Insper EDUCAÇÃO EXECUTIVA ONLINE

Live Learning AO VIVO

Programação para Não Programadores

Aula 4

Prof. Magno Severino e Prof. Marina Muradian

22/04/2021

Objetivos de aprendizagem

- Importar planilhas de dados no `R`.
- Realizar análise descritiva inicial.
- Obter o cálculo de estatísticas.

Importação e manupulação de bases de dados

O tipo de dados mais comum de ser importado para o `R` são aqueles que são armazenadas em planilhas. Nesta aula, vamos trabalhar com os dados contidos na biblioteca `nycflights13`, porém, vamos conosiderar a versão dos dados salvos em arquivos `.csv`.
Para importar dados do tipo `.csv` (valores separados por vírgula), utilize o comando abaixo.

```
flights <- read.csv("flights.csv")
airports <- read.csv("airports.csv")

class(flights)
```

```
## [1] "data.frame"
```

```
class(airports)
```

```
## [1] "data.frame"
```

Veja a seguinr os comandos para analisar a estrutura dos dados e obter uma descrição do tipo de cada coluna presente na base.

```
str(flights)
```

```
## 'data.frame':    336776 obs. of  20 variables:
##  $ X             : int  1 2 3 4 5 6 7 8 9 10 ...
##  $ year          : int  2013 2013 2013 2013 2013 2013 2013 2013 2013 2013 ...
##  $ month         : int  1 1 1 1 1 1 1 1 1 1 ...
##  $ day           : int  1 1 1 1 1 1 1 1 1 1 ...
##  $ dep_time      : int  517 533 542 544 554 554 555 557 557 558 ...
##  $ sched_dep_time: int  515 529 540 545 600 558 600 600 600 600 ...
##  $ dep_delay     : int  2 4 2 -1 -6 -4 -5 -3 -3 -2 ...
##  $ arr_time      : int  830 850 923 1004 812 740 913 709 838 753 ...
##  $ sched_arr_time: int  819 830 850 1022 837 728 854 723 846 745 ...
##  $ arr_delay     : int  11 20 33 -18 -25 12 19 -14 -8 8 ...
##  $ carrier       : Factor w/ 16 levels "9E","AA","AS",..: 12 12 2 4 5 12 4 6 4 2 ...
##  $ flight        : int  1545 1714 1141 725 461 1696 507 5708 79 301 ...
##  $ tailnum       : Factor w/ 4043 levels "D942DN","N0EGMQ",..: 180 524 2401 3204 2661 1142 1829 3300
##  $ origin        : Factor w/ 3 levels "EWR","JFK","LGA": 1 3 2 2 3 1 1 3 2 3 ...
##  $ dest          : Factor w/ 105 levels "ABQ","ACK","ALB",..: 44 44 59 13 5 70 36 43 55 70 ...
##  $ air_time      : int  227 227 160 183 116 150 158 53 140 138 ...
##  $ distance      : int  1400 1416 1089 1576 762 719 1065 229 944 733 ...
##  $ hour          : int  5 5 5 5 6 5 6 6 6 6 ...
##  $ minute        : int  15 29 40 45 0 58 0 0 0 0 ...
##  $ time_hour     : Factor w/ 6936 levels "2013-01-01 05:00:00",..: 1 1 1 1 2 1 2 2 2 2 ...
```

```
str(airports)
```

```
## 'data.frame':    1458 obs. of  9 variables:
##  $ X    : int  1 2 3 4 5 6 7 8 9 10 ...
##  $ faa  : Factor w/ 1458 levels "04G","06A","06C",..: 1 2 3 4 5 6 7 8 9 10 ...
##  $ name : Factor w/ 1440 levels "Aberdeen Regional Airport",..: 719 892 1171 1087 627 386 1404 422 1
##  $ lat  : num  41.1 32.5 42 41.4 31.1 ...
##  $ lon  : num  -80.6 -85.7 -88.1 -74.4 -81.4 ...
##  $ alt  : int  1044 264 801 523 11 1593 730 492 1000 108 ...
##  $ tz   : int  -5 -6 -6 -5 -5 -5 -5 -5 -5 -8 ...
##  $ dst  : Factor w/ 3 levels "A","N","U": 1 1 1 1 1 1 1 1 3 1 ...
##  $ tzone: Factor w/ 9 levels "America/Anchorage",..: 5 2 2 5 5 5 5 5 5 4 ...
```

Frequentemente, você deverá alterar o tipo de alguma(s) coluna(s), como, por exemplo, para tratar dados do tipo fator, caracteres e datas.

Nos dados armazenados em `flights`, podemos converter as colunas `carrier`, `orign` e `dest` para fator.

```
flights$carrier <- as.factor(flights$carrier)

flights$origin <- as.factor(flights$origin)

flights$dest <- as.factor(flights$dest)
```

Na tabela airport, podemos converter a coluna `tzone` para o tipo fator.

```
airports$tzone <- as.factor(airports$tzone)
```

Note que a coluna `flights$time_hour` representa a hora data e hora do voo. Com a função `head()` podemos visualizar as primeiras observações de uma coluna (e de um data frame também).

```r
head(flights$time_hour)
```

```
## [1] 2013-01-01 05:00:00 2013-01-01 05:00:00 2013-01-01 05:00:00
## [4] 2013-01-01 05:00:00 2013-01-01 06:00:00 2013-01-01 05:00:00
## 6936 Levels: 2013-01-01 05:00:00 2013-01-01 06:00:00 ... 2013-12-31 23:00:00
```

```r
class(flights$time_hour)
```

```
## [1] "factor"
```

Entretanto, a classe de `flights$time_hour` é caracter, o que nos impossibilita trabalhar com datas de maneira adequada. Uma alternativa, é utilizar a biblioteca `lubridate`. Para isso, execute o comando abaixo para instalá-la.

```r
install.packages("lubridate")
```

Em seguida, basta carregar a biblioteca e utilizar as funções que fazem conversão de caracteres em data e manipulações relacionadas a este tipo de dado. Abaixo, adicionamos períodos de tempos à uma data (anos, meses, dias), obtemos a diferença entre duas datas e selecionamos a unidade de tempo para exibir essa diferença.

```r
library(lubridate)

data <- "25/12/92"
class(data)
```

```
## [1] "character"
```

```r
data <- dmy(data)
class(data)
```

```
## [1] "Date"
```

```r
year(data)
```

```
## [1] 1992
```

```r
month(data)
```

```
## [1] 12
```

```r
day(data)
```

```
## [1] 25
```

```r
# adicionar 10 anos a uma data
data + years(10)
```

```
## [1] "2002-12-25"
```

```r
# adicionar 10 anos e 5 meses a uma data
data + years(10) + months(5)
```

```
## [1] "2003-05-25"
```

```r
# adicionar 10 anos, 5 meses e 7 dias a uma data
data + years(10) + months(5) + days(7)
```

```
## [1] "2003-06-01"
```

```r
# obter a diferença de duas datas em dias
interval(data, today()) / years(1)
```

```
## [1] 28.32055
```

```r
# obter a diferença de duas datas em dias
interval(data, today()) / days(1)
```

```
## [1] 10344
```

Este pacote nos permite trabalhar com data e hora em uma mesma variável. Note que agora a data está representada no formato MM/DD/AAAA, além de hora, minutos e segundos.

```r
data <- "12/25/92 10:07:35"
class(data)
```

```
## [1] "character"
```

```r
data <- mdy_hms(data)
class(data)
```

```
## [1] "POSIXct" "POSIXt"
```

```r
year(data)
```

```
## [1] 1992
```

```r
hour(data)
```

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

```
minute(data)
```

```
## [1] 7
```

```
second(data)
```

```
## [1] 35
```

Com isso, podemos coverter a coluna `flights$time_hour` para o tipo de dados `date`.

```
flights$time_hour <- ymd_hms(flights$time_hour)
```

Agora, todas as colunas de `flights` estão com os tipos de dados corretos.

```
str(flights)
```

```
## 'data.frame':    336776 obs. of  20 variables:
##  $ X             : int  1 2 3 4 5 6 7 8 9 10 ...
##  $ year          : int  2013 2013 2013 2013 2013 2013 2013 2013 2013 2013 ...
##  $ month         : int  1 1 1 1 1 1 1 1 1 1 ...
##  $ day           : int  1 1 1 1 1 1 1 1 1 1 ...
##  $ dep_time      : int  517 533 542 544 554 554 555 557 557 558 ...
##  $ sched_dep_time: int  515 529 540 545 600 558 600 600 600 600 ...
##  $ dep_delay     : int  2 4 2 -1 -6 -4 -5 -3 -3 -2 ...
##  $ arr_time      : int  830 850 923 1004 812 740 913 709 838 753 ...
##  $ sched_arr_time: int  819 830 850 1022 837 728 854 723 846 745 ...
##  $ arr_delay     : int  11 20 33 -18 -25 12 19 -14 -8 8 ...
##  $ carrier       : Factor w/ 16 levels "9E","AA","AS",..: 12 12 2 4 5 12 4 6 4 2 ...
##  $ flight        : int  1545 1714 1141 725 461 1696 507 5708 79 301 ...
##  $ tailnum       : Factor w/ 4043 levels "D942DN","N0EGMQ",..: 180 524 2401 3204 2661 1142 1829 3300
##  $ origin        : Factor w/ 3 levels "EWR","JFK","LGA": 1 3 2 2 3 1 1 3 2 3 ...
##  $ dest          : Factor w/ 105 levels "ABQ","ACK","ALB",..: 44 44 59 13 5 70 36 43 55 70 ...
##  $ air_time      : int  227 227 160 183 116 150 158 53 140 138 ...
##  $ distance      : int  1400 1416 1089 1576 762 719 1065 229 944 733 ...
##  $ hour          : int  5 5 5 5 6 5 6 6 6 6 ...
##  $ minute        : int  15 29 40 45 0 58 0 0 0 0 ...
##  $ time_hour     : POSIXct, format: "2013-01-01 05:00:00" "2013-01-01 05:00:00" ...
```

Análise descritiva

Realizar uma analise descritiva é fundamental para compreender, de forma resumida, a informação contida nos dados.

Uma função que resume de maneira "inteligente" todas as colunas de um data frame é a `summary`. É "inteligente" porque, a forma como o resumo é feito depente do tipo de cada coluna da tabela. Em `flights` há colunas numéricas, fatores, caracter e data.

```
summary(flights)
```

```
##        X                year          month             day
##  Min.   :     1   Min.   :2013   Min.   : 1.000   Min.   : 1.00
##  1st Qu.: 84195   1st Qu.:2013   1st Qu.: 4.000   1st Qu.: 8.00
##  Median :168389   Median :2013   Median : 7.000   Median :16.00
##  Mean   :168389   Mean   :2013   Mean   : 6.549   Mean   :15.71
##  3rd Qu.:252582   3rd Qu.:2013   3rd Qu.:10.000   3rd Qu.:23.00
##  Max.   :336776   Max.   :2013   Max.   :12.000   Max.   :31.00
##
##     dep_time    sched_dep_time   dep_delay          arr_time
##  Min.   :   1   Min.   : 106   Min.   : -43.00   Min.   :   1
##  1st Qu.: 907   1st Qu.: 906   1st Qu.:  -5.00   1st Qu.:1104
##  Median :1401   Median :1359   Median :  -2.00   Median :1535
##  Mean   :1349   Mean   :1344   Mean   :  12.64   Mean   :1502
##  3rd Qu.:1744   3rd Qu.:1729   3rd Qu.:  11.00   3rd Qu.:1940
##  Max.   :2400   Max.   :2359   Max.   :1301.00   Max.   :2400
##  NA's   :8255                  NA's   :8255      NA's   :8713
##  sched_arr_time   arr_delay           carrier          flight
##  Min.   :   1   Min.   : -86.000   UA     :58665   Min.   :   1
##  1st Qu.:1124   1st Qu.: -17.000   B6     :54635   1st Qu.: 553
##  Median :1556   Median :  -5.000   EV     :54173   Median :1496
##  Mean   :1536   Mean   :   6.895   DL     :48110   Mean   :1972
##  3rd Qu.:1945   3rd Qu.:  14.000   AA     :32729   3rd Qu.:3465
##  Max.   :2359   Max.   :1272.000   MQ     :26397   Max.   :8500
##                 NA's   :9430       (Other):62067
##     tailnum       origin            dest           air_time
##  N725MQ :   575   EWR:120835   ORD    : 17283   Min.   : 20.0
##  N722MQ :   513   JFK:111279   ATL    : 17215   1st Qu.: 82.0
##  N723MQ :   507   LGA:104662   LAX    : 16174   Median :129.0
##  N711MQ :   486                BOS    : 15508   Mean   :150.7
##  N713MQ :   483                MCO    : 14082   3rd Qu.:192.0
##  (Other):331700                CLT    : 14064   Max.   :695.0
##  NA's   :  2512                (Other):242450   NA's   :9430
##     distance         hour           minute
##  Min.   :  17   Min.   : 1.00   Min.   : 0.00
##  1st Qu.: 502   1st Qu.: 9.00   1st Qu.: 8.00
##  Median : 872   Median :13.00   Median :29.00
##  Mean   :1040   Mean   :13.18   Mean   :26.23
##  3rd Qu.:1389   3rd Qu.:17.00   3rd Qu.:44.00
##  Max.   :4983   Max.   :23.00   Max.   :59.00
##
##    time_hour
##  Min.   :2013-01-01 05:00:00
##  1st Qu.:2013-04-04 13:00:00
##  Median :2013-07-03 10:00:00
##  Mean   :2013-07-03 05:02:36
##  3rd Qu.:2013-10-01 07:00:00
##  Max.   :2013-12-31 23:00:00
##
```

Vamos obter a média de atrasos por aeroporto de origem. Antes, vamos verificar quantos aeroportos de origem existem na base.

```r
unique(flights$origin)
```

```
## [1] EWR LGA JFK
## Levels: EWR JFK LGA
```

Há quantos voos registrados saindo de cada aeroporto?

```r
table(flights$origin)
```

```
##
##    EWR    JFK    LGA
## 120835 111279 104662
```

Exercício: qual o nome dos aeroportos cujos códigos FAA estão listados acima? (Dica: fazer merge com `airports`).

Agora, vamos calcular a média de atraso geral para voos que partiram de "JFK".

```r
mean(flights$dep_delay[flights$origin == "JFK"])
```

```
## [1] NA
```

Note que existem dados faltantes (do tipo `NA`) na coluna `dep_delay`. Podemos desconsiderar esses dados no cálculo da média.

```r
mean(flights$dep_delay[flights$origin == "JFK"], na.rm = TRUE)
```

```
## [1] 12.11216
```

Podemos utilizar a função `tapply` para obter o resultado de uma função aplicada de acordo com certos grupos. Por exemplo, para obter a média de atraso dos voos de cada um dos aeroportos de origem, use o comando abaixo.

```r
tapply(flights$dep_delay, flights$origin, mean, na.rm = TRUE)
```

```
##      EWR      JFK      LGA
## 15.10795 12.11216 10.34688
```

Para arredondar números, podemos usar a função `round`, informando quantas casas decimais devem ser consideradas.

```r
round(tapply(flights$dep_delay, flights$origin, mean, na.rm = TRUE), 2)
```

```
##   EWR   JFK   LGA
## 15.11 12.11 10.35
```

Podemos fazer o mesmo considerando os aeroportos de destino.

```r
unique(flights$dest)
```

```
##   [1] IAH MIA BQN ATL ORD FLL IAD MCO PBI TPA LAX SFO DFW BOS LAS MSP DTW
##  [18] RSW SJU PHX BWI CLT BUF DEN SNA MSY SLC XNA MKE SEA ROC SYR SRQ RDU
##  [35] CMH JAX CHS MEM PIT SAN DCA CLE STL MYR JAC MDW HNL BNA AUS BTV PHL
##  [52] STT EGE AVL PWM IND SAV CAK HOU LGB DAY ALB BDL MHT MSN GSO CVG BUR
##  [69] RIC GSP GRR MCI ORF SAT SDF PDX SJC OMA CRW OAK SMF TUL TYS OKC PVD
##  [86] DSM PSE BHM CAE HDN BZN MTJ EYW PSP ACK BGR ABQ ILM MVY SBN LEX CHO
## [103] TVC ANC LGA
## 105 Levels: ABQ ACK ALB ANC ATL AUS AVL BDL BGR BHM BNA BOS BQN BTV ... XNA
```

```r
round(tapply(flights$arr_delay, flights$origin, mean, na.rm = TRUE), 2)
```

```
##  EWR  JFK  LGA
## 9.11 5.55 5.78
```

Saber somente a média de uma meida não é suficiente, devemos obter outras estatísticas, como a mediana, desvio padrão e variância, por exemplo.

```r
tapply(flights$dep_delay, flights$origin, median, na.rm = TRUE)
```

```
## EWR JFK LGA
##  -1  -1  -3
```

```r
tapply(flights$dep_delay, flights$origin, sd, na.rm = TRUE)
```

```
##      EWR      JFK      LGA
## 41.32370 39.03507 39.99302
```

```r
tapply(flights$dep_delay, flights$origin, var, na.rm = TRUE)
```

```
##      EWR      JFK      LGA
## 1707.649 1523.737 1599.442
```

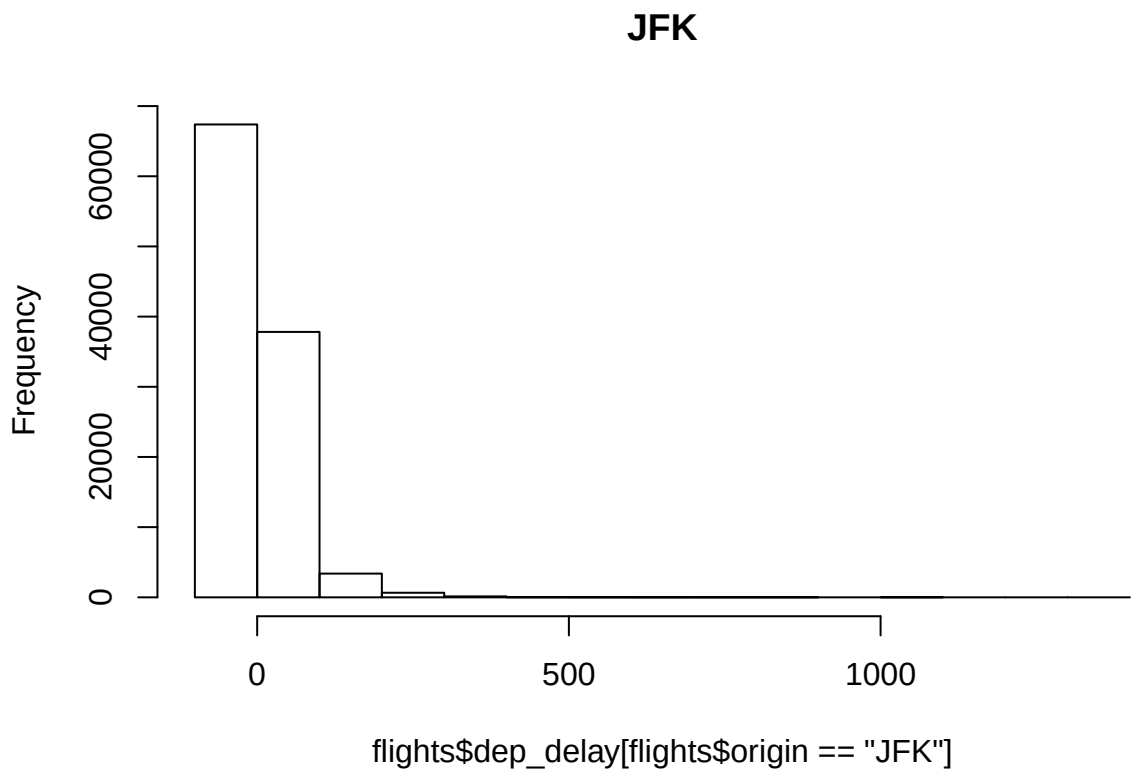
Podemos combinar todos esses resultados acima em uma única tabela, usando o comoando `cbind`. Abaixo, acrescentamos ainda os valores mínimo e máximo de atraso na partida observado em cada aerporto.

```r
cbind(Media = tapply(flights$dep_delay, flights$origin, mean, na.rm = TRUE),
      Mediana = tapply(flights$dep_delay, flights$origin, median, na.rm = TRUE),
      Minimo = tapply(flights$dep_delay, flights$origin, min, na.rm = TRUE),
      Maximo = tapply(flights$dep_delay, flights$origin, max, na.rm = TRUE),
      Desvio_padrao = tapply(flights$dep_delay, flights$origin, sd, na.rm = TRUE),
      Variancia = tapply(flights$dep_delay, flights$origin, var, na.rm = TRUE))
```

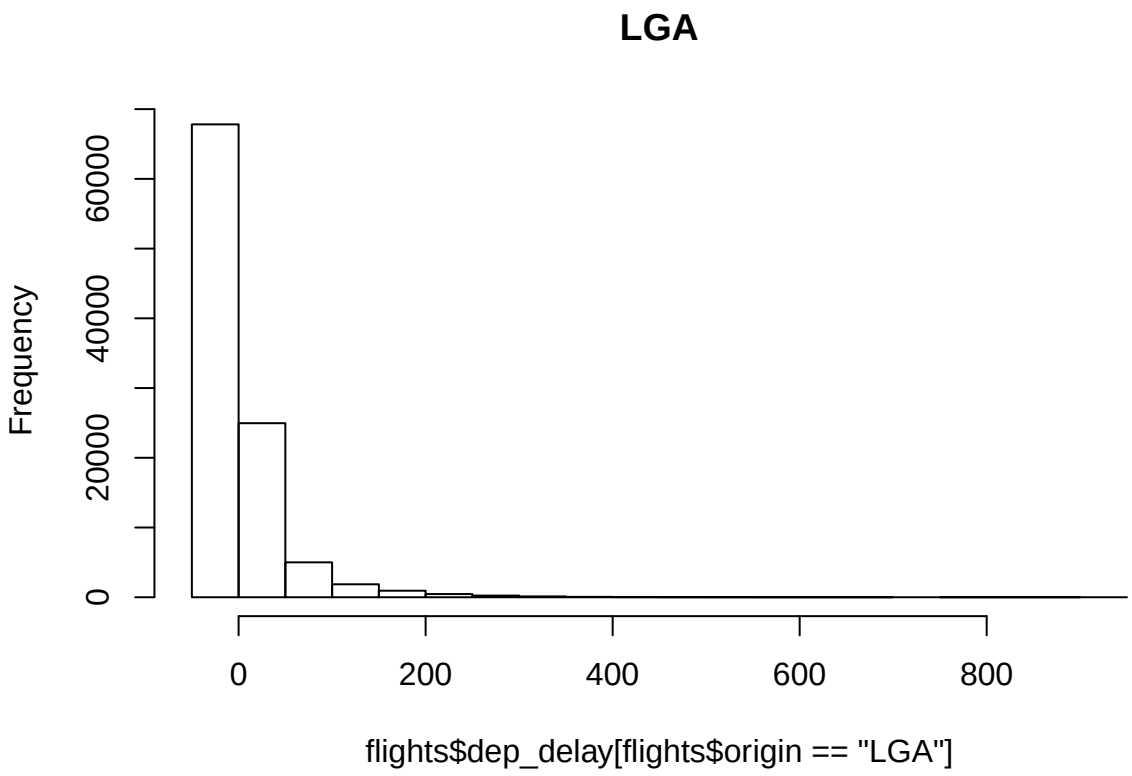
```
##        Media Mediana Minimo Maximo Desvio_padrao Variancia
## EWR 15.10795      -1    -25   1126      41.32370  1707.649
## JFK 12.11216      -1    -43   1301      39.03507  1523.737
## LGA 10.34688      -3    -33    911      39.99302  1599.442
```

Além disso, podemos construir gráficos, para analisar os dados de forma visual.

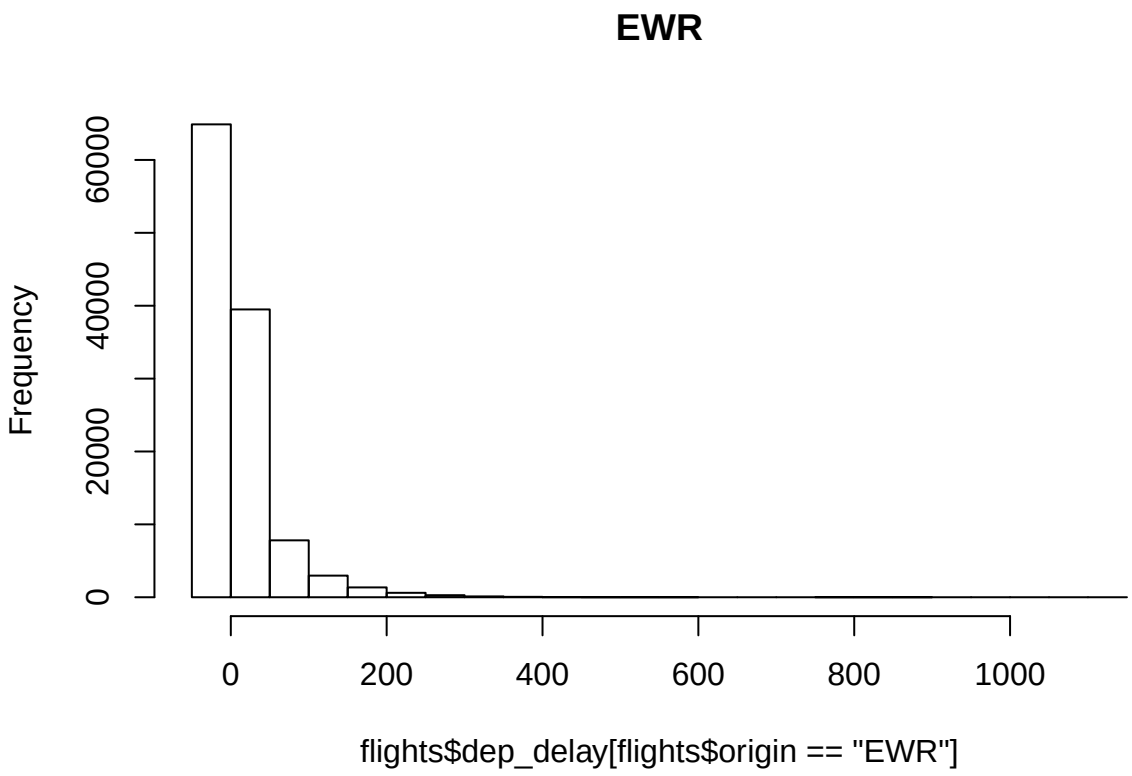
```r
hist(flights$dep_delay[flights$origin == "JFK"], main="JFK")
```




```r
hist(flights$dep_delay[flights$origin == "LGA"], main="LGA")
```




```r
hist(flights$dep_delay[flights$origin == "EWR"], main="EWR")
```



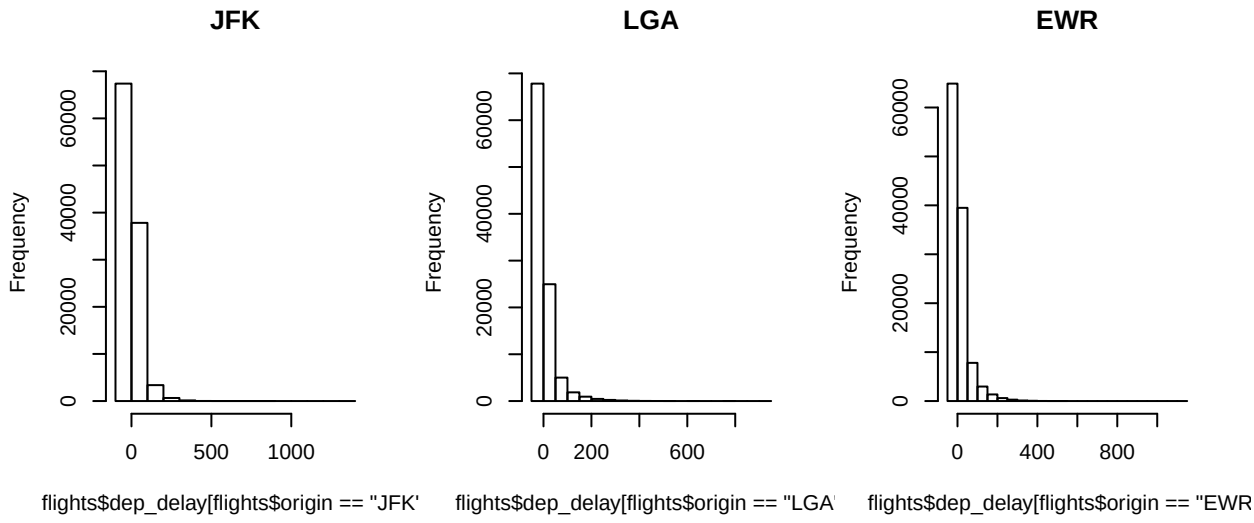
Os três gráficos podem ser exibidos de uma única vez se usarmos a opção `mfrow`, que permite definir o número de linhas e colunas do grid para plotar os gráficos.

```
par(mfrow=c(1,3))

hist(flights$dep_delay[flights$origin == "JFK"], main="JFK")

hist(flights$dep_delay[flights$origin == "LGA"], main="LGA")

hist(flights$dep_delay[flights$origin == "EWR"], main="EWR")
```



Lista de exercicios

1. Importe no `R` a planilha de dados correspondente ao seu projeto aplicado.
2. Verifique se alguma coluna deve ser transformada para um tipo de dados mais adequado.
3. Obtenha estatísticas descritivas dos dados.