Breve Introducción a R

Kamal Romero

noviembre 07, 2019

Contents

1 Operaciones aritméticas

Para introducirnos al uso del editor y de la linea de comandos de Rstudio, empezaremos con unas operaciones muy básica

```
2 + 2
```

```
## [1] 4
```

```
5 - 3
```

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

```
3 * 2
```

```
## [1] 6
```

```
6 / 3
```

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

Es posible aplicar reglas de asociación estándar así como operaciones más allá de las básicas (potencia, logaritmo, etc.)

```r
(5 + 3) / 4
```

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

```r
3^2
```

```
## [1] 9
```

```r
log(100, base = 10)
```

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

```r
5 %% 3
```

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

```r
sqrt(9)
```

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

2 Asignación de variables

Podemos guardar *valores* asignandoles un nombre, de modo que podamos acceder a dicho valor posteriormente

nota: esto es importante

```r
x <- 4

x
```

```
## [1] 4
```

```r
Numero_de_empleados <- 150

Numero_de_empleados
```

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

```r
Nuevos_analistas_enero <- 5

Nuevos_analistas_febrero <- 3

Analistas <- Nuevos_analistas_enero + Nuevos_analistas_febrero

Analistas
```

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

3 Tipos básicos en R

```r
Numero <-  1.0   # (real, flotante)
Entero <- 1
Caracter <- "ab"
Logico <- TRUE

Numero
```

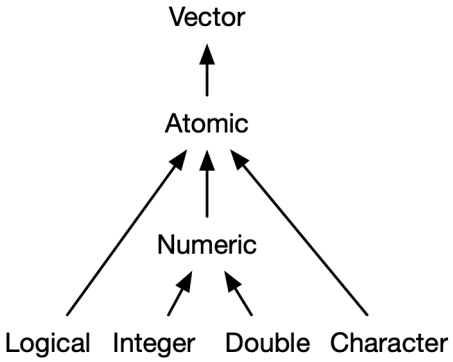


Figure 1:

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

```
Entero
```

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

```
Caracter
```

```
## [1] "ab"
```

```
Logico
```

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

Tener cuidado cuando trabajamos con tipos distintos

```
#Numero + Caracter
```

Cuando realizamos una operación con dos tipos numéricos distintos (real + entero), R *forza* (coerce) a que el resultado al tipo con mayor precisión, en este caso al tipo real

```
Suma <- Numero + Entero

Suma
```

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

```
class(Suma)
```

```
## [1] "numeric"
```

```
typeof(Suma)
```

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

Varias cosas acá, hemos tenido nuestra primera aproximación a una función en R, tópico que exploraremos con mayor detalle más adelante. Al igual que la idea intuitiva que tenemos de función de las matemáticas de secundaria, una función en R tiene un argumento (variable entre paréntesis) y nos arroja un resultado.

En R las funciones tienen su nombre seguido de paréntesis, donde colocamos la(s) variable(s) de argumento(s): `una_funcion(x)`

Las funcion `class` nos indica la clase de la variable (numérico, entero o lógico)

```
is.numeric(Numero)
```

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

```r
is.integer(Entero)
```

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

```r
is.character(Caracter)
```

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

```r
is.logical(Logico)
```

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

Vemos que R nos dice que `Entero` no es un entero, para especificar un entero debemos colocar una letra L al final del número

```r
Entero_2 <- 1L
Entero_2
```

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

```r
is.integer(Entero_2)
```

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

```r
class(Entero_2)
```

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

```r
typeof(Entero_2)
```

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

4 Vectores

Vamos a utilizar R para analizar datos y crear modelos estadísticos o algorítmicos a partir de los mismos.

La mayoría de los datos vienen representados en *tablas*: hojas de cálculo, bases de datos relacionales, ficheros .csv, etc.

La mayoría de los modelos estadísticos usan como input datos en forma de tabla.

El objeto más empleado para trabajar con tablas en R son los **data frame** y demás variantes (los tibble por ejemplo).

Antes de entender como trabajar tablas, repasamos el concepto de vector, que es el tipo básico sobre el cual están construidos los data frames.

```r
vector_numerico <- c(1, 10, 49)
vector_caracter <- c("a", "b", "c")

vector_numerico
```

```
## [1]  1 10 49
```

```r
vector_caracter
```

```
## [1] "a" "b" "c"
```

Los vectores son arreglos de una dimensión (fila o columna) que pueden almacenar números, carácteres o variables lógicas.

Los elementos deben ser del mismo tipo (todo números por ejemplo)

Como hemos visto arriba, los vectores se crean con el comando `c()` donde la c viene de *concatenar*

```
vector_mixto <- c(1,2,"a")
vector_mixto
```

```
## [1] "1" "2" "a"
```

En el ejemplo anterior hemos querido crear un vector con elementos de distintos tipos, numérico y carácter. R ha convertido todos los elementos a carácter.

Si hay carácteres en un vector R convierte todos los elementos a carácter, si todos son numéricos pero de distinto tipo, R los convierte al tipo de mayor precisión (doble). ¿Qué pasa con los vectores lógicos?

```
vector_mixto2 <- c(1,2,TRUE)
vector_mixto2
```

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

En este caso R ha convertido los elementos del vector a numérico

Observamos algo que posteriormente va a ser muy útil, R le ha asignado a la variable lógica `TRUE` el número 1. La variable `FALSE` tiene asignado un cero

¿Podemos cambiar una variable o un vector de tipo? SI

```
vector_numerico
```

```
## [1]  1 10 49
```

```
as.character(vector_numerico)
```

```
## [1] "1"  "10" "49"
```

```
vector_logico <- c(TRUE, FALSE)

vector_logico
```

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

```
as.numeric(vector_logico)
```

```
## [1] 1 0
```

Existen varias funciones en R que permiten realizar cambios de tipo

Existen otro par de formas de crear vectores

```
vector_1 <- 1:5

vector_2 <- seq(1,5)

vector_1
```

```
## [1] 1 2 3 4 5
```

```
vector_2
```

```
## [1] 1 2 3 4 5
```

Donde `seq` viene de secuencia

Se pueden crear secuencias más complejas con la función `seq()`

```
vector_3 <- seq(1,10, by = 2)
```

```r
vector_4 <- seq(1,10, length.out = 20)

vector_5 <- seq(1,10, along.with = vector_1)

vector_3
```

```
## [1] 1 3 5 7 9
```

```r
vector_4
```

```
##  [1]  1.000000  1.473684  1.947368  2.421053  2.894737  3.368421  3.842105
##  [8]  4.315789  4.789474  5.263158  5.736842  6.210526  6.684211  7.157895
## [15]  7.631579  8.105263  8.578947  9.052632  9.526316 10.000000
```

```r
vector_5
```

```
## [1]  1.00  3.25  5.50  7.75 10.00
```

Se pueden asignar nombres a los elementos del vector

```r
names(vector_1)
```

```
## NULL
```

```r
nombres <- c("uno", "dos", "tres", "cuatro", "cinco")

names(vector_1) <- nombres

names(vector_1)
```

```
## [1] "uno"    "dos"    "tres"   "cuatro" "cinco"
```

Otro comando para generar secuencias es `rep`

```r
repetido <- rep(4,10)

repetido
```

```
##  [1] 4 4 4 4 4 4 4 4 4 4
```

Podemos repetir no solo un valor sino vectores

```r
repetido_2 <- rep(1:4, 4)

repetido_2
```

```
##  [1] 1 2 3 4 1 2 3 4 1 2 3 4 1 2 3 4
```

```r
repetido_3 <- rep(1:4, each = 4)

repetido_3
```

```
##  [1] 1 1 1 1 2 2 2 2 3 3 3 3 4 4 4 4
```

```r
repetido_4 <- rep(1:4, each=2, times=2)

repetido_4
```

```
##  [1] 1 1 2 2 3 3 4 4 1 1 2 2 3 3 4 4
```

```r
repetido_5 <- rep(1:4, c(2,3,4,5))

repetido_5
```

```
##  [1] 1 1 2 2 2 3 3 3 3 4 4 4 4 4
```

4.1 Matrices

Al igual que la noción intuitiva que tenemos de matemáticas, las matrices son arreglos rectangulares de datos del mismo tipo

```r
A <- matrix(c(1, 2, 3, 4), nrow=2, ncol=2)
A
```

```
##      [,1] [,2]
## [1,]    1    3
## [2,]    2    4
```

4.2 Operaciones con vectores

Se pueden realizar operaciones con vectores

```r
vector_1
```

```
##    uno    dos   tres cuatro  cinco
##      1      2      3      4      5
```

```r
vector_2
```

```
## [1] 1 2 3 4 5
```

```r
Vec_suma <- vector_1 +vector_2

Vec_suma
```

```
##    uno    dos   tres cuatro  cinco
##      2      4      6      8     10
```

```r
vector_1
```

```
##    uno    dos   tres cuatro  cinco
##      1      2      3      4      5
```

```r
vector_2
```

```
## [1] 1 2 3 4 5
```

```r
Vec_producto <- vector_1 * vector_2

Vec_producto
```

```
##    uno    dos   tres cuatro  cinco
##      1      4      9     16     25
```

```r
vector_1
```

```
##    uno    dos   tres cuatro  cinco
##      1      2      3      4      5
```

```
vector_cuadrado <- vector_1^2

vector_cuadrado
```

```
##    uno    dos   tres cuatro  cinco
##      1      4      9     16     25
```

Se pueden pasar funciones a vectores

```
vector_1
```

```
##    uno    dos   tres cuatro  cinco
##      1      2      3      4      5
```

```
vector_raiz <- sqrt(vector_1)

vector_raiz
```

```
##      uno      dos     tres   cuatro    cinco
## 1.000000 1.414214 1.732051 2.000000 2.236068
```

4.3 Inspección de vectores

Es posible aplicar ciertas funciones para analizar algunas características de los vectores.

Recordar que el objetivo último es el de realizar análisis de datos, algunas de estas funciones serán empleadas de manera regular cuando querramos inspeccionar los datos de una tabla

Longitud de un vector

```
vector_4
```

```
##  [1]  1.000000  1.473684  1.947368  2.421053  2.894737  3.368421  3.842105
##  [8]  4.315789  4.789474  5.263158  5.736842  6.210526  6.684211  7.157895
## [15]  7.631579  8.105263  8.578947  9.052632  9.526316 10.000000
```

```
length(vector_4)
```

```
## [1] 20
```

Estadísticos de un vector

```
summary(vector_4)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    1.00    3.25    5.50    5.50    7.75   10.00
```

Valores al principio y al final

```
head(vector_4)
```

```
## [1] 1.000000 1.473684 1.947368 2.421053 2.894737 3.368421
```

```
tail(vector_4)
```

```
## [1]  7.631579  8.105263  8.578947  9.052632  9.526316 10.000000
```

Tabla de frecuencias

```
table(vector_4)
```

```
## vector_4
##                1 1.47368421052632 1.94736842105263 2.42105263157895
##                1                1                1                1
## 2.89473684210526 3.36842105263158 3.84210526315789 4.31578947368421
##                1                1                1                1
## 4.78947368421053 5.26315789473684 5.73684210526316 6.21052631578947
##                1                1                1                1
## 6.68421052631579 7.15789473684211 7.63157894736842 8.10526315789474
##                1                1                1                1
## 8.57894736842105 9.05263157894737 9.52631578947368               10
##                1                1                1                1
```

4.4 Selecciones

Es posible construir nuevos vectores a partir de un vector

```
vector_1
```

```
##    uno    dos   tres cuatro  cinco
##      1      2      3      4      5
```

```
vector_cuadrado <- vector_1^2

vector_cuadrado
```

```
##    uno    dos   tres cuatro  cinco
##      1      4      9     16     25
```

¿Como accedemos a los elementos de un vector?

```
vector_cuadrado[1]
```

```
## uno
##   1
```

```
vector_cuadrado[3]
```

```
## tres
##    9
```

Varios elementos seguidos (slice)

```
vector_cuadrado[1:3]
```

```
##  uno  dos tres
##    1    4    9
```

```
vector_cuadrado[3:1]
```

```
## tres  dos  uno
##    9    4    1
```

Varios elementos no seguidos

```
vector_cuadrado[c(1,3)]
```

```
##  uno tres
##    1    9
```

```
vector_cuadrado[c(3,1)]
```

```
## tres  uno
##    9    1
```

Todos los elementos excepto algunos

```
vector_cuadrado[-c(1,3)]
```

```
##    dos cuatro  cinco
##      4     16     25
```

```
vector_cuadrado[-length(vector_cuadrado)]
```

```
##    uno    dos   tres cuatro
##      1      4      9     16
```

Todos los elementos que cumplan una condición

```
vector_5
```

```
## [1]  1.00  3.25  5.50  7.75 10.00
```

```
vector_5[vector_5 < 5]
```

```
## [1] 1.00 3.25
```

```
vector_5[vector_5 > 5]
```

```
## [1]  5.50  7.75 10.00
```

La condición puede ser una igualdad

```
vector_5[vector_5 == 5.5]
```

```
## [1] 5.5
```

Estas condiciones son muy importantes para el análisis de datos. Suponga que desea localizar en una tabla con los datos financieros de un conjunto de personas, los datos de aquellos clientes cuyo ingreso sea mayor a 30.000 al año

También es útil para localizar datos y reemplazarlos por otro valor. Siguiendo el ejemplo anterior, sustituir la columna de ingresos de los clientes cuyo ingreso anual es menor de 12.000 al año por cero, ya que no trabajaremos con ese segmento

```
vector_5
```

```
## [1]  1.00  3.25  5.50  7.75 10.00
```

```
vector_5[vector_5 < 5] <- 0

vector_5
```

```
## [1]  0.00  0.00  5.50  7.75 10.00
```

En el caso que los vectores tengan nombres, podemos emplear estos para seleccionar elementos

```
vector_cuadrado["dos"]
```

```
## dos
##   4
```

```
vector_cuadrado[c("uno", "tres")]
```

```
##  uno tres
##    1    9
```

Más funciones aplicadas a vectores

```
vector_5 <- seq(1,10, along.with = vector_1)

vector_5
```

```
## [1]  1.00  3.25  5.50  7.75 10.00
```

```
max(vector_5)
```

```
## [1] 10
```

```
min(vector_5)
```

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

```
sum(vector_5)
```

```
## [1] 27.5
```

```
prod(vector_5)
```

```
## [1] 1385.312
```

5 Cosas Pendientes

5.1 Paquetes

Además de las funciones que vienen con R podemos instalar otros paquetes que tienen otras funciones incorporadas.

Un paquete no es más que código y/o datos que mejoran alguna funcionalidad de R o incorpora una nueva.

Para instalar un paquete solo tenemos que ejecutar `install.packages(nombre paquete)` y para usarlo lo cargamos en nuestro script con `library(nombre paquete)`.

Por lo general vamos a cargar e instalar paquetes con mucha frecuencia, ya que para muchos ejercicios estadísticos, de previsión, machine learning, etc. necesitaremos más y/o mejores funciones que las que trae el R base.

Para ver los paquetes que tenemos instalados podemos ir a la pestaña `Packages` o teclear en nuestra consola:

```
installed.packages()
```

Una búsqueda en google nos ayuda a buscar nuevos paquetes que nos ayuden a hacer cosas que necesitamos pero también tenemos la lista oficial de CRAN (The Comprehensible R Archive Network) o MetaCRAN que además nos ayuda a buscar los paquetes alojados en otras plataformas como GitHub.

5.2 Directorios de trabajo

Por higiene en el trabajo ya sea profesional o académico, es recomendable fijar los directorios (dirección de la carpeta) donde vamos a trabajar

Para saber en que carpeta estamos empleamos la instrucción `getwd()`

```
getwd()
```

```
## [1] "C:/Users/Usuario/Documents/Mapfre/IntroToR"
```

```
setwd("c:/Users/Usuario/Documents/Mapfre/IntroToR/")
```

Verificamos

```
getwd()
```

```
## [1] "C:/Users/Usuario/Documents/Mapfre/IntroToR"
```

5.3 Factores

En R se emplea el tipo factor para representar variables catergóricas (sexo, estado civil, código postal, cargo, etc.)

Existen variables categóricas las cuales poseen *niveles*, por ejemplo si nos referimos a tamaños, mediano es mayor a pequeño y estos dos a su vez están por debajo de grande.

El tipo factor permite que podamos realizar análisis estadísticos sobre variables categóricas tengan o no niveles u ordenaciones por jerarquía.

Se crean factores usando la función `factor()`

```
factor(1:3)
```

```
## [1] 1 2 3
## Levels: 1 2 3
```

Si deseamos colocar niveles al factor

```
factor(1:3, levels = 1:5)
```

```
## [1] 1 2 3
## Levels: 1 2 3 4 5
```

Nota: Cuando aprendamos a cargar datos externos en R, los data frames transforman por defecto cualquier carácter en factor, si no deseamos que eso sea así se lo debemos indicar a R

6 Data Frames

El `data.frame` será uno de los objetos que más utilizaremos en R para el análisis de datos.

Los data frames son *tablas* o arreglos rectangulares, podemos tratarlos como si fueran una hoja de Excel en la cual organizamos los datos que deseamos analizar.

Al igual que en Excel, podemos hacer cálculos con las tablas, crear nuevas columnas (variables), realizar análisis estadísticos, etc.

Una gran diferencia de R (u otros lenguajes) con Excel es que nuestro análisis puede ser reproducible, un tercero puede ver todos los pasos que hemos realizado y ser capaz de replicarlo.

6.1 Inspeccionar los datos

Por lo general, solemos inspeccionar los datos de una tabla para familiarizarnos con ellos y ver que tipo de análisis podemos realizar con ellos.

Vamos a usar datos de seguros de coches en el Reino Unido en 1975

Podemos abrir una versión amigable de la tabla en RStudio

```
View(ukaggclaim)
```

Existen varias formas de echar vistazos rápidos a los datos sin necesidad de visualizar la tabla entera

Por ejemplo podemos ver las 6 primeras filas

```
head(ukaggclaim)
```

```
##   OwnerAge Model CarAge NClaims AvCost
## 1    17-20     A    0-3       8    289
## 2    17-20     A    4-7       8    282
## 3    17-20     A    8-9       4    133
## 4    17-20     A    10+       1    160
## 5    17-20     B    0-3      10    372
## 6    17-20     B    4-7      28    249
```

O las 6 últimas

```
tail(ukaggclaim)
```

```
##     OwnerAge Model CarAge NClaims AvCost
## 123      60+     C    8-9      20    227
## 124      60+     C    10+       6    119
## 125      60+     D    0-3      62    385
## 126      60+     D    4-7      22    324
## 127      60+     D    8-9       6    192
## 128      60+     D    10+       6    123
```

Si en lugar de 6, queremos ver un determinado número de filas, simplemente se lo decimos a las funciones `heady tail`

```
head(ukaggclaim, n = 3)
```

```
##   OwnerAge Model CarAge NClaims AvCost
## 1    17-20     A    0-3       8    289
## 2    17-20     A    4-7       8    282
## 3    17-20     A    8-9       4    133
```

```
tail(ukaggclaim, n = 2)
```

```
##     OwnerAge Model CarAge NClaims AvCost
## 127      60+     D    8-9       6    192
## 128      60+     D    10+       6    123
```

En ocasiones utilizamos tablas muy grandes que almacenan muchas variables, en este caso antes de visualizar aunque sea solo una fracción de la tabla, es útil ver el nombre las columnas o lo que es lo mismo, las variables

```
colnames(ukaggclaim)
```

```
## [1] "OwnerAge" "Model"    "CarAge"   "NClaims"  "AvCost"
```

En general, para poder determinar cuantas variables y cuantas observaciones tenemos, usamos la función `dim`, la cual nos da el número de columnas y filas de la tabla

```
dim(ukaggclaim)
```

```
## [1] 128   5
```

Podemos ver por separado el número de filas y columnas con `nrow` y `ncol`

```
ncol(ukaggclaim)
```

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

```
nrow(ukaggclaim)
```

```
## [1] 128
```

Otra forma es usando la función `str`. Aunque no es intuitiva e incluso intimida a primera vista, no es tan complicado y aporta información útil

```
str(ukaggclaim)
```

```
## 'data.frame':    128 obs. of  5 variables:
##  $ OwnerAge: Factor w/ 8 levels "17-20","21-24",..: 1 1 1 1 1 1 1 1 1 1 ...
##  $ Model   : Factor w/ 4 levels "A","B","C","D": 1 1 1 1 2 2 2 2 3 3 ...
##  $ CarAge  : Factor w/ 4 levels "0-3","10+","4-7",..: 1 3 4 2 1 3 4 2 1 3 ...
##  $ NClaims : int  8 8 4 1 10 28 1 1 9 13 ...
##  $ AvCost  : int  289 282 133 160 372 249 288 11 189 288 ...
```

Una versión que aporta información estadística es `summary`

```
summary(ukaggclaim)
```

```
##     OwnerAge  Model  CarAge      NClaims           AvCost
##  17-20  :16   A:32   0-3:32   Min.   :  0.00   Min.   : 11.0
##  21-24  :16   B:32   10+:32   1st Qu.:  9.00   1st Qu.:158.5
##  25-29  :16   C:32   4-7:32   Median : 35.50   Median :213.0
##  30-34  :16   D:32   8-9:32   Mean   : 69.86   Mean   :231.1
##  35-39  :16                   3rd Qu.: 96.25   3rd Qu.:272.0
##  40-49  :16                   Max.   :434.00   Max.   :850.0
##  (Other):32                                    NA's   :5
```

Existen otras versiones de `summary` fuera de R base, elaboradas por otras personas (luego hablaremos de las librerías)

Una que aporta mucho detalle es la función `describe` de la librería `psich`

```
pacman::p_load(psych)
describe(ukaggclaim)
```

```
##           vars   n   mean     sd median trimmed   mad min max range skew
## OwnerAge*    1 128   4.50   2.30    4.5    4.50  2.97   1   8     7 0.00
## Model*       2 128   2.50   1.12    2.5    2.50  1.48   1   4     3 0.00
## CarAge*      3 128   2.50   1.12    2.5    2.50  1.48   1   4     3 0.00
## NClaims      4 128  69.86  91.85   35.5   51.31 43.00   0 434   434 2.00
## AvCost       5 123 231.13 117.05  213.0  217.73 84.51  11 850   839 2.26
##           kurtosis    se
## OwnerAge*    -1.27  0.20
## Model*       -1.39  0.10
## CarAge*      -1.39  0.10
## NClaims       3.83  8.12
## AvCost        8.61 10.55
```

La función `skim` de la librería `skimr` también aporta más información que `summary`

```
pacman::p_load(skimr)
skim(ukaggclaim)
```

```
## Skim summary statistics
```

```
##  n obs: 128
##  n variables: 5
##
## -- Variable type:factor ---------------------------------------------------------------------------
##  variable missing complete   n n_unique                          top_counts
##    CarAge       0      128 128        4 0-3: 32, 10+: 32, 4-7: 32, 8-9: 32
##     Model       0      128 128        4         A: 32, B: 32, C: 32, D: 32
##  OwnerAge       0      128 128        8 17-: 16, 21-: 16, 25-: 16, 30-: 16
##  ordered
##    FALSE
##    FALSE
##    FALSE
##
## -- Variable type:integer --------------------------------------------------------------------------
##  variable missing complete   n   mean     sd p0   p25   p50    p75 p100
##    AvCost       5      123 128 231.13 117.05 11 158.5 213   272     850
##   NClaims       0      128 128  69.86  91.85  0   9    35.5  96.25  434
##      hist
##  <U+2582><U+2587><U+2587><U+2582><U+2581><U+2581><U+2581><U+2581>
##  <U+2587><U+2582><U+2581><U+2581><U+2581><U+2581><U+2581><U+2581>
```

Una librería más reciente llamada `inspectdf` no solo aporta mucha información también en forma gráfica, sino que permite comparar dos dataframes que comparten columnas

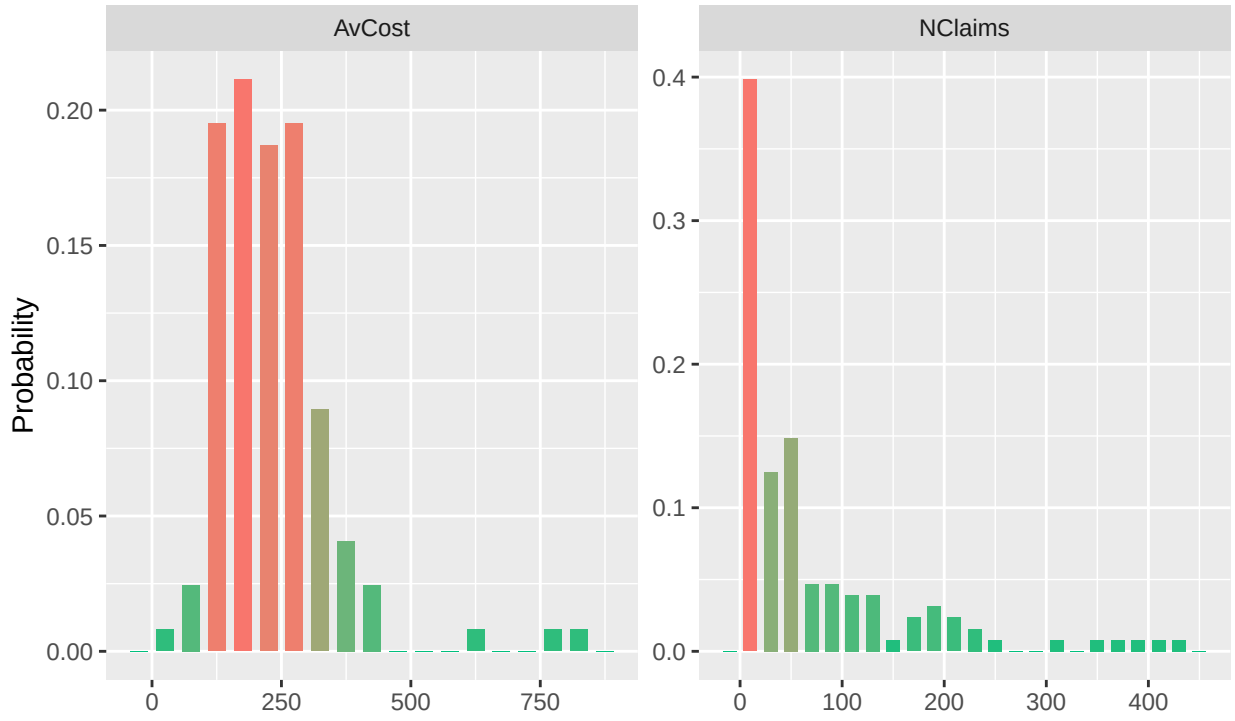
Por ejemplo podemos ver un resumen parecido al de la función `summary()`, con la función `show_plot()` graficamos los histogramas de las variables numéricas

```
pacman::p_load(inspectdf)
inspect_num(ukaggclaim)
```

```
## # A tibble: 2 x 10
##   col_name   min    q1 median  mean    q3   max    sd pcnt_na hist
##   <chr>    <int> <dbl>  <dbl> <dbl> <dbl> <int> <dbl>   <dbl> <list>
## 1 NClaims      0    9    35.5  69.9  96.2   434  91.9    0    <tibble [24 x 2]>
## 2 AvCost      11  158.  213   231.  272     850 117.     3.91 <tibble [19 x 2]>
```

```
show_plot(inspect_num(ukaggclaim))
```

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

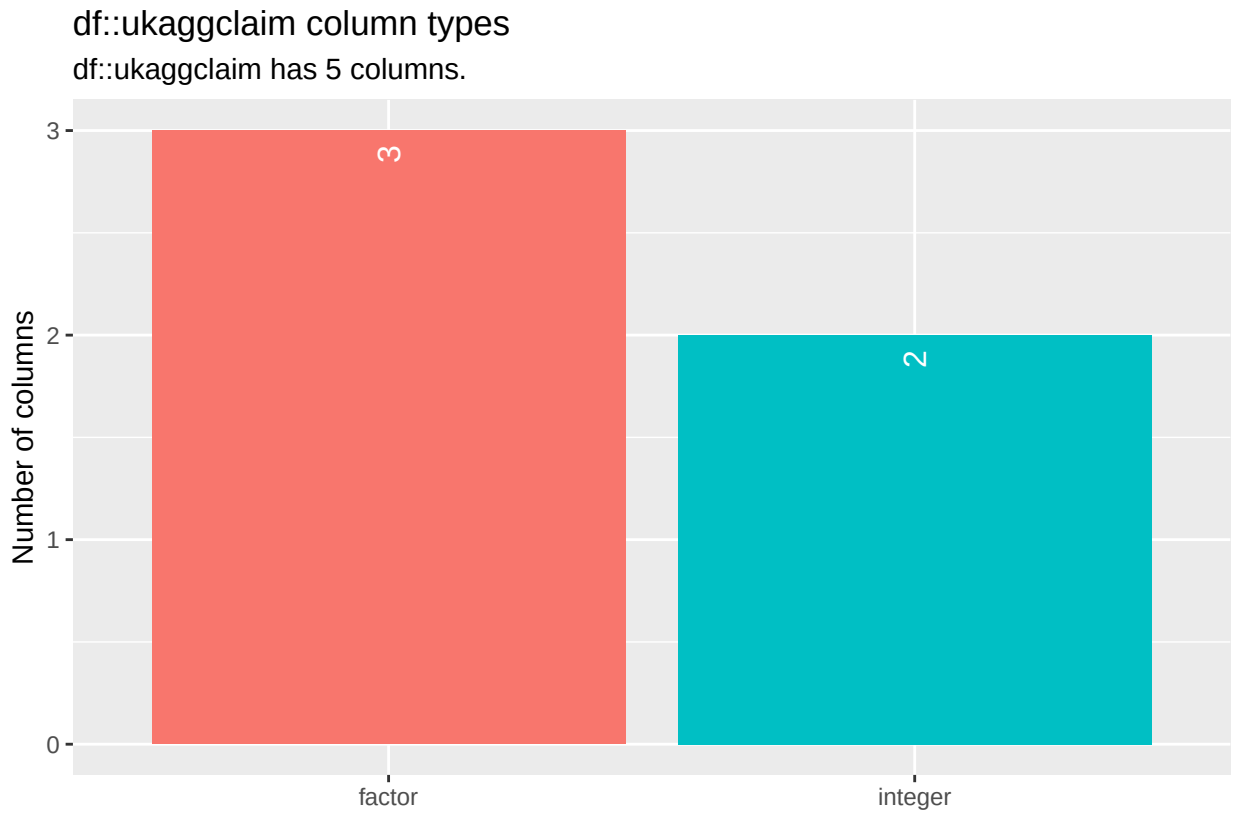


Podemos ver los distintos tipos de variables a través de una tabla y/o un gráfico

```
inspect_types(ukaggclaim)
```

```
## # A tibble: 2 x 4
##   type      cnt  pcnt col_name
##   <chr>   <int> <dbl> <list>
## 1 factor      3    60 <chr [3]>
## 2 integer     2    40 <chr [2]>
```

```
show_plot(inspect_types(ukaggclaim))
```

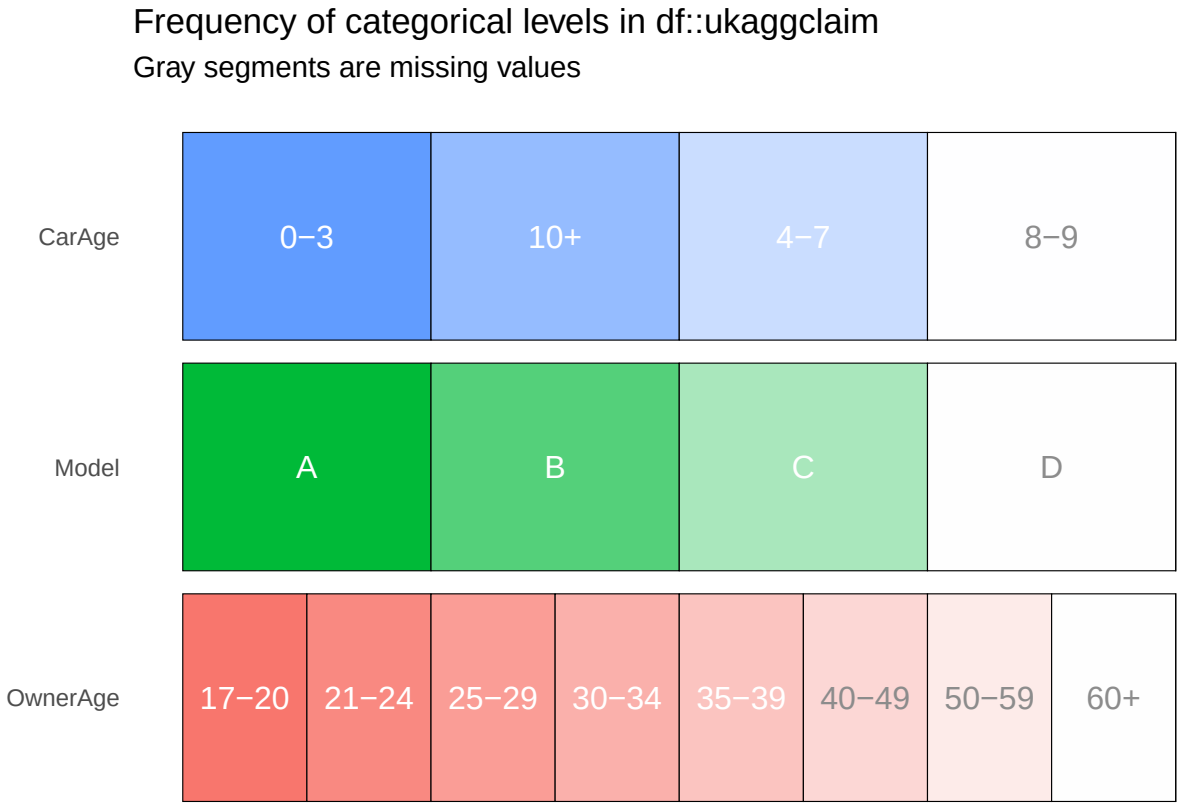


También inspeccionar solo las variables categóricas

```
inspect_cat(ukaggclaim)
```

```
## # A tibble: 3 x 5
##   col_name   cnt common common_pcnt levels
##   <chr>    <int> <chr>        <dbl> <list>
## 1 CarAge       4 0-3           25   <tibble [4 x 3]>
## 2 Model        4 A             25   <tibble [4 x 3]>
## 3 OwnerAge     8 17-20         12.5 <tibble [8 x 3]>
```

```
show_plot(inspect_cat(ukaggclaim))
```

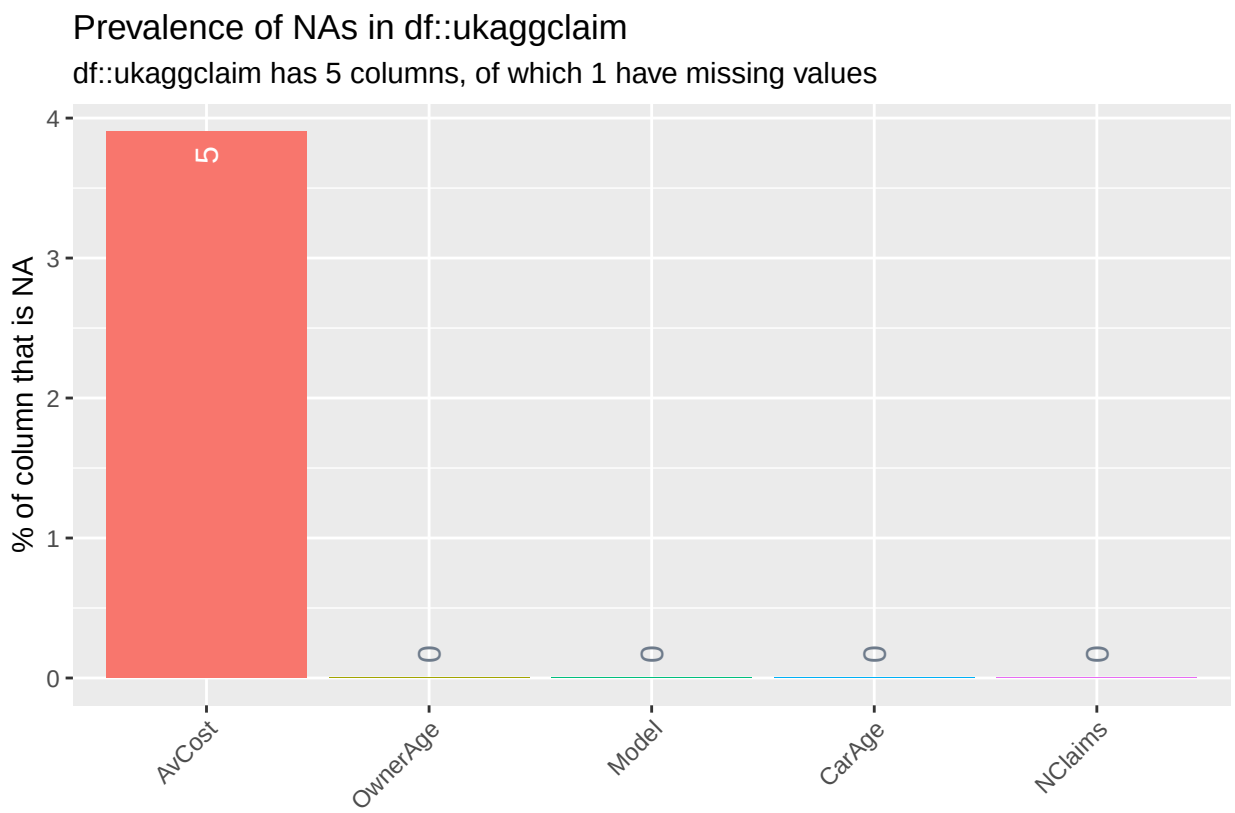


Ver el número de datos no disponibles o "celdas vacias"

```
inspect_na(ukaggclaim)
```

```
## # A tibble: 5 x 3
##   col_name   cnt  pcnt
##   <chr>    <int> <dbl>
## 1 AvCost       5  3.91
## 2 OwnerAge     0  0
## 3 Model        0  0
## 4 CarAge       0  0
## 5 NClaims      0  0
```

```
show_plot(inspect_na(ukaggclaim))
```

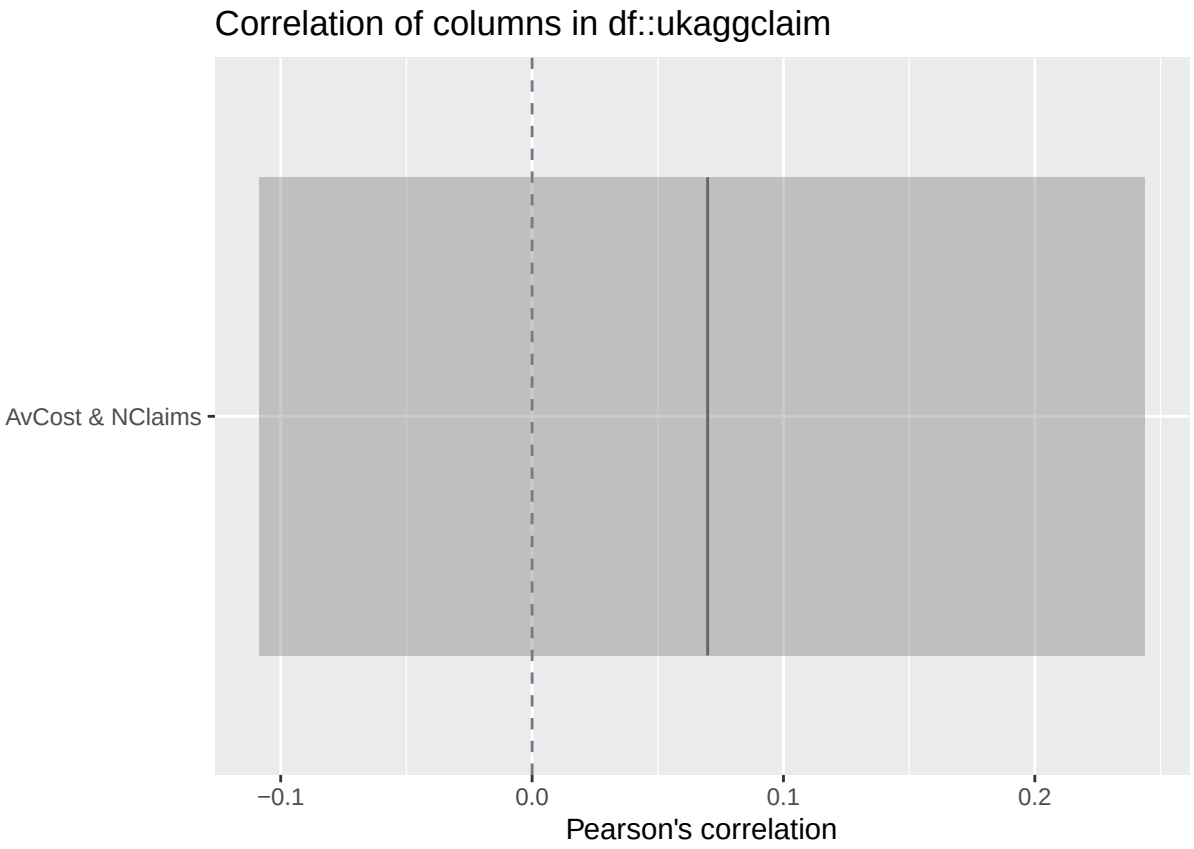


Y por último ver las correlaciones entre los datos de la tabla

```
inspect_cor(ukaggclaim)
```

```
## # A tibble: 1 x 6
##   col_1  col_2     corr p_value  lower upper
##   <chr>  <chr>    <dbl>   <dbl>  <dbl> <dbl>
## 1 AvCost NClaims 0.0698   0.444 -0.109 0.244
```

```
show_plot(inspect_cor(ukaggclaim))
```



6.2 Selecciones

Ahora que tenemos nuestros datos en un data frame, el próximo paso es acceder a ciertos datos de la tabla (una columna o grupo de filas que cumplan una condición), manipular y crear nuevas variables.

Podemos acceder a los datos del data frame del mismo modo que lo hacíamos con los vectores, los data frames no dejan de ser una especie de conjuntos de vectores

Recordar que podíamos acceder a elementos de los vectores por posición o nombre

```
head(ukaggclaim, n = 1)
```

```
##   OwnerAge Model CarAge NClaims AvCost
## 1    17-20     A    0-3       8    289
```

```
ukaggclaim[1,2]
```

```
## [1] A
## Levels: A B C D
```

```
ukaggclaim[1,4]
```

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

En el primer caso le decimos a R que queremos que nos muestre el dato de la primera fila y la segunda columna, en el segundo caso el dato de la primera fila y cuarta columna

En el caso que deseemos ver una columna o fila específica, dejamos una de las dimensiones en blanco

Si queremos acceder a la información de la segunda fila

```
ukaggclaim[2, ]
```

```
##   OwnerAge Model CarAge NClaims AvCost
## 2    17-20     A    4-7       8    282
```

Si queremos acceder a la información de la cuarta columna

```
ukaggclaim[ , 4]
```

```
##   [1]   8   8   4   1  10  28   1   1   9  13   1   0   3   2   0   0  18  31
##  [19]  10   4  59  96  13   3  44  39   7   2  24  18   2   0  56  55  17  12
##  [37] 125 172  36  10 163 129  18   8  72  50   6   1  43  53  15  12 179 211
##  [55]  39  19 197 125  30   9 104  55   8   2  43  73  21  14 191 219  46  23
##  [73] 210 131  32   8 119  43   4   0  90  98  35  22 380 434  97  59 401 253
##  [91]  50  15 199  88   8   9  69 120  42  35 366 353  95  45 310 148  33  13
## [109] 105  46  10   1  64 100  43  53 228 233  73  44 183 103  20   6  62  22
## [127]   6   6
```

Tambien podemos obtener la información por columna usando el nombre de la misma

```
ukaggclaim[ , "NClaims"]
```

```
##   [1]   8   8   4   1  10  28   1   1   9  13   1   0   3   2   0   0  18  31
##  [19]  10   4  59  96  13   3  44  39   7   2  24  18   2   0  56  55  17  12
##  [37] 125 172  36  10 163 129  18   8  72  50   6   1  43  53  15  12 179 211
##  [55]  39  19 197 125  30   9 104  55   8   2  43  73  21  14 191 219  46  23
##  [73] 210 131  32   8 119  43   4   0  90  98  35  22 380 434  97  59 401 253
##  [91]  50  15 199  88   8   9  69 120  42  35 366 353  95  45 310 148  33  13
## [109] 105  46  10   1  64 100  43  53 228 233  73  44 183 103  20   6  62  22
## [127]   6   6
```

Asimismo, podemos inspeccionar una columna de data frame escribiendo el nombre del mismo seguido de un signo $ y el nombre de la columna

La ventaja de este método es que los nombres de la columna se autocompletan

```
ukaggclaim$NClaims
```

```
##   [1]   8   8   4   1  10  28   1   1   9  13   1   0   3   2   0   0  18  31
##  [19]  10   4  59  96  13   3  44  39   7   2  24  18   2   0  56  55  17  12
##  [37] 125 172  36  10 163 129  18   8  72  50   6   1  43  53  15  12 179 211
##  [55]  39  19 197 125  30   9 104  55   8   2  43  73  21  14 191 219  46  23
##  [73] 210 131  32   8 119  43   4   0  90  98  35  22 380 434  97  59 401 253
##  [91]  50  15 199  88   8   9  69 120  42  35 366 353  95  45 310 148  33  13
## [109] 105  46  10   1  64 100  43  53 228 233  73  44 183 103  20   6  62  22
## [127]   6   6
```

Con la posición, al igual que los vectores, podemos acceder a conjuntos de filas y columnas (slices)

Por ejemplo, si nos queremos quedar solo con las 3 primeras filas y la cuarta y quinta columna

```
ukaggclaim[1:3, 4:5]
```

```
##   NClaims AvCost
## 1       8    289
## 2       8    282
## 3       4    133
```

Podemos construir una tabla nueva a partir de esta sub-tabla, asignando la última a un nombre

```r
uk_peque <-  ukaggclaim[1:3, 4:5]

uk_peque
```

```
##   NClaims AvCost
## 1       8    289
## 2       8    282
## 3       4    133
```

Sub-tabla solo con las 10 primeras filas

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

```
##    OwnerAge Model CarAge NClaims AvCost
## 1     17-20     A    0-3       8    289
## 2     17-20     A    4-7       8    282
## 3     17-20     A    8-9       4    133
## 4     17-20     A    10+       1    160
## 5     17-20     B    0-3      10    372
## 6     17-20     B    4-7      28    249
## 7     17-20     B    8-9       1    288
## 8     17-20     B    10+       1     11
## 9     17-20     C    0-3       9    189
## 10    17-20     C    4-7      13    288
```

Sub-tabla solo con las 2 últimas columnas

```r
ukaggclaim[, 4:5]
```

```
##     NClaims AvCost
## 1         8    289
## 2         8    282
## 3         4    133
## 4         1    160
## 5        10    372
## 6        28    249
## 7         1    288
## 8         1     11
## 9         9    189
## 10       13    288
## 11        1    179
## 12        0     NA
## 13        3    763
## 14        2    850
## 15        0     NA
## 16        0     NA
## 17       18    302
## 18       31    194
## 19       10    135
## 20        4    166
## 21       59    420
## 22       96    243
## 23       13    196
## 24        3    135
## 25       44    268
## 26       39    343
```

```
## 27        7    293
## 28        2    104
## 29       24    407
## 30       18    320
## 31        2    205
## 32        0     NA
## 33       56    268
## 34       55    285
## 35       17    181
## 36       12    110
## 37      125    275
## 38      172    243
## 39       36    179
## 40       10    264
## 41      163    334
## 42      129    274
## 43       18    208
## 44        8    150
## 45       72    383
## 46       50    305
## 47        6    116
## 48        1    636
## 49       43    236
## 50       53    270
## 51       15    160
## 52       12    110
## 53      179    259
## 54      211    226
## 55       39    161
## 56       19    107
## 57      197    340
## 58      125    260
## 59       30    189
## 60        9    104
## 61      104    400
## 62       55    349
## 63        8    147
## 64        2     65
## 65       43    207
## 66       73    129
## 67       21    157
## 68       14    113
## 69      191    208
## 70      219    214
## 71       46    149
## 72       23    137
## 73      210    251
## 74      131    232
## 75       32    203
## 76        8    141
## 77      119    233
## 78       43    325
## 79        4    207
## 80        0     NA
```

```
## 81       90    254
## 82       98    213
## 83       35    149
## 84       22     98
## 85      380    218
## 86      434    209
## 87       97    172
## 88       59    110
## 89      401    239
## 90      253    250
## 91       50    174
## 92       15    129
## 93      199    387
## 94       88    299
## 95        8    325
## 96        9    137
## 97       69    251
## 98      120    227
## 99       42    172
## 100      35     98
## 101     366    196
## 102     353    229
## 103      95    164
## 104      45    132
## 105     310    268
## 106     148    250
## 107      33    175
## 108      13    152
## 109     105    391
## 110      46    228
## 111      10    346
## 112       1    167
## 113      64    264
## 114     100    198
## 115      43    167
## 116      53    114
## 117     228    224
## 118     233    193
## 119      73    178
## 120      44    101
## 121     183    269
## 122     103    258
## 123      20    227
## 124       6    119
## 125      62    385
## 126      22    324
## 127       6    192
## 128       6    123
```

Podemos seleccionar datos con criterios según cumplan una condición

```
ukaggclaim[ukaggclaim$AvCost > 300, ]
```

```
##      OwnerAge Model CarAge NClaims AvCost
## 5       17-20     B    0-3      10    372
```

```
## NA       <NA>  <NA>   <NA>      NA     NA
## 13      17-20     D    0-3       3    763
## 14      17-20     D    4-7       2    850
## NA.1     <NA>  <NA>   <NA>      NA     NA
## NA.2     <NA>  <NA>   <NA>      NA     NA
## 17      21-24     A    0-3      18    302
## 21      21-24     B    0-3      59    420
## 26      21-24     C    4-7      39    343
## 29      21-24     D    0-3      24    407
## 30      21-24     D    4-7      18    320
## NA.3     <NA>  <NA>   <NA>      NA     NA
## 41      25-29     C    0-3     163    334
## 45      25-29     D    0-3      72    383
## 46      25-29     D    4-7      50    305
## 48      25-29     D    10+       1    636
## 57      30-34     C    0-3     197    340
## 61      30-34     D    0-3     104    400
## 62      30-34     D    4-7      55    349
## 78      35-39     D    4-7      43    325
## NA.4     <NA>  <NA>   <NA>      NA     NA
## 93      40-49     D    0-3     199    387
## 95      40-49     D    8-9       8    325
## 109     50-59     D    0-3     105    391
## 111     50-59     D    8-9      10    346
## 125       60+     D    0-3      62    385
## 126       60+     D    4-7      22    324
```

```
ukaggclaim[ukaggclaim$Model == "A", ]
```

```
##     OwnerAge Model CarAge NClaims AvCost
## 1      17-20     A    0-3       8    289
## 2      17-20     A    4-7       8    282
## 3      17-20     A    8-9       4    133
## 4      17-20     A    10+       1    160
## 17     21-24     A    0-3      18    302
## 18     21-24     A    4-7      31    194
## 19     21-24     A    8-9      10    135
## 20     21-24     A    10+       4    166
## 33     25-29     A    0-3      56    268
## 34     25-29     A    4-7      55    285
## 35     25-29     A    8-9      17    181
## 36     25-29     A    10+      12    110
## 49     30-34     A    0-3      43    236
## 50     30-34     A    4-7      53    270
## 51     30-34     A    8-9      15    160
## 52     30-34     A    10+      12    110
## 65     35-39     A    0-3      43    207
## 66     35-39     A    4-7      73    129
## 67     35-39     A    8-9      21    157
## 68     35-39     A    10+      14    113
## 81     40-49     A    0-3      90    254
## 82     40-49     A    4-7      98    213
## 83     40-49     A    8-9      35    149
## 84     40-49     A    10+      22     98
## 97     50-59     A    0-3      69    251
```

```
## 98     50-59     A    4-7     120    227
## 99     50-59     A    8-9      42    172
## 100    50-59     A    10+      35     98
## 113      60+     A    0-3      64    264
## 114      60+     A    4-7     100    198
## 115      60+     A    8-9      43    167
## 116      60+     A    10+      53    114
```

Lo podemos complicar

```
ukaggclaim[ukaggclaim$Model == "A" & ukaggclaim$NClaims >= 100, ]
```

```
##     OwnerAge Model CarAge NClaims AvCost
## 98     50-59     A    4-7     120    227
## 114      60+     A    4-7     100    198
```

Le hemos pedido a R que nos muestre los datos de la tabla que correspoden a los modelos de coches "A" y que el número de solicitudes sea mayor o igual a 100

Ambas condiciones se cumplen de manera simultanea

```
ukaggclaim[ukaggclaim$Model == "A" | ukaggclaim$NClaims >= 100, ]
```

```
##     OwnerAge Model CarAge NClaims AvCost
## 1      17-20     A    0-3       8    289
## 2      17-20     A    4-7       8    282
## 3      17-20     A    8-9       4    133
## 4      17-20     A    10+       1    160
## 17     21-24     A    0-3      18    302
## 18     21-24     A    4-7      31    194
## 19     21-24     A    8-9      10    135
## 20     21-24     A    10+       4    166
## 33     25-29     A    0-3      56    268
## 34     25-29     A    4-7      55    285
## 35     25-29     A    8-9      17    181
## 36     25-29     A    10+      12    110
## 37     25-29     B    0-3     125    275
## 38     25-29     B    4-7     172    243
## 41     25-29     C    0-3     163    334
## 42     25-29     C    4-7     129    274
## 49     30-34     A    0-3      43    236
## 50     30-34     A    4-7      53    270
## 51     30-34     A    8-9      15    160
## 52     30-34     A    10+      12    110
## 53     30-34     B    0-3     179    259
## 54     30-34     B    4-7     211    226
## 57     30-34     C    0-3     197    340
## 58     30-34     C    4-7     125    260
## 61     30-34     D    0-3     104    400
## 65     35-39     A    0-3      43    207
## 66     35-39     A    4-7      73    129
## 67     35-39     A    8-9      21    157
## 68     35-39     A    10+      14    113
## 69     35-39     B    0-3     191    208
## 70     35-39     B    4-7     219    214
## 73     35-39     C    0-3     210    251
## 74     35-39     C    4-7     131    232
```

```
## 77     35-39     D    0-3     119    233
## 81     40-49     A    0-3      90    254
## 82     40-49     A    4-7      98    213
## 83     40-49     A    8-9      35    149
## 84     40-49     A    10+      22     98
## 85     40-49     B    0-3     380    218
## 86     40-49     B    4-7     434    209
## 89     40-49     C    0-3     401    239
## 90     40-49     C    4-7     253    250
## 93     40-49     D    0-3     199    387
## 97     50-59     A    0-3      69    251
## 98     50-59     A    4-7     120    227
## 99     50-59     A    8-9      42    172
## 100    50-59     A    10+      35     98
## 101    50-59     B    0-3     366    196
## 102    50-59     B    4-7     353    229
## 105    50-59     C    0-3     310    268
## 106    50-59     C    4-7     148    250
## 109    50-59     D    0-3     105    391
## 113      60+     A    0-3      64    264
## 114      60+     A    4-7     100    198
## 115      60+     A    8-9      43    167
## 116      60+     A    10+      53    114
## 117      60+     B    0-3     228    224
## 118      60+     B    4-7     233    193
## 121      60+     C    0-3     183    269
## 122      60+     C    4-7     103    258
```

Podemos inspeccionar columnas individuales

```
summary(ukaggclaim$AvCost)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's
##    11.0   158.5   213.0   231.1   272.0   850.0       5
```

```
table(ukaggclaim$OwnerAge)
```

```
##
## 17-20 21-24 25-29 30-34 35-39 40-49 50-59   60+
##    16    16    16    16    16    16    16    16
```

Algo que solemos hacer con mucha frecuencia en Excel, es ordenar una tabla por fila, esto también lo podemos hacer en R con la función `order`

```
ukaggclaim[ order(ukaggclaim$AvCost),]
```

```
##     OwnerAge Model CarAge NClaims AvCost
## 8      17-20     B    10+       1     11
## 64     30-34     D    10+       2     65
## 84     40-49     A    10+      22     98
## 100    50-59     A    10+      35     98
## 120      60+     B    10+      44    101
## 28     21-24     C    10+       2    104
## 60     30-34     C    10+       9    104
## 56     30-34     B    10+      19    107
## 36     25-29     A    10+      12    110
## 52     30-34     A    10+      12    110
```

```
## 88     40-49     B    10+      59    110
## 68     35-39     A    10+      14    113
## 116      60+     A    10+      53    114
## 47     25-29     D    8-9       6    116
## 124      60+     C    10+       6    119
## 128      60+     D    10+       6    123
## 66     35-39     A    4-7      73    129
## 92     40-49     C    10+      15    129
## 104    50-59     B    10+      45    132
## 3      17-20     A    8-9       4    133
## 19     21-24     A    8-9      10    135
## 24     21-24     B    10+       3    135
## 72     35-39     B    10+      23    137
## 96     40-49     D    10+       9    137
## 76     35-39     C    10+       8    141
## 63     30-34     D    8-9       8    147
## 71     35-39     B    8-9      46    149
## 83     40-49     A    8-9      35    149
## 44     25-29     C    10+       8    150
## 108    50-59     C    10+      13    152
## 67     35-39     A    8-9      21    157
## 4      17-20     A    10+       1    160
## 51     30-34     A    8-9      15    160
## 55     30-34     B    8-9      39    161
## 103    50-59     B    8-9      95    164
## 20     21-24     A    10+       4    166
## 112    50-59     D    10+       1    167
## 115      60+     A    8-9      43    167
## 87     40-49     B    8-9      97    172
## 99     50-59     A    8-9      42    172
## 91     40-49     C    8-9      50    174
## 107    50-59     C    8-9      33    175
## 119      60+     B    8-9      73    178
## 11     17-20     C    8-9       1    179
## 39     25-29     B    8-9      36    179
## 35     25-29     A    8-9      17    181
## 9      17-20     C    0-3       9    189
## 59     30-34     C    8-9      30    189
## 127      60+     D    8-9       6    192
## 118      60+     B    4-7     233    193
## 18     21-24     A    4-7      31    194
## 23     21-24     B    8-9      13    196
## 101    50-59     B    0-3     366    196
## 114      60+     A    4-7     100    198
## 75     35-39     C    8-9      32    203
## 31     21-24     D    8-9       2    205
## 65     35-39     A    0-3      43    207
## 79     35-39     D    8-9       4    207
## 43     25-29     C    8-9      18    208
## 69     35-39     B    0-3     191    208
## 86     40-49     B    4-7     434    209
## 82     40-49     A    4-7      98    213
## 70     35-39     B    4-7     219    214
## 85     40-49     B    0-3     380    218
```

```
## 117      60+     B    0-3     228    224
## 54     30-34     B    4-7     211    226
## 98     50-59     A    4-7     120    227
## 123      60+     C    8-9      20    227
## 110    50-59     D    4-7      46    228
## 102    50-59     B    4-7     353    229
## 74     35-39     C    4-7     131    232
## 77     35-39     D    0-3     119    233
## 49     30-34     A    0-3      43    236
## 89     40-49     C    0-3     401    239
## 22     21-24     B    4-7      96    243
## 38     25-29     B    4-7     172    243
## 6      17-20     B    4-7      28    249
## 90     40-49     C    4-7     253    250
## 106    50-59     C    4-7     148    250
## 73     35-39     C    0-3     210    251
## 97     50-59     A    0-3      69    251
## 81     40-49     A    0-3      90    254
## 122      60+     C    4-7     103    258
## 53     30-34     B    0-3     179    259
## 58     30-34     C    4-7     125    260
## 40     25-29     B    10+      10    264
## 113      60+     A    0-3      64    264
## 25     21-24     C    0-3      44    268
## 33     25-29     A    0-3      56    268
## 105    50-59     C    0-3     310    268
## 121      60+     C    0-3     183    269
## 50     30-34     A    4-7      53    270
## 42     25-29     C    4-7     129    274
## 37     25-29     B    0-3     125    275
## 2      17-20     A    4-7       8    282
## 34     25-29     A    4-7      55    285
## 7      17-20     B    8-9       1    288
## 10     17-20     C    4-7      13    288
## 1      17-20     A    0-3       8    289
## 27     21-24     C    8-9       7    293
## 94     40-49     D    4-7      88    299
## 17     21-24     A    0-3      18    302
## 46     25-29     D    4-7      50    305
## 30     21-24     D    4-7      18    320
## 126      60+     D    4-7      22    324
## 78     35-39     D    4-7      43    325
## 95     40-49     D    8-9       8    325
## 41     25-29     C    0-3     163    334
## 57     30-34     C    0-3     197    340
## 26     21-24     C    4-7      39    343
## 111    50-59     D    8-9      10    346
## 62     30-34     D    4-7      55    349
## 5      17-20     B    0-3      10    372
## 45     25-29     D    0-3      72    383
## 125      60+     D    0-3      62    385
## 93     40-49     D    0-3     199    387
## 109    50-59     D    0-3     105    391
## 61     30-34     D    0-3     104    400
```

```
## 29     21-24     D    0-3      24    407
## 21     21-24     B    0-3      59    420
## 48     25-29     D    10+       1    636
## 13     17-20     D    0-3       3    763
## 14     17-20     D    4-7       2    850
## 12     17-20     C    10+       0     NA
## 15     17-20     D    8-9       0     NA
## 16     17-20     D    10+       0     NA
## 32     21-24     D    10+       0     NA
## 80     35-39     D    10+       0     NA
```

Hemos ordenado de menor a mayor, si quisieramos hacer lo contrario solo debemos colocar un signo menos - delante de la columna

```
ukaggclaim[ order(-ukaggclaim$AvCost),]
```

```
##     OwnerAge Model CarAge NClaims AvCost
## 14     17-20     D    4-7       2    850
## 13     17-20     D    0-3       3    763
## 48     25-29     D    10+       1    636
## 21     21-24     B    0-3      59    420
## 29     21-24     D    0-3      24    407
## 61     30-34     D    0-3     104    400
## 109    50-59     D    0-3     105    391
## 93     40-49     D    0-3     199    387
## 125      60+     D    0-3      62    385
## 45     25-29     D    0-3      72    383
## 5      17-20     B    0-3      10    372
## 62     30-34     D    4-7      55    349
## 111    50-59     D    8-9      10    346
## 26     21-24     C    4-7      39    343
## 57     30-34     C    0-3     197    340
## 41     25-29     C    0-3     163    334
## 78     35-39     D    4-7      43    325
## 95     40-49     D    8-9       8    325
## 126      60+     D    4-7      22    324
## 30     21-24     D    4-7      18    320
## 46     25-29     D    4-7      50    305
## 17     21-24     A    0-3      18    302
## 94     40-49     D    4-7      88    299
## 27     21-24     C    8-9       7    293
## 1      17-20     A    0-3       8    289
## 7      17-20     B    8-9       1    288
## 10     17-20     C    4-7      13    288
## 34     25-29     A    4-7      55    285
## 2      17-20     A    4-7       8    282
## 37     25-29     B    0-3     125    275
## 42     25-29     C    4-7     129    274
## 50     30-34     A    4-7      53    270
## 121      60+     C    0-3     183    269
## 25     21-24     C    0-3      44    268
## 33     25-29     A    0-3      56    268
## 105    50-59     C    0-3     310    268
## 40     25-29     B    10+      10    264
## 113      60+     A    0-3      64    264
```

```
## 58     30-34     C    4-7     125    260
## 53     30-34     B    0-3     179    259
## 122      60+     C    4-7     103    258
## 81     40-49     A    0-3      90    254
## 73     35-39     C    0-3     210    251
## 97     50-59     A    0-3      69    251
## 90     40-49     C    4-7     253    250
## 106    50-59     C    4-7     148    250
## 6      17-20     B    4-7      28    249
## 22     21-24     B    4-7      96    243
## 38     25-29     B    4-7     172    243
## 89     40-49     C    0-3     401    239
## 49     30-34     A    0-3      43    236
## 77     35-39     D    0-3     119    233
## 74     35-39     C    4-7     131    232
## 102    50-59     B    4-7     353    229
## 110    50-59     D    4-7      46    228
## 98     50-59     A    4-7     120    227
## 123      60+     C    8-9      20    227
## 54     30-34     B    4-7     211    226
## 117      60+     B    0-3     228    224
## 85     40-49     B    0-3     380    218
## 70     35-39     B    4-7     219    214
## 82     40-49     A    4-7      98    213
## 86     40-49     B    4-7     434    209
## 43     25-29     C    8-9      18    208
## 69     35-39     B    0-3     191    208
## 65     35-39     A    0-3      43    207
## 79     35-39     D    8-9       4    207
## 31     21-24     D    8-9       2    205
## 75     35-39     C    8-9      32    203
## 114      60+     A    4-7     100    198
## 23     21-24     B    8-9      13    196
## 101    50-59     B    0-3     366    196
## 18     21-24     A    4-7      31    194
## 118      60+     B    4-7     233    193
## 127      60+     D    8-9       6    192
## 9      17-20     C    0-3       9    189
## 59     30-34     C    8-9      30    189
## 35     25-29     A    8-9      17    181
## 11     17-20     C    8-9       1    179
## 39     25-29     B    8-9      36    179
## 119      60+     B    8-9      73    178
## 107    50-59     C    8-9      33    175
## 91     40-49     C    8-9      50    174
## 87     40-49     B    8-9      97    172
## 99     50-59     A    8-9      42    172
## 112    50-59     D    10+       1    167
## 115      60+     A    8-9      43    167
## 20     21-24     A    10+       4    166
## 103    50-59     B    8-9      95    164
## 55     30-34     B    8-9      39    161
## 4      17-20     A    10+       1    160
## 51     30-34     A    8-9      15    160
```

```
## 67     35-39     A    8-9      21    157
## 108    50-59     C    10+      13    152
## 44     25-29     C    10+       8    150
## 71     35-39     B    8-9      46    149
## 83     40-49     A    8-9      35    149
## 63     30-34     D    8-9       8    147
## 76     35-39     C    10+       8    141
## 72     35-39     B    10+      23    137
## 96     40-49     D    10+       9    137
## 19     21-24     A    8-9      10    135
## 24     21-24     B    10+       3    135
## 3      17-20     A    8-9       4    133
## 104    50-59     B    10+      45    132
## 66     35-39     A    4-7      73    129
## 92     40-49     C    10+      15    129
## 128      60+     D    10+       6    123
## 124      60+     C    10+       6    119
## 47     25-29     D    8-9       6    116
## 116      60+     A    10+      53    114
## 68     35-39     A    10+      14    113
## 36     25-29     A    10+      12    110
## 52     30-34     A    10+      12    110
## 88     40-49     B    10+      59    110
## 56     30-34     B    10+      19    107
## 28     21-24     C    10+       2    104
## 60     30-34     C    10+       9    104
## 120      60+     B    10+      44    101
## 84     40-49     A    10+      22     98
## 100    50-59     A    10+      35     98
## 64     30-34     D    10+       2     65
## 8      17-20     B    10+       1     11
## 12     17-20     C    10+       0     NA
## 15     17-20     D    8-9       0     NA
## 16     17-20     D    10+       0     NA
## 32     21-24     D    10+       0     NA
## 80     35-39     D    10+       0     NA
```

Hemos observado que hay una que otra fila con valores `NA`. Este tipo de variables sirve para denotar datos que no están disponibles, *celdas vacías* en lenguaje de excel

Si quisieramos eliminar dichas filas, usamos la función `na.omit`

```
dim(ukaggclaim)
```

```
## [1] 128   5
```

```
uk_limpio <- na.omit(ukaggclaim)

uk_limpio
```

```
##     OwnerAge Model CarAge NClaims AvCost
## 1      17-20     A    0-3       8    289
## 2      17-20     A    4-7       8    282
## 3      17-20     A    8-9       4    133
## 4      17-20     A    10+       1    160
## 5      17-20     B    0-3      10    372
```

```
## 6      17-20     B    4-7      28    249
## 7      17-20     B    8-9       1    288
## 8      17-20     B    10+       1     11
## 9      17-20     C    0-3       9    189
## 10     17-20     C    4-7      13    288
## 11     17-20     C    8-9       1    179
## 13     17-20     D    0-3       3    763
## 14     17-20     D    4-7       2    850
## 17     21-24     A    0-3      18    302
## 18     21-24     A    4-7      31    194
## 19     21-24     A    8-9      10    135
## 20     21-24     A    10+       4    166
## 21     21-24     B    0-3      59    420
## 22     21-24     B    4-7      96    243
## 23     21-24     B    8-9      13    196
## 24     21-24     B    10+       3    135
## 25     21-24     C    0-3      44    268
## 26     21-24     C    4-7      39    343
## 27     21-24     C    8-9       7    293
## 28     21-24     C    10+       2    104
## 29     21-24     D    0-3      24    407
## 30     21-24     D    4-7      18    320
## 31     21-24     D    8-9       2    205
## 33     25-29     A    0-3      56    268
## 34     25-29     A    4-7      55    285
## 35     25-29     A    8-9      17    181
## 36     25-29     A    10+      12    110
## 37     25-29     B    0-3     125    275
## 38     25-29     B    4-7     172    243
## 39     25-29     B    8-9      36    179
## 40     25-29     B    10+      10    264
## 41     25-29     C    0-3     163    334
## 42     25-29     C    4-7     129    274
## 43     25-29     C    8-9      18    208
## 44     25-29     C    10+       8    150
## 45     25-29     D    0-3      72    383
## 46     25-29     D    4-7      50    305
## 47     25-29     D    8-9       6    116
## 48     25-29     D    10+       1    636
## 49     30-34     A    0-3      43    236
## 50     30-34     A    4-7      53    270
## 51     30-34     A    8-9      15    160
## 52     30-34     A    10+      12    110
## 53     30-34     B    0-3     179    259
## 54     30-34     B    4-7     211    226
## 55     30-34     B    8-9      39    161
## 56     30-34     B    10+      19    107
## 57     30-34     C    0-3     197    340
## 58     30-34     C    4-7     125    260
## 59     30-34     C    8-9      30    189
## 60     30-34     C    10+       9    104
## 61     30-34     D    0-3     104    400
## 62     30-34     D    4-7      55    349
## 63     30-34     D    8-9       8    147
```

```
## 64     30-34     D    10+       2     65
## 65     35-39     A    0-3      43    207
## 66     35-39     A    4-7      73    129
## 67     35-39     A    8-9      21    157
## 68     35-39     A    10+      14    113
## 69     35-39     B    0-3     191    208
## 70     35-39     B    4-7     219    214
## 71     35-39     B    8-9      46    149
## 72     35-39     B    10+      23    137
## 73     35-39     C    0-3     210    251
## 74     35-39     C    4-7     131    232
## 75     35-39     C    8-9      32    203
## 76     35-39     C    10+       8    141
## 77     35-39     D    0-3     119    233
## 78     35-39     D    4-7      43    325
## 79     35-39     D    8-9       4    207
## 81     40-49     A    0-3      90    254
## 82     40-49     A    4-7      98    213
## 83     40-49     A    8-9      35    149
## 84     40-49     A    10+      22     98
## 85     40-49     B    0-3     380    218
## 86     40-49     B    4-7     434    209
## 87     40-49     B    8-9      97    172
## 88     40-49     B    10+      59    110
## 89     40-49     C    0-3     401    239
## 90     40-49     C    4-7     253    250
## 91     40-49     C    8-9      50    174
## 92     40-49     C    10+      15    129
## 93     40-49     D    0-3     199    387
## 94     40-49     D    4-7      88    299
## 95     40-49     D    8-9       8    325
## 96     40-49     D    10+       9    137
## 97     50-59     A    0-3      69    251
## 98     50-59     A    4-7     120    227
## 99     50-59     A    8-9      42    172
## 100    50-59     A    10+      35     98
## 101    50-59     B    0-3     366    196
## 102    50-59     B    4-7     353    229
## 103    50-59     B    8-9      95    164
## 104    50-59     B    10+      45    132
## 105    50-59     C    0-3     310    268
## 106    50-59     C    4-7     148    250
## 107    50-59     C    8-9      33    175
## 108    50-59     C    10+      13    152
## 109    50-59     D    0-3     105    391
## 110    50-59     D    4-7      46    228
## 111    50-59     D    8-9      10    346
## 112    50-59     D    10+       1    167
## 113      60+     A    0-3      64    264
## 114      60+     A    4-7     100    198
## 115      60+     A    8-9      43    167
## 116      60+     A    10+      53    114
## 117      60+     B    0-3     228    224
## 118      60+     B    4-7     233    193
```

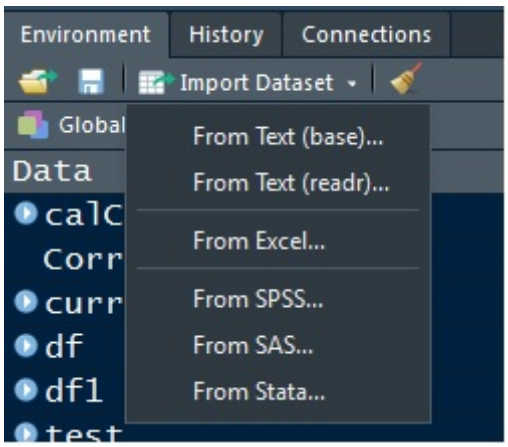


Figure 2:

```
## 119      60+     B    8-9      73    178
## 120      60+     B    10+      44    101
## 121      60+     C    0-3     183    269
## 122      60+     C    4-7     103    258
## 123      60+     C    8-9      20    227
## 124      60+     C    10+       6    119
## 125      60+     D    0-3      62    385
## 126      60+     D    4-7      22    324
## 127      60+     D    8-9       6    192
## 128      60+     D    10+       6    123
```

```
dim(uk_limpio)
```

```
## [1] 123   5
```

En ocasiones no deseamos eliminar las filas, sino imputar algún valor

```
ukaggclaim$AvCost[is.na(ukaggclaim$AvCost)] <- 0
```

7 Input - Output (o simplemente como cargar y guardar datos en R)

Hemos estado trabajando con variables, vectores, data frames (tablas), pero a estas alturas quizás ya nos hemos percatado que al cerrar RStudio todas estas variables se *pierden*

Para evitar esto podemos guardar los datos de diversas formas y formatos

Una opción es guardarlo como un archivo externo, por ejemplo un libro de Excel o un archivo csv

La otra opción es guardarlo como un archivo de R, al cual podamos acceder simplemente llamando la tabla o variable por su nombre

7.1 Cargar datos de fuentes externas

Podemos cargar datos directamente con el RStudio

Lo que hemos hecho antes vía menú, se ha ejecutado en R con este código

```
pacman::p_load(readxl)
empleados <- read_excel("empleados.xls", sheet = "Resultados", na = "n.d.")
```

Recordar que podemos visualizar la tabla con `View()`

```
View(empleados)
```

Observamos que la primera columna no es útil, la podemos eliminar facilmente, pero antes de hacerlo inspeccionemos información relativa al número de columnas

```
head(empleados)
```

```
## # A tibble: 6 x 8
##   X__1  `Nombre empresa` `NACE Rev. 2\nC~ `Número de iden~ `Número emplead~
##   <chr> <chr>            <chr>            <chr>                       <dbl>
## 1 1.    EBRO FOOD, S.A.  1081             ESA47412333                  6145
## 2 2.    BUNGE IBERICA SA 1041             ESA81473852                   329
## 3 3.    CAMPOFRIO FOOD ~ 1013             ESA09000928                  6794
## 4 4.    NESTLE ESPANA S~ 1086             ESA08005449                  4192
## 5 5.    MAHOU, SA        1105             ESA28078202                  2947
## 6 6.    SOCIEDAD ANONIM~ 1105             ESA08000820                  3132
## # ... with 3 more variables: `Número empleados\n2015` <dbl>, `Número
## #   empleados\n2014` <dbl>, `Número empleados\n2013` <dbl>
```

```
ncol(empleados)
```

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

```
colnames(empleados)
```

```
## [1] "X__1"
## [2] "Nombre empresa"
## [3] "NACE Rev. 2\nCódigo principal (4 digitos)"
## [4] "Número de identificación BvD"
## [5] "Número empleados\n2016"
## [6] "Número empleados\n2015"
## [7] "Número empleados\n2014"
## [8] "Número empleados\n2013"
```

Antes de eliminar la columna creamos una nueva tabla igual a la original, esto se suele hacer si por alguna razón deseas conservar los datos originales

```
empleados_2 <- empleados
```

Eliminamos la columna asignandole un valor `NULL`

```
empleados_2$X__1 <- NULL
```

Repetimos la inspección anterior para verificar que ha sido eliminada la columna

```
head(empleados_2)
```

```
## # A tibble: 6 x 7
##   `Nombre empresa` `NACE Rev. 2\nC~ `Número de iden~ `Número emplead~
##   <chr>            <chr>            <chr>                       <dbl>
## 1 EBRO FOOD, S.A.  1081             ESA47412333                  6145
## 2 BUNGE IBERICA SA 1041             ESA81473852                   329
## 3 CAMPOFRIO FOOD ~ 1013             ESA09000928                  6794
## 4 NESTLE ESPANA S~ 1086             ESA08005449                  4192
```

```
## 5 MAHOU, SA        1105             ESA28078202                  2947
## 6 SOCIEDAD ANONIM~ 1105             ESA08000820                  3132
## # ... with 3 more variables: `Número empleados\n2015` <dbl>, `Número
## #   empleados\n2014` <dbl>, `Número empleados\n2013` <dbl>
```

```
ncol(empleados_2)
```

```
## [1] 7
```

```
colnames(empleados_2)
```

```
## [1] "Nombre empresa"
## [2] "NACE Rev. 2\nCódigo principal (4 digitos)"
## [3] "Número de identificación BvD"
## [4] "Número empleados\n2016"
## [5] "Número empleados\n2015"
## [6] "Número empleados\n2014"
## [7] "Número empleados\n2013"
```

Como se puede observar en el menú, se pueden cargar archivos de distintos formatos, otro formato común es csv

```
airquality <- read.csv("airquality.csv", stringsAsFactors=FALSE)

View(airquality)
```

Si deseamos guardar un archivo en excel o csv lo hacemos del siguiente modo:

```
pacman::p_load(xlsx)

aire <- airquality

write.xlsx(x =  aire, file = "aire.xlsx", row.names = FALSE)

write.csv(x = aire, file = "aire2.csv")
```

Por otro lado podemos guardar datos como un archivo `Rdata`, el cual podremos simplemente cargar posteriormente en R

```
save(aire, file = "aire.RData")
```

Usamos el comando `rm()` para remover del espacio de trabajo la tabla aire

```
rm(aire)
```

El objeto ya no existe en memoria (ver la ventana de environment), pero como ya lo hemos guardado lo podemos recuperar con el comando `load()`

```
load(file = "aire.RData")
```

Si miramos la venta de environment observamos que hemos recuperado la tabla

```
aire
```

```
##     Ozone Solar.R Wind Temp Month Day
## 1      41     190  7.4   67     5   1
## 2      36     118  8.0   72     5   2
## 3      12     149 12.6   74     5   3
## 4      18     313 11.5   62     5   4
## 5      NA      NA 14.3   56     5   5
## 6      28      NA 14.9   66     5   6
```

```
## 7      23     299  8.6   65     5   7
## 8      19      99 13.8   59     5   8
## 9       8      19 20.1   61     5   9
## 10     NA     194  8.6   69     5  10
## 11      7      NA  6.9   74     5  11
## 12     16     256  9.7   69     5  12
## 13     11     290  9.2   66     5  13
## 14     14     274 10.9   68     5  14
## 15     18      65 13.2   58     5  15
## 16     14     334 11.5   64     5  16
## 17     34     307 12.0   66     5  17
## 18      6      78 18.4   57     5  18
## 19     30     322 11.5   68     5  19
## 20     11      44  9.7   62     5  20
## 21      1       8  9.7   59     5  21
## 22     11     320 16.6   73     5  22
## 23      4      25  9.7   61     5  23
## 24     32      92 12.0   61     5  24
## 25     NA      66 16.6   57     5  25
## 26     NA     266 14.9   58     5  26
## 27     NA      NA  8.0   57     5  27
## 28     23      13 12.0   67     5  28
## 29     45     252 14.9   81     5  29
## 30    115     223  5.7   79     5  30
## 31     37     279  7.4   76     5  31
## 32     NA     286  8.6   78     6   1
## 33     NA     287  9.7   74     6   2
## 34     NA     242 16.1   67     6   3
## 35     NA     186  9.2   84     6   4
## 36     NA     220  8.6   85     6   5
## 37     NA     264 14.3   79     6   6
## 38     29     127  9.7   82     6   7
## 39     NA     273  6.9   87     6   8
## 40     71     291 13.8   90     6   9
## 41     39     323 11.5   87     6  10
## 42     NA     259 10.9   93     6  11
## 43     NA     250  9.2   92     6  12
## 44     23     148  8.0   82     6  13
## 45     NA     332 13.8   80     6  14
## 46     NA     322 11.5   79     6  15
## 47     21     191 14.9   77     6  16
## 48     37     284 20.7   72     6  17
## 49     20      37  9.2   65     6  18
## 50     12     120 11.5   73     6  19
## 51     13     137 10.3   76     6  20
## 52     NA     150  6.3   77     6  21
## 53     NA      59  1.7   76     6  22
## 54     NA      91  4.6   76     6  23
## 55     NA     250  6.3   76     6  24
## 56     NA     135  8.0   75     6  25
## 57     NA     127  8.0   78     6  26
## 58     NA      47 10.3   73     6  27
## 59     NA      98 11.5   80     6  28
## 60     NA      31 14.9   77     6  29
```

```
## 61     NA     138  8.0   83     6  30
## 62    135     269  4.1   84     7   1
## 63     49     248  9.2   85     7   2
## 64     32     236  9.2   81     7   3
## 65     NA     101 10.9   84     7   4
## 66     64     175  4.6   83     7   5
## 67     40     314 10.9   83     7   6
## 68     77     276  5.1   88     7   7
## 69     97     267  6.3   92     7   8
## 70     97     272  5.7   92     7   9
## 71     85     175  7.4   89     7  10
## 72     NA     139  8.6   82     7  11
## 73     10     264 14.3   73     7  12
## 74     27     175 14.9   81     7  13
## 75     NA     291 14.9   91     7  14
## 76      7      48 14.3   80     7  15
## 77     48     260  6.9   81     7  16
## 78     35     274 10.3   82     7  17
## 79     61     285  6.3   84     7  18
## 80     79     187  5.1   87     7  19
## 81     63     220 11.5   85     7  20
## 82     16       7  6.9   74     7  21
## 83     NA     258  9.7   81     7  22
## 84     NA     295 11.5   82     7  23
## 85     80     294  8.6   86     7  24
## 86    108     223  8.0   85     7  25
## 87     20      81  8.6   82     7  26
## 88     52      82 12.0   86     7  27
## 89     82     213  7.4   88     7  28
## 90     50     275  7.4   86     7  29
## 91     64     253  7.4   83     7  30
## 92     59     254  9.2   81     7  31
## 93     39      83  6.9   81     8   1
## 94      9      24 13.8   81     8   2
## 95     16      77  7.4   82     8   3
## 96     78      NA  6.9   86     8   4
## 97     35      NA  7.4   85     8   5
## 98     66      NA  4.6   87     8   6
## 99    122     255  4.0   89     8   7
## 100    89     229 10.3   90     8   8
## 101   110     207  8.0   90     8   9
## 102    NA     222  8.6   92     8  10
## 103    NA     137 11.5   86     8  11
## 104    44     192 11.5   86     8  12
## 105    28     273 11.5   82     8  13
## 106    65     157  9.7   80     8  14
## 107    NA      64 11.5   79     8  15
## 108    22      71 10.3   77     8  16
## 109    59      51  6.3   79     8  17
## 110    23     115  7.4   76     8  18
## 111    31     244 10.9   78     8  19
## 112    44     190 10.3   78     8  20
## 113    21     259 15.5   77     8  21
## 114     9      36 14.3   72     8  22
```

```
## 115    NA     255 12.6   75     8  23
## 116    45     212  9.7   79     8  24
## 117   168     238  3.4   81     8  25
## 118    73     215  8.0   86     8  26
## 119    NA     153  5.7   88     8  27
## 120    76     203  9.7   97     8  28
## 121   118     225  2.3   94     8  29
## 122    84     237  6.3   96     8  30
## 123    85     188  6.3   94     8  31
## 124    96     167  6.9   91     9   1
## 125    78     197  5.1   92     9   2
## 126    73     183  2.8   93     9   3
## 127    91     189  4.6   93     9   4
## 128    47      95  7.4   87     9   5
## 129    32      92 15.5   84     9   6
## 130    20     252 10.9   80     9   7
## 131    23     220 10.3   78     9   8
## 132    21     230 10.9   75     9   9
## 133    24     259  9.7   73     9  10
## 134    44     236 14.9   81     9  11
## 135    21     259 15.5   76     9  12
## 136    28     238  6.3   77     9  13
## 137     9      24 10.9   71     9  14
## 138    13     112 11.5   71     9  15
## 139    46     237  6.9   78     9  16
## 140    18     224 13.8   67     9  17
## 141    13      27 10.3   76     9  18
## 142    24     238 10.3   68     9  19
## 143    16     201  8.0   82     9  20
## 144    13     238 12.6   64     9  21
## 145    23      14  9.2   71     9  22
## 146    36     139 10.3   81     9  23
## 147     7      49 10.3   69     9  24
## 148    14      20 16.6   63     9  25
## 149    30     193  6.9   70     9  26
## 150    NA     145 13.2   77     9  27
## 151    14     191 14.3   75     9  28
## 152    18     131  8.0   76     9  29
## 153    20     223 11.5   68     9  30
```