R for Stats

Statistics with R

Basel R Bootcamp

🕓 🏠 ✉ in

April 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

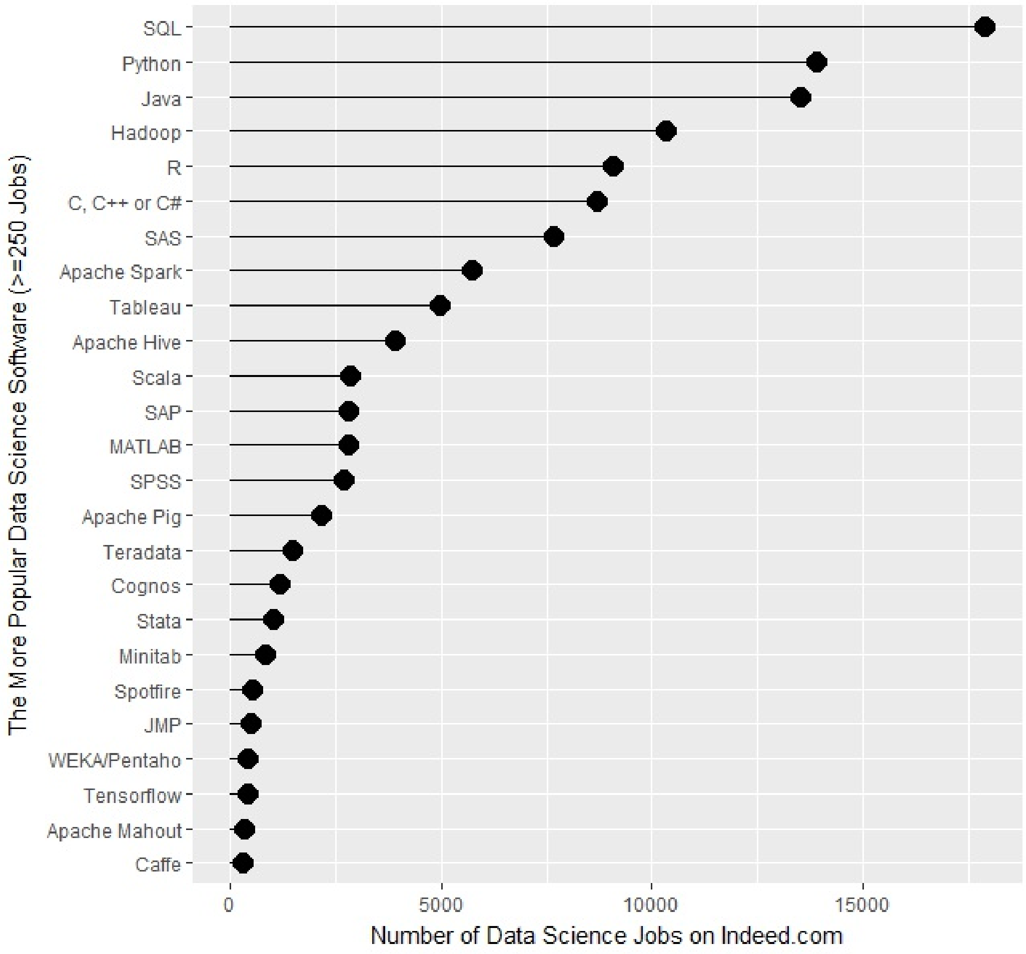
```
data <- read.table(filepath)
variables <- data %>% select(group, variable)
analysis <- lm(variable ~ group, data = variables)
summary(analysis)
```

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.



from 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

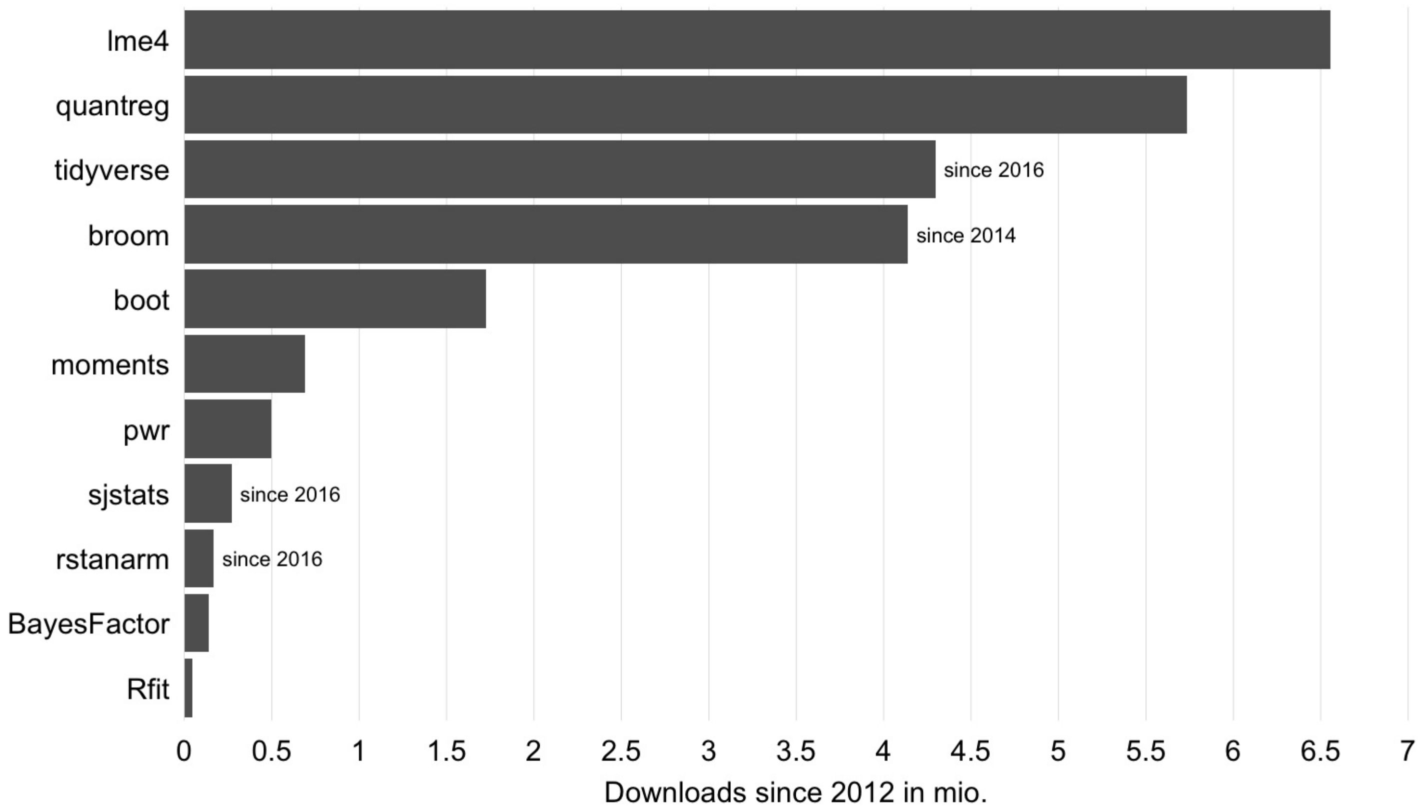
Con

It's slow, 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.

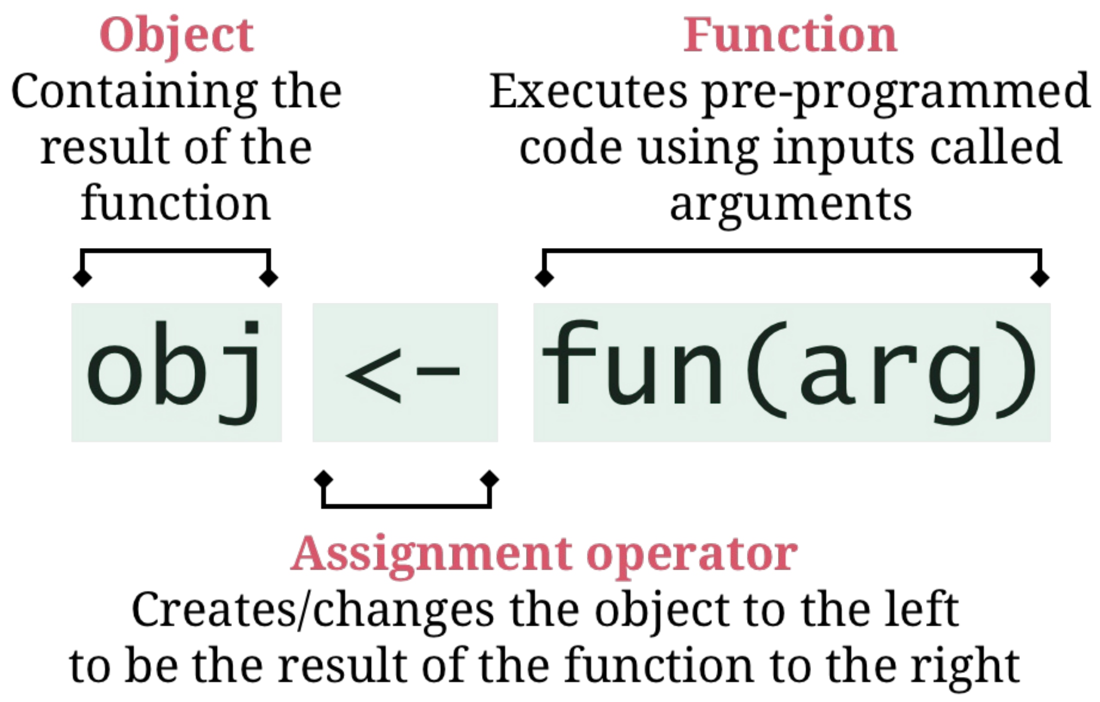
R is great for Statistics

…because of high-performance R packages (extensions) **downloaded and used millions of times**.



Essentials: 11 basic R lessons

1. **Everything is an object**
2. **<- creates/changes objects**
3. Everything happens through functions
4. Functions have (default) arguments
5. Functions live in packages
6. Find help with ?
7. Data lives in data frames
8. 3 data types + factors
9. `formula` and `data` specify a model
10. Use RStudio and projects
11. Use editor, shortcuts, auto-complete



Object
Containing the result of the function

Function
Executes pre-programmed code using inputs called arguments

```
obj <- fun(arg)
```

Assignment operator
Creates/changes the object to the left
to be the result of the function to the right

Essentials: 11 basic R lessons

1. **Everything is an object**
2. **`<-` creates/changes objects**
3. Everything happens through functions
4. Functions have (default) arguments
5. Functions live in packages
6. Find help with ?
7. Data lives in data frames
8. 3 data types + factors
9. `formula` and `data` specify a model
10. Use RStudio and projects
11. Use editor, shortcuts, auto-complete

```r
# an object called one_two_three
one_two_three <- c(1, 2, 3)

# print object
one_two_three
```

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

```r
# add 100 to the object's numbers (without <- )
one_two_three + 100
```

```
## [1] 101 102 103
```

```r
# print object (no <-, no change!)
one_two_three
```

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

Essentials: 11 basic R lessons

1. **Everything is an object**
2. **<- creates/changes objects**
3. Everything happens through functions
4. Functions have (default) arguments
5. Functions live in packages
6. Find help with ?
7. Data lives in data frames
8. 3 data types + factors
9. `formula` and `data` specify a model
10. Use RStudio and projects
11. Use editor, shortcuts, auto-complete

```
# print object
one_two_three
```

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

```
# make change permanent (with <- )
one_two_three <- one_two_three + 100

# print object (it has changed!)
one_two_three
```

```
## [1] 101 102 103
```

Essentials: 11 basic R lessons

1. Everything is an object
2. `<-` creates/changes objects
3. **Everything happens through functions**
4. Functions have (default) arguments
5. Functions live in packages
6. Find help with ?
7. Data lives in data frames
8. 3 data types + factors
9. `formula` and `data` specify a model
10. Use RStudio and projects
11. Use editor, shortcuts, auto-complete

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

# function `+`()
one_two_three + 100
```

```
## [1] 101 102 103
```

```r
# function print()
one_two_three
```

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

```r
# function mean()
mean(x = one_two_three)
```

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

Essentials: 11 basic R lessons

1. Everything is an object
2. `<-` creates/changes objects
3. Everything happens through functions
4. **Functions have (default) arguments**
5. Functions live in packages
6. Find help with `?`
7. Data lives in data frames
8. 3 data types + factors
9. `formula` and `data` specify a model
10. Use RStudio and projects
11. Use editor, shortcuts, auto-complete

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

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

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

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

```r
# assume a missing value (NA)
mean(x = c(1, 2, 3, NA))
```

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

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

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

Essentials: 11 basic R lessons

1. Everything is an object
2. `<-` creates/changes objects
3. Everything happens through functions
4. **Functions have (default) arguments**
5. Functions live in packages
6. Find help with ?
7. Data lives in data frames
8. 3 data types + factors
9. `formula` and `data` specify a model
10. Use RStudio and projects
11. Use editor, shortcuts, auto-complete

```
# mean with pipe %>%
c(1, 2, 3) %>% mean()
```

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

```
# mean with pipe %>% and NA
c(1, 2, 3, NA) %>% mean()
```

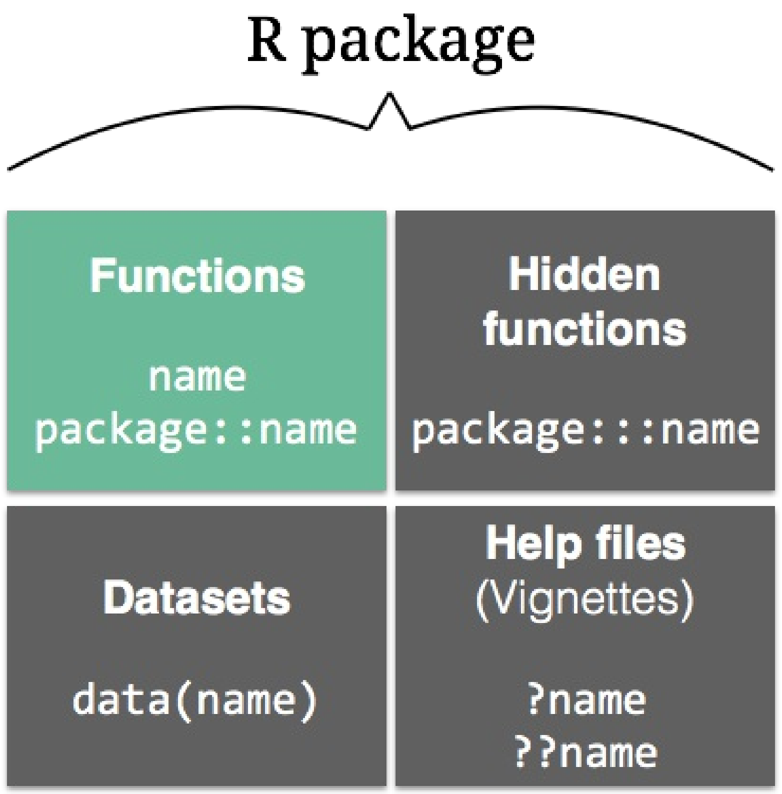
```
## [1] NA
```

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

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

Essentials: 11 basic R lessons

1. Everything is an object
2. `<-` creates/changes objects
3. Everything happens through functions
4. Functions have (default) arguments
5. **Functions live in packages**
6. Find help with `?`
7. Data lives in data frames
8. 3 data types + factors
9. `formula` and `data` specify a model
10. Use RStudio and projects
11. Use editor, shortcuts, auto-complete



R package

Functions

`name`
`package::name`

Hidden functions

`package:::name`

Datasets

`data(name)`

Help files
(Vignettes)

`?name`
`??name`

Essentials: 11 basic R lessons

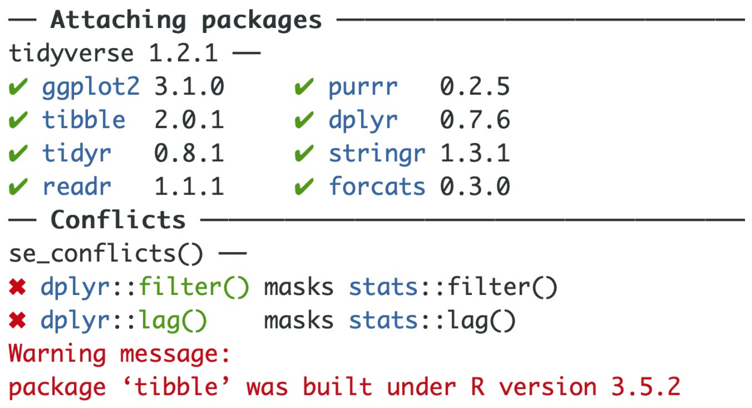
1. Everything is an object
2. `<-` creates/changes objects
3. Everything happens through functions
4. Functions have (default) arguments
5. **Functions live in packages**
6. Find help with ?
7. Data lives in data frames
8. 3 data types + factors
9. `formula` and `data` specify a model
10. Use RStudio and projects
11. Use editor, shortcuts, auto-complete

Install new packages with `install.packages()`

```
# install package: Only do this once!
install.packages("tidyverse")
```

Load existing packages with `library()`

```
# load package: EVERY TIME you write code
library(tidyverse)
```

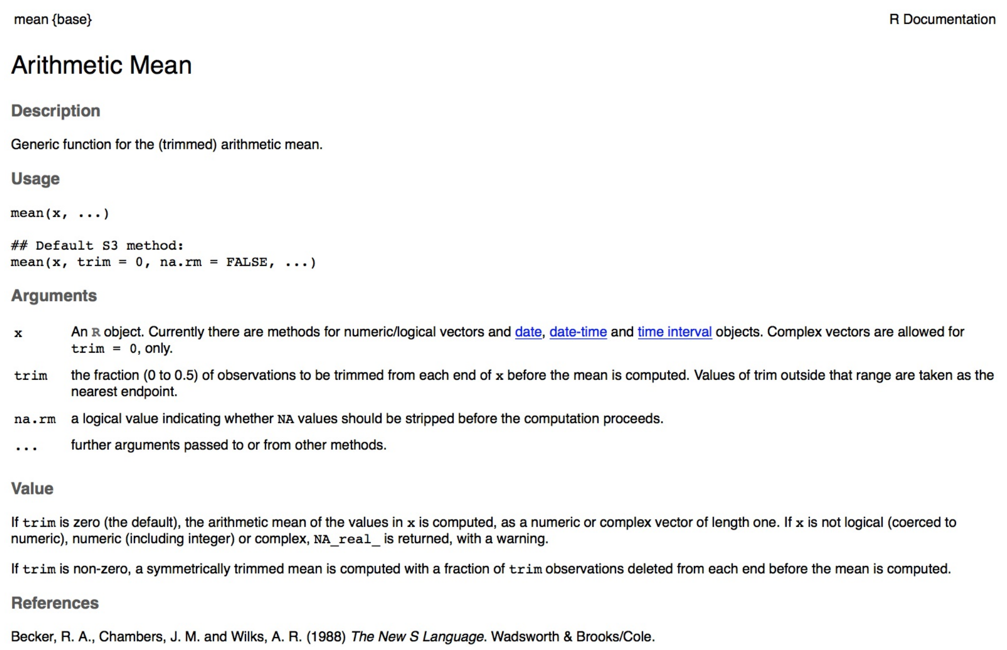


```
— Attaching packages ———————————————————————————
tidyverse 1.2.1 —
✔ ggplot2 3.1.0     ✔ purrr   0.2.5
✔ tibble  2.0.1     ✔ dplyr   0.7.6
✔ tidyr   0.8.1     ✔ stringr 1.3.1
✔ readr   1.1.1     ✔ forcats 0.3.0
— Conflicts ————————————————————————————————————
se_conflicts() —
✖ dplyr::filter() masks stats::filter()
✖ dplyr::lag()    masks stats::lag()
Warning message:
package ‘tibble’ was built under R version 3.5.2
```

Essentials: 11 basic R lessons

1. Everything is an object
2. `<-` creates/changes objects
3. Everything happens through functions
4. Functions have (default) arguments
5. Functions live in packages
6. **Find help with `?`**
7. Data lives in data frames
8. 3 data types + factors
9. `formula` and `data` specify a model
10. Use RStudio and projects
11. Use editor, shortcuts, auto-complete

```
?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.

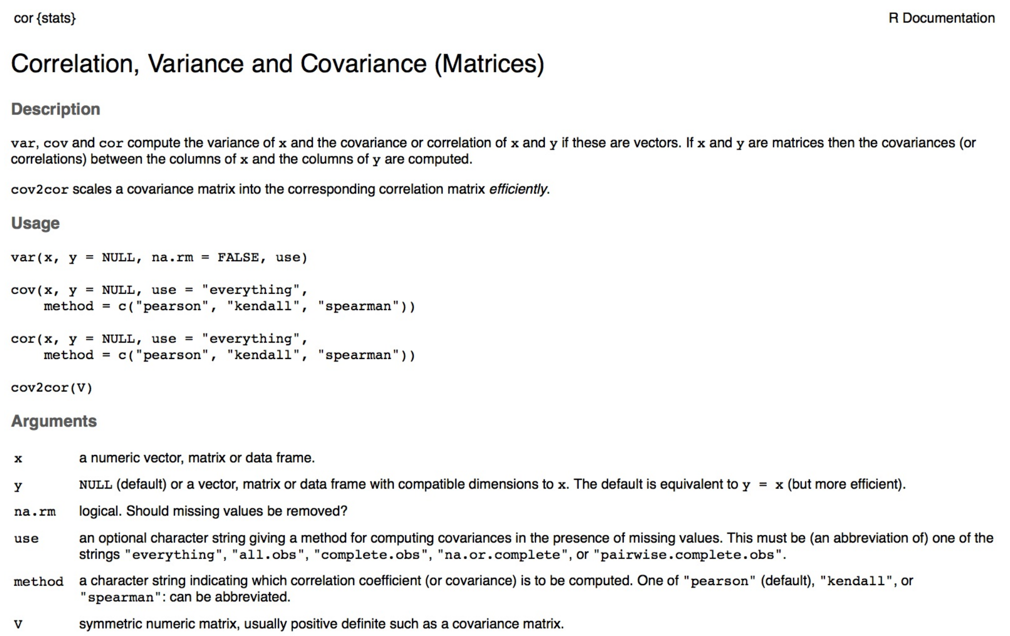
References

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

Essentials: 11 basic R lessons

1. Everything is an object
2. `<-` creates/changes objects
3. Everything happens through functions
4. Functions have (default) arguments
5. Functions live in packages
6. **Find help with `?`**
7. Data lives in data frames
8. 3 data types + factors
9. `formula` and `data` specify a model
10. Use RStudio and projects
11. Use editor, shortcuts, auto-complete

```
?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

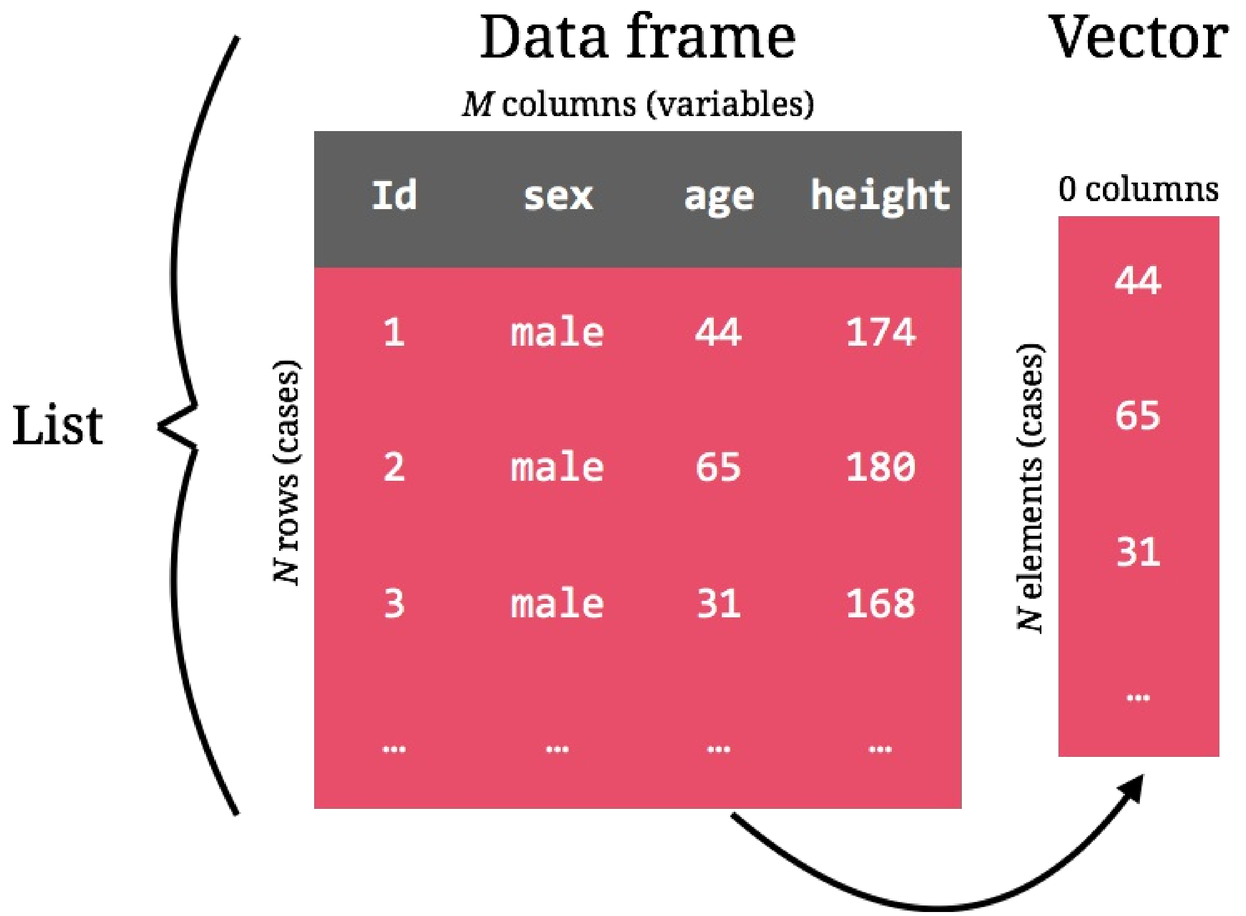
```
var(x, y = NULL, na.rm = FALSE, use)

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: 11 basic R lessons

1. Everything is an object
2. `<-` creates/changes objects
3. Everything happens through functions
4. Functions have (default) arguments
5. Functions live in packages
6. Find help with `?`
7. **Data lives in data frames**
8. 3 data types + factors
9. `formula` and `data` specify a model
10. Use RStudio and projects
11. Use editor, shortcuts, auto-complete

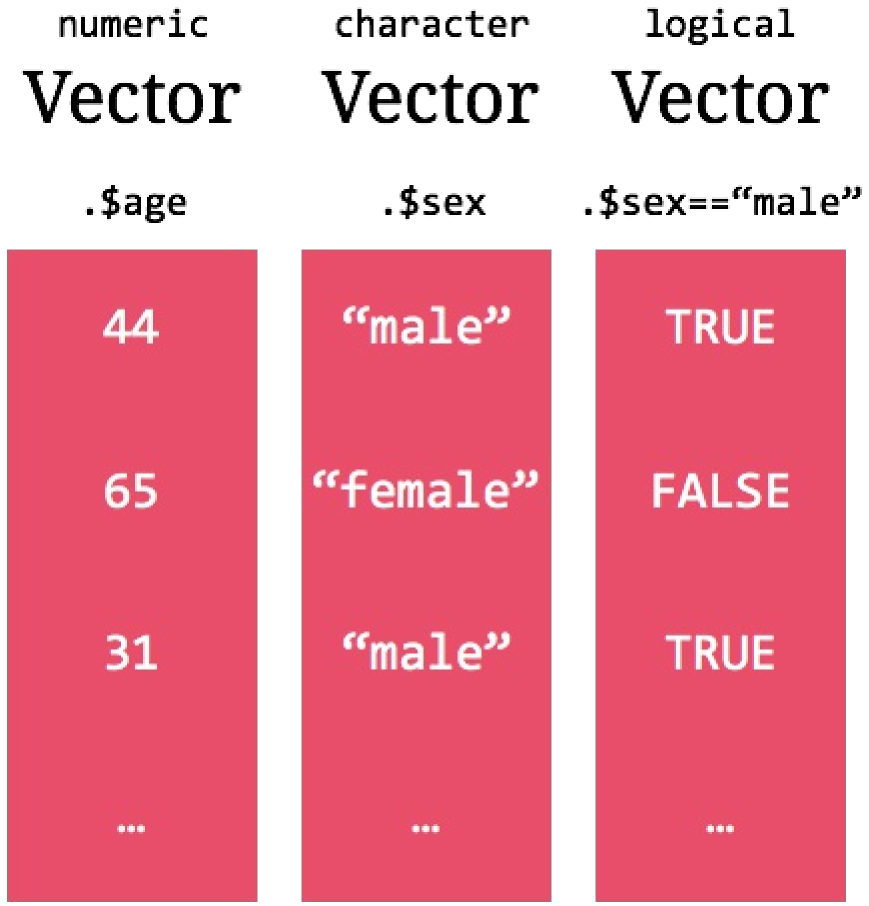


Id	sex	age	height
1	male	44	174
2	male	65	180
3	male	31	168
…	…	…	…

44
65
31
…

Essentials: 11 basic R lessons

1. Everything is an object
2. `<-` creates/changes objects
3. Everything happens through functions
4. Functions have (default) arguments
5. Functions live in packages
6. Find help with `?`
7. Data lives in data frames
8. **3 data types + factors**
9. `formula` and `data` specify a model
10. Use RStudio and projects
11. Use editor, shortcuts, auto-complete



numeric Vector .$age	character Vector .$sex	logical Vector .$sex==“male”
44	“male”	TRUE
65	“female”	FALSE
31	“male”	TRUE
…	…	…

Essentials: 11 basic R lessons

1. Everything is an object
2. `<-` creates/changes objects
3. Everything happens through functions
4. Functions have (default) arguments
5. Functions live in packages
6. Find help with ?
7. Data lives in data frames
8. **3 data types + factors**
9. `formula` and `data` specify a model
10. Use RStudio and projects
11. Use editor, shortcuts, auto-complete

```r
print(baselers)
```

```
## # A tibble: 10,000 x 20
##       id sex     age height weight income
##    <dbl> <chr> <dbl>  <dbl>  <dbl>  <dbl>
##  1     1 male     44   174.  113.    6300
##  2     2 male     65   180.   75.2  10900
##  3     3 fema…    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 fema…    41   156.   67.8   7600
##  9     9 male     43   176.   69.3   8500
## 10    10 fema…    31   166.   66.3   6100
## # … with 9,990 more rows, and 14 more variables
```

Essentials: 11 basic R lessons

1. Everything is an object
2. `<-` creates/changes objects
3. Everything happens through functions
4. Functions have (default) arguments
5. Functions live in packages
6. Find help with ?
7. Data lives in data frames
8. **3 data types + factors**
9. `formula` and `data` specify a model
10. Use RStudio and projects
11. Use editor, shortcuts, auto-complete

```r
# select sex veriable using $
baselers$sex
```

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

```r
# select sex veriable using %>% select
baselers %>% select(sex) %>% pull()
```

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

```r
# Possible, but less pretty...
baselers[['sex']]
baselers[[2]]
```

Essentials: 11 basic R lessons

1. Everything is an object
2. `<-` creates/changes objects
3. Everything happens through functions
4. Functions have (default) arguments
5. Functions live in packages
6. Find help with ?
7. Data lives in data frames
8. **3 data types + factors**
9. `formula` and `data` specify a model
10. Use RStudio and projects
11. Use editor, shortcuts, auto-complete

```r
# original sex vector
baselers$sex
```

```
## [1] "male"   "male"   "female" "male"   "male"  
## [6] "male"  
##  [ reached getOption("max.print") -- omitted 9994 entrie
```

```r
as.factor(baselers$sex)
```

```
## [1] male   male   female male   male   male  
##  [ reached getOption("max.print") -- omitted 9994 entrie
## Levels: female male
```

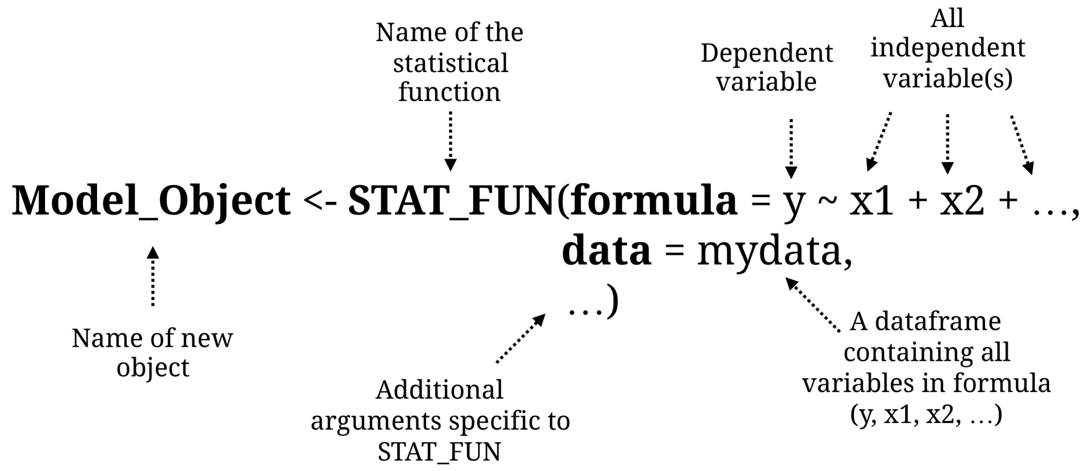
```r
as.factor(baselers$weight)
```

```
## [1] 113.4 75.2  55.5  93.8  <NA>  67.4 
##  [ reached getOption("max.print") -- omitted 9994 entrie
## 719 Levels: 37.9 38.3 39.2 39.6 40.3 ... 125.4
```

Essentials: 11 basic R lessons

1. Everything is an object
2. <- creates/changes objects
3. Everything happens through functions
4. Functions have (default) arguments
5. Functions live in packages
6. Find help with ?
7. Data lives in data frames
8. 3 data types + factors
9. **`formula` and `data` specify a model**
10. Use RStudio and projects
11. Use editor, shortcuts, auto-complete

```
# Run a regression and store result in my_lm
my_lm <- lm(formula = income ~ age + height,
             data = baselers)
```



Name of the statistical function

Dependent variable

All independent variable(s)

Model_Object <- **STAT_FUN**(**formula** = y ~ x1 + x2 + …,
data = mydata,
…)

Name of new object

Additional arguments specific to STAT_FUN

A dataframe containing all variables in formula (y, x1, x2, …)

Essentials: 11 basic R lessons

1. Everything is an object
2. `<-` creates/changes objects
3. Everything happens through functions
4. Functions have (default) arguments
5. Functions live in packages
6. Find help with `?`
7. Data lives in data frames
8. 3 data types + factors
9. **`formula` and `data` specify a model**
10. Use RStudio and projects
11. Use editor, shortcuts, auto-complete

Add variables using +

```
# Include multiple terms with +
my_lm <- lm(formula = income ~ age + height,
            data = baselers)
```

Include all variables using `formula = y ~ .`

```
# Use  y ~ . to include ALL variables
my_lm <- lm(formula = income ~ .,
            data = baselers)
```

Subtract variables using -

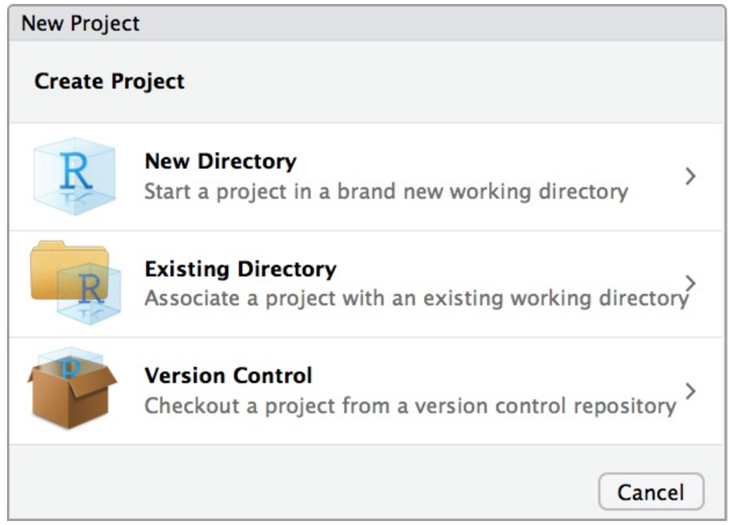
```
# Remove variables with -
my_lm <- lm(formula = income ~ . - id,
            data = baselers)
```

Essentials: 11 basic R lessons

1. Everything is an object
2. `<-` creates/changes objects
3. Everything happens through functions
4. Functions have (default) arguments
5. Functions live in packages
6. Find help with ?
7. Data lives in data frames
8. 3 data types + factors
9. `formula` and `data` specify a model
10. Use RStudio and projects
11. Use editor, shortcuts, auto-complete

Projects help...

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



New Project

Create Project

New Directory
Start a project in a brand new working directory

Existing Directory
Associate a project with an existing working directory

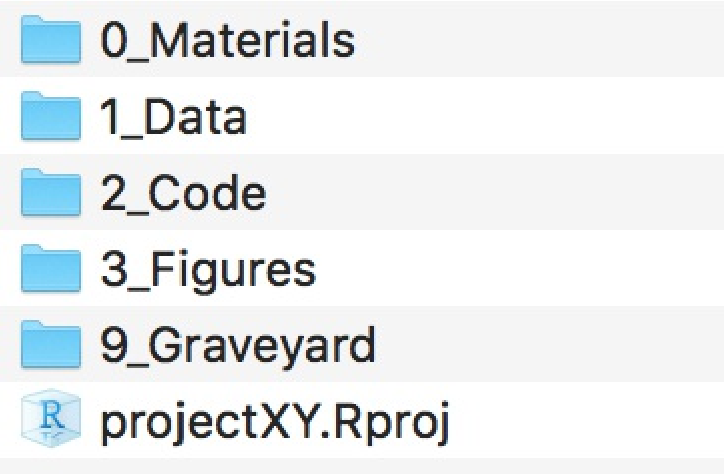
Version Control
Checkout a project from a version control repository

Essentials: 11 basic R lessons

1. Everything is an object
2. `<-` creates/changes objects
3. Everything happens through functions
4. Functions have (default) arguments
5. Functions live in packages
6. Find help with ?
7. Data lives in data frames
8. 3 data types + factors
9. `formula` and `data` specify a model
10. **Use RStudio and projects**
11. Use editor, shortcuts, auto-complete

Folder structure

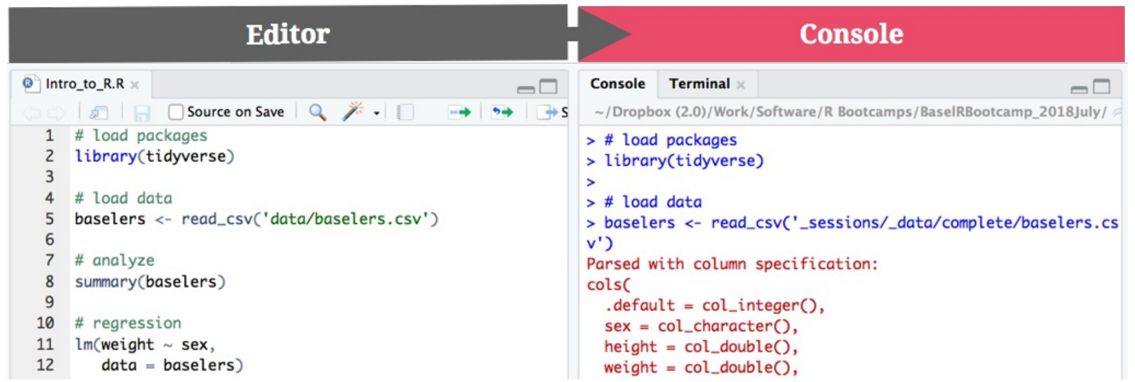
Complement projects by a **folder structure** appropriate for your project.



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

Essentials: 11 basic R lessons

1. Everything is an object
2. `<-` creates/changes objects
3. Everything happens through functions
4. Functions have (default) arguments
5. Functions live in packages
6. Find help with `?`
7. Data lives in data frames
8. 3 data types + factors
9. `formula` and `data` specify a model
10. Use RStudio and projects
11. **Use editor, shortcuts, auto-complete**


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

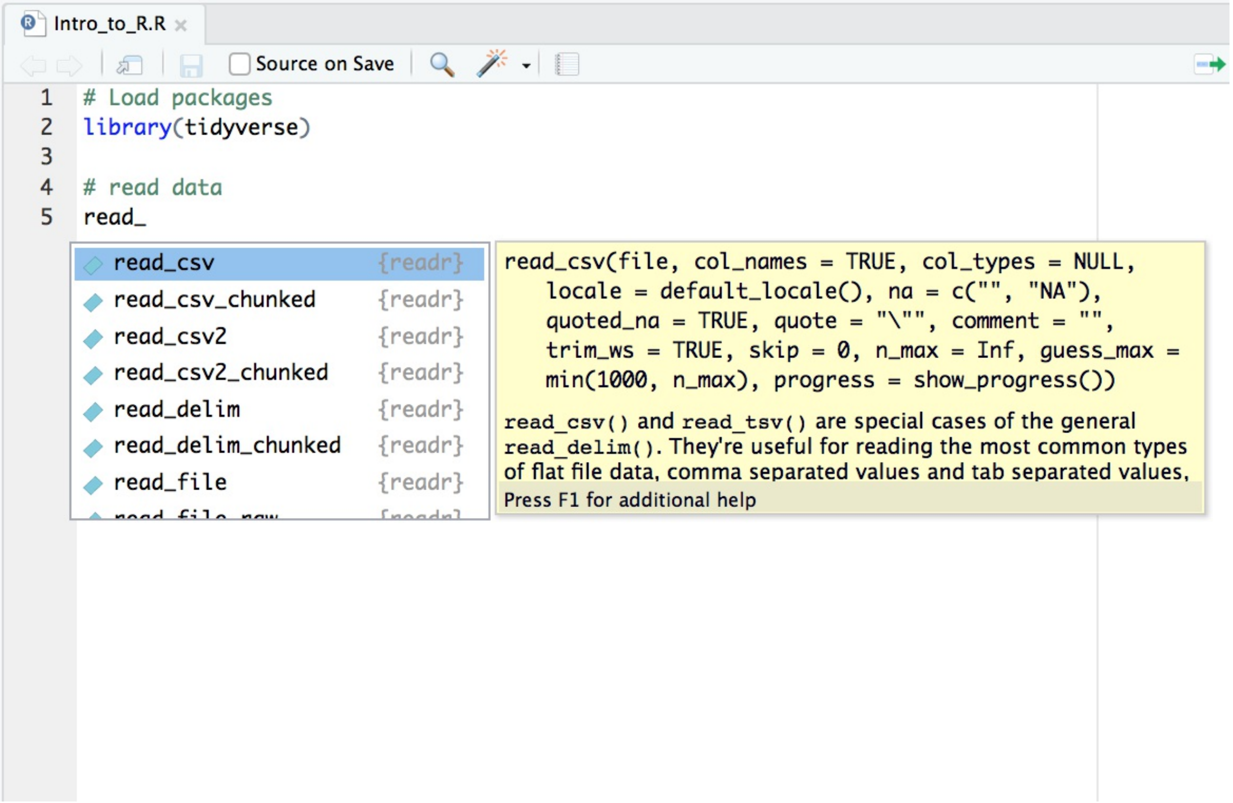
Essentials: 11 basic R lessons

1. Everything is an object
2. `<-` creates/changes objects
3. Everything happens through functions
4. Functions have (default) arguments
5. Functions live in packages
6. Find help with ?
7. Data lives in data frames
8. 3 data types + factors
9. `formula` and `data` specify a model
10. Use RStudio and projects
11. **Use editor, shortcuts, auto-complete**

```
# Load packages with library()
library(tidyverse)
library(yarrr)
library(lme4)

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

Essentials: 11 basic R lessons

1. Everything is an object
2. `<-` creates/changes objects
3. Everything happens through functions
4. Functions have (default) arguments
5. Functions live in packages
6. Find help with ?
7. Data lives in data frames
8. 3 data types + factors
9. `formula` and `data` specify a model
10. Use RStudio and projects
11. **Use editor, shortcuts, auto-complete**



```r
# Load packages
library(tidyverse)

# read data
read_
```

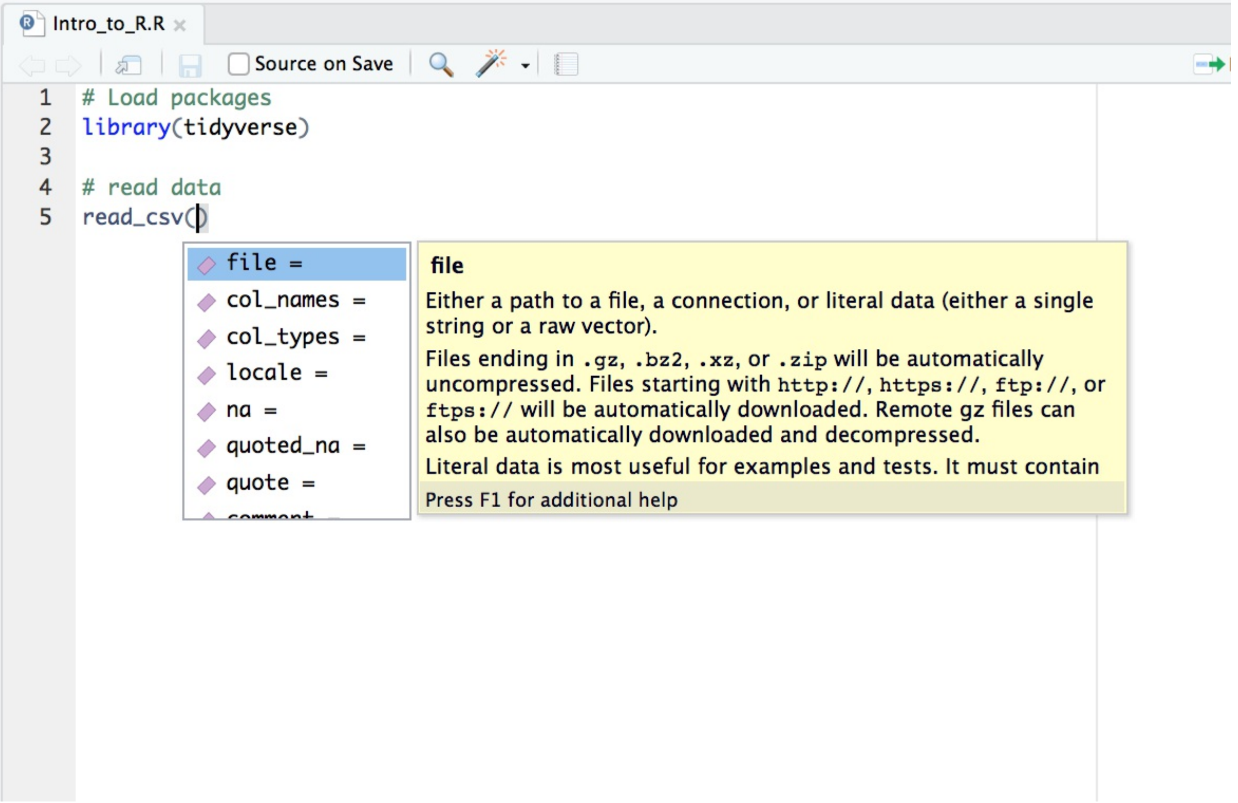
```
read_csv(file, col_names = TRUE, col_types = NULL,
    locale = default_locale(), na = c("", "NA"),
    quoted_na = TRUE, quote = "\"", comment = "",
    trim_ws = TRUE, skip = 0, n_max = Inf, guess_max =
    min(1000, n_max), progress = show_progress())
```

`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: 11 basic R lessons

1. Everything is an object
2. `<-` creates/changes objects
3. Everything happens through functions
4. Functions have (default) arguments
5. Functions live in packages
6. Find help with ?
7. Data lives in data frames
8. 3 data types + factors
9. `formula` and `data` specify a model
10. Use RStudio and projects
11. **Use editor, shortcuts, auto-complete**



```
# 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).

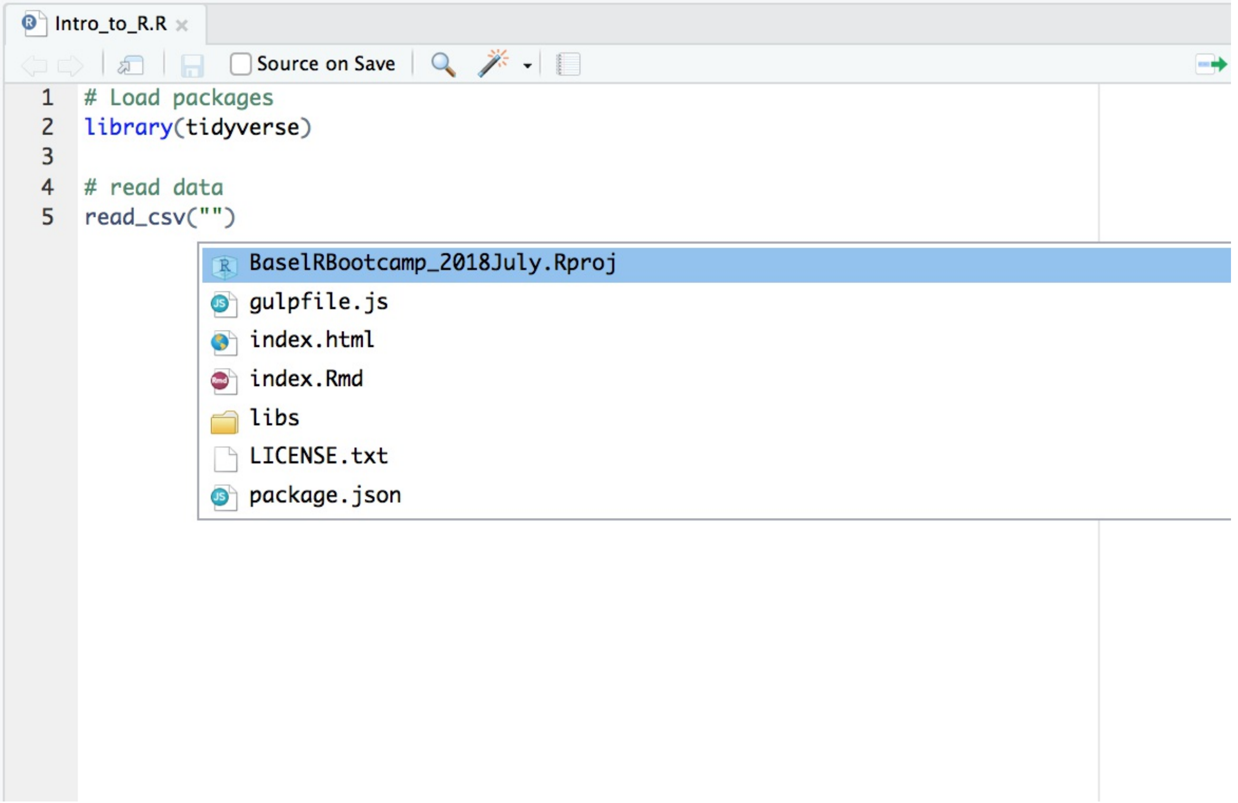
Files ending in `.gz`, `.bz2`, `.xz`, or `.zip` will be automatically uncompressed. Files starting with `http://`, `https://`, `ftp://`, or `ftps://` will be automatically downloaded. Remote gz files can also be automatically downloaded and decompressed.

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

Press F1 for additional help

Essentials: 11 basic R lessons

1. Everything is an object
2. <- creates/changes objects
3. Everything happens through functions
4. Functions have (default) arguments
5. Functions live in packages
6. Find help with ?
7. Data lives in data frames
8. 3 data types + factors
9. `formula` and `data` specify a model
10. Use RStudio and projects
11. **Use editor, shortcuts, auto-complete**



```
# Load packages
library(tidyverse)

# read data
read_csv("")
```

Download

Interactive