Universidade Federal do Ceará (UFC/Sobral)

Aula 05 - Métodos Computacionais Aplicados

Prof. Weligton Gomes

17/04/2023

```
library(ISwR)
data(energy)
```

Fator

É comum em dados estatísticos ter variáveis categóricas, indicando alguma subdivisão dos dados, como classe social, etc. Essas são inseridas por meio de um código numérico. Essas variáveis devem ser especificadas como fatores em R.

```
pain <- c(0,3,2,2,1)
fpain <- factor(pain,levels=0:3)
levels(fpain) <- c("none","mild","medium","severe")

fpain
```

```
## [1] none   severe medium medium mild
## Levels: none mild medium severe
```

```
as.numeric(fpain)
```

```
## [1] 1 4 3 3 2
```

```
levels(fpain)
```

```
## [1] "none"   "mild"   "medium" "severe"
```

Listas

Às vezes, é útil combinar uma coleção de objetos em um objeto composto maior. Isso pode ser feito usando listas.

function list

intake = ingestão

```
intake.pre <- c(5260,5470,5640,6180,6390,6515,6805,7515,7515,8230,8770)
intake.post <- c(3910,4220,3885,5160,5645,4680,5265,5975,6790,6900,7335)
length(intake.pre)
```

```
## [1] 11
```

```
length(intake.post)
```

```
## [1] 11
```

Para combinar esses vetores individuais em uma lista, você pode dizer

```
mylist <- list(before=intake.pre,after=intake.post)
mylist
```

```
## $before
##  [1] 5260 5470 5640 6180 6390 6515 6805 7515 7515 8230 8770
##
## $after
##  [1] 3910 4220 3885 5160 5645 4680 5265 5975 6790 6900 7335
```

```
mylist$after
```

```
##  [1] 3910 4220 3885 5160 5645 4680 5265 5975 6790 6900 7335
```

Dataframe

Um quadro de dados corresponde ao que outros pacotes estatísticos chamam de "matriz de dados" ou "conjunto de dados".

É uma lista de vetores e / ou fatores do mesmo comprimento que estão relacionados "transversalmente" de forma que os dados na mesma posição venham da mesma unidade experimental (sujeito, animal, etc.). Além disso, ele possui um conjunto exclusivo de nomes de linhas.

```
d <- data.frame(intake.pre,intake.post)
d$intake.pre
```

```
##  [1] 5260 5470 5640 6180 6390 6515 6805 7515 7515 8230 8770
```

Indexação: Caso de um ou mais índices

Se você precisa de um elemento específico em um vetor, por exemplo, o indivíduo de número 5, você pode fazer:

```
intake.pre[5] #Caso de um elemento
```

```
## [1] 6390
```

```
intake.pre[5]<-6395
intake.pre[c(3,5,7)] #Caso de mais de um elemento do vetor
```

```
## [1] 5640 6395 6805
```

ou de outra forma;

```
v <- c(3,5,7)
intake.pre[v]
```

```
## [1] 5640 6395 6805
```

```
intake.pre[1:5] #No caso de uma sequência de elementos
```

```
## [1] 5260 5470 5640 6180 6395
```

```
intake.pre[-c(3,5,7)]
```

```
## [1] 5260 5470 6180 6515 7515 7515 8230 8770
```

```
#Apresenta todas as informações, exceto aquelas
#com a indexação ou número 3, 5 e 7.

intake.pre1<-intake.pre[-c(3,5,7)]
#Criação de um vetor excluindo alguns elementos do vetor original.
```

Exercícios de Fixação

Questão 01 - Mostrar comandos que podem ser usados para criar os objetos e/ou executar as instruções a seguir.

a) o vetor formado pelos elementos 4, 8, 2.

```
c(4,8,2)
```

```
## [1] 4 8 2
```

```
a<-c(4,8,2)
a
```

```
## [1] 4 8 2
```

b) selecionar o primeiro e terceiro elemento do vetor formado pelos elementos 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20.

```
#Forma "trabalhosa"
b1<-c(10, 11 ,12 ,13, 14, 15, 16, 17, 18, 19, 20)

b2<-seq(10,20,1)
b1[c(1,3)]
```

```
## [1] 10 12
```

```
b1[c(1, 3, 5, 7)]
```

```
## [1] 10 12 14 16
```

c) o vetor com a sequência de valores: -3, -2, -1, 0, 1, 2, 3.

```
seq(-3,3,1)
```

```
## [1] -3 -2 -1  0  1  2  3
```

```
c<-seq(-3,3,1)
c
```

```
## [1] -3 -2 -1  0  1  2  3
```

d) o vetor com a sequência de notas dos alunos: 2.4, 3.4, 4.4, 5.4, 6.4, 7.4, 8.4, 9.4, 10.4.

```
seq(2.4,10.4,1)
```

```
## [1]  2.4  3.4  4.4  5.4  6.4  7.4  8.4  9.4 10.4
```

e) o vetor: 1, 3, 5, 7, 9, 11, 13, 15, 17, 19, 21, 23, 25, 27, 29, 31, 33, 35, 37, 39.

```
seq(1,39,2)
```

```
##  [1]  1  3  5  7  9 11 13 15 17 19 21 23 25 27 29 31 33 35 37 39
```

f) o vetor de elementos repetidos: 1, 2, 3, 1, 2, 3, 1, 2, 3, 1, 2, 3.

rep(objeto, n vezes) Sequência: seq(), 1:3

```
f1<-1:3
f2<-seq(1,3,1)

rep(f1,4)
```

```
##  [1] 1 2 3 1 2 3 1 2 3 1 2 3
```

```
rep(1:3,4)
```

```
##  [1] 1 2 3 1 2 3 1 2 3 1 2 3
```

g) o vetor de sequência repetida: 1, 1, 1, 2, 2, 2, 3, 3, 3, 4, 4, 4.

```
rep(c(1,2,3),c(3,3,3))
```

```
## [1] 1 1 1 2 2 2 3 3 3
```

h) o vetor alfanumérico: “Parana”, “Sao Paulo”, “Minas Gerais”.

```
c("Parana", "Sao Paulo", "Minas Gerais")
```

```
## [1] "Parana"       "Sao Paulo"    "Minas Gerais"
```

Questão 02 - Faça o que se pede:

a) Crie um vetor com os nomes do alunos Pedro, João e Maria;

```
nomes<-c("Pedro", "João", "Maria")
```

b) Crie uma matriz 3x3 com os números pares iniciando em 2 e finalizando em 18. Ordenar pela linha.

criação de matriz: função matrix(objeto, nrow = , ncol = , byrow = TRUE)

```
matriz<-matrix(seq(2,18,2), ncol = 3, byrow = TRUE)
```

c) Crie uma lista com os objetos criados nos itens (a) e (b):

```
lista<-list(nomes, matriz)
lista
```

```
## [[1]]
## [1] "Pedro" "João"  "Maria"
##
## [[2]]
##      [,1] [,2] [,3]
## [1,]    2    4    6
## [2,]    8   10   12
## [3,]   14   16   18
```

d) Consultar apenas o objeto que se encontra na primeira posição da lista

```
lista[[1]]
```

```
## [1] "Pedro" "João"  "Maria"
```

e) Suponha que você digitou um nome errado e que em vez de maria o nome correto seria mariana. Obs: proceder com a alteração a partir da lista criada.

```
lista[[1]][3]<-"mariana"
```

f) Substitua o valor 10 por 100 na matriz que se encontra dentro da lista.

```
lista[[2]]
```

```
##      [,1] [,2] [,3]
## [1,]    2    4    6
## [2,]    8   10   12
## [3,]   14   16   18
```

```
lista[[2]][[2,2]]<-100
lista[[2]]
```

```
##      [,1] [,2] [,3]
## [1,]    2    4    6
## [2,]    8  100   12
## [3,]   14   16   18
```

Questão 03: Mostre comando(s) para construir uma matriz 10 × 10 tal que as entradas são iguais a i x j, sendo i a linha e j a coluna.

```
matriz2<-matrix(1:100, ncol=10, byrow = TRUE)
matriz3<-matrix(seq(1,100,1), ncol=10, byrow = TRUE)
```

Questão 04: Construa um data-frame com uma tabela com três colunas: x, x^2 e exp(x), com x variando de 0 a 50.

```
x<-0:50
x2<-x^2
x3<-exp(x)

dados<-data.frame(x,x2,x3)
dados
```

```
##     x   x2           x3
## 1   0    0 1.000000e+00
## 2   1    1 2.718282e+00
## 3   2    4 7.389056e+00
## 4   3    9 2.008554e+01
## 5   4   16 5.459815e+01
## 6   5   25 1.484132e+02
## 7   6   36 4.034288e+02
## 8   7   49 1.096633e+03
## 9   8   64 2.980958e+03
## 10  9   81 8.103084e+03
## 11 10  100 2.202647e+04
## 12 11  121 5.987414e+04
## 13 12  144 1.627548e+05
## 14 13  169 4.424134e+05
## 15 14  196 1.202604e+06
## 16 15  225 3.269017e+06
## 17 16  256 8.886111e+06
## 18 17  289 2.415495e+07
## 19 18  324 6.565997e+07
## 20 19  361 1.784823e+08
## 21 20  400 4.851652e+08
## 22 21  441 1.318816e+09
## 23 22  484 3.584913e+09
## 24 23  529 9.744803e+09
## 25 24  576 2.648912e+10
```

```
## 26 25  625 7.200490e+10
## 27 26  676 1.957296e+11
## 28 27  729 5.320482e+11
## 29 28  784 1.446257e+12
## 30 29  841 3.931334e+12
## 31 30  900 1.068647e+13
## 32 31  961 2.904885e+13
## 33 32 1024 7.896296e+13
## 34 33 1089 2.146436e+14
## 35 34 1156 5.834617e+14
## 36 35 1225 1.586013e+15
## 37 36 1296 4.311232e+15
## 38 37 1369 1.171914e+16
## 39 38 1444 3.185593e+16
## 40 39 1521 8.659340e+16
## 41 40 1600 2.353853e+17
## 42 41 1681 6.398435e+17
## 43 42 1764 1.739275e+18
## 44 43 1849 4.727839e+18
## 45 44 1936 1.285160e+19
## 46 45 2025 3.493427e+19
## 47 46 2116 9.496119e+19
## 48 47 2209 2.581313e+20
## 49 48 2304 7.016736e+20
## 50 49 2401 1.907347e+21
## 51 50 2500 5.184706e+21
```

```
View(dados)
print(dados)
```

```
##     x   x2          x3
## 1   0    0 1.000000e+00
## 2   1    1 2.718282e+00
## 3   2    4 7.389056e+00
## 4   3    9 2.008554e+01
## 5   4   16 5.459815e+01
## 6   5   25 1.484132e+02
## 7   6   36 4.034288e+02
## 8   7   49 1.096633e+03
## 9   8   64 2.980958e+03
## 10  9   81 8.103084e+03
## 11 10  100 2.202647e+04
## 12 11  121 5.987414e+04
## 13 12  144 1.627548e+05
## 14 13  169 4.424134e+05
## 15 14  196 1.202604e+06
## 16 15  225 3.269017e+06
## 17 16  256 8.886111e+06
## 18 17  289 2.415495e+07
## 19 18  324 6.565997e+07
## 20 19  361 1.784823e+08
## 21 20  400 4.851652e+08
## 22 21  441 1.318816e+09
## 23 22  484 3.584913e+09
## 24 23  529 9.744803e+09
```

```
## 25 24  576 2.648912e+10
## 26 25  625 7.200490e+10
## 27 26  676 1.957296e+11
## 28 27  729 5.320482e+11
## 29 28  784 1.446257e+12
## 30 29  841 3.931334e+12
## 31 30  900 1.068647e+13
## 32 31  961 2.904885e+13
## 33 32 1024 7.896296e+13
## 34 33 1089 2.146436e+14
## 35 34 1156 5.834617e+14
## 36 35 1225 1.586013e+15
## 37 36 1296 4.311232e+15
## 38 37 1369 1.171914e+16
## 39 38 1444 3.185593e+16
## 40 39 1521 8.659340e+16
## 41 40 1600 2.353853e+17
## 42 41 1681 6.398435e+17
## 43 42 1764 1.739275e+18
## 44 43 1849 4.727839e+18
## 45 44 1936 1.285160e+19
## 46 45 2025 3.493427e+19
## 47 46 2116 9.496119e+19
## 48 47 2209 2.581313e+20
## 49 48 2304 7.016736e+20
## 50 49 2401 1.907347e+21
## 51 50 2500 5.184706e+21
```

Seleção condicional

Na prática, você geralmente precisa extrair dados que satisfaçam certos critérios. Isso pode ser feito simplesmente inserindo uma expressão relacional em vez do índice.

```
intake.post[intake.pre > 7000]
```

```
## [1] 5975 6790 6900 7335
```

```
intake.post[intake.pre > 7000 & intake.pre <= 8000]
```

```
## [1] 5975 6790
```

Indexação de Dataframe

```
d <- data.frame(intake.pre,intake.post)
d[5,1]
```

```
## [1] 6395
```

```
d[5,]
```

```
##   intake.pre intake.post
## 5       6395        5645
```

```
d[d$intake.pre>7000,]
```

```
##    intake.pre intake.post
```

```
## 8        7515        5975
## 9        7515        6790
## 10       8230        6900
## 11       8770        7335
```

```
d[1:2,]
```

```
##   intake.pre intake.post
## 1       5260        3910
## 2       5470        4220
```

```
head(d)
```

```
##   intake.pre intake.post
## 1       5260        3910
## 2       5470        4220
## 3       5640        3885
## 4       6180        5160
## 5       6395        5645
## 6       6515        4680
```

```
tail(d)
```

```
##    intake.pre intake.post
## 6        6515        4680
## 7        6805        5265
## 8        7515        5975
## 9        7515        6790
## 10       8230        6900
## 11       8770        7335
```

```
data(energy)
energy$expend
```

```
##  [1]  9.21  7.53  7.48  8.08  8.09 10.15  8.40 10.88  6.13  7.90 11.51 12.79
## [13]  7.05 11.85  9.97  7.48  8.79  9.69  9.68  7.58  9.19  8.11
```

```
energy$stature
```

```
##  [1] obese lean  lean  lean  lean  lean  lean  lean  lean  lean  obese obese
## [13] lean  obese obese lean  obese obese obese lean  obese lean
## Levels: lean obese
```

Ordenando um vetor com a função sort()

```
intake.pre <- c(5260,5470,5640,6180,6390,6515,6805,7515,7515,8230,8770)
intake.pre_ord <- sort(c(5260,5470,5640,6180,6390,6515,6805,7515,7515,8230,8770))
```

Listar objetos

```
ls()
```

```
##  [1] "a"              "b1"             "b2"             "c"
##  [5] "d"              "dados"          "energy"         "f1"
##  [9] "f2"             "fpain"          "intake.post"    "intake.pre"
## [13] "intake.pre_ord" "intake.pre1"    "lista"          "matriz"
```

```
## [17] "matriz2"        "matriz3"        "mylist"         "nomes"
## [21] "pain"           "v"              "x"              "x2"
## [25] "x3"
```

Remover objetos

```
rm()
```

Dados agrupados e Dataframe

Separação de vetores por cada grupo (lean (magro) e obese(obeso), gasto (expend) e altura (stature))

```
exp.lean <- energy$expend[energy$stature=="lean"]
length(exp.lean)
```

```
## [1] 13
```

```
exp.obese <- energy$expend[energy$stature=="obese"]
length(exp.obese)
```

```
## [1] 9
```

Alternativamente, utiliza-se a função split() que gera uma lista de vetores de acordo com o agrupamento.

```
l <- split(energy$expend, energy$stature)
l
```

```
## $lean
##  [1]  7.53  7.48  8.08  8.09 10.15  8.40 10.88  6.13  7.90  7.05  7.48  7.58
## [13]  8.11
##
## $obese
## [1]  9.21 11.51 12.79 11.85  9.97  8.79  9.69  9.68  9.19
```

Uma aplicação comum de loops é aplicar uma função a cada elemento de um conjunto de valores ou vetores e coletar os resultados em uma única estrutura;

Função sapply, lapply e tapply

Função sapply - tenta simplificar (daí o 's'): A função sapply aplica funções a cada elemento de um vetor, pode ser aplicada em vetores unidimensionais, data.frames (agrupamentos de vetores) e listas (também considerados vetores). Por padrão a função simplifica os resultados se possível, então pode retornar tanto único vetor, uma matriz ou listas.

Função lapply - sempre retorna uma lista (daí o 'l'): A função lapply funciona praticamente igual à sapply e permite a aplicação de uma função em cada elemento de um vetor unidimensional, data.frame ou lista.

Função tapply permite que você crie tabelas (daí o 't'): A função tapply divide as estruturas de dados e aplica funções a cada subconjunto, geralmente as funções passada calculam estatísticas descritivas. O argumento INDEX especifica um ou mais fatores para dividir os elementos da estrutura. O funcionamento dessa função é similar as funções aggregate e by.

ou de outra forma:

Função tapply permite que você crie tabelas do valor de uma função em subgrupos definidos por seu segundo argumento, que pode ser um fator ou uma lista de fatores.

```
library(ISwR)
data(thuesen)

mean(thuesen$blood.glucose)
```

```
## [1] 10.3
```

```
mean(thuesen$short.velocity, na.rm = TRUE)
```

```
## [1] 1.325652
```

```
sapply(thuesen, mean, na.rm=T)
```

```
##  blood.glucose short.velocity
##      10.300000       1.325652
```

```
lapply(thuesen, sd, na.rm=TRUE)
```

```
## $blood.glucose
## [1] 4.337501
##
## $short.velocity
## [1] 0.2329031
```

```
tapply(energy$expend, energy$stature, mean)
```

```
##      lean     obese
##  8.066154 10.297778
```

Utilize a função sort( ) para ordenar valores em um vetor (intake = ingestão)

```
intake.pre <- c(5260,5470,5640,6180,6390,6515,6805,7515,7515,8230,8770)
intake.pre_sort <- sort(c(5260,5470,5640,6180,6390,6515,6805,7515,7515,8230,8770))
```

Ambiente R

```
ls()
```

```
##  [1] "a"              "b1"             "b2"             "c"
##  [5] "d"              "dados"          "energy"         "exp.lean"
##  [9] "exp.obese"      "f1"             "f2"             "fpain"
## [13] "intake.post"    "intake.pre"     "intake.pre_ord" "intake.pre_sort"
## [17] "intake.pre1"    "l"              "lista"          "matriz"
## [21] "matriz2"        "matriz3"        "mylist"         "nomes"
## [25] "pain"           "thuesen"        "v"              "x"
## [29] "x2"             "x3"
```

```
rm()
```

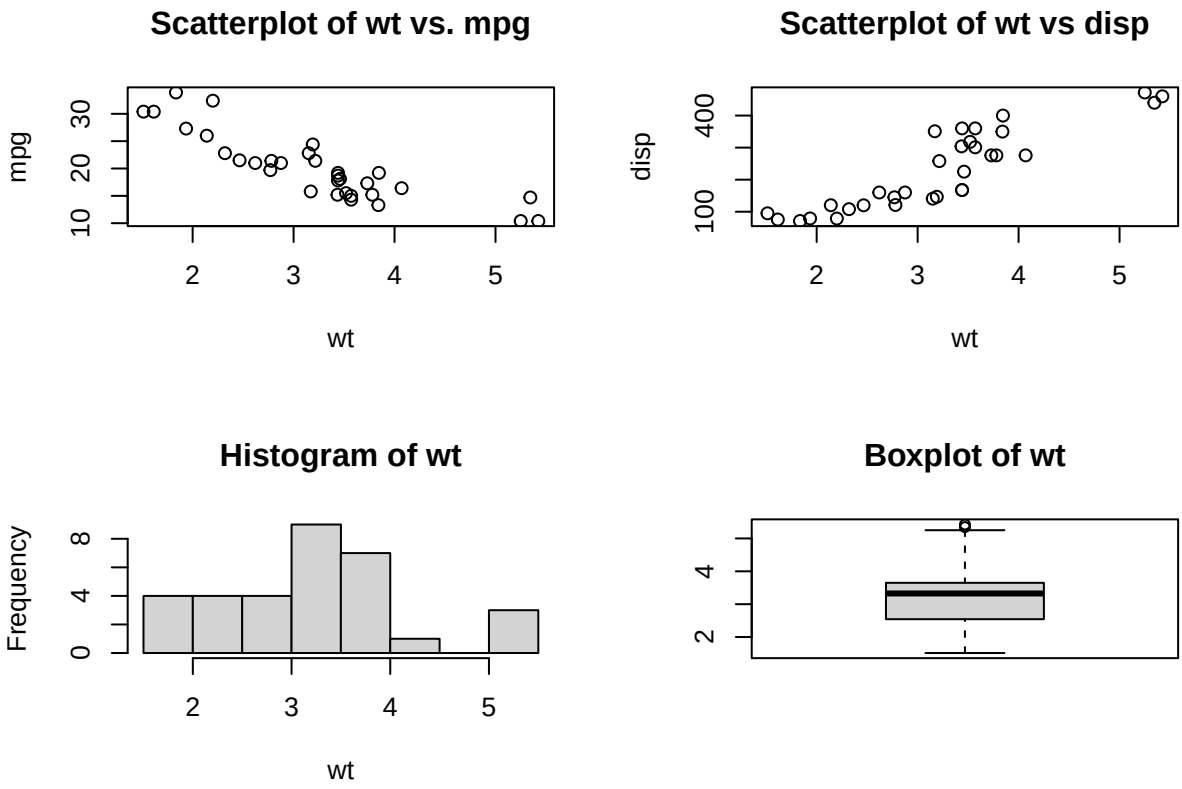
Funções Par ( ) e Layout ( )

Com a função par(), você pode incluir as opções mfrow=c(nrows, ncols) para criar uma matriz de gráficos de nrows x ncols que são ajustados pela linha.

mfcol=c(nrows, ncols) ajusta os gráficos na matriz por coluna.

Exemplo: # 4 figuras organizadas em 2 linhas e 2 colunas

```
attach(mtcars)
par(mfrow=c(2,2))
plot(wt,mpg, main="Scatterplot of wt vs. mpg")
plot(wt,disp, main="Scatterplot of wt vs disp")
hist(wt, main="Histogram of wt")
boxplot(wt, main="Boxplot of wt")
```

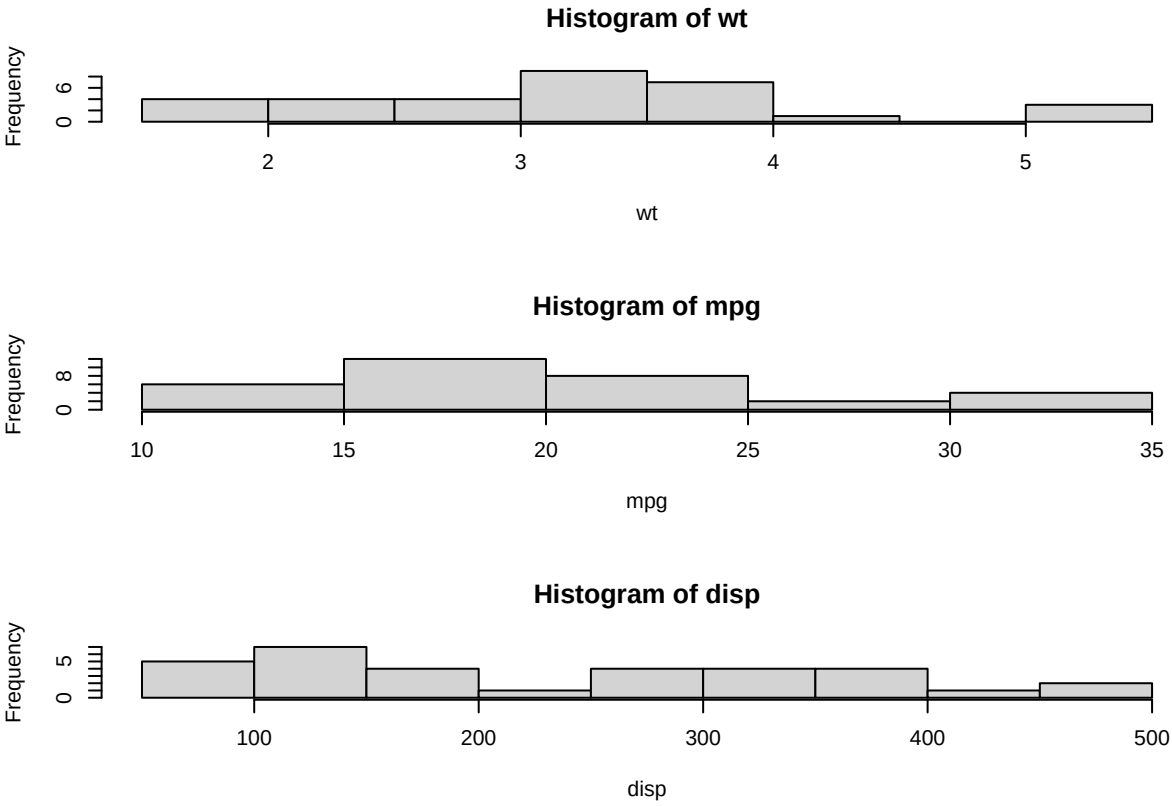


Exemplo: # 3 figuras organizadas em 3 linhas e 1 colunas

```
attach(mtcars)
```

```
## The following objects are masked from mtcars (pos = 3):
##
##     am, carb, cyl, disp, drat, gear, hp, mpg, qsec, vs, wt
```

```
par(mfrow=c(3,1))
hist(wt)
hist(mpg)
hist(disp)
```



A função layout() tem a forma layout(mat) onde mat é um objeto da matriz que especifica o local das N figuras para plotagem.

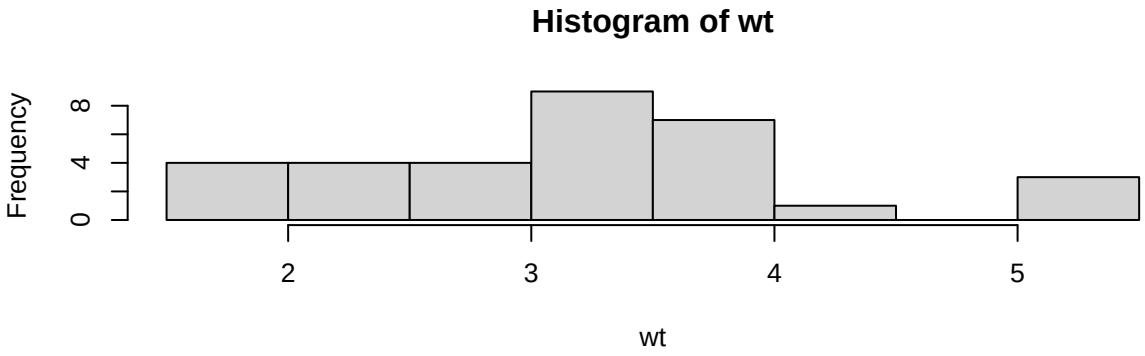
Exemplo: # Uma figura na linha 1 e duas figuras na linha 2

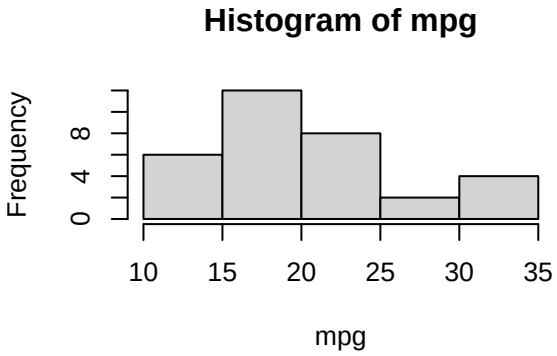
```
attach(mtcars)
```

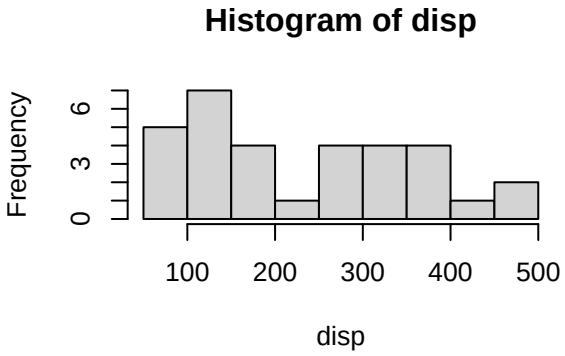
```
## The following objects are masked from mtcars (pos = 3):
##
##     am, carb, cyl, disp, drat, gear, hp, mpg, qsec, vs, wt
```

```
## The following objects are masked from mtcars (pos = 4):
##
##     am, carb, cyl, disp, drat, gear, hp, mpg, qsec, vs, wt
```

```
layout(matrix(c(1,1,2,3), 2, 2, byrow = TRUE))
hist(wt)
hist(mpg)
hist(disp)
```


Histogram of wt
Frequency
0
4
8
2
3
4
5
wt


Histogram of mpg
Frequency
0
4
8
10
15
20
25
30
35
mpg


Histogram of disp
Frequency
0
3
6
100
200
300
400
500
disp