Trabajo R

Miguel Ángel Dorado Maldonado

Se cargan las librerias necesarias para poder trabajar y se añade el directorio con los archivos como ruta principal del trabajo

```
library(tidyverse)
library(readr)
library(purrr)

# Añado mi directorio como ruta del trabajo
setwd("~/Universidad/Estadistica/Trabajo R")
```

```
> library(tidyverse)
> library(readr)
> library(purrr)
> # Añado mi directorio como ruta del trabajo
> setwd("~/Universidad/Estadistica/Trabajo R")
>
```

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

Haciendo uso de la librería readr se importa el archivo .csv como tibble. Para que las variables cualitativas sean leidas como factores se modifica el parametro col_types como se se indica sobre las variables correspondientes.

```
data <- read_csv("21425.csv", col_types=

cols(.default=col_double(),

sexo=col_factor(),

dietaEsp=col_factor(),

nivEstPad=col_factor(),

nivEstudios=col_factor(),

nivIngresos=col_factor())
)
```

```
> data <- read_csv("21425.csv", col_types=cols(.default=col_double(), sexo=col_factor(),
+                                              dietaEsp=col_factor(), nivEstPad=col_factor(),
+                                              nivEstudios=col_factor(), nivIngresos=col_factor()))

>
```

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.

Se accede a la columna IMC para modificarla. De esta forma si no existe se crea. Se asigna a cada fila el peso partido por la altura al cuadrado.

```
data$IMC <- data$peso/data$altura^2
```

```
> data$IMC <- data$peso/data$altura^2
>
```

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

Se eliminan todas las filas que contengan algún NA.

```
data <- na.omit(data)
```

```
> data <- na.omit(data)
>
```

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

Se crea un nuevo dataframe con las columnas numéricas filtrando con el método keep.
Para cada una de las posiciones de las columnas del dataframe se le calcula la media usando map_dbl.
De igual forma se calcula las desviaciones típicas. %>% te permite pasar parámetros a una función. (Función lambda).

```
dfNumerico <- keep(data,is.numeric)
medias <- map_dbl(dfNumerico, mean)
medias

desvTipicas <- dfNumerico %>%
  summarise_all(function(x) sqrt(mean(x^2) - mean(x)^2)) %>%
  map_dbl(function(x) x)
desvTipicas
```

```
> dfNumerico <- keep(data,is.numeric)
> medias <- map_dbl(dfNumerico, mean)
> medias
      peso     altura       edad     tabaco       ubes  carneRoja   verduras    deporte     drogas
73.7551000  1.7011594 40.7915573 20.7372248  4.0874571  1.7725712  6.0197940  4.1696627  0.5209049
       IMC
25.4362977
> desvTipicas <- dfNumerico %>%
+   summarise_all(function(x) sqrt(mean(x^2) - mean(x)^2)) %>%
+   map_dbl(function(x) x)
> desvTipicas
       peso      altura        edad      tabaco        ubes   carneRoja    verduras     deporte
16.46931617  0.07105482 14.29901759 41.35214413  5.90394962  2.12533530  7.11770305  4.75736220
     drogas         IMC
 1.49297056  5.21927270
>
```

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

Se cogen las variables predictorias(todas excluyendo peso, altura y el propio imc).
Se guardan en un vector los nombres de todas estas.

Se crea una función que dado un dataframe y dos variables del propio, genera un modelo y devuelve los coeficientes de regresión.
De igual forma se crea una función que devuelve el R2.

Se guardan los coeficientes de regresión y de determinación en dos variables por medio del método map_dbl donde para cada variable predictoria calcula su respectiva función.
"[[.]]" hace referencia al parametro de varPredictoria actual.

```
varPredictorias <- names(data[3:14])
varPredictorias

coefRegresion <- function(df,y,x){
  modelo <- lm(y ~ x, df)
  summary(modelo)$coefficients[2]
}

calcR2 <- function(df,y,x){
  modelo <- lm(y ~ x, df)
  summary(modelo)$r.squared
}

coeficientes <- map_dbl(varPredictorias, ~ coefRegresion(data,data$IMC,
data[[.]]))
coeficientes

valoresR2 <- map_dbl(varPredictorias, ~ calcR2(data,data$IMC, data[[.]]))
valoresR2
```

```
> varPredictorias <- names(data[3:14])
> varPredictorias
 [1] "sexo"        "edad"        "tabaco"      "ubes"        "carneRoja"   "verduras"    "deporte"
 [8] "drogas"      "dietaEsp"    "nivEstPad"   "nivEstudios" "nivIngresos"
> coefRegresion <- function(df,y,x){
+   modelo <- lm(y ~ x, df)
+   summary(modelo)$coefficients[2]
+ }
> calcR2 <- function(df,y,x){
+   modelo <- lm(y ~ x, df)
+   summary(modelo)$r.squared
+ }
> coeficientes <- map_dbl(varPredictorias, ~ coefRegresion(data,data$IMC, data[[.]]))
> coeficientes
 [1]  0.04473659  0.02910956 -0.09010006  0.09266687 -0.17296197 -0.01444383  0.15592474 -0.57331969
 [9]  0.33112131  1.26180353  0.16020597 -0.35658082
> valoresR2 <- map_dbl(varPredictorias, ~ calcR2(data,data$IMC, data[[.]]))
> valoresR2
 [1] 1.835485e-05 6.360095e-03 5.095963e-01 1.098789e-02 4.960623e-03 3.879936e-04 2.019959e-02
 [8] 2.689529e-02 1.723815e-04 4.825639e-03 2.577369e-04 1.179164e-03
>
```

Al analizar los valores obtenidos podemos observar como el R2 de una de las variables es de aproximadamente 0.5, mientras que el resto están muy próximas al 0. Esto quiere decir que esta variable, en este caso tabaco, genera un modelo capaz de explicar el 50% de los IMC. Es decir, el modelo se ajusta mucho más que el resto de variables.

6. Representa los gráficos de dispersión en el caso de variables numéricas y los box-plots 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.

Se crea una función del ajuste lineal que devuelve en una lista la variable y, la variable x y el modelo. Para calcular el modelo se usa el método lm. En este caso he utilizado str_c para concatenar los valores ya que de otra forma me daba problemas.

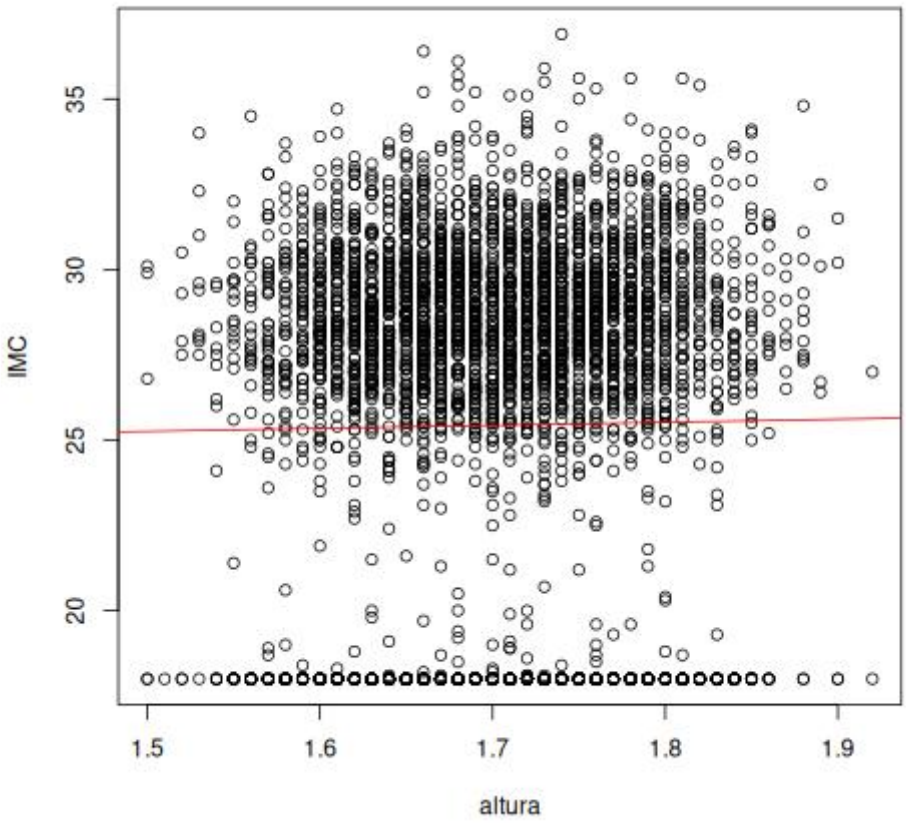
Se crea una función para generar la representación gráfica de la variable con respecto al IMC dado un modelo. En caso de ser una variable numérica se creará un plot y en caso de no serlo se creará un boxplot. Abline te permite dibujar la linea que representa la recta de la regresión sobre el gráfico.

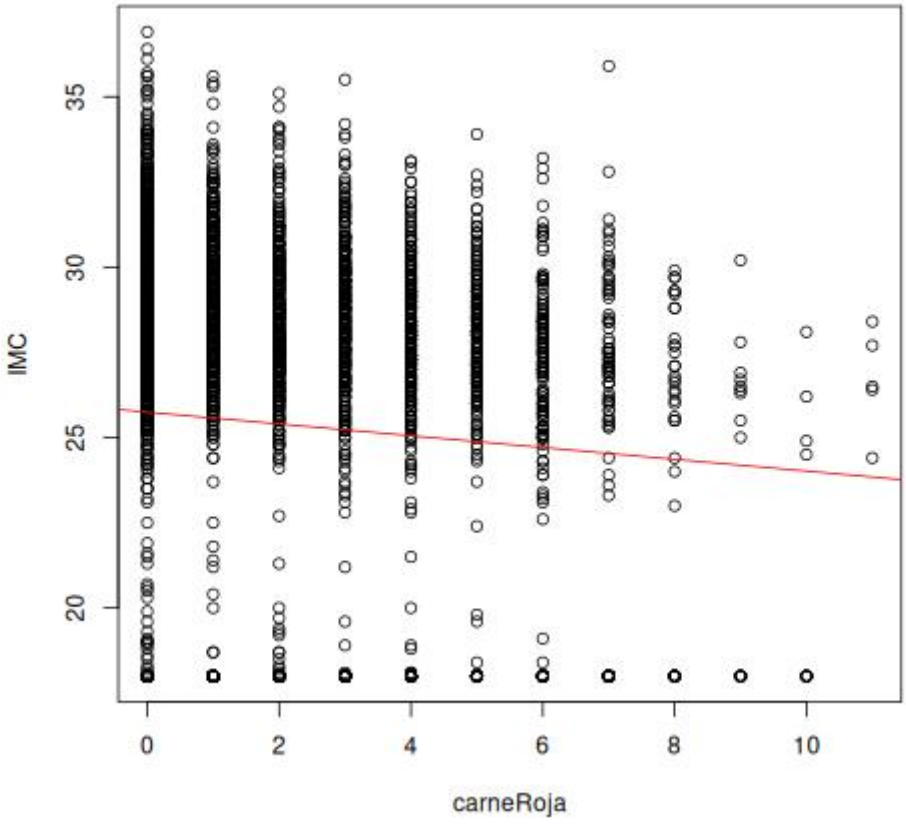
Una vez definidas las funciones se generan los modelos de cada una de las variables y se guarda en un vector (mods). A continuación se crea la carpeta donde se guardarán todas las imágenes. Mediante el método walk se generan las imagenes.

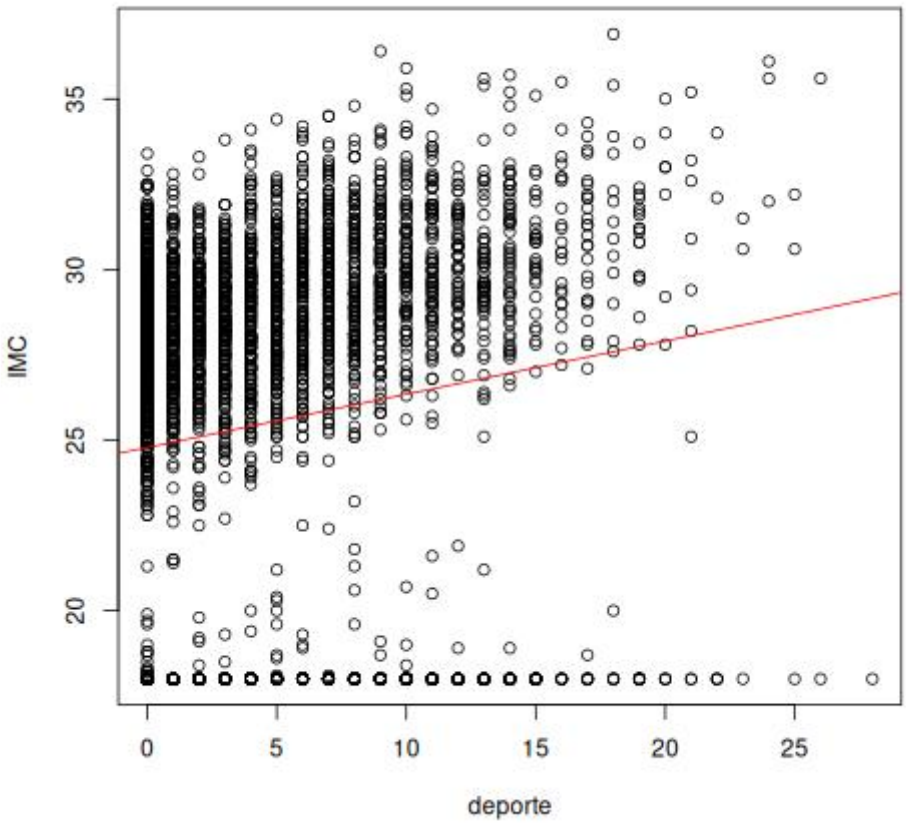
```r
linearAdjust <- function(df, y, x) {
  list(x=x, y=y, mod=lm(str_c(y, "~", x), df))
}

dibujarModelos <- function(mod) {
  jpeg(str_c("./Imagenes/", mod$x, ".jpeg"))

  if (is.numeric(data[[mod$x]])) {
    plot(data[[mod$x]], data[[mod$y]], xlab=mod$x, ylab=mod$y)
    abline(mod$mod, col="red")
  }else{
    boxplot(formula=data[[mod$y]] ~ data[[mod$x]], xlab=mod$x, ylab=mod$y)
```

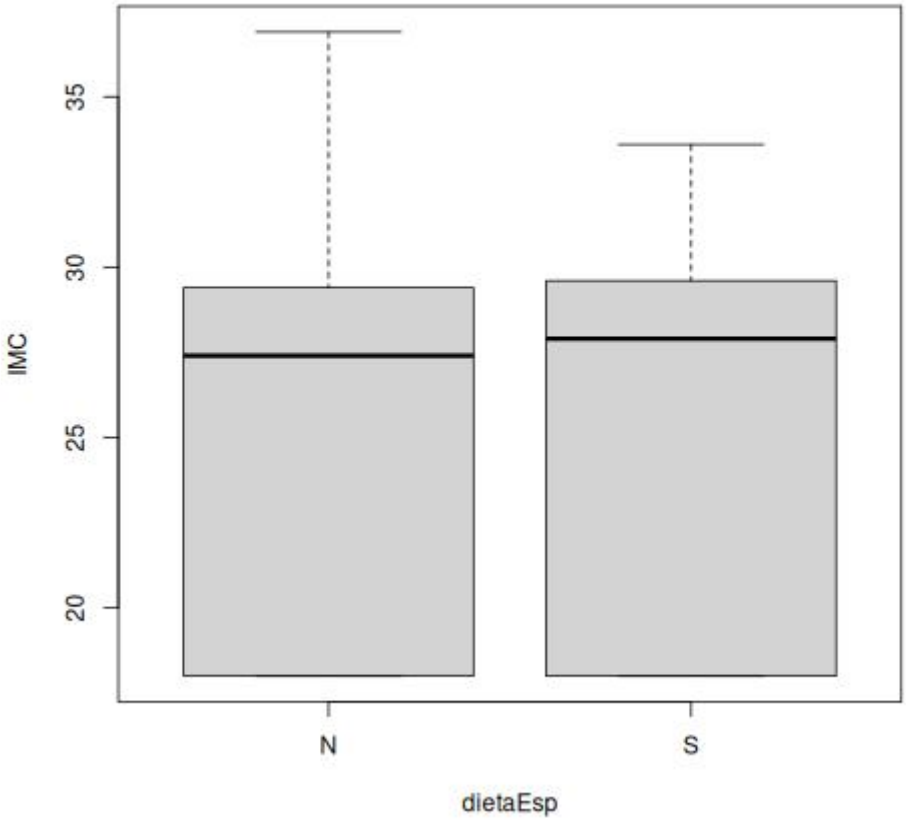
```
  }

  dev.off()
}

mods <- names(data) %>% map(~linearAdjust(data, "IMC", .))
dir.create("Imagenes")
mods %>% walk(dibujarModelos)
```

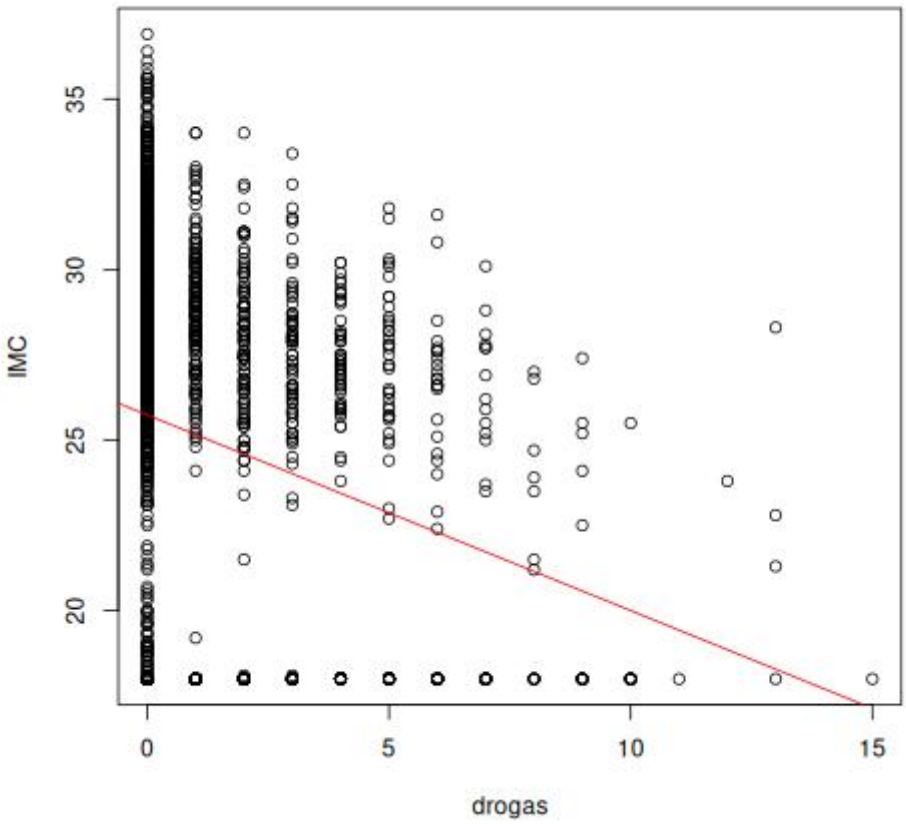
```
> linearAdjust <- function(df, y, x) {
+   list(x=x, y=y, mod=lm(str_c(y, "~", x), df))
+ }
> dibujarModelos <- function(mod) {
+   jpeg(str_c("./Imagenes/", mod$x, ".jpeg"))
+
+   if (is.numeric(data[[mod$x]])) {
+     plot(data[[mod$x]], data[[mod$y]], xlab=mod$x, ylab=mod$y)
+     abline(mod$mod, col="red")
+   }else{
+     boxplot(formula=data[[mod$y]] ~ data[[mod$x]], xlab=mod$x, ylab=mod$y)
+   }
+
+   dev.off()
+ }
> mods <- names(data) %>% map(~linearAdjust(data, "IMC", .))
Warning messages:
1: In model.matrix.default(mt, mf, contrasts) :
  the response appeared on the right-hand side and was dropped
2: In model.matrix.default(mt, mf, contrasts) :
  problem with term 1 in model.matrix: no columns are assigned
> # Utilizando el metodo walk para generar los graficos. Creo la carpeta imagenes
> # en caso de no estarlo.
> dir.create("Imagenes")
> mods %>% walk(dibujarModelos)
>
```

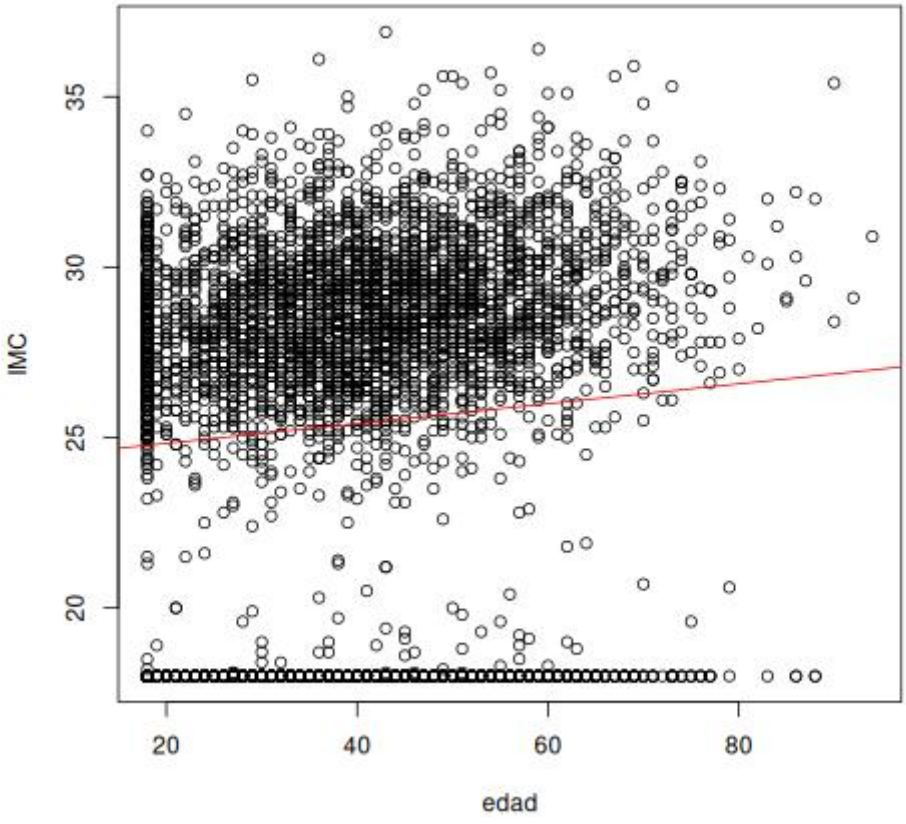

IMC
20
25
30
35
1.5
1.6
1.7
1.8
1.9
altura

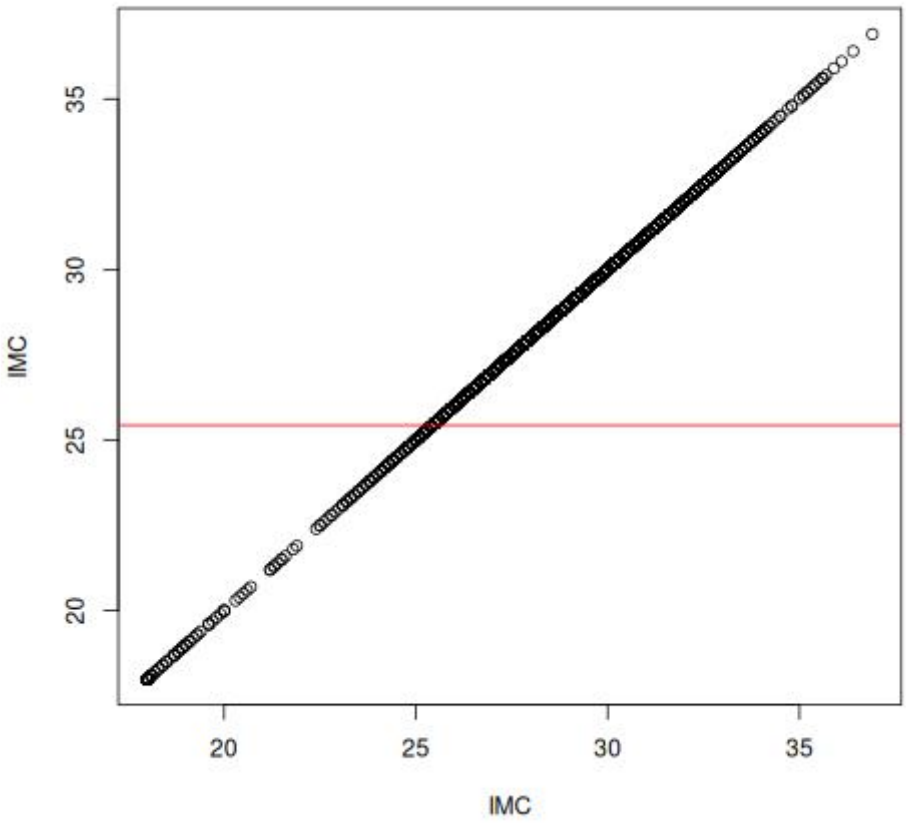

IMC
20
25
30
35
0
2
4
6
8
10
carneRoja

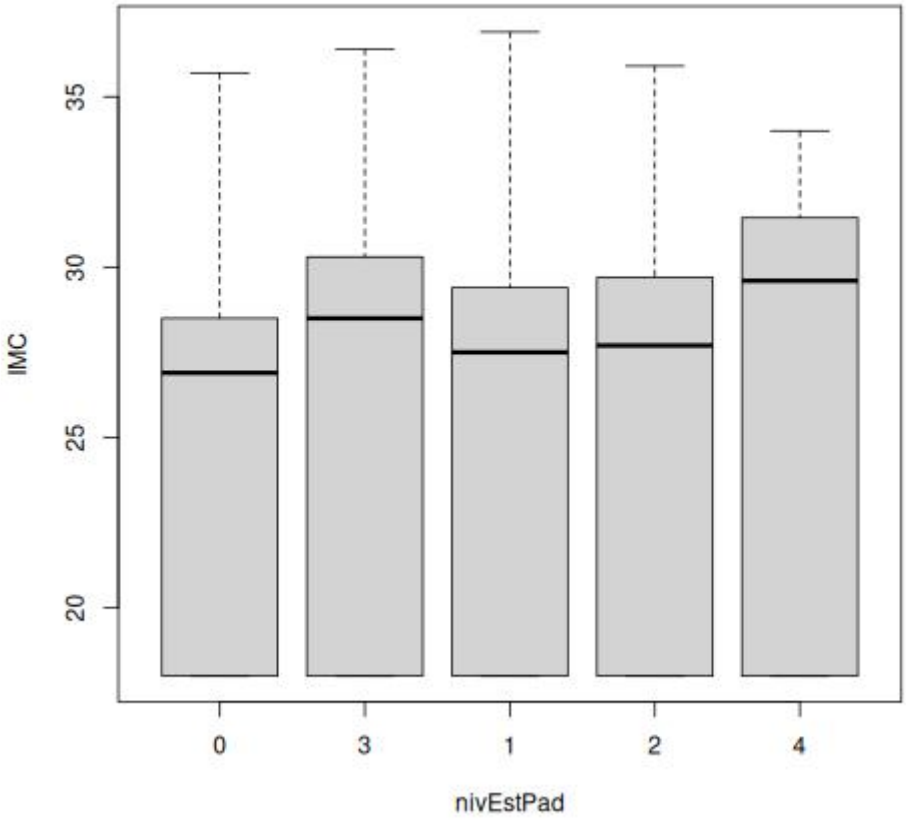

IMC
35
30
25
20
0
5
10
15
20
25
deporte

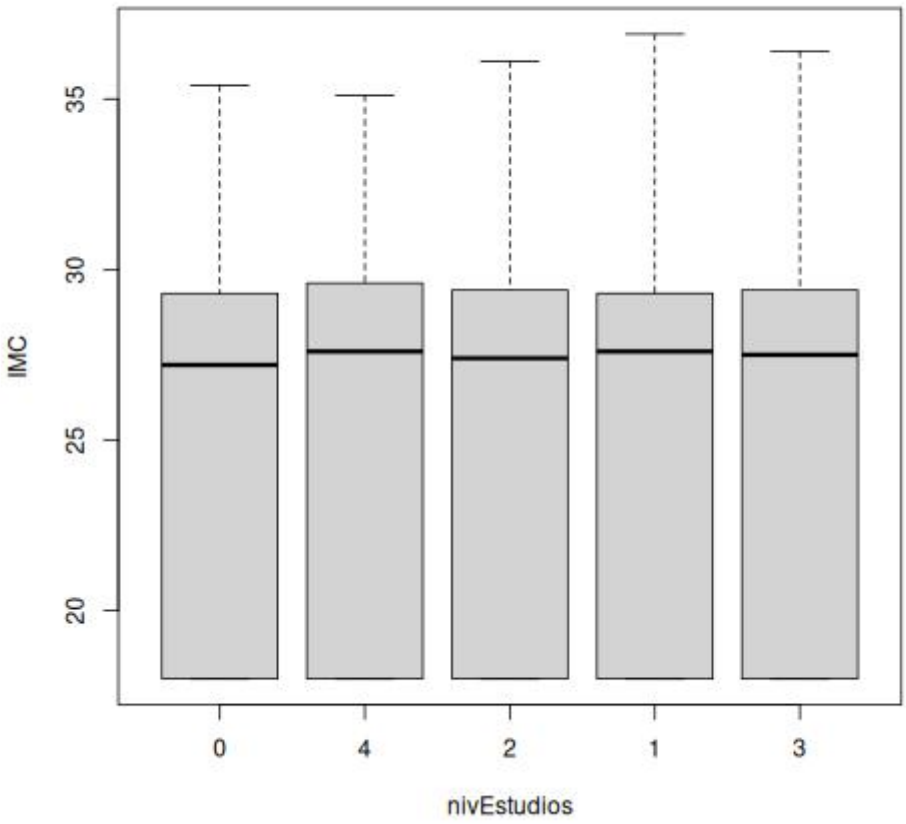

IMC
35
30
25
20
N
S
dietaEsp

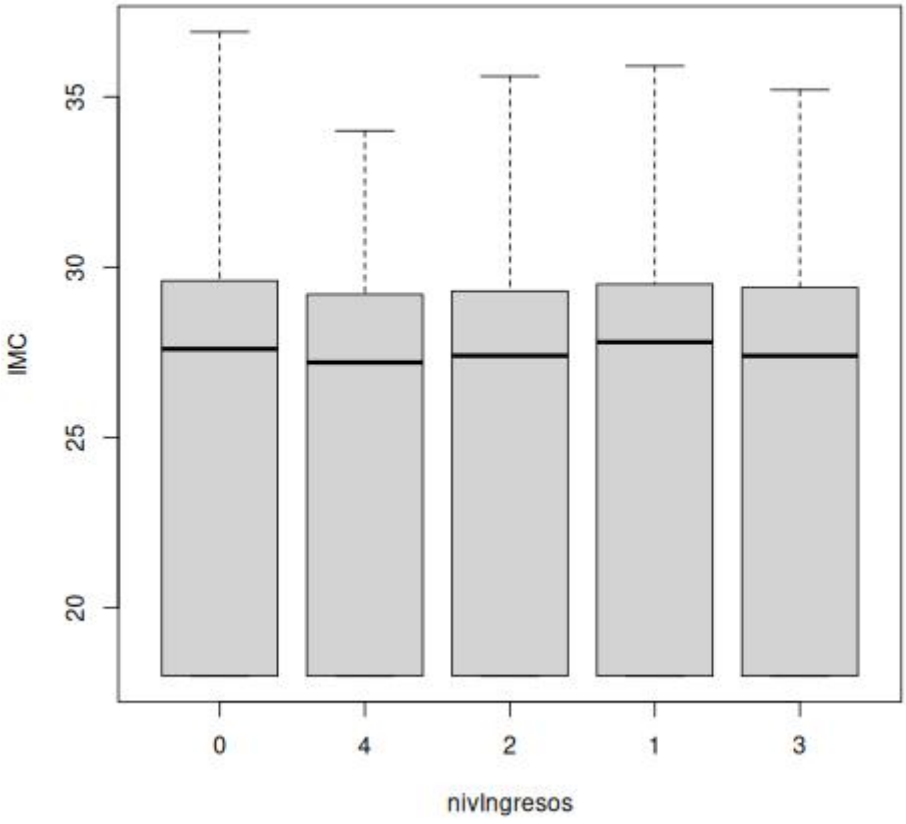

IMC
20
25
30
35
0
5
10
15
drogas

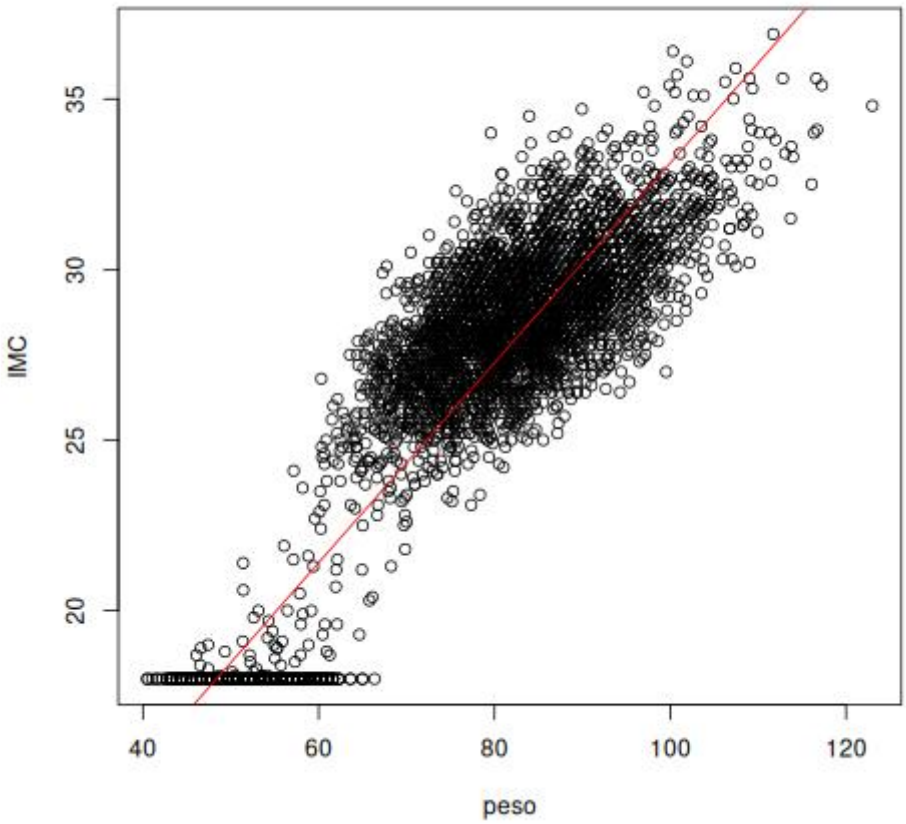

IMC
20
25
30
35
20
40
60
80
edad

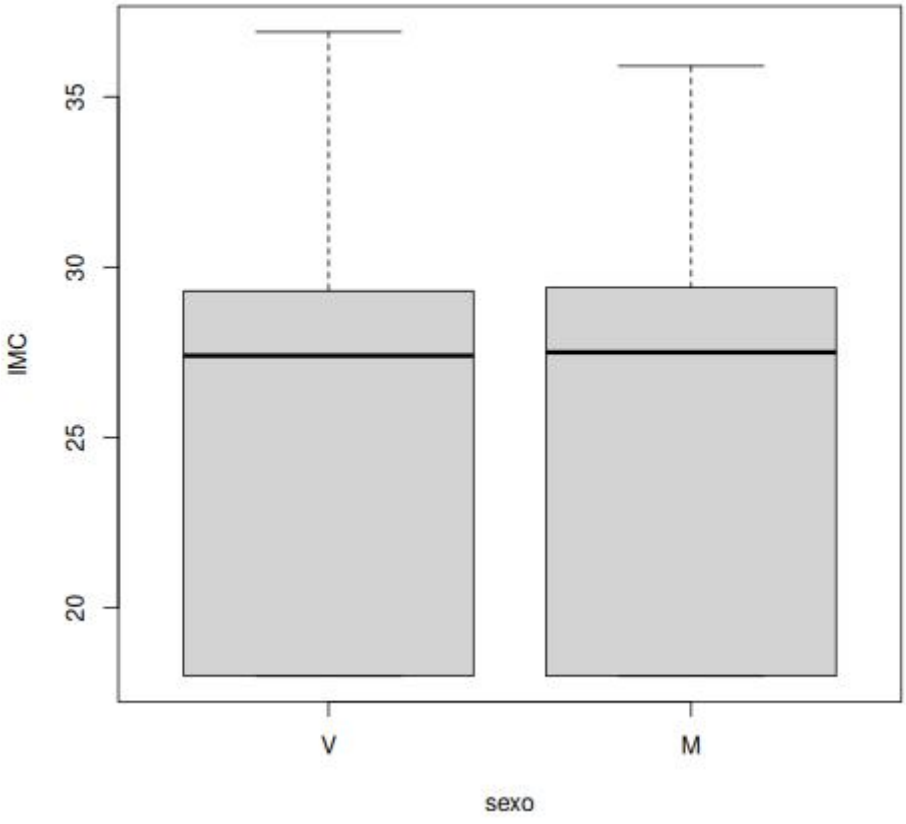

IMC
20
25
30
35
IMC

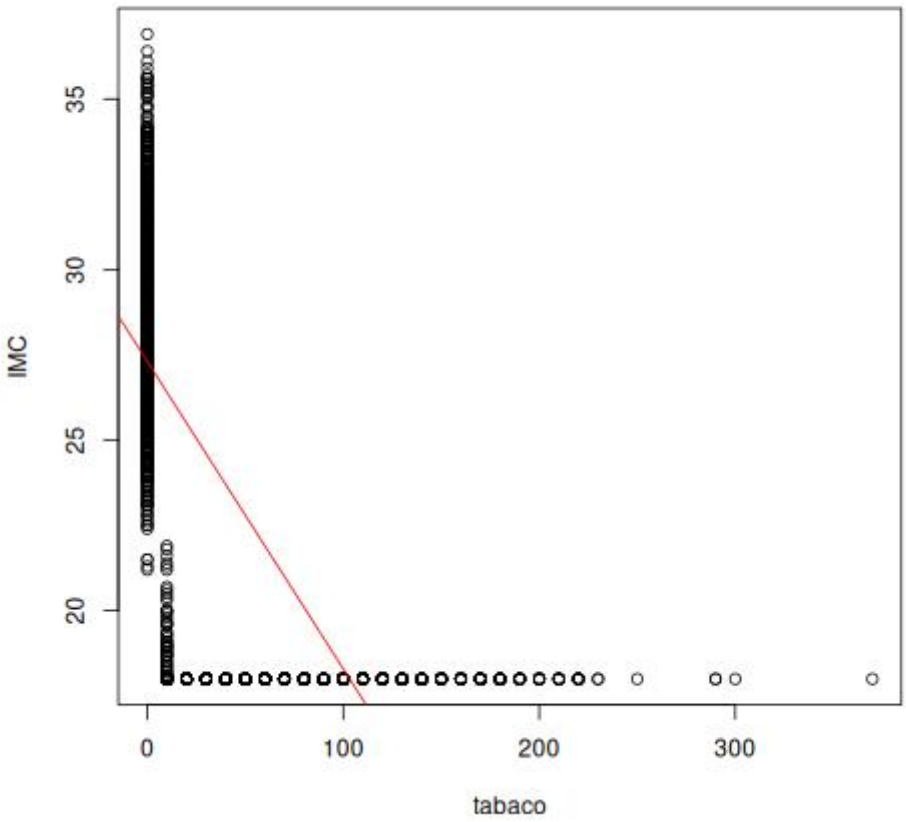

IMC
20
25
30
35
0
3
1
2
4
nivEstPad

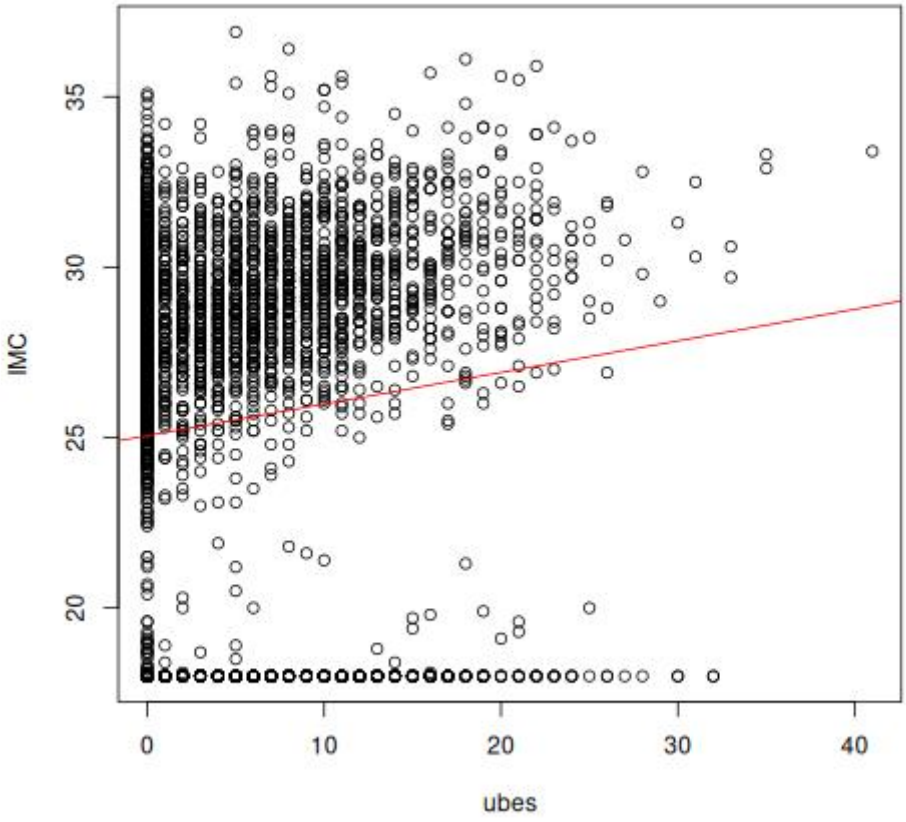

IMC
20
25
30
35
0
4
2
1
3
nivEstudios

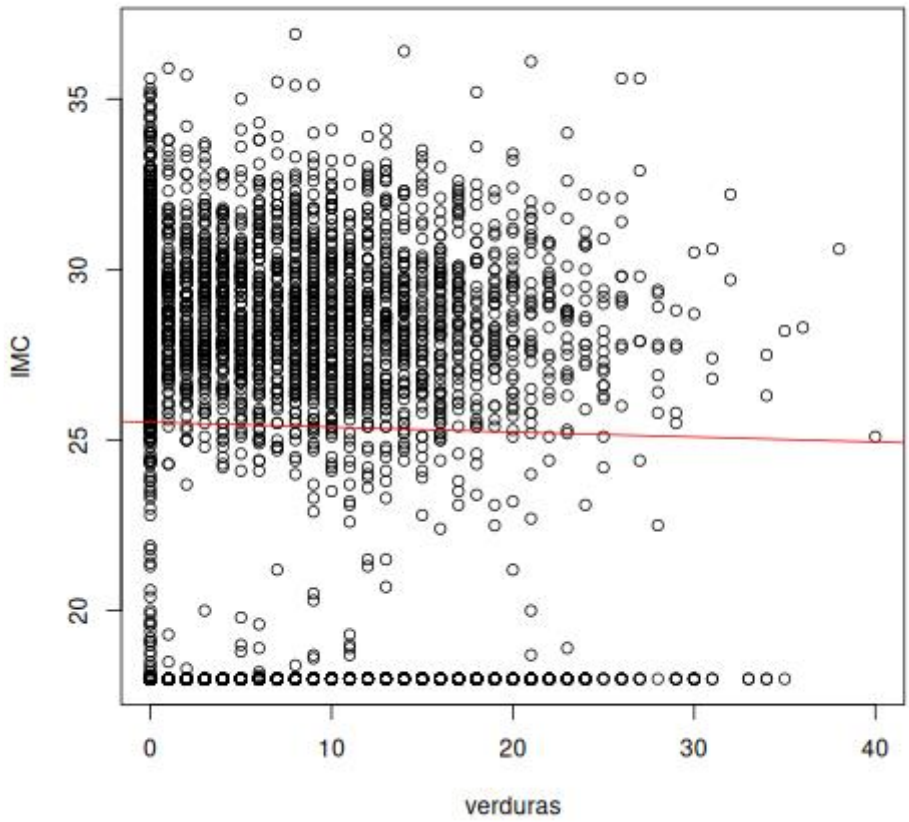

IMC
20
25
30
35
0
4
2
1
3
nivIngresos


IMC
peso
20
25
30
35
40
60
80
100
120


IMC
sexo
20
25
30
35
V
M


IMC
20
25
30
35
0
100
200
300
tabaco


IMC
20
25
30
35
0
10
20
30
40
ubes



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

Se crea la función separarSets que dados un dataframe y dos porcentajes crea tres sets.
De 0 a p1 el entrenamiento.
De p1 a p2 el test.
De p2 a 1 la validación.

El método sample te permite obtener una sección del dataframe pasandole una logitud.
Por último se devuelve una lista con los tres sets.

Se genera unos sets con los porcentajes pedidos. 60%, 20%, 20%.

```
separarSets <- function(df, p1, p2) {
  rDf <- 1:nrow(df)
  rTrain <- sample(rDf, p1 * length(rDf))
  rResto  <- setdiff(rDf, rTrain)
  rTest <- sample(rResto, p2*length(rDf))
  rValid <- setdiff(rResto, rTest)

  list(train=df[rTrain,], test=df[rTest,], valid=df[rValid,])
}

setsSeparados <- separarSets(data,.6,.2)
```

```
> separarSets <- function(df, p1, p2) {
+   rDf <- 1:nrow(df)
+   rTrain <- sample(rDf, p1 * length(rDf))
+   rResto  <- setdiff(rDf, rTrain)
+   rTest <- sample(rResto, p2*length(rDf))
+   rValid <- setdiff(rResto, rTest)
+
+   list(train=df[rTrain,], test=df[rTest,], valid=df[rValid,])
+ }
> setsSeparados <- separarSets(data,.6,.2)
>
```

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.

Se crea una función que permite calcular el R2 ajustado el cual relacionará de forma individual las dos variables.

Se guarda en AjstR2 el vecton con cada uno de los R2 ajustados de cada variable con respecto al IMC.

Para obtener la mejor variable sacamos el valor máximo de este vector y podemos observar como efectivamente el tabaco es la variable que más relación directa puede tener según los resultados.

```
calcR2ajst <- 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
  ajR2 <- 1 - (1- R2) * (nrow(df) - 1) / (nrow(df) - mod$rank)
  ajR2
}

calcModR2 <- function(dfTrain, dfTest, y, x) {
  mod <- linearAdjust(dfTrain, y, x)
  calcR2ajst(dfTest, mod$mod, y)
}

AjstR2 <- varPredictorias %>%
map_dbl(calcModR2,dfTrain=setsSeparados$train,dfTest=setsSeparados$test,y=
"IMC")

x <- which.max(AjstR2)
mejorVar <- varPredictorias[x]

mejorVar
AjstR2[x]
```

```
> calcModR2 <- function(dfTrain, dfTest, y, x) {
+   mod <- linearAdjust(dfTrain, y, x)
+   calcR2ajst(dfTest, mod$mod, y)
+ }
> AjstR2 <- varPredictorias %>%
+   map_dbl(calcModR2,dfTrain=setsSeparados$train,dfTest=setsSeparados$test,y="IMC")
> x <- which.max(AjstR2)
> mejorVar <- varPredictorias[x]
> mejorVar
[1] "tabaco"
> AjstR2[x]
[1] 0.4803668
>
```

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.

Se crea una función que permite encontrar el mejor ajuste añadiendo siempre la mejor variable. Para ello se usa el repeat donde cada vez se comprueban los R2 de cada una de las variables predictorias y se almacena la mejor de ellas (la más cercana a 1).

Se calcula el mejor ajuste con los sets de entrenamiento y testeo.

```
encontrarMejorAjuste <- function(dfTrain, dfTest, varPos) {
  bestVars <- character(0)
  aR2      <- 0

  repeat {
    aR2v <- map_dbl(varPos, ~calcModR2(dfTrain, dfTest, "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]
  }

  mod <- linearAdjust(dfTrain, "IMC", bestVars)

  list(vars=bestVars, mod=mod)
}

mejorAjuste <- encontrarMejorAjuste(setsSeparados$train,
setsSeparados$test, varPredictorias)
mejorMod <- mejorAjuste$mod$mod
```

```
> encontrarMejorAjuste <- function(dfTrain, dfTest, varPos) {
+   bestVars <- character(0)
+   aR2      <- 0
+
+   repeat {
+     aR2v <- map_dbl(varPos, ~calcModR2(dfTrain, dfTest, "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]
+   }
+
+   mod <- linearAdjust(dfTrain, "IMC", bestVars)
+
+   list(vars=bestVars, mod=mod)
+ }
> mejorAjuste <- encontrarMejorAjuste(setsSeparados$train, setsSeparados$test, varPredictorias)
0.4804 tabaco
There were 11 warnings (use warnings() to see them)
> mejorMod <- mejorAjuste$mod$mod
>
```

10. Evalúa el resultado en el conjunto de validación.

Se compara el resultado obtenido en el mejor ajuste con el R2 ajustado obtenido en el set de validación al que se le pasa como parametro el modelo generado por este primero.

```
calcR2ajst(setsSeparados$valid, mejorMod, "IMC")
```

```
> calcR2ajst(setsSeparados$valid, mejorMod, "IMC")
[1] 0.4749816
>
```

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.

Se lee el archivo eval y se crea una nueva columna con el método predict haciendo uso del mejor modelo obtenido.

Una vez calculado el IMC se puede obtener el peso despejando.

Por último se escribe en el archivo evalX.csv el cual será entregado junto al trabajo.

```
dfEval <- read_csv("eval.csv")

dfEval["IMC"] <- predict.lm(mejorMod, dfEval)
dfEval["Peso"] <- dfEval$IMC*dfEval$altura^2

write.csv(dfEval, "evalX.csv", row.names = FALSE)
```

```
> dfEval <- read_csv("eval.csv")
Rows: 1000 Columns: 13
── Column specification ────────────────────────────────────────────────────────
Delimiter: ","
chr  (2): sexo, dietaEsp
dbl (11): altura, edad, tabaco, ubes, carneRoja, verduras, deporte, drogas, nivEstPad, nivEstudios...

ℹ Use `spec()` to retrieve the full column specification for this data.
ℹ Specify the column types or set `show_col_types = FALSE` to quiet this message.
> dfEval["IMC"] <- predict.lm(mejorMod, dfEval)
> dfEval["Peso"] <- dfEval$IMC*dfEval$altura^2
> write.csv(dfEval, "evalX.csv", row.names = FALSE)
>
```

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

Se trata de un modelo muy básico con un R2 de 0.5. Es un modelo que toma una única variable predictoria por lo tanto no es un modelo fiable. Sin embargo suponiendo que tenemos estos datos y desconocemos el IMC y el peso es una implementación que te puede ayudar a tener al menos una base sobre la que trabajar. Según estos datos podemos ver como la gente que fuma tiene por normal general un IMC muy bajo con respecto a la media. Vemos que si que están relacionadas y por eso mismo nos pueden ayudar a predecirla.

Los principales problemas que he tenido con el trabajo han sido a la hora de implementar el código en R ya que no es un lenguaje con el que esté familiarizado. Cometía muchos errores con el manejo de map y las funciones lambda.

Me llama mucho la atención en general el estudio de datos. Me gusta averiguar que datos están relacionados y como puedo utilizar esto para llegar a conclusiones.

Se podría estudiar al completo los datos como por ejemplo que personas son las más propensas a comer verduras, que relación hay entre las personas que hacen deporte y el nivel de ingresos o si el nivel de estudios implica un nivel de ingresos mayor.