

# Introduction to R Programming

## Reading and Writing Tables

Pedro Fonseca

29 Abril 2020

## Before importing data

- ▶ Are you in the correct working directory? Are you in a project?
- ▶ Remember `getwd()` and `setwd()`.
- ▶ Typically we import tables stored in text files or CSV files (comma separated values).
- ▶ There are packages that allow importing xls and xlsx tables into R, but if you want to read data from excel I recommend saving your excel worksheet as a CSV or text (tab delimited) file instead.
- ▶ It is important to open the file where the table is stored with a text editor and pay attention to the structure of the data.

# The source file

What to look for:

- ▶ Headers (column names)
- ▶ Separator (" ", ";", "\t", ...)
- ▶ How are strings represented? (with "\" ?)
- ▶ Decimal separators (".", or ",", "?)
- ▶ Row names
- ▶ How are NAs represented? (" ", "NA", "na", "Na", ".", "-", ...)

# The source file

What to look for:

- ▶ Should we read all the lines of the file?
- ▶ Is there any metadata? Does the table start in the first line of the file?
- ▶ Are there comments? What is the comment character?

## Base R functions for reading tables

- ▶ All we need is `read.table()`!

But there are some useful wrappers around `read.table()`:

- ▶ `read.csv()`
- ▶ `read.csv2()`
- ▶ `read.delim()`
- ▶ `read.delim2()`

These functions read a file in table format and create a data frame that you can assign to an R object.

## read.table()

read.table() is our powerhouse function for reading tables from external files. These are the most important defaults:

```
read.table(file, header = FALSE, sep = "",  
           quote = "\"'", dec = ".", row.names,  
           col.names, na.strings = "NA",  
           nrows = -1, skip = 0, comment.char = "#")
```

- ▶ The file argument is where you specify the name of the file you want to read.
- ▶ If the file is not in the working directory you should indicate the path to the file instead.
- ▶ File names and paths must be provided as strings of characters.

## `read.table()`

- ▶ To the `row.names` argument you can either provide a vector with the names or a single number or character string indicating the position or the name of the column containing the names.
- ▶ If there is a header and the first row contains one fewer field than the number of columns, the first column is used for the row names. If `row.names` is missing, the rows are numbered.

## Wrappers around `read.table()`

`read.csv()` and `read.csv2()` are identical to `read.table()` except for the defaults. They are intended for reading CSV files with:

- ▶ `read.csv()`: `sep = ","` and `dec = "."`
- ▶ `read.csv2()`: `sep = ";"` and `dec = ","`

Also, both `read.csv()` and `read.csv2()` have `header = TRUE` and `comment.char = ""`, which means that the comment character is disabled.



## Wrappers around `read.table()`

`read.delim()` and `read.delim2()` are identical to `read.table()` except for the defaults. They are intended for reading TAB delimited files (`sep = "\t"`) with:

- ▶ `read.delim()`: `dec = "."`
- ▶ `read.delim2()`: `dec = ","`

Also, both `read.delim()` and `read.delim2()` have `header = TRUE` and `comment.char = ""`, which means that the comment character is disabled.

# Getting started

- ▶ Check your working directory
- ▶ Change it if you want
- ▶ Put the `mt.csv` file in your current working directory
- ▶ Now read it and save the table in an R object named `mt`
- ▶ Open the `mt` data frame using your global environment or with `View()`

```
mt <- read.csv(file = "mt.csv")
```

```
View(mt)
```

## Getting started

- ▶ Now create a new folder on your working directory and name it “datasets”.
- ▶ Store this lecture's datasets there (except `mt.csv`).

## Using read.csv()

Read the mtcars0.csv file from the datasets folder and assign it to an object:

```
mtcars0 <- read.csv(file = "datasets/mtcars0.csv")
```

## Using read.csv()

Read the mtcars1.csv file from the datasets folder and assign it to an object:

```
mtcars1 <- read.csv(file = "datasets/mtcars1.csv",  
                    row.names = 1)
```

## Using read.csv2()

Read the airquality0.csv file from the datasets folder and assign it to an object:

```
air0 <- read.csv2("datasets/airquality0.csv")
```

## Using read.csv2()

Read the airquality1.csv file from the datasets folder and assign it to an object:

```
air1 <- read.csv2("datasets/airquality1.csv",  
                  row.names = 1)
```

## Using read.delim()

Read the iris0.txt file from the datasets folder and assign it to an object:

```
iris0 <- read.delim("datasets/iris0.txt")
```



## Using read.delim()

Read the iris1.txt file from the datasets folder and assign it to an object:

```
iris1 <- read.delim("datasets/iris1.txt",  
                    header = FALSE,  
                    row.names = 1)
```

## Using read.delim2()

Read the weight file from the datasets folder and assign it to an object:

```
weight <- read.delim2("datasets/weight.txt")
```

## Using read.delim2()

Read the womanNA.txt file from the datasets folder and assign it to an object:

```
woman <- read.delim2("datasets/womanNA.txt",  
                      na.strings = c("NA", "Na"))
```

## Using read.table()

Read the woman.txt file from the datasets folder and assign it to an object:

```
woman2 <- read.table("datasets/woman.txt",  
                      sep = "\t",  
                      header = TRUE)
```

## Using read.table()

Read the AirPassengers.csv file from the datasets folder and assign it to an object:

```
airPass <- read.table("datasets/AirPassengers.csv",  
                      sep = "|")
```

Why didn't I use row.names = 1?

## Using read.table()

Read the ChickWeight.txt file from the datasets folder and assign it to an object:

```
chick <- read.table("datasets/ChickWeight.txt",  
                    sep = "&",  
                    row.names = 1)
```

## Using read.table()

Read the iris.csv file from the datasets folder and assign it to an object:

```
iris2 <- read.table("datasets/iris.csv",  
                    sep = ";",  
                    dec = ",",  
                    header = TRUE)
```

## Using read.table()

Read the irisNA.csv file from the datasets folder and assign it to an object:

```
iris3 <- read.table("datasets/irisNA.csv",  
                    sep = ";",  
                    dec = ",",  
                    header = TRUE,  
                    na.strings = "-")
```



## Using read.table()

Read the irisNA2.csv file from the datasets folder and assign it to an object:

```
iris4 <- read.table("datasets/irisNA2.csv",  
                    sep = ";",  
                    dec = ",",  
                    header = TRUE,  
                    na.strings = "-",  
                    skip = 5)
```

## Using read.table()

Read the irisNA3.csv file from the datasets folder and assign it to an object:

```
iris5 <- read.table("datasets/irisNA3.csv",  
                    sep = ";",  
                    dec = ",",  
                    header = TRUE,  
                    na.strings = "-",  
                    skip = 3,  
                    comment.char = "%")
```

## Reading a file from the internet

It's also possible to use any of the functions that we've just learned to import files from the web:

```
my_data <- read.delim(  
  "http://www.sthda.com/upload/boxplot_format.txt")  
  
head(my_data)
```

	Nom	variable	Group
## 1	IND1	10	A
## 2	IND2	7	A
## 3	IND3	20	A
## 4	IND4	14	A
## 5	IND5	14	A
## 6	IND6	12	A

# Writing data

After doing calculations with imported data, we usually want to store tables with our results in a local file.

This can be done with:

- ▶ `write.table()`
- ▶ `write.csv()`
- ▶ `write.csv2()`

## Writing data

- ▶ `write.table()` is like `read.table()` in reverse.
- ▶ Likewise, `write.csv()` and `write.csv2()` are wrappers around `write.table()` differing only in the default values.

## Writing data

These are the most important default values of `write.table()`:

```
write.table(x, file = "", quote = TRUE, sep = " ",  
           na = "NA", dec = ".", row.names = TRUE,  
           col.names = TRUE)
```

- ▶ `write.table()` prints object `x` (after converting it to a data frame if it is not one nor a matrix) to a local file.
- ▶ In the `file` argument you should indicate a character string naming the output file (with the respective path, in case you don't want to store the file in the working directory).
- ▶ The `quote` argument indicates if character and factor columns should be surrounded by double quotes in the output.

## Writing data

- ▶ The `sep` argument stipulates how the values of the table should be separated in the output file.
- ▶ The `na` arguments indicates how the NA values should be displayed in the output file
- ▶ `row.names` and `col.names` indicate whether or not row names and column names should be printed to the output file.

## Writing data

- ▶ `write.csv` uses `","` as separator and `"."` as decimal point
- ▶ `write.csv2` uses `";"` as separator and `","` as decimal point



# Writing data

By default:

- ▶ In your working directory, create a new folder called outputs.

## Writing data

Write the mtcars1 dataframe to a CSV file with row names and column names.

```
write.csv(x = mtcars1,  
          file = "outputs/mtcars1.csv")
```

Or:

```
write.table(x = mtcars1,  
            file = "outputs/mtcars1.csv",  
            sep = ";")
```

## Writing data

Write the `mtcars1` dataframe to a CSV file with row names, column names, and values separated with ";". Use ",", as the decimal separator.

```
write.csv2(x = mtcars1,  
           file = "outputs/mtcars1_v2.csv")
```

Or:

```
write.table(x = mtcars1,  
           file = "outputs/mtcars1_v2.csv",  
           sep = ";",  
           dec = ",")
```

## Writing data

Write the `air0` dataframe to a tab delimited file (.txt). Include column names but not row names, and identify the NAs with "-".

```
write.table(air0, file = "outputs/air0.txt",  
            sep = "\t",  
            row.names = FALSE,  
            na = "-")
```

## Writing data

Write the chick dataframe to a text file without row names nor column names. Table values in the output should be separated by "|".

```
write.table(chick, file = "outputs/chick.txt",  
            sep = "|",  
            row.names = FALSE,  
            col.names = FALSE)
```