Who survived on the Titanic

Mathieu Klimczak

16/06/2019

Introduction

The goal here will be to predict whether or not passenger of the titanic will survive, thus a classification task.

This data set is a classical one available on Kaggle, that you can find here at the following adress.

https://www.kaggle.com/c/titanic/data

Since the test set given by Kaggle obviously don't have the result, we'll use the train set and split it into train set and validation set to perform our ML task.

Let's first import the train set, see what it looks like and transform the class of some predictor for later use.

```
df <- read_csv("C:/Users/Mathieu/Documents/Dropbox/DataAnalysis/R/titanic/datas/train.csv")
```

```
## Parsed with column specification:
## cols(
##   PassengerId = col_double(),
##   Survived = col_double(),
##   Pclass = col_double(),
##   Name = col_character(),
##   Sex = col_character(),
##   Age = col_double(),
##   SibSp = col_double(),
##   Parch = col_double(),
##   Ticket = col_character(),
##   Fare = col_double(),
##   Cabin = col_character(),
##   Embarked = col_character()
## )
```

```
df <- df %>% mutate(Survived = as.factor(Survived), Pclass = as.factor(Pclass),
    Embarked = as.factor(Embarked), Sex = as.factor(Sex))
y <- df$Survived
```

The data dictionnary is the following one.

- Survival : 0 = No, 1 = Yes
- Pclass : Ticket class 1 = 1st, 2 = 2nd, 3 = 3rd
- sex : Sex
- Age : Age in years
- sibsp : # of siblings / spouses aboard the Titanic
- parch : # of parents / children aboard the Titanic
- ticket : Ticket number
- fare : Passenger fare
- cabin : Cabin number
- embarked : Port of Embarkation C = Cherbourg, Q = Queenstown, S = Southampton

Exploratory Data Analysis, new predictors

Structure and missing values

Let's check the global structure of the dataset.

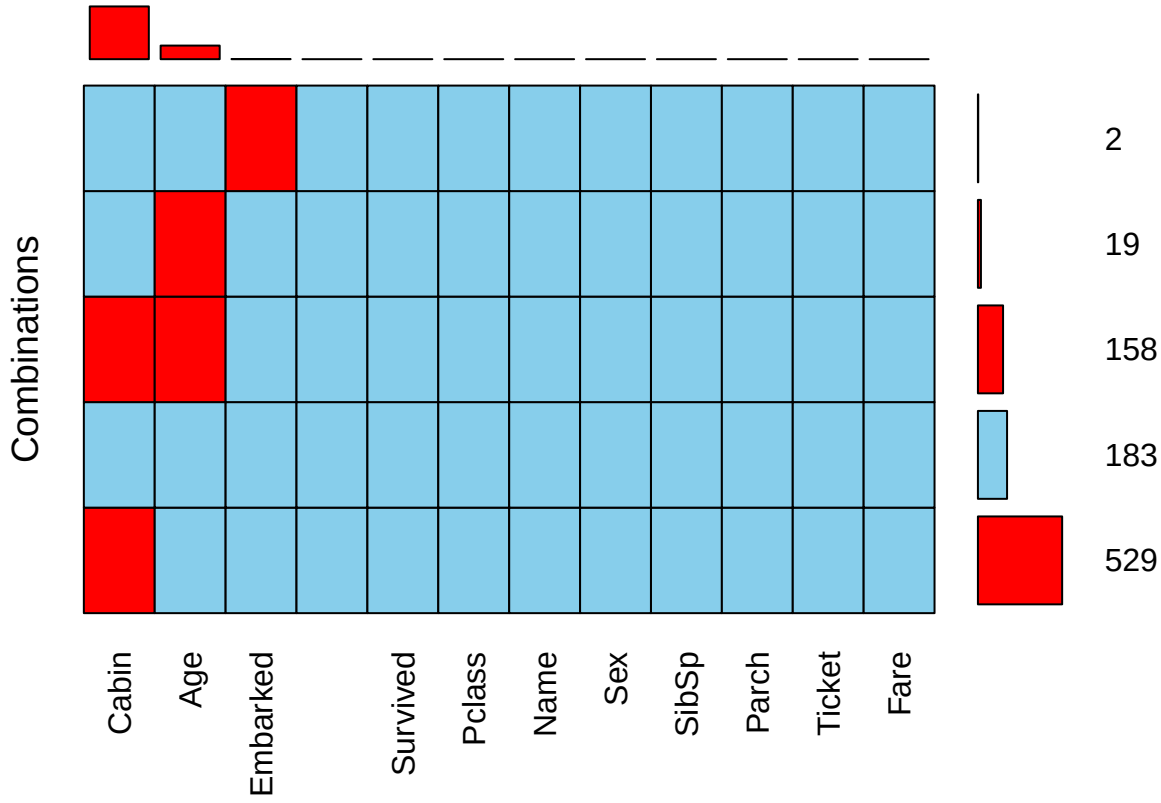
```
summary(df)
```

```
##   PassengerId    Survived Pclass      Name              Sex
##  Min.   :  1.0   0:549    1:216   Length:891         female:314
##  1st Qu.:223.5   1:342    2:184   Class :character   male  :577
##  Median :446.0            3:491   Mode  :character
##  Mean   :446.0
##  3rd Qu.:668.5
##  Max.   :891.0
##
##       Age            SibSp           Parch           Ticket
##  Min.   : 0.42   Min.   :0.000   Min.   :0.0000   Length:891
##  1st Qu.:20.12   1st Qu.:0.000   1st Qu.:0.0000   Class :character
##  Median :28.00   Median :0.000   Median :0.0000   Mode  :character
##  Mean   :29.70   Mean   :0.523   Mean   :0.3816
##  3rd Qu.:38.00   3rd Qu.:1.000   3rd Qu.:0.0000
##  Max.   :80.00   Max.   :8.000   Max.   :6.0000
##  NA's   :177
##       Fare           Cabin           Embarked
##  Min.   :  0.00   Length:891         C   :168
##  1st Qu.:  7.91   Class :character   Q   : 77
##  Median : 14.45   Mode  :character   S   :644
##  Mean   : 32.20                      NA's:  2
##  3rd Qu.: 31.00
##  Max.   :512.33
##
```

```
glimpse(df)
```

```
## Observations: 891
## Variables: 12
## $ PassengerId <dbl> 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15,...
## $ Survived    <fct> 0, 1, 1, 1, 0, 0, 0, 0, 1, 1, 1, 1, 0, 0, 0, 1, 0,...
## $ Pclass      <fct> 3, 1, 3, 1, 3, 3, 1, 3, 3, 2, 3, 1, 3, 3, 3, 2, 3,...
## $ Name        <chr> "Braund, Mr. Owen Harris", "Cumings, Mrs. John Bra...
## $ Sex         <fct> male, female, female, female, male, male, male, ma...
## $ Age         <dbl> 22, 38, 26, 35, 35, NA, 54, 2, 27, 14, 4, 58, 20, ...
## $ SibSp       <dbl> 1, 1, 0, 1, 0, 0, 0, 3, 0, 1, 1, 0, 0, 1, 0, 0, 4,...
## $ Parch       <dbl> 0, 0, 0, 0, 0, 0, 0, 1, 2, 0, 1, 0, 0, 5, 0, 0, 1,...
## $ Ticket      <chr> "A/5 21171", "PC 17599", "STON/O2. 3101282", "1138...
## $ Fare        <dbl> 7.2500, 71.2833, 7.9250, 53.1000, 8.0500, 8.4583, ...
## $ Cabin       <chr> NA, "C85", NA, "C123", NA, NA, "E46", NA, NA, NA, ...
## $ Embarked    <fct> S, C, S, S, S, Q, S, S, S, C, S, S, S, S, S, Q,...
```

```
aggr(df, prop = FALSE, combined = TRUE, numbers = TRUE, sortVars = TRUE, sortCombs = TRUE)
```




```
##
##  Variables sorted by number of missings:
##      Variable Count
##         Cabin   687
##           Age   177
##      Embarked     2
##   PassengerId     0
##      Survived     0
##        Pclass     0
##          Name     0
##           Sex     0
##         SibSp     0
##         Parch     0
##        Ticket     0
##          Fare     0
```

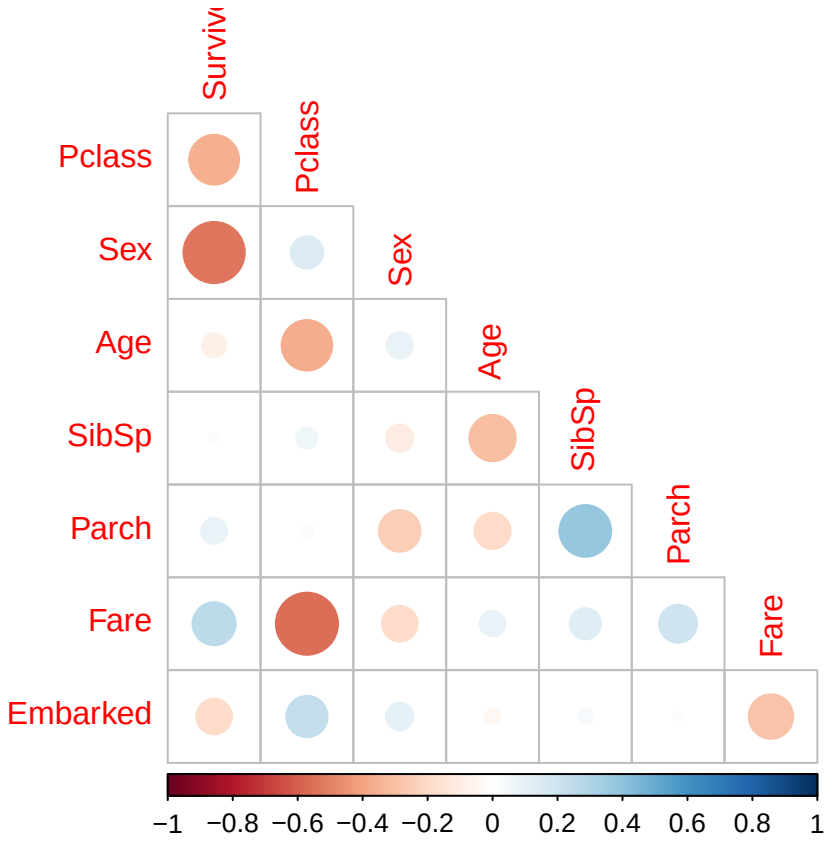
For missing values we use the VIM library, each row is a combination of predictors with missing values. There are 529 observations where the only missing value is the cabin, 158 obsevations where it is the cabin and the age, 19 obsevartions where only the age is missing and finally, 2 observations where the place of embarkation is missing.

We have the following correlations for the predictors. Some of the interesting corelations are Pclass/Survived, Sex/Survived, Age/Pclass, Fare/Pclass

```
df %>%
  select(-PassengerId, -Name, -Cabin, -Ticket) %>%
  mutate(Sex = fct_recode(Sex,
                          "0" = "male",
```

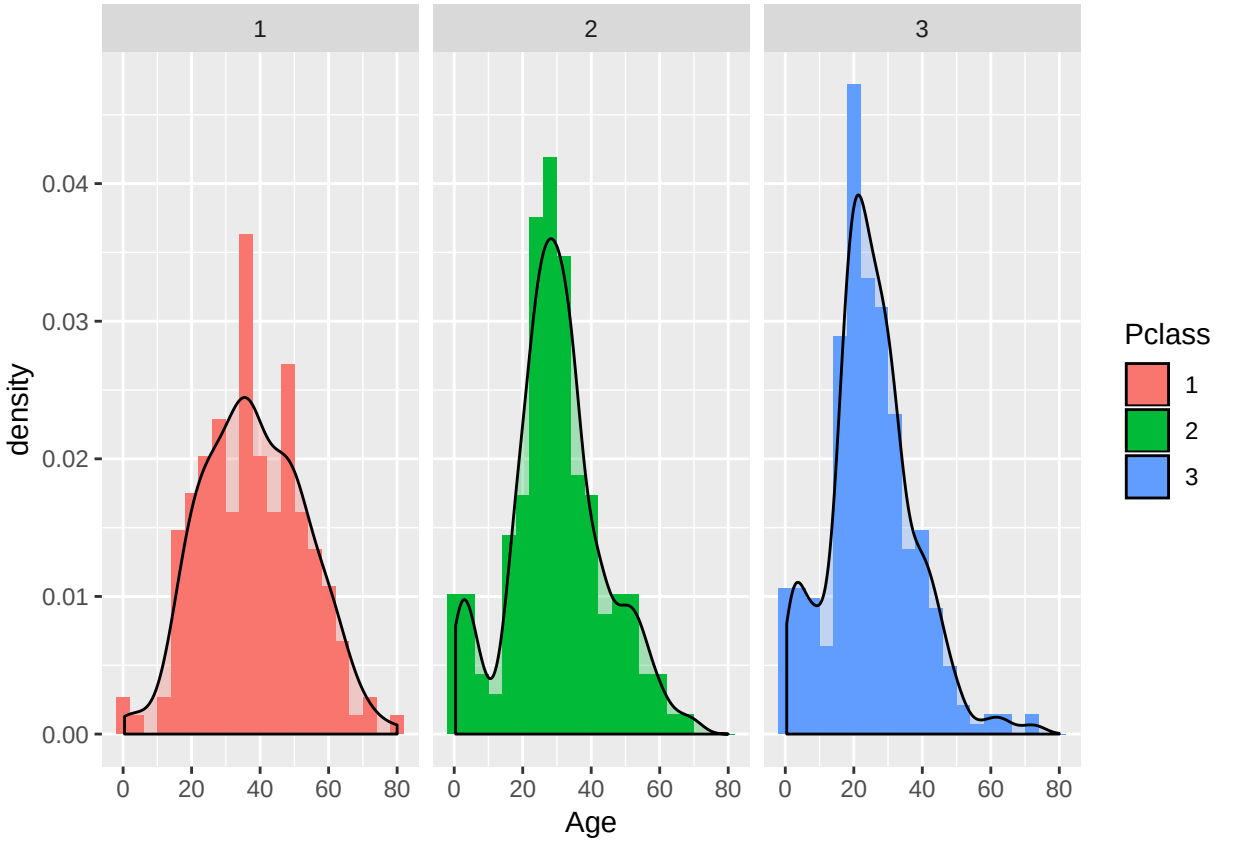
```
                          "1" = "female")) %>%

  mutate(
    Sex = as.integer(Sex),
    Pclass = as.integer(Pclass),
    Survived = as.integer(Survived),
    Embarked = as.integer(Embarked)
  ) %>%
  cor(use = "complete.obs", method = "pearson") %>%
  corrplot(type = "lower", diag = FALSE)
```



the Age

```
df %>% ggplot(aes(x = Age, y = ..density.., fill = Pclass)) + geom_histogram(binwidth = 4) +
    geom_density(alpha = 0.3) + facet_grid(. ~ Pclass)
```




```
df %>% group_by(Pclass) %>% summarize(mean(Survived == 1))
```

```
## # A tibble: 3 x 2
##   Pclass `mean(Survived == 1)`
##   <fct>                  <dbl>
## 1 1                      0.630
## 2 2                      0.473
## 3 3                      0.242
```

```
df %>% group_by(Sex) %>% summarize(mean(Survived == 1))
```

```
## # A tibble: 2 x 2
##   Sex    `mean(Survived == 1)`
##   <fct>                  <dbl>
## 1 female                 0.742
## 2 male                   0.189
```

Although the ship is mostly filled with adults around 30-40 years old, young adults and children seem more common in the 2nd and 3rd class, whereas older people, ie $>$ 40 years old are more common in the first class

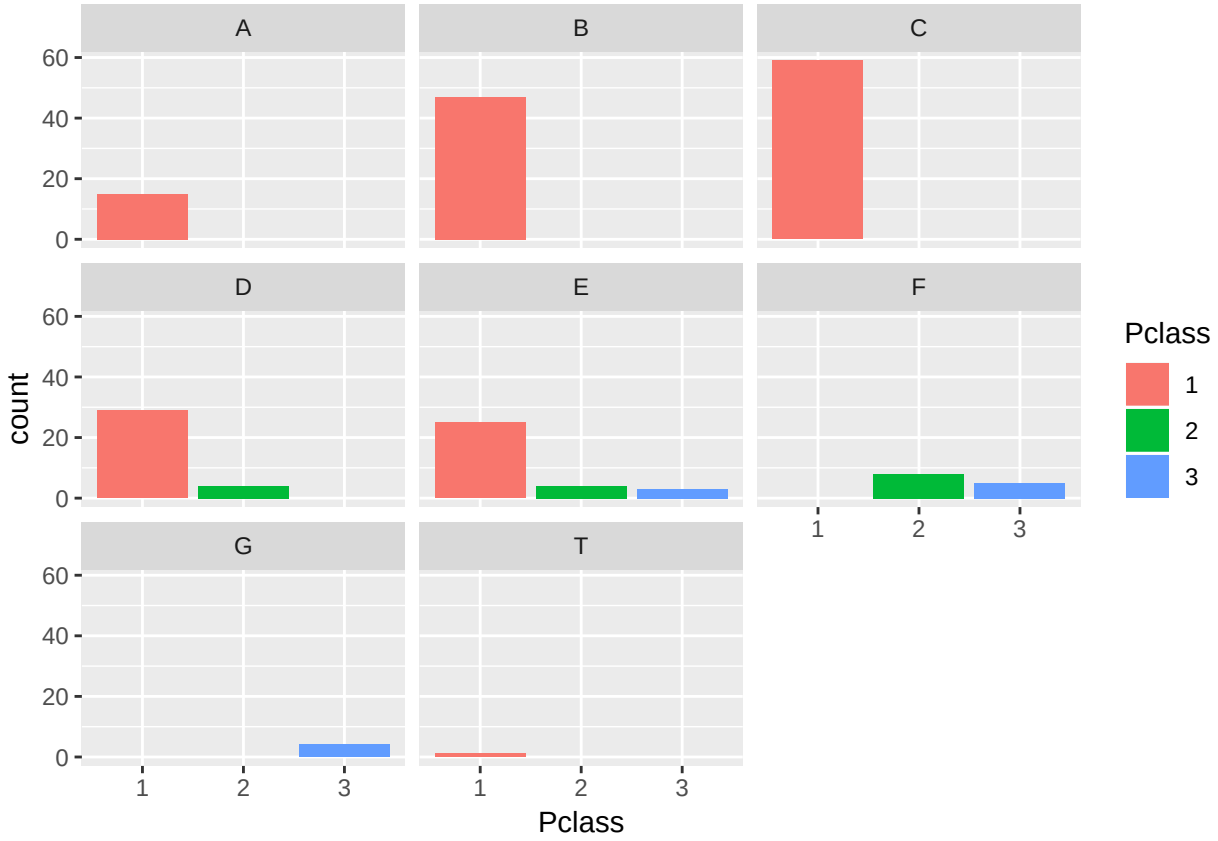
The class with the highest rate of survival if the first one. In the third class, nearly 75% died. Nearly 75% of the women survived

the cabin

We now look at the cabin, the first letter corresponds to the deck. Let's see what we can deduce from it

```
df <- df %>% mutate(Deck = substring(Cabin, 1, 1))
df %>% filter(!is.na(Deck)) %>% ggplot(aes(x = Pclass, fill = Pclass)) + geom_bar() +
```

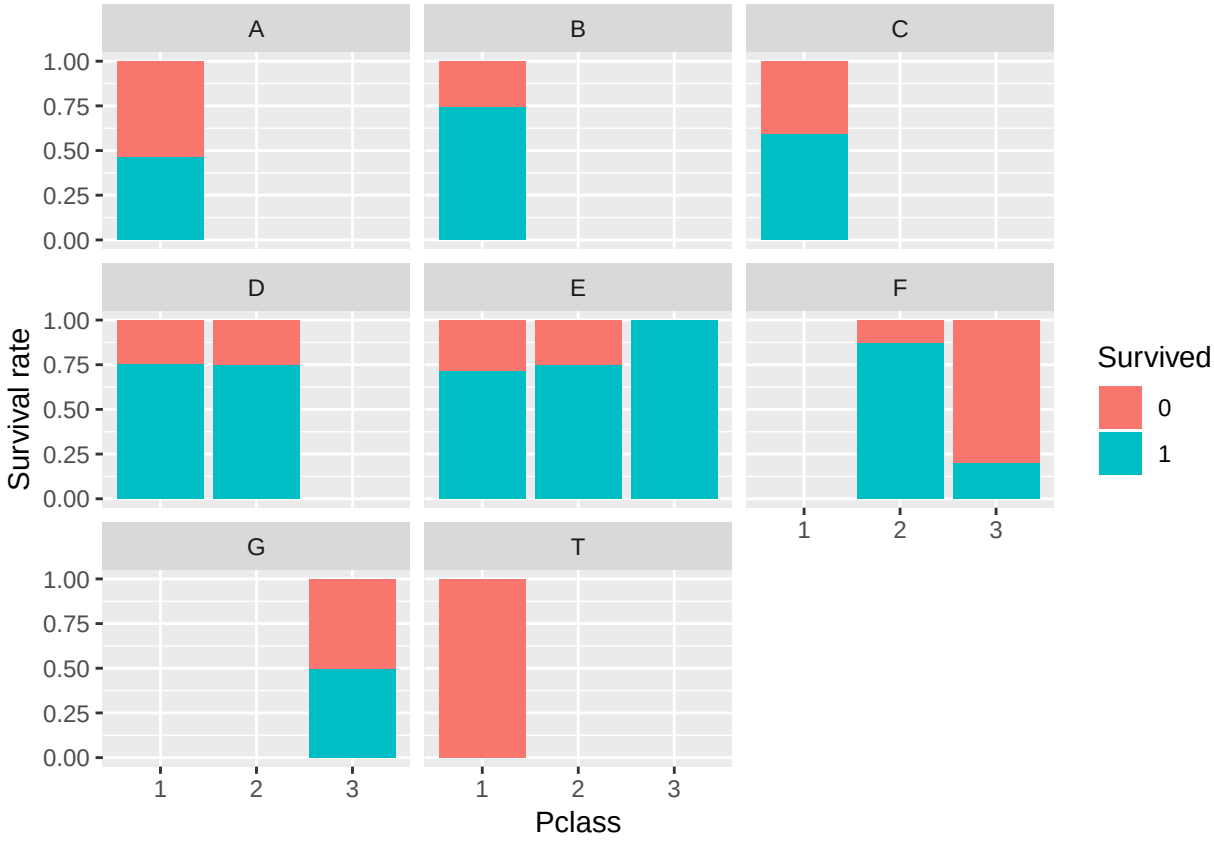
```
    facet_wrap(. ~ Deck)
```




```
df %>% filter(!is.na(Deck)) %>% group_by(Deck) %>% summarize(Surv_rate = mean(Survived ==
    1)) %>% arrange(desc(Surv_rate))
```

```
## # A tibble: 8 x 2
##   Deck  Surv_rate
##   <chr>     <dbl>
## 1 D         0.758
## 2 E         0.75
## 3 B         0.745
## 4 F         0.615
## 5 C         0.593
## 6 G         0.5
## 7 A         0.467
## 8 T         0
```

```
df %>% filter(!is.na(Deck)) %>% ggplot(aes(x = Pclass, fill = Survived)) + geom_bar(position = "fill") +
    facet_wrap(. ~ Deck) + ylab("Survival rate")
```



Decks A,B,C, and T are exclusively filled with 1st class, G with 3rd class, 2nd class is only found on decks D,E,F. Everybody died on Deck T.

The name

The full name won't be usefull as it is, however, we can extract the title of the passenger. Master accounts for young children, apart from female and male title, one can also noticed some "rare titles" like "Sir", "Don" etc.

```
df <- df %>% mutate(Title = str_extract(Name, "[A-Z][a-z]*\\.")) %>% mutate(Title = str_replace_all(Tit
    "\\.", "")) %>% mutate(Title = str_trim(Title))
table(df$Title, df$Sex)
```

```
##
##             female male
##   Capt           0    1
##   Col            0    2
##   Countess       1    0
##   Don            0    1
##   Dr             1    6
##   Jonkheer       0    1
##   Lady           1    0
##   Major          0    2
##   Master         0   40
##   Miss         182    0
##   Mlle           2    0
##   Mme            1    0
##   Mr             0  517
```

```
##   Mrs         125    0
##   Ms            1    0
##   Rev           0    6
##   Sir           0    1
```

```
df %>% group_by(Title) %>% summarize(rate = mean(Survived == 1)) %>% arrange(desc(rate))
```

```
## # A tibble: 17 x 2
##    Title     rate
##    <chr>    <dbl>
##  1 Countess 1
##  2 Lady     1
##  3 Mlle     1
##  4 Mme      1
##  5 Ms       1
##  6 Sir      1
##  7 Mrs      0.792
##  8 Miss     0.698
##  9 Master   0.575
## 10 Col      0.5
## 11 Major    0.5
## 12 Dr       0.429
## 13 Mr       0.157
## 14 Capt     0
## 15 Don      0
## 16 Jonkheer 0
## 17 Rev      0
```

```
df <- df %>% mutate(Title = factor(Title)) %>% mutate(Title = fct_collapse(Title,
    Miss = c("Mlle", "Ms"), Mrs = "Mme", Ranked = c("Major", "Dr", "Capt", "Col",
        "Rev"), Royalty = c("Lady", "Countess", "Don", "Sir", "Jonkheer")))
df %>% group_by(Title) %>% summarize(rate = mean(Survived == 1)) %>% arrange((desc(rate)))
```

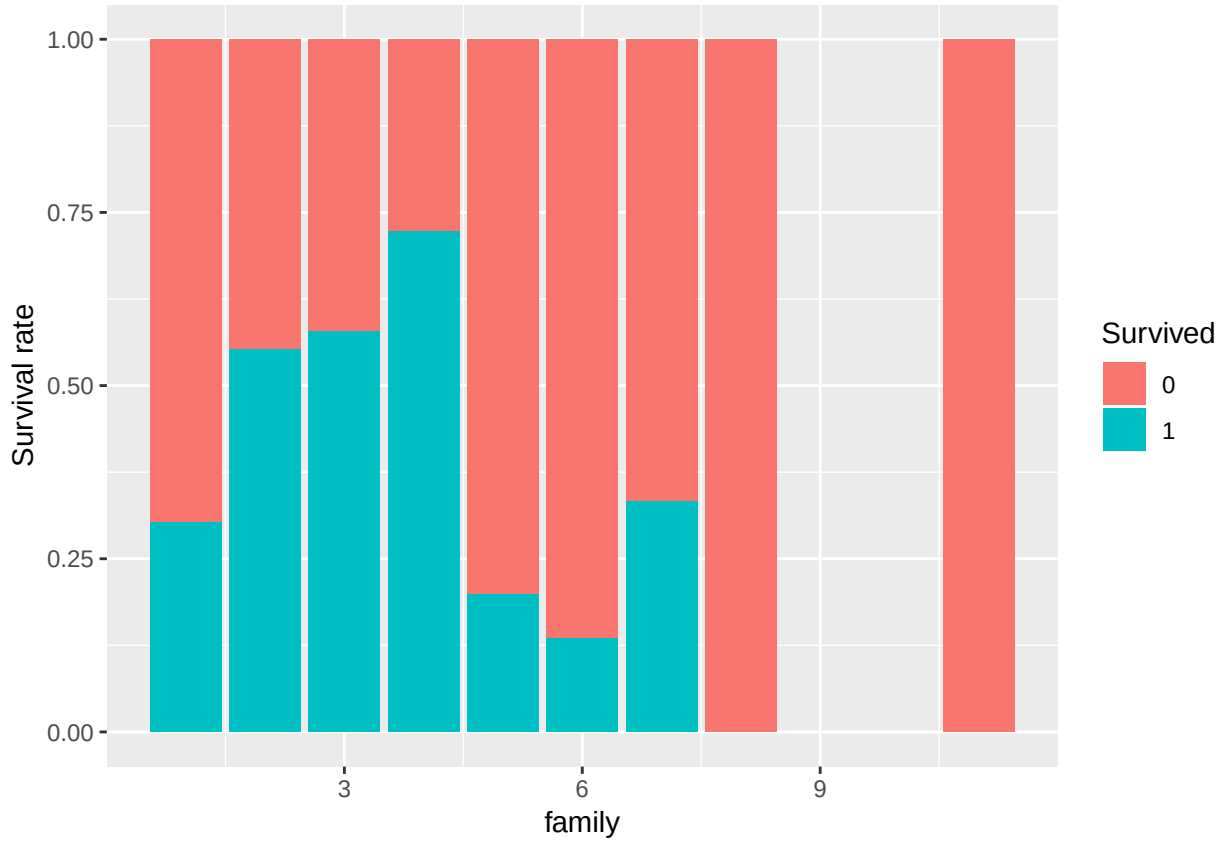
```
## # A tibble: 6 x 2
##   Title    rate
##   <fct>   <dbl>
## 1 Mrs     0.794
## 2 Miss    0.703
## 3 Royalty 0.6
## 4 Master  0.575
## 5 Ranked  0.278
## 6 Mr      0.157
```

Grouping titles as we have done, we have a better distinction better people.

The family : Parch + SibSp

Small families, ie between 2-4 people, were much more likely to survive as we can see in the following plot. Let's create a predictor for that

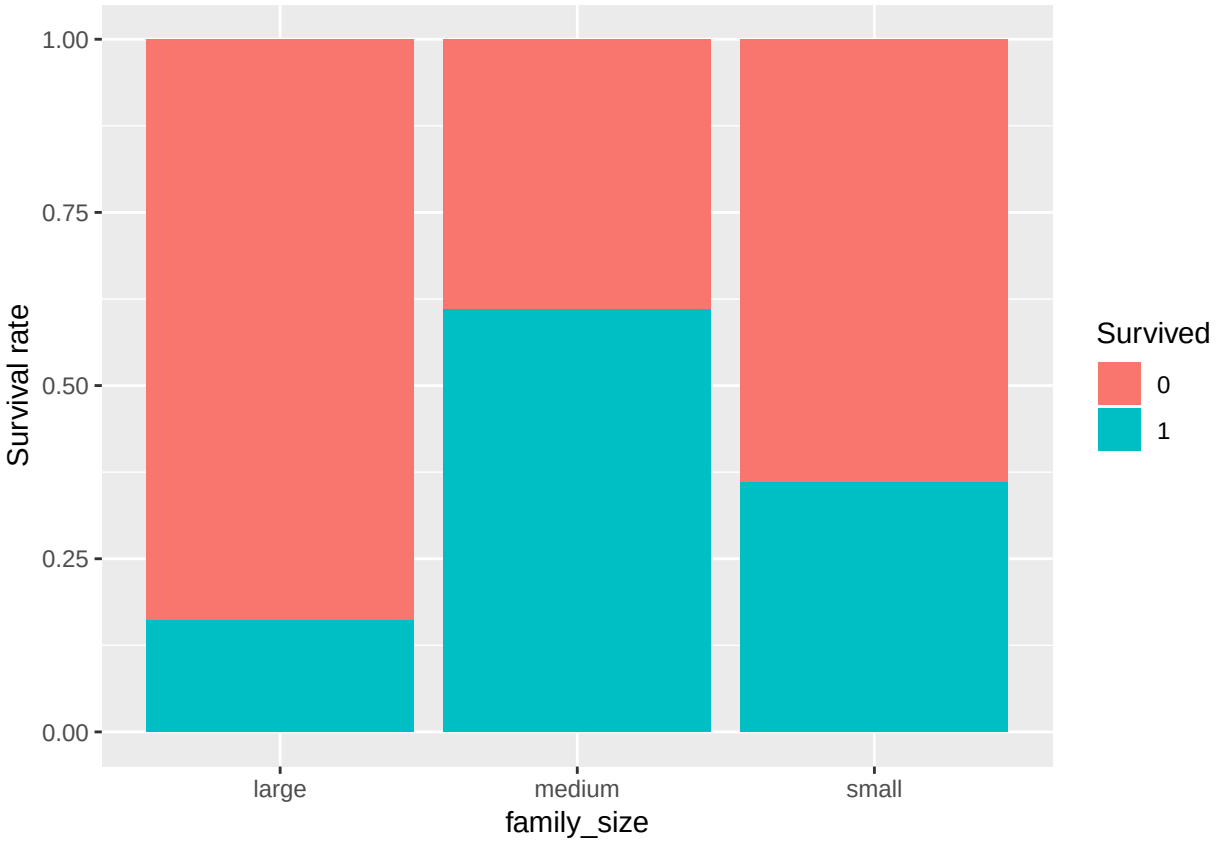
```
df <- df %>% mutate(family = Parch + SibSp + 1)
df %>% ggplot(aes(x = family, fill = Survived)) + geom_bar(position = "fill") +
    ylab("Survival rate")
```




```
df %>% group_by(family) %>% summarize(rate = mean(Survived == 1)) %>% arrange(desc(rate))
```

```
## # A tibble: 9 x 2
##   family  rate
##    <dbl> <dbl>
## 1      4 0.724
## 2      3 0.578
## 3      2 0.553
## 4      7 0.333
## 5      1 0.304
## 6      5 0.2
## 7      6 0.136
## 8      8 0
## 9     11 0
```

```
df <- df %>% mutate(family_size = factor(ifelse(family > 4, "large", ifelse(2 <
    family, "medium", "small"))))
df %>% ggplot(aes(x = family_size, fill = Survived)) + geom_bar(position = "fill") +
    ylab("Survival rate")
```

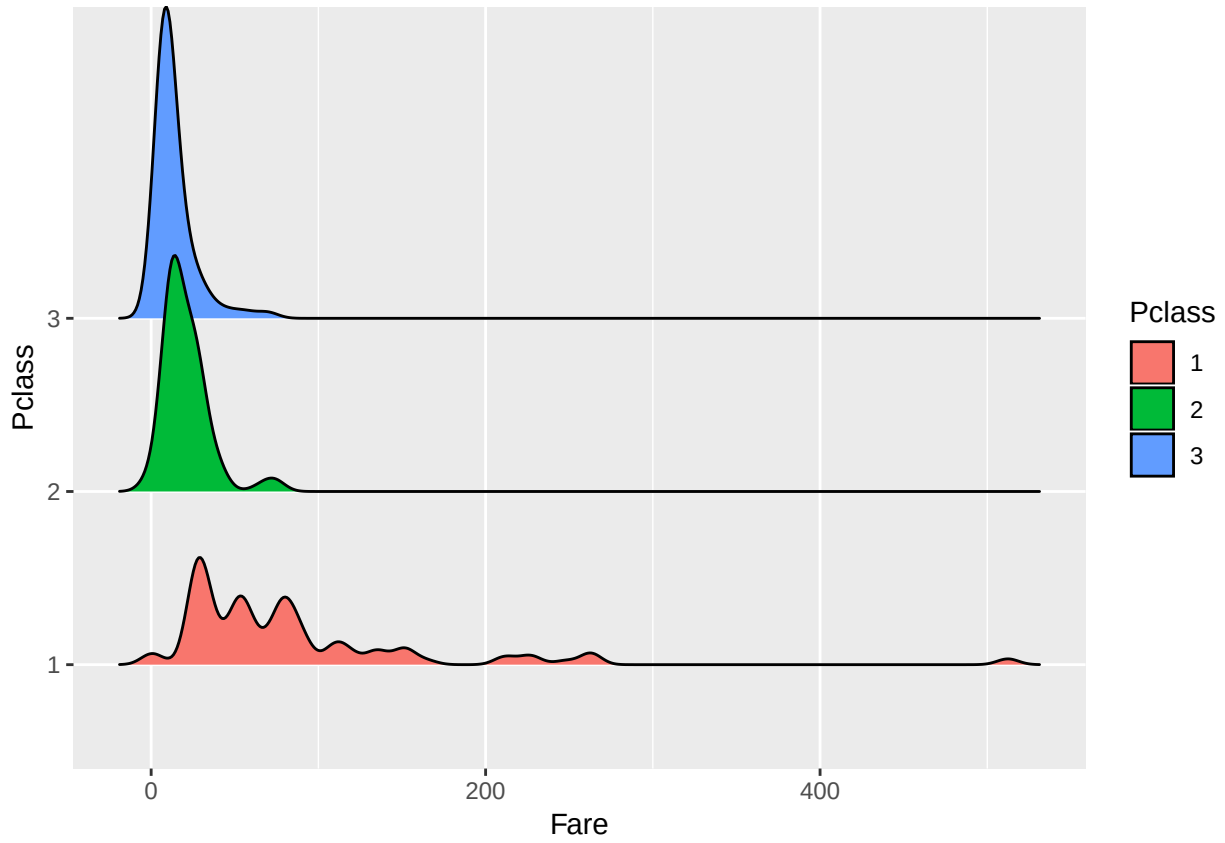


The fare

As it is, the Fare doesn't seem to be usefull. The strange thing here is that the class here overlap a lot, which shouldn't be. The price of the travel in the 3rd class shouldn't overlap as much the price of the 1rst class, which is the most expensive one.

```
df %>% ggplot(aes(x = Fare, y = Pclass, fill = Pclass)) + geom_density_ridges()
```

```
## Picking joint bandwidth of 6.31
```

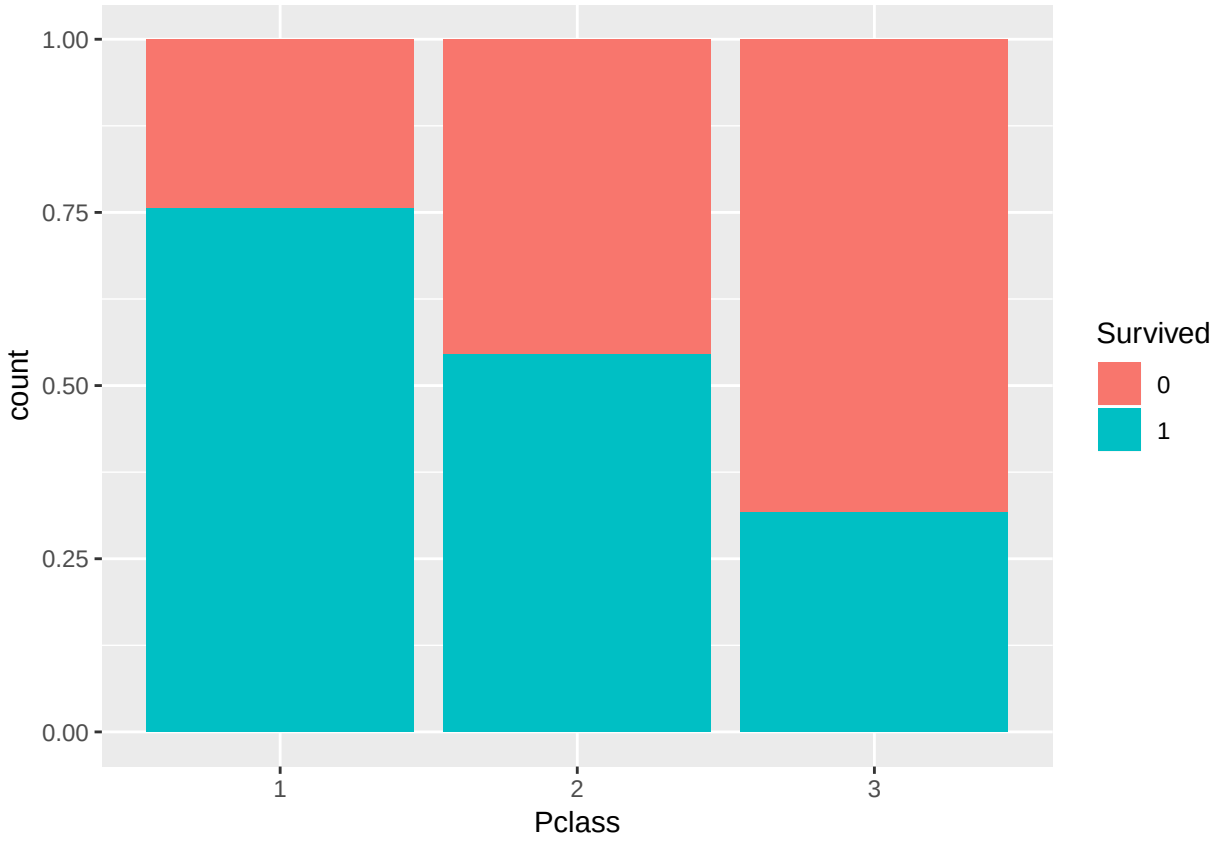



```
df %>% group_by(Ticket, Pclass, Fare) %>% summarize(n = n()) %>% arrange(desc(n))
```

```
## # A tibble: 682 x 4
## # Groups:   Ticket, Pclass [681]
##    Ticket       Pclass  Fare     n
##    <chr>        <fct>  <dbl> <int>
##  1 1601         3       56.5     7
##  2 347082       3       31.3     7
##  3 CA. 2343     3       69.6     7
##  4 3101295      3       39.7     6
##  5 347088       3       27.9     6
##  6 CA 2144      3       46.9     6
##  7 382652       3       29.1     5
##  8 S.O.C. 14879 2       73.5     5
##  9 113760       1      120       4
## 10 113781       1      152.      4
## # ... with 672 more rows
```

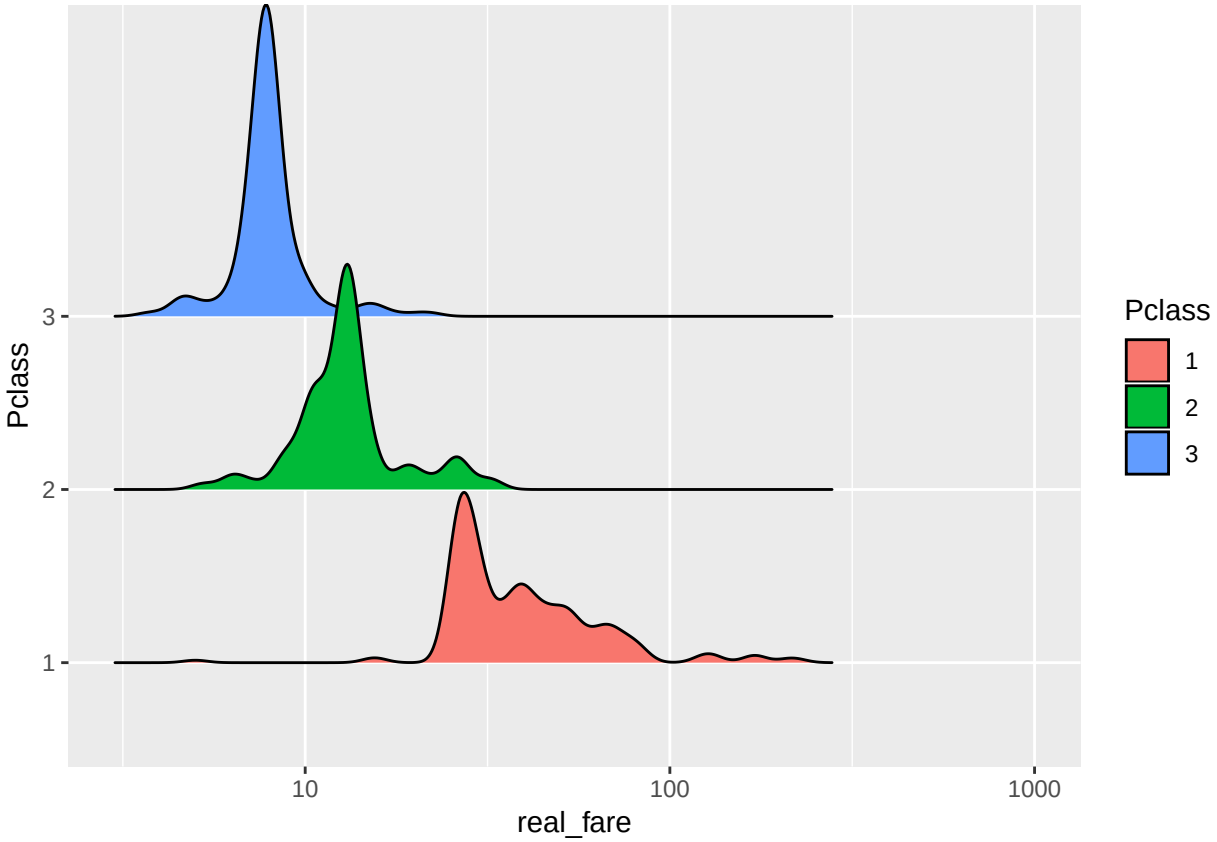
As we can see here, the Fare predictor does not seem to correspond to the price of one person, but to the whole price for a group, i.e. the ticket 1601 seems to account for 7 people traveling together. The real price should then be given by the following code.

```
df <- df %>% group_by(Ticket) %>% mutate(real_fare = Fare/n(), shared_ticket = ifelse(n() >
    1, 1, 0)) %>% ungroup()
df %>% filter(shared_ticket == 1) %>% ggplot(aes(Pclass, fill = Survived)) +
    geom_bar(position = "fill")
```




```
df %>% filter(real_fare > 0) %>% ggplot(aes(real_fare, Pclass, fill = Pclass)) +
    geom_density_ridges() + scale_x_log10(lim = c(3, 1000)) + labs(x = "real_fare")
```

```
## Picking joint bandwidth of 0.033
```

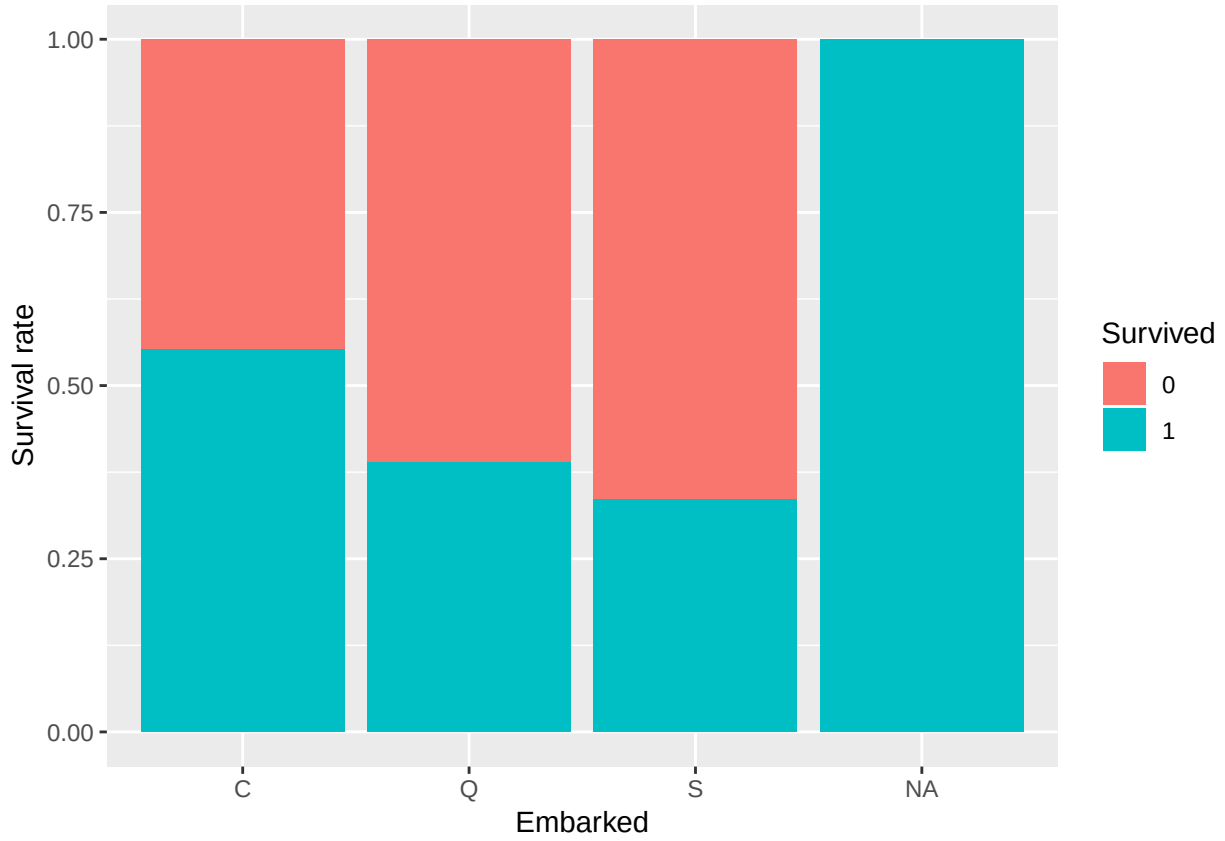


We now have a much clearer separation between the different classes.

Port of embarkation

Does the port of embarkation has an influence on the rate of survival ?

```
df %>% ggplot(aes(x = Embarked, fill = Survived)) + geom_bar(position = "fill") +
    ylab("Survival rate")
```




```
df %>% filter(!is.na(Embarked)) %>% group_by(Embarked) %>% summarize(n = n(),
    Survival_rate = mean(Survived == 1) * 100)
```

```
## # A tibble: 3 x 3
##   Embarked     n Survival_rate
##   <fct>    <int>         <dbl>
## 1 C          168          55.4
## 2 Q           77          39.0
## 3 S          644          33.7
```

```
df %>% filter(!is.na(Embarked)) %>% group_by(Embarked, Pclass) %>% summarize(n = n(),
    Survival_rate = mean(Survived == 1) * 100)
```

```
## # A tibble: 9 x 4
## # Groups:   Embarked [3]
##   Embarked Pclass     n Survival_rate
##   <fct>    <fct>  <int>         <dbl>
## 1 C        1         85          69.4
## 2 C        2         17          52.9
## 3 C        3         66          37.9
## 4 Q        1          2          50
## 5 Q        2          3          66.7
## 6 Q        3         72          37.5
## 7 S        1        127          58.3
## 8 S        2        164          46.3
## 9 S        3        353          19.0
```

Clearly, the port of embarkation seems to have an influence on the survival rate.

Preprocessing

Missing port of embarkation values

Only two people have a missing port of embarkation, they have the same ticket number, so they traveled together and thus have the same port of embarkation. They are from the first class and are located on deck. We'll use the the most frequent port of embarkation under these criterions to fill in the blanks.

```
df %>% filter(is.na(Embarked))
```

```
## # A tibble: 2 x 18
##   PassengerId Survived Pclass Name  Sex     Age SibSp Parch Ticket  Fare
##         <dbl> <fct>    <fct>  <chr> <fct> <dbl> <dbl> <dbl> <chr>  <dbl>
## 1          62 1        1      Icar~ fema~    38     0     0 113572    80
## 2         830 1        1      Ston~ fema~    62     0     0 113572    80
## # ... with 8 more variables: Cabin <chr>, Embarked <fct>, Deck <chr>,
## #   Title <fct>, family <dbl>, family_size <fct>, real_fare <dbl>,
## #   shared_ticket <dbl>
```

```
df %>% filter(!is.na(Embarked) & Pclass == 1 & Deck == 'B') %>%
  group_by(Embarked) %>%
  summarize(n = n())
```

```
## # A tibble: 2 x 2
##   Embarked     n
##   <fct>    <int>
## 1 C           22
## 2 S           23
```

```
df <- df %>%
  mutate(Embarked = as.character(Embarked)) %>%
  mutate(Embarked = case_when(is.na(Embarked) ~ "S",
                              TRUE ~ Embarked)) %>%
  mutate(Embarked = as.factor(Embarked))
```

Missing decks values

Concerning the missing deck values, it seems we won't be predict anything, especially for the third class, where almost all of them have a missing value. What we can do however is create a new predictor on whether or not the deck is known.

```
df %>%
  group_by(Pclass) %>%
  summarize(
    n = n(),
    missing = sum(is.na(Deck)),
    ratio_missing_deck = sum(is.na(Deck)) * 100 / n()
  )
```

```
## # A tibble: 3 x 4
##   Pclass     n missing ratio_missing_deck
##   <fct>  <int>   <int>              <dbl>
## 1 1        216      40               18.5
## 2 2        184     168               91.3
## 3 3        491     479               97.6
```

```
df <- df %>% mutate(known_deck = ifelse(!is.na(Deck), 1, 0))
```

Missing Age values

Here, we'll the preProcess function of the Caret package, which allows imputations of missing values. First we drop the predictors we won't use and transfrom into factors the newly created predictors.

```
df <-
  select(df,
         -c(Cabin,
            PassengerId,
            Name,
            Ticket,
            Fare,
            Deck,
            SibSp,
            Parch,
            family))

df <- df %>% mutate(
  Sex = factor(Sex),
  Pclass = factor(Pclass),
  Survived = factor(Survived),
  Embarked = factor(Embarked),
  shared_ticket = factor(shared_ticket),
  known_deck = factor(known_deck)
)



str(df)
```

```
## Classes 'tbl_df', 'tbl' and 'data.frame':    891 obs. of  10 variables:
##  $ 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 NA 54 2 27 14 ...
##  $ Embarked     : Factor w/ 3 levels "C","Q","S": 3 1 3 3 3 2 3 3 3 1 ...
##  $ Title        : Factor w/ 6 levels "Ranked","Royalty",..: 6 5 4 5 6 6 6 3 5 5 ...
##  $ family_size  : Factor w/ 3 levels "large","medium",..: 3 3 3 3 3 3 3 1 2 3 ...
##  $ real_fare    : num  7.25 71.28 7.92 26.55 8.05 ...
##  $ shared_ticket: Factor w/ 2 levels "0","1": 1 1 1 2 1 1 1 2 2 2 ...
##  $ known_deck   : Factor w/ 2 levels "0","1": 1 2 1 2 1 1 2 1 1 1 ...
```

```
df_preProcess <- preProcess(df, method = 'bagImpute')
df_preProcess
```

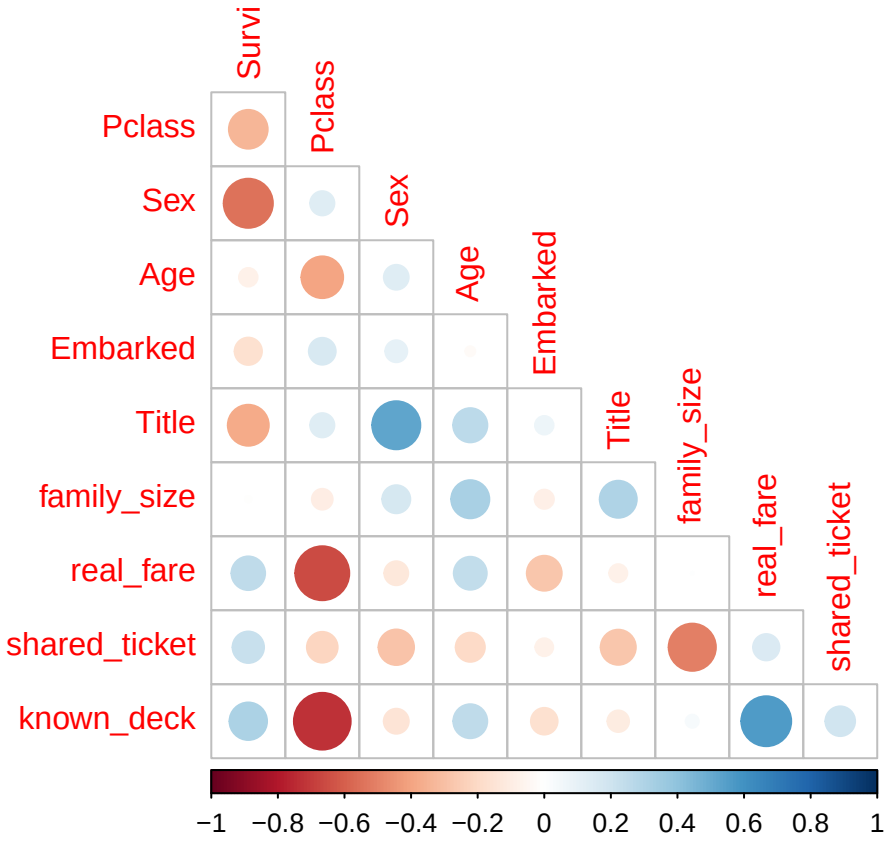
```
## Created from 714 samples and 10 variables
##
## Pre-processing:
##   - bagged tree imputation (2)
##   - ignored (8)
```

```
df <- predict(df_preProcess, newdata = df)
anyNA(df)
```

```
## [1] FALSE
```

New Correlations

Here's the new corelations we have now.



As we can see, some of the newly created predictors seems to be correlated. Thus, the shared_ticket predictor is correlated with the family size, and the real_fare is indeed correlated with Pclass.

ML algorithms

First let's create the partition into train and validation set.

Partition creation

```
set.seed(1)
val_index <- createDataPartition(y, times = 1, p = 0.8, list = FALSE)
df_train <- df[val_index, ]
y_train <- as.factor(y[val_index])
df_validation <- df[-val_index, ]
y_validation <- as.factor(y[-val_index])
```

Training

We are going to use multiple models and combine their results, since we are on a classification task, we'll mostly use tree based methods, as well as perceptron and gradient boosting methods.

```
models <- c("rf", "ranger", "RRF", "wsrf", "Rborist", "avNNet", "mlp", "monmlp",
    "adaboost", "gbm")
```

Some algorithms might take some time to run, the timeElapsed column here is in seconds.

```
## Registered S3 method overwritten by 'RRF':
##   method      from
##   plot.margin randomForest
```

event	start	end	timeElapsed	comment
rf	2019-06-17 22:34:13	2019-06-17 22:34:42	29	train
ranger	2019-06-17 22:34:42	2019-06-17 22:34:56	14	train
RRF	2019-06-17 22:34:56	2019-06-17 22:45:59	663	train
wsrf	2019-06-17 22:45:59	2019-06-17 22:46:33	34	train
Rborist	2019-06-17 22:46:33	2019-06-17 22:52:09	336	train
avNNet	2019-06-17 22:52:09	2019-06-17 22:53:12	63	train
mlp	2019-06-17 22:53:12	2019-06-17 22:53:24	12	train
monmlp	2019-06-17 22:53:24	2019-06-17 22:54:28	64	train
adaboost	2019-06-17 22:54:28	2019-06-17 22:58:39	251	train
gbm	2019-06-17 22:58:39	2019-06-17 22:58:46	7	train

```
set.seed(1)
fits <- lapply(models, function(model) {
    print(model)
    train(Survived ~ ., method = model, data = df_train)
})
names(fits) <- models
```

Predicting

Let's apply the predicting to the validation and see the accuracy of each models. Note here that some models which took really long to run, like RRF and Rborist, had some pretty low results comapred to gbm which only 7 seconds.

```
fits_predicts <- sapply(fits, function(fits) {
    predict(fits, newdata = df_validation)
})
dim(fits_predicts)
```

```
## [1] 177  10
```

the overall accuracy is around 0.80, but it was really long to run all of those models. So let's check the accuracy of the models obtained via cross validation and take only the best performing ones.

```
acc <- colMeans(fits_predicts == y_validation)
acc %>% knitr::kable()
```

	x
rf	0.7909605
ranger	0.8022599
RRF	0.7796610
wsrf	0.7966102
Rborist	0.7570621
avNNet	0.7966102
mlp	0.5988701
monmlp	0.8135593
adaboost	0.8022599

	x
gbm	0.8248588

```
mean(acc)
```

```
## [1] 0.7762712
```

```
votes <- rowMeans(fits_predicts == "1")
y_hat <- ifelse(votes > 0.5, "1", "0")
mean(y_hat == y_validation)
```

```
## [1] 0.8079096
```

```
results <- tibble(method = "All the models", Sensitivity = confusionMatrix(reference = y_validation,
    data = factor(y_hat))$byClass[1], Specificity = confusionMatrix(reference = y_validation,
    data = factor(y_hat))$byClass[2])
results %>% knitr::kable()
```

method	Sensitivity	Specificity
All the models	0.8807339	0.6911765

```
acc_hat <- sapply(fits, function(fit) min(fit$results$Accuracy))
acc_hat %>% knitr::kable()
```

	x
rf	0.8110983
ranger	0.8055634
RRF	0.7887512
wsrf	0.8235530
Rborist	0.6215730
avNNet	0.8234858
mlp	0.6395313
monmlp	0.8096709
adaboost	0.7974977
gbm	0.8109375

```
mean(acc_hat)
```

```
## [1] 0.7731662
```

Let's predict again but just with the selected models.

```
names(which(acc_hat >= 0.8))
new_models <- names(which(acc_hat >= 0.8))
new_fits <- lapply(new_models, function(model) {
    print(model)
    train(Survived ~ ., method = model, data = df_train)
})
names(new_fits) <- new_models
```

```
new_fits_predicts <- sapply(new_fits, function(fits) {
  predict(fits, newdata = df_validation)
```

```
})

votes <- rowMeans(new_fits_predicts == "1")
y_hat <- ifelse(votes > 0.5, "1", "0")
mean(y_hat == y_validation)
```

```
## [1] 0.7966102
```

```
results <- bind_rows(
  results,
  tibble(
    method = "The best models",
    Sensitivity = confusionMatrix(reference = y_validation, data = factor(y_hat))$byClass[1],
    Specificity = confusionMatrix(reference = y_validation, data = factor(y_hat))$byClass[2]
  )
)
results %>% knitr::kable()
```

method	Sensitivity	Specificity
All the models	0.8807339	0.6911765
The best models	0.8899083	0.6470588

Conclusion

As a result, we slightly improved the overall accuracy, but we also improved the running time of the model as well as the specificity.