Cuarto Laboratorio - Introducción al R

Bioestadística 2020

Data Frames

Los data frames es la manera usual de guardar y usar datos en R. Los data frame se parecen a las matrices, donde cada fila suele representar una observación (pacientes, por ejemplo), y las columnas representan variables (nombre, sexo, edad, temperatura, etc).

El software *R* trae varios data frames por defecto. Usaremos el data frame *mtcars* como ejemplo.

```
?mtcars
```

```
## starting httpd help server ... done
```

```
mtcars # Observar que visualizar el data frame entero no es práctico.
```

```
##                    mpg cyl  disp  hp drat    wt  qsec vs am gear carb
## Mazda RX4         21.0   6 160.0 110 3.90 2.620 16.46  0  1    4    4
## Mazda RX4 Wag     21.0   6 160.0 110 3.90 2.875 17.02  0  1    4    4
## Datsun 710        22.8   4 108.0  93 3.85 2.320 18.61  1  1    4    1
## Hornet 4 Drive    21.4   6 258.0 110 3.08 3.215 19.44  1  0    3    1
## Hornet Sportabout 18.7   8 360.0 175 3.15 3.440 17.02  0  0    3    2
## Valiant           18.1   6 225.0 105 2.76 3.460 20.22  1  0    3    1
## Duster 360        14.3   8 360.0 245 3.21 3.570 15.84  0  0    3    4
## Merc 240D         24.4   4 146.7  62 3.69 3.190 20.00  1  0    4    2
## Merc 230          22.8   4 140.8  95 3.92 3.150 22.90  1  0    4    2
## Merc 280          19.2   6 167.6 123 3.92 3.440 18.30  1  0    4    4
## Merc 280C         17.8   6 167.6 123 3.92 3.440 18.90  1  0    4    4
## Merc 450SE        16.4   8 275.8 180 3.07 4.070 17.40  0  0    3    3
## Merc 450SL        17.3   8 275.8 180 3.07 3.730 17.60  0  0    3    3
## Merc 450SLC       15.2   8 275.8 180 3.07 3.780 18.00  0  0    3    3
## Cadillac Fleetwood  10.4   8 472.0 205 2.93 5.250 17.98  0  0    3    4
## Lincoln Continental 10.4   8 460.0 215 3.00 5.424 17.82  0  0    3    4
## Chrysler Imperial   14.7   8 440.0 230 3.23 5.345 17.42  0  0    3    4
## Fiat 128          32.4   4  78.7  66 4.08 2.200 19.47  1  1    4    1
## Honda Civic       30.4   4  75.7  52 4.93 1.615 18.52  1  1    4    2
## Toyota Corolla    33.9   4  71.1  65 4.22 1.835 19.90  1  1    4    1
## Toyota Corona     21.5   4 120.1  97 3.70 2.465 20.01  1  0    3    1
## Dodge Challenger  15.5   8 318.0 150 2.76 3.520 16.87  0  0    3    2
## AMC Javelin       15.2   8 304.0 150 3.15 3.435 17.30  0  0    3    2
## Camaro Z28        13.3   8 350.0 245 3.73 3.840 15.41  0  0    3    4
## Pontiac Firebird  19.2   8 400.0 175 3.08 3.845 17.05  0  0    3    2
## Fiat X1-9         27.3   4  79.0  66 4.08 1.935 18.90  1  1    4    1
## Porsche 914-2     26.0   4 120.3  91 4.43 2.140 16.70  0  1    5    2
## Lotus Europa      30.4   4  95.1 113 3.77 1.513 16.90  1  1    5    2
## Ford Pantera L    15.8   8 351.0 264 4.22 3.170 14.50  0  1    5    4
## Ferrari Dino      19.7   6 145.0 175 3.62 2.770 15.50  0  1    5    6
## Maserati Bora     15.0   8 301.0 335 3.54 3.570 14.60  0  1    5    8
## Volvo 142E        21.4   4 121.0 109 4.11 2.780 18.60  1  1    4    2
```

Tenemos varias funciones que nos ayudan a visualizar el data frame, y tener una mejor idea acerca de los datos

```r
summary(mtcars)
```

```
##       mpg             cyl             disp             hp
##  Min.   :10.40   Min.   :4.000   Min.   : 71.1   Min.   : 52.0
##  1st Qu.:15.43   1st Qu.:4.000   1st Qu.:120.8   1st Qu.: 96.5
##  Median :19.20   Median :6.000   Median :196.3   Median :123.0
##  Mean   :20.09   Mean   :6.188   Mean   :230.7   Mean   :146.7
##  3rd Qu.:22.80   3rd Qu.:8.000   3rd Qu.:326.0   3rd Qu.:180.0
##  Max.   :33.90   Max.   :8.000   Max.   :472.0   Max.   :335.0
##       drat             wt             qsec             vs
##  Min.   :2.760   Min.   :1.513   Min.   :14.50   Min.   :0.0000
##  1st Qu.:3.080   1st Qu.:2.581   1st Qu.:16.89   1st Qu.:0.0000
##  Median :3.695   Median :3.325   Median :17.71   Median :0.0000
##  Mean   :3.597   Mean   :3.217   Mean   :17.85   Mean   :0.4375
##  3rd Qu.:3.920   3rd Qu.:3.610   3rd Qu.:18.90   3rd Qu.:1.0000
##  Max.   :4.930   Max.   :5.424   Max.   :22.90   Max.   :1.0000
##        am              gear            carb
##  Min.   :0.0000   Min.   :3.000   Min.   :1.000
##  1st Qu.:0.0000   1st Qu.:3.000   1st Qu.:2.000
##  Median :0.0000   Median :4.000   Median :2.000
##  Mean   :0.4062   Mean   :3.688   Mean   :2.812
##  3rd Qu.:1.0000   3rd Qu.:4.000   3rd Qu.:4.000
##  Max.   :1.0000   Max.   :5.000   Max.   :8.000
```

```r
str(mtcars)
```

```
## 'data.frame':    32 obs. of  11 variables:
##  $ mpg : num  21 21 22.8 21.4 18.7 18.1 14.3 24.4 22.8 19.2 ...
##  $ cyl : num  6 6 4 6 8 6 8 4 4 6 ...
##  $ disp: num  160 160 108 258 360 ...
##  $ hp  : num  110 110 93 110 175 105 245 62 95 123 ...
##  $ drat: num  3.9 3.9 3.85 3.08 3.15 2.76 3.21 3.69 3.92 3.92 ...
##  $ wt  : num  2.62 2.88 2.32 3.21 3.44 ...
##  $ qsec: num  16.5 17 18.6 19.4 17 ...
##  $ vs  : num  0 0 1 1 0 1 0 1 1 1 ...
##  $ am  : num  1 1 1 0 0 0 0 0 0 0 ...
##  $ gear: num  4 4 4 3 3 3 3 4 4 4 ...
##  $ carb: num  4 4 1 1 2 1 4 2 2 4 ...
```

```r
#View(mtcars) # para ver en RStudio
head(mtcars)
```

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

Podemos acceder a los datos de un data frame de la misma manera que en una matriz,

```r
mtcars[1,1]
```

```
## [1] 21
```

```
mtcars[2,]
```

```
##               mpg cyl disp  hp drat    wt  qsec vs am gear carb
## Mazda RX4 Wag  21   6  160 110  3.9 2.875 17.02  0  1    4    4
```

pero tenemos otras opciones. Si nos interesa seleccionar una variable (columna), lo podemos usar usando *$*

```
mtcars$mpg
```

```
##  [1] 21.0 21.0 22.8 21.4 18.7 18.1 14.3 24.4 22.8 19.2 17.8 16.4 17.3 15.2
## [15] 10.4 10.4 14.7 32.4 30.4 33.9 21.5 15.5 15.2 13.3 19.2 27.3 26.0 30.4
## [29] 15.8 19.7 15.0 21.4
```

Por ejemplo, si queremos las observaciones (filas) para las cuales el rendimiento de combustible ('mtcars$mpg') es mayor a 20, lo podemos hacer de la siguiente manera:

```
mtcars_r <- mtcars[mtcars$mpg > 20,]
mtcars_r
```

```
##                 mpg cyl  disp  hp drat    wt  qsec vs am gear carb
## Mazda RX4      21.0   6 160.0 110 3.90 2.620 16.46  0  1    4    4
## Mazda RX4 Wag  21.0   6 160.0 110 3.90 2.875 17.02  0  1    4    4
## Datsun 710     22.8   4 108.0  93 3.85 2.320 18.61  1  1    4    1
## Hornet 4 Drive 21.4   6 258.0 110 3.08 3.215 19.44  1  0    3    1
## Merc 240D      24.4   4 146.7  62 3.69 3.190 20.00  1  0    4    2
## Merc 230       22.8   4 140.8  95 3.92 3.150 22.90  1  0    4    2
## Fiat 128       32.4   4  78.7  66 4.08 2.200 19.47  1  1    4    1
## Honda Civic    30.4   4  75.7  52 4.93 1.615 18.52  1  1    4    2
## Toyota Corolla 33.9   4  71.1  65 4.22 1.835 19.90  1  1    4    1
## Toyota Corona  21.5   4 120.1  97 3.70 2.465 20.01  1  0    3    1
## Fiat X1-9      27.3   4  79.0  66 4.08 1.935 18.90  1  1    4    1
## Porsche 914-2  26.0   4 120.3  91 4.43 2.140 16.70  0  1    5    2
## Lotus Europa   30.4   4  95.1 113 3.77 1.513 16.90  1  1    5    2
## Volvo 142E     21.4   4 121.0 109 4.11 2.780 18.60  1  1    4    2
```

Importar datos

La mayoría de las veces, tenemos datos que son anotados a partir de un experimento que realizamos, o simplemente un colega tiene datos que quiere que analicemos. En dichos casos, tenemos la necesidad de entrar los datos en R. La gran mayoría de las veces, hacerlo a mano sería engorroso. Por suerte, los datos suelen estar guardados en formas estándar (ejemplo: archivos .csv), y R los puede interpretar e importar automáticamente. Consideremos que tenemos que importar el data frame de juguete "dataFrame.csv". Podemos observar varias cosas:

- El archivo de texto tiene tantas lineas como filas (más la fila de nombres de las variables, si existe).
- El fin de linea "\n" es el separador más usual para delimitar filas del data frame.
- Dentro de una fila, las variables de columnas adyacentes son separadas usando un separador (en este caso, punto y coma ";"). Otros separadores comunes son la coma "," (que puede causar problemas si también es el delimitador de decimales en los números), el espacio en blanco " ", o el tabulador"\t".
- Las variables de texto o factores (etiquetas) están entrecomilladas (quotes), mientras que las variables numéricas no lo están.

Una vez que sabemos las características del archivo del cual disponemos, lo guardamos como archivo *txt* o *csv* (ejemplo: 'dataFrame.csv') en la carpeta fijada como directorio de trabajo

```r
getwd() # El archivo debe estar guardado en la carpeta que se muestra.
```

```
## [1] "C:/Users/Nie Lie/Nextcloud/Bioestadistica/Laboratorio/Lab_2020"
```

```r
?setwd
# En caso de querer cambiar la carpeta de directorio de trabajo,
# se puede hacer usando la función 'setwd'
```

y usaremos la función *read.table*.

```r
?read.table
# Aquí las opciones se tienen que elegir en el código.
# Asegurarse que el archivo "dataFrame.csv" esté guardado en el directorio de trabajo.
dataFrame = read.table("dataFrame.csv", header = TRUE, sep = ";", quote = "\"'", dec = ",")
```

En caso de que el archivo no esté guardado en el directorio de trabajo, debemos pasar la ruta del archivo (ya sea partiendo desde el directorio de trabajo, o la ruta absoluta).

Observar que marcamos:

- header = TRUE: las variables tienen nombre.
- sep = ";": las columnas están separadas por punto y coma
- quote = " \' " : las variables que no son numéricas están entrecomilladas
- dec = ",": el delimitador de decimales es la coma

La manera de elegir bien esas opciones es abrir el archivo de texto y observar sus características.

Hay otras funciones similares de importación de datos que deben tenerse en cuenta. Distintas funciones pueden servir para importar distintos tipos de objetos, o contienen distintos valores por defecto de alguno de sus parámetros. A continuación hay dos de esas funciones, pero hay varias más.

```r
?read.delim
?read.csv
```

Exportar datos

Muchas veces, deseamos guardar los objetos que creamos en *R* en un archivo de texto. Por lo general, desearemos compartir eso con algún colega, o usaremos esos datos en otro software. El caso más común es querer guardar un data frame, o una matriz.

Como ejemplo, usaremos la función *write.table*, que nos permitirá crear un archivo del estilo *.csv* a partir de un data frame o matriz.

```r
m = matrix(rnorm(100*2), ncol = 2)
# matriz 100x2 con entradas normales N(0,1)
d_f = as.data.frame(m)
names(d_f) = c("Columna 1", "Columna 2")
write.table(d_f, file = "datos_normales.csv", sep = "\t", col.names = TRUE)
```

El funcionamiento de write.table es similar al de read.table. Una manera de pensarlo es que, los parámetros que fijamos al usar write.table, son los que necesitaremos para recuperar los datos cuando usemos read.table.

También contamos con otras funciones para guardar datos

```r
?read.csv
?read.delim
```

Función apply

La función *apply* es muy útil cuando trabajamos con matrices. Si queremos realizar una función sobre todas las columnas (por ejemplo, tomar la suma), lo podemos hacer de manera elegante usando la función *apply*.

```
v <- c(2, 4, 6, 8)
N <- matrix(v, nrow = 2, byrow = TRUE)

?apply
# Observar la descripción de la variable 'MARGIN' (segunda variable).
#'MARGIN = 2' corresponde a aplicar la función 'FUN' a cada columna.
v.Ncsum <- apply(N, 2, sum)
# 'v.Ncsum' es un vector, donde la entrada 'i'
# representa la suma de todas las entradas de la columna
# 'i' de N
v.Ncsum
```

```
## [1]  8 12
```

```
m.Nsqrt <- apply(N, c(1, 2), sqrt)
# Al poner 'MARGIN = c(1, 2)',
# aplicamos la función 'FUN = sqrt' a cada elemento de 'N'.
# Por lo tanto, 'v.Nsqrt' es una matriz del mismo tamaño que 'N'.
m.Nsqrt
```

```
##          [,1]     [,2]
## [1,] 1.414214 2.000000
## [2,] 2.449490 2.828427
```

```
sqrt(N)
```

```
##          [,1]     [,2]
## [1,] 1.414214 2.000000
## [2,] 2.449490 2.828427
```

```
# Comparar con 'm.Nsqrt'.
```

Si queremos aplicar una función más compleja, la podeos definir dentro de apply de la siguiente manera:

```
resta_media = function(v){
return(v - mean(v))
}
m.Nrc <- apply(N, 1, resta_media)
```

Al poner *MARGIN = 1*, sabemos que la función resta_media será aplicada sobre vectores (filas de N). Dicha función toma cada fila de N, le calcula el promedio *(mean())*, y se lo resta a cada entrada de la fila. Los resultados de estas operaciones siempre se guardan como columnas. Esto es, el resultado de aplicar *resta_media* a la primera fila de N se guarda como la primera columna de *m.Nrc*.

```
v.Nrcm <- apply(m.Nrc, 2, mean)
v.Nrcm
```

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

Observar que los promedios de las columnas de m.Nrc son cero, por ser los resultados de v - mean(v) para las filas de N.

Ejercicio 1 En la base de datos *mtcars*, obtener los nombres de los autos con 6 cilindros y 4 carburadores.

Ejercicio 2 En la base de datos *mtcars*, dentro de los autos con 6 cilindros, hallar el promedio de los caballos de fuerza.

Ejercicio 3 En la base de datos *mtcars*, graficar el número de cilindros (eje x) y los caballos de fuerza (eje y). Luego, graficar el peso de los autos (eje x) y los caballos de fuerza (eje y). Finalmente, graficar la tasa de millas por galón (eje x) y los caballos de fuerza (eje y).

Ejercicio 4 Importar la base de datos *dataFrame.csv*, aplicarle las funciones *View*, *head*, *summary*, y *str*. Además, graficar *Variable1* contra *Variable2*.

Ejercicio 5 Crear una matriz de tamaño 100×5 cuyas entradas sean uniformes $U(0, 1)$. Luego, exportar esa matriz a un archivo *.csv* (los separadores entre celdas deben ser comas).