Trabajo de modelización estadística

1. Carga en memoria el fichero CSV como tibble, asegurándote de que las variables cualitativas sean leídas como factores

```
library(tidyverse)
#1
#Cargamos el csv
datos<- read_csv("./17456.csv",
                 col_types =
                   cols(.default = col_double(),
                        sexo = col_factor(),
                        dietaEsp = col_factor()))
datos
```

Para este apartado simplemente importamos la librería que vamos a usar y cargamos el archivo CSV como se pide en el enunciado. Así se vería:

```
> datos
# A tibble: 5,000 × 14
    peso altura sexo   edad tabaco  ubes carneRoja verduras deporte drogas dietaEsp nivEstPad
   <dbl>  <dbl> <fct> <dbl>  <dbl> <dbl>     <dbl>    <dbl>   <dbl>  <dbl> <fct>        <dbl>
 1  86.1   1.75 V        52      0     0         5        0       4      0 N                1
 2  72.1   1.73 V        60      0     0         2       16       5      0 N                0
 3  89.7   1.85 V        41      0     0         6       11       6      0 N                0
 4  75.1   1.7  V        59      0     3         3        7       0      0 S                1
 5  69.6   1.74 V        40    140     0         0       22       5      0 N                0
 6  71.9   1.6  M        40    120     0         1        6       4      0 N                1
 7  89.0   1.78 V        37      0     2         0        1       6      3 N                1
 8  76.9   1.66 M        45      0     9         5        1       2      0 N                3
 9  87.8   1.81 V        44      0     0         2        0       0      0 N                1
10  71.4   1.7  M        27      0     0         5        8       3      0 N                0
# i 4,990 more rows
# i 2 more variables: nivEstudios <dbl>, nivIngresos <dbl>
```

2. Construye una nueva columna llamada IMC que sea igual al peso dividido por la altura al cuadrado. La variable explicada será IMC, las variables explicatorias serán el resto de 12 variables exceptuando peso y altura.

```
#2
#Añadimos la columna IMC
datos <- add_column(datos, IMC = datos$peso/(datos$altura^2))
datos
```

Creamos una columna nueva aplicando la formula para aplicar IMC a cada fila. Así se vería:

```
> datos <- add_column(datos, IMC = datos$peso/(datos$altura^2))
> datos
# A tibble: 5,000 × 15
    peso altura sexo   edad tabaco  ubes carneRoja verduras deporte drogas dietaEsp nivEstPad nivEstudios nivIngresos   IMC
   <dbl>  <dbl> <fct> <dbl>  <dbl> <dbl>     <dbl>    <dbl>   <dbl>  <dbl> <fct>        <dbl>       <dbl>       <dbl> <dbl>
 1  74.0   1.65 M        34      0     0         4       10      10      6 N                2           4           4  27.2
 2  75.6   1.67 V        41     30     0         0        0       4      0 N                3           4           4  27.1
 3  67.8   1.72 V        42     10     9         1        0       7      0 N                2           2           2  22.9
 4  59.9   1.57 M        27      0     6         2        0       0      0 N                2           1           0  24.3
 5  98.6   1.85 V        56     80     7         3        3       0      0 S                0           1           3  28.8
 6  62.7   1.77 V        18     10     6         6        8       4      0 N                2           4           4  20.0
 7 113.    1.75 V        48    100     0         1        0       3      0 N                1           0           0  37.0
 8 110.    1.83 V        40      0     0         2       15      16      0 N                1           2           2  32.8
 9  61.4   1.6  M        33     20    13         0       11      15      1 N                1           3           2  24
10 120.    1.73 V        40    190     0         4       15      10      0 N                0           0           0  40.0
# i 4,990 more rows
# i Use `print(n = ...)` to see more rows
```

3. Elimina completamente las filas que tengan algún valor NA en una de sus columnas.

```
#3
#Eliminamos las columnas con valores na
#Primero vemos que columnas tienen valores na
na <- as.data.frame(
  cbind(
    lapply(
      lapply(datos, is.na), sum)
  )
)
na

#Ahora seleccionamos las columnas que no tengan valores na
datos <- datos %>% select(-contains(rownames(subset(na, na$V1 != 0))))
datos
```

En este apartado primero comprobamos que columnas tienen valores NA. Si ejecutamos na veríamos lo siguiente:

```
> na
            V1
peso         0
altura       0
sexo         0
edad         0
tabaco       0
ubes         0
carneRoja    0
verduras     0
deporte      0
drogas      51
dietaEsp     0
nivEstPad    0
nivEstudios  0
nivIngresos  0
IMC          0
```

Por último, seleccionamos todas las filas de datos que en el dataframe no tengan un valor de NA, se vería así:

```
> datos
# A tibble: 5,000 × 14
    peso altura sexo   edad tabaco  ubes carneRoja verduras deporte dietaEsp nivEstPad
   <dbl>  <dbl> <fct> <dbl>  <dbl> <dbl>     <dbl>    <dbl>   <dbl> <fct>        <dbl>
 1  86.1   1.75 V        52      0     0         5        0       4 N                1
 2  72.1   1.73 V        60      0     0         2       16       5 N                0
 3  89.7   1.85 V        41      0     0         6       11       6 N                0
 4  75.1   1.7  V        59      0     3         3        7       0 S                1
 5  69.6   1.74 V        40    140     0         0       22       5 N                0
 6  71.9   1.6  M        40    120     0         1        6       4 N                1
 7  89.0   1.78 V        37      0     2         0        1       6 N                1
 8  76.9   1.66 M        45      0     9         5        1       2 N                3
 9  87.8   1.81 V        44      0     0         2        0       0 N                1
10  71.4   1.7  M        27      0     0         5        8       3 N                0
# i 4,990 more rows
# i 3 more variables: nivEstudios <dbl>, nivIngresos <dbl>, IMC <dbl>
```

4. Calcula las medias y desviaciones típicas (no cuasidesviación) de todas las variables numéricas.

```
#4
#Nos quedamos con las columnas que tengan valores solo numericos
numeric <- as.data.frame(
  cbind(
    lapply(
      datos, is.numeric)
  )
)
numeric
numeric_df <- datos %>% select(any_of(rownames(subset(numeric, numeric$V1 == TRUE))))
numeric_df
```

Hacemos algo parecido al apartado anterior, guardamos en un dataframe que columnas son numéricas:

```
> numeric
                 V1
peso           TRUE
altura         TRUE
sexo          FALSE
edad           TRUE
tabaco         TRUE
ubes           TRUE
carneRoja      TRUE
verduras       TRUE
deporte        TRUE
dietaEsp      FALSE
nivEstPad      TRUE
nivEstudios    TRUE
nivIngresos    TRUE
IMC            TRUE
```

Y hacemos una selección igual que antes, solo que esta vez nos quedamos solo con los valores TRUE:

```
> numeric_df
# A tibble: 5,000 × 12
    peso altura  edad tabaco  ubes carneRoja verduras deporte nivEstPad nivEstudios nivIngresos
   <dbl>  <dbl> <dbl>  <dbl> <dbl>     <dbl>    <dbl>   <dbl>     <dbl>       <dbl>       <dbl>
 1  86.1   1.75    52      0     0         5        0       4         1           3           2
 2  72.1   1.73    60      0     0         2       16       5         0           1           0
 3  89.7   1.85    41      0     0         6       11       6         0           1           1
 4  75.1   1.7     59      0     3         3        7       0         1           1           1
 5  69.6   1.74    40    140     0         0       22       5         0           2           0
 6  71.9   1.6     40    120     0         1        6       4         1           2           2
 7  89.0   1.78    37      0     2         0        1       6         1           0           0
 8  76.9   1.66    45      0     9         5        1       2         3           4           4
 9  87.8   1.81    44      0     0         2        0       0         1           2           3
10  71.4   1.7     27      0     0         5        8       3         0           2           3
# i 4,990 more rows
# i 1 more variable: IMC <dbl>
```

Ahora calculamos la media de todas las columnas de esta manera:

```
#Calculamos la media de todas las columnas
colMeans(numeric_df)
```

Y nos daría:

```
> colMeans(numeric_df)
      peso     altura       edad     tabaco       ubes  carneRoja   verduras    deporte nivEstPad nivEstudios nivIngresos        IMC
 77.412028   1.701312  40.208800  19.578000   3.885800   1.726600   5.968200   4.151200  1.254600   2.213600   2.160000  26.698030
```

Y calculamos la desviación típica:

```r
#Calculas la desviacion tipica
#Creamos una funcion que la calcule
desviacion_tipica_fn <- function(ap) {
  return(sqrt(mean(ap^2) - mean(ap)^2))
}

#Aplicamos funcion a todo el dataframe
desviacion_tipica <-
  cbind(
    lapply(
      numeric_df, desviacion_tipica_fn)
  )

desviacion_tipica
```

Primero creamos la función para calcular la desviación típica, tras esto aplicamos la formula a todo el dataframe obteniendo:

```
> desviacion_tipica
              [,1]
peso      14.64644
altura    0.07090472
edad      13.91024
tabaco    41.11936
ubes      5.758399
carneRoja 2.084383
verduras  7.01196
deporte   4.656688
nivEstPad 0.9663223
nivEstudios 1.246906
nivIngresos 1.369964
IMC       4.519736
```

5. Calcula los coeficientes de regresión y el coeficiente de determinación para las 12 regresiones lineales unidimensionales.

```r
#5
#Calculamos la regresion lineal de todas las variables usando  y=IMC
#Creamos funcion para hacerlo de manera mas comoda
regresion_lineal_unidimensional <- function(dt, x, y) {
  rl <- lm(y~x, dt)
  list(coeficiente_de_regresion = coef(rl)[[1]], coeficiente_de_determinacion = summary(rl)$r.squared, regresion_lineal = rl)
}

#Guardamos las regresiones lineales
rlineales <- numeric_df %>% map(regresion_lineal_unidimensional, dt=numeric_df, y=numeric_df$IMC)
rlineales
```

Ahora creamos una función que calcule la regresión lineal unidimensional entre dos variables dado un dataframe y que guarde el coeficiente de regresión, el de determinación y la regresión lineal en una formula.

Tras esto aplicamos esta formula a todo el dataframe y obtenemos:

```
> rlineales
$peso
$peso$coeficiente_de_regresion
[1] 5.322004

$peso$coeficiente_de_determinacion
[1] 0.8007103

$peso$regresion_lineal

Call:
lm(formula = y ~ x, data = dt)

Coefficients:
(Intercept)            x
     5.3220       0.2761
```

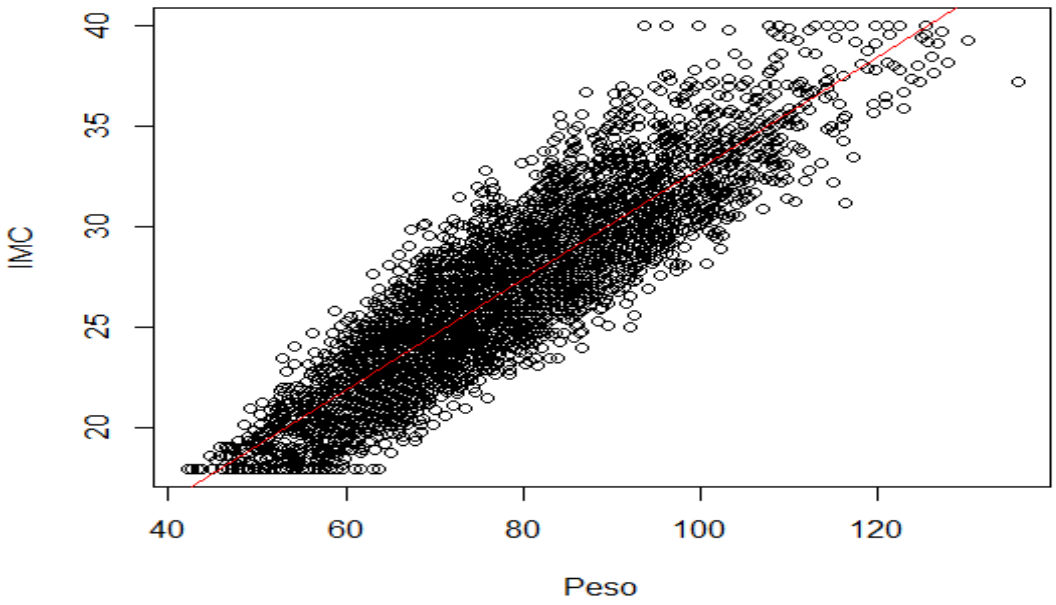
Veríamos algo así para cada columna del dataframe

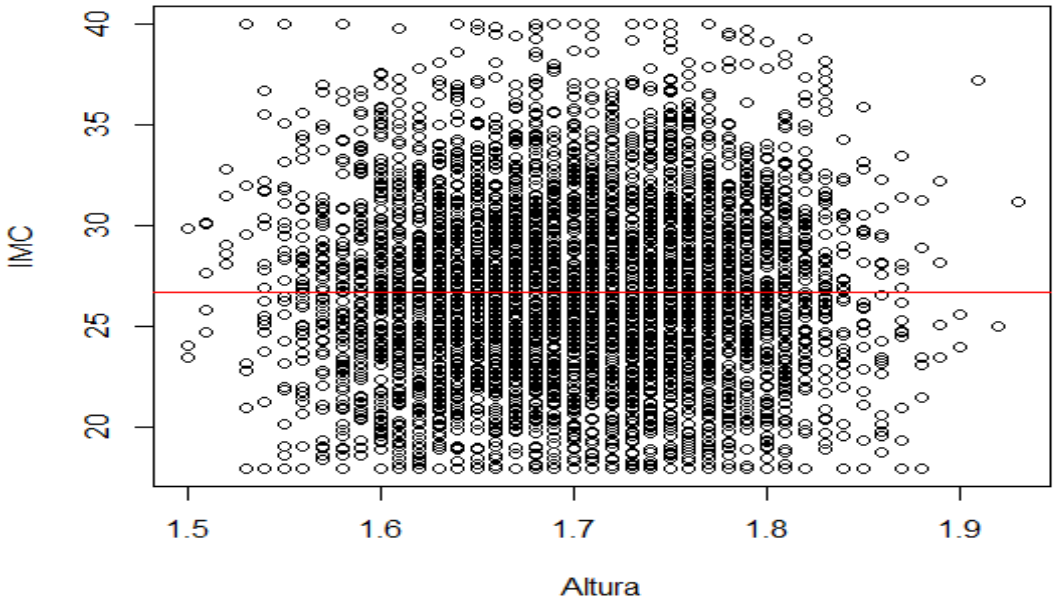
6. Representa los gráficos de dispersión en el caso de variables numéricas y los boxplots en el caso de variables cualitativas. En el caso de las variables numéricas (y sólo en ese caso) el gráfico debe tener sobreimpresa la recta de regresión simple correspondiente.

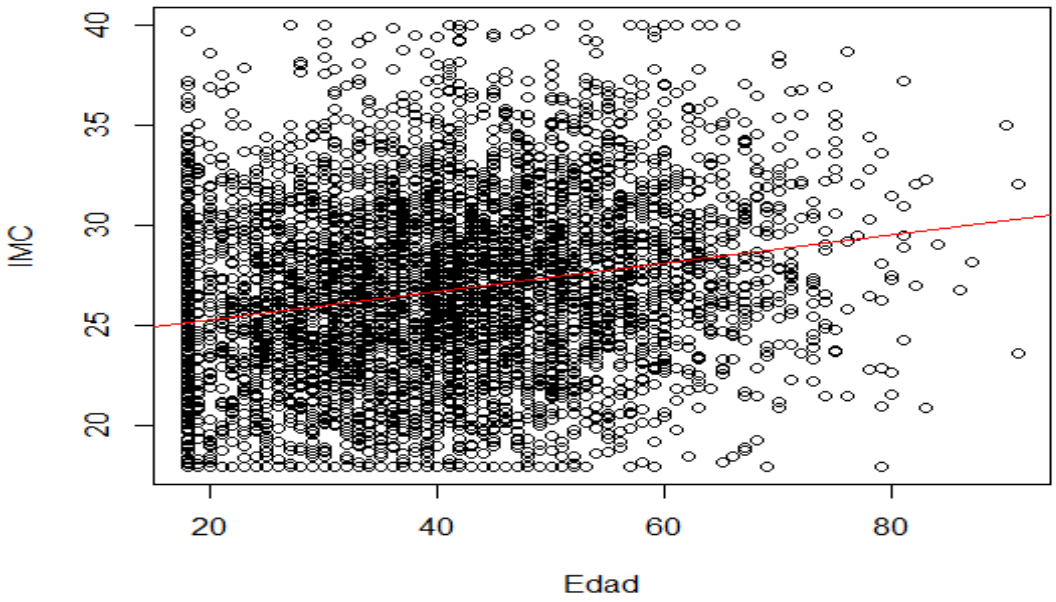
En este caso tenemos que crear gráficos, en el caso de las variables numéricas tendrá la siguiente estructura:

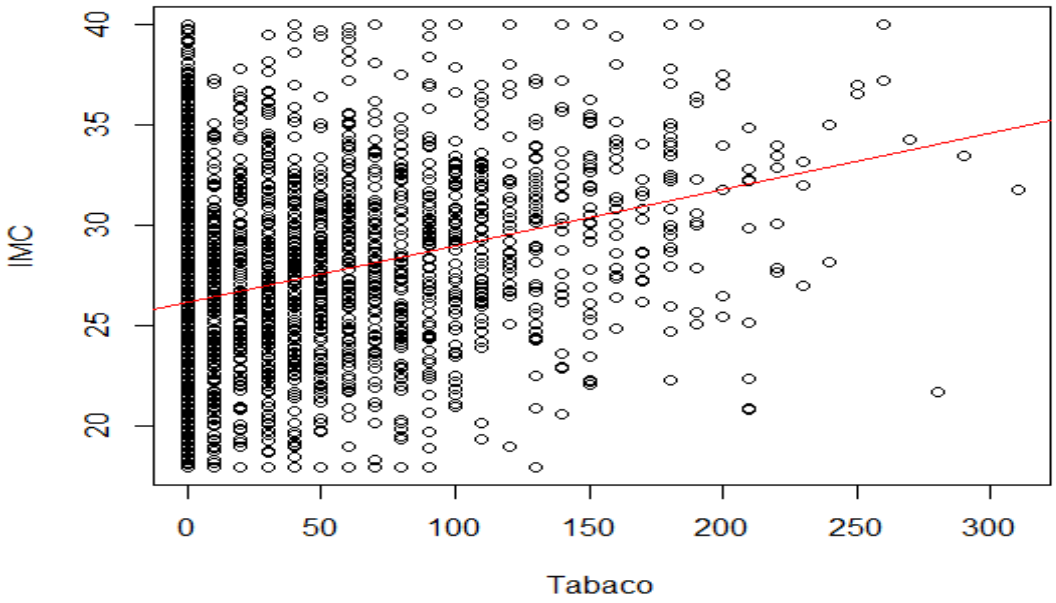
```
#6
#Grafico variables numericas

plot(x=numeric_df$peso, y=numeric_df$IMC, xlab = "Peso", ylab="IMC")
abline(rlineales$peso$regresion_lineal,col="red")
```

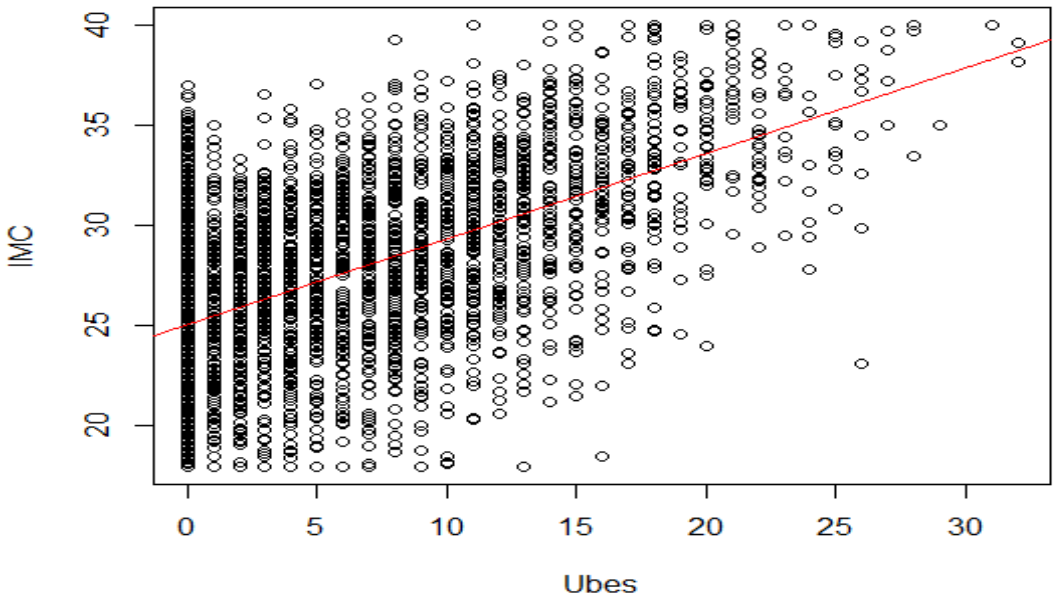
Donde solo ira variando la parte de peso. Las gráficas se verían así:

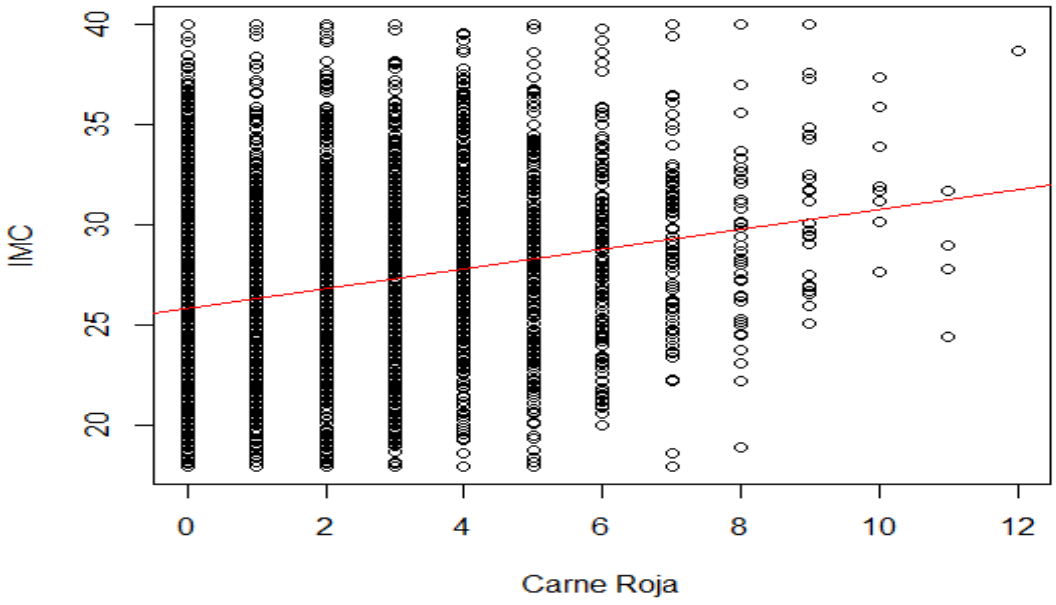

IMC
40
35
30
25
20
40
60
80
100
120
Peso

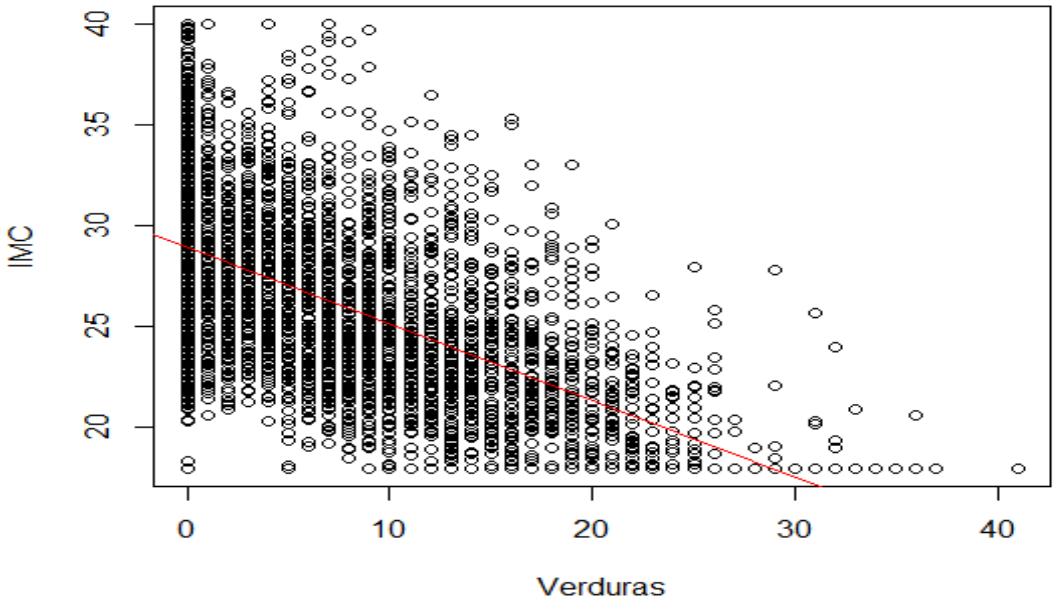

IMC
40
35
30
25
20
1.5
1.6
1.7
1.8
1.9
Altura

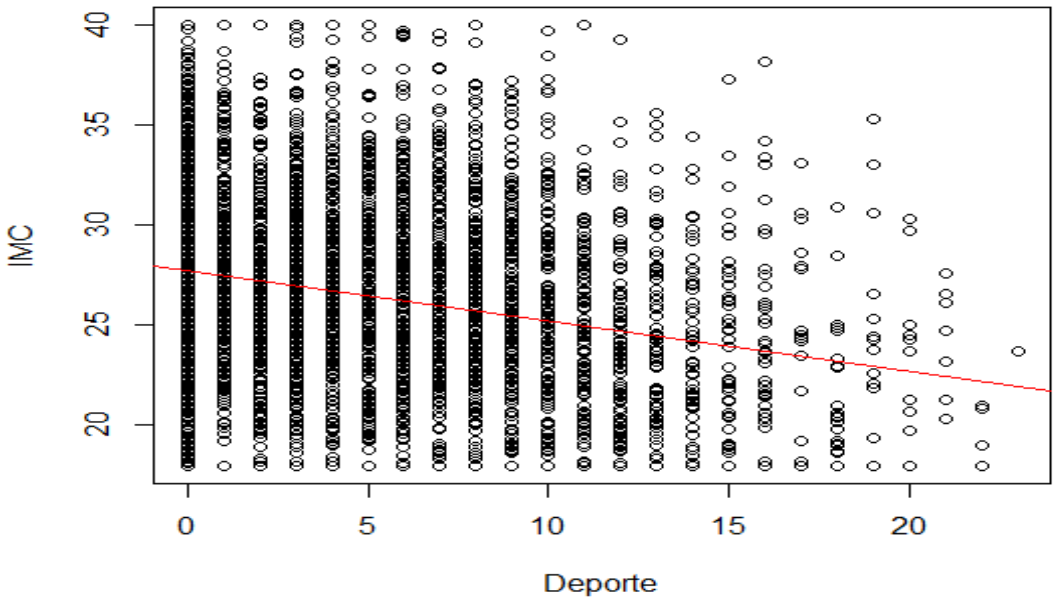

IMC
20
25
30
35
40
20
40
60
80
Edad

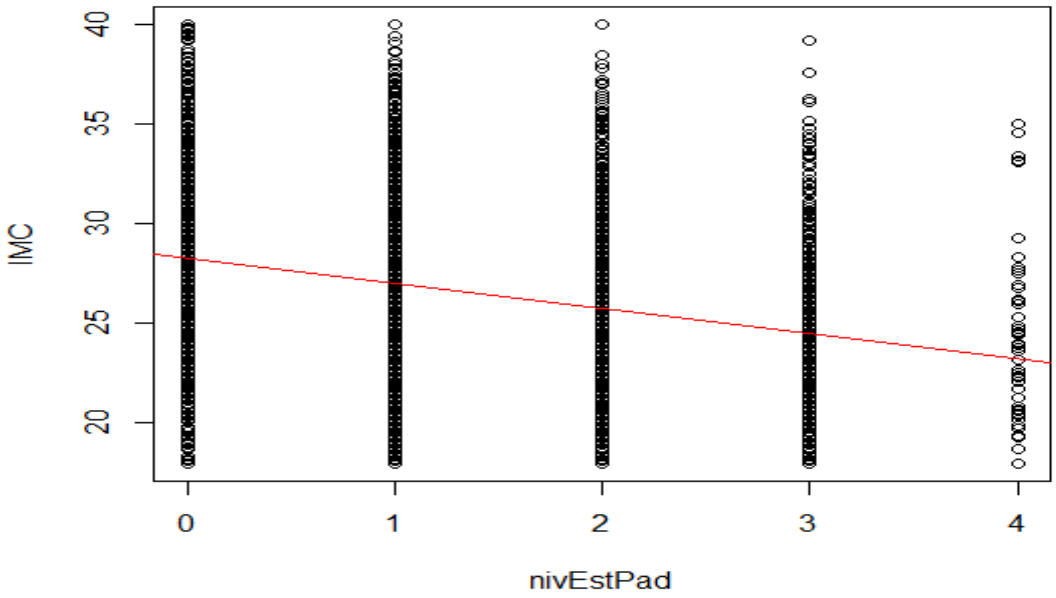

IMC
20
25
30
35
40
0
50
100
150
200
250
300
Tabaco

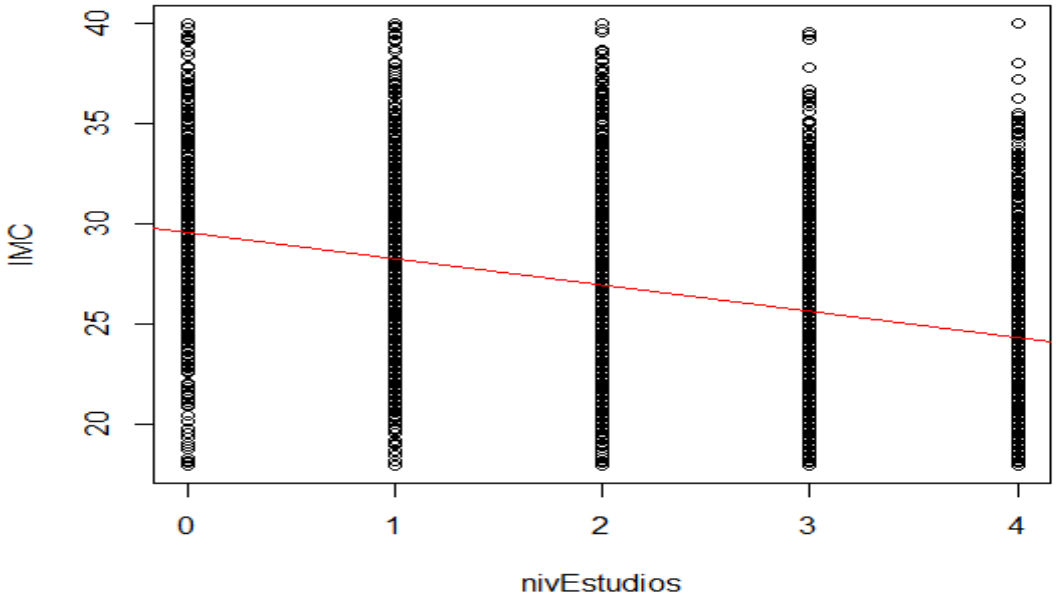

IMC
20
25
30
35
40
0
5
10
15
20
25
30
Ubes

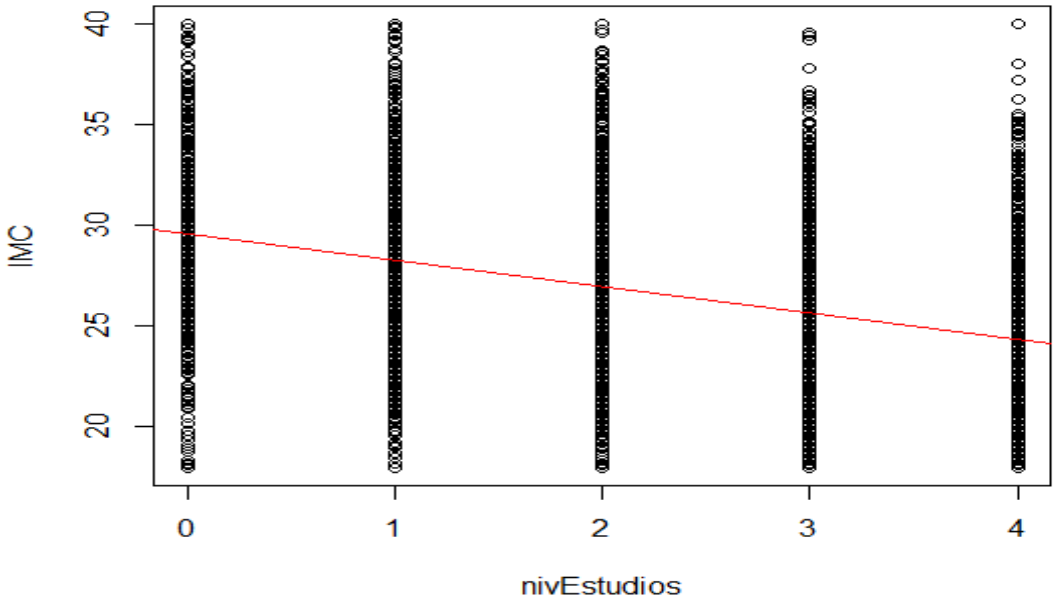

IMC
20
25
30
35
40
0
2
4
6
8
10
12
Carne Roja


IMC
20
25
30
35
40
0
10
20
30
40
Verduras


IMC
20
25
30
35
40
0
5
10
15
20
Deporte


IMC
20
25
30
35
40
0
1
2
3
4
nivEstPad

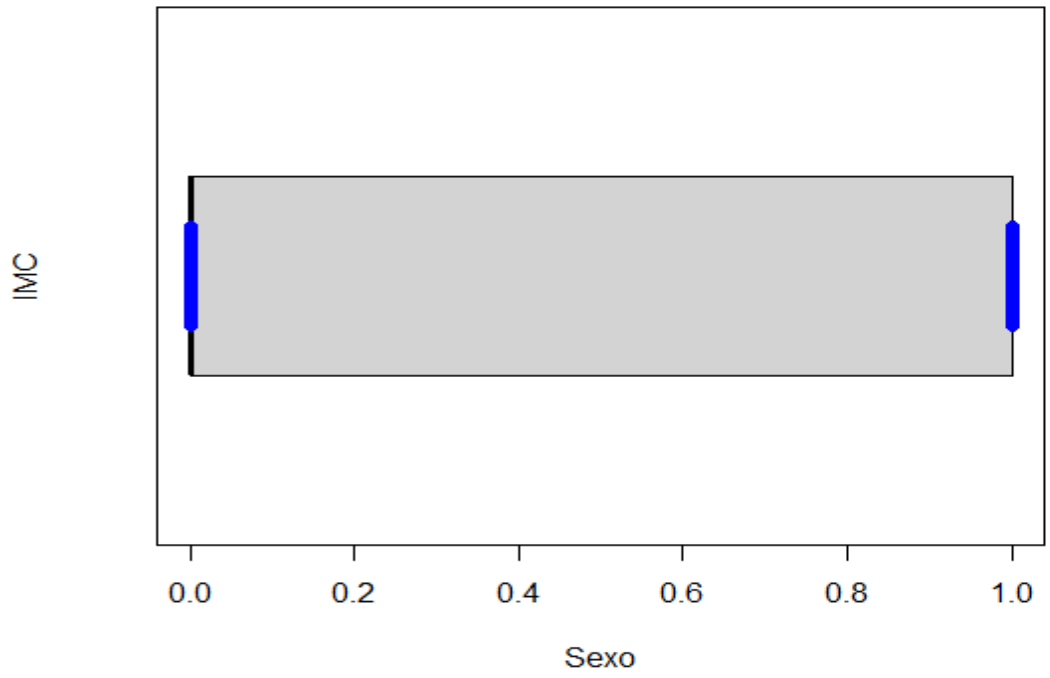

IMC
20
25
30
35
40
0
1
2
3
4
nivEstudios

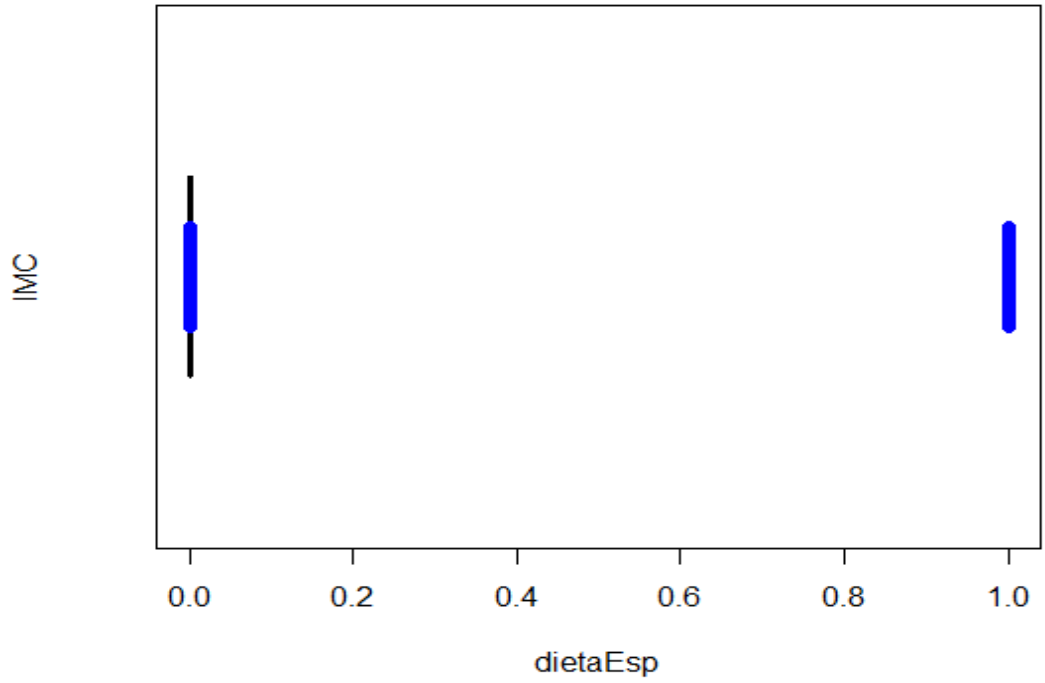

IMC
20
25
30
35
40
0
1
2
3
4
nivEstudios

Para las variables no numéricas tendremos el siguiente código:

```
#Boxplot
#Grafico variables no numericas
boxplot(as.numeric(datos$sexo=="M"), horizontal = TRUE, xlab = "Sexo", ylab="IMC")
stripchart(as.numeric(datos$sexo=="M"), method = "jitter", pch = 19, add = TRUE, col = "blue")


boxplot(as.numeric(datos$dietaEsp=="S"), horizontal = TRUE, xlab = "dietaEsp", ylab="IMC")
stripchart(as.numeric(datos$dietaEsp=="S"), method = "jitter", pch = 19, add = TRUE, col = "blue")
```

Y las gráficas se verían así:





7. Separa el conjunto original de datos en tres conjuntos de entrenamiento, test y validación en las proporciones 60%, 20% y 20%

```r
#7
#Dividimos los datos en 3 partes: Entrenamiento, test y validacion
#Creamos la funcion para que tome samples aleatorias del dataframe
dividirDataSet <- function(data, p1, p2){
  rDf    <- 1:nrow(data)
  rTrain <- sample(rDf, p1 * length(rDf))
  rTemp  <- setdiff(rDf, rTrain)
  rTest  <- sample(rTemp, p2 * length(rTemp))
  rValid <- setdiff(rTemp, rTest)
  return(list(entrenamiento = data[rTrain, ], test=  data[rTest, ], validacion=data[rValid, ]))
}

#Guardamos los dataset
dataSet <- dividirDataSet(datos[-c(1,2)], 0.6, 0.2)
```

Primero creamos una función, la cual dada un dataframe y dos porcentajes, nos separa de manera aleatoria en dataframe en 3 partes:

```
> dataSet
$entrenamiento
# A tibble: 3,000 × 12
   sexo   edad tabaco  ubes carneRoja verduras deporte dietaEsp nivEstPad nivEstudios
   <fct> <dbl>  <dbl> <dbl>     <dbl>    <dbl>   <dbl> <fct>        <dbl>        <dbl>
 1 V        51      0    15         1       18       7 N                0            3
 2 V        26      0    19         0       16      16 N                1            2
 3 V        33     40     0         5       10       0 N                0            1
 4 V        50     50     2         4        0       0 N                2            4
 5 V        63    130     0         0       22      14 N                1            3
 6 M        42      0     0         0        7      13 N                1            2
 7 V        28      0     0         3        0       9 N                2            4
 8 M        27      0     0         0       22      14 N                1            2
 9 V        50    120    13         1        0       1 N                1            1
10 V        36      0     0         1        8       3 S                0            0
# i 2,990 more rows
# i 2 more variables: nivIngresos <dbl>, IMC <dbl>
# i Use `print(n = ...)` to see more rows

$test
```

8. Selecciona cuál de las 12 variables sería la que mejor explica la variable IMC de manera individual, entrenando con el conjunto de entrenamiento y testeando con el conjunto de test.

```r
#8
#Calculamos la regresion lineal de todas las variables con IMC para el conjunto de entrenamiento
modTrain <- dataSet$entrenamiento %>% map(regresion_lineal_unidimensional, dt=dataSet$entrenamiento, y=dataSet$entrenamiento$IMC)

#Creamos una funcion que calcule r2 ajustado
R2ajustado<-function(df,mod, y){
  MSE  <- mean((y - predict.lm(mod[["regresion_lineal"]], df)) ^ 2)
  varY <- mean(y ^ 2) - mean(y) ^ 2
  print(MSE/varY)
  R2   <- 1 - (MSE / varY)
  aR2  <- 1 - (1- R2) * (nrow(df) - 1) / (nrow(df) - mod$regresion_lineal$rank)
  aR2
}
#Aplicamos la funcion de r2 ajustado a todo el modelo creado con el conjunto de entrenamiento
#Le pasamos como parametros todas las variables excepto IMC ya que no tiene sentiendo, lo estamos entrenando para predecir IMC
bestFit <-modTrain[-which(names(modTrain) == "IMC")]%>% map_dbl(R2ajustado, df=dataSet$test, y=dataSet$test$IMC)
bestFit

#Mostramos que variable es la que mejor predice IMC
which.max(bestFit)[1]
```

Para este apartado, primero calculamos las regresiones lineales del dataSet de entrenamiento.

Tras esto creamos una función para calcular R2 ajustado tomando como parámetros un dataframe, un modelo linear y la variable a predecir.

Por último, con el modelo creado con el dataSet de entrenamiento encontramos que variable predice mejor la variable “IMC”. Para esto, le aplicamos a todas las regresiones lineales calculadas antes la función “R2ajustado” y vemos cual es el máximo:

```
> bestFit
      sexo        edad      tabaco        ubes    carneRoja     verduras      deporte     dietaEsp    nivEstPad  nivEstudios  nivIngresos
-0.004539517 -0.050523188 -0.073824757 -0.262324408 -0.055343530 -0.323986257 -0.080071533 -0.004320117 -0.100463045 -0.168418282 -0.133756725
> #Mostramos que variable es la que mejor predice IMC
> which.max(bestFit)[1]
dietaEsp
       8
```

El que nos de un R2 ajustado, será el que mejor prediga IMC para regresiones lineales simples. En este caso diestaEsp.

9. Selecciona un modelo óptimo lineal de regresión, entrenando en el conjunto de entrenamiento, testeando en el conjunto de test el coeficiente de determinación ajustado y utilizando una técnica progresiva de ir añadiendo la mejor variable.

```r
#9
#Creamos una funcion para hacer calculos de regresiones lineales multiples
regresion_lineal_multiple <- function(dt, x, y) {
  rl<-lm(str_c(y, "~", str_c(x, collapse="+")), dt)
  list(coeficiente_de_regresion = coef(rl)[[1]], coeficiente_de_determinacion = summary(rl)$r.squared, regresion_lineal = rl)
}

#Creamos una funcion que haga r2 ajustado directamente
calcModR2 <- function(dfTrain, dfTest, varPre,y, x) {
  mod <- regresion_lineal_multiple(dfTrain, y, x)
  R2ajustado(dfTest, mod, varPre)
}
```

Primero creamos una función que acepte regresiones lineales múltiples, de esta manera podemos pasarle una lista de variables y que las concatene para hacerlo.

Después una función a la que le pasaremos el conjunto de entrenamiento, el de test, la variable a predecir y las variables predictoras que calculara su regresión lineal y R2ajustado.

```r
#Creamos una funcion que calcule que variable es mejor
encontrarMejorAjuste <- function(dfTrain, dfTest, varPos) {
  bestVars <- varPos[1]
  aR2      <- 0
  #Repetimos en bucle
  #Añadimos cada vez una variable nueva para calcular R2 hasta que encontremos la mejor variable
  repeat {
    aR2v <- map_dbl(varPos, ~calcModR2(dfTrain, dfTest, dfTest$IMC,"IMC", c(bestVars, .)))
    i    <- which.max(aR2v)
    aR2M <- aR2v[i]
    if (aR2M <= aR2) break

    cat(sprintf("%1.4f %s\n", aR2M, varPos[i]))
    aR2 <- aR2M
    bestVars <- c(bestVars, varPos[i])
    varPos   <- varPos[-i]
  }
  #Guardamos en una lista
  mod <- regresion_lineal_multiple(dt = dfTrain, y="IMC", x=bestVars)

  list(vars=bestVars, mod=mod)
}
```

Ahora creamos una función a la que le pasaremos el conjunto de entrenamiento, el de test y el nombre de las variables predictoras. Esta ira haciendo bucles, añadiendo cada vez mas variables a la regresión lineal múltiple para encontrar el mejor R2 ajustado y ver que variable predice mejor IMC.

```r
#Vemos que variable predice mejor IMC
bestMod1 <- encontrarMejorAjuste(dataSet$entrenamiento, dataSet$test, names(datos[-c(1,2,14)]))
bestMod1$vars
```

Por último, aplicamos esta función para ver cuál es la variable:

```
> bestMod1$vars
[1] "sexo"
```

10.Evalúa el resultado en el conjunto de validación.

```r
#10
#Comprobamos como de bien lo hace nuestro modelo
R2ajustado(df=dataSet$validacion, mod=bestMod1$mod, y=dataSet$validacion$IMC)
```

Para este apartado vamos a comprobar lo bueno que es nuestro modelo, para ello llamamos a la función de R2ajustado con dataset de validación y sus valores de IMC pero usando como modelo el que acabamos de calcular:

```
> R2ajustado(df=dataSet$validacion, mod=bestMod1$mod, y=dataSet$validacion$IMC)
[1] 1.001328
[1] -0.001954408
```

Como vemos, R2 ajustado sigue saliendo negativo aunque sigue siendo mejor que el calculado en el apartado 8.

11. Lee el dataframe de evaluación que te habrá llegado (eval.csv) y utiliza el modelo creado para añadirle una nueva columna con el valor de la variable IMC y, a continuación, otra columna con el valor de la variable Peso. Salva el resultado como evalX.csv para enviarlo como parte de la solución al trabajo.

```r
#11
#Cargamos el archivo de evaluacion
eval <- read_csv("./eval.csv", col_types = cols(.default = col_double(), sexo = col_factor(), dietaEsp = col_factor()))
#Predecimos la columna IMC con nuestro modelo
eval$IMC <- predict.lm(bestMod1$mod$regresion_lineal, eval)
eval
#Y calculamos la columna peso con los datos predichos
eval <- add_column(eval, peso = eval$IMC*(eval$altura^2))
eval
#Guardamos el dataframe en evalX.csv
write.csv(eval, "./evalX.csv", row.names=TRUE)
```

Por último cargamos el archivo “eval.csv”. Tras esto predecimos la columna IMC para los datos de “eval.csv” usando el modelo que acabamos de calcular:

```
> eval
# A tibble: 1,000 x 15
   sexo  altura  edad tabaco  ubes carneRoja verduras deporte drogas dietaEsp nivEstPad nivEstudios nivIngresos   IMC  peso
   <fct>  <dbl> <dbl>  <dbl> <dbl>     <dbl>    <dbl>   <dbl>  <dbl> <fct>        <dbl>       <dbl>       <dbl> <dbl> <dbl>
 1 M       1.63    18      0     0         0        5       0      2 S                0           1           2  26.7  71.0
 2 V       1.68    35      0     1         1        6       0      0 N                1           3           2  26.6  75.2
 3 V       1.77    39     30    13         0        0       6      0 N                2           3           1  26.6  83.4
 4 M       1.59    57     60     4         4       16      15      0 N                2           2           2  26.7  67.6
 5 M       1.58    18      0     3         1        0       7      0 N                2           4           4  26.7  66.7
 6 M       1.66    35    160     0         0        0       4      0 N                0           2           3  26.7  73.6
 7 V       1.75    46    140     3         1        0       0      0 S                0           1           0  26.6  81.5
 8 M       1.73    54      0     0         0        9       4      0 N                0           0           1  26.7  80.0
 9 V       1.67    18      0     0         0       11       0      0 S                1           2           3  26.6  74.3
10 V       1.74    53      0     3         1        4       0      0 N                1           2           1  26.6  80.6
```

Que como podemos ver, son unos valores que parecen correctos.

Ahora toca calcular peso con los valores IMC predichos:

```
> eval
# A tibble: 1,000 x 15
   sexo  altura  edad tabaco  ubes carneRoja verduras deporte drogas dietaEsp nivEstPad nivEstudios nivIngresos   IMC  peso
   <fct>  <dbl> <dbl>  <dbl> <dbl>     <dbl>    <dbl>   <dbl>  <dbl> <fct>        <dbl>       <dbl>       <dbl> <dbl> <dbl>
 1 M       1.63    18      0     0         0        5       0      2 S                0           1           2  26.7  71.0
 2 V       1.68    35      0     1         1        6       0      0 N                1           3           2  26.6  75.2
 3 V       1.77    39     30    13         0        0       6      0 N                2           3           1  26.6  83.4
 4 M       1.59    57     60     4         4       16      15      0 N                2           2           2  26.7  67.6
 5 M       1.58    18      0     3         1        0       7      0 N                2           4           4  26.7  66.7
 6 M       1.66    35    160     0         0        0       4      0 N                0           2           3  26.7  73.6
 7 V       1.75    46    140     3         1        0       0      0 S                0           1           0  26.6  81.5
 8 M       1.73    54      0     0         0        9       4      0 N                0           0           1  26.7  80.0
 9 V       1.67    18      0     0         0       11       0      0 S                1           2           3  26.6  74.3
10 V       1.74    53      0     3         1        4       0      0 N                1           2           1  26.6  80.6
```

Que también se ven datos de peso correctos.

Por último, guardaríamos estos datos en un archivo CSV “evalX.csv”.

12.Expresa tus conclusiones sobre el modelo creado. Incluyendo, al menos, respuestas a las siguientes cuestiones:

- Que utilidad podría tener el modelo matemático que has obtenido.
- Que se puede deducir a partir del modelo sobre la relación entre las variables.
- Problemas que has encontrado en el desarrollo.
- Qué te ha llamado la atención en el proceso.
- Qué más podría hacerse y cómo plantearlo.

Hemos podido ver que en la relación entre variables para este caso es bastante pobre ya que al calcular R2 ajustado salían siempre números negativos, por lo que el modelo no seria algo apto para hacer uso del más allá de simplemente ver como crear un modelo y predecir datos.

El único problema encontrado es que al salir los R2 ajustados negativos pensaba que había hecho algo mal, tras repasarlo más, me di cuenta de que simplemente los datos no eran especialmente buenos para predecir.

Lo único que se me ocurre que se podría hacer es conseguir o mas datos o mejores datos para mejorar el modelo.