

Data

Introduction to Data Science with R

www.therbootcamp.com

@therbootcamp

October 2018

Data

In this session you will get to know...

- R's 3 main **data types**
- a little more about **functions**
- R's **Import/Export** functions



3 Object types for data

R has 3 main data objects...

list - R's multi-purpose container

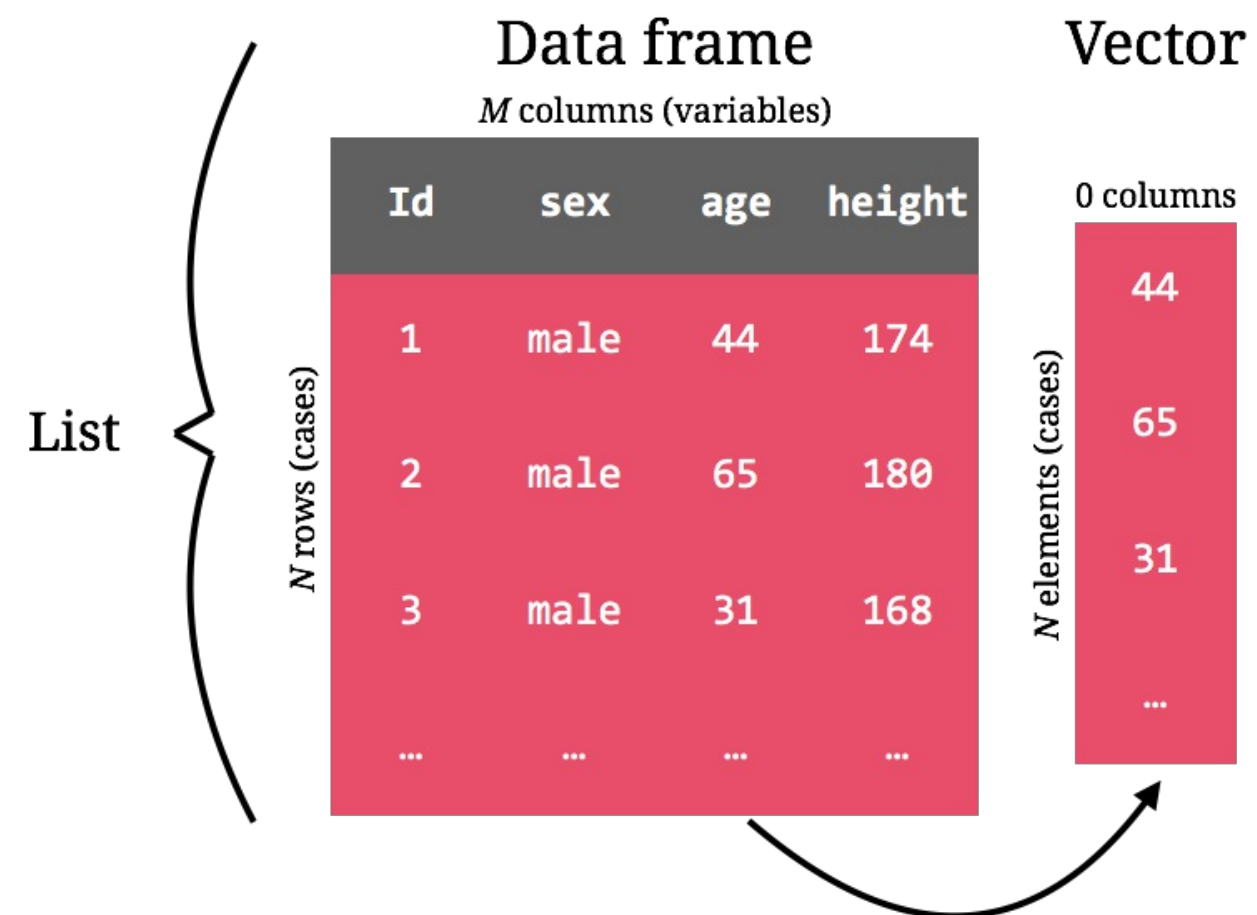
- Can carry any data, incl. lists
- Often used for function outputs

data_frame - R's spreadsheet

- Specific type of list
- Typical data format
- For multi-variable data sets

vectors - R's data container

- Actually carries the data
- Contain data of 1 of many types



list

- 1 - Can **carry any data**, incl. lists, data_frames, vectors, etc.
- 2 - Are often used for **function outputs**
- 3 - Have **named elements**.
- 4 - Elements can be **inspected** via `names()` or `str()`.
- 5 - Elements are (typically) **selected** by `$`.

List

N **named** Elements (objects)

coefficients			df	resid- uals	r square
var	est.	T	99	1.74	0.37
				1.42	
			99	8.21	
x1	1.17	1.86		4.24	
				0.45	
				43.1	
x2	3.32	2.65		2.1	
				...	

List: Select element using \$

```
# regression
reg_model <- lm(height ~ sex + age,
                 data = baselers)
reg_results <- summary(reg_model)

# get element names
names(reg_results)
```

```
## [1] "call"      "terms"
## [3] "residuals" "coefficients"
## [5] "aliased"    "sigma"
## [7] "df"         "r.squared"
```

```
# select element using $
reg_results$coefficients
```

```
##           Estimate t value
## (Intercept) 164.171266 499.5339
## sexmale      13.993699  66.4724
## age          -0.003753  -0.5819
```

List

N named Elements (objects)

coefficients			df	resid- uals	r square
var	est.	T	99	1.74	0.37
				1.42	
				8.21	
x1	1.17	1.86		4.24	
				0.45	
			99	43.1	
x2	3.32	2.65		2.1	
				...	

data_frame

- 1 - Are lists containing **vectors of equal length** representing the variables.
- 2 - Contain vectors of different types: numeric, character, etc.
- 3 - Have named elements.
- 4 - Elements can be **inspected** via `names()`, `str()`, `print()`, `View()`, or `skimr::skim()`.
- 5 - Elements are (typically) **selected** by `$`.
- 6 - Come in different flavors: `data.frame()`, `data.table()`, `tibble()`.

Data frame
4 columns (variables)

N rows (cases)	Id	sex	age	height
	1	male	44	174
	2	male	65	180
	3	male	31	168

Inspect content

```
# inspect baselers via 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 fema...   31   168.    55.5   5100
## 4     4  male    27   209    93.8   4200
## 5     5  male    24   177.    NA    4000
##   education confession children
##   <chr>      <chr>      <int>
## 1 SEK_III   catholic      2
## 2 obligato... confessio...      2
## 3 SEK_III   <NA>          2
## 4 SEK_III   catholic      2
## 5 SEK_III   catholic      1
## # ... with 9,995 more rows, and 11 more
## #   variables
```

Data frame

4 columns (variables)

N rows (cases)	Id	sex	age	height
	1	male	44	174
	2	male	65	180
	3	male	31	168

Inspect content

```
# inspect baselers via print  
View(baselers)
```

	id	sex	age	height	weight	income
1	1	male	44	174.3	113.4	6300
2	2	male	65	180.3	75.2	10900
3	3	female	31	168.3	55.5	5100
4	4	male	27	209.0	93.8	4200
5	5	male	24	176.7	NA	4000
6	6	male	63	186.6	67.4	11400
7	7	male	71	151.6	83.3	12000
8	8	female	41	155.7	67.8	7600
9	9	male	43	176.1	69.3	8500
10	10	female	31	166.1	66.3	6100
11	11	female	42	157.8	51.9	8000

Data frame

4 columns (variables)

N rows (cases)	Id	sex	age	height
	1	male	44	174
	2	male	65	180
	3	male	31	168

Select via \$

```
# select age variable  
baselers$age
```

```
## [1] 44 65 31 27 24 63 71 41 43 31 42 31  
## [13] 38 49 39 54 78 62 88 74
```

```
# select age variable  
baselers$education
```

```
## [1] "SEK_III"  
## [2] "obligatory_school"  
## [3] "SEK_III"  
## [4] "SEK_III"  
## [5] "SEK_III"  
## [6] "SEK_III"  
## [7] "SEK_III"  
## [8] "SEK_III"  
## [9] "apprenticeship"  
## [10] "SEK_II"
```

Data frame

4 columns (variables)

N rows (cases)	Id	sex	age	height
	1	male	44	174
	2	male	65	180
	3	male	31	168

Change/Add via \$

```
# compute age in months
baselers$age <- baselers$age * 2

# inspect baselers
baselers
```

```
## # A tibble: 10,000 x 20
##       id sex    age height weight income
##   <int> <chr> <dbl> <dbl> <dbl> <dbl>
## 1     1 male    88   174.  113.   6300
## 2     2 male   130   180.   75.2  10900
## 3     3 fema...  62   168.   55.5   5100
## 4     4 male    54   209    93.8   4200
## 5     5 male    48   177.    NA   4000
##   education confession children
##   <chr>      <chr>      <int>
## 1 SEK_III   catholic      2
## 2 obligato... confessio...      2
## 3 SEK_III   <NA>          2
## 4 SEK_III   catholic      2
## 5 SEK_III   catholic      1
## # ... with 9,995 more rows, and 11 more
```

Data frame

4 columns (variables)

N rows (cases)	Id	sex	age	height
	1	male	44	174
	2	male	65	180
	3	male	31	168

Tidy data

- 1 - Each variable you measure should be in one column.
- 2 - Each different observation of that variable should be in a different row.
- 3 - There should be one table for each "kind" of variable.
- 4 - If you have multiple tables, they should include a column in the table that allows them to be linked.

see [The Elements of Data Analytic Style](#) by Jeff Leek

Data frame

4 columns (variables)

N rows (cases)	Id	sex	age	height
	1	male	44	174
	2	male	65	180
	3	male	31	168

vector

1 - R's **basic and, in a way, only data container**.

2 - Can contain only a **single type of data** and missing values.

3 - Data types

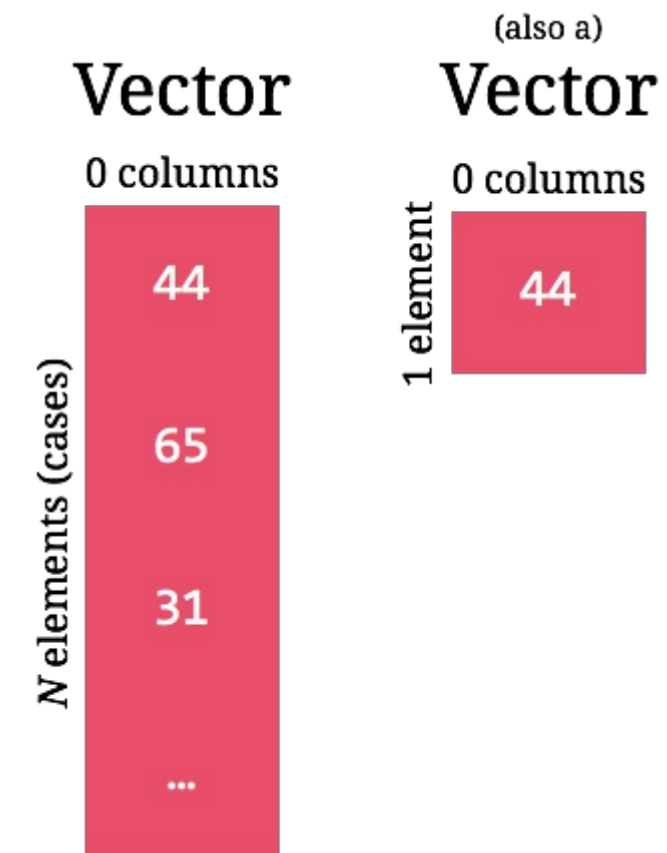
numeric - All numbers

character - All characters (e.g., names)

logical - TRUE or FALSE

...

NA - missing values



Select/Change/(Add) via []

```
# extract vector containing age
age <- baselers$age
age
```

```
## [1] 88 130 62 54 48 126 142 82 86
```

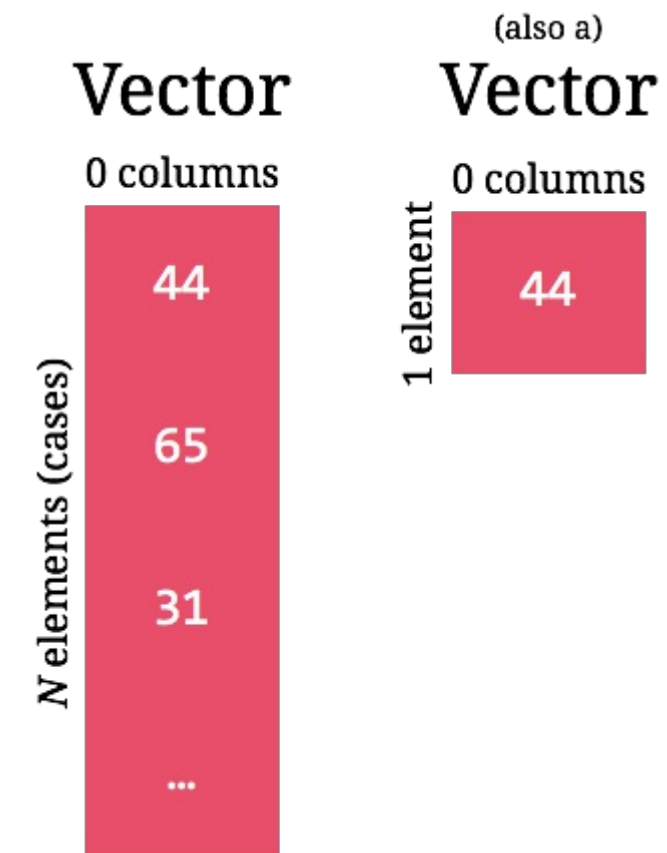
```
# select value
age[2]
```

```
## [1] 130
```

```
# change value
age[2] <- 2
age
```

```
## [1] 88 2 62 54 48 126 142 82 86
```

Find more info on indexing [here](#).



Data types: numeric

numeric vectors are used to store numbers and only numbers.

```
baselers$age

## [1] 88 130 62 54 48 126 142 82 86

# evaluate type
typeof(baselers$age)

## [1] "double"

is.numeric(baselers$age)

## [1] TRUE
```

numeric Vector	character Vector	logical Vector
.\$age	.\$sex	.\$sex=="male"
44	"male"	TRUE
65	"female"	FALSE
31	"male"	TRUE
...

Data types: character

character vector are used to store data represented by **letters and symbols, and all other data**.

```
baselers$sex
```

```
## [1] "male" "male" "female" "male"  
## [5] "male" "male" "male" "female"
```

```
# evaluate type  
as.character(baselers$age)
```

```
## [1] "88" "130" "62" "54" "48" "126"  
## [7] "142" "82" "86"
```

numeric Vector	character Vector	logical Vector
.\$age	.\$sex	.\$sex=="male"
44	"male"	TRUE
65	"female"	FALSE
31	"male"	TRUE
...

Data types: Logical

logical vector are used to **data** aka to select elements or rows. logical are typically created from other vectors via **logical comparisons**.

```
baselers$sex == "male"
```

```
## [1] TRUE TRUE FALSE TRUE TRUE TRUE  
## [7] TRUE FALSE
```

```
# evaluate type  
baselers$age < 30
```

```
## [1] TRUE TRUE TRUE TRUE TRUE TRUE TRUE  
## [8] TRUE TRUE
```

numeric Vector	character Vector	logical Vector
.\$age	.\$sex	.\$sex=="male"
44	"male"	TRUE
65	"female"	FALSE
31	"male"	TRUE
...

Data types: Logical

logical vector are used to **data** aka to select elements or rows. logical are typically created from other vectors via **logical comparisons**.

Logical operators

== - is equal to

<, > - smaller/greater than

≤, ≥ - smaller/greater than or equal

&, && - logical AND

|, || - logical OR

numeric Vector	character Vector	logical Vector
.\$age	.\$sex	.\$sex=="male"
44	"male"	TRUE
65	"female"	FALSE
31	"male"	TRUE
...

Object Classes

- 1 - R's objects have **content and attributes**.
- 2 - Attributes include always **names**, **dimensions**, and the **class** (or type) of the object.
- 3 - **Classes** are critical because they determine **when and how they can be used in functions!**

R (data) object

data	attributes
<i>numbers</i> -0.62, -0.08, 0.6 1, 0.52, -0.47, 0. 44, -0.57, -0.52	<i>names</i> names
<i>~ and/or ~</i> <i>character strings</i> ZRU, ZJB, PTK, QBD , CWK, HZM, LKB, RN Y, KFB, GOF	<i>dimensions, length</i> dim, length
<i>~ and/or ~</i> <i>many other types</i>	<i>class, type</i> class
	<i>other attributes</i> attributes

essential

Functions

Functions have 3 elements:

1 - **Name**: Used to refer to the function and call (execute) it.

2 - **Arguments**: Used to provide (data) inputs and to control what the function does.

Arguments with default values (e.g., `use = "everything"`) need not be specified.

Arguments without default values (e.g., `x`) need be specified. **Inputs must have the appropriate class!**

3 - **Body**: The code that uses the inputs (arguments) to produce the desired output. The code of the functions body is based **copies of the inputs**, which are named according to the arguments names.

A function

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

```
na.method <- pmatch(use, c("all.obs",  
  "complete.obs", "pairwise.complete.obs",  
  "everything", "na.or.complete"))  
...  
if (method == "pearson")  
  .Call(C_cor, x, y, na.method, FALSE)  
else if (na.method %in% c(2L, 5L)) {  
  if (is.null(y))  
    .Call(C_cor, Rank(na.omit(x)), NULL,  
          na.method, method == "kendall")  
  }  
...  
}
```

Documentation

R documentation (**help files** and **vignettes**) will become very easy to use once you are familiar with the basic R vocabulary.

Pay attention to...

Usage - shows how to use function, its arguments and their defaults.

Arguments - describes arguments, and their class.

Value - describes what the function returns.

Examples - provide working R code.

```
# To access help files
?name_of_function

# search help files
??name_of_function
```

?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.

Raw (structured) Data

delim-separated data

```
id,sex,age,height,weight,income,education,confession,children
1,male,44,174.3,113.4,6300,SEK_III,catholic,2,5,7,610,40,6,4
2,male,65,180.3,75.2,10900,obligatory_school,confessionless,
3,female,31,168.3,55.5,5100,SEK_III,NA,2,7,6,720,14,3,6,102,
4,male,27,209,93.8,4200,SEK_III,catholic,2,7,8,680,39,6,0,11
5,male,24,176.7,NA,4e3,SEK_III,catholic,1,5,4,260,19,0,1,82,
6,male,63,186.6,67.4,11400,SEK_III,evangelical-reformed,0,7,
7,male,71,151.6,83.3,12e3,SEK_III,evangelical-reformed,2,8,5
8,female,41,155.7,67.8,7600,SEK_III,confessionless,1,7,2,135
9,male,43,176.1,69.3,8500,apprenticeship,catholic,2,7,5,150,
10,female,31,166.1,66.3,6100,SEK_II,catholic,1,6,7,700,0,0,3
11,female,42,157.8,51.9,8e3,obligatory_school,catholic,2,9,7
12,male,31,165.9,66,5900,apprenticeship,evangelical-reforme
13,female,38,162.5,73.4,6200,apprenticeship,confessionless,2
14,female,49,182.8,46.9,NA,SEK_III,evangelical-reformed,1,6,
15,female,39,160,NA,5600,SEK_III,other,2,7,4,540,35,7,4,122,
16,female,54,139.7,50.3,10900,SEK_III,evangelical-reformed,3
17,female,78,153.1,64.1,11e3,SEK_III,confessionless,2,7,2,95
18,female,62,174.6,63.8,11500,SEK_III,confessionless,2,9,7,1
19,male,88,191.4,99.8,14200,SEK_III,confessionless,2,7,3,121
20,male,74,183.8,78.1,12100,apprenticeship,catholic,2,5,7,11
```

markup data

```
<!doctype html>
<html lang="en" class="gr__therbootcamp_github_io">
  <head>...</head>
  <body data-gr-c-s-loaded="true">
    <script async src="https://www.google-analytics.com/analytics.js">
    <script type="text/javascript" async src="https://snap.licdn.com/li
    insight.min.js"></script>
    <script type="text/javascript">
      _linkedin_data_partner_id = "111419";
    </script>
    <script type="text/javascript">...</script>
  <div id="particles-js">
    <div class="content">
      <h1>
        <span class="site-title">TheRBootcamp</span>
        <span class="site-description">Learn Data Science in R</
      <a class="link" href="#upcoming" data-scroll>
        <font size="6" color="#FF3A2A">Basel July 21 22 28 29
      </a>
    </h1>
```

Delim-separated data

- 1 - Most typical file format.
- 2 - Requires **delimiter** to separate entries.



delim-separated data

```
id,sex,age,height,weight,income,education,confession,children
1,male,44,174.3,113.4,6300,SEK_III,catholic,2,5,7,610,40,6,4
2,male,65,180.3,75.2,10900,obligatory_school,confessionless,
3,female,31,168.3,55.5,5100,SEK_III,NA,2,7,6,720,14,3,6,102,
4,male,27,209,93.8,4200,SEK_III,catholic,2,7,8,680,39,6,0,11
5,male,24,176.7,NA,4e3,SEK_III,catholic,1,5,4,260,19,0,1,82,
6,male,63,186.6,67.4,11400,SEK_III,evangelical-reformed,0,7,
7,male,71,151.6,83.3,12e3,SEK_III,evangelical-reformed,2,8,5
8,female,41,155.7,67.8,7600,SEK_III,confessionless,1,7,2,135
9,male,43,176.1,69.3,8500,apprenticeship,catholic,2,7,5,150,
10,female,31,166.1,66.3,6100,SEK_II,catholic,1,6,7,700,0,0,3
11,female,42,157.8,51.9,8e3,obligatory_school,catholic,2,9,7
12,male,31,165.9,66,5900,apprenticeship,evangelical-reforme
13,female,38,162.5,73.4,6200,apprenticeship,confessionless,2
14,female,49,182.8,46.9,NA,SEK_III,evangelical-reformed,1,6,
15,female,39,160,NA,5600,SEK_III,other,2,7,4,540,35,7,4,122,
16,female,54,139.7,50.3,10900,SEK_III,evangelical-reformed,3
17,female,78,153.1,64.1,11e3,SEK_III,confessionless,2,7,2,97
18,female,62,174.6,63.8,11500,SEK_III,confessionless,2,9,7,1
19,male,88,191.4,99.8,14200,SEK_III,confessionless,2,7,3,121
20,male,74,183.8,78.1,12100,apprenticeship,catholic,2,5,7,11
```

readr

readr is a tidyverse package that provides convenient functions to **read in** (non-nested) data files into data frames (tibbles to be precise):



```
# Importing data from a file
```

```
data <- read_csv(file, ...)  # comma-delimited  
data <- read_csv2(file, ...) # semicolon-delimited  
data <- read_delim(file, ...) # arbitrary-delimited
```

```
# Writing a data frame to a file
```

```
write_csv(data_object, file, ...)  # comma-delimited  
write_delim(data_object, file, ...) # arbitrary-delimited
```

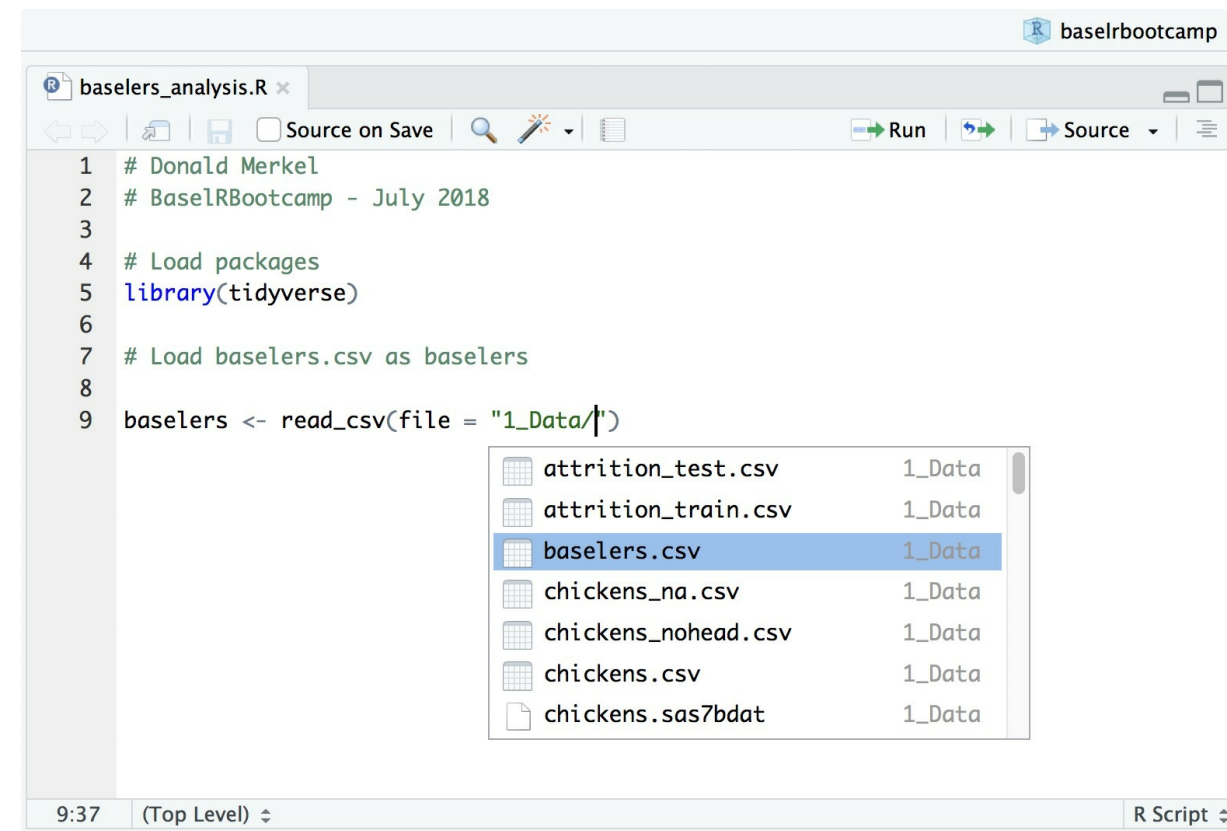

Finding the file path

1 - Identify the file path using the **auto-complete**.

2 - Initiate auto-complete and browse through the folder structure by placing the cursor between two quotation marks and using the **tab key**.



3 - Auto-complete begins with the project folder - **place your data inside your project folder!**

A screenshot of the RStudio IDE. The main editor window shows a script named 'baselers_analysis.R' with the following code:

```
1 # Donald Merkel
2 # BaselRBootcamp - July 2018
3
4 # Load packages
5 library(tidyverse)
6
7 # Load baselers.csv as baselers
8
9 baselers <- read_csv(file = "1_Data/|")
```

The cursor is at the end of the string "1_Data/|". An auto-complete dropdown menu is visible, listing several files in the '1_Data' directory: 'attrition_test.csv', 'attrition_train.csv', 'baselers.csv' (which is highlighted), 'chickens_na.csv', 'chickens_nohead.csv', 'chickens.csv', and 'chickens.sas7bdat'. The status bar at the bottom shows '9:37 (Top Level) R Script'.

Identifying the delimiter

- 1 - **Find the file** on your hard drive. Should be in your data folder inside your project.
- 2 - **Open the file** in RStudio (right-click on the file in the pane) a text viewer, e.g., `code editor` (Mac), `Notepad++` (Mac), `Notepad` (Windows).



baselers.csv

```
id,sex,age,height,weight,income,education,confession,children
1,male,44,174.3,113.4,6300,SEK_III,catholic,2,5,7,610,40,6,4
2,male,65,180.3,75.2,10900,obligatory_school,confessionless,
3,female,31,168.3,55.5,5100,SEK_III,NA,2,7,6,720,14,3,6,102,
4,male,27,209,93.8,4200,SEK_III,catholic,2,7,8,680,39,6,0,11
5,male,24,176.7,NA,4e3,SEK_III,catholic,1,5,4,260,19,0,1,82,
6,male,63,186.6,67.4,11400,SEK_III,evangelical-reformed,0,7,
7,male,71,151.6,83.3,12e3,SEK_III,evangelical-reformed,2,8,5
8,female,41,155.7,67.8,7600,SEK_III,confessionless,1,7,2,135
9,male,43,176.1,69.3,8500,apprenticeship,catholic,2,7,5,150,
10,female,31,166.1,66.3,6100,SEK_II,catholic,1,6,7,700,0,0,3
11,female,42,157.8,51.9,8e3,obligatory_school,catholic,2,9,7
12,male,31,165.9,66,5900,apprenticeship,evangelical-reforme
13,female,38,162.5,73.4,6200,apprenticeship,confessionless,2
14,female,49,182.8,46.9,NA,SEK_III,evangelical-reformed,1,6,
15,female,39,160,NA,5600,SEK_III,other,2,7,4,540,35,7,4,122,
16,female,54,139.7,50.3,10900,SEK_III,evangelical-reformed,3
17,female,78,153.1,64.1,11e3,SEK_III,confessionless,2,7,2,97
18,female,62,174.6,63.8,11500,SEK_III,confessionless,2,9,7,1
19,male,88,191.4,99.8,14200,SEK_III,confessionless,2,7,3,121
20,male,74,183.8,78.1,12100,apprenticeship,catholic,2,5,7,11
```

Identifying the delimiter

- 1 - **Find the file** on your hard drive. Should be in your data folder inside your project.
- 2 - **Open the file** in RStudio (right-click on the file in the pane) a text viewer, e.g., `vim` (Mac), `notepad` (Mac), `notepad++` (Windows).

```
# Read with explicit column names
baselers <- read_delim(file = ".../baselers.csv",
                      delim = c(",",""))
```

baselers.csv

```
id,sex,age,height,weight,income,education,confession,children
1,male,44,174.3,113.4,6300,SEK_III,catholic,2,5,7,610,40,6,4
2,male,65,180.3,75.2,10900,obligatory_school,confessionless,
3,female,31,168.3,55.5,5100,SEK_III,NA,2,7,6,720,14,3,6,102,
4,male,27,209,93.8,4200,SEK_III,catholic,2,7,8,680,39,6,0,11
5,male,24,176.7,NA,4e3,SEK_III,catholic,1,5,4,260,19,0,1,82,
6,male,63,186.6,67.4,11400,SEK_III,evangelical-reformed,0,7,
7,male,71,151.6,83.3,12e3,SEK_III,evangelical-reformed,2,8,5
8,female,41,155.7,67.8,7600,SEK_III,confessionless,1,7,2,135
9,male,43,176.1,69.3,8500,apprenticeship,catholic,2,7,5,150,
10,female,31,166.1,66.3,6100,SEK_II,catholic,1,6,7,700,0,0,3
11,female,42,157.8,51.9,8e3,obligatory_school,catholic,2,9,7
12,male,31,165.9,66,5900,apprenticeship,evangelical-reforme
13,female,38,162.5,73.4,6200,apprenticeship,confessionless,2
14,female,49,182.8,46.9,NA,SEK_III,evangelical-reformed,1,6,
15,female,39,160,NA,5600,SEK_III,other,2,7,4,540,35,7,4,122,
16,female,54,139.7,50.3,10900,SEK_III,evangelical-reformed,3
17,female,78,153.1,64.1,11e3,SEK_III,confessionless,2,7,2,97
18,female,62,174.6,63.8,11500,SEK_III,confessionless,2,9,7,1
19,male,88,191.4,99.8,14200,SEK_III,confessionless,2,7,3,121
20,male,74,183.8,78.1,12100,apprenticeship,catholic,2,5,7,11
```


Handling headers

1 - readr- functions typically expect the **column names** in the first line.

2 - If no column names are available, use the **col_names-argument** to provide them.

```
# Read with explicit column names
baselers <- read_csv(file = ".../baselers.csv",
                     col_names = c("id",
                                   "age",
                                   ...))
```

baselers.csv

```
id,sex,age,height,weight,income,education,confession,children
1,male,44,174.3,113.4,6300,SEK_III,catholic,2,5,7,610,40,6,4
2,male,65,180.3,75.2,10900,obligatory_school,confessionless,
3,female,31,168.3,55.5,5100,SEK_III,NA,2,7,6,720,14,3,6,102,
4,male,27,209,93.8,4200,SEK_III,catholic,2,7,8,680,39,6,0,11
5,male,24,176.7,NA,4e3,SEK_III,catholic,1,5,4,260,19,0,1,82,
6,male,63,186.6,67.4,11400,SEK_III,evangelical-reformed,0,7,
7,male,71,151.6,83.3,12e3,SEK_III,evangelical-reformed,2,8,5
8,female,41,155.7,67.8,7600,SEK_III,confessionless,1,7,2,135
9,male,43,176.1,69.3,8500,apprenticeship,catholic,2,7,5,150,
10,female,31,166.1,66.3,6100,SEK_II,catholic,1,6,7,700,0,0,3
11,female,42,157.8,51.9,8e3,obligatory_school,catholic,2,9,7
12,male,31,165.9,66,5900,apprenticeship,evangelical-reforme
13,female,38,162.5,73.4,6200,apprenticeship,confessionless,2
14,female,49,182.8,46.9,NA,SEK_III,evangelical-reformed,1,6,
15,female,39,160,NA,5600,SEK_III,other,2,7,4,540,35,7,4,122,
16,female,54,139.7,50.3,10900,SEK_III,evangelical-reformed,3
17,female,78,153.1,64.1,11e3,SEK_III,confessionless,2,7,2,97
18,female,62,174.6,63.8,11500,SEK_III,confessionless,2,9,7,1
19,male,88,191.4,99.8,14200,SEK_III,confessionless,2,7,3,121
20,male,74,183.8,78.1,12100,apprenticeship,catholic,2,5,7,11
```

Handling data types

Reading in data, **readr infers the type of data** for each column.

```
# Read baselers
read_csv(file = "1_Data/baselers.csv")

## Parsed with column specification:
## cols(
##   .default = col_integer(),
##   sex = col_character(),
##   height = col_double(),
##   weight = col_double(),
##   income = col_double(),
##   education = col_character(),
##   confession = col_character(),
##   food = col_double(),
##   fasnacht = col_character(),
##   eyecor = col_character()
## )

## See spec(...) for full column specifications.
```

baselers.csv

```
id,sex,age,height,weight,income,education,confession,children
1,male,44,174.3,113.4,6300,SEK_III,catholic,2,5,7,610,40,6,4
2,male,65,180.3,75.2,10900,obligatory_school,confessionless,
3,female,31,168.3,55.5,5100,SEK_III,NA,2,7,6,720,14,3,6,102,
4,male,27,209,93.8,4200,SEK_III,catholic,2,7,8,680,39,6,0,11
5,male,24,176.7,NA,4e3,SEK_III,catholic,1,5,4,260,19,0,1,82,
6,male,63,186.6,67.4,11400,SEK_III,evangelical-reformed,0,7,
7,male,71,151.6,83.3,12e3,SEK_III,evangelical-reformed,2,8,5
8,female,41,155.7,67.8,7600,SEK_III,confessionless,1,7,2,135
9,male,43,176.1,69.3,8500,apprenticeship,catholic,2,7,5,150,
10,female,31,166.1,66.3,6100,SEK_II,catholic,1,6,7,700,0,0,3
11,female,42,157.8,51.9,8e3,obligatory_school,catholic,2,9,7
12,male,31,165.9,66,5900,apprenticeship,evangelical-reforme
13,female,38,162.5,73.4,6200,apprenticeship,confessionless,2
14,female,49,182.8,46.9,NA,SEK_III,evangelical-reformed,1,6,
15,female,39,160,NA,5600,SEK_III,other,2,7,4,540,35,7,4,122,
16,female,54,139.7,50.3,10900,SEK_III,evangelical-reformed,3
17,female,78,153.1,64.1,11e3,SEK_III,confessionless,2,7,2,97
18,female,62,174.6,63.8,11500,SEK_III,confessionless,2,9,7,1
19,male,88,191.4,99.8,14200,SEK_III,confessionless,2,7,3,121
20,male,74,183.8,78.1,12100,apprenticeship,catholic,2,5,7,11
```


Handling data types

Incorrect data types can be fixed. Typically this involves:

- 1 - **removing character elements** from otherwise numeric variables.
- 2 - Setting **explicit NA strings** using the `na`-argument.
- 3 - Re-running `type_convert`.

```
# Read baselers
baseslers <- read_csv(file = ".../baselers.csv",
                      na = c('NA'))

# Try to fix incorrect data types
baselers <- type_convert(baseslers)
```

baselers.csv

```
id,sex,age,height,weight,income,education,confession,children
1,male,44,174.3,113.4,6300,SEK_III,catholic,2,5,7,610,40,6,4
2,male,65,180.3,75.2,10900,obligatory_school,confessionless,
3,female,31,168.3,55.5,5100,SEK_III,NA,2,7,6,720,14,3,6,102,
4,male,27,209,93.8,4200,SEK_III,catholic,2,7,8,680,39,6,0,11
5,male,24,176.7,NA,4e3,SEK_III,catholic,1,5,4,260,19,0,1,82,
6,male,63,186.6,67.4,11400,SEK_III,evangelical-reformed,0,7,
7,male,71,151.6,83.3,12e3,SEK_III,evangelical-reformed,2,8,5
8,female,41,155.7,67.8,7600,SEK_III,confessionless,1,7,2,135
9,male,43,176.1,69.3,8500,apprenticeship,catholic,2,7,5,150,
10,female,31,166.1,66.3,6100,SEK_II,catholic,1,6,7,700,0,0,3
11,female,42,157.8,51.9,8e3,obligatory_school,catholic,2,9,7
12,male,31,165.9,66,5900,apprenticeship,evangelical-reforme
13,female,38,162.5,73.4,6200,apprenticeship,confessionless,2
14,female,49,182.8,46.9,NA,SEK_III,evangelical-reformed,1,6,
15,female,39,160,NA,5600,SEK_III,other,2,7,4,540,35,7,4,122,
16,female,54,139.7,50.3,10900,SEK_III,evangelical-reformed,3
17,female,78,153.1,64.1,11e3,SEK_III,confessionless,2,7,2,97
18,female,62,174.6,63.8,11500,SEK_III,confessionless,2,9,7,1
19,male,88,191.4,99.8,14200,SEK_III,confessionless,2,7,3,121
20,male,74,183.8,78.1,12100,apprenticeship,catholic,2,5,7,11
```

Other data

R provides **read and write functions** for practically all data file formats. See [rio](#).

readr



```
# read fixed width files (can be fast)
data <- read_fwf(file, ...)

# read Apache style log files
data <- read_log(file, ...)
```

haven



```
# read SAS's .sas7bat and sas7bcat files
data <- read_sas(file, ...)

# read SPSS's .sav files
data <- read_sav(file, ...)

# etc
```

readxl



```
# read Excel's .xls and .xlsx files
data <- read_excel(file, ...)
```

Other

```
# Read Matlab .mat files
data <- R.matlab::readMat(file, ...)

# Read and wrangle .xml and .html
data <- XML::xmlParseParse(file, ...)

# from package jsonlite: read .json files
data <- jsonlite::read_json(file, ...)
```

Remote databases

R provides **all necessary tools to pull data from or directly work with** remote databases such as, e.g., a SQL database. Find out more at:

db.rstudio.com

Practical

[Link to practical](#)