INF 1514

Introdução à Análise de Dados

Material Gráficos

PUC
RIO

Gráficos

- O R já vem com funções básicas que fazem gráficos estatísticos de todas as naturezas.
- Essas funções são rápidas e simples de usar.
- Além das funções do R, vamos utilizar as funções do pacote **ggplot2**.
- Outros pacotes que também podem ser utilizados são o **lattice** e o **ggvis**.

Gráfico de dispersão

- Para construir um **gráfico de dispersão**, utilizamos a função **plot()**, cujos principais parâmetros são:
  - x, y - vetores com os dados dos eixos x e y, respectivamente.
  - *type* - tipo de gráfico, podendo ser pontos, linhas, escada, entre outros.

```
n <- 200
x <- 1:n
y <- 10 + 3 * x
plot(x, y)
```

Uma reta formada por 200 pares (x, y).

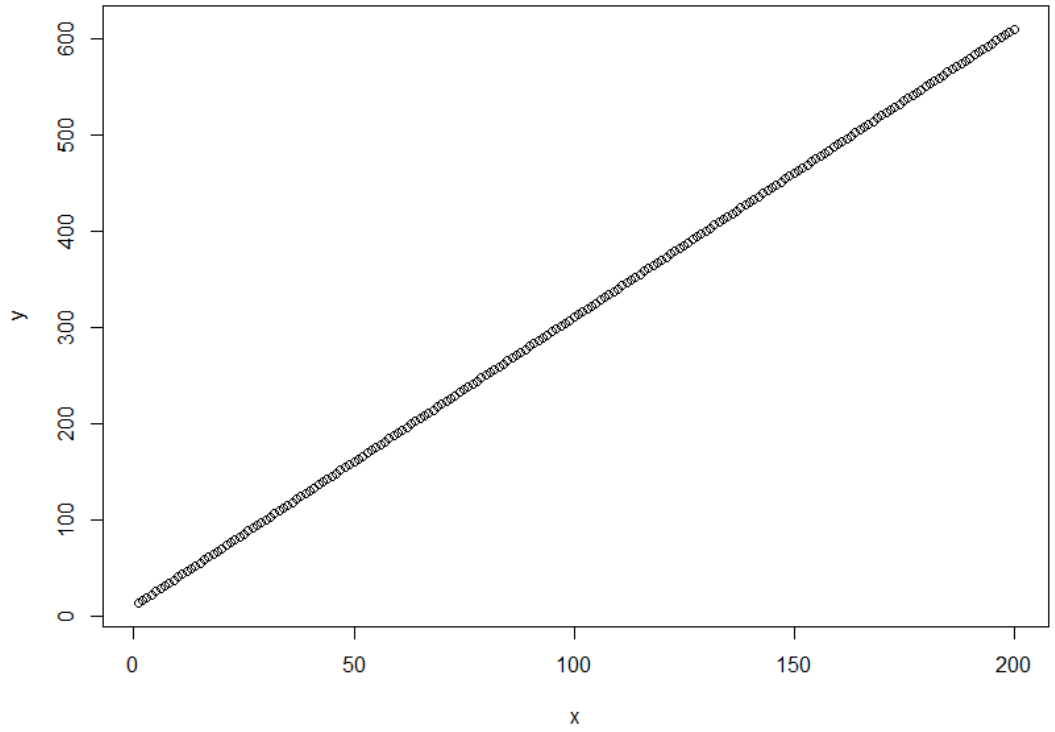


Gráfico de dispersão

- Para construir um **gráfico de dispersão**, utilizamos a função **plot()**, cujos principais parâmetros são:
  - x, y - vetores com os dados dos eixos x e y, respectivamente.
  - *type* - tipo de gráfico, podendo ser pontos, linhas, escada, entre outros.

```
n <- 200
x <- 1:n
y <- 10 + 3 * x
y <- y + rnorm(n, sd = 50)
plot(x, y)
```

rnorm é uma função que simula variáveis normais.
Sua forma de uso é: x <- rnorm(n, mean, sd)

- n é o tamanho da amostra;
- mean é a média (opcional com default 0);
- sd é o desvio-padrão (opcional com default 1);

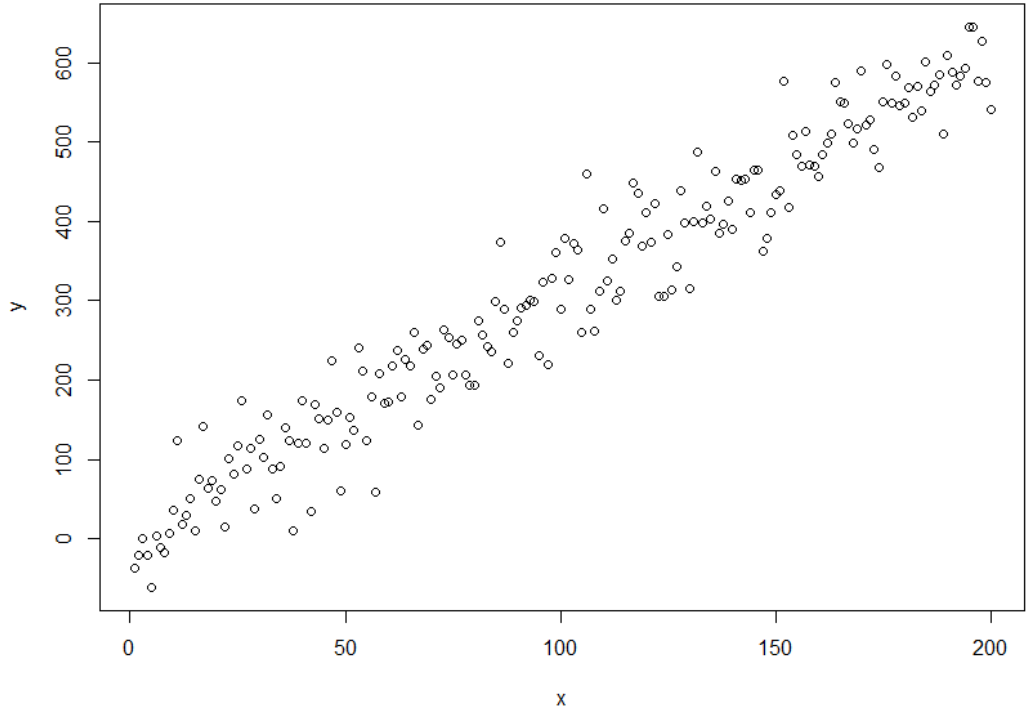
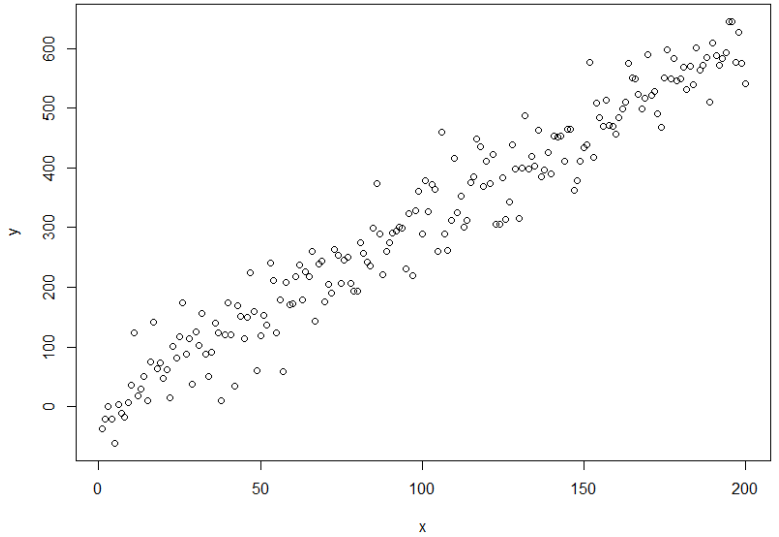


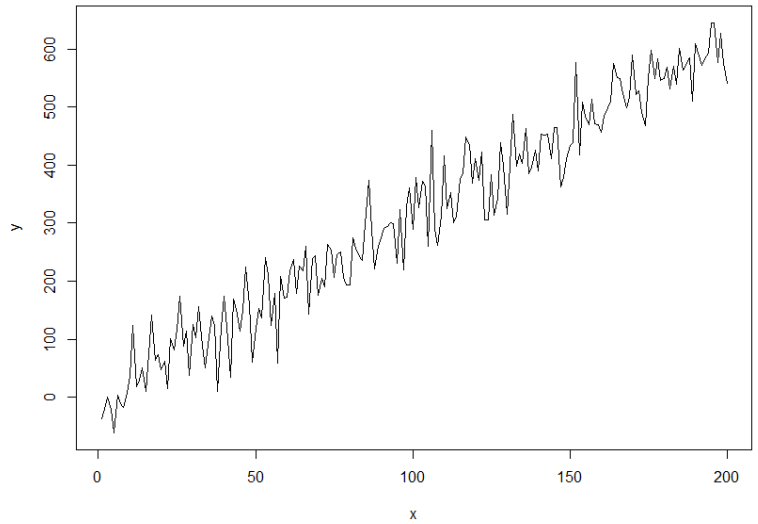
Gráfico de dispersão

- Para construir um **gráfico de dispersão**, utilizamos a função **plot()**, cujos principais parâmetros são:
  - x, y - vetores com os dados dos eixos x e y, respectivamente.
  - *type* - tipo de gráfico, podendo ser pontos, linhas, escada, entre outros.

```
n <- 200
x <- 1:n
y <- 10 + 3 * x + rnorm(n, sd = 50)
plot(x, y)
```




```
n <- 200
x <- 1:n
y <- 10 + 3 * x + rnorm(n, sd = 50)
plot(x, y, type = "l")
```

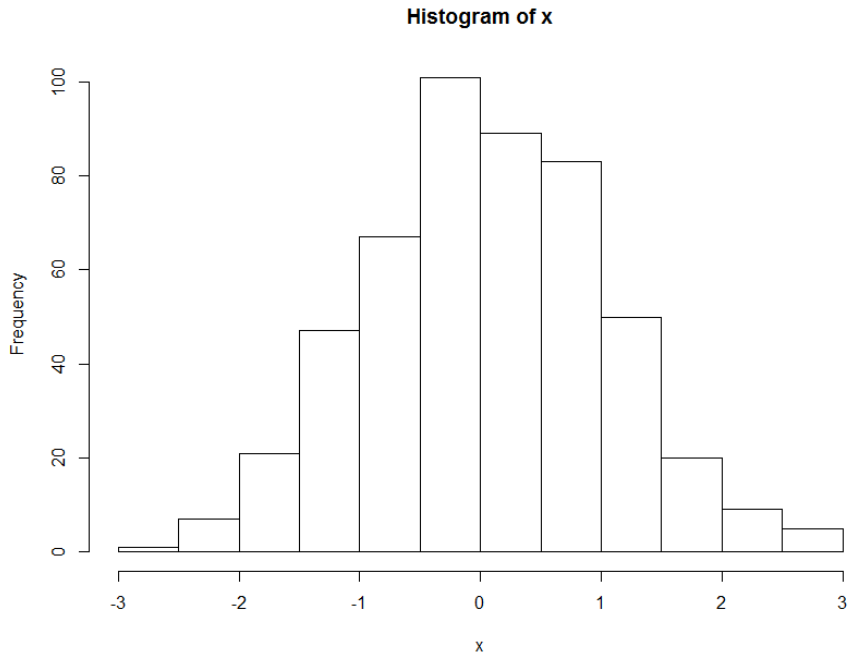


Histograma

- Para construir **histogramas**, utilizamos a função **hist()**, cujos principais parâmetros são:
  - x - o vetor numérico para o qual o histograma será construído.
  - *breaks* - o número aproximado de barras.

```
x <- rnorm(500)
hist(x)
```

Média 0 e desvio padrão 1.

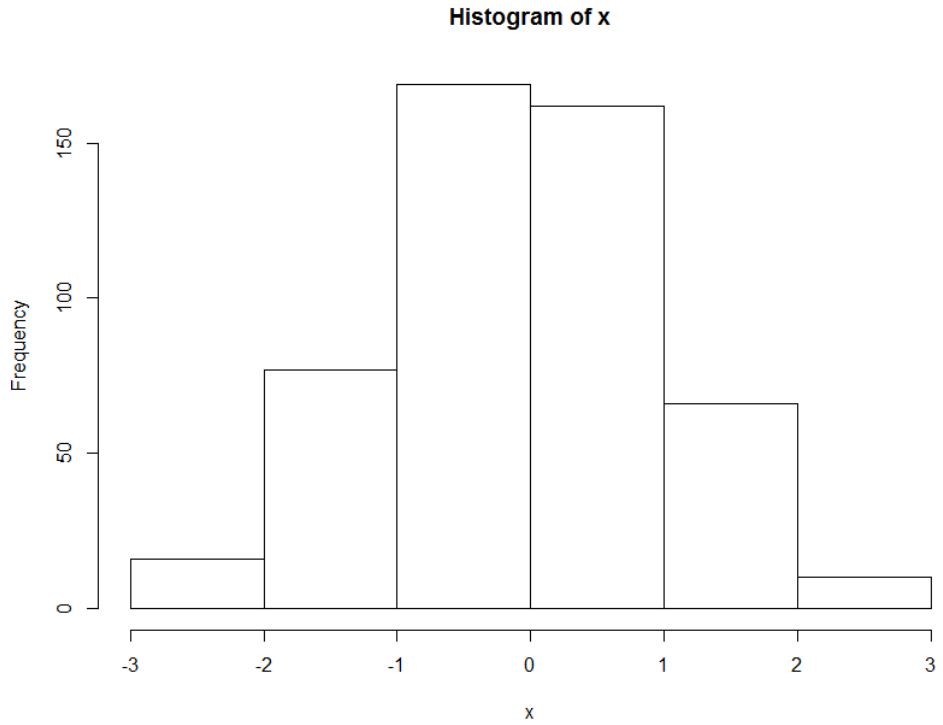


Histograma

- Para construir **histogramas**, utilizamos a função **hist()**, cujos principais parâmetros são:
  - x - o vetor numérico para o qual o histograma será construído.
  - *breaks* - o número aproximado de barras.

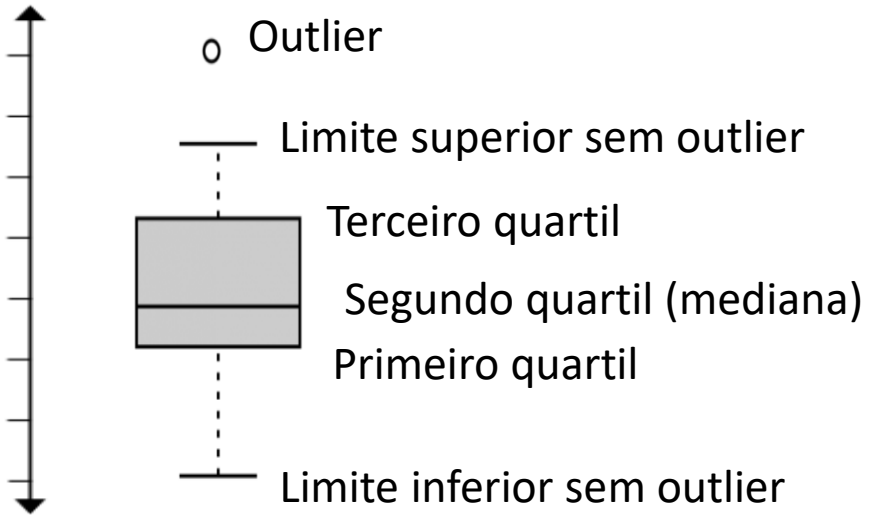
```
x <- rnorm(500)
hist(x, breaks = 6)
```

Histograma com 6 barras.



Boxplot

- O ***boxplot*** é um gráfico utilizado para estudar o comportamento de variáveis, sendo composto por:
  - Uma caixa (*box*) que representa a região entre o primeiro e o terceiro quartis (quantis 25% e 75%), ou seja, 50% dos dados estão dentro da caixa.
  - Uma linha dentro da caixa que representa a posição da mediana (segundo quartil ou quantil 50%).
  - Linhas que se prolongam a partir da caixa até no máximo 1,5 vezes a distância interquartil (diferença entre o 1º e 3º quartis).
  - As observações que passarem essa distância são representadas individualmente por pontos.



Boxplot

- Para construir ***boxplots***, utilizamos a função **boxplot()**, cujos principais parâmetros são:
  - x - o vetor numérico para o qual o ***bloxplot*** será construído.

```
> idade <- c(8, 9, 12, 20, 22, 18, 21, 18, 35, 23, 10, 12, 17)
> summary(idade)
Min. 1st Qu. Median Mean 3rd Qu. Max.
8.00 12.00 18.00 17.31 21.00 35.00
> boxplot(idade, main="Boxplot: Idade", col="blue")
```

Suponhamos um vetor contendo uma amostra da idade de pessoas registradas em um determinado plano de saúde.

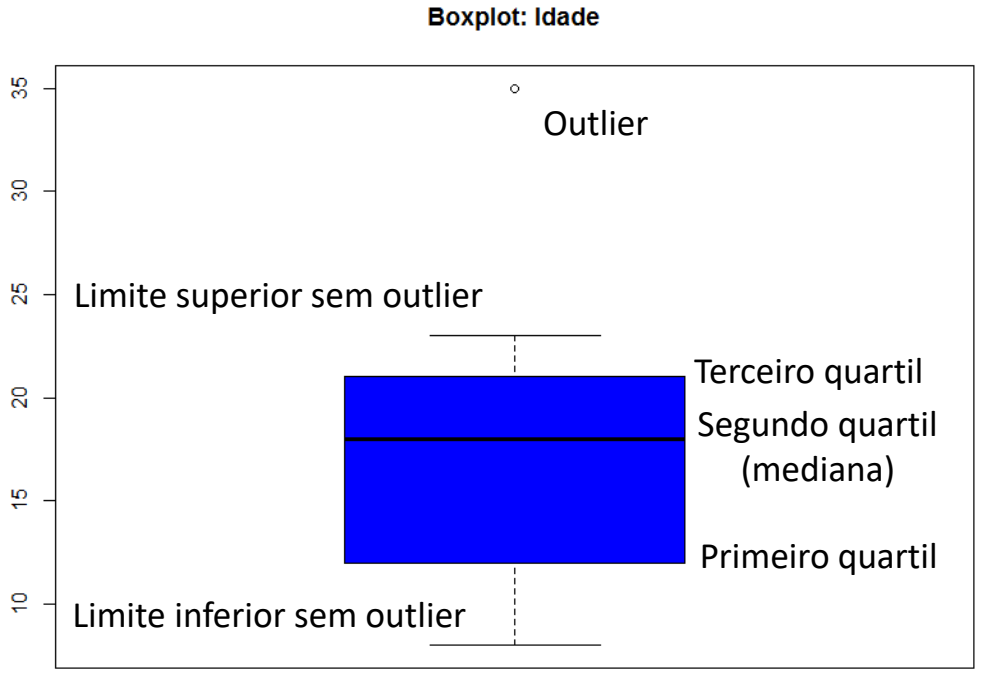


Gráfico de barras

- Uma forma simples de descrever quantitativamente observações é agrupá-las em categorias e contar quantas observações pertencem a cada categoria.
- No R a forma mais simples e direta de obter contagens (frequências) é através da função **table()**.
- Para construir **gráficos de barras**, utilizamos as funções **table()** e **barplot()** combinadas.

Gráfico de barras

- Para construir **gráficos de barras**, utilizamos as funções **table()** e **barplot()** combinadas.

```
filhos <- c(0, 2, 1, 3, 2, 0, 1, 3, 1, 1, 1, 0, 0)
tabela <- table(filhos)
barplot(tabela, main = "Paternidade",
          xlab = "Quantidade de filhos", ylab = "Quantidade de pessoas")
```

Suponhamos um vetor contendo uma amostra do número de filhos de pessoas registradas em um determinado plano de saúde.

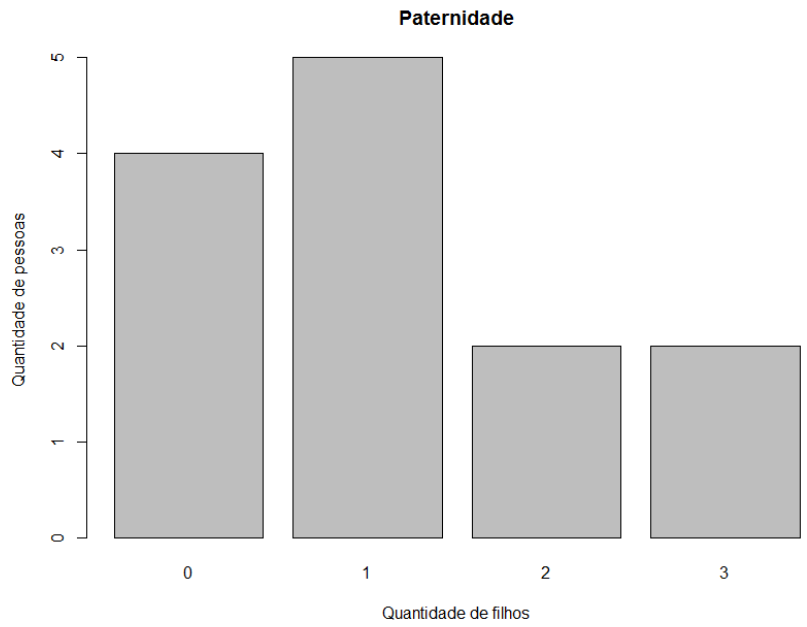
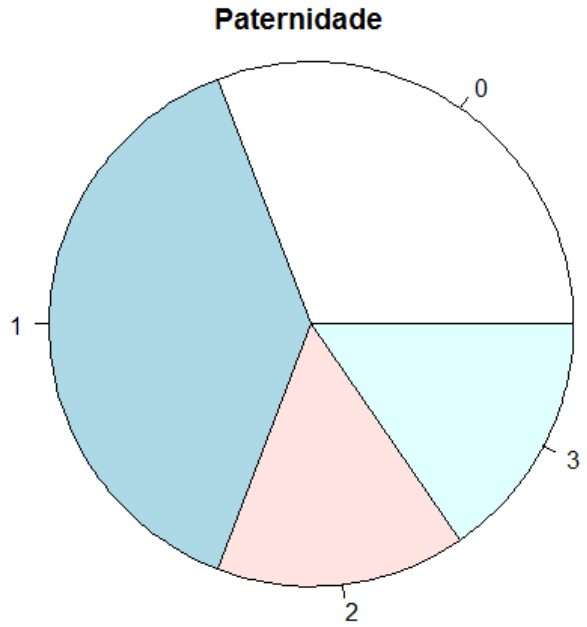


Gráfico de setores

- **Gráficos de setores (pizza)** são normalmente utilizados quando se quer demonstrar a representatividade de cada categoria de uma determinada variável na constituição do total.
- Para construir **setores**, utilizamos a função **pie()**.

```
filhos <- c(0, 2, 1, 3, 2, 0, 1, 3, 1, 1, 1, 0, 0)
tabela <- table(filhos)
pie(tabela, main = "Paternidade")
```

Suponhamos um vetor contendo uma amostra do número de filhos de pessoas registradas em um determinado plano de saúde.



Exemplo 1

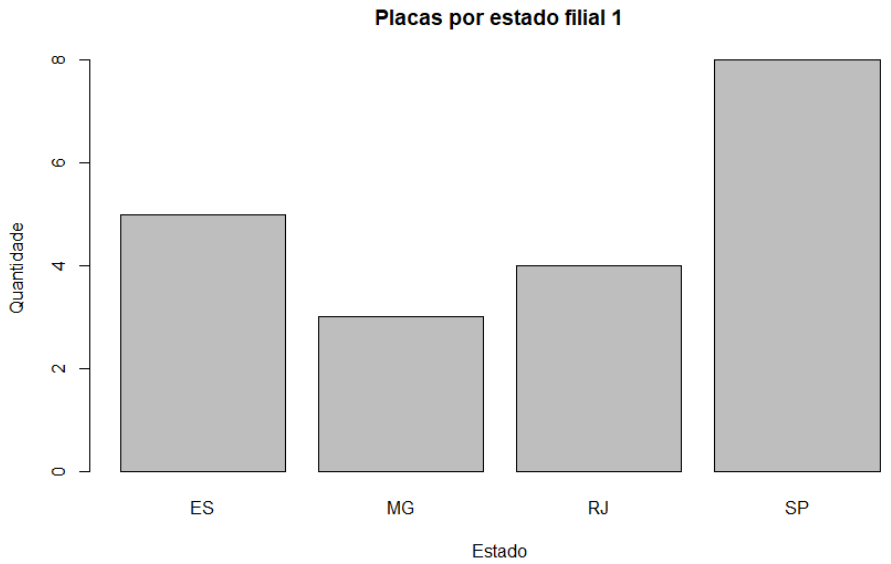
- Uma empresa produz de placas aço em suas 4 filiais cujo valor de referência é 1.200kg. Após verificar que lotes com placas não passaram pelo controle de qualidade em todas as filiais, as equipes das filiais 1 e 3 passaram por um treinamento. Com o objetivo de verificar o quão eficaz foram os treinamentos, foram selecionadas 10 placas produzidas em cada uma das 4 filiais. Os dados das filiais estão armazenados em arquivos **CSV** com o nome **placafilialX.csv**.
  - Elabore um gráfico de barras para o quantidade de placas da amostra da filial 1 por estado. O gráfico deve ter: título “Placas por estado filial 1”; eixo x com rótulo “Estado”; eixo y com rótulo “Quantidade”.
  - Elabore um gráfico de pizza para a quantidade de placas da amostra da filial 1 por estado. O gráfico deve ter o título “Placas por estado filial 1”.
  - Elabore um histograma com cinco barras para o peso das placas da amostra da filial 1. O gráfico deve ter: o título “Histograma de placas filial 1”.

Exemplo 1 - Resolução

- Elabore um gráfico de barras para o quantidade de placas da amostra da filial 1 por estado. O gráfico deve ter: título “Placas por estado filial 1”; eixo x com rótulo “Estado”; eixo y com rótulo “Quantidade”.

```
filial1 <- read.table("C:\\temp\\placafilial1.csv", sep = ";", dec = ",")
names(filial1) <- c("peso", "estado")
tabela <- table(filial1$estado)
barplot(tabela, main = "Placas por estado filial 1", xlab = "Estado", ylab =
"Quantidade")
```

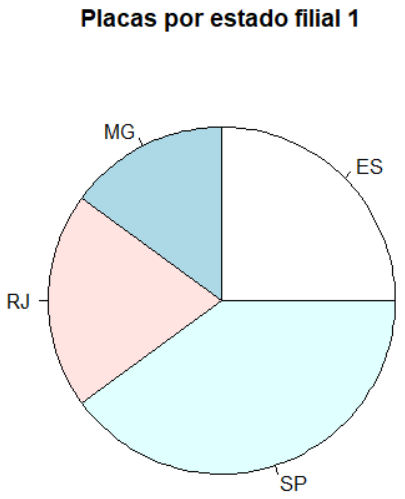
- A leitura do arquivo foi realizada com o comando **read.table**, interno do R, mas poderia ter sido utilizado o comando **read.csv** ou **read.csv2**.
- **names** foi utilizado para definir o nome das colunas do data frame uma vez que o arquivo placafilial1.csv não possui.
- O comando **table** permite criar uma representação categórica dos dados com o nome da variável e a frequência na forma de uma tabela.



Exemplo 1 - Resolução

- Elabore um gráfico de pizza para a quantidade de placas da amostra da filial 1 por estado. O gráfico deve ter o título “Placas por estado filial 1”.

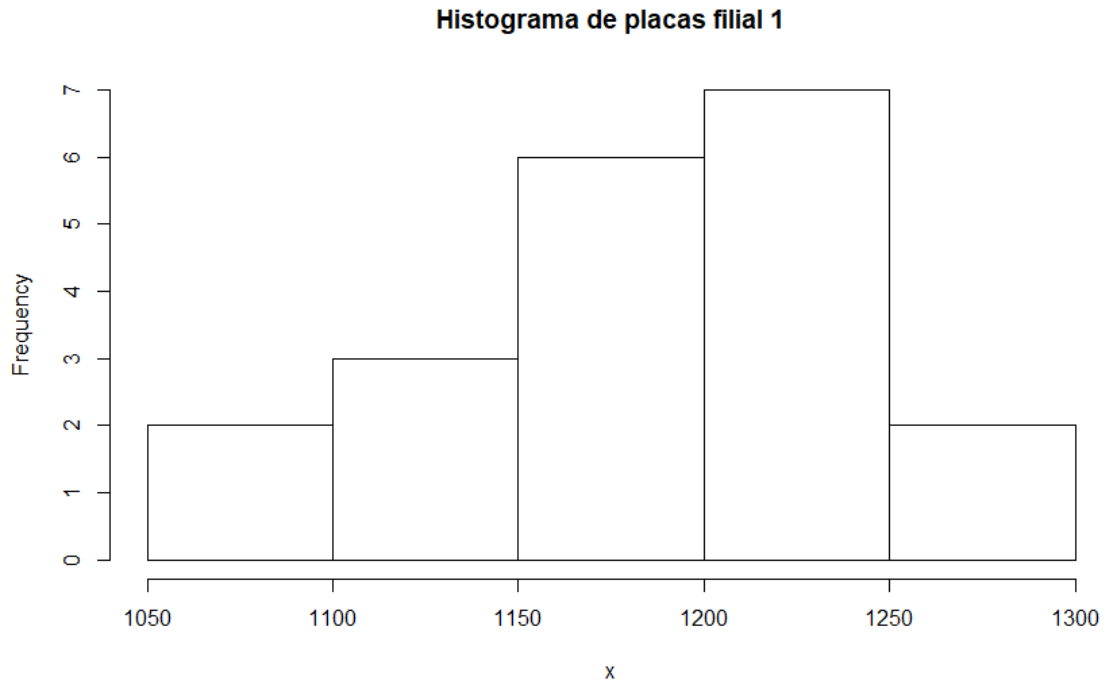
```
pie (tabela, main = "Placas por estado filial 1")
```



Exemplo 1 - Resolução

- Elabore um histograma com cinco barras para o peso das placas da amostra da filial 1. O gráfico deve ter: o título “Histograma de placas filial 1”.

```
x <- filial1$peso
hist(x, main = "Histograma de placas filial 1", breaks = 5)
```



Exemplo 2

- Sabendo que as filiais 2 e 4 não passaram pelo treinamento, analise se são falsas ou verdadeiras as afirmativas abaixo:
  - As equipes das filiais 1 e 3 produzem placas com menor variabilidade.
  - A equipe 4 é a que produz peças com maior variabilidade.
  - Na amostra da equipe 2, foi detectado um outlier.
  - Na amostra da equipe 3, foi detectada a menor amplitude.
  - Como as placas das equipes 1 e 3 apresentam menor variabilidade e com valor médio próximo do valor de referência, pode-se concluir que o treinamento teve efeito positivo e que as demais filiais também deveriam realizar o treinamento.

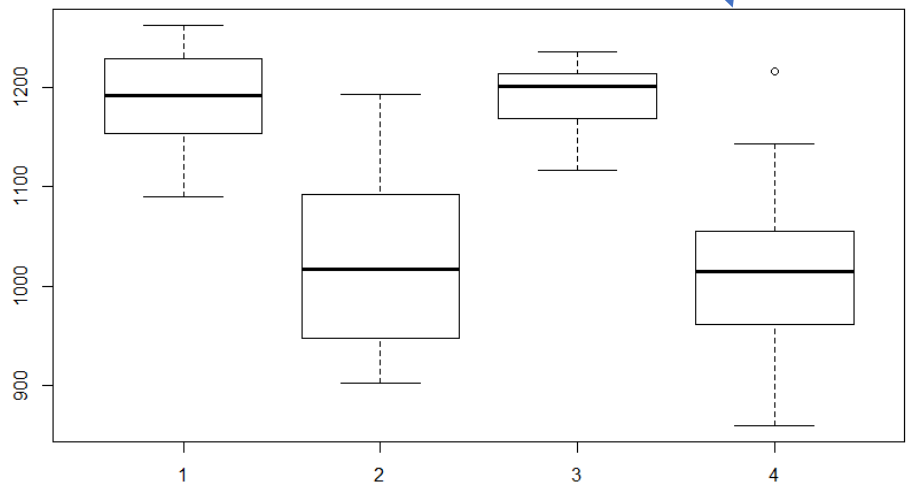
Exemplo 2

```
filial1 <- read.table("C:\\temp\\placafilial1.csv", sep = ";", dec = ",")
names(filial1) <- c("peso", "estado")

filial2 <- read.table("C:\\temp\\placafilial2.csv", sep = ";", dec = ",")
names(filial2) <- c("peso", "estado")

filial3 <- read.table("C:\\temp\\placafilial3.csv", sep = ";", dec = ",")
names(filial3) <- c("peso", "estado")

filial4 <- read.table("C:\\temp\\placafilial4.csv", sep = ";", dec = ",")
names(filial4) <- c("peso", "estado")
```

Exemplo 2

```
> filial1$filial <- rep("1", 20)
> filial2$filial <- rep("2", 20)
> filial3$filial <- rep("3", 20)
> filial4$filial <- rep("4", 20)
>
> empresa.merge.1e2 <- merge(filial1, filial2, all = TRUE)
> empresa.merge.3e4 <- merge(filial3, filial4, all = TRUE)
> empresa <- merge(empresa.merge.1e2, empresa.merge.3e4, all = TRUE)
> summary(empresa)
      peso          estado      filial
 Min.   : 860.2   ES:21   Length:80
 1st Qu.:1017.0   MG:13   Class :character
 Median :1120.6   RJ:20   Mode  :character
 Mean   :1104.9   SP:26
 3rd Qu.:1201.1
 Max.   :1262.3
>
> boxplot(empresa$peso ~ empresa$filial)
```

Variabilidade = Q3 – Q1
Amplitude = Valor máximo – Valor mínimo



Exemplo 2

- Sabendo que as filiais 2 e 4 não passaram pelo treinamento, analise se são falsas ou verdadeiras as afirmativas abaixo:
  - As equipes das filiais 1 e 3 produzem placas com menor variabilidade.
  - A equipe 4 é a que produz peças com maior variabilidade.
  - Na amostra da equipe 2, foi detectado um outlier.
  - Na amostra da equipe 3, foi detectada a menor amplitude.
  - Como as placas das equipes 1 e 3 apresentam menor variabilidade e com valor médio próximo do valor de referência, pode-se concluir que o treinamento teve efeito positivo e que as demais filiais também deveriam realizar o treinamento.

Gráficos com ggplot2

- É um pacote do R voltado para a criação de gráficos estatísticos baseado na **Gramática dos Gráficos** criada por Leland Wilkinson.
- Essa gramática diz que um **gráfico estatístico** é um mapeamento dos dados a partir de **atributos estéticos** (cores, formas, tamanho) de **formas geométricas** (pontos, linhas, barras).
- Seguindo essa gramática, os gráficos no **ggplot2** são criados em camadas, construídas uma a uma.
- Para a criação dos gráficos serão utilizadas duas bases:
  - mtcars – contém dados como consumo, cilindradas, número de carburadores, etc. sobre 32 automóveis.
  - ais - contém dados como sangue e peso de atletas australianos. Para utilizar esta base, deve ser instalado o pacote DAAG.

Bancos ais e mtcars

- Antes de seguir, vamos visualizar as primeiras linhas dos bancos **ais** e **mtcars**, utilizando os comandos:

```
> library(DAAG)
Carregando pacotes exigidos: lattice
> head(ais)
   rcc wcc   hc   hg ferr   bmi   ssf pcBfat   lbm    ht   wt sex  sport
1 3.96 7.5 37.5 12.3   60 20.56 109.1  19.75 63.32 195.9 78.9   f B_Ball
2 4.41 8.3 38.2 12.7   68 20.67 102.8  21.30 58.55 189.7 74.4   f B_Ball
3 4.14 5.0 36.4 11.6   21 21.86 104.6  19.88 55.36 177.8 69.1   f B_Ball
4 4.11 5.3 37.3 12.6   69 21.88 126.4  23.66 57.18 185.0 74.9   f B_Ball
5 4.45 6.8 41.5 14.0   29 18.96 80.3   17.64 53.20 184.6 64.6   f B_Ball
6 4.10 4.4 37.4 12.5   42 21.04 75.2   15.58 53.77 174.0 63.7   f B_Ball
> head(mtcars)
                   mpg cyl disp  hp drat    wt  qsec vs am gear carb
Mazda RX4         21.0   6  160 110 3.90 2.620 16.46  0  1    4    4
Mazda RX4 Wag     21.0   6  160 110 3.90 2.875 17.02  0  1    4    4
Datsun 710        22.8   4  108  93 3.85 2.320 18.61  1  1    4    1
Hornet 4 Drive    21.4   6  258 110 3.08 3.215 19.44  1  0    3    1
Hornet Sportabout 18.7   8  360 175 3.15 3.440 17.02  0  0    3    2
Valiant           18.1   6  225 105 2.76 3.460 20.22  1  0    3    1
```

Primeira camada

- A primeira camada é criada pela função **ggplot()** do pacote **ggplot2**, que deve receber como parâmetro um data frame.
- A primeira camada é um painel em branco.

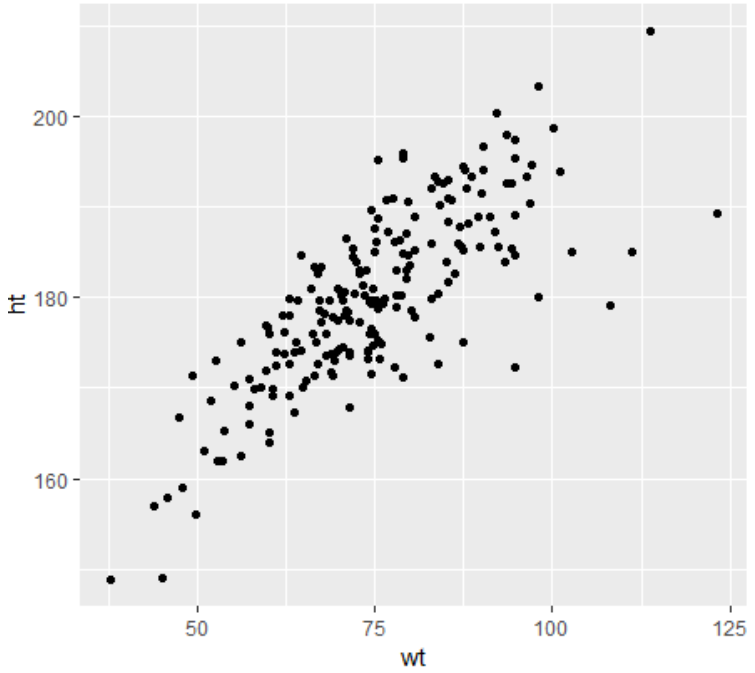
```
library(ggplot2)
ggplot(data = mtcars)
```

Camada por camada

- Cada camada representa um tipo de mapeamento ou personalização do gráfico.
- A partir da primeira camada, podemos, então, construir as demais.

```
library(DAAG)
library(ggplot2)
ggplot(data = ais) + geom_point(mapping = aes(x = wt, y = ht))
```

Gráfico de dispersão envolvendo os dados peso (eixo x) e altura (eixo y).



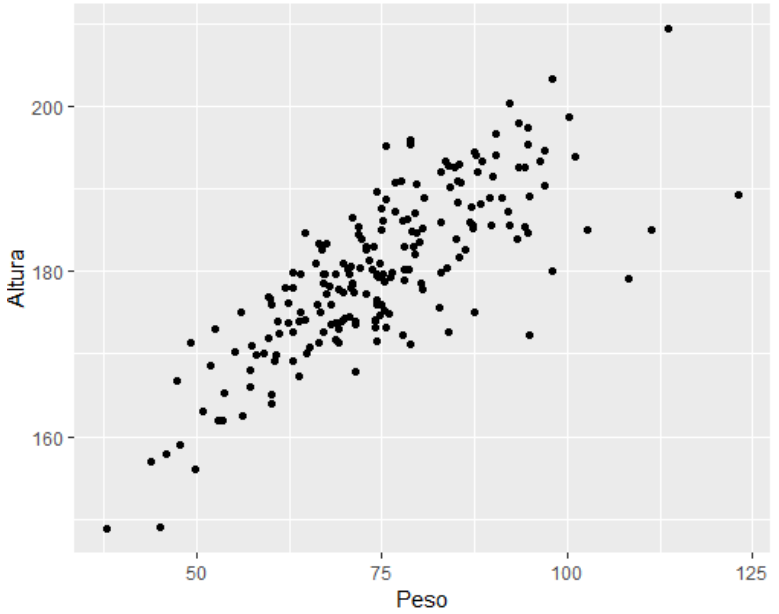
Segunda camada

- A primeira camada foi criada pela função **ggplot()**, que recebeu um data frame.
- A segunda camada foi criada pela função **geom_point()** que especificou a **forma geométrica** (gráfico dispersão transformando pares (x, y) em pontos, no caso) utilizada no mapeamento das observações.
- A função **aes()** foi usada para indicar qual variável seria representada no eixo **x** e qual seria representada no eixo **y**.
- As duas camadas foram unidas pelo operador “+”.
- A função **aes()** pode ser utilizada, por exemplo, para:
  - definir a estética (***aesthetics***) das variáveis.
  - indicar a relação entre os dados, no exemplo, par (x, y).
  - definir a cor e o tamanho dos componentes geométricos.

Terceira camada

- A primeira camada foi criada pela função **ggplot()**, que recebeu um data frame.
- A segunda camada foi criada pela função **geom_point()** que especificou a **forma geométrica** (gráfico dispersão, no caso) utilizada no mapeamento das observações.
- As funções **labs()**, **xlab()** e **ylab()** podem ser utilizadas para acrescentar rótulos aos eixos **x** e **y**.

```
library(DAAG)
library(ggplot2)
ggplot(data = ais) +
  geom_point(mapping = aes(x = wt, y = ht)) +
  labs(x = "Peso", y = "Altura")
```

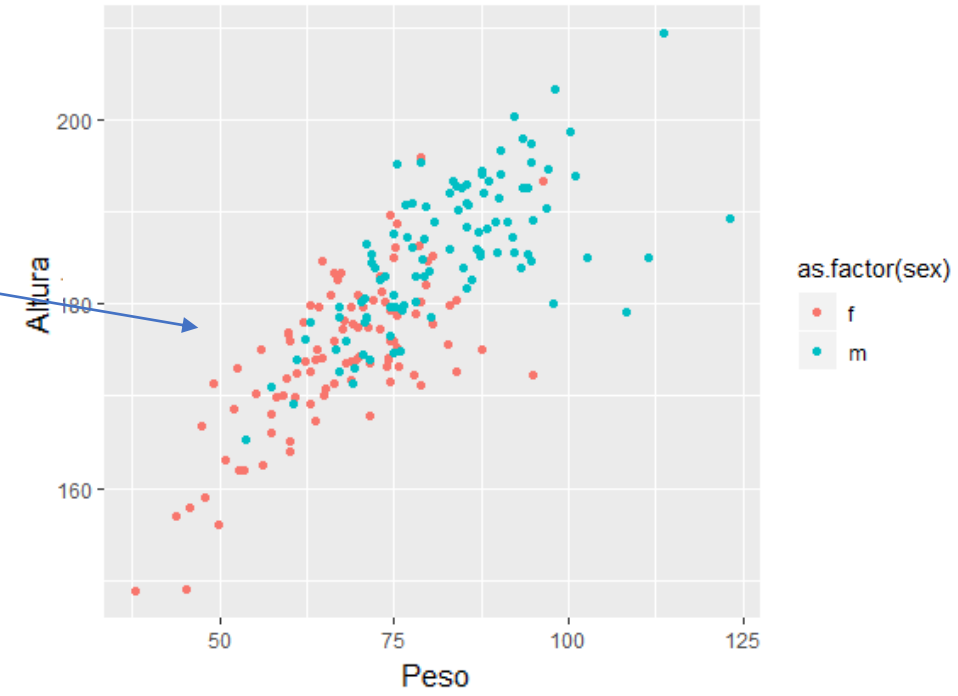


Estética

- Outro aspecto que pode ser mapeado nesse gráfico é a cor dos pontos.

```
library(DAAG)
library(ggplot2)
ggplot(data = ais) +
  geom_point(mapping = aes(x = wt, y = ht, color = as.factor(sex))) +
  labs(x = "Peso", y = "Altura")
```

A variável categórica sex (f ou m) do data frame foi utilizada como fator para definir as cores dos pontos.

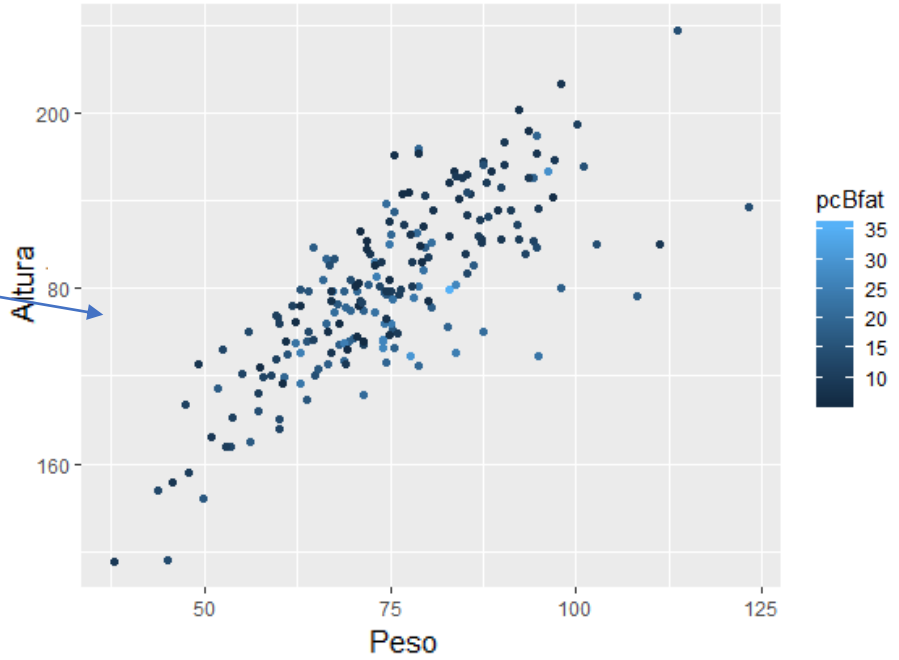


Estética

- Outro aspecto que pode ser mapeado nesse gráfico é a cor dos pontos.

```
library(DAAG)
library(ggplot2)
ggplot(data = ais) +
  geom_point(mapping = aes(x = wt, y = ht, color = pcBfat)) +
  labs(x = "Peso", y = "Altura")
```

A variável contínua pcBfat do data frame foi utilizada para definir as cores dos pontos.



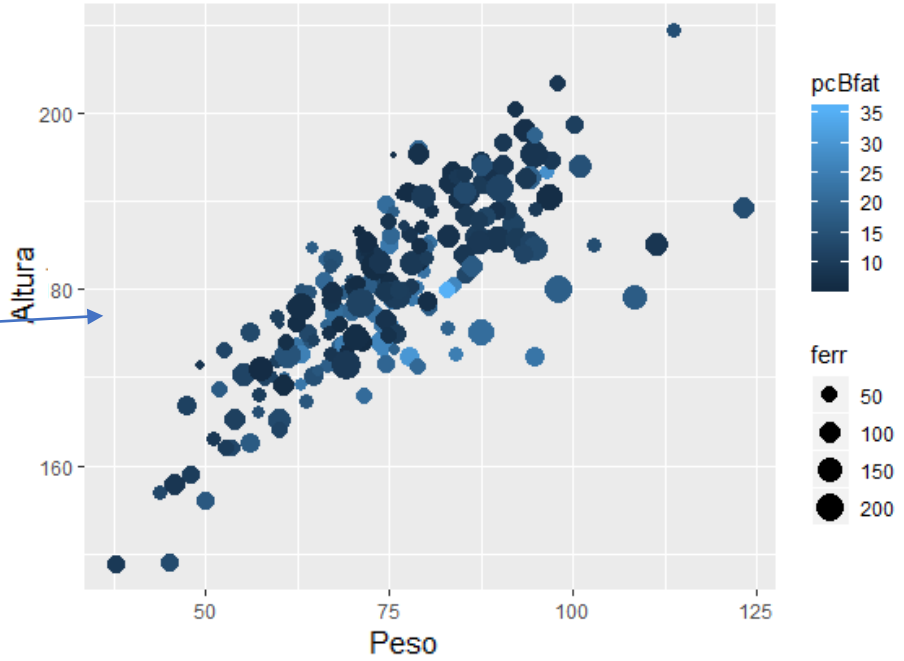
Estética

- Outro aspecto que pode ser mapeado nesse gráfico é a cor dos pontos.

```
library(DAAG)
library(ggplot2)
ggplot(data = ais) +
  geom_point(mapping = aes(x = wt, y = ht, color = pcBfat, size = ferr)) +
  labs(x = "Peso", y = "Altura")
```

A variável contínua pcBfat do data frame foi utilizada para definir as cores dos pontos.

A variável contínua ferr do data frame foi utilizada para mapear o tamanho dos pontos

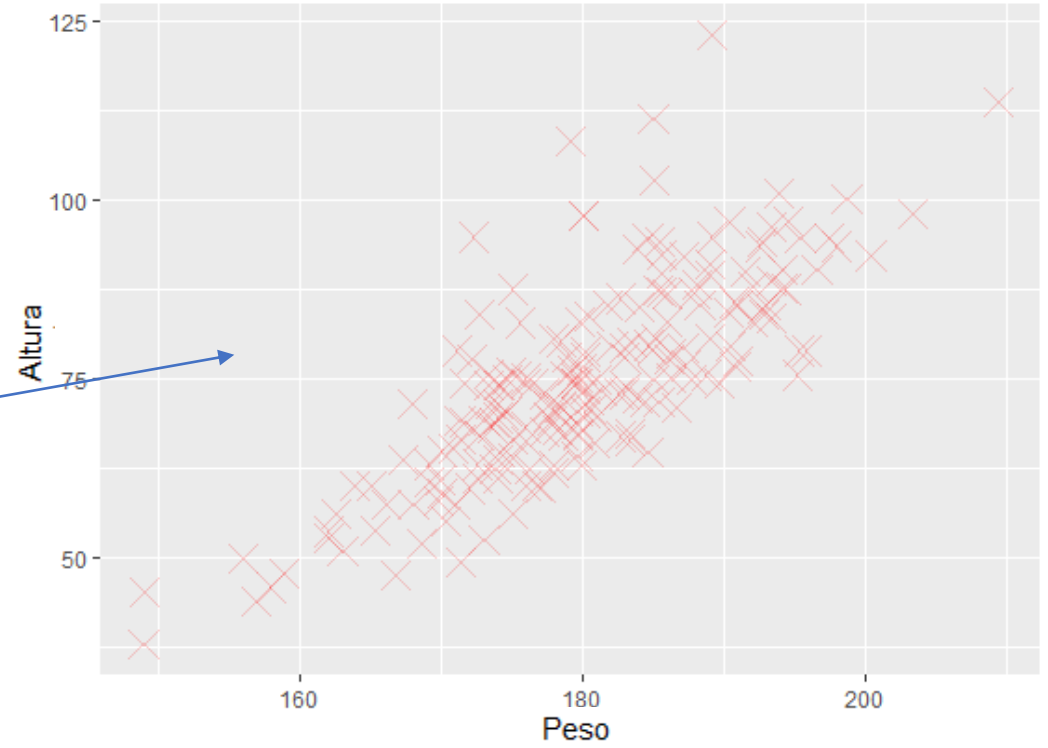


Estética

- Outros aspectos também podem ser definidos.

```
library(DAAG)
library(ggplot2)
ggplot(ais, aes(y = wt, x = ht)) +
  geom_point(color = "red", size = 5, shape = 4, alpha = 0.2) +
  labs(x = "Peso", y = "Altura")
```

Em geom_point color define a cor do ponto, size, o tamanho do ponto, shape, o formato do ponto e alpha, a transparência (entre 1 e 0).



Geoms_XXXX

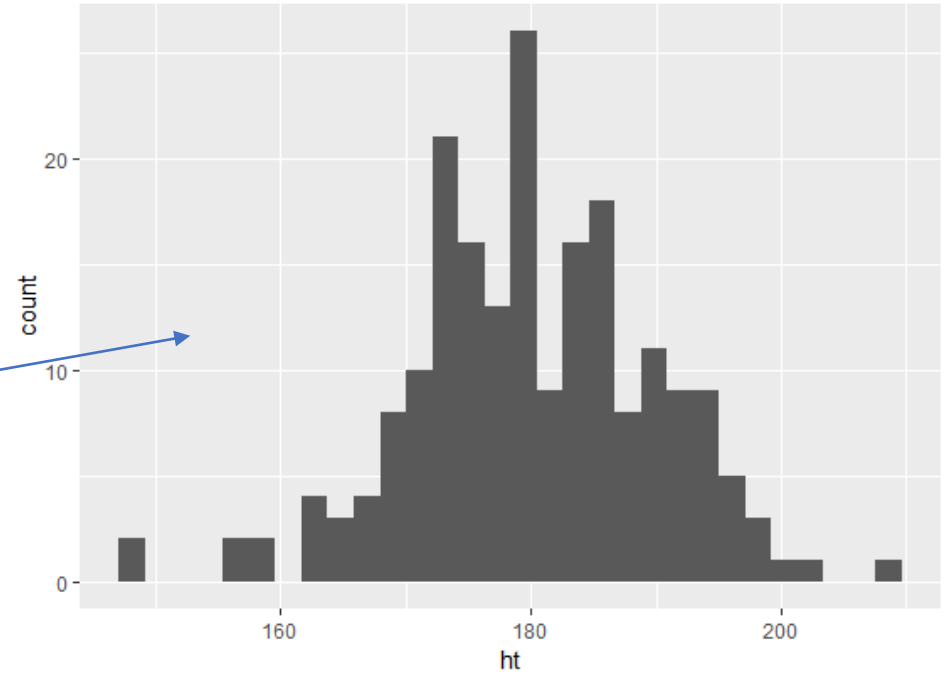
- Exemplos de outros geoms:
  - geom_point – para gráficos de dispersão.
  - geom_line - para linhas definidas por pares (x,y).
  - geom_abline - para retas definidas por um intercepto e uma inclinação.
  - geom_hline - para retas horizontais.
  - geom_bar - para barras.
  - geom_histogram - para histogramas.
  - geom_boxplot - para boxplots.
  - geom_density - para densidades.
  - geom_area - para áreas.
- Para fazer um gráfico usando ggplot2 e a gramática de gráficos, seguimos o padrão:
  - ggplot(data = DATA) + GEOM_FUNCTION(mapping = aes(MAPPINGS))

Geoms_histogram

- Para fazer um histograma, precisamos passar uma variável x, sendo que y é a frequência calculada.

```
library(DAAG)
library(ggplot2)
ggplot(ais) + geom_histogram(aes(x = ht), bins = 30)
```

A variável ht é usada como base para gerar o histograma.

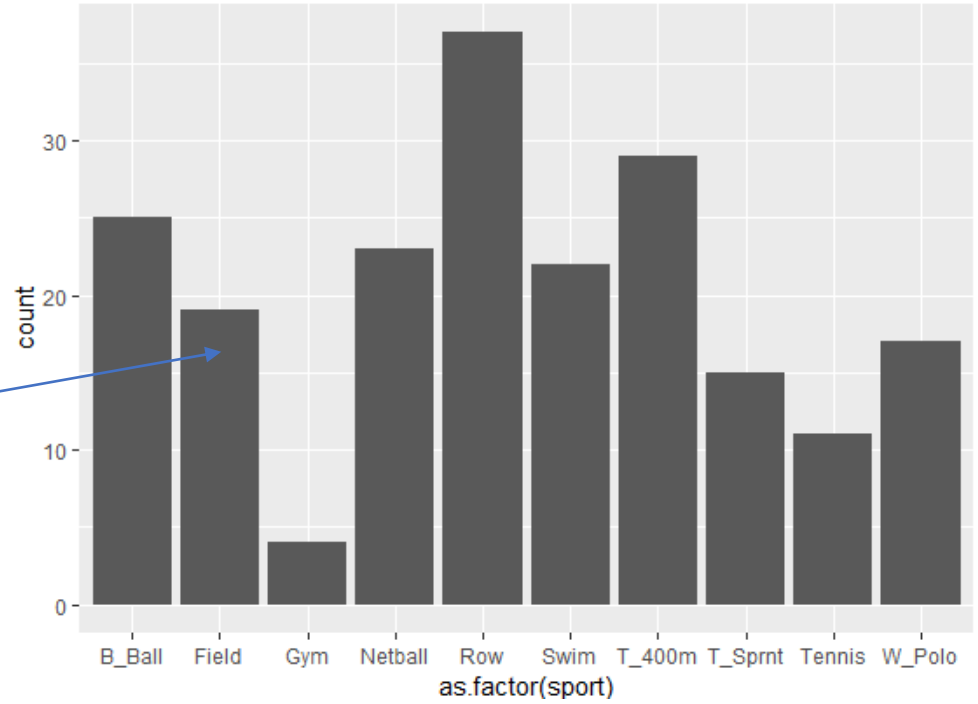


Geoms_bar

- Para fazer um histograma, precisamos passar uma variável x, sendo que y é a frequência calculada.

```
library(DAAG)
library(ggplot2)
ggplot(ais) + geom_bar(aes(x = as.factor(sport)))
```

A variável sport é usada como base para gerar o gráfico de barras.

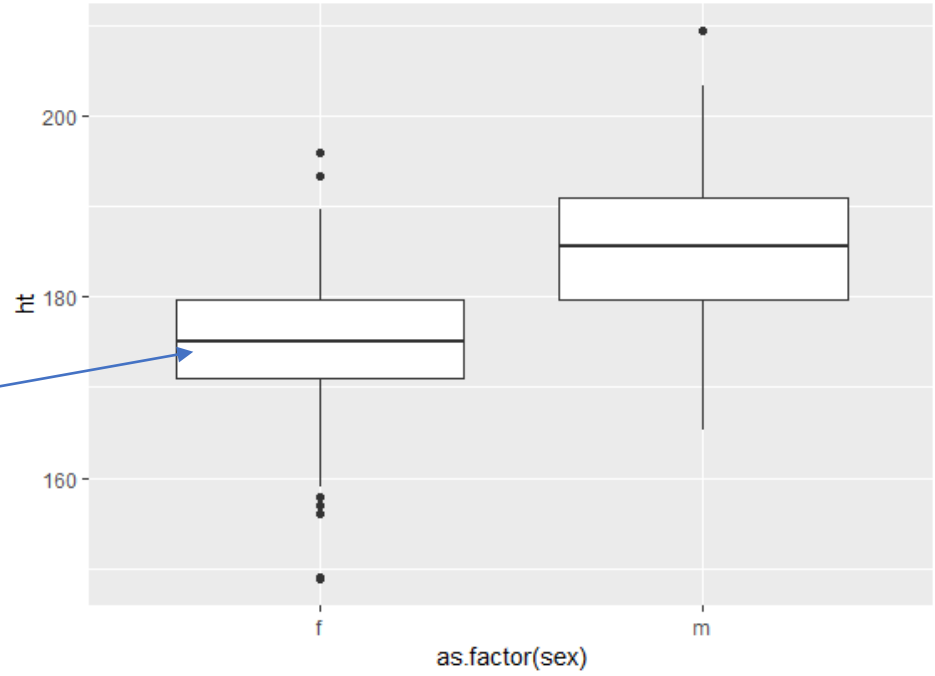


Geoms_boxplot

- Para fazer um boxplot para cada grupo, precisamos passar um fator para o aspecto x do gráfico.

```
library(DAAG)
library(ggplot2)
ggplot(ais) + geom_boxplot(aes(x = as.factor(sex), y = ht))
```

A variável sex é usada como fator para construção do boxplot considerando a variável altura.



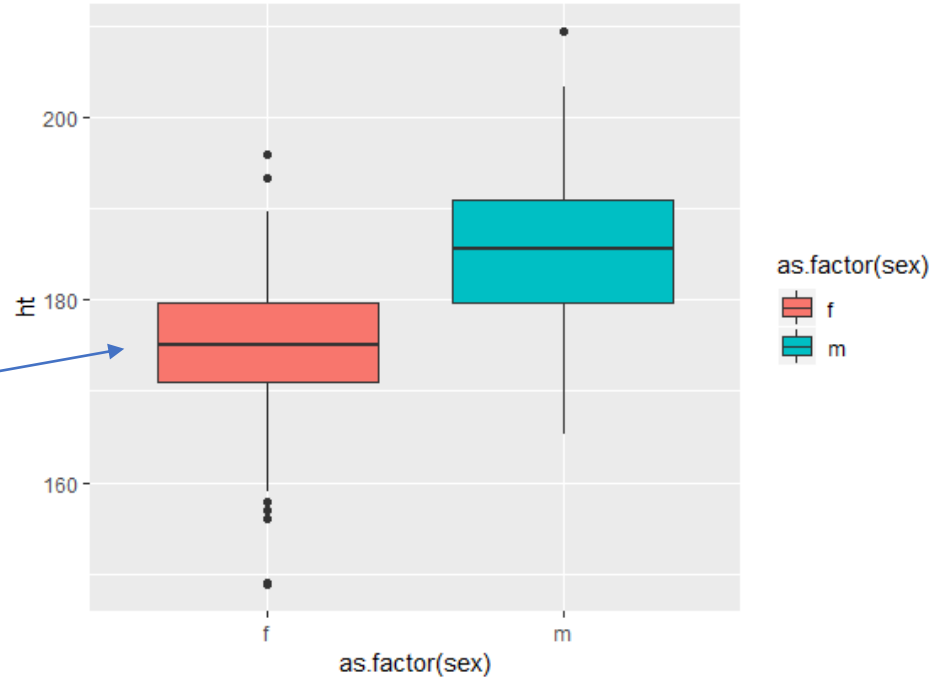
Geoms_boxplot

- Para fazer um boxplot para cada grupo, precisamos passar um fator para o aspecto x do gráfico.

```
library(DAAG)
library(ggplot2)
ggplot(ais) + geom_boxplot(aes(x = as.factor(sex), y = ht, fill = as.factor(sex)))
```

Legenda.

A variável sex é usada como fator para construção do boxplot considerando a variável altura.



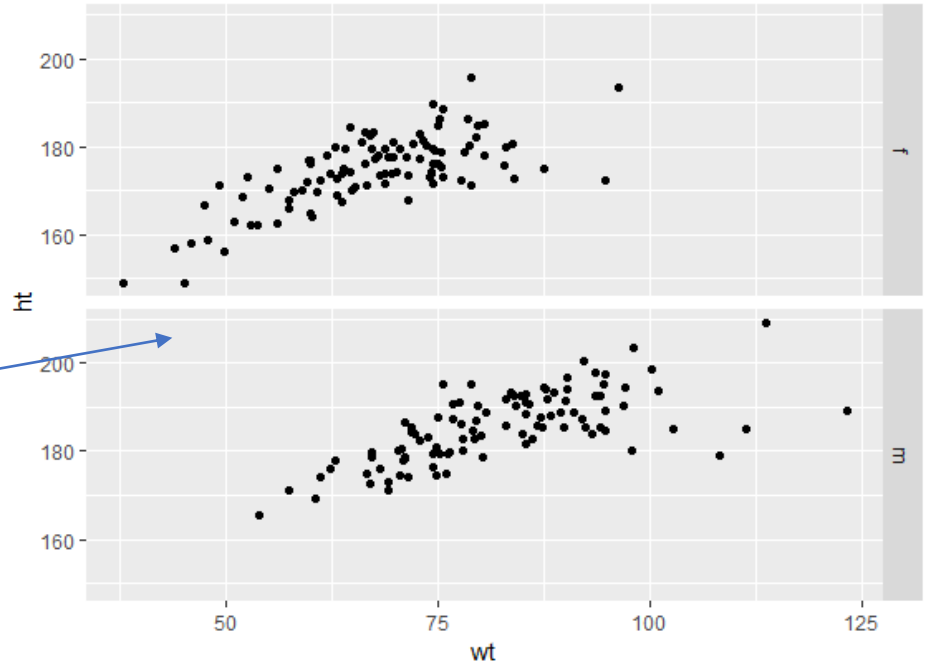
Facets

- Auxilia na visualização de diferentes subconjuntos dos dados em gráficos separados.

```
library(DAAG)
library(ggplot2)
ggplot(data = ais) + geom_point(mapping = aes(x = wt, y = ht)) + facet_grid(sex~.)
```

Conjuntos na horizontal.

A variável sex é usada na função facet_grid para dividir o conjunto de dados em masculino e feminino.



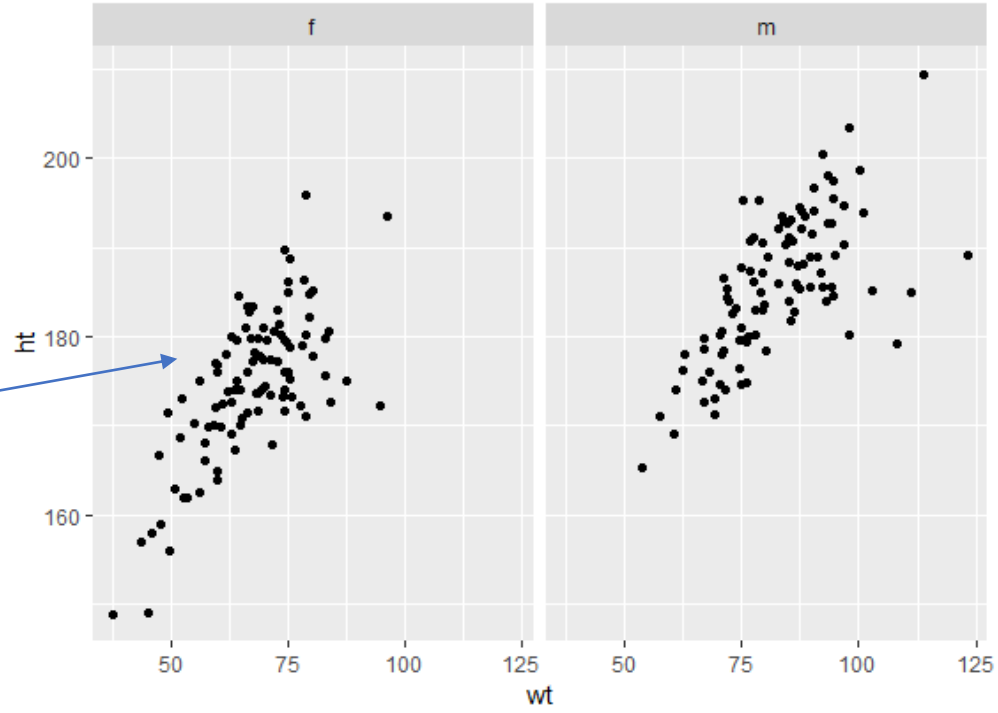
Facets

- Auxilia na visualização de diferentes subconjuntos dos dados em gráficos separados.

```
library(DAAG)
library(ggplot2)
ggplot(data = ais) + geom_point(mapping = aes(x = wt, y = ht)) + facet_grid(.~sex)
```

Conjuntos na vertical.

A variável sex é usada na função facet_grid para dividir o conjunto de dados em masculino e feminino.



Exercício 1

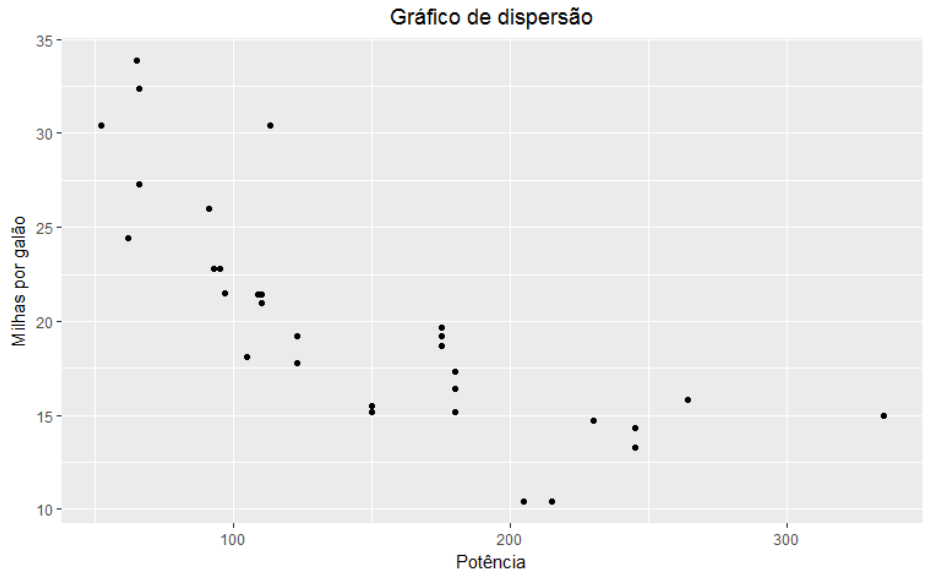
- Usando o banco **mtcars** e o pacote **ggplot2,** elabore um gráfico de dispersão, um gráfico barras, um histograma e um boxplot. Todos os gráficos deverão ter título e rótulos nos eixos x e y quando aplicável. Para o gráfico de dispersão criado, elabore um novo utilizando **facets**.

Exercício 1 - Resolução

- Usando o banco **mtcars** e o pacote **ggplot2,** elabore um gráfico de dispersão, um gráfico barras, um histograma e um boxplot. Todos os gráficos deverão ter título e rótulos nos eixos x e y quando aplicável. Para o gráfico de dispersão criado, elabore um novo utilizando **facets**.

```
library(ggplot2)
ggplot(data = mtcars) +
 geom_point(mapping = aes(x = hp, y = mpg)) +
 labs(x = "Potência", y = "Milhas por galão") +
 ggtitle("Gráfico de dispersão") +
 theme(plot.title = element_text(hjust = 0.50))
```

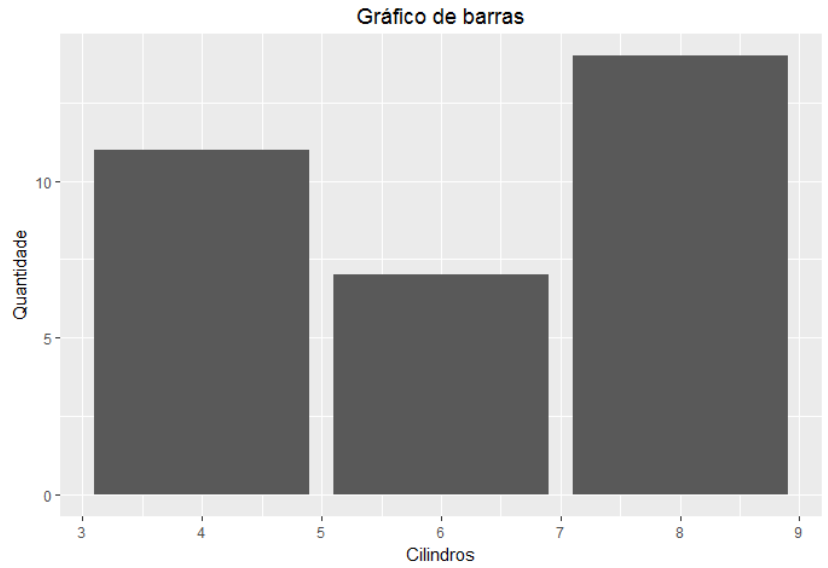
Por padrão o título é alinhado à esquerda. Use theme conforme descrito acima para centralizar o título.



Exercício 1 - Resolução

- Usando o banco **mtcars** e o pacote **ggplot2,** elabore um gráfico de dispersão, um gráfico barras, um histograma e um boxplot. Todos os gráficos deverão ter título e rótulos nos eixos x e y quando aplicável. Para o gráfico de dispersão criado, elabore um novo utilizando **facets**.

```
library(ggplot2)
ggplot(data = mtcars) +
 geom_bar(mapping = aes(x = cyl)) +
 labs(x = "Cilindros", y = "Quantidade") +
 ggtitle("Gráfico de barras") +
 theme(plot.title = element_text(hjust = 0.50))
```

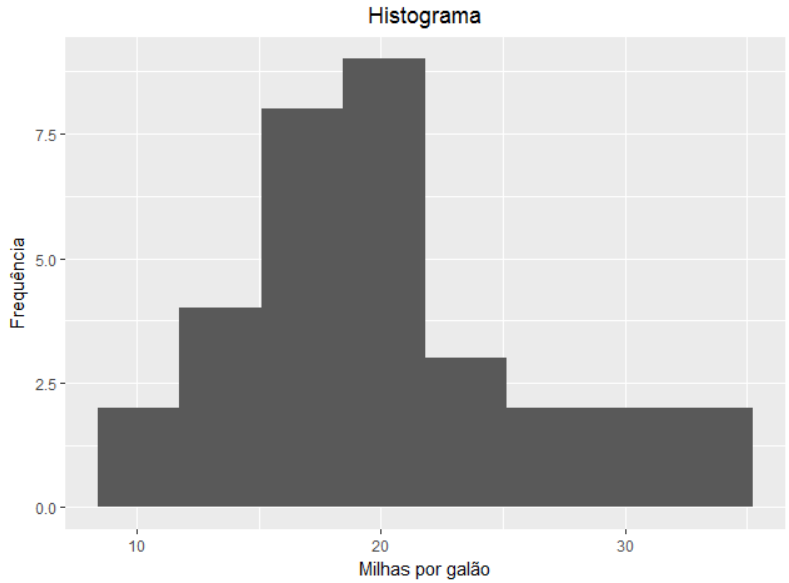


Exercício 1 - Resolução

- Usando o banco **mtcars** e o pacote **ggplot2,** elabore um gráfico de dispersão, um gráfico barras, um histograma e um boxplot. Todos os gráficos deverão ter título e rótulos nos eixos x e y quando aplicável. Para o gráfico de dispersão criado, elabore um novo utilizando **facets**.

```
library(ggplot2)
ggplot(data = mtcars) +
 geom_histogram(mapping = aes(mtcars$mpg), bins = 8) +
 labs(x = "Milhas por galão", y = "Frequência") +
 ggtitle("Histograma") +
 theme(plot.title = element_text(hjust = 0.50))
```

O parâmetro bins define o número de divisões do histograma.

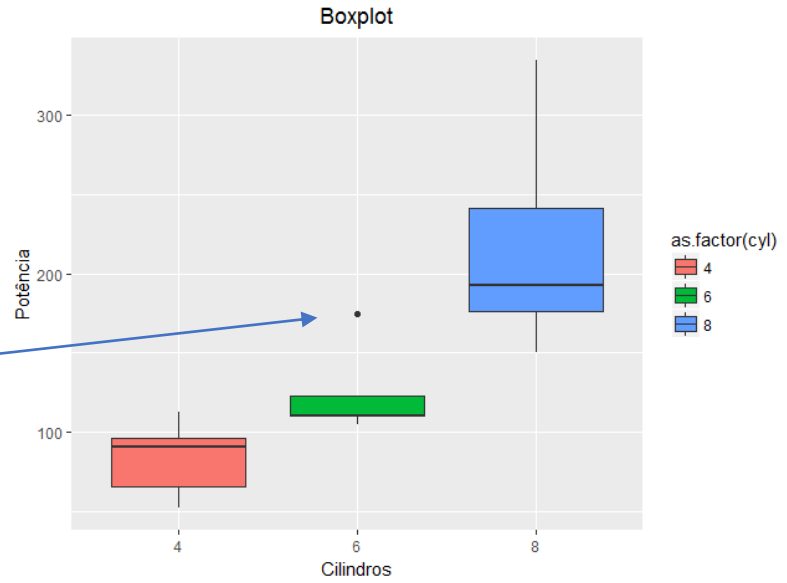


Exercício 1 - Resolução

- Usando o banco **mtcars** e o pacote **ggplot2,** elabore um gráfico de dispersão, um gráfico barras, um histograma e um boxplot. Todos os gráficos deverão ter título e rótulos nos eixos x e y quando aplicável. Para o gráfico de dispersão criado, elabore um novo utilizando **facets**.

```
library(ggplot2)
ggplot(data = mtcars) +
 geom_boxplot(mapping = aes(x = as.factor(cyl), y = hp, fill = as.factor(cyl))) +
 labs(x = "Cilindros", y = "Potência") +
 ggtitle("Boxplot") +
 theme(plot.title = element_text(hjust = 0.50))
```

Exemplo de um outlier.

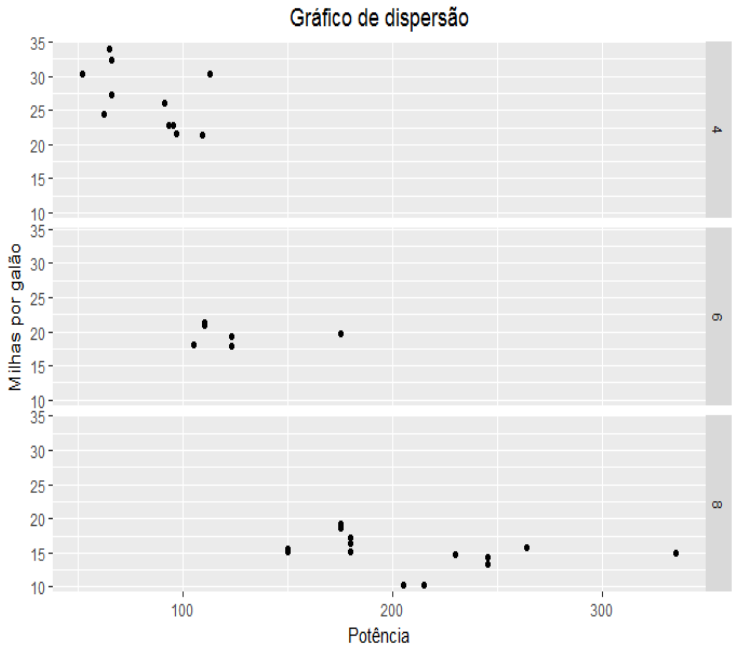


Exercício 1 - Resolução

- Usando o banco **mtcars** e o pacote **ggplot2,** elabore um gráfico de dispersão, um gráfico barras, um histograma e um boxplot. Todos os gráficos deverão ter título e rótulos nos eixos x e y quando aplicável. Para o gráfico de dispersão criado, elabore um novo utilizando **facets**.

```
library(ggplot2)
ggplot(data = mtcars) +
 geom_point(mapping = aes(x = hp, y = mpg)) +
 facet_grid (cyl~.) +
 labs(x = "Potência", y = "Milhas por galão") +
 ggtitle("Gráfico de dispersão") +
 theme(plot.title = element_text(hjust = 0.50))
```

Uso do recurso facets.



Gerando mapas

- Existem diferentes bibliotecas para a elaboração de mapas no R. No exemplo que será apresentado, foram utilizadas as funções **readOGR** (pacote **rgdal**) e **spplot** nativa do R.
- Um dos recursos utilizados no R para a plotagem de mapas são os *shapefiles,* que é um formato popular de arquivo contendo dados geoespaciais em forma de vetor usado por Sistemas de Informações Geográficas (SIG).
- Os *shapefiles* normalmente possuem não só as informações para o desenho do mapa propriamente dito, como também informações como código e nome de estados e municípios, entre outras.
- Muitos *shapefile* podem ser encontrados nos próprios pacotes de mapas e também em fontes como o IBGE (https://www.ibge.gov.br/geociencias/downloads-geociencias.html).
- Será necessário realizar o "download" do arquivo rj_municipios.zip seguindo o caminho: organização_do_territorio\malhas_territoriais\Malhas_municipais\Município_2018\Ufs\RJ.

Gerando mapas

- Inicialmente precisamos importar para o R o *shapefile* do mapa a ser trabalhado. Será utilizado o *shapefile* 33MUE250GC_SIR dos municípios do RJ que estão no arquivo rj_municípios.zip obtido.
- A carga do *shapefile* pode ser feita através do comando *readOGR*.
- Observe no *workspace* do RStudio (canto superior direito) a estrutura do objeto Rio, onde temos dados não espaciais, data.frame **data**, e dados espaciais, list **polygons**.

```
library(rgdal)
Rio <- readOGR("C:\\Temp\\MapaRJ\\.", "33MUE250GC_SIR", use_iconv=T, encoding="UTF-8")
n <- length(Rio@data$CD_GEOCMU) // ou length(Rio$CD_GEOCMU)
n
```

No exemplo, o shapefile 33MUE250GC_SIR, extraído do arquivo rj_municípios.zip, possui a informação geoespacial (polygons) da divisão do estado do Rio em municípios bem como um data.frame (data) com o código IBGE e o nome dos municípios.

O valor de n é 92, ou seja, são 92 municípios.

Gerando mapas

- A seguir, deve ser escolhido o conjunto de dados (população, número de escolas, índice de analfabetismo, entre outros) que será utilizado como base para colorir as regiões (no caso os municípios) do mapa. Os dados podem ser obtidos de fontes como IBGE, FGV, entre outros ou mesmo gerados por você.
- Iremos utilizar o arquivo DadosRJ.CSV que: possui cabeçalho (codigo;populacao;pibpercapta); os dados estão separados por “;”; o separador de casas decimais é “,”; e está no formato UTF-8.
- O arquivo DadosRJ.CSV foi gerado a partir de dados obtidos no site do IBGE e possui 3 colunas: codigo, contendo o código IBGE do município, ou seja, o mesmo usado no arquivo de mapas; população, contendo a população residente do município em 2010; pibpercapta, contendo o PIB *per capta* do município no ano de 2010.

```
library(descr)
file.head("C:\\Temp\\MapaRJ\\DadosRJ.CSV")
dados <- read.table("C:\\Temp\\MapaRJ\\DadosRJ.CSV", header=T, sep=";", dec=",",
encoding="UTF-8")
```

Gerando mapas

- Já temos os dados geoespaciais e os atributos básicos dos municípios (NM_MUNICIP e CD_GEOCMU) na variável espacial **Rio** e o conjunto de dados escolhido na variável **dados**.
- No passo seguinte, deve ser feita a junção das duas variáveis em uma única variável utilizando, por exemplo, o comando **merge**.

```
mapa.com.dados <- merge(Rio, dados, by.x = "CD_GEOCMU", by.y = "codigo")
str(mapa.com.dados)
```

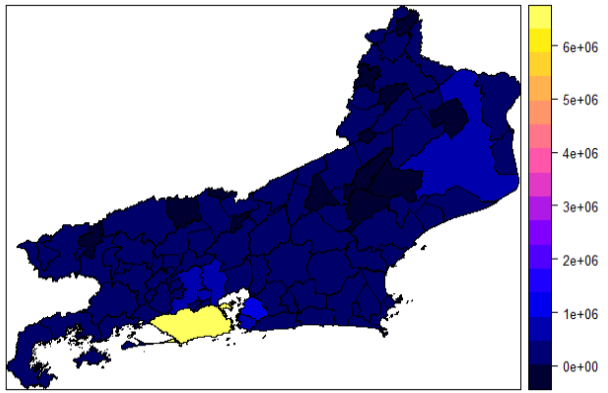
- **by.x** e **by.y** indicam quais colunas devem ser utilizadas como índices para realizar a junção.
- Observe que deve ser utilizada a variável **Rio** e não **Rio@data** no comando **merge**, pois se for utilizado **Rio**, os dados geoespaciais não serão copiados.
- Observe que a coluna codigo do data.frame **dados** (segundo data.frame no comando **merge**) não é copiada para o data.frame **mapa.com.dados** pois seria redundante em termos de informações.

Gerando mapas

- Finalmente podemos realizar a plotagem do mapa através do comando **spplot** utilizando uma das colunas de dados escolhidos como referência.

```
library(rgdal)
Rio <- readOGR("C:\\Temp\\MapaRJ\\.", "33MUE250GC_SIR", stringsAsFactors = F,
use_iconv = T, encoding="UTF-8")

dados <- read.table("C:\\Temp\\MapaRJ\\DadosRJ.CSV", header = T, sep=";", dec=",",
encoding="UTF-8")

mapa.com.dados <- merge(Rio, dados, by.x = "CD_GEOCMU", by.y = "codigo")
spplot(mapa.com.dados, "populacao")
```

Use o comando file.head do pacote descr para explorar o arquivo DadosRJ.CSV.



Gerando mapas

- Finalmente podemos realizar a plotagem do mapa através do comando **spplot** utilizando uma das colunas de dados escolhidos como referência.

```
library(rgdal)
Rio <- readOGR("C:\\Temp\\MapaRJ\\.", "33MUE250GC_SIR", stringsAsFactors = F,
use_iconv = T, encoding="UTF-8")

dados <- read.table("C:\\Temp\\MapaRJ\\DadosRJ.CSV", header = T, sep=";", dec=",",
encoding="UTF-8")

mapa.com.dados <- merge(Rio, dados, by.x = "CD_GEOCMU", by.y = "codigo")
spplot(mapa.com.dados, "pibpercapta")
```

Use o comando file.head do pacote descr para explorar o arquivo DadosRJ.CSV.