

Tipología 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	1
Ejemplo de estudio visual con el juego de datos Titanic	2
Procesos de limpieza del juego de datos	2
Procesos de análisis del juego de datos	7

Introducción

A título de muestra, 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”.

De momento dejaremos para las siguientes prácticas el estudio de algoritmos predictivos y nos centraremos por ahora en el estudio de las variables de un juego de datos, es decir, haremos un trabajo descriptivo del mismo.

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

Las técnicas que trabajaremos son las siguientes:

1. Normalización
 2. Discretización
 3. Gestión de valores nulos
 4. Estudio de correlaciones
 5. Reducción de la dimensionalidad
 6. Análisis visual del juego de datos
-

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 único dataset
test <- read.csv('./data/titanic-test.csv', stringsAsFactors = FALSE)
train <- read.csv('./data/titanic-train.csv', stringsAsFactors = FALSE)

# Unimos los dos juegos 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 10
## $ 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 vacíos.

```
# Estadísticas de valores vacíos
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 vacíos 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 tendría 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
```

```
# 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 estimaciones 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	ípicos					

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

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

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

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

```
#Los valores atípicos 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 atípicos

```
#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 atípicos Parch.
boxplot.stats(totalData$Parch)$out
```

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

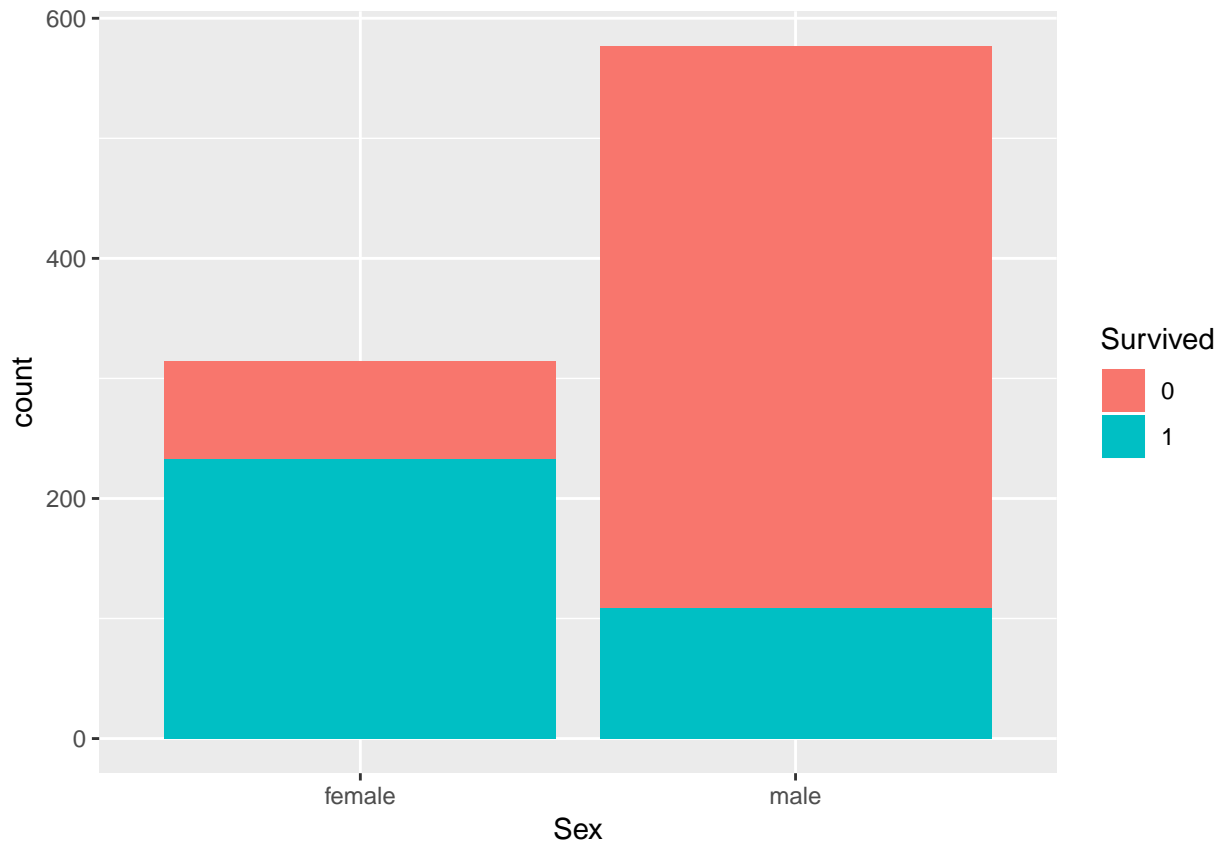
```
#Los valores atípicos Fare.
boxplot.stats(totalData$Fare)$out
```

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

Procesos de análisis del juego de datos

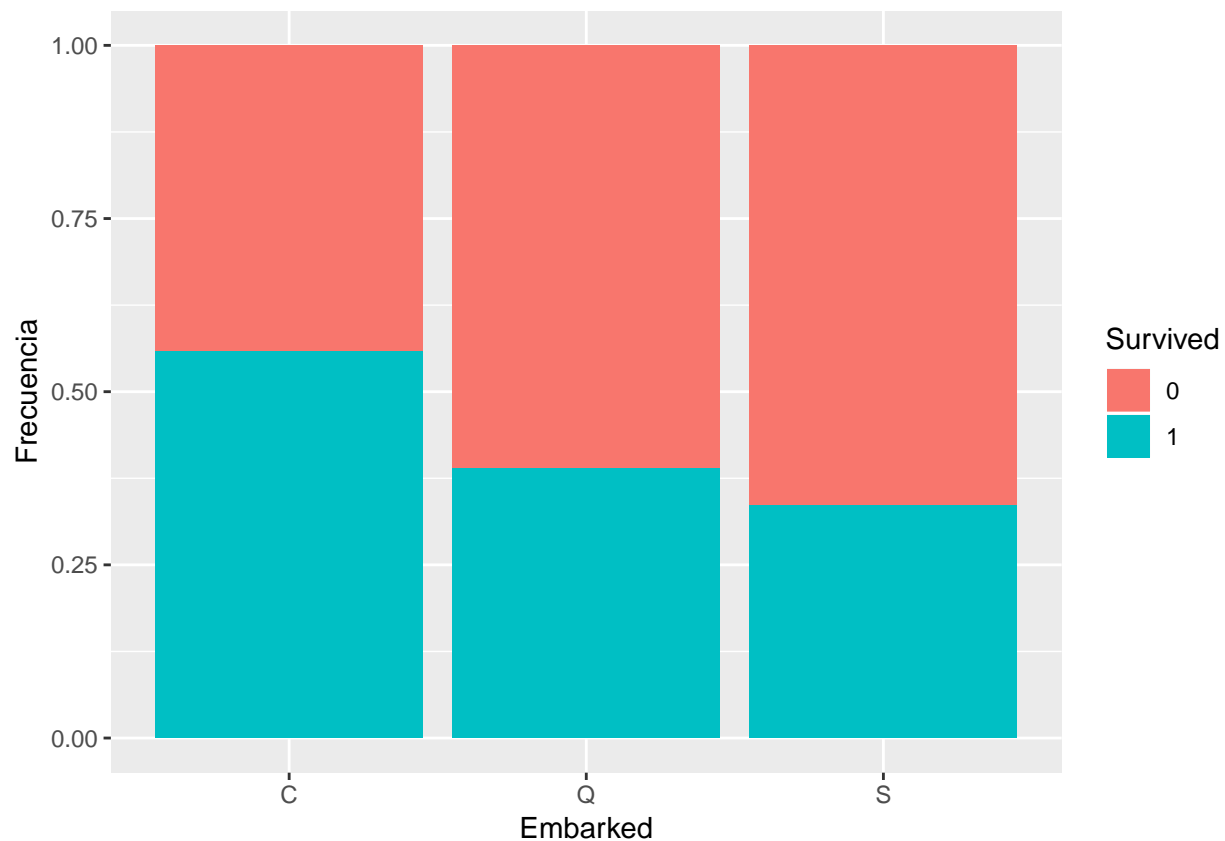
Nos proponemos analizar las relaciones entre las diferentes variables del juego de datos.

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



```
# Otro punto de vista. Survival como función de Embarked:
```

```
ggplot(data = totalData[1:filas,],aes(x=Embarked,fill=Survived))+geom_bar(position="fill")+ylab("Frecuencia")
```



Obtenemos una matriz de porcentajes de frecuencia.

Vemos, por ejemplo que la probabilidad de sobrevivir si se embarcó en “C” es de un 55,88%

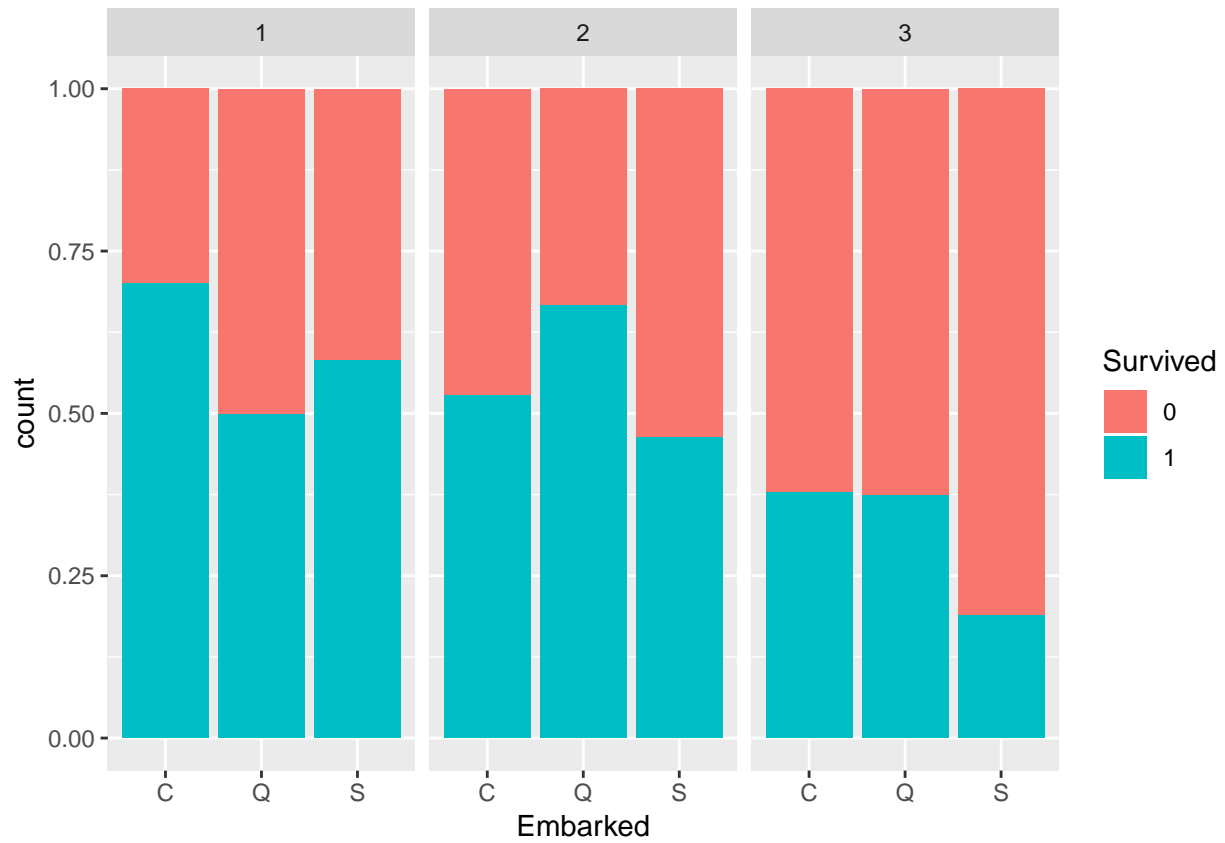
```
t<-table(totalData[1:filas,]$Embarked,totalData[1:filas,]$Survived)
for (i in 1:dim(t)[1]){
  t[i,]<-t[i,]/sum(t[i,])*100
}
t
```

```
##
##           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.

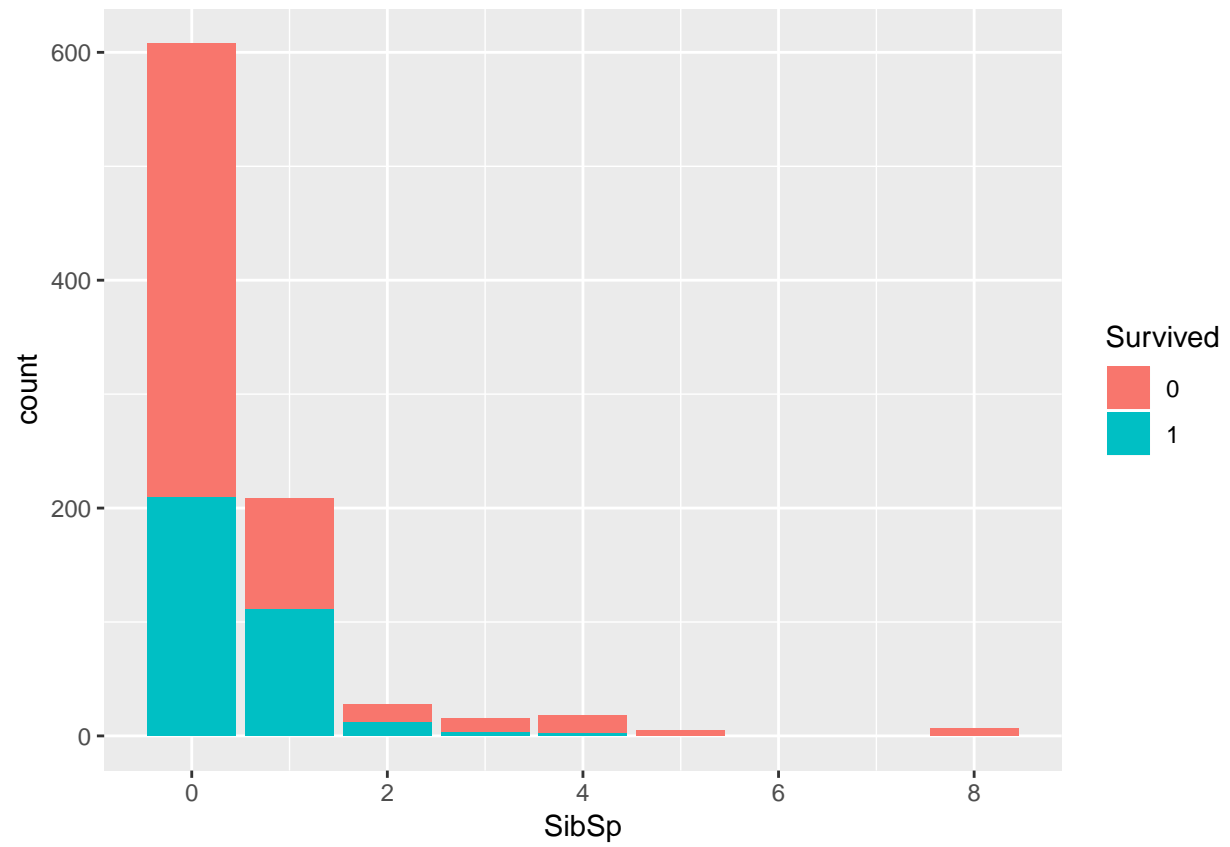
```
# Now, let's divide the graph of Embarked by Pclass:
```

```
ggplot(data = totalData[1:filas,],aes(x=Embarked,fill=Survived))+geom_bar(position="fill")+facet_wrap(~Pclass)
```

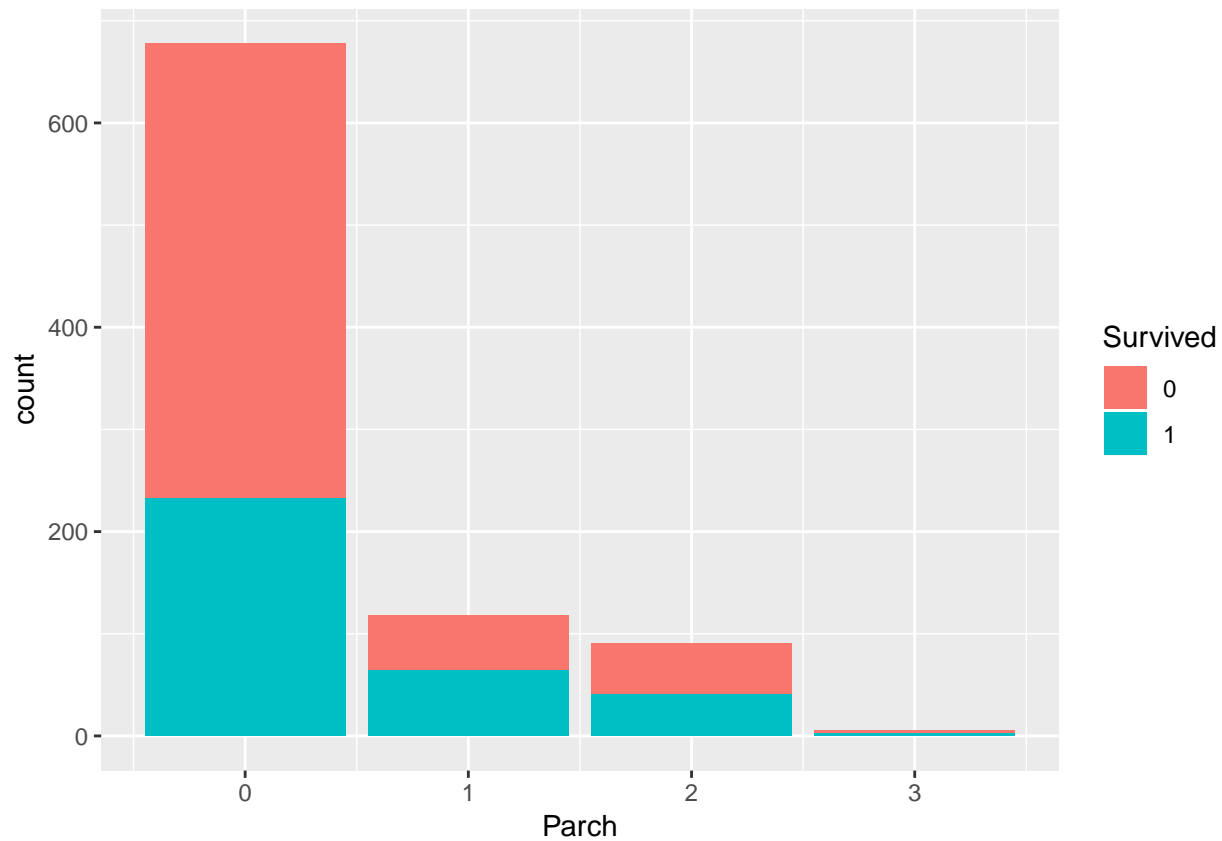



Comparemos ahora dos gráficos de frecuencias: Survived-SibSp y Survived-Parch

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



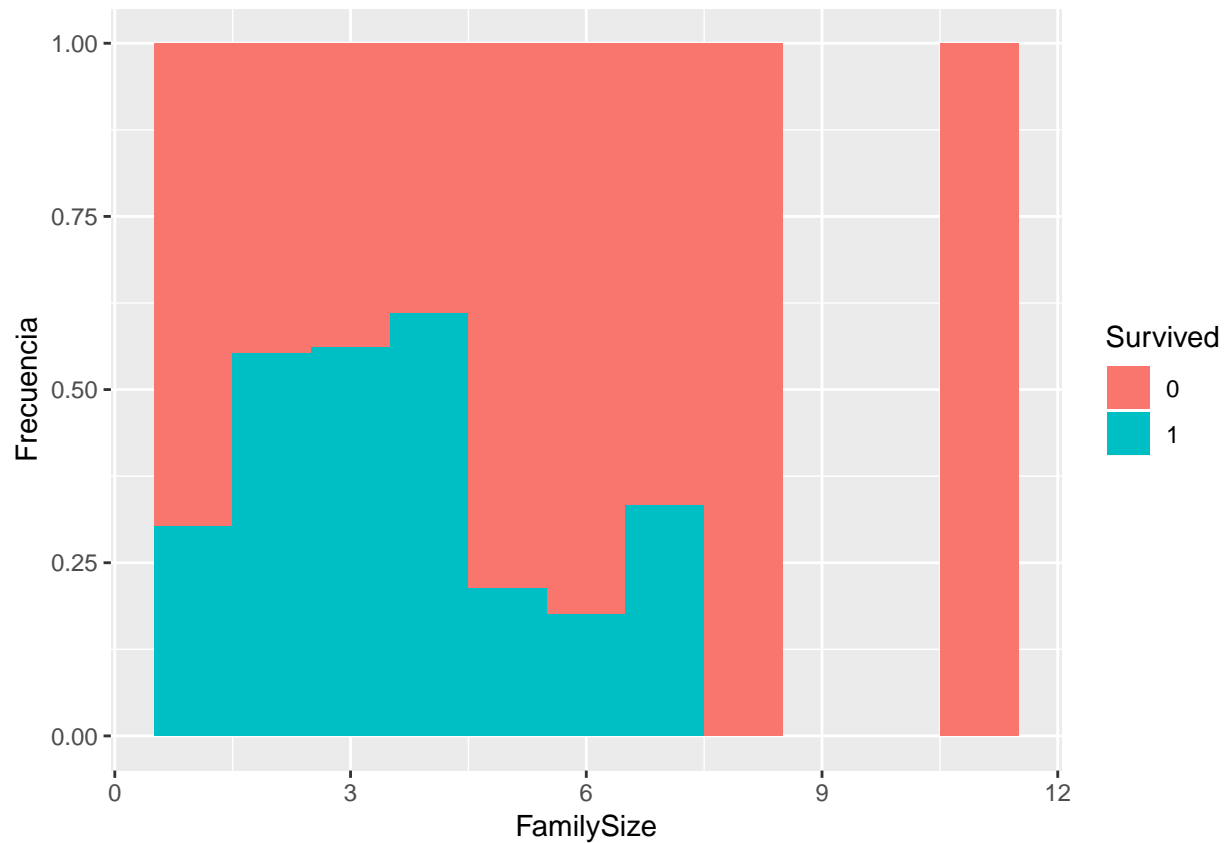
```
ggplot(data = totalData[1:filas,],aes(x=SibSp,fill=Survived))+geom_bar()
```



Vemos como la forma de estos dos gráficos es similar. Este hecho nos puede indicar presencia de correlación

Veamos un ejemplo de construcción de una variable nueva: Tamaño 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_bar()
```

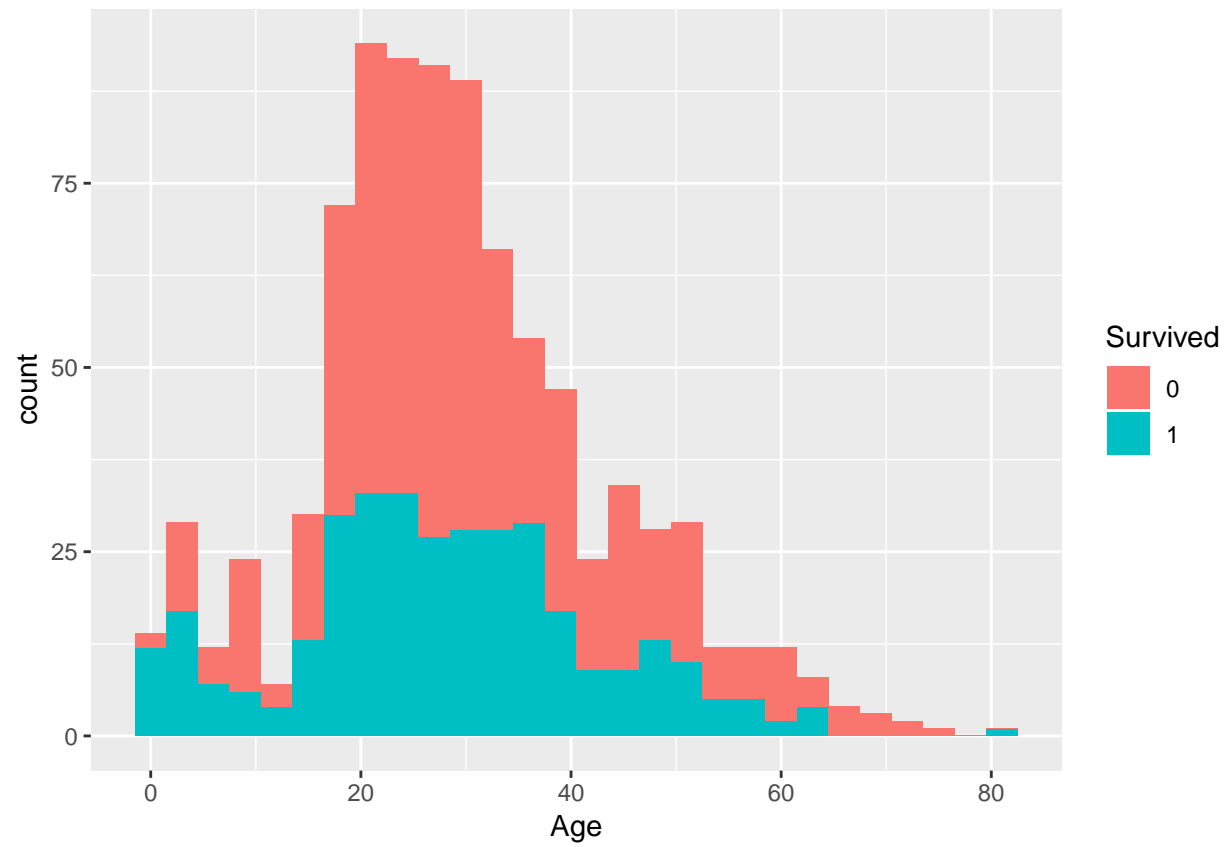


Observamos como familias de entre 2 y 6 miembros tienen más del 50% de posibilidades de supervivencia

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(binwidth=5)
```

