Spaceship-Titanic

Juan Luis González Rodríguez & Rocío González Martínez

2022-06-05

Índice


```
# Package names
packages <- c("tidyr", "dplyr","ggplot2", "keras","reshape2","tidyverse",
              "caret","ROCR", "knitr", 'nortest', "bestNormalize","corrplot",
              "arules")

# Install packages not yet installed
installed_packages <- packages %in% rownames(installed.packages())
if (any(installed_packages == FALSE)) {
install.packages(packages[!installed_packages])
}
# Packages loading
invisible(lapply(packages, library, character.only = TRUE))
```

```
## Warning: package 'keras' was built under R version 4.1.3
```

```
## Warning: package 'ROCR' was built under R version 4.1.3
```

```
## Warning: package 'bestNormalize' was built under R version 4.1.3
```

```
set.seed(15463)
```

1 Contexto

1.1 Descripción del Dataset

El dataset *Spaceship Titanic* [1]. Este dataset es parte de la competición homónima y tiene por objetivo crear un algoritmo para predecir qué pasajeros han desaparecido al colisionar una nave espacial denominada Titanic con una anomalía espaciotemporal. Con el conjunto de datos, se pretende predecir si el pasajero ha desaparecido o no, para enviar a un equipo a rescatarle. Para ello, se facilitan 2 ficheros (separados por entrenamiento y test), Se usará el fichero *train.csv* en uno para limpiar todos los registros y posteriormente se usará este para entrenar al modelo. Con el fichero test podremos probar el modelo (no incluye la variable objetivo).

Descripción de **Train.csv**: Conjunto de datos con información de unos 8 700 pasajeros. Este consta de los campos que se especifican más abajo.

Nombre	Tipo	Descripción
PassengerId	chr	Identificador de cada pasajero. El formato es gggg_pp (gggg hace referencia al grupo de pasajeros y pp al número dentro del grupo). Normalmente los miembros del grupo son familia.
HomePlanet	factor	Platena de origen del pasajero.
CryoSleep	logical	Indica si el pasajero está en animación suspendida durante el viaje o no.
Cabin	chr	Indican la cabina del pasajero. El formato es "plataforma/numero/lado". Lado será P o S
Destination	factor	Indica el nombre del planeta de destino del pasajero.
Age	integer	Indica la edad biológica del pasajero en años en el momento del viaje.
VIP	logical	Indica si el pasajero ha pagado por un servicio VIP o no
RoomService, FoodCouert, ShopingMall, Spa, VRDeck	numeric	Indica la cantidad de dinero que el pasajero ha gastado en cada uno de los servicios
Name	chr	Indica el nombre y apellido del pasajero

Nombre	Tipo	Descripción
Transported	logical	Variable objetivo, indica si el pasajero ha sido transportado a otra dimensión o no (es decir si ha desaparecido).

La estructura del dataset es la siguiente:

```
df <- read.csv("~/MASTER CIENCIA DE DATOS/Tipologia y ciclo de vida de los datos/Practicas/Práctica2/Ej
               colClasses=c("HomePlanet"="factor",
                            "CryoSleep"="logical",
                            "Destination"="factor",
                            "VIP"="logical",
                            "Transported"="logical"))
df$Age <- as.integer(df$Age)

str(df)
```

```
## 'data.frame':    8693 obs. of  14 variables:
##  $ PassengerId : chr  "0001_01" "0002_01" "0003_01" "0003_02" ...
##  $ HomePlanet  : Factor w/ 4 levels "","Earth","Europa",..: 3 2 3 3 2 2 2 2 2 3 ...
##  $ CryoSleep   : logi  FALSE FALSE FALSE FALSE FALSE FALSE ...
##  $ Cabin       : chr  "B/0/P" "F/0/S" "A/0/S" "A/0/S" ...
##  $ Destination : Factor w/ 4 levels "","55 Cancri e",..: 4 4 4 4 4 3 4 4 4 2 ...
##  $ Age         : int  39 24 58 33 16 44 26 28 35 14 ...
##  $ VIP         : logi  FALSE FALSE TRUE FALSE FALSE FALSE ...
##  $ RoomService : num  0 109 43 0 303 0 42 0 0 0 ...
##  $ FoodCourt   : num  0 9 3576 1283 70 ...
##  $ ShoppingMall: num  0 25 0 371 151 0 3 0 17 0 ...
##  $ Spa         : num  0 549 6715 3329 565 ...
##  $ VRDeck      : num  0 44 49 193 2 0 0 NA 0 0 ...
##  $ Name        : chr  "Maham Ofracculy" "Juanna Vines" "Altark Susent" "Solam Susent" ...
##  $ Transported : logi  FALSE TRUE FALSE FALSE TRUE TRUE ...
```

1.2 ¿Por qué es importante y qué pregunta/problema pretende responder?

El objetivo que se persigue con el proyecto es el de, partiendo del conjunto de datos anteriormente comentado, desarrollar un modelo supervisado que permita responder a la pregunta: **¿Ha desaparecido el pasajero que se indica?**

Con ello, la tripulación podrá dirigir los esfuerzos de una manera más eficiente y maximizar las vidas salvadas.

2 Integración y selección de los datos de interes.

Solo hay 1 fichero de origen, por lo que no hay que combinar los datos de diferentes fuentes.

Como ya se tiene a los usuarios identificados a los usuarios en base a los identificadores, no es necesario almacenar sus nombres de cara al análisis. Por otro lado, de los campos *Passenger_id* y *Cabin* se pueden extraer aún más campos como el grupo y número dentro del grupo en el primer caso y la plataforma, número de cabina y lado en el segundo.

Se elimina la variable *Name* y se crean las nuevas variables derivadas.

```
df <- select(df, -Name)
```

```
df <- df %>%
  mutate(PassengerGroup=
           as.character(sapply(strsplit(PassengerId,"_"), `[`, 1))) %>%
  mutate(PassengerNumInGroup=
           as.factor(sapply(strsplit(PassengerId,"_"), `[`, 2))) %>%
  mutate(CabinPlatform =
           as.factor(sapply(strsplit(Cabin,"/"), `[`, 1))) %>%
    mutate(CabinNumber =
           as.integer(sapply(strsplit(Cabin,"/"), `[`, 2))) %>%
    mutate(CabinSide =
           as.factor(sapply(strsplit(Cabin,"/"), `[`, 3)))

df <- select(df, -Cabin)
```

Tras crear las nuevas variables derivadas se elimina *Cabin* porque ya tenemos su información separada. PassengerId no se eliminará porque sirve para identificar los registros. Se muestra un resumen de los campos con la función *summary*.

```
summary(df)
```

```
##  PassengerId         HomePlanet   CryoSleep              Destination
##  Length:8693              : 201   Mode :logical                : 182
##  Class :character   Earth :4602   FALSE:5439      55 Cancri e  :1800
##  Mode  :character   Europa:2131   TRUE :3037      PSO J318.5-22: 796
##                     Mars  :1759   NA's :217       TRAPPIST-1e  :5915
##
##
##
##       Age            VIP           RoomService        FoodCourt
##  Min.   : 0.00   Mode :logical   Min.   :    0.0   Min.   :    0.0
##  1st Qu.:19.00   FALSE:8291      1st Qu.:    0.0   1st Qu.:    0.0
##  Median :27.00   TRUE :199       Median :    0.0   Median :    0.0
##  Mean   :28.83   NA's :203       Mean   :  224.7   Mean   :  458.1
##  3rd Qu.:38.00                   3rd Qu.:   47.0   3rd Qu.:   76.0
##  Max.   :79.00                   Max.   :14327.0   Max.   :29813.0
##  NA's   :179                     NA's   :181       NA's   :183
##   ShoppingMall          Spa              VRDeck        Transported
##  Min.   :    0.0   Min.   :    0.0   Min.   :    0.0   Mode :logical
##  1st Qu.:    0.0   1st Qu.:    0.0   1st Qu.:    0.0   FALSE:4315
##  Median :    0.0   Median :    0.0   Median :    0.0   TRUE :4378
##  Mean   :  173.7   Mean   :  311.1   Mean   :  304.9
##  3rd Qu.:   27.0   3rd Qu.:   59.0   3rd Qu.:   46.0
##  Max.   :23492.0   Max.   :22408.0   Max.   :24133.0
##  NA's   :208       NA's   :183       NA's   :188
##  PassengerGroup     PassengerNumInGroup CabinPlatform   CabinNumber
##  Length:8693        01     :6217        F      :2794   Min.   :   0.0
##  Class :character   02     :1412        G      :2559   1st Qu.: 167.2
##  Mode  :character   03     : 571        E      : 876   Median : 427.0
##                     04     : 231        B      : 779   Mean   : 600.4
##                     05     : 128        C      : 747   3rd Qu.: 999.0
```

```
##                     06     :  75        (Other): 739   Max.   :1894.0
##                     (Other):  59        NA's   : 199   NA's   :199
##  CabinSide
##  P   :4206
##  S   :4288
##  NA's: 199
##
##
##
##
```

Cabe destacar que en *HomePlanet* y en *Destination* hay campos con valores vacíos que no se han considerado como NA's. Por otro lado, Hay algunos campos que presenta NA's que podrán tratarse o desestimarse. También se observan valores extremos en algunos campos.

3 Limpieza de los Datos.

En este apartado se tratará de mejorar la calidad de los datos presentes en base a la falta de calidad. Por límite de extensión del proyecto, nos centraremos en el tratamiento de outliers y de valores nulos.

3.1 Tratamiento valores nulos.

Como se puede ver en el Summary del apartado anterior, en muchos de los campos del conjunto de datos hay registros vacíos que se deben tratar. El listado de campos afectados es el siguiente:

1. HomePlanet
2. Destination
3. Age
4. CryoSleep
5. VIP
6. RoomService
7. FoodCourt
8. ShoppingMall
9. Spa
10. VRDeck
11. CabinPlatform, CabinNumber, CabinSide

En los dos primeros casos, se remapean los registros en blanco de los campos *HomePlanet* y *Destination* por el valor *Unknown*. Con esto, no perdemos información y evitamos confundir a las personas que interpreten los resultados.

```
levels(df$HomePlanet) <- c("Unknown", "Earth", "Europa", "Mars")
levels(df$Destination) <- c("Unknown", "55 Cancri e", "PSO J318.5-22",
                            "TRAPPIST-1e")
```

De los otros casos, siguen existiendo el siguiente número de valores nulos.

```
sapply(df, function(x) sum(length(which(is.na(x)))))
```

```
##         PassengerId          HomePlanet           CryoSleep         Destination
##                   0                   0                 217                   0
##                 Age                 VIP         RoomService           FoodCourt
##                 179                 203                 181                 183
##        ShoppingMall                 Spa              VRDeck         Transported
##                 208                 183                 188                   0
##      PassengerGroup PassengerNumInGroup       CabinPlatform         CabinNumber
##                   0                   0                 199                 199
##           CabinSide
##                 199
```

En primer lugar, se eliminarán los registros del conjunto de datos que contengan un valor vacío en los campos *Age* y *CryoSleep*, así como en los campos *CabinPlatform*, *CabinNumber* y *CabinSide*, ya que el volumen de registros afectados es muy pequeño en comparación con el total de registros del dataset:

```
df <- subset(df, !is.na(df$Age) &
               !is.na(df$CryoSleep) &
               !is.na(df$CabinPlatform) &
               !is.na(df$CabinNumber) &
               !is.na(df$CabinSide),

             select = colnames(df))
```

También eliminamos del conjunto de datos los registros en los que no se informe el campo VIP, pues los nulos de los campos numéricos se informarán a partir de este.

```
df <- subset(df, !is.na(df$VIP), select = colnames(df))
```

Observemos ahora, en un dataframe en el que eliminásemos todos los registros nulos, cómo se comporta el gasto de los diferentes campos en función de si el pasajero es VIP:

```
dfGr <- na.omit(df)

dfGr %>%
  group_by(dfGr$VIP) %>%
  summarize(across(c(RoomService, FoodCourt, ShoppingMall, Spa, VRDeck),
                   mean,
                   na.rm = TRUE))
```

```
## # A tibble: 2 x 6
##   `dfGr$VIP` RoomService FoodCourt ShoppingMall   Spa VRDeck
##   <lgl>            <dbl>     <dbl>        <dbl> <dbl>  <dbl>
## 1 FALSE             221.      436.         177.  303.   281.
## 2 TRUE              350.     1932.         225.  776.  1217.
```

Como se puede ver en la tabla anterior, la diferencia entre el gasto de un pasajero VIP y uno no VIP es considerable. Por esta razón, se decide informar los valores nulos de campos asociados a gastos a partir de la media del gasto en el grupo VIP y no VIP.

```
    roomServiceVIP = mean(df[df$VIP == TRUE,]$RoomService, na.rm = TRUE)
    roomServiceNVIP = mean(df[df$VIP == FALSE,]$RoomService, na.rm = TRUE)
```

```
    foodCourtVIP = mean(df[df$VIP == TRUE,]$FoodCourt, na.rm = TRUE)
    foodCourtNVIP = mean(df[df$VIP == FALSE,]$FoodCourt, na.rm = TRUE)

    shoppingMallVIP = mean(df[df$VIP == TRUE,]$ShoppingMall, na.rm = TRUE)
    shoppingMallNVIP = mean(df[df$VIP == FALSE,]$ShoppingMall, na.rm = TRUE)

    spaVIP = mean(df[df$VIP == TRUE,]$Spa, na.rm = TRUE)
    spaNVIP = mean(df[df$VIP == FALSE,]$Spa, na.rm = TRUE)

    vrdeckVIP = mean(df[df$VIP == TRUE,]$VRDeck, na.rm = TRUE)
    vrdeckNVIP = mean(df[df$VIP == FALSE,]$VRDeck, na.rm = TRUE)
```

```
    df$RoomService[is.na(df$RoomService) & df$VIP == TRUE] <- roomServiceVIP
    df$RoomService[is.na(df$RoomService) & df$VIP == FALSE] <- roomServiceNVIP

    df$FoodCourt[is.na(df$FoodCourt) & df$VIP == TRUE] <- foodCourtVIP
    df$FoodCourt[is.na(df$FoodCourt) & df$VIP == FALSE] <- foodCourtNVIP

    df$ShoppingMall[is.na(df$ShoppingMall) & df$VIP == TRUE] <- shoppingMallVIP
    df$ShoppingMall[is.na(df$ShoppingMall) & df$VIP == FALSE] <- shoppingMallNVIP

    df$Spa[is.na(df$Spa) & df$VIP == TRUE] <- spaVIP
    df$Spa[is.na(df$Spa) & df$VIP == FALSE] <- spaNVIP

    df$VRDeck[is.na(df$VRDeck) & df$VIP == TRUE] <- vrdeckVIP
    df$VRDeck[is.na(df$VRDeck) & df$VIP == FALSE] <- vrdeckNVIP
```

Después de estos tratamientos ya podemos observar cómo el Dataset aparece limpio de valores vacíos:

```
sapply(df, function(x) sum(length(which(is.na(x)))))
```

```
##         PassengerId          HomePlanet           CryoSleep         Destination
##                   0                   0                   0                   0
##                 Age                 VIP         RoomService           FoodCourt
##                   0                   0                   0                   0
##        ShoppingMall                 Spa              VRDeck         Transported
##                   0                   0                   0                   0
##      PassengerGroup PassengerNumInGroup       CabinPlatform         CabinNumber
##                   0                   0                   0                   0
##           CabinSide
##                   0
```

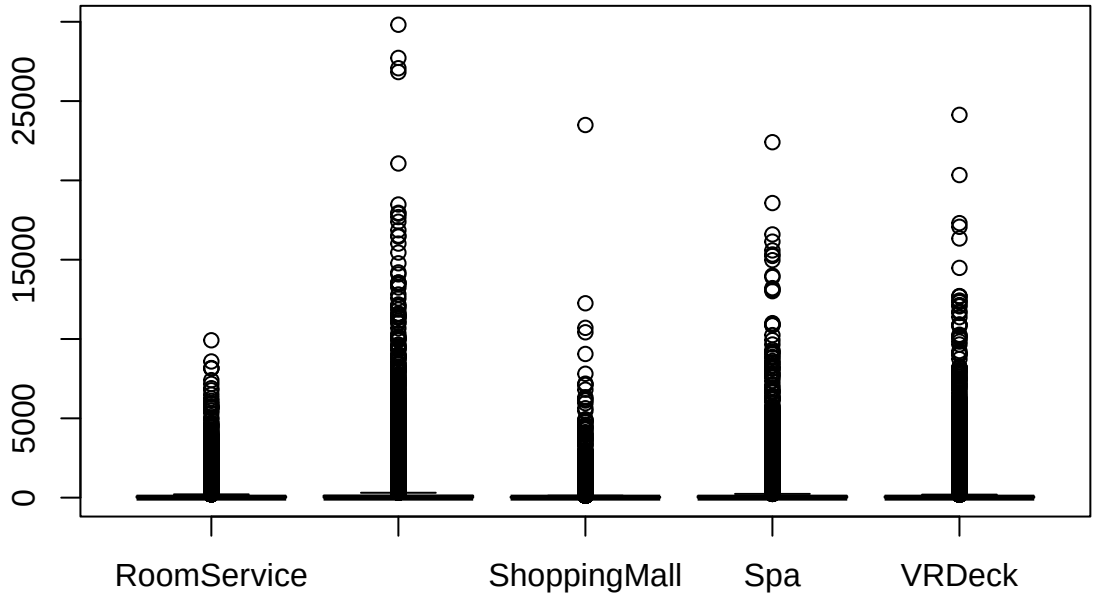
3.2 Tratamiento valores extremos.

Para realizar el análisis de valores extremos, se representan los valores de los campos numéricos en un BoxPlot:

```
boxplotCols <- c("RoomService","FoodCourt","ShoppingMall","Spa","VRDeck")
boxplot(select(df, boxplotCols), col = rainbow(length(boxplotCols)))
```

```
## Note: Using an external vector in selections is ambiguous.
```

```
## i Use ‘all_of(boxplotCols)‘ instead of ‘boxplotCols‘ to silence this message.
## i See <https://tidyselect.r-lib.org/reference/faq-external-vector.html>.
## This message is displayed once per session.
```



Los valores alejados que se pueden ver en el gráfico anterior sí parecen valores atípicos dentro del conjunto de datos, pero en ningún caso se consideran en este proyecto valores extremo, ya que como se puede observar, la distribución de los datos está por encima de los cuartiles. Estos datos atípicos que se comentan, siguen siendo representativos de la variedad de la muestra, y por tanto formarán parte del conjunto de datos que se utilizará para entrenar al modelo.

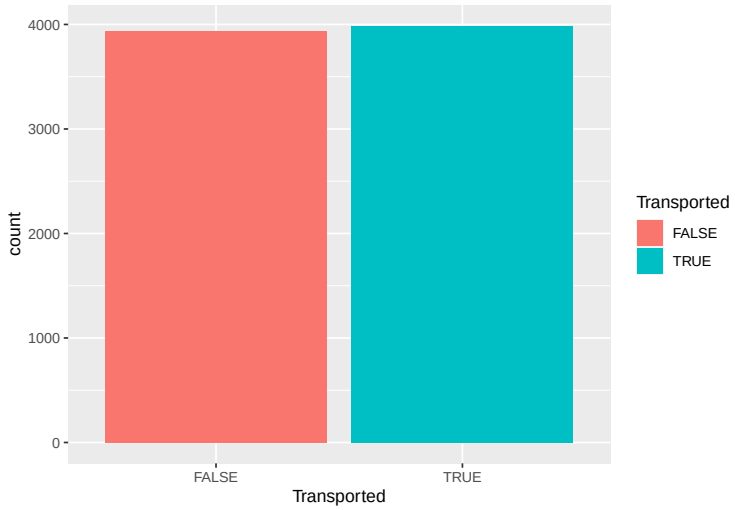
4 Análisis de los datos

4.1 Selección de los grupos de datos

Dentro del conjunto de datos, encontramos un campo que será nuestra variable objetivo a predecir en el modelo que se quiere construir. Esta variable es *Transported*, e indica como se comentaba en el primer apartado, si un pasajero ha desaparecido después de colisionar. Por tanto, los grupos de datos a analizar son, por un lado los pasajeros transportados, y por otro, los no transportados.

Para realizar un primer análisis exploratorio, comprobamos en primer lugar el volumen de datos distribuidos en nuestra variable objetivo (Transported):

```
ggplot(data = df) + geom_bar(mapping = aes(x = Transported, fill = Transported))
```



Como vemos, la mitad de los datos indican que el pasajero desapareció, y la otra mitad indica lo contrario.

```
df %>%
  count(Transported)
```

```
##   Transported    n
## 1       FALSE 3936
## 2        TRUE 3989
```

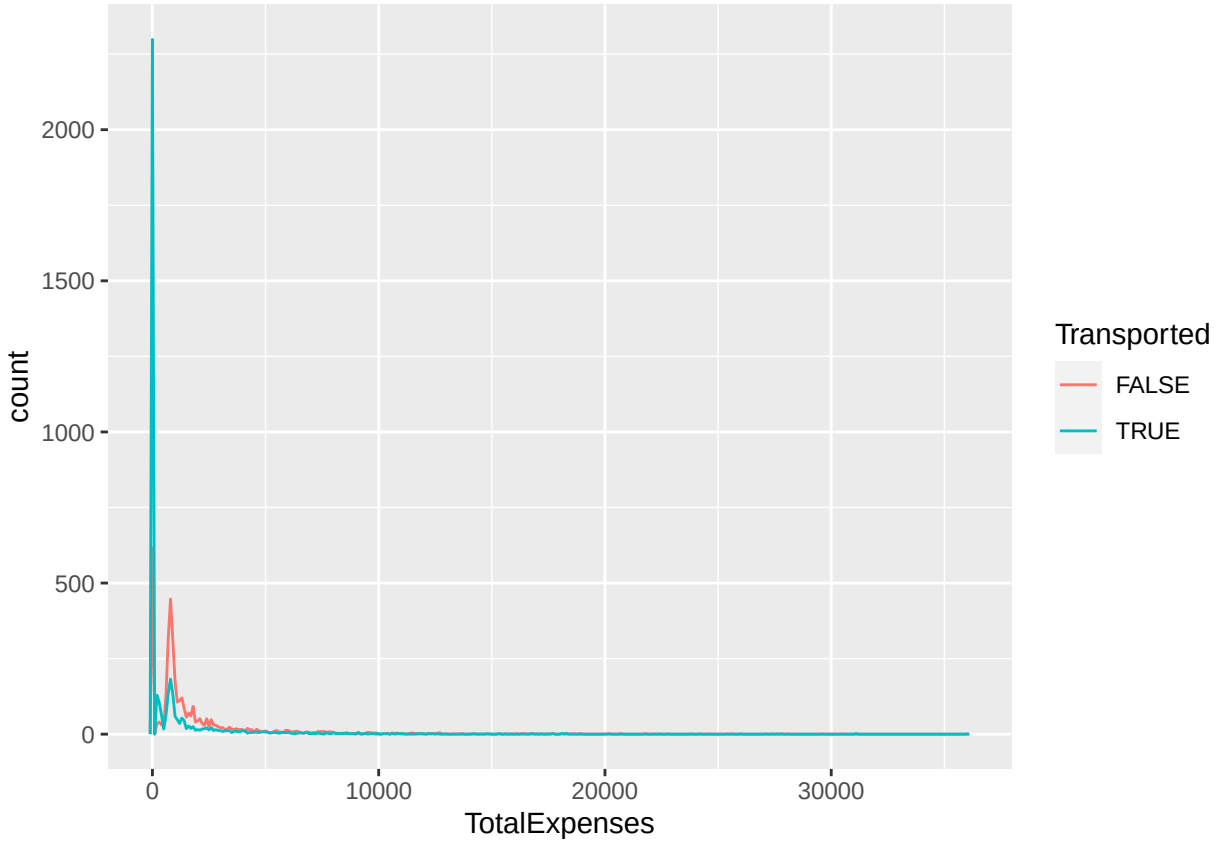
¿Cómo se relaciona esta variable con el resto de campos del conjunto de datos? ¿Se ve afectada por alguna de las otras variables?

Para ello, crearemos un nuevo campo que mida el gasto total de cada pasajero:

```
df_exp <- df
df_exp$TotalExpenses <- df_exp$RoomService + df_exp$FoodCourt + df_exp$ShoppingMall + df_exp$Spa + df_e
df_exp <- select(df_exp, c("HomePlanet", "CryoSleep", "Destination", "Age", "VIP", "PassengerGroup", "C
```

Si miramos la relación entre esta nueva variable y la variable objetivo, obtenemos lo siguiente:

```
ggplot(data = df_exp, mapping = aes(x = TotalExpenses)) + geom_freqpoly(mapping = aes(colour = Transpor
```



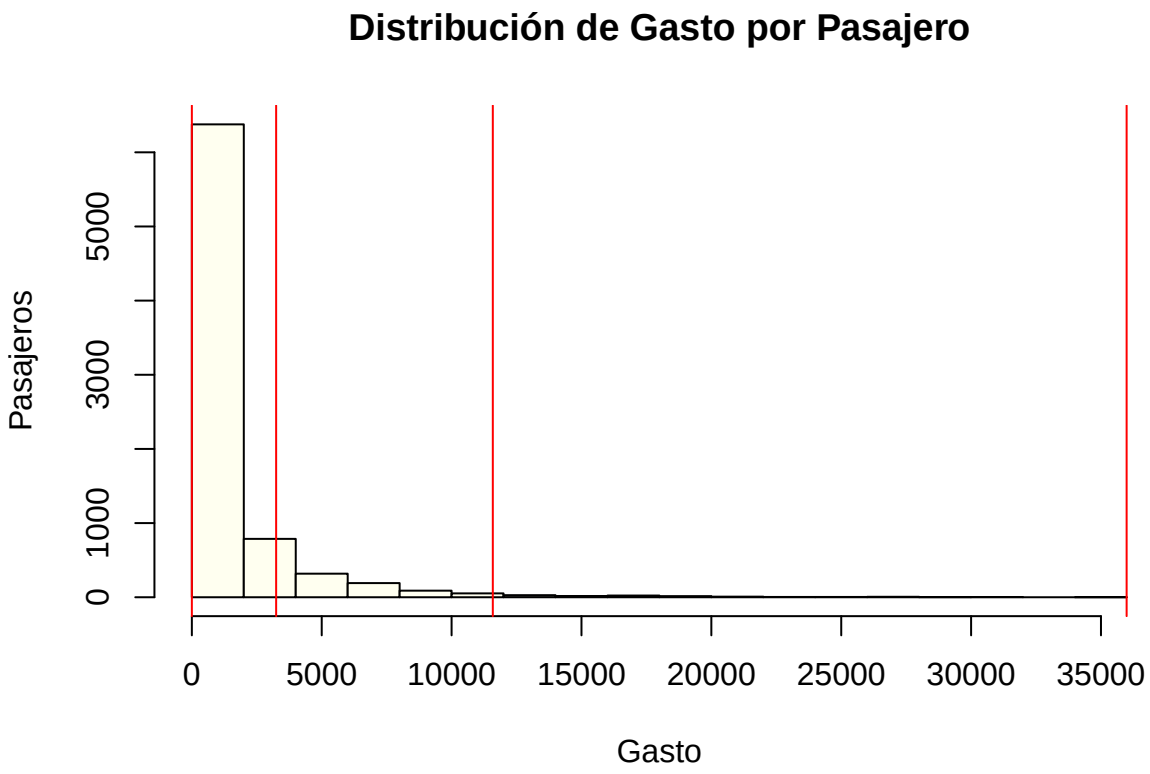
Lo que indica el gráfico anterior es que la mayoría de pasajeros tienen un gasto más cercano a los valores pequeños. Pero está claro que de esta manera no se puede apreciar si existe alguna relación entre ambos campos. Por esta razón, procedemos a discretizar *TotalExpenses* según el método KN:

```
table(discretize(df_exp$TotalExpenses, "cluster" ))
```

```
##
##        [0,3.2e+03) [3.2e+03,1.14e+04) [1.14e+04,3.6e+04]
##               6957                840                 128
```

Si se discretiza mediante KM, la distribución que seguiría es la siguiente:

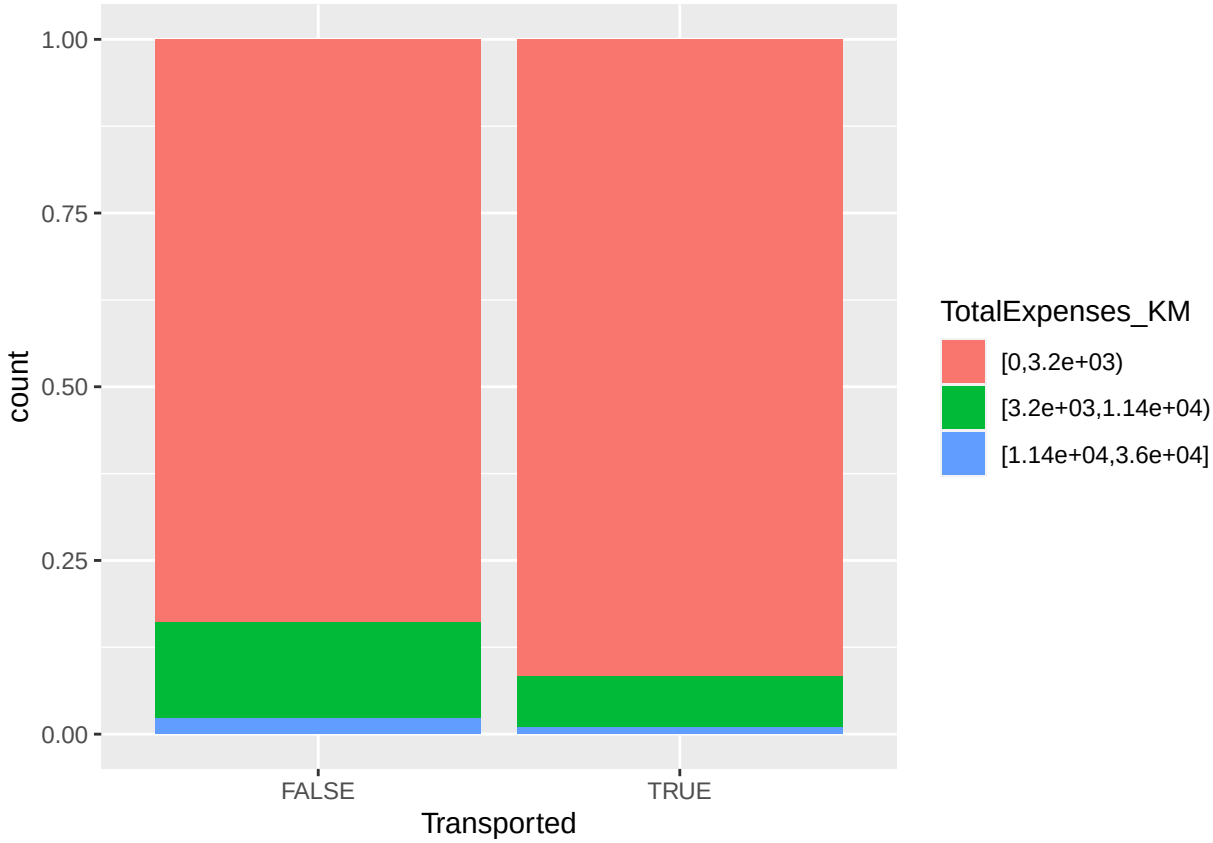
```
hist(df_exp$TotalExpenses,main="Distribución de Gasto por Pasajero",xlab="Gasto", ylab="Pasajeros",col =
abline(v=discretize(df_exp$TotalExpenses, method="cluster", onlycuts=TRUE),col="red")
```




```
df_exp$TotalExpenses_KM <- discretize(df_exp$TotalExpenses, "cluster" )
```

Y se puede, mediante un gráfico de barras, ver la relación entre la variable anterior y la variable *Transported*.

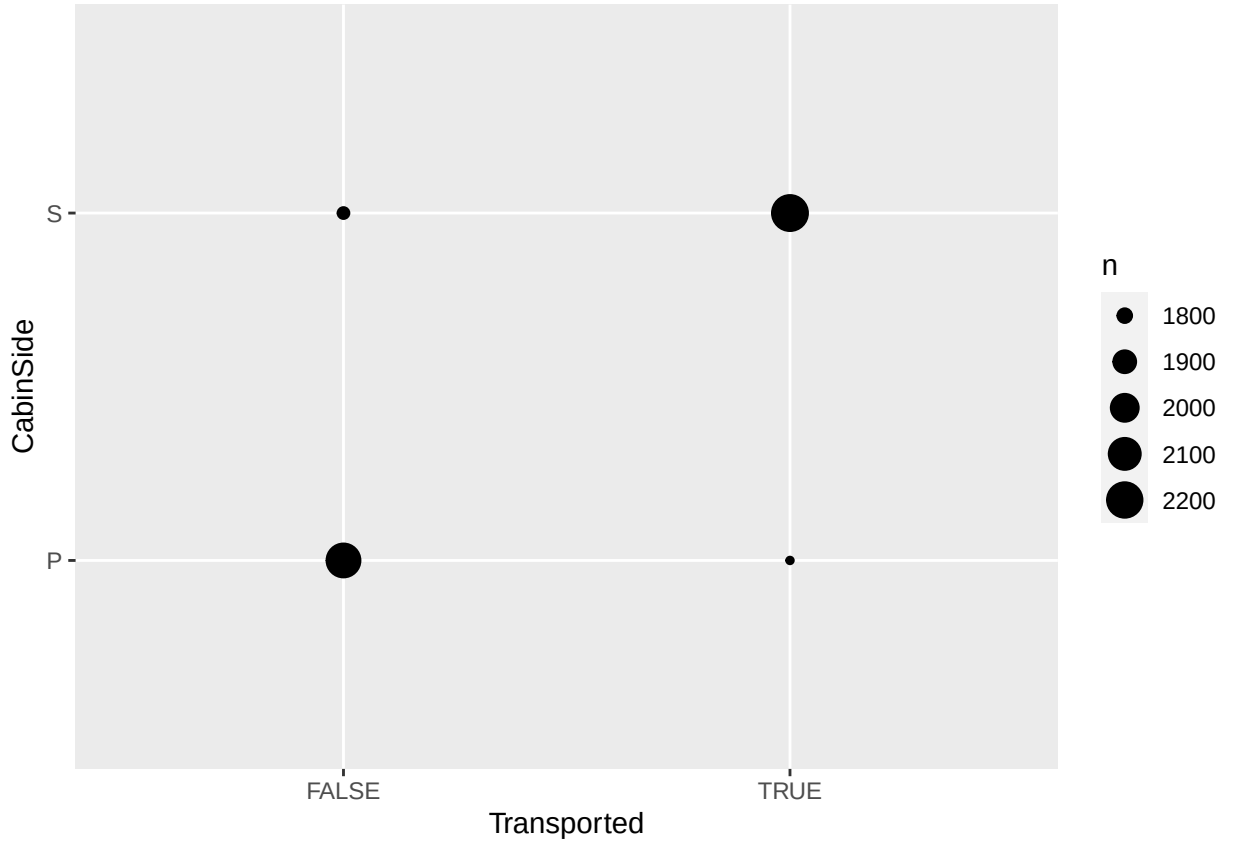
```
ggplot(data = df_exp) + geom_bar(mapping = aes(x = Transported, fill = TotalExpenses_KM), position = "fi
```



En este caso, no parece haber ninguna relación entre ambas variables, ya que el mayor porcentaje de pasajeros transportados y no transportados estaban en el primer segmento de gasto.

Sí parece haber una correlación entre las variables *CabinSide* y *Transported*:

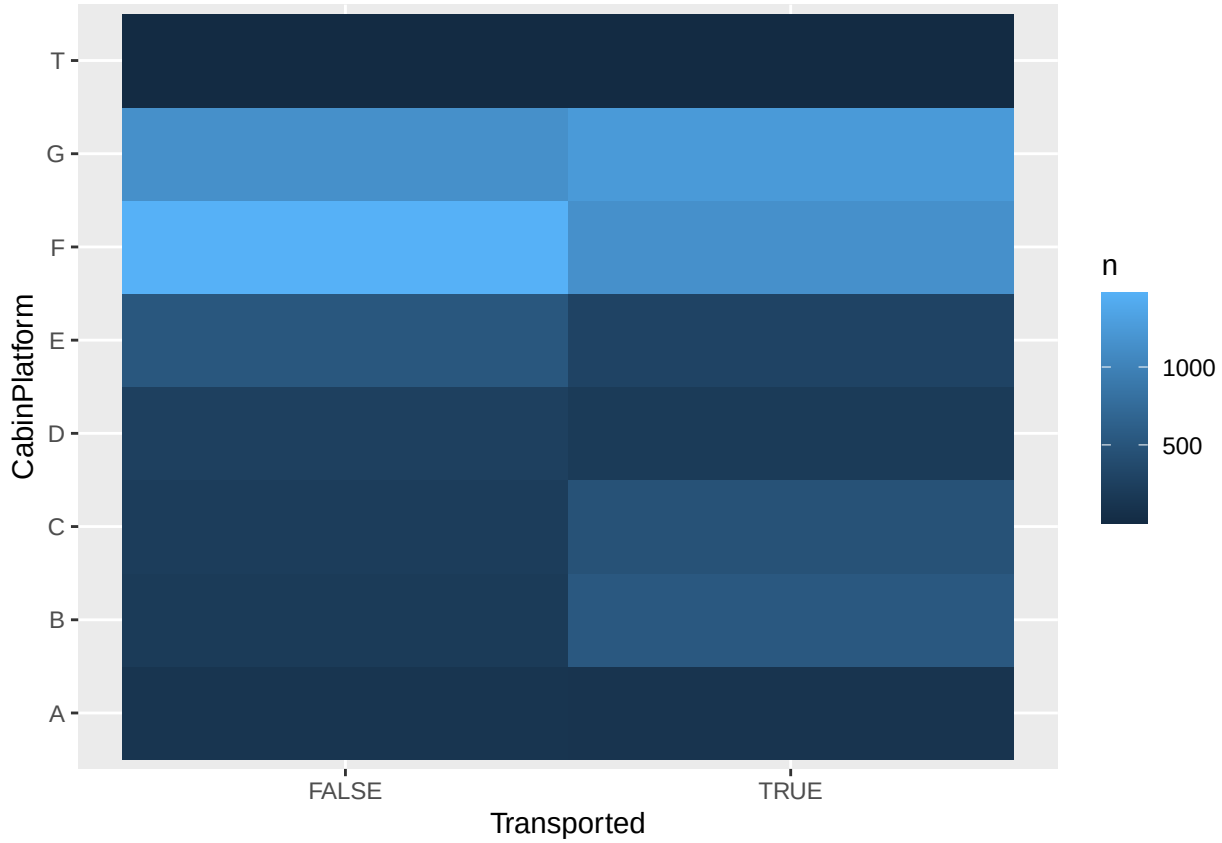
```
ggplot(data = df_exp) + geom_count(mapping = aes(x = Transported, y = CabinSide))
```



En este caso se ve claramente cómo el volumen grande de pasajeros transportados estaban en el lado S, mientras que los no transportados pertenecían al lado P.

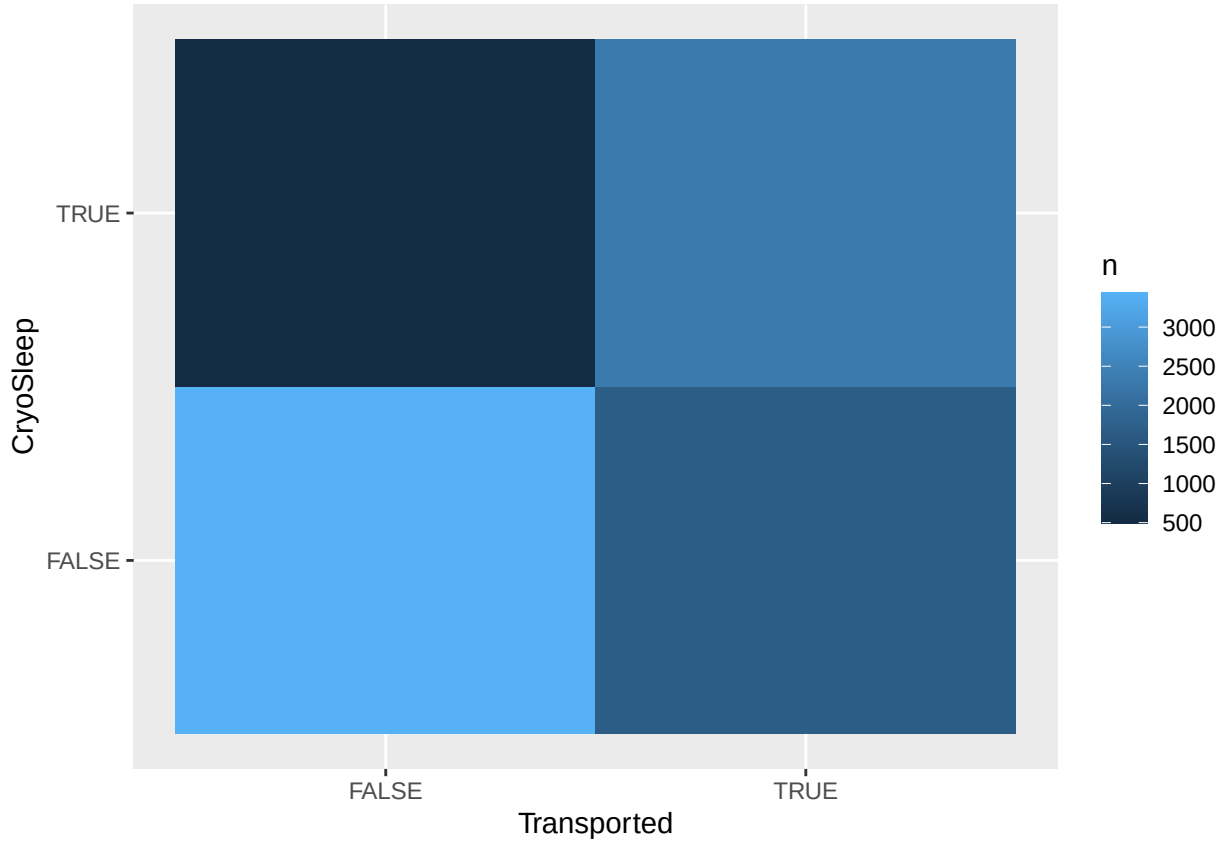
No es tan claro, sin embargo, la afectación de la plataforma de la cabina en la transportación de los pasajeros:

```
df_exp %>%
 count(Transported, CabinPlatform) %>%
 ggplot(mapping = aes(x = Transported, y = CabinPlatform)) + geom_tile(mapping = aes(fill = n))
```



Por último, comprobamos también la relación entre la variable objetivo *Transported* y la variable *CryoSleep*, que indicaba si el pasajero estaba en animación suspendida o no:

```
df_exp %>%
 count(Transported, CryoSleep) %>%
 ggplot(mapping = aes(x = Transported, y = CryoSleep)) + geom_tile(mapping = aes(fill = n))
```



En este caso, aunque no es igual de claro que con la variable *CabinSide*, también parece haber una correlación entre las variables.

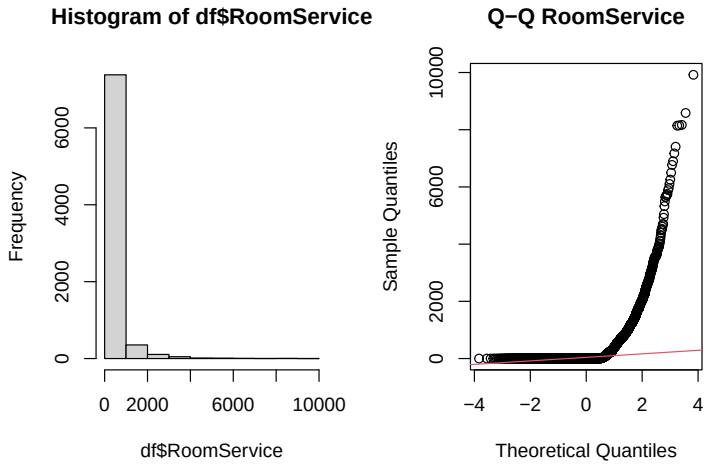
4.2 Comprobación de la normalidad y homogeneidad de la varianza

Debido a la naturaleza de las variables, se deberá estudiar la normalidad y varianza para las variables numéricas. No tiene sentido estudiar si se distribuye normalmente o con que varianza lo hace una variable categórica. Es decir, se estudiará sobre las variables *Age*,*RoomService*, *FoodCourt*, *ShoppingMall*, *Spa* y *VRDeck*.

A continuación se detalla el **estudio de la normalidad**. Debido a que, la cantidad de registros es superior a 5 000, se usará el test de normalidad de *Anderson-Darling*, ya que el test de *Shapiro-Wilk* tiene como limitación un valor máx. de 5 000 registros.

```
par(mfrow=c(1,2))

hist(df$RoomService)
qqnorm(df$RoomService, main="Q-Q RoomService")
qqline(df$RoomService,col=2)
```



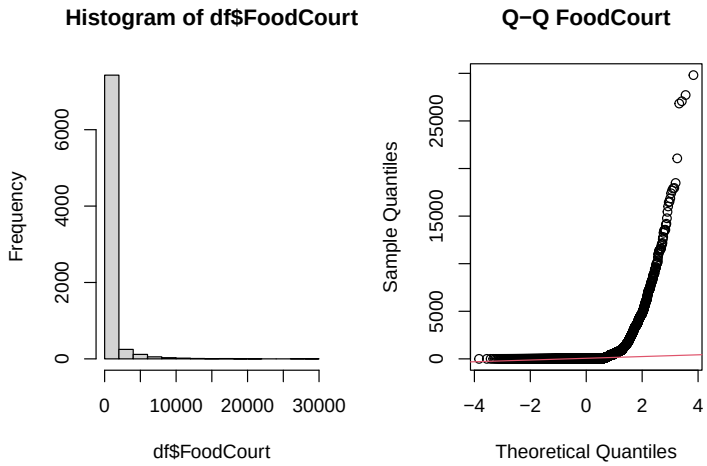

```
ad.test(df$RoomService)
```

```
##
##  Anderson-Darling normality test
##
## data:  df$RoomService
## A = 1696, p-value < 2.2e-16
```

Gráficamente se observa que no sigue una distribución normal. Si realizamos el test de normalidad de *Anderson-Daling* nos da un p-valor $< 0{,}05$. Por lo que se rechaza la hipótesis nula que indica que los datos siguen una distribución normal.

```
par(mfrow=c(1,2))

hist(df$FoodCourt)
qqnorm(df$FoodCourt, main="Q-Q FoodCourt")
qqline(df$FoodCourt,col=2)
```



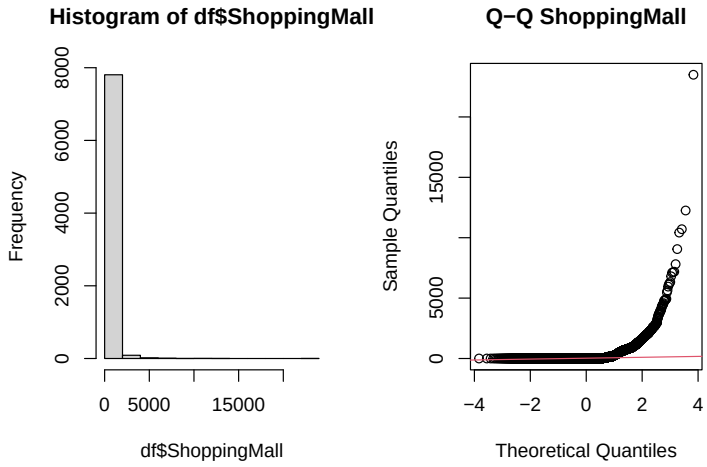
```
ad.test(df$FoodCourt)
```

```
##
##  Anderson-Darling normality test
##
## data:  df$FoodCourt
## A = 1946.7, p-value < 2.2e-16
```

Gráficamente se observa que no sigue una distribución normal. Si realizamos el test de normalidad de *Anderson-Daling* nos da un p-valor $< 0{,}05$. Por lo que se rechaza la hipótesis nula que indica que los datos siguen una distribución normal.

```
par(mfrow=c(1,2))

hist(df$ShoppingMall)
qqnorm(df$ShoppingMall, main="Q-Q ShoppingMall")
qqline(df$ShoppingMall,col=2)
```



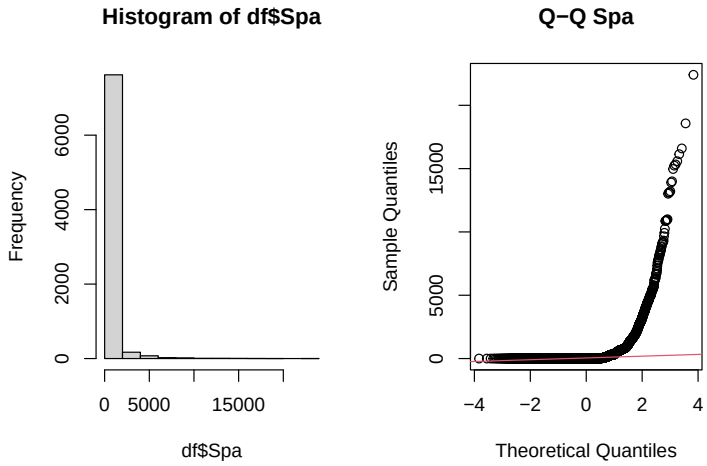

```
ad.test(df$ShoppingMall)
```

```
##
##  Anderson-Darling normality test
##
## data:  df$ShoppingMall
## A = 1788.3, p-value < 2.2e-16
```

Gráficamente se observa que no sigue una distribución normal. Si realizamos el test de normalidad de *Anderson-Daling* nos da un p-valor $< 0{,}05$. Por lo que se rechaza la hipótesis nula que indica que los datos siguen una distribución normal.

```
par(mfrow=c(1,2))

hist(df$Spa)
qqnorm(df$Spa, main="Q-Q Spa")
qqline(df$Spa,col=2)
```



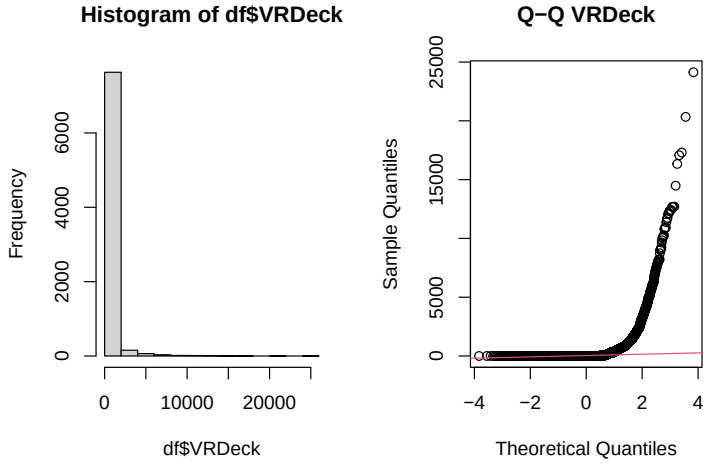

```
ad.test(df$Spa)
```

```
##
##  Anderson-Darling normality test
##
## data:  df$Spa
## A = 1937.5, p-value < 2.2e-16
```

Gráficamente se observa que no sigue una distribución normal. Si realizamos el test de normalidad de *Anderson-Daling* nos da un p-valor $< 0{,}05$. Por lo que se rechaza la hipótesis nula que indica que los datos siguen una distribución normal.

```
par(mfrow=c(1,2))

hist(df$VRDeck)
qqnorm(df$VRDeck, main="Q-Q VRDeck")
qqline(df$VRDeck,col=2)
```



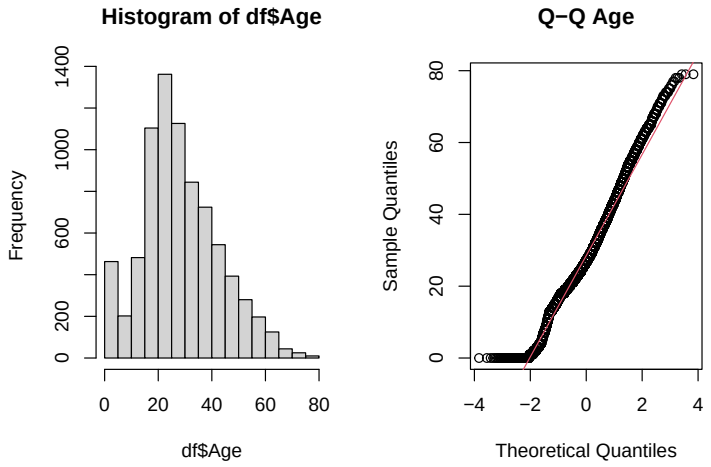
```
ad.test(df$VRDeck)
```

```
##
##  Anderson-Darling normality test
##
## data:  df$VRDeck
## A = 1967.5, p-value < 2.2e-16
```

Gráficamente se observa que no sigue una distribución normal. Si realizamos el test de normalidad de *Anderson-Daling* nos da un p-valor $< 0{,}05$. Por lo que se rechaza la hipótesis nula que indica que los datos siguen una distribución normal.

```
par(mfrow=c(1,2))

hist(df$Age)
qqnorm(df$Age, main="Q-Q Age")
qqline(df$Age,col=2)
```




```
ad.test(df$Age)
```

```
##
##  Anderson-Darling normality test
##
## data:  df$Age
## A = 46.535, p-value < 2.2e-16
```

Gráficamente se observa que no sigue del todo una distribución normal, hay más datos de los esperados en la parte izquierda de la distribución, por lo que aunque se aproxima no aporta información segura. Si realizamos un test formal de normalidad, como el de *Anderson-Daling*, indica un p-valor $< 0{,}05$. Por lo que se rechaza la hipótesis nula que indica que los datos siguen una distribución normal. Si en los siguientes análisis se requiere que los datos sigan una distribución normales, se deberá de proceder a corregir esta.

Tras comprobar la normalidad, se procede a realizar el **análisis de homocedasticidad**. Es decir, se comprueba la igualdad de varianza entre los grupos que se van a comparar. Como los datos no siguen una

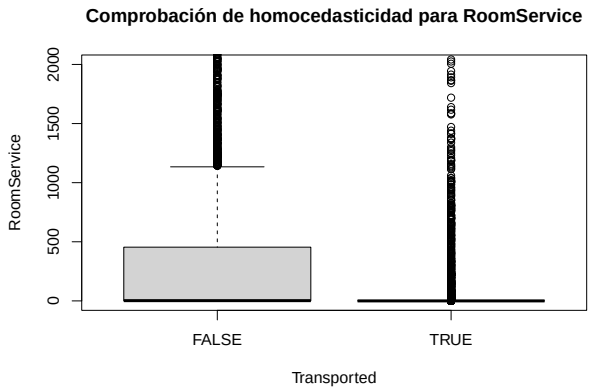
distribución normal no se aplica el test de *Levene*. En su defecto se usará el test de *Fligner-Killeen*. Con ello comprobaremos si la varianza entre los dos grupos (pasajeros transportados y no transportados) es la misma o no.

Aplicamos a todas las variables numéricas:

```
fligner.test(RoomService ~Transported, data=df)
```

```
##
##  Fligner-Killeen test of homogeneity of variances
##
## data:  RoomService by Transported
## Fligner-Killeen:med chi-squared = 2575.7, df = 1, p-value < 2.2e-16
```

```
boxplot(RoomService~Transported, data = df,
        main="Comprobación de homocedasticidad para RoomService",
        ylim=c(0,2000))
```

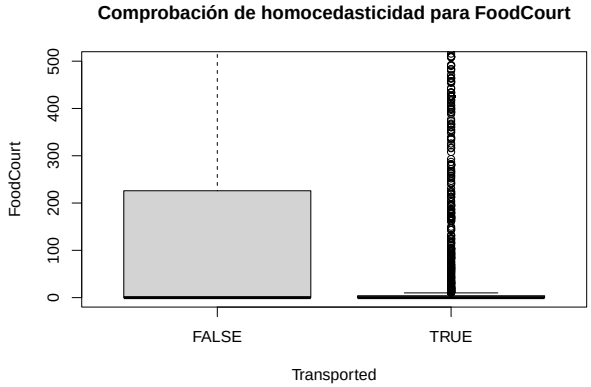


Se rechaza la hipótesis nula de homocedastitcidad. Por lo que la variable *RoomService* presenta varianzas estadísticamente diferentes para los pasajeros transportados y los no transportados. Se puede observar gráficamente como, los gráficos Boxplot apuntan a la misma conclusión. Se ha establecido en el plot un valor máx. de la variable del eje y como 2000 para observar mejor los IQR.

```
fligner.test(FoodCourt ~Transported, data=df)
```

```
##
##  Fligner-Killeen test of homogeneity of variances
##
## data:  FoodCourt by Transported
## Fligner-Killeen:med chi-squared = 100.61, df = 1, p-value < 2.2e-16
```

```
boxplot(FoodCourt~Transported, data = df,
        main="Comprobación de homocedasticidad para FoodCourt",
        ylim=c(0,500))
```

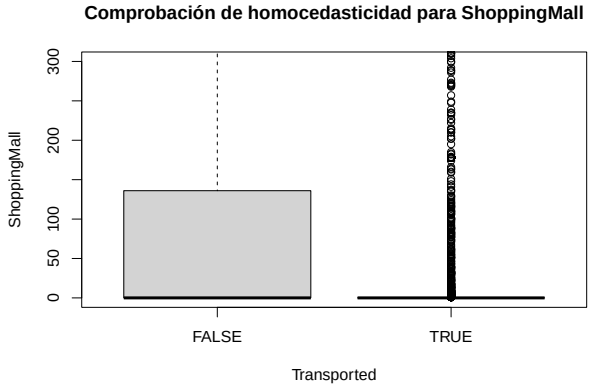


Se rechaza la hipótesis nula, la variable *FoodCour* presenta heterocestasicidad en los grupos definidos por *Transported*. Se puede observar gráficamente como, los gráficos Boxplot apuntan a la misma conclusión. Se ha establecido en el plot un valor máx. de la variable del eje y como 500 para observar mejor los IQR.

```
fligner.test(ShoppingMall ~Transported, data=df)
```

```
##
##  Fligner-Killeen test of homogeneity of variances
##
## data:  ShoppingMall by Transported
## Fligner-Killeen:med chi-squared = 183.47, df = 1, p-value < 2.2e-16
```

```
boxplot(ShoppingMall~Transported, data = df,
        main="Comprobación de homocedasticidad para ShoppingMall",
        ylim=c(0,300))
```

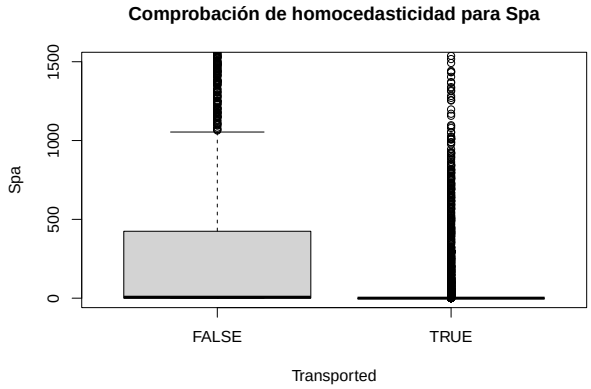


Se rechaza la hipótesis nula de homocedastitcidad. Por lo que la variable *ShoppingMall* presenta varianzas estadísticamente diferentes para los pasajeros transportados y los no transportados. Se puede observar gráficamente como, los gráficos Boxplot apuntan a la misma conclusión. Se ha establecido en el plot un valor máx. de la variable del eje y como 300 para observar mejor los IQR.

```
fligner.test(Spa ~Transported, data=df)
```

```
##
##  Fligner-Killeen test of homogeneity of variances
##
## data:  Spa by Transported
## Fligner-Killeen:med chi-squared = 2572.1, df = 1, p-value < 2.2e-16
```

```
boxplot(Spa~Transported, data = df,
        main="Comprobación de homocedasticidad para Spa",
        ylim=c(0,1500))
```

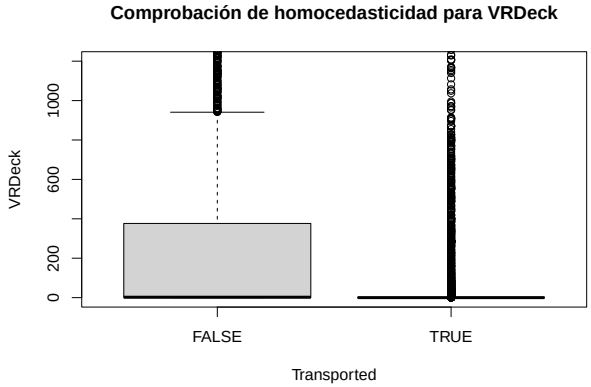


Se rechaza la hipótesis nula, la variable *Spa* presenta heterocestasicidad en los grupos definidos por *Transported*. Se puede observar gráficamente como, los gráficos Boxplot apuntan a la misma conclusión. Se ha establecido en el plot un valor máx. de la variable del eje y como 1500 para observar mejor los IQR.

```
fligner.test(VRDeck ~Transported, data=df)
```

```
##
##  Fligner-Killeen test of homogeneity of variances
##
## data:  VRDeck by Transported
## Fligner-Killeen:med chi-squared = 2399.1, df = 1, p-value < 2.2e-16
```

```
boxplot(VRDeck~Transported, data = df,
        main="Comprobación de homocedasticidad para VRDeck", ylim=c(0,1200))
```

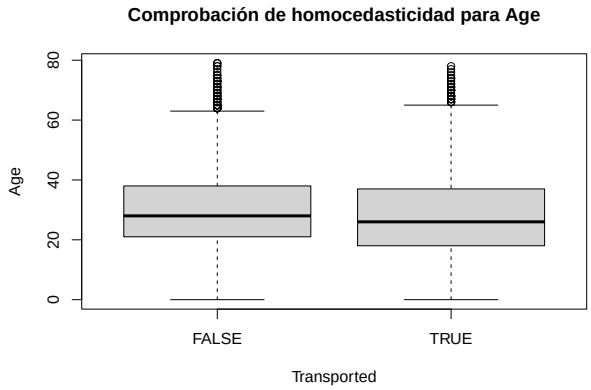


Se rechaza la hipótesis nula, la variable *VRDeck* presenta heterocestasicidad en los grupos definidos por *Transported*. Se puede observar gráficamente como, los gráficos Boxplot apuntan a la misma conclusión. Se ha establecido en el plot un valor máx. de la variable del eje y como 1200 para observar mejor los IQR.

```
fligner.test(Age ~ Transported, data=df)
```

```
##
##  Fligner-Killeen test of homogeneity of variances
##
## data:  Age by Transported
## Fligner-Killeen:med chi-squared = 34.571, df = 1, p-value = 4.109e-09
```

```
boxplot(Age~Transported, data = df,
        main="Comprobación de homocedasticidad para Age")
```



Aunque gráficamente parece que la varianza es homogénea para los dos grupos de la variable *Age* definidos por *Transported*, el test de *Fligner-Killeen* apunta a que no son iguales. El p-valor apportado es inferior a 0.05 por lo que se rechaza la hipótesis nula de igualdad de varianzas.

Por lo que se afirma que **ninguno de los campos numéricos sigue una distribución normal, ni presentan homocedasticidad** para los dos grupos definidos por la variable *Transported*. Se observa en cuanto a la distribución normal. Que en todas las variables (salvo en *Age*) el grupo asignado como "TRUE" tiene muchos más valores en 0 (lo que indica que no han gastado nada en esos servicios) que el asignado como "FALSE."

4.3 Aplicación de pruebas estadísticas

4.3.1 Análisis de Correlaciones

Para realizar el análisis de correlaciones, en primer lugar se creará un conjunto de datos nuevo a partir del anterior, que contenga las mismas variables pero transformadas todas a numéricas:

```
df_cor <- df[,2:17]
df_cor <- df_cor[,-12:-13]
df_cor$HomePlanet <- unclass(df_cor$HomePlanet)
df_cor$Destination <- unclass(df_cor$Destination)
df_cor$VIP <- unclass(df_cor$VIP)
df_cor$Transported <- unclass(df_cor$Transported)
df_cor$CabinPlatform <- unclass(df_cor$HomePlanet)
df_cor$CabinSide <- unclass(df_cor$HomePlanet)
```

```
df_cor$CryoSleep[df_cor$CryoSleep==TRUE] = 1
df_cor$CryoSleep[df_cor$CryoSleep==FALSE] = 0

df_cor$VIP[df_cor$VIP==TRUE] = 1
df_cor$VIP[df_cor$VIP==FALSE] = 0

df_cor$Transported[df_cor$Transported==TRUE] = 1
df_cor$Transported[df_cor$Transported==FALSE] = 0

head(df_cor)
```

```
##   HomePlanet CryoSleep Destination Age VIP RoomService FoodCourt ShoppingMall
## 1          3         0           4  39   0           0         0            0
## 2          2         0           4  24   0         109         9           25
## 3          3         0           4  58   1          43      3576            0
## 4          3         0           4  33   0           0      1283          371
## 5          2         0           4  16   0         303        70          151
## 6          2         0           3  44   0           0       483            0
##    Spa VRDeck Transported CabinPlatform CabinNumber CabinSide
## 1    0      0           0             3           0         3
## 2  549     44           1             2           0         2
## 3 6715     49           0             3           0         3
## 4 3329    193           0             3           0         3
## 5  565      2           1             2           1         2
## 6  291      0           1             2           0         2
```

```
df_cor[] <- lapply(df_cor, function(x) as.numeric(as.character(x)))
head(df_cor)
```

```
##   HomePlanet CryoSleep Destination Age VIP RoomService FoodCourt ShoppingMall
## 1          3         0           4  39   0           0         0            0
## 2          2         0           4  24   0         109         9           25
## 3          3         0           4  58   1          43      3576            0
## 4          3         0           4  33   0           0      1283          371
## 5          2         0           4  16   0         303        70          151
## 6          2         0           3  44   0           0       483            0
```

```
##    Spa VRDeck Transported CabinPlatform CabinNumber CabinSide
## 1    0      0           0             3           0         3
## 2  549     44           1             2           0         2
## 3 6715     49           0             3           0         3
## 4 3329    193           0             3           0         3
## 5  565      2           1             2           1         2
## 6  291      0           1             2           0         2
```

Una vez se tiene el dataset preparado, se calcula la correlación entre todas las variables, teniendo en cuenta que el análisis de Normalidad y Homocedasticidad ha salido negativo en ambos casos (ni los datos son normales ni se cumple la homocedasticidad). Por tanto, se usa el método de Spearman:

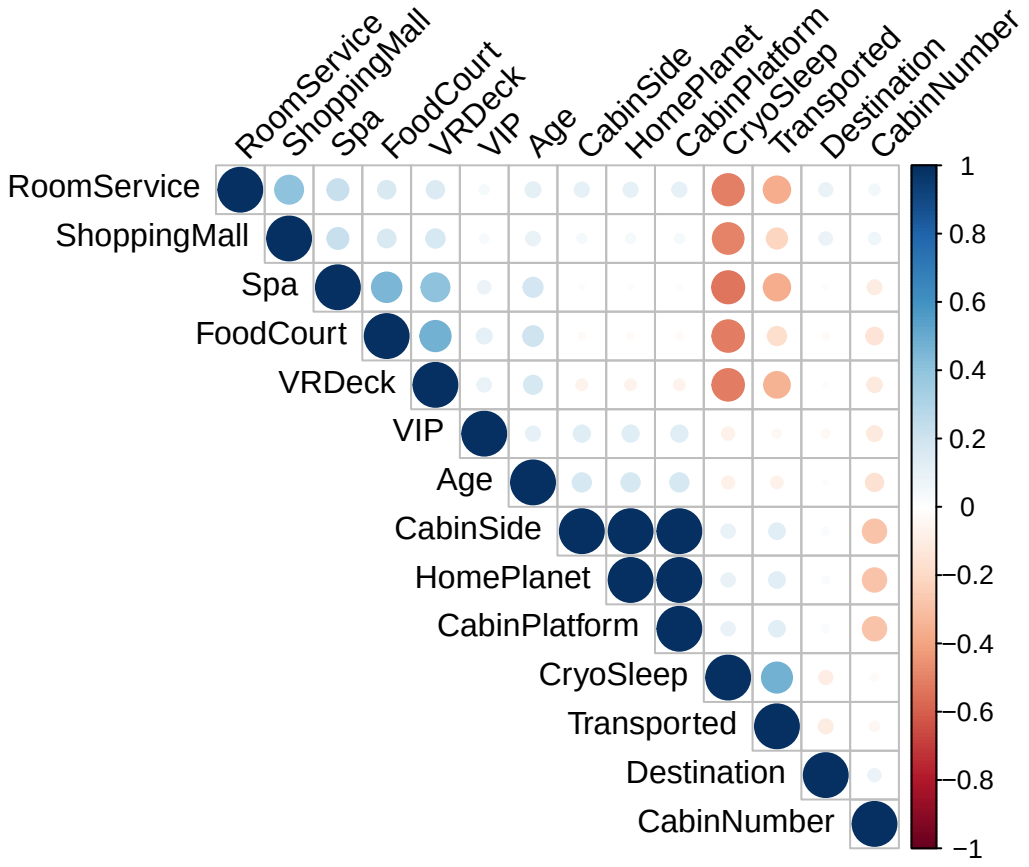
```
res<- cor(df_cor, method = "spearman", use = "complete.obs")
round(res, 2)
```

```
##               HomePlanet CryoSleep Destination   Age   VIP RoomService
## HomePlanet          1.00      0.09        0.03  0.17  0.14        0.11
## CryoSleep           0.09      1.00       -0.09 -0.08 -0.08       -0.50
## Destination         0.03     -0.09        1.00 -0.01 -0.03        0.09
## Age                 0.17     -0.08       -0.01  1.00  0.10        0.12
## VIP                 0.14     -0.08       -0.03  0.10  1.00        0.04
## RoomService         0.11     -0.50        0.09  0.12  0.04        1.00
## FoodCourt          -0.02     -0.52       -0.02  0.20  0.12        0.16
## ShoppingMall        0.04     -0.49        0.08  0.10  0.04        0.41
## Spa                 0.01     -0.53        0.02  0.19  0.08        0.23
## VRDeck             -0.06     -0.52       -0.01  0.17  0.09        0.16
## Transported         0.13      0.47       -0.10 -0.07 -0.03       -0.36
## CabinPlatform       1.00      0.09        0.03  0.17  0.14        0.11
## CabinNumber        -0.28     -0.03        0.08 -0.16 -0.12        0.06
## CabinSide           1.00      0.09        0.03  0.17  0.14        0.11
##               FoodCourt ShoppingMall   Spa VRDeck Transported CabinPlatform
## HomePlanet        -0.02         0.04  0.01  -0.06        0.13          1.00
## CryoSleep         -0.52        -0.49 -0.53  -0.52        0.47          0.09
## Destination       -0.02         0.08  0.02  -0.01       -0.10          0.03
## Age                0.20         0.10  0.19   0.17       -0.07          0.17
## VIP                0.12         0.04  0.08   0.09       -0.03          0.14
## RoomService        0.16         0.41  0.23   0.16       -0.36          0.11
## FoodCourt          1.00         0.16  0.45   0.48       -0.17         -0.02
## ShoppingMall       0.16         1.00  0.23   0.18       -0.21          0.04
## Spa                0.45         0.23  1.00   0.41       -0.36          0.01
## VRDeck             0.48         0.18  0.41   1.00       -0.34         -0.06
## Transported       -0.17        -0.21 -0.36  -0.34        1.00          0.13
## CabinPlatform     -0.02         0.04  0.01  -0.06        0.13          1.00
## CabinNumber       -0.15         0.07 -0.10  -0.11       -0.05         -0.28
## CabinSide         -0.02         0.04  0.01  -0.06        0.13          1.00
##               CabinNumber CabinSide
## HomePlanet          -0.28      1.00
## CryoSleep           -0.03      0.09
## Destination          0.08      0.03
## Age                 -0.16      0.17
## VIP                 -0.12      0.14
## RoomService          0.06      0.11
## FoodCourt           -0.15     -0.02
```

```
## ShoppingMall          0.07      0.04
## Spa                  -0.10      0.01
## VRDeck               -0.11     -0.06
## Transported          -0.05      0.13
## CabinPlatform        -0.28      1.00
## CabinNumber           1.00     -0.28
## CabinSide            -0.28      1.00
```

Como se puede ver en los casos anteriores, no parece haber correlación entre ninguna variable. A continuación, se presentan este coeficiente de correlación en un gráfico:

```
corrplot(res, type = "upper", order = "hclust",
         tl.col = "black", tl.srt = 45)
```



4.3.2 Contraste de hipótesis

Se realizará un contraste de hipótesis en el que se estudiará si el gato total de dos submuestras, siendo una las personas con TRUE en Transported y las segundas las personas con FALSE.

En primer lugar, se crea una variable denominada *amountSpent*, esta será la suma de todas las variables relacionadas con el gasto. Se hace un head para ver los valores de los primeros registros.

```
df <- df %>%
  mutate(amountSpent = RoomService+FoodCourt+ShoppingMall+Spa+VRDeck)
kable(cbind("amountSpent"=head(df)$amountSpent), align = 'c')
```

amountSpent
0
736
10383
5176
1091
774

Con esta, se plantea la pregunta de investigación:

¿Es diferente la media del total gastado de las personas transportadas μ_1 que la media del total gastado de las personas no transportadas μ_2 ?

La formulación del contraste de hipótesis nula y alternativa es la siguiente:

$$H_0 : \mu_1 = \mu_2$$
$$H_1 : \mu_1 \neq \mu_2$$

Como las muestras no siguen una distribución normal se usará un test para realizar el contraste NO paramétrico de dos muestras independientes. Este test será el **test de suma de rangos de Wilcoxon**.

```
amountSpenteTransported <- filter(df, Transported==TRUE)$amountSpent

amountSpenteNOTransported <- filter(df, Transported==FALSE)$amountSpent

wilcox.test(amountSpenteNOTransported, amountSpenteTransported)
```

```
##
##  Wilcoxon rank sum test with continuity correction
##
## data:  amountSpenteNOTransported and amountSpenteTransported
## W = 11486818, p-value < 2.2e-16
## alternative hypothesis: true location shift is not equal to 0
```

Con un p-valor inferior al valor de significancia de 0.05, no hay evidencia suficiente para aceptar la hipótesis nula. Por lo que se puede concluir que la media de la cantidad total de dinero gastada por las personas Transportadas y las No transportadas no es estadísticamente igual.

4.3.3 Modelo de regresión logarítmica

Se usará un modelo de regresivo logística que permitirá identificar a los pasajeros entre transportados o no transportados. Para ello hará uso de las variables explicativas que determinarán en la medida de lo posible si estos pasajeros toman un valor u otro.

Se parten los datos en *df* para entrenar al modelo con un 70% de los casos y en *df_validation* para medir la bondad del modelo con el 30% restante.

```
df_raw <- df

Index <- createDataPartition(df_raw$Transported, p=0.7, list=FALSE,times=1)

df <- df_raw[Index,]
df_validation <- df_raw[-Index,]
```

Aunque no se incluye en la práctica por motivos de extensión. Se han realizado análisis previos y se ha determinado que hay variables que empeoran el modelo, por lo que estas se eliminan en el entrenamiento y en la validación.

```
df1 <- select(df, -c(PassengerId, PassengerGroup, HomePlanet, Destination,VIP, PassengerNumInGroup, Cabi
```

Se entrena el modelo con el dataset previamente creado. Se presupone que lasvariables no son bariables de confusión y no presentan interacción entre ellas.

```
model_log <- glm(Transported ~ ., data = df1, family = binomial(link='logit'))
```

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

```
summary(model_log)
```

```
##
## Call:
## glm(formula = Transported ~ ., family = binomial(link = "logit"),
##     data = df1)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.8133  -0.7359   0.0393   0.6877   3.3571
##
## Coefficients:
##                 Estimate Std. Error z value Pr(>|z|)
## (Intercept)   -4.757e-01  8.715e-02  -5.458 4.81e-08 ***
## CryoSleepTRUE  1.842e+00  8.468e-02  21.751  < 2e-16 ***
## RoomService   -1.283e-03  1.057e-04 -12.140  < 2e-16 ***
## FoodCourt      7.307e-04  4.722e-05  15.473  < 2e-16 ***
## ShoppingMall   7.306e-04  7.684e-05   9.508  < 2e-16 ***
## Spa           -1.513e-03  1.209e-04 -12.516  < 2e-16 ***
## VRDeck        -1.323e-03  1.088e-04 -12.159  < 2e-16 ***
## CabinNumber   -1.478e-04  6.779e-05  -2.181   0.0292 *
## CabinSideS     5.530e-01  6.888e-02   8.028 9.91e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 7692.3  on 5548  degrees of freedom
## Residual deviance: 5228.0  on 5540  degrees of freedom
## AIC: 5246
##
## Number of Fisher Scoring iterations: 7
```

A continuación se evalúa el mismo. Para ello se predicen los valores del conjunto *df_validation* y se comparan con los valres reales del mismo.

```
fitted.results <- predict(model_log,
                          newdata = select(df_validation, c(CryoSleep,
                                                            RoomService,
```

```
                                                        FoodCourt,
                                                        ShoppingMall,
                                                        Spa,VRDeck,
                                                        CabinNumber,
                                                        CabinSide))
                           ,type = "response")
```

```
fitted.results_FACTOR <- ifelse(fitted.results > 0.5,TRUE,FALSE)

resultado_comprobacion <- cbind("ID"=df_validation$PassengerId,
                                "Transported"=df_validation$Transported,
                                "Prediction"=fitted.results_FACTOR)

df_validation_info <- as.data.frame(resultado_comprobacion)
kable(head(df_validation_info),align='c', row.names=FALSE)
```

ID	Transported	Prediction
0004_01	TRUE	FALSE
0007_01	TRUE	TRUE
0008_02	TRUE	TRUE
0009_01	TRUE	TRUE
0016_01	TRUE	TRUE
0020_01	FALSE	TRUE

Se convierten las variables *Transported* (valor real) y *Prediction* (valor predicho) y se mide la bondad del modelo con la matriz de confusión mediante la función *confusionMatrix).

```
df_validation_info$Transported <- as.factor(df_validation_info$Transported)
df_validation_info$Prediction <- as.factor(df_validation_info$Prediction)

confusionMatrix(df_validation_info$Transported, df_validation_info$Prediction)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction FALSE TRUE
##      FALSE   924  256
##      TRUE    239  957
##
##                Accuracy : 0.7917
##                  95% CI : (0.7748, 0.8078)
##     No Information Rate : 0.5105
##     P-Value [Acc > NIR] : <2e-16
##
##                   Kappa : 0.5833
##
##  Mcnemar's Test P-Value : 0.4721
##
##             Sensitivity : 0.7945
##             Specificity : 0.7890
```

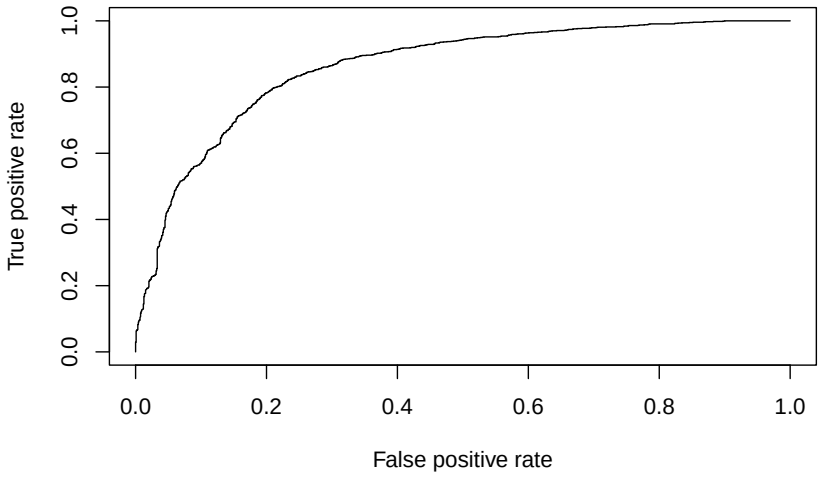
```
##          Pos Pred Value : 0.7831
##          Neg Pred Value : 0.8002
##              Prevalence : 0.4895
##          Detection Rate : 0.3889
##    Detection Prevalence : 0.4966
##       Balanced Accuracy : 0.7917
##
##        'Positive' Class : FALSE
##
```

La clase positiva se categoriza como FALSE (no desaparecido).

Se observa una buena predicción de las clases. Con una exactitud del 80,38%. La tasa de positivos que se han asignado como positivos es del 81,27%. mientras que los negativos identificado como auténticos negativos es del 79,55%. De los verdaderos.

La precisión, es decir, los datos clasificados como positivos y que realmente lo son es del 78,73%.

```
pr <- prediction(fitted.results, df_validation$Transported)
prf <- performance(pr, measure = "tpr", x.measure = "fpr")
plot(prf)
```




```
auc <- performance(pr, measure = "auc")
auc <- auc@y.values[[1]]
auc
```

```
## [1] 0.8619569
```

Vemos como la curva ROC nos da un valor próxima a la esquina superior derecha y un área del 0.85, muy próximo a 1.

Se observa que el modelo es bueno prediciendo si los pasajeros han desaparecido o no.

[1] Kaggle, "Spaceship titanic." kaggle.com; Kaggle, 2022.Available: https://www.kaggle.com/competitions/spaceship-titanic/overview