Universidade de Brasília

DEPARTAMENTO DE ESTATÍSTICA

05 fevereiro 2023

Trabalho 2

Profª. Drª. Thais Carvalho Valadares Rodrigues

Computação em Estatística 2

Aluno: Bruno Gondim Toledo | Matrícula: 15/0167636

Instruções

1) Escreva seu código com esmero, evitando operações redundantes, comentando os resultados e usando as melhores práticas em programação. **Utilize o pacote purr**.

2) O aluno deve enviar 1 arquivo pdf contendo **ambos**: o código utilizado para realizar cada questão e o resultado obtido (ou as primeiras linhas do resultado obtido, caso o resultado seja extenso), de preferência, utilizem o Rmarkdown. Além disso, o aluno deve enviar o arquivo .R com o código desenvolvido.

3) Os arquivos devem ser enviados pelo Teams até às 23h59 do dia 17/02/2023.

4) O trabalho é individual! Respostas semelhantes serão penalizadas severamente.

Questão 1) Com base no banco de dados `nycflights13::flights`, execute os comandos a seguir, a fim de transformar o banco de dados em uma lista por companhia aerea (*carrier*), e responda os itens abaixo utilizando a lista banco e as funcionalidades do pacote purr. `banco <- nycflights13::flights %>% split(.$carrier)`

```
if (!require("pacman")) install.packages("pacman")
```

```
## Carregando pacotes exigidos: pacman
```

```
p_load(knitr,tidyverse,pracma,signal)
banco <- nycflights13::flights %>% split(.$carrier)
```

a) A partir da lista banco, selecione as colunas que contém a palavra 'delay' para cada companhia aerea (*carrier*). Retorne o resultado como uma lista. (1 ponto)

```
lista <- banco %>%
  map(~select(.x, contains("delay")))

head(lista,n=2)
```

```
## $`9E`
## # A tibble: 18,460 x 2
##    dep_delay arr_delay
##        <dbl>     <dbl>
##  1         0        11
##  2        -9        -2
##  3        -3        -2
##  4        -6        -1
##  5        -8        -5
##  6         0        -5
##  7         6         5
##  8         0        13
##  9        -8        -8
## 10        -6       -33
## # ... with 18,450 more rows
##
## $AA
## # A tibble: 32,729 x 2
##    dep_delay arr_delay
##        <dbl>     <dbl>
##  1         2        33
##  2        -2         8
##  3        -1        31
##  4        -4       -12
##  5        13         5
##  6        -2        -3
##  7        -1        14
```

```
##  8         0        48
##  9        -4         4
## 10        -3       -10
## # ... with 32,719 more rows
```

b) A partir da lista obtida na letra A, calcule a média dos atrasos de chegada e de saída para cada companhia aerea (*carrier*). Retorne o resultado como um dataframe. (1 ponto)

```
dataframe <- lista %>%
  map_dfr(~ data.frame(
    mean_dep_delay  = mean(.x$dep_delay,na.rm = T),
    mean_arr_delay = mean(.x$arr_delay,na.rm = T))
    )

dataframe
```

```
##    mean_dep_delay mean_arr_delay
## 1       16.725769      7.3796692
## 2        8.586016      0.3642909
## 3        5.804775     -9.9308886
## 4       13.022522      9.4579733
## 5        9.264505      1.6443409
## 6       19.955390     15.7964311
## 7       20.215543     21.9207048
## 8       18.726075     20.1159055
## 9        4.900585     -6.9152047
## 10      10.552041     10.7747334
## 11      12.586207     11.9310345
## 12      12.106073      3.5580111
## 13       3.782418      2.1295951
## 14      12.869421      1.7644644
## 15      17.711744      9.6491199
## 16      18.996330     15.5569853
```

c) Crie uma função para repetir a tarefa da letra B, mas permita ao usuário escolher a operação que será efetuada nas colunas (média, mínimo, *summary*, etc . . .). Permita ao usuário escolher se `na.rm = T/F`. Retorne o resultado como uma lista. (1 ponto)

```
funcao <- function(operacao=mean,na=T){
  lista2 <- lista %>%
  map(~list(mean_dep_delay  = operacao(.x$dep_delay,na.rm = na),
        mean_arr_delay = operacao(.x$arr_delay,na.rm = na)))
  return(lista2)
}

funcao()
```

```
## $`9E`
## $`9E`$mean_dep_delay
## [1] 16.72577
##
## $`9E`$mean_arr_delay
## [1] 7.379669
##
##
## $AA
## $AA$mean_dep_delay
## [1] 8.586016
##
## $AA$mean_arr_delay
## [1] 0.3642909
##
##
## $AS
## $AS$mean_dep_delay
## [1] 5.804775
##
## $AS$mean_arr_delay
## [1] -9.930889
##
##
## $B6
## $B6$mean_dep_delay
## [1] 13.02252
##
## $B6$mean_arr_delay
## [1] 9.457973
##
##
## $DL
## $DL$mean_dep_delay
## [1] 9.264505
##
## $DL$mean_arr_delay
## [1] 1.644341
##
##
## $EV
## $EV$mean_dep_delay
```

```
## [1] 19.95539
##
## $EV$mean_arr_delay
## [1] 15.79643
##
##
## $F9
## $F9$mean_dep_delay
## [1] 20.21554
##
## $F9$mean_arr_delay
## [1] 21.9207
##
##
## $FL
## $FL$mean_dep_delay
## [1] 18.72607
##
## $FL$mean_arr_delay
## [1] 20.11591
##
##
## $HA
## $HA$mean_dep_delay
## [1] 4.900585
##
## $HA$mean_arr_delay
## [1] -6.915205
##
##
## $MQ
## $MQ$mean_dep_delay
## [1] 10.55204
##
## $MQ$mean_arr_delay
## [1] 10.77473
##
##
## $OO
## $OO$mean_dep_delay
## [1] 12.58621
##
## $OO$mean_arr_delay
## [1] 11.93103
##
##
## $UA
## $UA$mean_dep_delay
## [1] 12.10607
##
## $UA$mean_arr_delay
## [1] 3.558011
##
##
## $US
## $US$mean_dep_delay
## [1] 3.782418
##
```

```
## $US$mean_arr_delay
## [1] 2.129595
##
##
## $VX
## $VX$mean_dep_delay
## [1] 12.86942
##
## $VX$mean_arr_delay
## [1] 1.764464
##
##
## $WN
## $WN$mean_dep_delay
## [1] 17.71174
##
## $WN$mean_arr_delay
## [1] 9.64912
##
##
## $YV
## $YV$mean_dep_delay
## [1] 18.99633
##
## $YV$mean_arr_delay
## [1] 15.55699
```

```
funcao(operacao=summary,na=F)
```

```
## $`9E`
## $`9E`$mean_dep_delay
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's
##  -24.00   -6.00   -2.00   16.73   17.00  747.00    1044
##
## $`9E`$mean_arr_delay
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's
##  -68.00  -21.00   -7.00    7.38   15.00  744.00    1166
##
##
## $AA
## $AA$mean_dep_delay
##     Min.  1st Qu.   Median     Mean  3rd Qu.     Max.     NA's
##  -24.000   -6.000   -3.000    8.586    4.000 1014.000      636
##
## $AA$mean_arr_delay
##      Min.   1st Qu.    Median      Mean   3rd Qu.      Max.      NA's
##  -75.0000  -21.0000   -9.0000    0.3643    8.0000 1007.0000       782
##
##
## $AS
## $AS$mean_dep_delay
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's
## -21.000  -7.000  -3.000   5.805   3.000 225.000       2
##
## $AS$mean_arr_delay
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's
## -74.000 -32.000 -17.000  -9.931   2.000 198.000       5
##
```

```
##
## $B6
## $B6$mean_dep_delay
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's
##  -43.00   -5.00   -1.00   13.02   12.00  502.00     466
##
## $B6$mean_arr_delay
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's
## -71.000 -14.000  -3.000   9.458  17.000 497.000     586
##
##
## $DL
## $DL$mean_dep_delay
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's
## -33.000  -5.000  -2.000   9.264   5.000 960.000     349
##
## $DL$mean_arr_delay
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's
## -71.000 -20.000  -8.000   1.644   8.000 931.000     452
##
##
## $EV
## $EV$mean_dep_delay
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's
##  -32.00   -5.00   -1.00   19.96   25.00  548.00    2817
##
## $EV$mean_arr_delay
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's
##   -62.0   -14.0    -1.0    15.8    26.0   577.0    3065
##
##
## $F9
## $F9$mean_dep_delay
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's
##  -27.00   -4.00    0.50   20.22   18.00  853.00       3
##
## $F9$mean_arr_delay
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's
##  -47.00   -9.00    6.00   21.92   31.00  834.00       4
##
##
## $FL
## $FL$mean_dep_delay
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's
##  -22.00   -4.00    1.00   18.73   17.00  602.00      73
##
## $FL$mean_arr_delay
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's
##  -44.00   -7.00    5.00   20.12   24.00  572.00      85
##
##
## $HA
## $HA$mean_dep_delay
##     Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
##  -16.000   -7.000   -4.000    4.901   -1.000 1301.000
##
## $HA$mean_arr_delay
##     Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
```

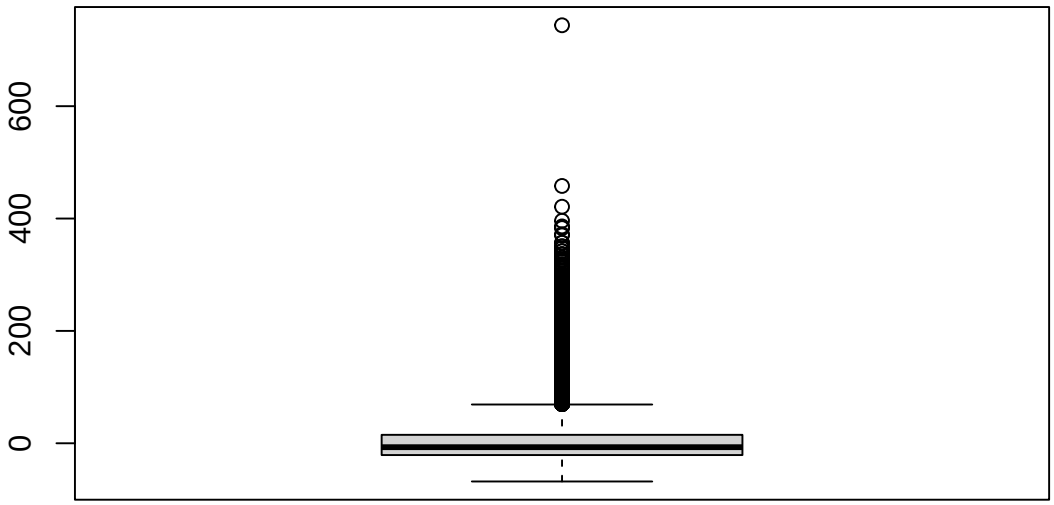
```
##  -70.000  -27.750  -13.000   -6.915    2.750 1272.000
##
##
## $MQ
## $MQ$mean_dep_delay
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's
##  -26.00   -7.00   -3.00   10.55    9.00 1137.00    1234
##
## $MQ$mean_arr_delay
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's
##  -53.00  -13.00   -1.00   10.77   18.00 1127.00    1360
##
##
## $OO
## $OO$mean_dep_delay
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's
##  -14.00   -9.00   -6.00   12.59    4.00  154.00       3
##
## $OO$mean_arr_delay
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's
##  -26.00  -16.00   -7.00   11.93    6.00  157.00       3
##
##
## $UA
## $UA$mean_dep_delay
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's
##  -20.00   -4.00    0.00   12.11   11.00  483.00     686
##
## $UA$mean_arr_delay
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's
## -75.000 -18.000  -6.000   3.558  12.000 455.000     883
##
##
## $US
## $US$mean_dep_delay
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's
## -19.000  -7.000  -4.000   3.782   0.000 500.000     663
##
## $US$mean_arr_delay
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's
##  -70.00  -15.00   -6.00    2.13    8.00  492.00     705
##
##
## $VX
## $VX$mean_dep_delay
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's
##  -20.00   -4.00    0.00   12.87    8.00  653.00      31
##
## $VX$mean_arr_delay
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's
## -86.000 -23.000  -9.000   1.764   8.000 676.000      46
##
##
## $WN
## $WN$mean_dep_delay
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's
##  -13.00   -2.00    1.00   17.71   17.00  471.00     192
##
```

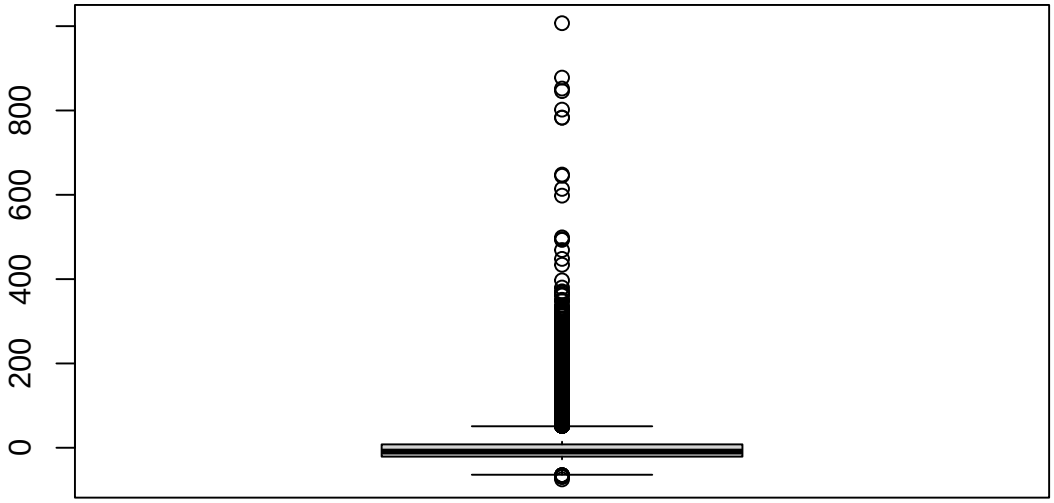
```
## $WN$mean_arr_delay
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's
## -58.000 -15.000  -3.000   9.649  15.000 453.000     231
##
##
## $YV
## $YV$mean_dep_delay
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's
##     -16      -7      -2      19      23     387      56
##
## $YV$mean_arr_delay
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's
##  -46.00  -16.00   -2.00   15.56   24.25  381.00      57
```

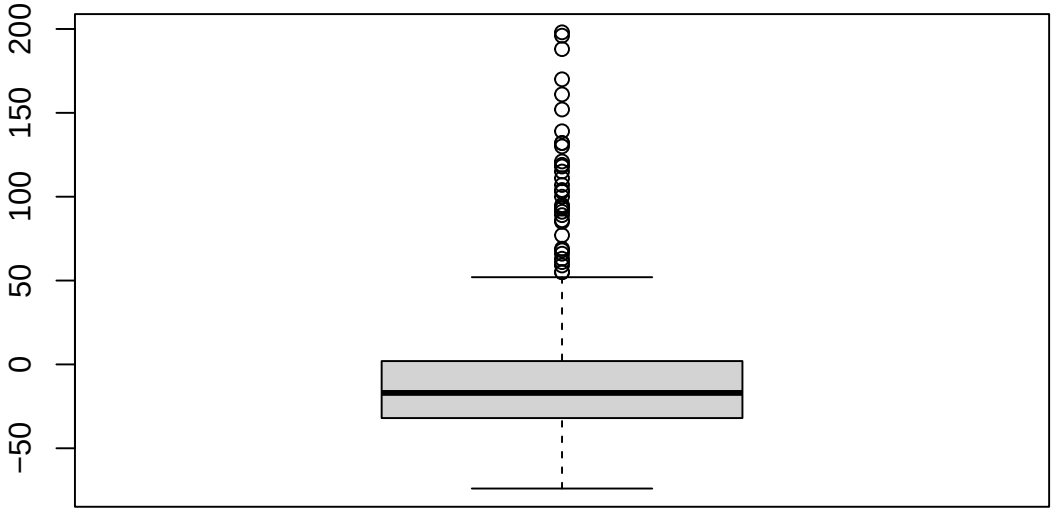
```
# Mais exemplos de operação da função no arquivo .R;
# suprimido aqui para não poluir demais o documento!
```

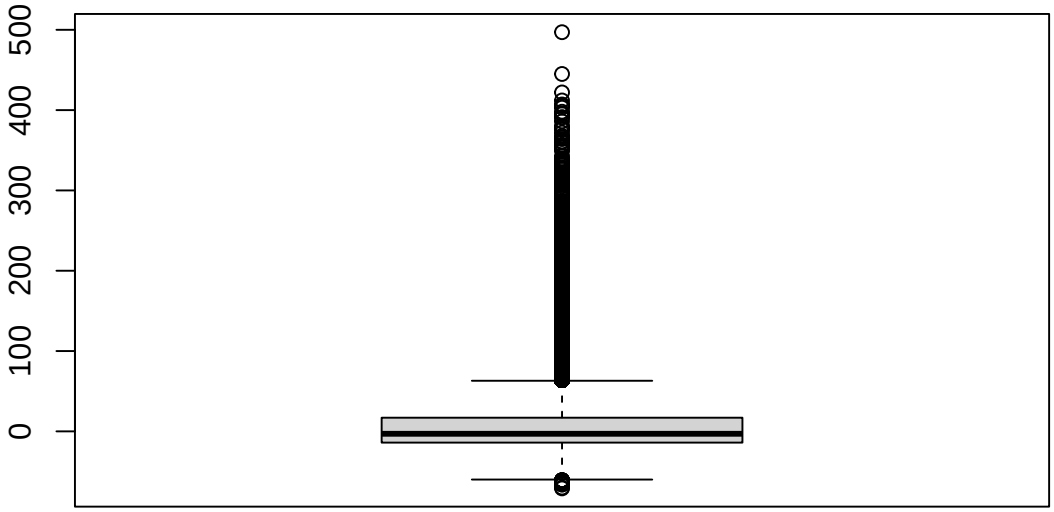
d) A partir da lista inicial banco, faça um *boxplot* da coluna 'arr_delay' para cada companhia aérea (utilize as funções map e boxplot). (1 ponto)

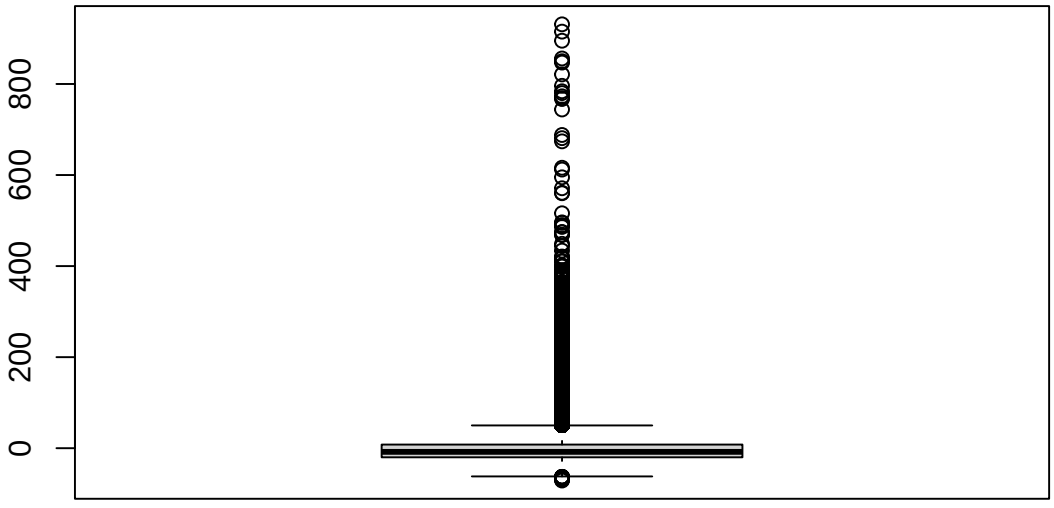
```
lista3 <- banco %>%
  map(~select(.x, contains("arr_delay"))) %>%
  map(~boxplot(.))
```

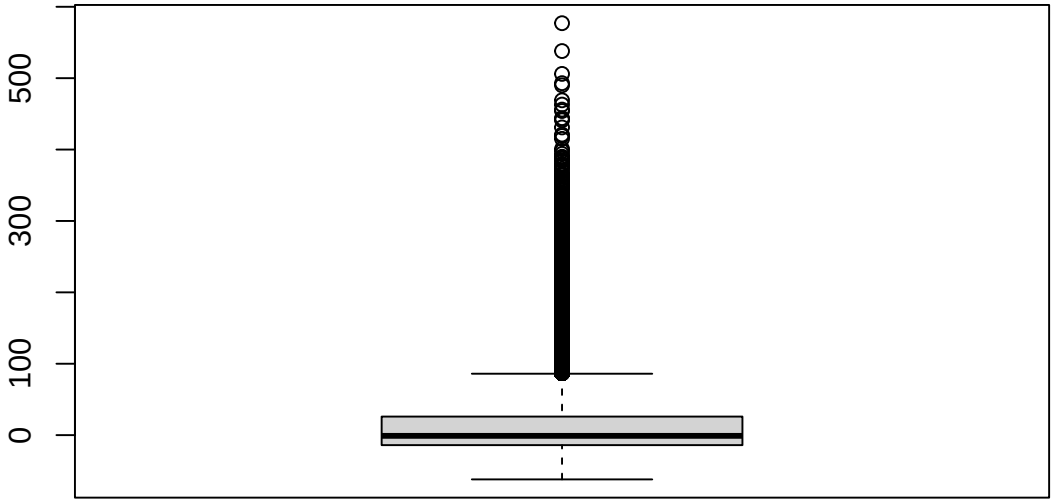


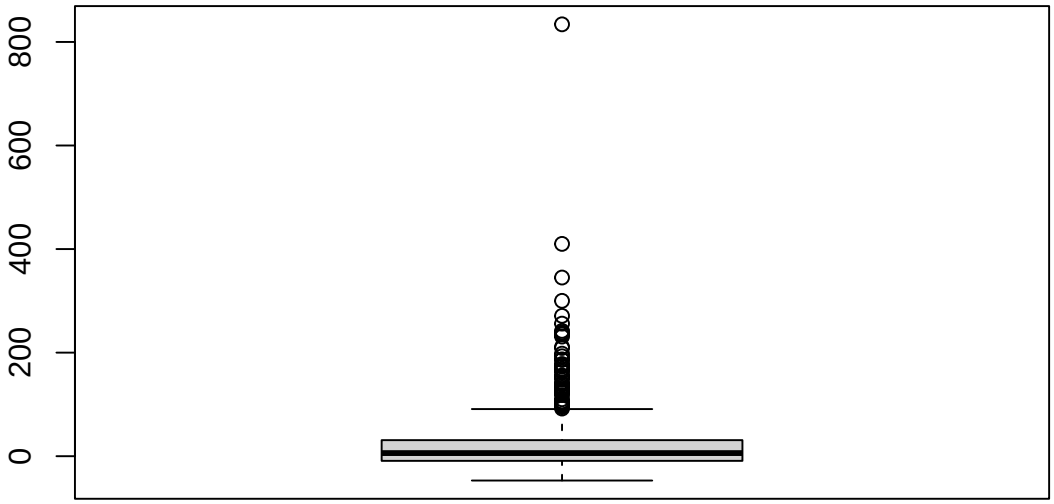

0
200
400
600
800

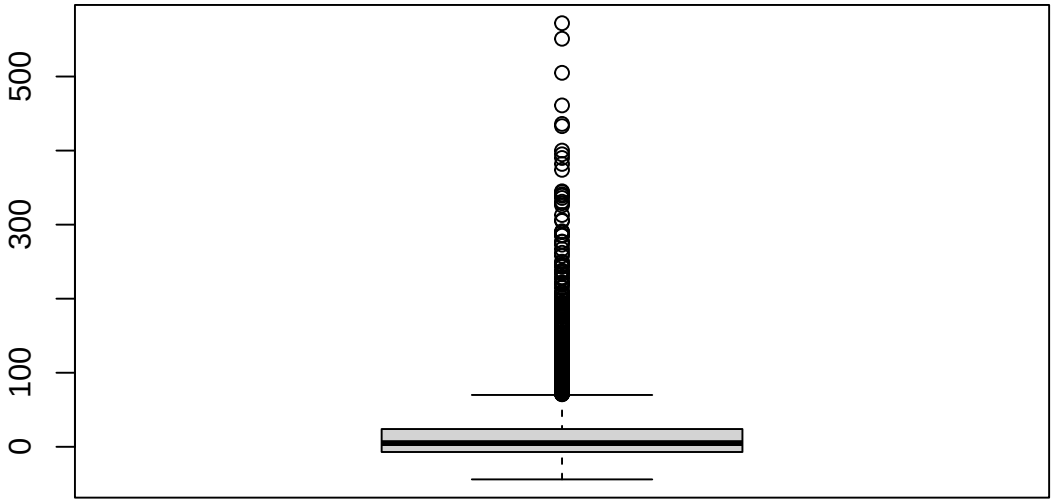

−50
0
50
100
150
200

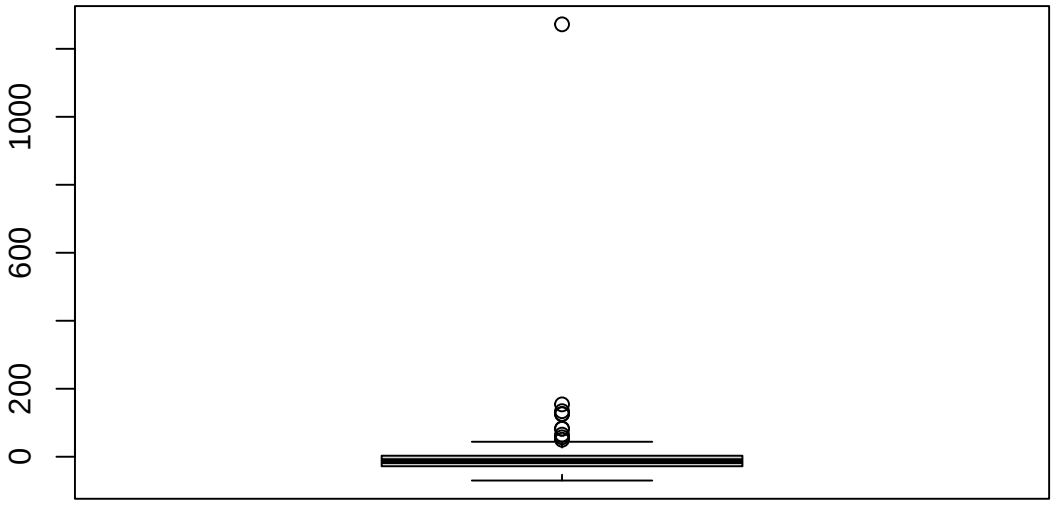

0
100
200
300
400
500

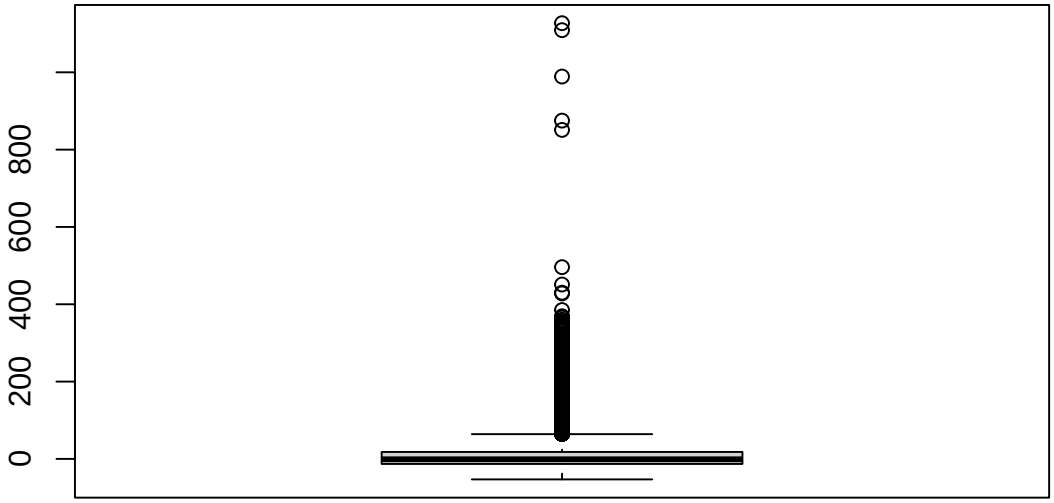

0
200
400
600
800

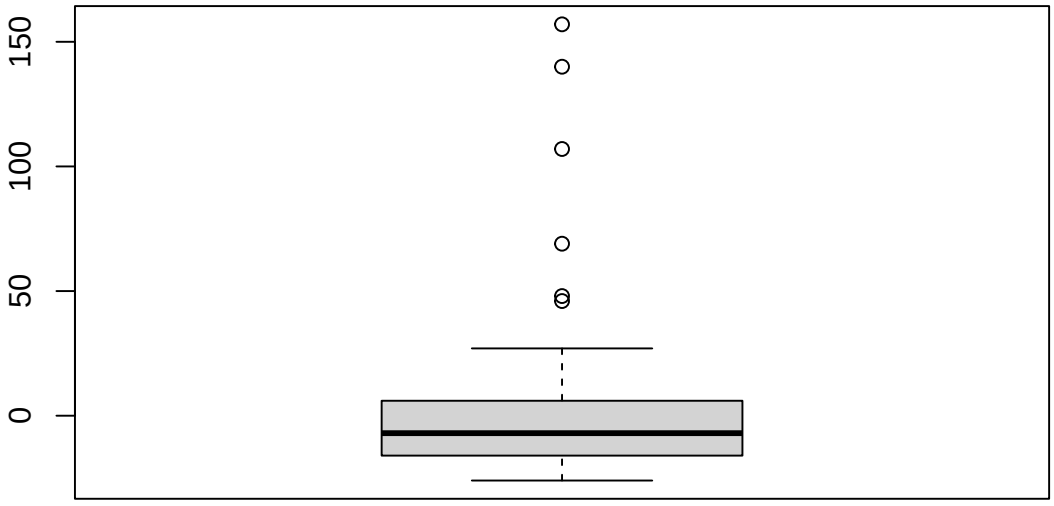

0
100
300
500

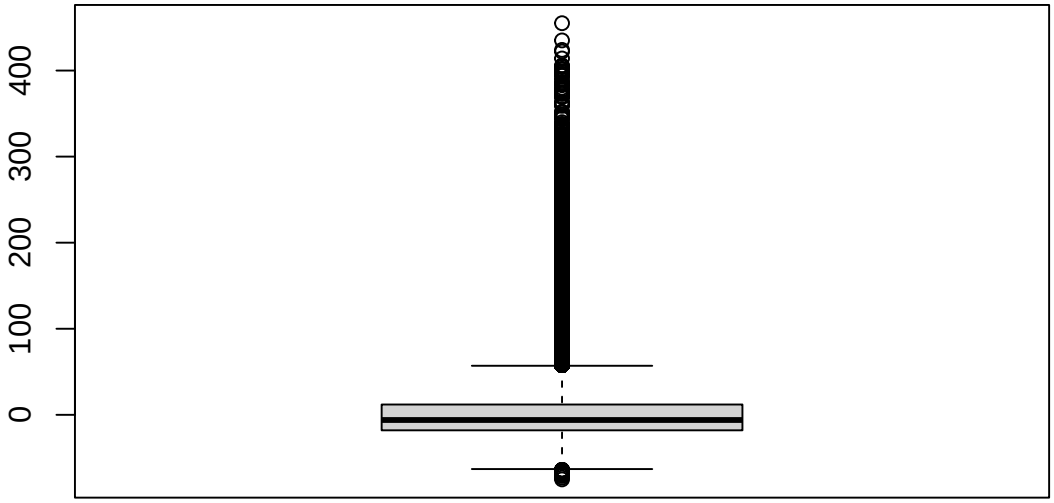

0
200
400
600
800

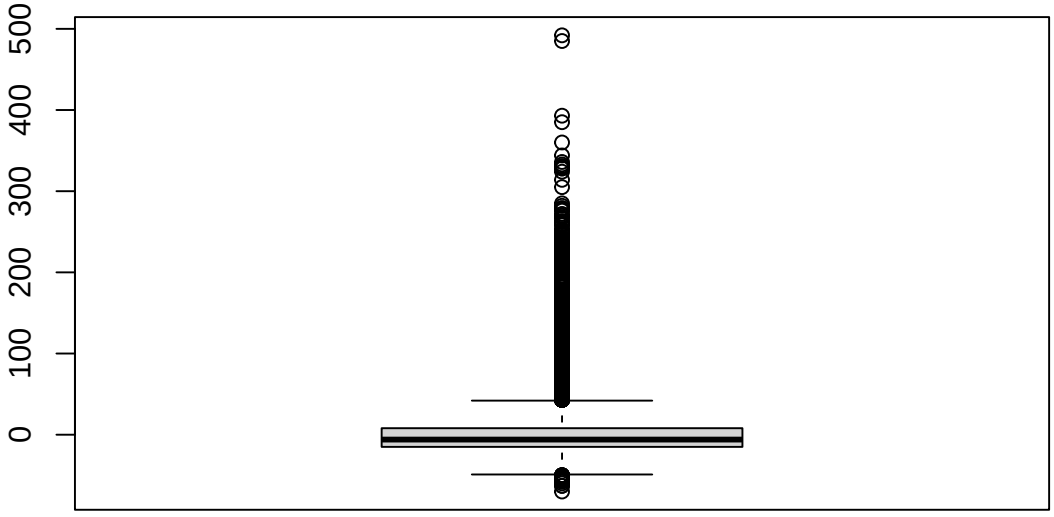

0
100
300
500

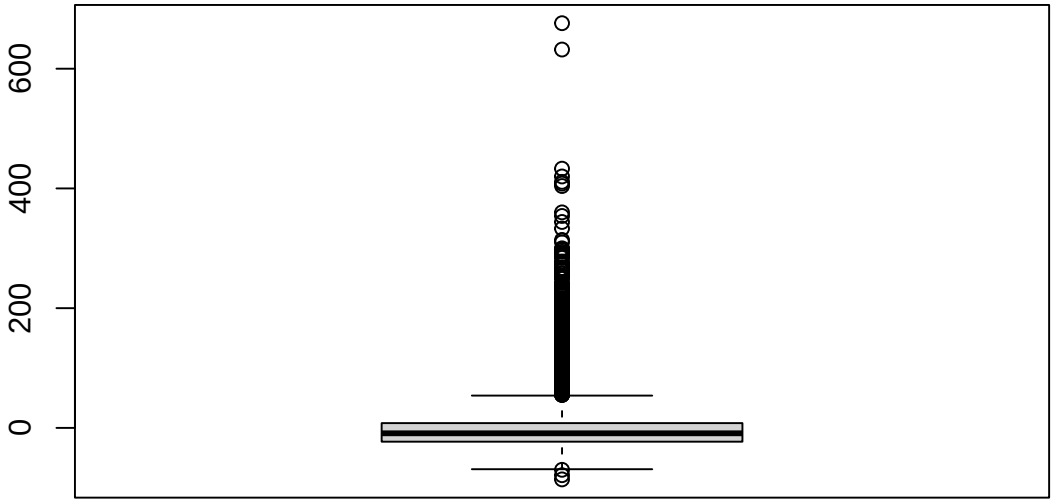

0
200
600
1000

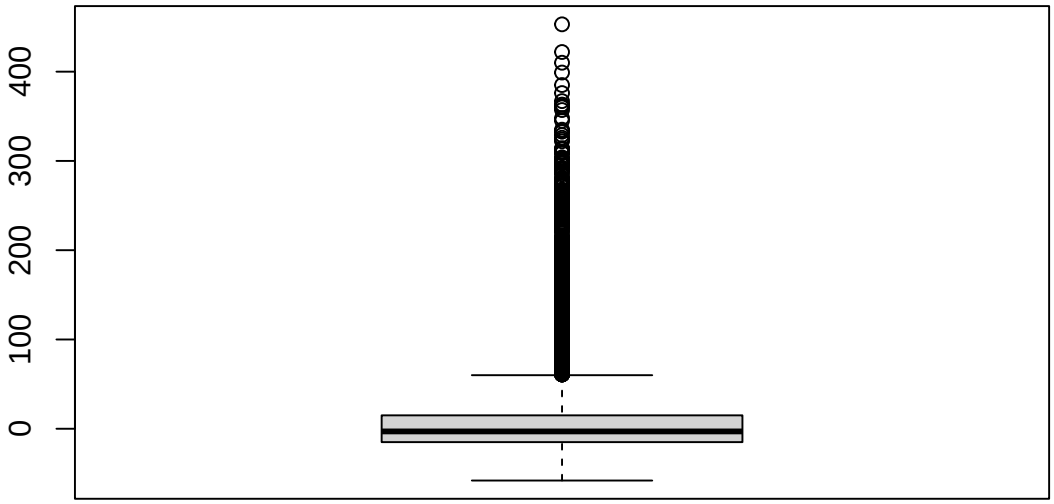

0
200
400
600
800

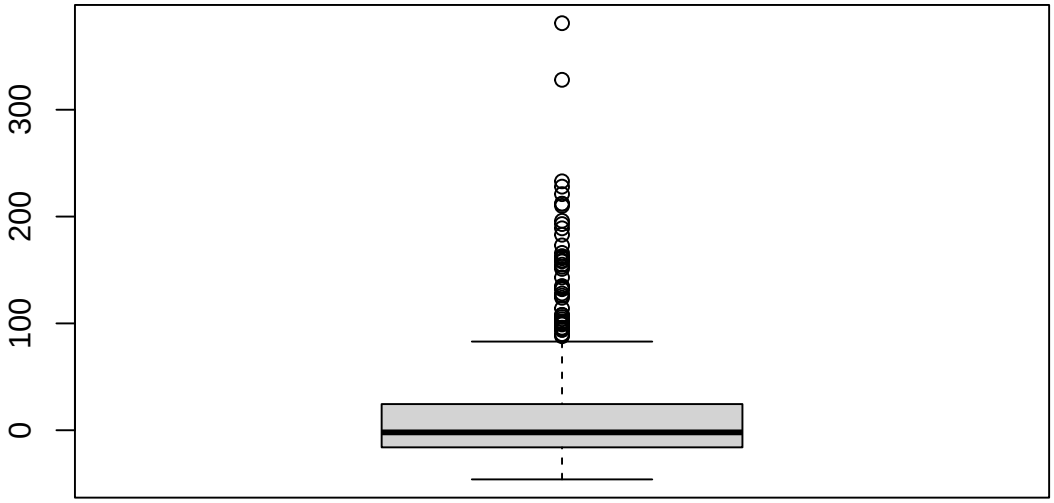

0
50
100
150


0
100
200
300
400


0
100
200
300
400
500


0
200
400
600


0
100
200
300
400


0
100
200
300

e) Note que, além do gráfico, a função `boxplot` retorna uma série de informações relevantes. Com base nisso, retorne quantos outliers na variável `'arr_delay'` cada companhia aérea registrou. Retorne o resultado como um vetor. (1 ponto)

```
outliers <- lista3 %>%
  map(~.$out) %>%
  map(as.data.frame)

quantidade_outliers <- outliers %>%
  map_int(nrow) %>%
  unlist()

quantidade_outliers
```

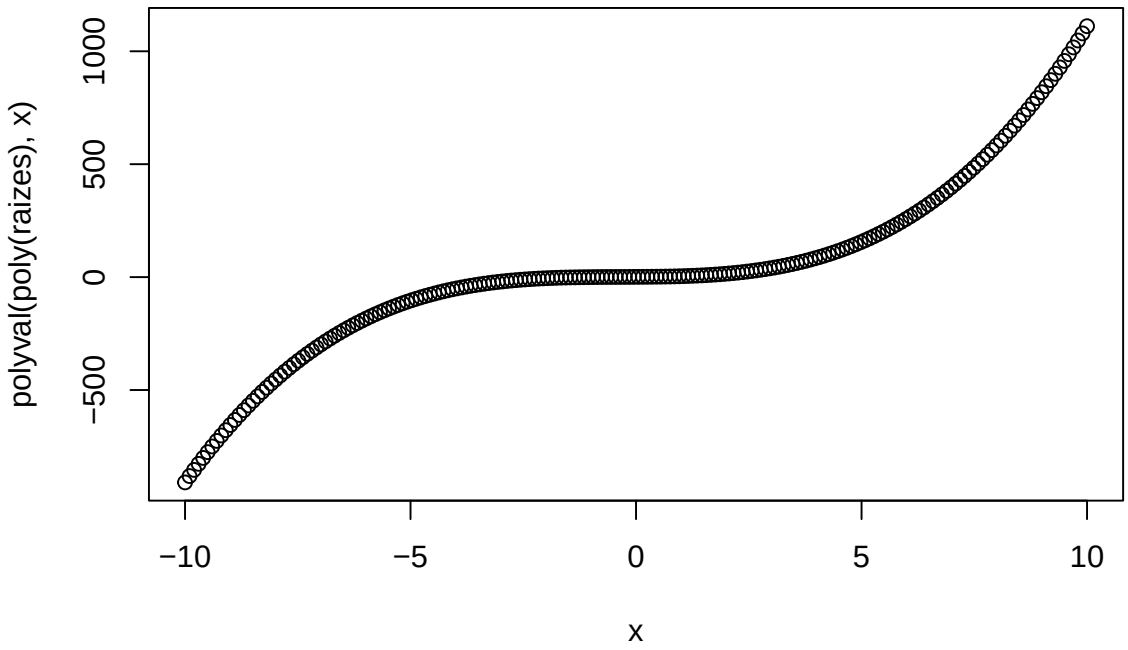
```
##   9E   AA   AS   B6   DL   EV   F9   FL   HA   MQ   OO   UA   US   VX   WN   YV
## 1600 2482   36 4706 3586 4379   56  311   10 2120    6 4217 1478  405 1063   49
```

Questão 2) Construa uma função para calcular as raízes de um polinômio de grau 3. Caso hajam duas raízes iguais, lançar uma mensagem de aviso para notificar o usuário. Permita ao usuário a opção de que o grafico da função seja construído e as raízes identificadas. Use o elemento ... para dar ao usuário controle dos parâmetros do gráfico. Por fim, teste sua função com alguns exemplos interessantes. (3 ponto)

```
pol3 <- function(a,b,c,d,grafico=T,resultado=T, ...){
  raizes <- roots(c(a,b,c,d))
  lista_return <- list()
  lista_return[[1]] <- raizes
  if (length(unique(raizes)) != length(raizes)){
    lista_return[[2]] <- "Existem raízes iguais"}
  if (grafico==T & resultado==T){
    x <- seq(-10,10,length=200)
    plot(x,polyval(poly(raizes),x), ...)
    return(lista_return)
  }
  else if (grafico==T & resultado==F){
    x <- seq(-10,10,length=200)
    plot(x,polyval(poly(raizes),x), ...)
  }
  else if(grafico==F & resultado==T){
    return(lista_return)
  }
  else {
    return("Girafas são criaturas sem coração")
  }
}

pol3(1,1,1,1)
```

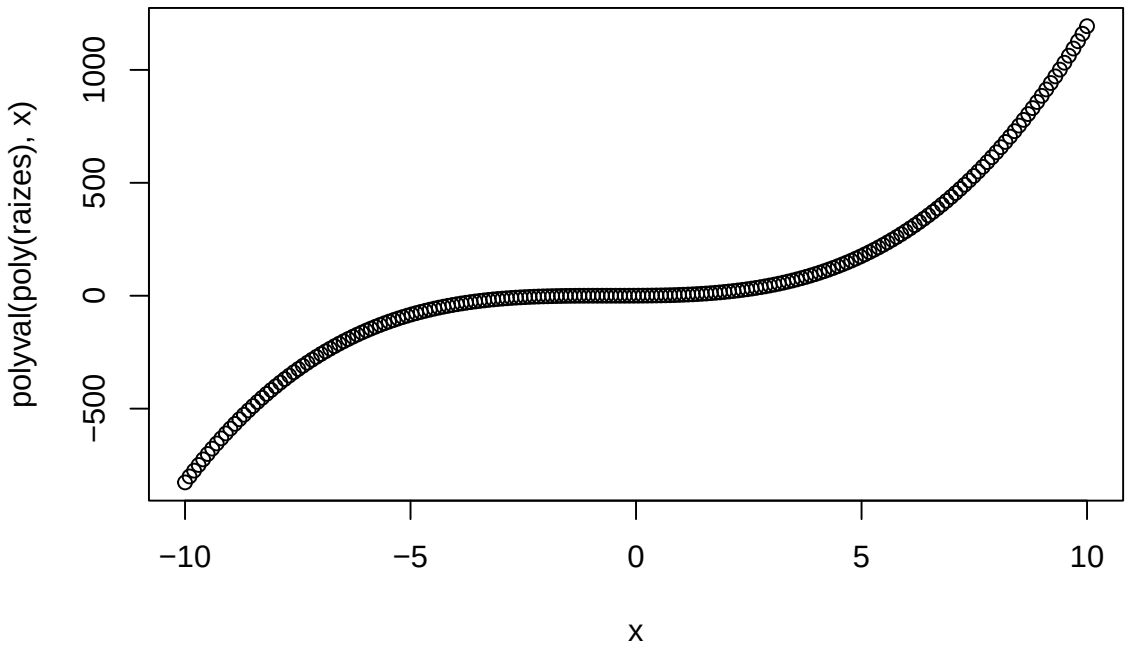
```
## Warning in xy.coords(x, y, xlabel, ylabel, log): partes imaginárias descartadas
## na coerção
```




```
## [[1]]
## [1]  0+1i -1+0i  0-1i
```

```
pol3(6,11,6,1)
```

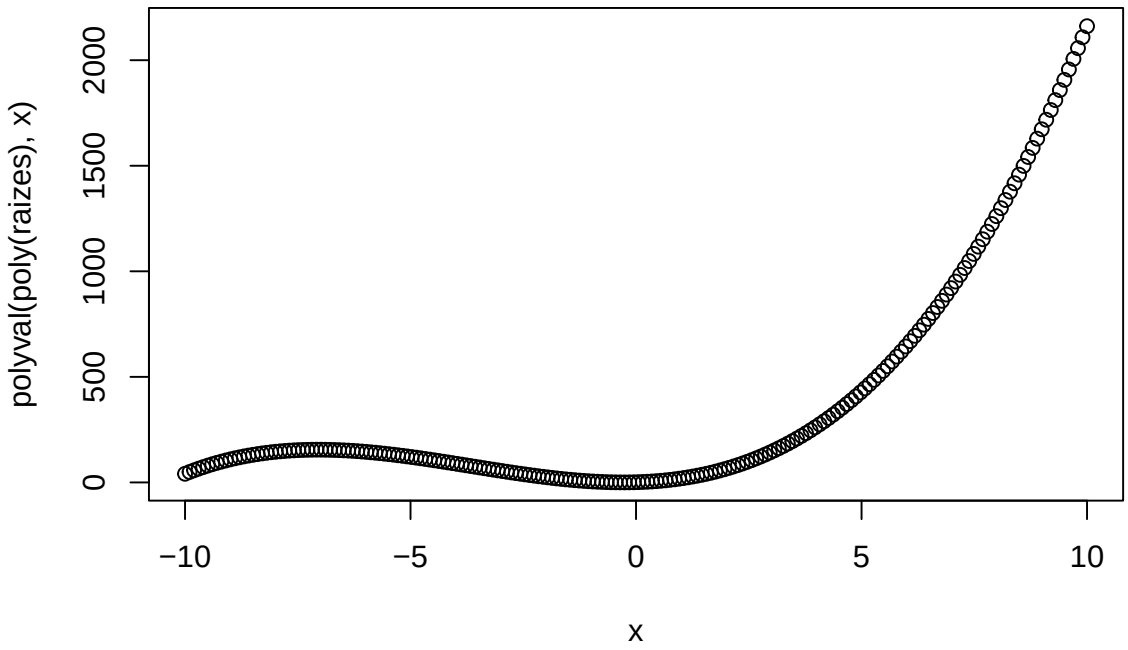
```
## Warning in xy.coords(x, y, xlabel, ylabel, log): partes imaginárias descartadas
## na coerção
```




```
## [[1]]
## [1] -0.3333333+0i -0.5000000+0i -1.0000000-0i
```

```
pol3(1,11,6,1)
```

```
## Warning in xy.coords(x, y, xlabel, ylabel, log): partes imaginárias descartadas
## na coerção
```




```
## [[1]]
## [1]  -0.2829249+0.1256689i  -0.2829249-0.1256689i -10.4341503+0.0000000i
```

```
pol3(1,3,3,1,grafico=F)
```

```
## [[1]]
## [1] -1+0i -1+0i -1-0i
```

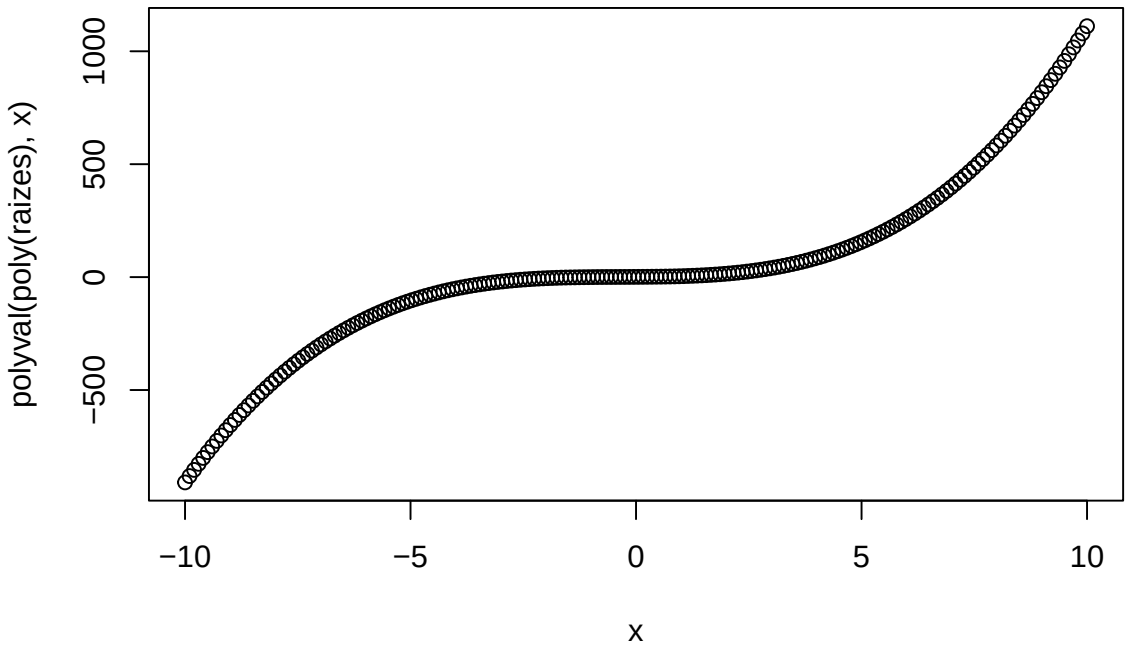
```
pol3(1,1,1,1,grafico =T ,resultado =T )
```

```
## Warning in xy.coords(x, y, xlabel, ylabel, log): partes imaginárias descartadas
## na coerção
```

```
## [[1]]
## [1]  0+1i -1+0i  0-1i
```

```
pol3(1,1,1,1,grafico =T ,resultado =F )
```

```
## Warning in xy.coords(x, y, xlabel, ylabel, log): partes imaginárias descartadas
## na coerção
```




```
pol3(1,1,1,1,grafico =F ,resultado =T )
```

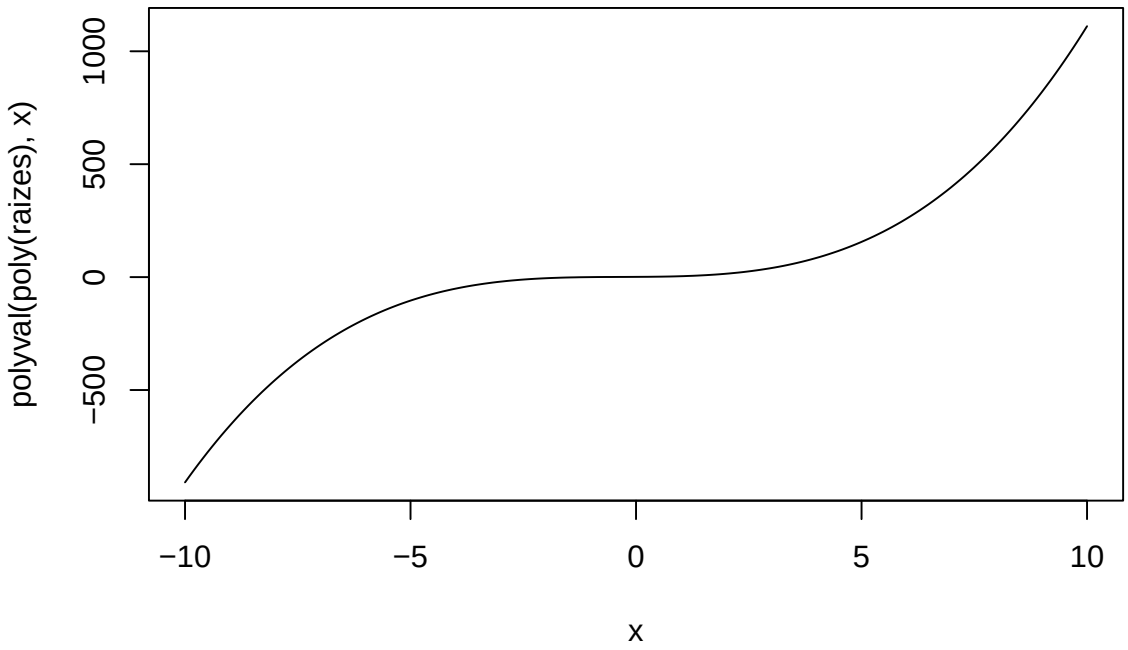
```
## [[1]]
## [1]  0+1i -1+0i  0-1i
```

```
pol3(1,1,1,1,grafico =F ,resultado =F )
```

```
## [1] "Girafas são criaturas sem coração"
```

```
pol3(1,1,1,1, type='l')
```

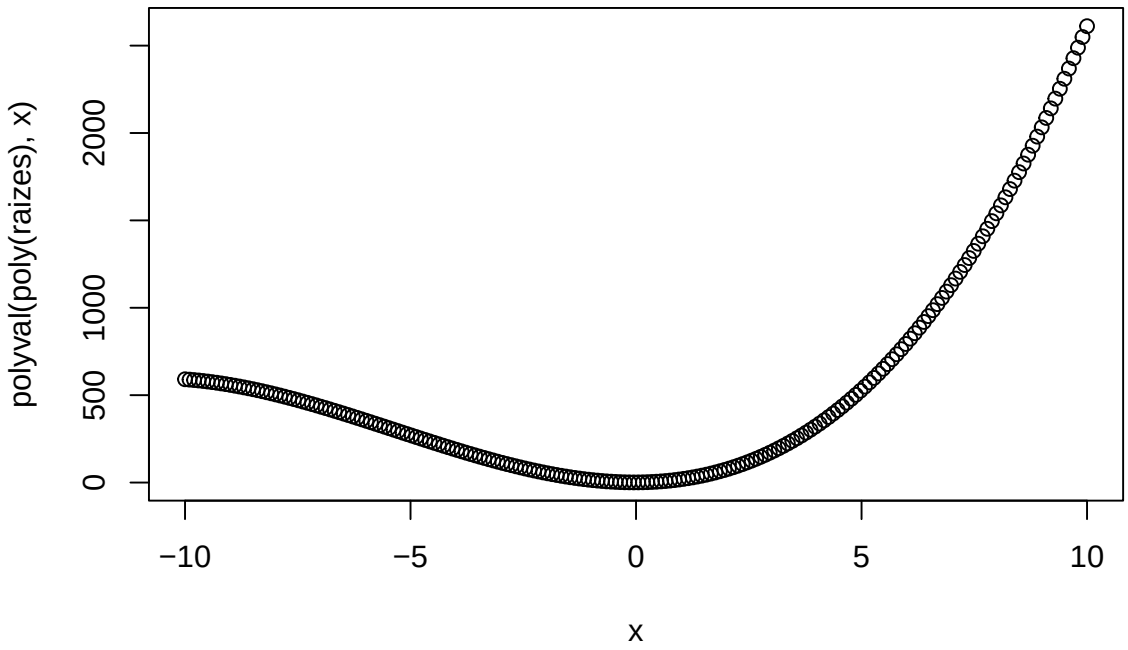
```
## Warning in xy.coords(x, y, xlabel, ylabel, log): partes imaginárias descartadas
## na coerção
```




```
## [[1]]
## [1]  0+1i -1+0i  0-1i
```

```
pol3(1,16,1,1)
```

```
## Warning in xy.coords(x, y, xlabel, ylabel, log): partes imaginárias descartadas
## na coerção
```



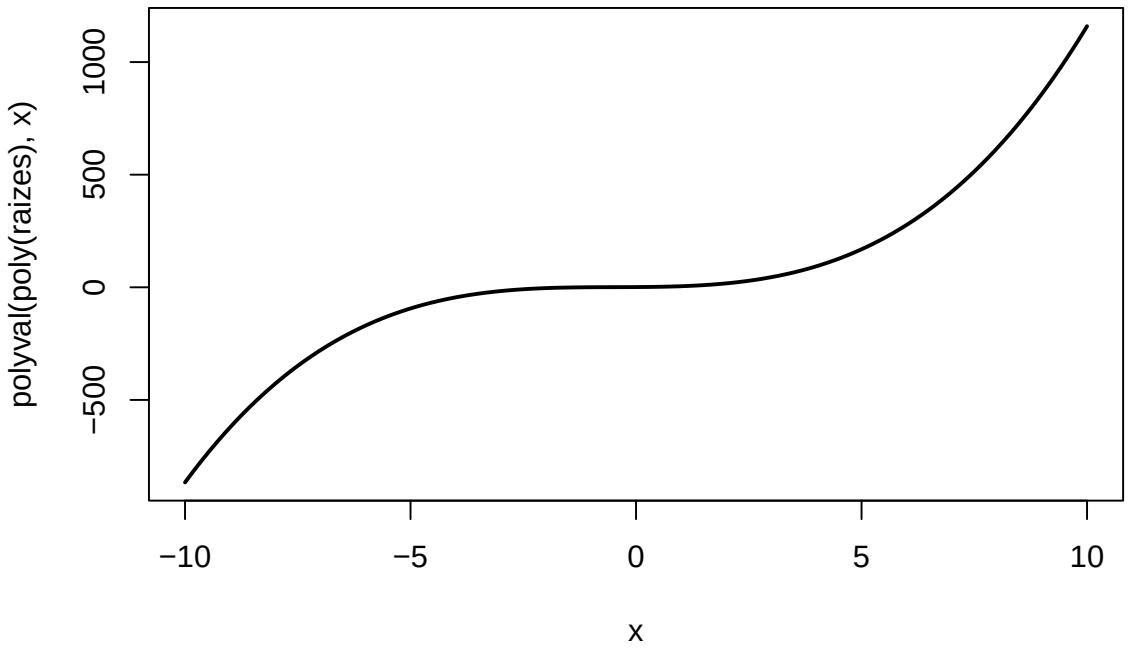

```
## [[1]]
## [1]  -0.0293977+0.2487294i  -0.0293977-0.2487294i -15.9412046-0.0000000i
```

```
pol3(1,5,1,1,grafico=F)
```

```
## [[1]]
## [1] -0.082012+0.4472779i -0.082012-0.4472779i -4.835976+0.0000000i
```

```
pol3(11,16,14,12, type='l',lwd='2')
```

```
## Warning in xy.coords(x, y, xlabel, ylabel, log): partes imaginárias descartadas
## na coerção
```




```
## [[1]]
## [1] -0.1444762+0.956584i -1.1655930+0.000000i -0.1444762-0.956584i
```

Questão 3) Com relação ao tópico do trabalho final do seu grupo, escreva sobre a utilidade do tema e explique 1 função interessante do pacote utilizado. (2 ponto)

Webscrapping, ou raspagem de dados, pode ser extremamente útil na vida do estatístico, cientista de dados ou qualquer um que queira manipular dados em *html*.

Basicamente, estamos acostumados a trabalhar com bancos de dados tabulados em formatos como *.csv*, *.xlsx* e até *.txt*, seja pelo `R`, seja utilizando outra ferramenta. Porém, as vezes queremos trabalhar com dados, especialmente aqueles disponíveis em sites, muitas vezes de nichos específicos e que não existe a opção de baixar ou obter os dados diretamente em formato de *excel* ou similar. Porém, utilizando técnicas de raspagem de dados, este impeditivo se torna facilmente superável.

Trabalhando com o `R`, dispomos de pacotes como o `selenium`, ou, como mostraremos em nossa apresentação, o pacote `rvest`, em conluio com o pacote `xml2`, que nos será útil na rotina de raspagem de dados da internet.

Com certeza, as funções mais auspciosas desses pacotes são a `rvest::html_nodes` ou `rvest::html_elements`, que servirão justamente para captar da marcação *html* as informações que gostaríamos de extrair em formato de texto ou numérico para enfim os manipular conforme vontade/necessidade.

É muito comum utilizar raspagem de dados para, por exemplo, fazer pesquisas de preços em sites que dispoem seu catálogo de produtos em formato *html* (basicamente quase todos os sites comerciais). Além disso, tem se popularizado o uso para coletar *tweets* da conta do *twitter* de determinadas pessoas, a fim de se fazer a análise desejável (text mining, núvem de palavras, etc).