Proyecto FMAD

ICAI. Máster en Big Data. Fundamentos Matemáticos del Análisis de Datos (FMAD).

Curso 2021-22. Última actualización: 2021-10-10

Índice

Introducción

Cargamos las librerías

```
library(tidyverse)
library(lubridate)
library(caret)
library(grid)
library(gridExtra)
```

Leemos los datos

```
datos <- read.csv("marketing_campaign.csv", header = TRUE, sep = "")
```

Definición de las variables

- **ID**: El ID del cliente.
- **Year_Birth:** Año de nacimiento.
- **Education:** Nivel de educación del cliente.
- **Marital_Status:** Estado civil del cliente.
- **Income:** Ingreso familiar anual del cliente.
- **Kidhome:** Número de niños pequeños en casa del cliente.
- **Teenhome:** Número de adolescentes en el hogar del cliente.
- **Dt_Customer:** Fecha de inscripción del cliente en la empresa.
- **Recency:** Número de días desde la última compra.
- **MntWines:** Gasto en productos vitivinícolas en los últimos 2 años.
- **MntGoldProds:** Gasto en productos premium en los últimos 2 años.
- **NumDealsPurchases:** Número de compras con uso de descuento.
- **NumWebPurchases:** Número de compras a través de la web.
- **NumCatalogPurchases:** Número de compras usando catalogo.
- **NumWebVisitsMonth:** Visitas por mes a la web.
- **AcceptedCmp1:** 1 si el cliente acepta la oferta en la 1ra campaña, 0 si no lo acepta.
- **AcceptedCmp2:** 1 si el cliente acepta la oferta en la 2nd campaña, 0 si no lo acepta.
- **Complain:** 1 si el cliente se ha quejado en los dos últimos años.
- **Z_CostContact:** Coste de contactar con cliente.
- **Z_Revenue:** Ingresos/Beneficios después de que el cliente acepte la campaña
- **Response:** 1 si el cliente acepta la oferta en la última campaña y 0 si no la acepta.

TODO Resumen de datos

```
cat(cat(cat(cat("El conjunto de datos tiene", nrow(datos)), "filas y"), ncol(datos)), "columnas")
```

```
## El conjunto de datos tiene 2440 filas y 29 columnas
```

```
str(datos)
```

```
## 'data.frame':    2440 obs. of  29 variables:
##  $ ID                 : int  5524 2174 4141 6182 5324 7446 965 6177 4855 5899 ...
##  $ Year_Birth         : int  1957 1954 1965 1984 1981 1967 1971 1985 1974 1950 ...
##  $ Education          : chr  "Graduation" "Graduation" "Graduation" "Graduation" ...
##  $ Marital_Status     : chr  "Single" "Single" "Together" "Together" ...
##  $ Income             : chr  "58138" "46344" "71613" "26646" ...
##  $ Kidhome            : int  0 1 0 1 1 0 0 1 1 1 ...
##  $ Teenhome           : chr  "0" "1" "0" "0" ...
##  $ Dt_Customer        : chr  "04-09-2012" "08-03-2014" "21-08-2013" "10-02-2014" ...
##  $ Recency            : chr  "58" "38" "26" "26" ...
##  $ MntWines           : int  635 11 426 11 173 520 235 76 14 28 ...
##  $ MntFruits          : int  88 1 49 4 43 42 65 10 0 0 ...
##  $ MntMeatProducts    : int  546 6 127 20 118 98 164 56 24 6 ...
##  $ MntFishProducts    : int  172 2 111 10 46 0 50 3 3 1 ...
##  $ MntSweetProducts   : int  88 1 21 3 27 42 49 1 3 1 ...
##  $ MntGoldProds       : int  88 6 42 5 15 14 27 23 2 13 ...
##  $ NumDealsPurchases  : int  3 2 1 2 5 2 4 2 1 1 ...
##  $ NumWebPurchases    : int  8 1 8 2 5 6 7 4 3 1 ...
##  $ NumCatalogPurchases: int  10 1 2 0 3 4 3 0 0 0 ...
##  $ NumStorePurchases  : int  4 2 10 4 6 10 7 4 2 0 ...
##  $ NumWebVisitsMonth  : int  7 5 4 6 5 6 6 8 9 20 ...
##  $ AcceptedCmp3       : int  0 0 0 0 0 0 0 0 0 1 ...
##  $ AcceptedCmp4       : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ AcceptedCmp5       : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ AcceptedCmp1       : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ AcceptedCmp2       : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ Complain           : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ Z_CostContact      : int  3 3 3 3 3 3 3 3 3 3 ...
##  $ Z_Revenue          : int  11 11 11 11 11 11 11 11 11 11 ...
##  $ Response           : int  1 0 0 0 0 0 0 0 1 0 ...
```

```
head(datos)
```

```
##     ID Year_Birth  Education Marital_Status Income Kidhome Teenhome Dt_Customer
## 1 5524       1957 Graduation         Single  58138       0        0  04-09-2012
## 2 2174       1954 Graduation         Single  46344       1        1  08-03-2014
## 3 4141       1965 Graduation       Together  71613       0        0  21-08-2013
## 4 6182       1984 Graduation       Together  26646       1        0  10-02-2014
```

```
## 5 5324       1981        PhD        Married  58293       1        0  19-01-2014
## 6 7446       1967     Master       Together  62513       0        1  09-09-2013
##   Recency MntWines MntFruits MntMeatProducts MntFishProducts MntSweetProducts
## 1      58      635        88             546             172               88
## 2      38       11         1               6               2                1
## 3      26      426        49             127             111               21
## 4      26       11         4              20              10                3
## 5      94      173        43             118              46               27
## 6      16      520        42              98               0               42
##   MntGoldProds NumDealsPurchases NumWebPurchases NumCatalogPurchases
## 1           88                 3               8                  10
## 2            6                 2               1                   1
## 3           42                 1               8                   2
## 4            5                 2               2                   0
## 5           15                 5               5                   3
## 6           14                 2               6                   4
##   NumStorePurchases NumWebVisitsMonth AcceptedCmp3 AcceptedCmp4 AcceptedCmp5
## 1                 4                 7            0            0            0
## 2                 2                 5            0            0            0
## 3                10                 4            0            0            0
## 4                 4                 6            0            0            0
## 5                 6                 5            0            0            0
## 6                10                 6            0            0            0
##   AcceptedCmp1 AcceptedCmp2 Complain Z_CostContact Z_Revenue Response
## 1            0            0        0             3        11        1
## 2            0            0        0             3        11        0
## 3            0            0        0             3        11        0
## 4            0            0        0             3        11        0
## 5            0            0        0             3        11        0
## 6            0            0        0             3        11        0
```

```
cat(cat("Existen", sum(is.na(datos))), "valores nulos")
```

```
## Existen 4421 valores nulos
```

TODO Preprocesamiento de los datos

Antes de todo yo diria que habría que eliminar todos los valores nulos ya que hay bastantes y revisar que los datos estan dentro de su variable ya que en varios casos que he visto hay datos de fechas que estan dentro de una variable de tipo int

En un primer momento hemos eliminado las filas de los datos nulos. Además, tras observar que había algunos datos erróneos y tener suficientes clientes hemos eliminado las filas de dichos datos. Y cambiamos algún tipo de columna que esté mal

```
datos <- na.omit(datos)
datos <- datos %>%
  filter(ID != 0 & ID != 1 & Education != "2n" & Income > 10 & Income != "2")
```

En una segunda fase del preprocesado hemos cambiado algunas columnas mal tipadas

```
datos$Teenhome <- as.numeric(datos$Teenhome)
datos$Income <- as.numeric(datos$Income)
datos$Recency <- as.numeric(datos$Recency)
```

Ya que el conjunto de datos tiene muchas columnas, vamos a eliminar algunas que no nos resultan muy interesantes: “NumDealsPurchases”, “Receny”, “AcceptedCmp1”, “AcceptedCmp2”, “AcceptedCmp3”, “AcceptedCmp4” y “AcceptedCmp5”, “Z_CostContact” y “Z_Revenue”.

```
datos <- datos %>%
  select(-c(AcceptedCmp3:AcceptedCmp2), -NumDealsPurchases, -Recency,
         -Z_CostContact, -Z_Revenue)
```

En el dataset original en vez de aparecer las edades aparecen los años de nacimiento de los clientes. Para un procesamiento mejor y más útil realizaremos un mutate a la tabla con el fin de generar una nueva columna formada por la edad de los clientes.

```
datos <- datos %>%
  mutate(edad = 2021 - Year_Birth) %>%
  select(-Year_Birth)
```

Sumaremos el total de niños de cada cliente agrupando las columnas Kidhome y Teenhome

```
datos <- datos %>%
  mutate(totalHijos = Kidhome + Teenhome) %>%
  select(-Kidhome, -Teenhome)
```

Cogemos las columnas “NumWebPurchases”, “NumCatalogPurchases” y “NumStorePurchases” que indican el numero de compras hechas en cada sitio, en tiendas, por catalogo y por la web y las sumaría todas en una única columna que indique el total de compras que ha realizado el cliente.

```
datos <- datos %>%
  rowwise(ID) %>%
  mutate(suma_compras = sum(c(NumWebPurchases, NumCatalogPurchases,
                              NumStorePurchases)))
```

Sumamos las columans: “MntWines”, “MntFruits”, “MntMeatProducts”, “MntFishProducts”, “MntSweetProducts”, “MntGoldProds” para saber cuánto dinero se ha gastado un cliente en total.

```
datos <- datos %>%
  rowwise(ID) %>%
  mutate(Dinero_Gastado = sum(c(MntWines, MntFruits, MntMeatProducts, MntFishProducts,
                                MntSweetProducts, MntGoldProds)))
```

Agrupamos el estado civil de cada cliente y lo simplificamos para ver si vive solo o en pareja

```
datos <- datos %>%
  mutate(Marital_Status = factor(Marital_Status== "Single" | Marital_Status== "Divorced",
                                 levels = c(TRUE, FALSE), labels = c('Single','Not single')))
```

Cambiamos la columna Dt_Customer y establecemos 4 grupos que representan la longevidad del cliente en la empresa

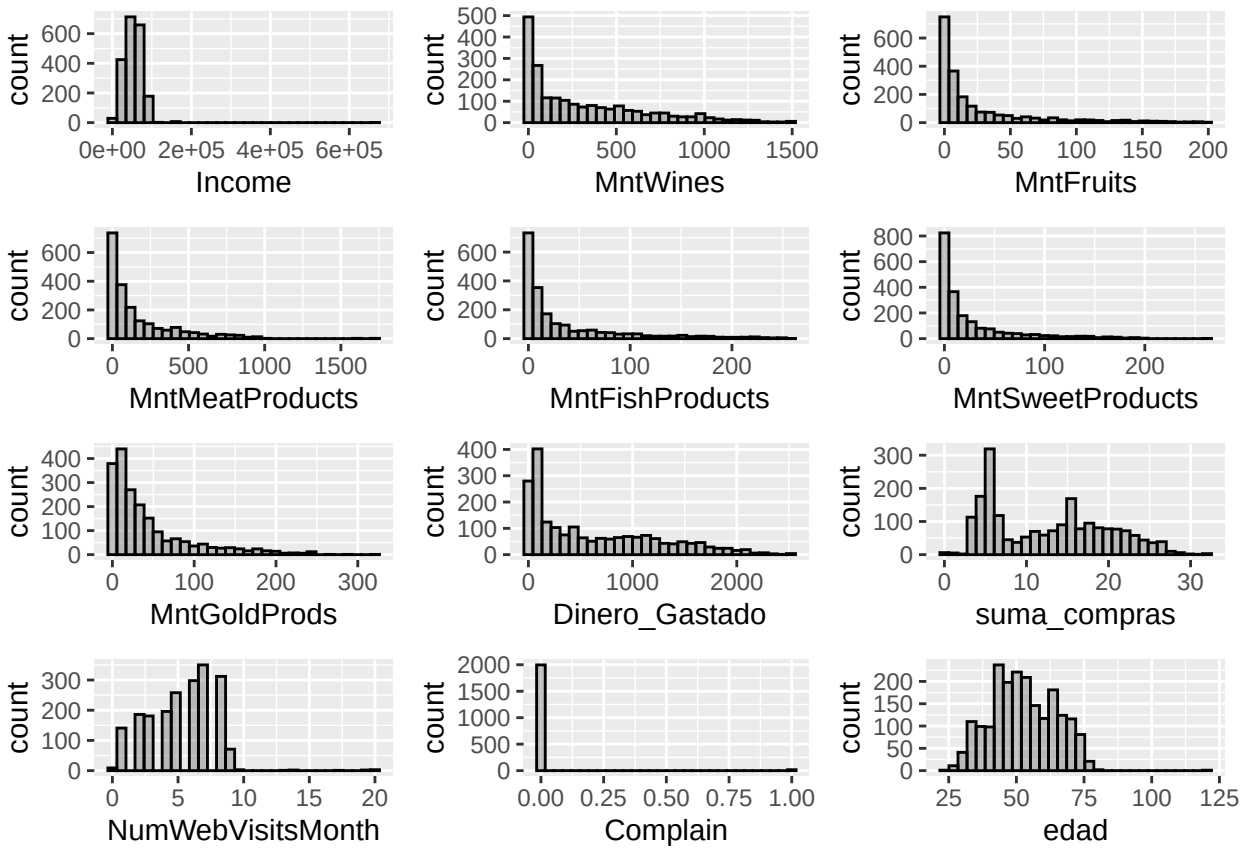
```
datos$Dt_Customer = as.Date(datos$Dt_Customer, "%d-%m-%Y")
fechaMinima = min(datos$Dt_Customer)
datos$Dt_Customer <-  factor(cut_number(as.duration(datos$Dt_Customer-fechaMinima),
                      n = 3), labels = c("Nuevo", "Con Experiencia", "Muy Antiguo"),
                      ordered = TRUE)
```

TODO Visualización de los datos

En este apartado se realizará el análisis mediante gráficas de las variables según su tipo, viendo si siguen una distribución normal y si presentarían outliers entre otros factores.

Estas son las variables continuas

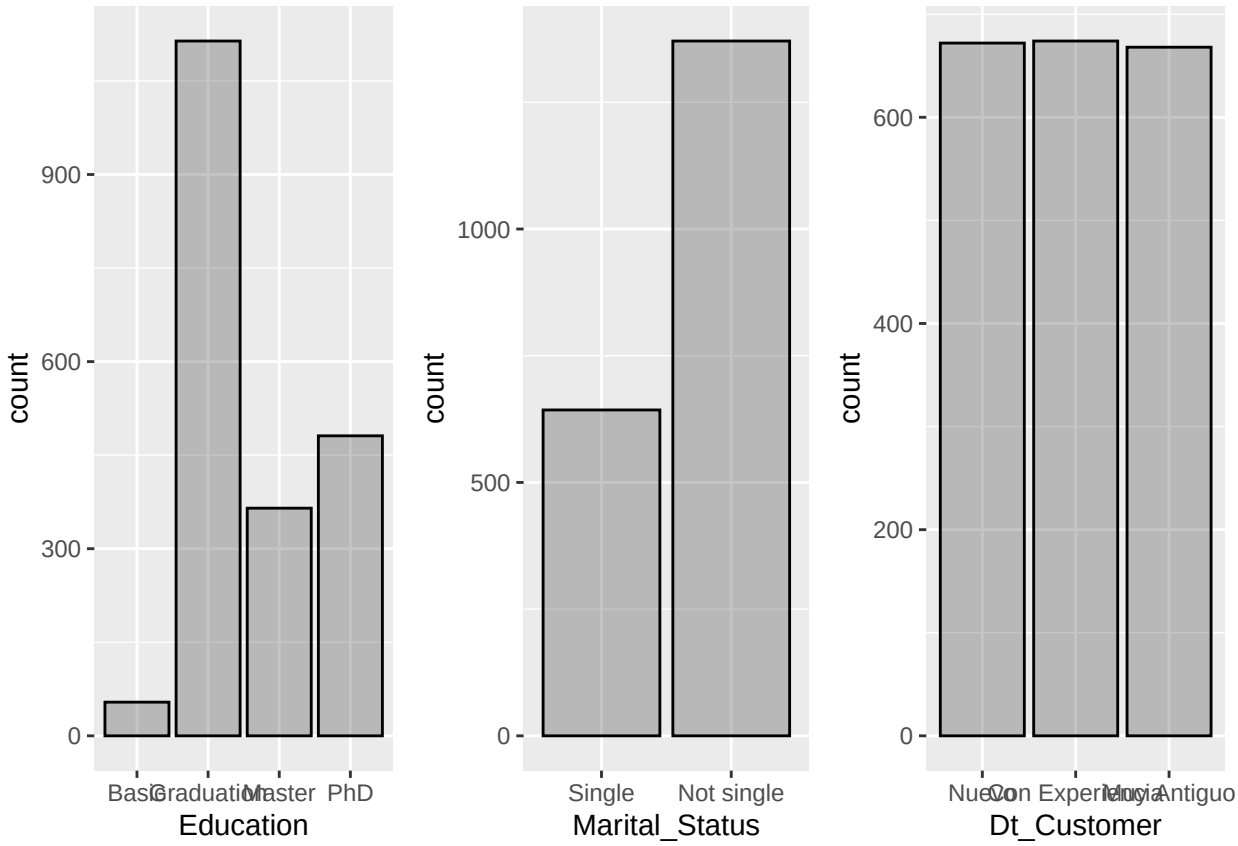
```
h1 <- ggplot(datos)+
  geom_histogram(aes(x = Income), color = "black", alpha = 0.35)
h2 <- ggplot(datos)+
  geom_histogram(aes(x = MntWines ), color = "black", alpha = 0.35)
h3 <- ggplot(datos)+
  geom_histogram(aes(x = MntFruits ), color = "black", alpha = 0.35)
h4 <- ggplot(datos)+
  geom_histogram(aes(x = MntMeatProducts ), color = "black", alpha = 0.35)
h5 <- ggplot(datos)+
  geom_histogram(aes(x = MntFishProducts ), color = "black", alpha = 0.35)
h6 <- ggplot(datos)+
  geom_histogram(aes(x = MntSweetProducts ), color = "black", alpha = 0.35)
h7 <- ggplot(datos) +
  geom_histogram(aes(x = MntGoldProds ), color = "black", alpha = 0.35)
h8 <- ggplot(datos) +
  geom_histogram(aes(x = Dinero_Gastado), color = "black", alpha = 0.35)
h9 <- ggplot(datos)+
  geom_histogram(aes(x = suma_compras ), color = "black", alpha = 0.35)
h10 <- ggplot(datos)+
  geom_histogram(aes(x = NumWebVisitsMonth ), color = "black", alpha = 0.35)
h11 <- ggplot(datos)+
  geom_histogram(aes(x = Complain ), color = "black", alpha = 0.35)
h12 <- ggplot(datos)+
  geom_histogram(aes(x = edad ), color = "black", alpha = 0.35)
grid.arrange(h1,h2,h3,h4,h5,h6,h7,h8,h9,h10,h11,h12)
```



Estas son las varibles discretas

```
hb1 <- ggplot(datos)+
  geom_bar(aes(x = Education), color = "black", alpha = 0.35)
hb2 <- ggplot(datos) +
  geom_bar(aes(x = Marital_Status), color = "black", alpha = 0.35)
hb3 <- ggplot(datos) +
  geom_bar(aes(x = Dt_Customer), color = "black", alpha = 0.35)

grid.arrange(hb1, hb2, hb3, ncol = 3)
```



TODO Buscar la relación posible entre distintas variable y realizr modelos de ML

En esta sección usaremos técnicas de Machine Learning tomando distintos outputs.

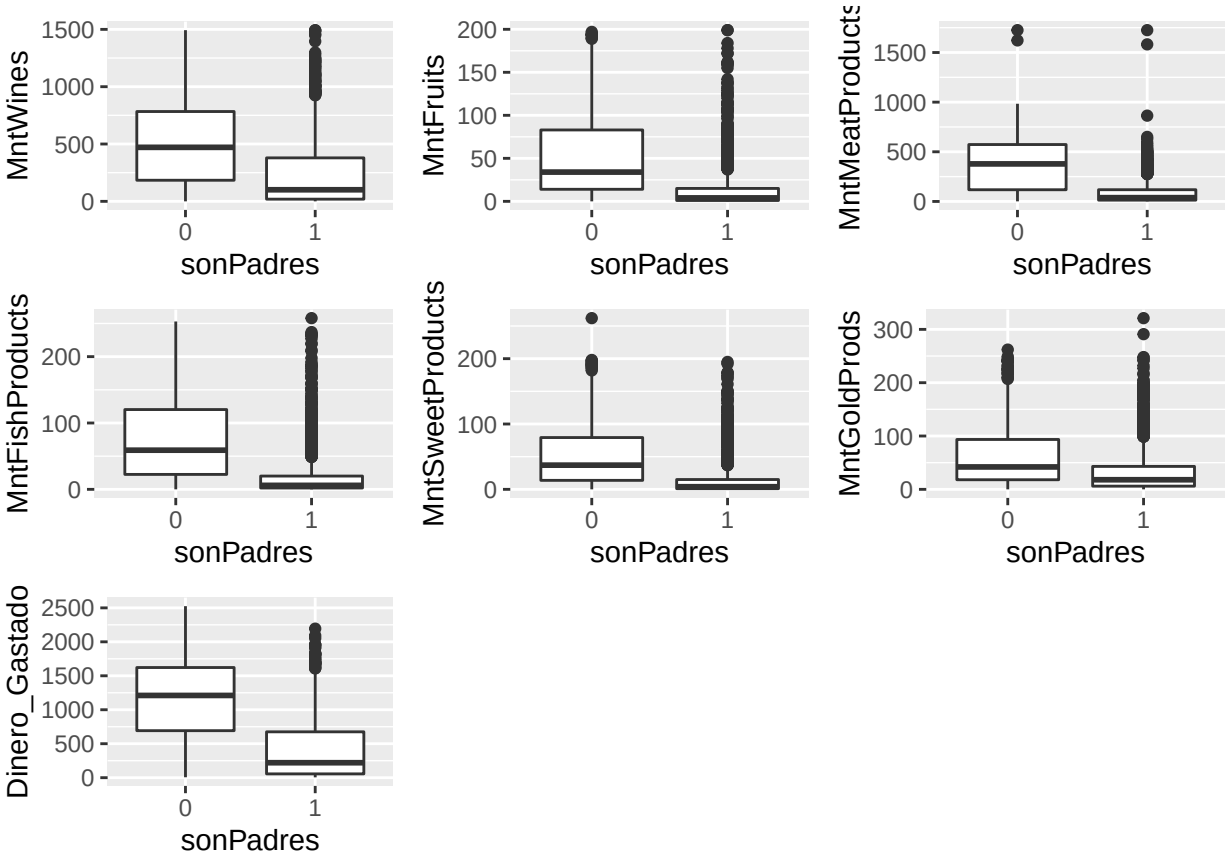
En un primer momento hemos analizado la relación entre tener hijos/el número de hijos y los gastos en compras de cada tipo.

```
datos_ML<-datos %>%
  mutate(sonPadres = factor(totalHijos>0, levels = c(FALSE, TRUE), labels = c(0,1)))
borrar <- c("MntWines", "MntFruits", "MntMeatProducts",  "MntFishProducts", "MntSweetProducts", "MntGol
datos_ML <- datos_ML[(names(datos_ML) %in% borrar)]
head(datos_ML)
```

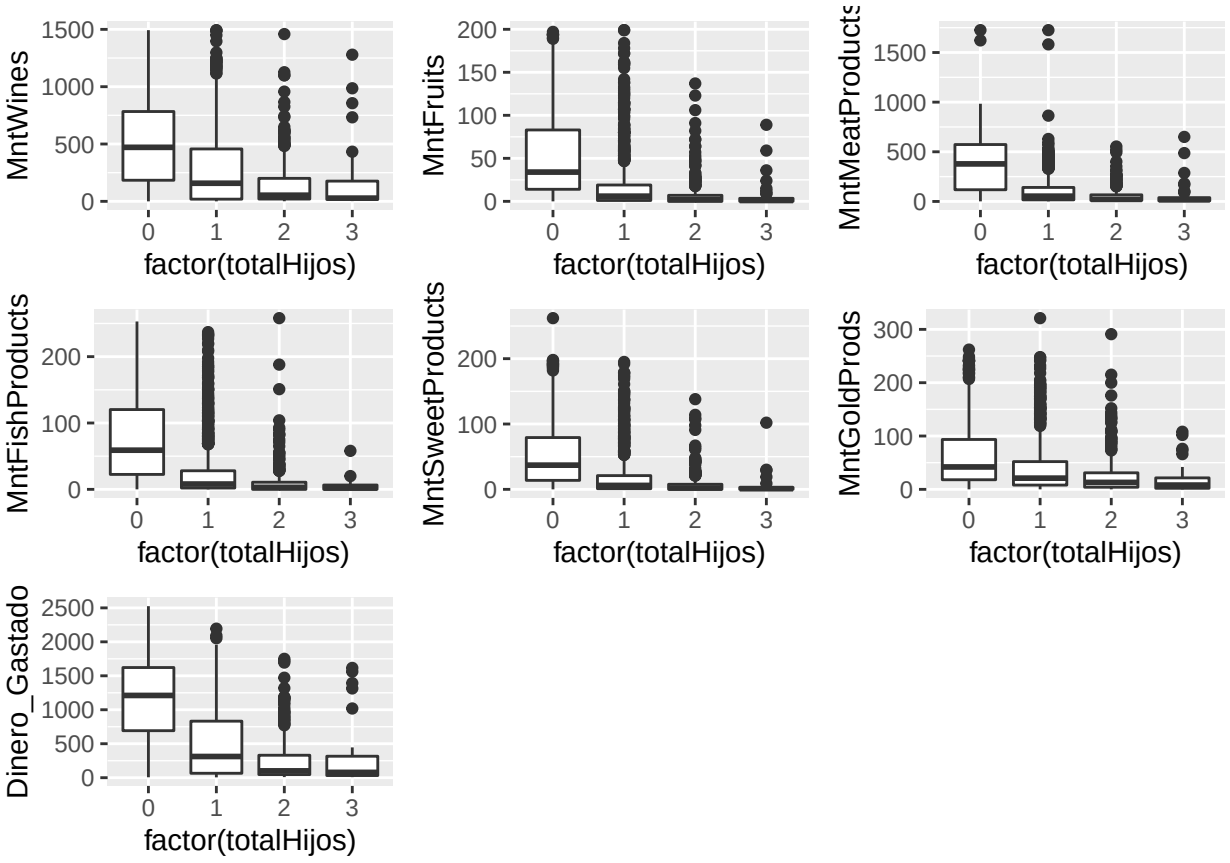
```
## # A tibble: 6 x 9
## # Rowwise:
##   MntWines MntFruits MntMeatProducts MntFishProducts MntSweetProducts
##      <int>     <int>           <int>            <int>             <int>
## 1      635        88             546              172                88
## 2       11         1               6                2                 1
## 3      426        49             127              111                21
## 4       11         4              20               10                 3
## 5      173        43             118               46                27
## 6      520        42              98                0                42
## # ... with 4 more variables: MntGoldProds <int>, totalHijos <dbl>,
## #   Dinero_Gastado <int>, sonPadres <fct>
```

Veamos como se relacionan las diferentes variables con las variables de output, empezando por si tienen hijos.

```
g1 <-ggplot(datos_ML) +
  geom_boxplot(aes(x = sonPadres, y = MntWines))
g2<-ggplot(datos_ML) +
  geom_boxplot(aes(x = sonPadres, y = MntFruits))
g3<-ggplot(datos_ML) +
  geom_boxplot(aes(x = sonPadres, y = MntMeatProducts))
g4<-ggplot(datos_ML) +
  geom_boxplot(aes(x = sonPadres, y = MntFishProducts))
g5<-ggplot(datos_ML) +
  geom_boxplot(aes(x = sonPadres, y = MntSweetProducts))
g6<-ggplot(datos_ML) +
  geom_boxplot(aes(x = sonPadres, y = MntGoldProds))
g7<-ggplot(datos_ML) +
  geom_boxplot(aes(x = sonPadres, y = Dinero_Gastado))
gridExtra::grid.arrange(g1,g2,g3,g4, g5, g6, g7)
```

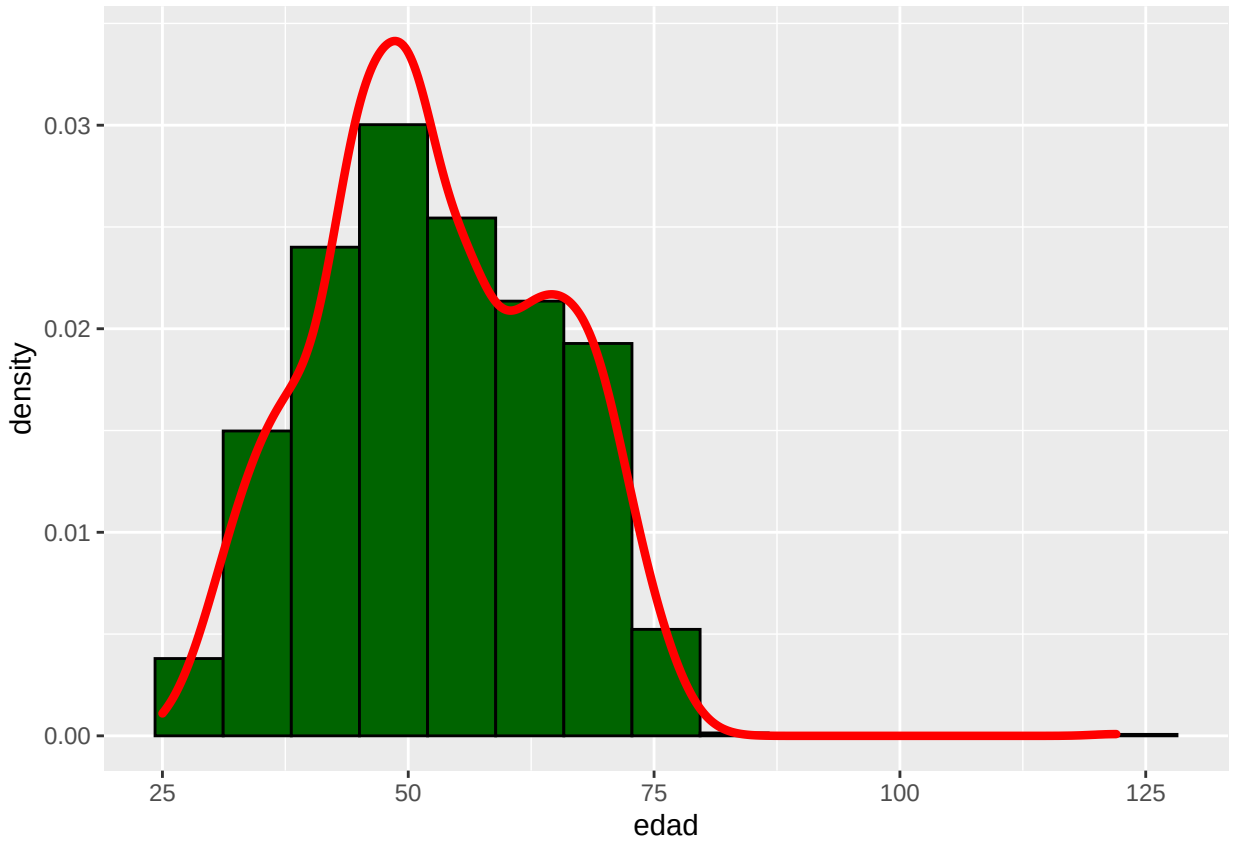



```
gb1 <-ggplot(datos_ML) +
  geom_boxplot(aes(x = factor(totalHijos), y = MntWines))
gb2<-ggplot(datos_ML) +
  geom_boxplot(aes(x = factor(totalHijos), y = MntFruits))
gb3<-ggplot(datos_ML) +
  geom_boxplot(aes(x = factor(totalHijos), y = MntMeatProducts))
gb4<-ggplot(datos_ML) +
  geom_boxplot(aes(x = factor(totalHijos), y = MntFishProducts))
gb5<-ggplot(datos_ML) +
  geom_boxplot(aes(x = factor(totalHijos), y = MntSweetProducts))
gb6<-ggplot(datos_ML) +
  geom_boxplot(aes(x = factor(totalHijos), y = MntGoldProds))
gb7<-ggplot(datos_ML) +
  geom_boxplot(aes(x = factor(totalHijos), y = Dinero_Gastado))
gridExtra::grid.arrange(gb1,gb2,gb3,gb4, gb5, gb6, gb7)
```

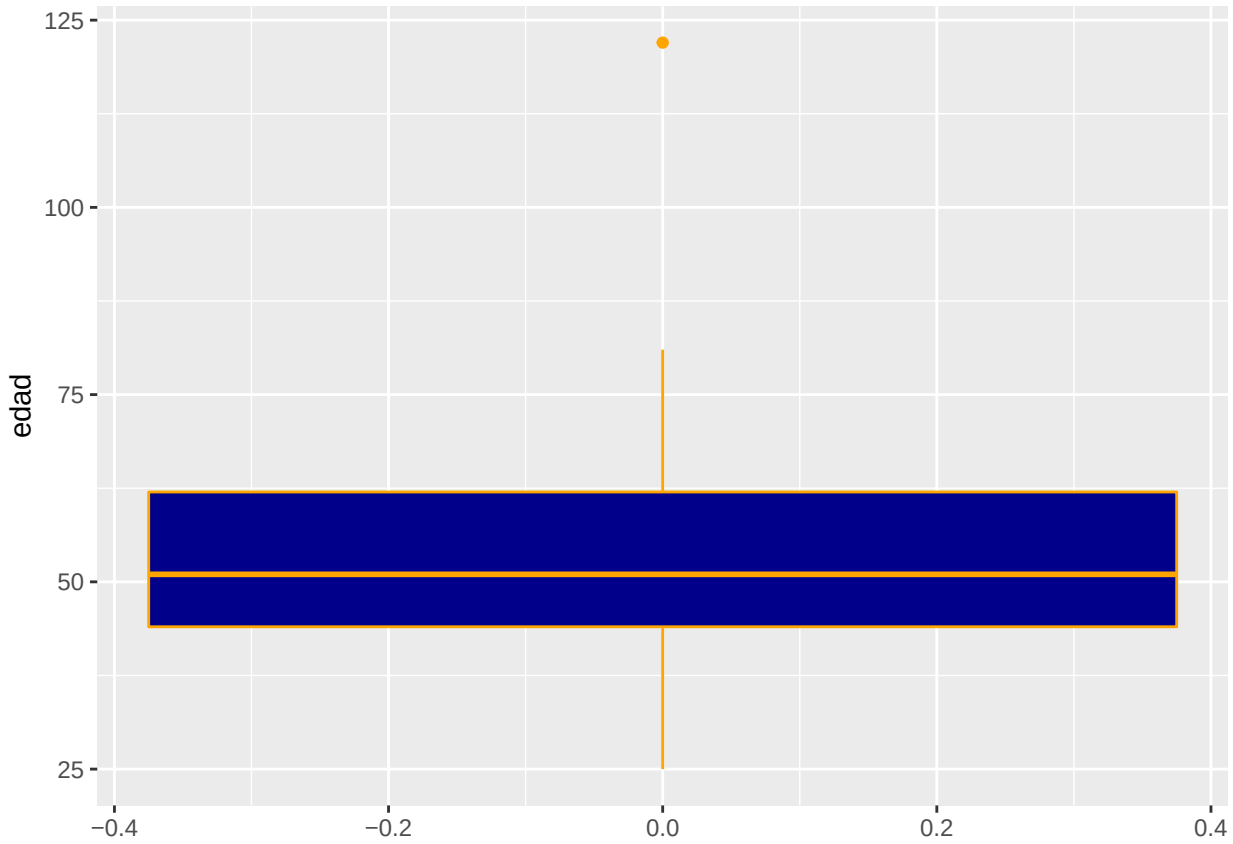


Vemos la relación entre la edad y los gastos de compras. Primero vamos a ver la distribución de la edad. Haría un violin así guapo pero lo he intentado y no sé

```
ggplot(datos) +
  geom_histogram(aes(x = edad, y =stat(density)), bins = 15, fill = "darkgreen", color = "black") +
  geom_density(aes(x = edad), color="red", size=1.5)
```

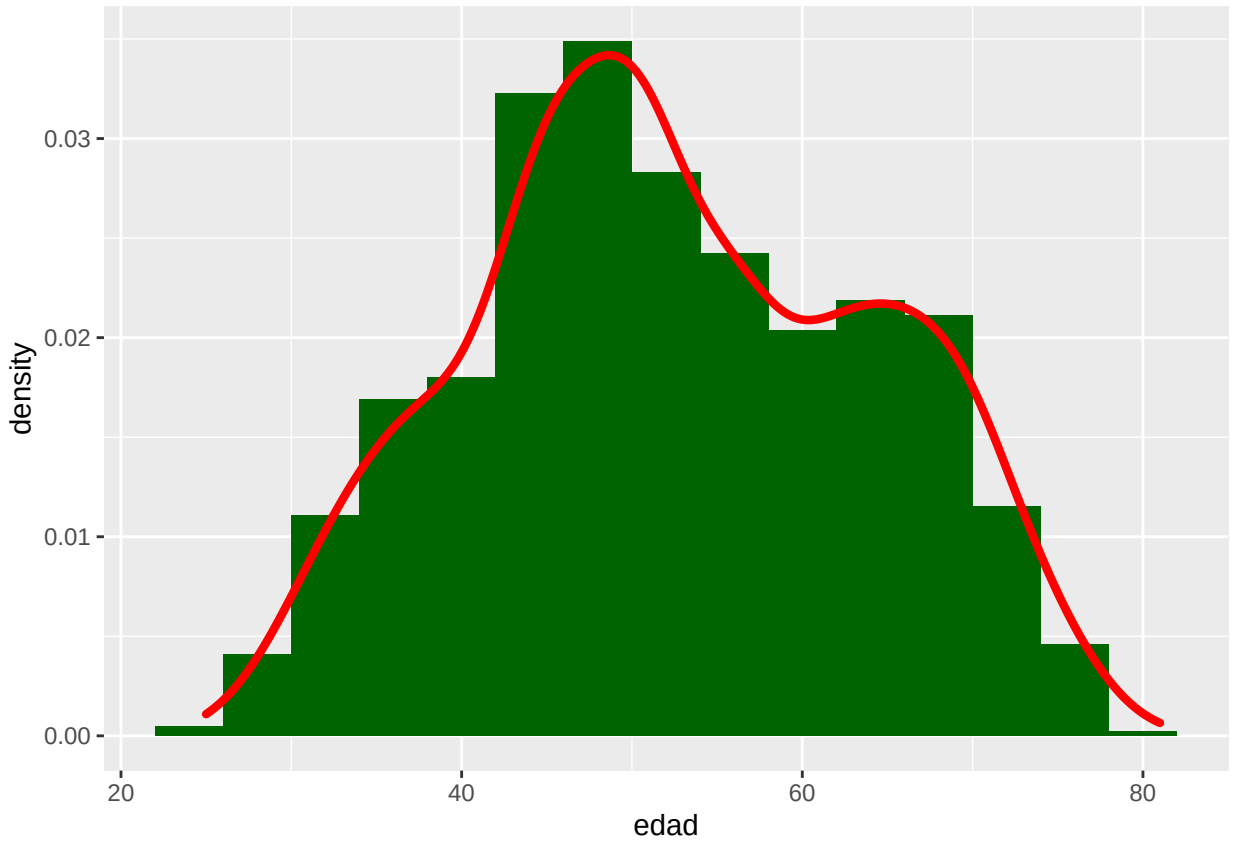



```
ggplot(datos) +
  geom_boxplot(aes(y = edad), fill = "darkblue", color = "orange")
```

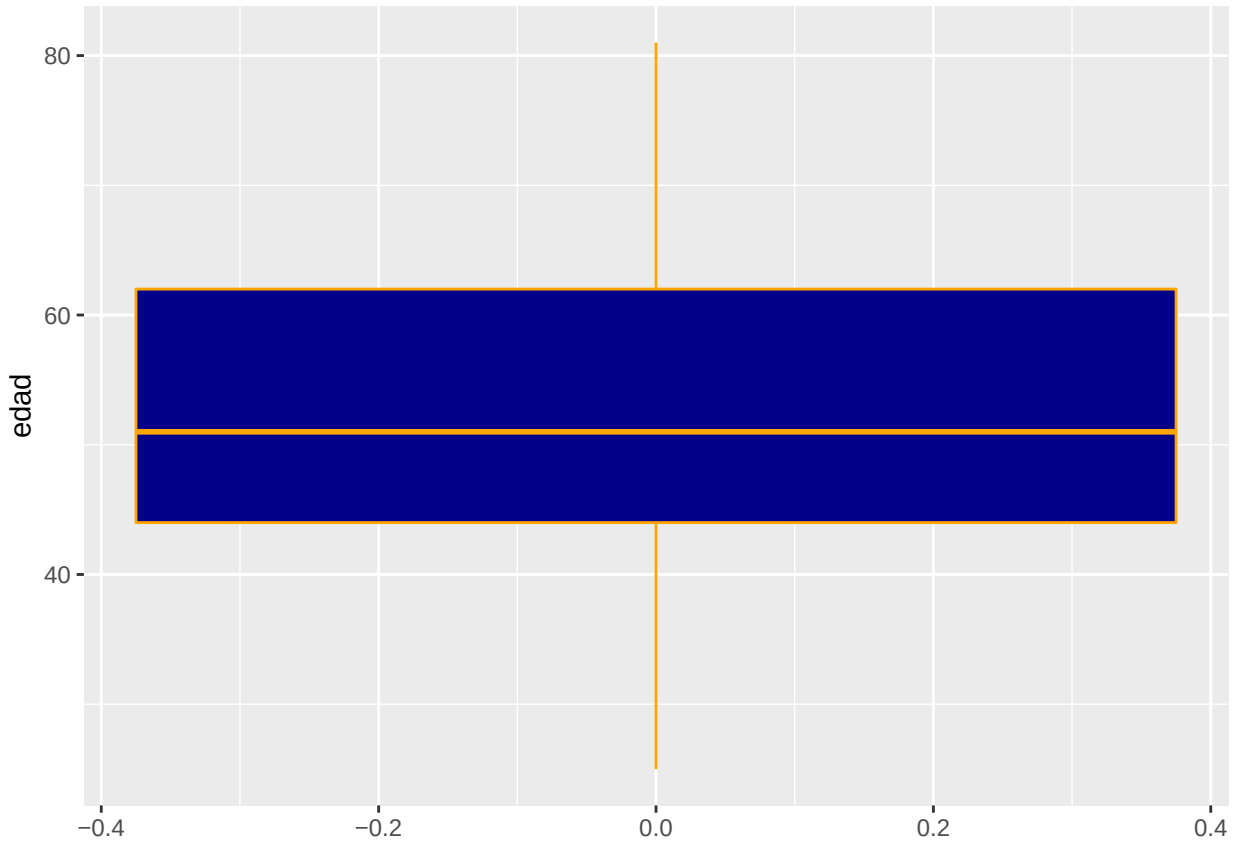


Como hemos visto que parece que hay un dato mal introducido (ya que no creemos que haya un cliente con 120 años) lo eliminamos y volvemos a examinar los datos

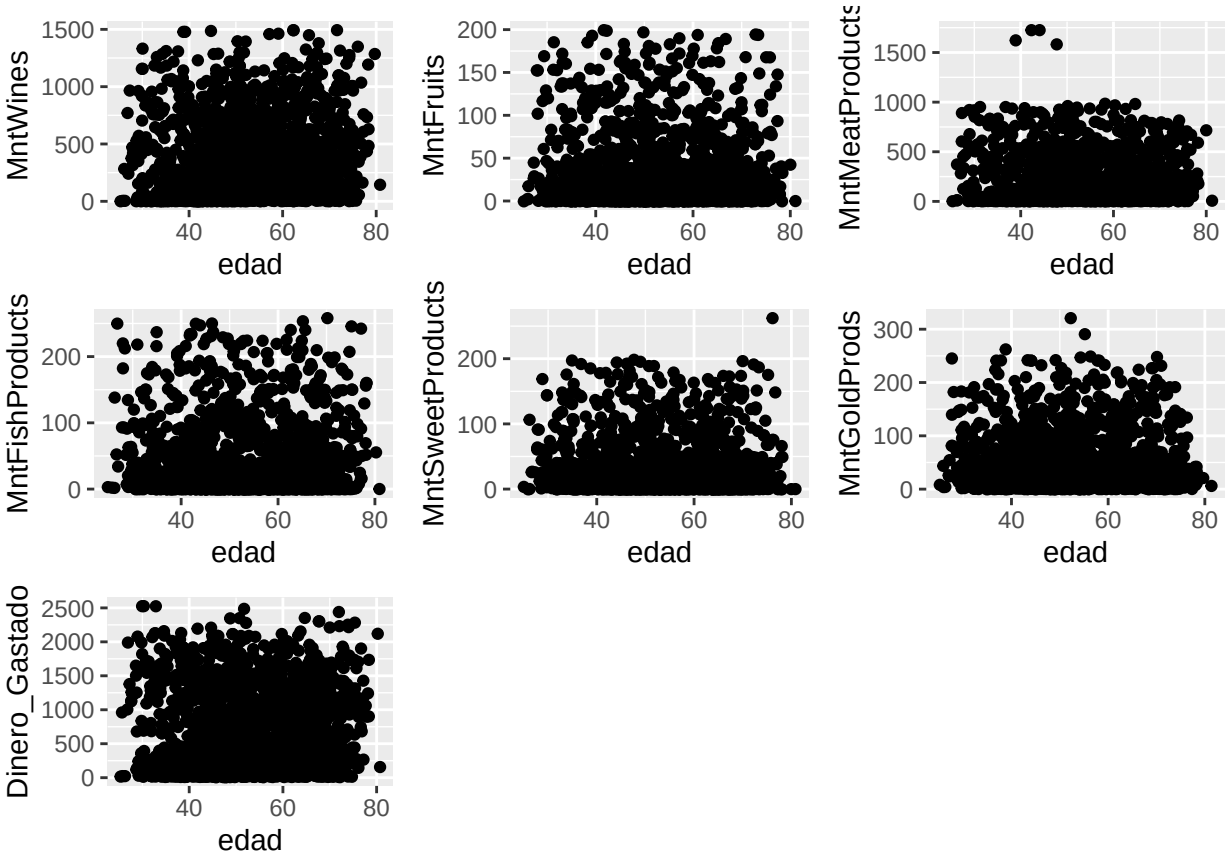
```
datos <- datos %>%
  filter(edad < 100)
ggplot(datos) +
  geom_histogram(aes(x = edad, y =stat(density)), bins = 15, fill = "darkgreen") +
  geom_density(aes(x = edad), color="red", size=1.5)
```




```
ggplot(datos) +
  geom_boxplot(aes(y = edad), fill = "darkblue", color = "orange")
```




```
g1 <-ggplot(datos) +
  geom_jitter(aes(x = edad, y = MntWines))
g2<-ggplot(datos) +
  geom_jitter(aes(x = edad, y = MntFruits))
g3<-ggplot(datos) +
  geom_jitter(aes(x = edad, y = MntMeatProducts))
g4<-ggplot(datos) +
  geom_jitter(aes(x = edad, y = MntFishProducts))
g5<-ggplot(datos) +
  geom_jitter(aes(x = edad, y = MntSweetProducts))
g6<-ggplot(datos) +
  geom_jitter(aes(x = edad, y = MntGoldProds))
g7<-ggplot(datos) +
  geom_jitter(aes(x = edad, y = Dinero_Gastado))
gridExtra::grid.arrange(g1,g2,g3,g4, g5, g6, g7)
```



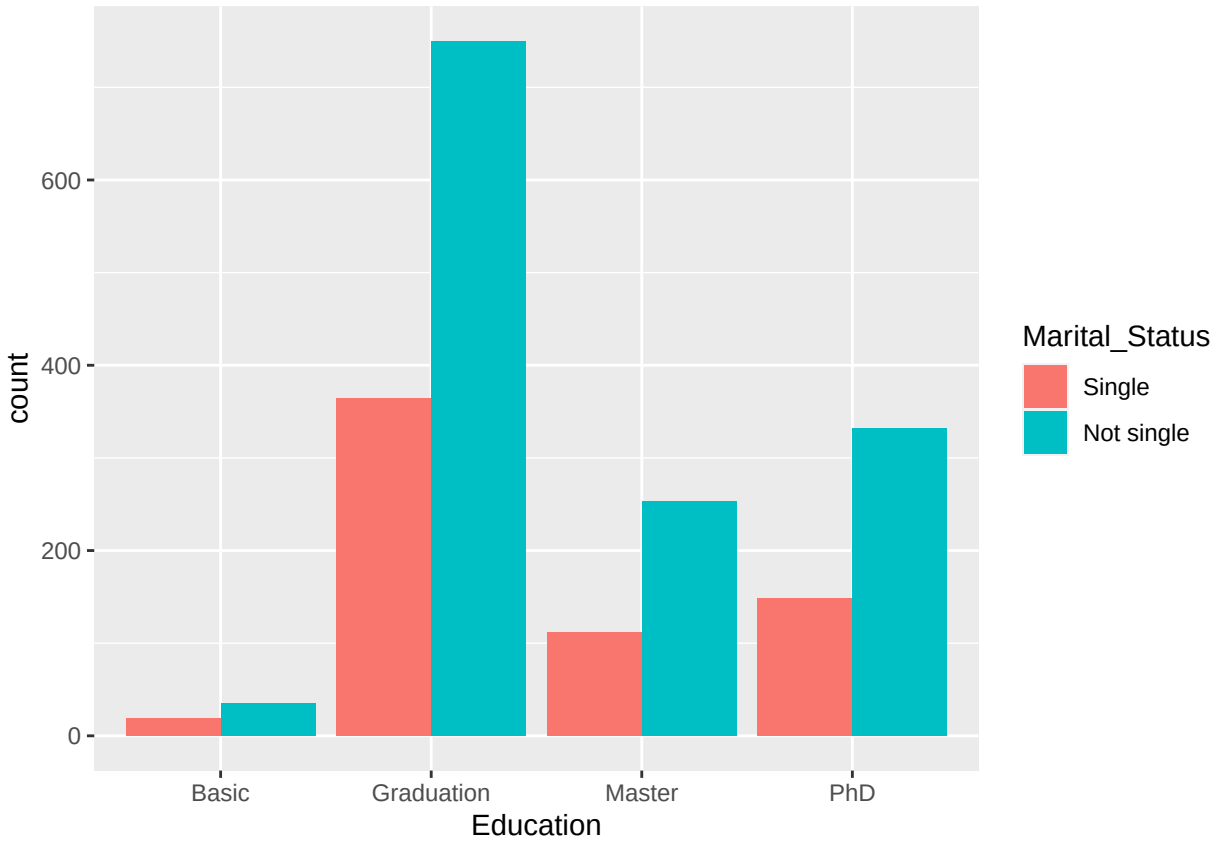
En una primera observación podemos ver que el gasto en vino es mucho mayor que en cualquiera de las otras secciones y para cualquier edad. Por otra parte, noo parece que existan patrones muy claros en ninguna de las variables. Sin embargo, podríamos decir que las personas de mayor edad gastan más dinero en vino y eso probablemente repercuten en que gasten más dinero en general. pero se puede observar en el último gráfico que las personas de una edad más alta gastan más dinero

Realizamos un último estudio en función del nivel de estudios y el estado sentimental. Pero primero vamos a ver cuantos datos hay de cada tio

```
table(datos$Education, datos$Marital_Status)
```

```
##
##              Single Not single
##   Basic          19         35
##   Graduation    364        750
##   Master        112        253
##   PhD           148        332
```

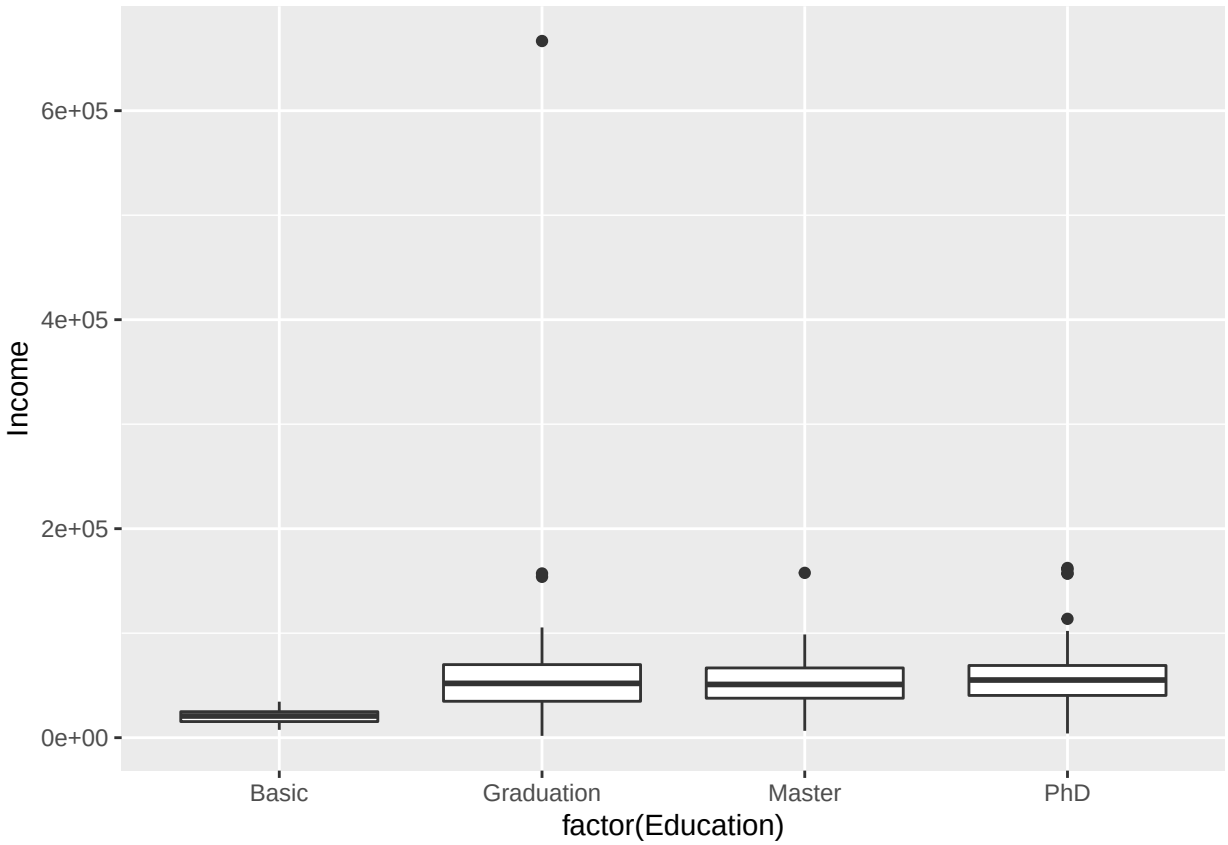
```
ggplot(datos) +
  geom_bar(aes(x = Education, fill = Marital_Status), position = "dodge")
```



Podemos ver que las los clientes en pareja son el doble prácticamente que los clientes sin pareja para cada nivel de estudios.

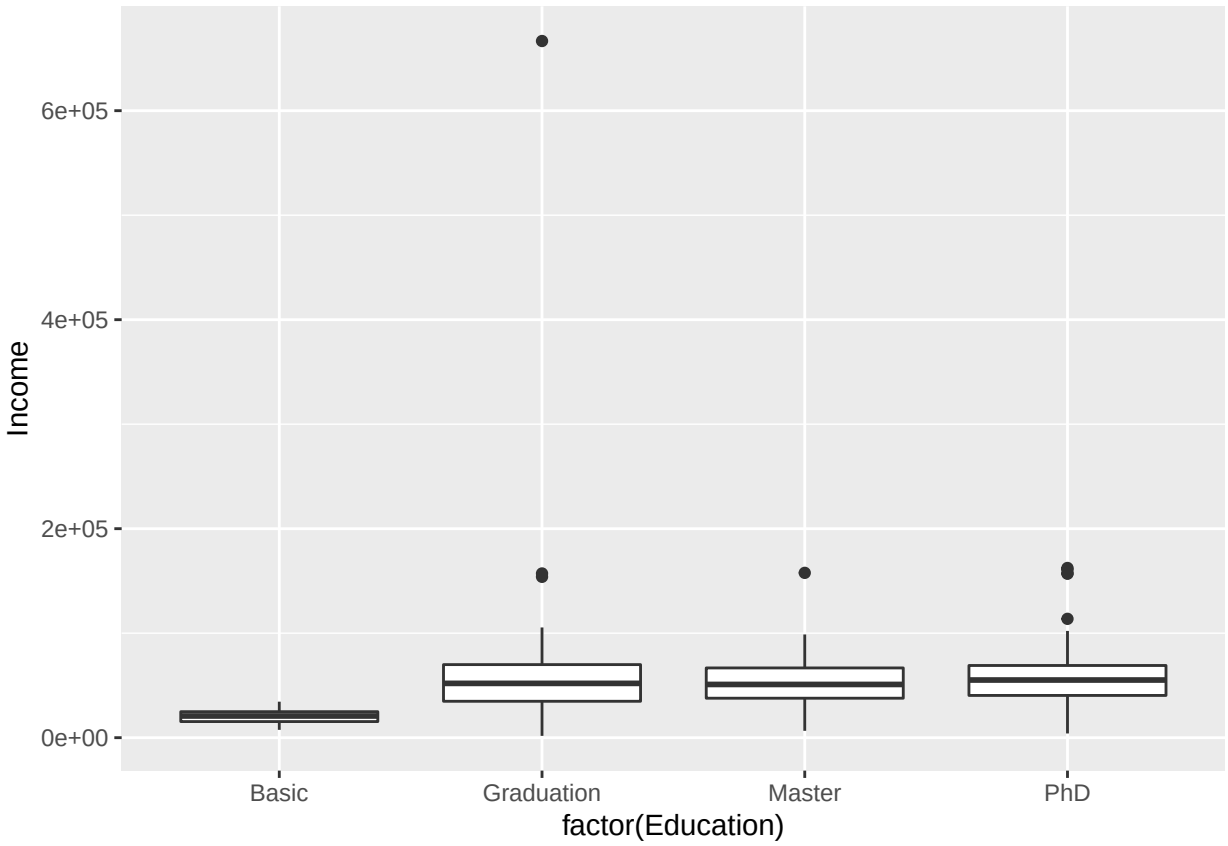
Veamos la relación entre el salario y el nivel de estudios

```
ggplot(datos) +
  geom_boxplot(aes(x = factor(Education), y = Income))
```



Vemos que hay un outlier que no nos permite ver correctamente los gráficos. Por este motivo lo quitamos

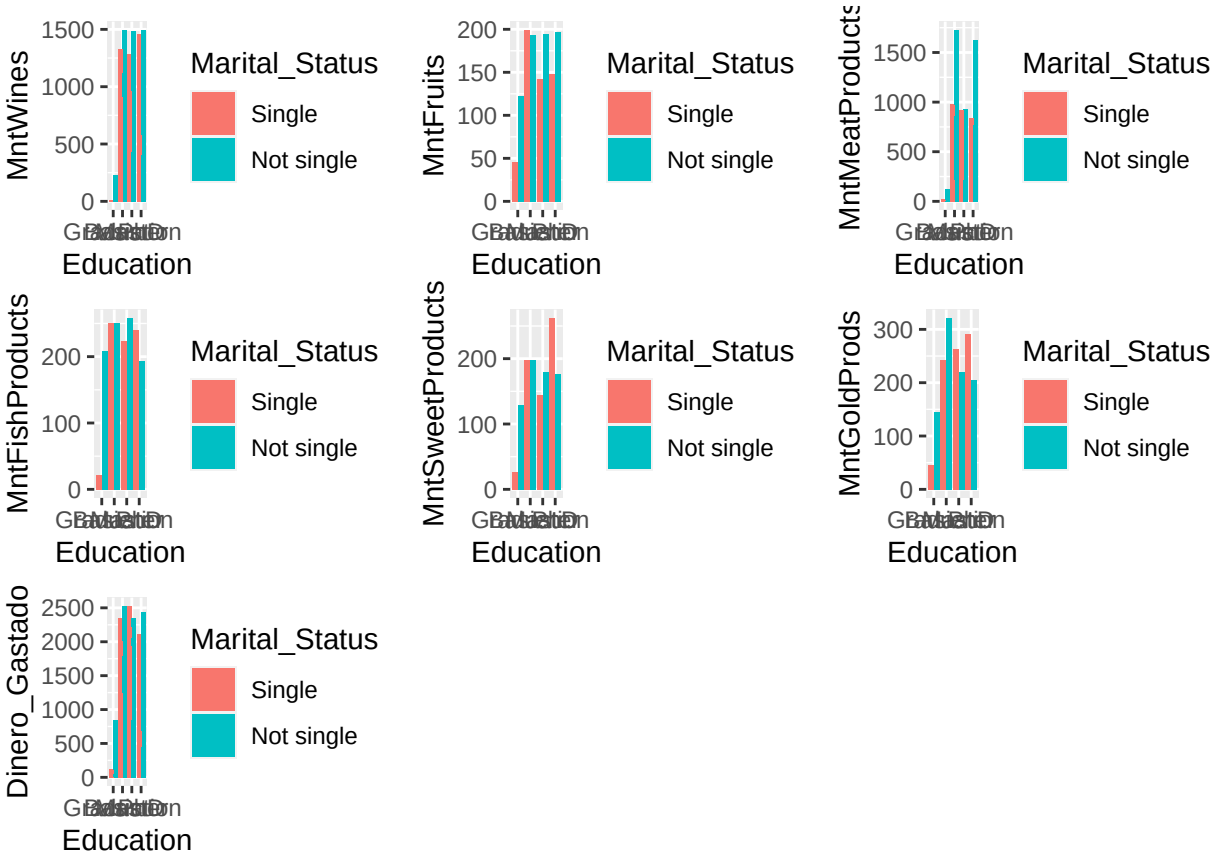
```
aux <- datos %>%
  filter(Income < 500000)
ggplot(datos) +
  geom_boxplot(aes(x = factor(Education), y = Income))
```



Se puede observar cláramente que las personas con un nivel de estudio "Basic" ganan mucho menos que cualquiera de los otros 3 grupos. Otra cosa que hay que tener en cuenta es que entre los 3 grupos restantes no existen deiferencias significativas.

Por último vemos la relación entre el gasto y el nivel de estudios y la situación sentimental.

```
g1 <- ggplot(datos) +
  geom_col(aes(x = Education, y = MntWines, fill = Marital_Status), position = "dodge")
g2<-ggplot(datos) +
  geom_col(aes(x = Education, y = MntFruits, fill = Marital_Status), position = "dodge")
g3<-ggplot(datos) +
  geom_col(aes(x = Education, y = MntMeatProducts, fill = Marital_Status), position = "dodge")
g4<-ggplot(datos) +
  geom_col(aes(x = Education, y = MntFishProducts, fill = Marital_Status), position = "dodge")
g5<-ggplot(datos) +
  geom_col(aes(x = Education, y = MntSweetProducts, fill = Marital_Status), position = "dodge")
g6<-ggplot(datos) +
  geom_col(aes(x = Education, y = MntGoldProds, fill = Marital_Status), position = "dodge")
g7<-ggplot(datos) +
  geom_col(aes(x = Education, y = Dinero_Gastado, fill = Marital_Status), position = "dodge")
gridExtra::grid.arrange(g1,g2,g3,g4, g5, g6, g7)
```



Estos gráficos reflejan lo visto anteriormente, ya que en todas las secciones, los clientes con un nivel de estudio "Basic" gastan mucho menos dinero que cualquiera de los otros 3 grupos, lo que concuerda con que su salario sea menor.

Split del dataset

```
set.seed(150)
trainIndex <- createDataPartition(datos_ML$sonPadres, p = 0.8, list = FALSE, times = 1)
train_set <- datos_ML[trainIndex,]
test_set <- datos_ML[-trainIndex,]
train_set_eval <- train_set
test_set_eval <- test_set
```

Ahora realizamos el control y realizamos el modelo de regresión logística

```
#Control del train
ctrl <- trainControl(method = "cv", number = 10,summaryFunction = defaultSummary, classProbs = TRUE)

#Regresión logística
train_set <- train_set %>%
  mutate(sonPadres = factor(sonPadres == 1, levels = c(TRUE, FALSE), labels = c("Si", "No")))
LogReg.fit <- train(form = sonPadres ~ .,
                    data = train_set,
                    method = "glm",
```

```
                     preProcess = c("center","scale"),
                     trControl = ctrl,
                     metric = "Accuracy")
```

```
## Warning: glm.fit: algorithm did not converge

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Warning in predict.lm(object, newdata, se.fit, scale = 1, type = if (type == :
## prediction from a rank-deficient fit may be misleading

## Warning in predict.lm(object, newdata, se.fit, scale = 1, type = if (type == :
## prediction from a rank-deficient fit may be misleading

## Warning: glm.fit: algorithm did not converge

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Warning in predict.lm(object, newdata, se.fit, scale = 1, type = if (type == :
## prediction from a rank-deficient fit may be misleading

## Warning in predict.lm(object, newdata, se.fit, scale = 1, type = if (type == :
## prediction from a rank-deficient fit may be misleading

## Warning: glm.fit: algorithm did not converge

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Warning in predict.lm(object, newdata, se.fit, scale = 1, type = if (type == :
## prediction from a rank-deficient fit may be misleading

## Warning in predict.lm(object, newdata, se.fit, scale = 1, type = if (type == :
## prediction from a rank-deficient fit may be misleading

## Warning: glm.fit: algorithm did not converge

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Warning in predict.lm(object, newdata, se.fit, scale = 1, type = if (type == :
## prediction from a rank-deficient fit may be misleading

## Warning in predict.lm(object, newdata, se.fit, scale = 1, type = if (type == :
## prediction from a rank-deficient fit may be misleading

## Warning: glm.fit: algorithm did not converge

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Warning in predict.lm(object, newdata, se.fit, scale = 1, type = if (type == :
## prediction from a rank-deficient fit may be misleading

## Warning in predict.lm(object, newdata, se.fit, scale = 1, type = if (type == :
```

```
## prediction from a rank-deficient fit may be misleading

## Warning: glm.fit: algorithm did not converge

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Warning in predict.lm(object, newdata, se.fit, scale = 1, type = if (type == :
## prediction from a rank-deficient fit may be misleading


## Warning in predict.lm(object, newdata, se.fit, scale = 1, type = if (type == :
## prediction from a rank-deficient fit may be misleading

## Warning: glm.fit: algorithm did not converge

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Warning in predict.lm(object, newdata, se.fit, scale = 1, type = if (type == :
## prediction from a rank-deficient fit may be misleading


## Warning in predict.lm(object, newdata, se.fit, scale = 1, type = if (type == :
## prediction from a rank-deficient fit may be misleading

## Warning: glm.fit: algorithm did not converge

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Warning in predict.lm(object, newdata, se.fit, scale = 1, type = if (type == :
## prediction from a rank-deficient fit may be misleading


## Warning in predict.lm(object, newdata, se.fit, scale = 1, type = if (type == :
## prediction from a rank-deficient fit may be misleading

## Warning: glm.fit: algorithm did not converge

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Warning in predict.lm(object, newdata, se.fit, scale = 1, type = if (type == :
## prediction from a rank-deficient fit may be misleading


## Warning in predict.lm(object, newdata, se.fit, scale = 1, type = if (type == :
## prediction from a rank-deficient fit may be misleading

## Warning: glm.fit: algorithm did not converge

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Warning in predict.lm(object, newdata, se.fit, scale = 1, type = if (type == :
## prediction from a rank-deficient fit may be misleading


## Warning in predict.lm(object, newdata, se.fit, scale = 1, type = if (type == :
## prediction from a rank-deficient fit may be misleading

## Warning: glm.fit: algorithm did not converge
```

```
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
```

```
LogReg.fit
```

```
## Generalized Linear Model
##
## 1612 samples
##    8 predictor
##    2 classes: 'Si', 'No'
##
## Pre-processing: centered (8), scaled (8)
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 1451, 1452, 1450, 1450, 1451, 1451, ...
## Resampling results:
##
##   Accuracy  Kappa
##   1         1
```

```
summary(LogReg.fit)
```

```
##
## Call:
## NULL
##
## Deviance Residuals:
##        Min          1Q      Median          3Q         Max
## -9.955e-06  -4.150e-06  -3.717e-06   3.354e-06   5.967e-06
##
## Coefficients: (1 not defined because of singularities)
##                     Estimate Std. Error z value Pr(>|z|)
## (Intercept)       -23.21066 6660.04594  -0.003    0.997
## MntWines           -0.02899 6143.84853   0.000    1.000
## MntFruits           0.03077 6948.74863   0.000    1.000
## MntMeatProducts     0.23567 6328.15385   0.000    1.000
## MntFishProducts     0.06874 7164.86150   0.000    1.000
## MntSweetProducts    0.04734 6858.88803   0.000    1.000
## MntGoldProds       -0.04918 6065.71907   0.000    1.000
## totalHijos        -38.10873 9227.21533  -0.004    0.997
## Dinero_Gastado           NA         NA      NA       NA
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 1.9185e+03  on 1611  degrees of freedom
## Residual deviance: 2.4015e-08  on 1604  degrees of freedom
## AIC: 16
```

```
##
## Number of Fisher Scoring iterations: 25
```

```
str(LogReg.fit)
```

```
## List of 23
##  $ method      : chr "glm"
##  $ modelInfo   :List of 15
##   ..$ label     : chr "Generalized Linear Model"
##   ..$ library   : NULL
##   ..$ loop      : NULL
##   ..$ type      : chr [1:2] "Regression" "Classification"
##   ..$ parameters:'data.frame':   1 obs. of  3 variables:
##   .. ..$ parameter: chr "parameter"
##   .. ..$ class    : chr "character"
##   .. ..$ label    : chr "parameter"
##   ..$ grid      :function (x, y, len = NULL, search = "grid")
##   .. ..- attr(*, "srcref")= 'srcref' int [1:8] 8 26 8 99 26 99 8 8
##   .. .. ..- attr(*, "srcfile")=Classes 'srcfilecopy', 'srcfile' <environment: 0x7fdfc5efe7e0>
##   ..$ fit       :function (x, y, wts, param, lev, last, classProbs, ...)
##   .. ..- attr(*, "srcref")= 'srcref' int [1:8] 9 25 30 19 25 19 9 30
##   .. .. ..- attr(*, "srcfile")=Classes 'srcfilecopy', 'srcfile' <environment: 0x7fdfc5efe7e0>
##   ..$ predict   :function (modelFit, newdata, submodels = NULL)
##   .. ..- attr(*, "srcref")= 'srcref' int [1:8] 31 29 42 19 29 19 31 42
##   .. .. ..- attr(*, "srcfile")=Classes 'srcfilecopy', 'srcfile' <environment: 0x7fdfc5efe7e0>
##   ..$ prob      :function (modelFit, newdata, submodels = NULL)
##   .. ..- attr(*, "srcref")= 'srcref' int [1:8] 43 26 51 19 26 19 43 51
##   .. .. ..- attr(*, "srcfile")=Classes 'srcfilecopy', 'srcfile' <environment: 0x7fdfc5efe7e0>
##   ..$ varImp    :function (object, ...)
##   .. ..- attr(*, "srcref")= 'srcref' int [1:8] 52 28 59 19 28 19 52 59
##   .. .. ..- attr(*, "srcfile")=Classes 'srcfilecopy', 'srcfile' <environment: 0x7fdfc5efe7e0>
##   ..$ predictors:function (x, ...)
##   .. ..- attr(*, "srcref")= 'srcref' int [1:8] 60 32 60 67 32 67 60 60
##   .. .. ..- attr(*, "srcfile")=Classes 'srcfilecopy', 'srcfile' <environment: 0x7fdfc5efe7e0>
##   ..$ levels    :function (x)
##   .. ..- attr(*, "srcref")= 'srcref' int [1:8] 61 28 61 93 28 93 61 61
##   .. .. ..- attr(*, "srcfile")=Classes 'srcfilecopy', 'srcfile' <environment: 0x7fdfc5efe7e0>
##   ..$ trim      :function (x)
##   .. ..- attr(*, "srcref")= 'srcref' int [1:8] 62 26 85 19 26 19 62 85
##   .. .. ..- attr(*, "srcfile")=Classes 'srcfilecopy', 'srcfile' <environment: 0x7fdfc5efe7e0>
##   ..$ tags      : chr [1:4] "Generalized Linear Model" "Linear Classifier" "Two Class Only" "Accepts
##   ..$ sort      :function (x)
##   .. ..- attr(*, "srcref")= 'srcref' int [1:8] 88 26 88 38 26 38 88 88
##   .. .. ..- attr(*, "srcfile")=Classes 'srcfilecopy', 'srcfile' <environment: 0x7fdfc5efe7e0>
```

```
##  $ modelType   : chr "Classification"
##  $ results     :'data.frame':    1 obs. of  5 variables:
##   ..$ parameter : chr "none"
##   ..$ Accuracy  : num 1
##   ..$ Kappa     : num 1
##   ..$ AccuracySD: num 0
##   ..$ KappaSD   : num 0
##  $ pred        : NULL
##  $ bestTune    :'data.frame':    1 obs. of  1 variable:
##   ..$ parameter: chr "none"
##  $ call        : language train.formula(form = sonPadres ~ ., data = train_set, method = "glm", preP
##  $ dots        : list()
##  $ metric      : chr "Accuracy"
##  $ control     :List of 27
##   ..$ method           : chr "cv"
##   ..$ number           : num 10
##   ..$ repeats          : logi NA
##   ..$ search           : chr "grid"
##   ..$ p                : num 0.75
##   ..$ initialWindow    : NULL
##   ..$ horizon          : num 1
##   ..$ fixedWindow      : logi TRUE
##   ..$ skip             : num 0
##   ..$ verboseIter      : logi FALSE
##   ..$ returnData       : logi TRUE
##   ..$ returnResamp     : chr "final"
##   ..$ savePredictions  : chr "none"
##   ..$ classProbs       : logi TRUE
##   ..$ summaryFunction  :function (data, lev = NULL, model = NULL)
##   ..$ selectionFunction: chr "best"
##   ..$ preProcOptions   :List of 6
##   .. ..$ thresh   : num 0.95
##   .. ..$ ICAcomp  : num 3
##   .. ..$ k        : num 5
##   .. ..$ freqCut  : num 19
##   .. ..$ uniqueCut: num 10
##   .. ..$ cutoff   : num 0.9
##   ..$ sampling         : NULL
##   ..$ index            :List of 10
##   .. ..$ Fold01: int [1:1451] 1 2 3 4 5 6 7 8 9 10 ...
##   .. ..$ Fold02: int [1:1452] 1 2 3 4 6 7 8 9 11 12 ...
##   .. ..$ Fold03: int [1:1450] 2 3 4 5 6 7 8 9 10 11 ...
##   .. ..$ Fold04: int [1:1450] 1 2 3 5 6 7 8 9 10 11 ...
```

```
##   .. ..$ Fold05: int [1:1451] 1 2 3 4 5 6 7 8 9 10 ...
##   .. ..$ Fold06: int [1:1451] 1 2 3 4 5 6 8 9 10 11 ...
##   .. ..$ Fold07: int [1:1451] 1 3 4 5 6 7 9 10 11 12 ...
##   .. ..$ Fold08: int [1:1450] 1 2 3 4 5 7 8 9 10 11 ...
##   .. ..$ Fold09: int [1:1451] 1 2 4 5 6 7 8 10 11 12 ...
##   .. ..$ Fold10: int [1:1451] 1 2 3 4 5 6 7 8 9 10 ...
##   ..$ indexOut         :List of 10
##   .. ..$ Resample01: int [1:161] 12 53 55 57 61 70 74 79 81 83 ...
##   .. ..$ Resample02: int [1:160] 5 10 14 18 41 43 84 102 104 105 ...
##   .. ..$ Resample03: int [1:162] 1 13 25 37 40 75 88 95 114 116 ...
##   .. ..$ Resample04: int [1:162] 4 24 29 31 35 54 56 62 73 80 ...
##   .. ..$ Resample05: int [1:161] 11 21 28 34 44 46 89 98 112 141 ...
##   .. ..$ Resample06: int [1:161] 7 16 42 52 58 63 82 119 128 134 ...
##   .. ..$ Resample07: int [1:161] 2 8 17 36 47 59 60 68 72 78 ...
##   .. ..$ Resample08: int [1:162] 6 15 23 26 32 33 64 66 67 91 ...
##   .. ..$ Resample09: int [1:161] 3 9 22 27 30 45 48 49 65 69 ...
##   .. ..$ Resample10: int [1:161] 19 20 38 39 50 51 71 77 99 115 ...
##   ..$ indexFinal       : NULL
##   ..$ timingSamps      : num 0
##   ..$ predictionBounds : logi [1:2] FALSE FALSE
##   ..$ seeds            :List of 11
##   .. ..$ : int 533752
##   .. ..$ : int 891356
##   .. ..$ : int 856374
##   .. ..$ : int 414352
##   .. ..$ : int 710866
##   .. ..$ : int 942913
##   .. ..$ : int 983122
##   .. ..$ : int 877848
##   .. ..$ : int 209121
##   .. ..$ : int 228196
##   .. ..$ : int 219816
##   ..$ adaptive         :List of 4
##   .. ..$ min     : num 5
##   .. ..$ alpha   : num 0.05
##   .. ..$ method  : chr "gls"
##   .. ..$ complete: logi TRUE
##   ..$ trim             : logi FALSE
##   ..$ allowParallel    : logi TRUE
##  $ finalModel  :List of 34
##   ..$ coefficients     : Named num [1:9] -23.2107 -0.029 0.0308 0.2357 0.0687 ...
##   .. ..- attr(*, "names")= chr [1:9] "(Intercept)" "MntWines" "MntFruits" "MntMeatProducts" ...
##   ..$ residuals        : Named num [1:1612] 1 -1 1 -1 -1 ...
```

```
##   .. ..- attr(*, "names")= chr [1:1612] "X1" "X2" "X3" "X4" ...
##   ..$ fitted.values    : Named num [1:1612] 1.00 2.22e-16 1.00 8.62e-12 1.12e-11 ...
##   .. ..- attr(*, "names")= chr [1:1612] "X1" "X2" "X3" "X4" ...
##   ..$ effects          : Named num [1:1612] 0.001302 0.000906 -0.000733 0.000841 0.000309 ...
##   .. ..- attr(*, "names")= chr [1:1612] "(Intercept)" "MntWines" "MntFruits" "MntMeatProducts" ...
##   ..$ R                : num [1:9, 1:9] -0.000181 0 0 0 0 ...
##   .. ..- attr(*, "dimnames")=List of 2
##   .. .. ..$ : chr [1:9] "(Intercept)" "MntWines" "MntFruits" "MntMeatProducts" ...
##   .. .. ..$ : chr [1:9] "(Intercept)" "MntWines" "MntFruits" "MntMeatProducts" ...
##   ..$ rank             : int 8
##   ..$ qr               :List of 5
##   .. ..$ qr   : num [1:1612, 1:9] -1.81e-04 8.25e-05 3.03e-02 2.68e-02 3.06e-02 ...
##   .. .. ..- attr(*, "dimnames")=List of 2
##   .. .. .. ..$ : chr [1:1612] "X1" "X2" "X3" "X4" ...
##   .. .. .. ..$ : chr [1:9] "(Intercept)" "MntWines" "MntFruits" "MntMeatProducts" ...
##   .. ..$ rank : int 8
##   .. ..$ qraux: num [1:9] 1.02 1 1.01 1.01 1.01 ...
##   .. ..$ pivot: int [1:9] 1 2 3 4 5 6 7 8 9
##   .. ..$ tol  : num 1e-11
##   .. ..- attr(*, "class")= chr "qr"
##   ..$ family           :List of 12
##   .. ..$ family    : chr "binomial"
##   .. ..$ link      : chr "logit"
##   .. ..$ linkfun   :function (mu)
##   .. ..$ linkinv   :function (eta)
##   .. ..$ variance  :function (mu)
##   .. ..$ dev.resids:function (y, mu, wt)
##   .. ..$ aic       :function (y, n, mu, wt, dev)
##   .. ..$ mu.eta    :function (eta)
##   .. ..$ initialize: language {     if (NCOL(y) == 1) { ...
##   .. ..$ validmu   :function (mu)
##   .. ..$ valideta  :function (eta)
##   .. ..$ simulate  :function (object, nsim)
##   .. ..- attr(*, "class")= chr "family"
##   ..$ linear.predictors: Named num [1:1612] 25.8 -76 25.2 -25.5 -25.2 ...
##   .. ..- attr(*, "names")= chr [1:1612] "X1" "X2" "X3" "X4" ...
##   ..$ deviance         : num 2.4e-08
##   ..$ aic              : num 16
##   ..$ null.deviance    : num 1919
##   ..$ iter             : int 25
##   ..$ weights          : Named num [1:1612] 1.72e-11 2.22e-16 3.00e-11 2.34e-11 3.05e-11 ...
##   .. ..- attr(*, "names")= chr [1:1612] "X1" "X2" "X3" "X4" ...
##   ..$ prior.weights    : Named num [1:1612] 1 1 1 1 1 1 1 1 1 1 ...
```

```
##   .. ..- attr(*, "names")= chr [1:1612] "X1" "X2" "X3" "X4" ...
##   ..$ df.residual      : int 1604
##   ..$ df.null          : int 1611
##   ..$ y                : Named num [1:1612] 1 0 1 0 0 0 0 0 1 1 ...
##   .. ..- attr(*, "names")= chr [1:1612] "X1" "X2" "X3" "X4" ...
##   ..$ converged        : logi FALSE
##   ..$ boundary         : logi FALSE
##   ..$ model            :'data.frame':    1612 obs. of  9 variables:
##   .. ..$ .outcome        : Factor w/ 2 levels "Si","No": 2 1 2 1 1 1 1 1 2 2 ...
##   .. ..$ MntWines        : num [1:1612] 0.942 -0.879 0.332 -0.879 -0.225 ...
##   .. ..$ MntFruits       : num [1:1612] 1.575 -0.624 0.589 -0.548 0.993 ...
##   .. ..$ MntMeatProducts : num [1:1612] 1.6553 -0.711 -0.1808 -0.6496 -0.0186 ...
##   .. ..$ MntFishProducts : num [1:1612] 2.599 -0.639 1.437 -0.486 0.275 ...
##   .. ..$ MntSweetProducts: num [1:1612] 1.552 -0.621 -0.121 -0.571 0.578 ...
##   .. ..$ MntGoldProds    : num [1:1612] 0.886 -0.7241 -0.0172 -0.7438 -0.3118 ...
##   .. ..$ totalHijos      : num [1:1612] -1.2695 1.3796 -1.2695 0.0551 0.0551 ...
##   .. ..$ Dinero_Gastado  : num [1:1612] 1.6625 -0.9635 0.2735 -0.9206 -0.0337 ...
##   .. ..- attr(*, "terms")=Classes 'terms', 'formula'  language .outcome ~ MntWines + MntFruits + Mntl
##   .. .. .. ..- attr(*, "variables")= language list(.outcome, MntWines, MntFruits, MntMeatProducts, Mn
##   .. .. .. ..- attr(*, "factors")= int [1:9, 1:8] 0 1 0 0 0 0 0 0 0 0 ...
##   .. .. .. .. ..- attr(*, "dimnames")=List of 2
##   .. .. .. .. .. ..$ : chr [1:9] ".outcome" "MntWines" "MntFruits" "MntMeatProducts" ...
##   .. .. .. .. .. ..$ : chr [1:8] "MntWines" "MntFruits" "MntMeatProducts" "MntFishProducts" ...
##   .. .. .. ..- attr(*, "term.labels")= chr [1:8] "MntWines" "MntFruits" "MntMeatProducts" "MntFishPro
##   .. .. .. ..- attr(*, "order")= int [1:8] 1 1 1 1 1 1 1 1
##   .. .. .. ..- attr(*, "intercept")= int 1
##   .. .. .. ..- attr(*, "response")= int 1
##   .. .. .. ..- attr(*, ".Environment")=<environment: 0x7fdfad5d4bf8>
##   .. .. .. ..- attr(*, "predvars")= language list(.outcome, MntWines, MntFruits, MntMeatProducts, Mnt
##   .. .. .. ..- attr(*, "dataClasses")= Named chr [1:9] "factor" "numeric" "numeric" "numeric" ...
##   .. .. .. .. ..- attr(*, "names")= chr [1:9] ".outcome" "MntWines" "MntFruits" "MntMeatProducts" ..
##   ..$ formula          :Class 'formula'  language .outcome ~ .
##   .. .. ..- attr(*, ".Environment")=<environment: 0x7fdfad5d4bf8>
##   ..$ terms            :Classes 'terms', 'formula'  language .outcome ~ MntWines + MntFruits + MntMea
##   .. .. ..- attr(*, "variables")= language list(.outcome, MntWines, MntFruits, MntMeatProducts, MntFr
##   .. .. ..- attr(*, "factors")= int [1:9, 1:8] 0 1 0 0 0 0 0 0 0 0 ...
##   .. .. .. ..- attr(*, "dimnames")=List of 2
##   .. .. .. .. ..$ : chr [1:9] ".outcome" "MntWines" "MntFruits" "MntMeatProducts" ...
##   .. .. .. .. ..$ : chr [1:8] "MntWines" "MntFruits" "MntMeatProducts" "MntFishProducts" ...
##   .. .. ..- attr(*, "term.labels")= chr [1:8] "MntWines" "MntFruits" "MntMeatProducts" "MntFishProduc
##   .. .. ..- attr(*, "order")= int [1:8] 1 1 1 1 1 1 1 1
##   .. .. ..- attr(*, "intercept")= int 1
##   .. .. ..- attr(*, "response")= int 1
```

```
##   .. .. ..- attr(*, ".Environment")=<environment: 0x7fdfad5d4bf8>
##   .. .. ..- attr(*, "predvars")= language list(.outcome, MntWines, MntFruits, MntMeatProducts, MntFis
##   .. .. ..- attr(*, "dataClasses")= Named chr [1:9] "factor" "numeric" "numeric" "numeric" ...
##   .. .. .. ..- attr(*, "names")= chr [1:9] ".outcome" "MntWines" "MntFruits" "MntMeatProducts" ...
##   ..$ data             :'data.frame':    1612 obs. of  9 variables:
##   .. ..$ MntWines        : num [1:1612] 0.942 -0.879 0.332 -0.879 -0.225 ...
##   .. ..$ MntFruits       : num [1:1612] 1.575 -0.624 0.589 -0.548 0.993 ...
##   .. ..$ MntMeatProducts : num [1:1612] 1.6553 -0.711 -0.1808 -0.6496 -0.0186 ...
##   .. ..$ MntFishProducts : num [1:1612] 2.599 -0.639 1.437 -0.486 0.275 ...
##   .. ..$ MntSweetProducts: num [1:1612] 1.552 -0.621 -0.121 -0.571 0.578 ...
##   .. ..$ MntGoldProds    : num [1:1612] 0.886 -0.7241 -0.0172 -0.7438 -0.3118 ...
##   .. ..$ totalHijos      : num [1:1612] -1.2695 1.3796 -1.2695 0.0551 0.0551 ...
##   .. ..$ Dinero_Gastado  : num [1:1612] 1.6625 -0.9635 0.2735 -0.9206 -0.0337 ...
##   .. ..$ .outcome        : Factor w/ 2 levels "Si","No": 2 1 2 1 1 1 1 1 2 2 ...
##   ..$ offset           : NULL
##   ..$ control          :List of 3
##   .. ..$ epsilon: num 1e-08
##   .. ..$ maxit  : num 25
##   .. ..$ trace  : logi FALSE
##   ..$ method           : chr "glm.fit"
##   ..$ contrasts        : NULL
##   ..$ xlevels          : Named list()
##   ..$ xNames           : chr [1:8] "MntWines" "MntFruits" "MntMeatProducts" "MntFishProducts" ...
##   ..$ problemType      : chr "Classification"
##   ..$ tuneValue        :'data.frame':    1 obs. of  1 variable:
##   .. ..$ parameter: chr "none"
##   ..$ obsLevels        : chr [1:2] "Si" "No"
##   .. ..- attr(*, "ordered")= logi FALSE
##   ..$ param            : list()
##   ..- attr(*, "class")= chr [1:2] "glm" "lm"
##  $ preProcess  :List of 22
##   ..$ dim              : int [1:2] 1612 8
##   ..$ bc               : NULL
##   ..$ yj               : NULL
##   ..$ et               : NULL
##   ..$ invHyperbolicSine: NULL
##   ..$ mean             : Named num [1:8] 312.2 25.7 168.3 35.5 25.9 ...
##   .. ..- attr(*, "names")= chr [1:8] "MntWines" "MntFruits" "MntMeatProducts" "MntFishProducts" ...
##   ..$ std              : Named num [1:8] 342.6 39.6 228.2 52.5 40 ...
##   .. ..- attr(*, "names")= chr [1:8] "MntWines" "MntFruits" "MntMeatProducts" "MntFishProducts" ...
##   ..$ ranges           : NULL
##   ..$ rotation         : NULL
##   ..$ method           :List of 3
```

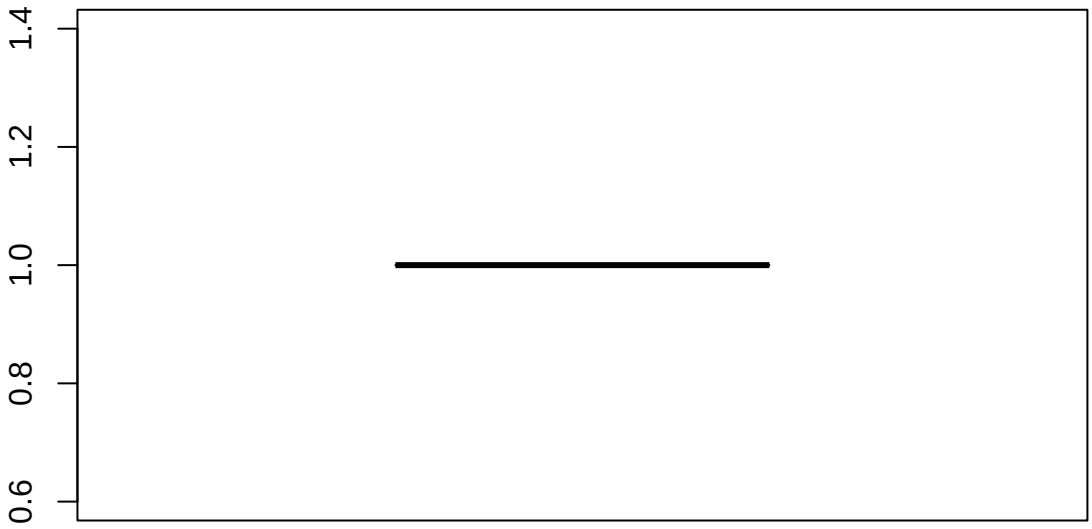
```
##   .. ..$ center: chr [1:8] "MntWines" "MntFruits" "MntMeatProducts" "MntFishProducts" ...
##   .. ..$ scale : chr [1:8] "MntWines" "MntFruits" "MntMeatProducts" "MntFishProducts" ...
##   .. ..$ ignore: chr(0)
##   ..$ thresh           : num 0.95
##   ..$ pcaComp          : NULL
##   ..$ numComp          : NULL
##   ..$ ica              : NULL
##   ..$ wildcards        :List of 2
##   .. ..$ PCA: chr(0)
##   .. ..$ ICA: chr(0)
##   ..$ k                : num 5
##   ..$ knnSummary       :function (x, ...)
##   ..$ bagImp           : NULL
##   ..$ median           : NULL
##   ..$ data             : NULL
##   ..$ rangeBounds      : num [1:2] 0 1
##   ..$ call             : chr "scrubed"
##   ..- attr(*, "class")= chr "preProcess"
##  $ trainingData: rowwise_df [1,612 x 9] (S3: rowwise_df/tbl_df/tbl/data.frame)
##   ..$ .outcome        : Factor w/ 2 levels "Si","No": 2 1 2 1 1 1 1 1 2 2 ...
##   ..$ MntWines        : int [1:1612] 635 11 426 11 235 76 14 28 194 3 ...
##   ..$ MntFruits       : int [1:1612] 88 1 49 4 65 10 0 0 61 14 ...
##   ..$ MntMeatProducts : int [1:1612] 546 6 127 20 164 56 24 6 480 17 ...
##   ..$ MntFishProducts : int [1:1612] 172 2 111 10 50 3 3 1 225 6 ...
##   ..$ MntSweetProducts: int [1:1612] 88 1 21 3 49 1 3 1 112 1 ...
##   ..$ MntGoldProds    : int [1:1612] 88 6 42 5 27 23 2 13 30 5 ...
##   ..$ totalHijos      : num [1:1612] 0 2 0 1 1 1 1 2 0 0 ...
##   ..$ Dinero_Gastado  : int [1:1612] 1617 27 776 53 590 169 46 49 1102 46 ...
##   ..- attr(*, "groups")= tibble [1,612 x 1] (S3: tbl_df/tbl/data.frame)
##   .. ..$ .rows: list<int> [1:1612]
##   .. .. ..$ : int 1
##   .. .. ..$ : int 2
##   .. .. ..$ : int 3
##   .. .. ..$ : int 4
##   .. .. ..$ : int 5
##   .. .. ..$ : int 6
##   .. .. ..$ : int 7
##   .. .. ..$ : int 8
##   .. .. ..$ : int 9
##   .. .. ..$ : int 10
##   .. .. ..$ : int 11
##   .. .. ..$ : int 12
##   .. .. ..$ : int 13
```

```
##   .. .. ..$ : int 14
##   .. .. ..$ : int 15
##   .. .. ..$ : int 16
##   .. .. ..$ : int 17
##   .. .. ..$ : int 18
##   .. .. ..$ : int 19
##   .. .. ..$ : int 20
##   .. .. ..$ : int 21
##   .. .. ..$ : int 22
##   .. .. ..$ : int 23
##   .. .. ..$ : int 24
##   .. .. ..$ : int 25
##   .. .. ..$ : int 26
##   .. .. ..$ : int 27
##   .. .. ..$ : int 28
##   .. .. ..$ : int 29
##   .. .. ..$ : int 30
##   .. .. ..$ : int 31
##   .. .. ..$ : int 32
##   .. .. ..$ : int 33
##   .. .. ..$ : int 34
##   .. .. ..$ : int 35
##   .. .. ..$ : int 36
##   .. .. ..$ : int 37
##   .. .. ..$ : int 38
##   .. .. ..$ : int 39
##   .. .. ..$ : int 40
##   .. .. ..$ : int 41
##   .. .. ..$ : int 42
##   .. .. ..$ : int 43
##   .. .. ..$ : int 44
##   .. .. ..$ : int 45
##   .. .. ..$ : int 46
##   .. .. ..$ : int 47
##   .. .. ..$ : int 48
##   .. .. ..$ : int 49
##   .. .. ..$ : int 50
##   .. .. ..$ : int 51
##   .. .. ..$ : int 52
##   .. .. ..$ : int 53
##   .. .. ..$ : int 54
##   .. .. ..$ : int 55
##   .. .. ..$ : int 56
```

```
##   .. .. ..$ : int 57
##   .. .. ..$ : int 58
##   .. .. ..$ : int 59
##   .. .. ..$ : int 60
##   .. .. ..$ : int 61
##   .. .. ..$ : int 62
##   .. .. ..$ : int 63
##   .. .. ..$ : int 64
##   .. .. ..$ : int 65
##   .. .. ..$ : int 66
##   .. .. ..$ : int 67
##   .. .. ..$ : int 68
##   .. .. ..$ : int 69
##   .. .. ..$ : int 70
##   .. .. ..$ : int 71
##   .. .. ..$ : int 72
##   .. .. ..$ : int 73
##   .. .. ..$ : int 74
##   .. .. ..$ : int 75
##   .. .. ..$ : int 76
##   .. .. ..$ : int 77
##   .. .. ..$ : int 78
##   .. .. ..$ : int 79
##   .. .. ..$ : int 80
##   .. .. ..$ : int 81
##   .. .. ..$ : int 82
##   .. .. ..$ : int 83
##   .. .. ..$ : int 84
##   .. .. ..$ : int 85
##   .. .. ..$ : int 86
##   .. .. ..$ : int 87
##   .. .. ..$ : int 88
##   .. .. ..$ : int 89
##   .. .. ..$ : int 90
##   .. .. ..$ : int 91
##   .. .. ..$ : int 92
##   .. .. ..$ : int 93
##   .. .. ..$ : int 94
##   .. .. ..$ : int 95
##   .. .. ..$ : int 96
##   .. .. ..$ : int 97
##   .. .. ..$ : int 98
##   .. .. ..$ : int 99
```

```
##   .. .. .. [list output truncated]
##   .. .. ..@ ptype: int(0)
##  $ resample    :'data.frame':    10 obs. of  3 variables:
##   ..$ Accuracy: num [1:10] 1 1 1 1 1 1 1 1 1 1
##   ..$ Kappa   : num [1:10] 1 1 1 1 1 1 1 1 1 1
##   ..$ Resample: chr [1:10] "Fold01" "Fold02" "Fold03" "Fold04" ...
##  $ resampledCM :'data.frame':    10 obs. of  6 variables:
##   ..$ cell1    : num [1:10] 116 115 116 116 115 115 116 116 116 116
##   ..$ cell2    : num [1:10] 0 0 0 0 0 0 0 0 0 0
##   ..$ cell3    : num [1:10] 0 0 0 0 0 0 0 0 0 0
##   ..$ cell4    : num [1:10] 45 45 46 46 46 46 45 46 45 45
##   ..$ parameter: chr [1:10] "none" "none" "none" "none" ...
##   ..$ Resample : chr [1:10] "Fold01" "Fold02" "Fold03" "Fold04" ...
##  $ perfNames   : chr [1:2] "Accuracy" "Kappa"
##  $ maximize    : logi TRUE
##  $ yLimits     : NULL
##  $ times       :List of 3
##   ..$ everything: 'proc_time' Named num [1:5] 0.957 0.051 1.025 0 0
##   .. ..- attr(*, "names")= chr [1:5] "user.self" "sys.self" "elapsed" "user.child" ...
##   ..$ final     : 'proc_time' Named num [1:5] 0.027 0.003 0.03 0 0
##   .. ..- attr(*, "names")= chr [1:5] "user.self" "sys.self" "elapsed" "user.child" ...
##   ..$ prediction: logi [1:3] NA NA NA
##  $ levels      : chr [1:2] "Si" "No"
##   ..- attr(*, "ordered")= logi FALSE
##  $ terms       :Classes 'terms', 'formula'  language sonPadres ~ MntWines + MntFruits + MntMeatProduc
##   .. ..- attr(*, "variables")= language list(sonPadres, MntWines, MntFruits, MntMeatProducts, MntFish
##   .. ..- attr(*, "factors")= int [1:9, 1:8] 0 1 0 0 0 0 0 0 0 0 ...
##   .. .. ..- attr(*, "dimnames")=List of 2
##   .. .. .. ..$ : chr [1:9] "sonPadres" "MntWines" "MntFruits" "MntMeatProducts" ...
##   .. .. .. ..$ : chr [1:8] "MntWines" "MntFruits" "MntMeatProducts" "MntFishProducts" ...
##   .. ..- attr(*, "term.labels")= chr [1:8] "MntWines" "MntFruits" "MntMeatProducts" "MntFishProducts
##   .. ..- attr(*, "order")= int [1:8] 1 1 1 1 1 1 1 1
##   .. ..- attr(*, "intercept")= int 1
##   .. ..- attr(*, "response")= int 1
##   .. ..- attr(*, ".Environment")=<environment: R_GlobalEnv>
##   .. ..- attr(*, "predvars")= language list(sonPadres, MntWines, MntFruits, MntMeatProducts, MntFish
##   .. ..- attr(*, "dataClasses")= Named chr [1:9] "factor" "numeric" "numeric" "numeric" ...
##   .. .. ..- attr(*, "names")= chr [1:9] "sonPadres" "MntWines" "MntFruits" "MntMeatProducts" ...
##  $ coefnames   : chr [1:8] "MntWines" "MntFruits" "MntMeatProducts" "MntFishProducts" ...
##  $ xlevels     : Named list()
##  - attr(*, "class")= chr [1:2] "train" "train.formula"
```

```
boxplot(LogReg.fit$resample$Accuracy, xlab = "Accuracy")
```



Idea: Alomejor ya es un poco de machine learning pero se me ocurre tratar de encontrar al mejor grupo de personas a las que ofrecerles descuentos para aumentar las ventas. Por ejemplo si ves que la gente con ingresos altos compra independientemente de si tiene descuento o no a ese tipo de gente no mandar descuento pero si se ve que otro grupo aumenta el consumo cuando dispone de descuentos centralizar los descuentos en ese grupo. (Espero que se haya entendido)