Practica 2

M2.951 - Tipologia i cicle de vida de les dades aula 1
2019-1 · Màster universitari en Ciència de dades (Data science)
Estudis de Informàtica, Multimèdia i Telecomunicacions
Joan Carles Badia Purroy

Carreguem el fitxer. Afegim la opció per a que els camps de text no els consideri un Factor.

```
library(tidyverse)
```

```
## ── Attaching packages ─────────────────────────────────────
──────────────────────────────────────────────── tidyverse 1.3.0 ──
```

```
## ✓ ggplot2 3.2.1     ✓ purrr   0.3.3
## ✓ tibble  2.1.3     ✓ dplyr   0.8.3
## ✓ tidyr   1.0.0     ✓ stringr 1.4.0
## ✓ readr   1.3.1     ✓ forcats 0.4.0
```

```
## ── Conflicts ────────────────────────────────────────────────
──────────────────────────────────────────── tidyverse_conflicts() ──
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```
train <- read.csv("~/Documents/Master UOC/Tipologia i cicle de vi
da de dades/Practica2/train.csv", stringsAsFactors=FALSE)
test <- read.csv("~/Documents/Master UOC/Tipologia i cicle de vid
a de dades/Practica2/test.csv", stringsAsFactors=FALSE)
str(train)
```

```
## 'data.frame':    891 obs. of  12 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 Harris" "Cumings, Mrs.
John Bradley (Florence Briggs Thayer)" "Heikkinen, Miss. Laina" "
Futrelle, Mrs. Jacques Heath (Lily May Peel)" ...
##  $ Sex        : chr  "male" "female" "female" "female" ...
##  $ Age        : num  22 38 26 35 35 NA 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 ...
##  $ Ticket     : chr  "A/5 21171" "PC 17599" "STON/O2. 3101282"
"113803" ...
##  $ Fare       : num  7.25 71.28 7.92 53.1 8.05 ...
##  $ Cabin      : chr  "" "C85" "" "C123" ...
##  $ Embarked   : chr  "S" "C" "S" "S" ...
```

1. Descripció del dataset. Perquè és important i quina pregunta/problema pretén respondre?

El dataset conté informació sobre els passatgers del Titanic. De cada passatger tenim una sèrie de variables com ara el nom, edat, gènere, classe econòmica-social,.. i la variable que indica si es va salvar o no. La idea que hi ha al darrera d'aquest fitxer és la de trobar un model automàtic capaç de predir si el passatger es va salvar o no en funció de la resta de variables.

El dataset train conté 891 registres, amb 12 variables cadascun. El dataset test conté 418 registres amb 11 variables cadascun.

El dataset test no conté la variable objectiu Survived. Això és perquè forma part d'una competició en la que es tracta d"entrenar el model per tal de predir aquesta variable per al conjunt de dades test.

PassengerId : variable identificativa $ Survived : Variable objectiu qualitativa $ Pclass : Variable qualitativa $ Name : Qualitativa $ Sex : qualitativa $ Age : Quantitativa discreta $ SibSp : Quantitativa discreta $ Parch : Quantitativa discreta $ Ticket : Qualitativa $ Fare : Quantitativa contínua $ Cabin : Qualitativa $ Embarked : Qualitativa

Analitzem ara cadascuna de les variables:

La variable objectiu : Survived

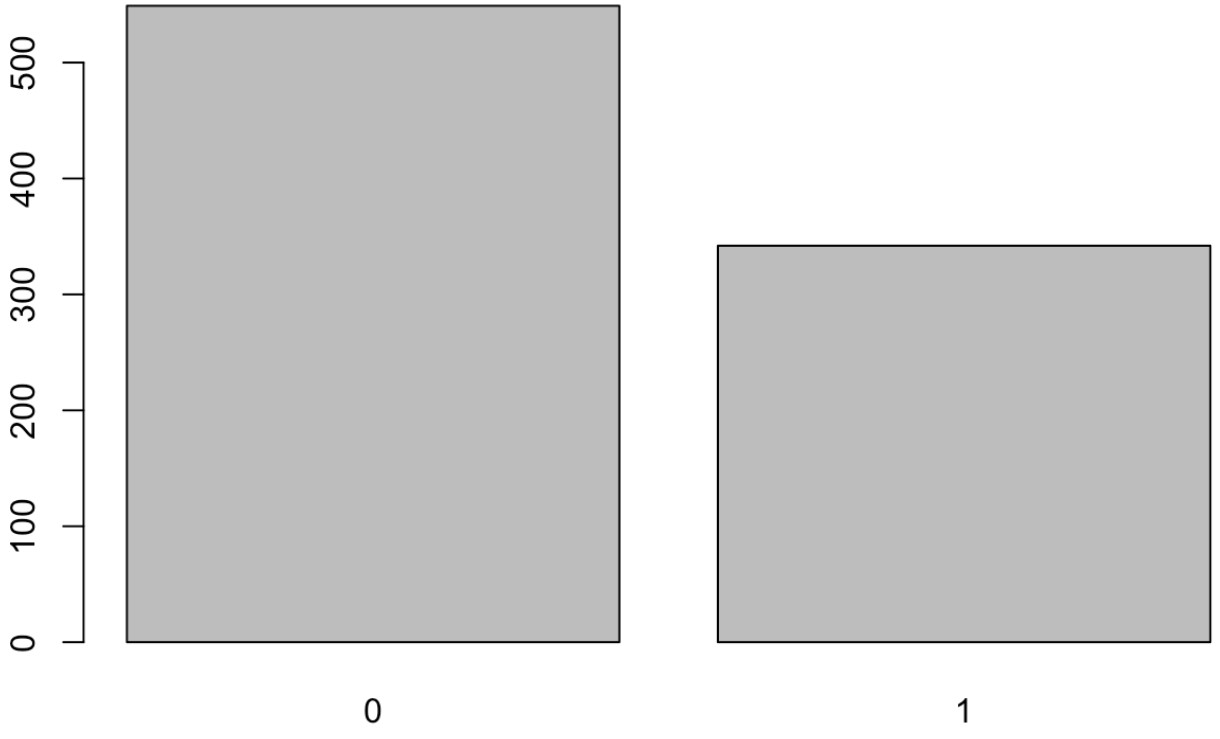
La variable Survived és de tipus booleà. Ens indica si el passatger és viu o mort. El domini de valors és el següent: 0 = No, 1 = Yes

Passem el camp Survived a factors:

```
train$Survived <- as.factor(train$Survived)
print(levels(train$Survived))
```

```
## [1] "0" "1"
```

```
plot(train$Survived)
```



La variable Pclass

Ens indica la classe del bitllet en la que viatjava el passatger. Ens indica de retruc el

estatus social. El domini de valors és:
1 = 1st, 2 = 2nd, 3 = 3rd
Tot i tractar-se d'un enter, és una variable categòrica ja que descriu un valor categòric amb el que no es pot fer operacions aritmètiques.

Passem el camp Pclass a factors:

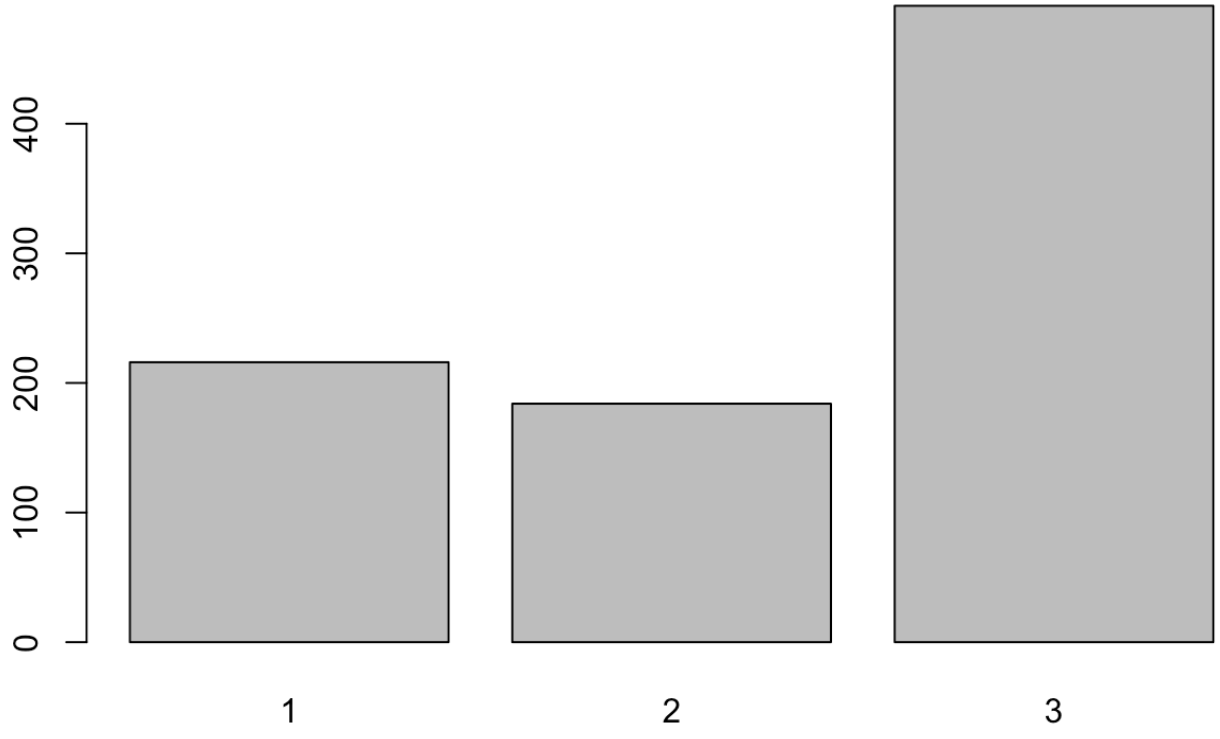
```
train$Pclass <- as.factor(train$Pclass)
print(levels(train$Pclass))
```

```
## [1] "1" "2" "3"
```

```
test$Pclass <- as.factor(test$Pclass)
print(levels(test$Pclass))
```

```
## [1] "1" "2" "3"
```

```
plot(train$Pclass)
```



La variable Sex És indica si el passatger era home o dona.
el domini de valors és: male / female

Passem el camp Sex a factors:

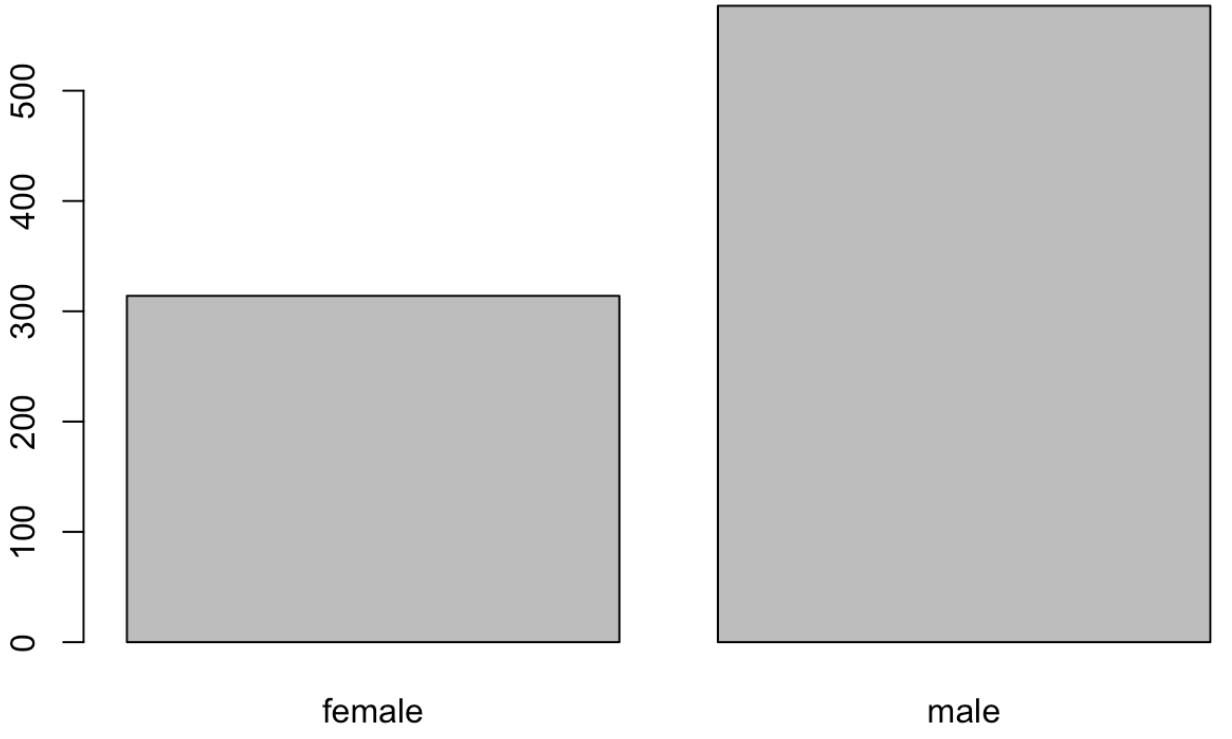
```
train$Sex <- as.factor(train$Sex)
print(levels(train$Sex))
```

```
## [1] "female" "male"
```

```
test$Sex <- as.factor(test$Sex)
print(levels(test$Sex))
```

```
## [1] "female" "male"
```

```
plot(train$Sex)
```



La variable Embarked
Variable categòrica. Indica el port en que van embarcar. Pot adoptar els següents valors:
C = Cherbourg, Q = Queenstown, S = Southampton

Passem el camp Embarked a factors:

```
train$Embarked <- as.factor(train$Embarked)
print(levels(train$Embarked))
```

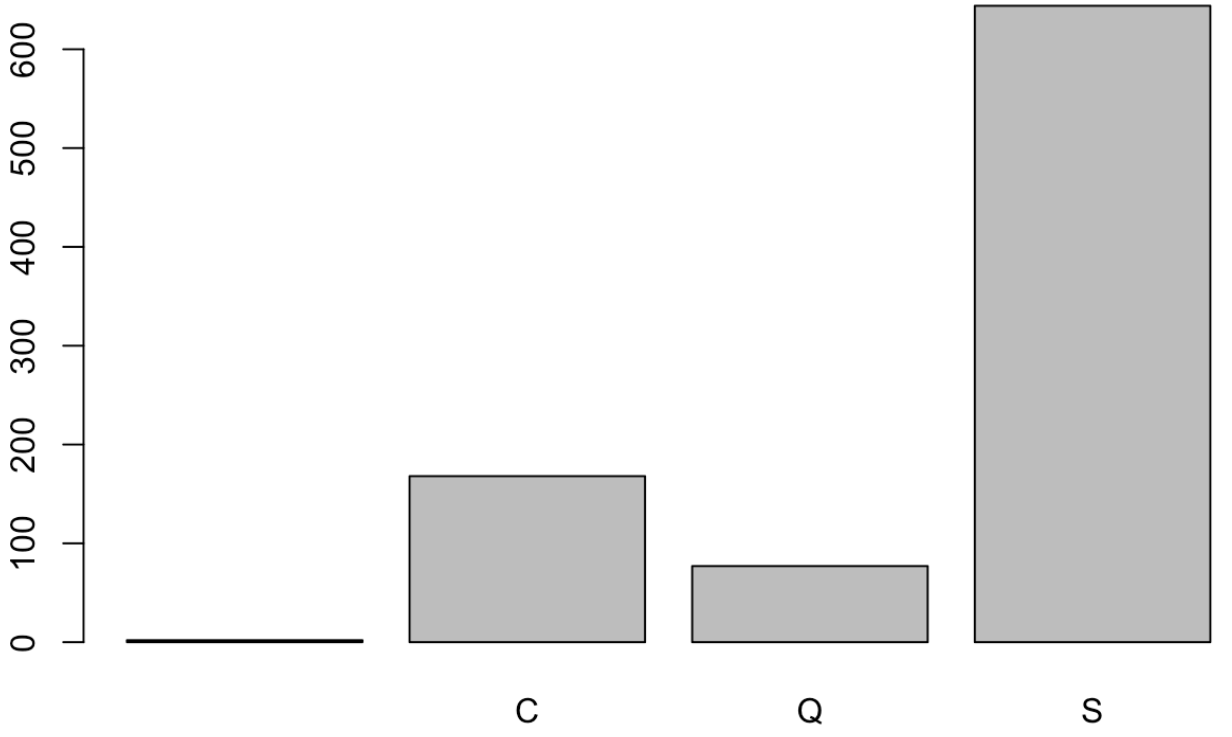
```
## [1] ""  "C" "Q" "S"
```

```
test$Embarked <- as.factor(test$Embarked)
print(levels(test$Embarked))
```

```
## [1] "C" "Q" "S"
```

N'hi ha que tenen valor desconegut.

```
plot(train$Embarked)
```



La variable cabin
el número de la cabina del passatger

Variables numèriques

Comprovem el tipus de les variables numèriques:

```
is.numeric(train$Age)
```

```
## [1] TRUE
```

```
is.numeric(test$Age)
```

```
## [1] TRUE
```

```
is.numeric(train$SibSp)
```

```
## [1] TRUE
```

```
is.numeric(test$SibSp)
```

```
## [1] TRUE
```

```
is.numeric(train$Parch)
```

```
## [1] TRUE
```

```
is.numeric(test$Parch)
```

```
## [1] TRUE
```

```
is.numeric(train$Fare)
```

```
## [1] TRUE
```

```
is.numeric(test$Fare)
```

```
## [1] TRUE
```

La variable Age

Variable numèrica. Indica la edat del passatger en anys. Si la edat és inferior a 1 el nombre és fraccional. Si l'edat és estimada, pren la forma xx.5.

Busquem possibles valors nuls:

```
which(is.na(train$Age))
```

```
##   [1]   6  18  20  27  29  30  32  33  37  43  46  47  48  49
56  65  66  77
##  [19]  78  83  88  96 102 108 110 122 127 129 141 155 159 160
167 169 177 181
##  [37] 182 186 187 197 199 202 215 224 230 236 241 242 251 257
261 265 271 275
##  [55] 278 285 296 299 301 302 304 305 307 325 331 335 336 348
352 355 359 360
##  [73] 365 368 369 376 385 389 410 411 412 414 416 421 426 429
432 445 452 455
##  [91] 458 460 465 467 469 471 476 482 486 491 496 498 503 508
512 518 523 525
## [109] 528 532 534 539 548 553 558 561 564 565 569 574 579 585
590 594 597 599
## [127] 602 603 612 613 614 630 634 640 644 649 651 654 657 668
670 675 681 693
## [145] 698 710 712 719 728 733 739 740 741 761 767 769 774 777
779 784 791 793
## [163] 794 816 826 827 829 833 838 840 847 850 860 864 869 879
889
```

```
which(is.na(test$Age))
```

```
##  [1]  11  23  30  34  37  40  42  48  55  59  66  77  84  85
86  89  92  94 103
## [20] 108 109 112 117 122 125 128 133 134 147 149 152 161 164 1
69 171 174 184 189
## [39] 192 200 201 206 212 217 220 226 228 234 244 245 250 256 2
57 266 267 268 269
## [58] 272 274 275 283 287 289 290 291 293 298 302 305 313 333 3
40 343 345 358 359
## [77] 366 367 381 383 385 409 411 414 417 418
```

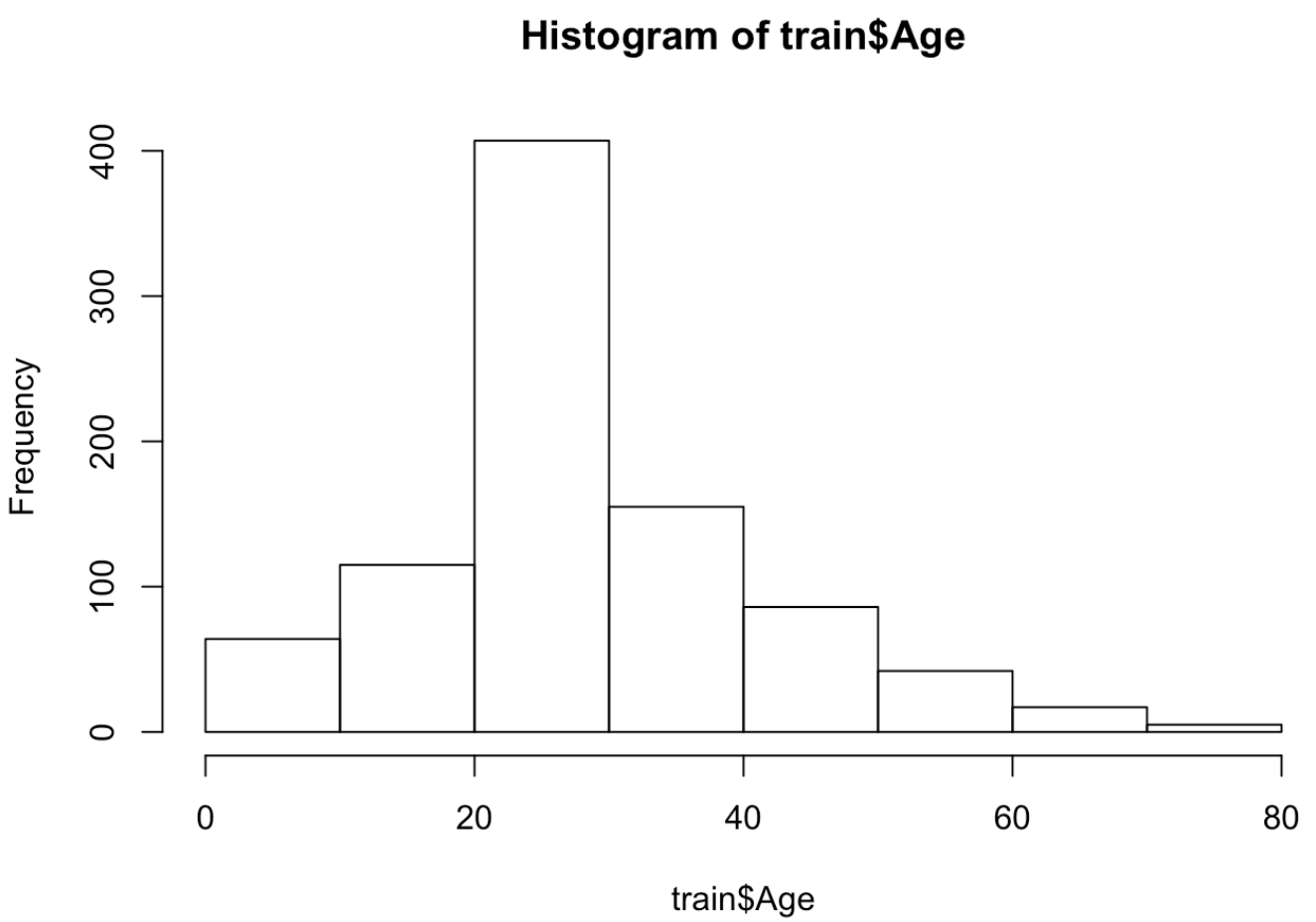
Hi ha valors nuls. Podem optar per substituir-ho per la mitjana per tal de que no alteri les dades. Calculem la mitjana dels valors que no son NaN:

```
mitjanaAge = mean(c(train$Age,test$Age), na.rm = TRUE)
print(mitjanaAge)
```

```
## [1] 29.88114
```

```
library("imputeTS")
train$Age <- na_replace(train$Age, mitjanaAge)
test$Age <- na_replace(test$Age, mitjanaAge)
```

```
graphics::hist(train$Age)
```



La variable Ticket
El nombre del tiquet de viatge

La variable Fare

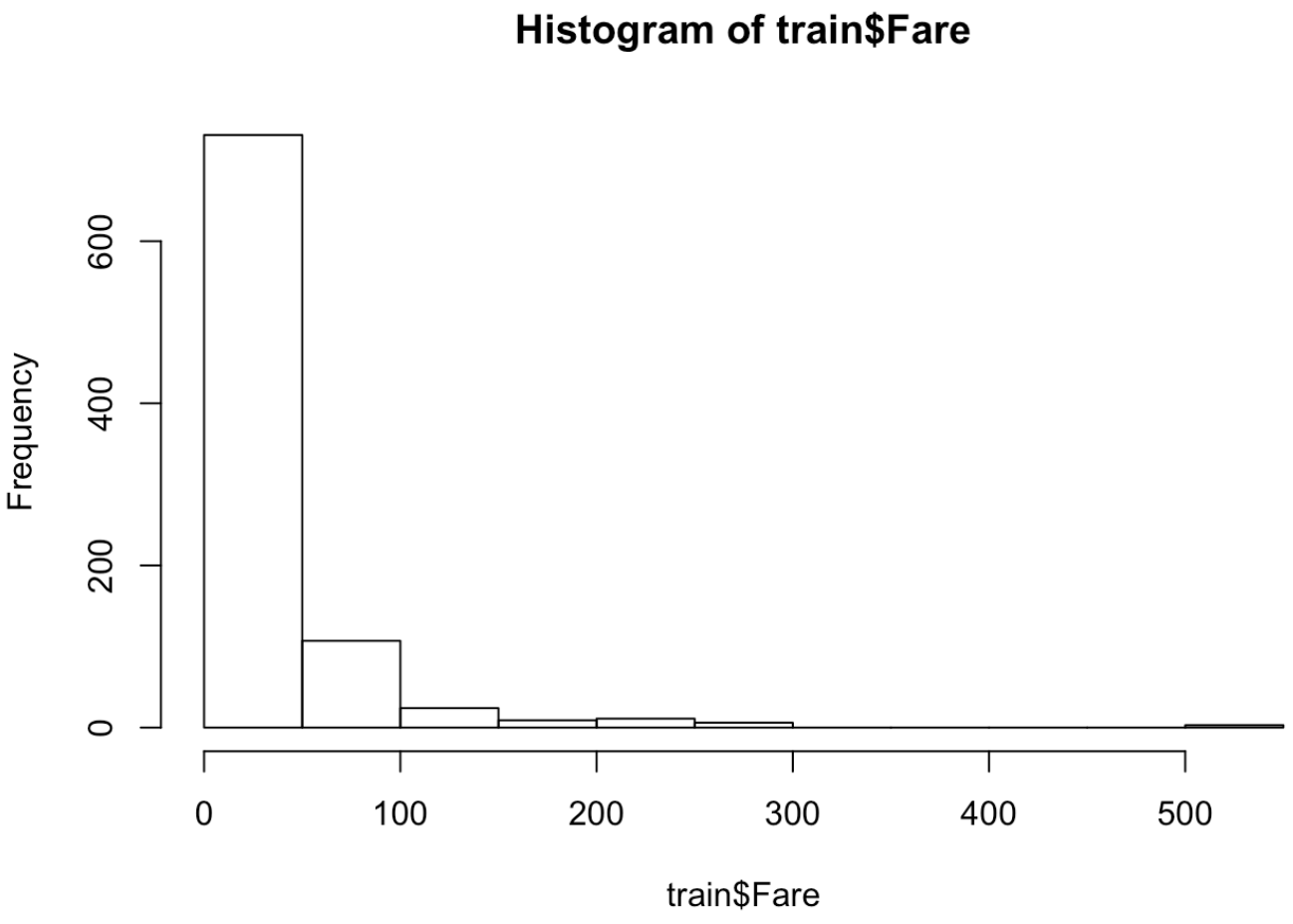
El preu del bitllet que va pagar el passatger

Busquem possibles valors nuls:

```
which(is.nan(train$Fare))
```

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

```
graphics::hist(train$Fare)
```



La variable Cabin
el número de la cabina del passatger

```
print(head(train$Cabin,10))
```

```
##  [1] ""     "C85"  ""     "C123" ""     ""     "E46"  ""     "
"     ""
```

La variable PassengerId

Indica l'identificador únic per al passatger. És un comptador.

La variable Name

Indica el nom del passatger. No és rellevant per a l'estudi, ja que no ens proporciona cap eina per saber si va sobreviure o no.

Analitzem ara quines son les variables que poden ser útils per a la predicció de supervivència.

Descartem:

PassengerId:

És un simple identificador únic de cada registre que no aporta cap informació.
Name:
El nom del passatger tampoc és analitzable.
Ticket:
El número del ticket no ens aporta a priori massa informació útil.
Cabin:
Per si no es de fàcil estudi. A no se que se'n pugui extreure alguna altra informació com ara la coberta a la que pertany.

Variables que, a priori, poden influir en la predicció :

Pclass:

La classe social és important. A major classe social més possibilitats a priori de sobreviure. #### Sex: El sexe també es important. En un accident solen evacuar primer als passatgers de sexe femení. #### Age: L'edat també influeix. Els nens es solen evacuar primer… #### SibSp: nombre de parents per persona
Parch: Nombre de fills/pares per persona #### Fare:
El preu del bitllet a priori hauria de estar correlacionat amb la classe Pclass, per tant estudiarem si podem eliminar-lo i quedar-nos amb un dels dos.
Embarked: En principi no sembla que hagi d'influir en la possibilitat de sobreviure o no, però potser inclou alguna una relació que desconeixem.

Valors extrems de les variables quantitatives discretes

Busquem els valors extrems a les variables quantitatives discretes que ens poden alterar els estadístics.

Age

```
library(car)
```

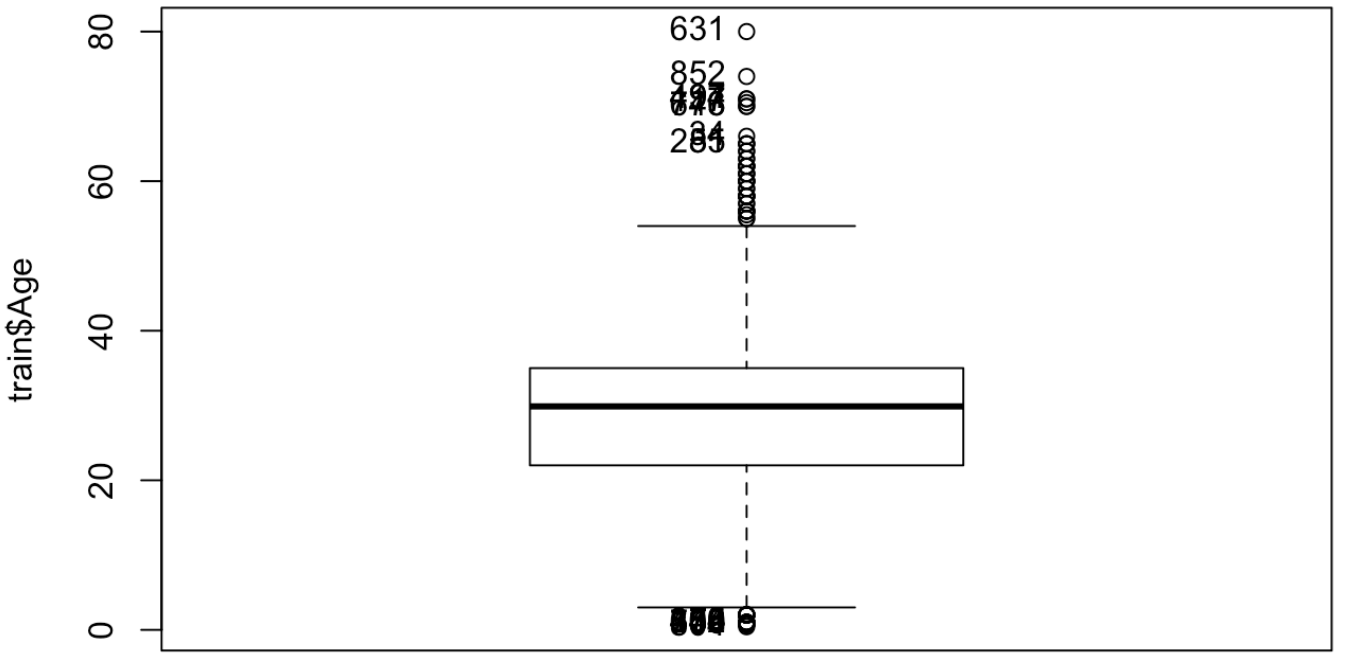
```
## Loading required package: carData
```

```
##
## Attaching package: 'car'
```

```
## The following object is masked from 'package:dplyr':
##
##     recode
```

```
## The following object is masked from 'package:purrr':
##
##     some
```

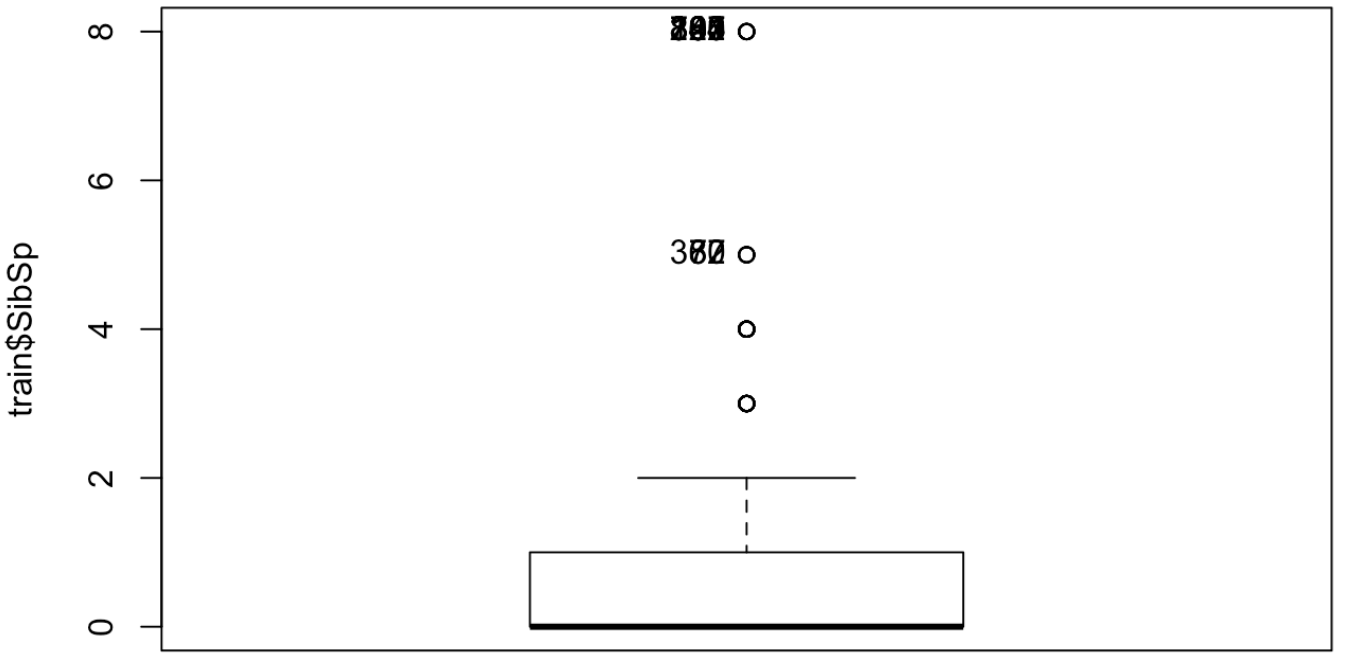
```
Boxplot(train$Age,id=TRUE)
```



```
##  [1] 804 756 470 645  79 832 306 165 173 184 631 852  97 494 1
17 673 746  34  55
## [20] 281
```

SibSp

```
library(car)
Boxplot(train$SibSp,id=TRUE)
```




```
##  [1] 160 181 202 325 793 847 864  60  72 387
```

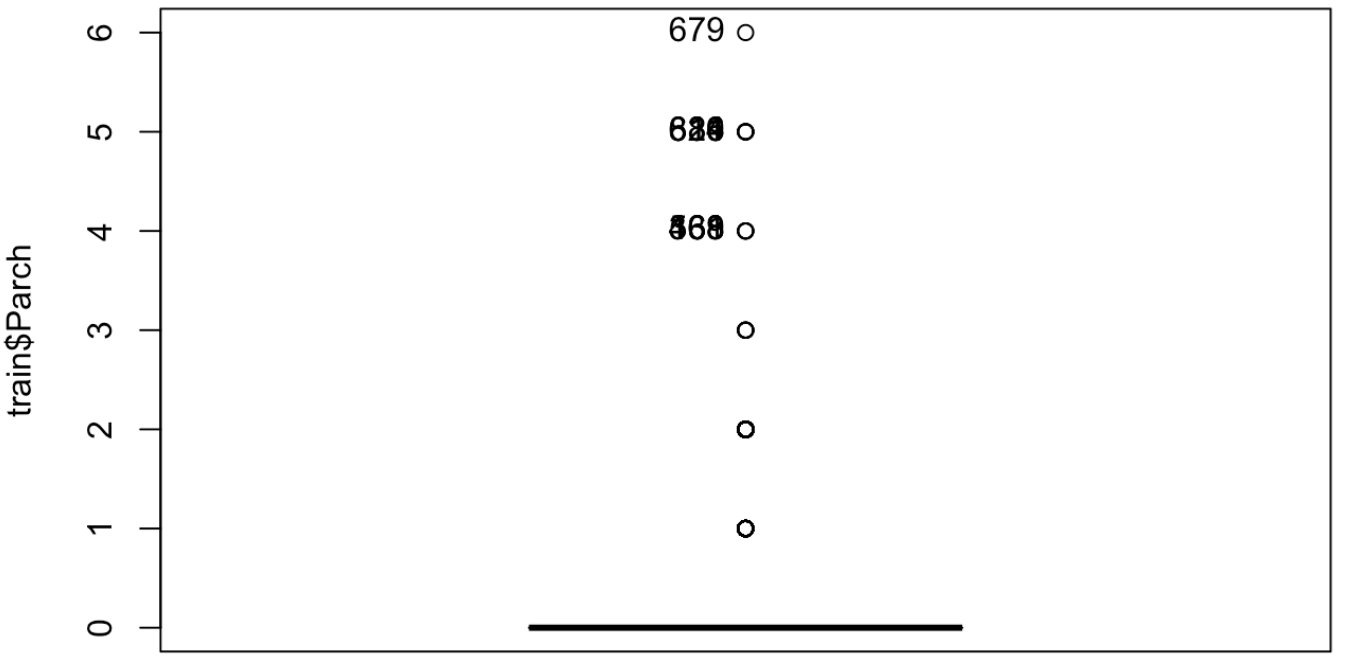
```
outliers8 = which(train$SibSp==8)
print(train[outliers8,])
```

```
##     PassengerId Survived Pclass                                  N
ame    Sex
## 160         160        0      3        Sage, Master. Thomas He
nry   male
## 181         181        0      3      Sage, Miss. Constance Gla
dys female
## 202         202        0      3               Sage, Mr. Freder
ick   male
## 325         325        0      3          Sage, Mr. George John
Jr    male
## 793         793        0      3           Sage, Miss. Stella A
nna female
## 847         847        0      3          Sage, Mr. Douglas Bul
len   male
## 864         864        0      3 Sage, Miss. Dorothy Edith "Dol
ly" female
##          Age SibSp Parch   Ticket  Fare Cabin Embarked
## 160 29.88114     8     2 CA. 2343 69.55              S
## 181 29.88114     8     2 CA. 2343 69.55              S
## 202 29.88114     8     2 CA. 2343 69.55              S
## 325 29.88114     8     2 CA. 2343 69.55              S
## 793 29.88114     8     2 CA. 2343 69.55              S
## 847 29.88114     8     2 CA. 2343 69.55              S
## 864 29.88114     8     2 CA. 2343 69.55              S
```

Comprovem, pel nom, que els que son 8 parents, son familia realment.

Parch

```
library(car)
Boxplot(train$Parch,id=TRUE)
```

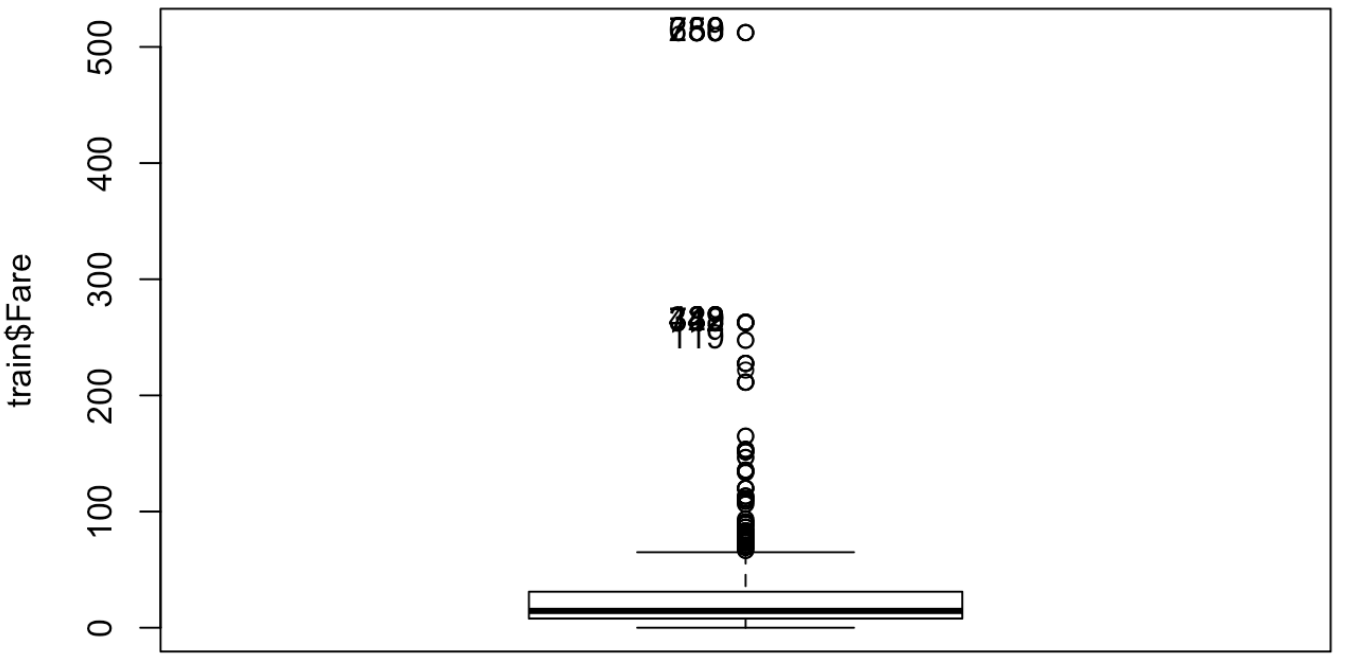



```
##  [1] 679  14  26 611 639 886 168 361 439 568
```

Tot i que hi ha un valor amb 6 parents, no es considera desmesurat.

Fare

```
library(car)
Boxplot(train$Fare,id=TRUE)
```




```
##  [1] 259 680 738  28  89 342 439 312 743 119
```

```
outliers = which(train$Fare>500)
print(train[outliers,])
```

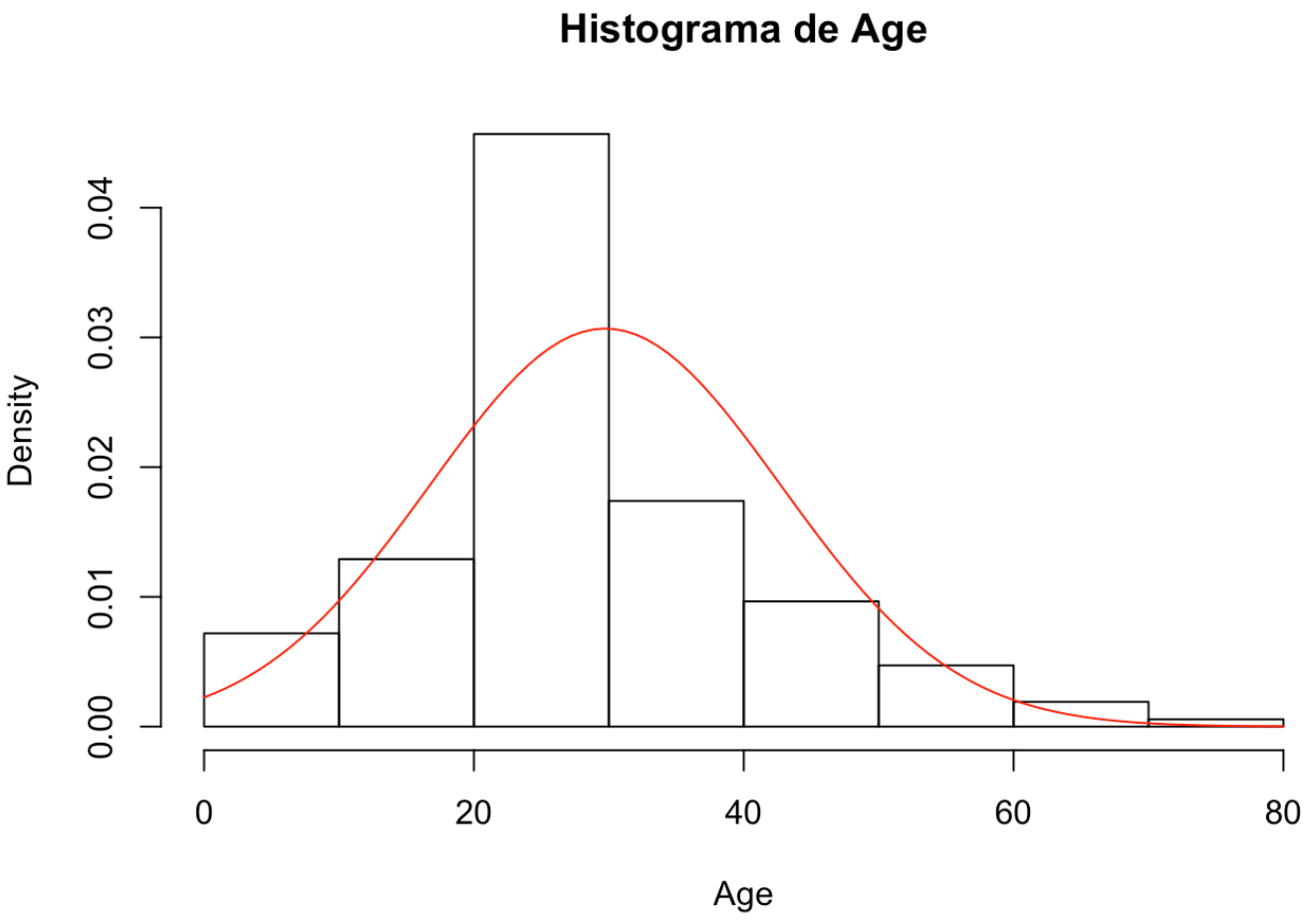
```
##     PassengerId Survived Pclass
Name    Sex Age
## 259         259        1      1                     Ward, Miss.
Anna female  35
## 680         680        1      1 Cardeza, Mr. Thomas Drake Mart
inez   male  36
## 738         738        1      1              Lesurer, Mr. Gusta
ve J   male  35
##     SibSp Parch   Ticket     Fare       Cabin Embarked
## 259     0     0 PC 17755 512.3292                   C
## 680     0     1 PC 17755 512.3292 B51 B53 B55        C
## 738     0     0 PC 17755 512.3292        B101        C
```

Normalitat de les dades

Comproveu si es compleix l'assumpció de normalitat en les dades.

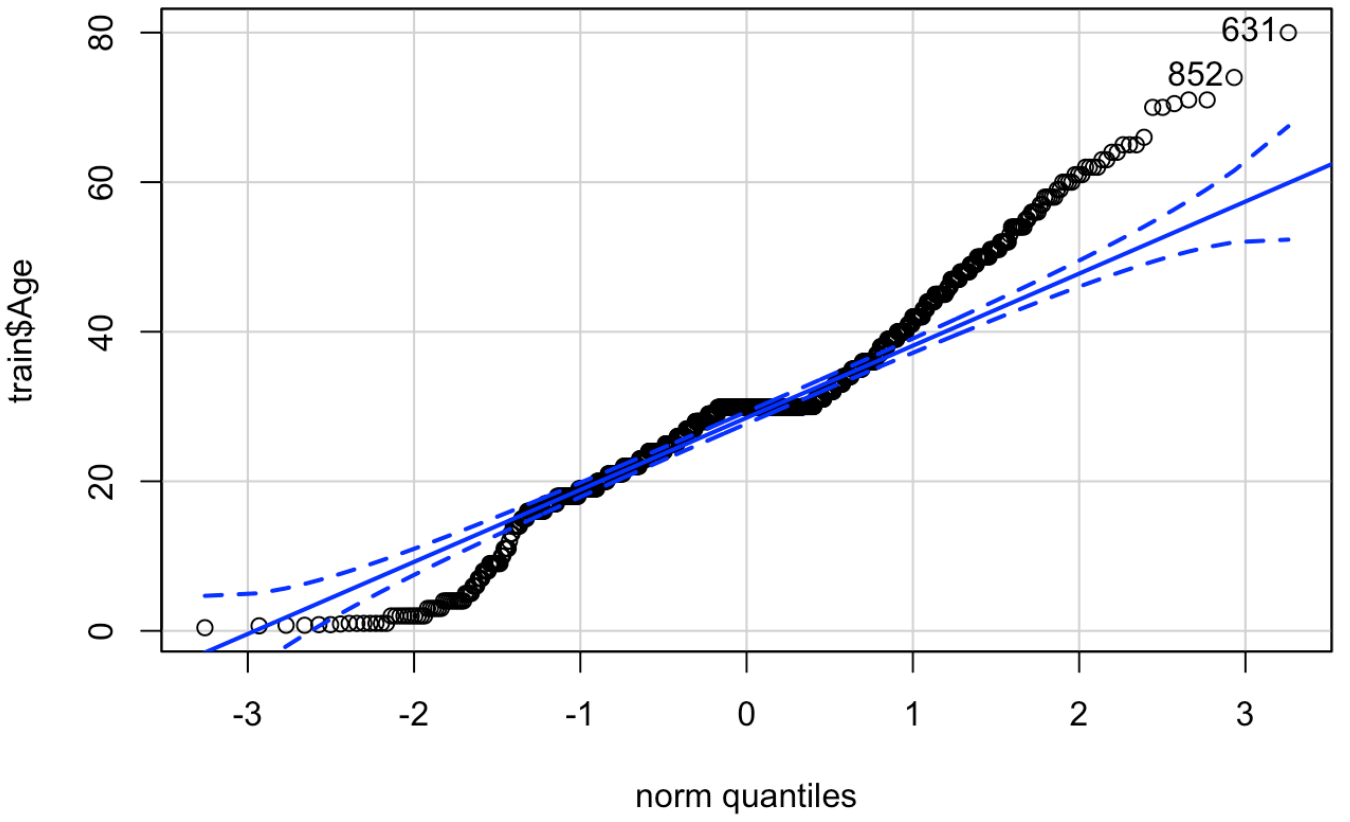
Age

```
hist(train$Age,xlim=range(0:80),main="Histograma de Age",xlab="Age",freq=FALSE)
curve(dnorm(x,mean(train$Age),sd=sd(train$Age)), add=TRUE,col="red")
```



Amb l'histograma podem veure una forma de tipus Gaussià, que tot i que no és la forma de campana familiar, és una aproximació.

```
qqPlot(train$Age)
```




```
## [1] 631 852
```

Podem veure el QQPlot que hi ha valors dels extrems que es surten de la linea. Està desviat cap a la dreta. Això indica que la major part de les dades es concentren cap a la part baixa.

Correlació entre variables

Sospitem que hi ha una relació entre la classe del bitllet i el cost del bitllet.

```
library(MASS)
```

```
##
## Attaching package: 'MASS'
```

```
## The following object is masked from 'package:dplyr':
##
##     select
```

```
train_pclass_fare = subset(train, select=c("Pclass", "Fare"))
table_pclass_fare <-table(train_pclass_fare)
x <- prop.table(table_pclass_fare)            #Por defecto propor
ciones totales.

#cor.test(train_pclass_fare$Fare , train_pclass_fare$Pclass , met
hod = "spearman", alternative = "two.sided")
```

La prova de Chi^2 ens donara la probabilitat d'independència.
h0: Les dades son independents
h1: No hi ha independencia entre les dos variables

```
chisq.test(table_pclass_fare)
```

```
## Warning in chisq.test(table_pclass_fare): Chi-squared approxim
ation may be
## incorrect
```

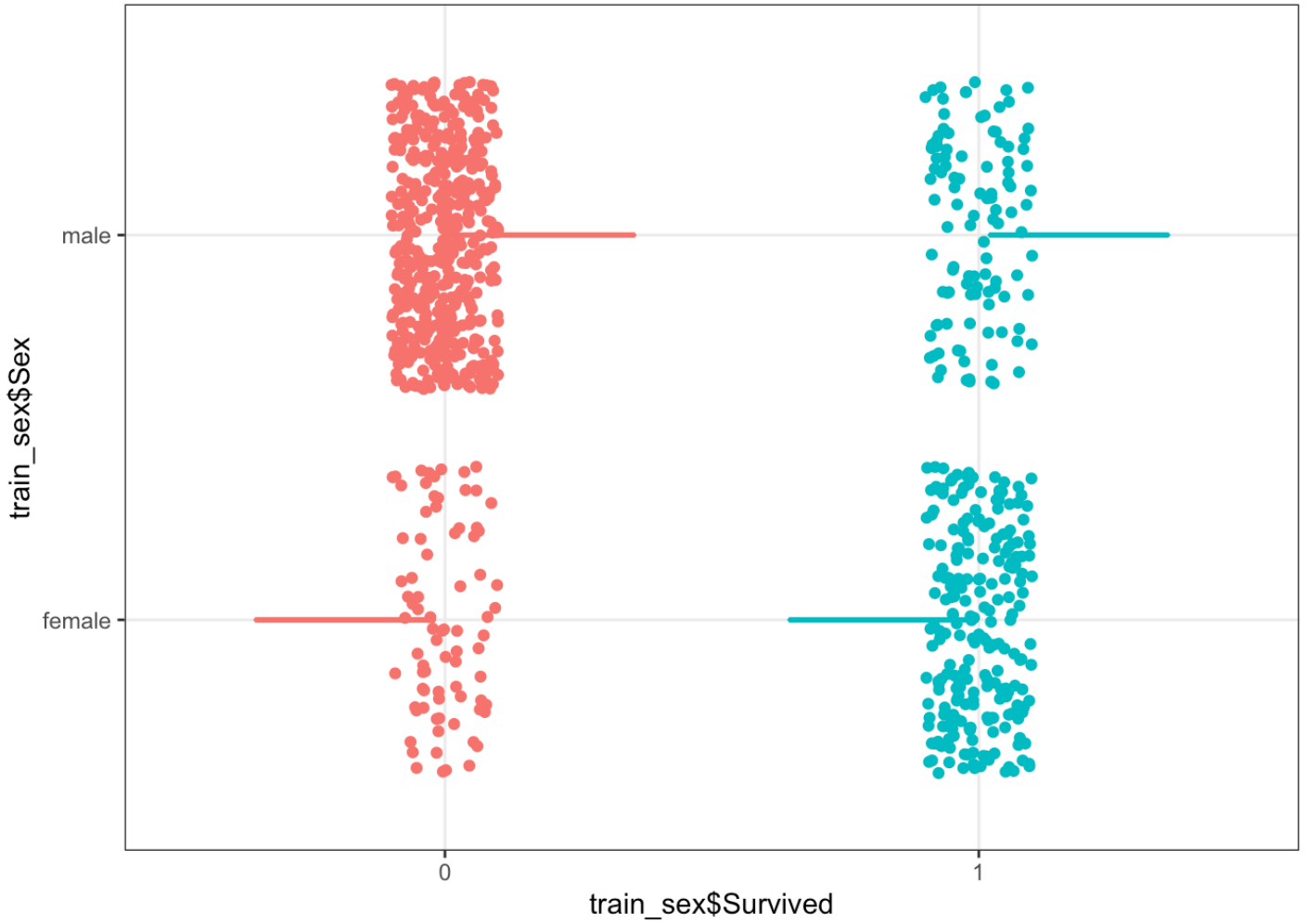
```
##
##  Pearson's Chi-squared test
##
## data:  table_pclass_fare
## X-squared = 1697.8, df = 494, p-value < 2.2e-16
```

Com p<0.05 rebutjem la hipotesis h0. No podem afirmar que les dues variables siguin independents.

Ens preguntem ara si existeixen diferències significatives en el grau de supervivència (Survived) dels homes en relació a les dones. Distribució de Survived d'homes i dones per separat en un boxplot

```
library(ggplot2)
train_sex = subset(train, select=c("Sex", "Survived"))
```

```
ggplot(data = train_sex, aes(x = train_sex$Survived, y = train_se
x$Sex, color = train_sex$Survived)) +
  geom_boxplot(outlier.shape = NA) +
  geom_jitter(width = 0.1) +
  theme_bw() +
  theme(legend.position = "null")
```



Com es pot veure pel gràfic, si que hi ha diferències significatives pel que fa al sexe. Els Survived es decanten pel sexe female i els Survived=0 pel male.

Anem a fer un model de regressió simple utilitzant com a variable dependent Survived i independent sex.

```
modelo <- glm(train_sex$Survived ~ train_sex$Sex, data = train_se
x, family = "binomial")
summary(modelo)
```

```
##
## Call:
## glm(formula = train_sex$Survived ~ train_sex$Sex, family = "bi
nomial",
##     data = train_sex)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.6462  -0.6471  -0.6471   0.7725   1.8256
##
## Coefficients:
##                   Estimate Std. Error z value Pr(>|z|)
## (Intercept)         1.0566     0.1290   8.191 2.58e-16 ***
## train_sex$Sexmale  -2.5137     0.1672 -15.036  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 1186.7  on 890  degrees of freedom
## Residual deviance:  917.8  on 889  degrees of freedom
## AIC: 921.8
##
## Number of Fisher Scoring iterations: 4
```

```
confint(object = modelo, level = 0.95 )
```

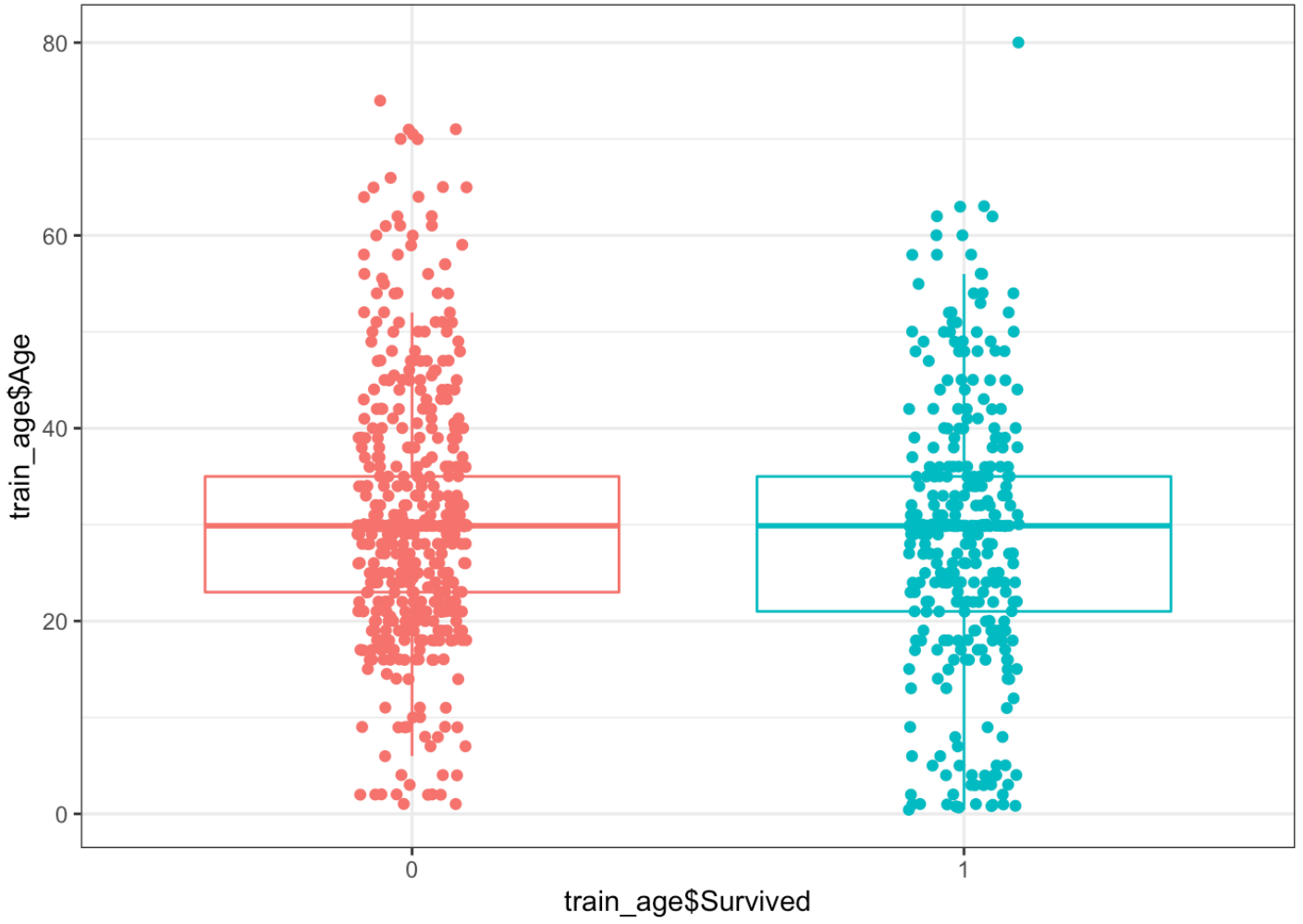
```
## Waiting for profiling to be done...
```

```
##                       2.5 %    97.5 %
## (Intercept)        0.8085881  1.314934
## train_sex$Sexmale -2.8465007 -2.190728
```

Sembla que el Sexe és significatiu, i el valor ‘male’ determina negativament la possibilitat de supervivència.

Provem a veure si la edat es significativa:

```
train_age = subset(train, select=c("Age", "Survived"))
ggplot(data = train_age, aes(x = train_age$Survived, y = train_ag
e$Age, color = train_age$Survived)) +
  geom_boxplot(outlier.shape = NA) +
  geom_jitter(width = 0.1) +
  theme_bw() +
  theme(legend.position = "null")
```



Pel que podem veure no sembla massa determinant. No hi ha grans diferències. Afegim l'edat al model de regressió:

```
modelo <- glm(train_age$Survived ~ train_age$Age, data = train_ag
e, family = "binomial")
summary(modelo)
```

```
##
## Call:
## glm(formula = train_age$Survived ~ train_age$Age, family = "bi
nomial",
##     data = train_age)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.1137  -0.9863  -0.9429   1.3613   1.6400
##
## Coefficients:
##                Estimate Std. Error z value Pr(>|z|)
## (Intercept)   -0.14045    0.17226  -0.815   0.4149
## train_age$Age -0.01128    0.00539  -2.093   0.0364 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 1186.7  on 890  degrees of freedom
## Residual deviance: 1182.2  on 889  degrees of freedom
## AIC: 1186.2
##
## Number of Fisher Scoring iterations: 4
```

Sembla que l'edat és lleugerament significativa però no massa.

Provem ara, la classe del bitllet. Pclass. Fem un model de regressió simple:

```
train_pclass = subset(train, select=c("Pclass", "Survived"))

modelo <- glm(train_pclass$Survived ~ train_pclass$Pclass, data =
train_pclass, family = "binomial")
summary(modelo)
```

```
##
## Call:
## glm(formula = train_pclass$Survived ~ train_pclass$Pclass, fam
ily = "binomial",
##     data = train_pclass)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.4094  -0.7450  -0.7450   0.9619   1.6836
##
## Coefficients:
##                       Estimate Std. Error z value Pr(>|z|)
## (Intercept)             0.5306     0.1409   3.766 0.000166 ***
## train_pclass$Pclass2   -0.6394     0.2041  -3.133 0.001731 **
## train_pclass$Pclass3   -1.6704     0.1759  -9.496  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 1186.7  on 890  degrees of freedom
## Residual deviance: 1083.1  on 888  degrees of freedom
## AIC: 1089.1
##
## Number of Fisher Scoring iterations: 4
```

La classe del bitllet si es significativa.

Model de regressió logísti múltiple.

Afegirem la variable de l'edat i la classe a la del sexe per a fer un model logístic múltiple:

```
train_sex_age_pclass = subset(train, select=c("Sex", "Age", "Pcla
ss", "Survived"))


modelo.log.m <- glm(train_sex_age_pclass$Survived ~ . , data = tr
ain_sex_age_pclass, family = binomial)
summary(modelo.log.m)
```

```
##
## Call:
## glm(formula = train_sex_age_pclass$Survived ~ ., family = bino
mial,
##     data = train_sex_age_pclass)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.6494  -0.6638  -0.4189   0.6327   2.4276
##
## Coefficients:
##              Estimate Std. Error z value Pr(>|z|)
## (Intercept)  3.545791   0.365480   9.702  < 2e-16 ***
## Sexmale     -2.611304   0.186717 -13.985  < 2e-16 ***
## Age         -0.033306   0.007366  -4.521 6.15e-06 ***
## Pclass2     -1.122789   0.257775  -4.356 1.33e-05 ***
## Pclass3     -2.328491   0.240816  -9.669  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 1186.66  on 890  degrees of freedom
## Residual deviance:  805.27  on 886  degrees of freedom
## AIC: 815.27
##
## Number of Fisher Scoring iterations: 5
```

```
confint(object = modelo.log.m, level = 0.95)
```

```
## Waiting for profiling to be done...
```

```
##                  2.5 %      97.5 %
## (Intercept)  2.84722560  4.28137163
## Sexmale     -2.98508659 -2.25231692
## Age         -0.04797559 -0.01906295
## Pclass2     -1.63402766 -0.62242054
## Pclass3     -2.81016276 -1.86500109
```

La edat no sembla massa determinant per la supervivència combinada tot i que influeix negativament. Aleshores, la variable que més influeix és el sexe. En el cas dels homes en sentit negatiu, es a dir, els homes tenen menys possibilitats de

sobreviure. L'altre paràmetre que afecta més en segona posició és la classe social. Com més baixa menys possibilitats de supervivència. I finalment la edat, com més alta menys possibilitats de supervivència, tot i que influeix molt poc.

Algorismes de classificació

Aplicarem ara alguns algorismes de classificació que ens proporcionaran algun model de predicció.

Primer que res dividim el fitxer de train en train i test i n'agafarem les columnes que volem utilitzar:

```
library(caret)
```

```
## Loading required package: lattice
```

```
##
## Attaching package: 'caret'
```

```
## The following object is masked from 'package:purrr':
##
##     lift
```

```
library(rpart.plot)
```

```
## Loading required package: rpart
```

```
library(lattice)
library(ggplot2)
library(rpart)

trainIndex = createDataPartition(train$Survived,
                      p=0.7, list=FALSE,times=1)
```

```
subtrain = subset(train[trainIndex,],select=c("Sex", "Age", "Pcla
ss","SibSp","Parch", "Survived"))
subtest = subset(train[-trainIndex,],select=c("Sex", "Age", "Pcla
ss","SibSp","Parch", "Survived"))

#print(subtrain)
#print(subtest)
```

```
summary(subtrain)
```

```
##      Sex           Age        Pclass      SibSp            Par
ch     Survived
##  female:228   Min.   : 0.42   1:151   Min.   :0.0000   Min.
:0.0   0:385
##  male  :397   1st Qu.:22.00   2:126   1st Qu.:0.0000   1st Qu.
:0.0   1:240
##               Median :29.88   3:348   Median :0.0000   Median
:0.0
##               Mean   :29.39           Mean   :0.5296   Mean
:0.4
##               3rd Qu.:35.00           3rd Qu.:1.0000   3rd Qu.
:0.0
##               Max.   :80.00           Max.   :8.0000   Max.
:6.0
```

Arbre de decisió

Un cop tenim les dades preparades aplicarem l'algorisme d'arbre de decisió per a obtenir un model de predicció. Utilitzarem "cross-validation repeated" per a provar amb diferents subconjunts d'entrenament (10) amb 3 repeticions. Passem a entrenar l'algorisme amb train. El mètode "rpart" indica de quina manera es divideixen els nodes al classificar a l'arbre.

```
trctrl <- trainControl(method = "repeatedcv", number = 10, repeat
s = 3)
set.seed(3333)
dtree_fit <- train(Survived ~., data = subtrain, method = "rpart"
,
                   parms = list(split = "information"),
                   trControl=trctrl,
                   tuneLength = 10)
```

EL resultat de l'entrenament és el següent:

```
dtree_fit
```

```
## CART
##
## 625 samples
##   5 predictor
##   2 classes: '0', '1'
##
## No pre-processing
## Resampling: Cross-Validated (10 fold, repeated 3 times)
## Summary of sample sizes: 563, 563, 562, 562, 562, 563, ...
## Resampling results across tuning parameters:
##
##   cp          Accuracy   Kappa
##   0.00000000  0.7940690  0.5541891
##   0.04537037  0.7567844  0.4595466
##   0.09074074  0.7728025  0.5154845
##   0.13611111  0.7728025  0.5154845
##   0.18148148  0.7728025  0.5154845
##   0.22685185  0.7728025  0.5154845
##   0.27222222  0.7728025  0.5154845
##   0.31759259  0.7728025  0.5154845
##   0.36296296  0.7728025  0.5154845
##   0.40833333  0.6570148  0.1619415
##
## Accuracy was used to select the optimal model using the larges
t value.
## The final value used for the model was cp = 0.
```

```
test_pred <- predict(dtree_fit, newdata = subtest)
confusionMatrix(test_pred, subtest$Survived )  #check accuracy
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 152  33
##          1  12  69
##
##                Accuracy : 0.8308
##                  95% CI : (0.7803, 0.8738)
##     No Information Rate : 0.6165
##     P-Value [Acc > NIR] : 2.211e-14
##
##                   Kappa : 0.6277
##
##  Mcnemar's Test P-Value : 0.002869
##
##             Sensitivity : 0.9268
##             Specificity : 0.6765
##          Pos Pred Value : 0.8216
##          Neg Pred Value : 0.8519
##              Prevalence : 0.6165
##          Detection Rate : 0.5714
##    Detection Prevalence : 0.6955
##       Balanced Accuracy : 0.8016
##
##        'Positive' Class : 0
##
```

Ara provarem d'entrenar l'algorisme amb un altre criteri de partició de fulles. L'index gini

```
 set.seed(3333)
dtree_fit_gini <- train(Survived ~., data = subtrain, method = "r
part",
                  parms = list(split = "gini"),
                  trControl=trctrl,
                  tuneLength = 10)
dtree_fit_gini
```

```
## CART
##
## 625 samples
##   5 predictor
##   2 classes: '0', '1'
##
## No pre-processing
## Resampling: Cross-Validated (10 fold, repeated 3 times)
## Summary of sample sizes: 563, 563, 562, 562, 562, 563, ...
## Resampling results across tuning parameters:
##
##   cp          Accuracy   Kappa
##   0.00000000  0.7951357  0.5555204
##   0.04537037  0.7567844  0.4595466
##   0.09074074  0.7728025  0.5154845
##   0.13611111  0.7728025  0.5154845
##   0.18148148  0.7728025  0.5154845
##   0.22685185  0.7728025  0.5154845
##   0.27222222  0.7728025  0.5154845
##   0.31759259  0.7728025  0.5154845
##   0.36296296  0.7728025  0.5154845
##   0.40833333  0.6570148  0.1619415
##
## Accuracy was used to select the optimal model using the larges
t value.
## The final value used for the model was cp = 0.
```

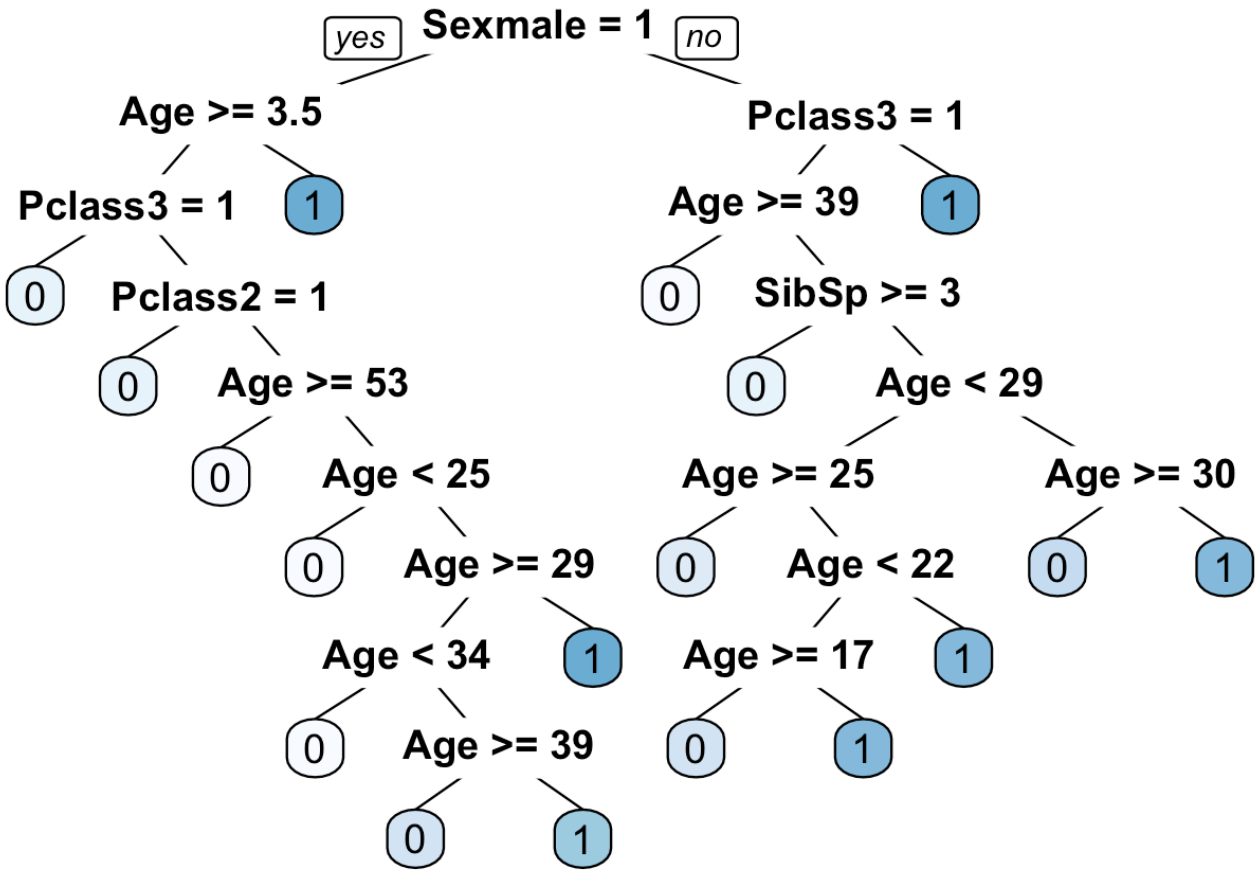
provem de predir els valors del conjunt de test, a veure que tal funciona l'algorisme. Treiem la matriu de confusió:

```
test_pred_gini <- predict(dtree_fit_gini, newdata = subtest)
confusionMatrix(test_pred_gini, subtest$Survived )  #check accura
cy
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 152  33
##          1  12  69
##
##                Accuracy : 0.8308
##                  95% CI : (0.7803, 0.8738)
##     No Information Rate : 0.6165
##     P-Value [Acc > NIR] : 2.211e-14
##
##                   Kappa : 0.6277
##
##  Mcnemar's Test P-Value : 0.002869
##
##             Sensitivity : 0.9268
##             Specificity : 0.6765
##          Pos Pred Value : 0.8216
##          Neg Pred Value : 0.8519
##              Prevalence : 0.6165
##          Detection Rate : 0.5714
##    Detection Prevalence : 0.6955
##       Balanced Accuracy : 0.8016
##
##        'Positive' Class : 0
##
```

La precissió és de 0.78.
La representació de l'arbre és la següent:

```
prp(dtree_fit_gini$finalModel, box.palette = "Blues", tweak = 1.2
)
```



Gradient Boosting

Ara provarem un algorisme de classificació que és una combinació d'algorismes classificadors. El Gradient Boosting és una tècnica d Machine Learning per a problemes de regressió i classificació que produeix un model de predicció ensamblant un conjunt de models de predicció febles, tipicament arbres de decisió. La idea és anar seqüèncialment prioritzant els punts que els models cataloguen malament per a que el següent algorisme es centri en ells.

```
library(gbm)
```

```
## Loaded gbm 2.1.5
```

```
library(MASS)
```

Utilitzem Cross Validation per a repetir l'entrenament amb diferents subconjunts de dades i trobar així, la combinació de paràmetres més òptima:

```
fitControl <- trainControl(method = "repeatedcv", number = 10, re
peats = 3)
gbmFit1 <- train(Survived ~ ., data = subtrain, method = "gbm", t
rControl = fitControl,verbose = FALSE)
gbmFit1
```

```
## Stochastic Gradient Boosting
##
## 625 samples
##   5 predictor
##   2 classes: '0', '1'
##
## No pre-processing
## Resampling: Cross-Validated (10 fold, repeated 3 times)
## Summary of sample sizes: 562, 562, 563, 563, 562, 563, ...
## Resampling results across tuning parameters:
##
##   interaction.depth  n.trees  Accuracy   Kappa
##   1                   50      0.7957501  0.5625913
##   1                  100      0.8000000  0.5705130
##   1                  150      0.7994794  0.5690756
##   2                   50      0.8021676  0.5641022
##   2                  100      0.8064345  0.5752726
##   2                  150      0.8122717  0.5887473
##   3                   50      0.8048131  0.5702022
##   3                  100      0.8112221  0.5864092
##   3                  150      0.8128179  0.5897409
##
## Tuning parameter 'shrinkage' was held constant at a value of 0
.1
##
## Tuning parameter 'n.minobsinnode' was held constant at a value
of 10
## Accuracy was used to select the optimal model using the larges
t value.
## The final values used for the model were n.trees = 150, intera
ction.depth =
##  3, shrinkage = 0.1 and n.minobsinnode = 10.
```

Un cop tenim el model entrenat, executem les prediccions sobre les dades de prova:

```
test_pred_gbmFit1 <- predict(gbmFit1, newdata = subtest)
confusionMatrix(test_pred_gbmFit1, subtest$Survived )  #check acc
uracy
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 153  31
##          1  11  71
##
##                Accuracy : 0.8421
##                  95% CI : (0.7926, 0.8838)
##     No Information Rate : 0.6165
##     P-Value [Acc > NIR] : 6.804e-16
##
##                   Kappa : 0.6532
##
##  Mcnemar's Test P-Value : 0.00337
##
##             Sensitivity : 0.9329
##             Specificity : 0.6961
##          Pos Pred Value : 0.8315
##          Neg Pred Value : 0.8659
##              Prevalence : 0.6165
##          Detection Rate : 0.5752
##    Detection Prevalence : 0.6917
##       Balanced Accuracy : 0.8145
##
##        'Positive' Class : 0
##
```

La precisió ha pujat lleugerament fins a 0.82 respecte a l'arbre de decisió.

Finalment, provarem l'algorisme Extreme Gradient Boost:

Preparem les dades:
Separem el conjunt en training i testing

```
set.seed(100)  # For reproducibility

# Create index for testing and training data
inTrain <- createDataPartition(y = train$Survived, p = 0.8, list
= FALSE)

# subset power_plant data to training
training <- train[inTrain,]


# subset the rest to test
 testing <- train[-inTrain,]

# print(training)
# print(testing)
```

```
library(xgboost)
```

```
##
## Attaching package: 'xgboost'
```

```
## The following object is masked from 'package:dplyr':
##
##     slice
```

```
library(onehot)

training = subset(training,select=c("PassengerId","Sex", "Age", "
Pclass","Survived"))
testing = subset(testing,select=c("PassengerId","Sex", "Age", "Pc
lass","Survived"))

#print(training)
#print(testing)
```

Per a executar aquest algorisme les dades han de ser numèriques, per tant, hem de transformar les dades categòriques en una matriu de columnes per a cada valor possible del factor. Utilitzem la funció onehot:

```
y_train <- training$Survived
y_test <- testing$Survived

X_train_onehot <- onehot(subset(training,select=c("PassengerId","
Sex", "Pclass")))
X_test_onehot <- onehot(subset(testing,select=c("PassengerId","Se
x", "Pclass")))

X_train <- predict(X_train_onehot,training)
X_test <- predict(X_test_onehot,testing)
```

Un cop tenim les dades preparades les transformem en un format de matriu que necessita l'algorisme:

```
#X_train = xgb.DMatrix(as.matrix(X_train %>% select(-Survived)))

X_train = xgb.DMatrix(as.matrix(X_train))
X_test = xgb.DMatrix(as.matrix(X_test))
```

```
#X_train
#X_test
```

Utilitzem cross validation per a entrenar el model amb 5 subconjunts de dades :

```
xgb_trcontrol = trainControl(
  method = "cv",
  number = 5,
  allowParallel = TRUE,
  verboseIter = FALSE,
  returnData = FALSE
)
```

Configurem uns paràmetres concrets:

```
xgbGrid <- expand.grid(nrounds = c(100,200),
                       max_depth = c(10, 15, 20, 25),
                       colsample_bytree = seq(0.5, 0.9, length.ou
t = 5),
                       eta = 0.1,
                       gamma=0,
                       min_child_weight = 1,
                       subsample = 1
                      )
```

Entrenem el model amb aquests paràmetres:

```
set.seed(0)

xgb_model = train(
  X_train, y_train,
  trControl = xgb_trcontrol,
  tuneGrid = xgbGrid,
  method = "xgbTree"
)
```

Obtenim la combinació òptima de paràmetres:

```
xgb_model$bestTune
```

	nroun…	max_de…	…	ga…	colsample_bytree	min_child_weight
	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>
7	100	10	0.1	0	0.8	1

1 row

Executem la predicció sobre el conjunt de test:

```
predicted = predict(xgb_model, X_test)

predicted<-ifelse(predicted=='0',0,1)
y_test<-ifelse(y_test=='0',0,1)

resultat = as.data.frame(cbind(predicted = predicted,
                               observed = y_test))
print(resultat)
```

```
##     predicted observed
## 1           0        0
## 2           0        0
## 3           0        0
## 4           1        1
## 5           0        0
## 6           0        1
## 7           1        0
## 8           0        1
## 9           1        1
## 10          1        1
## 11          0        0
## 12          0        1
## 13          0        0
## 14          1        1
## 15          0        1
## 16          0        0
## 17          1        0
## 18          0        0
## 19          0        0
## 20          1        0
## 21          0        1
## 22          1        1
## 23          0        0
## 24          0        0
## 25          1        1
## 26          0        0
## 27          0        0
## 28          0        0
## 29          0        1
## 30          0        1
## 31          0        0
## 32          1        1
## 33          0        1
## 34          0        1
## 35          0        1
## 36          0        0
## 37          0        0
## 38          0        0
## 39          0        1
## 40          1        1
## 41          1        0
## 42          1        0
## 43          0        0
## 44          0        1
```

```
## 45           0         0
## 46           0         0
## 47           1         1
## 48           0         1
## 49           0         0
## 50           0         0
## 51           0         0
## 52           0         0
## 53           0         0
## 54           0         0
## 55           0         1
## 56           0         0
## 57           0         0
## 58           0         1
## 59           1         1
## 60           1         0
## 61           1         1
## 62           0         0
## 63           0         1
## 64           0         0
## 65           0         0
## 66           0         1
## 67           0         0
## 68           0         0
## 69           1         1
## 70           0         0
## 71           0         0
## 72           0         0
## 73           1         1
## 74           1         0
## 75           1         1
## 76           0         0
## 77           1         0
## 78           0         0
## 79           1         1
## 80           0         0
## 81           1         1
## 82           0         0
## 83           1         1
## 84           1         1
## 85           0         0
## 86           0         0
## 87           0         0
## 88           1         1
## 89           0         1
```

```
## 90           0        1
## 91           0        0
## 92           0        0
## 93           1        1
## 94           1        1
## 95           0        0
## 96           0        0
## 97           1        0
## 98           0        0
## 99           0        0
## 100          0        1
## 101          0        0
## 102          0        0
## 103          0        0
## 104          0        0
## 105          0        1
## 106          1        1
## 107          0        0
## 108          1        1
## 109          0        0
## 110          0        1
## 111          0        0
## 112          0        0
## 113          0        1
## 114          0        0
## 115          1        1
## 116          0        0
## 117          1        1
## 118          0        0
## 119          0        1
## 120          0        0
## 121          0        0
## 122          0        0
## 123          0        0
## 124          0        0
## 125          0        1
## 126          0        0
## 127          1        0
## 128          0        0
## 129          1        1
## 130          0        0
## 131          0        1
## 132          0        0
## 133          0        0
## 134          0        1
```

```
## 135          1          0
## 136          0          0
## 137          1          1
## 138          1          0
## 139          0          0
## 140          0          0
## 141          0          1
## 142          0          0
## 143          0          0
## 144          0          0
## 145          0          1
## 146          0          0
## 147          0          0
## 148          0          0
## 149          0          1
## 150          0          0
## 151          0          0
## 152          0          1
## 153          0          0
## 154          0          1
## 155          0          0
## 156          0          1
## 157          0          0
## 158          0          1
## 159          0          1
## 160          1          0
## 161          0          1
## 162          1          1
## 163          1          1
## 164          0          0
## 165          0          0
## 166          0          0
## 167          0          0
## 168          0          0
## 169          0          0
## 170          0          1
## 171          0          0
## 172          1          1
## 173          0          0
## 174          0          0
## 175          1          0
## 176          1          1
## 177          1          0
```

```
confusionMatrix (xgb_model)
```

```
## Cross-Validated (5 fold) Confusion Matrix
##
## (entries are percentual average cell counts across resamples)
##
##            Reference
## Prediction    0    1
##          0 55.6 13.7
##          1  6.0 24.6
##
##  Accuracy (average) : 0.8025
```

Obtenim una precissió de 80%. A partir d'aquí caldria jugar amb els paràmetres per a millorar si es pot els resultats.

Conclusions generals.

L'objectiu d'aquest dataset és obtenir un model capaç de predir amb la màxima precisió la supervivència d'un passatger en funció d'una sèrie de dades que tenim sobre cadascun.
Hem fet models de regressió simple i múltiple per veure la influència de cada variable. Donat el tipus de dades, hem fet un model logístic. Els resultats obtinguts ens diuen que la variable del sexe és la més determinant a l'hora d'establir la supervivència (els homes tenen menys probabilitats de sobreviure), seguida de la classe social (reflexada en la categoria del bitllet) (com més baixa menys possibilitats) i de l'edat (en molt baix percentatge).

Després hem provat algorismes de classificació. L'algorisme de l'arbre de decisió ens donava una precisió a l'hora de fer la predicció d'un 78%-80% en funció dels paràmetres. La combinació d'algorismes de classificació en la forma del Gradient Boosting fa pujar la precisió lleugerament fins a 82%. Finalment provem l'algorisme de combinació d'algorismes més emprat actualment, el Extreme Gradient Boosting, una variant del Gradient Boosting amb capacitat per provar diferents combinacions de paràmetres.

Contribuciones	Firma
Investigació prèvia	Joan Carles Badia Purroy
Redacció de les respostes	Joan Carles Badia Purroy
Desenvolupament codi	Joan Carles Badia Purroy