

Intro to R

The R Bootcamp
www.therbootcamp.com
[@therbootcamp](https://twitter.com/therbootcamp)

July 2018

R

From [Wikipedia](#) (emphasis added):

R is an **open source programming language** and software environment for **statistical computing and graphics** that is supported by the **R Foundation for Statistical Computing**. The R language is **widely used among statisticians and data miners** for developing statistical software and data analysis. Polls, surveys of data miners, and studies of scholarly literature databases show that **R's popularity has increased substantially in recent years**.

R is a GNU package. The source code for the R software environment is written primarily in **C, Fortran, and R**. R is freely available under the GNU General Public License, and pre-compiled binary versions are provided for various operating systems. While R has a command line interface, there are several **graphical front-ends available**.

Programming language

From [Wikipedia](#) (emphasis added):

A programming language is a **formal language** that specifies a set of instructions that can be used to produce various kinds of output. Programming languages generally consist of **instructions for a computer**. Programming languages can be used to create programs that **implement specific algorithms**.

Algorithm

1. Load data
2. Extract variables
3. Run analysis
4. Print result

Implementation in R

```
data <- read.table(link)
variables <- data[,c('group', 'variable')]
analysis <- lm(variable ~ group, data = variables)
summary(analysis)
```

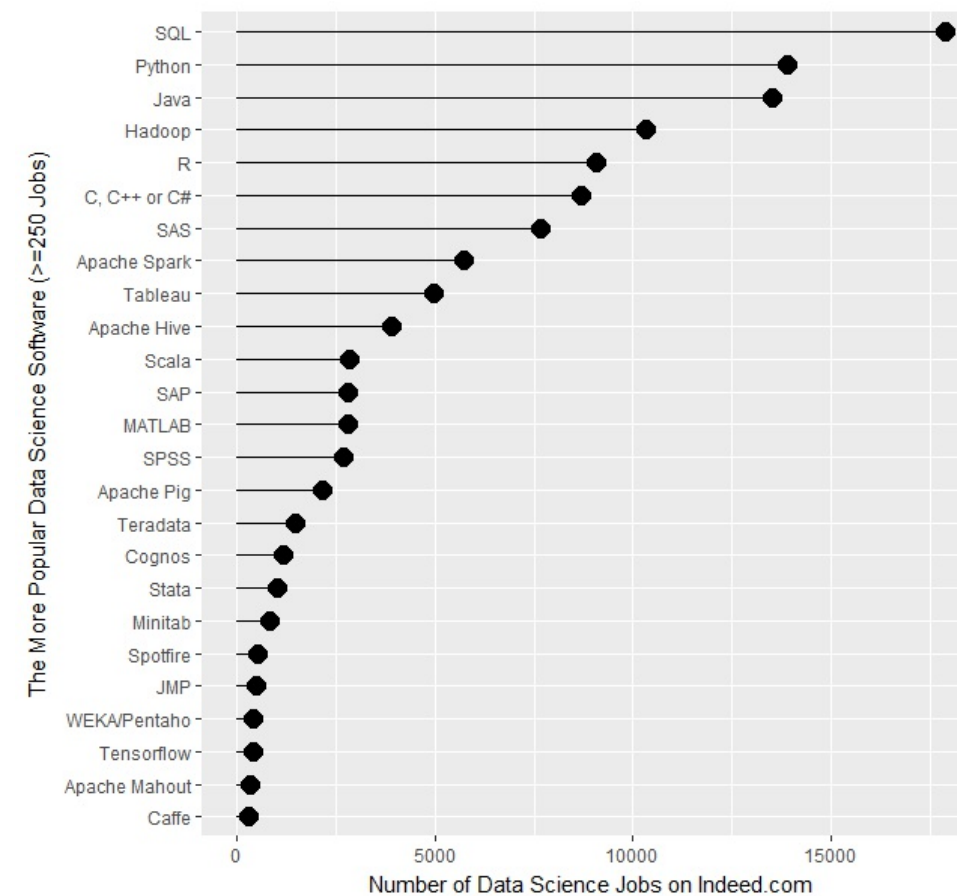
Why R?

R steadily **grows in popularity**.

Today, R is one of the **most popular languages for data science** and overall.

In terms of the number of data science jobs, **R beats SAS and Matlab**, and is on par with Python.

Image source: <https://i0.wp.com/r4stats.com/>



R is so popular because

There are many good reasons to prefer R over superficially more user friendly software such as **Excel** or **SPSS** or more complex programming languages like **C++** or **Python**.

Pro

1. **It's free**
2. Relatively **easy**
3. **Extensibility** (**CRAN**, packages)
4. **User base** (e.g., **stackoverflow**)
5. **Tidyverse** (dplyr, ggplot, etc.)
6. **RStudio**
7. **Productivity** options: **Latex**, **Markdown**, **GitHub**

Con

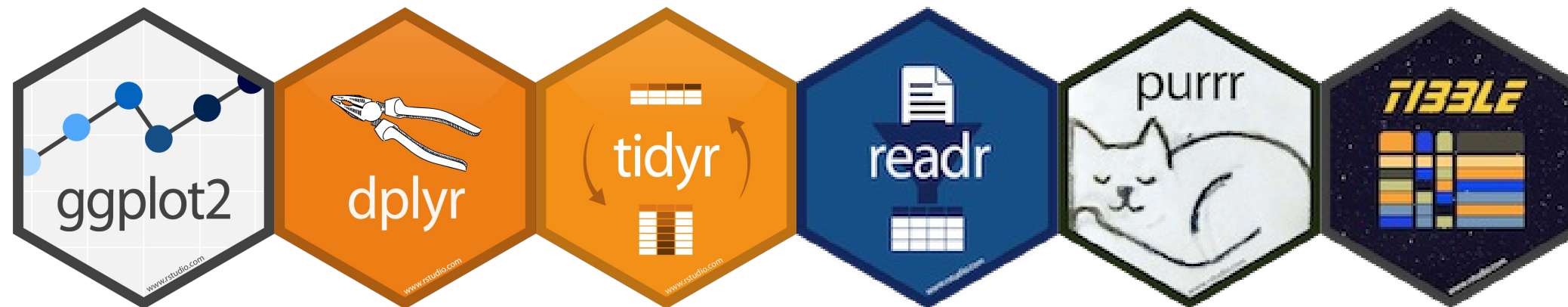
Sometimes slow and awkward, but...

Tidyverse Rcpp, **BH**: Links R to C++ and high-performance C++ libraries
rPython: Links R to Python
RHadoop: Links R to Hadoop for big data applications.

The almighty tidyverse

Among its many packages, R newly contains a collection of high-performance, user-friendly packages (libraries) known as the **tidyverse**. The tidyverse includes:

1. `ggplot2` -- creating graphics.
2. `dplyr` -- data manipulation.
3. `tidyr` -- tidying data.
4. `readr` -- read wild data.
5. `purrr` -- functional programming.
6. `tibble` -- modern data frame.



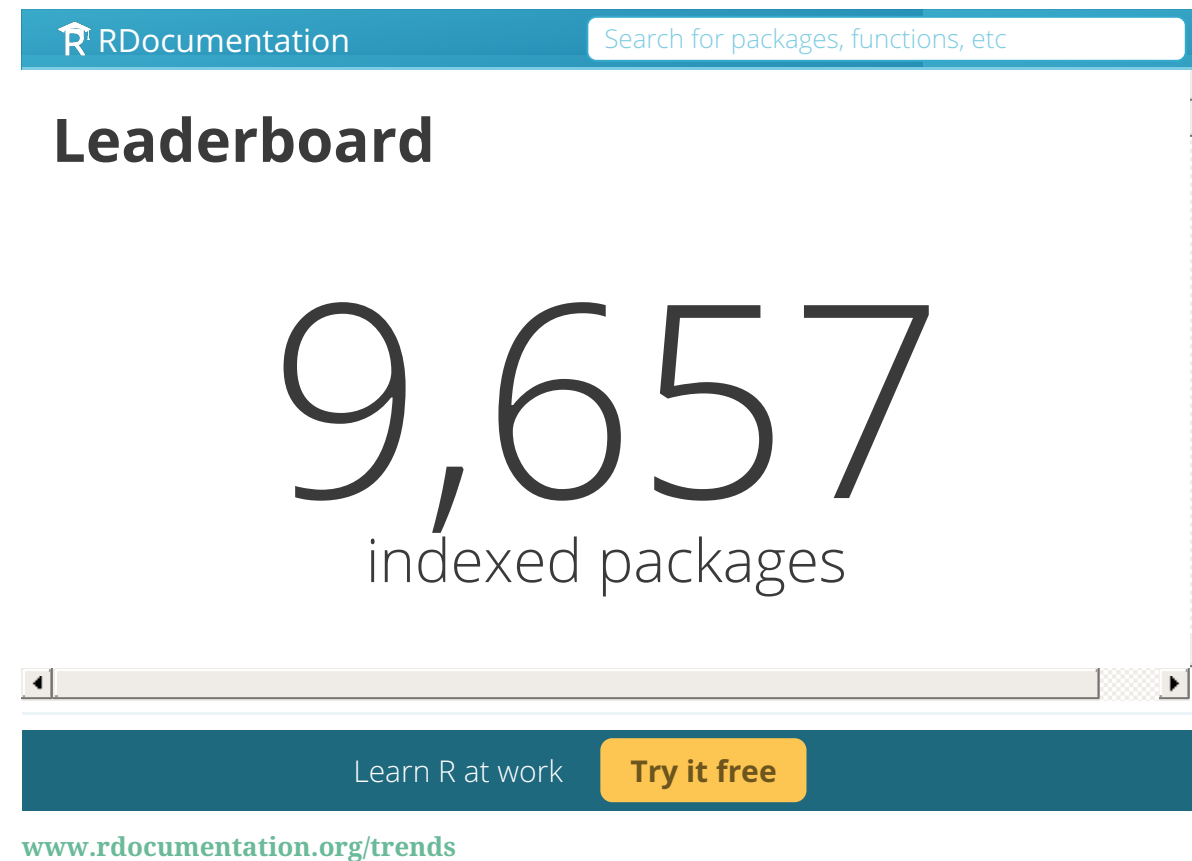
Packages

R features a vast and cutting-edge collection of **packages** provided on **CRAN** and **Git/GitHub** by R's large and highly active user base and the work of .

```
# To install a package  
install.packages('package_name')
```

```
# load a package  
library(package_name)  
require(package_name)
```

```
#Note:  
# Don't forget that packages  
# must also be loaded.
```



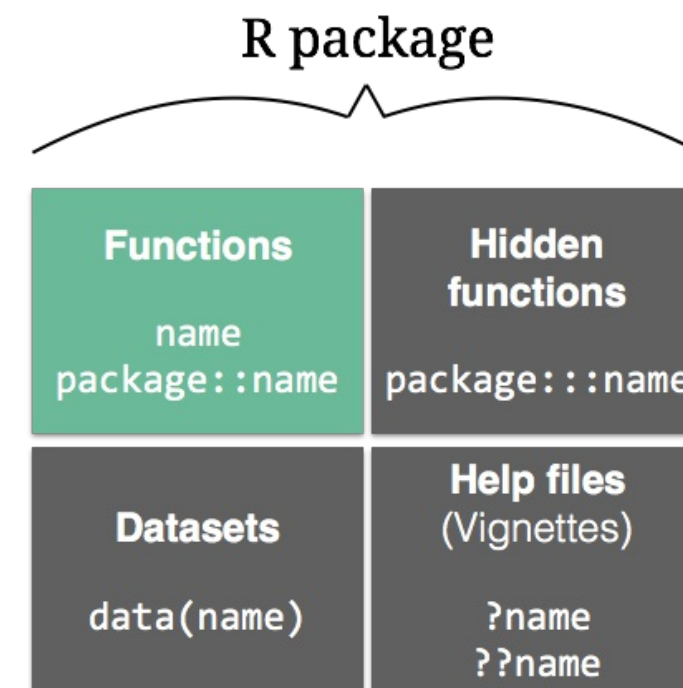
Packages

R features a vast and cutting-edge collection of **packages** provided on **CRAN** and **Git/GitHub** by R's large and highly active user base and the work of .

```
# To install a package
install.packages('package_name')

# load a package
library(package_name)
require(package_name)

#Note:
# Don't forget that packages
# must also be loaded.
```



RStudio: R's favorite environment

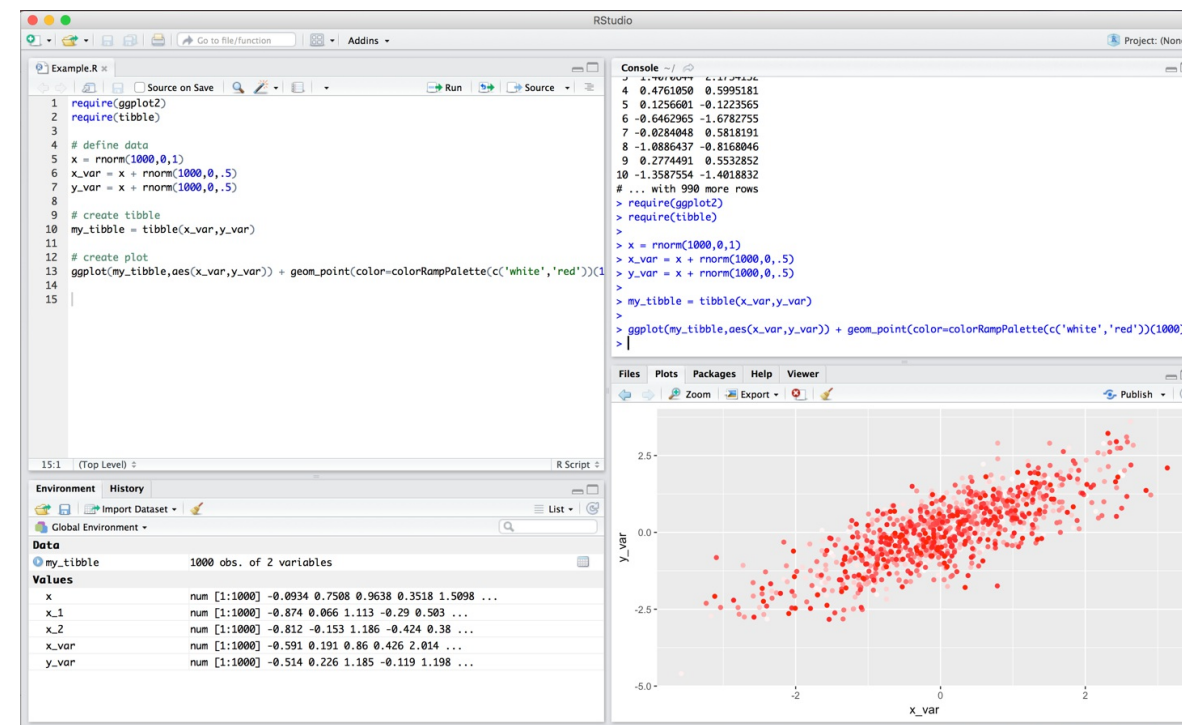
Next to many useful packages, R users greatly benefit from R's integrated development environment **RStudio**. Rstudio is a **graphical user interface** that allows you to (a) edit code, (b) run code, (c) access files and history, and (d) create plots. RStudio also helps you with **project management**, **version control** via **Github**, writing **reports** using **markdown** and **knitr**, and many other aspects of working with R.

Script editor

This is where you write your code.

Environment & History

Here you can track what you have done.



Console

This is where you talk to R. Here you run your code.

Plot, Help, Files, etc.

This window pane is mostly used for plotting and help files.

The 2⁴ Lessons of the R Bootcamp

Essentials: The 2⁴ Lessons of the R Bootcamp

1. Everything is an object
2. Use `<-` to create/change objects
3. Name objects using `_`
4. Objects have classes
5. Everything happens through functions
6. Functions have (default) arguments
7. Functions expect certain object classes
8. View help files using `?`
9. Study errors and warnings
10. Data is stored in data frames
11. Select variables (vectors) using `$`
12. Use RStudio and projects
13. Use editor and shortcuts
14. First load packages and data
15. Use auto-complete
16. Comment and format for readability

Essentials: The 2⁴ Lessons of the R Bootcamp

1. **Everything is an object**
2. **Use < - to create/change objects**
3. **Name objects using _**
4. Objects have classes
5. Everything happens through functions
6. Functions have (default) arguments
7. Functions expect certain object classes
8. View help files using ?
9. Study errors and warnings
10. Data is stored in data frames
11. Select variables (vectors) using \$
12. Use RStudio and projects
13. Use editor and shortcuts
14. First load packages and data
15. Use auto-complete
16. Comment and format for readability

```
# an object called some_name  
some_name <- c(1, 2, 3)  
  
# add 2 to the object's numbers  
some_name + 2
```

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

```
# print object  
some_name
```

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

```
# make change permanent  
some_name <- some_name + 2  
  
# print object  
some_name
```

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

Essentials: The 2⁴ Lessons of the R Bootcamp

1. Everything is an object
2. Use <- to create/change objects
3. Name objects using _
4. **Objects have classes**
5. Everything happens through functions
6. Functions have (default) arguments
7. Functions expect certain object classes
8. View help files using ?
9. Study errors and warnings
10. Data is stored in data frames
11. Select variables (vectors) using \$
12. Use RStudio and projects
13. Use editor and shortcuts
14. First load packages and data
15. Use auto-complete
16. Comment and format for readability

```
# an object called some_name  
class(some_name)
```

```
## [1] "numeric"
```

```
typeof(some_name)
```

```
## [1] "double"
```

```
# an object called some_name  
class(list())
```

```
## [1] "list"
```

```
# an object called some_name  
class(tibble())
```

```
## [1] "tbl_df"      "tbl"        "data.frame"
```

Essentials: The 2⁴ Lessons of the R Bootcamp

1. Everything is an object
2. Use <- to create/change objects
3. Name objects using _
4. Objects have classes
5. **Everything happens through functions**
6. Functions have (default) arguments
7. Functions expect certain object classes
8. View help files using ?
9. Study errors and warnings
10. Data is stored in data frames
11. Select variables (vectors) using \$
12. Use RStudio and projects
13. Use editor and shortcuts
14. First load packages and data
15. Use auto-complete
16. Comment and format for readability

```
# function c()  
some_name <- c(1, 2, 3)
```

```
# function `+`()  
some_name + 2
```

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

```
# function print()  
some_name
```

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

```
# function class()  
class(some_name)
```

```
## [1] "numeric"
```

Essentials: The 2⁴ Lessons of the R Bootcamp

1. Everything is an object
2. Use <- to create/change objects
3. Objects have classes
4. Name objects using _
5. Everything happens through functions
6. **Functions have (default) arguments**
7. Functions expect certain object classes
8. View help files using ?
9. Study errors and warnings
10. Data is stored in data frames
11. Select variables (vectors) using \$
12. Use RStudio and projects
13. Use editor and shortcuts
14. First load packages and data
15. Use auto-complete
16. Comment and format for readability

```
# no argument  
mean()
```

```
## Error in mean.default(): argument "x" is missing, with no
```

```
# required argument  
mean(c(1, 2, 3))
```

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

```
# introducing NA  
mean(c(1, 2, 3, NA))
```

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

```
# changing default to handle NA  
mean(c(1, 2, 3, NA), na.rm = TRUE)
```

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

Essentials: The 2⁴ Lessons of the R Bootcamp

1. Everything is an object
2. Use <- to create/change objects
3. Objects have classes
4. Name objects using _
5. Everything happens through functions
6. Functions have (default) arguments
7. **Functions expect certain object classes**
8. View help files using ?
9. Study errors and warnings
10. Data is stored in data frames
11. Select variables (vectors) using \$
12. Use RStudio and projects
13. Use editor and shortcuts
14. First load packages and data
15. Use auto-complete
16. Comment and format for readability

```
# mean works also for logical  
mean(c(TRUE, FALSE, TRUE))
```

```
## [1] 0.6667
```

```
# but not for character  
mean(c("a", "b", "c"))
```

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

```
# classes relevant for all arg's  
mean(c(1, 2, 3), na.rm = "test")
```

```
## Error in if (na.rm) x <- x[!is.na(x)]: argument is not i
```


Essentials: The 2⁴ Lessons of the R Bootcamp

1. Everything is an object
2. Use `<-` to create/change objects
3. Objects have classes
4. Name objects using `_`
5. Everything happens through functions
6. Functions have (default) arguments
7. Functions expect certain object classes
8. **View help files using ?**
9. Study errors and warnings
10. Data is stored in data frames
11. Select variables (vectors) using `$`
12. Use RStudio and projects
13. Use editor and shortcuts
14. First load packages and data
15. Use auto-complete
16. Comment and format for readability

?mean

mean (base)

R Documentation

Arithmetic Mean

Description

Generic function for the (trimmed) arithmetic mean.

Usage

```
mean(x, ...)
```

```
## Default S3 method:
```

```
mean(x, trim = 0, na.rm = FALSE, ...)
```

Arguments

x An R object. Currently there are methods for numeric/logical vectors and [date](#), [date-time](#) and [time interval](#) objects. Complex vectors are allowed for `trim = 0`, only.

trim the fraction (0 to 0.5) of observations to be trimmed from each end of `x` before the mean is computed. Values of `trim` outside that range are taken as the nearest endpoint.

na.rm a logical value indicating whether NA values should be stripped before the computation proceeds.

... further arguments passed to or from other methods.

Value

If `trim` is zero (the default), the arithmetic mean of the values in `x` is computed, as a numeric or complex vector of length one. If `x` is not logical (coerced to numeric), numeric (including integer) or complex, `NA_real_` is returned, with a warning.

If `trim` is non-zero, a symmetrically trimmed mean is computed with a fraction of `trim` observations deleted from each end before the mean is computed.

References

Becker, R. A., Chambers, J. M. and Wilks, A. R. (1988) *The New S Language*. Wadsworth & Brooks/Cole.

Essentials: The 2⁴ Lessons of the R Bootcamp

1. Everything is an object
2. Use `<-` to create/change objects
3. Objects have classes
4. Name objects using `_`
5. Everything happens through functions
6. Functions have (default) arguments
7. Functions expect certain object classes
8. **View help files using ?**
9. Study errors and warnings
10. Data is stored in data frames
11. Select variables (vectors) using `$`
12. Use RStudio and projects
13. Use editor and shortcuts
14. First load packages and data
15. Use auto-complete
16. Comment and format for readability

?cor

cor (stats)

R Documentation

Correlation, Variance and Covariance (Matrices)

Description

`var`, `cov` and `cor` compute the variance of `x` and the covariance or correlation of `x` and `y` if these are vectors. If `x` and `y` are matrices then the covariances (or correlations) between the columns of `x` and the columns of `y` are computed.

`cov2cor` scales a covariance matrix into the corresponding correlation matrix *efficiently*.

Usage

```
var(x, y = NULL, na.rm = FALSE, use)
```

```
cov(x, y = NULL, use = "everything",  
    method = c("pearson", "kendall", "spearman"))
```

```
cor(x, y = NULL, use = "everything",  
    method = c("pearson", "kendall", "spearman"))
```

```
cov2cor(v)
```

Arguments

x a numeric vector, matrix or data frame.

y `NULL` (default) or a vector, matrix or data frame with compatible dimensions to `x`. The default is equivalent to `y = x` (but more efficient).

na.rm logical. Should missing values be removed?

use an optional character string giving a method for computing covariances in the presence of missing values. This must be (an abbreviation of) one of the strings "everything", "all.obs", "complete.obs", "na.or.complete", or "pairwise.complete.obs".

method a character string indicating which correlation coefficient (or covariance) is to be computed. One of "pearson" (default), "kendall", or "spearman": can be abbreviated.

v symmetric numeric matrix, usually positive definite such as a covariance matrix.

Essentials: The 2⁴ Lessons of the R Bootcamp

1. Everything is an object
2. Use <- to create/change objects
3. Objects have classes
4. Name objects using _
5. Everything happens through functions
6. Functions have (default) arguments
7. Functions expect certain object classes
8. View help files using ?
9. **Study errors and warnings**
10. Data is stored in data frames
11. Select variables (vectors) using \$
12. Use RStudio and projects
13. Use editor and shortcuts
14. First load packages and data
15. Use auto-complete
16. Comment and format for readability

```
# message - attend  
basel <- type_convert(baselers)
```

```
## Parsed with column specification:  
## cols(  
##   sex = col_character()  
## )
```

```
# warning - attend closely  
result <- mean('NA')
```

```
## Warning in mean.default("NA"): argument is not numeric or
```

```
# error - fix  
lenth(1)
```

```
## Error in lenth(1): could not find function "lenth"
```

Essentials: The 2⁴ Lessons of the R Bootcamp

1. Everything is an object
2. Use <- to create/change objects
3. Objects have classes
4. Name objects using _
5. Everything happens through functions
6. Functions have (default) arguments
7. Functions expect certain object classes
8. View help files using ?
9. **Study errors and warnings**
10. Data is stored in data frames
11. Select variables (vectors) using \$
12. Use RStudio and projects
13. Use editor and shortcuts
14. First load packages and data
15. Use auto-complete
16. Comment and format for readability

```
length(1)
```

```
## Error in length(1): could not find function "length"
```

Error	Description
'could not find function'	Typo or package not loaded
'error in eval'	An object is used in function that does not exist.
'cannot open()'	Typo or missing path.
'no applicable method'	Function inapplicable for type
package errors	Unable to install, compile, or load package.

Essentials: The 2⁴ Lessons of the R Bootcamp

1. Everything is an object
2. Use `<-` to create/change objects
3. Objects have classes
4. Name objects using `_`
5. Everything happens through functions
6. Functions have (default) arguments
7. Functions expect certain object classes
8. View help files using `?`
9. Study errors and warnings
10. **Data is stored in data frames**
11. Select variables (vectors) using `$`
12. Use RStudio and projects
13. Use editor and shortcuts
14. First load packages and data
15. Use auto-complete
16. Comment and format for readability

```
print(baselers)
```

```
## # A tibble: 10,000 x 20
##       id sex    age height weight income
##   <int> <chr> <int> <dbl> <dbl> <dbl>
## 1     1 male    44   174.  113.  6300.
## 2     2 male    65   180.   75.2 10900.
## 3     3 female   31   168.   55.5  5100.
## 4     4 male    27   209.   93.8  4200.
## 5     5 male    24   177.    NA   4000.
## 6     6 male    63   187.   67.4 11400.
## 7     7 male    71   152.   83.3 12000.
## 8     8 female   41   156.   67.8  7600.
## 9     9 male    43   176.   69.3  8500.
## 10    10 female   31   166.   66.3  6100.
## # ... with 9,990 more rows, and 14 more
## # variables
```

Essentials: The 2⁴ Lessons of the R Bootcamp

1. Everything is an object
2. Use <- to create/change objects
3. Objects have classes
4. Name objects using _
5. Everything happens through functions
6. Functions have (default) arguments
7. Functions expect certain object classes
8. View help files using ?
9. Study errors and warnings
10. Data is stored in data frames
11. **Select variables (vectors) using \$**
12. Use RStudio and projects
13. Use editor and shortcuts
14. First load packages and data
15. Use auto-complete
16. Comment and format for readability

```
# select sex variable using $  
baselers$sex
```

```
## [1] "male" "male" "female" "male" "male"  
## [6] "male" "male" "female"  
## [ reached getOption("max.print") -- omitted 9992 entries]
```

```
# Wherever possible, AVOID...  
baselers[['sex']]
```

```
## [1] "male" "male" "female" "male" "male"  
## [6] "male" "male" "female"  
## [ reached getOption("max.print") -- omitted 9992 entries]
```

```
baselers[[2]]
```

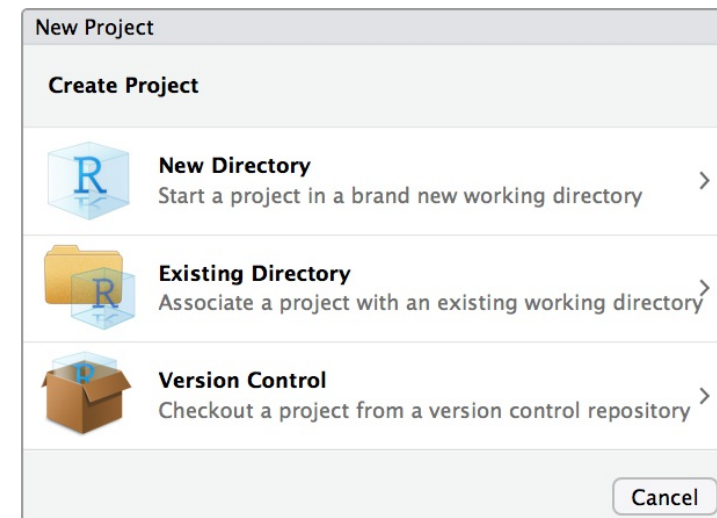
```
## [1] "male" "male" "female" "male" "male"  
## [6] "male" "male" "female"  
## [ reached getOption("max.print") -- omitted 9992 entries]
```

Essentials: The 2⁴ Lessons of the R Bootcamp

1. Everything is an object
2. Use `<-` to create/change objects
3. Objects have classes
4. Name objects using `_`
5. Everything happens through functions
6. Functions have (default) arguments
7. Functions expect certain object classes
8. View help files using `?`
9. Study errors and warnings
10. Data is stored in data frames
11. Select variables (vectors) using `$`
12. **Use RStudio and projects**
13. Use editor and shortcuts
14. First load packages and data
15. Use auto-complete
16. Comment and format for readability

Projects help...

save workspace and history • set project specific options •
access files • version control • etc.

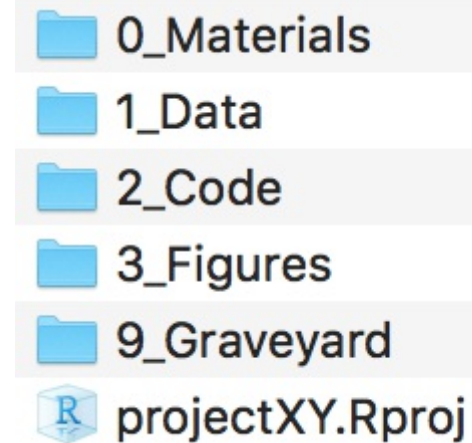


Essentials: The 2⁴ Lessons of the R Bootcamp

1. Everything is an object
2. Use `<-` to create/change objects
3. Objects have classes
4. Name objects using `_`
5. Everything happens through functions
6. Functions have (default) arguments
7. Functions expect certain object classes
8. View help files using `?`
9. Study errors and warnings
10. Data is stored in data frames
11. Select variables (vectors) using `$`
12. **Use RStudio and projects**
13. Use editor and shortcuts
14. First load packages and data
15. Use auto-complete
16. Comment and format for readability

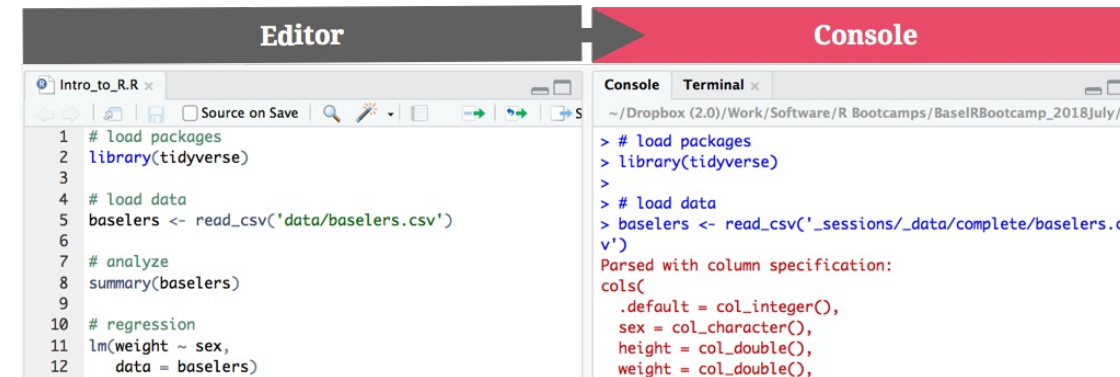
Folder structure

Complement projects by a **folder structure** appropriate for your project.



Essentials: The 2⁴ Lessons of the R Bootcamp

1. Everything is an object
2. Use <- to create/change objects
3. Objects have classes
4. Name objects using _
5. Everything happens through functions
6. Functions have (default) arguments
7. Functions expect certain object classes
8. View help files using ?
9. Study errors and warnings
10. Data is stored in data frames
11. Select variables (vectors) using \$
12. Use RStudio and projects
13. **Use editor and shortcuts**
14. First load packages and data
15. Use auto-complete
16. Comment and format for readability



```
1 # load packages
2 library(tidyverse)
3
4 # load data
5 baselers <- read_csv('data/baselers.csv')
6
7 # analyze
8 summary(baselers)
9
10 # regression
11 lm(weight ~ sex,
12     data = baselers)
```

```
> # load packages
> library(tidyverse)
>
> # load data
> baselers <- read_csv('_sessions/_data/complete/baselers.csv')
Parsed with column specification:
cols(
  .default = col_integer(),
  sex = col_character(),
  height = col_double(),
  weight = col_double(),
```

Shortcut to **send to console**:

⌘/ctrl + ↵

Shortcut to **rerun chunk**:

⌘/ctrl + ⇧ + p

Essentials: The 2⁴ Lessons of the R Bootcamp

1. Everything is an object
2. Use <- to create/change objects
3. Objects have classes
4. Name objects using _
5. Everything happens through functions
6. Functions have (default) arguments
7. Functions expect certain object classes
8. View help files using ?
9. Study errors and warnings
10. Data is stored in data frames
11. Select variables (vectors) using \$
12. Use RStudio and projects
13. Use editor and shortcuts
14. **First load packages and data**
15. Use auto-complete
16. Comment and format for readability

```
# import packages
library(tidyverse)
library(yarr)
library(lme4)

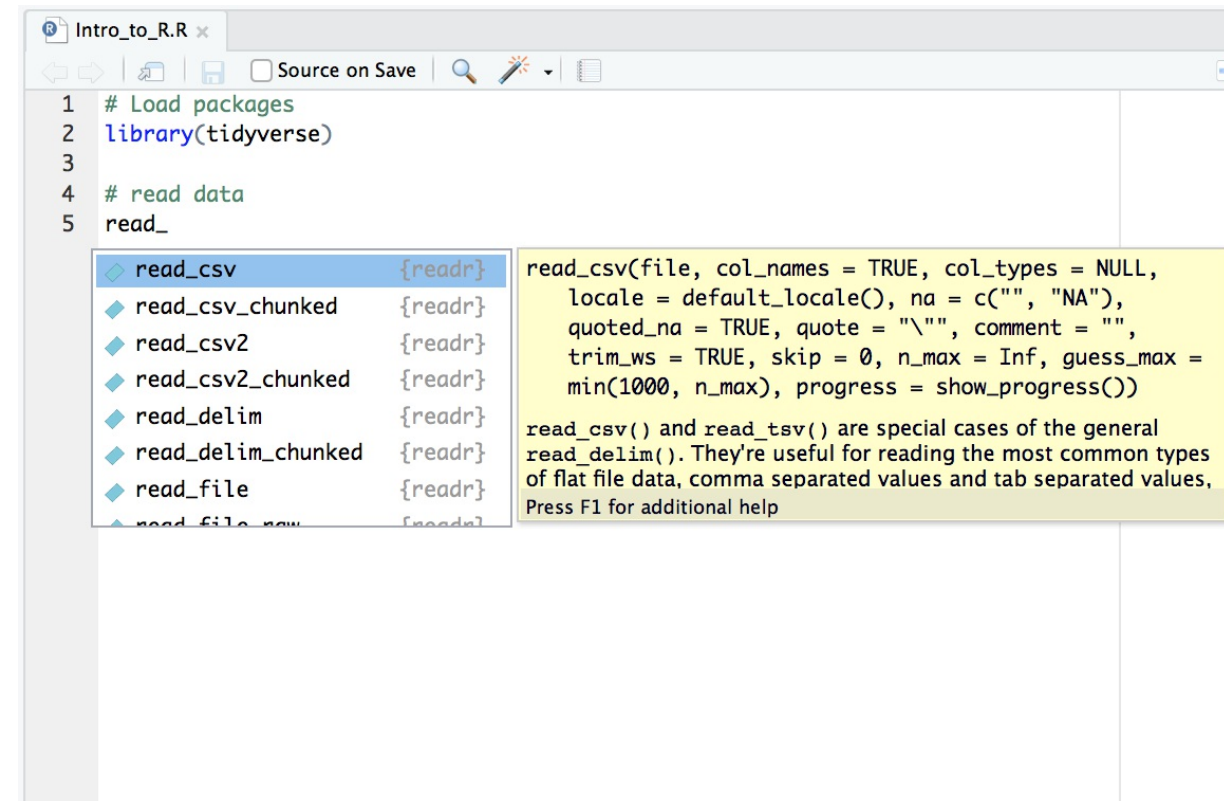
# import data
baselers <- read_delim("baselers.txt",
                      delim = '\t')
```

The goal is...

... to create self-contained scripts that run uninterrupted from beginning to start.

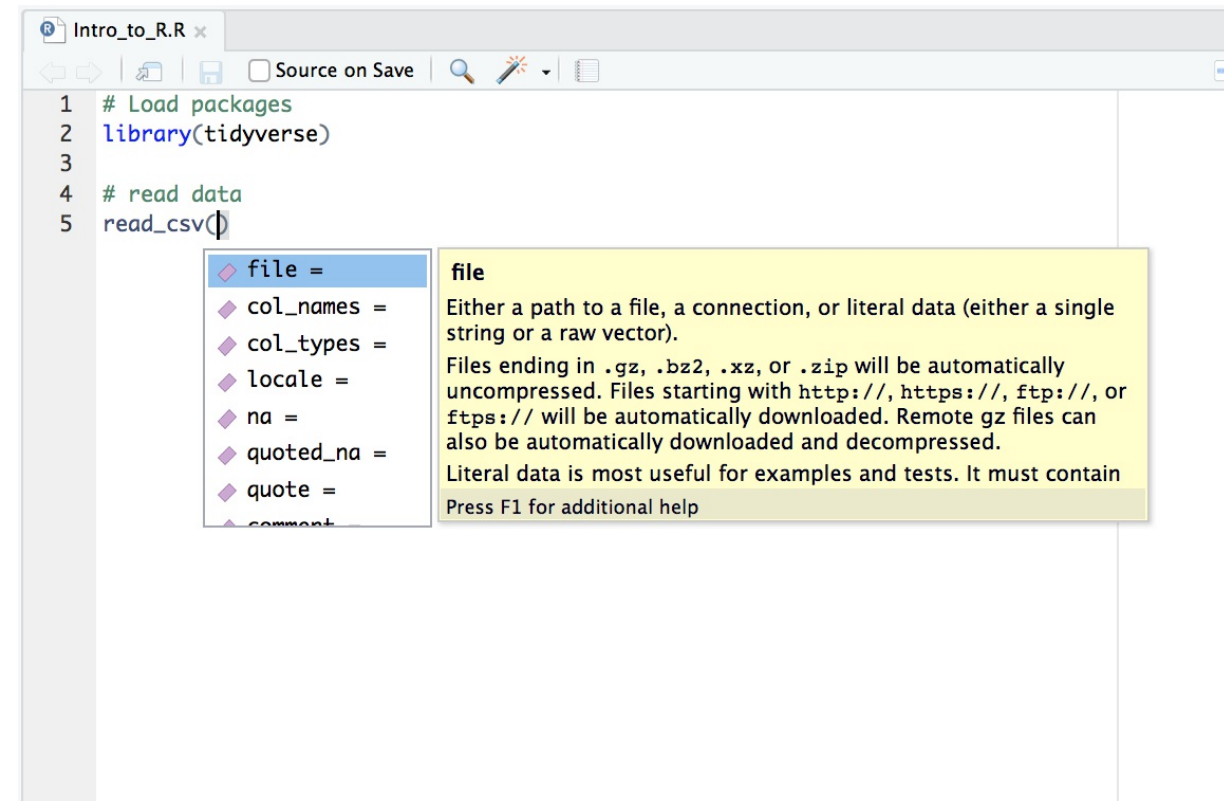
Essentials: The 2⁴ Lessons of the R Bootcamp

1. Everything is an object
2. Use <- to create/change objects
3. Objects have classes
4. Name objects using _
5. Everything happens through functions
6. Functions have (default) arguments
7. Functions expect certain object classes
8. View help files using ?
9. Study errors and warnings
10. Data is stored in data frames
11. Select variables (vectors) using \$
12. Use RStudio and projects
13. Use editor and shortcuts
14. First load packages and data
15. **Use auto-complete**
16. Comment and format for readability



Essentials: The 2⁴ Lessons of the R Bootcamp

1. Everything is an object
2. Use <- to create/change objects
3. Objects have classes
4. Name objects using _
5. Everything happens through functions
6. Functions have (default) arguments
7. Functions expect certain object classes
8. View help files using ?
9. Study errors and warnings
10. Data is stored in data frames
11. Select variables (vectors) using \$
12. Use RStudio and projects
13. Use editor and shortcuts
14. First load packages and data
15. **Use auto-complete**
16. Comment and format for readability



The screenshot shows the RStudio interface with a script editor titled 'Intro_to_R.R'. The script contains the following code:

```
1 # Load packages
2 library(tidyverse)
3
4 # read data
5 read_csv()
```

An auto-complete tooltip is displayed for the `read_csv()` function. It lists the following arguments:

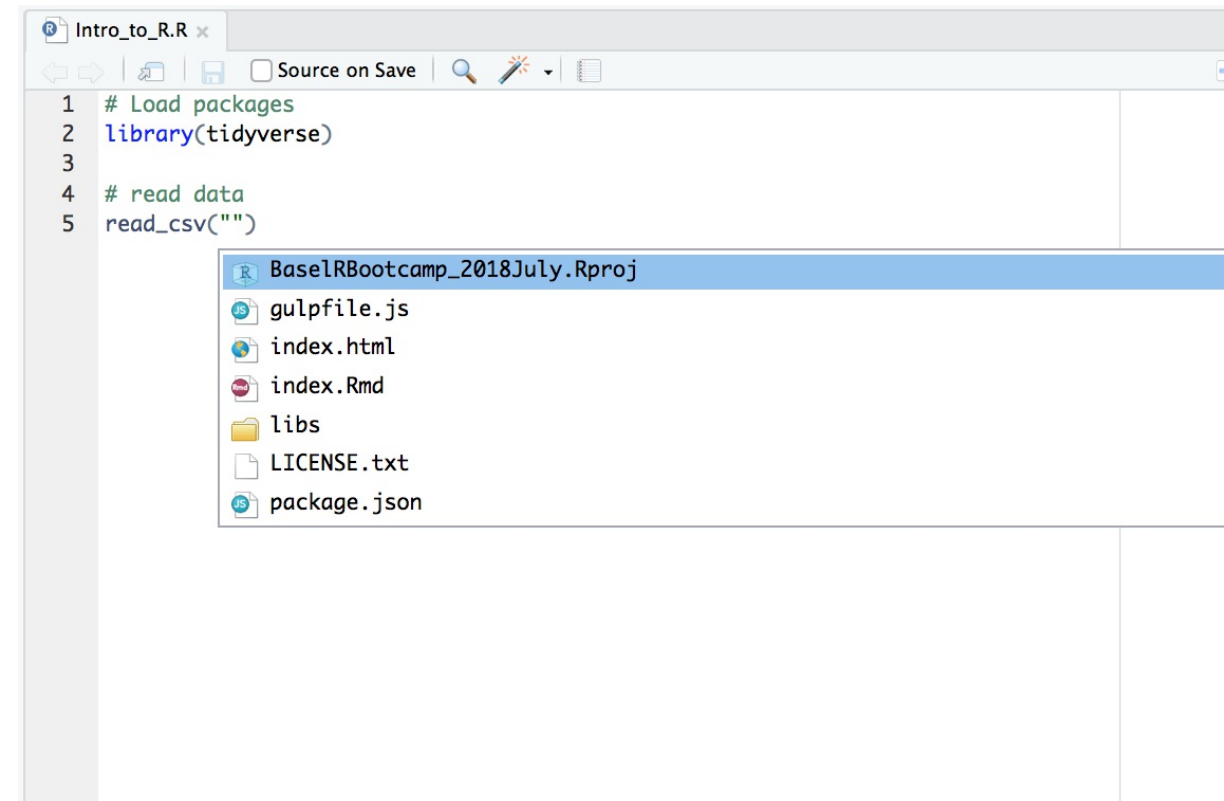
- file =
- col_names =
- col_types =
- locale =
- na =
- quoted_na =
- quote =
- comment =

The tooltip also includes a description of the `file` argument:

file
Either a path to a file, a connection, or literal data (either a single string or a raw vector).
Files ending in .gz, .bz2, .xz, or .zip will be automatically uncompressed. Files starting with http://, https://, ftp://, or ftps:// will be automatically downloaded. Remote gz files can also be automatically downloaded and decompressed.
Literal data is most useful for examples and tests. It must contain
Press F1 for additional help

Essentials: The 2⁴ Lessons of the R Bootcamp

1. Everything is an object
2. Use `<-` to create/change objects
3. Objects have classes
4. Name objects using `_`
5. Everything happens through functions
6. Functions have (default) arguments
7. Functions expect certain object classes
8. View help files using `?`
9. Study errors and warnings
10. Data is stored in data frames
11. Select variables (vectors) using `$`
12. Use RStudio and projects
13. Use editor and shortcuts
14. First load packages and data
15. **Use auto-complete**
16. Comment and format for readability



Essentials: The 2⁴ Lessons of the R Bootcamp

1. Everything is an object
2. Use <- to create/change objects
3. Objects have classes
4. Name objects using _
5. Everything happens through functions
6. Functions have (default) arguments
7. Functions expect certain object classes
8. View help files using ?
9. Study errors and warnings
10. Data is stored in data frames
11. Select variables (vectors) using \$
12. Use RStudio and projects
13. Use editor and shortcuts
14. First load packages and data
15. Use auto-complete
16. **Comment and format for readability**

Bad

```
mean(subset((tibble(c('a','b'),runif(1000,0,1))),c..a....b..=='a')[,'runif.1000..0..1.'])
```

Good

```
# create my data.frame
my_data <- tibble('group' = c('a','b'),
                  'value' = runif(1000,0,1))

# subset data to group a
# and compute average value
my_data %>%
  filter(group == 'a') %>%
  summarize(mean(value))
```

Essentials: The 2⁴ Lessons of the R Bootcamp

1. Everything is an object
2. Use `<-` to create/change objects
3. Objects have classes
4. Name objects using `_`
5. Everything happens through functions
6. Functions have (default) arguments
7. Functions expect certain object classes
8. View help files using `?`
9. Study errors and warnings
10. Data is stored in data frames
11. Select variables (vectors) using `$`
12. Use RStudio and projects
13. Use editor and shortcuts
14. First load packages and data
15. Use auto-complete
16. **Comment and format for readability**

Short style guide

```
# Choose appropriate names
analyze_baselers.R
trial_id

# Leave spaces around operators
var_rt <- var(rt, na.rm = TRUE)

# indent code
if (var_rt < 2){
  print('small variance')
} else {
  print('large variance')
}

# Create sections using
# Data wrangling section -----
```

See also style.tidyverse.org/

Essentials: Lesson 17

Struggle,
ask for help,
struggle,
...



Downloads

Data sets

Interactive session

Open up Rstudio...