Data Frame

Daniel Eduardo Macias Estrada

9/5/2021

Data frames

Un data frame es una tabla de doble entrada, formada por variables en las columnas y observaciones de estas variables en las filas, de manera que cada fila contiene los valores de las variables para un mismo caso o un mismo individuo

- `data()`: para abrir una ventana con la lista de los objetos de datos a los que tenemos acceso en la sesión actual de R (los que lleva la instalación básica de R y los que aportan los paquetes que tengamos cargados.
  - Si entramos `data(package=.packages(all.available = TRUE))` obtendremos la lista de todos los objetos de datos a los que tenemos acceso, incluyendo los de los paquetes que tengamos instalados, pero que no estén cargados en la sesión actual

Obteniendo datos de un dataframe

Para evitar usar siempre el nombre del dataframe, podemos realizar una copia con una variable

Sin embargo al ser muchos elementos podemos hacer uso de las siguientes funciones

- `head(d.f, n)`: para mostrar las n primeras filas del data frame. Por defecto se muestran las 6 primeras filas
- `tail(d.f, n)`: para mostrar las n últimas filas del data frame. Por defecto se muestran las 6 últimas
- `str(d.f)`: para conocer la estructura global de un data frame
- `names(d.f)`: para producir un vector con los nombres de las columnas

```
df = iris
head(df, 5)
```

```
##   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
```

```r
tail(df, 5)
```

```
##     Sepal.Length Sepal.Width Petal.Length Petal.Width   Species
## 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
```

```r
str(df)
```

```
## '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 ...
```

```r
names(df)
```

```
## [1] "Sepal.Length" "Sepal.Width"  "Petal.Length" "Petal.Width"  "Species"
```

Otras funciones son

- `rownames(d.f)`: para producir un vector con los identificadores de las filas
  - R entiende siempre que estos identificadores son palabras, aunque sean números, de ahí que los imprima entre comillas.
- `colnames(d.f)`: para producir un vector con los identificadores de las columnas
- `dimnames(d.f)`: para producir una list formada por dos vectores (el de los identificadores de las filas y el de los nombres de las columnas)
- `nrow(d.f)`: para consultar el número de filas de un data frame
- `ncol(d.f)`: para consultar el número de columnas de un data frame
- `dim(d.f)`: para producir un vector con el número de filas y el de columnas

```r
rownames(df)
```

```
##   [1] "1"   "2"   "3"   "4"   "5"   "6"   "7"   "8"   "9"   "10"  "11"  "12"
##  [13] "13"  "14"  "15"  "16"  "17"  "18"  "19"  "20"  "21"  "22"  "23"  "24"
##  [25] "25"  "26"  "27"  "28"  "29"  "30"  "31"  "32"  "33"  "34"  "35"  "36"
##  [37] "37"  "38"  "39"  "40"  "41"  "42"  "43"  "44"  "45"  "46"  "47"  "48"
##  [49] "49"  "50"  "51"  "52"  "53"  "54"  "55"  "56"  "57"  "58"  "59"  "60"
##  [61] "61"  "62"  "63"  "64"  "65"  "66"  "67"  "68"  "69"  "70"  "71"  "72"
##  [73] "73"  "74"  "75"  "76"  "77"  "78"  "79"  "80"  "81"  "82"  "83"  "84"
##  [85] "85"  "86"  "87"  "88"  "89"  "90"  "91"  "92"  "93"  "94"  "95"  "96"
##  [97] "97"  "98"  "99"  "100" "101" "102" "103" "104" "105" "106" "107" "108"
## [109] "109" "110" "111" "112" "113" "114" "115" "116" "117" "118" "119" "120"
## [121] "121" "122" "123" "124" "125" "126" "127" "128" "129" "130" "131" "132"
## [133] "133" "134" "135" "136" "137" "138" "139" "140" "141" "142" "143" "144"
## [145] "145" "146" "147" "148" "149" "150"
```

```r
colnames(df)
```

```
## [1] "Sepal.Length" "Sepal.Width"  "Petal.Length" "Petal.Width"  "Species"
```

```r
dimnames(df)
```

```
## [[1]]
##   [1] "1"   "2"   "3"   "4"   "5"   "6"   "7"   "8"   "9"   "10"  "11"  "12"
##  [13] "13"  "14"  "15"  "16"  "17"  "18"  "19"  "20"  "21"  "22"  "23"  "24"
##  [25] "25"  "26"  "27"  "28"  "29"  "30"  "31"  "32"  "33"  "34"  "35"  "36"
##  [37] "37"  "38"  "39"  "40"  "41"  "42"  "43"  "44"  "45"  "46"  "47"  "48"
##  [49] "49"  "50"  "51"  "52"  "53"  "54"  "55"  "56"  "57"  "58"  "59"  "60"
##  [61] "61"  "62"  "63"  "64"  "65"  "66"  "67"  "68"  "69"  "70"  "71"  "72"
##  [73] "73"  "74"  "75"  "76"  "77"  "78"  "79"  "80"  "81"  "82"  "83"  "84"
##  [85] "85"  "86"  "87"  "88"  "89"  "90"  "91"  "92"  "93"  "94"  "95"  "96"
##  [97] "97"  "98"  "99"  "100" "101" "102" "103" "104" "105" "106" "107" "108"
## [109] "109" "110" "111" "112" "113" "114" "115" "116" "117" "118" "119" "120"
## [121] "121" "122" "123" "124" "125" "126" "127" "128" "129" "130" "131" "132"
## [133] "133" "134" "135" "136" "137" "138" "139" "140" "141" "142" "143" "144"
## [145] "145" "146" "147" "148" "149" "150"
##
## [[2]]
## [1] "Sepal.Length" "Sepal.Width"  "Petal.Length" "Petal.Width"  "Species"
```

```r
dim(df)
```

```
## [1] 150   5
```

- `df$nombre_variable`: para obtener una columna concreta de un dataframe.
  - El resultado será un vector o un factor, según cómo esté definida la columna dentro del dataframe
  - Las variables de un dataframe son internas, no están definidas en el entorno global de trabajo de R

```r
df$Sepal.Length[1:10] # 10 primeras longitudes de sepalos
```

```
##  [1] 5.1 4.9 4.7 4.6 5.0 5.4 4.6 5.0 4.4 4.9
```

Sub-data frames

- `d.f[n,m]`: para extraer "trozos" del data frame por filas y columnas (funciona exactamente igual que en matrices) donde n y m pueden definirse como:
  - Intervalos
  - condiciones
  - números naturales
  - no poner nada
  - Si sólo queremos definir la subtabla qudándonos con algunas variables, basta aplicar el nombre del data frame al vector de variables
  - Estas construcciones se pueden usar también para reordenar las filas o columnas

```r
df[1:10, ]
```

```
##    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
```

```r
df[1:10, 2:4]
```

```
##    Sepal.Width Petal.Length Petal.Width
## 1          3.5          1.4         0.2
## 2          3.0          1.4         0.2
## 3          3.2          1.3         0.2
## 4          3.1          1.5         0.2
## 5          3.6          1.4         0.2
## 6          3.9          1.7         0.4
## 7          3.4          1.4         0.3
## 8          3.4          1.5         0.2
## 9          2.9          1.4         0.2
## 10         3.1          1.5         0.1
```

```r
df[df$Species == "setosa" & df$Sepal.Width > 4, ]
```

```
##    Sepal.Length Sepal.Width Petal.Length Petal.Width Species
## 16          5.7         4.4          1.5         0.4  setosa
## 33          5.2         4.1          1.5         0.1  setosa
## 34          5.5         4.2          1.4         0.2  setosa
```

Leyendo tablas de datos

- `read.table()`: para definir un data frame a partir de una tabla de datos contenida en un fichero
  - Este fichero puede estar guardado nuestro ordenador o bien podemos conocer su url. Sea cual sea el caso, se aplica la función al nombre del fichero o a la dirección entre comillas.

```r
df = read.table("../../data/bulls.dat")
df
```

```
##    V1   V2   V3   V4   V5 V6   V7   V8   V9
## 1   1 2200 51.0 1128 70.9  7 0.25 54.8 1720
## 2   1 2250 51.9 1108 72.1  7 0.25 55.3 1575
## 3   1 1625 49.9 1011 71.6  6 0.15 53.1 1410
## 4   1 4600 53.1  993 68.9  8 0.35 56.4 1595
```

```
## 5   1 2150 51.2  996 68.6  7 0.25 55.0 1488
## 6   1 1225 49.2  985 71.4  6 0.15 51.4 1500
## 7   1 2250 51.0  959 72.1  7 0.20 54.0 1522
## 8   1 4000 51.5 1060 69.3  7 0.30 55.6 1765
## 9   1 1600 50.1  979 71.2  6 0.25 51.5 1365
## 10  1 1525 49.6 1083 75.8  6 0.30 54.6 1640
## 11  1 1850 50.6 1036 69.2  6 0.15 54.8 1570
## 12  1 2850 51.1  870 70.9  7 0.15 52.9 1450
## 13  1 2650 51.1  998 65.5  7 0.40 54.6 1505
## 14  1 1550 50.2  973 69.5  6 0.40 53.0 1530
## 15  1 2000 49.0  893 73.9  6 0.20 51.9 1470
## 16  1 2300 49.6  975 68.2  6 0.50 52.9 1842
## 17  1 1900 49.1  997 67.9  6 0.30 54.0 1500
## 18  1 1400 48.4  946 68.6  5 0.20 51.2 1480
## 19  1 1650 50.9  928 67.2  6 0.25 54.1 1480
## 20  1 1500 49.5  963 69.4  6 0.35 53.1 1670
## 21  1 1375 49.2  911 67.4  6 0.20 53.4 1490
## 22  1 1500 48.1 1003 70.5  5 0.25 54.7 1748
## 23  1 2400 51.1  915 64.9  7 0.25 54.6 1725
## 24  1 1425 48.9  924 72.7  5 0.15 52.1 1374
## 25  1 1525 49.4  959 68.4  6 0.15 52.6 1565
## 26  1 1800 47.7  944 66.5  5 0.40 53.3 1556
## 27  1 2500 50.6  897 67.2  6 0.30 54.9 1688
## 28  1 1600 48.9  974 71.0  5 0.30 54.2 1722
## 29  1 1300 49.9  872 70.7  6 0.20 53.3 1325
## 30  1 1400 48.4  841 71.3  5 0.15 51.5 1365
## 31  1 1300 48.6  920 71.4  5 0.15 52.9 1450
## 32  1 1400 47.6  974 69.7  5 0.15 51.9 1570
## 33  5 2000 50.5 1002 68.8  6 0.20 54.4 1735
## 34  5 1300 50.2  998 68.7  6 0.15 52.9 1540
## 35  5 1300 49.0 1015 69.8  6 0.30 51.9 1550
## 36  5 1300 48.7 1056 72.9  5 0.15 52.6 1525
## 37  5 1500 49.6  984 71.4  6 0.15 53.4 1650
## 38  5 1225 48.9  934 66.0  5 0.20 52.1 1430
## 39  5 2750 49.7  929 66.9  6 0.25 53.3 1688
## 40  5 1500 49.9  919 67.1  6 0.20 54.3 1425
## 41  5 1325 47.8  931 67.1  5 0.25 51.5 1520
## 42  5 1800 49.6  952 69.4  6 0.25 52.3 1512
## 43  5 1375 51.0 1002 72.1  7 0.25 51.9 1410
## 44  5  975 48.6  936 65.3  5 0.35 51.4 1550
## 45  5 1325 48.3  870 65.6  5 0.30 52.5 1588
## 46  5 1850 50.1  853 67.9  6 0.15 52.9 1390
## 47  5 1025 48.8  843 67.3  5 0.20 50.4 1390
## 48  5 1000 47.7  913 68.2  5 0.15 49.4 1345
## 49  5  975 47.2  844 70.6  5 0.15 50.1 1285
## 50  8 1750 54.0 1252 76.5  8 0.15 56.9 1648
## 51  8 1450 53.3 1383 81.4  8 0.20 59.6 1904
## 52  8 1200 52.8 1076 74.0  7 0.15 55.5 1615
## 53  8 2000 53.5 1175 74.5  8 0.10 57.4 1686
## 54  8 1450 53.2 1027 71.2  8 0.10 56.9 1696
## 55  8 1800 52.3 1116 71.1  7 0.10 57.5 1620
## 56  8 1525 51.8 1095 71.1  7 0.15 54.6 1712
## 57  8 1925 52.7 1141 78.5  7 0.15 55.6 1572
## 58  8 3450 54.8 1039 70.6  8 0.10 58.7 1600
```

```
## 59  8 1650 52.8  981 74.1  7 0.10 56.9 1750
## 60  8 1900 52.4  933 71.5  7 0.10 56.2 1640
## 61  8 1850 51.2 1083 74.5  7 0.20 55.9 1752
## 62  8 1550 52.3 1143 77.7  7 0.10 56.1 1785
## 63  8 1825 53.0 1055 76.8  8 0.10 56.7 1526
## 64  8 1475 52.9 1037 75.0  7 0.10 55.5 1406
## 65  8 2200 51.8 1076 74.5  7 0.15 55.8 1475
## 66  8 1850 53.1  964 70.8  8 0.10 55.5 1535
## 67  8 1550 51.2 1057 74.8  7 0.10 55.5 1520
## 68  8 1250 50.8 1040 74.5  6 0.10 55.8 1516
## 69  8 1350 52.7 1079 75.5  7 0.15 56.1 1595
## 70  8 1725 51.4 1034 71.2  7 0.10 56.0 1655
## 71  8 1750 50.7 1012 71.6  6 0.10 54.3 1480
## 72  8 1450 51.4  997 73.4  7 0.10 55.2 1454
## 73  8 1200 49.8  991 70.8  6 0.15 54.6 1475
## 74  8 1425 50.0  928 70.8  6 0.10 53.9 1375
## 75  8 1250 50.1  990 71.0  6 0.10 54.9 1564
## 76  8 1500 51.7  992 70.6  7 0.15 55.1 1458
```

Algunos de sus parámetros son

- `header = TRUE`: para indicar si la tabla que importamos tiene una primera fila con los nombres de las columnas. El valor por defecto es False
- `col.names = c(...)`: para especificar el nombre de las columnas. No olviden que cada nombre debe ir entre comillas
- `sep`: para especificar las separaciones entre columnas en el fichero (si no es un espacio en blanco). Si es así, hay que introducir el parámetro pertinente entre comillas
- `dec`: para especificar el signo que separa la parte entera de la decimal (si no es un punto). Si es así, hay que introducir el parámetro pertinente entre comillas.

```
df = read.table("../../data/bulls.dat",
                header = FALSE,
                col.names = c("breed", "sale_price", "shoulder",
                              "fat_free", "percent_ff", "frame_scale",
                              "back_fat", "sale_height", "sale_weight"),
                sep = "", dec = ".")

head(df)
```

```
##   breed sale_price shoulder fat_free percent_ff frame_scale back_fat
## 1     1       2200     51.0     1128       70.9           7     0.25
## 2     1       2250     51.9     1108       72.1           7     0.25
## 3     1       1625     49.9     1011       71.6           6     0.15
## 4     1       4600     53.1      993       68.9           8     0.35
## 5     1       2150     51.2      996       68.6           7     0.25
## 6     1       1225     49.2      985       71.4           6     0.15
##   sale_height sale_weight
## 1        54.8        1720
## 2        55.3        1575
## 3        53.1        1410
## 4        56.4        1595
```

```
## 5        55.0        1488
## 6        51.4        1500
```

```
str(df)
```

```
## 'data.frame':    76 obs. of  9 variables:
##  $ breed      : int  1 1 1 1 1 1 1 1 1 ...
##  $ sale_price : int  2200 2250 1625 4600 2150 1225 2250 4000 1600 1525 ...
##  $ shoulder   : num  51 51.9 49.9 53.1 51.2 49.2 51 51.5 50.1 49.6 ...
##  $ fat_free   : int  1128 1108 1011 993 996 985 959 1060 979 1083 ...
##  $ percent_ff : num  70.9 72.1 71.6 68.9 68.6 71.4 72.1 69.3 71.2 75.8 ...
##  $ frame_scale: int  7 7 6 8 7 6 7 7 6 6 ...
##  $ back_fat   : num  0.25 0.25 0.15 0.35 0.25 0.15 0.2 0.3 0.25 0.3 ...
##  $ sale_height: num  54.8 55.3 53.1 56.4 55 51.4 54 55.6 51.5 54.6 ...
##  $ sale_weight: int  1720 1575 1410 1595 1488 1500 1522 1765 1365 1640 ...
```

Factores en un data frame

- `stringsAsFactors`: para prohibir la transformación de las columnas de palabras en factores debemos usar `stringsAsFactors=FALSE` (ya que por defecto, R realiza dicha transformación)
- Para importar un fichero de una página web segura (cuyo url empiece con https), no podemos entrar directamente la dirección en `read.table()`, una solución es instalar y cargar el paquete RCurl y entonces usar la instrucción `read.table(textConnection(getURL("url")), ...)`

```
library(RCurl)
df2 = read.csv(textConnection(getURL("https://archive.ics.uci.edu/ml/machine-learning-databases/00591/na
                fill = TRUE,
                header = TRUE,
                stringsAsFactors = FALSE)

head(df2)
```

```
##   ï..Name Gender   Count Probability
## 1   James      M 5304407  0.01451679
## 2    John      M 5260831  0.01439753
## 3  Robert      M 4970386  0.01360266
## 4 Michael      M 4579950  0.01253414
## 5 William      M 4226608  0.01156713
## 6    Mary      F 4169663  0.01141129
```

Leyendo diferentes tipos de fichero

- `read.csv()`: para importar ficheros en formato CSV7
- `read.xls()` o `read.xlsx`: para importar hojas de cálculo tipo Excel u OpenOffice en formato XLS o XLSX, respectivamente. Se necesita el paquete xlsx
- `read.mtb()`: para importar tablas de datos Minitab. Se necesita el paquete foreign
- `read.spss()`: para importar tablas de datos SPSS. Se necesita el paquete foreign
- `help.search()`: para seguir

Exportando datos a ficheros (Guardar DF)

- `write.table(df, file="")`: para exportar un data frame a un fichero
  - `file = ""`: es donde indicaremos el nombre que queremos darle al fichero
  - Podemos usar el parámetro `sep` para indicar el símbolo de separación de columnas. Siempre entre comillas
  - También podemos utilizar el parámetro `dec` para indicar la separación entre la parte entera y decimal de los datos

```
df3 = read.table("https://archive.ics.uci.edu/ml/machine-learning-databases/wine/wine.data",
                 col.names = c("class_id", "alcohol", "Malic_acid", "Ash",
                               "Alcalinity_of_ash", "Magnesium", "Total_phenols", "Flavanoids",
                               "Nonflavanoid_phenols", "Proanthocyanins", "Color_instensity",
                               "Hue", "OD280/OD315", "Proline"),
                 fill = TRUE, sep = ",")

write.table(df3, file = "../../data/wines.txt", dec = ".")
df4 = read.table("../../data/wines.txt", header = TRUE, fill = TRUE)
head(df4)
```

```
##   class_id alcohol Malic_acid  Ash Alcalinity_of_ash Magnesium Total_phenols
## 1        1   14.23       1.71 2.43              15.6       127          2.80
## 2        1   13.20       1.78 2.14              11.2       100          2.65
## 3        1   13.16       2.36 2.67              18.6       101          2.80
## 4        1   14.37       1.95 2.50              16.8       113          3.85
## 5        1   13.24       2.59 2.87              21.0       118          2.80
## 6        1   14.20       1.76 2.45              15.2       112          3.27
##   Flavanoids Nonflavanoid_phenols Proanthocyanins Color_instensity  Hue
## 1       3.06                0.28            2.29             5.64 1.04
## 2       2.76                0.26            1.28             4.38 1.05
## 3       3.24                0.30            2.81             5.68 1.03
## 4       3.49                0.24            2.18             7.80 0.86
## 5       2.69                0.39            1.82             4.32 1.04
## 6       3.39                0.34            1.97             6.75 1.05
##   OD280.OD315 Proline
## 1        3.92    1065
## 2        3.40    1050
## 3        3.17    1185
## 4        3.45    1480
## 5        2.93     735
## 6        2.85    1450
```

Construyendo data frames

- `data.frame(vector_1,...,vector_n)`: para construir un data frame a partir de vectores introducidos en el orden en el que queremos disponer las columnas de la tabla
  - R considera del mismo tipo de datos todas las entradas de una columna de un data frame‘
  - Las variables tomarán los nombres de los vectores. Estos nombres se pueden especificar en el argumento de `data.frame` entrando una construcción de la forma `nombre_variable = vector`
  - `rownames`: para especificar los identificadores de las filas

- También en esta función podemos hacer uso del parámetro `stringAsFactors` para evitar la transformación de las columnas de tipo palabra en factores

```
gender = c("H", "M", "M", "M", "H")
age =    c( 23,  45,  20,  30,  18)
family = c(  2,   3,   4,   2,   5)
df5 = data.frame(genero = gender, edad = age, familia = family, stringsAsFactors = TRUE)
rownames(df5) = c("P1", "P2", "P3", "P4", "P5")
df5
```

```
##    genero edad familia
## P1      H   23       2
## P2      M   45       3
## P3      M   20       4
## P4      M   30       2
## P5      H   18       5
```

```
## Añadir un nombres en la columna y fila
dimnames(df5) = list(
  c("Antonio", "Susana", "Angela", "Ximena", "Felipe"),
  c("Sexo", "Años", "MiembrosFamilia")
)
df5 = rbind(df5, c("H", 30, 1))
df5
```

```
##         Sexo Años MiembrosFamilia
## Antonio    H   23               2
## Susana     M   45               3
## Angela     M   20               4
## Ximena     M   30               2
## Felipe     H   18               5
## 6          H   30               1
```

```
## Uso de parámetros para transformar de factor a caracter
df5$Sexo = as.character(df5$Sexo)

## Añadir nuevo parámetro
df5$Ingresos = c(10000,15000,8000,9000,10400,20000)
df5
```

```
##         Sexo Años MiembrosFamilia Ingresos
## Antonio    H   23               2    10000
## Susana     M   45               3    15000
## Angela     M   20               4     8000
## Ximena     M   30               2     9000
## Felipe     H   18               5    10400
## 6          H   30               1    20000
```

Cambiando los tipos de datos

- `as.character`: para transformar todos los datos de un objeto en palabras

- `as.integer`: para transformar todos los datos de un objeto a números enteros
- `as.numeric`: para transformar todo los datos de un objeto a número reales

```
str(df5)
```

```
## 'data.frame':    6 obs. of  4 variables:
##  $ Sexo           : chr  "H" "M" "M" "M" ...
##  $ Años           : chr  "23" "45" "20" "30" ...
##  $ MiembrosFamilia: chr  "2" "3" "4" "2" ...
##  $ Ingresos       : num  10000 15000 8000 9000 10400 20000
```

```
as.factor(df5$Años)
```

```
## [1] 23 45 20 30 18 30
## Levels: 18 20 23 30 45
```

```
as.factor(df5$Sexo)
```

```
## [1] H M M M H H
## Levels: H M
```

```
as.numeric(df5$Años)
```

```
## [1] 23 45 20 30 18 30
```

Otras formas de hacer subdata frames

- `droplevels(d.f)`: para borrar los niveles sobrantes de todos los factores, ya que las columnas que son factores heredan en los subdata frames todos los niveles del factor original aunque no aparezcan en el trozo que hemos extraído
- `select(d.f, parámetros)`: para especificar que queremos extraer de un data frame
  - `starts_with("x")`: extrae del data frame las variables cuyo nombre empieza con la palabra "x"
  - `ends_with("x")`: extrae del data frame las variables cuyo nombre termina con la palabra "x"
  - `contains("x")`: extrae del data frame las variables cuyo nombre contiene la palabra "x"
  - Se necesita el paquete `dplyr` o mejor aún `tidyverse`

```
gender = c("H", "M", "M", "M", "H")
age =    c( 23,  45,  20,  30,  18)
family = c(  2,   3,   4,   2,   5)
df5 = data.frame(genero = gender, edad = age, familia = family, stringsAsFactors = TRUE)
df5[df5$genero == "M", ] -> df_m
str(df_m)
```

```
## 'data.frame':    3 obs. of  3 variables:
##  $ genero : Factor w/ 2 levels "H","M": 2 2 2
##  $ edad   : num  45 20 30
##  $ familia: num  3 4 2
```

```r
df_m = droplevels(df_m)
str(df_m)
```

```
## 'data.frame':    3 obs. of  3 variables:
##  $ genero : Factor w/ 1 level "M": 1 1 1
##  $ edad   : num  45 20 30
##  $ familia: num  3 4 2
```

Tidyverse

```r
library(tidyverse)
```

```
## -- Attaching packages --------------------------------------- tidyverse 1.3.1 --
```

```
## v ggplot2 3.3.3     v purrr   0.3.4
## v tibble  3.1.1     v dplyr   1.0.6
## v tidyr   1.1.3     v stringr 1.4.0
## v readr   1.4.0     v forcats 0.5.1
```

```
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x tidyr::complete() masks RCurl::complete()
## x dplyr::filter()   masks stats::filter()
## x dplyr::lag()      masks stats::lag()
```

```r
iris_petal = select(iris, starts_with("Petal"))
head(iris_petal)
```

```
##   Petal.Length Petal.Width
## 1          1.4         0.2
## 2          1.4         0.2
## 3          1.3         0.2
## 4          1.5         0.2
## 5          1.4         0.2
## 6          1.7         0.4
```

```r
iris_length = select(iris, ends_with("Length"))
head(iris_length)
```

```
##   Sepal.Length Petal.Length
## 1          5.1          1.4
## 2          4.9          1.4
## 3          4.7          1.3
## 4          4.6          1.5
## 5          5.0          1.4
## 6          5.4          1.7
```

- `subset(d.f, condición, select = columnas)` para extraer del data frame las filas que cumplen la condición y las columnas especificadas

- Si queremos todas las filas, no hay que especificar ninguna condición
- Si queremos todas las columnas, no hace especificar el parámetro `select`
- Las variables en la condición se especifican con su nombre, sin añadir antes el nombre del data frame

Subset

```
versicolor <- subset(iris, Species == "versicolor", select = c(1,3))
rownames(versicolor) = 1:nrow(versicolor) # Para empezar desde 1 hasta la cantidad de elementos
head(versicolor, 5)
```

```
##   Sepal.Length Petal.Length
## 1          7.0          4.7
## 2          6.4          4.5
## 3          6.9          4.9
## 4          5.5          4.0
## 5          6.5          4.6
```

```
str(versicolor)
```

```
## 'data.frame':    50 obs. of  2 variables:
##  $ Sepal.Length: num  7 6.4 6.9 5.5 6.5 5.7 6.3 4.9 6.6 5.2 ...
##  $ Petal.Length: num  4.7 4.5 4.9 4 4.6 4.5 4.7 3.3 4.6 3.9 ...
```