Trabajo de modelización en R

Sergio Camacho Marín

Grupo 1ºD

Curso 2022/23

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

Dentro de todas las variables que hay en el .csv que se me ha pasado, he considerado que las variables cualitativas son las siguientes: sexo, dietaEsp, nivEstPad, nicEstudios, nivIngresos. Estas 3 últimas debido a que son escalas y van desde el 0 al 2. Por lo demás, he puesto que por defecto sean doubles. Todo esto se ha establecido con "coltypes" de la siguiente manera:

```
#Cargamos el CSV como tibble, esto
datos <- read_csv('C:/Users/sergi/Desktop/TrabajoR/15312.csv',col_types =
                        cols(
                          .default = col_double(),
                          sexo= col_factor(),
                          dietaEsp= col_factor(),
                          nivEstPad =col_factor(),
                          nivEstudios =col_factor(),
                          nivIngresos = col_factor()
                        ))
```

```
> #Cargamos el CSV como tibble, esto
> datos <- read_csv('C:/Users/sergi/Desktop/TrabajoR/15312.csv',col_types =
+                         cols(
+                           .default = col_double(),
+                           sexo= col_factor(),
+                           dietaEsp= col_factor(),
+                           nivEstPad =col_factor(),
+                           nivEstudios =col_factor(),
+                           nivIngresos = col_factor()
+                         ))
```

Y ahora para saber si es correcto usamos "str":

```
str(datos)
```

```
 .default = col_double(),
 peso = col_double(),
 altura = col_double(),
 sexo = col_factor(levels = NULL, ordered = FALSE, include_na = FALSE),
 edad = col_double(),
 tabaco = col_double(),
 ubes = col_double(),
 carneRoja = col_double(),
 verduras = col_double(),
 deporte = col_double(),
 drogas = col_double(),
 dietaEsp = col_factor(levels = NULL, ordered = FALSE, include_na = FALSE),
 nivEstPad = col_factor(levels = NULL, ordered = FALSE, include_na = FALSE),
 nivEstudios = col_factor(levels = NULL, ordered = FALSE, include_na = FALSE),
 nivIngresos = col_factor(levels = NULL, ordered = FALSE, include_na = FALSE)
```

Apartado 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.

Hay que añadir al final de la tabla un campo que sea relacionado con la fórmula aplicada del apartado, IMC = peso / altura^2. Lo he realizado de la siguiente manera:

```
#Calculamos la columna IMC mediante la fórmula nueva del apartdo y se añade al
#final de la tabla
datos$IMC <- datos$peso / (datos$altura ^ 2)
datos
```

```
> 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>    <fct>     <fct>       <fct>       <dbl>
 1  90.3   1.75 V        43      0     0         1        3       0      3 N        1         2           1            29.5
 2  55.8   1.76 V        63    130     0         3        0       0      0 S        2         3           4            18.0
 3  91.9   1.75 V        26      0     8         1        0       0      0 N        1         3           2            30.0
 4  97.3   1.66 M        33      0     0         0       15       4      0 N        1         0           0            35.3
 5  55.8   1.76 V        41     40     0         8        0       0      0 N        2         2           2            18.0
 6  76.8   1.6  M        29      0     0         4        5       0      0 N        2         3           4            30
 7  52.6   1.71 M        48    170     0         0       26      20      0 N        3         3           3            18.0
 8  91.4   1.74 V        49      0    10         0        7       5      0 N        2         2           2            30.2
 9  81.2   1.6  M        66      0    14         0        0       3      0 N        0         1           0            31.7
10  89.6   1.74 V        18      0     5         0        4       1      0 N        0         0           1            29.6
# … with 4,990 more rows
```

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

Para este apartado hay una función bastante útil que es na.omit(“datos con NA”):

```
#Eliminamos las filas que tengan algún valor NA
datos <-na.omit(datos)
datos
```

```
    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>    <fct>     <fct>       <fct>       <dbl>
 1  90.3   1.75 V        43      0     0         1        3       0      3 N        1         2           1            29.5
 2  55.8   1.76 V        63    130     0         3        0       0      0 S        2         3           4            18.0
 3  91.9   1.75 V        26      0     8         1        0       0      0 N        1         3           2            30.0
 4  97.3   1.66 M        33      0     0         0       15       4      0 N        1         0           0            35.3
 5  55.8   1.76 V        41     40     0         8        0       0      0 N        2         2           2            18.0
 6  76.8   1.6  M        29      0     0         4        5       0      0 N        2         3           4            30
 7  52.6   1.71 M        48    170     0         0       26      20      0 N        3         3           3            18.0
 8  91.4   1.74 V        49      0    10         0        7       5      0 N        2         2           2            30.2
 9  81.2   1.6  M        66      0    14         0        0       3      0 N        0         1           0            31.7
10  89.6   1.74 V        18      0     5         0        4       1      0 N        0         0           1            29.6
# … with 4,940 more rows
```

Y como podemos ver, antes teníamos 4990 filas y ahora tenemos 4949 filas, ha quitado un total de 50 filas de valores NA.

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

Para calcular la desviación típica hay un problema, en R está implementada la cuasivarianza, así que nos haremos una función para aplicarla luego. Empecemos con la media:

```
#Calculamos las medias y desviaciones tipicas de cada una de las variables
#numéricas, y para no mezclar usaremos keep, para que mantenga lo que queremos
#en este caso numericos, no factores  luego con map se aplica a toda los datos,
#la función
medias <- datos %>%  keep(is.numeric) %>% map_dbl(mean)
medias
```

```
> medias <- datos %>%  keep(is.numeric) %>% map_dbl(mean)
> medias
      peso     altura       edad     tabaco       ubes  carneRoja   verduras    deporte     drogas        IMC
76.2577475  1.7002909 40.5935354 18.9151515  4.0820202  1.7911111  5.9567677  4.1684848  0.4745455 26.3331918
```

Y ahora con las desviaciones típicas, la fórmula para la función la tome de unos de los scripts que están al inicio de la estadística en R:

```
#DesvTipicas
desvtip <- function(x){
  return(sqrt(mean(x^2)-mean(x)^2))
}
desv <- datos %>%  keep(is.numeric) %>% map_dbl(desvtip)
desv
```

```
> desv <- datos %>%  keep(is.numeric) %>% map_dbl(desvtip)
> desv
      peso     altura       edad     tabaco       ubes  carneRoja   verduras    deporte     drogas        IMC
20.60415110  0.07066778 14.03916738 40.81363004  5.89357979  2.15078134  7.08349552  4.68794895  1.40016541  6.76277097
```

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

Para realizar esta tarea se ha realizado una función que coge la variable x(12 variables para la regresión, quitamos altura y peso) y variable y(IMC). Esta función nos devolverá los coeficientes de regresión(intercept y pendiente) y el coeficiente de correlación(R2), ambos presente en la función "lm".

```
#Hay que realizar la regresión lineal a las 12 variables
#Calculamos los coeficientes de regresión para cada uno de los valores con respecto al IMC
#Creamos un función que haga la regresión con lm y nos quedamos los coeficientes
#de regresión y de determinación y se tiene que quitar peso y altura, ya que
#ahora está IMC
reg <- function(dt, x, y) {
  temp <- lm(y ~ x, dt)
  list(coeficientes_regresion=coef(temp),
       R2=summary(temp)$r.squared)
}
```

```
> #Hay que realizar la regresión lineal a las 12 variables
> #Calculamos los coeficientes de regresión para cada uno de los valores con respecto al IMC
> #Creamos un función que haga la regresión con lm y nos quedamos los coeficientes
> #de regresión y de determinación y se tiene que quitar peso y altura, ya que
> #ahora está IMC
> reg <- function(dt, x, y) {
+   temp <- lm(y ~ x, dt)
+   list(coeficientes_regresion=coef(temp),
+        R2=summary(temp)$r.squared)
+ }
```

Y ahora mediante una tubería, quitamos primero y segundo de la tabla que corresponde a peso y altura, y aplicamos la función quehemos hecho, luego quito IMC~IMC debido a que es absurdo ponerlo:

```
#Quitamos peso y altura y aplicamos reg a los datos
regresiones <- datos[c(-1:-2)] %>% map(reg,dt=datos, y=datos$IMC)
regresiones <- regresiones[c(-13)]#Quitar IMC~IMC, que no tiene sentido
regresiones
```

```
> regresiones <- datos[c(-1:-2)] %>% map(reg,dt=datos, y=datos$IMC)
> regresiones <- regresiones[c(-13)]#Quitar IMC~IMC, que no tiene sentido
> regresiones
$sexo
$sexo$coeficientes_regresion
(Intercept)          xM
 26.2665396   0.1322357

$sexo$R2
[1] 9.557846e-05
```

```
$edad
$edad$coeficientes_regresion
(Intercept)           x
 25.5141646   0.0201763

$edad$R2
[1] 0.00175435
```

```
$verduras
$verduras$coeficientes_regresion
(Intercept)           x
 24.8099772   0.2557116

$verduras$R2
[1] 0.07173758
```

```
$tabaco
$tabaco$coeficientes_regresion
(Intercept)           x
28.05271011 -0.09090693

$tabaco$R2
[1] 0.3009921
```

```
$deporte
$deporte$coeficientes_regresion
(Intercept)           x
26.73112643 -0.09546266

$deporte$R2
[1] 0.004379086
```

```
$ubes
$ubes$coeficientes_regresion
(Intercept)           x
 24.6864795   0.4034062

$ubes$R2
[1] 0.1235931
```

```
$drogas
$drogas$coeficientes_regresion
(Intercept)           x
 26.3936785  -0.1274625

$drogas$R2
[1] 0.0006964249
```

```
$carneRoja
$carneRoja$coeficientes_regresion
(Intercept)           x
 24.8704822   0.8166493

$carneRoja$R2
[1] 0.06745503
```

```
$dietaEsp
$dietaEsp$coeficientes_regresion
(Intercept)          xS
 26.3248934   0.1762959

$dietaEsp$R2
[1] 3.048221e-05
```

```
$nivEstPad
$nivEstPad$coeficientes_regresion
(Intercept)          x2          x3          x0          x4
 26.4463060  -0.5124606  -1.5974027   0.8688336  -1.5071649

$nivEstPad$R2
[1] 0.01086246
```

```
$nivEstudios
$nivEstudios$coeficientes_regresion
(Intercept)          x3          x0          x1          x4
 26.3688377  -0.8088542   1.6261254   1.3025796  -1.2738901

$nivEstudios$R2
[1] 0.02313314
```

```
$nivIngresos
$nivIngresos$coeficientes_regresion
(Intercept)          x4          x2          x0          x3
 27.2512260  -2.6032631  -0.3406617   1.1989816  -1.7928043

$nivIngresos$R2
[1] 0.03666581
```

Apartado 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 los videos del tema 2 se puede encontrar una función en el archivo de R: "Purr.R", es la que he utilizado para realizar este ejercicio. La primera línea determina como se va a llamar el archivo, la segunda hace un gráfico de puntos(dispersión) y pone los nombres de las variables que se están usando, la tercera pinta una línea que en nuestro caso es la línea de regresión y último es para limpiar.

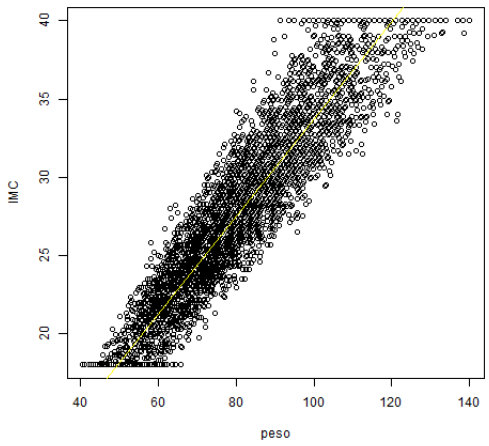
```
dibujarGraficos <- function(nombre) {
  png(file=str_c("IMC~",nombre,".png"))
  plot(y=datos$IMC,datos[[nombre]],xlab=nombre , ylab="IMC")
  abline(lm(datos$IMC ~ datos[[nombre]]), col="yellow")
  dev.off()
}
```

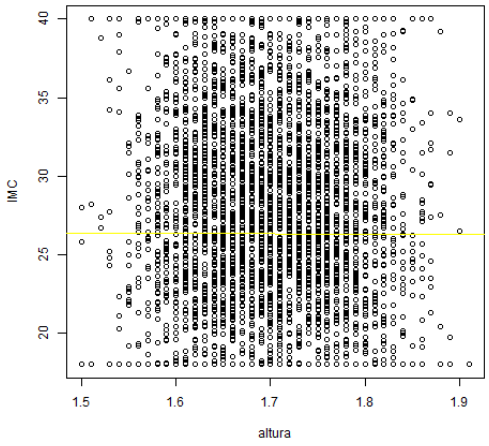
```
> #Mirando en el campus en el archivo de R, "Purr.R", encontre esta manera de
> #realizar los gráficos de dispersión y bowplots, que es una función que
> #hace imágenes de los gráficos y los deja en la carpeta de trabajo.
> dibujarGraficos <- function(nombre) {
+   png(file=str_c("IMC~",nombre,".png"))
+   plot(y=datos$IMC,datos[[nombre]],xlab=nombre , ylab="IMC")
+   abline(lm(datos$IMC ~ datos[[nombre]]), col="yellow")
+   dev.off()
+ }
```

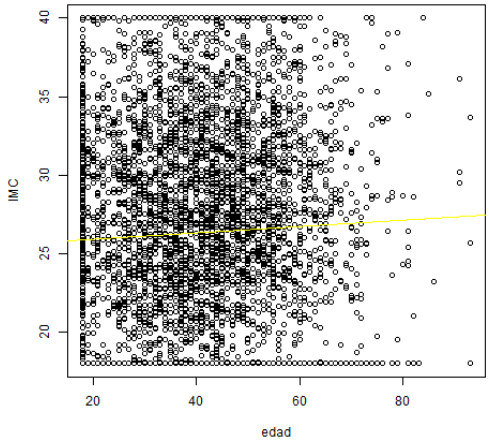
Usando la función de la siguiente manera, tenemos:

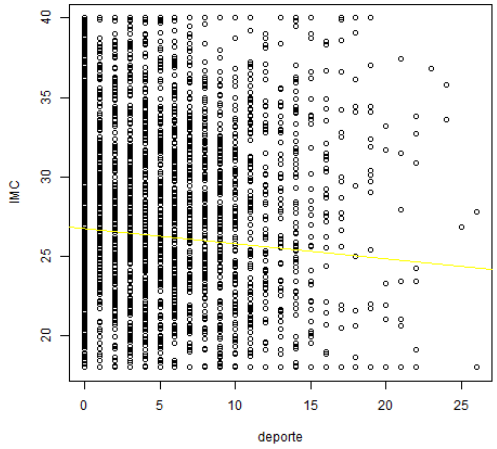
```
#Con esto generamos las gráficas
datos %>% colnames() %>% walk(dibujarGraficos)
```

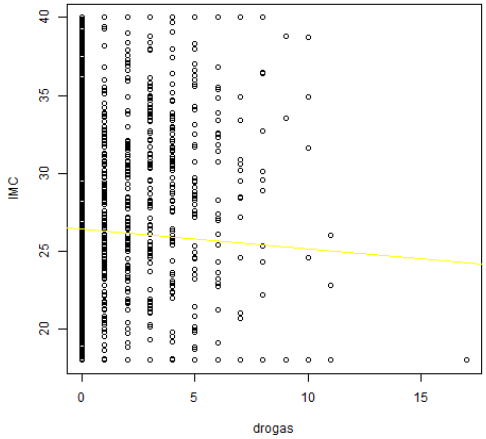
Y produce las siguientes gráficas:

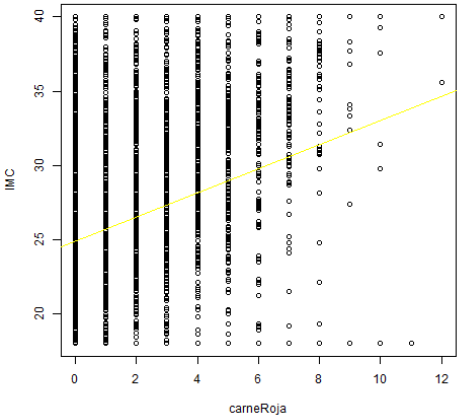

IMC
peso
20
25
30
35
40
40
60
80
100
120
140

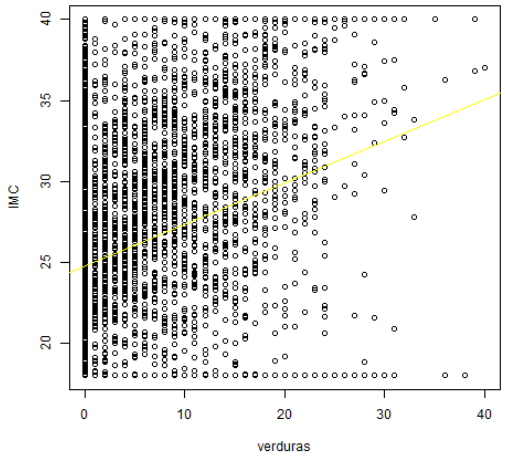

IMC
altura
20
25
30
35
40
1.5
1.6
1.7
1.8
1.9

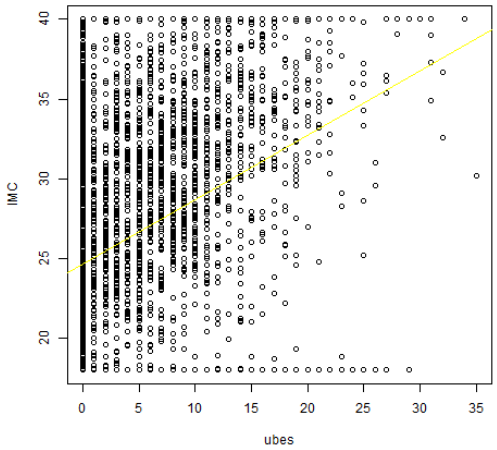

IMC
edad
20
25
30
35
40
20
40
60
80

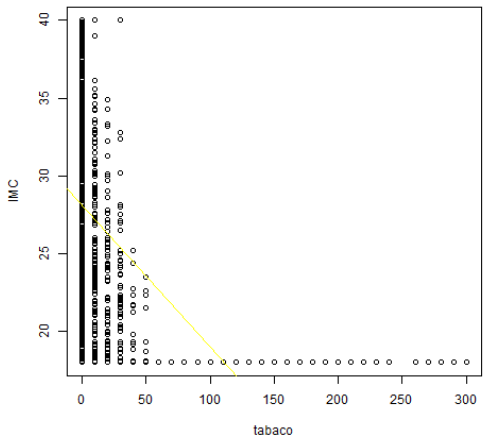

IMC
deporte
20
25
30
35
40
0
5
10
15
20
25

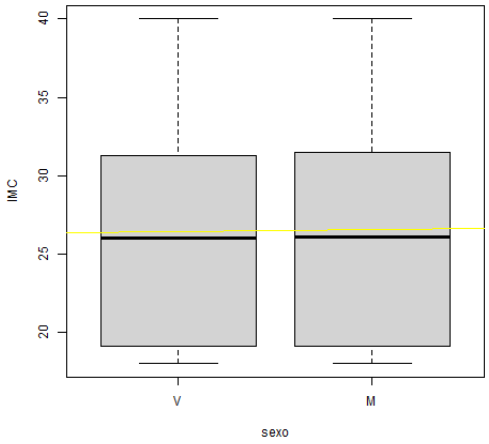

IMC
drogas
20
25
30
35
40
0
5
10
15

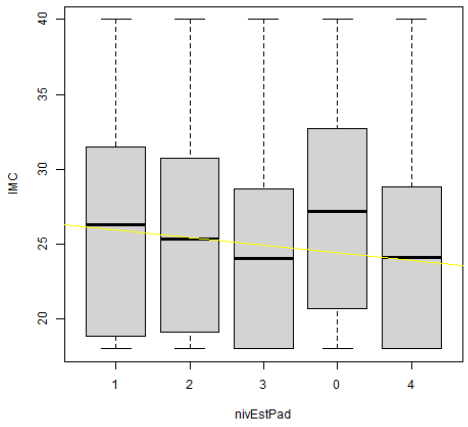

IMC
carneRoja
20
25
30
35
40
0
2
4
6
8
10
12

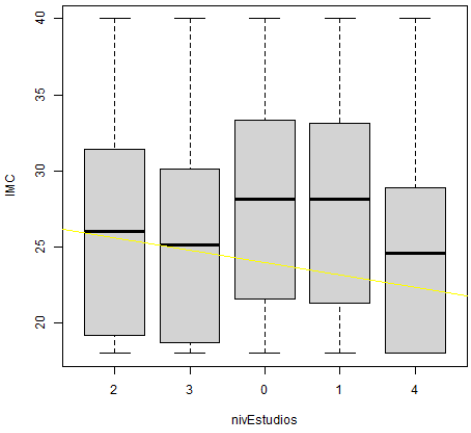

IMC
verduras

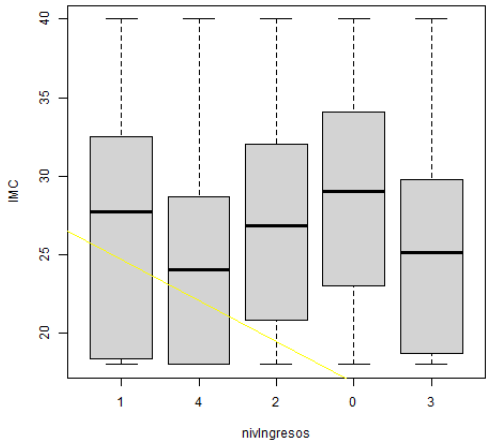

IMC
ubes


IMC
tabaco


IMC
V
M
sexo


IMC
nivEstPad


IMC
nivEstudios



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

En el archivo de R del campus: “S16a_SelecciónModelos_TrainTestSets.R” hay una función ya hecha para separar los conjuntos y para separarlos hay otra en el “S16b_SelecciónModelos_AdjR2ValidSet.R” que te explica como separarlos en 60%, 20% y 20%.

```
#Función auxilar para separar conjuntos, usando la función
#sample de R junto con setdiff para calcular la diferencia
#de conjuntos
separarSets <- function(df, p) {
  rDf <- 1:nrow(df)
  n1 <- sample(rDf, p * length(rDf))
  n2 <- setdiff(rDf, n1)
  list(n1=df[n1,], n2=df[n2,])
}
```

Y ahora tenemos que separar en entrenamiento, test y validación. Por lo que usando la primera vez separarsets con un 0.6, no dará que el primer valor tendrá un 60% que metemos en una variable llamada entrenamiento, y en el segundo valor un 40%, que tendremos que utilizar de nuevo separarsets al 0.5, para terminar de dividirlo en 20% y 20% y asignarlos a test y validación.

```
#Separamos en 60%, 40% y lo almacenamos en conjunto
conjunto <- separarSets(datos, .6)
#El primer valor será de entrenamiento, por lo tanto lo guardamos ahí
entrenamiento <- conjunto$n1
#El segundo valor tendrá el 40% restante, por lo que si separamos a la mitad
#Tendremos 20% y 20%
conjunto <- separarSets(conjunto$n2,.5)
#El primero será de test y el último de validación
test <- conjunto$n1
validación <- conjunto$n2
```

Y la ejecución da lo siguiente:

Entrenamiento

	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>	<fct>	<fct>	<fct>	<dbl>
1	88.3	1.68	V	33	0	0	7	0	0	0	N	2	3	4	31.3
2	91.4	1.83	V	20	0	3	2	3	7	0	S	0	2	1	27.3
3	82.7	1.76	V	50	0	4	1	2	0	0	N	1	3	3	26.7
4	53.2	1.72	M	45	140	0	6	0	0	1	N	1	2	2	18.0
5	51.4	1.69	M	62	60	4	1	4	6	1	N	2	4	3	18.0
6	66.8	1.78	M	59	10	14	1	1	8	0	N	1	1	2	21.1
7	75.7	1.65	M	53	0	9	0	7	6	0	N	2	2	3	27.8
8	55.1	1.75	V	18	80	7	9	0	11	0	N	2	4	4	18.0
9	75.1	1.59	M	65	0	0	4	1	0	0	N	0	0	2	29.7
10	76.2	1.68	M	23	10	9	5	0	0	0	N	2	3	4	27.0

… with 2,960 more rows

Test

	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>	<fct>	<fct>	<fct>	<dbl>
1	56.4	1.77	V	18	90	4	0	4	3	0	N	3	4	4	18.0
2	43.8	1.56	M	64	240	0	1	2	0	5	N	3	2	4	18.0
3	112.	1.67	M	30	0	5	4	15	0	0	N	2	3	2	40.0
4	95.4	1.67	V	48	0	0	3	22	10	0	N	1	3	4	34.2
5	83.0	1.61	M	53	0	0	4	0	0	4	N	1	1	0	32.0
6	110.	1.66	M	30	0	12	0	29	13	0	N	0	1	2	40.0
7	104.	1.76	V	42	0	11	1	2	0	0	N	3	4	2	33.5
8	69.4	1.76	V	18	0	0	1	0	0	0	N	1	2	3	22.4
9	77.6	1.65	M	48	0	8	1	2	4	0	N	3	3	3	28.5
10	65.4	1.64	M	42	0	5	0	1	5	1	N	0	0	1	24.3

… with 980 more rows

Validación

	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>	<fct>	<fct>	<fct>	<dbl>
1	76.8	1.6	M	29	0	0	4	5	0	0	N	2	3	4	30
2	89.6	1.74	V	18	0	5	0	4	1	0	N	0	0	1	29.6
3	59.2	1.76	V	36	0	0	0	0	4	0	N	2	3	3	19.1
4	81.5	1.76	V	40	0	0	0	8	5	0	N	1	2	2	26.3
5	59.6	1.82	V	36	20	0	0	0	0	8	S	2	3	1	18.0
6	82.5	1.85	V	23	0	3	2	0	1	0	N	2	4	4	24.1
7	49.4	1.63	M	18	20	0	0	8	0	0	N	1	1	2	18.6
8	68.0	1.63	M	54	0	0	3	6	13	0	N	2	2	0	25.6
9	109.	1.82	V	45	0	0	4	11	7	0	N	0	2	3	32.9
10	91.1	1.76	V	51	10	0	7	1	0	0	N	0	2	2	29.4

… with 980 more rows

Apartado 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.

Para este apartado si reutilizamos la función “reg” se nos facilita el trabajo, calculamos las regeresiones de las 12 variables, se nos cuela IMC, pero no cuenta.

```
#Para analizar cual de las 12 variables explica mejor la variable IMC en los conjuntos de entrenamiento
#y test, podemos usar la función reg previamente definida
regresionesEntrenamiento <- entrenamiento[c(-1:-2)] %>% map(reg,dt=entrenamiento, y=entrenamiento$IMC)
regresionesTest <- test[c(-1:-2)] %>% map(reg,dt=test, y=test$IMC)
```

Luego de esto debemos ordenar y visualizar los modelos que mejor R2 obtienen con respecto a la variable IMC:

```
#Ordenamos la lista por los valores de R2 de forma creciente
regresionesEntrenamiento <- regresionesEntrenamiento[order(sapply(regresionesEntrenamiento, `[[`, i = "R2"))]
regresionesTest <-regresionesTest[order(sapply(regresionesTest, `[[`, i = "R2"))]
#Con los conjuntos que hemos creado, obtenemos que el tabaco presentan la mayor correlación
#linear con el IMC dentro del conjunto Entrenamiento, con un 0.3092068 como coeficiente de determinación
#Asi mismo, el tabaco es el que presentan la mayor correlación dentro del conjunto de test
#con 0.280177 como coeficiente de determinación
```

Los 5 mejores modelos para determinar para predecir el valor de IMC son los siguientes:

1. Tabaco
2. Ubes
3. Verduras
4. CarneRoja
5. NivIngresos

```
$ubes$coeficientes_regresion
(Intercept)           x
 24.6128109   0.4564059

$ubes$R2
[1] 0.153336

$tabaco
$tabaco$coeficientes_regresion
(Intercept)           x
27.93901313 -0.08724706

$tabaco$R2
[1] 0.2801077
```

```
$nivIngresos
$nivIngresos$coeficientes_regresion
(Intercept)          x4          x2          x0          x3
 27.0712735  -2.3776446   0.1083124   1.4676934  -1.3951744

$nivIngresos$R2
[1] 0.03693948

$carneRoja
$carneRoja$coeficientes_regresion
(Intercept)           x
 24.9684806   0.8203992

$carneRoja$R2
[1] 0.0683961

$verduras
$verduras$coeficientes_regresion
(Intercept)           x
 24.8822758   0.2529749

$verduras$R2
[1] 0.07679322
```

Apartado 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.

Para este último punto, me he fijado bastante en el script que se presenta en el campus para poder realizar este apartado. El script en cuestión es "S16c_SelecciónModelos_ModeloFinal.R".

He usado diversas funciones que se necesitan:

La función para realizar ajustes lineales múltiples para múltiples x:

```
#Función para realizar ajustes lineales múltiples
linearAdjust <- function(df, y, x) {
  lm(str_c(y, "~", str_c(x, collapse="+")), df)
}
```

Otra función para que devuelva el coeficiente de determinación calculado sobre un dataframe:

```
calcR2A <- function(df, mod) {
  R2 <- summary(mod)$r.squared
  1 - (1- R2) * (nrow(df) - 1) / (nrow(df) - mod$rank)
}
```

La función de ajuste lineal que da el coeficiente de determinación, que utiliza la función anterior:

```
calcModR2 <- function(dfTrain, dfTest, y, x) {
  mod <- linearAdjust(dfTrain, y, x)
  calcR2A(dfTest, mod)
}
```

Y por último, la función que calcula el mejor ajuste lineal:

```
#Función para buscar el mejor ajuste lineal
MejorAjuste <- function(dfTrain, dfTest, varPos) {
  #Inicializamos las variables para el bucle
  bestVars <- character(0)
  aR2      <- 0

  repeat {
    #Bloque de código a repetir hasta que se produzca el break,
    #puesto que no queremos empeorar el modelo actual

    aR2v <- map_dbl(varPos, ~calcModR2(dfTrain, dfTest, "IMC", c(bestVars, .)))
    i    <- which.max(aR2v)
    aR2M <- aR2v[i]

    if (aR2M <= aR2) break

    aR2 <- aR2M
    bestVars <- c(bestVars, varPos[i])
    varPos   <- varPos[-i]
  }

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

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

La función del mejor ajuste lineal busca el mejor ajuste lineal para todo el modelo de manera que si empeora el modelo salga del mismo para evitar empeorarlo. Para ello va cogiendo el mejor coeficiente de cada variable y lo añade al ajuste, si es peor que el anterior no introduce la variable, sino continua y sigue metiendo los mejores ajustes.

Ejecutamos, (quito peso y altura debido a que con IMC ya tiene bastante):

```
#Calculamos el mejor ajuste
mAjuste <-MejorAjuste(entrenamiento,test,names(datos)[-1:-2])
```

Y en mi caso, ocurre el siguiente ajuste, está compuesto de 8 variables y son las siguientes:

```
> mAjuste <-MejorAjuste(entrenamiento,test,names(datos)[-1:-2])
There were 18 warnings (use warnings() to see them)
> mAjuste
$vars
[1] "tabaco"      "ubes"        "verduras"    "carneRoja"   "deporte"     "nivIngresos" "edad"        "drogas"

$mod

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

Coefficients:
  (Intercept)        tabaco          ubes      verduras     carneRoja       deporte  nivIngresos4  nivIngresos2  nivIngresos0
     24.47503      -0.09272       0.37870       0.50264       1.02137      -0.48722      -3.11215      -1.34891       1.21312
nivIngresos3          edad        drogas
     -2.05325       0.01451       0.13097
```

Apartado 10: Evalúa el resultado en el conjunto de validación.

Por último comprobamos el modelo con el conjunto de validación y obtenemos como resultado lo siguiente:

```
#Obtenemos que las variables de tabaco, ubes, verduras, carneRoja, deporte,
#nivIngresos, edad, drogas componen el mejor ajuste(8 variables)
#comprobamos los resultados obtenidos con el conjunto de validación
modFinal <- lm(str_c("IMC", " ~ ", str_c(mAjuste$vars,collapse = " + ")) , validación)
calcR2A(validación,modFinal)
```

```
> modFinal <- lm(str_c("IMC", " ~ ", str_c(mAjuste$vars,collapse = " + ")) , validación)
> calcR2A(validación,modFinal)
[1] 0.6941491
```

```
> modFinal <- lm(str_c("IMC", " ~ ", str_c(mAjuste$vars,collapse = " + ")) , validación)
> modFinal

Call:
lm(formula = str_c("IMC", " ~ ", str_c(mAjuste$vars, collapse = " + ")),
    data = validación)

Coefficients:
  (Intercept)        tabaco          ubes      verduras     carneRoja       deporte  nivIngresos4  nivIngresos2  nivIngresos0
     23.27689      -0.08985       0.41306       0.49888       1.05508      -0.49568      -2.16566       0.01480       0.74033
nivIngresos3          edad        drogas
     -1.37633       0.01911       0.03093
```

Apartado 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.

Primero se carga como tibble los nuevos datos de entrada del fichero "eval.csv" y luego se utiliza na.omit() por si acaso hay algún dato NA, para que no moleste en la predicción. Posterior a esto se utiliza la función predict y se le pasa el modelo y el nuevo dataframe, con eso ya tenemos el IMC. Para calcular el peso, hacemos la fórmula despejando en función del peso y guardamos el csv con write.csv.

```
#Cargamos el CSV como tibble, esto
datosEval <- read_csv('C:/Users/sergi/Desktop/TrabajoR/eval.csv',col_types =
                      cols(
                        .default = col_double(),
                        sexo= col_factor(),
                        dietaEsp= col_factor(),
                        nivEstPad =col_factor(),
                        nivEstudios =col_factor(),
                        nivIngresos = col_factor()
                      ))
str(datosEval)
datosEval<- na.omit(datosEval)
#Le calculo la columna IMC con los datos que tiene el eval.csv
datosEval$IMC <- predict(object = modFinal, newdata = datosEval)
datosEval
#Para meter peso, lo suyo es hacer la conversión con IMC*(altura^2) = peso
datosEval$peso <- datosEval$IMC * (datosEval$altura^2)
datosEval
#Y ahora falta guardar el nuevo eval como evalX.csv
write.csv(datosEval, file = "evalX.csv")
```

```
# 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>    <fct>     <fct>       <fct>       <dbl> <dbl>
 1 M       1.63    18      0     0         0        5       0      2 S        0         1           2           26.2  69.6
 2 V       1.68    35      0     1         1        6       0      0 N        1         3           2           28.4  80.2
 3 V       1.77    39     30    13         0        0       6      0 N        2         3           1           23.7  74.3
 4 M       1.59    57     60     4         4       16      15      0 N        2         2           2           25.4  64.2
 5 M       1.58    18      0     3         1        0       7      0 N        2         4           4           20.3  50.6
 6 M       1.66    35    160     0         0        0       4      0 N        0         2           3            6.21 17.1
 7 V       1.75    46    140     3         1        0       0      0 S        0         1           0           14.6  44.7
 8 M       1.73    54      0     0         0        9       4      0 N        0         0           1           26.8  80.3
 9 V       1.67    18      0     0         0       11       0      0 S        1         2           3           27.7  77.3
10 V       1.74    53      0     3         1        4       0      0 N        1         2           1           28.6  86.5
# … with 990 more rows
# ℹ Use `print(n = ...)` to see more rows
> #Y ahora falta guardar el nuevo eval como evalX.csv
> write.csv(datosEval, file = "evalX.csv")
```

Apartado 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.

  El modelo que he construido tiene bastante utilidad con respecto a definir uno de los parámetros de la tabla como es el IMC o dos parámetros, debido a que está relacionado con la altura y el peso, por lo tanto teniendo una de ellas y las otras variables también se puede sacar.

- Que se puede deducir a partir del modelo sobre la relación entre las variables.

  El modelo está compuesto de 8 variables de las cuáles el tabaco y el deporte son negativos lo que quiere decir que reducen el IMC, lo cuál no veo sentido en relación al deporte(a menos que sea ejercicio de ganancia muscular). Por otro lado si veo sentido que las ubes suban este índice junto con la carne roja, pero tampoco veo el sentido que aumente el IMC por el consumo de verduras.

- Problemas que has encontrado en el desarrollo.

  El uso de tuberías me ha matado, aunque suene coloquial, debido a que se me ha hecho bastante complicado entenderlas y usarlas al principio, haciendo scripts del todo demoledores para la vista y sin reutilizar código.

- Qué te ha llamado la atención en el proceso.

  El hecho de tener que descomponer en varios conjuntos para poder realizar un modelo que sea más o menos preciso, debido a que en mi caso es de 0.69 que está bastante bien, aunque no es lo suficiente preciso.

- Qué más podría hacerse y cómo plantearlo.

  Desde mi punto de vista, hay varios factores que faltan para calcular el IMC, como puede ser trastornos mentales, tiempo de sueño, productos consumidos(índice calórico total) o deporte intenso(como ejercicios que requieran mover objetos pesados, estilo halterofilia). Aunque esto podría derivar en un modelo demasiado complejo, se poodrían sustituir algunas de las variables como los niveles por unas de las que mencioné anteriormente.