

Primeras exploraciones:

Importando y limpiando datos

Guillermo de Anda-Jáuregui y Cristóbal Fresno

Instituto Nacional de Medicina Genómica

2019

Problema

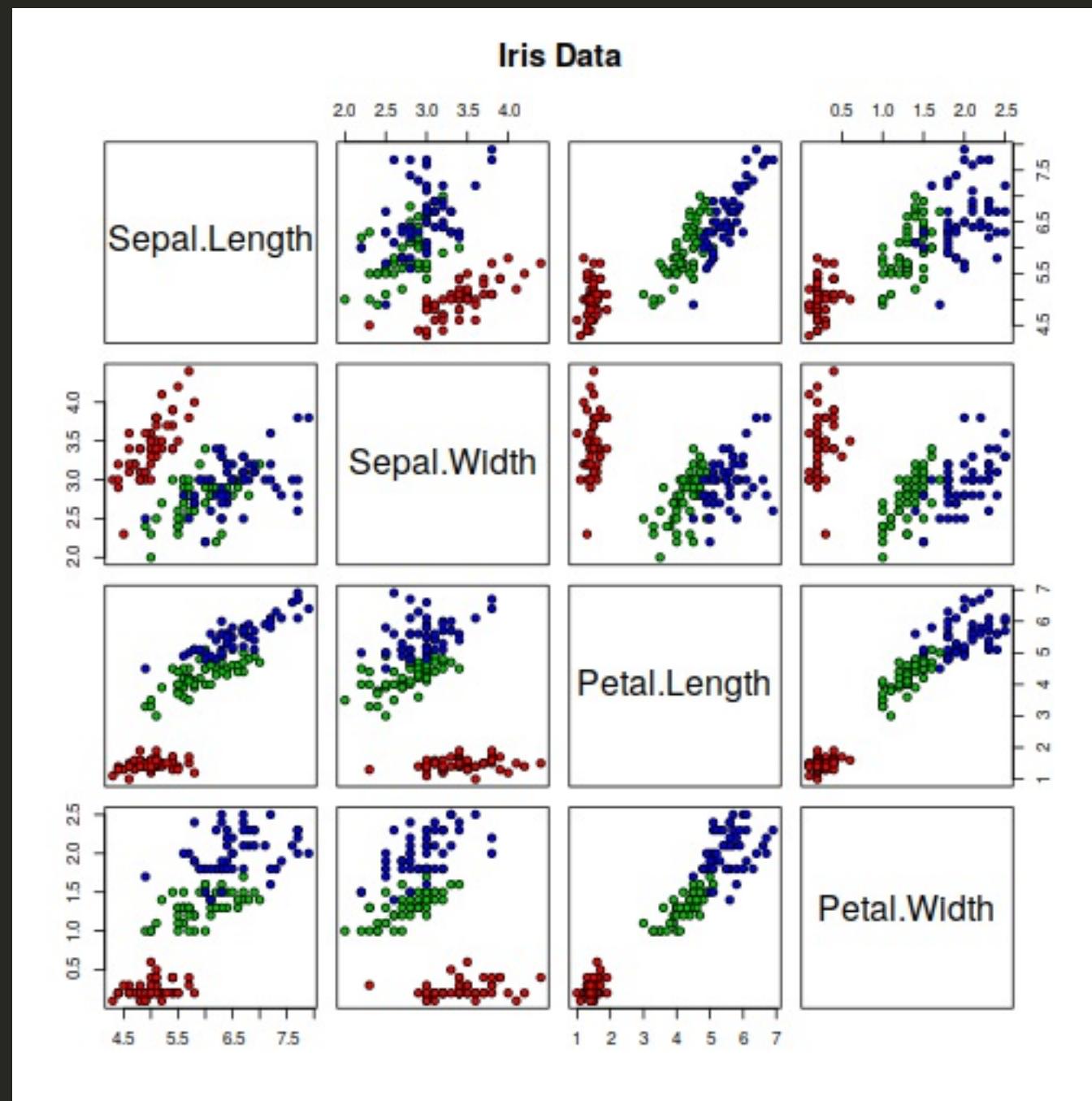
Distinguir entre:



Iris dataset

Fisher, Anderson 1936

```
##   Sepal.Length Sepal.Width Petal.Length Petal.Width Species
## 1          5.1       3.5        1.4       0.2  setosa
## 2          4.9       3.0        1.4       0.2  setosa
## 3          4.7       3.2        1.3       0.2  setosa
## 4          4.6       3.1        1.5       0.2  setosa
## 5          5.0       3.6        1.4       0.2  setosa
## 6          5.4       3.9        1.7       0.4  setosa
```



Leyendo datos tabulares

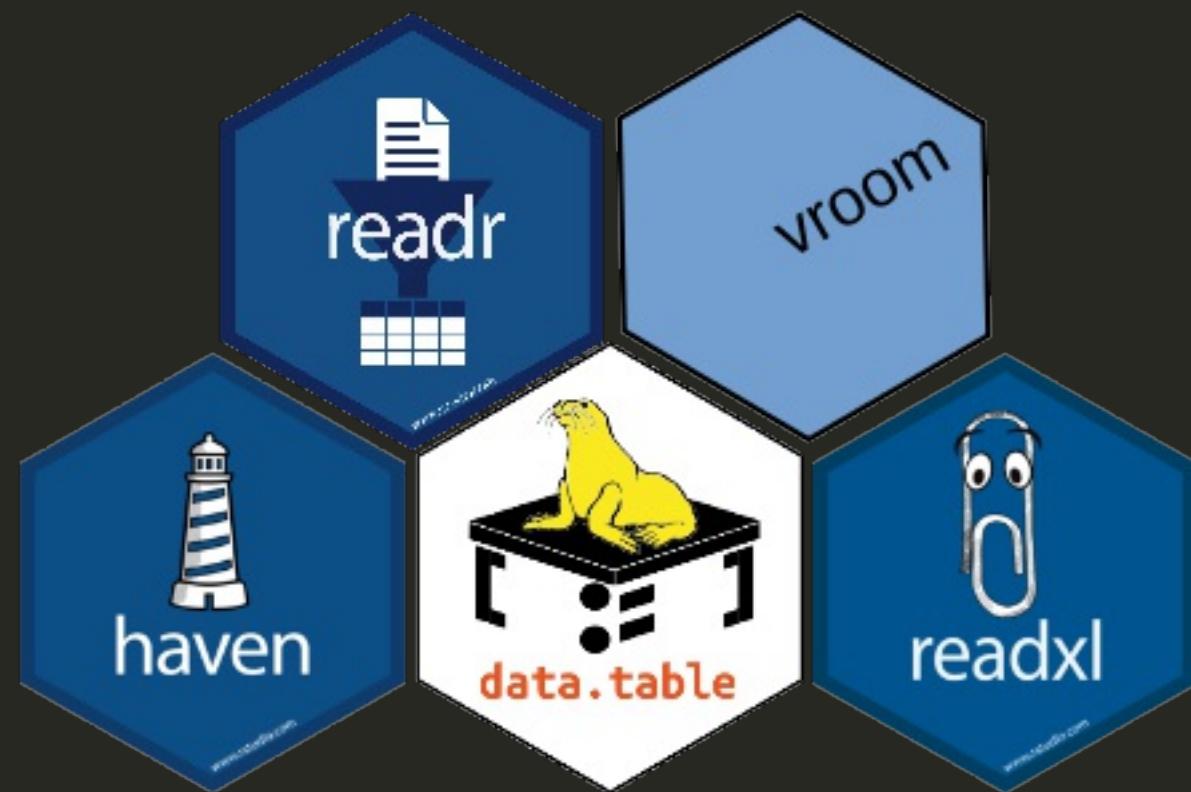
- *Iris* es un conjunto de datos que viene por default en R...
- Para el ejercicio vamos a leer un archivo externo de *Iris*

--

¿Cómo leer?

Hay varias opciones...

¿Cómo leer?



R base

- Entrada: archivo de texto

- Salida: un data frame

Estructura tabular (rectangular), diferentes tipos de datos.

R base

```
read.table(file = "data/iris_01.csv")
```

```
##                                     V1
## 1 Sepal.Length,Sepal.Width,Petal.Length,Petal.Width,Species
## 2                               5.1,3.5,1.4,0.2, setosa
## 3                               4.9,3,1.4,0.2, setosa
## 4                               4.7,3.2,1.3,0.2, setosa
## 5                               4.6,3.1,1.5,0.2, setosa
## 6                               5,3.6,1.4,0.2, setosa
## 7                               5.4,3.9,1.7,0.4, setosa
## 8                               4.6,3.4,1.4,0.3, setosa
## 9                               5,3.4,1.5,0.2, setosa
## 10                             4.4,2.9,1.4,0.2, setosa
## 11                             4.9,3.1,1.5,0.1, setosa
## 12                             5.4,3.7,1.5,0.2, setosa
## 13                             4.8,3.4,1.6,0.2, setosa
## 14                             4.8,3,1.4,0.1, setosa
## 15                             4.3,3,1.1,0.1, setosa
## 16                             5.8,4,1.2,0.2, setosa
## 17                             5.7,4.4,1.5,0.4, setosa
## 18                             5.4,3.9,1.3,0.4, setosa
## 19                             5.1,3.5,1.4,0.3, setosa
## 20                             5.7,3.8,1.7,0.3, setosa
## 21                             5.1,3.8,1.5,0.3, setosa
```

Oh, Oh!

```
read.table(file = "data/iris_01.csv", #el archivo
           header = , #¿tiene cabezal?
           sep = , #¿cuál separador?
           quote = , #¿Valores con comillas?
                      #nombres de ...
           row.names = , #renglones
           col.names = , #columnas
           #renglones que no se lean...
           skip = #al principio
           nrows = #cuantos leer despues
                      #y el famoso
           stringsAsFactors = #...
                      )
```

Un breve paréntesis...

Tipos de datos en R

Ordenados de menos a más flexible:

- Lógico

```
my_logic = c(TRUE, TRUE, FALSE)  
my_logic
```

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

```
typeof(my_logic)
```

```
## [1] "logical"
```

Tipos de datos en R

Ordenados de menos a más flexible:

- Entero

```
my_integer = c(7L, 2L, 5L)  
my_integer
```

```
## [1] 7 2 5
```

```
typeof(my_integer)
```

```
## [1] "integer"
```

Tipos de datos en R

Ordenados de menos a más flexible:

- Doble

```
my_double = c(7.25, 3, 14)  
my_double
```

```
## [1] 7.25 3.00 14.00
```

```
typeof(my_double)
```

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

Tipos de datos en R

Ordenados de menos a más flexible:

- "Carácter"

```
my_char = c("Esto", "Es", "1", "de", "muchos", "textos", "TRUE")  
my_char
```

```
## [1] "Esto"    "Es"      "1"       "de"      "muchos"  "textos"  "TRUE"
```

```
typeof(my_char)
```

```
## [1] "character"
```

Los temidos factores

- Los factores son vectores atómicos que contienen **valores predefinidos**
- Permiten almacenar variables categóricas

```
my_factor <- factor(c("interno", "externo", "externo", "interno"))
my_factor
```

```
## [1] interno externo externo interno
## Levels: externo interno
```

```
levels(my_factor)
```

```
## [1] "externo" "interno"
```

```
class(my_factor)
```

```
## [1] "factor"
```

Los temidos factores

- Los factores son en realidad numéricos

```
typeof(my_factor)
```

```
## [1] "integer"
```

```
as.numeric(my_factor)
```

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

Más adelante aprenderemos a manipular factores


```
my_iris <-
read.table(file = "data/iris_01.csv", #el archivo
          header = TRUE, #¿tiene cabezal?
          sep = ",", #¿cuál separador?
          stringsAsFactors = TRUE#...
                    )

head(my_iris)
```

```
##      Sepal.Length Sepal.Width Petal.Length Petal.Width Species
## 1            5.1         3.5          1.4         0.2  setosa
## 2            4.9         3.0          1.4         0.2  setosa
## 3            4.7         3.2          1.3         0.2  setosa
## 4            4.6         3.1          1.5         0.2  setosa
## 5            5.0         3.6          1.4         0.2  setosa
## 6            5.4         3.9          1.7         0.4  setosa
```

Veamos nuestros datos

```
str(my_iris)
```

```
## 'data.frame':    150 obs. of  5 variables:  
## $ Sepal.Length: num  5.1 4.9 4.7 4.6 5 5.4 4.6 5 4.4 4.9 ...  
## $ Sepal.Width : num  3.5 3 3.2 3.1 3.6 3.9 3.4 3.4 2.9 3.1 ...  
## $ Petal.Length: num  1.4 1.4 1.3 1.5 1.4 1.7 1.4 1.5 1.4 1.5 ...  
## $ Petal.Width : num  0.2 0.2 0.2 0.2 0.2 0.4 0.3 0.2 0.2 0.1 ...  
## $ Species      : Factor w/ 3 levels "setosa","versicolor",...: 1 1 1 1 1 1 1 1 1 1
```

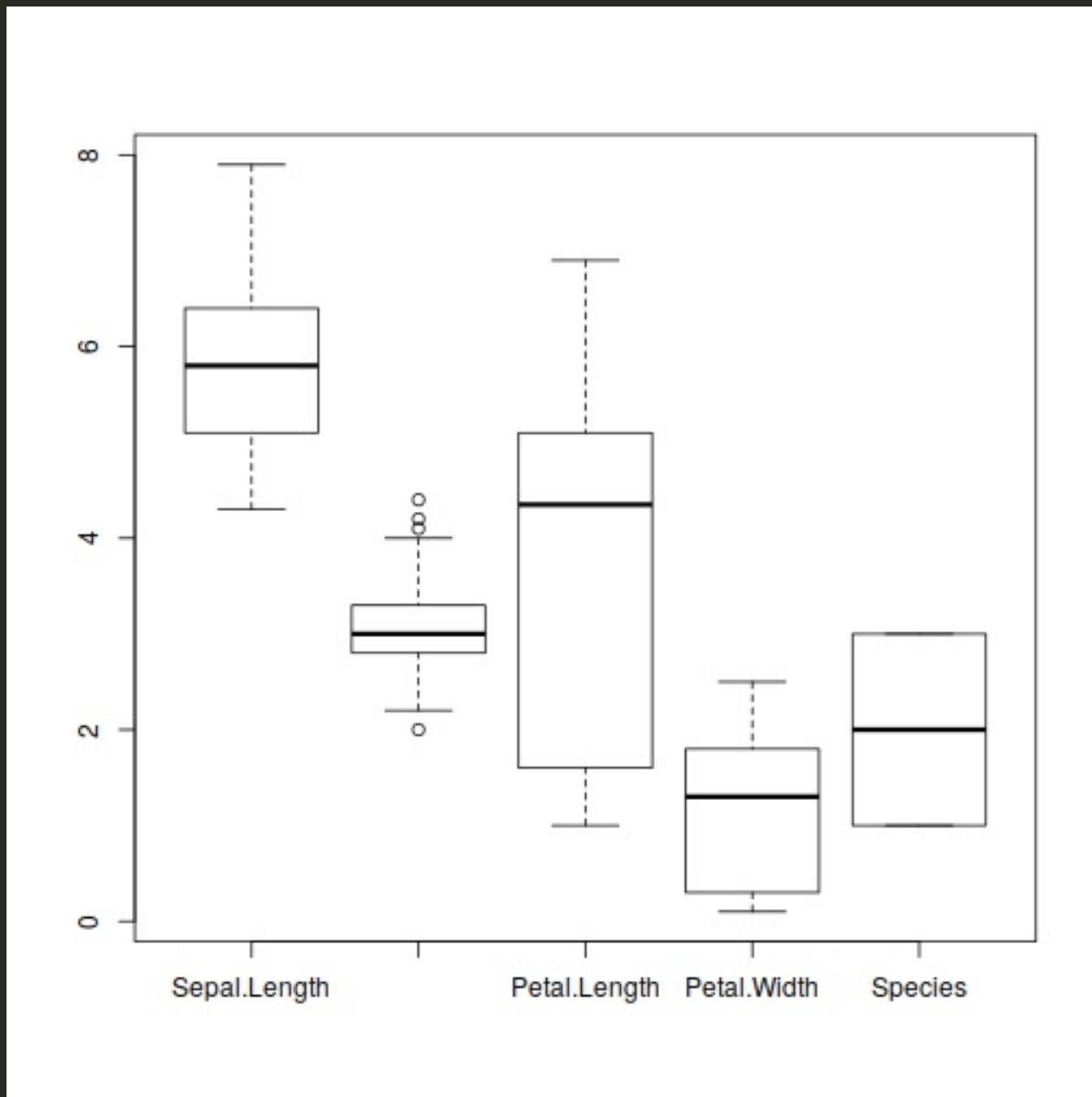
Veamos nuestros datos

```
summary(my_iris)
```

```
##   Sepal.Length   Sepal.Width   Petal.Length   Petal.Width
##   Min.   :4.300   Min.   :2.000   Min.   :1.000   Min.   :0.100
##   1st Qu.:5.100   1st Qu.:2.800   1st Qu.:1.600   1st Qu.:0.300
##   Median :5.800   Median :3.000   Median :4.350   Median :1.300
##   Mean    :5.843   Mean    :3.057   Mean    :3.758   Mean    :1.199
##   3rd Qu.:6.400   3rd Qu.:3.300   3rd Qu.:5.100   3rd Qu.:1.800
##   Max.    :7.900   Max.    :4.400   Max.    :6.900   Max.    :2.500
##                               ##          Species
##                               ##  setosa     :50
##                               ##  versicolor:50
##                               ##  virginica :50
##                               ##
##                               ##
##                               ##
```

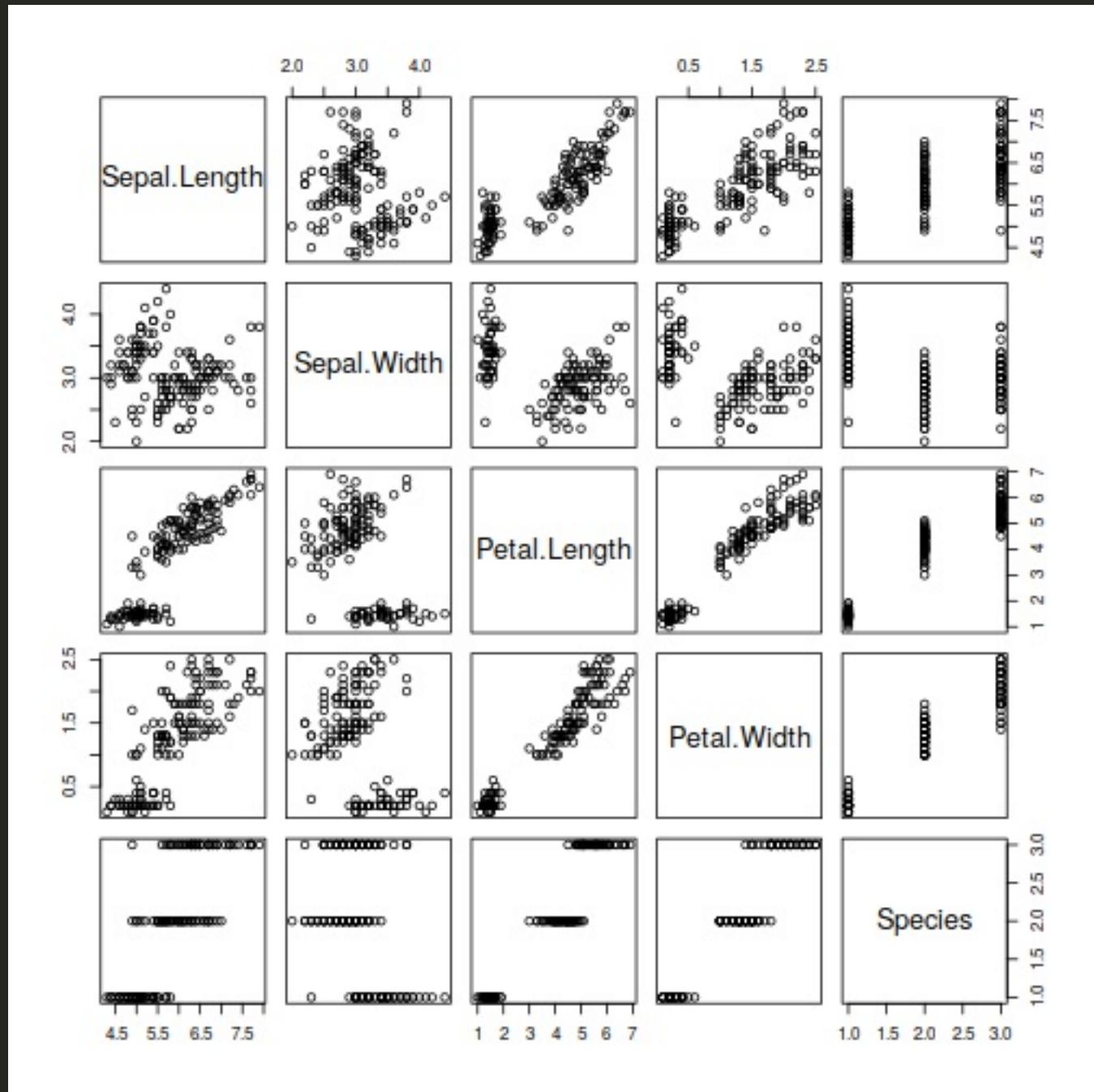
Veamos nuestros datos

```
boxplot(my_iris)
```



Veamos nuestros datos

```
pairs(my_iris)
```



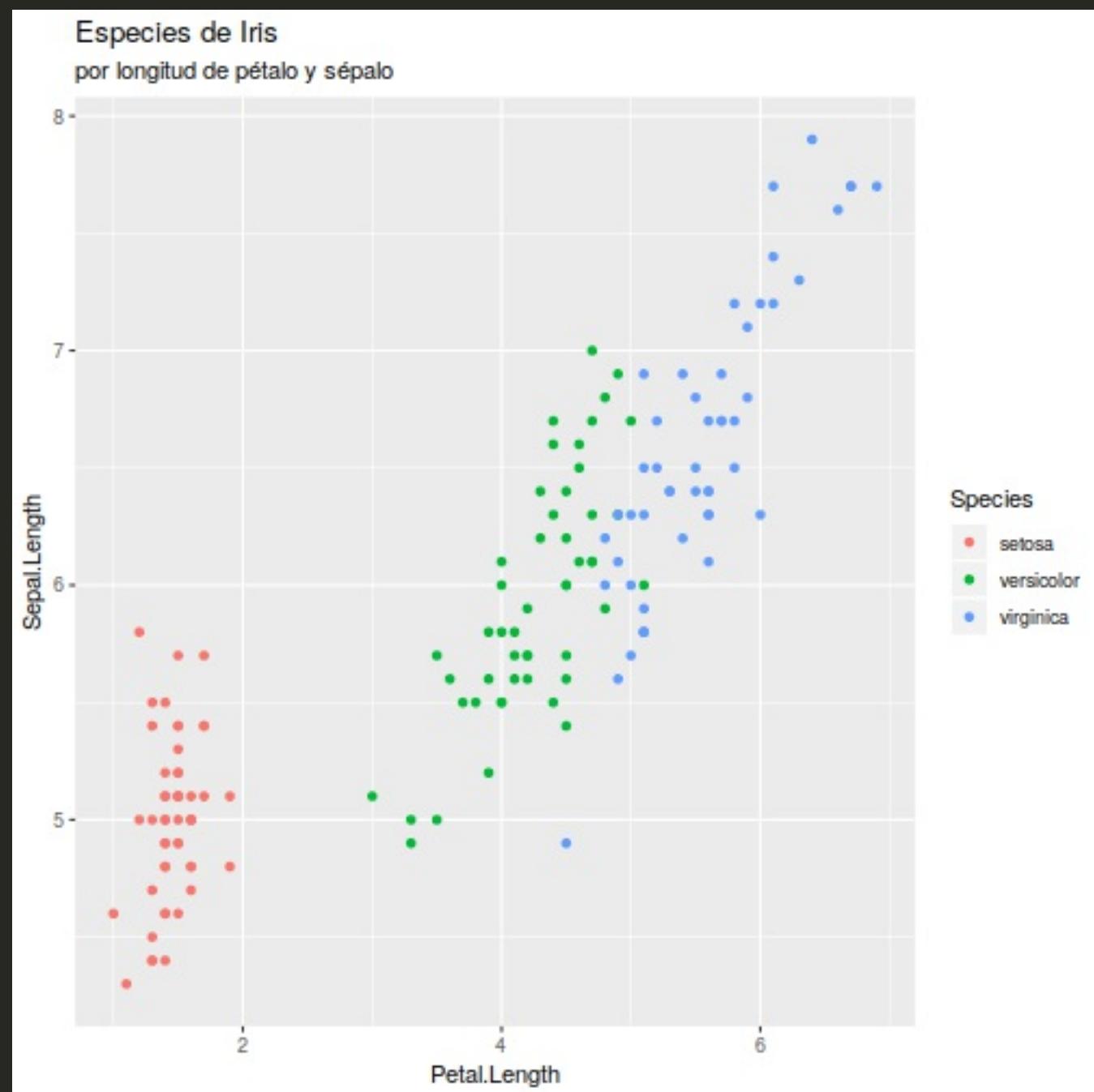
Tratemos de ver si se separan ...

```
my_iris %>%
  ggplot(aes(x = Petal.Length,
              y = Sepal.Length,
              colour = Species)) +
  geom_point() +
  ggtitle('Especies de Iris',
  subtitle = "por longitud de pétalo y sépalo")
```

Nota: el código para graficar con ggplot lo revisaremos con más detalle durante el curso

- Noten el uso de %>% ... este es el operador *pipe* del paquete magrittr; lo usaremos mucho

Tratemos de ver si se separan ...



¿Dudas?

Ejercicio: mtcars



Ejercicio: mtcars

```
mtcars %>% head
```

```
##          mpg cyl disp  hp drat    wt  qsec vs am gear carb
## Mazda RX4     21.0   6 160 110 3.90 2.620 16.46  0  1    4    4
## Mazda RX4 Wag 21.0   6 160 110 3.90 2.875 17.02  0  1    4    4
## Datsun 710    22.8   4 108  93 3.85 2.320 18.61  1  1    4    1
## Hornet 4 Drive 21.4   6 258 110 3.08 3.215 19.44  1  0    3    1
## Hornet Sportabout 18.7   8 360 175 3.15 3.440 17.02  0  0    3    2
## Valiant       18.1   6 225 105 2.76 3.460 20.22  1  0    3    1
```

Ejercicio: mtcars

- 1) ¿Cuántas variables y observaciones tiene el conjunto de datos mtcars?
- 2) Encuentre una pareja de variables tales que los autos se separen por el número de cilindros (cyl)

Pistas:

- Considere usar las funciones dim, ncol, nrow
- Si quiero agrupar por una variable, me conviene que esa variable sea un factor; si no lo es, as.factor puede convertirla...

Leer con Readr



El paquete **Readr** permite leer muchos tipos de datos rectangulares

- `read_delim` : datos delimitados (cualquier separador)
- `read_csv` : datos separados por comas
- `read_tsv` : datos separados por tabuladores
- `read_table` : datos separados por espacio en blanco (espacios, tabs...)

Leer con Readr

Ejemplo

```
my_iris_tbl <-  
readr::read_csv(file = "data/iris_01.csv")
```

```
## Parsed with column specification:  
## cols(  
##   Sepal.Length = col_double(),  
##   Sepal.Width = col_double(),  
##   Petal.Length = col_double(),  
##   Petal.Width = col_double(),  
##   Species = col_character()  
## )
```

¡Readr tomo varias decisiones por nosotros!

Veamos como se ve my_iris_tbl

my_iris_tbl

```
## # A tibble: 150 x 5
##   Sepal.Length Sepal.Width Petal.Length Petal.Width Species
##       <dbl>        <dbl>        <dbl>        <dbl>      <chr>
## 1         5.1         3.5         1.4         0.2  setosa
## 2         4.9         3.0         1.4         0.2  setosa
## 3         4.7         3.2         1.3         0.2  setosa
## 4         4.6         3.1         1.5         0.2  setosa
## 5         5.0         3.6         1.4         0.2  setosa
## 6         5.4         3.9         1.7         0.4  setosa
## 7         4.6         3.4         1.4         0.3  setosa
## 8         5.0         3.4         1.5         0.2  setosa
## 9         4.4         2.9         1.4         0.2  setosa
## 10        4.9         3.1         1.5         0.1 setosa
## # ... with 140 more rows
```

--

¡Se ve diferente!

¿Qué son esas notas ahí?

Tibbles



Una versión más moderna del `data.frame`... con algunas ventajas

--> print más limpio:

- Solo algunos renglones (no necesitamos `head`)
- Indica dimensiones y tipo de variables (no necesitamos `str`, `dim`)

Tibbles



- > No obliga a que los strings sean factores por default
- > No usa rownames (y eso es algo bueno)

Los tibbles pueden hacer todo lo que los data frames tradicionales

```
my_iris_tbl %>% str
```

```
## Classes 'spec_tbl_df', 'tbl_df', 'tbl' and 'data.frame': 150 obs. of 5 variables:
##   $ Sepal.Length: num 5.1 4.9 4.7 4.6 5 5.4 4.6 4.6 5 4.4 ...
##   $ Sepal.Width : num 3.5 3 3.2 3.1 3.6 3.9 3.4 3.4 2.9 3.1 ...
##   $ Petal.Length: num 1.4 1.4 1.3 1.5 1.4 1.7 1.4 1.5 1.4 1.5 ...
##   $ Petal.Width : num 0.2 0.2 0.2 0.2 0.2 0.4 0.3 0.2 0.2 0.1 ...
##   $ Species      : chr "setosa" "setosa" "setosa" "setosa" ...
##     - attr(*, "spec")=
##       .. cols(
##         .. Sepal.Length = col_double(),
##         .. Sepal.Width = col_double(),
##         .. Petal.Length = col_double(),
##         .. Petal.Width = col_double(),
##         .. Species = col_character()
##           .. )
```

¡Pero cuidado! Los métodos de tibble NO adivinan cosas como los de base

```
my_iris_tbl %>% boxplot()
```

```
## Error in x[floor(d)] + x[ceiling(d)]: non-numeric argument to binary operator
```

... y eso es algo bueno

Pregunta: ¿Por qué falló ese boxplot? ¿Cómo solucionarlo?

data.table



- El paquete `data.table` está orientado a big data
- Es una filosofía distinta a la de Tidy; inspirada en SQL
- Bastante más rápido; sintaxis es más espartana

data.table

```
my_iris_dt = data.table::fread(input = "data/iris_01.csv")  
my_iris_dt
```

	Sepal.Length	Sepal.Width	Petal.Length	Petal.Width	Species
## 1:	5.1	3.5	1.4	0.2	setosa
## 2:	4.9	3.0	1.4	0.2	setosa
## 3:	4.7	3.2	1.3	0.2	setosa
## 4:	4.6	3.1	1.5	0.2	setosa
## 5:	5.0	3.6	1.4	0.2	setosa
## ---					
## 146:	6.7	3.0	5.2	2.3	virginica
## 147:	6.3	2.5	5.0	1.9	virginica
## 148:	6.5	3.0	5.2	2.0	virginica
## 149:	6.2	3.4	5.4	2.3	virginica
## 150:	5.9	3.0	5.1	1.8	virginica

data.table

```
my_iris_dt %>% str
```

```
## Classes 'data.table' and 'data.frame': 150 obs. of 5 variables:  
## $ Sepal.Length: num 5.1 4.9 4.7 4.6 5 5.4 4.6 5 4.4 4.9 ...  
## $ Sepal.Width : num 3.5 3 3.2 3.1 3.6 3.9 3.4 3.4 2.9 3.1 ...  
## $ Petal.Length: num 1.4 1.4 1.3 1.5 1.4 1.7 1.4 1.5 1.4 1.5 ...  
## $ Petal.Width : num 0.2 0.2 0.2 0.2 0.2 0.4 0.3 0.2 0.2 0.1 ...  
## $ Species      : chr "setosa" "setosa" "setosa" "setosa" ...  
## - attr(*, ".internal.selfref")=<externalptr>
```

Vroom

¡Lo mejor de dos mundos!

- Súper rápido
- Eficiente con la memoria
- Devuelve tibbles --> tidy

Vroom

¡Lo mejor de dos mundos!

```
my_iris_vroom = vroom::vroom(file = "data/iris_01.csv")  
  
## Observations: 150  
## Variables: 5  
## chr [1]: Species  
## dbl [4]: Sepal.Length, Sepal.Width, Petal.Length, Petal.Width  
##  
## Call `spec()` for a copy-pastable column specification  
## Specify the column types with `col_types` to quiet this message
```

Vroom

¡Lo mejor de dos mundos!

```
my_iris_vroom
```

```
## # A tibble: 150 x 5
##   Sepal.Length Sepal.Width Petal.Length Petal.Width Species
##       <dbl>      <dbl>       <dbl>       <dbl>   <chr>
## 1         5.1        3.5        1.4        0.2  setosa
## 2         4.9        3.0        1.4        0.2  setosa
## 3         4.7        3.2        1.3        0.2  setosa
## 4         4.6        3.1        1.5        0.2  setosa
## 5         5.0        3.6        1.4        0.2  setosa
## 6         5.4        3.9        1.7        0.4  setosa
## 7         4.6        3.4        1.4        0.3  setosa
## 8         5.0        3.4        1.5        0.2  setosa
## 9         4.4        2.9        1.4        0.2  setosa
## 10        4.9        3.1        1.5        0.1 setosa
## # ... with 140 more rows
```

¿Qué otros paquetes podría necesitar?

¡Datos de Excel!

--

readxl

readxl

Este paquete leer datos de Excel, en diferentes formatos (xls, xlsx...)

readxl

```
readxl::read_xlsx(path = "data/datasets_clasicos.xlsx")
```

```
## # A tibble: 150 x 5
##   Sepal.Length Sepal.Width Petal.Length Petal.Width Species
##       <dbl>      <dbl>       <dbl>       <dbl>   <chr>
## 1         5.1        3.5        1.4        0.2  setosa
## 2         4.9        3.0        1.4        0.2  setosa
## 3         4.7        3.2        1.3        0.2  setosa
## 4         4.6        3.1        1.5        0.2  setosa
## 5         5.0        3.6        1.4        0.2  setosa
## 6         5.4        3.9        1.7        0.4  setosa
## 7         4.6        3.4        1.4        0.3  setosa
## 8         5.0        3.4        1.5        0.2  setosa
## 9         4.4        2.9        1.4        0.2  setosa
## 10        4.9        3.1        1.5        0.1 setosa
## # ... with 140 more rows
```

--

¿Qué tal si tengo más de una hoja?

readxl

```
readxl::read_xlsx(path = "data/datasets_clasicos.xlsx",  
                   sheet = 1)
```

```
## # A tibble: 150 x 5  
##   Sepal.Length Sepal.Width Petal.Length Petal.Width Species  
##       <dbl>        <dbl>        <dbl>        <dbl>      <chr>  
## 1         5.1         3.5         1.4         0.2  setosa  
## 2         4.9         3           1.4         0.2  setosa  
## 3         4.7         3.2          1.3         0.2  setosa  
## 4         4.6         3.1          1.5         0.2  setosa  
## 5           5          3.6          1.4         0.2  setosa  
## 6         5.4         3.9          1.7         0.4  setosa  
## 7         4.6         3.4          1.4         0.3  setosa  
## 8           5          3.4          1.5         0.2  setosa  
## 9         4.4         2.9          1.4         0.2  setosa  
## 10        4.9         3.1          1.5         0.1  setosa  
## # ... with 140 more rows
```

readxl

```
readxl::read_xlsx(path = "data/datasets_clasicos.xlsx",  
                   sheet = 2)
```

```
## # A tibble: 32 x 12  
##   ...1     mpg     cyl   disp     hp   drat     wt   qsec     vs     am   gear   carb  
##   <chr>   <dbl>   <dbl>  <dbl>   <dbl>   <dbl>   <dbl>  <dbl>   <dbl>   <dbl>   <dbl>   <dbl>  
## 1 Mazda...  21      6    160    110    3.9    2.62   16.5     0      1     4     4  
## 2 Mazda...  21      6    160    110    3.9    2.88   17.0     0      1     4     4  
## 3 Datsu...  22.8    4    108     93    3.85   2.32   18.6     1      1     4     1  
## 4 Horne...  21.4    6    258    110    3.08   3.22   19.4     1      0     3     1  
## 5 Horne...  18.7    8    360    175    3.15   3.44   17.0     0      0     3     2  
## 6 Valia...  18.1    6    225    105    2.76   3.46   20.2     1      0     3     1  
## 7 Duste...  14.3    8    360    245    3.21   3.57   15.8     0      0     3     4  
## 8 Merc ...  24.4    4    147.    62     3.69   3.19    20       1      0     4     2  
## 9 Merc ...  22.8    4    141.    95     3.92   3.15   22.9     1      0     4     2  
## 10 Merc ... 19.2    6    168.    123    3.92   3.44   18.3     1      0     4     4  
## # ... with 22 more rows
```

readxl

También podemos escoger por nombre

```
readxl::read_xlsx(path = "data/datasets_clasicos.xlsx",  
                   sheet = "USArrests")
```

```
## # A tibble: 50 x 5  
##   state      Murder Assault UrbanPop    Rape  
##   <chr>     <dbl>    <dbl>     <dbl>   <dbl>  
## 1 Alabama     13.2     236       58   21.2  
## 2 Alaska      10.0     263       48   44.5  
## 3 Arizona      8.1     294       80   31.0  
## 4 Arkansas     8.8     190       50   19.5  
## 5 California    9.0     276       91   40.6  
## 6 Colorado      7.9     204       78   38.7  
## 7 Connecticut   3.3     110       77   11.1  
## 8 Delaware      5.9     238       72   15.8  
## 9 Florida      15.4     335       80   31.9  
## 10 Georgia     17.4     211       60   25.8  
## # ... with 40 more rows
```

```
readxl::read_xlsx(path = "data/datasets_clasicos.xlsx",
                   sheet = 4)

## # A tibble: 51 x 5
##   `#this is a dataset of arrests in the USA` ...2     ...3     ...4     ...5
##   <chr>          <chr>    <chr>    <chr>    <chr>
## 1 state           Murder   Assault  UrbanPop Rape
## 2 Alabama         13.2    236      58       21.2
## 3 Alaska          10      263      48       44.5
## 4 Arizona         8.1     294      80       31
## 5 Arkansas        8.8     190      50       19.5
## 6 California      9       276      91       40.6
## 7 Colorado        7.9     204      78       38.7
## 8 Connecticut     3.3     110      77       11.1
## 9 Delaware        5.9     238      72       15.8
## 10 Florida        15.4    335      80       31.9
## # ... with 41 more rows
```

Se ve mal...

Tenemos que "saltarnos" la primera línea

```
readxl::read_xlsx(path = "data/datasets_clasicos.xlsx",
                    sheet = 4,
                    skip = 1)
```

```
## # A tibble: 50 x 5
##   state      Murder Assault UrbanPop    Rape
##   <chr>     <dbl>   <dbl>     <dbl>   <dbl>
## 1 Alabama     13.2     236       58   21.2
## 2 Alaska      10        263       48   44.5
## 3 Arizona      8.1      294       80   31
## 4 Arkansas     8.8      190       50   19.5
## 5 California    9        276       91   40.6
## 6 Colorado     7.9      204       78   38.7
## 7 Connecticut   3.3      110       77   11.1
## 8 Delaware      5.9      238       72   15.8
## 9 Florida      15.4     335       80   31.9
## 10 Georgia     17.4     211       60   25.8
## # ... with 40 more rows
```

¡Mucho mejor!

¿que más tenemos en ese archivo?

Limpieza de headers: Janitor

A veces tenemos datos con cabezales que no son limpios: espacios, símbolos raros, MayúsculaS MAL usadas...

Cuando eso pasa, podemos usar janitor::clean_names

```
my_genes <- vroom::vroom("data/genes_500.txt")
```

```
## Observations: 499
## Variables: 7
## chr [4]: Gene stable ID, Chromosome/scaffold name, Gene type, HGNC symbol
## dbl [3]: Gene start (bp), Gene end (bp), Gene % GC content
##
## Call `spec()` for a copy-pastable column specification
## Specify the column types with `col_types` to quiet this message
```

```
my_genes
```

```
## # A tibble: 499 x 7
##   `Gene stable ID` `Chromosome/sca... `Gene start (bp... `Gene end (bp)` 
##   <chr>              <chr>                  <dbl>                <dbl>
## 1 ENSG00000210049  MT                      577                 647
## 2 ENSG00000211459  MT                      648                 1601
## 3 ENSG00000210077  MT                     1602                1670
## 4 ENSG00000210082  MT                     1671                3229
## 5 ENSG00000209082  MT                     3230                3304
## 6 ENSG00000198888  MT                     3307                4262
## 7 ENSG00000210100  MT                     4263                4331
## 8 ENSG00000210107  MT                     4329                4400
## 9 ENSG00000210112  MT                     4402                4469
## 10 ENSG00000198763 MT                    4470                5511
## # ... with 489 more rows, and 3 more variables: `Gene % GC content` <dbl>,
## #       `Gene type` <chr>, `HGNC symbol` <chr>
```

```
my_genes <- janitor::clean_names(dat = my_genes,  
                                case = "snake")  
my_genes
```

```
## # A tibble: 499 x 7  
##   gene_stable_id chromosome_scaf... gene_start_bp gene_end_bp  
##   <chr>          <chr>            <dbl>        <dbl>  
## 1 ENSG000002100... MT                 577         647  
## 2 ENSG000002114... MT                 648        1601  
## 3 ENSG000002100... MT                1602        1670  
## 4 ENSG000002100... MT                1671        3229  
## 5 ENSG000002090... MT                3230        3304  
## 6 ENSG000001988... MT                3307        4262  
## 7 ENSG000002101... MT                4263        4331  
## 8 ENSG000002101... MT                4329        4400  
## 9 ENSG000002101... MT                4402        4469  
## 10 ENSG000001987... MT               4470        5511  
## # ... with 489 more rows, and 3 more variables:  
## #   gene_percent_gc_content <dbl>, gene_type <chr>, hgnc_symbol <chr>
```

Cases:

snake : snake_case

screaming_snake : SERPIENTE_GRITONA

lower_camel : camello_Jorobado

upper_camel : Dromedario_Multijorobado

lower_upper : algoASI

upper_lower : ASItambien

¿Otros tipos de datos?

- haven
- googledrive
- googlesheets ...

¿Datos binarios?

`saveRDS` --> `RDS` --> Guarda un objeto

`save` --> `RData` --> Guarda objeto(s) en el espacio de trabajo, nombrado(s)

`save.image` --> `RData` --> Guarda TODOS los objetos en el espacio de trabajo, nombrados

Los podemos leer con `load` (carga todo con los nombres con los que se escribió)

o `readRDS` (lo asignamos)

```
    otro_iris_mas <- iris
  save(otro_iris_mas, file = "results/iris_forever.RData")
    rm(otro_iris_mas)
      otro_iris_mas
```

```
## Error in eval(expr, envir, enclos): object 'otro_iris_mas' not found
```

```
load("results/iris_forever.RData")
  otro_iris_mas
```

	Sepal.Length	Sepal.Width	Petal.Length	Petal.Width	Species
## 1	5.1	3.5	1.4	0.2	setosa
## 2	4.9	3.0	1.4	0.2	setosa
## 3	4.7	3.2	1.3	0.2	setosa
## 4	4.6	3.1	1.5	0.2	setosa
## 5	5.0	3.6	1.4	0.2	setosa
## 6	5.4	3.9	1.7	0.4	setosa
## 7	4.6	3.4	1.4	0.3	setosa
## 8	5.0	3.4	1.5	0.2	setosa
## 9	4.4	2.9	1.4	0.2	setosa
## 10	4.9	3.1	1.5	0.1	setosa
## 11	5.4	3.7	1.5	0.2	setosa
## 12	4.8	3.4	1.6	0.2	setosa
## 13	4.8	3.0	1.4	0.1	setosa
## 14	4.3	3.0	1.1	0.1	setosa
## 15	5.8	4.0	1.2	0.2	setosa
## 16	5.7	4.4	1.5	0.4	setosa
## 17	5.4	3.9	1.3	0.4	setosa

```
otro_iris_mas <- iris  
save.image(file = "results/sesion_01.RData")
```

Borramos todo

```
otro_iris_mas <- iris  
rm(list=ls()) # ¡cuidado con este comando!  
otro_iris_mas
```

```
## Error in eval(expr, envir, enclos): object 'otro_iris_mas' not found
```

```
load("results/sesion_01.RData")
otro_iris_mas %>% head
```

```
##   Sepal.Length Sepal.Width Petal.Length Petal.Width Species
## 1           5.1         3.5          1.4         0.2  setosa
## 2           4.9         3.0          1.4         0.2  setosa
## 3           4.7         3.2          1.3         0.2  setosa
## 4           4.6         3.1          1.5         0.2  setosa
## 5           5.0         3.6          1.4         0.2  setosa
## 6           5.4         3.9          1.7         0.4  setosa
```

```
my_iris_vroom
```

```
## # A tibble: 150 x 5
##   Sepal.Length Sepal.Width Petal.Length Petal.Width Species
##       <dbl>        <dbl>        <dbl>        <dbl>    <chr>
## 1           5.1         3.5          1.4         0.2  setosa
## 2           4.9         3.0          1.4         0.2  setosa
## 3           4.7         3.2          1.3         0.2  setosa
## 4           4.6         3.1          1.5         0.2  setosa
## 5           5.0         3.6          1.4         0.2  setosa
## 6           5.4         3.9          1.7         0.4  setosa
## 7           4.6         3.4          1.4         0.3  setosa
## 8           5.0         3.4          1.5         0.2  setosa
## 9           4.4         2.9          1.4         0.2  setosa
## 10          4.9         3.1          1.5         0.1 setosa
## # ... with 140 more rows
```

```
saveRDS(object = my_iris_vroom,  
        file = "results/my_iris_vroom.RDS"  
        )  
  
load("results/my_iris_vroom.RDS")
```

```
## Warning: file 'my_iris_vroom.RDS' has magic number 'X'  
##   Use of save versions prior to 2 is deprecated
```

```
## Error in load("results/my_iris_vroom.RDS"): bad restore file magic number (file ma
```

```
my_iris_vroom_again <- readRDS(file = "results/my_iris_vroom.RDS")
my_iris_vroom_again
```

```
## # A tibble: 150 x 5
##   Sepal.Length Sepal.Width Petal.Length Petal.Width Species
##       <dbl>      <dbl>       <dbl>       <dbl>   <chr>
## 1         5.1        3.5        1.4        0.2  setosa
## 2         4.9        3.0        1.4        0.2  setosa
## 3         4.7        3.2        1.3        0.2  setosa
## 4         4.6        3.1        1.5        0.2  setosa
## 5         5.0        3.6        1.4        0.2  setosa
## 6         5.4        3.9        1.7        0.4  setosa
## 7         4.6        3.4        1.4        0.3  setosa
## 8         5.0        3.4        1.5        0.2  setosa
## 9         4.4        2.9        1.4        0.2  setosa
## 10        4.9        3.1        1.5        0.1  setosa
## # ... with 140 more rows
```

Todas los paquetes de lectura tienen su versión de escritura

`read.table <--> write.table`

`data.table::fread <--> data.table::fwrite`

`vroom::vroom <--> vroom::vroom_write`

```
write.table(x = mtcars,
            file = "results/otro_mtcars.txt",
            sep = "\t",
            quote = FALSE,
            row.names = TRUE,
            col.names = TRUE)
```

-- ¿Cómo se ve el archivo? ¿Qué pasa si cambio los parámetros?

Ejercicios:

- Leer y manipular el archivo: `data/iris_esp.csv`
- Leer y manipular el archivo: `mundiales_femenil.xlsx`

¿Qué acciones realizaron en cada caso?