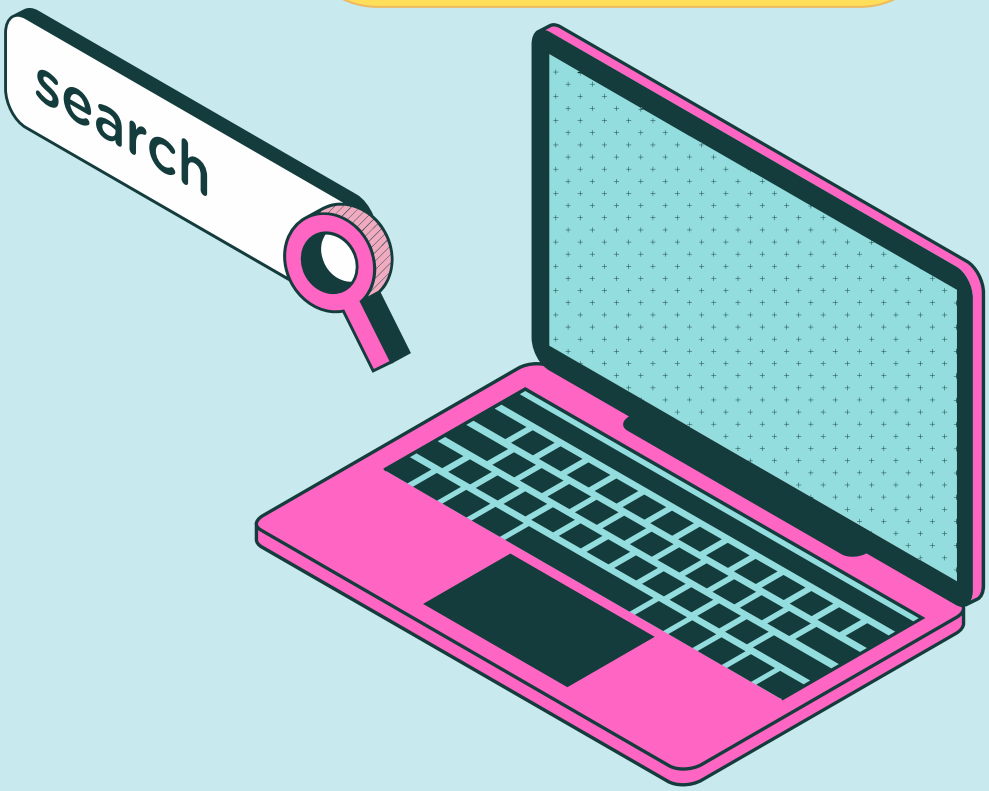
PROYECTO R

ESTADISTICA



Nombre: Rocío Guzmán Arroyo
Ingeniería Informática B

INTRODUCCIÓN

En este proyecto resolveremos todos los apartados que se nos piden y explicaremos adecuadamente todos los resultados obtenidos y las funciones utilizadas.

Como primer objetivo debemos crear un data frame en que incluyamos una nueva variable la cual depende del peso y la altura. Y eliminaremos todos los elementos que sean incoherentes(NA) como pueden ser divisiones por cero y raíces negativas.

```
#TRABAJO DE ESTADÍSTICA

#Cargamos las librerias necesarias para ejecutar el programa


install.packages("tidyverse")
install.packages("dplyr")
install.packages("magrittr")
install.packages("knitr")

library(tidyverse)
library(dplyr)
library(magrittr)
library(knitr)

# Introducimos la direccion donde se encuentra el archivo que quiero abrir

pathPlots <- 'C:/Users/HP/OneDrive/Documentos'
name = 'MEC19827'
```

```
IMC = x$peso/ (x$altura)^2
y <- IMC
z <- all_vars(!is.na(.))

#Asignamos el parametro  IMC a la valiable y, utilizamos mutate y filtrar todo,donde z es la condicion,
#para añadirla a nuestra nueva tabla y eliminar los valores vacios.


df <- g(f(x, y), z)
df <- filter_all(mutate(read_csv(str_c(name, ".csv"), col_types=cols(.default = col_double(),
                                                                     sexo       = col_factor(NULL),
                                                                     dietaEsp   = col_factor(NULL))), IMC = peso/altura^2), all_vars(!is.na(.)))
```

Hemos ejecutado todas las librerías necesarias para poder llevar a cabo el proyecto, posterior mente hemos creado la variable IMC con su formula correspondiente, hemos identificado las variables no numéricas para que sean leídas y utilizado los comandos mutate y filtrar para poder añadir nuestra nueva variable y filtrar bajo la condición de eliminar valores incoherente. Pasamos IMC a 'y' para que en la table el nombre de la columna sea IMC.

La tabla resultante es la siguiente:

	peso	altura	sexo	edad	tabaco	ubes	carneRoja	verduras	deporte	drogas	dietaEsp	nivEstPad	nivEstudios	nivIngresos	IMC
1	84.54	1.69	V	28	0	0	3	0	0	0	N	2	2	1	29.59980
2	86.18	1.67	M	45	0	2	0	0	4	0	N	1	0	1	30.90107
3	77.81	1.72	V	38	0	7	1	4	7	0	N	1	2	1	26.30138
4	70.78	1.65	M	18	0	0	0	2	1	2	N	2	4	4	25.99816
5	49.00	1.65	M	18	210	18	2	16	7	5	S	1	3	3	17.99816
6	48.41	1.64	V	49	130	0	3	11	4	0	N	0	1	1	17.99896
7	92.62	1.76	M	40	0	11	3	0	0	0	N	2	4	4	29.90057
8	61.41	1.62	M	45	0	4	0	10	12	0	N	2	3	4	23.39963
9	102.66	1.73	M	71	0	5	3	0	0	0	N	3	3	3	34.30118
10	69.63	1.60	M	42	0	0	0	0	3	0	N	2	4	4	27.19922
11	72.08	1.64	M	32	0	0	2	5	0	0	N	1	2	1	26.79952
12	56.51	1.72	V	42	0	6	5	20	12	0	N	0	2	2	19.10154
13	93.60	1.64	M	58	0	5	0	0	0	0	N	3	3	3	34.80071
14	54.10	1.67	V	25	0	0	0	14	6	0	N	1	2	2	19.39833
15	80.54	1.76	V	60	0	0	1	3	10	0	N	2	3	2	26.00077
16	100.88	1.77	V	41	0	6	2	0	0	0	N	0	1	3	32.20020
17	84.21	1.71	V	39	0	0	8	0	1	5	N	1	2	3	28.79860
18	76.01	1.81	V	31	0	4	0	14	6	0	N	1	2	2	23.20137
19	57.73	1.67	M	59	0	0	3	12	13	0	N	1	4	4	20.69992
20	68.55	1.63	M	31	0	0	2	4	2	2	N	1	3	3	25.80075

Podemos ver un fragmento de la tabla ya que como nos muestra el comando 'nrow' nuestra tabla tiene cerca de 5000 filas. A través del comando 'view(df)' hemos podido generar la tabla.

El siguiente apartado en cuestión es calcular las medias y desviaciones típicas de todas las variables numéricas, la hemos calculado con sus formulas correspondientes y los resultados adquiridos son:

```
#Peso

media_peso <- mean(df[[1]])
media_peso

desV_peso <- sqrt (mean((df[[1]])^2) - media_peso^2)
desV_peso

#Altura

media_altura <- mean(df[[2]])
media_altura

desV_altura <- sqrt (mean((df[[2]])^2) - media_altura^2)
desV_altura

#Edad

media_edad <- mean(df[[4]])
media_edad

desV_edad <- sqrt (mean((df[[4]])^2) - media_edad^2)
desV_edad

#Tabaco

media_tabaco <- mean(df[[5]])
media_tabaco

desV_tabaco <- sqrt (mean((df[[5]])^2) - media_tabaco^2)
```

```r
#Ubes

media_ubes <- mean(df[[6]])
media_ubes

desV_ubes <- sqrt (mean((df[[6]])^2) - media_altura^2)
desV_ubes

#Carne roja

media_roja <- mean(df[[7]])
media_roja

desV_roja <- sqrt (mean((df[[7]])^2) - media_roja^2)
desV_roja

#Verdura

media_verdura <- mean(df[[8]])
media_verdura

desV_verdura <- sqrt (mean((df[[8]])^2) - media_verdura^2)
desV_verdura
```

```r
#Deporte

media_deporte <- mean(df[[9]])
media_deporte

desV_deporte <- sqrt (mean((df[[9]])^2) - media_deporte^2)
desV_deporte

#Drogas

media_drogas <- mean(df[[10]])
media_drogas

desV_drogas <- sqrt (mean((df[[10]])^2) - media_drogas^2)
desV_drogas

#Nivel padres


media_padres <- mean(df[[12]])
media_padres

desV_padres <- sqrt (mean((df[[12]])^2) - media_padress^2)
desV_padres
```

```r
#Nivel estudios


media_estudios <- mean(df[[13]])
media_estudios

desV_estudios <- sqrt (mean((df[[13]])^2) - media_estudios^2)
desV_estudios

#Nivel de ingresos

media_ingresos <- mean(df[[14]])
media_ingresos

desV_ingresos <- sqrt (mean((df[[14]])^2) - media_ingresos^2)
desV_ingresos

#IMC

media_IMC <- mean(df[[15]])
media_IMC

desV_IMC <- sqrt (mean((df[[15]])^2) - media_IMC^2)
desV_IMC
```

Y los resultados obtenidos al compilar el programa son:

```
> media_peso
[1] 72.82414
> desV_peso
[1] 16.17376
> media_altura
[1] 1.699542
> desV_altura
[1] 0.07051662
> media_edad
[1] 40.2661
> desV_edad
[1] 13.99738
> media_tabaco
[1] 19.99394
> desV_tabaco
[1] 41.60543
> media_ubes
[1] 4.103977
> desV_ubes
[1] 6.969122
> media_roja
[1] 1.725823
> desV_roja
[1] 2.097159
> media_verdura
[1] 5.77125
> desV_verdura
[1] 6.902509
> media_deporte
[1] 4.041187
> desV_deporte
[1] 4.579768
> media_drogas
[1] 0.4853624
> desV_drogas
[1] 1.397785
> media_padres
[1] 1.233192
> desV_padres
[1] 0.9546197
> media_estudios
[1] 2.174238
> desV_estudios
[1] 1.250127
> media_ingresos
[1] 2.144963
> desV_ingresos
[1] 1.369236
> media_IMC
[1] 25.16752
> desV_IMC
[1] 5.170356
```

El siguiente apartado nos pide calcular los coeficientes de regresión y determinación de todas las variables unidimensionales, asique utilizamos el comando 'setdiff' para eliminar las variables IMC, peso y altura ya que al depender directamente de IMC no nos importa mucho la imformación que nos brinda.

```
#Calculamos los coeficientos de regresion y determinancion
selected <- setdiff(names(df), c("altura", "IMC", "peso"))
selected
names(selected) <- selected

linearAdjust <- function(df, y, x) lm(str_c(y, "~", str_c(x, collapse="+")), df)


modelos <- map(selected, linearAdjust, df=df, y= "IMC")
modelos %>% map("coefficients")

#Una vez optenemos los coeficientes de regresión ahora calculamos
#el coeficiente de determinacion
modelos %>% map(summary)
map_dbl(map(modelos, summary), "r.squared")
```

Principalmente hemos seleccionado las variables que necesitábamos las cuales serían:

```
> selected
 [1] "sexo"       "edad"       "tabaco"     "ubes"       "carneRoja"   "verduras"
 [7] "deporte"    "drogas"     "dietaEsp"   "nivEstPad"  "nivEstudios" "nivIngresos"
```

Después de eso como estamos comparando todas las variables con IMC utilizamos la función de ajuste linear que crea un ajuste linear de todas las variables seleccionadas.

Y creamos los modelos de R que tienen los ajustes de todas las variables seleccionadas frente a la variable 'y'.

Utilizamos el comando 'modelos %>% map("coefficients")' para obtener los coeficientes de regresión, ya que la función map nos puede proporcionar gran cantidad de información acerca de nuestras variables.

A través de Sumary obtenemos un resumen de información de las variables pero solo seleccionamos el coeficiente de determinación.

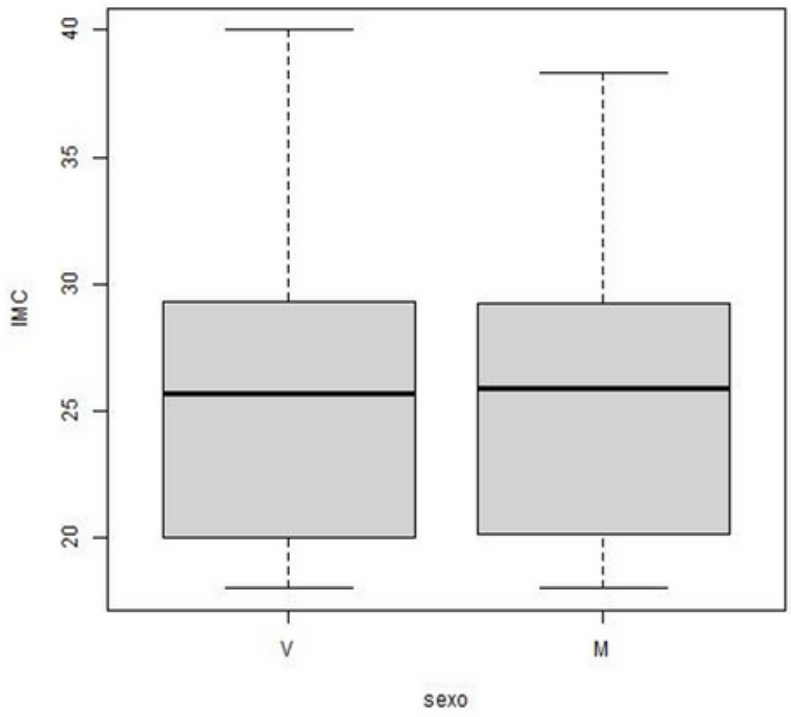
```
$sexo
  (Intercept)         sexoM
25.1676699265 -0.0003034691

$edad
(Intercept)        edad
 21.0208183   0.1029824

$tabaco
(Intercept)      tabaco
26.63910689 -0.07360181

$ubes
(Intercept)        ubes
 24.2854420   0.2149316

$carneRoja
(Intercept)   carneRoja
 24.9627434   0.1186525

$verduras
(Intercept)    verduras
 27.4230173  -0.3908167
```

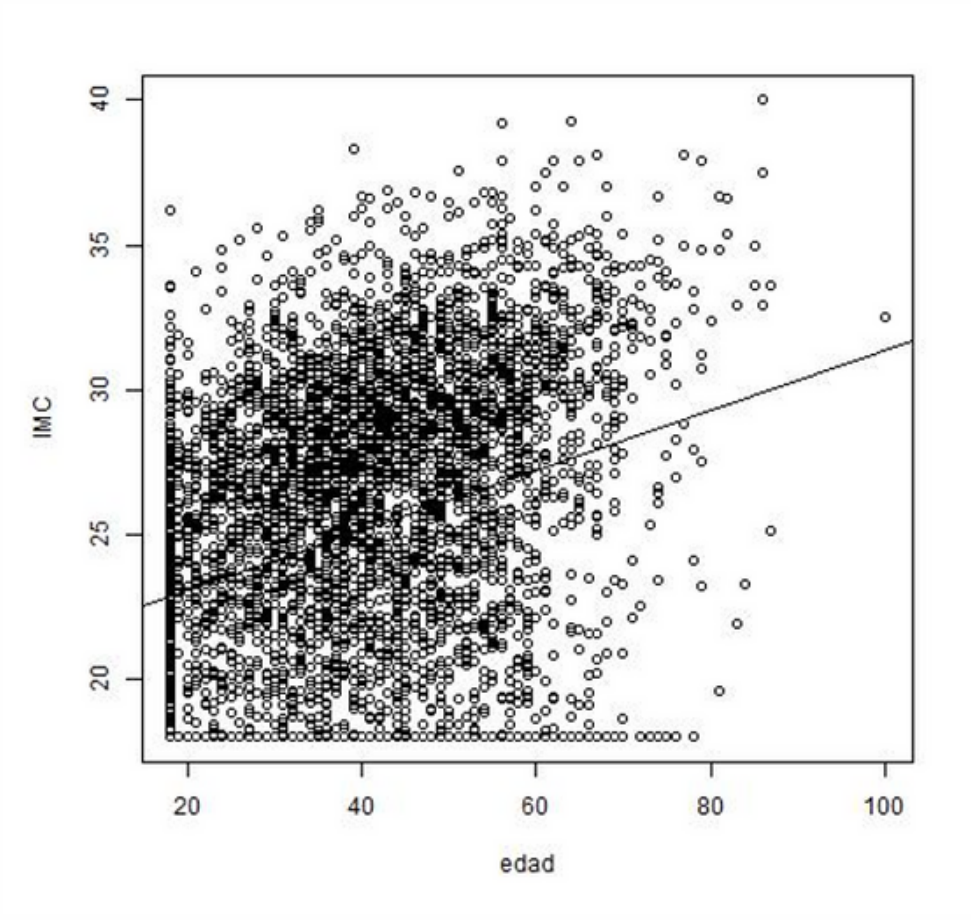
```
$verduras
(Intercept)    verduras
 27.4230173  -0.3908167

$deporte
(Intercept)     deporte
 27.0483041  -0.4654047

$drogas
(Intercept)      drogas
25.15465339  0.02650222

$dietaEsp
(Intercept)   dietaEspS
25.17186564 -0.09205534

$nivEstPad
(Intercept)   nivEstPad
 25.3139920  -0.1187775

$nivEstudios
(Intercept) nivEstudios
  25.968371   -0.368338

$nivIngresos
(Intercept) nivIngresos
 26.1272111  -0.4474178
```

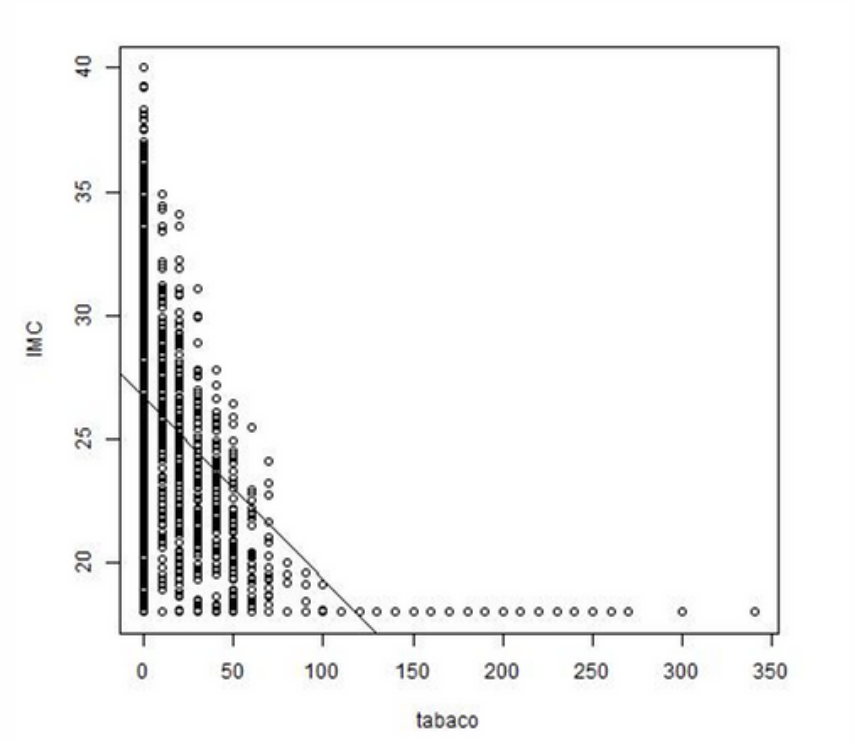
```
> map_dbl(map(modelos, summary), "r.squared")
        sexo         edad       tabaco         ubes    carneRoja     verduras
8.611487e-10 7.772818e-02 3.507809e-01 5.981600e-02 2.316198e-03 2.722194e-01
     deporte       drogas     dietaEsp    nivEstPad  nivEstudios  nivIngresos
1.699447e-01 5.133393e-05 1.426876e-05 4.809367e-04 7.931590e-03 1.403919e-02
```

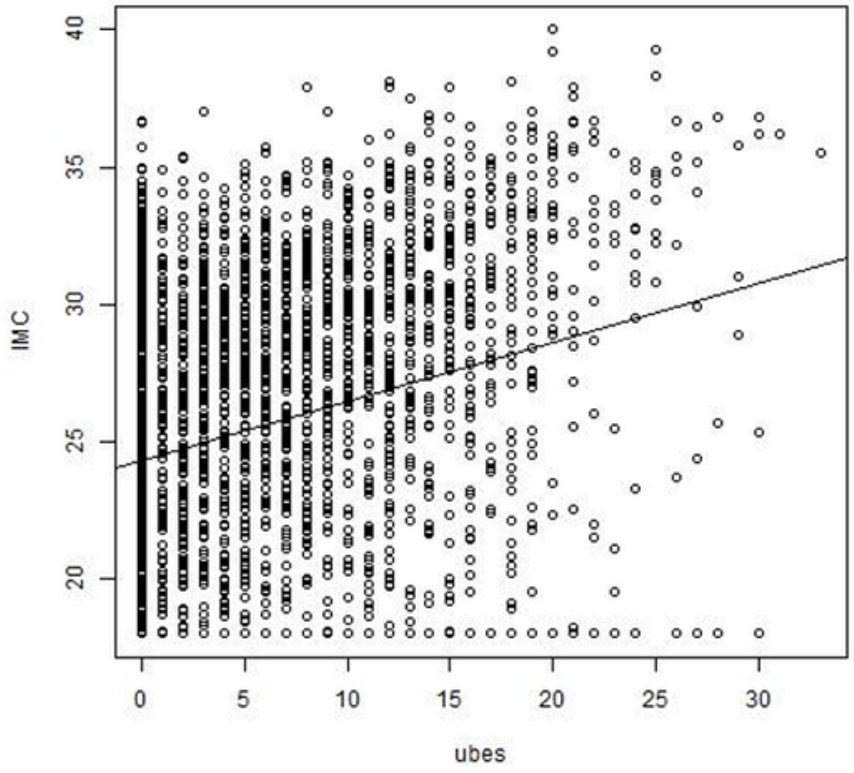
Lo siguiente que se nos pide es representar gráficamente todas las variables seleccionadas frente a IMC, si son no numéricas se representa con box-plots que son diagramas de cajas como los generados a continuación.

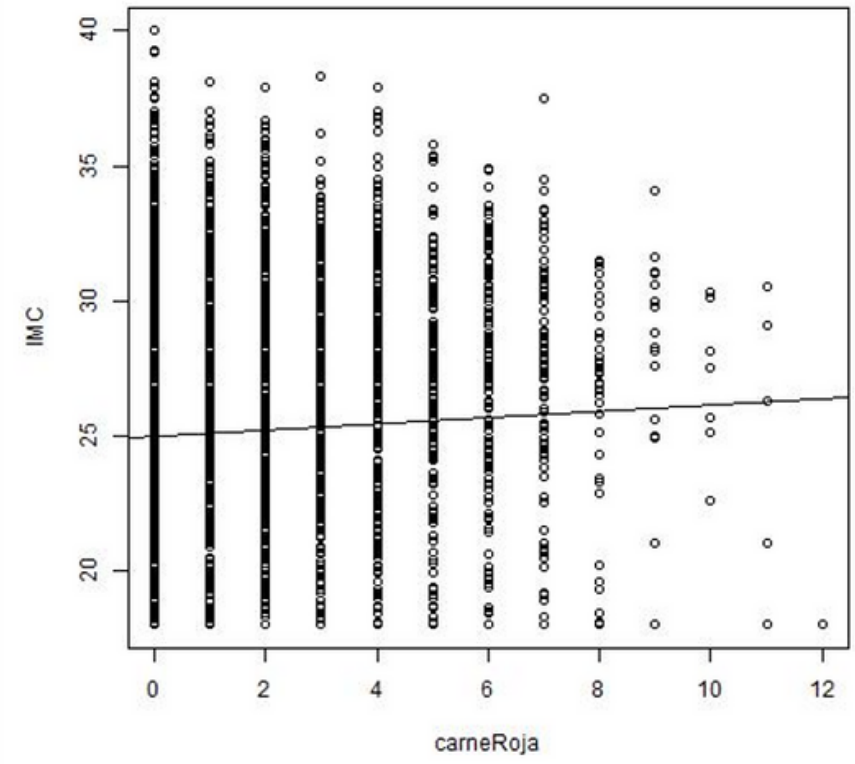
```
#Representa los graficos
dibujarModelos <- function(df, mod, y, x) {
  jpeg(str_c(pathPlots, 'grafico_', x, '.jpeg'))
  plot(df[[x]], df[[y]], xlab=x, ylab=y)
  if (is.numeric(df[[x]])) abline(mod)
  dev.off()
}

iwalk(modelos, dibujarModelos, df=df, y="IMC")
modelos %>% iwalk(dibujarModelos)
```

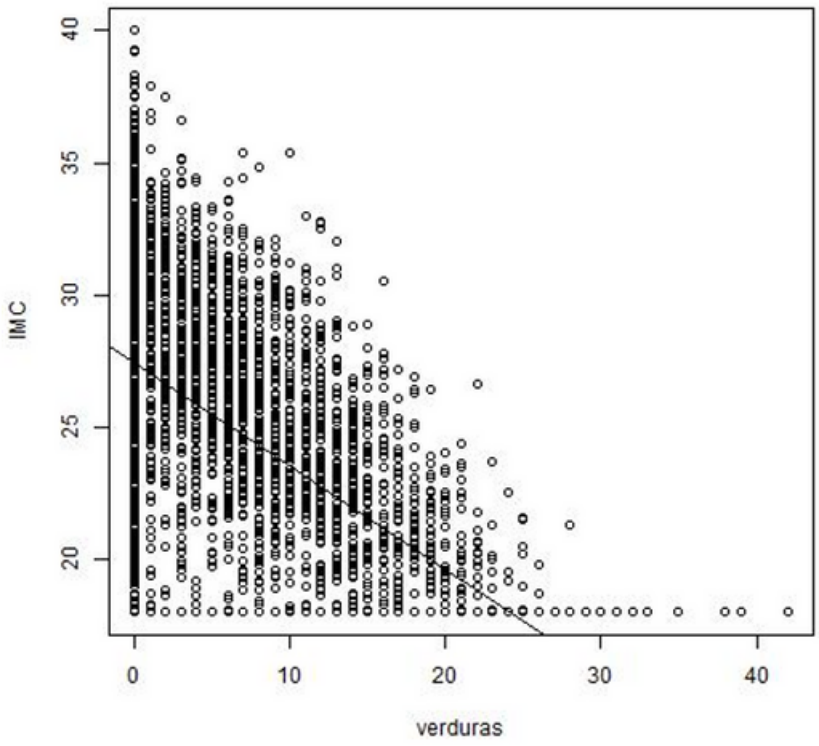


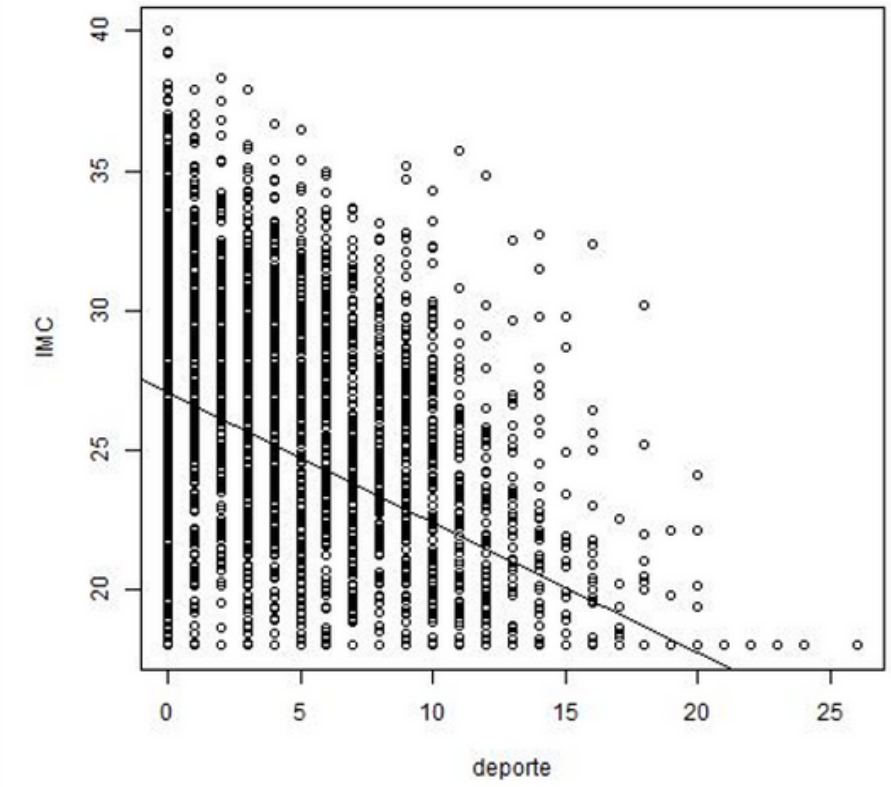

IMC
edad
20
25
30
35
40
20
40
60
80
100

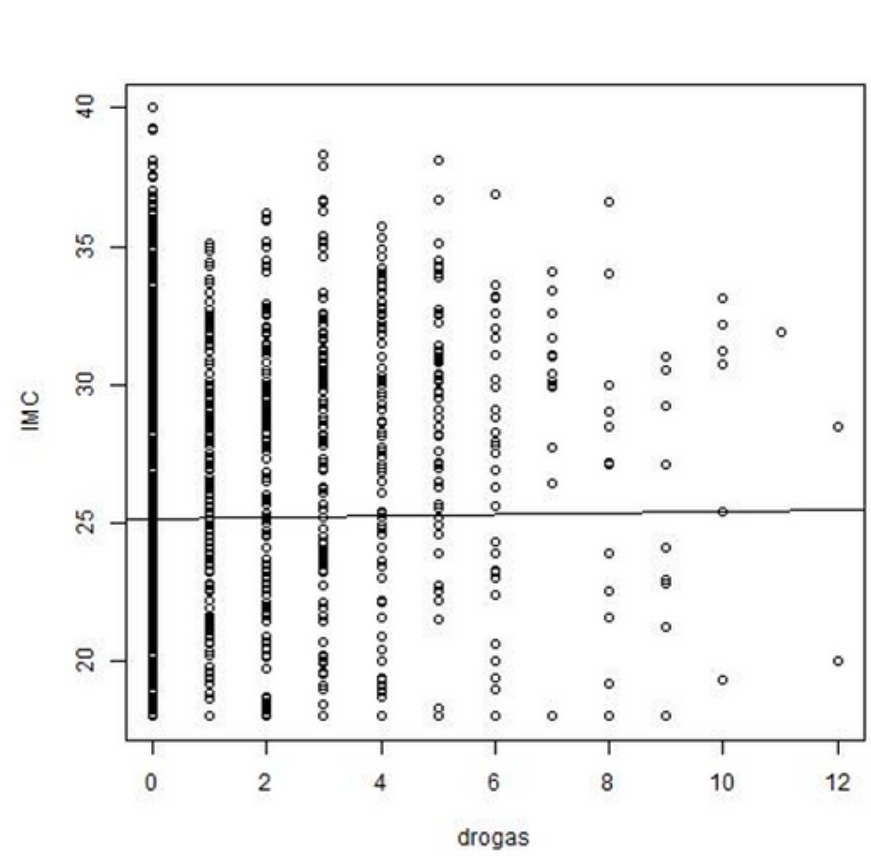

IMC
tabaco
20
25
30
35
40
0
50
100
150
200
250
300
350

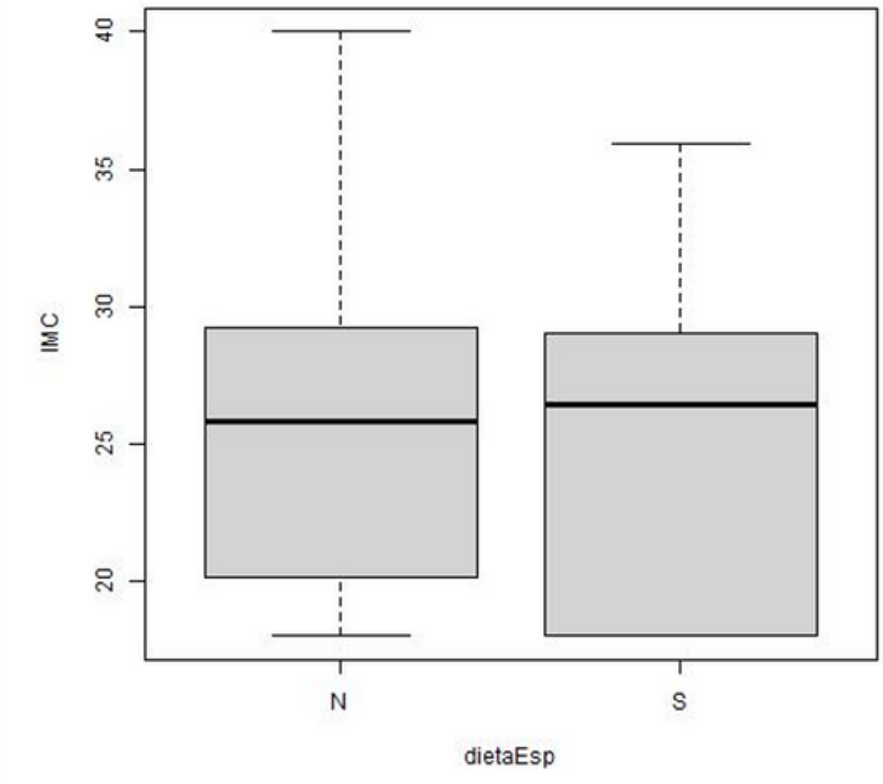

IMC
ubes
0
5
10
15
20
25
30
20
25
30
35
40

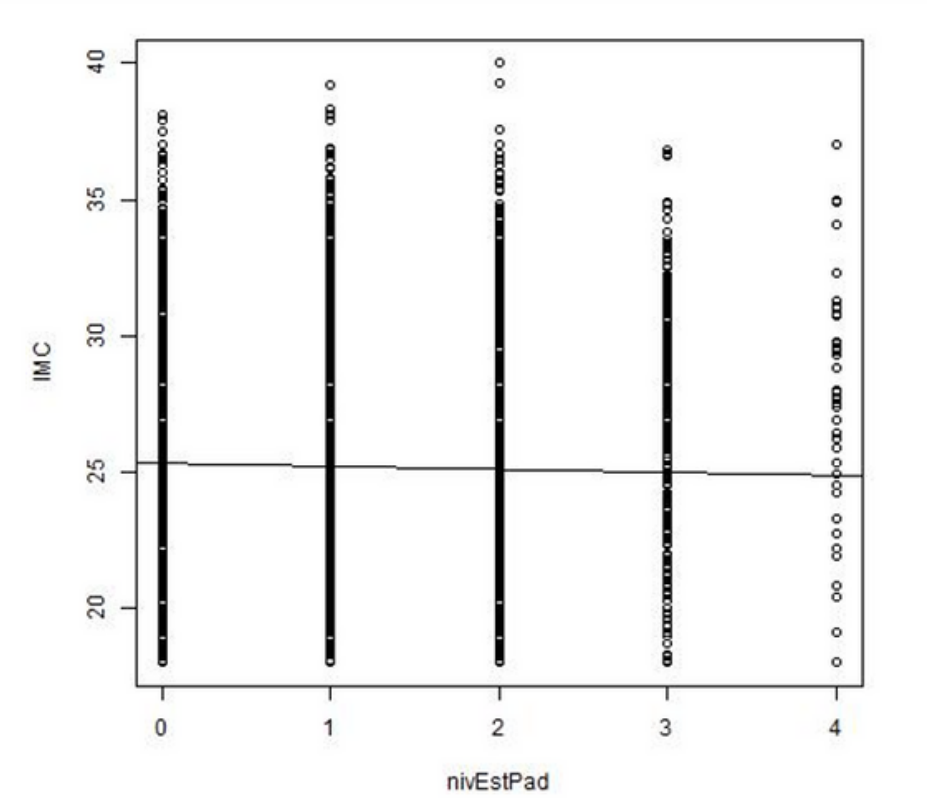

IMC
carneRoja
0
2
4
6
8
10
12
20
25
30
35
40

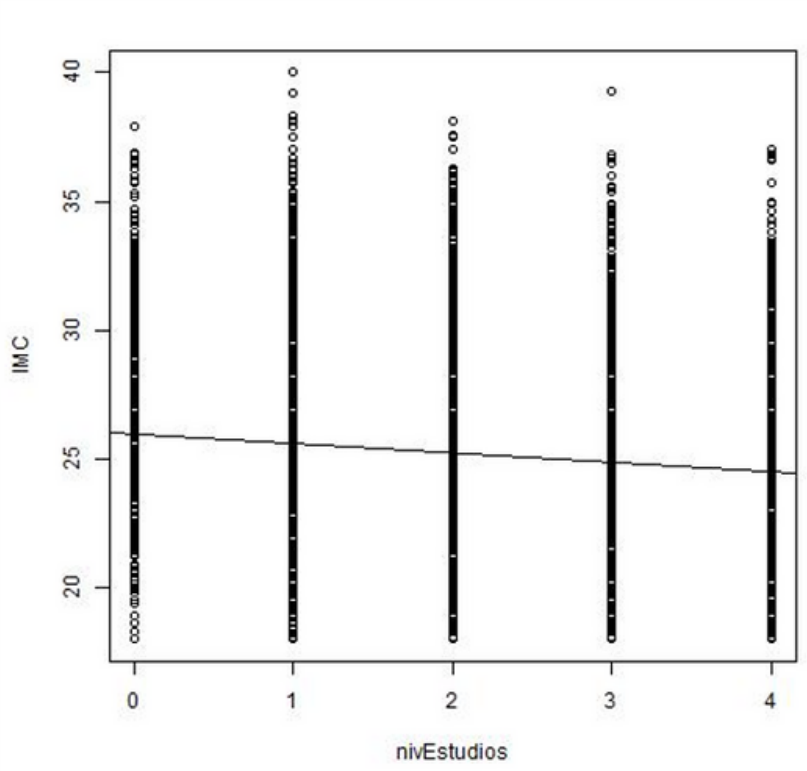

IMC
verduras
0
10
20
30
40
20
25
30
35
40

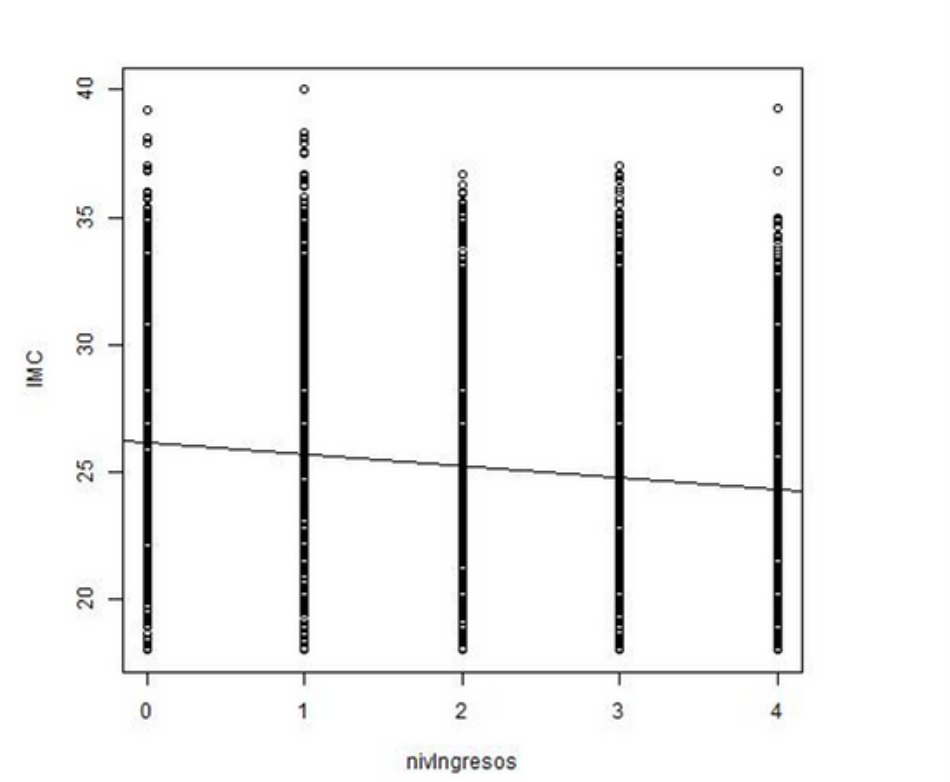

IMC
deporte
0
5
10
15
20
25
20
25
30
35
40


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


IMC
40
35
30
25
20
N
S
dietaEsp


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


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



Para comprender la función dibujar modelos lo único que debemos saber es que primeramente le decimos el sitio donde queremos que lo guarde el formato y su nombre, segundo creo un grafico con la función plot y además comprueba si es numérica o no.

El siguiente apartado nos pide separar la muestra en tres grupos train , que corresponde al 60% y test y valid que corresponden al 20 % cada uno.

```
separarSets <-function(df,p1,p2){

rDf    <- 1:nrow(df)
rTrain <- sample(rDf, p1* length(rDf))
B <- setdiff(rDf,rTrain)
rTest <- sample(B, p2* length(B))
C<- setdiff(B,rTest)

list(train=df[rTrain,] ,test=df[rTest,],val=df[C,])
}

datos <- separarSets(df,0.6,0.5)
datos
```

Lo que hacemos es, creamos la función 'separarset' que va a hacer unas determinadas acciones , luego invocamos a la función y lo guardamos en datos dando valores específicos para que nos de un resultados

Dentro de la función separarSets utilizamos sample para obtener los valores de la muestra del tamaño que necesitamos y utilizamos setdiff para no utilizar valores repetidos ya que eliminamos del conjunto los ya seleccionados

El resultado obtenido es el siguiente:

```
$train
# A tibble: 2,971 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  90.9   1.71 M        34      0     3         3        0       0      0 N
 2  76.8   1.84 V        45      0     0         2       10       8      0 N
 3  78.1   1.72 V        60      0    17         0       19      11      0 N
 4  93.2   1.76 V        54      0     1         0        4       0      3 N
 5  83.2   1.81 V        21      0     0         3        2       0      0 N
 6  54.5   1.74 V        35    120     0         5        6       0      0 N
 7  55.8   1.76 V        47     40     8         0       19       6      0 S
 8  82.8   1.65 V        27      0     0         0        0       0      0 N
 9  54.0   1.7  V        28      0     6         5       26       8      2 N
10  93.4   1.75 V        56      0     0         0        0       1      0 N
# ... with 2,961 more rows, and 4 more variables: nivEstPad <dbl>, nivEstudios <dbl>,
#   nivIngresos <dbl>, IMC <dbl>

$test
# A tibble: 991 x 15
    peso altura sexo   edad tabaco  ubes carneRoja verduras deporte drogas dietaEsp
   <dbl>  <dbl> <fct> <dbl>  <dbl> <dbl>     <dbl>    <dbl>   <dbl>  <dbl> <fct>
 1 134.    1.83 V        86      0    20         0        0       0      0 N
 2  95.2   1.75 V        59      0     8         9        0       6      0 N
 3  66.4   1.54 M        44      0    17         0       15      10      0 N
 4  77.4   1.77 V        35     10     0         0        0       1      0 N
 5  64.1   1.68 M        28      0     0         1       10       7      0 N
 6  48.4   1.64 M        52    150     0         1        0       0      0 S
 7  50.8   1.64 M        18      0     0         1       17       5      0 N
 8  66.2   1.65 M        36      0     2         3       10       0      0 N
 9  76.8   1.65 M        63     10     0         0        2       0      0 N
10  47.8   1.63 M        30    140     0         0        0       2      1 N
# ... with 981 more rows, and 4 more variables: nivEstPad <dbl>, nivEstudios <dbl>,
#   nivIngresos <dbl>, IMC <dbl>

$val
# A tibble: 991 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  70.8   1.65 M        18      0     0         0        2       1      2 N
 2  56.5   1.72 V        42      0     6         5       20      12      0 N
 3  57.7   1.67 M        59      0     0         3       12      13      0 N
 4  75.3   1.76 V        47      0     0         5        8       7      0 N
 5  89.3   1.66 M        56      0     5         1        0       0      0 N
 6  47.2   1.62 M        24    100     0         4       13       7      0 N
 7  57.7   1.79 V        27    130     0         0       15      13      2 N
 8  62.3   1.53 M        43      0     0         4        8       1      0 N
 9  90.7   1.84 V        28      0     0         4        0       3      0 N
10  62.3   1.69 M        56      0     2         0       13      10      0 N
# ... with 981 more rows, and 4 more variables: nivEstPad <dbl>, nivEstudios <dbl>,
```

El siguiente apartado nos pide que seleccionemos la variable que mejor se ajusta a IMC con los conjuntos train y test.

```
modelos$rank

calcR2 <- 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
  aR2 <- 1 - (1- R2) * (nrow(df) - 1) / (nrow(df) - mod$rank)

  tibble(MSE=MSE, varY=varY, R2=R2, aR2=aR2)
}
calcModR2 <- function(dfTrain, dfTest, y, x) {
  mod <- linearAdjust(dfTrain, y, x)
  calcR2(dfTest, mod, y)$aR2
}
a2 <- map_dbl(selected,calcModR2, dfTrain=datos$train,
                           dfTest=datos$test, y="IMC")
a2
bestVar <- which.max(a2)
bestVar
```

El espacio que utilizamos son las muestras train y test, calculamos los coeficientes con la formula aR2.Pasamos todos los coeficientes a a2 los cuales son:

```
> a2
        sexo         edad       tabaco         ubes     carneRoja      verduras
-0.0028294283  0.0536769361  0.3318501402  0.0742995558 -0.0002320924  0.2851643440
     deporte       drogas      dietaEsp     nivEstPad   nivEstudios   nivIngresos
 0.1646783256 -0.0019039653 -0.0019930087 -0.0019172058 -0.0002769027 -0.0003045644
> |
```

Y which.max selecciona el valor más alto que sin duda es tabaco con un valor de 0.3318

La función encontrarMejorAjuste va introduciendo en bestVars la cual hemos inicializado a cero, las variables que mejoren el aR2.
Como la mejor variable es la que sea mayor cuando ya no podamos mejorar más nos salimos del bucle.
Utilizamos cat para añadir a la consola las variables que se van añadiendo a aR2

En best1 tenemos el modelo generado al ir añadiendo variables simples al modelo,
pero en best2 tenemos pares de valiables que hemos separado mediente :

```
linearAdjust <- function(df, y, x) {
  lm(str_c(y, "~", str_c(x, collapse="+")), df)
}
# Funcion que calcula el mejor ajuste lineal
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)
}

best1 <- encontrarMejorAjuste(datos$train, datos$test, selected)

selected2 <- crossing(var1=selected, var2=selected) %>% pmap_chr(str_c,
                                                                 sep=":")
best2 <- encontrarMejorAjuste(datos$train, datos$test, selected2)
```

YLos resultados obtenidos son;

```
> best1 <- encontrarMejorAjuste(datos$train, datos$test, selected)
0.3319 tabaco
0.6122 verduras
0.6815 ubes
0.7466 edad
0.7695 deporte
0.7835 nivIngresos
0.7865 nivEstPad
0.7891 carneRoja
0.7915 drogas
0.7927 nivEstudios
```

```
> best2 <- encontrarMejorAjuste(datos$train, datos$test, selected2)
0.3319 tabaco:tabaco
0.6124 dietaEsp:verduras
0.7007 edad:ubes
0.7411 tabaco:verduras
0.7805 edad:edad
0.8040 deporte:nivIngresos
0.8225 edad:tabaco
0.8285 ubes:ubes
0.8354 tabaco:ubes
0.8383 carneRoja:nivIngresos
0.8422 deporte:edad
0.8462 edad:nivIngresos
0.8495 deporte:verduras
0.8520 nivEstPad:nivEstPad
0.8542 deporte:deporte
0.8560 nivIngresos:tabaco
0.8586 nivEstPad:tabaco
0.8600 drogas:drogas
0.8612 nivEstudios:nivIngresos
0.8622 carneRoja:tabaco
0.8637 carneRoja:carneRoja
0.8652 deporte:tabaco
0.8660 ubes:verduras
0.8664 nivEstPad:nivIngresos
0.8666 sexo:verduras
0.8667 nivIngresos:nivIngresos
0.8668 nivEstPad:nivEstudios
0.8668 carneRoja:ubes
0.8668 drogas:tabaco
0.8668 deporte:nivEstPad
```

Evaluamos el resultado y vemos que best2 es mejor ajuste.

```
calcR2(datos$val, best1$mod, 'IMC')
calcR2(datos$val, best2$mod, 'IMC')
```

```
> calcR2(datos$val, best1$mod, 'IMC')
# A tibble: 1 x 4
    MSE  varY    R2   aR2
  <dbl> <dbl> <dbl> <dbl>
1  5.70  26.1 0.782 0.779
> calcR2(datos$val, best2$mod, 'IMC')
# A tibble: 1 x 4
    MSE  varY    R2   aR2
  <dbl> <dbl> <dbl> <dbl>
1  3.35  26.1 0.872 0.867
>
```

CONCLUSIÓN

Verdaderamente el software matemático es muy útil y eficaz con el se pueden resolver gran cantidad de problemas y es muy visual, si que es cierto a que debido a mis pocos conocimientos sobre me él me ha sido muy tedioso llevar la practica a cabo y me ha tomado muchas horas de investigación en internet.