Titanic Passenger Survival Prediction - EDA and Feature Engineering

July 16, 2017

This is the Exploration Data Analysis (EDA) part of kaggle competition - Titanic. The contents include: data overview and inspect, data cleaning (filling missing data and factorization) and some feature engineering.

```
setwd('C:/Users/Bangda/Desktop/kaggle/titanic')
library(ggplot2)
library(reshape2)
library(stringr)
library(dplyr)
library(rpart)
library(e1071)
train <- read.csv('train.csv', header = TRUE)
test  <- read.csv('test.csv', header = TRUE)
```

1. Data Overview and Simple Feature Engineering

First let's check the data we have,

```
glimpse(train)
```

```
## Observations: 891
## Variables: 12
## $ PassengerId <int> 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15,...
## $ Survived    <int> 0, 1, 1, 1, 0, 0, 0, 0, 1, 1, 1, 1, 0, 0, 0, 1, 0,...
## $ Pclass      <int> 3, 1, 3, 1, 3, 3, 1, 3, 3, 2, 3, 1, 3, 3, 3, 2, 3,...
## $ Name        <fctr> Braund, Mr. Owen Harris, Cumings, Mrs. John Bradl...
## $ Sex         <fctr> male, female, female, female, male, male, male, m...
## $ Age         <dbl> 22, 38, 26, 35, 35, NA, 54, 2, 27, 14, 4, 58, 20, ...
## $ SibSp       <int> 1, 1, 0, 1, 0, 0, 0, 3, 0, 1, 1, 0, 0, 1, 0, 0, 4,...
## $ Parch       <int> 0, 0, 0, 0, 0, 0, 0, 1, 2, 0, 1, 0, 0, 5, 0, 0, 1,...
## $ Ticket      <fctr> A/5 21171, PC 17599, STON/O2. 3101282, 113803, 37...
## $ Fare        <dbl> 7.2500, 71.2833, 7.9250, 53.1000, 8.0500, 8.4583, ...
## $ Cabin       <fctr> , C85, , C123, , , E46, , , , G6, C103, , , , , ,...
## $ Embarked    <fctr> S, C, S, S, S, Q, S, S, S, C, S, S, S, S, S, S, Q...
```

From `train` data, we can see that:
(1) number of *Ticket* is less than *Name*, some passengers share one ticket;
(2) *Embarked* has unidentified levels;
(3) *Cabin* has unidentified levels;

```
glimpse(test)
```

```
## Observations: 418
## Variables: 11
## $ PassengerId <int> 892, 893, 894, 895, 896, 897, 898, 899, 900, 901, ...
## $ Pclass      <int> 3, 3, 2, 3, 3, 3, 3, 2, 3, 3, 3, 1, 1, 2, 1, 2, 2,...
## $ Name        <fctr> Kelly, Mr. James, Wilkes, Mrs. James (Ellen Needs...
## $ Sex         <fctr> male, female, male, male, female, male, female, m...
```

```
## $ Age         <dbl> 34.5, 47.0, 62.0, 27.0, 22.0, 14.0, 30.0, 26.0, 18...
## $ SibSp       <int> 0, 1, 0, 0, 1, 0, 0, 1, 0, 2, 0, 0, 1, 1, 1, 1, 0,...
## $ Parch       <int> 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0,...
## $ Ticket      <fctr> 330911, 363272, 240276, 315154, 3101298, 7538, 33...
## $ Fare        <dbl> 7.8292, 7.0000, 9.6875, 8.6625, 12.2875, 9.2250, 7...
## $ Cabin       <fctr> , , , , , , , , , , , , B45, , E31, , , , , , , ,...
## $ Embarked    <fctr> Q, S, Q, S, S, S, Q, S, C, S, S, S, S, S, S, C, Q...
```

From `test` data we could draw same conclusion on *Ticket* and *Cabin*.

(1) Family

We can create a new variable called *Family*, where if the sum of *SibSp* and *Parch* is 0 means the passengers came alone. Additionally, we can treat the sum of *SibSp* and *Parch* to be *num_family*.

```r
train$Family <- ifelse((train$SibSp + train$Parch) == 0, 0, 1)
test$Family  <- ifelse((test$SibSp + test$Parch) == 0, 0, 1)
train$num_family <- train$SibSp + train$Parch
test$num_family  <- test$SibSp + test$Parch
```

(2) Title and Name

We can inspect the *Name* variable,

```r
head(train$Name)
```

```
## [1] Braund, Mr. Owen Harris
## [2] Cumings, Mrs. John Bradley (Florence Briggs Thayer)
## [3] Heikkinen, Miss. Laina
## [4] Futrelle, Mrs. Jacques Heath (Lily May Peel)
## [5] Allen, Mr. William Henry
## [6] Moran, Mr. James
## 891 Levels: Abbing, Mr. Anthony ... Zimmerman, Mr. Leo
```

and we can find there are title informations contained in *Name*.

```r
extract_name <- function(name, type) {
  # split Name to title, first and last name
  splited_name <- str_split(name, '[,.]') %>% unlist() %>% str_trim()
  return(splited_name[type])
}
# add new variables to train and test
train$first_name <- sapply(train$Name, extract_name, 1)
train$title      <- sapply(train$Name, extract_name, 2)
train$last_name  <- sapply(train$Name, extract_name, 3)
test$first_name  <- sapply(test$Name, extract_name, 1)
test$title       <- sapply(test$Name, extract_name, 2)
test$last_name   <- sapply(test$Name, extract_name, 3)
# check the categories of title
train$title %>% unique()
```

```
##  [1] "Mr"           "Mrs"          "Miss"         "Master"
##  [5] "Don"          "Rev"          "Dr"           "Mme"
##  [9] "Ms"           "Major"        "Lady"         "Sir"
## [13] "Mlle"         "Col"          "Capt"         "the Countess"
```

```
## [17] "Jonkheer"
```

```r
test$title  %>% unique()
```

```
## [1] "Mr"     "Mrs"    "Miss"   "Master" "Ms"     "Col"    "Rev"    "Dr"
## [9] "Dona"
```

```r
# get the list of all categories
titles <- base::union(train$title %>% unique(),
                      test$title %>% unique())
# factorize title
train$title <- train$title %>% factor(levels = titles)
test$title  <- test$title %>% factor(levels = titles)
```

2. Missing Data

Then we check if `NA` exists in numeric variables

```r
apply(train, 2, function(x) sum(is.na(x)) )
```

```
## PassengerId    Survived      Pclass        Name         Sex         Age
##           0           0           0           0           0         177
##       SibSp       Parch      Ticket        Fare       Cabin    Embarked
##           0           0           0           0           0           0
##      Family  num_family  first_name       title   last_name
##           0           0           0           0           0
```

```r
apply(test,  2, function(x) sum(is.na(x)) )
```

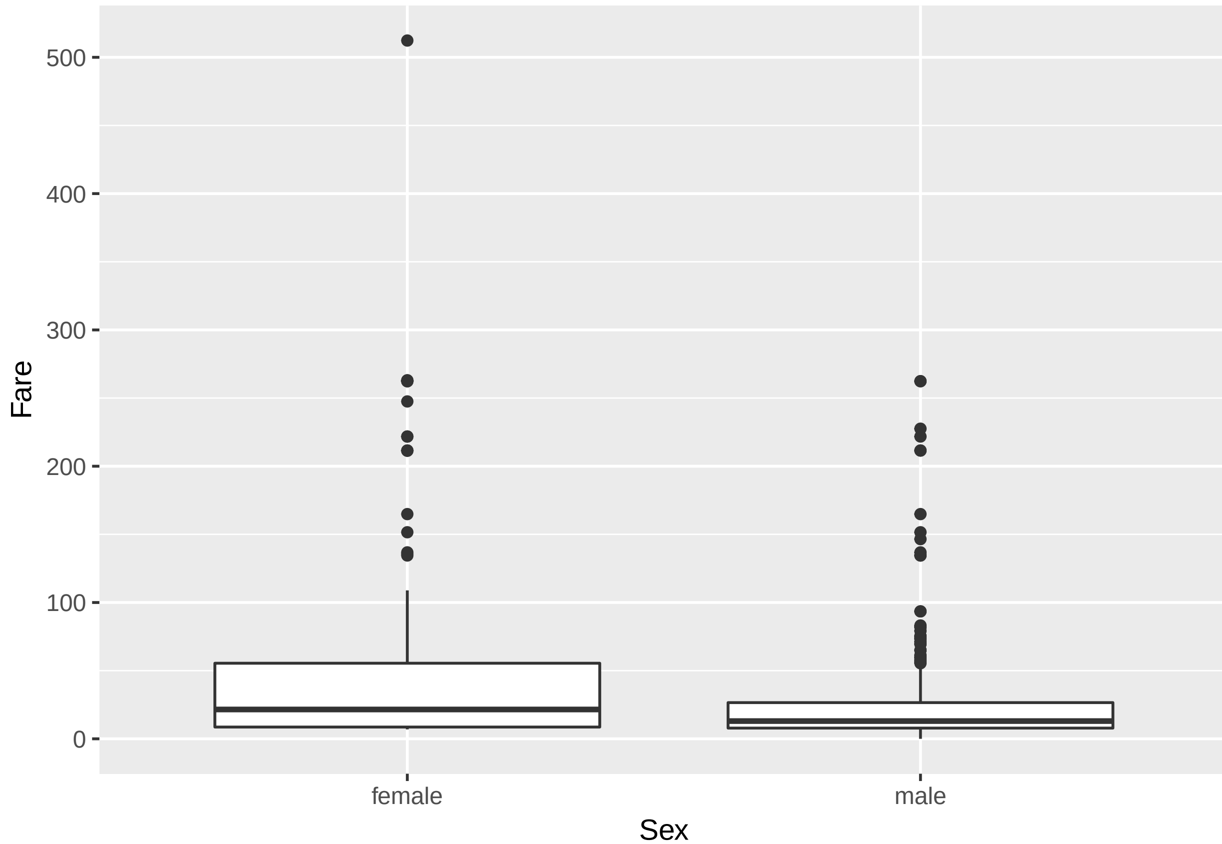
```
## PassengerId      Pclass        Name         Sex         Age       SibSp
##           0           0           0           0          86           0
##       Parch      Ticket        Fare       Cabin    Embarked      Family
##           0           0           1           0           0           0
##  num_family  first_name       title   last_name
##           0           0           0           0
```

We can see that there are 177 `NA` for *Age* in `train` and 86 `NA` for *Age*, 1 `NA` for *Fare* in test.

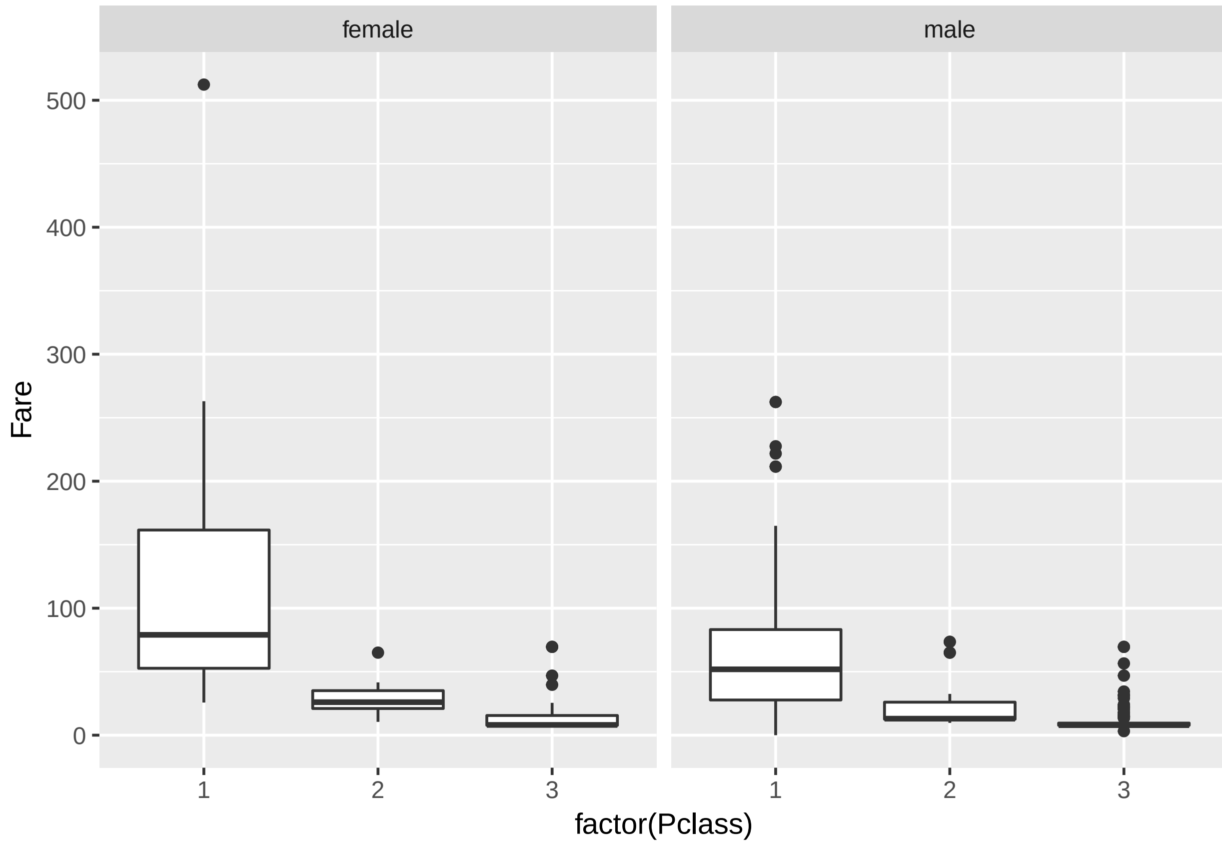
Combined with conclusions in above section, there are also some missing values in *Cabin* and *Embark*. We gonna impute these missing values and start from the simpliest case.

(1) Fare

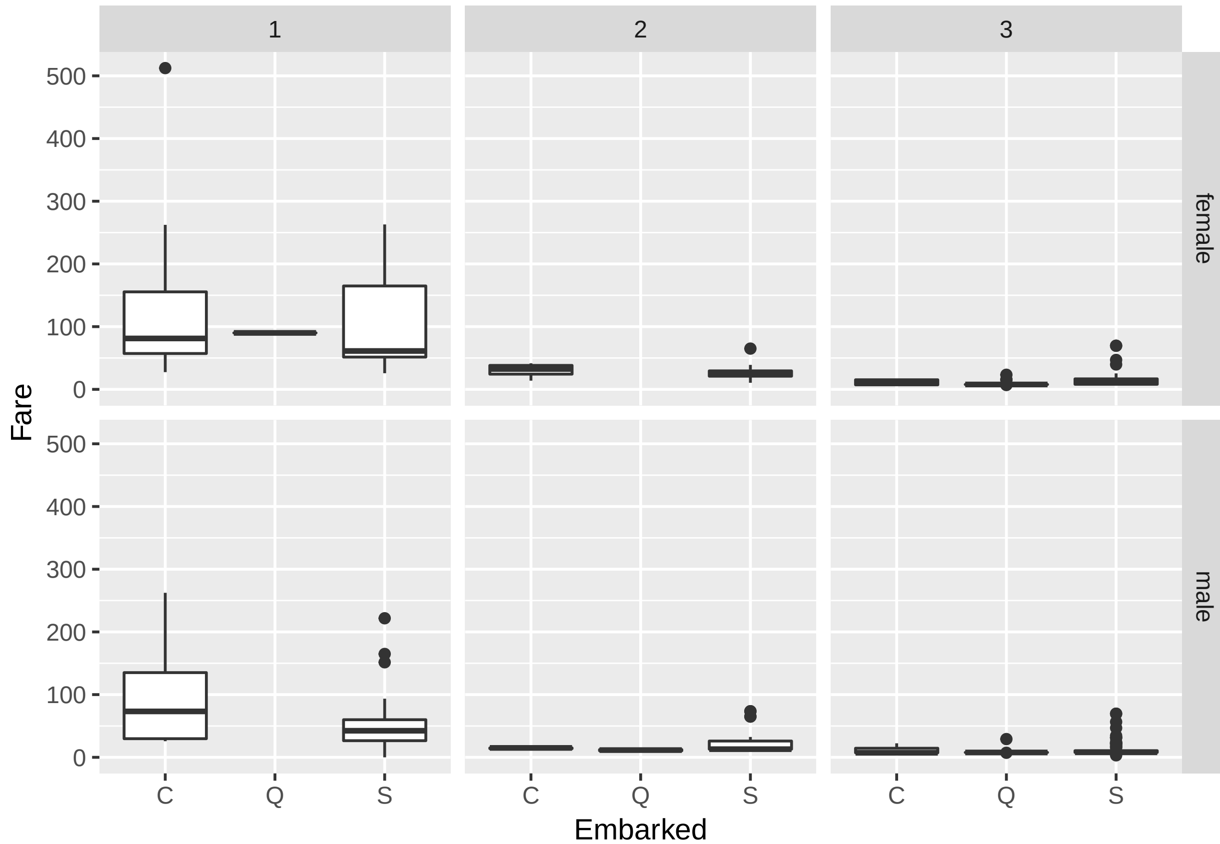
```r
idx_na_fare <- which(is.na(test$Fare))
# with sex
test[-idx_na_fare, ] %>%
  ggplot() +
  geom_boxplot(aes(x = Sex, y = Fare))
```




```r
# with pclass
test[-idx_na_fare, ] %>%
  ggplot() +
  geom_boxplot(aes(x = factor(Pclass), y = Fare)) +
  facet_wrap(~ Sex)
```




```
# with embarked
test[-idx_na_fare, ] %>%
  ggplot() +
  geom_boxplot(aes(x = Embarked, y = Fare)) +
  facet_grid(Sex ~ Pclass)
```



Therefore, for the one missing *Fare* we can use the mean or median value of the group (group by *Sex*, *Pclass*, *Embarked*) that the case fall in.

```
test[idx_na_fare, c('Sex', 'Pclass', 'Embarked')]
```

```
##      Sex Pclass Embarked
## 153 male      3        S
```

```
# use mean and median
test[-idx_na_fare, ] %>%
  select(Sex, Pclass, Embarked, Fare) %>%
  filter(Sex == 'male', Pclass == 3, Embarked == 'S') %>%
  summarise(avg_fare = mean(Fare), mid_fare = median(Fare))
```

```
##   avg_fare mid_fare
## 1 12.71887   7.9875
```

From the above figure, the missing case fall in the figure at position (2, 3), we can see that there are some outliers in that boxplot. Therefore here we decide to use median to fill the missing *Fare*,

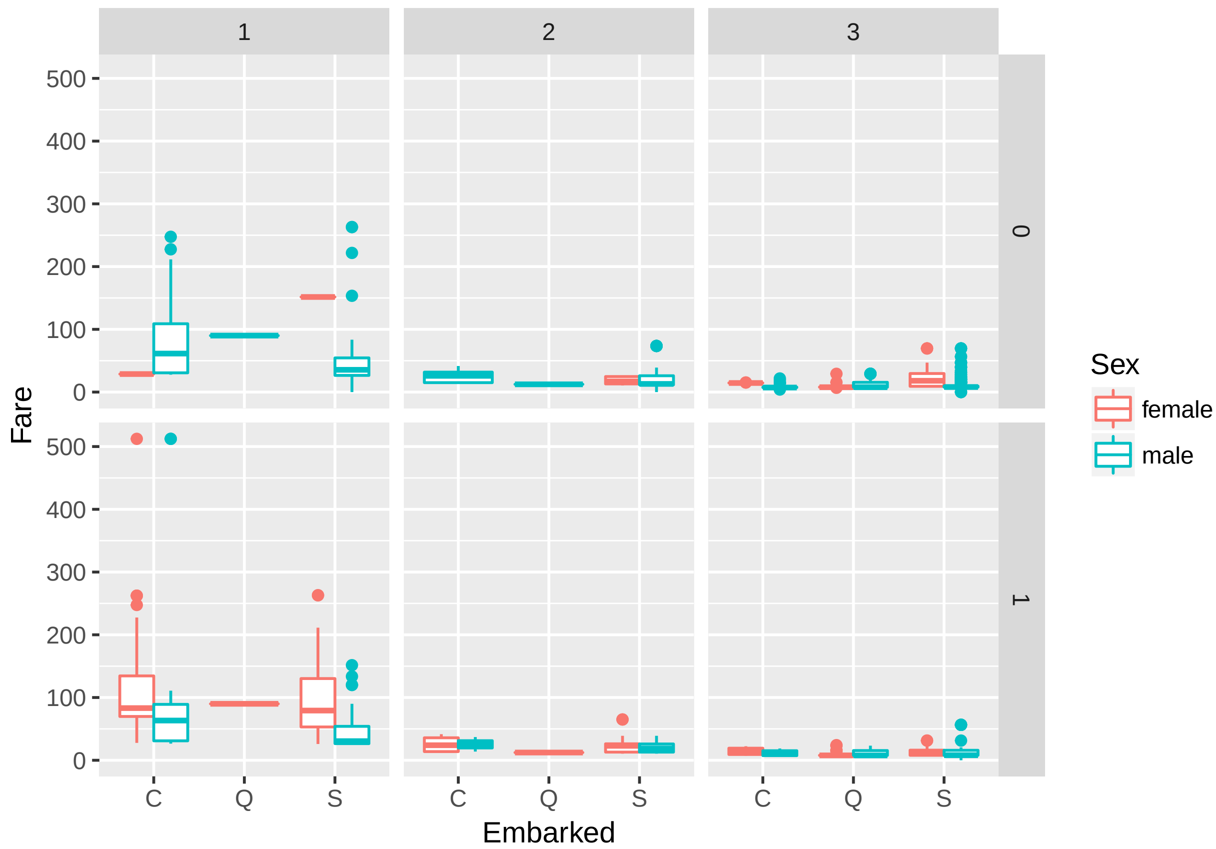
```
test[idx_na_fare, 'Fare'] <- 7.9875
```

(2) Embarked

```
idx_na_embark <- ((train$Embarked %>% as.character()) == '') %>% which()
train[idx_na_embark, ] %>%
  select(Survived, Fare, Sex, Pclass)
```

```
##     Survived Fare    Sex Pclass
## 62          1   80 female      1
## 830         1   80 female      1
```

```
# with fare, embarked, pclass
train[-idx_na_embark, ] %>%
  ggplot() +
  geom_boxplot(aes(x = Embarked, y = Fare, col = Sex)) +
  facet_grid(Survived ~ factor(Pclass))
```

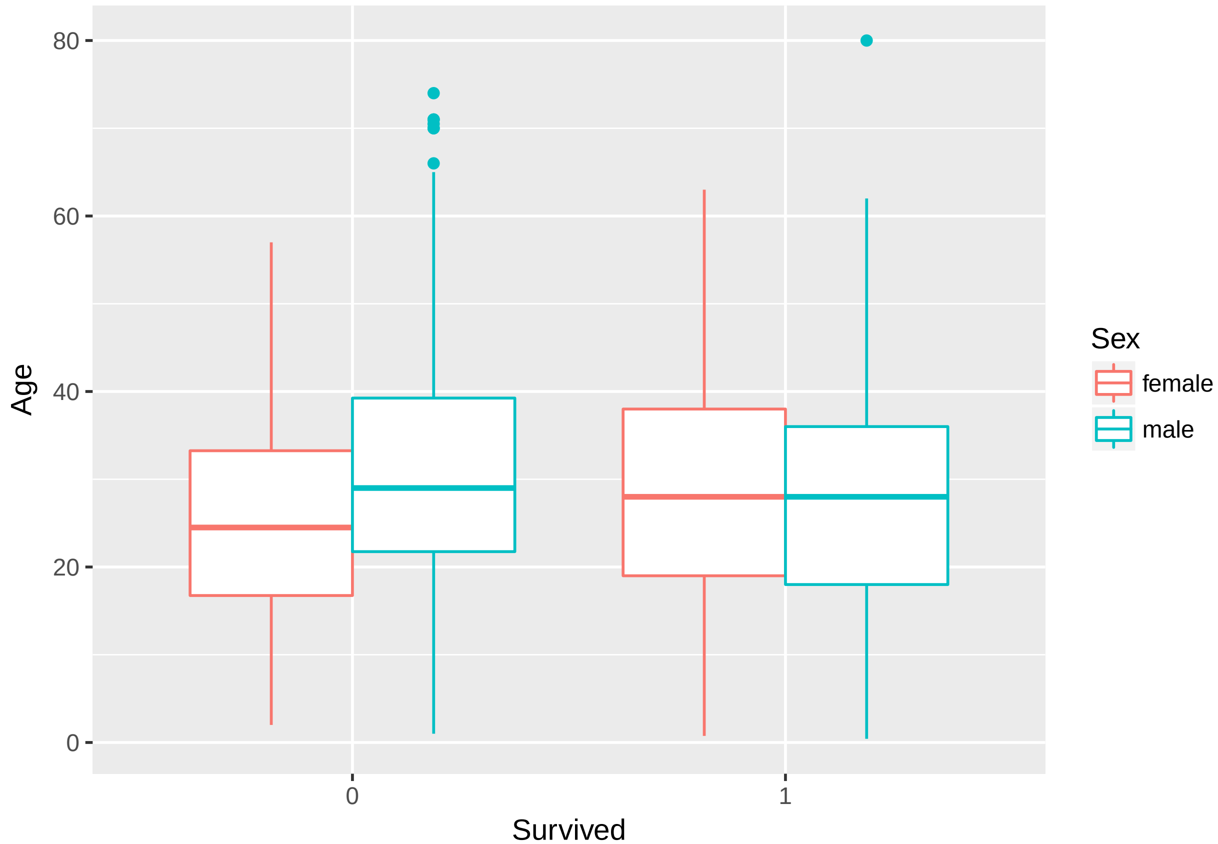


From the above box plots, we classify the *Embarked* of two missing cases to be `C`.

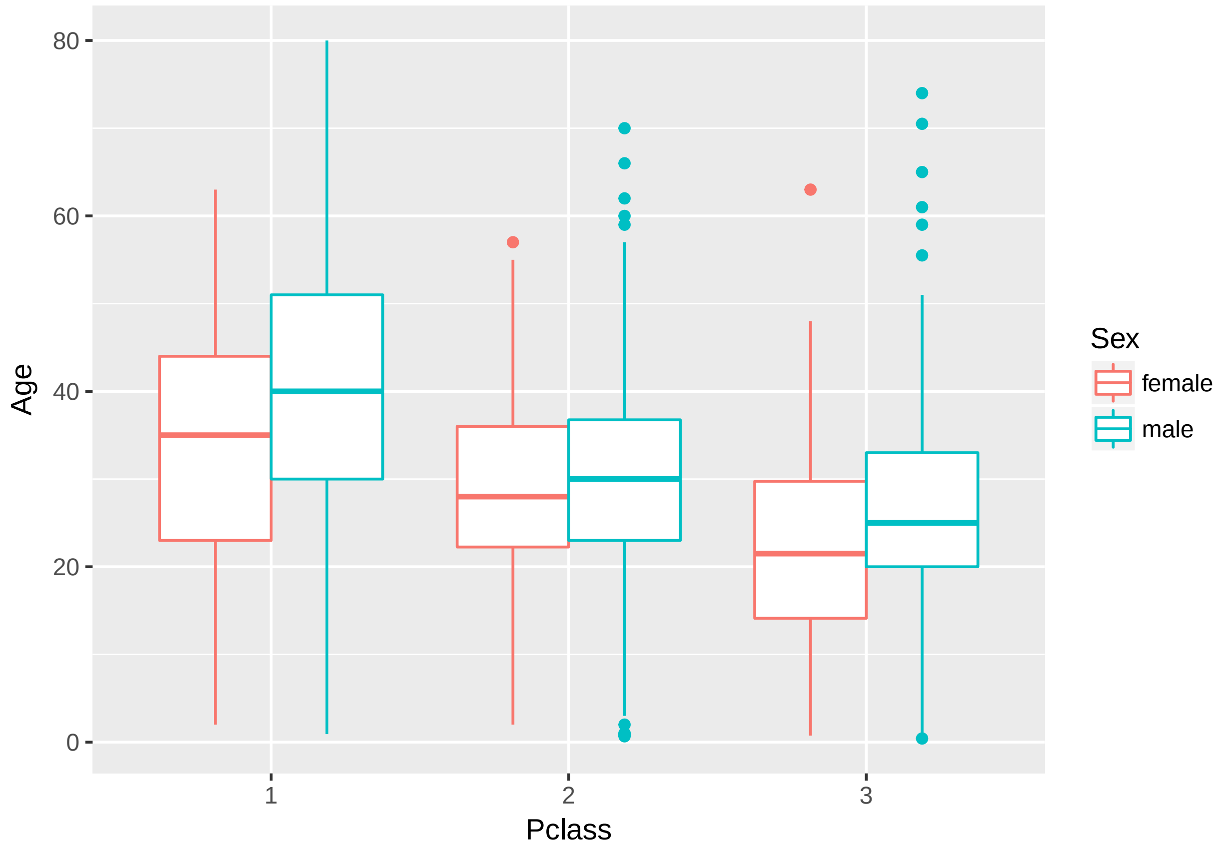
```
train[idx_na_embark, 'Embarked'] <- 'C'
train$Embarked <- factor(train$Embarked, levels = c('C', 'Q', 'S'))
```

(3) Age

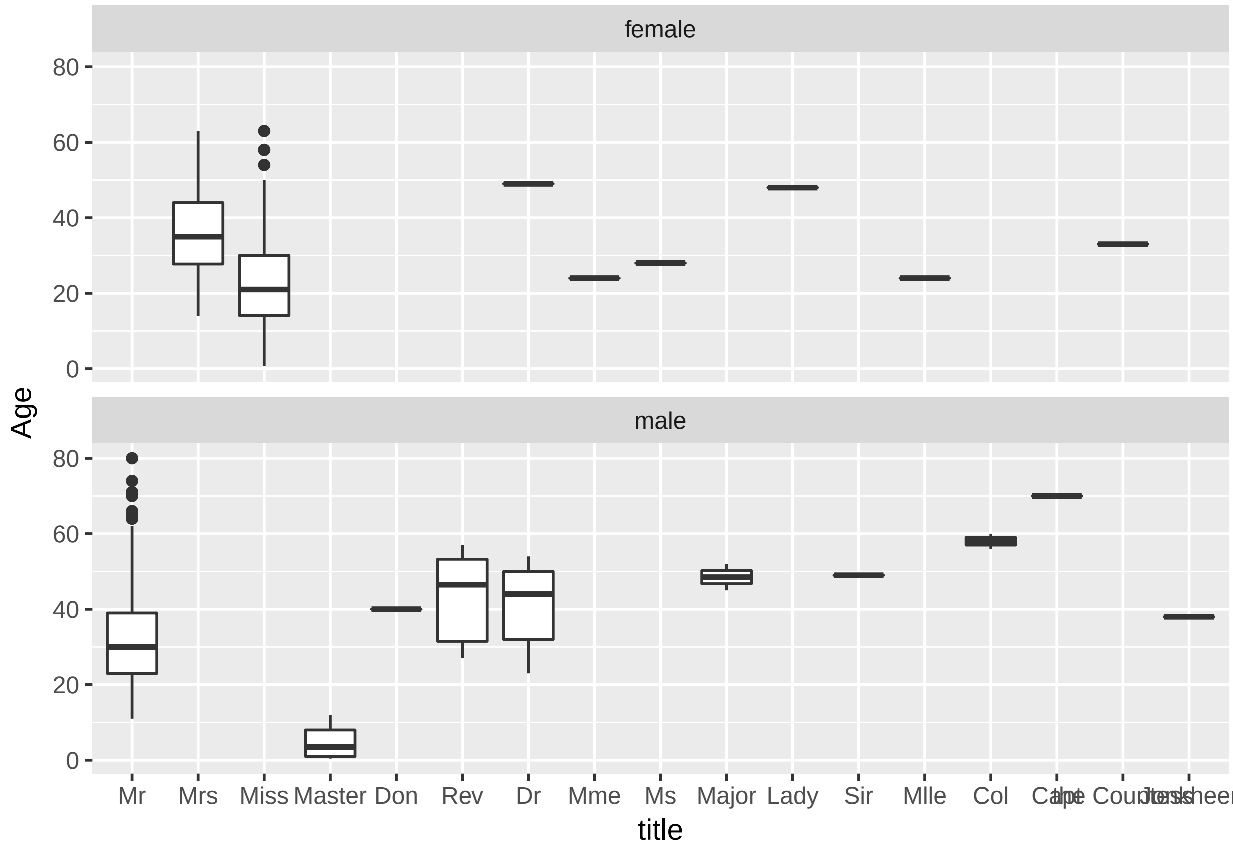
```
idx_na_age_train <- which(is.na(train$Age))
train_wna_age  <- train[idx_na_age_train, ]
train_wona_age <- train[-idx_na_age_train, ]
# age ~ sex | survived
train_wona_age %>%
  ggplot() +
  geom_boxplot(aes(x = factor(Survived), y = Age, col = Sex)) +
  xlab('Survived')
```



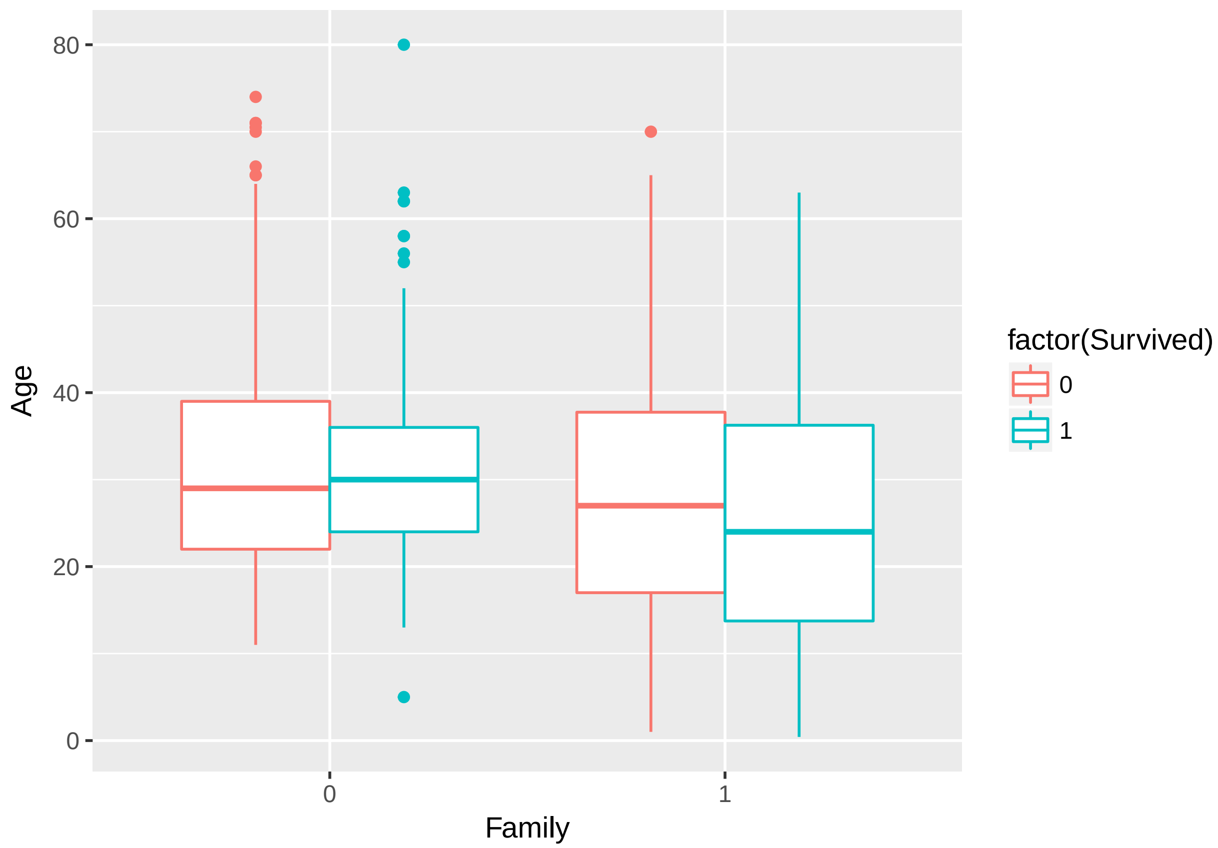

```
# age ~ sex | pclass
train_wona_age %>%
  ggplot() +
  geom_boxplot(aes(x = factor(Pclass), y = Age, col = Sex)) +
  xlab('Pclass')
```



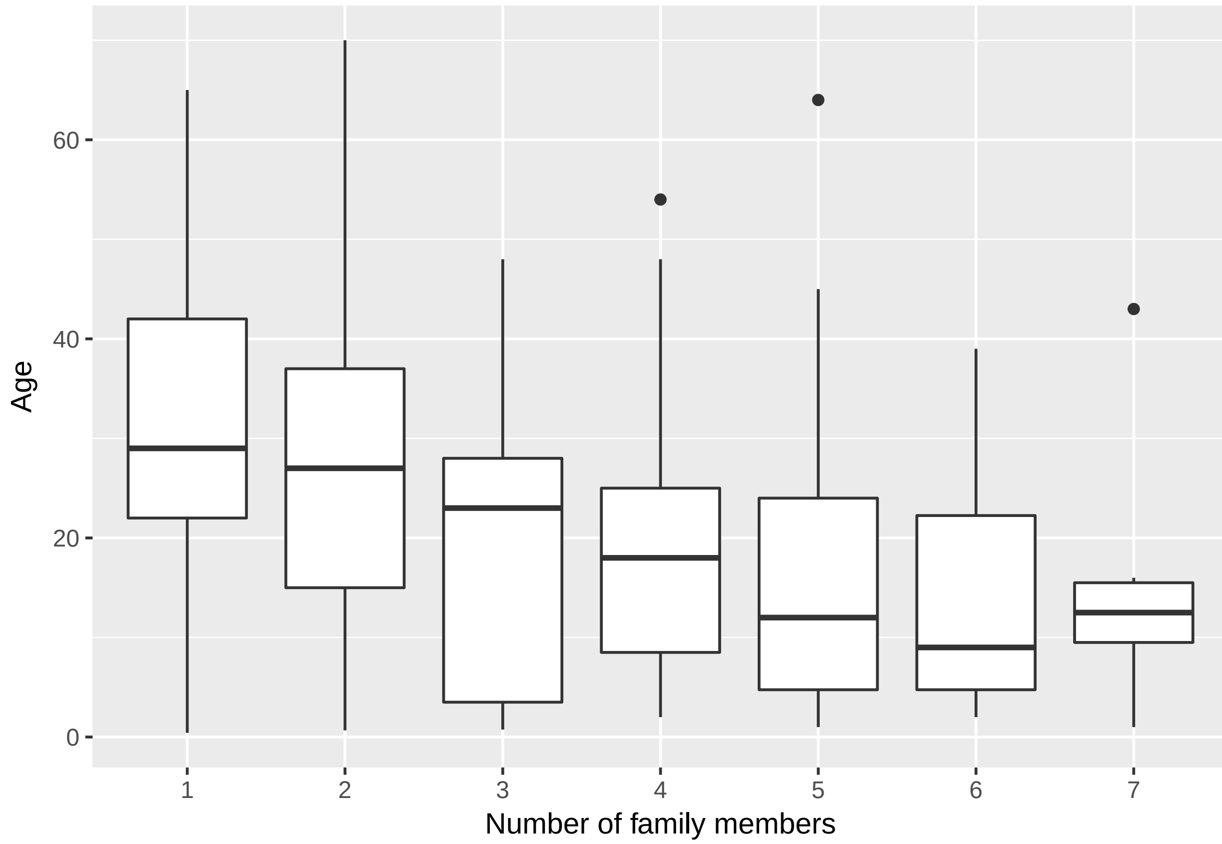

```
# age ~ title
train_wona_age %>%
  ggplot() +
  geom_boxplot(aes(x = title, y = Age)) +
  facet_wrap(~ Sex, nrow = 2)
```



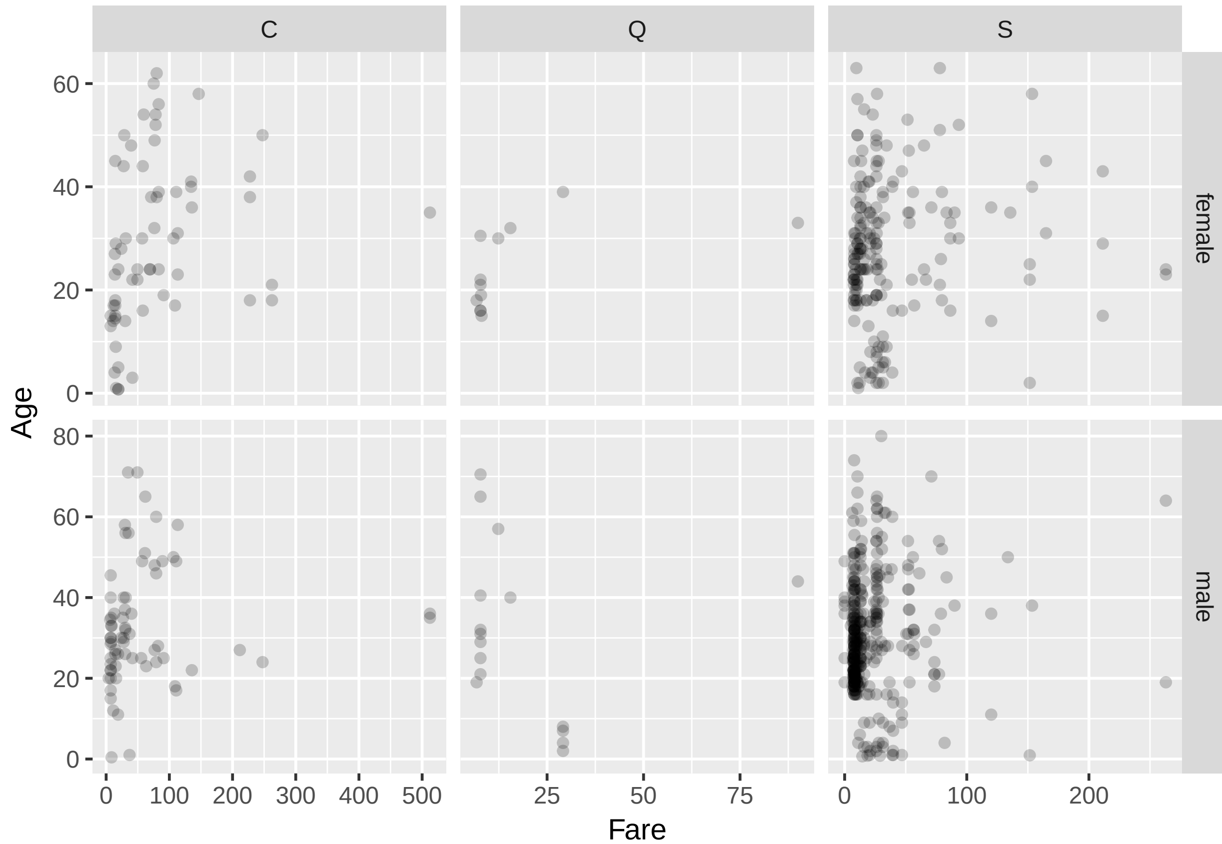

```
# age ~ family | survived
train_wona_age %>%
  ggplot() +
  geom_boxplot(aes(x = factor(Family), y = Age, col = factor(Survived))) +
  xlab('Family')
```




```r
# age ~ numfamily
train_wona_age %>%
  filter(num_family > 0) %>%
  ggplot() +
  geom_boxplot(aes(x = factor(num_family), y = Age)) +
  xlab('Number of family members')
```




```r
# age ~ fare | pclass + sex
train_wona_age %>%
  ggplot() +
  geom_point(aes(x = Fare, y = Age), alpha = I(1/5)) +
  facet_grid(Sex ~ Embarked, scale = 'free')
```



In conclusion, we can find that *Age* relate to *Sex*, *Pclass*, *title*, *family*, *numfamily* and *fare*, also there could be some interactions among those variables. The reason why we didn't include *Survived* is it's not exist in `test`.

To fill the missing values, for simplicity, we can use linear regression or regression tree.

First we gonna split the `train_wona` into two parts: one for training and one for testing the prediction of *Age*.

```
# train and validation set
train_age <- train_wona_age %>%
  dplyr::select(PassengerId, Age, Pclass, Fare, Family, num_family, title) %>%
  dplyr::mutate(Pclass = factor(Pclass), Family = factor(Family))
# need to fill age
pred_age  <- train_wna_age %>%
  dplyr::select(PassengerId, Age, Pclass, Fare, Family, num_family, title) %>%
  dplyr::mutate(Pclass = factor(Pclass), Family = factor(Family))
# test set
test <- test %>%
  dplyr::mutate(Pclass = factor(Pclass), Family = factor(Family))
set.seed(123)
train_train_idx <- sample(1:nrow(train_age), round(nrow(train_age)/5), replace = FALSE)
train_train_age <- train_age[train_train_idx, ]
train_test_age  <- train_age[-train_train_idx, ]
```

Next we use regression tree to fill in the `NA` in *Age*.

```
# training model
tr_model1 <- rpart(Age ~., data = train_train_age)
```

```r
# training error
pred_tr_model1_train <- predict(tr_model1, newdata = train_train_age)
(rmse_tr_model1_train <- (pred_tr_model1_train - train_train_age$Age)^2 %>%
    mean() %>% sqrt())
```

```
## [1] 9.162199
```

```r
# test error
pred_tr_model1_test <- predict(tr_model1, newdata = train_test_age)
(rmse_tr_model1_test <- (pred_tr_model1_test - train_test_age$Age)^2 %>%
    mean() %>% sqrt())
```

```
## [1] 12.82945
```

Since tree methods usually have overfitting issue, we try to tune the model by test several parameters, we can use `plot(tr_model1)` to check that `tr_model1` has depth 5, we also set `minsplit` which denote the minimum number of observations that must exist in a node for a split to be attempted.

```r
# tunning model
tr_model2 <- tune.rpart(Age ~., data = train_train_age, maxdepth = 2:7, minsplit = 2:10,
                        cp = c(0.001, 0.002, 0.005, 0.01, 0.02, 0.03))
tr_model2$best.parameters
```

```
##     minsplit    cp maxdepth
## 111        4 0.001        4
```

```r
# relative best model
tr_model3 <- rpart(Age ~., data = rbind(train_train_age, train_test_age),
                   maxdepth = 5, minsplit = 5, cp = .005)
# training error
pred_tr_model3_train <- predict(tr_model3, newdata = train_train_age)
(rmse_tr_model3_train <- (pred_tr_model3_train - train_train_age$Age)^2 %>%
    mean() %>% sqrt())
```

```
## [1] 9.718318
```

```r
# test error
pred_tr_model3_test <- predict(tr_model3, newdata = train_test_age)
(rmse_tr_model3_test <- (pred_tr_model3_test - train_test_age$Age)^2 %>%
    mean() %>% sqrt())
```

```
## [1] 10.83036
```

Then we apply this model on the data set with `NA` in *Age*, that's the last step to imputate missing values.

```r
pred_age$Age <- predict(tr_model3, newdata = pred_age)
# fill in train
train_wna_age <- train_wna_age %>%
  left_join(pred_age[, c('PassengerId', 'Age')], by = 'PassengerId') %>%
  select(-Age.x) %>%
  mutate(Age = Age.y) %>%
  select(-Age.y)
train <- rbind(train_wna_age, train_wona_age)
# fill in test
test_wna_age  <- test[is.na(test$Age), ]
test_wona_age <- test[!is.na(test$Age), ]
test_wna_age$Age <- predict(tr_model3, newdata = test_wna_age)
test <- rbind(test_wna_age, test_wona_age)
```

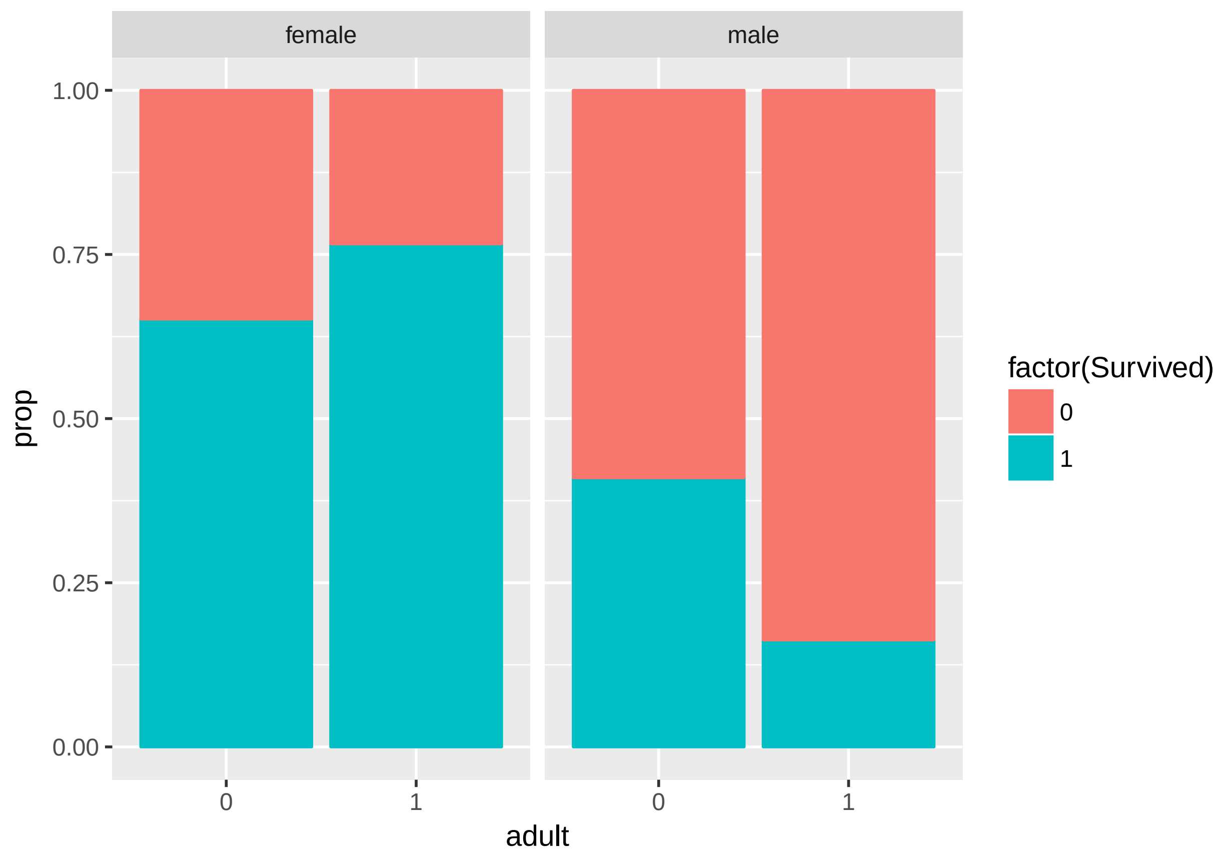
```r
# save the data
save.image("C:/Users/Bangda/Desktop/kaggle/titanic/eda1.RData")
```

3. EDA and more Feature Engineering

(1) Age

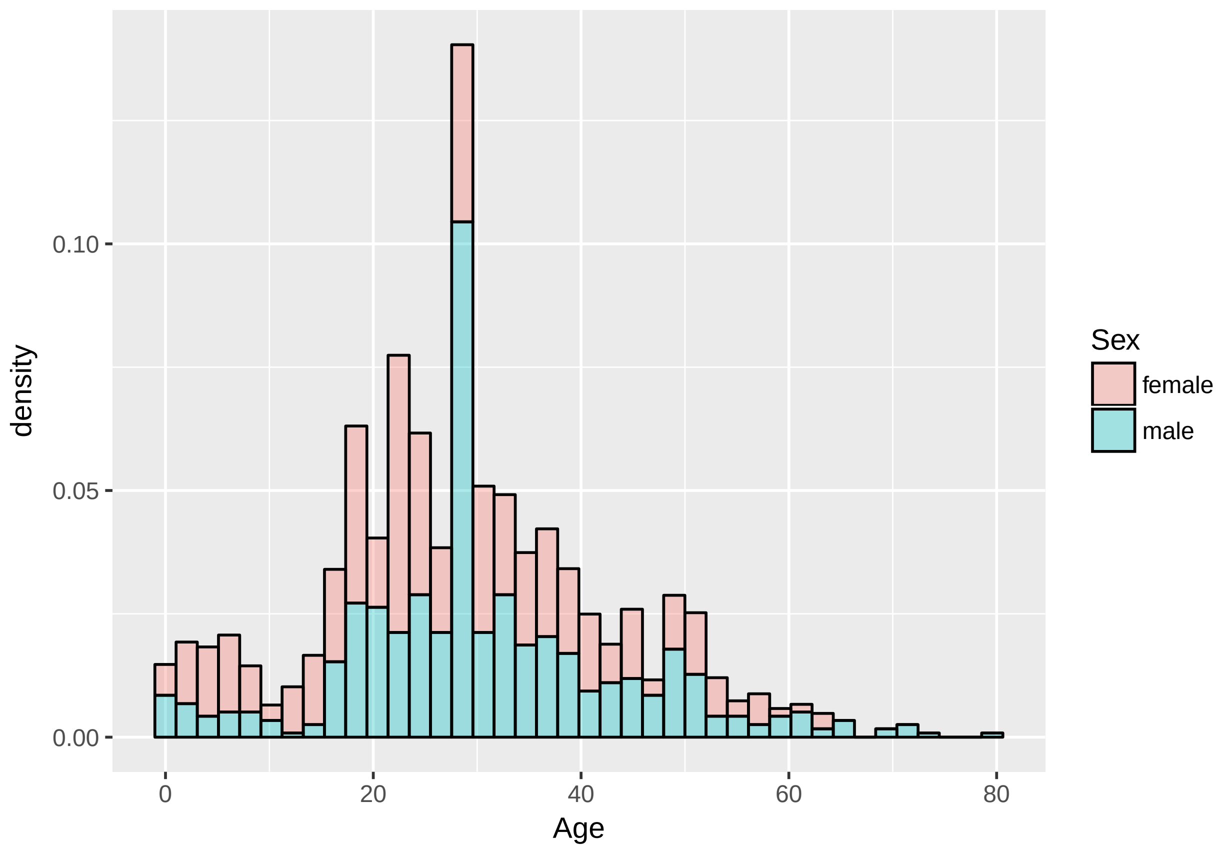
We check if the survival rate has significant difference between adults and teenagers, we can create a binary variable to denote whether the passenger was adult,

```r
train$adult <- ifelse(train$Age < 18, 0, 1) %>% factor()
test$adult  <- ifelse(test$Age  < 18, 0, 1) %>% factor()
# survived ~ adult | sex
train %>%
  ggplot(aes(x = adult, fill = factor(Survived),
             col = factor(Survived))) +
  geom_bar(position = 'fill') +
  facet_wrap(~ Sex) + ylab('prop')
```

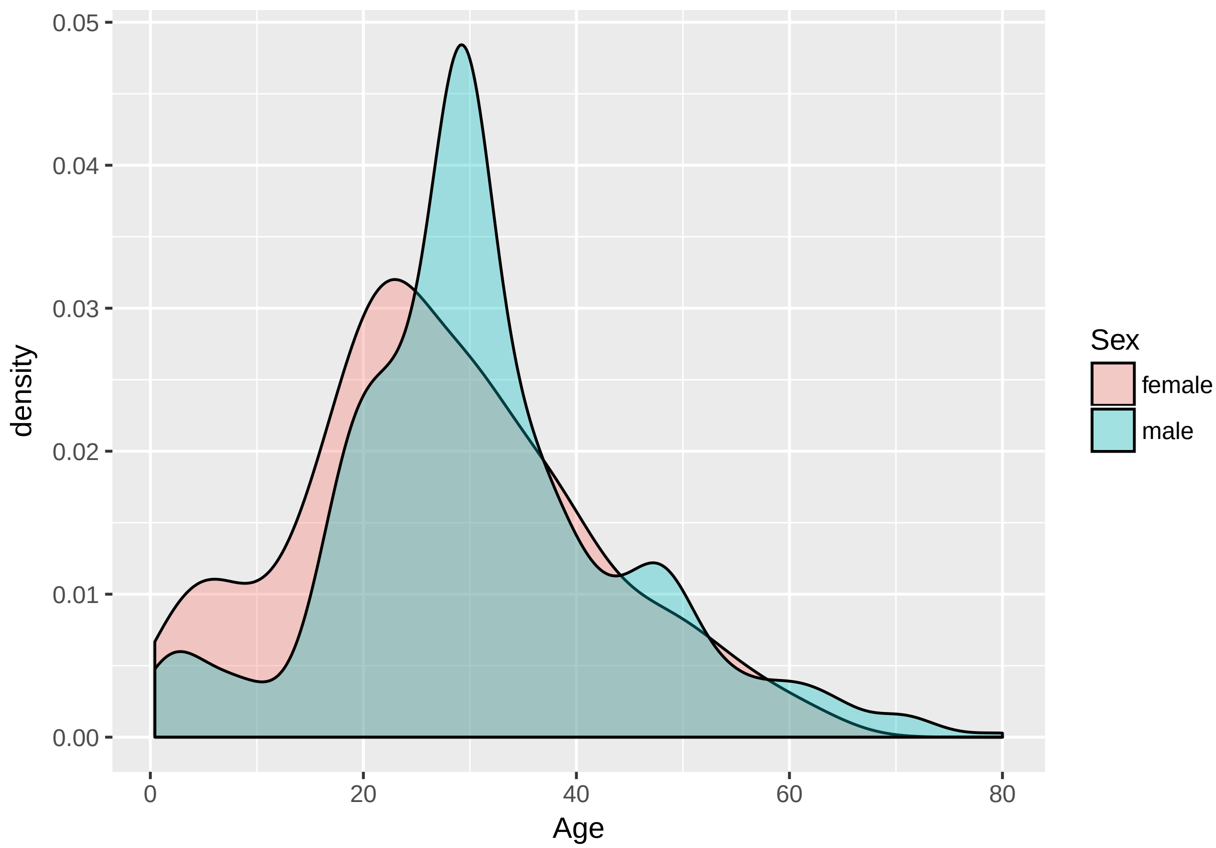


Next we gonna cut *Age* with more points, we see the distribution of *Age* looks like

```r
# distribution of age
train %>%
  ggplot(aes(x = Age, fill = Sex)) +
  geom_histogram(aes(y = ..density..),
                 col = 'black', alpha = I(1/3), bins = 40)
```

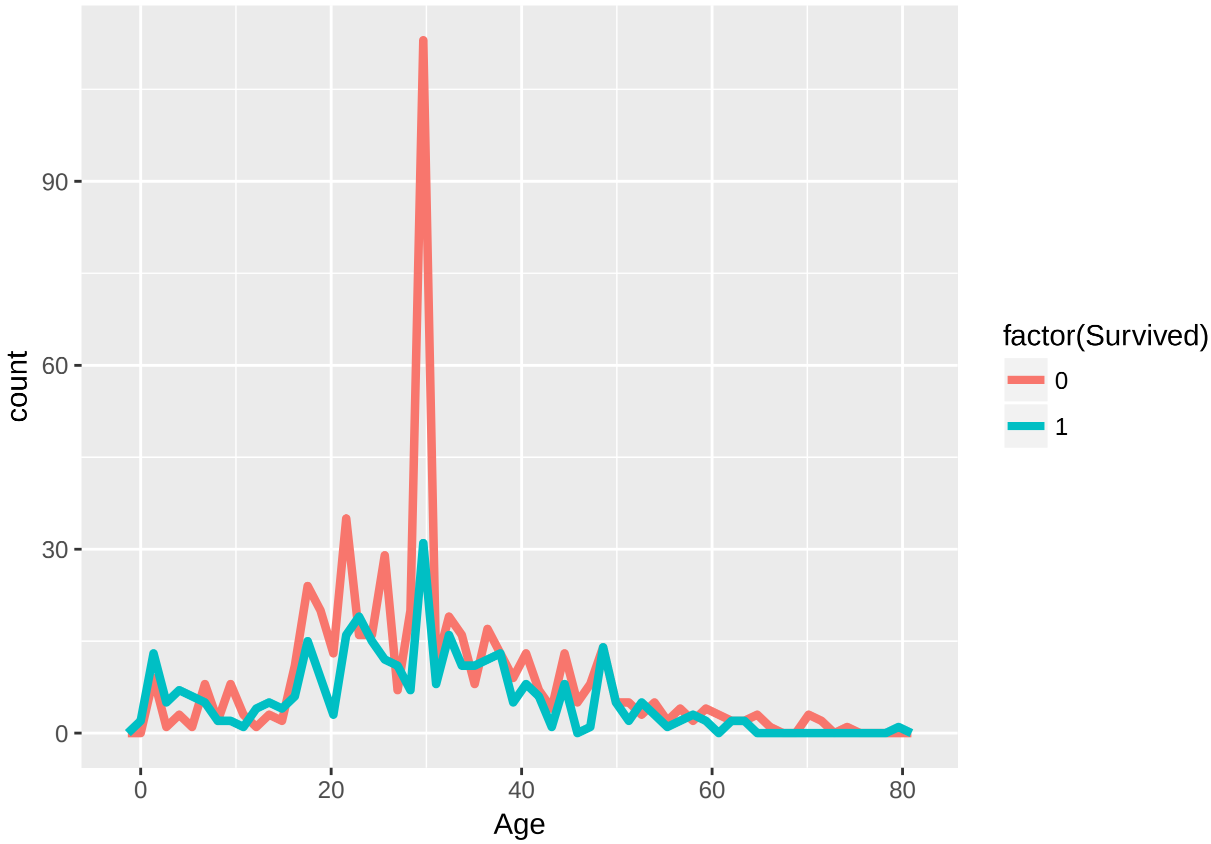



```
train %>%
  ggplot(aes(x = Age, fill = Sex)) +
  geom_density(aes(y = ..density..), alpha = I(1/3))
```

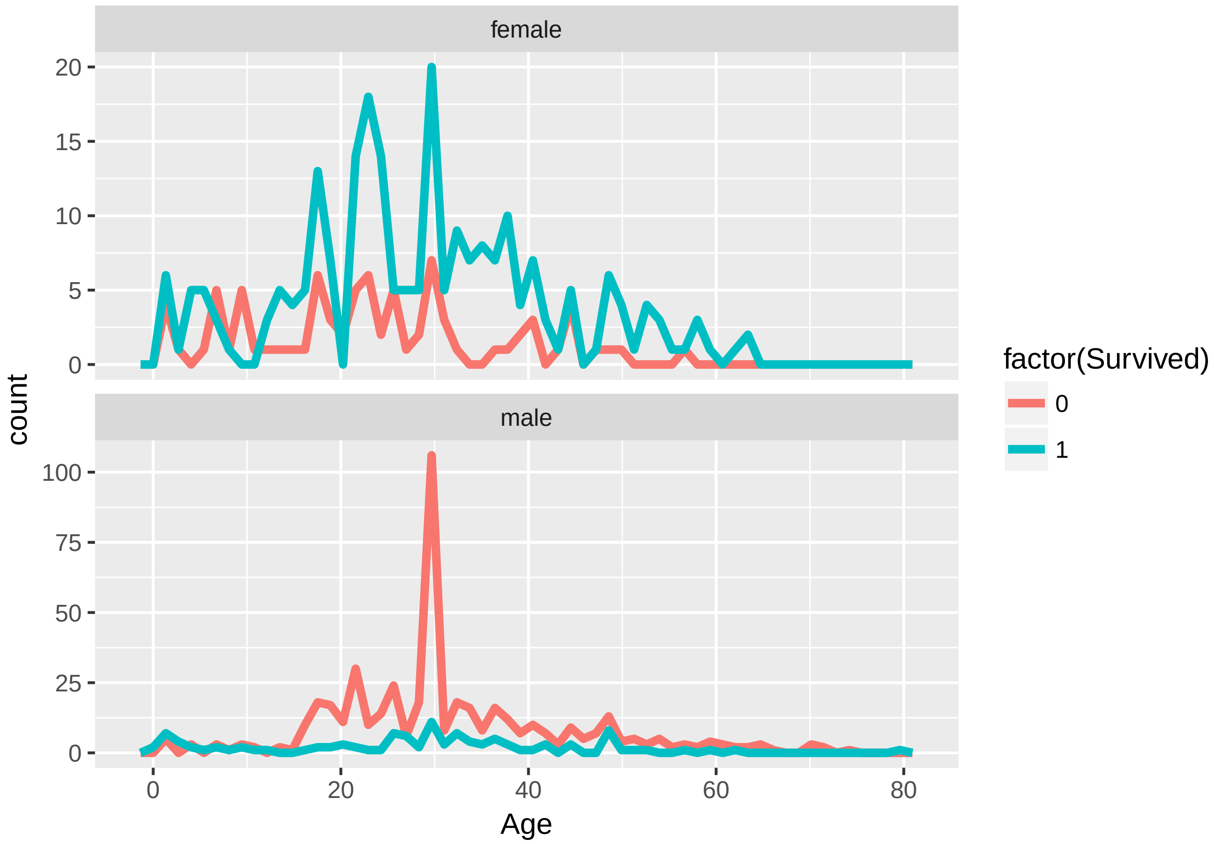


Also check *Survived* with *Age*

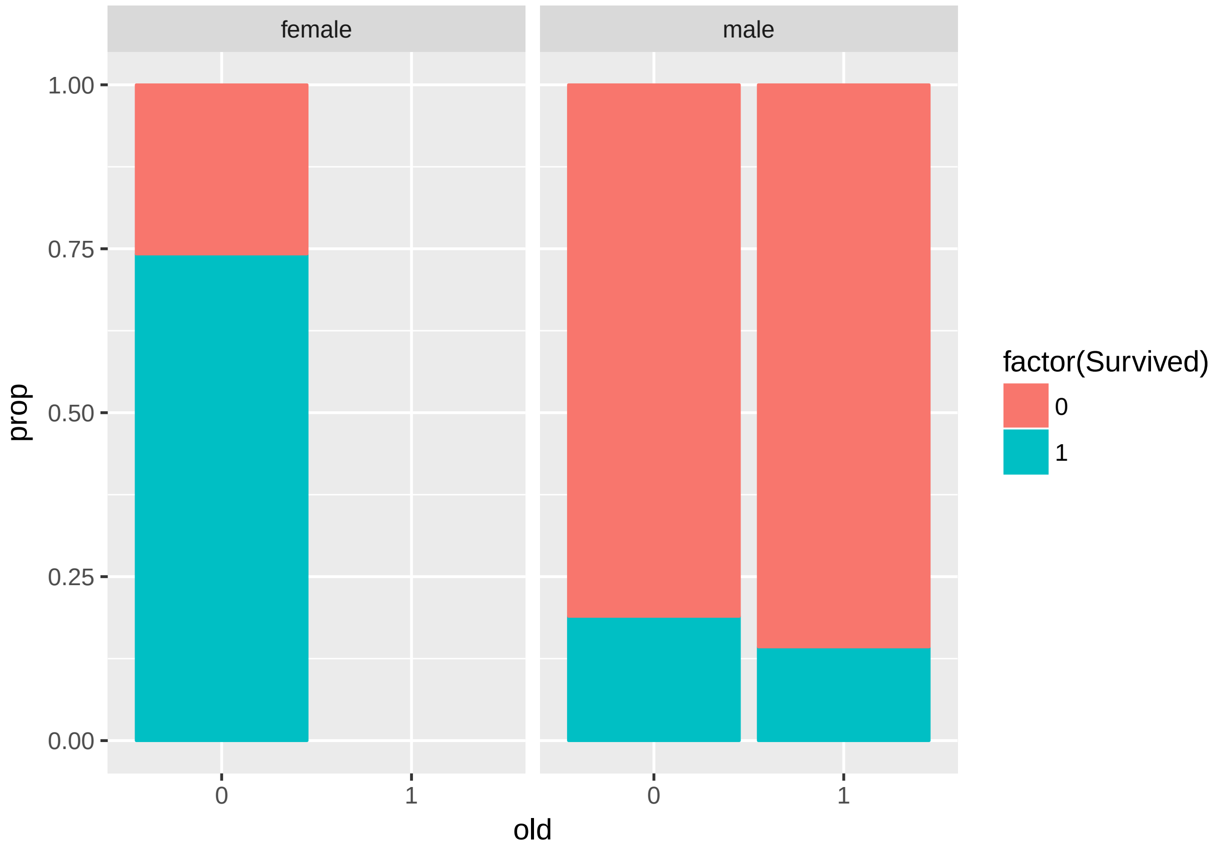
```
# survival at different age
train %>%
  ggplot() +
  geom_freqpoly(aes(x = Age, col = factor(Survived)), bins = 60, size = 1.5)
```




```r
# survival at different age | sex
train %>%
  ggplot() +
  geom_freqpoly(aes(x = Age, col = factor(Survived)), bins = 60, size = 1.5) +
  facet_wrap(~ Sex, nrow = 2, scale = 'free_y')
```




```r
train$old <- ifelse(train$Age < 70, 0, 1) %>% factor()
test$old  <- ifelse(test$Age  < 70, 0, 1) %>% factor()
# survived ~ old | sex
train %>%
  ggplot(aes(x = old, fill = factor(Survived), col = factor(Survived))) +
  geom_bar(position = 'fill') +
  facet_wrap(~ Sex) + ylab('prop')
```



We can conclude that if we define an *old* variable to denote people who are greater than 70 years old, the survival rate will not display significant difference. Therefore, we only expect the cutoff at 18 would be helpful.

(2) Ticket

We already know that the type of *Ticker* is much less than the number of observations.

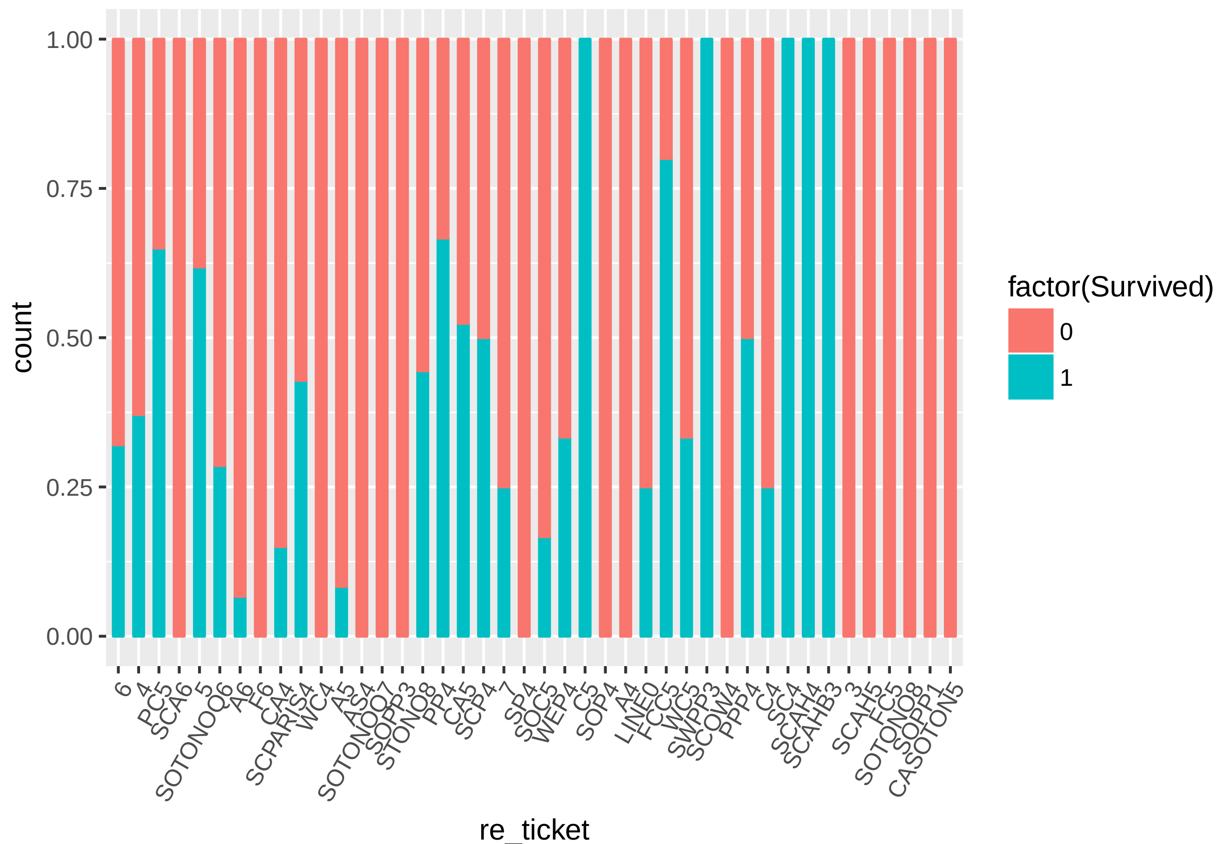
```
all_ticket <- union(unique(train$Ticket) %>% as.character(),
                    unique(test$Ticket) %>% as.character())
c(length(all_ticket), nrow(train) + nrow(test))
```

```
## [1]  929 1309
```

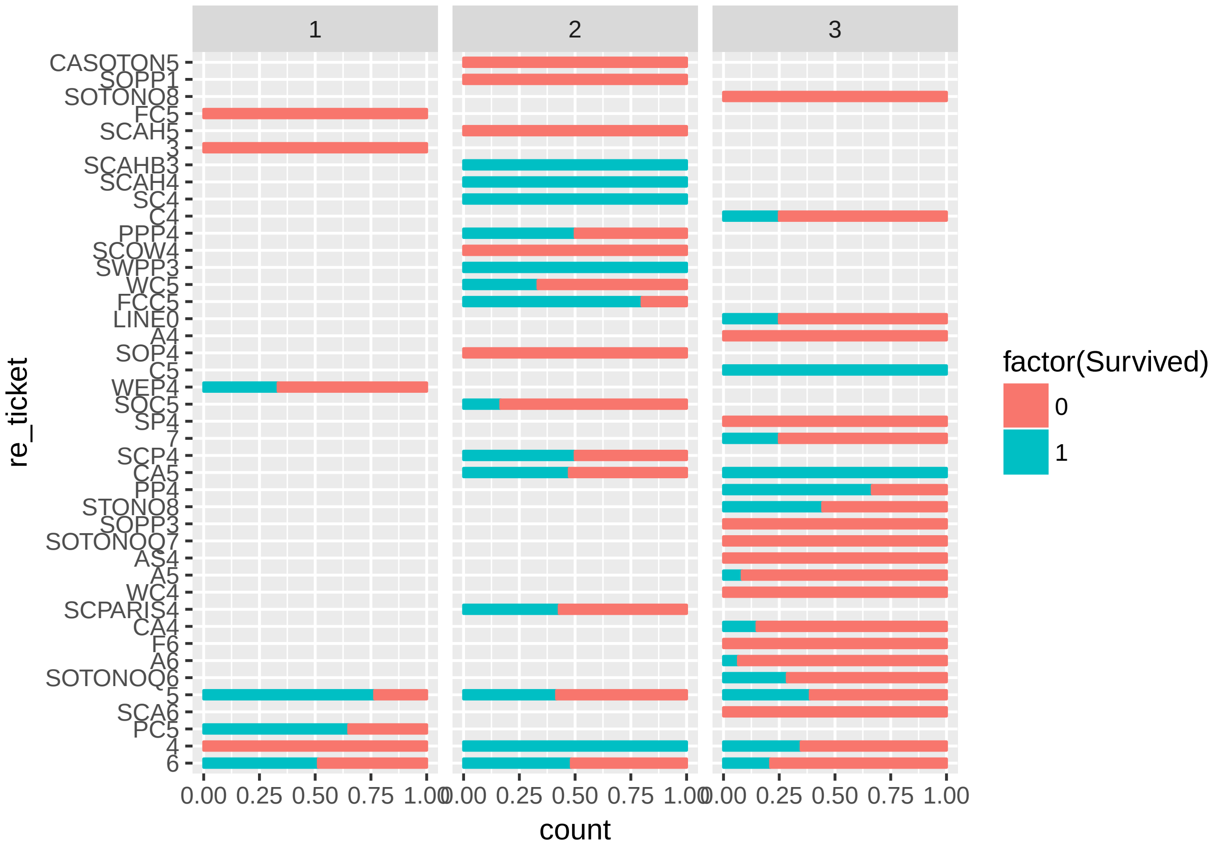
We can see that the *Ticket* are consist of letters and numbers, here for simplicity, we can reconstruct the *Ticket* to be the letter and the number of digits.

```
# extract letters
extract_letters <- function(ticket) {
  letter <- str_extract_all(ticket, '[A-Z]+') %>% '[[' (1)
  return(paste(letter, collapse = ''))
}
# extract numbers
extract_numbers <- function(ticket) {
  number <- str_extract_all(ticket, '[0-9]+') %>% '[[' (1)
  return(nchar(paste(number, collapse = '')))
}
```

```r
# extract
letter <- sapply(c(train$Ticket %>% as.character(),
                   test$Ticket %>% as.character()), extract_letters)
number <- sapply(c(train$Ticket %>% as.character(),
                   test$Ticket %>% as.character()), extract_numbers)
re_ticket <- paste0(letter, number)
# add re_ticket into data
train$re_ticket <- re_ticket[1:nrow(train)]
test$re_ticket <- re_ticket[(nrow(train) + 1):(nrow(test) + nrow(train))]
level <- union(train$re_ticket, test$re_ticket)
train$re_ticket <- factor(train$re_ticket, levels = level)
test$re_ticket <- factor(test$re_ticket, levels = level)
# check the survival rate of different type of ticket
ggplot(train, aes(x = re_ticket, fill = factor(Survived))) +
  geom_bar(aes(col = factor(Survived)),width = 0.5, position = 'fill') +
  theme(axis.text.x = element_text(angle = 60, hjust = 1))
```




```r
ggplot(train, aes(x = re_ticket, fill = factor(Survived))) +
  geom_bar(aes(col = factor(Survived)),width = 0.5, position = 'fill') +
  facet_wrap(~ Pclass, nrow = 1) +
  coord_flip()
```

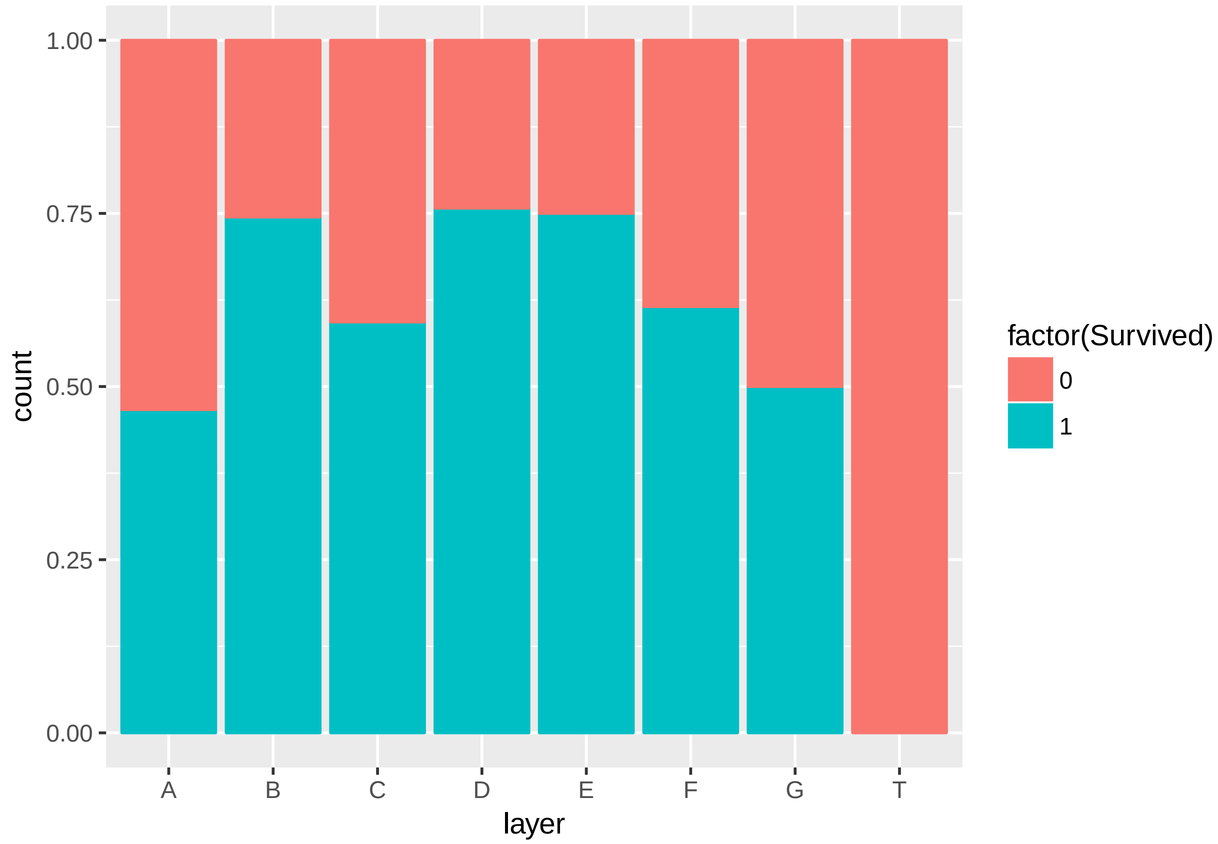


From the 'enhanced' bar plot, We can clearly see that *Survived* has conspicuous differences among different *Ticket*, and we also explore its relationship with *Pclass*.

(3) Cabin

Take a further step, we gonna deal wit *Cabin*. Here is one reference: https://www.kaggle.com/c/titanic/discussion/4693#25690, we can see that there are 7 layers of *Cabin* which is from `A` to `G`.

```
idx_na_cabin <- which(train$Cabin == '')
train[-idx_na_cabin, ] %>%
  mutate(layer = str_sub(Cabin, 1, 1) %>% factor) %>%
  ggplot(aes(x = layer, fill = factor(Survived))) +
  geom_bar(aes(col = factor(Survived)), position = 'fill')
```



Since the number of observations are not large enough, we cannot get a clear pattern about the relationship between *Cabin* and *Survived*.

(4) Family

This idea is from some references and blogs, we can get the Family Id From the *Name* variable.

```
idx_family_train <- which(train$Family == 1)
idx_family_test  <- which(test$Family == 1)
train_family <- train[idx_family_train, ]
test_family  <- test[idx_family_test, ]
train_nfamily <- train[-idx_family_train, ]
test_nfamily  <- test[-idx_family_test, ]
```

First we set the *FamilyId* of non-Family passengers to be 0,

```
train_nfamily$FamilyId <- 0
test_nfamily$FamilyId  <- 0
```

Then we assign FamilyId to the passengers who have Family.

```
family_key <- union(train_family$first_name,
                    test_family$first_name)
family_ref <- train_family %>%
  select(first_name, num_family) %>%
  rbind(test_family[, c('first_name', 'num_family')]) %>%
  group_by(first_name) %>%
```

```r
  summarise(count = n())
family_ref$FamilyId <- 1:nrow(family_ref)
train_family <- train_family %>%
  left_join(family_ref[, c('first_name', 'FamilyId')], by = 'first_name')
test_family  <- test_family %>%
  left_join(family_ref[, c('first_name', 'FamilyId')], by = 'first_name')
# concatenate back
train <- rbind(train_nfamily, train_family)
test  <- rbind(test_nfamily, test_family)
```

Finally, save the data again.

```r
save.image("C:/Users/Bangda/Desktop/kaggle/titanic/eda2.RData")
```

4. Summary

Compared with the original data, we derived 9 extra variables in our final data set which could be as the base to be modeled:

```r
glimpse(train)
```

```
## Observations: 891
## Variables: 21
## $ PassengerId <int> 6, 18, 20, 27, 29, 30, 33, 37, 43, 46, 48, 56, 65,...
## $ Survived    <int> 0, 1, 1, 0, 1, 0, 1, 1, 0, 0, 1, 1, 0, 0, 0, 1, 0,...
## $ Pclass      <int> 3, 2, 3, 3, 3, 3, 3, 3, 3, 3, 3, 1, 1, 3, 3, 3, 3,...
## $ Name        <fctr> Moran, Mr. James, Williams, Mr. Charles Eugene, M...
## $ Sex         <fctr> male, male, female, male, female, male, female, m...
## $ SibSp       <int> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,...
## $ Parch       <int> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,...
## $ Ticket      <fctr> 330877, 244373, 2649, 2631, 330959, 349216, 33567...
## $ Fare        <dbl> 8.4583, 13.0000, 7.2250, 7.2250, 7.8792, 7.8958, 7...
## $ Cabin       <fctr> , , , , , , , , , , , , C52, , , , , , , , , , , ,
## $ Embarked    <fctr> Q, S, C, C, Q, S, Q, C, C, S, Q, S, C, S, S, Q, S...
## $ Family      <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,...
## $ num_family  <int> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,...
## $ first_name  <chr> "Moran", "Williams", "Masselmani", "Emir", "O'Dwye...
## $ title       <fctr> Mr, Mr, Mrs, Mr, Miss, Mr, Miss, Mr, Mr, Mr, Miss...
## $ last_name   <chr> "James", "Charles Eugene", "Fatima", "Farred Cheha...
## $ Age         <dbl> 29.32824, 33.57576, 29.32824, 29.32824, 22.26389, ...
## $ adult       <fctr> 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1...
## $ old         <fctr> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0...
## $ re_ticket   <fctr> 6, 6, 4, 4, 6, 6, 6, 4, 6, SCA6, 5, 5, PC5, 6, 6,...
## $ FamilyId    <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,...
```

Also there are many points that we can expect to make some improvement:
(1) we can use specific packages to deal with missing *Age*, for instance: `mice`;
(2) we could narrow the range of value for *title* variable since some of the categories has only a few observations;
(3) we could use better representation for *FamilyId*;
(4) optimize some chunks of code - more readable and efficient.