



# Modelización Estadística

- [Apartado 01](#)
- [Apartado 02](#)
- [Apartado 03](#)
- [Apartado 04](#)
- [Apartado 05](#)
- [Apartado 06](#)
- [Gráficos](#)
- [Apartado 07](#)
- [Apartado 08](#)
- [Apartado 09](#)
- [Apartado 10](#)
- [Apartado 11](#)

E.T.S de Ingeniería Informática - Universidad de Málaga.

Trabajo elaborado por: Antonio J. Galán Herrera

### Aclaraciones previas:

- Se ha evitado el uso de acentos en el script para evitar caracteres raros al guardarlo en formato UTF-8.
- Las rutas a ficheros están escritas de forma relativa a la carpeta de la entrega, por lo que el script fallará al leer los datos si se ejecuta en otra ubicación.
- El apartado 11 solo se encuentra en este documento, no en el script.
- Se han eliminado algunos de los comentarios del script en las secciones de código del documento para que no haya explicaciones duplicadas.

Proyecto y rama del trabajo.

## Apartado 01

### Código de R

```
# Usando la funcion 'read_csv' del paquete 'readr'
# se exporta el fichero con formato 'tibble'.
datos <- read_csv("../Recursos/12306.csv", col_types=cols(sexo=col_factor(),
                                                         dietaEsp=col_factor(),
                                                         nivEstPad=col_factor(),
                                                         nivEstudios=col_factor(),
                                                         nivIngresos=col_factor()))
```

### Resultado

```
> datos
# A tibble: 5,000 x 14
  peso altura sexo  edad tabaco  ubes carneRoja verduras deporte drogas dietaEsp
<dbl>  <dbl> <fct>  <dbl>  <dbl> <dbl>    <dbl>    <dbl>    <dbl>  <dbl>  <fct>
1  81.6   1.71 V      18      0      0         3        11        4      0 N
2 101.   1.61 M      56      0      0         3         5        5      0 N
3 103.   1.83 V      42      0      0         4         7        0      0 N
```

```
4 113.    1.75 V      39      0      0      1      16      7      0 N
5  55.8    1.76 V      43     110     0      2       0      0      1 N
6  68.0    1.67 M      21      0     11      0     13      0      1 N
7  66.6    1.58 M      74     130      6      0      0     11      0 N
8 101.     1.7  V      50      0      3      0     29     18      0 N
9  81.4    1.69 M      35      0      4      0     14      3      0 N
10 76.1    1.67 M      18      0      0      6      0     10      2 S
# ... with 4,990 more rows, and 3 more variables: nivEstPad <fct>, nivEstudios <fct>,
#   nivIngresos <fct>
```

Variables:

- `datos` con los valores del fichero ordenados en 14 columnas, 9 de ellas variables cuantitativas y 5 de ellas variables cualitativas.

Explicación

Los datos del fichero `12306.csv` se exportan en un formato de valores separados por comas.

Con R puede leerse dichos valores haciendo uso de la función `read_csv()` de la librería `tidyverse` ; estos se almacenarán en un dataframe en formato `tibble` , que facilitará el manejo de los valores durante el resto de apartados del trabajo.

Como se puede ver, la lectura incorpora la factorización de 5 variables, debido a que son cualitativas.

Apartado 02

Código de R

```
# Se crea la columna 'IMC' segun el enunciado
datos <- mutate(datos, IMC = peso / altura^2)
```

Resultado

```
> datos
# A tibble: 5,000 x 15
  peso altura sexo  edad tabaco  ubes carneRoja verduras deporte drogas dietaEsp
  <dbl>  <dbl> <fct>  <dbl>  <dbl> <dbl>    <dbl>    <dbl>    <dbl>  <dbl> <fct>
1  81.6   1.71 V      18      0      0      3      11      4      0 N
2 101.    1.61 M      56      0      0      3       5      5      0 N
3 103.    1.83 V      42      0      0      4       7      0      0 N
4 113.    1.75 V      39      0      0      1     16      7      0 N
5  55.8    1.76 V      43     110     0      2       0      0      1 N
6  68.0    1.67 M      21      0     11      0     13      0      1 N
7  66.6    1.58 M      74     130      6      0      0     11      0 N
8 101.     1.7  V      50      0      3      0     29     18      0 N
9  81.4    1.69 M      35      0      4      0     14      3      0 N
10 76.1    1.67 M      18      0      0      6      0     10      2 S
# ... with 4,990 more rows, and 4 more variables: nivEstPad <fct>, nivEstudios <fct>,
#   nivIngresos <fct>, IMC <dbl>
```

Variables:

- `datos` con una nueva columna llamada *IMC*, cuyos valores se obtienen operando los de las columnas *peso* y *altura*.

Explicación

En R, es habitual usar la función `mutate()` para modificar un dataframe añadiéndole columnas, lo que permite reutilizar la misma variable actualizando su contenido.

En este caso, como los valores están en `datos` , basta con añadir una nueva columna a esa variable, en lugar de generar otra variable nueva que contenga la misma información junto a la añadida.

Apartado 03

Código de R

```
# Se limpian las filas con algun NA de los datos,
# usando la funcion 'na.exclude()' o 'na.omit()'
datos <- na.exclude(datos)
```

### Resultado

```
> datos
# A tibble: 4,950 x 15
  peso altura sexo  edad tabaco  ubes carneRoja verduras deporte drogas dietaEsp
  <dbl>  <dbl> <fct> <dbl>  <dbl> <dbl>    <dbl>    <dbl>    <dbl>  <dbl> <fct>
1  81.6   1.71 V      18     0     0      3      11      4      0 N
2 101.   1.61 M      56     0     0      3       5      5      0 N
3 103.   1.83 V      42     0     0      4       7      0      0 N
4 113.   1.75 V      39     0     0      1     16      7      0 N
5  55.8   1.76 V      43    110     0      2       0      0      1 N
6  68.0   1.67 M      21     0    11     0     13      0      1 N
7  66.6   1.58 M      74    130     6     0       0     11      0 N
8 101.   1.7  V      50     0     3     0     29     18      0 N
9  81.4   1.69 M      35     0     4     0     14      3      0 N
10 76.1   1.67 M      18     0     0     6       0     10      2 S
# ... with 4,940 more rows, and 4 more variables: nivEstPad <fct>, nivEstudios <fct>,
#   nivIngresos <fct>, IMC <dbl>
```

Variables:

- `datos` con 10 filas menos, donde en cada una de esas filas había al menos un valor NA para alguna de las columnas.

### Explicación

Durante el resto de apartados será necesario que no haya valores NA, ya que interfieren con los procesos de cálculo.

Usando la funcion `na.exclude()` se eliminan de un dataframe todas las filas completas que contengan un NA en alguna de sus columnas, obteniendo un `tibble` donde todos los valores son computables.

## Apartado 04

### Código de R

```
# Calcula la desviacion tipica.
# * datos:  vector con los valores para el calculo
dt <- function(datos) {
  sqrt(mean(datos^2) - mean(datos)^2)
}

# Quitar variables cualitativas (factores) de los datos
datosN <- Filter(is.numeric, datos)

# Medias
medias <- colMeans(datosN)

# Desviaciones tipicas
desviaciones <- map_dbl(datosN, dt)
```

### Resultado

```
> datosN
# A tibble: 4,950 x 10
  peso altura  edad tabaco  ubes carneRoja verduras deporte drogas  IMC
  <dbl>  <dbl> <dbl>  <dbl> <dbl>    <dbl>    <dbl>    <dbl>  <dbl> <dbl>
1  81.6   1.71   18     0     0      3      11      4      0 27.9
2 101.   1.61   56     0     0      3       5      5      0 39.1
3 103.   1.83   42     0     0      4       7      0      0 30.9
4 113.   1.75   39     0     0      1     16      7      0 37.0
5  55.8   1.76   43    110     0      2       0      0      1 18.0
6  68.0   1.67   21     0    11     0     13      0      1 24.4
7  66.6   1.58   74    130     6     0       0     11      0 26.7
8 101.   1.7    50     0     3     0     29     18      0 35
```

```
 9  81.4  1.69  35    0    4    0    14    3    0  28.5
10  76.1  1.67  18    0    0    6    0   10    2  27.3
# ... with 4,940 more rows

> medias
      peso      altura      edad      tabaco      ubes carneRoja  verduras
86.9590485  1.7007192 40.5456566 19.4121212  3.9446465  1.7507071  6.0218182
      deporte      drogas      IMC
4.0812121  0.4967677 30.0140191

> desviaciones
      peso      altura      edad      tabaco      ubes carneRoja  verduras
18.57074700  0.06994665 13.97470429 41.67645769  5.79335797  2.11002286  7.08233888
      deporte      drogas      IMC
4.63049516  1.47034307  5.89044675
```

Funciones:

- `dt()` que calcula la desviación típica de un conjunto de datos.

Variables:

- `datosN` que contiene todas las columnas asociadas a variables cuantitativas.
- `medias` que contiene la media de todos los valores de cada variable cuantitativa.
- `desviaciones` que contiene las desviaciones típicas de todos los valores de cada variable cuantitativa.

Explicación

Pese a que R tiene una función llamada `sd()` que dice calcular la desviación típica, en realidad lo que calcula es la cuasivarianza; por eso, para este apartado se ha generado una función `dt()` que calcula la desviación típica real.

Por otra parte, aunque se hayan limpiado los datos de valores NA, aún quedan las columnas de tipo factor, que también deben ser suprimidas; usando la función `Filter()` se extraen las columnas numéricas y una vez obtenidos todos los datos numéricos en `datosN`, se puede calcular tanto las medias como las desviaciones típicas, usando `colMeans()` en la primera y un `map_dbl(sd, ...)` en la segunda.

La primera también podría haberse hecho con un `map_dbl(mean, ...)`, pero de la otra forma es más directo.

Apartado 05

Código de R

```
# Realiza el ajuste lineal sobre un conjunto de datos.
# * datos: conjunto de datos
# * y: variable a explicar
# * x: variable explicatoria
ajusteLineal <- function(datos, y, x) {

  # lm(str_c(y, "~", x), df) # Deprecado: Apartado 09

  lm(str_c(y, "~", str_c(x, collapse="+")), datos)
}

# Variables para los modelos
variables <- names(datos[3:14]) # Todas excepto "altura", "peso" e "IMC"

# Modelos
modelos <- variables %>% map(ajusteLineal, datos=datos, y="IMC")

# Coeficientes de regresion y determinacion de los modelos
regresion <- modelos %>% map(coefficients)
determinacion <- modelos %>% map(summary) %>% map("r.squared")
```

Resultado

```
> variables
[1] "sexo"      "edad"      "tabaco"    "ubes"      "carneRoja"
[6] "verduras" "deporte"   "drogas"    "dietaEsp"  "nivEstPad"
[11] "nivEstudios" "nivIngresos"

> modelos
```

```
[[1]]

Call:
lm(formula = str_c(y, "~", str_c(x, collapse = "+")), data = datos)

Coefficients:
(Intercept)      sexoM
   30.03841    -0.04953

[[2]]

Call:
lm(formula = str_c(y, "~", str_c(x, collapse = "+")), data = datos)

Coefficients:
(Intercept)      edad
   21.2284    0.2167

[[3]]

Call:
lm(formula = str_c(y, "~", str_c(x, collapse = "+")), data = datos)

Coefficients:
(Intercept)      tabaco
   31.85370   -0.09477

[[4]]

Call:
lm(formula = str_c(y, "~", str_c(x, collapse = "+")), data = datos)

Coefficients:
(Intercept)      ubes
   30.6699   -0.1663

[[5]]

Call:
lm(formula = str_c(y, "~", str_c(x, collapse = "+")), data = datos)

Coefficients:
(Intercept)  carneRoja
   29.93258    0.04652

[[6]]

Call:
lm(formula = str_c(y, "~", str_c(x, collapse = "+")), data = datos)

Coefficients:
(Intercept)  verduras
   29.59500    0.06958

[[7]]

Call:
lm(formula = str_c(y, "~", str_c(x, collapse = "+")), data = datos)

Coefficients:
(Intercept)  deporte
   29.2560    0.1857

[[8]]

Call:
lm(formula = str_c(y, "~", str_c(x, collapse = "+")), data = datos)

Coefficients:
(Intercept)  drogas
   30.02405   -0.02019

[[9]]

Call:
lm(formula = str_c(y, "~", str_c(x, collapse = "+")), data = datos)

Coefficients:
(Intercept)  dietaEspS
   30.0075    0.1405
```

```
[[10]]

Call:
lm(formula = str_c(y, "~", str_c(x, collapse = "+")), data = datos)

Coefficients:
(Intercept)  nivEstPad2  nivEstPad1  nivEstPad3  nivEstPad4
      32.117      -3.476      -1.810      -4.471      -4.863

[[11]]

Call:
lm(formula = str_c(y, "~", str_c(x, collapse = "+")), data = datos)

Coefficients:
(Intercept)  nivEstudios0  nivEstudios4  nivEstudios1  nivEstudios2
      28.409       5.557      -1.696       4.072       1.915

[[12]]

Call:
lm(formula = str_c(y, "~", str_c(x, collapse = "+")), data = datos)

Coefficients:
(Intercept)  nivIngresos0  nivIngresos4  nivIngresos1  nivIngresos2
      28.929       4.303      -1.566       2.790       1.103


> regresion
[[1]]
(Intercept)      sexoM
30.03840547 -0.04953317

[[2]]
(Intercept)      edad
21.2283553  0.2166857

[[3]]
(Intercept)      tabaco
31.85369517 -0.09476945

[[4]]
(Intercept)      ubes
30.6698557 -0.1662599

[[5]]
(Intercept)  carneRoja
29.93257548  0.04652044

[[6]]
(Intercept)  verduras
29.595003  0.069583

[[7]]
(Intercept)  deporte
29.2559836  0.1857379

[[8]]
(Intercept)  drogas
30.02404815 -0.02018853

[[9]]
(Intercept)  dietaEspS
30.0074624  0.1405014

[[10]]
(Intercept)  nivEstPad2  nivEstPad1  nivEstPad3  nivEstPad4
      32.116791  -3.476127  -1.809982  -4.470905  -4.862785

[[11]]
(Intercept)  nivEstudios0  nivEstudios4  nivEstudios1  nivEstudios2
      28.408778    5.556721   -1.696364    4.071621    1.915136

[[12]]
(Intercept)  nivIngresos0  nivIngresos4  nivIngresos1  nivIngresos2
      28.929490    4.303432   -1.566490    2.789610    1.102659


> determinacion
[[1]]
[1] 1.767394e-05

[[2]]
[1] 0.2642709

[[3]]
[1] 0.4495954
```

```
[[4]]
[1] 0.02673864

[[5]]
[1] 0.0002776933

[[6]]
[1] 0.006999438

[[7]]
[1] 0.02131863

[[8]]
[1] 2.539513e-05

[[9]]
[1] 2.531141e-05

[[10]]
[1] 0.06195507

[[11]]
[1] 0.1527215

[[12]]
[1] 0.1123886
```

Funciones:

- `ajusteLineal()` que realiza un ajuste lineal sobre los datos.
- `variables` que contiene el nombre de todas las variables excepto *peso*, *altura* e *IMC*.
- `modelos` que es una lista de los modelos de regresión simple unidimensionales para todas las variables de `variables`.
- `regresion` que contiene los coeficientes de los modelos de `modelos`, donde se puede observar el coeficiente de regresión.
- `determinacion` que contiene los coeficientes de determinación de los modelos de `modelos`, correspondiente a  $R^2$ .

Explicación

Tras definir la función `ajusteLineal()` para que el proceso sea un poco más modular, se obtienen los nombres de las variables que la función va a recibir como argumento para generar los diferentes modelos.

Se excluye *IMC* por ser la variable explicada, así como *peso* y *altura* por generar el *IMC*.

Una vez obtenidos los modelos aplicando la función con `map()`, pueden obtenerse los coeficientes de regresión y de determinación en el contenido de la propia variable `modelos`.

Esto es posible debido a que `ajusteLineal()` usa `lm()` en su interior, que genera dichos valores de forma automática, por lo que solo hay que llamarlos.

Apartado 06

Código de R

```
# Igual que 'ajusteLineal()', pero devuelve una lista con los valores
# 'x' e 'y' del modelo, ademas del propio modelo y los datos.
# * datos: conjunto de datos
# * y: variable a explicar
# * x: variable explicatoria
ajusteLinealLista <- function(datos, y, x) {
  list(x=x, y=y, modelo=lm(str_c(y, "~", str_c(x, collapse="+")), datos))
}

# Genera los graficos de dispersion junto a la recta de regresion
# * datos: conjunto de datos del que extraer el grafico
# * modelo: modelo del que extraer la recta de regresion
dibujarModelos <- function(datos, modelo) {

  # Crear el fichero de la imagen (la carpeta debe existir)
  jpeg(str_c("Graficos/", modelo$x, ".jpeg"))

  # Valores numericos
  x <- datos[[modelo$x]]
  y <- datos[[modelo$y]]
```



```
# Nombres
ejeX <- modelo$x
ejeY <- modelo$y

# Representar los graficos
plot(x, y, xlab=ejeX, ylab=ejeY, col=4)

# Añadir la recta de regresion solo si es una variable cuantitativa
if (is.numeric(datos[[modelo$x]])) {
  abline(modelo$modelo, col=1)
}

# Terminar de escribir en el fichero
dev.off()
}

# Generar los modelos con los valores necesarios para las graficas
modelos <- variables %>% map(ajusteLinealLista, datos=datos, y="IMC")

# Dibujar todos los modelos
modelos %>% walk(~dibujarModelos(datos, .))
```

**Resultado**

```
> modelos
[[1]]
[[1]]$x
[1] "sexo"

[[1]]$y
[1] "IMC"

[[1]]$modelo

Call:
lm(formula = str_c(y, "~", str_c(x, collapse = "+")), data = datos)

Coefficients:
(Intercept)      sexoM
   30.03841    -0.04953

[[2]]
[[2]]$x
[1] "edad"

[[2]]$y
[1] "IMC"

[[2]]$modelo

Call:
lm(formula = str_c(y, "~", str_c(x, collapse = "+")), data = datos)

Coefficients:
(Intercept)      edad
   21.2284    0.2167

[[3]]
[[3]]$x
[1] "tabaco"

[[3]]$y
[1] "IMC"

[[3]]$modelo

Call:
lm(formula = str_c(y, "~", str_c(x, collapse = "+")), data = datos)

Coefficients:
(Intercept)      tabaco
   31.85370   -0.09477

[[4]]
[[4]]$x
[1] "ubes"
```



```
[[4]]$y
[1] "IMC"

[[4]]$modelo

Call:
lm(formula = str_c(y, "~", str_c(x, collapse = "+")), data = datos)

Coefficients:
(Intercept)      ubes
    30.6699    -0.1663

[[5]]
[[5]]$x
[1] "carneRoja"

[[5]]$y
[1] "IMC"

[[5]]$modelo

Call:
lm(formula = str_c(y, "~", str_c(x, collapse = "+")), data = datos)

Coefficients:
(Intercept) carneRoja
    29.93258     0.04652

[[6]]
[[6]]$x
[1] "verduras"

[[6]]$y
[1] "IMC"

[[6]]$modelo

Call:
lm(formula = str_c(y, "~", str_c(x, collapse = "+")), data = datos)

Coefficients:
(Intercept)  verduras
    29.59500     0.06958

[[7]]
[[7]]$x
[1] "deporte"

[[7]]$y
[1] "IMC"

[[7]]$modelo

Call:
lm(formula = str_c(y, "~", str_c(x, collapse = "+")), data = datos)

Coefficients:
(Intercept)  deporte
    29.2560     0.1857

[[8]]
[[8]]$x
[1] "drogas"

[[8]]$y
[1] "IMC"

[[8]]$modelo

Call:
lm(formula = str_c(y, "~", str_c(x, collapse = "+")), data = datos)

Coefficients:
(Intercept)  drogas
    30.02405    -0.02019

[[9]]
[[9]]$x
[1] "dietaEsp"
```

```
[[9]]$y
[1] "IMC"

[[9]]$modelo

Call:
lm(formula = str_c(y, "~", str_c(x, collapse = "+")), data = datos)

Coefficients:
(Intercept)      dietaEspS
      30.0075         0.1405

[[10]]
[[10]]$x
[1] "nivEstPad"

[[10]]$y
[1] "IMC"

[[10]]$modelo

Call:
lm(formula = str_c(y, "~", str_c(x, collapse = "+")), data = datos)

Coefficients:
(Intercept)  nivEstPad2  nivEstPad1  nivEstPad3  nivEstPad4
      32.117      -3.476      -1.810      -4.471      -4.863

[[11]]
[[11]]$x
[1] "nivEstudios"

[[11]]$y
[1] "IMC"

[[11]]$modelo

Call:
lm(formula = str_c(y, "~", str_c(x, collapse = "+")), data = datos)

Coefficients:
(Intercept)  nivEstudios0  nivEstudios4  nivEstudios1  nivEstudios2
      28.409         5.557       -1.696         4.072         1.915

[[12]]
[[12]]$x
[1] "nivIngresos"

[[12]]$y
[1] "IMC"

[[12]]$modelo

Call:
lm(formula = str_c(y, "~", str_c(x, collapse = "+")), data = datos)

Coefficients:
(Intercept)  nivIngresos0  nivIngresos4  nivIngresos1  nivIngresos2
      28.929         4.303       -1.566         2.790         1.103
```

Variables:

- `modelos` actualizada que contiene, para cada valor del vector:
  - `x` con la variable  $x$  de ese modelo.
  - `y` con la variable  $y$  de ese modelo.
  - `modelo` con el propio modelo calculado.
  - `datos` con los datos que se usaron para el modelo.

Además, este apartado en concreto genera gráficos de dispersión para los modelos con variables cuantitativas, así como gráficos de caja y bigote para las variables cualitativas.

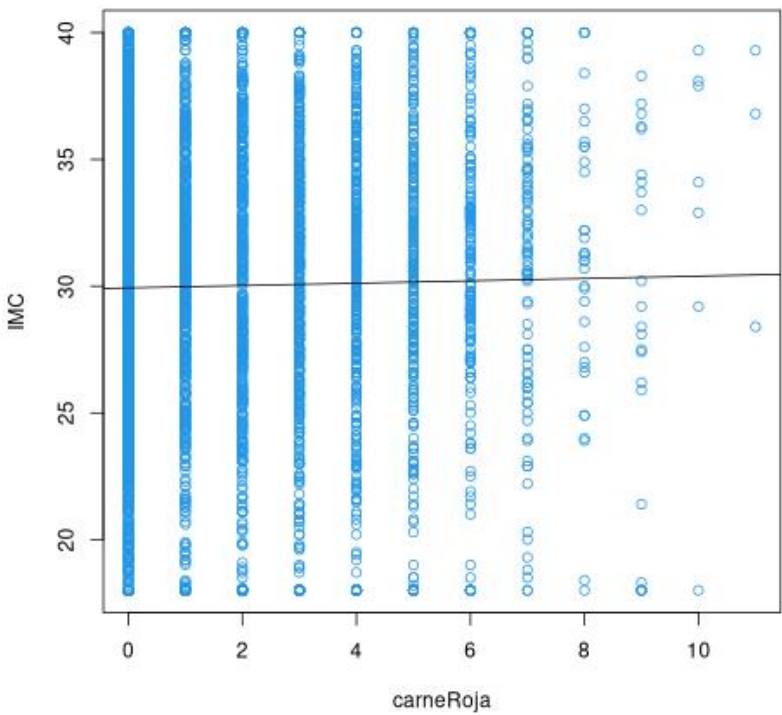
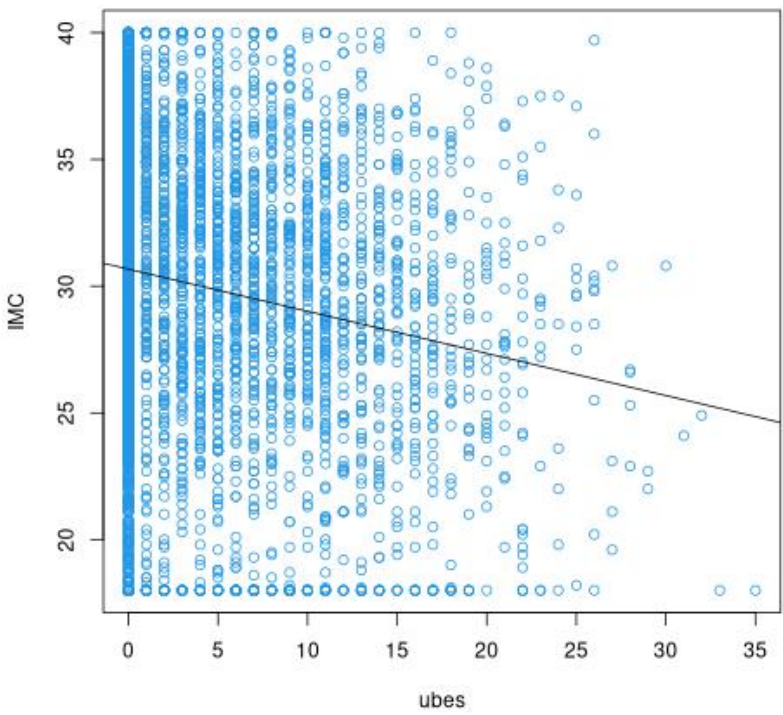
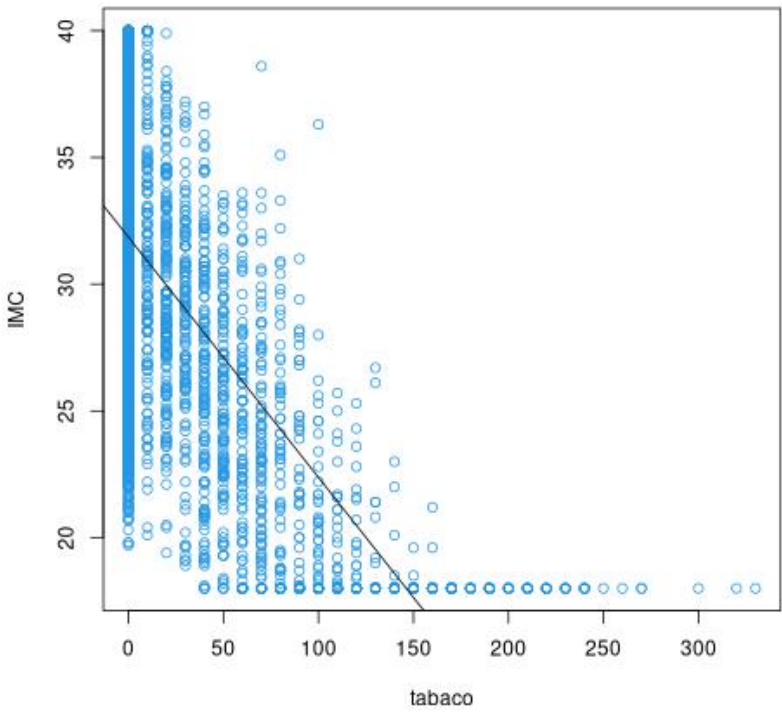
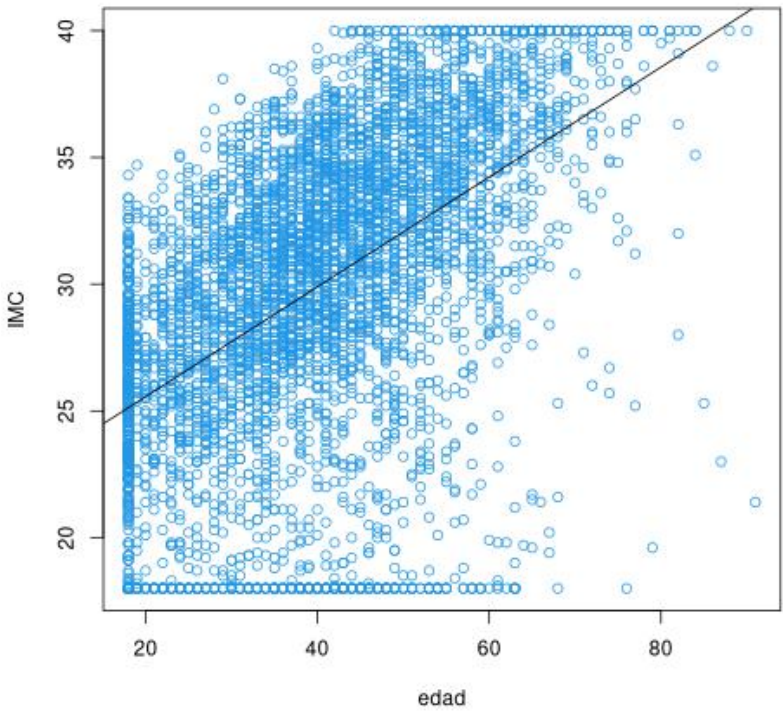
# Gráficos

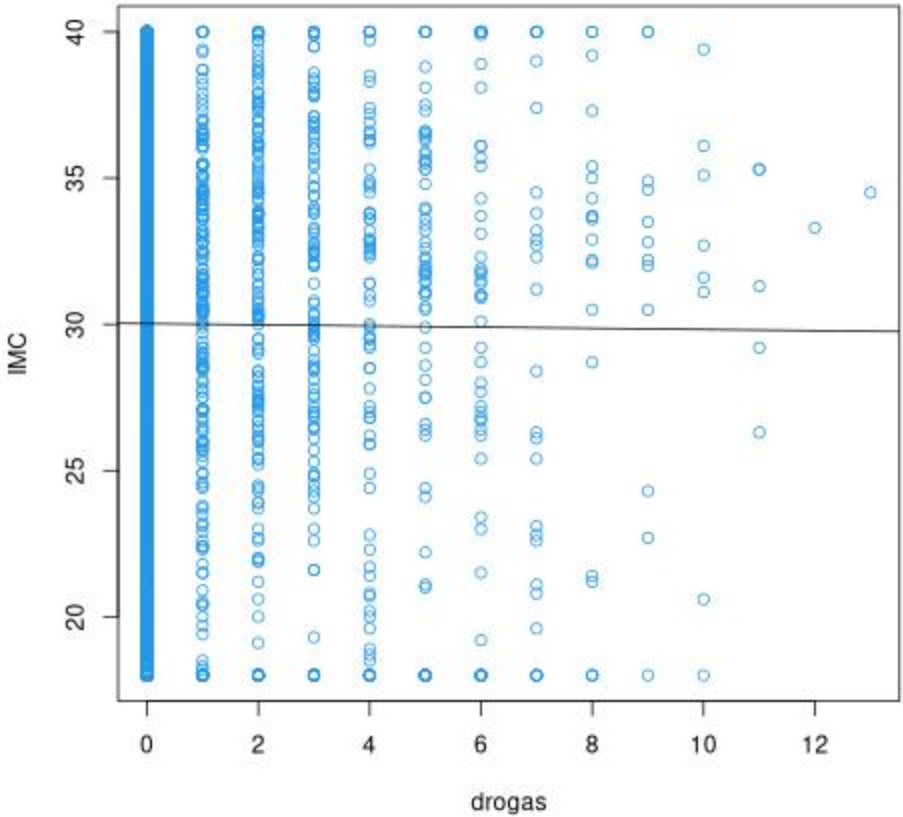
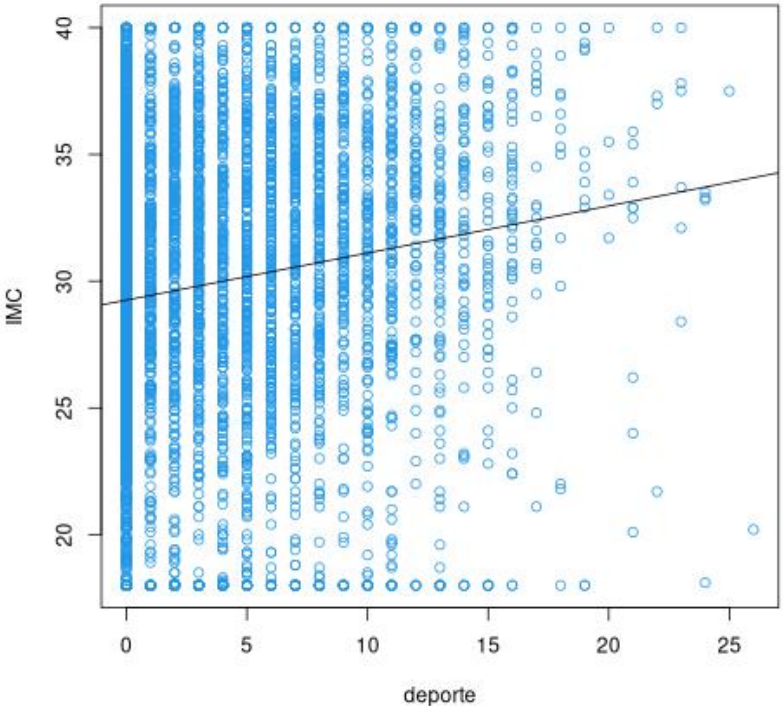
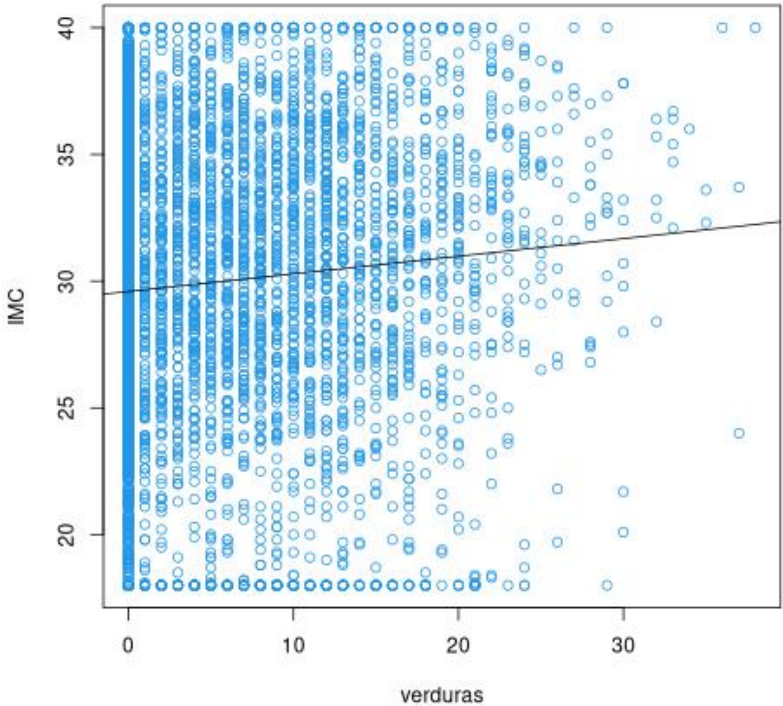


Las imágenes originales (mejor resolución) se encuentran en el directorio *Graficos* de la misma entrega.

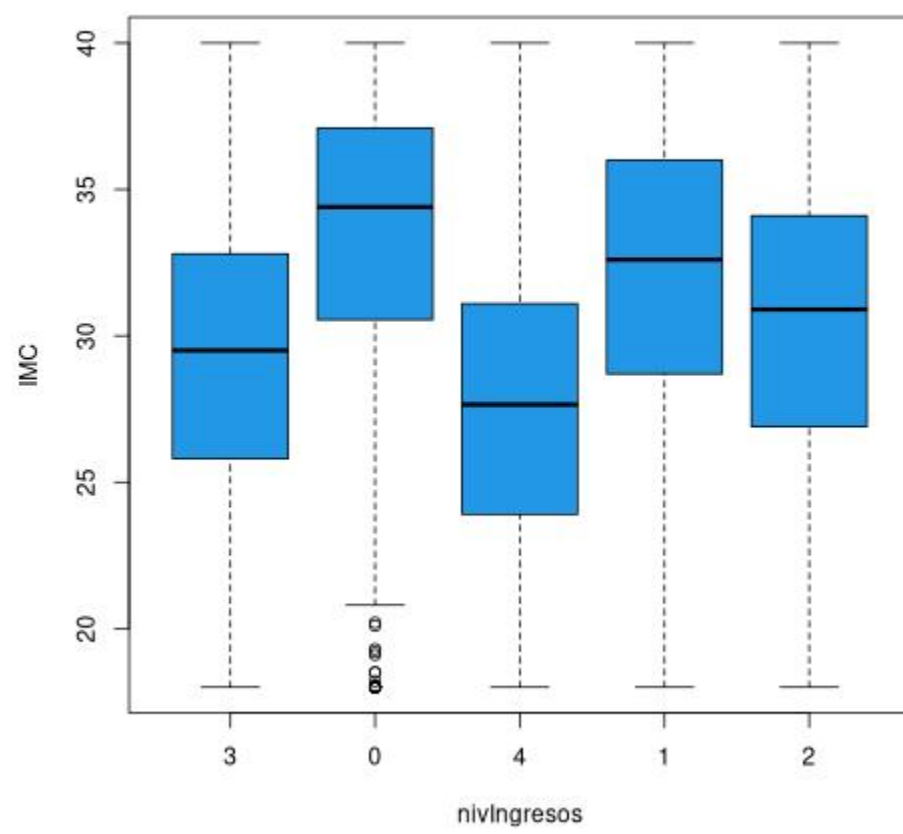
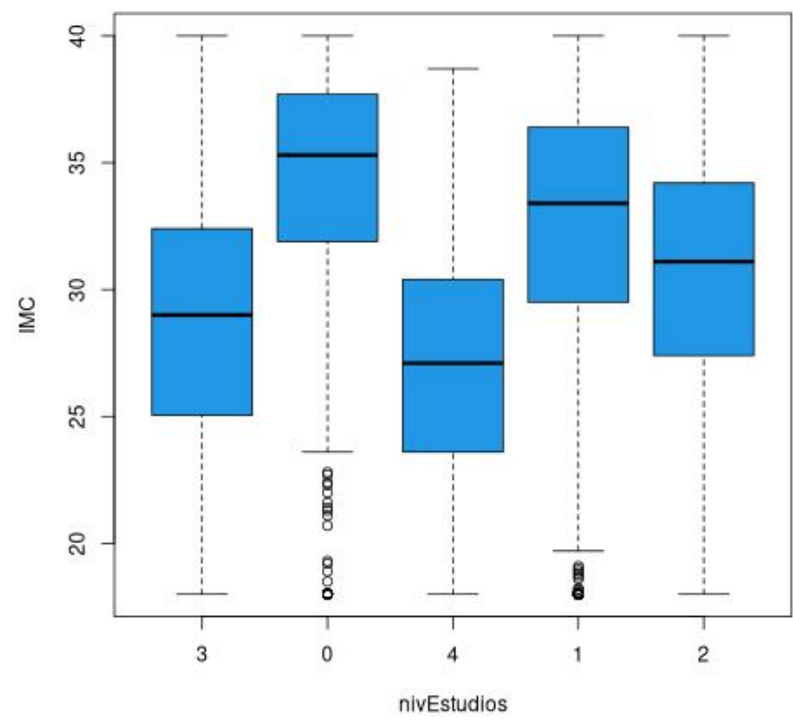
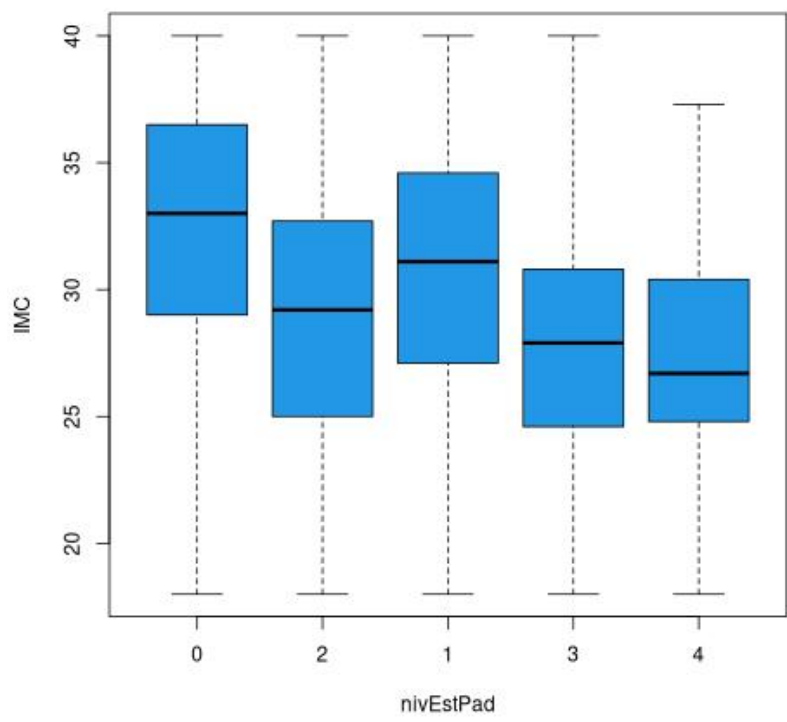
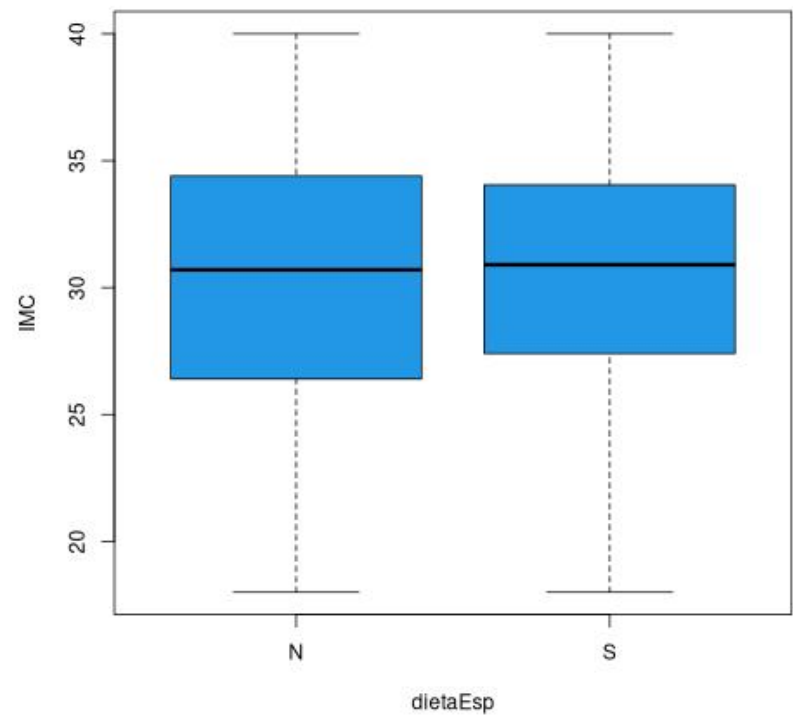
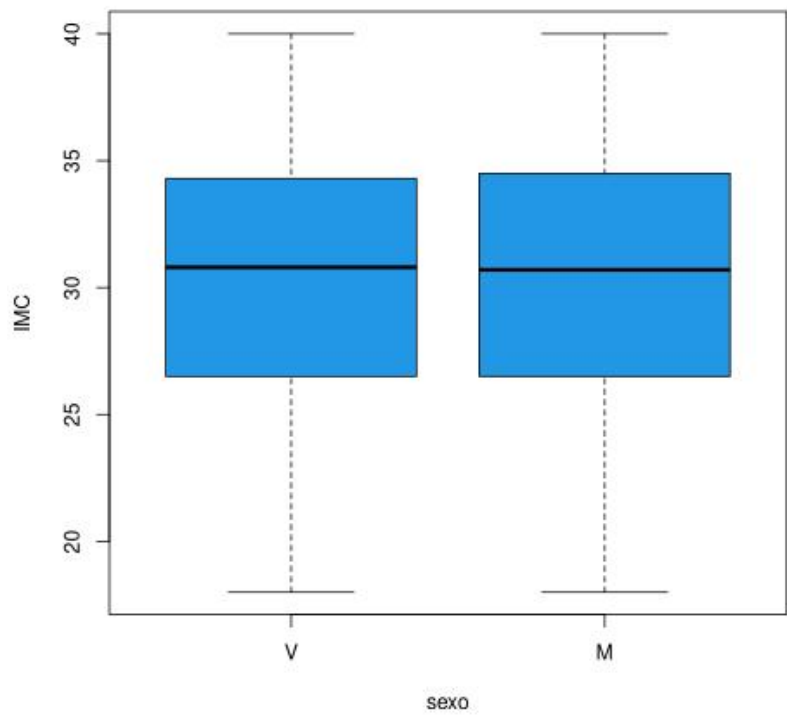
Se ha decidido usar el color azul para los datos ya que de esta forma, puede resaltarse la recta de regresión.

## Variables cuantitativas





Variables cualitativas





Explicación

A la hora de graficar los datos y su regresión lineal, hay que tener en cuenta si son variables cuantitativas o cualitativas.

Para el primer caso, basta con usar una función `plot(x, y)`, siendo  $x$  cada una de las variables explicatorias y siendo  $y$  la variable explicada (*IMC*); tras graficar la dispersión de los datos, con `abline(modelo)` se superpone la recta de regresión del modelo generado.

Para el segundo caso, se usa `boxplot(x, y)` y no se superpone la recta de regresión.

Todo el proceso anterior se ha compactado en la función `dibujarModelos()`, que además genera las imágenes como ficheros y tiene en cuenta el tipo de variable, por lo que genera cada gráfico correspondiente atendiendo a la variable  $x$  que recibe.

Apartado 07

Código de R

```
# Separa un conjunto de datos en 3 subconjuntos disjuntos: Entrenamiento,
# Test y Validacion; que se dividiran segun las proporciones indicadas.
# * datos: conjunto de datos del que obtener los subconjuntos
# * p1: porcentaje del subconjunto de Entrenamiento
# * p2: porcentaje que divide a '1-p1'
obtenerConjuntos <- function(datos, p1, p2) {
  rdatos <- 1:nrow(datos)
  rTrain <- sample(rdatos, p1 * length(rdatos))
  rTemp <- setdiff(rdatos, rTrain)
  rTest <- sample(rTemp, p2 * length(rTemp))
  rValid <- setdiff(rTemp, rTest)

  list(entrenamiento=datos[rTrain,], test=datos[rTest,], validacion=datos[rValid,])
}

# Conjuntos obtenidos
conjuntos <- obtenerConjuntos(datos, .6, .5)# 0.6 : Entrenamiento
```

Resultado

```
> conjuntos
$entrenamiento
# A tibble: 2,970 x 15
  peso altura sexo  edad tabaco  ubes carneRoja verduras deporte drogas dietaEsp
  <dbl>  <dbl> <fct>  <dbl>  <dbl> <dbl>    <dbl>    <dbl>    <dbl>  <dbl> <fct>
1  98.0   1.81 V      44     30     0         8         6         4         0 N
2  92.8   1.64 M      55      0     0         0         1         1         0 N
3  74.6   1.58 M      51     40     0         5         0         0         0 N
4  76.5   1.68 V      21      0     0         5         0         3         0 N
5 120.   1.78 M      52      0     0         0         0         0         0 N
6  94.1   1.58 M      58      0     0         1         5         0         2 N
7 100.   1.71 M      24      0     0         0         0         7         0 N
8  48.4   1.64 M      59    140     5         0        12         1         0 S
9  78.8   1.79 V      35     10     0         3         0         0         0 N
10 82.4   1.78 V      22      0    17         1         6         0         0 N
# ... with 2,960 more rows, and 4 more variables: nivEstPad <fct>, nivEstudios <fct>,
#   nivIngresos <fct>, IMC <dbl>

$test
# A tibble: 990 x 15
  peso altura sexo  edad tabaco  ubes carneRoja verduras deporte drogas dietaEsp
  <dbl>  <dbl> <fct>  <dbl>  <dbl> <dbl>    <dbl>    <dbl>    <dbl>  <dbl> <fct>
1  52.6   1.71 M      63    180     1         7         0         2         0 N
2  48.4   1.64 M      58    220    13         0         0         0         1 N
3  55.1   1.75 V      18    110     3         0         9         8         6 N
4 114.   1.69 V      63      0     0         2        19         7         0 N
5  97.2   1.74 V      41      0     5         1         9        10         0 N
6  76.3   1.69 V      18      0     0         1         0         0         0 N
7  94.2   1.57 M      47      0     0         2         1         7         0 N
8  87.3   1.57 M      47      0     0         0        10         2         0 N
9  99.7   1.73 V      33      0     0         3        12        16         2 N
10 103.   1.63 M      66      0    11         0         8         6         2 N
# ... with 980 more rows, and 4 more variables: nivEstPad <fct>, nivEstudios <fct>,
#   nivIngresos <fct>, IMC <dbl>

$validacion
# A tibble: 990 x 15
  peso altura sexo  edad tabaco  ubes carneRoja verduras deporte drogas dietaEsp
  <dbl>  <dbl> <fct>  <dbl>  <dbl> <dbl>    <dbl>    <dbl>    <dbl>  <dbl> <fct>
```

```
1 68.0 1.67 M 21 0 11 0 13 0 1 N
2 75.3 1.62 M 30 0 10 2 0 0 0 N
3 81.2 1.7 V 44 30 0 3 0 0 5 N
4 90.7 1.8 V 28 0 13 0 0 0 0 N
5 86.0 1.74 V 33 0 8 11 4 0 2 N
6 54.8 1.6 M 34 80 3 2 2 5 8 N
7 92.0 1.72 M 27 10 11 1 8 0 0 N
8 104. 1.76 M 54 0 3 4 12 13 0 N
9 107. 1.71 V 56 0 4 0 13 4 0 N
10 80.4 1.68 V 38 0 23 0 1 0 0 N
# ... with 980 more rows, and 4 more variables: nivEstPad <fct>, nivEstudios <fct>,
# nivIngresos <fct>, IMC <dbl>
```

Funciones:

- `obtenerConjuntos()` que divide un conjunto de datos en 3 subconjuntos.

Variables:

- `conjuntos` que contiene 3 subconjuntos disjuntos:
  - `entrenamiento` , correspondiente al 60 % de los datos.
  - `test` , correspondiente al 20 % de los datos.
  - `validacion` , correspondiente al 20 % de los datos.

Explicación

Para poder evaluar correctamente la predicción de un modelo, primero es necesario eliminar fluctuaciones estadísticas que existen entre las variables; para ello, se usa la técnica de separar el conjunto de datos en 2 subconjuntos disjuntos: uno para entrenar el modelo, y otro más pequeño para probarlo. Esto además, elimina el problema de que se produzca sobreajuste.

En este caso sin embargo, se usará un tercer conjunto del mismo tamaño que el conjunto de test, que permitirá medir la calidad del modelo de una forma más objetiva.

Para ello, se almacenan los subconjuntos en `conjuntos` , obtenidos a través de `obtenerConjuntos(datos, 0.6, 0.5)` .

Se obtiene un conjunto de entrenamiento con un tamaño del 60 % del original, y los datos restantes (40 %) se dividen en 2 subconjuntos de un 50 % cada uno ( $0.4 \cdot 0.5 = 0.2$ ), resultando en los otros 2 subconjuntos: test con un 20 %, y validación con otro 20 %.

Apartado 08

Código de R

```
# Funciones -----

# Calcula los valores: MSE, varY, R2 y R² a partir de un modelo.
# * datos: conjunto de datos del modelo
# * modelo: modelo
# * y: variable a explicar
calcularR2 <- function(df, mod, y) {
  MSE <- mean((df[[y]] - predict.lm(mod, df)) ^ 2)
  varY <- mean(df[[y]] ^ 2) - mean(df[[y]]) ^ 2

  R2 <- 1 - MSE / varY
  R2a <- 1 - (1- R2) * (nrow(df) - 1) / (nrow(df) - mod$rank)

  tibble(MSE=MSE, varY=varY, R2=R2, R2a=R2a)
}

# Calcula el coeficiente de determinación (R²) ajustado a partir
# de un conjunto de entrenamiento y un conjunto de test.
# * dfTrain: conjunto de entrenamiento
# * dfTest: conjunto de test
# * y: variable explicada
# * x: variable explicatoria
calcularR2ajustado <- function(dfTrain, dfTest, y, x) {
  modelo <- ajustelLineal(dfTrain, y, x)
  coeficientes <- calcularR2(dfTest, modelo, y)

  coeficientes$R2a
```



```

}

# Calculos -----

# Variables predictoras
predictoras <- names(datos[-length(datos)]) # Todas menos 'IMC' (la ultima)

# Subconjuntos
entrenamiento <- conjuntos$entrenamiento
test          <- conjuntos$test
validacion    <- conjuntos$validacion

# Calcular los 14 R² de los 14 modelos lineales unidimensionales
r2a <- predictoras %>% map_dbl(calcularR2ajustado, dfTrain=entrenamiento, dfTest=test, y="IMC")

# Esto permitira escoger el mejor modelo eliminando fluctuaciones estadisticas

# Obtener la mejor variable para una prediccion unidimensional
mejorVariable <- predictoras[which.max(r2a)]

# Obtener el mejor modelo usando el conjunto de entrenamiento
mejorModelo <- ajusteLineal(entrenamiento, "IMC", mejorVariable)

# Obtener su R² con el conjunto de test
evaluacionTest <- calcularR2(test, mejorModelo, "IMC")

# Obtener su R² con el conjunto de validacion
evaluacionValidacion <- calcularR2(validacion, mejorModelo, "IMC")

```

### Resultado

```

> predictoras
[1] "peso"      "altura"    "sexo"      "edad"      "tabaco"
[6] "ubres"     "carneRoja" "verduras"  "deporte"   "drogas"
[11] "dietaEsp"  "nivEstPad" "nivEstudios" "nivIngresos"

> r2a
[1] 0.8466999320 -0.0015412904 -0.0009151768 0.3058570048 0.4616058156
[6] 0.0076416906 -0.0012383140 0.0065498871 0.0121926236 -0.0014304505
[11] -0.0014070115 0.0651550267 0.1592479228 0.1285421465

> mejorVariable
[1] "peso"

> mejorModelo

Call:
lm(formula = str_c(y, "~", str_c(x, collapse = "+")), data = datos)

Coefficients:
(Intercept)      peso
      4.6684      0.2915

> evaluacionTest
# A tibble: 1 x 4
  MSE varY    R2  R2a
<dbl> <dbl> <dbl> <dbl>
1  5.55  36.3 0.847 0.847

> evaluacionValidacion
# A tibble: 1 x 4
  MSE varY    R2  R2a
<dbl> <dbl> <dbl> <dbl>
1  5.55  35.1 0.842 0.842

```

Funciones:

- `calcularR2()` que calcula tanto el  $R^2$ , como el  $MSE$ , la  $\sigma_y^2$  y el  $R^2$  ajustado.
- `calcularR2ajustado()` que calcula el  $R^2$  ajustado usando la anterior.

Variables:

- `predictoras` con el nombre de todas las variables explicatorias.
- `mejorVariable` con el nombre de la variable que mejor predice *IMC* en un modelo unidimensional.
- `mejorModelo` con el modelo resultante de aplicar la mejor variable.

- `evaluacionTest` con los datos resultantes de evaluar el modelo con el conjunto de test.
- `evaluacionValidacion` con los datos resultantes de evaluar el modelo con el conjunto de validación.

### Explicación

Usando los subconjuntos obtenidos en el apartado anterior y aplicando programación funcional, se crean las funciones `calcularR2()` y `calcularR2ajustado()` para realizar una búsqueda de la variable explicatoria más significativa para generar un modelo de regresión lineal unidimensional.

Se calculan los  $R^2$  ajustados de todos los modelos posibles con las 14 variables explicatorias en la variable `predictoras`, porque la variable que genere un modelo con mayor  $R^2$  ajustado será el mejor de todos los generados.

Finalmente, se evalúa el `mejorModelo` con el conjunto de test así como con el conjunto de validación, obteniendo en ambos casos un  $R^2$  bastante cercano a 1.

## Apartado 09

### Código de R

```
# Funciones -----

# La funcion 'ajusteLineal()' del apartado 08 se ha modificado para
# que pueda reutilizarse en este apartado (ahora es compatible con ambos)

# Encuentra el mejor ajuste lineal usando un conjunto de variables y
# calculando sus R^2 ajustados con conjuntos de Entrenamiento y de Test.
# * dfTrain:      conjunto de entrenamiento
# * dfTest:       conjunto de test
# * predictoras:  variables explicatorias del modelo
obtenerMejorAjusteLineal <- function(dfTrain, dfTest, predictoras) {

  # Variables iniciales
  mejoresVariables <- character(0)
  R2a              <- 0

  # Bucle del algoritmo
  repeat {
    # Inicializar
    R2as <- map_dbl(predictoras, ~calcularR2ajustado(dfTrain, dfTest, "IMC", c(mejoresVariables, .)))
    i    <- which.max(R2as)
    mejorR2a <- R2as[i]

    if (mejorR2a <= R2a) {
      break
    }

    # Mostrar por la consola los valores calculados durante la ejecucion
    cat(sprintf("%1.8f %s\n", mejorR2a, predictoras[i]))

    # Actualizar los valores para la siguiente iteracion
    R2a <- mejorR2a
    mejoresVariables <- c(mejoresVariables, predictoras[i])
    predictoras <- predictoras[-i]
  }

  modelo <- ajusteLineal(dfTrain, "IMC", mejoresVariables)

  list(variables=mejoresVariables, modelo=modelo)
}

# Calculos -----

# Variables simples
predictorasSimple <- names(datos[-length(datos)]) # Todas menos 'IMC' (la ultima)
mejorModeloSimple <- obtenerMejorAjusteLineal(conjuntos$entrenamiento, conjuntos$test, predictorasSimple)$modelo

# Variables combinadas
predictorasCombo <- crossing(var1=predictoras, var2=predictoras) %>% pmap_chr(str_c, sep=":")
mejorModeloCombo <- obtenerMejorAjusteLineal(conjuntos$entrenamiento, conjuntos$test, predictorasCombo)$modelo
```

### Resultado

```
> predictorasSimple
[1] "peso"      "altura"    "sexo"      "edad"      "tabaco"
[6] "ubes"      "carneRoja" "verduras"  "deporte"   "drogas"
[11] "dietaEsp"  "nivEstPad" "nivEstudios" "nivIngresos"

> mejorModeloSimple

Call:
lm(formula = str_c(y, "~", str_c(x, collapse = "+")), data = datos)

Coefficients:
(Intercept)      peso      altura      tabaco      edad      drogas
 58.024723    0.323750  -33.321498  -0.005901   0.013008   0.037291
nivEstudios0 nivEstudios4 nivEstudios1 nivEstudios2      ubes      verduras
 0.309555   -0.154860   0.215460   0.130442  -0.009296   0.004310
dietaEspS
 0.056532

> predictorasCombo
[1] "altura:altura"      "altura:carneRoja"      "altura:deporte"
[4] "altura:dietaEsp"    "altura:drogas"         "altura:edad"
[7] "altura:nivEstPad"   "altura:nivEstudios"    "altura:nivIngresos"
[10] "altura:peso"        "altura:sexo"          "altura:tabaco"
[13] "altura:ubes"       "altura:verduras"      "carneRoja:altura"
[16] "carneRoja:carneRoja" "carneRoja:deporte"    "carneRoja:dietaEsp"
[19] "carneRoja:drogas"   "carneRoja:edad"       "carneRoja:nivEstPad"
[22] "carneRoja:nivEstudios" "carneRoja:nivIngresos" "carneRoja:peso"
[25] "carneRoja:sexo"     "carneRoja:tabaco"     "carneRoja:ubes"
[28] "carneRoja:verduras" "deporte:altura"       "deporte:carneRoja"
[31] "deporte:deporte"    "deporte:dietaEsp"     "deporte:drogas"
[34] "deporte:edad"       "deporte:nivEstPad"    "deporte:nivEstudios"
[37] "deporte:nivIngresos" "deporte:peso"         "deporte:sexo"
[40] "deporte:tabaco"     "deporte:ubes"         "deporte:verduras"
[43] "dietaEsp:altura"    "dietaEsp:carneRoja"   "dietaEsp:deporte"
[46] "dietaEsp:dietaEsp"  "dietaEsp:drogas"      "dietaEsp:edad"
[49] "dietaEsp:nivEstPad" "dietaEsp:nivEstudios" "dietaEsp:nivIngresos"
[52] "dietaEsp:peso"      "dietaEsp:sexo"        "dietaEsp:tabaco"
[55] "dietaEsp:ubes"     "dietaEsp:verduras"    "drogas:altura"
[58] "drogas:carneRoja"   "drogas:deporte"       "drogas:dietaEsp"
[61] "drogas:drogas"      "drogas:edad"          "drogas:nivEstPad"
[64] "drogas:nivEstudios" "drogas:nivIngresos"   "drogas:peso"
[67] "drogas:sexo"        "drogas:tabaco"        "drogas:ubes"
[70] "drogas:verduras"   "edad:altura"          "edad:carneRoja"
[73] "edad:deporte"      "edad:dietaEsp"        "edad:drogas"
[76] "edad:edad"         "edad:nivEstPad"       "edad:nivEstudios"
[79] "edad:nivIngresos"  "edad:peso"            "edad:sexo"
[82] "edad:tabaco"       "edad:ubes"            "edad:verduras"
[85] "nivEstPad:altura"   "nivEstPad:carneRoja"   "nivEstPad:deporte"
[88] "nivEstPad:dietaEsp" "nivEstPad:drogas"      "nivEstPad:edad"
[91] "nivEstPad:nivEstPad" "nivEstPad:nivEstudios" "nivEstPad:nivIngresos"
[94] "nivEstPad:peso"     "nivEstPad:sexo"       "nivEstPad:tabaco"
[97] "nivEstPad:ubes"     "nivEstPad:verduras"    "nivEstudios:altura"
[100] "nivEstudios:carneRoja" "nivEstudios:deporte"   "nivEstudios:dietaEsp"
[103] "nivEstudios:drogas"  "nivEstudios:edad"     "nivEstudios:nivEstPad"
[106] "nivEstudios:nivEstudios" "nivEstudios:nivIngresos" "nivEstudios:peso"
[109] "nivEstudios:sexo"    "nivEstudios:tabaco"    "nivEstudios:ubes"
[112] "nivEstudios:verduras" "nivIngresos:altura"    "nivIngresos:carneRoja"
[115] "nivIngresos:deporte" "nivIngresos:dietaEsp"  "nivIngresos:drogas"
[118] "nivIngresos:edad"    "nivIngresos:nivEstPad" "nivIngresos:nivEstudios"
[121] "nivIngresos:nivIngresos" "nivIngresos:peso"      "nivIngresos:sexo"
[124] "nivIngresos:tabaco"  "nivIngresos:ubes"     "nivIngresos:verduras"
[127] "peso:altura"        "peso:carneRoja"       "peso:deporte"
[130] "peso:dietaEsp"      "peso:drogas"          "peso:edad"
[133] "peso:nivEstPad"     "peso:nivEstudios"     "peso:nivIngresos"
[136] "peso:peso"          "peso:sexo"            "peso:tabaco"
[139] "peso:ubes"          "peso:verduras"        "sexo:altura"
[142] "sexo:carneRoja"     "sexo:deporte"         "sexo:dietaEsp"
[145] "sexo:drogas"        "sexo:edad"            "sexo:nivEstPad"
[148] "sexo:nivEstudios"   "sexo:nivIngresos"     "sexo:peso"
[151] "sexo:sexo"          "sexo:tabaco"          "sexo:ubes"
[154] "sexo:verduras"      "tabaco:altura"        "tabaco:carneRoja"
[157] "tabaco:deporte"     "tabaco:dietaEsp"      "tabaco:drogas"
[160] "tabaco:edad"        "tabaco:nivEstPad"     "tabaco:nivEstudios"
[163] "tabaco:nivIngresos" "tabaco:peso"          "tabaco:sexo"
[166] "tabaco:tabaco"      "tabaco:ubes"          "tabaco:verduras"
[169] "ubes:altura"        "ubes:carneRoja"       "ubes:deporte"
[172] "ubes:dietaEsp"      "ubes:drogas"          "ubes:edad"
[175] "ubes:nivEstPad"     "ubes:nivEstudios"     "ubes:nivIngresos"
[178] "ubes:peso"          "ubes:sexo"            "ubes:tabaco"
[181] "ubes:ubes"          "ubes:verduras"        "verduras:altura"
[184] "verduras:carneRoja" "verduras:deporte"     "verduras:dietaEsp"
[187] "verduras:drogas"    "verduras:edad"        "verduras:nivEstPad"
[190] "verduras:nivEstudios" "verduras:nivIngresos" "verduras:peso"
[193] "verduras:sexo"      "verduras:tabaco"      "verduras:ubes"
[196] "verduras:verduras"

> mejorModeloCombo
```

```
Call:
lm(formula = str_c(y, "~", str_c(x, collapse = "+")), data = datos)

Coefficients:
(Intercept)          altura          sexoM          carneRoja
  1.045e+01      -5.989e+00      7.630e+00      7.236e-02
      tabaco          edad          deporte          dietaEspS
 -5.935e-02      1.312e-01      9.402e-02      1.176e-02
 peso:sexoV      peso:sexoM      peso:altura      sexoM:altura
  8.140e-01      8.261e-01      -2.780e-01      -4.959e+00
 tabaco:verduras      carneRoja:edad      tabaco:edad      peso:carneRoja
  1.289e-07      8.351e-05      -4.522e-06      -2.249e-05
 peso:tabaco      altura:tabaco      sexoM:tabaco      tabaco:nivEstPad2
 -3.912e-06      3.424e-02      3.032e-03      8.621e-05
 tabaco:nivEstPad1      tabaco:nivEstPad3      tabaco:nivEstPad4      altura:carneRoja
  3.278e-05      -1.879e-04      1.454e-03      -4.128e-02
 sexoM:carneRoja      edad:nivIngresos0      edad:nivIngresos4      edad:nivIngresos1
 -3.551e-03      -9.655e-05      -1.363e-04      1.250e-05
 edad:nivIngresos2      tabaco:dietaEspS      edad:deporte      peso:deporte
  1.078e-04      -2.918e-04      4.324e-05      -2.784e-05
 peso:edad      altura:edad      sexoM:edad      drogas:nivEstudios3
 -4.780e-06      -7.498e-02      -6.826e-03      1.502e-01
 drogas:nivEstudios0      drogas:nivEstudios4      drogas:nivEstudios1      drogas:nivEstudios2
  1.495e-01      1.509e-01      1.567e-01      1.472e-01
 altura:drogas      altura:deporte      sexoM:deporte      dietaEspS:deporte
 -8.692e-02      -5.330e-02      -4.796e-03      -3.057e-03
 verduras:carneRoja      altura:verduras      sexoM:drogas
 -1.642e-04      2.803e-04      -3.794e-03
```

Funciones:

- `obtenerMejorAjusteLineal()` que genera el mejor modelo a partir de un conjunto de entrenamiento, uno de test, y unas variables predictoras.

Variables:

- `predictorasSimples` nombres de las variables para generar el modelo de variables simple.
- `mejorModeloSimple` modelo generado con la mejor de las variables simples.
- `predictorasCombo` nombres de los pares de variables para generar el modelo de variables combinadas.
- `mejorModeloCombo` modelo generado con la mejor de las variables combinadas.

Si se habilita que `obtenerMejorAjusteLineal()` muestre los datos de forma iterativa, la evolución de los valores obtenidos es la siguiente (he usado 8 decimales porque en mi caso, salen valores muy parecidos):

```
> mejorModeloSimple <- obtenerMejorAjusteLineal(conjuntos$entrenamiento, conjuntos$test, predictorasSimple)$modelo
0.85138148 peso
0.99348556 altura
0.99352425 tabaco
0.99362600 edad
0.99368498 nivIngresos
0.99375586 drogas
0.99379060 ubes
0.99380214 deporte

> mejorModeloCombo <- obtenerMejorAjusteLineal(conjuntos$entrenamiento, conjuntos$test, predictorasCombo)$modelo
0.91473195 peso:sexo
0.99894855 altura:peso
0.99917853 altura:altura
0.99923559 altura:sexo
0.99965227 sexo:sexo
0.99965253 carneRoja:tabaco
0.99965263 edad:verduras
```

Explicación

Finalmente, se usarán varias variables para generar un modelo óptimo.

Pese a que podría hacerse de muchas formas (aplicar logaritmos, modelos aditivos...), se estudiará qué modelo es mejor: uno con variables simples, o bien, uno con variables relacionadas en pares.

Igual que en el apartado anterior, primero se obtendrán las variables sobre la que iterar una función que genere el modelo (en este caso, `obtenerMejorAjusteLineal()` ) y se obtendrá el modelo con la mejor de esas variables. Este proceso se repetirá 2 veces: una para el modelo con variables simples y otra para el modelo con variables combinadas.

En el siguiente apartado, se analizarán los datos obtenidos en este.

# Apartado 10

## Código de R

```
# Modelo unidimensional
evaluacionSimple <- calcularR2(validacion, mejorModeloSimple, "IMC")

# Modelo multidimensional
evaluacionCombo <- calcularR2(validacion, mejorModeloCombo, "IMC")
```

## Resultado

```
> evaluacionSimple
# A tibble: 1 x 4
  MSE varY    R2  R2a
<dbl> <dbl> <dbl> <dbl>
1 0.207  35.1 0.994 0.994

> evaluacionCombo
# A tibble: 1 x 4
  MSE varY    R2  R2a
<dbl> <dbl> <dbl> <dbl>
1 0.00912  35.1  1.00  1.00
```

Variables:

- `valuacionSimple` que ofrece datos sobre la calidad del modelo de variables simples.
- `evaluacionCombo` que ofrece datos sobre la calidad del modelo de variables combinadas.

## Explicación

Se puede observar que el modelo con variables simples, pese a que tiene un  $R^2$  ajustado menor, se aproxima más a la "realidad", ya que el modelo con variables combinadas parece haber sufrido un sobreajuste.

No es normal que  $R^2 = 1$ , ese es solo un valor teórico y vendría a representar un modelo predictivo perfecto, cosa que no es posible.

# Apartado 11

## ¿Qué utilidad podría tener el modelo matemático obtenido?

Ya que el modelo se ha construido a partir de un conjunto de datos que giran en torno al cálculo del *IMC*, un posible uso podría ser el de predecir el *IMC* en base a un mayor número de factores o cualquier otra medida parecida.

## ¿Qué puede deducirse sobre la relación entre las variables a partir del modelo?

Como la variable *IMC* es un cálculo directo de las variables *peso* y *altura*, una de ellas ha estado siempre presente en las variables más influyentes a la hora de calcular los mejores modelos, tanto en el caso simple como en el combinado.

## ¿Qué problemas se han encontrado en el desarrollo?

Quitando mi poca experiencia con R y entender algunos conceptos estadísticos -porque no se me da especialmente bien-, diría que el principal problema que me he econtrado ha sido la complejidad que supone realmente generar un modelo adecuado y qué camino seguir, qué métodos utilizar, qué variables... el proceso de selección.

## ¿Qué más me ha llamado la atención en el proceso?

Lo grande que es la estadística y que, para una sola cosa, existen infinitas formas de hacerlas; en este caso, he evaluado al final un modelo de variables simples contra uno de variables combinadas, pero solo he combinado de 2 en 2, podrían haberse hecho muchas más, haber elegido otro ajuste diferente...

Y sin embargo, la potencia de R no queda desapercibida, ya que puede dar lugar a algoritmos y cálculos muy cómodos, junto a una presentación intuitiva de los datos en la propia consola.

### **¿Qué más podría hacerse y cómo plantearlo?**

Podrían usarse variables combinadas más grandes, modificando el apartado 09, y estudiar qué modelo presenta una mejor predicción del modelo simple al modelo con mayor variables combinadas.

También podrían usarse otras técnicas, como aplicar logaritmos para generar el modelo.