiplied by 6
30       # the result is 30
```

- Any object (a dataset, variable etc) can be assigned to a variable

- You can recall previous commands using the up arrow key

- Entering a letter and then hitting the Tab key twice will list the commands and objects starting with that letter

Kingston University London

R Syntax

- You can split a command line to as many pieces as you like with no harm:

```
boxplot(iris[1:4],
   boxwex = 0.2,      # you can even insert a comment
   pch =        14)
```

```
# The above script is equivalent to:
boxplot(iris[1:4], boxwex = 0.2, pch = 14)
```

- You can also join multiple commands in different lines to a single line command separated by semicolons:

```
data(iris); boxplot(iris[1:4]); abline(h = 2.5)
```

- You can have two graphics windows by using `windows()`

- To quit R, type:

```
quit()
```

and, say "No" the question asked (unless you want to save the session image)

Kingston University London

R Syntax

- Double column (`::`) separates a function name from its package name (like `fBasics::basicStats`)
- When generating a set of numbers, setting a seed number generator with `set.seed()` is not necessary, but it will make your results reproducible (see R Function of the Day)

For more on R syntax, see:

Common Uncommon Notations that Confuse New R Coders
Good Practices in R Programming
Syntax Quirks You'll Want to Know
R Reference Card & v.2

Kingston University London

How to Enter Data into R

Vector, matrix, array

- For a small dataset (vector), create an R object using `c()`:

```
    x <- c(2, 4, 3, 5, 1, 4, 7, 2, 5, 3, 8, 5, 6, 9, 2)
    boxplot(x)
```

- For a data frame, create multiple vectors for rows or columns, and bind them using `rbind() / cbind()` or `data.frame()` functions.

- For a matrix, use the `matrix()` function:

```
x <- matrix(c(2,12,6,10), nrow=2, byrow=TRUE)
```
or
```
x <- matrix(c(2,12,6,10), nrow=2, byrow=FALSE)
```

- For an array, use the `array()` function:

```
x <- array(c(matrix1, matrix2), dim=c(2,2,2))
```
   # multiple matrices (of size 2x2) are merged to create an array: two 2x2 tables (hence, `dim = c(2,2,2)`)

Creating matrices and dataframes

15 Easy Solutions To Your Data Frame Problems In R

Kingston University London

How to Enter Data into R

2x2 table

- To create a 2x2 (contingency) table, create a matrix:

```
x  <-  matrix(c(22, 46, 66, 58), nrow = 2)
```

- R can also provide a blank spreadsheet to enter the numbers:

```
x <- data.frame()  # assigns a name to the contingency table to be created
fix(x)  # opens the data editor to enter the cell values (rxc)
```

- To see the newly created contingency table:

```
x    # prints the newly created contingency table

       var1 var2
    1   22   66
    2   46   58
```

- To use the contingency table, for example, for Fisher's test, use the assigned name of the 2x2 table (x):

```
fisher.test(x)
```

Kingston University London

How to Enter Data into R

Excel (CSV) file

- Save your Excel file as CSV in the working directory (if in doubt, check with: `getwd()`)

- Use the `read.csv()` function to read your file into R (ideally, assign it to a name):

```
file <- read.csv("filename.csv", header = TRUE)
```

- `dim(file), str(file), names(file), head(file), tail(file)` can be used to explore the dataset

- You can also export (save) an R dataframe (like `iris`) as a CSV file (by default to the working directory):

```
write.csv(iris, "iris.csv")
      OR
write.csv2(iris, "iris.csv")   # write.csv2()
```
uses a comma for the decimal point and a semicolon for the separator

Kingston University London

How to Enter Data into R

Excel (CSV) file

Function	What It Does	Example
read.table()	Reads any tabular data where the columns are separated (for example by commas or tabs). You can specify the separator (for example, commas or tabs), as well as other arguments to precisely describe your data.	read.table(file="myfile", sep="t", header=TRUE)
read.csv()	A simplified version of read.table() with all the arguments preset to read CSV files, like Microsoft Excel spreadsheets.	read.csv(file="myfile")
read.csv2()	A version of read.csv() configured for data with a comma as the decimal point and a semicolon as the field separator.	read.csv2(file="myfile", header=TRUE)
read.delim()	Useful for reading delimited files, with tabs as the default separator.	read.delim(file="myfile", header=TRUE)

Kingston University London

R FOR DUMMIES CHEAT SHEET

How to Enter Data into R

Excel (CSV) file

If you are not sure what your working directory is, and you want to read a CSV file into R, use the following command:

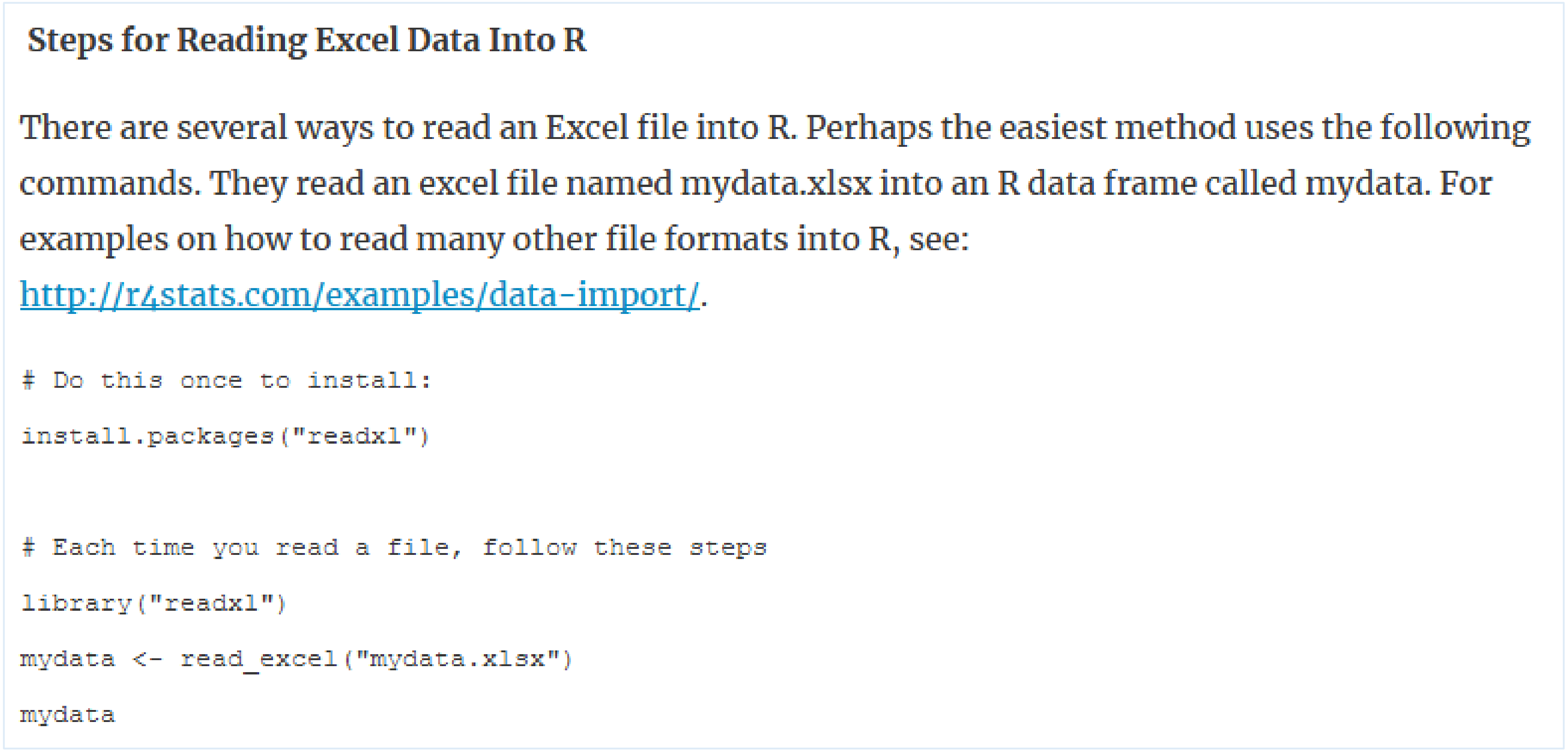
```
x <- read.csv(file.choose(), header=TRUE)
     # if no header, use FALSE
```

This will allow you to browse your computer and locate the file like you do on Windows Explorer.

Kingston University London

How to Enter Data into R

Excel (XLSX) file

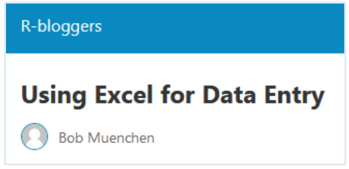


Steps for Reading Excel Data Into R

There are several ways to read an Excel file into R. Perhaps the easiest method uses the following commands. They read an excel file named mydata.xlsx into an R data frame called mydata. For examples on how to read many other file formats into R, see: http://r4stats.com/examples/data-import/.

```
# Do this once to install:
install.packages("readxl")


# Each time you read a file, follow these steps
library("readxl")
mydata <- read_excel("mydata.xlsx")
mydata
```



R-bloggers

Using Excel for Data Entry

Bob Muenchen

How to Enter Data into R

Excel file

An easier way for smaller spreadsheet tables

Copy the spreadsheet on clipboard, and import it to R using the following function:

```
x <- read.delim("clipboard", header=TRUE)
```

# if no header, use FALSE

How to Enter Data into R

SPSS, Stata, SAS and other files

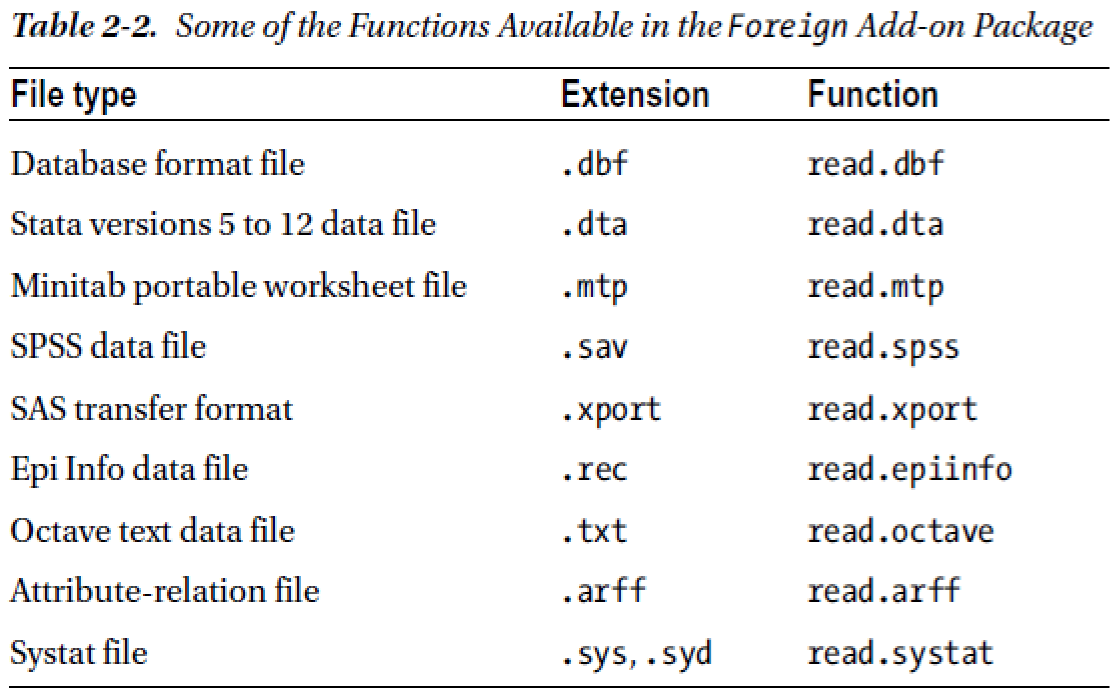
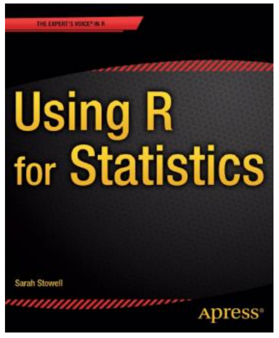


Table 2-2. Some of the Functions Available in the `Foreign` Add-on Package

File type	Extension	Function
Database format file	`.dbf`	`read.dbf`
Stata versions 5 to 12 data file	`.dta`	`read.dta`
Minitab portable worksheet file	`.mtp`	`read.mtp`
SPSS data file	`.sav`	`read.spss`
SAS transfer format	`.xport`	`read.xport`
Epi Info data file	`.rec`	`read.epiinfo`
Octave text data file	`.txt`	`read.octave`
Attribute-relation file	`.arff`	`read.arff`
Systat file	`.sys, .syd`	`read.systat`

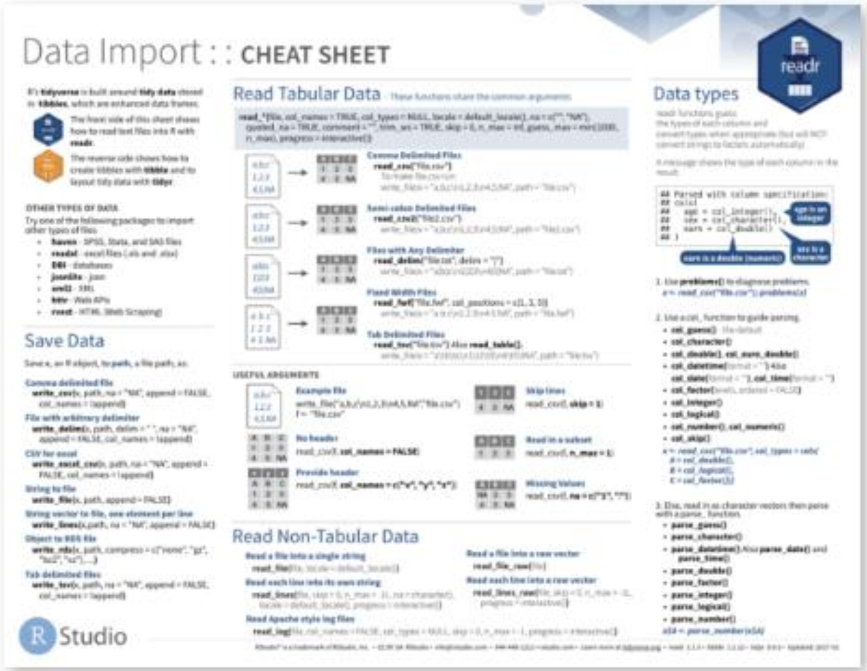


How to Enter Data into R

Data Import Cheat Sheet

The Data Import cheat sheet reminds you how to read in flat files with http://readr.tidyverse.org/, work with the results as tibbles, and reshape messy data with tidyr. Use tidyr to reshape your tables into tidy data, the data format that works the most seamlessly with R and the tidyverse. Updated 01/17.

DOWNLOAD


Data Import : : CHEAT SHEET

Read Tabular Data

Data types

Save Data

Read Non-Tabular Data

RStudio

DataCamp

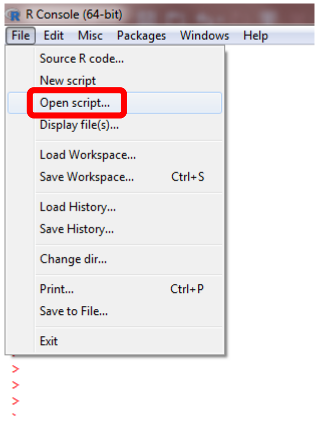
This R Data Import Tutorial Is Everything You Need

Honing Data Science
A portal for aspiring data scientist

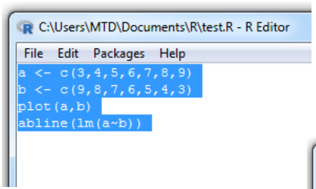
10 techniques to load data into R

Running R Scripts

Save your script as a text file with a file extension "R" like `script.R`
Preferentially, save the file in the working directory (to find out, use: `getwd()` in R)
Open your script file using the File menu in R (Open script)



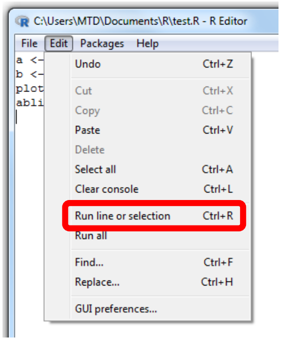
Once the file is open, select the script



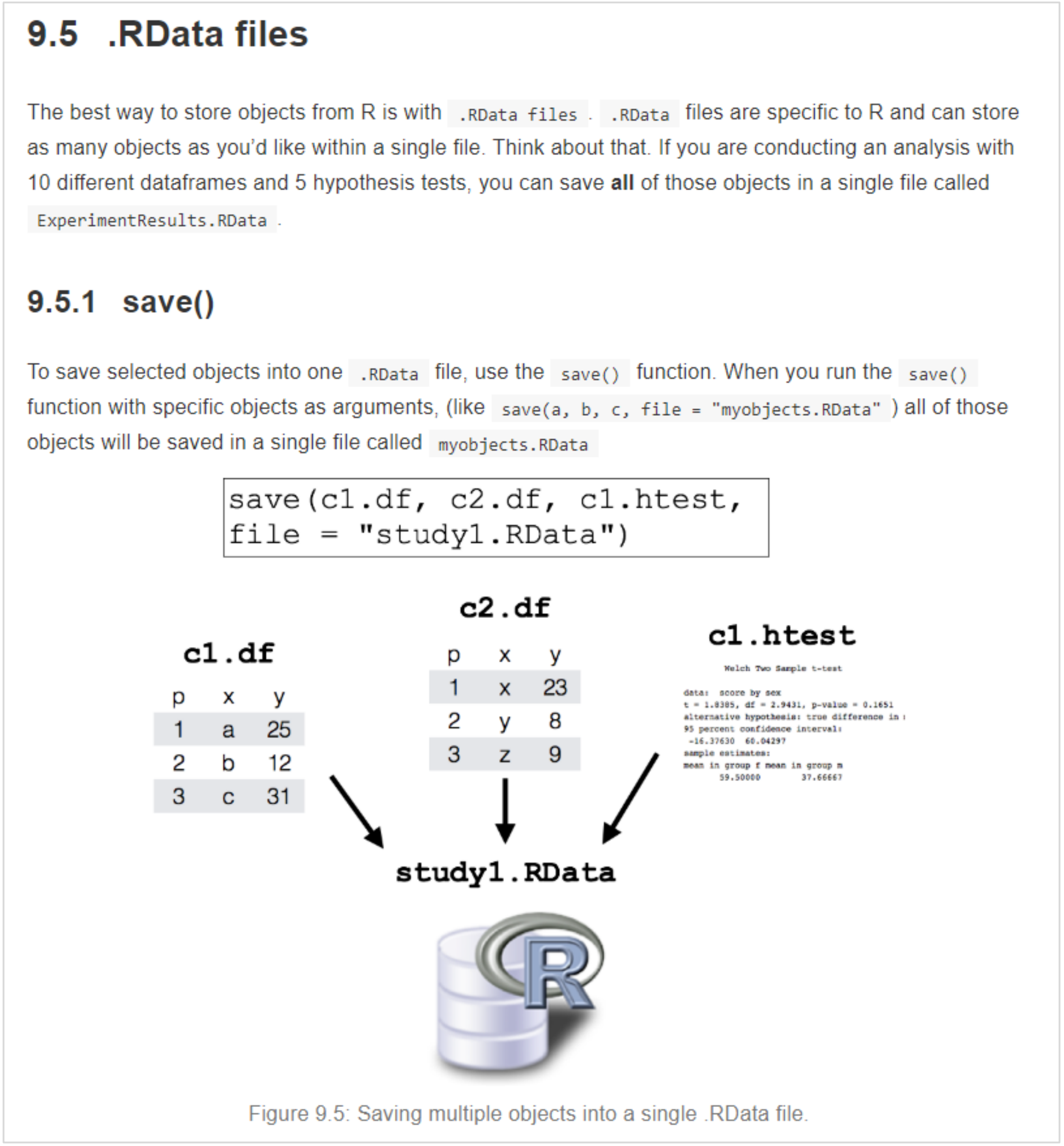
```
a <- c(3,4,5,6,7,8,9)
b <- c(9,8,7,6,5,4,3)
plot(a,b)
abline(lm(a~b))
```

Press CTRL + R on the keyboard (or click on Run line or selection under Edit)

The script will run in its entirety



How to Save Data as an R Object

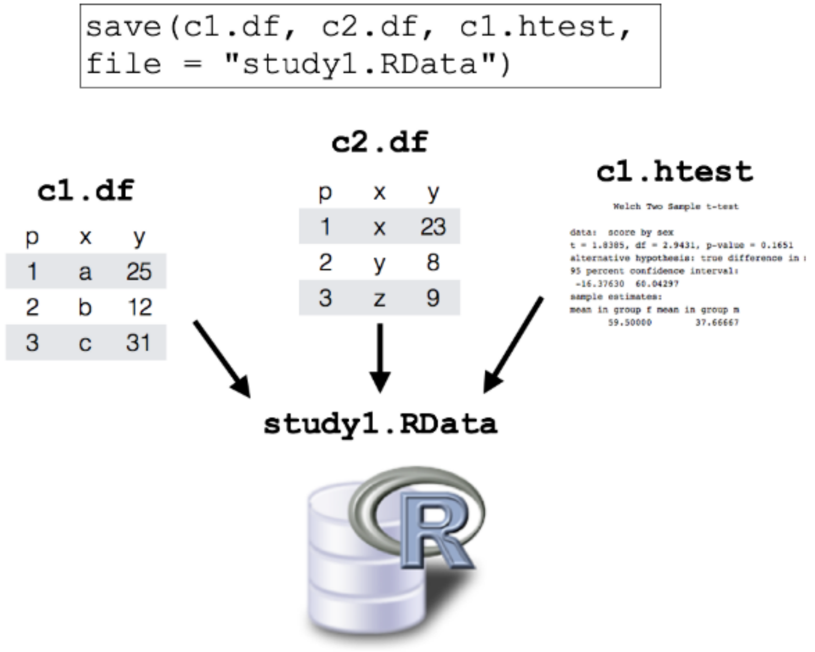


9.5 .RData files

The best way to store objects from R is with `.RData files`. `.RData` files are specific to R and can store as many objects as you'd like within a single file. Think about that. If you are conducting an analysis with 10 different dataframes and 5 hypothesis tests, you can save **all** of those objects in a single file called `ExperimentResults.RData`.

9.5.1 save()

To save selected objects into one `.RData` file, use the `save()` function. When you run the `save()` function with specific objects as arguments, (like `save(a, b, c, file = "myobjects.RData"`) all of those objects will be saved in a single file called `myobjects.RData`



```
save(c1.df, c2.df, c1.htest,
file = "study1.RData")
```

c1.df

p	x	y
1	a	25
2	b	12
3	c	31

c2.df

p	x	y
1	x	23
2	y	8
3	z	9

c1.htest

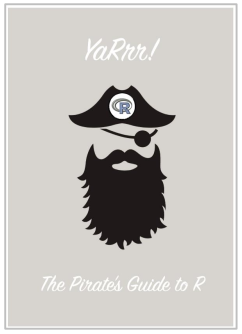
```
        Welch Two Sample t-test

data:  score by sex
t = 1.8385, df = 2.9431, p-value = 0.1651
alternative hypothesis: true difference in :
95 percent confidence interval:
 -16.37630  60.04297
sample estimates:
mean in group f mean in group m
       59.50000        37.66667
```

Figure 9.5: Saving multiple objects into a single .RData file.

`save()` function saves your R object (for example, a data frame) in your working directory, and it can be loaded by recalling it at the next session.

Use `load()` to recall and load the R object.

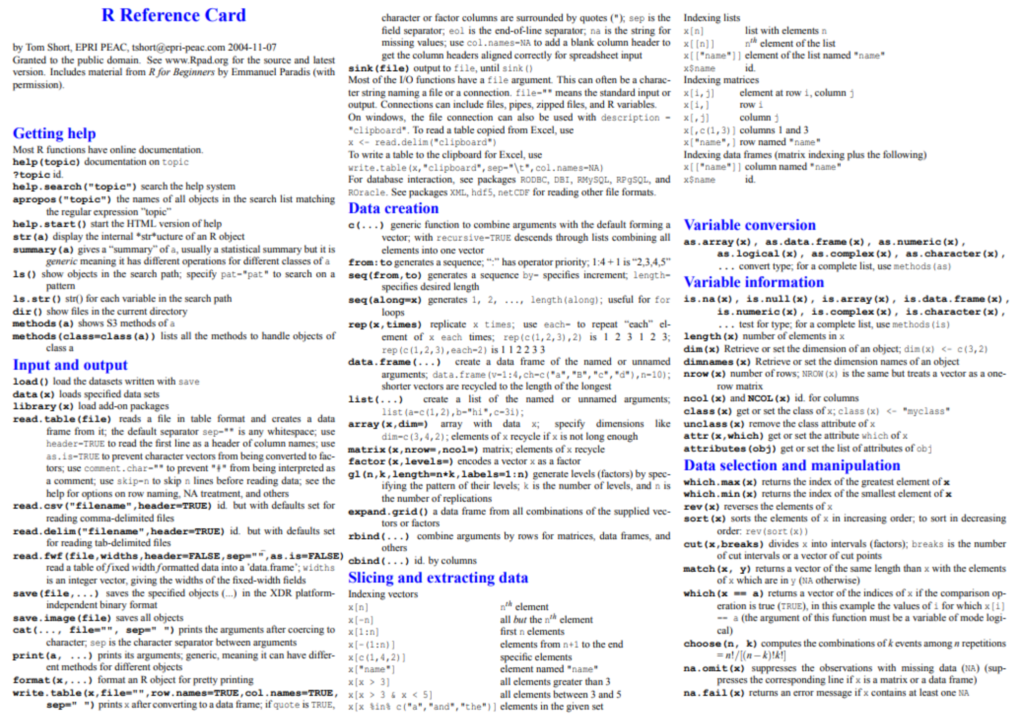


Most Common Sources of Error Messages

- Typos (comma instead of dot; pound sign or ampersand instead of dollar sign; leaving out the dot in multi-word function names like `read.delimit()` or `as.factor()`; case sensitivity (including WORD changing the initial to a capital letter)
- Parentheses (type, unequal opening and closing parentheses)
- Quotes; including WORD changing your straight quotes (" … ") with smart quotes (“ … ”); inconsistent use of single and double quotes
- Missing commas (e.g., between function arguments)
- Your grouping variable is NOT a factor (but numerical or string/character variable; check with `class()`)
- Missing data is causing trouble
- The function you use exists in two different libraries in use (loaded)
- The library you intend to use is not loaded
- In path definition, R uses forward slash like C:/R (not back slash as in Windows Explorer like C:\R)

Kingston University London

R Reference Cards



R Reference Card

by Tom Short, EPRI PEAC, tshort@epri-peac.com 2004-11-07
Granted to the public domain. See www.Rpad.org for the source and latest version. Includes material from *R for Beginners* by Emmanuel Paradis (with permission).

Getting help

Most R functions have online documentation.

`help(topic)` documentation on `topic`
`?topic` id.
`help.search("topic")` search the help system
`apropos("topic")` the names of all objects in the search list matching the regular expression "topic"
`help.start()` start the HTML version of help
`str(a)` display the internal *str*ucture of an R object
`summary(a)` gives a "summary" of `a`, usually a statistical summary but it is *generic* meaning it has different operations for different classes of `a`
`ls()` show objects in the search path; specify `pat="pat"` to search on a pattern
`ls.str()` str() for each variable in the search path
`dir()` show files in the current directory
`methods(a)` shows S3 methods of `a`
`methods(class=class(a))` lists all the methods to handle objects of class a

Input and output

`load()` load the datasets written with `save`
`data(x)` loads specified data sets
`library(x)` load add-on packages
`read.table(file)` reads a file in table format and creates a data frame from it; the default separator `sep=""` is any whitespace; use `header=TRUE` to read the first line as a header of column names; use `as.is=TRUE` to prevent character vectors from being converted to factors; use `comment.char=""` to prevent `"#"` from being interpreted as a comment; use `skip=n` to skip `n` lines before reading data; see the help for options on row naming, NA treatment, and others
`read.csv("filename",header=TRUE)` id. but with defaults set for reading comma-delimited files
`read.delim("filename",header=TRUE)` id. but with defaults set for reading tab-delimited files
`read.fwf(file,widths,header=FALSE,sep="",as.is=FALSE)` read a table of *f*ixed *w*idth *f*ormatted data into a 'data.frame'; `widths` is an integer vector, giving the widths of the fixed-width fields
`save(file,...)` saves the specified objects (...) in the XDR platform-independent binary format
`save.image(file)` saves all objects
`cat(..., file="", sep=" ")` prints the arguments after coercing to character; `sep` is the character separator between arguments
`print(a, ...)` prints its arguments; generic, meaning it can have different methods for different objects
`format(x,...)` format an R object for pretty printing
`write.table(x,file="",row.names=TRUE,col.names=TRUE, sep=" ")` prints `x` after converting to a data frame; if `quote` is `TRUE`, character or factor columns are surrounded by quotes ("); `sep` is the field separator; `eol` is the end-of-line separator; `na` is the string for missing values; use `col.names=NA` to add a blank column header to get the column headers aligned correctly for spreadsheet input
`sink(file)` output to `file`, until `sink()`

Most of the I/O functions have a `file` argument. This can often be a character string naming a file or a connection. `file=""` means the standard input or output. Connections can include files, pipes, zipped files, and R variables.

On windows, the file connection can also be used with `description = "clipboard"`. To read a table copied from Excel, use
`x <- read.delim("clipboard")`
To write a table to the clipboard for Excel, use
`write.table(x,"clipboard",sep="\t",col.names=NA)`
For database interaction, see packages `RODBC`, `DBI`, `RMySQL`, `RPgSQL`, and `ROracle`. See packages `XML`, `hdf5`, `netCDF` for reading other file formats.

Data creation

`c(...)` generic function to combine arguments with the default forming a vector; with `recursive=TRUE` descends through lists combining all elements into one vector
`from:to` generates a sequence; ":" has operator priority; 1:4 + 1 is "2,3,4,5"
`seq(from,to)` generates a sequence `by=` specifies increment; `length=` specifies desired length
`seq(along=x)` generates `1, 2, ..., length(along)`; useful for `for` loops
`rep(x,times)` replicate `x times`; use `each=` to repeat "each" element of `x each` times; `rep(c(1,2,3),2)` is 1 2 3 1 2 3; `rep(c(1,2,3),each=2)` is 1 1 2 2 3 3
`data.frame(...)` create a data frame of the named or unnamed arguments; `data.frame(v=1:4,ch=c("a","B","c","d"),n=10)`; shorter vectors are recycled to the length of the longest
`list(...)` create a list of the named or unnamed arguments; `list(a=c(1,2),b="hi",c=3i)`;
`array(x,dim=)` array with data `x`; specify dimensions like `dim=c(3,4,2)`; elements of `x` recycle if `x` is not long enough
`matrix(x,nrow=,ncol=)` matrix; elements of `x` recycle
`factor(x,levels=)` encodes a vector `x` as a factor
`gl(n,k,length=n*k,labels=1:n)` generate levels (factors) by specifying the pattern of their levels; `k` is the number of levels, and `n` is the number of replications
`expand.grid()` a data frame from all combinations of the supplied vectors or factors
`rbind(...)` combine arguments by rows for matrices, data frames, and others
`cbind(...)` id. by columns

Slicing and extracting data

Indexing vectors

`x[n]`	n-th element
`x[-n]`	all *but* the n-th element
`x[1:n]`	first `n` elements
`x[-(1:n)]`	elements from `n+1` to the end
`x[c(1,4,2)]`	specific elements
`x["name"]`	element named `"name"`
`x[x > 3]`	all elements greater than 3
`x[x > 3 & x < 5]`	all elements between 3 and 5
`x[x %in% c("a","and","the")]`	elements in the given set

Indexing lists

`x[n]`	list with elements `n`
`x[[n]]`	n-th element of the list
`x[["name"]]`	element of the list named `"name"`
`x$name`	id.

Indexing matrices

`x[i,j]`	element at row `i`, column `j`
`x[i,]`	row `i`
`x[,j]`	column `j`
`x[,c(1,3)]`	columns 1 and 3
`x["name",]`	row named `"name"`

Indexing data frames (matrix indexing plus the following)

`x[["name"]]`	column named `"name"`
`x$name`	id.

Variable conversion

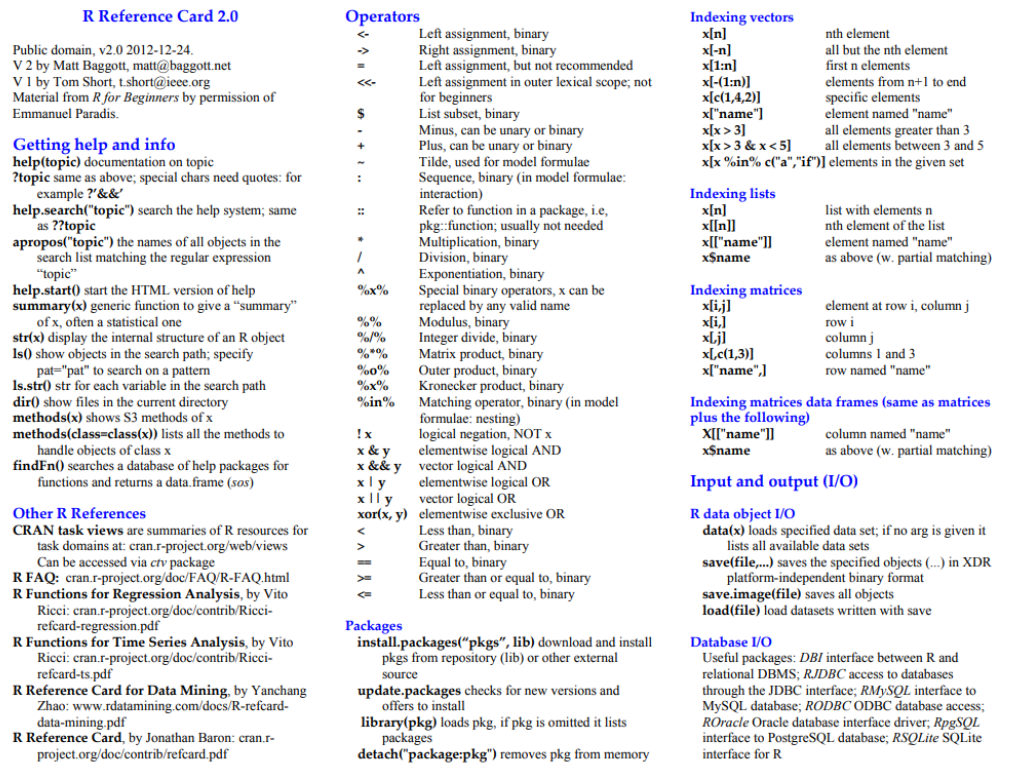
`as.array(x), as.data.frame(x), as.numeric(x), as.logical(x), as.complex(x), as.character(x),` ... convert type; for a complete list, use `methods(as)`

Variable information

`is.na(x), is.null(x), is.array(x), is.data.frame(x), is.numeric(x), is.complex(x), is.character(x),` ... test for type; for a complete list, use `methods(is)`
`length(x)` number of elements in `x`
`dim(x)` Retrieve or set the dimension of an object; `dim(x) <- c(3,2)`
`dimnames(x)` Retrieve or set the dimension names of an object
`nrow(x)` number of rows; `NROW(x)` is the same but treats a vector as a one-row matrix
`ncol(x)` and **`NCOL(x)`** id. for columns
`class(x)` get or set the class of `x`; `class(x) <- "myclass"`
`unclass(x)` remove the class attribute of `x`
`attr(x,which)` get or set the attribute `which` of `x`
`attributes(obj)` get or set the list of attributes of `obj`

Data selection and manipulation

`which.max(x)` returns the index of the greatest element of **x**
`which.min(x)` returns the index of the smallest element of **x**
`rev(x)` reverses the elements of `x`
`sort(x)` sorts the elements of `x` in increasing order; to sort in decreasing order: `rev(sort(x))`
`cut(x,breaks)` divides `x` into intervals (factors); `breaks` is the number of cut intervals or a vector of cut points
`match(x, y)` returns a vector of the same length than `x` with the elements of `x` which are in `y` (`NA` otherwise)
`which(x == a)` returns a vector of the indices of `x` if the comparison operation is true (`TRUE`), in this example the values of `i` for which `x[i] == a` (the argument of this function must be a variable of mode logical)
`choose(n, k)` computes the combinations of k events among n repetitions $= n!/[(n-k)!k!]$
`na.omit(x)` suppresses the observations with missing data (`NA`) (suppresses the corresponding line if `x` is a matrix or a data frame)
`na.fail(x)` returns an error message if `x` contains at least one `NA`



R Reference Card 2.0

Public domain, v2.0 2012-12-24.
V 2 by Matt Baggott, matt@baggott.net
V 1 by Tom Short, t.short@ieee.org
Material from *R for Beginners* by permission of Emmanuel Paradis.

Getting help and info

help(topic) documentation on topic
?topic same as above; special chars need quotes: for example ?'&&'
help.search("topic") search the help system; same as ??topic
apropos("topic") the names of all objects in the search list matching the regular expression "topic"
help.start() start the HTML version of help
summary(x) generic function to give a "summary" of x, often a statistical one
str(x) display the internal structure of an R object
ls() show objects in the search path; specify pat="pat" to search on a pattern
ls.str() str for each variable in the search path
dir() show files in the current directory
methods(x) shows S3 methods of x
methods(class=class(x)) lists all the methods to handle objects of class x
findFn() searches a database of help packages for functions and returns a data.frame (*sos*)

Other R References

CRAN task views are summaries of R resources for task domains at: cran.r-project.org/web/views Can be accessed via *ctv* package
R FAQ: cran.r-project.org/doc/FAQ/R-FAQ.html
R Functions for Regression Analysis, by Vito Ricci: cran.r-project.org/doc/contrib/Ricci-refcard-regression.pdf
R Functions for Time Series Analysis, by Vito Ricci: cran.r-project.org/doc/contrib/Ricci-refcard-ts.pdf
R Reference Card for Data Mining, by Yanchang Zhao: www.rdatamining.com/docs/R-refcard-data-mining.pdf
R Reference Card, by Jonathan Baron: cran.r-project.org/doc/contrib/refcard.pdf

Operators

<-	Left assignment, binary
->	Right assignment, binary
=	Left assignment, but not recommended
<<-	Left assignment in outer lexical scope; not for beginners
$	List subset, binary
-	Minus, can be unary or binary
+	Plus, can be unary or binary
~	Tilde, used for model formulae
:	Sequence, binary (in model formulae: interaction)
::	Refer to function in a package, i.e, pkg::function; usually not needed
*****	Multiplication, binary
/	Division, binary
^	Exponentiation, binary
%x%	Special binary operators, x can be replaced by any valid name
%%	Modulus, binary
%/%	Integer divide, binary
%*%	Matrix product, binary
%o%	Outer product, binary
%x%	Kronecker product, binary
%in%	Matching operator, binary (in model formulae: nesting)
! x	logical negation, NOT x
x & y	elementwise logical AND
x && y	vector logical AND
x | y	elementwise logical OR
x || y	vector logical OR
xor(x, y)	elementwise exclusive OR
<	Less than, binary
>	Greater than, binary
==	Equal to, binary
>=	Greater than or equal to, binary
<=	Less than or equal to, binary

Packages

install.packages("pkgs", lib) download and install pkgs from repository (lib) or other external source
update.packages checks for new versions and offers to install
library(pkg) loads pkg, if pkg is omitted it lists packages
detach("package:pkg") removes pkg from memory

Indexing vectors

x[n]	nth element
x[-n]	all but the nth element
x[1:n]	first n elements
x[-(1:n)]	elements from n+1 to end
x[c(1,4,2)]	specific elements
x["name"]	element named "name"
x[x > 3]	all elements greater than 3
x[x > 3 & x < 5]	all elements between 3 and 5
x[x %in% c("a","if")]	elements in the given set

Indexing lists

x[n]	list with elements n
x[[n]]	nth element of the list
x[["name"]]	element named "name"
x$name	as above (w. partial matching)

Indexing matrices

x[i,j]	element at row i, column j
x[i,]	row i
x[,j]	column j
x[,c(1,3)]	columns 1 and 3
x["name",]	row named "name"

Indexing matrices data frames (same as matrices plus the following)

X[["name"]]	column named "name"
x$name	as above (w. partial matching)

Input and output (I/O)

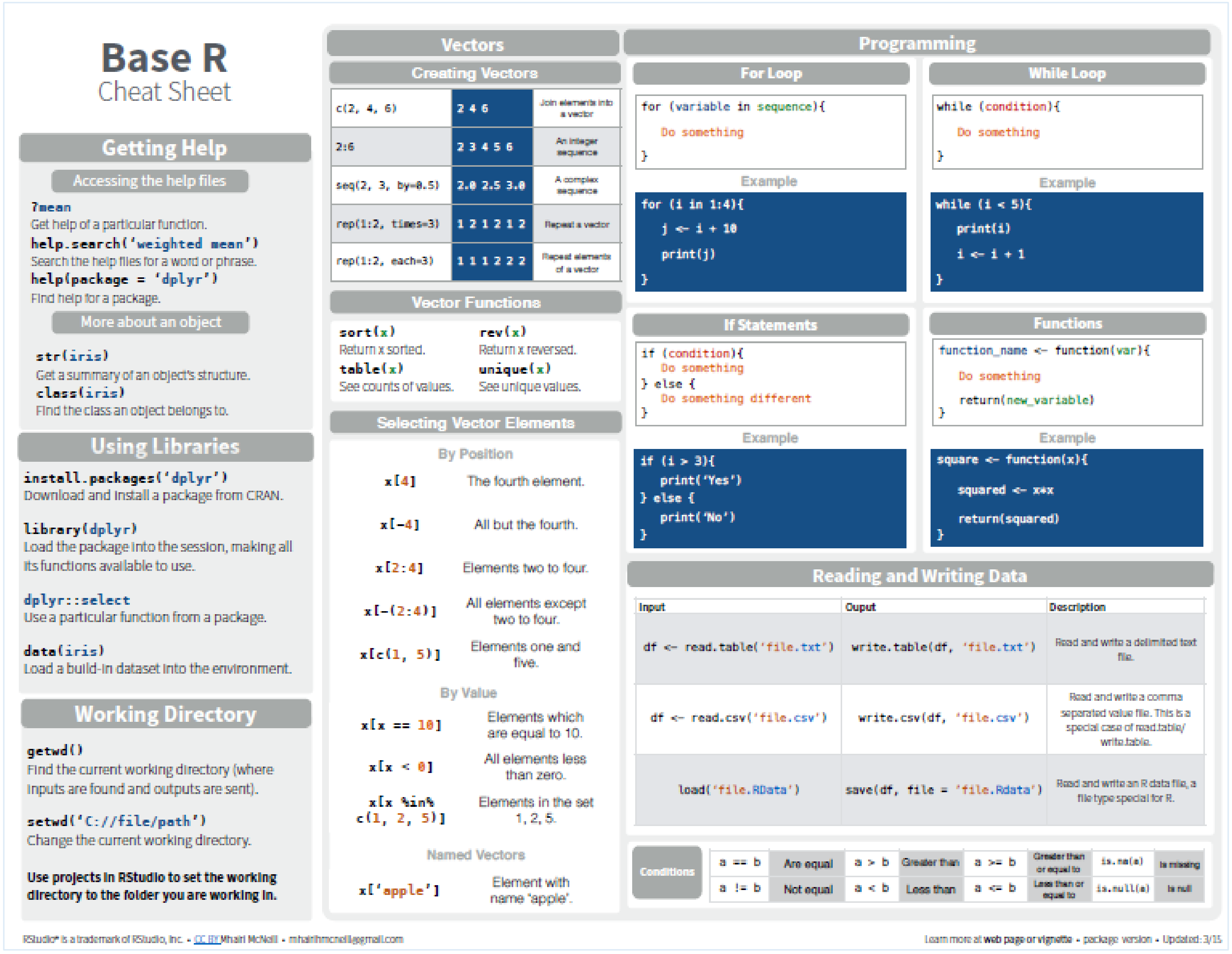
R data object I/O

data(x) loads specified data set; if no arg is given it lists all available data sets
save(file,...) saves the specified objects (...) in XDR platform-independent binary format
save.image(file) saves all objects
load(file) load datasets written with save

Database I/O

Useful packages: *DBI* interface between R and relational DBMS; *RJDBC* access to databases through the JDBC interface; *RMySQL* interface to MySQL database; *RODBC* ODBC database access; *ROracle* Oracle database interface driver; *RpgSQL* interface to PostgreSQL database; *RSQLite* SQLite interface for R

R Cheat Sheet



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 Libraries

```
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 build-in dataset into the environment.

Working Directory

```
getwd()
```
Find the current working directory (where inputs are found and outputs are sent).

```
setwd('C://file/path')
```
Change the current working directory.

Use projects in RStudio to set the working directory to the folder you are working in.

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.

By Value

`x[x == 10]`	Elements which are equal to 10.
`x[x < 0]`	All elements less than zero.
`x[x %in% c(1, 2, 5)]`	Elements in the set 1, 2, 5.

Named Vectors

`x['apple']`	Element with name 'apple'.

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

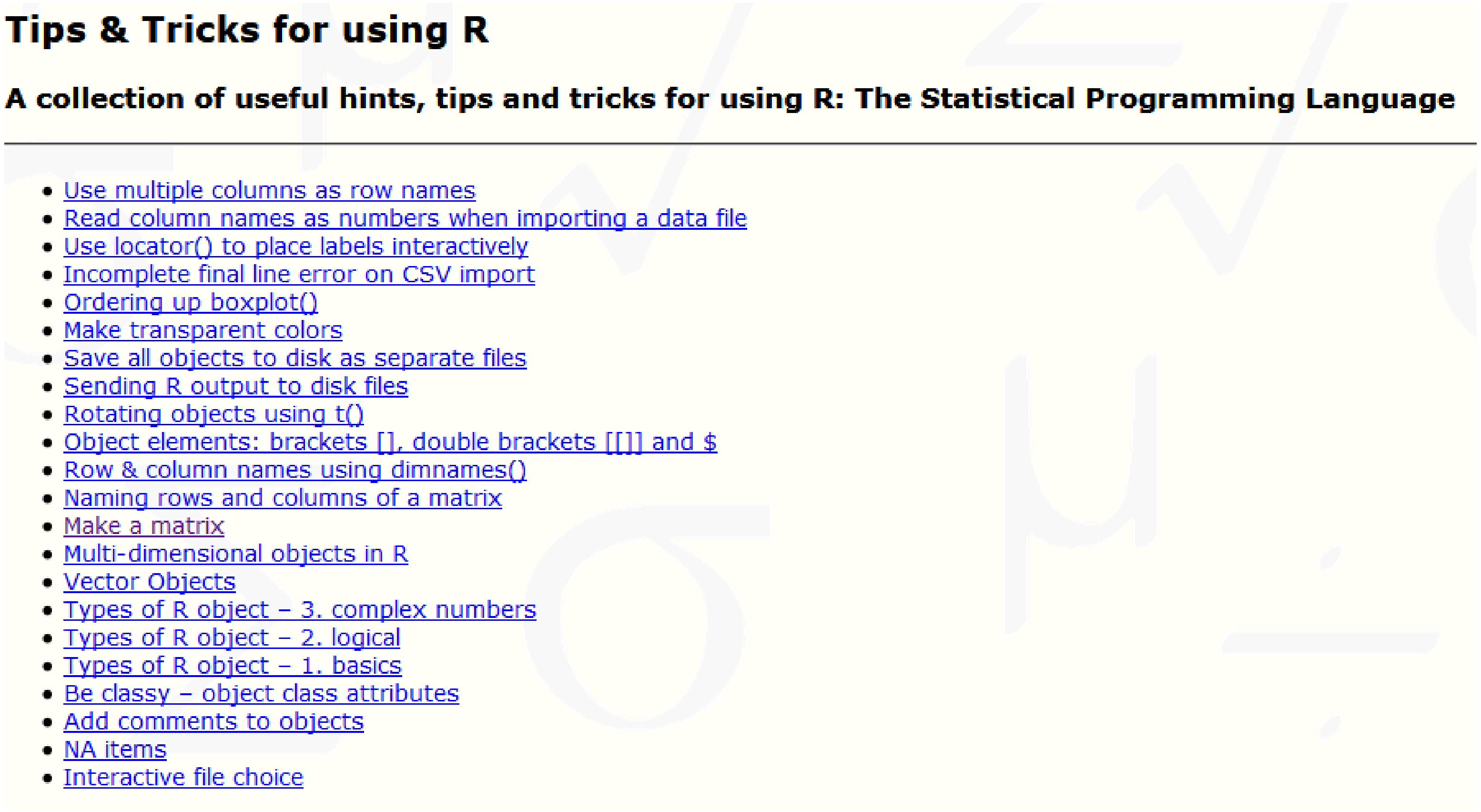
Input	Ouput	Description
`df <- read.table('file.txt')`	`write.table(df, 'file.txt')`	Read and write a delimited text file.
`df <- read.csv('file.csv')`	`write.csv(df, 'file.csv')`	Read and write a comma separated value file. This is a special case of read.table/write.table.
`load('file.RData')`	`save(df, file = 'file.Rdata')`	Read and write an R data file, a file type special for R.

Conditions								
	`a == b`	Are equal	`a > b`	Greater than	`a >= b`	Greater than or equal to	`is.na(a)`	Is missing
	`a != b`	Not equal	`a < b`	Less than	`a <= b`	Less than or equal to	`is.null(a)`	Is null

RStudio® is a trademark of RStudio, Inc. • CC BY Mhairi McNeill • mhairihmcneill@gmail.com

Learn more at web page or vignette • package version • Updated: 3/15

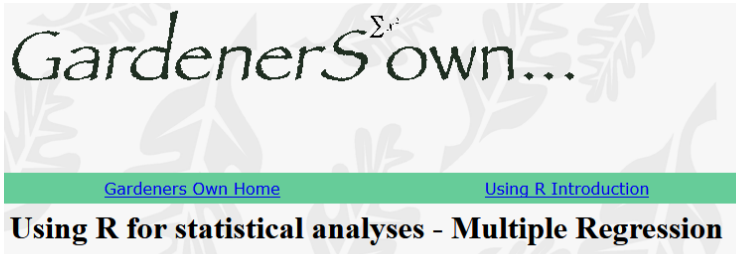
More on R



Tips & Tricks for using R

A collection of useful hints, tips and tricks for using R: The Statistical Programming Language

- Use multiple columns as row names
- Read column names as numbers when importing a data file
- Use locator() to place labels interactively
- Incomplete final line error on CSV import
- Ordering up boxplot()
- Make transparent colors
- Save all objects to disk as separate files
- Sending R output to disk files
- Rotating objects using t()
- Object elements: brackets [], double brackets [[]] and $
- Row & column names using dimnames()
- Naming rows and columns of a matrix
- Make a matrix
- Multi-dimensional objects in R
- Vector Objects
- Types of R object – 3. complex numbers
- Types of R object – 2. logical
- Types of R object – 1. basics
- Be classy – object class attributes
- Add comments to objects
- NA items
- Interactive file choice

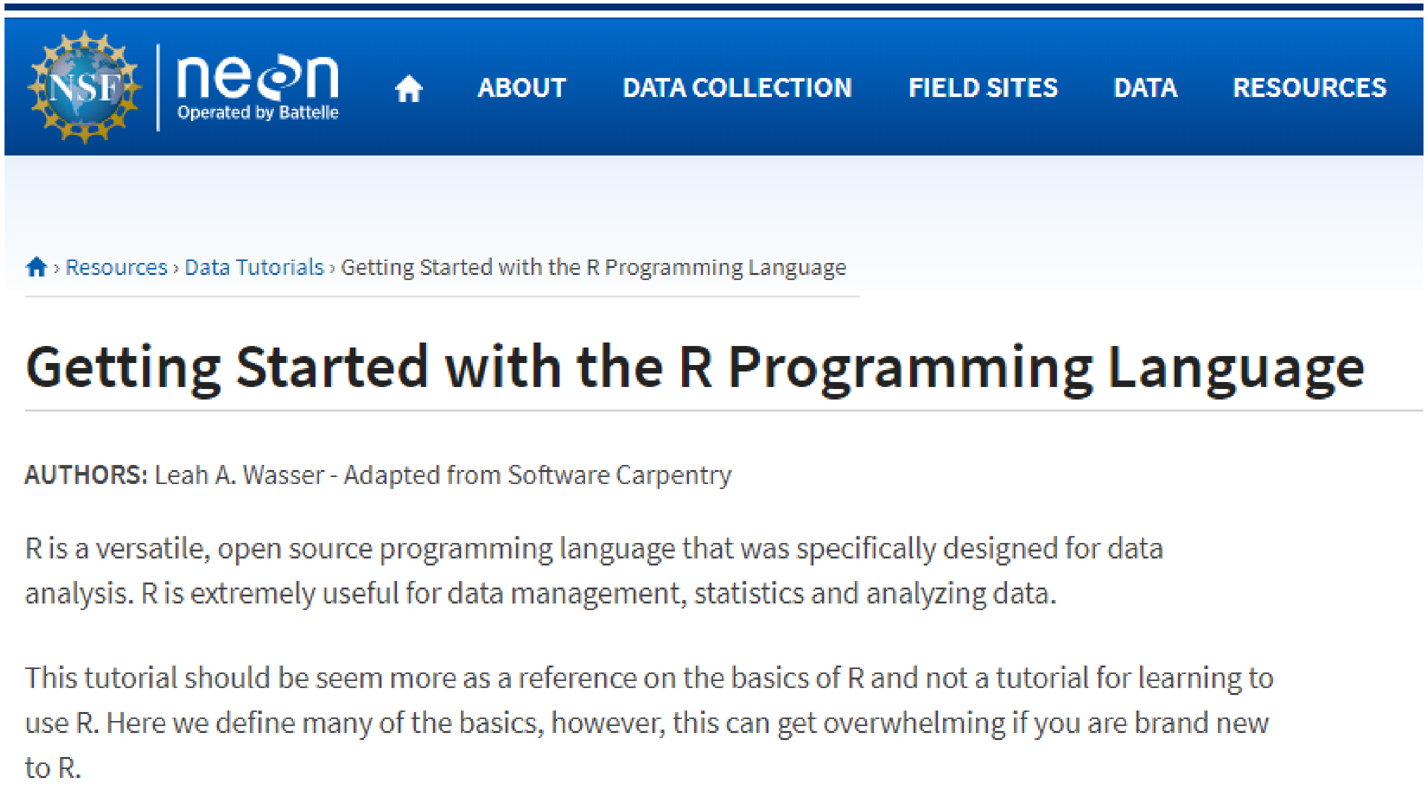


Using R for statistical analyses - Multiple Regression

Kingston University London

More on R

Perfect for Self-review of the Basics


NSF
neon
Operated by Battelle
ABOUT
DATA COLLECTION
FIELD SITES
DATA
RESOURCES
› Resources › Data Tutorials › Getting Started with the R Programming Language

Getting Started with the R Programming Language

AUTHORS: Leah A. Wasser - Adapted from Software Carpentry

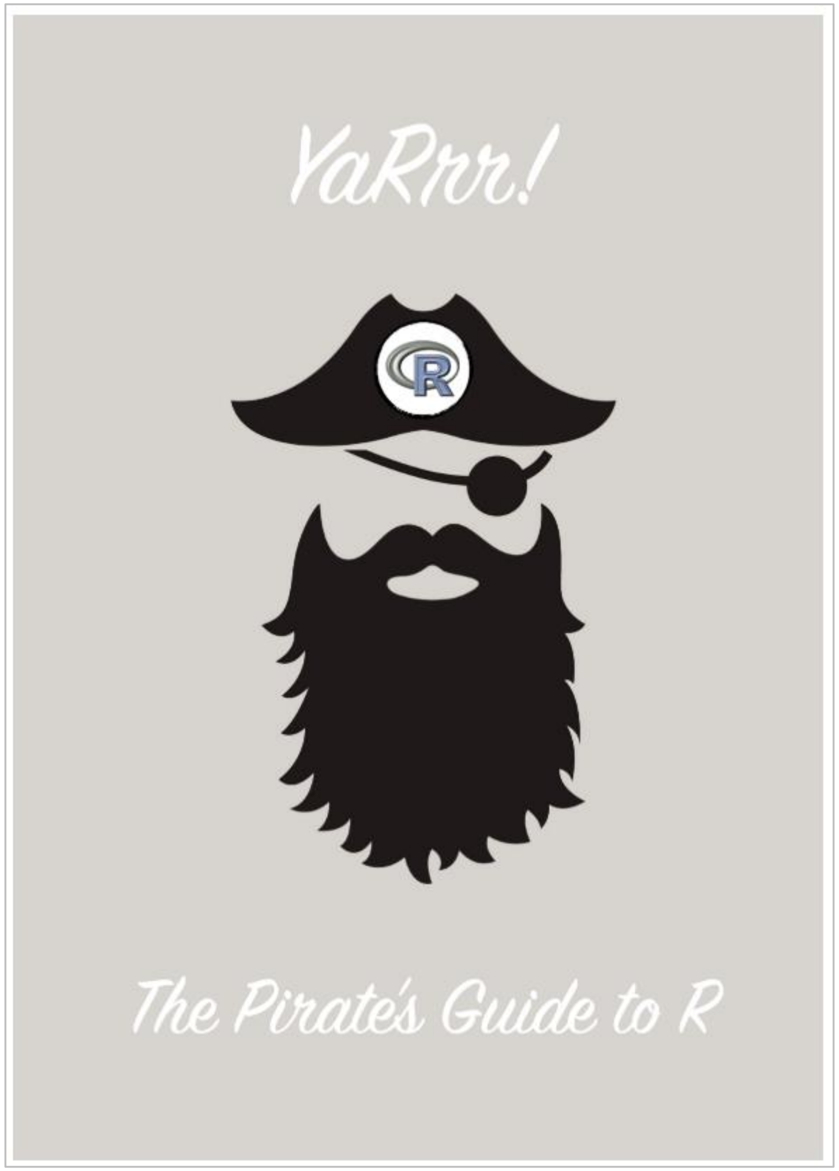
R is a versatile, open source programming language that was specifically designed for data analysis. R is extremely useful for data management, statistics and analyzing data.

This tutorial should be seem more as a reference on the basics of R and not a tutorial for learning to use R. Here we define many of the basics, however, this can get overwhelming if you are brand new to R.

And more: https://www.neonscience.org/resources/data-tutorials
(especially: https://www.neonscience.org/packages-in-r)

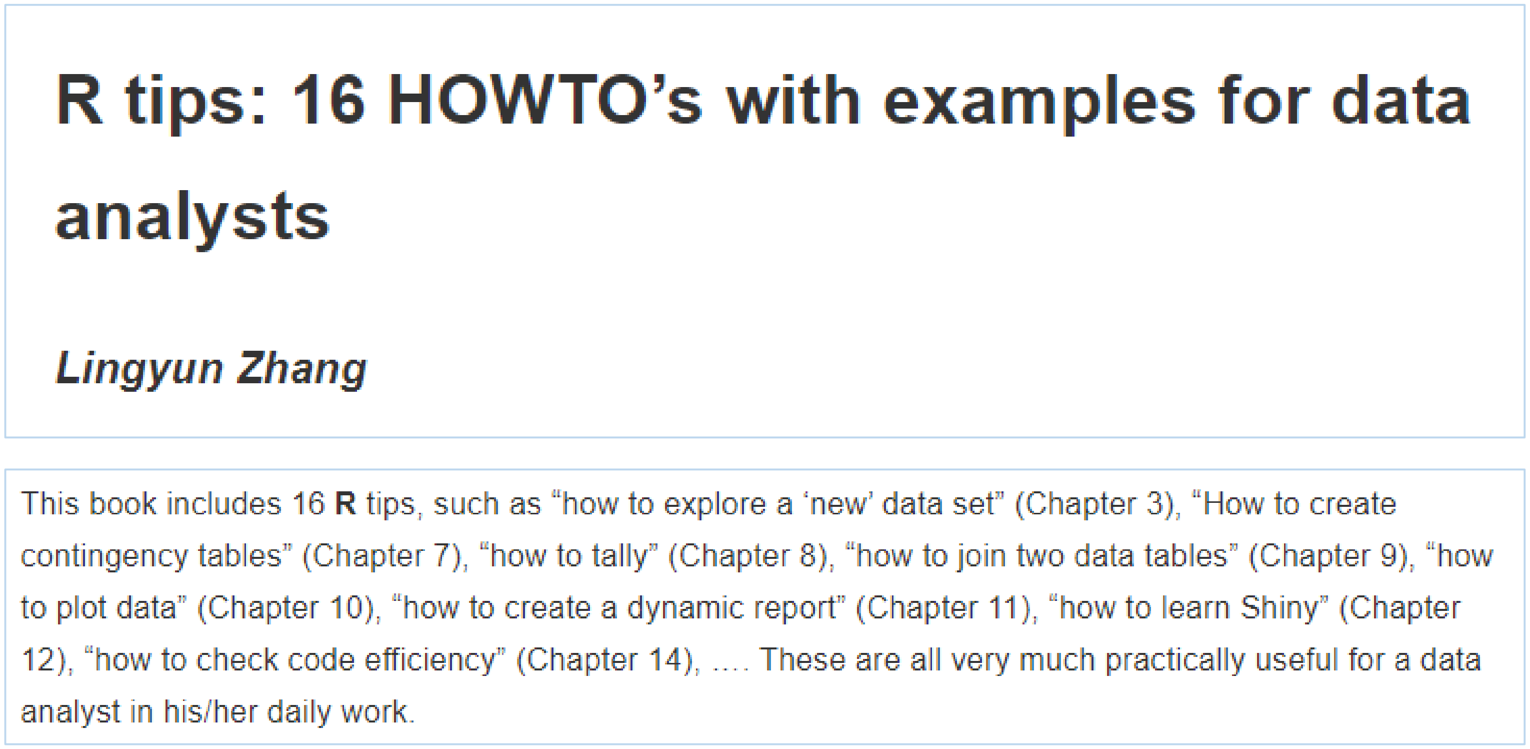
Kingston University London

More on R



The purpose of this book is to help you learn R from the ground-up.

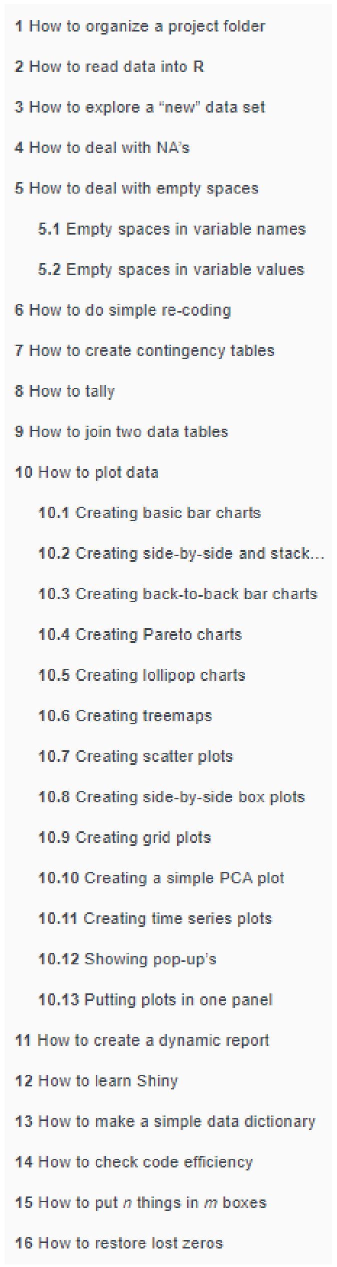
More on R



R tips: 16 HOWTO's with examples for data analysts

Lingyun Zhang

This book includes 16 **R** tips, such as "how to explore a 'new' data set" (Chapter 3), "How to create contingency tables" (Chapter 7), "how to tally" (Chapter 8), "how to join two data tables" (Chapter 9), "how to plot data" (Chapter 10), "how to create a dynamic report" (Chapter 11), "how to learn Shiny" (Chapter 12), "how to check code efficiency" (Chapter 14), …. These are all very much practically useful for a data analyst in his/her daily work.



More on R

http://www.dorak.info/r

R Notes and R Links

Next

R for Descriptive Statistics