Métodos Computacionais para Estatística e Otimização

Avaliação 1 - Luiz Henrique Barretta Francisco

1. Crie uma função em R que recebe suas matrizes quaisquer verifica se são compatíveis para a subtração. Se sim, retorna o resultado da subtração em uma nova matriz. Você deve usar a estrutura *for()*.

```
subtrair_matrizes <- function(A, B) {
  if (!all(dim(A) == dim(B))) {
    message("As matrizes não têm dimensões compatíveis para subtração.")
    return(NULL)}
  resultado <- matrix(0, nrow = nrow(A), ncol = ncol(A))
  for (i in 1:nrow(A)) {
    for (j in 1:ncol(A)) {
      resultado[i, j] <- A[i, j] - B[i, j]}}
  return(resultado)}

A <- matrix(1:9, nrow = 3)
B <- matrix(9:1, nrow = 3)
subtrair_matrizes(A, B)
```

```
##      [,1] [,2] [,3]
## [1,]   -8   -2    4
## [2,]   -6    0    6
## [3,]   -4    2    8
```

```
A <- matrix(1:9, nrow = 3)
B <- matrix(9:24, nrow = 4)
subtrair_matrizes(A, B)
```

```
## NULL
```

2. Considere o conjunto de dados mtcars disponível no pacote datasets do R. Carregue esse conjunto de dados e faça:

• Calcule as medidas descritivas mínimo, primeiro quartil, mediana, média, terceiro quartil e máximo para cada coluna que faça sentido calcular tais medidas.

```
data(mtcars)
summary(mtcars)
```

```
##       mpg             cyl             disp             hp
##  Min.   :10.40   Min.   :4.000   Min.   : 71.1   Min.   : 52.0
##  1st Qu.:15.43   1st Qu.:4.000   1st Qu.:120.8   1st Qu.: 96.5
##  Median :19.20   Median :6.000   Median :196.3   Median :123.0
##  Mean   :20.09   Mean   :6.188   Mean   :230.7   Mean   :146.7
```

```
##  3rd Qu.:22.80   3rd Qu.:8.000   3rd Qu.:326.0   3rd Qu.:180.0
##  Max.   :33.90   Max.   :8.000   Max.   :472.0   Max.   :335.0
##       drat             wt             qsec             vs
##  Min.   :2.760   Min.   :1.513   Min.   :14.50   Min.   :0.0000
##  1st Qu.:3.080   1st Qu.:2.581   1st Qu.:16.89   1st Qu.:0.0000
##  Median :3.695   Median :3.325   Median :17.71   Median :0.0000
##  Mean   :3.597   Mean   :3.217   Mean   :17.85   Mean   :0.4375
##  3rd Qu.:3.920   3rd Qu.:3.610   3rd Qu.:18.90   3rd Qu.:1.0000
##  Max.   :4.930   Max.   :5.424   Max.   :22.90   Max.   :1.0000
##        am              gear            carb
##  Min.   :0.0000   Min.   :3.000   Min.   :1.000
##  1st Qu.:0.0000   1st Qu.:3.000   1st Qu.:2.000
##  Median :0.0000   Median :4.000   Median :2.000
##  Mean   :0.4062   Mean   :3.688   Mean   :2.812
##  3rd Qu.:1.0000   3rd Qu.:4.000   3rd Qu.:4.000
##  Max.   :1.0000   Max.   :5.000   Max.   :8.000
```

- Capture o nome das linhas deste dataset e faça eles virarem uma nova coluna, chamada car_type.

```
mtcars$car_type <- rownames(mtcars)
rownames(mtcars) <- NULL
head(mtcars$car_type)
```

```
## [1] "Mazda RX4"         "Mazda RX4 Wag"     "Datsun 710"
## [4] "Hornet 4 Drive"    "Hornet Sportabout" "Valiant"
```

- Suponha que a coluna gear representa um variável categórica. Indique para o R a sua suposição.

```
mtcars$gear <- as.factor(mtcars$gear)
str(mtcars$gear)
```

```
##  Factor w/ 3 levels "3","4","5": 2 2 2 1 1 1 1 2 2 2 ...
```

- Calcule a média e o desvio-padrão da coluna mpg para cada nível da variável gear.

```
aggregate(mpg ~ gear, data = mtcars, FUN = mean)
```

```
##   gear      mpg
## 1    3 16.10667
## 2    4 24.53333
## 3    5 21.38000
```

```
aggregate(mpg ~ gear, data = mtcars, FUN = sd)
```

```
##   gear      mpg
## 1    3 3.371618
## 2    4 5.276764
## 3    5 6.658979
```

- Calcule a correlação de Pearson entre as colunas mpg e hp.

```r
cor(mtcars$mpg, mtcars$hp)
```

```
## [1] -0.7761684
```

• Calcule a média da coluna mpg para carros em que a coluna drat é maior que sua mediana (medianada coluna drat).

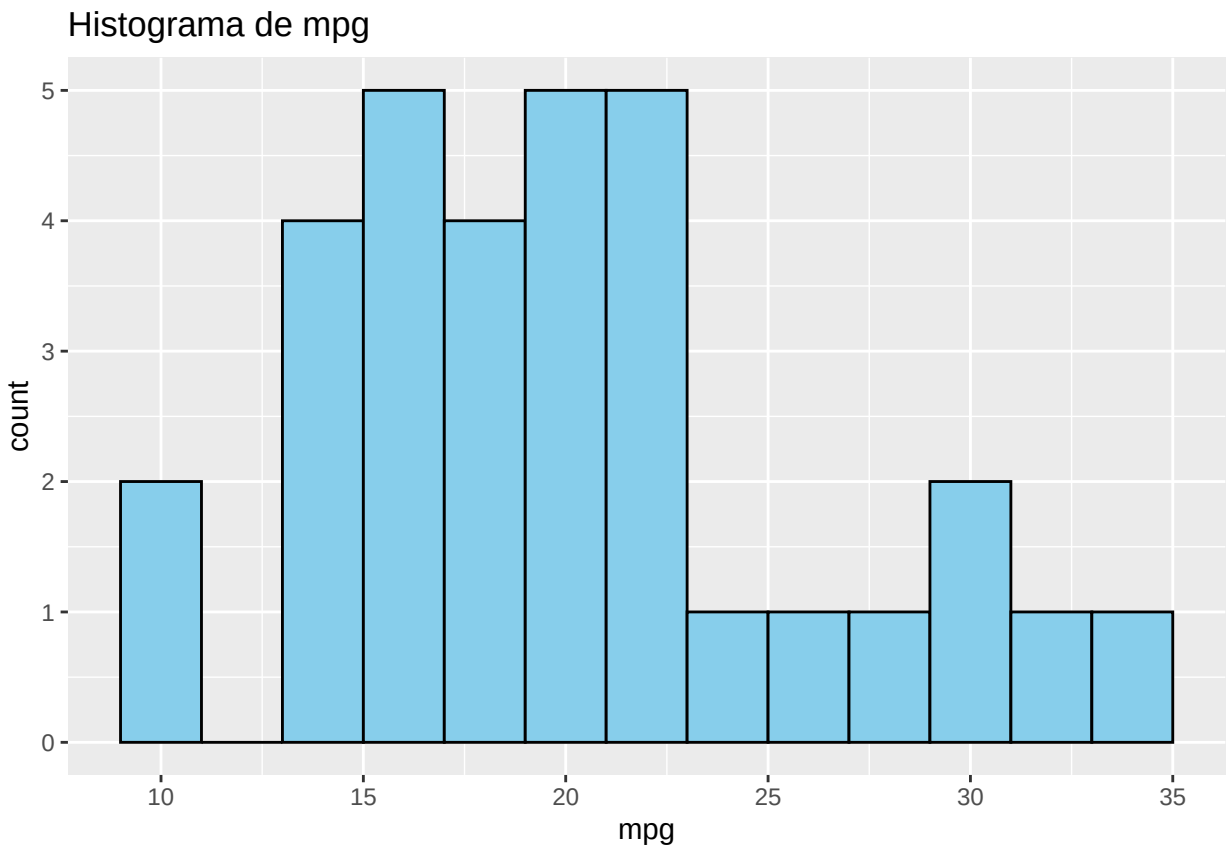
```r
mediana_drat <- median(mtcars$drat)
mean(mtcars$mpg[mtcars$drat > mediana_drat])
```

```
## [1] 23.5625
```

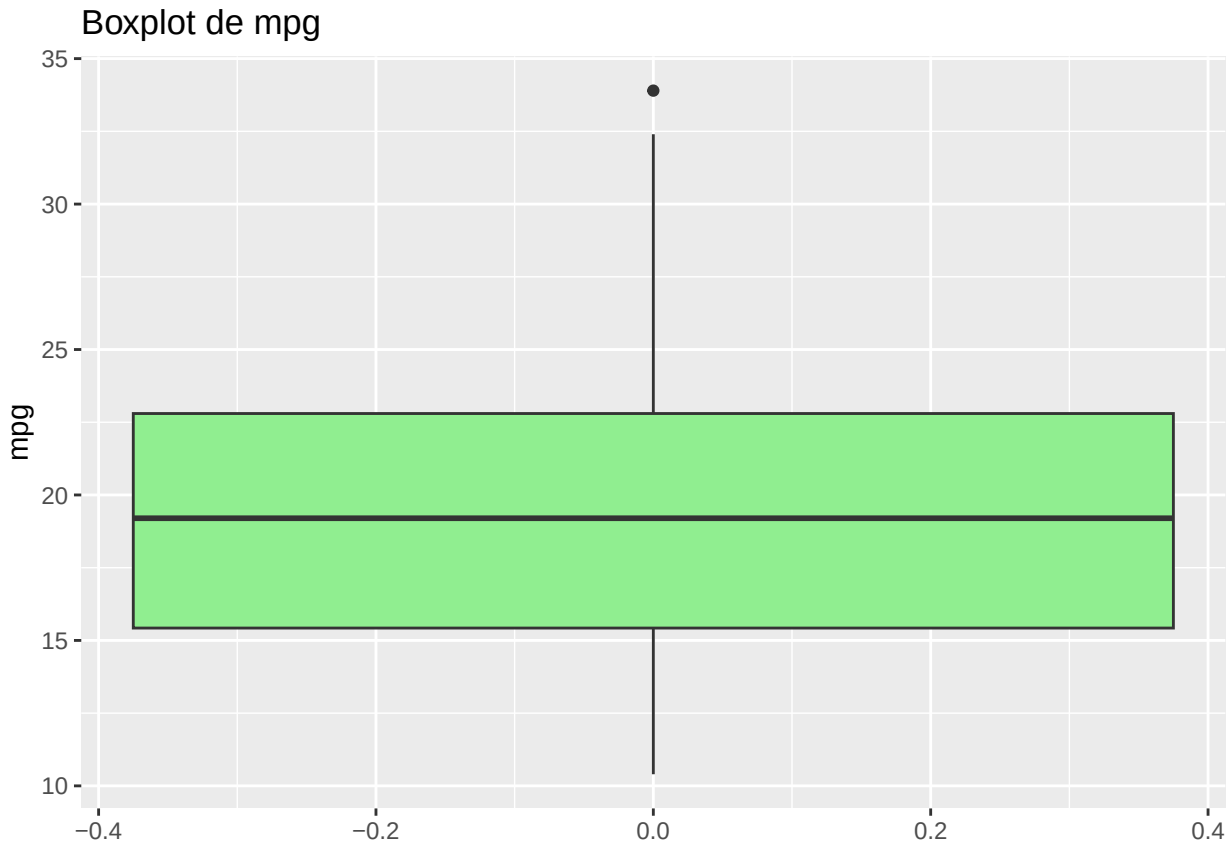
3. Considere o conjunto de dados mtcars disponível no pacote datasets do R. Carregue esse conjunto dedados e faça:

• Selecione uma coluna numérica e faça suas representações gráficas adequadas.

```r
library(ggplot2)

ggplot(mtcars, aes(x = mpg)) +
  geom_histogram(binwidth = 2, fill = "skyblue", color = "black") +
  ggtitle("Histograma de mpg")
```

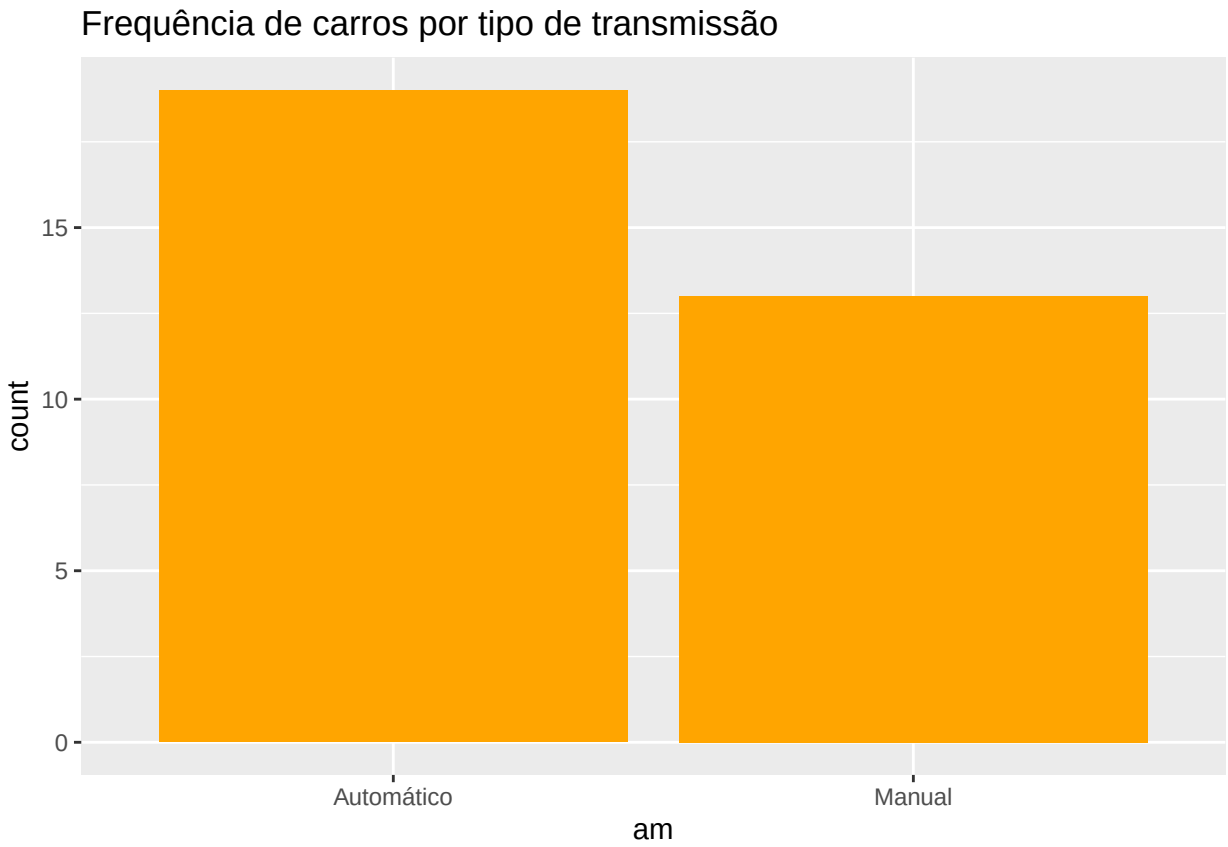


```r
ggplot(mtcars, aes(y = mpg)) +
  geom_boxplot(fill = "lightgreen") +
  ggtitle("Boxplot de mpg")
```



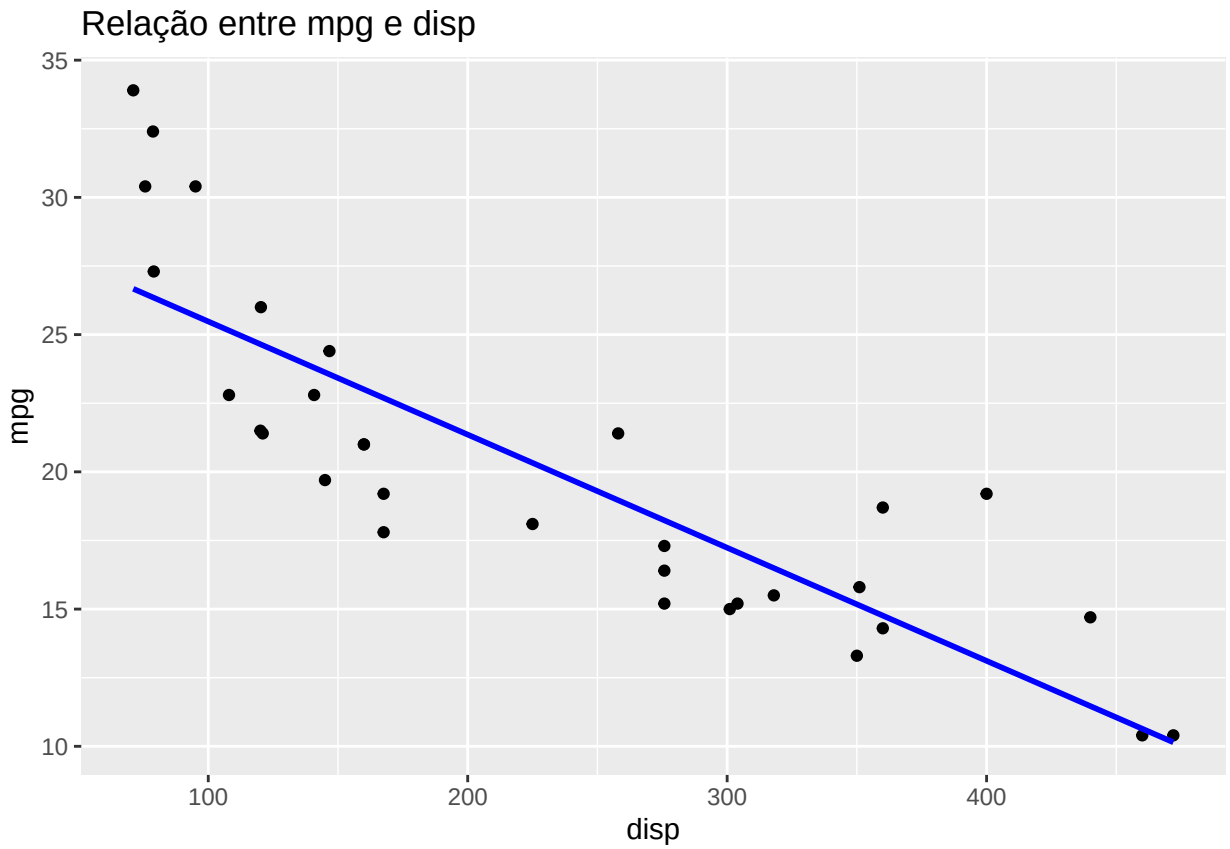
• Considere que as colunas vs, am, gear e carb são categóricas (qualitativas ordinais). Escolha uma delas e faça uma representação gráfica adequada.

```r
mtcars$am <- factor(mtcars$am, labels = c("Automático", "Manual"))
ggplot(mtcars, aes(x = am)) +
  geom_bar(fill = "orange") +
  ggtitle("Frequência de carros por tipo de transmissão")
```



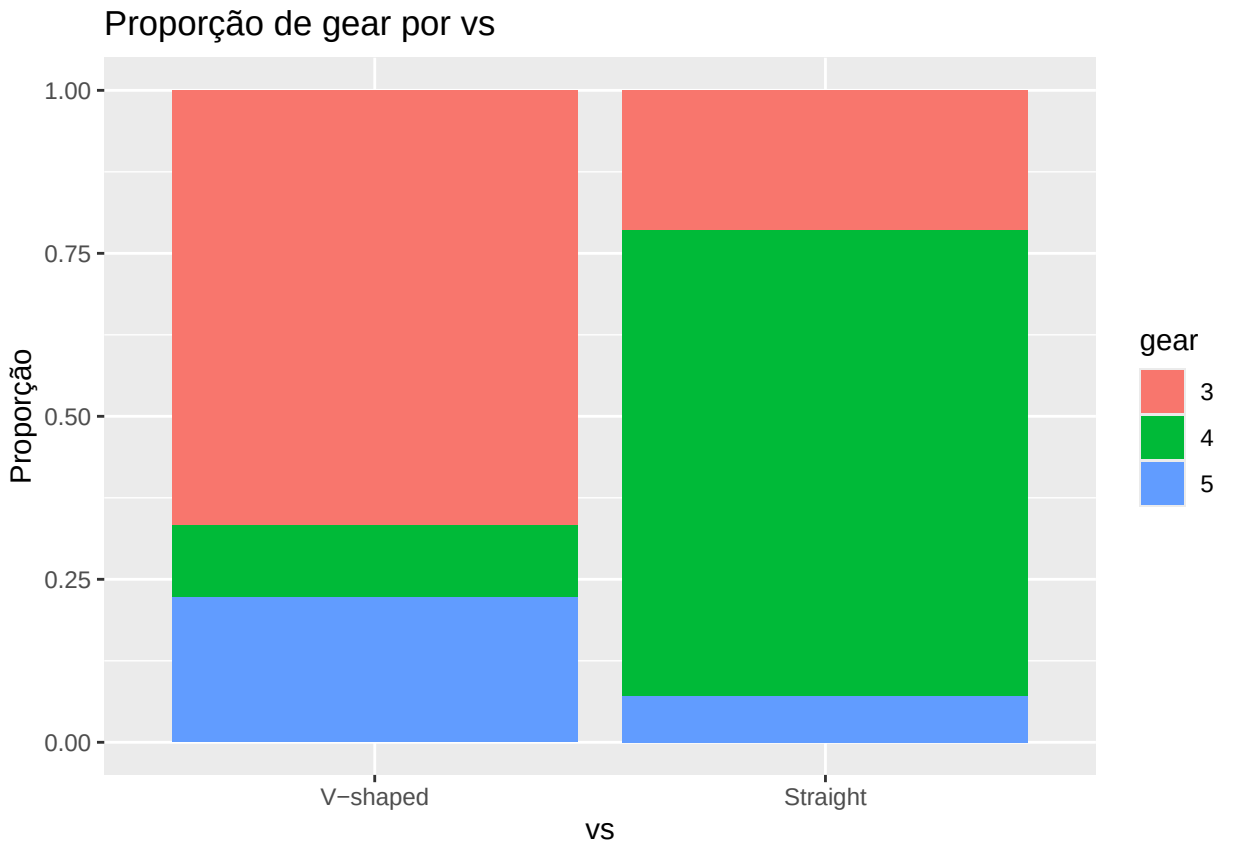
• Suponha que é de interesse verificar a relação entre as colunas mpg e disp. Faça uma representação gráfica adequada para ilustrar tal relação.

```
ggplot(mtcars, aes(x = disp, y = mpg)) +
  geom_point() +
  geom_smooth(method = "lm", se = FALSE, color = "blue") +
  ggtitle("Relação entre mpg e disp")
```



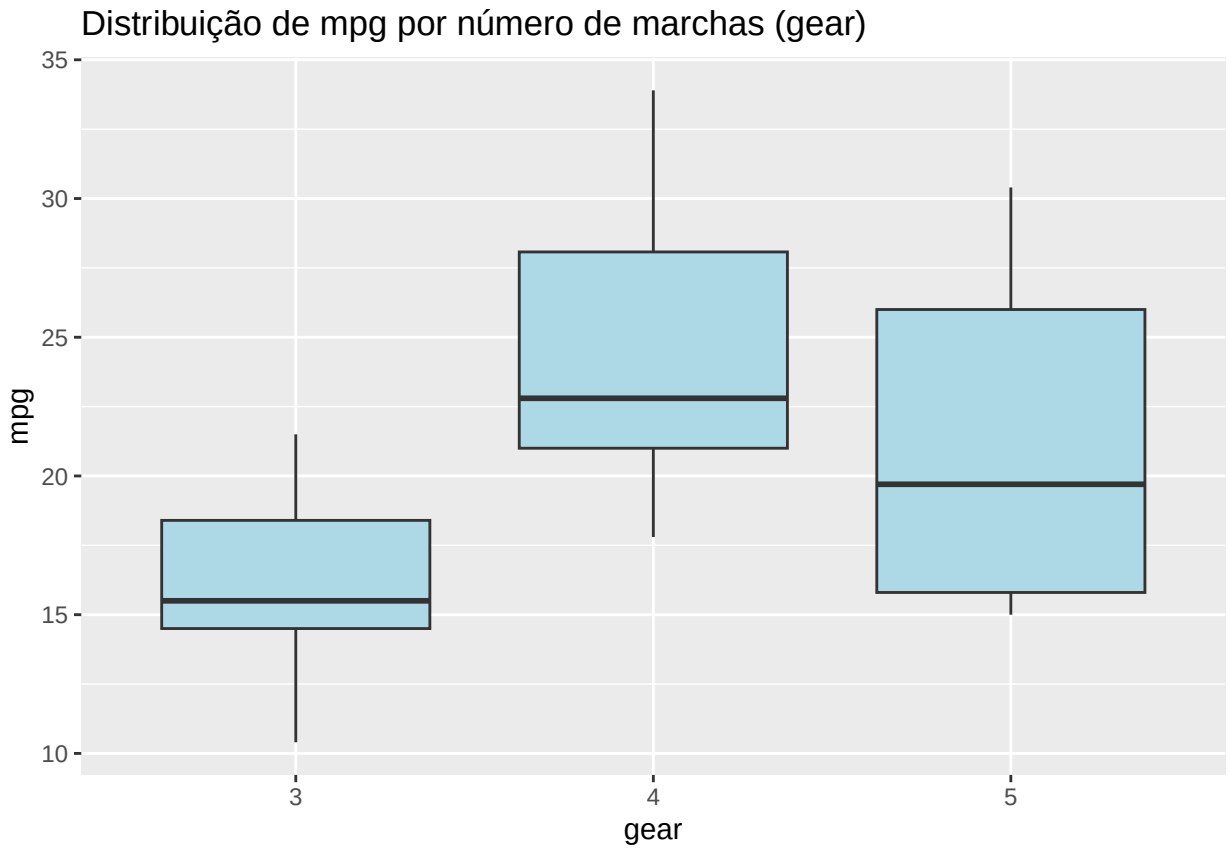
• Suponha que é de interesse verificar a relação entre as colunas vs e gear. Faça uma representação gráfica adequada para ilustrar tal relação.

```
mtcars$vs <- factor(mtcars$vs, labels = c("V-shaped", "Straight"))
mtcars$gear <- factor(mtcars$gear)
ggplot(mtcars, aes(x = vs, fill = gear)) +
  geom_bar(position = "fill") +
  ylab("Proporção") +
  ggtitle("Proporção de gear por vs")
```



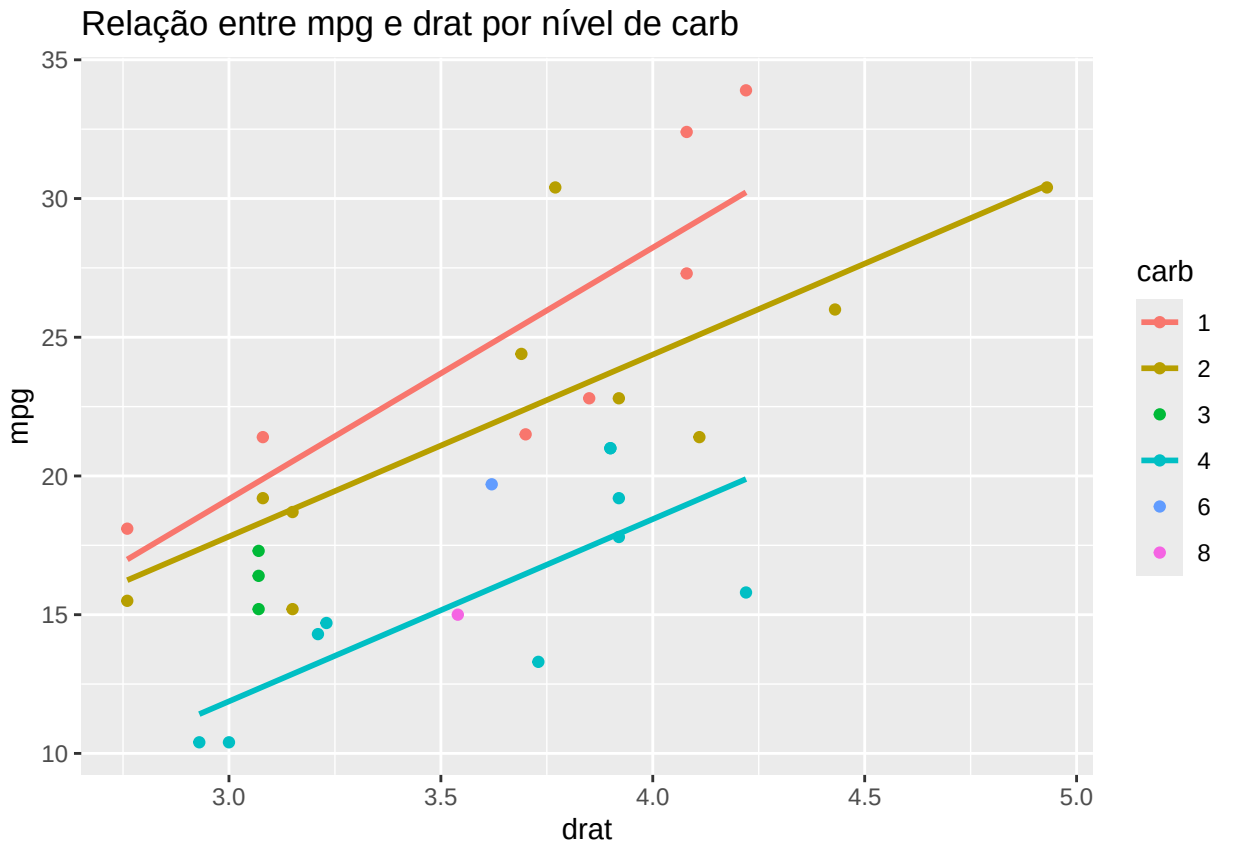
- Suponha que é de interesse verificar a relação entre gear e mpg faça uma representação gráfica adequada.

```
ggplot(mtcars, aes(x = gear, y = mpg)) +
  geom_boxplot(fill = "lightblue") +
  ggtitle("Distribuição de mpg por número de marchas (gear)")
```



• Suponha que é de interesse verificar se a relação entre mpg e drat muda de acordo com os níveis da variável carb. Faça uma representação gráfica adequada para visualizar tal relação.

```
mtcars$carb <- factor(mtcars$carb)
ggplot(mtcars, aes(x = drat, y = mpg, color = carb)) +
  geom_point() +
  geom_smooth(method = "lm", se = FALSE) +
  ggtitle("Relação entre mpg e drat por nível de carb")
```



4. Crie uma função em C++ que recebe suas matrizes compatíveis para a subtração e retorna uma nova matriz com o resultado. Use a estrutura for de forma similar a que você usou em R. Use sua função em R usando o pacote Rcpp.

```
library(Rcpp)

cppFunction('
NumericMatrix subtrairMatrizesRcpp(NumericMatrix A, NumericMatrix B) {
  int nrow = A.nrow(), ncol = A.ncol();
  NumericMatrix resultado(nrow, ncol);

  if (nrow != B.nrow() || ncol != B.ncol()) {
    stop("As matrizes não são compatíveis para subtração.");
  }

  for (int i = 0; i < nrow; i++) {
    for (int j = 0; j < ncol; j++) {
      resultado(i, j) = A(i, j) - B(i, j);
    }
  }
  return resultado;
}
')

A <- matrix(1:9, nrow = 3)
```

```r
B <- matrix(9:1, nrow = 3)

subtrairMatrizesRcpp(A, B)
```

```
##      [,1] [,2] [,3]
## [1,]   -8   -2    4
## [2,]   -6    0    6
## [3,]   -4    2    8
```

5. Use a biblioteca Armadillo para criar uma função em R que faça a subtração entre duas matrizes.

```r
library(RcppArmadillo)

Rcpp::cppFunction(depends = "RcppArmadillo", code = '
arma::mat subtrairArmadillo(arma::mat A, arma::mat B) {
  if (A.n_rows != B.n_rows || A.n_cols != B.n_cols) {
    Rcpp::stop("As matrizes não são compatíveis para subtração.");
  }
  return A - B;
}
')

A <- matrix(1:9, nrow = 3)
B <- matrix(9:1, nrow = 3)

subtrairArmadillo(A, B)
```

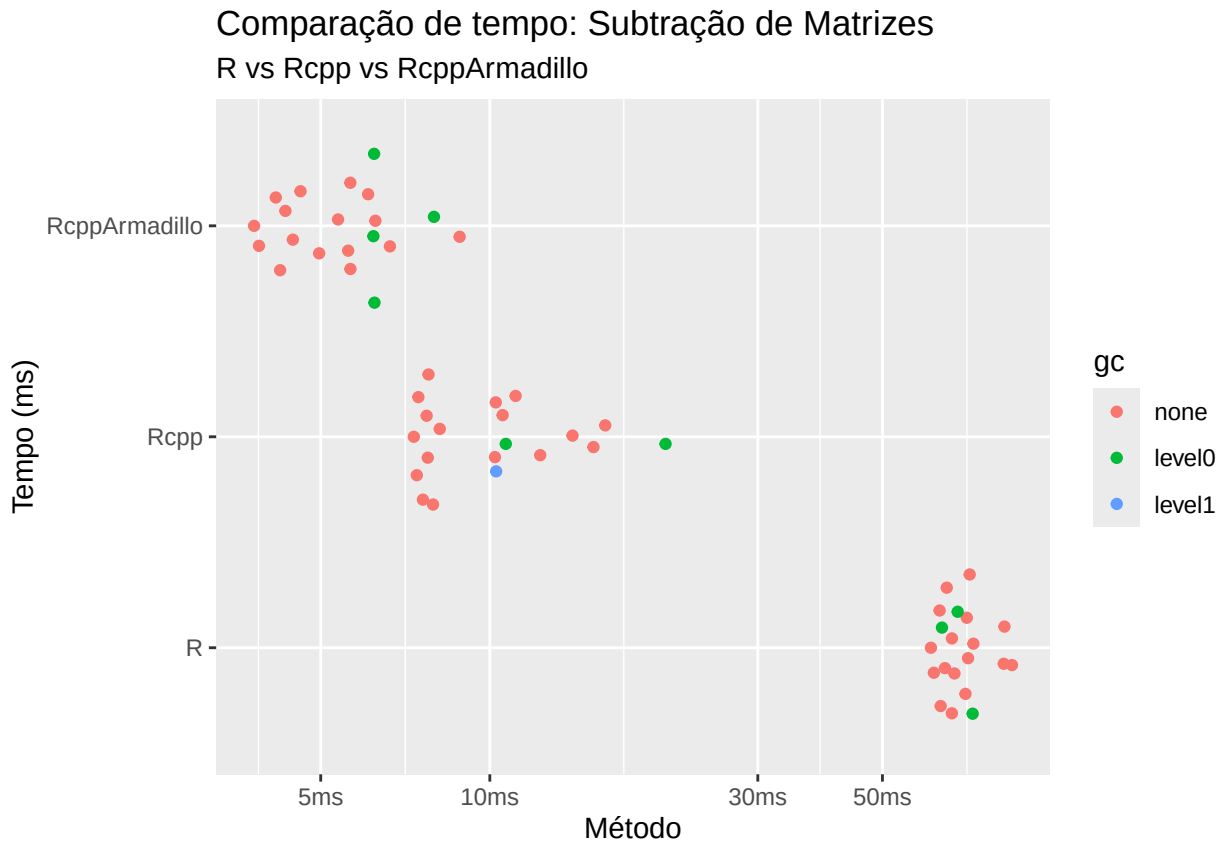
```
##      [,1] [,2] [,3]
## [1,]   -8   -2    4
## [2,]   -6    0    6
## [3,]   -4    2    8
```

6. Compare o tempo computacional das três funções criadas anteriormente para fazer a subtração entre duas matrizes. Reporte o seus resultados com uma tabela e representação gráfica. Interprete os resultados.

```r
library(bench)
library(dplyr)

set.seed(123)
A <- matrix(runif(1e6), nrow = 1000)
B <- matrix(runif(1e6), nrow = 1000)

resultados <- bench::mark(
  R = subtrair_matrizes(A, B),
  Rcpp = subtrairMatrizesRcpp(A, B),
  RcppArmadillo = subtrairArmadillo(A, B),
  iterations = 20,
  check = TRUE)

summary(resultados)
```

```
## # A tibble: 3 x 6
##   expression         min   median `itr/sec` mem_alloc `gc/sec`
##   <bch:expr>    <bch:tm> <bch:tm>     <dbl> <bch:byt>    <dbl>
## 1 R               61ms  67.17ms      14.3    7.63MB     2.53
## 2 Rcpp          7.32ms   8.14ms     101.     7.63MB    17.8
## 3 RcppArmadillo 3.81ms   5.17ms     189.     7.63MB    47.3
```

```
autoplot(resultados) +
  labs(title = "Comparação de tempo: Subtração de Matrizes",
       subtitle = "R vs Rcpp vs RcppArmadillo",
       y = "Tempo (ms)", x = "Método")
```



Com base na tabela e no gráfico gerado pelo *bench::mark*, é possível observar diferenças significativas de desempenho entre as três abordagens de subtração de matrizes. A implementação em R puro foi a mais lenta, com tempo mediano de execução em torno de 59 milissegundos, e a menor taxa de iterações por segundo. Em contraste, a versão com Rcpp obteve um desempenho substancialmente melhor, reduzindo o tempo para cerca de 7,7 milissegundos, o que representa uma taxa de execução quase 8 vezes maior. A abordagem mais eficiente foi a com RcppArmadillo, que alcançou tempo mediano de apenas 4 milissegundos e 228 execuções por segundo, destacando-se pela superioridade na velocidade.

O gráfico confirma essas diferenças, tornando visível a clara vantagem das abordagens em C++, especialmente com RcppArmadillo. Isso se deve à eficiência computacional da biblioteca Armadillo, que é otimizada para operações matriciais de baixo nível. Em resumo, para tarefas computacionalmente intensivas como subtração de grandes matrizes, o uso de RcppArmadillo é claramente a escolha mais vantajosa.