Análise exploratória de dados — Dataset Titanic

12 min read · Jun 26, 2023

Ana Paula Ferrari Januário Follow

Share

Conforme mencionado anteriormente, cumpriremos nossa promessa de realizar uma análise exploratória de dados sobre os arquivos contendo informações do Titanic. Estes conjuntos de dados estão disponíveis no kaggle e foram utilizados em uma competição envolvendo Machine Learning. Antes de aplicar qualquer técnica, é crucial enfatizar que a análise de dados segue seis passos, conforme listados abaixo:

Os 6 passos da análise de dados

Considerando os passos de uma análise de dados, como já recebemos os dados podemos prosseguir para a definição do problema, o processamento e a limpeza dos dados, a análise propriamente dita e a criação de alguns gráficos para compartilhar as informações.

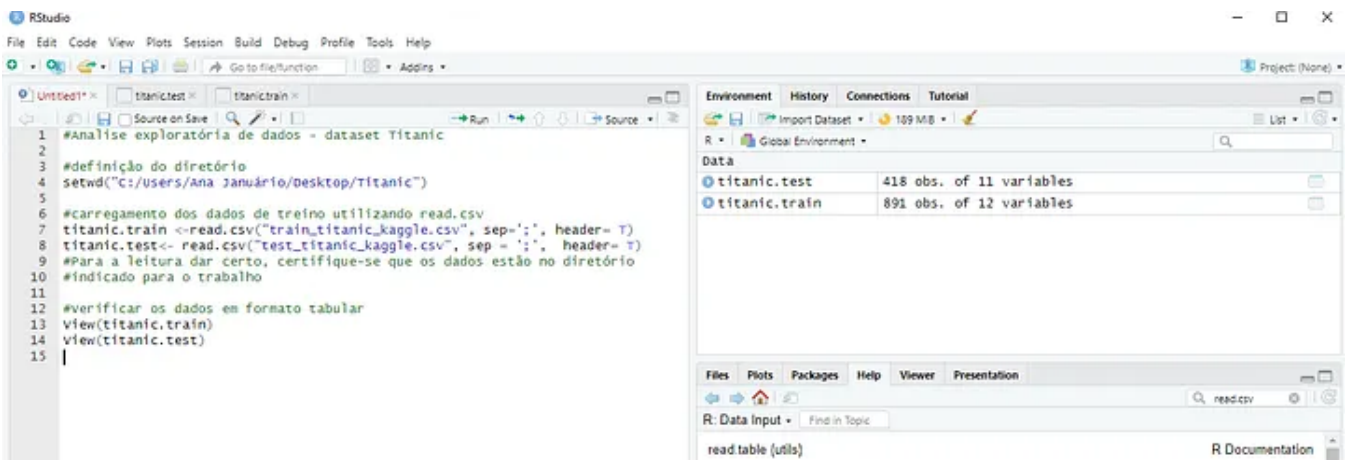
Nesta fase da análise exploratória, como não estamos familiarizados com os datasets, podemos sugerir algumas etapas essenciais. Primeiramente, é importante definir a questão que queremos responder, compreender o significado das variáveis

presentes, avaliar o formato dos dados e obter informações básicas sobre eles. Além disso, é recomendado verificar se existem classes dentro do conjunto de dados e explorar alguns conceitos estatísticos relacionados a essas classes.

Com essas considerações em mente, procederemos ao carregamento dos dados no RStudio. Nesta fase você já deve ter percebido que temos dois dois conjuntos de dados: um de treino e outro de teste. Essa divisão dos dados é comum em machine learning para realizar a avaliação das taxas de acertos dos algoritmos utilizados.

```
#definição do diretório
setwd("C:/Users/Ana Januário/Desktop/Titanic")

#carregamento dos dados de treino utilizando read.csv
titanic.train <-read.csv("train_titanic_kaggle.csv", sep=';', header= T)
titanic.test<- read.csv("test_titanic_kaggle.csv", sep = ';',  header= T)
#Para a leitura dar certo, certifique-se que os dados estão no diretório
#indicado para o trabalho

#Verificar os dados em formato tabular
View(titanic.train)
View(titanic.test)
```

Ao examinar os dados, você notará que o conjunto de dados de teste possui uma coluna a menos em comparação com o conjunto de dados de treinamento. Se você leu a página do Kaggle, já deve saber que essa coluna ausente é a variável “Survived”, que indica se a pessoa sobreviveu ao naufrágio ou não. No entanto, caso não tivéssemos essa informação, como poderíamos proceder?



Uma abordagem inicial seria verificar se existem colunas em comum nos dois conjuntos de dados. Podemos fazer isso manualmente, ou seja, comparando o nome das colunas de cada conjunto de dados individualmente. Outra abordagem mais eficiente é utilizar um método automático por meio da criação de um loop. Vamos explorar essas duas opções separadamente.

```
# Verificar se as duas listas de nomes de colunas são iguais,
# Metodo do olhometro - manual
names(titanic.train)
names(titanic.test)

##console
> names(titanic.train)
 [1] "PassengerId" "Survived"    "Pclass"      "Name"        "Sex"
 [6] "Age"         "SibSp"       "Parch"       "Ticket"      "Fare"
[11] "Cabin"       "Embarked"
> names(titanic.test)
 [1] "PassengerId" "Pclass"      "Name"        "Sex"         "Age"
 [6] "SibSp"       "Parch"       "Ticket"      "Fare"        "Cabin"
[11] "Embarked"
```

No método manual, ao compararmos os nomes das colunas, podemos observar que as variáveis têm os mesmos nomes, com exceção de uma coluna faltante. No entanto, esse método é propenso a erros, portanto, optaremos por realizar uma verificação automática usando um loop. Primeiramente, criaremos duas variáveis contendo os nomes das colunas dos conjuntos de treinamento e teste. Em seguida, construiremos uma função para verificar se o nome da coluna do conjunto de treinamento é idêntico ao nome das colunas do conjunto de teste. Por fim, solicitaremos a impressão dos nomes das colunas de treinamento, juntamente com o resultado do teste.

```
# Criar uma variável com os nomes das colunas de cada dataset
train_columns <- colnames(titanic.train)
test_columns <- colnames(titanic.test)

# Imprimir o resultado para cada nome de coluna
for (column_name in train_columns) {
  identical_name <- column_name %in% test_columns
  print(paste(column_name, identical_name, sep = ": "))
}
```

```
##Console
[1] "PassengerId: TRUE"
[1] "Survived: FALSE"
[1] "Pclass: TRUE"
[1] "Name: TRUE"
[1] "Sex: TRUE"
[1] "Age: TRUE"
[1] "SibSp: TRUE"
[1] "Parch: TRUE"
[1] "Ticket: TRUE"
[1] "Fare: TRUE"
[1] "Cabin: TRUE"
[1] "Embarked: TRUE"
```

Portanto, podemos confirmar que a variável “Survived” não está presente no conjunto de dados de teste. Para combinar os dois conjuntos de dados, é necessário lidar com esse detalhe. Na competição do Kaggle, adicionei a coluna “Survived” aos dados de teste e a preenchi com valores “NA” (em um estágio inicial). No entanto, nesta análise exploratória, não faz sentido ter uma coluna repleta de valores “NA”. Portanto, vamos construir um novo dataframe a partir dos dados de treinamento, excluindo a variável “Survived” (pessoal do Kaggle, por favor, não me julguem!).

```
#vamos remover a coluna survived do dataset de treino para esta analise inicial
df.train<- titanic.train[, -which(names(titanic.train) == "Survived")]
```

Pronto! Agora vamos adicionar uma coluna que será útil caso queiramos separar os conjuntos de dados novamente. Para isso, criaremos uma coluna chamada “isTrainSet”, que indicará se as informações são provenientes do conjunto de treinamento ou de teste. Feito isso, poderemos unir os dois conjuntos de dados usando a função “rbind”.

```
#indicação de qual dataset a informação é proveniente
df.train$isTrainSet <- TRUE
titanic.test$isTrainSet <- FALSE

#combinar os dados
titanic.full<- rbind(df.train, titanic.test)

#verificar se o numero de linhas por dataset está igual
```

```
table(titanic.full$isTrainSet)

##Console
> table(titanic.full$isTrainSet)

FALSE  TRUE
  418   891
```

Agora sim, temos um único dataset com 1309 observações e 12 colunas, podemos começar a avaliar as nossas variáveis. Para isso vamos usar a função “str”, que nos dá a estrutura de cada coluna.

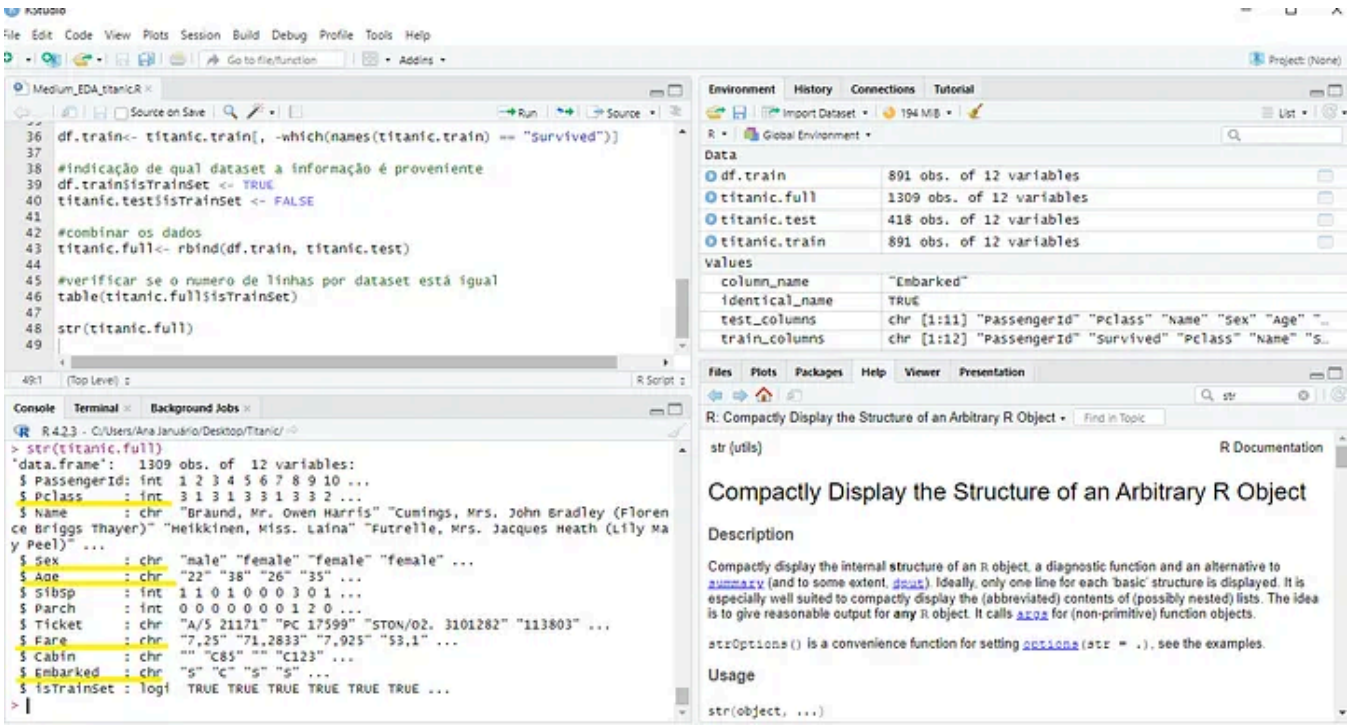
Estrutura e tratamento das variáveis

Ao avaliar inicialmente as variáveis, observamos que a coluna “Age” (idade) está definida como um caractere (chr), ou seja, como uma string, assim como a variável “Fare” (tarifa), que indica o valor pago pela passagem. No entanto, essas variáveis são numéricas e, portanto, estão mal estruturadas. Além disso, as variáveis “Sex” e “Embarked” que também estão como string, podem ser tratadas como fator.

Por outro lado a variável “PClass” (classe) que indica o status económico do passageiro, está definida como numérica. No entanto, na realidade, trata-se de uma variável categórica.

Para aqueles que não estão familiarizados ou não se recordam, no R, uma coluna do tipo “character” (string) contém valores de texto ou sequências de caracteres. Esses valores são tratados como dados categóricos e são armazenados como vetores de caracteres. Por outro lado, uma coluna do tipo “fator” é uma variável categórica que representa um conjunto fixo de valores possíveis. Esses valores são armazenados como inteiros e mapeados para rótulos de texto. Isso é útil quando se lida com dados categóricos, onde a ordem ou o significado específico dos valores é importante. A

principal vantagem do uso de fatores é que eles economizam memória e podem melhorar o desempenho em algumas operações, especialmente quando há um grande número de repetições dos valores.



Estrutura dos dados e avaliação da colunas que precisam de ser modificadas

Vamos iniciar as conversões necessárias, a começar pela idade. No início, logo quando executamos a conversão, recebemos um aviso de “NA” introduzido por coerção, o que significa que ainda precisamos lidar com esses valores ausentes mais adiante.

```
## Console
> titanic.full$Age<- as.numeric(titanic.full$Age)
Warning message:
NAs introduced by coercion
```

Um detalhe importante a ser considerado é na variável “Tarifa” (Fare). Ao observarmos sua estrutura com o comando “str”, percebemos que os decimais estão sendo separados por “,” em vez de “.”. Isso pode causar problemas ao converter a variável para o formato numérico, uma vez que no R os valores decimais são separados por “.”. Portanto, antes de realizar a conversão, devemos substituir todas as ocorrências de “,” por “.”.

```
#A variável Fare tem um porém, nela os valores usam , ou inves de .
#temos que substitui-los
titanic.full$Fare <- gsub(",", ".", titanic.full$Fare)
titanic.full$Fare<- as.numeric(titanic.full$Fare)
```

Após tratar as variáveis numéricas, vamos proceder com as variáveis categóricas. Faremos a conversão para o formato de fator e, em seguida, começaremos a avaliar os casos de valores ausentes. No caso das variáveis do tipo “character” (string), podemos utilizar o comando “table” para verificar quais classes estão presentes em cada variável, bem como o número de observações em cada classe. Isso nos ajudará a ter uma visão geral dos dados e identificar possíveis valores omissos.

```
#tratar a variável sexo
table(titanic.full$Sex) #não temos NA, nem problemas de escrita

##console
female   male
   466    843

titanic.full$Sex<- as.factor(titanic.full$Sex)

#Variavel Embarked
table(titanic.full$Embarked)

      C   Q   S
  2 270 123 914
# Nesse caso também não temos problema de escrita,
# mas verificamos uma classe de NA
# Neste caso, preenchemos os valores omisssos com a classe com maior n de obser
titanic.full[titanic.full$Embarked== '', "Embarked"]<-'S'
table(titanic.full$Embarked)

  C   Q   S
270 123 916

titanic.full$Embarked<- as.factor(titanic.full$Embarked)

#Variável PClass
#essa variável pode ser interpretada como primeira classe, segunda classe e ter
#então vamos apenas coloca-la como factor
titanic.full$Pclass<- as.factor(titanic.full$Pclass)
table(titanic.full$Pclass)
```

```
  1   2   3
323 277 709
```

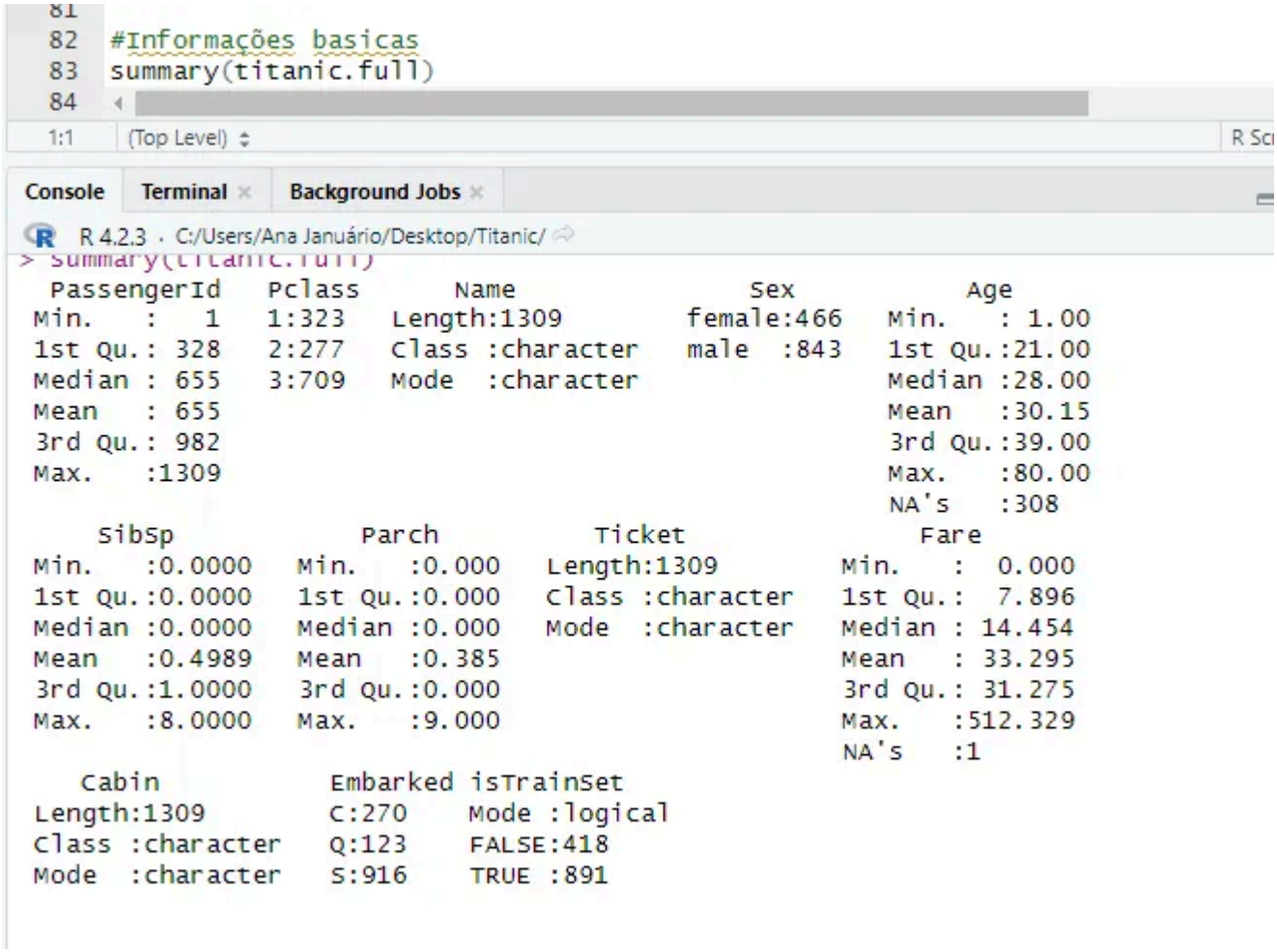
Após o tratamento das variáveis, podemos observar a presença de valores ausentes em cada uma delas utilizando as funções “colSums” e “is.na”.

```
#verificando NAs
colSums(is.na(titanic.full))
PassengerId      Pclass        Name         Sex         Age       SibSp
          0           0           0           0         308           0
      Parch      Ticket        Fare       Cabin    Embarked  isTrainSet
          0           0           1           0           0           0
```

Antes de aplicar qualquer técnica de modelagem estatística ou de aprendizado de máquina, é necessário lidar com os valores ausentes. Podemos optar por removê-los dos dados, mas essa abordagem não é recomendada, pois resulta na perda de informações. Outra opção é substituí-los pela média ou mediana dos valores presentes na variável. Além disso, podemos até mesmo criar um modelo de regressão para preencher cada um dos valores ausentes. No entanto, por enquanto, vamos deixar de lado a questão dos valores ausentes e prosseguir com a análise exploratória dos dados.

Análise Exploratória

Agora que os dados foram tratados de forma preliminar, podemos prosseguir com a análise exploratória. Um bom ponto de partida é obter um resumo estatístico do conjunto de dados usando a função “summary”. Para variáveis numéricas, essa função fornecerá informações como valores mínimo, máximo, média, mediana, quartis e a contagem de valores ausentes. Já para variáveis categóricas em formato de string, ela mostrará as classes presentes e o comprimento. No caso de variáveis categóricas no formato de fatores, a função “summary” exibirá os níveis de cada fator e a contagem de observações em cada nível.



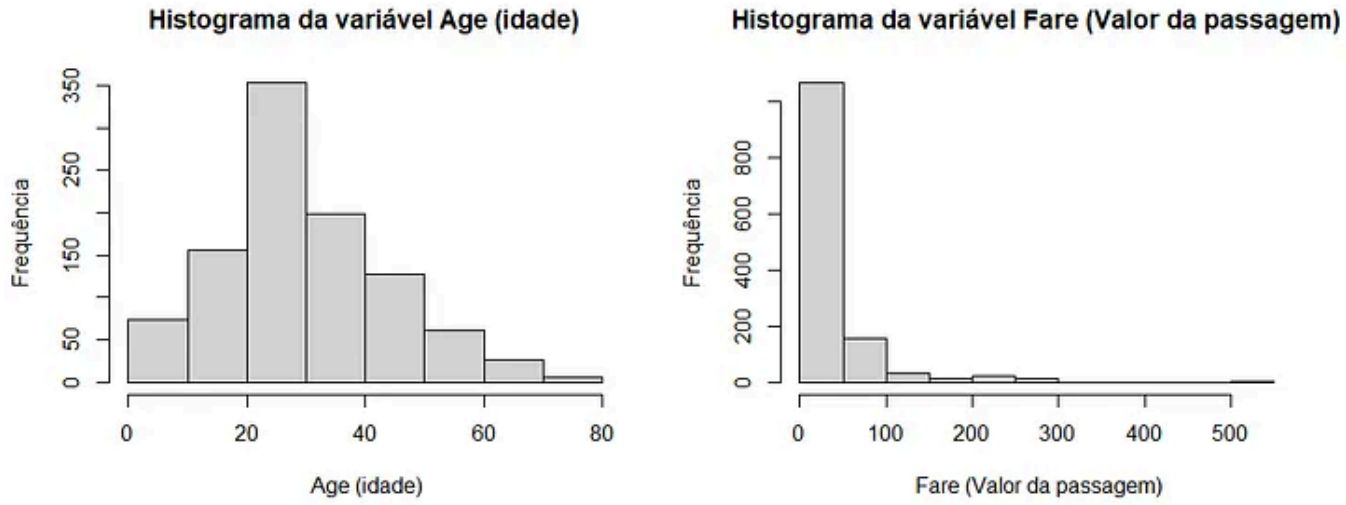
Informações estatísticas fornecidas pela função summary

Isso já nos proporciona uma visão geral bastante útil, não é mesmo? No entanto, caso você precise de informações adicionais, como o desvio padrão ou a variância de uma variável numérica, você pode usar a função “stdev” disponível na biblioteca base do R. Alternativamente, você pode baixar o pacote “fBasics” e utilizar a função “basicStats” para obter estatísticas básicas mais abrangentes.

```
#Vamos supor que eu queira saber o desvio padrão da idade
#podemos usar a função stdev
# mas atenção que essa função não tolera omissos
stdev(titanic.full$Age, na.rm=TRUE)

#ou ainda podemos utilizar o pacote fBasics e usufruir da função basicStats
install.packages("fBasics")
library(fBasics)
basicStats(titanic.full$Age)

##Console
 X..titanic.full.Age
nobs                1309.000000
NAs                  308.000000
Minimum                1.000000
```

```
Maximum              80.000000
1. Quartile          21.000000
3. Quartile          39.000000
Mean                 30.147852
Median               28.000000
Sum                30178.000000
SE Mean               0.448175
LCL Mean             29.268380
UCL Mean             31.027324
Variance            201.062118
Stdev                14.179637
Skewness              0.457463
Kurtosis              0.133755
```

Após obter o resumo estatístico do conjunto de dados, podemos avançar para uma análise univariada das variáveis numéricas, utilizando recursos visuais para uma melhor compreensão da distribuição dos valores em cada variável. Uma ferramenta visual útil nesse contexto é o histograma. Ao observarmos os histogramas, podemos notar que nenhuma das duas variáveis parece seguir uma distribuição normal. Para confirmar essa suposição, realizamos os testes de normalidade de Shapiro-Wilk e Kolmogorov-Smirnov, disponíveis no pacote “nortest”. Esses testes são importantes, pois em muitos modelos estatísticos, a suposição de normalidade dos dados é considerada um requisito fundamental.



Histograma das variáveis age e fare

```
#Histograma
par(mfrow = c(1,2)) #comando para apresentar os gráficos lado a lado
```

```
hist(titanic.full$Age, main = "Histograma da variável Age (idade)",
     xlab= " Age (idade)", ylab= "Frequência")

hist(titanic.full$Fare, main = "Histograma da variável Fare (Valor da passagem)
     xlab= " Fare (Valor da passagem)", ylab= "Frequência")

#teste de normalidade
#install.packages("nortest") #caso não tenha instalado
library(nortest) #chama o pacote

shapiro.test(titanic.full$Age) #teste de shapiro-wilk
lillie.test(titanic.full$Age) #teste de Kolmogorov-Smirnov com a correção de Li

##Console

> shapiro.test(titanic.full$Age)

 Shapiro-Wilk normality test

data:  titanic.full$Age
W = 0.9774, p-value = 2.318e-11

> lillie.test(titanic.full$Age)

 Lilliefors (Kolmogorov-Smirnov) normality test

data:  titanic.full$Age
D = 0.086577, p-value < 2.2e-16
```

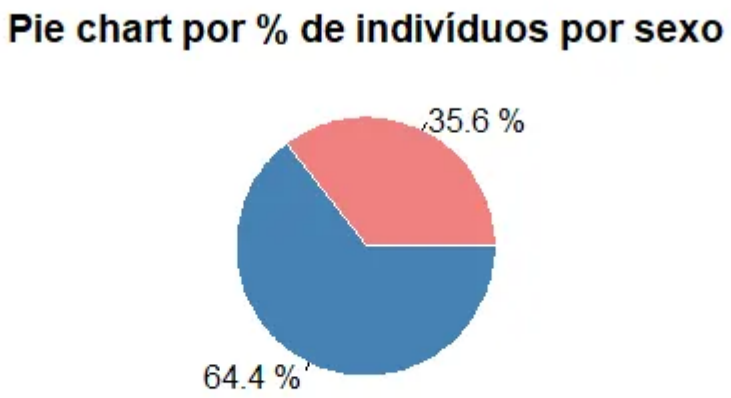
Após a análise univariada das variáveis numéricas, vamos agora avaliar as variáveis categóricas. Para isso, iremos observar as quantidades de homens e mulheres presentes no conjunto de dados, e também, observar o número de indivíduos por classe e porto de embarque.

Quando lidamos com variáveis categóricas que possuem duas ou três classes, é comum utilizarmos pie charts para representá-las. Entretanto o pie chart em R exige alguns parâmetro que para quem esta iniciando pode não ser tão claro. Podemos abordar gráficos, pacotes de gráficos e parâmetros em outro momento. O importante a observar nesta fase é a existência de variáveis desbalanceadas, ou seja, quando temos um número significativamente maior de indivíduos em uma classe do que em outra. Para verificar isso, realizamos uma avaliação visual dos gráficos e, de forma mais formal, aplicamos o teste do qui-quadrado.

O teste do qui-quadrado é uma ferramenta estatística utilizada para avaliar se há associação entre duas variáveis categóricas. Mas nesse caso, utilizaremos o teste para verificar se há desequilíbrio significativo na distribuição dos indivíduos entre as classes das variáveis categóricas em análise.

```
##resultado console
> table_sex<-table(titanic.full$Sex)
> table_sex

female   male
   466    843
> table_percent<-round(table_sex/sum(table_sex)*100, 2) #vamos apresentar em pe
> #Definir cores para cada categoria
> cores <- c("female" = "#F08080", "male" = "#4682B4")
> #produzir o piechart com a função pie
> pie(table_sex, col = cores, radius=1, lty=1, border="white",
    labels = paste(table_percent, "%"), main = "Pie chart por % de indivíduos p
> # Adicionar a legenda ao gráfico
> legend("topright", legend = names(table_sex), fill = cores, title = "Sex")
> #teste estatistico para verificar se as frequências observadas
> #são significativamente diferentes das frequências esperadas.
> qui.quad<-chisq.test(table_sex)
> qui.quad

 Chi-squared test for given probabilities

data:  table_sex
X-squared = 108.58, df = 1, p-value < 2.2e-16
> qui.quad$residuals

   female      male
-7.368117  7.368117
```

O pie chart resultante é este:



Pie chart gerado pelo comando pie()

Como podemos verificar, a variável Sex apresenta um desequilíbrio significativo, com um maior número de homens do que mulheres no conjunto de dados. Além disso, ao utilizar o comando “pie” para criar o pie chart, notamos que o resultado não é visualmente agradável e pode ser um pouco trabalhoso de ajustar. No entanto, existem outros pacotes e parâmetros disponíveis que permitem a criação de pie charts mais esteticamente atraentes. Se você estiver interessado em explorar essas opções, pode ver este site aqui para obter mais informações e exemplos práticos.

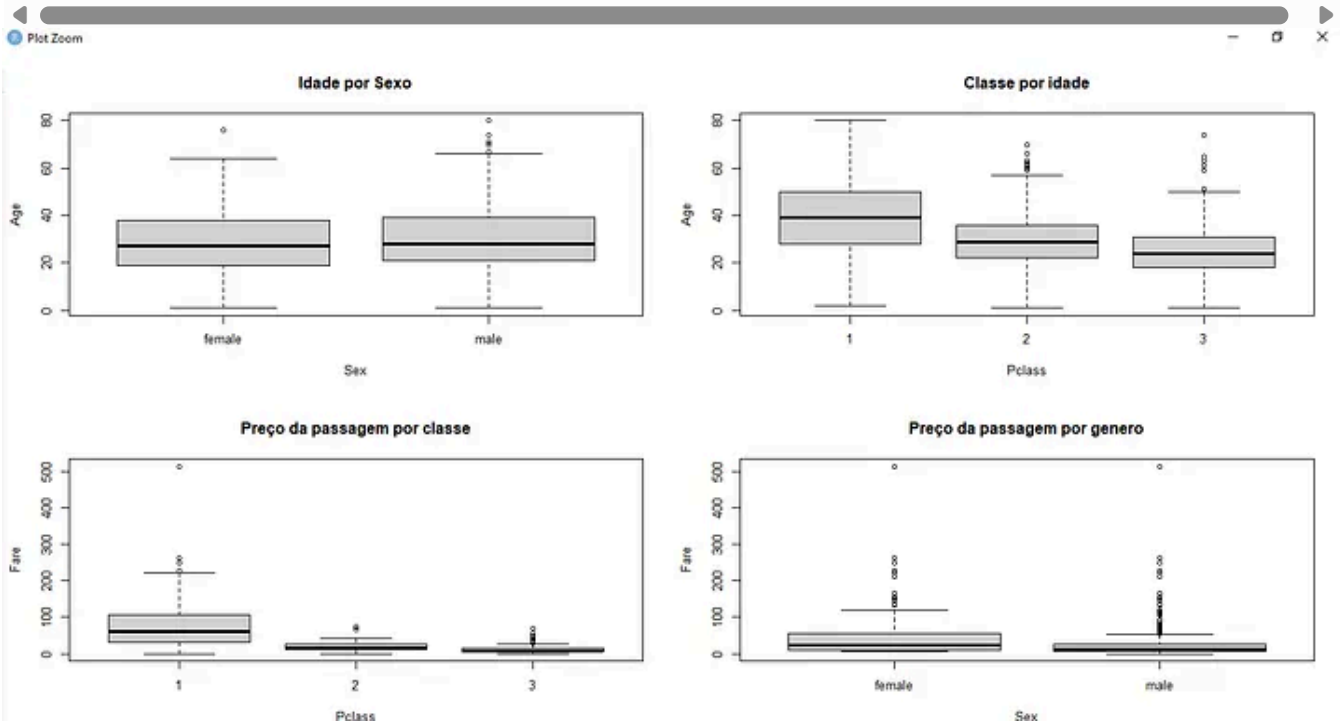
Como o texto está a ficar muito extenso, vou disponibilizar o código para criar o barchart e vamos iniciar a analise exploratória bivariada.

```
#Pclass
#criar label, nomes inspiradas no Kaggle
Pclass_label <- ifelse(titanic.full$Pclass == "1", "Alta",
                       ifelse(titanic.full$Pclass == "2", "Média", "Baixa"))

pclass_table<-table(Pclass_label)
pclass_table
cores <- c("#DAA520", "#8B4513", "#BDB76B")

barplot(pclass_table, main = "Observação por classe", col = cores)

#teste estatistico para classes
qui.quad.class<-chisq.test(pclass_table)
qui.quad.class
```

```
qui.quad.class$residuals
#também apresenta classe desbalanceadas
```

Análise Bivariada

A análise bivariada envolve a análise de pares de variáveis. Nesta análise observamos que as medianas das idades por género parecem próximas e há poucos outliers. Realizamos o teste de Mann-Whitney-Wilcoxon para avaliar se as distribuições são iguais. O resultado do teste apresentou um valor de p-valor igual a 0.058, o que indica que não há evidências suficientes para rejeitar a hipótese nula a um nível de significância de 5%. No entanto, é importante ressaltar que a interpretação desse resultado deve ser feita com cautela, uma vez que poderia ser considerado significativo em níveis de significância mais permissivos. Portanto, não podemos afirmar com certeza que as distribuições são “idênticas”, mas sim que não há evidências suficientes para concluir que são diferentes.



Boxplot para representar variáveis numéricas por níveis das variáveis categóricas.

Considerando que a análise bivariada não apresentou associação significativa entre as variáveis consideradas, a próxima etapa consistirá na análise do dataset de treino, com foco nas características dos sobreviventes desse desastre. Ao direcionar nossa atenção para esses dados específicos, poderemos obter insights mais relevantes e significativos sobre os fatores que contribuíram para a sobrevivência dos passageiros.

O código trabalhado nesta postagem está disponível em meu github.

Análise De Dados | Análise Exploratória | Pie Charts | Rstudio | Bar Chart

Follow

Written by Ana Paula Ferrari Januário

27 followers · 23 following

Data Scientist & Master in Statistics & PhD student. Brazilian based in Portugal, passionate about learning and sharing knowledge. Writing on Medium in PT only.

No responses yet

Write a response

What are your thoughts?

More from Ana Paula Ferrari Januário