Intro to R

R for Data Science

Basel R Bootcamp

February 2019

R is a 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?

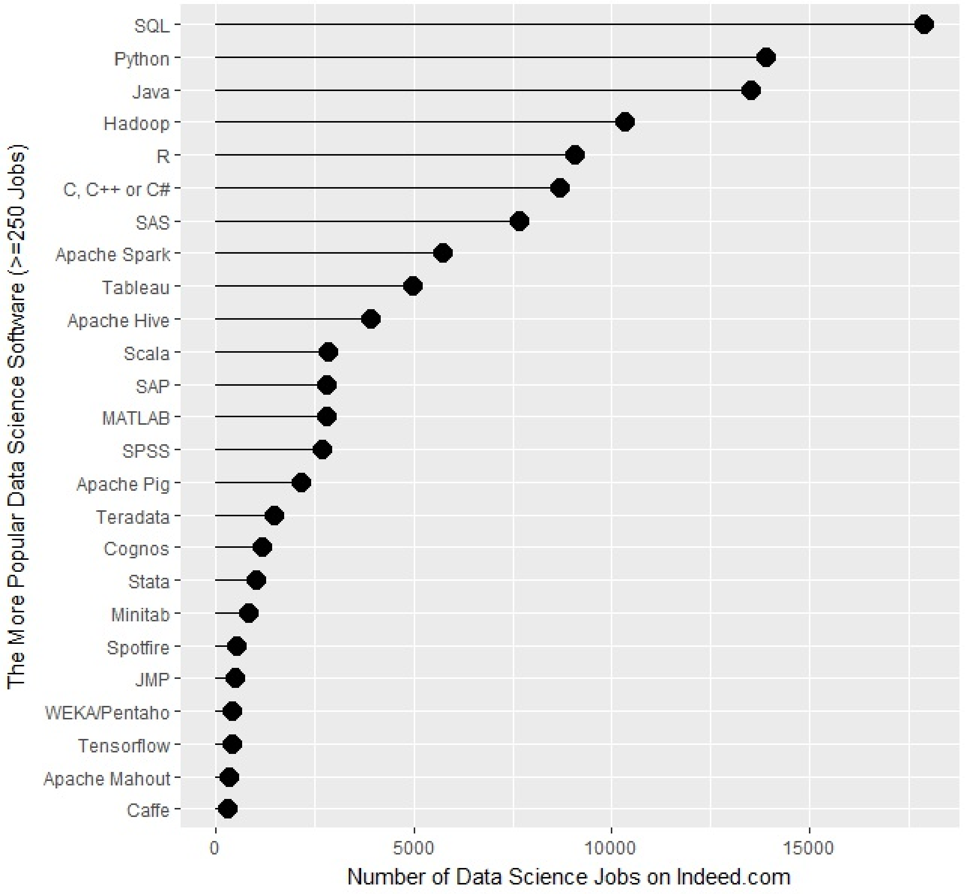
R steadily **grows in popularity**.

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.

In many sciences it becomes the de factor **lingua franca** for statistics.

Image source: http://r4stats.com/blog/



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`, `ggplot2`, etc.)
6. RStudio
7. **Productivity** options: Latex, Markdown, GitHub

Con

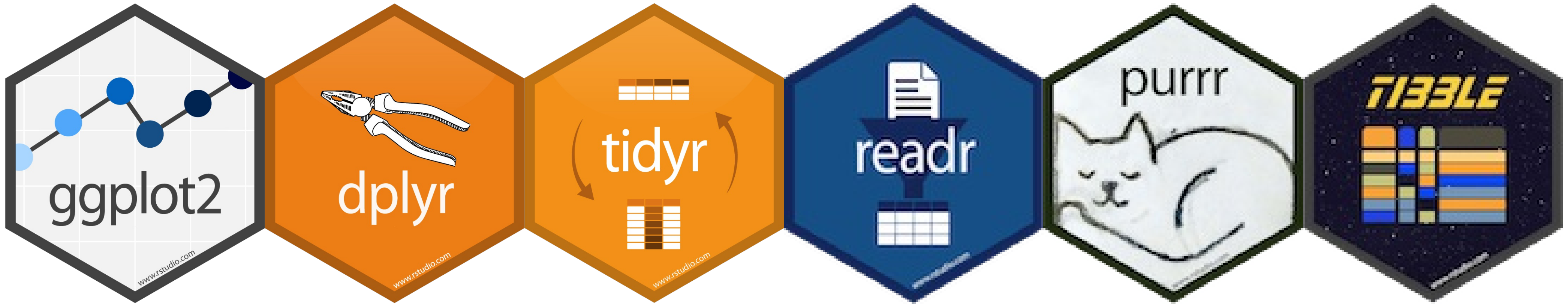
Some consider it slow and convuluted, but...

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:

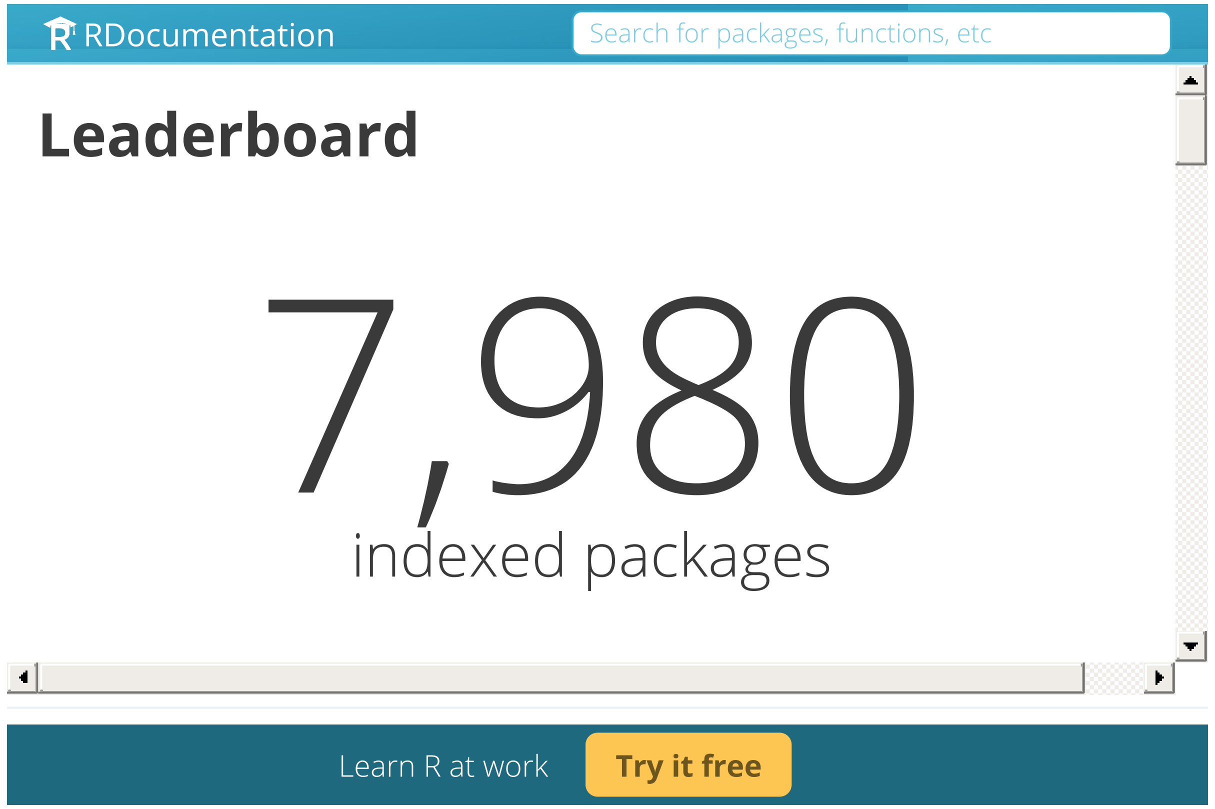
1. `ggplot2` -- creating graphics.
2. `dplyr` -- data manipulation.
3. `tidyr` -- tidying data.
4. `readr` -- read wild data.
5. `purrr` -- functional programming.
6. `tibble` -- modern data frame.



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.
```



Leaderboard

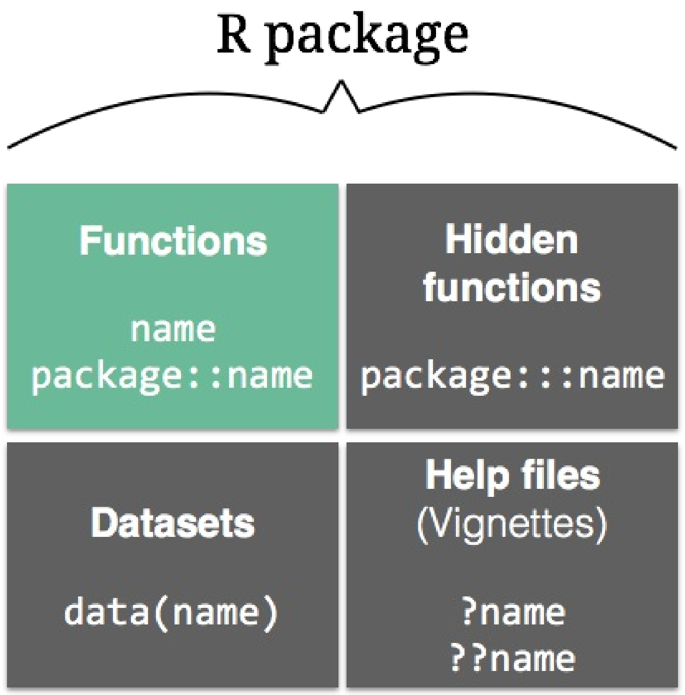
7,980
indexed packages

www.rdocumentation.org/trends

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.
```



R package

Functions

name
package::name

Hidden functions

package:::name

Datasets

data(name)

Help files
(Vignettes)

?name
??name

12 basic R lessons

Essentials: 12 basic R lessons

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. Use RStudio and projects
11. Use editor and auto-complete
12. Comment and format for readability

Essentials: 12 basic R lessons

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. Use RStudio and projects
11. Use editor and auto-complete
12. 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: 12 basic R lessons

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. Use RStudio and projects
11. Use editor and auto-complete
12. Comment and format for readability

```
# What is the class of some_name
class(some_name)
```

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

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

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

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

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

Essentials: 12 basic R lessons

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. Use RStudio and projects
11. Use editor and auto-complete
12. Comment and format for readability

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

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

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

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

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

```r
# function class()
class(x = some_name)
```

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

Essentials: 12 basic R lessons

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. Use RStudio and projects
11. Use editor and auto-complete
12. Comment and format for readability

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

```
## Error in mean.default(): argument "x" is missing, with n
```

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

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

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

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

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

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

Essentials: 12 basic R lessons

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. Use RStudio and projects
11. Use editor and auto-complete
12. Comment and format for readability

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

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

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

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

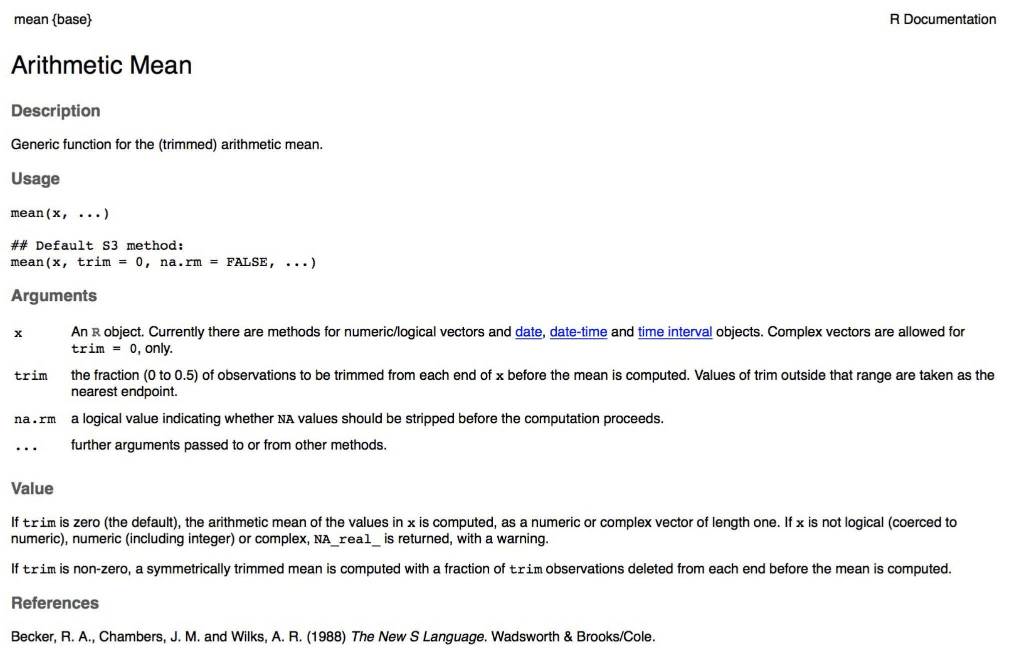
```r
# classes relevant for all arg's
mean(c(1, 2, 3), na.rm = "test")
```

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

Essentials: 12 basic R lessons

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. Use RStudio and projects
11. Use editor and auto-complete
12. 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.

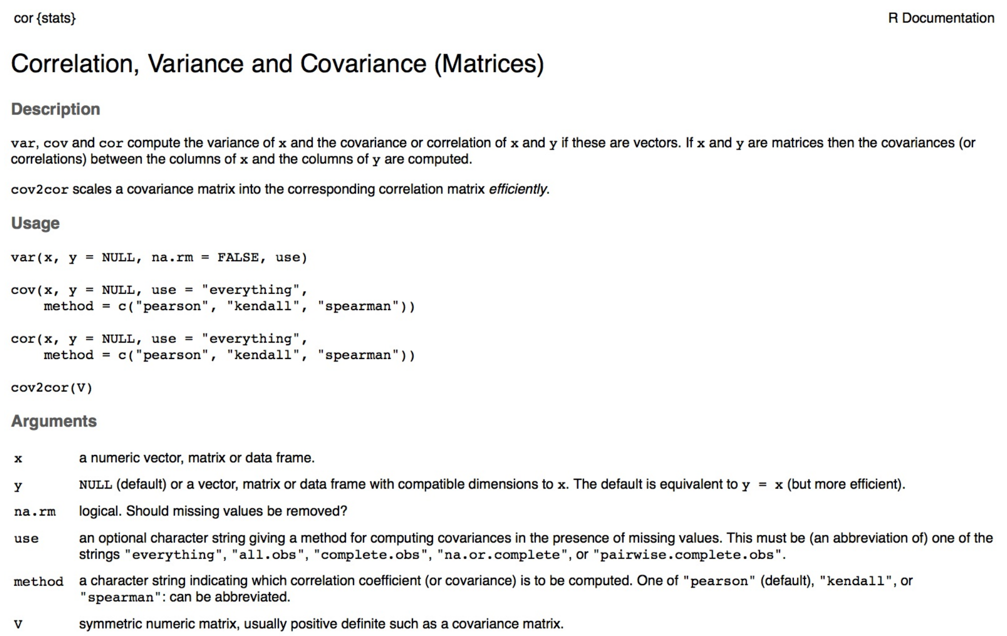
References

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

Essentials: 12 basic R lessons

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. Use RStudio and projects
11. Use editor and auto-complete
12. 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: 12 basic R lessons

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. Use RStudio and projects
11. Use editor and auto-complete
12. Comment and format for readability

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

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

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

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

```r
# error - fix
lenth(x = 1)
```

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

Essentials: 12 basic R lessons

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. Use RStudio and projects
11. Use editor and auto-complete
12. Comment and format for readability

```
lenth(x = 1)
```

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

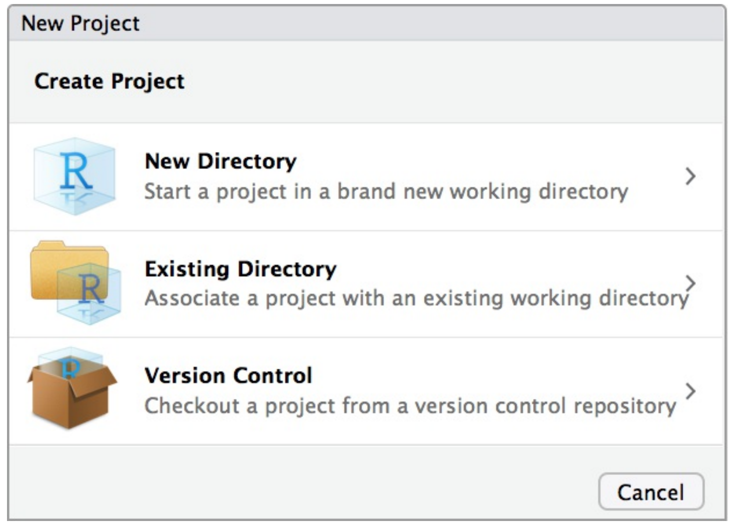
Error	Description
`'could not find function'`	Typo or package not loaded
`'error in eval'`	An object is used in function that does not exist.
`'cannot open()'`	Typo or missing path.
`'no applicable method'`	Function inapplicable for type
package errors	Unable to install, compile, or load package.

Essentials: 12 basic R lessons

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. **Use RStudio and projects**
11. Use editor and auto-complete
12. Comment and format for readability

Projects help ... USE projects!

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



New Directory
Start a project in a brand new working directory

Existing Directory
Associate a project with an existing working directory

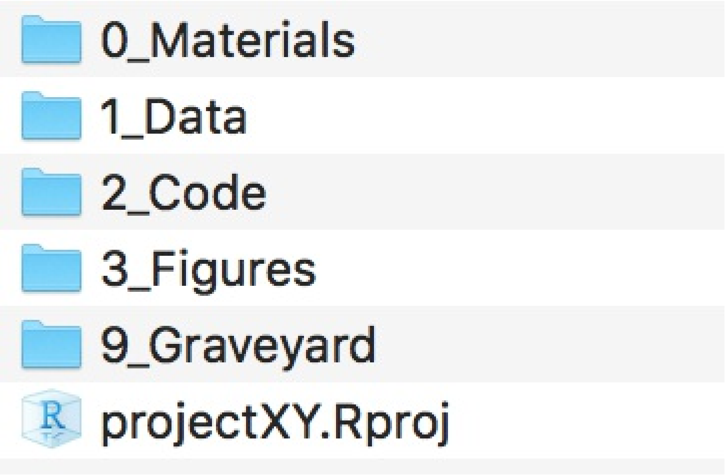
Version Control
Checkout a project from a version control repository

Essentials: 12 basic R lessons

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. **Use RStudio and projects**
11. Use editor and auto-complete
12. Comment and format for readability

Folder structure

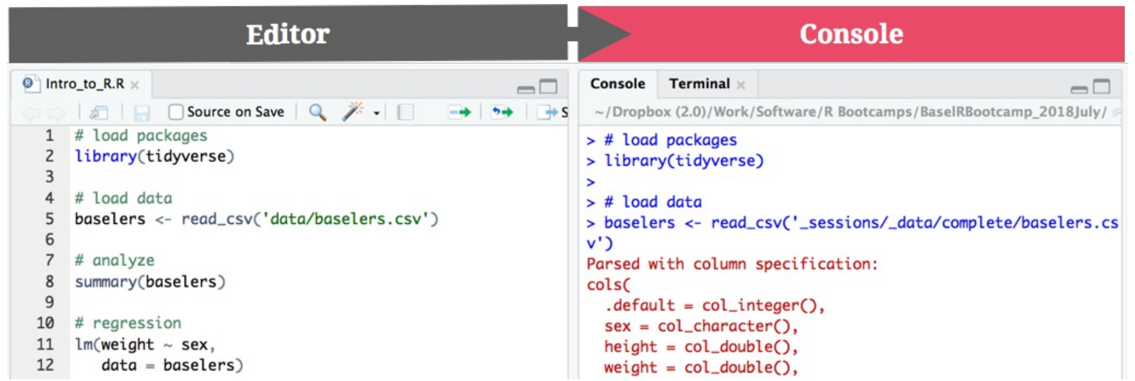
Complement projects by a **folder structure** appropriate for your project.



0_Materials
1_Data
2_Code
3_Figures
9_Graveyard
projectXY.Rproj

Essentials: 12 basic R lessons

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. Use RStudio and projects
11. **Use editor and auto-complete**
12. Comment and format for readability


Editor
Console
Intro_to_R.R
Source on Save
Console
Terminal
~/Dropbox (2.0)/Work/Software/R Bootcamps/BaselRBootcamp_2018July/

```r
# load packages
library(tidyverse)

# load data
baselers <- read_csv('data/baselers.csv')

# analyze
summary(baselers)

# regression
lm(weight ~ sex,
   data = baselers)
```

```
> # load packages
> library(tidyverse)
>
> # load data
> baselers <- read_csv('_sessions/_data/complete/baselers.cs
v')
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: 12 basic R lessons

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. Use RStudio and projects
11. **Use editor and auto-complete**
12. Comment and format for readability

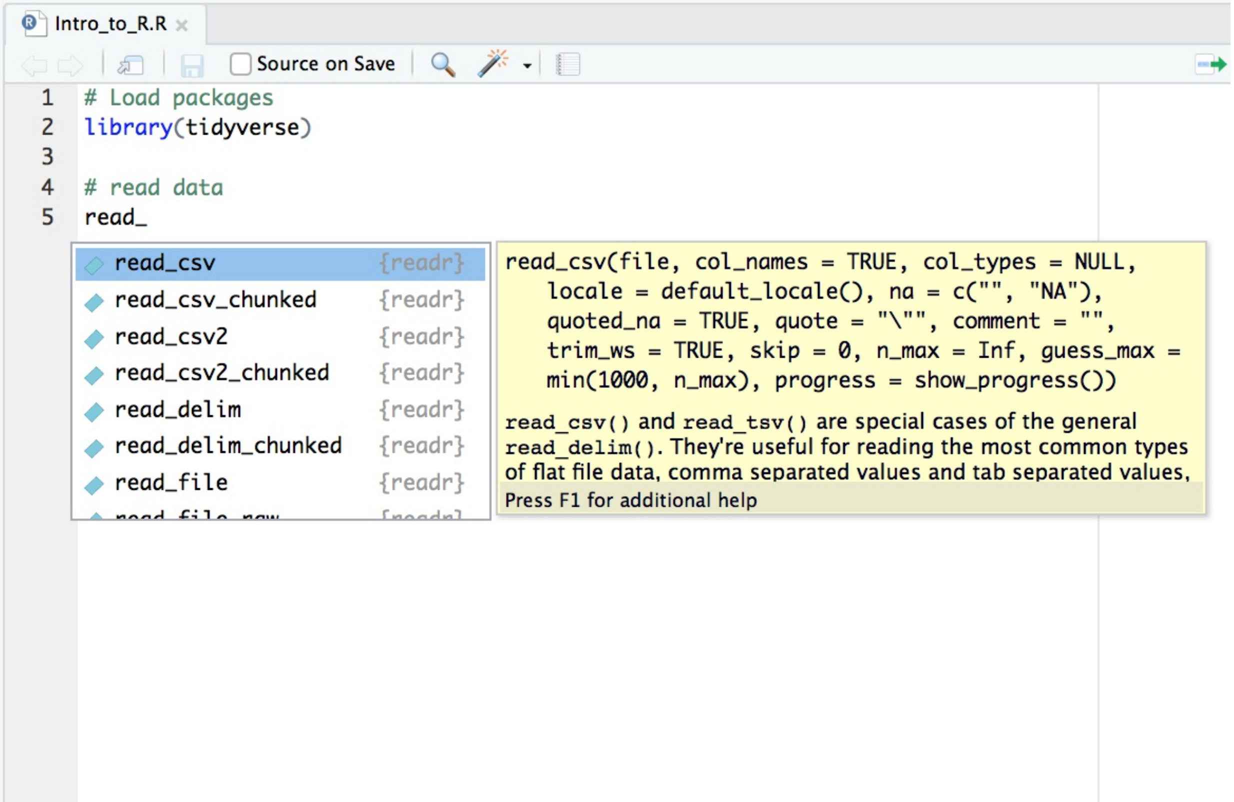
```
# import packages
library(tidyverse)
library(yarrr)
library(lme4)

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

The goal is...

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

Essentials: 12 basic R lessons

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. Use RStudio and projects
11. **Use editor and auto-complete**
12. Comment and format for readability



```
# Load packages
library(tidyverse)

# read data
read_
```

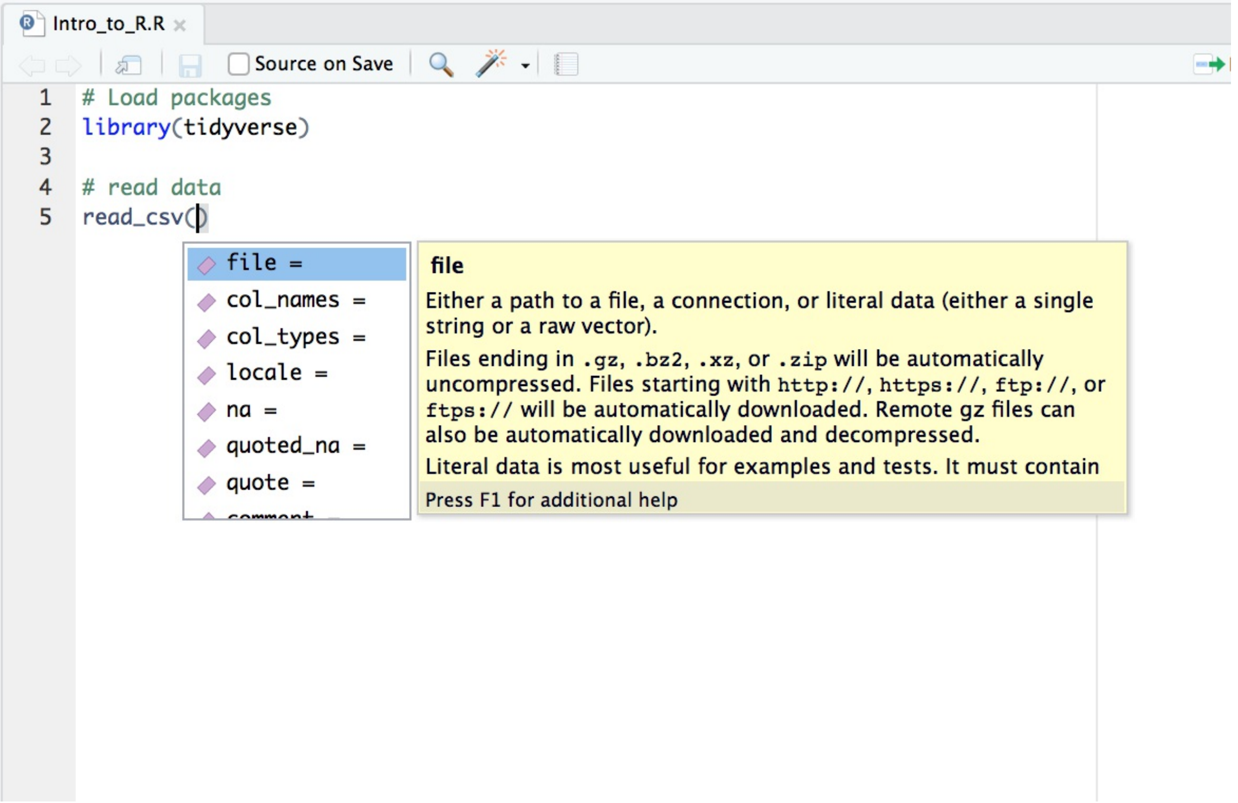
```
read_csv(file, col_names = TRUE, col_types = NULL,
    locale = default_locale(), na = c("", "NA"),
    quoted_na = TRUE, quote = "\"", comment = "",
    trim_ws = TRUE, skip = 0, n_max = Inf, guess_max =
    min(1000, n_max), progress = show_progress())
```

`read_csv()` and `read_tsv()` are special cases of the general `read_delim()`. They're useful for reading the most common types of flat file data, comma separated values and tab separated values,

Press F1 for additional help

Essentials: 12 basic R lessons

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. Use RStudio and projects
11. **Use editor and auto-complete**
12. Comment and format for readability



```
# Load packages
library(tidyverse)

# read data
read_csv()
```

file

Either a path to a file, a connection, or literal data (either a single string or a raw vector).

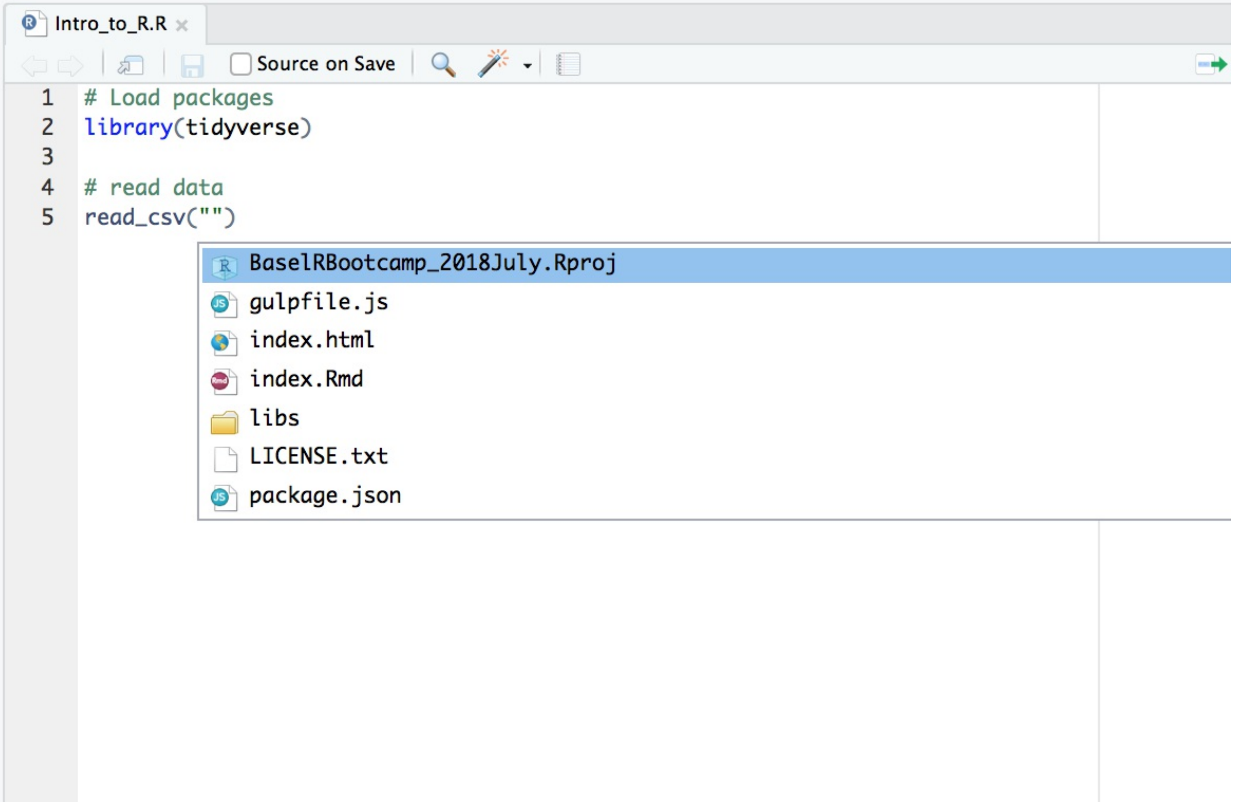
Files ending in `.gz`, `.bz2`, `.xz`, or `.zip` will be automatically uncompressed. Files starting with `http://`, `https://`, `ftp://`, or `ftps://` will be automatically downloaded. Remote gz files can also be automatically downloaded and decompressed.

Literal data is most useful for examples and tests. It must contain

Press F1 for additional help

Essentials: 12 basic R lessons

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. Use RStudio and projects
11. Use editor and auto-complete
12. Comment and format for readability



```
# Load packages
library(tidyverse)

# read data
read_csv("")
```

Essentials: 12 basic R lessons

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. Use RStudio and projects
11. Use editor and auto-complete
12. **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: 12 basic R lessons

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. Use RStudio and projects
11. Use editor and auto-complete
12. **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')
}

# Data wrangling section ---------
```

See also **style.tidyverse.org/**

Downloads

Interactive

Open RStudio