Carregamento e Pré-processamento dos Dados

```r
# Carregamento dos dados
data_train = read.csv('train.csv')
data_test = read.csv('test.csv')
gen_sub = read.csv('gender_submission.csv')
head(data_train)
```

```
##   PassengerId Survived Pclass
## 1           1        0      3
## 2           2        1      1
## 3           3        1      3
## 4           4        1      1
## 5           5        0      3
## 6           6        0      3
##                                                  Name    Sex Age SibSp Parch
## 1                             Braund, Mr. Owen Harris   male  22     1     0
## 2 Cumings, Mrs. John Bradley (Florence Briggs Thayer) female  38     1     0
## 3                              Heikkinen, Miss. Laina female  26     0     0
## 4        Futrelle, Mrs. Jacques Heath (Lily May Peel) female  35     1     0
## 5                            Allen, Mr. William Henry   male  35     0     0
## 6                                    Moran, Mr. James   male  NA     0     0
##             Ticket    Fare Cabin Embarked
## 1        A/5 21171  7.2500              S
## 2         PC 17599 71.2833   C85        C
## 3 STON/O2. 3101282  7.9250              S
## 4           113803 53.1000  C123        S
## 5           373450  8.0500              S
## 6           330877  8.4583              Q
```

```r
head(data_test)
```

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

```r
head(gen_sub)
```

```
##   PassengerId Survived
## 1         892        0
## 2         893        1
## 3         894        0
## 4         895        0
## 5         896        1
## 6         897        0
```

Removendo variáveis irrelevantes

As variáveis “Name”, “Ticket” e “Cabin” não serão úteis para a nossa análise.

```r
# Remoção de variáveis irrelevantes
data_train = subset(data_train, select = -c(Name, Ticket, Cabin))
data_test = subset(data_test, select = -c(Name, Ticket, Cabin))
# Combinação dos datasets
data_test$Survived = gen_sub$Survived
data = rbind.fill(data_train, data_test)
head(data)
```

```
##   PassengerId Survived Pclass    Sex Age SibSp Parch    Fare Embarked
## 1           1        0      3   male  22     1     0  7.2500        S
## 2           2        1      1 female  38     1     0 71.2833        C
## 3           3        1      3 female  26     0     0  7.9250        S
## 4           4        1      1 female  35     1     0 53.1000        S
## 5           5        0      3   male  35     0     0  8.0500        S
## 6           6        0      3   male  NA     0     0  8.4583        Q
```

Imputação de valores faltantes

```r
# Identificação de valores irregulares e torná-los NA
data[data == ''] = NA

summary(is.na(data)) # 263 NAs em Age / 1 NA em Fare / 2 NA em Embarked
```

```
##  PassengerId      Survived         Pclass           Sex
##  Mode :logical   Mode :logical   Mode :logical   Mode :logical
##  FALSE:1309      FALSE:1309      FALSE:1309      FALSE:1309
##
##     Age            SibSp           Parch            Fare
##  Mode :logical   Mode :logical   Mode :logical   Mode :logical
##  FALSE:1046      FALSE:1309      FALSE:1309      FALSE:1308
##  TRUE :263                                       TRUE :1
##   Embarked
##  Mode :logical
##  FALSE:1307
##  TRUE :2
```

Temos alguns dados faltantes: 263 na variável “Age”, 1 na variável “Fare” e 2 na variável “Embarked”.

Imputação da variável Embarked

Como Embarked é uma variável categórica, vamos imputar com a moda.

```r
table(data$Embarked)
```

```
##
##   C   Q   S
## 270 123 914
```

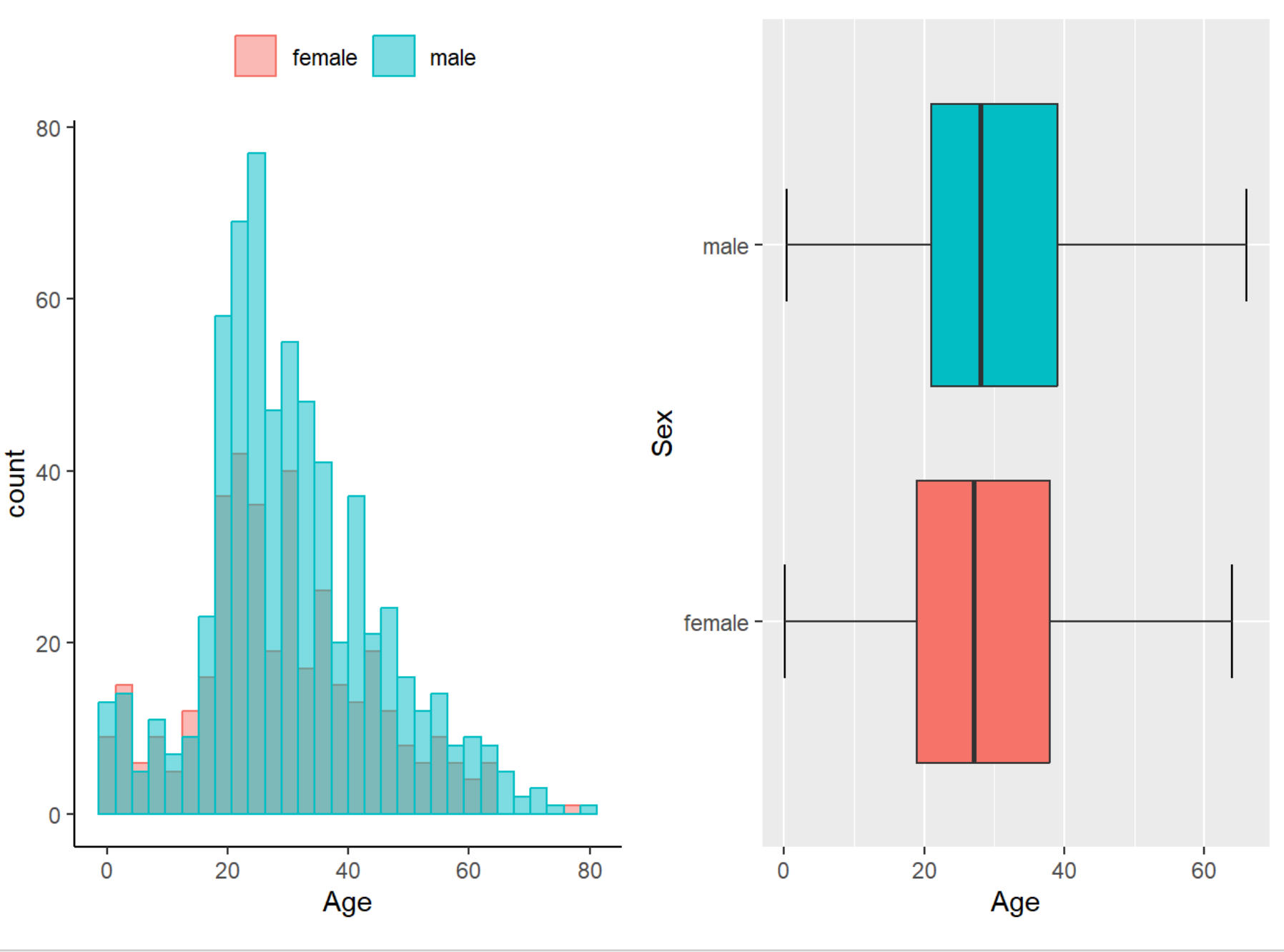
Moda = S

```r
data$Embarked = replace_na(data$Embarked, 'S')
```

Imputação da variável Age

Como a variável Age é contínua vamos analisar se podemos imputar a mediana. Iremos verificar a distribuição dos dados para cada categoria da variável Sex sem os outliers. Caso as distribuições de Age em cada categoria de Sex forem diferentes, não podemos imputar com a mediana.

```r
plt1 = ggplot(data, aes(x = Age, color = Sex, fill = Sex)) +
  geom_histogram( position="identity", alpha = 0.5) +
  theme_classic()+ theme(legend.position="top") +
  scale_fill_discrete(name = "") +
  scale_color_discrete(name = "")

plt2 = ggplot(data, aes(x = Age, y = Sex, fill = Sex)) +
  stat_boxplot(geom = 'errorbar', width = 0.3) +
  geom_boxplot(outliers = F) +
  theme(legend.position = 'none')

grid.arrange(plt1, plt2, ncol=2)
```



```r
AgeMale = data['Age'][data['Sex'] == 'male']
AgeFemale = data['Age'][data['Sex'] == 'female']
ks.test(AgeMale, AgeFemale)
```

```
##
##  Asymptotic two-sample Kolmogorov-Smirnov test
##
## data:  AgeMale and AgeFemale
## D = 0.084636, p-value = 0.06059
## alternative hypothesis: two-sided
```

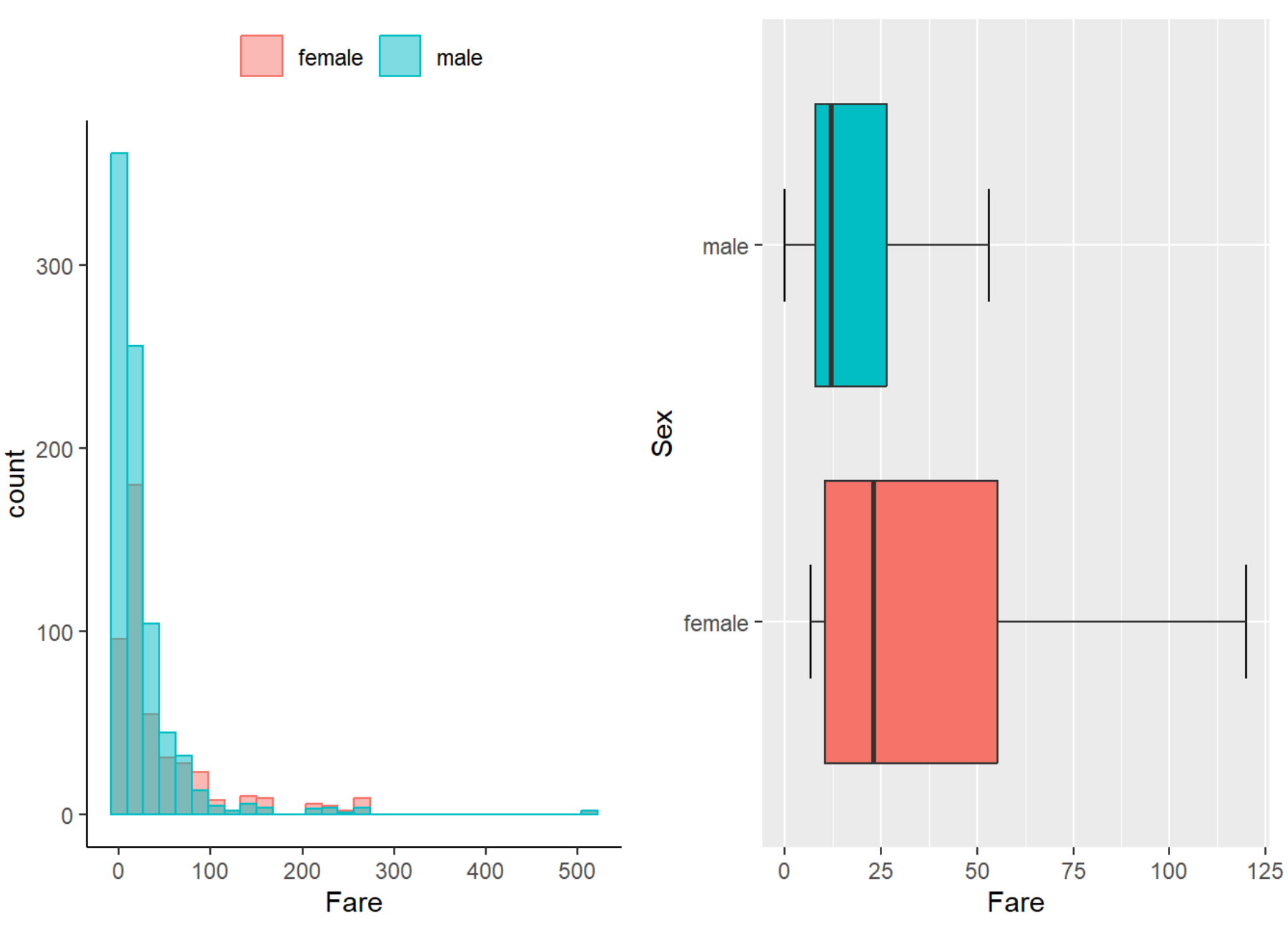
O teste Kolmogorov-Smirnov de duas amostras não encontrou evidências para afirmarmos que as distribuições são diferentes, portanto podemos imputar os dados faltantes de Age com a mediana.

```r
data$Age = replace_na(data$Age, median(data$Age, na.rm = T))
```

Imputação da variável Fare

Faremos o mesmo que fizemos na variável Age, na variável Fare.

```r
plt1 = ggplot(data, aes(x = Fare, color = Sex, fill = Sex)) +
  geom_histogram( position="identity", alpha = 0.5) +
  theme_classic()+ theme(legend.position="top") +
  scale_fill_discrete(name = "") +
  scale_color_discrete(name = "")

plt2 = ggplot(data, aes(x = Fare, y = Sex, fill = Sex)) +
  stat_boxplot(geom = 'errorbar', width = 0.3) +
  geom_boxplot(outliers = F) +
  theme(legend.position = 'none')

grid.arrange(plt1, plt2, ncol=2)
```



```r
FareMale = data['Fare'][data['Sex'] == 'male']
FareFemale = data['Fare'][data['Sex'] == 'female']
ks.test(FareMale, FareFemale)
```

```
##
##  Asymptotic two-sample Kolmogorov-Smirnov test
##
## data:  FareMale and FareFemale
## D = 0.24152, p-value = 1.267e-15
## alternative hypothesis: two-sided
```

O teste de Kolmogorov-Smirnov para duas amostras encontrou evidências para afirmarmos que as distribuições de Fare para cada nível de Sex são diferentes. Vamos utilizar MICE para imputar na observação faltante já que não há problemas em usar.
Antes de utilizarmos o MICE precisamos remover as variáveis que não podemos usar (PassengerId, pois é somente um índice; e Survived, pois é nossa variável resposta, logo comprometeria nosso modelo preditivo) e colocar as variáveis categóricas (Sex, Pclass e Embarked) como dummies.

```r
# Criação de variáveis dummy
data = dummy_cols(
  data,
  select_columns = c('Sex', 'Pclass', 'Embarked'),
  remove_first_dummy = TRUE,
  remove_selected_columns = TRUE
)

# Imputação final com MICE para Fare
data_aux = data[,-c(1,2)]
set.seed(28051996)
imp = mice(data_aux)
data_aux = complete(imp, 1)

# Reconstrução do dataset final
data = cbind(data[,c(1,2)], data_aux)
data_train = data[1:891,-1]
data_test = data[892:1309,]
```

Modelo preditivo usando random forest

Criação do modelo

```r
# Modelo inicial
set.seed(28051996)
model = randomForest(factor(Survived) ~ ., data = data_train, proximity = T)
```

Validação do modelo

Precisamos validar nosso modelo, portanto faremos a cross validation utilizando 5 Kfolds estratificados para diminuir a variabilidade dentro dos subdatasets, dando maior confiabilidade à avaliação do modelo.

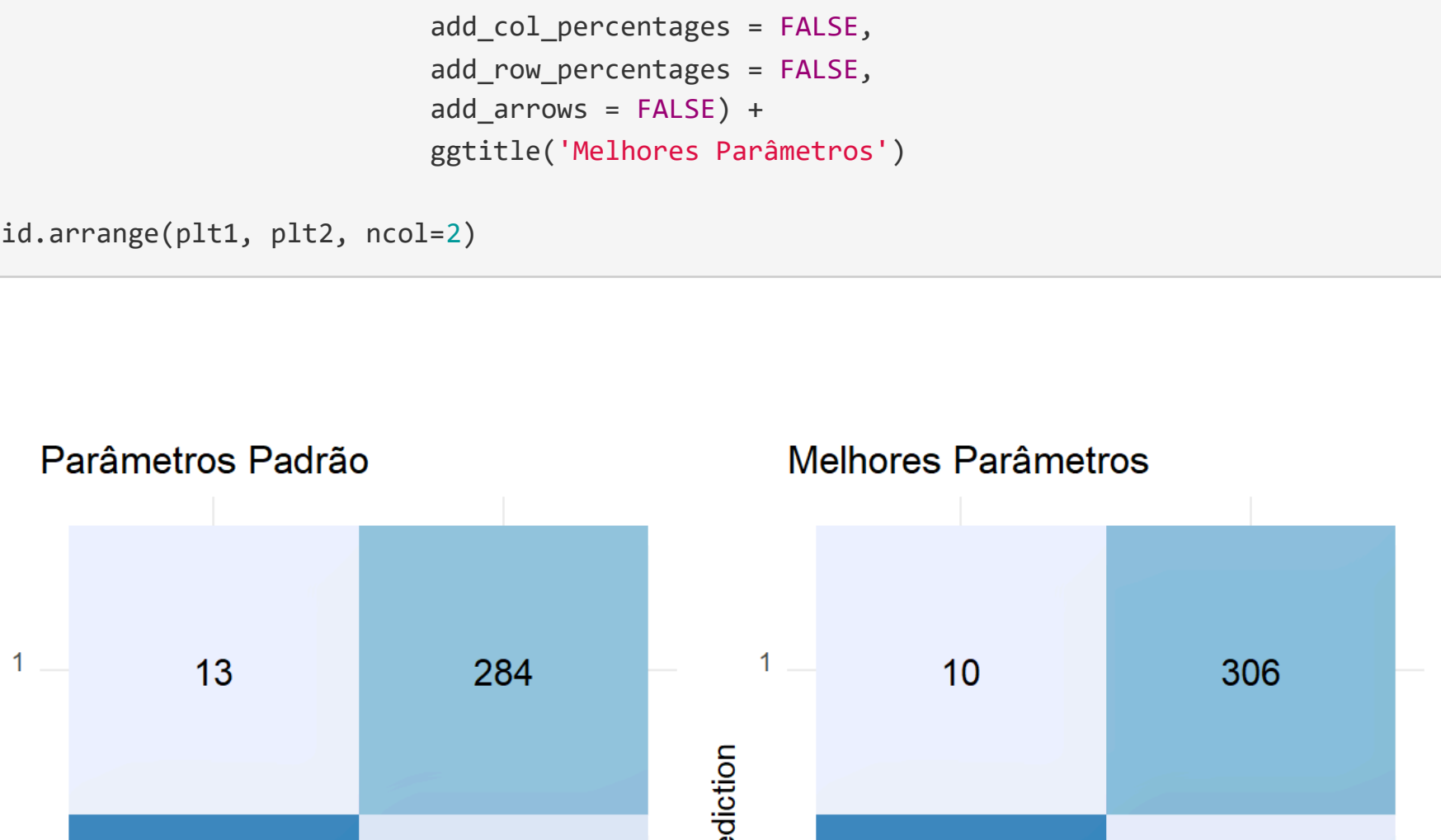
```r
# Validação cruzada
set.seed(28051996)
train_control = trainControl(
  method = 'cv',
  number = 5, # 5 kfolds
  search = 'random', # otimização aleatória
  classProbs = TRUE,
  summaryFunction = twoClassSummary
)

# Função customizada de métricas
customSummary = function(data, lev = NULL, model = NULL) {
  out = twoClassSummary(data, lev, model)
  tp = sum(data$obs == lev[2] & data$pred == lev[2]) # true positive
  fp = sum(data$obs == lev[1] & data$pred == lev[2]) # false positive
  out = c(out, Precision = tp/(tp+fp))
  return(out)
}

train_control$summaryFunction = customSummary

# Ajuste dos rótulos para validação cruzada
data_train_aux = data_train
data_train_aux$Survived <- factor(
  data_train_aux$Survived,
  levels = c(0, 1),
  labels = c("No", "Yes")
)

set.seed(28051996)
cv_best_model = train(
  factor(Survived) ~ .,
  data = data_train_aux,
  trControl = train_control,
  method = 'rf',
  metric = 'ROC',
)
```

O melhor valor para o hiperparâmetro mtry é 4.

Avaliação do Modelo

Vamos plotar as matrizes de confusão para avaliar a otimização do modelo

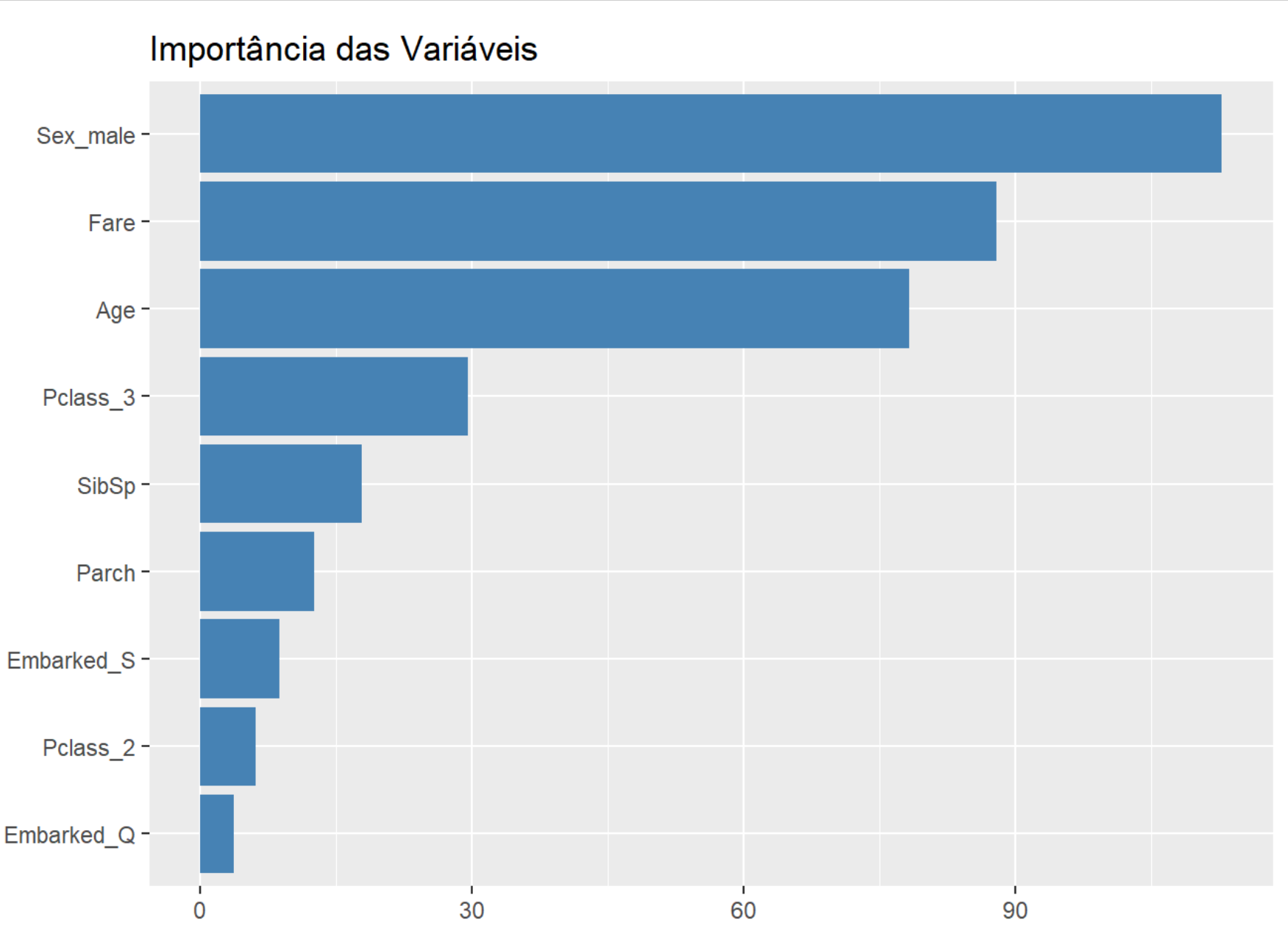
```r
# Matrizes de confusão
y_hat_train = predict(model, data_train)
cm_train = confusionMatrix(y_hat_train, factor(data_train$Survived), mode = 'everything')
cm_train_df = as.data.frame(cm_train$table)
names(cm_train_df) <- c("Prediction", "Reference", "N")

y_hat_best_train = factor(ifelse(predict(cv_best_model, data_train) == 'No', 0, 1))
cm_best_train = confusionMatrix(y_hat_best_train, factor(data_train$Survived), mode = 'everything')
cm_best_train_df = as.data.frame(cm_best_train$table)
names(cm_best_train_df) <- c("Prediction", "Reference", "N")

# Visualização comparativa
plt1 = plot_confusion_matrix(cm_train_df,
                             target_col = "Reference",
                             prediction_col = "Prediction",
                             palette = "Blues",
                             place_x_axis_above = F,
                             add_normalized = FALSE,
                             rotate_y_text = F,
                             add_col_percentages = FALSE,
                             add_row_percentages = FALSE,
                             add_arrows = FALSE) +
                             ggtitle('Parâmetros Padrão')

plt2 = plot_confusion_matrix(cm_best_train_df,
                             target_col = "Reference",
                             prediction_col = "Prediction",
                             palette = "Blues",
                             place_x_axis_above = F,
                             add_normalized = FALSE,
                             rotate_y_text = F,
                             add_col_percentages = FALSE,
                             add_row_percentages = FALSE,
                             add_arrows = FALSE) +
                             ggtitle('Melhores Parâmetros')

grid.arrange(plt1, plt2, ncol=2)
```



Ocorreu uma boa melhora nas predições do modelo.

Importância das variáveis

```r
# Importância das variáveis
imp_model = importance(cv_best_model$finalModel)
ggplot(data = data.frame(Variable = rownames(imp_model), Importance = imp_model[,1]),
       aes(x = reorder(Variable, Importance), y = Importance)) +
       geom_col(fill = 'steelblue') +
       coord_flip() +
       ggtitle('Importância das Variáveis')
```



Temos 3 variáveis que suas importâncias se sobressaem em relação às outras: Age, Fare e Sex

Previsões no Conjunto de Teste

```r
y_hat_best_test = factor(ifelse(predict(cv_best_model, data_test) == 'No', 0, 1))
cm_best_test = confusionMatrix(y_hat_best_test, factor(gen_sub$Survived), mode = 'everything')
cm_best_test_df = as.data.frame(cm_best_test$table)
names(cm_best_test_df) <- c("Prediction", "Reference", "N")

plot_confusion_matrix(cm_best_test_df,
                      target_col = "Reference",
                      prediction_col = "Prediction",
                      palette = "Blues",
                      place_x_axis_above = F,
                      add_normalized = FALSE,
                      rotate_y_text = F,
                      add_col_percentages = FALSE,
                      add_row_percentages = FALSE,
                      add_arrows = FALSE) +
                      ggtitle('Desempenho no Conjunto de Teste')
```