Apuntes Métodos Avanzados en Estadística

Álvaro José Álvarez Arranz

02/01/2024

Tema 0: Introducción a R y repaso

Características de R

- Distingue entre mayúsculas y minúsculas.
- Los nombres de todo empiezan por una letra (Como en cualquier lenguaje, vaya).
- Operador de asignación. Se suele usar '<-', pero tambíen vale con el signo '='.

```
x = 2
y <- 3
x
```

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

```
y
```

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

- Creamos arrays de elementos. Los elementos no tiene por qué compartir tipos.

```
x = c(1,2,3,4,5)
y = c(3,"Alvaro", 9.5)
x
```

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

```
y
```

```
## [1] "3"      "Alvaro" "9.5"
```

- Podemos listar el nombre de las variables que se están usando.

```
ls()
```

```
## [1] "x" "y"
```

- Podemos borrar las cosas que ya no nos interesan.

```
rm(x)
```

- Obtener ayuda con help() o help.search()

Vectores. Cómo generarlos

- Usando ':'.

```
1:6
```

```
## [1] 1 2 3 4 5 6
```

- Usando la función 'seq()'. Lo que hace es crear una secuencia.

```
seq(2, 10, 2)
```

```
## [1]  2  4  6  8 10
```

- Usando la función 'rep()'. Lo que hace es repetir un elemento x veces.

```
rep(c(1,3), 4)
```

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

Filtrado de vectores

A continuación se muestra cómo seleccionar algunos elementos

```
x = (1:5)^2
x[2]
```

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

Notese que en los arrays de R el valor 0 es el tipo de datos que contiene y los valores empiezan en la posición 1.

Se pueden elegir los elementos que queramos poniendo un vector en las coordenadas del array.

```
x[c(2, 5)]
```

```
## [1]  4 25
```

Si queremos excluir las posiciones, basta con poner el signo - en el array.

```
x[-c(2, 5)]
```

```
## [1]  1  9 16
```

Podemos seleccionar los elementos que cumplan con una condición.

```r
x[x > 15]
```

```
## [1] 16 25
```

Y también obtener las posiciones de los elementos que cumplen con la condición.

```r
which(x > 15)
```

```
## [1] 4 5
```

Factores

Son estructuras que se utilizan para manejar variables cualitativas en análisis estadísticos.

Se pueden contemplar como un vector al que se añade un poco más de información consistente en los distintos valores presentes en el vector, llamados *niveles*.

Se crean usando la función 'factor()'.

```r
respuesta = c("si","no","si","no","no","no","si")
respuesta = factor(respuesta)
respuesta
```

```
## [1] si no si no no no si
## Levels: no si
```

Matrices

Las matrices **son vectores**, pero con dos atributos extra: número de filas y de columnas. Pero los vectores **NO SON** matrices de una fila o columna. Así se crea una matriz:

```r
matrix(1:3, nrow=2, ncol=3)
```

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

Como no hay suficientes valores, la función se encarga de repetir el vector introducido hasta rellenar la matriz. Para añadirle dificulatd, la función primero rellena las matrices de manera vertical.

Las matrices también son vectores

Así es como se crean las matrices.

```r
x = matrix(2:10, ncol=3, nrow=3)
x
```

```
##      [,1] [,2] [,3]
## [1,]    2    5    8
## [2,]    3    6    9
## [3,]    4    7   10
```

Podemos obtener los valores que cumplen con una condición en forma de vector.

```
x[x>6]
```

```
## [1]  7  8  9 10
```

Seleccionar parte de la matriz

Se pueden seleccionar de distintas formas: * Coordenada fila-columna:

```
x[1,2]
```

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

- Una columna completa

```
x[,3]
```

```
## [1]  8  9 10
```

- Un conjunto de filas o columnas

```
x[,c(1,3)];
```

```
##      [,1] [,2]
## [1,]    2    8
## [2,]    3    9
## [3,]    4   10
```

```
x[c(2,3),]
```

```
##      [,1] [,2] [,3]
## [1,]    3    6    9
## [2,]    4    7   10
```

Operaciones útiles con matrices

Tenemos dos matrices

```
A=matrix(1:4,2,2);
B=matrix(5:8,2,2)
```

- Producto matricial

```
A %*% B
```

```
##      [,1] [,2]
## [1,]   23   31
## [2,]   34   46
```

- Determinante

```
det(A)
```

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

- Producto componente a componente

```
A * B
```

```
##      [,1] [,2]
## [1,]    5   21
## [2,]   12   32
```

- Traspuesta

```
t(A)
```

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

- Extrear la diagonal de una matriz

```
diag(A)
```

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

- Inversa

```
solve(A)
```

```
##      [,1] [,2]
## [1,]   -2  1.5
## [2,]    1 -0.5
```

- Resolver un sistema de ecuaciones lineales $Ax = b$:

```
b = c(1,3);
solve(A,b)
```

```
## [1]  2.5 -0.5
```

- Autovalores y autovectores:

```
eigen(A)
```

```
## eigen() decomposition
## $values
## [1]  5.3722813 -0.3722813
##
## $vectors
##            [,1]       [,2]
## [1,] -0.5657675 -0.9093767
## [2,] -0.8245648  0.4159736
```

Listas

Una lista es un vector de objetos de tipos distintos, que conviene agrupar en la misma estructura.

Es importante su comprensión porque unchas funciones devuelven resultados de esta forma.

Para crearlas se usa el comando *list*:

```
milista = list(nombre='Pepe', no.hijos=3, edades.hijos=c(4,7,9));
str(milista)
```

```
## List of 3
##  $ nombre      : chr "Pepe"
##  $ no.hijos    : num 3
##  $ edades.hijos: num [1:3] 4 7 9
```

Extraer información de una lista

Hay varias formas de extraer la información de una lista.

```
milista$nombre;
```

```
## [1] "Pepe"
```

```
milista[[1]];
```

```
## [1] "Pepe"
```

```
milista[['nombre']];
```

```
## [1] "Pepe"
```

```
milista$edades.hijos[2]
```

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

Data frames

Es la estructura más importante en R.

Intuitivamente son matrices con entradas de distintos tipos. Tecnicamente es una lista formada por vectores de la misma longitud.

Para crearlos se utiliza el comando data.frame:

```
x = 7:9;
y = c('a','b','c');
mifichero = data.frame(edad=x, grupo=y);
mifichero
```

```
##   edad grupo
## 1    7     a
## 2    8     b
## 3    9     c
```

Admiten comandos para matrices y para listas:

```
mifichero$edad
```

```
## [1] 7 8 9
```

```
mifichero[1,]
```

```
##   edad grupo
## 1    7     a
```

Importar datos a R desde RStudio

En la pestaña 'Environment' hay un boton para añadir datos a R provenientes de un fichero.

```
notas <- read.csv2("~/Master Ciencia de Datos/MAE/notas.txt", sep="") # Esto solo funciona en mi equipo
```

También puede usarse mediante comando.

Aplicar funciones por columnas

Se utilizan las funciones de la familia *apply*

```
apply(X = notas[c(2,3)], MARGIN = 2, FUN = mean) # MARGIN = 2 indica que hay que hacerlo por columnas
```

```
##   nota09   nota10
## 6.604877 5.439943
```

```
sapply(X = notas[c(2,3)], FUN = mean) # Este ya lo hace por columnas
```

```
##   nota09   nota10
## 6.604877 5.439943
```

```
lapply(X = notas[c(2,3)], FUN = mean) # Lo hace por columnas y también lo separa
```

```
## $nota09
## [1] 6.604877
##
## $nota10
## [1] 5.439943
```

Aplicar funciones según factor

Usamos otra función de la familia *apply*

```r
notas$tipo = factor(notas$tipo) # El índice debe ser tipo para aplicar la siguiente linea
tapply(X = notas$nota09, INDEX = notas$tipo, FUN = mean)
```

```
## concertado    privado    publico
##   6.888444   7.422755   6.343525
```

- Se puede ver la media de las notas según el tipo de colegio en el año 2009
- Puede verse lo mismo cambiando a notas$nota10

```r
anova = aov(notas$nota09 ~ notas$nota10)
summary(anova)
```

```
##                Df Sum Sq Mean Sq F value Pr(>F)
## notas$nota10    1  361.6   361.6   469.8 <2e-16 ***
## Residuals    1220  939.0     0.8
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Esta tabla muestra una serie de datos que se verán más adelante en el curso

Manejo de dplyr

Inttroducción

Es un paquete de tidyverse que incluye las acciones más comunes que se realizan sobre un conjunto de datos:

- Contar: *count*
- Seleccionar filas: *filter*
- Seleccionar columnas: *select*
- Ordenar: *arrange*
- Sintaxis en cadena o *pipes*
- Añadir nuevas variables: *mutate*
- Resumir: *summarise*

Las características generales son:
* El primer argumento siempre es un *data.frame* * El resto de argumentos indican lo que queremos hacer con el *data.frame* * El resultado siempre tiene la estructura de un *data.frame*

Seleccionamos 15 observaciones (5 de cada especie) del fichero *iris* a la manera *tradicional*

```r
lirios = iris[c(1:5, 51:55, 101:105),]
head(lirios)
```

```
##    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
## 51          7.0         3.2          4.7         1.4 versicolor
```

Usaremos *lirios* como fichero de datos en los ejemplos que siguen. Para ello tenemos que cargar *dlpyr*:

```
library(dplyr)
```

```
## Warning: package 'dplyr' was built under R version 4.3.2
```

```
##
## Attaching package: 'dplyr'
```

```
## The following objects are masked from 'package:stats':
##
##     filter, lag
```

```
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

Para saber cuántos lirios hay de cada especie hay que usar la función *count*:

```
count(lirios, Species)
```

```
##      Species n
## 1     setosa 5
## 2 versicolor 5
## 3  virginica 5
```

Uso de *filter*

Para seleccionar los lirios de la especie *setosa* usamos la función *filter*:

```
filter(lirios, Species == 'setosa')
```

```
##   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
```

También se pueden seleccionar todos los datos pertenecientes a dos especies:

```
filter(lirios, Species == 'setosa' | Species == 'virginica')
```

```
##    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           6.3         3.3          6.0         2.5 virginica
## 7           5.8         2.7          5.1         1.9 virginica
## 8           7.1         3.0          5.9         2.1 virginica
## 9           6.3         2.9          5.6         1.8 virginica
## 10          6.5         3.0          5.8         2.2 virginica
```

Uso de *select*

Seleccionamos las variables relaccionadas con el sépalo:

```
select(lirios, Sepal.Length, Sepal.Width)
```

```
##     Sepal.Length Sepal.Width
## 1            5.1         3.5
## 2            4.9         3.0
## 3            4.7         3.2
## 4            4.6         3.1
## 5            5.0         3.6
## 51           7.0         3.2
## 52           6.4         3.2
## 53           6.9         3.1
## 54           5.5         2.3
## 55           6.5         2.8
## 101          6.3         3.3
## 102          5.8         2.7
## 103          7.1         3.0
## 104          6.3         2.9
## 105          6.5         3.0
```

Selección de un rango de variables:

```
select(lirios, Petal.Length:Sepal.Length)
```

```
##     Petal.Length Sepal.Width Sepal.Length
## 1            1.4         3.5          5.1
## 2            1.4         3.0          4.9
## 3            1.3         3.2          4.7
## 4            1.5         3.1          4.6
## 5            1.4         3.6          5.0
## 51           4.7         3.2          7.0
## 52           4.5         3.2          6.4
## 53           4.9         3.1          6.9
## 54           4.0         2.3          5.5
## 55           4.6         2.8          6.5
## 101          6.0         3.3          6.3
## 102          5.1         2.7          5.8
## 103          5.9         3.0          7.1
## 104          5.6         2.9          6.3
## 105          5.8         3.0          6.5
```

Seleccionar todas las variables menos *Species*:

```
select(lirios, -Species)
```

```
##     Sepal.Length Sepal.Width Petal.Length Petal.Width
## 1            5.1         3.5          1.4         0.2
## 2            4.9         3.0          1.4         0.2
```

```
## 3           4.7         3.2          1.3         0.2
## 4           4.6         3.1          1.5         0.2
## 5           5.0         3.6          1.4         0.2
## 51          7.0         3.2          4.7         1.4
## 52          6.4         3.2          4.5         1.5
## 53          6.9         3.1          4.9         1.5
## 54          5.5         2.3          4.0         1.3
## 55          6.5         2.8          4.6         1.5
## 101         6.3         3.3          6.0         2.5
## 102         5.8         2.7          5.1         1.9
## 103         7.1         3.0          5.9         2.1
## 104         6.3         2.9          5.6         1.8
## 105         6.5         3.0          5.8         2.2
```

Seleccionamos las variables cuyo nombre contenga *Petal*:

```
select(lirios, contains('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
## 51           4.7         1.4
## 52           4.5         1.5
## 53           4.9         1.5
## 54           4.0         1.3
## 55           4.6         1.5
## 101          6.0         2.5
## 102          5.1         1.9
## 103          5.9         2.1
## 104          5.6         1.8
## 105          5.8         2.2
```

Uso de *arrange*

Por defecto, esta función ordena de **menor a mayor** el valor de una variable.

```
arrange(lirios, Sepal.Length)
```

```
##    Sepal.Length Sepal.Width Petal.Length Petal.Width    Species
## 1           4.6         3.1          1.5         0.2     setosa
## 2           4.7         3.2          1.3         0.2     setosa
## 3           4.9         3.0          1.4         0.2     setosa
## 4           5.0         3.6          1.4         0.2     setosa
## 5           5.1         3.5          1.4         0.2     setosa
## 6           5.5         2.3          4.0         1.3 versicolor
## 7           5.8         2.7          5.1         1.9  virginica
## 8           6.3         3.3          6.0         2.5  virginica
## 9           6.3         2.9          5.6         1.8  virginica
## 10          6.4         3.2          4.5         1.5 versicolor
```

```
## 11          6.5         2.8          4.6         1.5 versicolor
## 12          6.5         3.0          5.8         2.2  virginica
## 13          6.9         3.1          4.9         1.5 versicolor
## 14          7.0         3.2          4.7         1.4 versicolor
## 15          7.1         3.0          5.9         2.1  virginica
```

Para ordenar *de mayor a menor* se hace de lña siguiente manera:

```
arrange(lirios, -Sepal.Length)
```

```
##    Sepal.Length Sepal.Width Petal.Length Petal.Width    Species
## 1           7.1         3.0          5.9         2.1  virginica
## 2           7.0         3.2          4.7         1.4 versicolor
## 3           6.9         3.1          4.9         1.5 versicolor
## 4           6.5         2.8          4.6         1.5 versicolor
## 5           6.5         3.0          5.8         2.2  virginica
## 6           6.4         3.2          4.5         1.5 versicolor
## 7           6.3         3.3          6.0         2.5  virginica
## 8           6.3         2.9          5.6         1.8  virginica
## 9           5.8         2.7          5.1         1.9  virginica
## 10          5.5         2.3          4.0         1.3 versicolor
## 11          5.1         3.5          1.4         0.2     setosa
## 12          5.0         3.6          1.4         0.2     setosa
## 13          4.9         3.0          1.4         0.2     setosa
## 14          4.7         3.2          1.3         0.2     setosa
## 15          4.6         3.1          1.5         0.2     setosa
```

Se puede ordenar las especies por orden alfabético y dentro de cada especie por otra variable de menor a mayor:

```
arrange(lirios, Species, Sepal.Length)
```

```
##    Sepal.Length Sepal.Width Petal.Length Petal.Width    Species
## 1           4.6         3.1          1.5         0.2     setosa
## 2           4.7         3.2          1.3         0.2     setosa
## 3           4.9         3.0          1.4         0.2     setosa
## 4           5.0         3.6          1.4         0.2     setosa
## 5           5.1         3.5          1.4         0.2     setosa
## 6           5.5         2.3          4.0         1.3 versicolor
## 7           6.4         3.2          4.5         1.5 versicolor
## 8           6.5         2.8          4.6         1.5 versicolor
## 9           6.9         3.1          4.9         1.5 versicolor
## 10          7.0         3.2          4.7         1.4 versicolor
## 11          5.8         2.7          5.1         1.9  virginica
## 12          6.3         3.3          6.0         2.5  virginica
## 13          6.3         2.9          5.6         1.8  virginica
## 14          6.5         3.0          5.8         2.2  virginica
## 15          7.1         3.0          5.9         2.1  virginica
```

Sintaxis en cadena o *pipes*

Se usa el operador '%>%'. El elemento que lo precede es el primer argumento para el comando que le sigue:

```
lirios %>%
  select(contains('Petal')) %>%
  filter(Petal.Length > 4) %>%
  arrange(Petal.Length)
```

```
##   Petal.Length Petal.Width
## 1          4.5         1.5
## 2          4.6         1.5
## 3          4.7         1.4
## 4          4.9         1.5
## 5          5.1         1.9
## 6          5.6         1.8
## 7          5.8         2.2
## 8          5.9         2.1
## 9          6.0         2.5
```

Añadir nuevas variables: *mutate*

Añadimos una nueva variable que corresponda al cociente entre anchura y longitud del pétalo (lo que podría cuantificar la forma del pétalo):

```
lirios_nuevo = lirios %>%
  select(contains('Petal')) %>%
  mutate(forma = round(Petal.Width/Petal.Length, 2))
```

mutate tiene variantes:

- *mutate*: Añade una nueva variable al data frame
- *transmute*: Añade una nueva variable y elimina las antiguas
- *mutate_all*: Cambia todas las variables de un data frame simultaneamente
- *mutate_at*: Cambia las variables especificadas por nombre
- *mutate_if*: Cambia las variables que cumplen una condición

Resumir: summarise

Se usa para hacer resúmenes. Se suele usar en combinación con *group_by*.

```
lirios %>%
  group_by(Species) %>%
  summarise(media_PL = mean(Petal.Length),
            varianza_PL = var(Petal.Length))
```

```
## # A tibble: 3 x 3
##   Species    media_PL varianza_PL
##   <fct>         <dbl>       <dbl>
## 1 setosa         1.4       0.0050
## 2 versicolor     4.54      0.113
## 3 virginica      5.68      0.127
```

Operaciones por columnas: across()

Esta función calcula las medias y las desviaciones típicas de cada medida de pétalo en cada una de las tres especias

```
lirios %>%
  group_by(Species) %>%
  summarise(across(contains('Petal'),
                   list(Media = mean, DT = sd)))
```

```
## # A tibble: 3 x 5
##   Species    Petal.Length_Media Petal.Length_DT Petal.Width_Media Petal.Width_DT
##   <fct>                   <dbl>           <dbl>             <dbl>          <dbl>
## 1 setosa                   1.4           0.0707               0.2          0
## 2 versicolor               4.54          0.336                1.44         0.0894
## 3 virginica                5.68          0.356                2.1          0.274
```

Visualización de datos

Para la visualización de datos se usará *ggplot2*, que es otro paquete del tidyverse.

El esquema que se sigue para la creación de cualquier gráfico con ggplot2 es:

```
ggplot(data=<DATA>) +
  <GEOM_FUNCTION>(mapping = aes(<MAPPINGS>))
```

Donde:
* `ggplot` crea un sistema de coordenadas vacio. El argumento 'data' fija el data frame donde están los datos.
* `<GEOM_FUNCTION>` añade una capa al gráfico con el tipo de elementos que se van a representar. * `mapping = aes(<MAPPINGS>)` asigna variables del fichero a las propiedades visuales del gráfico.

Las `<GEOM_FUNCTION>` pueden ser las siguientes:

- `geom_point()`: Gráfico de dispersión.
- `geom_smooth()`: Junto a `geom_point` para recta de mínimos cuadrados.
- `geom_line()`: Gráfico de líneas.
- `geom_bar()`: Gráfico de barras.
- `geom_histogram()`: Histograma.
- `geom_boxplot()`: Gráfico de caja y bigote.

Estos son algunos de los muchos diagramas disponibles.