Tipologáa y ciclo de vida de los datos: Práctica 2: Limpieza y validación de los datos

Autores: Youness El Guennouni y Mario Gutiérrez Calvo de Mora

Mayo 2019

Contents

Introducción

---

trabajaremos con el juego de datos “Titanic” que recoge datos sobre el famoso crucero y sobre el que es fácil realizar tareas de clasificación predictiva sobre la variable “Survived”.

Las actividades que llevaremos a cabo en esta práctica suelen enmarcarse en las fases iniciales de un proyecto de mineráa de datos y consisten en la selección de caraterásticas o variables y la preparación del juego de datos para posteriormente ser consumido por un algoritmo.

Primeramente realizaremos el estudio de las variables de un juego de datos, es decir, haremos un trabajo descriptivo del mismo. Y de forma posterior, realizaremos el estudio de algoritmos predictivos y las conclusiones que se pueden extraer del estudio.

Siguiendo las principales etapas de un proyecto analático, las diferentes tareas a realizar (y justificar) son las siguientes:

1. Descripción del dataset. Â¿Por qué es importante y qué pregunta/problema pretende responder?
2. Integración y selección de los datos de interés a analizar.
3. Limpieza de los datos. 3.1. ¿Los datos contienen ceros o elementos vacáos? ¿Cómo gestionaráas cada uno de estos casos? 3.2. Identificación y tratamiento de valores extremos.
4. Análisis de los datos. 4.1. Selección de los grupos de datos que se quieren analizar/comparar (planificación de los análisis a aplicar). 4.2. Comprobación de la normalidad y homogeneidad de la varianza. 4.3. Aplicación de pruebas estadásticas para comparar los grupos de datos. En función de los datos y el objetivo del estudio, aplicar pruebas de contraste de hipótesis, correlaciones, regresiones, etc. Aplicar al menos tres métodos de análisis diferentes.
5. Representación de los resultados a partir de tablas y gráficas.
6. Resolución del problema. A partir de los resultados obtenidos, Â¿cuáles son las conclusiones? Â¿Los resultados permiten responder al problema?
7. Código: Hay que adjuntar el código, preferiblemente en R, con el que se ha realizado la limpieza, análisis y representación de los datos. Si lo preferás, también podéis trabajar en Python.

Descripción del dataset. Â¿Por qué es importante y qué pregunta/problema pretende responder?

El conjunto de datos objeto de análisis se ha obtenido a partir de este enlace en Kaggle y está constituido por 12 caracterásticas (columnas) que presentan a los 891 pasajeros (filas o registros). Entre los campos de este conjunto de datos, encontramos los siguientes:

- **PassengerId**: identificador unico del pasajero.
- **Survived**: Si el pasajero ha sobrevivido (1) o no (0)
- **Pclass**: En que clase viajaba
- **Name**: Nombre de pasajero

- **Sex**: genéro de pasajero
- **SibSp**: Numero de hermanos / cónyuges a bordo
- **Parch**: Numero de padres / hijos a bordo
- **Ticket**: Numero de ticket
- **Fare**: tárifa del viaje
- **Cabin**: Cabina
- **Embarked**: El puerto desde el cual ha embarcado el pasajero (C- Cherbourg, S - Southampton, Q - Queenstown)

Es importante saber a que preguntas debemos de responder para definir un objetivo claro y no desviarnos de ello. En nuestro caso el problema que debemos de solventar seráa Â¿Que factores influyen directamente o indirectamente en la superviviencia o no de un pasajero? Además, se podrá proceder a crear modelos de regresión que permitan predecir la supervivencia o no de un pasajero en función de sus caráctiristas y contrastes de hipótesis que ayuden a identificar propiedades interesantes en las muestras que puedan ser inferidas con respecto a la población.

Integración y selección de los datos de interés a analizar

Desde el análisis de los datos podemos descartar algunos factores desde el inicio como el nÃºmero de ticket, tarifa o nombre de pasajero. Otro dato que decidimos no tenerle en cuenta es la cabina ya que más de 70% vienen vacios.

Lectura de ficheros

Cargar a los archivos `train.csv` y`test.csv`. Una vez cargado el archivo, valida que los tipos de datos son los correctos. Si no es asá, conviertelos al tipo oportuno.

- Archivo de datos (`train.csv`)

```
train <- read.csv( "train.csv")
head(train)
```

```
##   PassengerId Survived Pclass
## 1           1        0      3
## 2           2        1      1
## 3           3        1      3
## 4           4        1      1
## 5           5        0      3
## 6           6        0      3
##                                                 Name    Sex Age SibSp
## 1                            Braund, Mr. Owen Harris   male  22     1
## 2 Cumings, Mrs. John Bradley (Florence Briggs Thayer) female  38     1
## 3                             Heikkinen, Miss. Laina female  26     0
## 4       Futrelle, Mrs. Jacques Heath (Lily May Peel) female  35     1
```

```
## 5                            Allen, Mr. William Henry   male  35     0
## 6                                    Moran, Mr. James   male  NA     0
##   Parch           Ticket    Fare Cabin Embarked
## 1     0        A/5 21171  7.2500              S
## 2     0         PC 17599 71.2833   C85        C
## 3     0 STON/O2. 3101282  7.9250              S
## 4     0           113803 53.1000  C123        S
## 5     0           373450  8.0500              S
## 6     0           330877  8.4583              Q
```

```
sapply( train, class)
```

```
## PassengerId    Survived      Pclass        Name         Sex         Age
##   "integer"   "integer"   "integer"    "factor"    "factor"   "numeric"
##       SibSp       Parch      Ticket        Fare       Cabin    Embarked
##   "integer"   "integer"    "factor"   "numeric"    "factor"    "factor"
```

- Archivo de los tests (`test.csv`)

```
test <- read.csv( "test.csv")
head(test)
```

```
##   PassengerId Pclass                                          Name    Sex
## 1         892      3                              Kelly, Mr. James   male
## 2         893      3              Wilkes, Mrs. James (Ellen Needs) female
## 3         894      2                     Myles, Mr. Thomas Francis   male
## 4         895      3                              Wirz, Mr. Albert   male
## 5         896      3 Hirvonen, Mrs. Alexander (Helga E Lindqvist) female
## 6         897      3                    Svensson, Mr. Johan Cervin   male
##    Age SibSp Parch  Ticket    Fare Cabin Embarked
## 1 34.5     0     0  330911  7.8292              Q
## 2 47.0     1     0  363272  7.0000              S
## 3 62.0     0     0  240276  9.6875              Q
## 4 27.0     0     0  315154  8.6625              S
## 5 22.0     1     1 3101298 12.2875              S
## 6 14.0     0     0    7538  9.2250              S
```

```
sapply( test, class)
```

```
## PassengerId      Pclass        Name         Sex         Age       SibSp
##   "integer"   "integer"    "factor"    "factor"   "numeric"   "integer"
##       Parch      Ticket        Fare       Cabin    Embarked
##   "integer"    "factor"   "numeric"    "factor"    "factor"
```

Borrar a las columnas innecesarias

```
train <- select(train, -Name, -Fare, -Ticket, -Cabin)
test  <- select(test, -Name, -Fare, -Ticket, -Cabin)
head (train)
```

```
##   PassengerId Survived Pclass    Sex Age SibSp Parch Embarked
## 1           1        0      3   male  22     1     0        S
## 2           2        1      1 female  38     1     0        C
## 3           3        1      3 female  26     0     0        S
## 4           4        1      1 female  35     1     0        S
## 5           5        0      3   male  35     0     0        S
## 6           6        0      3   male  NA     0     0        Q
```

Limpieza de datos

En esta sección vamos a llevar a cabo el proceso de limpieza de datos que consiste en:

Los datos contienen ceros o elementos vacíos

A continuación vamos a detectar a los valores vac?os y nulos.

```
# Estad?sticas de valores vacáos
colSums(is.na(train))
```

```
## PassengerId    Survived      Pclass         Sex         Age       SibSp
##           0           0           0           0         177           0
##       Parch    Embarked
##           0           0
```

```
colSums(train=="")
```

```
## PassengerId    Survived      Pclass         Sex         Age       SibSp
##           0           0           0           0          NA           0
##       Parch    Embarked
##           0           2
```

Llegados a este punto debemos decidir cómo manejar estos registros que contienen valores desconocidos para algÃºn campo. Una opción podráa ser eliminar esos registros que incluyen este tipo de valores, pero ello supondráa desaprovechar información.
Como alternativa, se empleará un método de imputación de valores basado en la similitud o diferencia entre los registros: la imputación basada en k vecinos más próximos. La elección de esta alternativa se realiza bajo la hipótesis de que nuestros registros guardan cierta relación. No obstante, es mejor trabajar con datos aproximados que con los propios elementos vacíos, ya que obtendremos análisis con menor margen de error.

```
#Para los valores perdidos procedemos con aplicar la distancia de Gower
#train$Age <- kNN(train)$Age
#train$Embarked <- kNN(train)$Embarked
#train$Parch <- kNN(train)$Parch
#train$SibSp <- kNN(train)$SibSp
#train$Survived <- kNN(train)$Survived
train <- kNN(train)

#Rempazar la edad por la media
train$Age[is.na(train$Age)] <- winsor.mean(train$Age,trim=0.05)

sapply(train, function(x) sum(is.na(x)))
```

```
##     PassengerId        Survived          Pclass             Sex
##               0               0               0               0
##             Age           SibSp           Parch        Embarked
##               0               0               0               0
## PassengerId_imp    Survived_imp      Pclass_imp         Sex_imp
##               0               0               0               0
##         Age_imp       SibSp_imp       Parch_imp    Embarked_imp
##               0               0               0               0
```

Discretizamos cuando tiene sentido y en función de cada variable.

```
# ¿Para qué variables tendráa sentido un proceso de discretización?
apply(train,2, function(x) length(unique(x)))
```

```
##     PassengerId        Survived          Pclass             Sex
##             891               2               3               2
##             Age           SibSp           Parch        Embarked
##              88               7               7               4
## PassengerId_imp    Survived_imp      Pclass_imp         Sex_imp
##               1               1               1               1
##         Age_imp       SibSp_imp       Parch_imp    Embarked_imp
##               2               1               1               1
```

```
# Discretizamos las variables con pocas clases
cols<-c("Survived","Pclass","Sex","Embarked")
for (i in cols){
  train[,i] <- as.factor(train[,i])
}

# Después de los cambios, analizamos la nueva estructura del juego de datos
str(train)
```

```
## 'data.frame':    891 obs. of  16 variables:
##  $ PassengerId    : int  1 2 3 4 5 6 7 8 9 10 ...
##  $ Survived       : Factor w/ 2 levels "0","1": 1 2 2 2 1 1 1 1 2 2 ...
##  $ Pclass         : Factor w/ 3 levels "1","2","3": 3 1 3 1 3 3 1 3 3 2 ...
##  $ Sex            : Factor w/ 2 levels "female","male": 2 1 1 1 2 2 2 2 1 1 ...
##  $ Age            : num  22 38 26 35 35 21 54 2 27 14 ...
##  $ SibSp          : int  1 1 0 1 0 0 0 3 0 1 ...
##  $ Parch          : int  0 0 0 0 0 0 0 1 2 0 ...
##  $ Embarked       : Factor w/ 4 levels "","C","Q","S": 4 2 4 4 4 3 4 4 4 2 ...
##  $ PassengerId_imp: logi  FALSE FALSE FALSE FALSE FALSE FALSE ...
##  $ Survived_imp   : logi  FALSE FALSE FALSE FALSE FALSE FALSE ...
##  $ Pclass_imp     : logi  FALSE FALSE FALSE FALSE FALSE FALSE ...
##  $ Sex_imp        : logi  FALSE FALSE FALSE FALSE FALSE FALSE ...
##  $ Age_imp        : logi  FALSE FALSE FALSE FALSE FALSE TRUE ...
##  $ SibSp_imp      : logi  FALSE FALSE FALSE FALSE FALSE FALSE ...
##  $ Parch_imp      : logi  FALSE FALSE FALSE FALSE FALSE FALSE ...
##  $ Embarked_imp   : logi  FALSE FALSE FALSE FALSE FALSE FALSE ...
```

Identificación y tratamiento de valores extremos

- Cuadro de las estimaciónes no robustas y robustas.

En el siguiente cuadro se van a mostrar a las estimaciones no robustas y robustas por un posible uso a la hora de remplazar a los valores perdidos o a los extremos.

```
age_s<-summary(train$Age)
pclass_s<-summary(train$Pclass)
sibSp_s<-summary(train$SibSp)
parch_s<-summary(train$Parch)

table_s <- suppressWarnings(rbind(age_s,pclass_s,sibSp_s,parch_s))
age_r <- c(sd(train$Age), winsor.mean(train$Age,trim=0.05), IQR(train$Age))
pclass_r <- c(sd(train$Pclass), "NA", IQR(train$Pclass))
sibSp_r <- c(sd(train$SibSp), winsor.mean(train$SibSp,trim=0.05), IQR(train$SibSp))
parch_r <- c(sd(train$Parch), winsor.mean(train$Parch,trim=0.05), IQR(train$Parch))

table_r <- rbind(age_r,pclass_r,sibSp_r, parch_r)
table_res <- cbind(table_s, table_r)
colnames( table_res) <- c("Min", "1st Qu", "Median", "Mean", "3rd Qu", "Max", "SD", "WINSOR", "IQR")
kable(table_res)
```

	Min	1st Qu	Median	Mean	3rd Qu	Max	SD	WINSOR	IQR
age_s	0.42	21	28	29.3537261503928	36.25	80	13.8619246774438	29.1416947250281	15.2
pclass_s	216	184	491	216	184	491	0.836071240977049	NA	1
sibSp_s	0	0	0	0.52300785634119	1	8	1.10274343229343	0.452300785634119	1
parch_s	0	0	0	0.381593714927048	0	6	0.806057221129948	0.345679012345679	0

- Detactamos a los valores extremos

```
#Los valores atápicos SibSp.
boxplot.stats(train$SibSp)$out
```

```
##  [1] 3 4 3 3 4 5 3 4 5 3 3 4 8 4 4 3 8 4 8 3 4 4 4 4 8 3 3 5 3 5 3 4 4 3 3
## [36] 5 4 3 4 8 4 3 4 8 4 8
```

```
#Los valores atápicos Parch.
boxplot.stats(train$Parch)$out
```

```
##   [1] 1 2 1 5 1 1 5 2 2 1 1 2 2 2 1 2 2 2 3 2 2 1 1 1 1 2 1 1 2 2 1 2 2 2 1
##  [36] 2 1 1 2 1 4 1 1 1 1 2 2 1 2 1 1 1 2 1 1 2 2 2 1 1 2 2 1 2 1 1 1 1 1 1
##  [71] 1 2 1 2 2 1 1 2 1 1 2 1 1 1 1 2 1 1 1 4 1 1 2 2 2 2 2 1 1 1 2 2 1 1 2
## [106] 2 3 4 1 2 1 1 2 1 2 1 2 1 1 2 2 1 1 1 1 2 2 2 2 2 2 1 1 2 1 4 1 1 2 1
## [141] 2 1 1 2 5 2 1 1 1 2 1 5 2 1 1 1 2 1 6 1 2 1 2 1 1 1 1 1 1 1 3 2 1 1 1
## [176] 1 2 1 2 3 1 2 1 2 2 1 1 2 1 2 1 2 1 1 1 2 1 1 2 1 2 1 1 1 1 3 2 1 1 1
## [211] 1 5 2
```

- Remplazamos a los valores extremos

Se van a sustituir a los valores atápicos por los valores de l primer o tercer cuadro dependiendo si es un valor atipico minimo o máximo. Dicha sustitución nos aportará unos modelos más fiables.

```
#SibSp
qnt <- quantile(train$SibSp, probs=c(.25, .75), na.rm = T)
caps <- quantile(train$SibSp, probs=c(.10, .90), na.rm = T)
H_SibSp <- 1.5 * IQR(train$SibSp, na.rm = T)
train$SibSp[train$SibSp < (qnt[1] - H_SibSp)] <- caps[1]
train$SibSp[train$SibSp > (qnt[1] + H_SibSp)] <- caps[2]

#Los valores atápicos Parch.
boxplot.stats(train$SibSp)$out
```

```
## numeric(0)
```

```
#Parch
qnt <- quantile(train$Parch, probs=c(.25, .75), na.rm = T)
caps <- quantile(train$Parch, probs=c(.10, .90), na.rm = T)
H_Parch <- 1.5 * IQR(train$Parch, na.rm = T)
train$Parch[train$Parch < (qnt[1] - H_Parch)] <- caps[1]
train$Parch[train$Parch > (qnt[1] + H_Parch)] <- caps[2]

#Los valores atápicos Parch.
boxplot.stats(train$Parch)$out
```

```
##   [1] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
##  [36] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
##  [71] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
## [106] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
## [141] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
## [176] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
## [211] 2 2 2
```

En el caso de Número de hermanos|cónyuges a bordo hemos podido eleminar a los valores atipicos, en cambio para el Número de padres|hijos a bordo sigue habiendo valores extremos pero que contienen un valor menos agresivo.

Análisis de los datos

Selección de los grupos de datos que se quieren analizar/comparar (planificación de los análisis a aplicar)

A continuación, se seleccionan los grupos dentro de nuestro conjunto de datos que pueden resultar interesantes para analizar y/o comparar. No obstante, como se verá en el apartado consistente en la realización de pruebas estadásticas, no todos se utilizarán.

```
# Agrupación por genero
train.female <- train[train$Sex == "female",]
train.male <- train[train$Sex == "male",]

# Agrupación por puerta de embarque
train.c <- train[train$Embarked == "C",]
train.s <- train[train$Embarked == "S",]
```

```
train.q <- train[train$Embarked == "Q",]

# Agrupación por Parch
train.parch.cero <- train[train$Parch == "0",]
train.parch.mayor <- train[train$Parch > "0",]

# Agrupación por Parch
train.sibSp.cero <- train[train$SibSp == "0",]
train.sibSp.mayor <- train[train$SibSp > "0",]

#Usar el termino de tamaÃ±o de la familia sumando parch y SibSp
train$FamilySize <- train$SibSp + train$Parch +1;
```

Comprobación de la normalidad y homogeneidad de la varianza

Para la comprobación de que los valores que toman nuestras variables cuantitativas provienen de una población distribuida normalmente, utilizaremos la prueba de normalidad de Anderson- Darling.

Así, se comprueba que para que cada prueba se obtiene un p-valor superior al nivel de significación prefijado $\alpha = 0,05$. Si esto se cumple, entonces se considera que variable en cuestión sigue una distribución normal.

```
alpha = 0.05
col.names = colnames(train)
for (i in 1:ncol(train)) {
  if (i == 1) cat("Variables que no siguen una distribución normal:\n")
  if (is.integer(train[,i]) | is.numeric(train[,i])) {
    p_val = ad.test(train[,i])$p.value
    if (p_val < alpha) {
      cat(col.names[i])
      # Format output
      if (i < ncol(train) - 1) cat(", ")
      if (i %% 3 == 0) cat("\n")
    }
  }
}
```

```
## Variables que no siguen una distribución normal:
## PassengerId, Age, SibSp,
## Parch, FamilySize
```

Aplicación de pruebas estadásticas para comparar los grupos de datos. En función de los datos y el objetivo del estudio, aplicar pruebas de contraste de hipótesis, correlaciones, regresiones, etc. Aplicar al menos tres métodos de análisis diferentes

Vamos a estudiar la homogeneidad de varianzas mediante la aplicación de un test de Fligner-Killeen. En este caso, estudiaremos esta homogeneidad en cuanto al genero del pasajero. En el siguiente test, la hipótesis nula consiste en que ambas varianzas son iguales.

```
#fligner.test(Survived ~ Sex, data = train)
fligner.test(train)
```

```
##
##  Fligner-Killeen test of homogeneity of variances
##
## data:  train
## Fligner-Killeen:med chi-squared = 10799, df = 16, p-value <
## 2.2e-16
```

Puesto que obtenemos un p-valor inferior a 0,05, no aceptamos la hipótesis de que las varianzas de ambas muestras son homogéneas. Detectamos que el genero de pasajero es un factor que ha influido en la supervivencia de los pasajeros.

```
#fligner.test(Survived ~ Embarked, data = train)
```

Procedemos a realizar un análisis de correlación entre las distintas variables para determinar cuáles de ellas ejercen una mayor influencia sobre el precio final del vehículo. Para ello, se utilizará el coeficiente de correlación de Spearman, puesto que hemos visto que tenemos datos que no siguen una distribución normal.

```
corr_matrix <- matrix(nc = 2, nr = 0)
colnames(corr_matrix) <- c("estimate", "p-value")
# Calcular el coeficiente de correlación para cada variable cuantitativa
# con respecto al campo "precio"
#for (i in 3:(ncol(train) - 1)) {
 # if (is.integer(train[,i]) | is.numeric(train[,i])) {
  #  spearman_test = cor.test(train[,i], train[,2], method = "spearman")
   # corr_coef = spearman_test$estimate
    #p_val = spearman_test$p.value
    # Add row to matrix
    #pair = matrix(ncol = 2, nrow = 1)
    #pair[1][1] = corr_coef
    #pair[2][1] = p_val
    #corr_matrix <- rbind(corr_matrix, pair)
    #rownames(corr_matrix)[nrow(corr_matrix)] <- colnames(train)[i]
  #}
#}
print(corr_matrix)
```

```
##      estimate p-value
```

Representación de los resultados a partir de tablas y gráficas

Resolución del problema. A partir de los resultados obtenidos, Â¿cuáles son las conclusiones? Â¿Los resultados permiten responder al problema?

Ejemplo de estudio visual con el juego de datos Titanic

Procesos de limpieza del juego de datos

Primer contacto con el juego de datos, visualizamos su estructura.

```
# Guardamos el juego de datos test y train en un Ãºnico dataset
test <- read.csv('./data/titanic-test.csv',stringsAsFactors = FALSE)
train <- read.csv('./data/titanic-train.csv', stringsAsFactors = FALSE)

# Unimos los dos juetos de datos en uno solo
totalData <- bind_rows(train,test)
filas=dim(train)[1]

# Verificamos la estructura del juego de datos
str(totalData)
```

```
## 'data.frame':    1309 obs. of  13 variables:
##  $ PassengerId                                                  : int  1 2 3 4 5 6 7 8 9 1
##  $ Survived                                                     : int  0 1 1 1 0 0 0 0 1 1
##  $ Pclass                                                       : int  3 1 3 1 3 3 1 3 3 2
##  $ Name                                                         : chr  "Braund, Mr. Owen Ha
##  $ Sex                                                          : chr  "male" "female" "fe
##  $ Age                                                          : num  22 38 26 35 35 NA 5
##  $ SibSp                                                        : int  1 1 0 1 0 0 0 3 0 1
##  $ Parch                                                        : int  0 0 0 0 0 0 0 1 2 0
##  $ Ticket                                                       : chr  "A/5 21171" "PC 175
##  $ Fare                                                         : num  7.25 71.28 7.92 53.
##  $ Cabin                                                        : chr  "" "C85" "" "C123"
##  $ Embarked                                                     : chr  "S" "C" "S" "S" ...
##  $ PassengerId.Pclass.Name.Sex.Age.SibSp.Parch.Ticket.Fare.Cabin.Embarked.: chr  NA NA NA NA ...
```

Trabajamos los atributos con valores vacáos.

```
# Estadásticas de valores vacáos
colSums(is.na(totalData))
```

```
##                                                  PassengerId
##                                                          418
##                                                     Survived
##                                                          418
##                                                       Pclass
##                                                          418
##                                                         Name
##                                                          418
##                                                          Sex
##                                                          418
##                                                          Age
##                                                          595
##                                                        SibSp
##                                                          418
##                                                        Parch
##                                                          418
##                                                       Ticket
##                                                          418
```

```
##                                                                    Fare
##                                                                     418
##                                                                   Cabin
##                                                                     418
##                                                                Embarked
##                                                                     418
## PassengerId.Pclass.Name.Sex.Age.SibSp.Parch.Ticket.Fare.Cabin.Embarked.
##                                                                     891
```

```
colSums(totalData=="")
```

```
##                                                                PassengerId
##                                                                         NA
##                                                                   Survived
##                                                                         NA
##                                                                     Pclass
##                                                                         NA
##                                                                       Name
##                                                                         NA
##                                                                        Sex
##                                                                         NA
##                                                                        Age
##                                                                         NA
##                                                                      SibSp
##                                                                         NA
##                                                                      Parch
##                                                                         NA
##                                                                     Ticket
##                                                                         NA
##                                                                       Fare
##                                                                         NA
##                                                                      Cabin
##                                                                         NA
##                                                                   Embarked
##                                                                         NA
## PassengerId.Pclass.Name.Sex.Age.SibSp.Parch.Ticket.Fare.Cabin.Embarked.
##                                                                         NA
```

```
# Tomamos valor "C" para los valores vacáos de la variable "Embarked"
totalData$Embarked[totalData$Embarked==""]="C"

#Para los valores perdidos procedemos con aplicar la distancia de Gower
totalData <- kNN(totalData)
```

Discretizamos cuando tiene sentido y en función de cada variable.

```
# ¿Para qué variables tendráa sentido un proceso de discretización?
apply(totalData,2, function(x) length(unique(x)))
```

```
##                                                                PassengerId
##                                                                        891
##                                                                   Survived
```

```
##                                                                      2
##                                                                 Pclass
##                                                                      3
##                                                                   Name
##                                                                    891
##                                                                    Sex
##                                                                      2
##                                                                    Age
##                                                                     88
##                                                                  SibSp
##                                                                      7
##                                                                 Parch
##                                                                      7
##                                                                Ticket
##                                                                    681
##                                                                 Fare
##                                                                    248
##                                                                Cabin
##                                                                    148
##                                                             Embarked
##                                                                      3
##     PassengerId.Pclass.Name.Sex.Age.SibSp.Parch.Ticket.Fare.Cabin.Embarked.
##                                                                    397
##                                                          PassengerId_imp
##                                                                      2
##                                                             Survived_imp
##                                                                      2
##                                                               Pclass_imp
##                                                                      2
##                                                                Name_imp
##                                                                      2
##                                                                 Sex_imp
##                                                                      2
##                                                                 Age_imp
##                                                                      2
##                                                               SibSp_imp
##                                                                      2
##                                                                Parch_imp
##                                                                      2
##                                                               Ticket_imp
##                                                                      2
##                                                                Fare_imp
##                                                                      2
##                                                               Cabin_imp
##                                                                      2
##                                                             Embarked_imp
##                                                                      2
## PassengerId.Pclass.Name.Sex.Age.SibSp.Parch.Ticket.Fare.Cabin.Embarked._imp
##                                                                      2
```

```r
# Discretizamos las variables con pocas clases
cols<-c("Survived","Pclass","Sex","Embarked")
for (i in cols){
  totalData[,i] <- as.factor(totalData[,i])
```

```
}

# Después de los cambios, analizamos la nueva estructura del juego de datos
str(totalData)
```

```
## 'data.frame':    1309 obs. of  26 variables:
##  $ PassengerId                                                    : int  1 2 3 4 5 6 7 8
##  $ Survived                                                       : Factor w/ 2 levels "
##  $ Pclass                                                         : Factor w/ 3 levels "
##  $ Name                                                           : chr  "Braund, Mr. Ow
##  $ Sex                                                            : Factor w/ 2 levels "
##  $ Age                                                            : num  22 38 26 35 35 
##  $ SibSp                                                          : int  1 1 0 1 0 0 0 3
##  $ Parch                                                          : int  0 0 0 0 0 0 0 1
##  $ Ticket                                                         : chr  "A/5 21171" "PC
##  $ Fare                                                           : num  7.25 71.28 7.92
##  $ Cabin                                                          : chr  "" "C85" "" "C1
##  $ Embarked                                                       : Factor w/ 3 levels "
##  $ PassengerId.Pclass.Name.Sex.Age.SibSp.Parch.Ticket.Fare.Cabin.Embarked.    : chr  " Mrs. James (E
##  $ PassengerId_imp                                                : logi  FALSE FALSE FA
##  $ Survived_imp                                                   : logi  FALSE FALSE FA
##  $ Pclass_imp                                                     : logi  FALSE FALSE FA
##  $ Name_imp                                                       : logi  FALSE FALSE FA
##  $ Sex_imp                                                        : logi  FALSE FALSE FA
##  $ Age_imp                                                        : logi  FALSE FALSE FA
##  $ SibSp_imp                                                      : logi  FALSE FALSE FA
##  $ Parch_imp                                                      : logi  FALSE FALSE FA
##  $ Ticket_imp                                                     : logi  FALSE FALSE FA
##  $ Fare_imp                                                       : logi  FALSE FALSE FA
##  $ Cabin_imp                                                      : logi  FALSE FALSE FA
##  $ Embarked_imp                                                   : logi  FALSE FALSE FA
##  $ PassengerId.Pclass.Name.Sex.Age.SibSp.Parch.Ticket.Fare.Cabin.Embarked._imp: logi  TRUE TRUE TRUE
```

Cuadro de las estimaciónes no robustas y robustas.

```
age_s<-summary(totalData$Age)
pclass_s<-summary(totalData$Pclass)
sibSp_s<-summary(totalData$SibSp)
parch_s<-summary(totalData$Parch)
fare_s<-summary(totalData$Fare)

table_s <- suppressWarnings(rbind(age_s,pclass_s,sibSp_s,parch_s, fare_s))
age_r <- c(sd(totalData$Age), winsor.mean(totalData$Age,trim=0.05), IQR(totalData$Age))
pclass_r <- c(sd(totalData$Pclass), "NA", IQR(totalData$Pclass))
#pclass_r <- c(sd(totalData$Pclass), winsor.mean(totalData$Pclass,trim=0.05), IQR(totalData$Pclass))
sibSp_r <- c(sd(totalData$SibSp), winsor.mean(totalData$SibSp,trim=0.05), IQR(totalData$SibSp))
parch_r <- c(sd(totalData$Parch), winsor.mean(totalData$Parch,trim=0.05), IQR(totalData$Parch))
fare_r <- c(sd(totalData$Fare), winsor.mean(totalData$Fare,trim=0.05), IQR(totalData$Fare))

table_r <- rbind(age_r,pclass_r,sibSp_r, parch_r, fare_r)
table_res <- cbind(table_s, table_r)
colnames( table_res) <- c("Min", "1st Qu", "Median", "Mean", "3rd Qu", "Max", "SD", "WINSOR", "IQR")
kable(table_res)
```

	Min	1st Qu	Median	Mean	3rd Qu	Max	SD	WINSOR
age_s	0.42	11	21	23.5459663865546	32	80	14.3387964848852	23.3907563025
pclass_s	216	184	909	216	184	909	0.761315077742776	NA
sibSp_s	0	0	1	1.95263559969442	5	8	2.2775770777165	1.93659281894
parch_s	0	0	0	0.898395721925134	2	6	1.00590254596032	0.87394957983
fare_s	0	9.5	30	36.896981894576	46.9	512.3292	41.5602225355617	32.8266262032
Detactamos	a los v	alores at	ápicos					

```
#Los valores atápicos SibSp.
boxplot.stats(totalData$SibSp)$out
```

```
## integer(0)
```

```
#Los valores atápicos Parch.
boxplot.stats(totalData$Parch)$out
```

```
## [1] 6
```

```
#Los valores atápicos Fare.
boxplot.stats(totalData$Fare)$out
```

```
##  [1] 263.0000 146.5208 263.0000 247.5208 146.5208 113.2750 512.3292
##  [8] 153.4625 135.6333 151.5500 247.5208 151.5500 110.8833 108.9000
## [15] 262.3750 164.8667 134.5000 135.6333 153.4625 133.6500 134.5000
## [22] 263.0000 135.6333 211.5000 227.5250 120.0000 113.2750 120.0000
## [29] 263.0000 151.5500 108.9000 221.7792 106.4250 106.4250 110.8833
## [36] 227.5250 110.8833 153.4625 113.2750 133.6500 512.3292 211.3375
## [43] 110.8833 227.5250 151.5500 227.5250 211.3375 512.3292 262.3750
## [50] 120.0000 211.3375 120.0000 164.8667
```

Remplazamos a los valores atápicos

```
#Parch
qnt <- quantile(totalData$Parch, probs=c(.25, .75), na.rm = T)
caps <- quantile(totalData$Parch, probs=c(.05, .95), na.rm = T)
H <- 1.5 * IQR(totalData$Parch, na.rm = T)
totalData$Parch[totalData$Parch < (qnt[1] - H)] <- caps[1]
totalData$Parch[totalData$Parch > (qnt[1] + H)] <- caps[2]

#Fare
qnt <- quantile(totalData$Fare, probs=c(.25, .75), na.rm = T)
caps <- quantile(totalData$Fare, probs=c(.05, .95), na.rm = T)
H <- 1.5 * IQR(totalData$Fare, na.rm = T)
totalData$Fare[totalData$Fare < (qnt[1] - H)] <- caps[1]
totalData$Fare[totalData$Fare > (qnt[1] + H)] <- caps[2]

#Los valores atápicos Parch.
boxplot.stats(totalData$Parch)$out
```

```
## numeric(0)
```

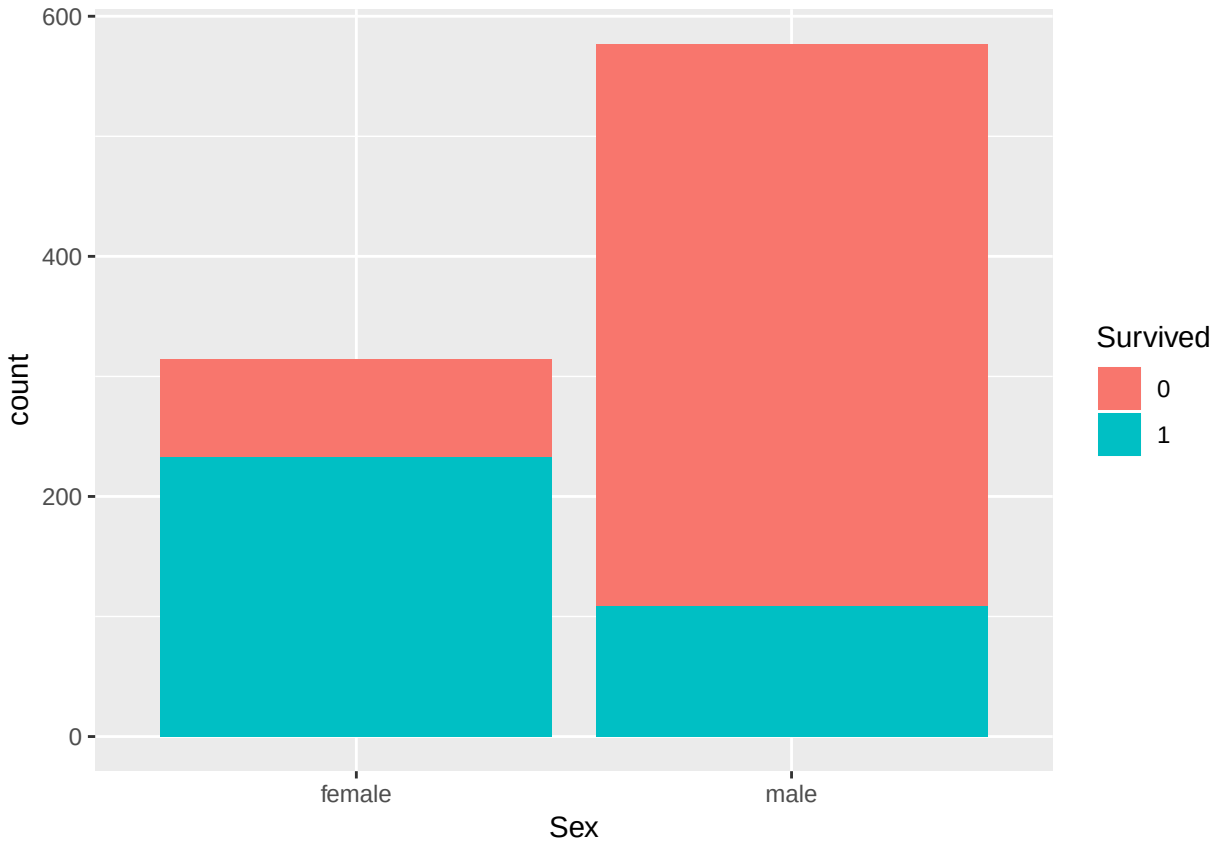
```r
#Los valores atápicos Fare.
boxplot.stats(totalData$Fare)$out
```

```
## numeric(0)
```

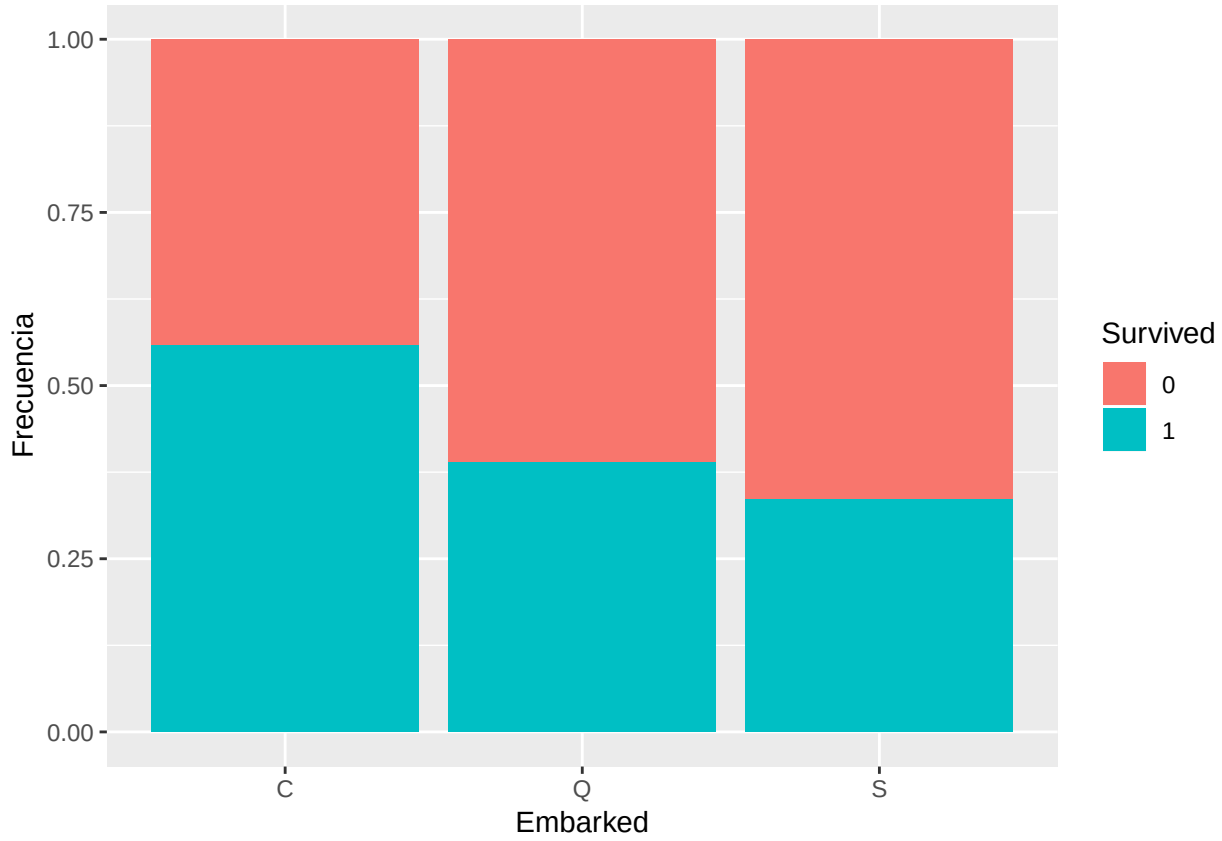
Procesos de análisis del juego de datos

Nos proponemos analizar las relaciones entre las diferentes variables del juego de datos.

```r
# Visualizamos la relación entre las variables "sex" y "survival":
ggplot(data=totalData[1:filas,],aes(x=Sex,fill=Survived))+geom_bar()
```




```r
# Otro punto de vista. Survival como función de Embarked:
ggplot(data = totalData[1:filas,],aes(x=Embarked,fill=Survived))+geom_bar(position="fill")+ylab("Frecuer
```



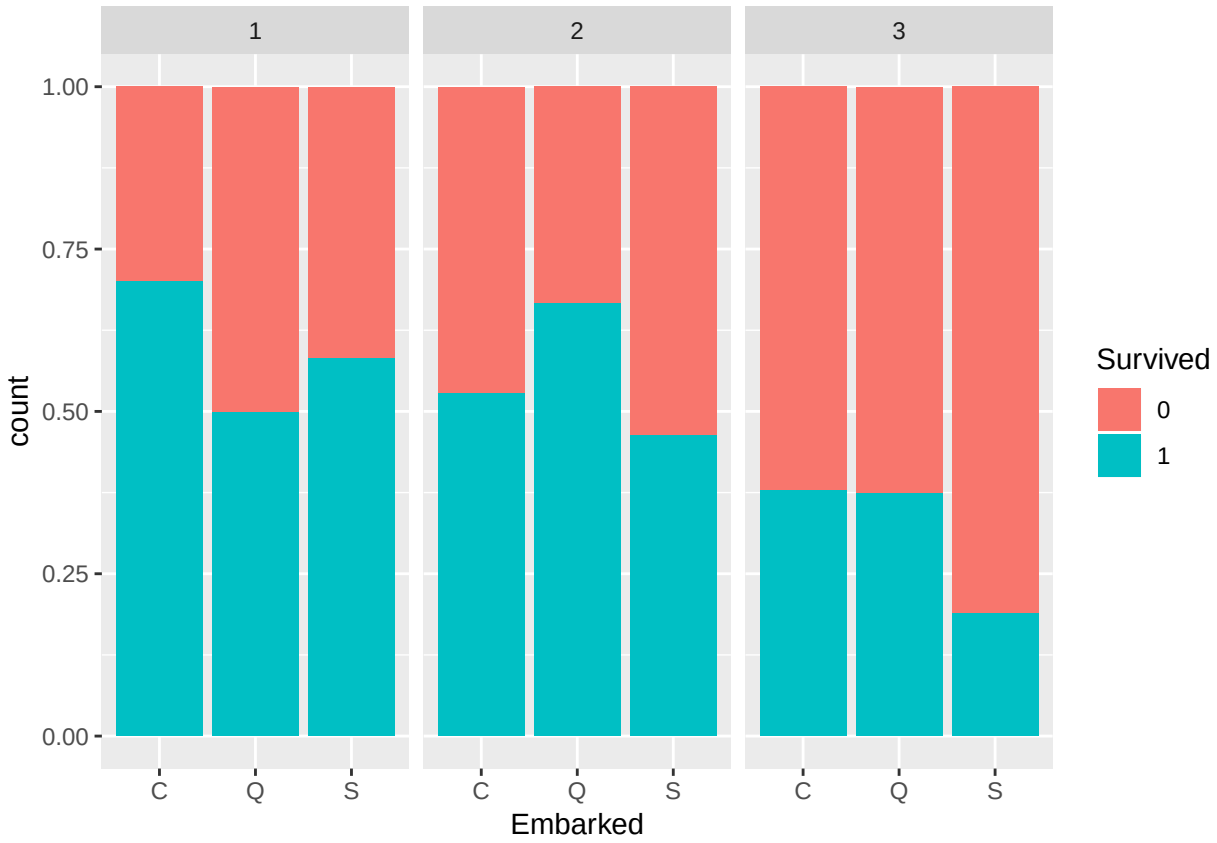
Obtenemos una matriz de porcentages de frecuencia.
Vemos, por ejemplo que la probabilidad de sobrebivir si se embarcó en "C" es de un 55,88%

```
t<-table(totalData[1:filas,]$Embarked,totalData[1:filas,]$Survived)
for (i in 1:dim(t)[1]){
    t[i,]<-t[i,]/sum(t[i,])*100
}
t
```

```
##
##            0        1
##   C 44.11765 55.88235
##   Q 61.03896 38.96104
##   S 66.30435 33.69565
```

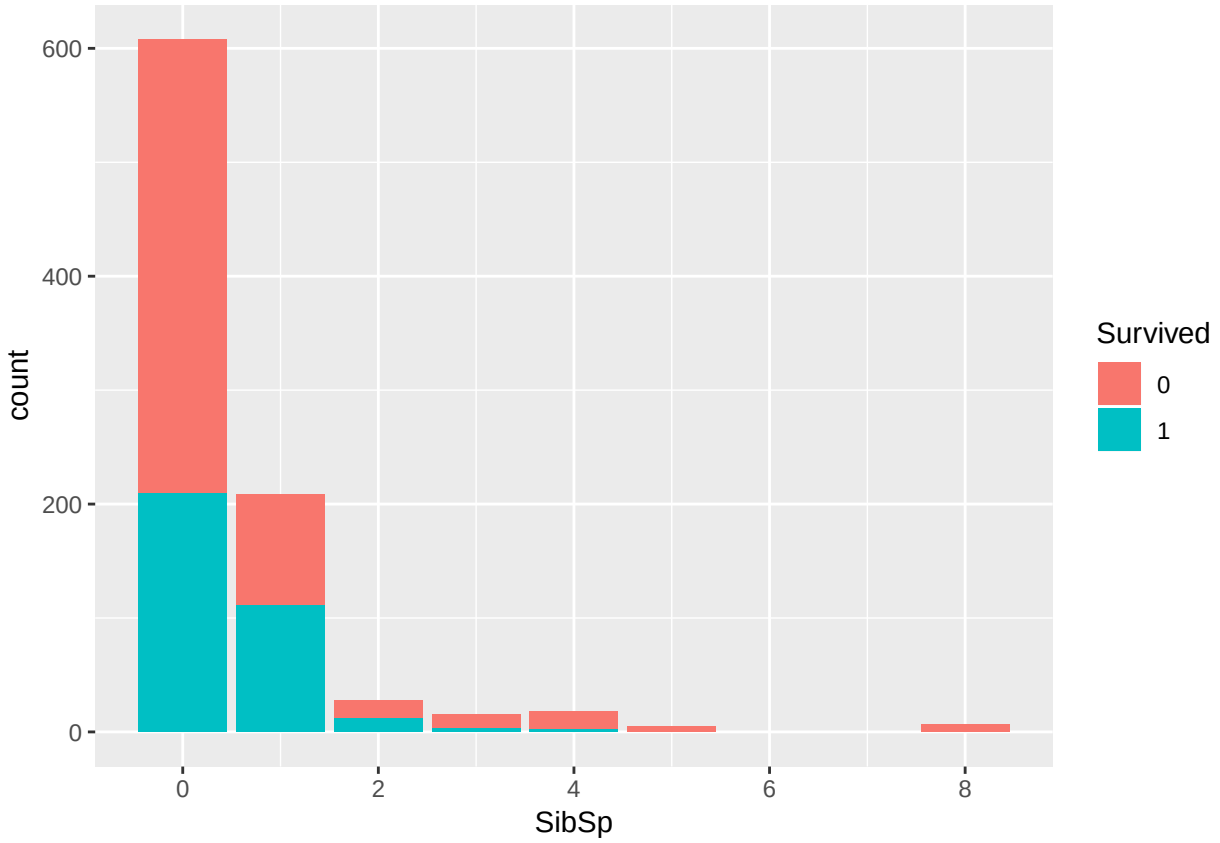
Veamos ahora como en un mismo gráfico de frecuencias podemos trabajar con 3 variables: Embarked, Survived y Pclass.

```
# Now, let's devide the graph of Embarked by Pclass:
ggplot(data = totalData[1:filas,],aes(x=Embarked,fill=Survived))+geom_bar(position="fill")+facet_wrap(~
```

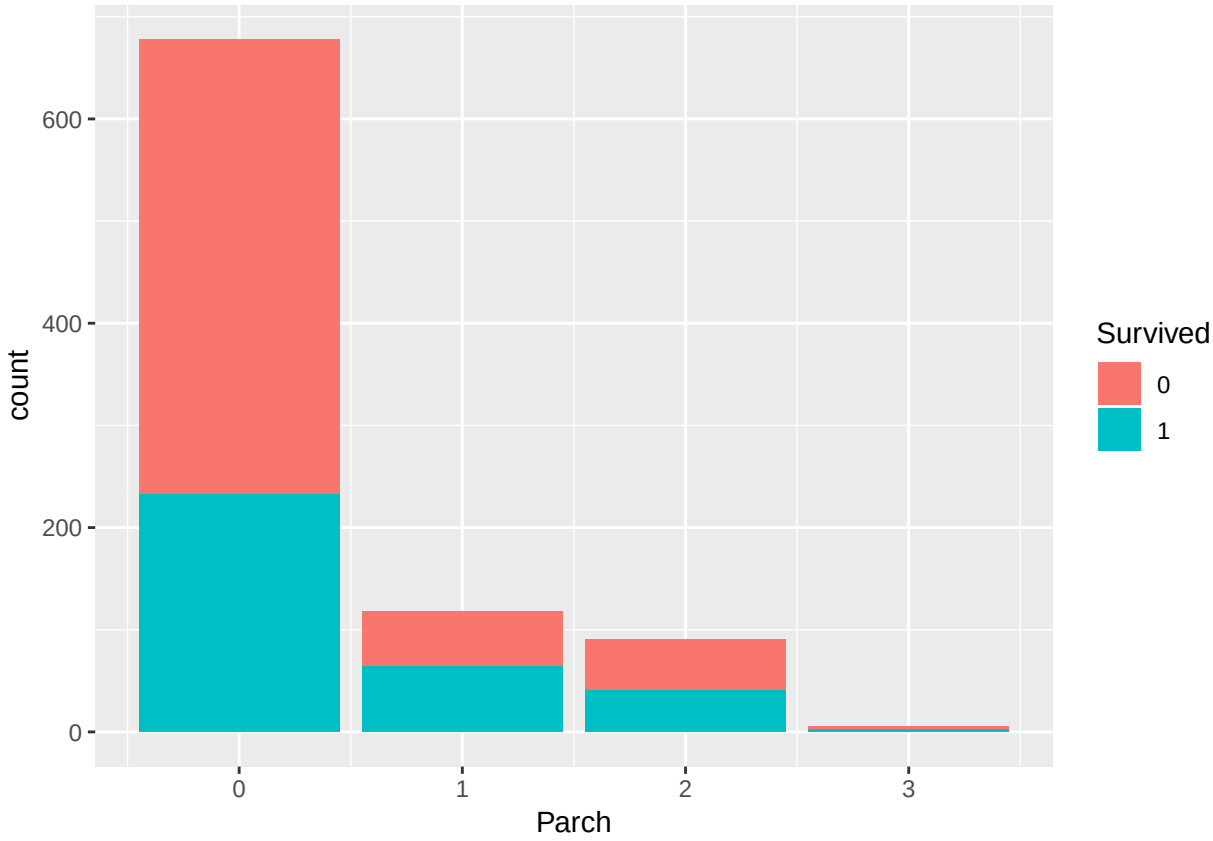


Comparemos ahora dos gráficos de frecuencias: Survived-SibSp y Survived-Parch

```
# Survivial como función de SibSp y Parch
ggplot(data = totalData[1:filas,],aes(x=SibSp,fill=Survived))+geom_bar()
```



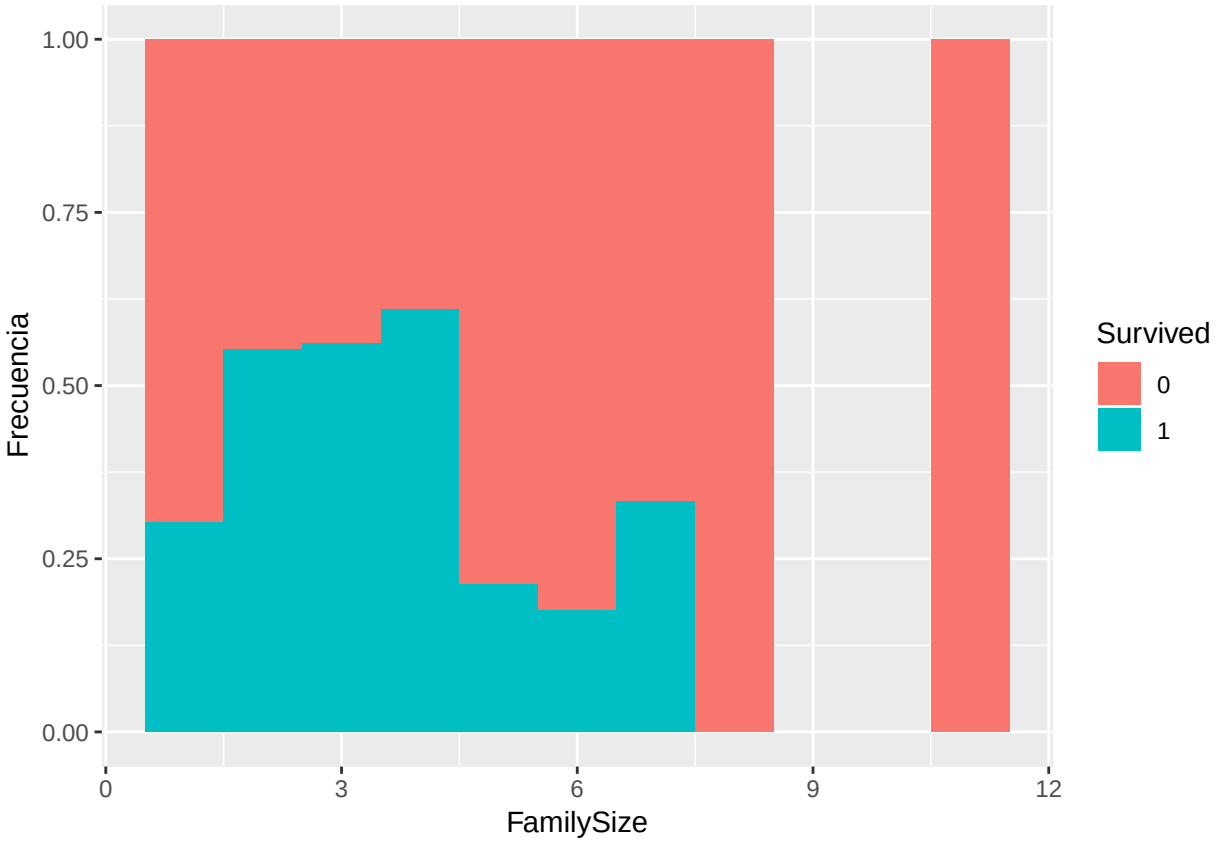

```
ggplot(data = totalData[1:filas,],aes(x=Parch,fill=Survived))+geom_bar()
```




```
# Vemos como las forma de estos dos gráficos es similar. Este hecho nos puede indicar presencia de corr
```

Veamos un ejemplo de construcción de una variable nueva: TamaÃ±o de familia

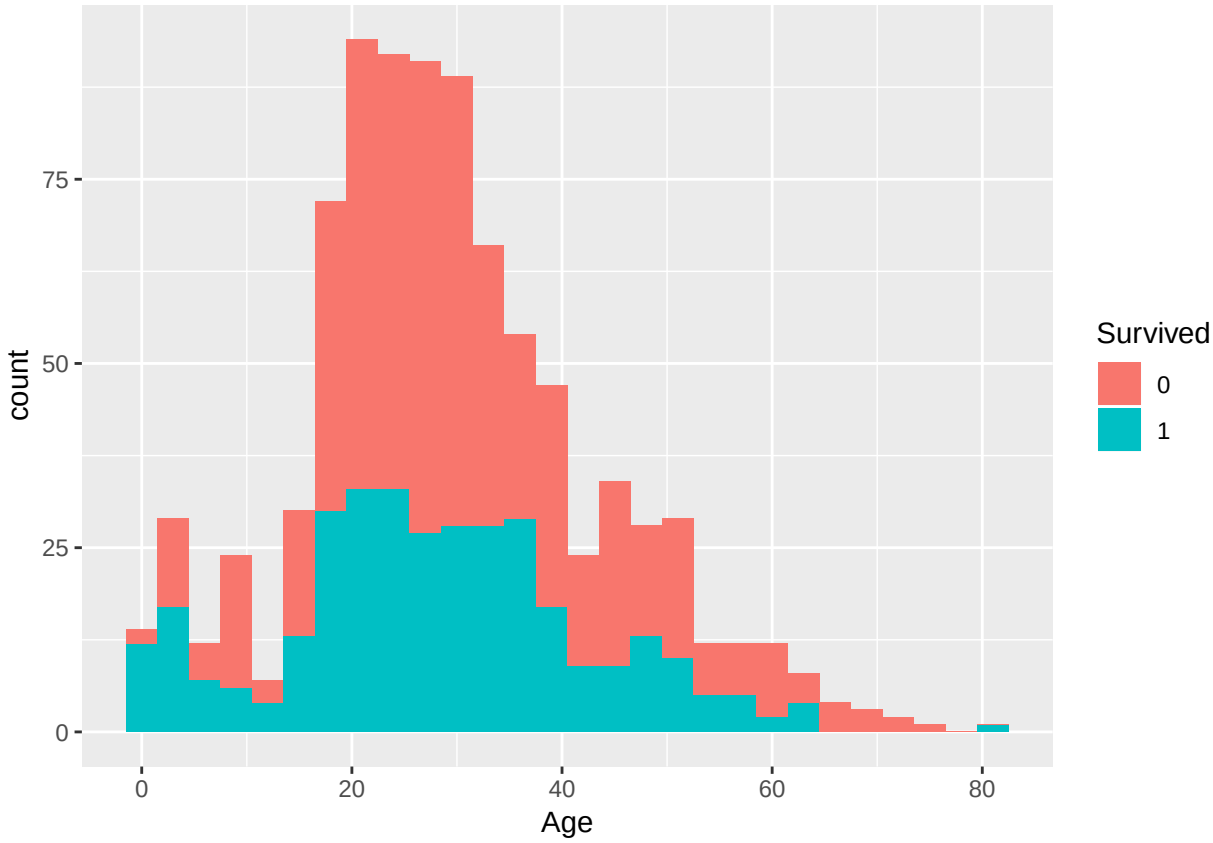
```
# Construimos un atributo nuevo: family size.
totalData$FamilySize <- totalData$SibSp + totalData$Parch +1;
totalData1<-totalData[1:filas,]
ggplot(data = totalData1[!is.na(totalData[1:filas,]$FamilySize),],aes(x=FamilySize,fill=Survived))+geom
```




```
# Observamos como familias de entre 2 y 6 miembros tienen más del 50% de posibilidades de supervivencia
```

Veamos ahora dos gráficos que nos compara los atributos Age y Survived.
Observamos como el parámetro position="fill" nos da la proporción acumulada de un atributo dentro de otro

```
# Survival como función de age:
ggplot(data = totalData1[!(is.na(totalData[1:filas,]$Age)),],aes(x=Age,fill=Survived))+geom_histogram(b
```




```
ggplot(data = totalData1[!is.na(totalData[1:filas,]$Age),],aes(x=Age,fill=Survived))+geom_histogram(bin
```


1.00
0.75
0.50
0.25
0.00
Frecuencia
0
20
40
60
80
Age
Survived
0
1