Visualização de dados R

Mineração de Dados - Laboratório 1

Thiago Ferreira Covões

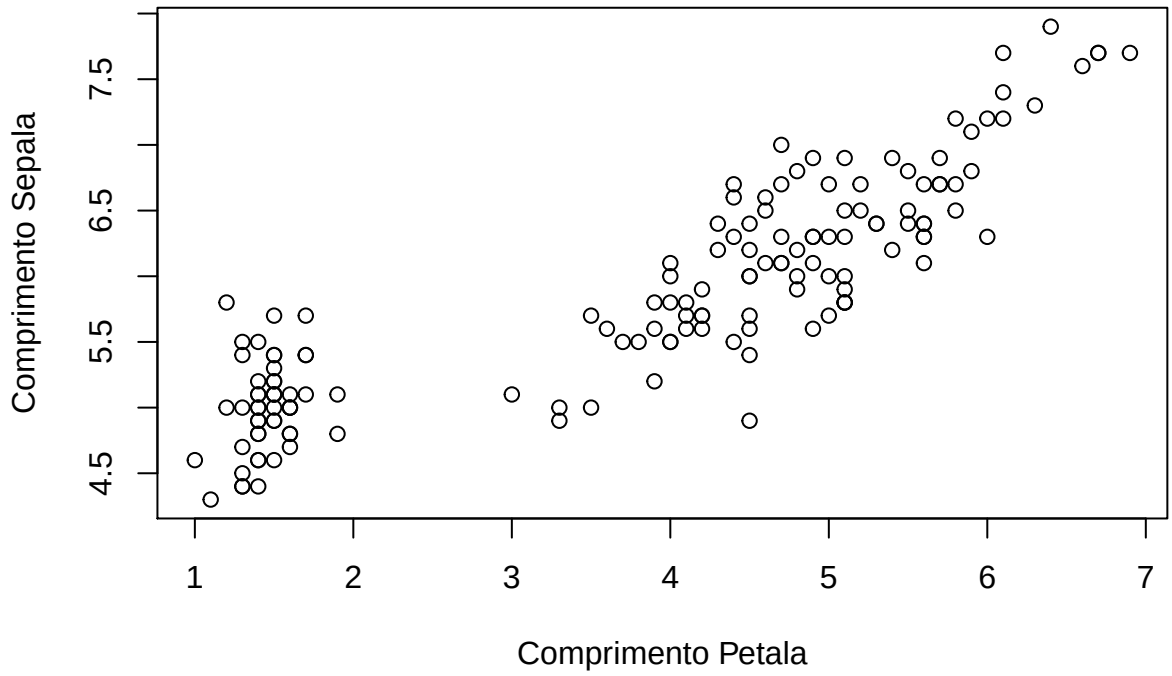
Este material foi baseado no material do Prof. Carlos da Silva dos Santos.

Visualização

Vamos construir alguns gráficos que nos permitam visualizar a geometria dos dados, sua distribuição e tendências.

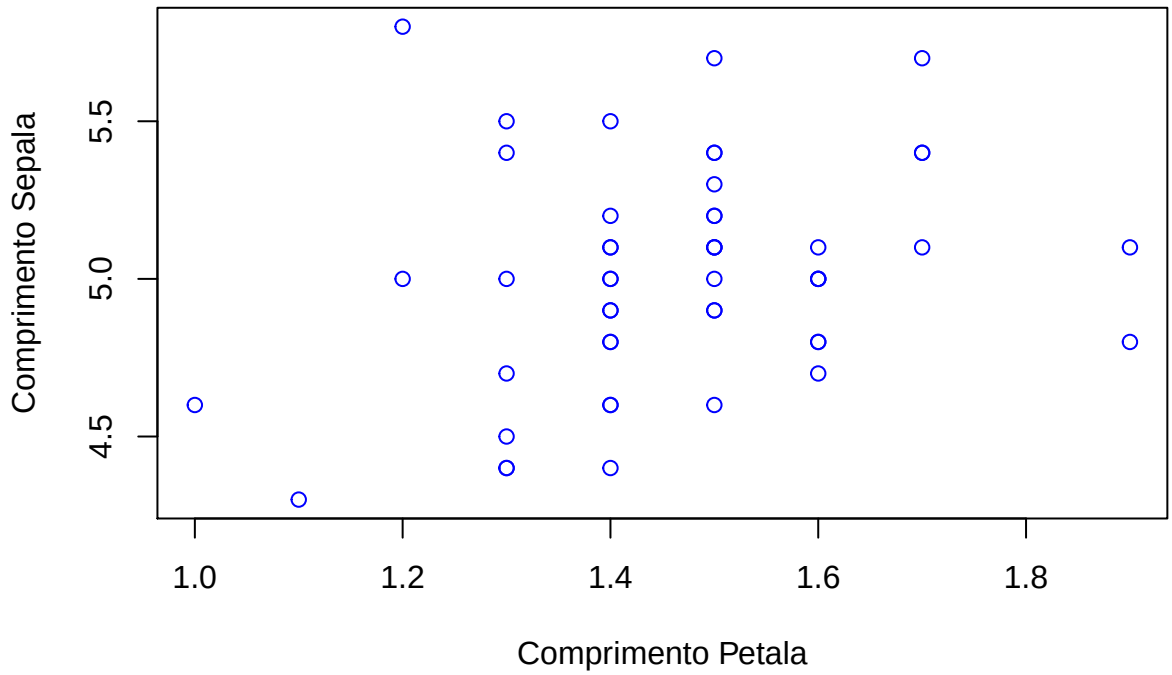
Muitas vezes estamos interessados em entender como dois parâmetros variam conjuntamente. Uma maneira de visualizar esse tipo de comportamento é por meio de um *gráfico de dispersão*, que consiste em uma representação em que pares de atributos (x, y) são desenhados em um plano cartesiano como pontos. O exemplo abaixo mostra como construir um gráfico para examinar a relação entre comprimento de pétala e comprimento de sépala:

```
plot(iris$Petal.Length, iris$Sepal.Length,
     xlab="Comprimento Petala", ylab="Comprimento Sepala", type="p")
```

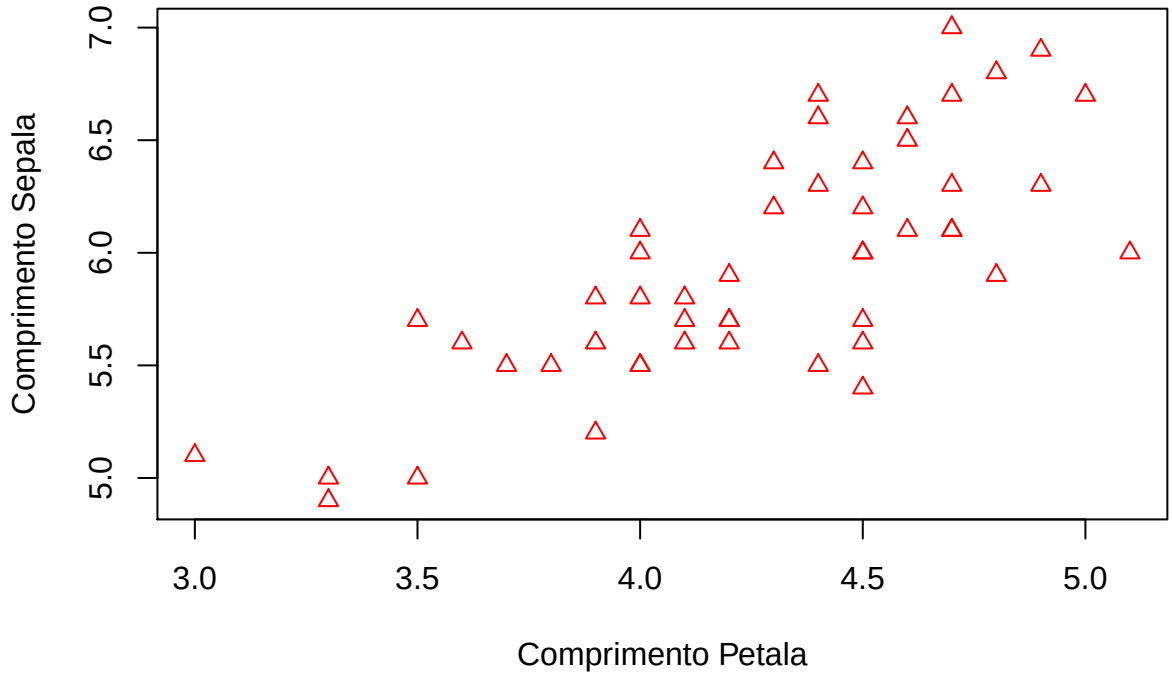


Usando indexação e símbolos diferentes para cada espécie, podemos visualizar o mesmo gráfico de dispersão agrupado por espécie:

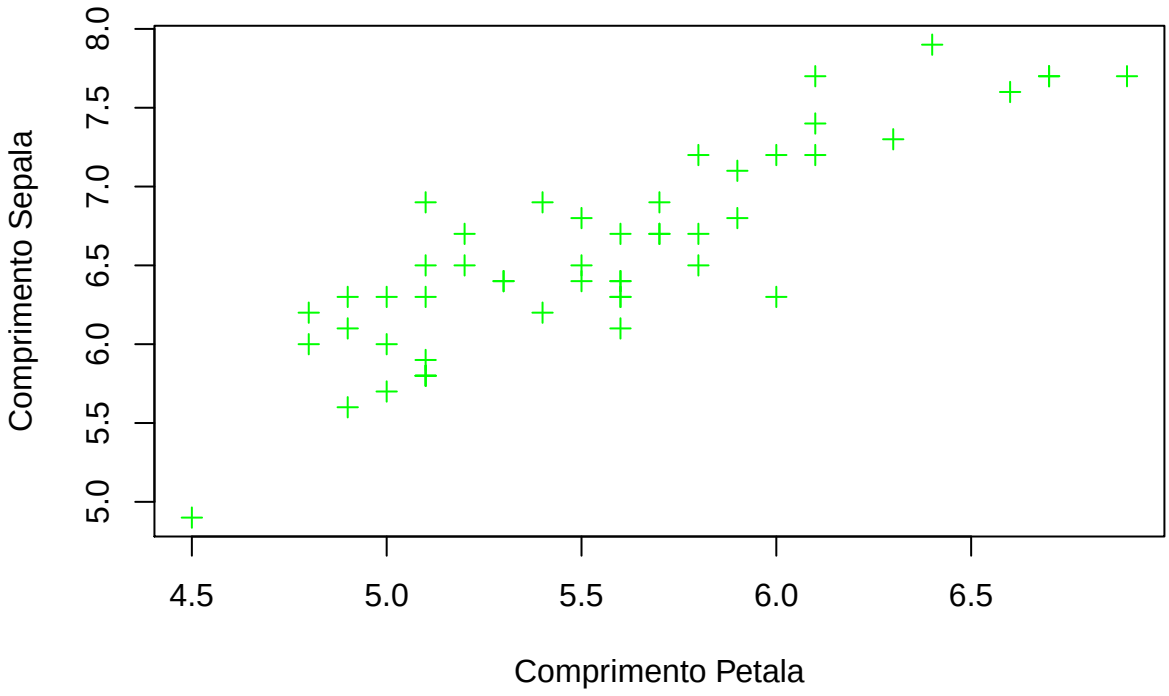
```
plot(iris[iris$Species=="setosa", "Petal.Length"],
     iris[iris$Species=="setosa", "Sepal.Length"],
     xlab="Comprimento Petala", ylab="Comprimento Sepala", type="p", pch=1, col='blue')
```




```
plot(iris[iris$Species=="versicolor", "Petal.Length"],
     iris[iris$Species=="versicolor", "Sepal.Length"],
     xlab="Comprimento Petala", ylab="Comprimento Sepala", type="p", pch=2, col='red')
```

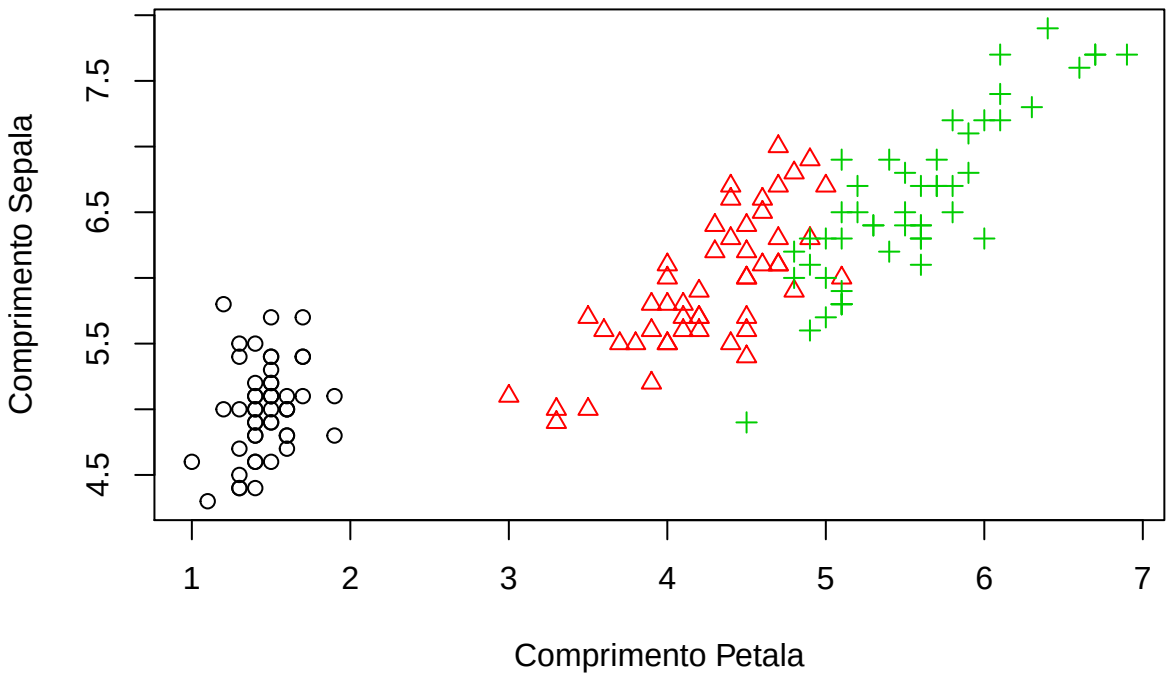



```
plot(iris[iris$Species=="virginica", "Petal.Length"],
     iris[iris$Species=="virginica", "Sepal.Length"],
     xlab="Comprimento Petala", ylab="Comprimento Sepala", type="p", pch=3, col='green')
```



Podemos também plotar todos os dados e colorir de acordo com a espécie em um único gráfico.

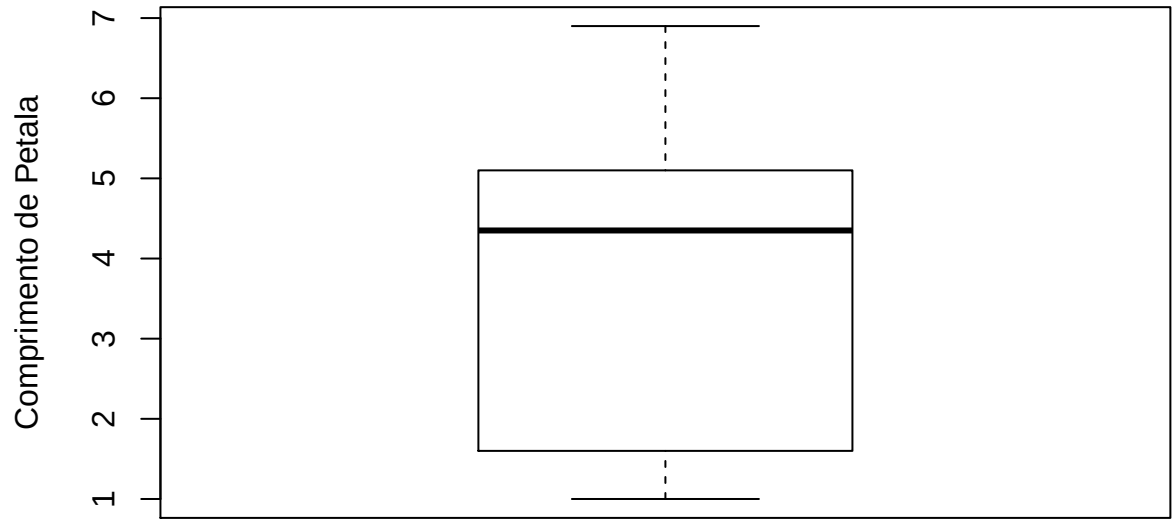
```
plot(iris[, "Petal.Length"],
     iris[, "Sepal.Length"],
     xlab="Comprimento Petala", ylab="Comprimento Sepala", type="p",
     pch=as.numeric(iris$Species), col=as.numeric(iris$Species))
```



Uma visualização interessante da distribuição dos dados é fornecida pelo gráfico de caixa (*boxplot*), que representa as mesmas medidas fornecidas pela função *summary*. A Figura abaixo mostra o gráfico de caixa para o comprimento de pétala. O traço horizontal central, dentro do retângulo representa a mediana. O lado inferior do retângulo representa o primeiro quartil, enquanto o topo representa o terceiro quartil. A altura do

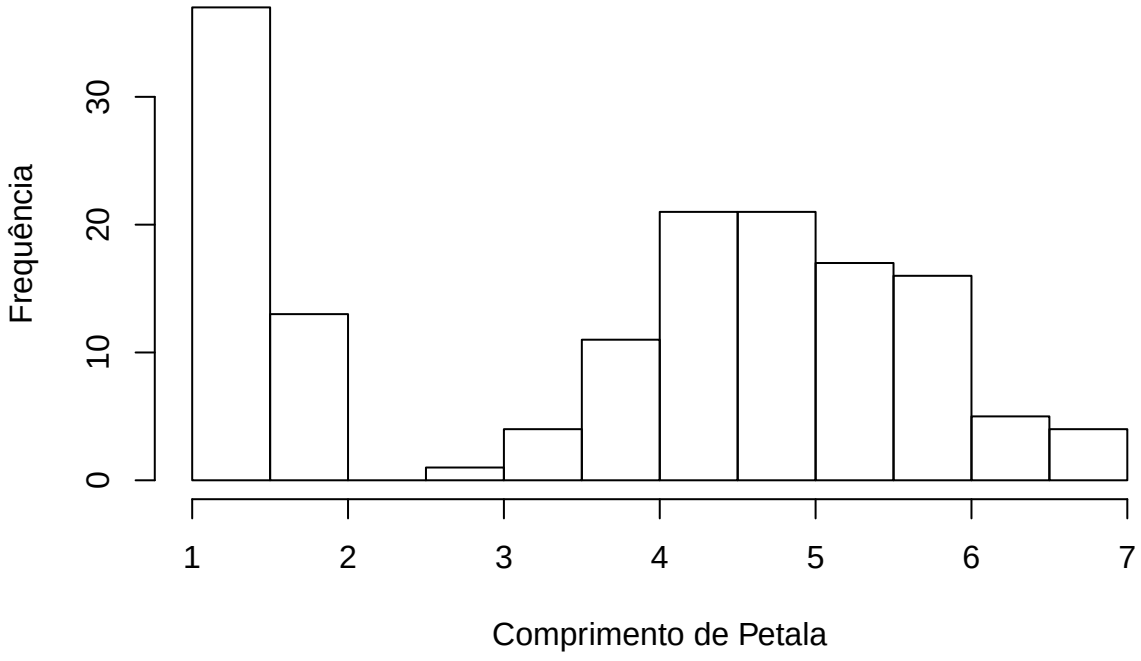
retângulo representa então o intervalo em que em situam os 50~% centrais dos dados.

```
boxplot(iris$Petal.Length, main="", ylab="Comprimento de Petala")
```



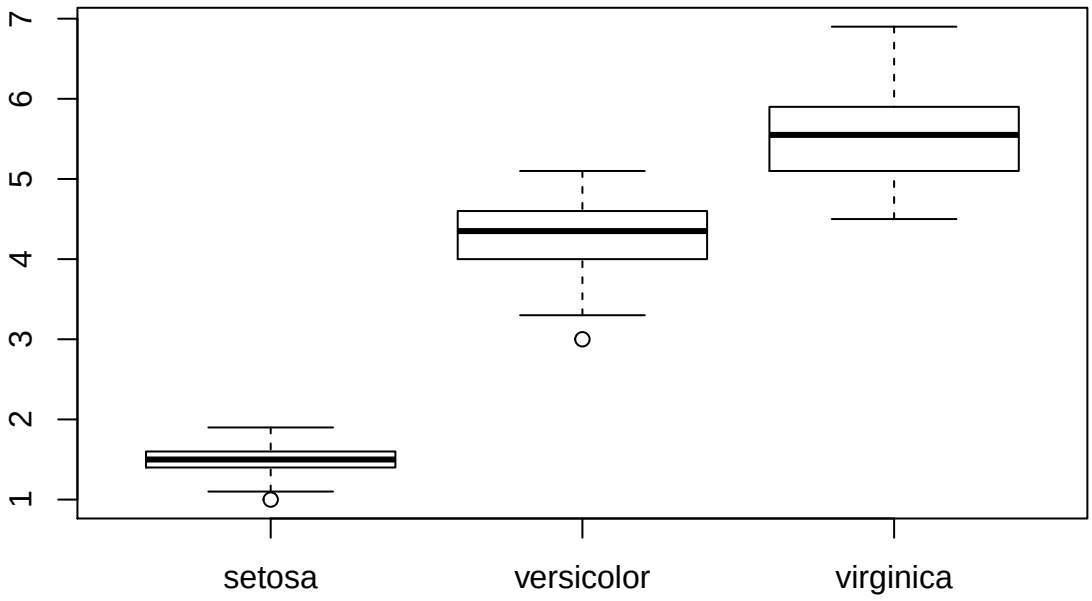
A Figura indica uma assimetria da mediana em relação ao primeiro e terceiro quartis. Nesse caso, podemos construir um *histograma* para melhor investigar o formato da distribuição. O histograma sugere que existem dois grupos dentro dos dados: o primeiro formado por instância com comprimento de pétala menor (entre 1 e 2) e um segundo grupo em que o comprimento se concentra em torno de 4.5.

```
hist(iris$Petal.Length, main="", xlab="Comprimento de Petala", ylab="Frequência")
```



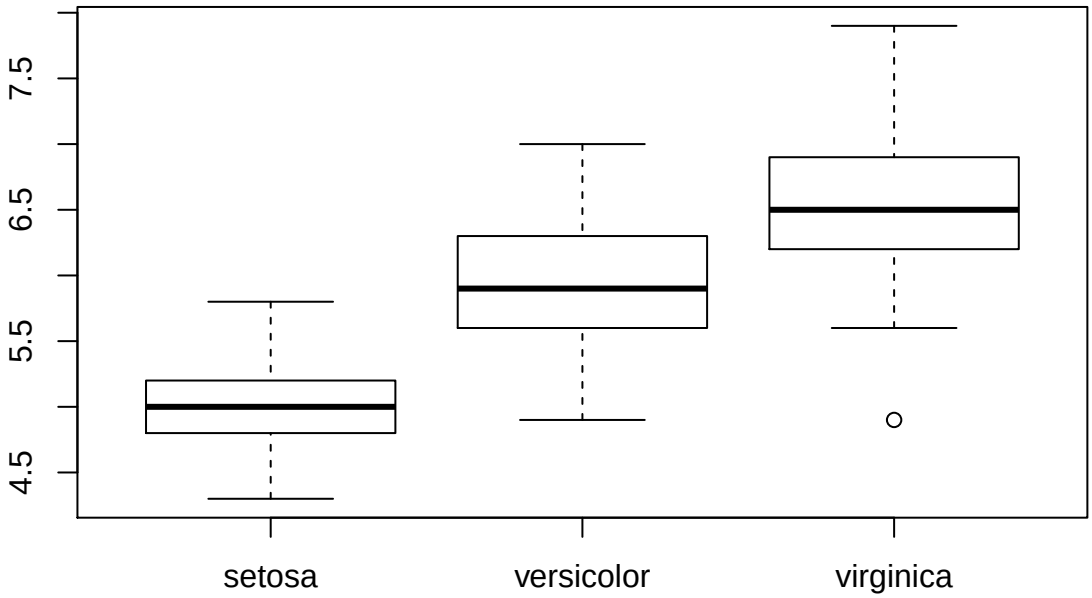
Uma aplicação útil do gráfico é caixa é comparar a distribuição de atributos agrupados por categoria. No exemplo a seguir, vamos visualizar a distribuição do comprimento de pétala, agrupado por espécie (setosa, versicolor ou virginica).

```
boxplot(Petal.Length ~ Species, data=iris)
```



A sintaxe *par1 ~ par2* indica que o gráfico de *par1* deve ser agrupado de acordo com as categorias definidas em *par2*. O parâmetro *data* especifica o dataframe que deve ser utilizado como fonte dos dados. Os pontos representados por círculos representam instâncias com valores destoantes (*outliars*). A Figura abaixo mostra o mesmo tipo de gráfico agrupado por espécie, desta vez para o comprimento da sépala, obtido com o comando abaixo.

```
boxplot(Sepal.Length ~ Species, data=iris)
```



Exercícios

1 - Faça o gráfico de caixa para cada um dos atributos numéricos do conjunto de dados iris.

2 - Faça o gráfico de caixa agrupado por espécie para cada um dos atributos numéricos do conjunto de dados

iris.

3 - Faça o gráfico de dispersão para todos os pares diferentes de atributos numéricos do conjunto de dados iris.

4 - A partir dos gráficos, que conclusões você consegue tirar sobre a relação entre atributos e espécies? Quais atributos são mais informativos se quisermos separar as três espécies?

Análise de correlação

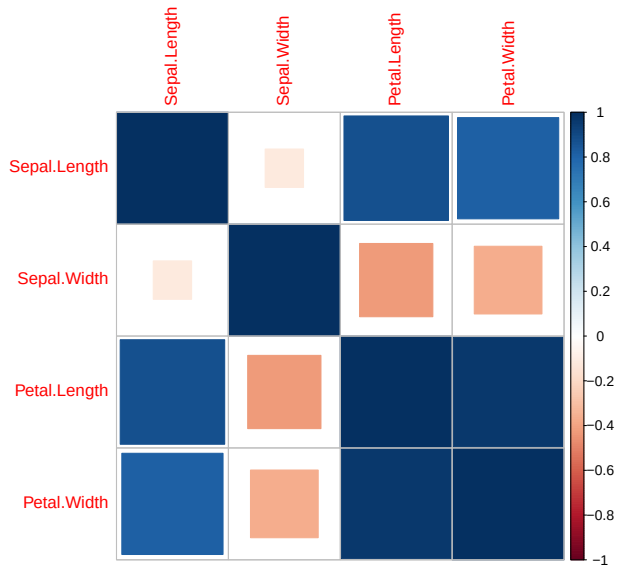
Muitas vezes queremos avaliar a correlação entre as variáveis que iremos usar em uma análise. Variáveis com valores altos de correlação podem trazer para alguns modelos, e em geral, não faz sentido utilizá-las em conjunto na indução de modelos. No R, podemos fazer uma avaliação rápida da correlação entre todas as variáveis de uma base de dados utilizando a função *cor*, que retorna a matriz de correlação.

```
mat_cor <- cor(iris[, -5])
mat_cor
```

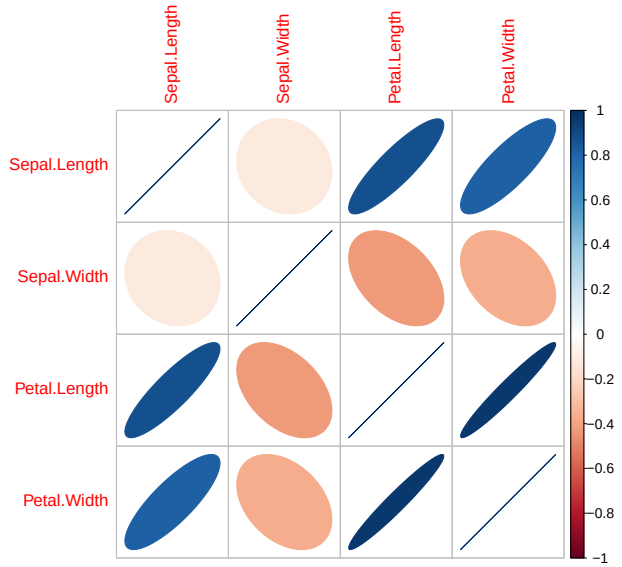
```
##              Sepal.Length Sepal.Width Petal.Length Petal.Width
## Sepal.Length    1.0000000  -0.1175698    0.8717538   0.8179411
## Sepal.Width    -0.1175698   1.0000000   -0.4284401  -0.3661259
## Petal.Length    0.8717538  -0.4284401    1.0000000   0.9628654
## Petal.Width     0.8179411  -0.3661259    0.9628654   1.0000000
```

Conforme o número de variáveis cresce, analisar a matriz pode se tornar uma tarefa pouco prática. Diversas visualizações podem auxiliar, por exemplo, *heatmaps* e visualizações de grafos. O pacote *corrplot* já possui diversas formas de visualizar uma matriz de correlação implementadas, conforme pode ser visto abaixo.

```
#se você não possuir o pacote será necessário o seguinte comando:
# install.packages("corrplot")
library(corrplot)
corrplot(mat_cor, method = "square")
```




```
corrplot(mat_cor, method = "ellipse")
```



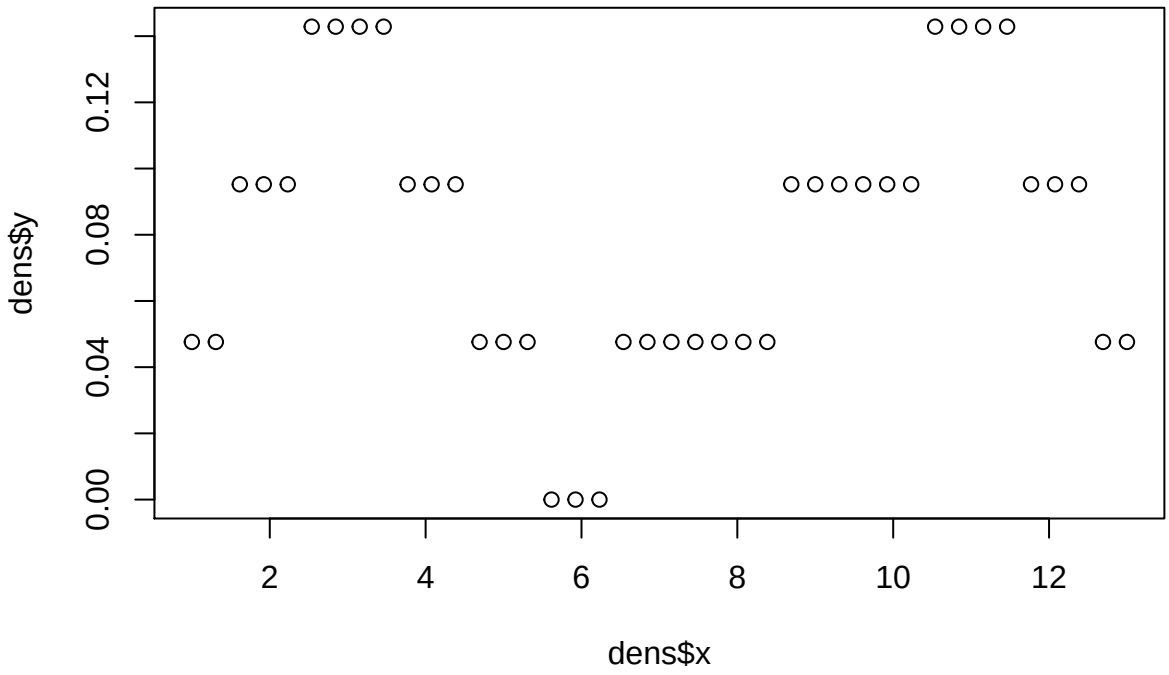
Estimação de densidade

Na última aula aprendemos sobre estimação de densidade não-paramétrica utilizando *Parzen Windows*. Vamos ver como ficaria a implementação da técnica em R utilizando o *kernel* hiper-cubo unitário.

```
parzen <- function(x, h=3, nro_pontos=100){
  pontos <- seq(min(x)-1, max(x)+1, length.out = nro_pontos)
  dens_estimada <- rep(NA, length(pontos))
  for (i in 1:length(pontos)) {
    #cálculo das distâncias/diferenças escalonadas em relação ao bandwith
    distancias <- abs(pontos[i] - x)/h
    #aplicação do kernel hipercubo unitário
    #contagem de número de pontos dentro do hipercubo com aresta h
    contagem_box <- rep(0, length(distancias))
    contagem_box[distancias < 1/2] <- 1
    #aplicação da normalização
    dens_estimada[i] <- sum(contagem_box)/(length(x)*h)
  }
  list(x=pontos, y=dens_estimada)
}
```

A função *seq* facilita a geração de sequências de números, neste caso ela é utilizada para a geração dos pontos para os quais a densidade será computada. O parâmetro *length.out* permite especificar o número de pontos que queremos gerar e a rotina se encarrega de obter os pontos espaçados de forma uniforme no intervalo especificado. Para gerar um vetor com um mesmo valor em todas as posições é utilizada a função *rep*, que simplesmente repete seu primeiro parâmetro de acordo com o número de vezes indicado pelo segundo parâmetro. Podemos usar a função da seguinte forma:

```
x <- c(2,3,4,8,10,11,12)
dens <- parzen(x, h=3, nro_pontos=40)
plot(dens)
```



Para entendermos o resultado, podemos inspecionar os dados gerados em intervalos de interesse. Por exemplo, se temos um hiper-cubo com aresta de tamanho 3, cada ponto será contado se estiver a uma distância 1,5 de uma amostra dos dados. Isso pode ser verificado na fronteira relacionada à amostra de valor 4:

```
dens$x[14:17]
```

```
## [1] 5.000000 5.307692 5.615385 5.923077
```

```
dens$y[14:17]
```

```
## [1] 0.04761905 0.04761905 0.00000000 0.00000000
```

Exercício

1. Modifique a função *parzen* para utilizar um *kernel* gaussiano. Lembrando que o *kernel* gaussiano é definido como:

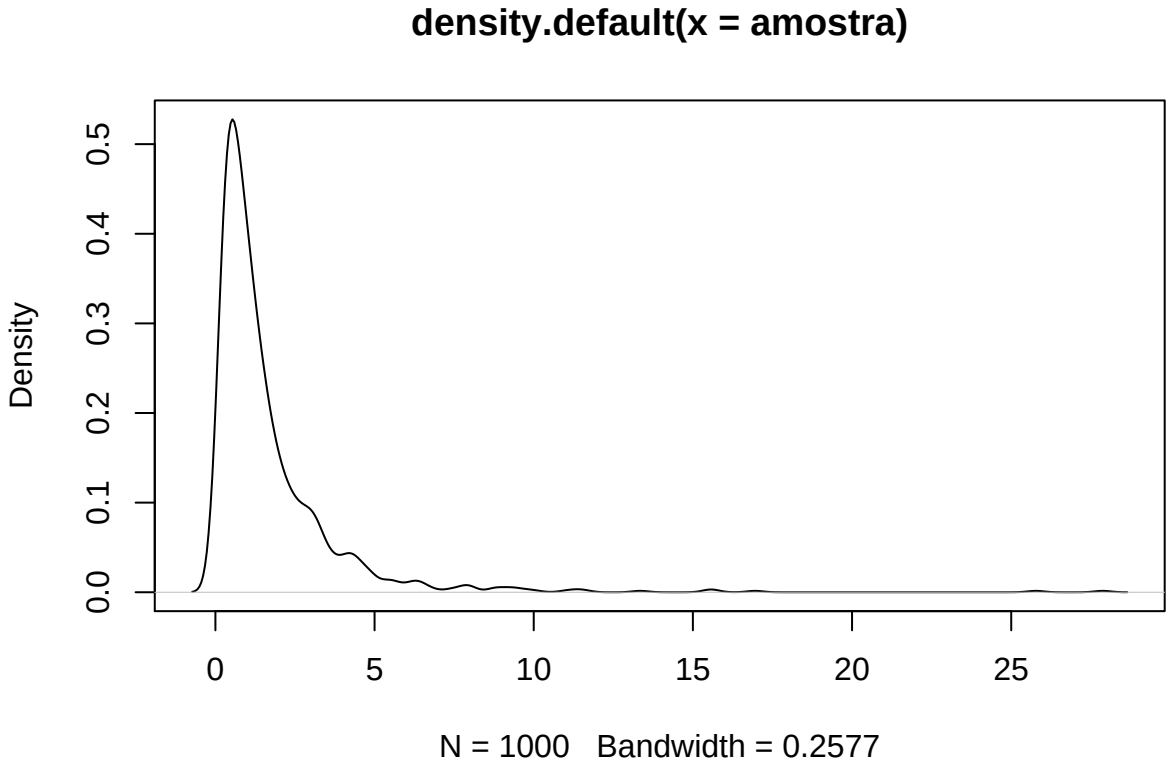
$$K(u) = \frac{1}{\sqrt{2\pi}} exp\left[-\frac{u^2}{2}\right]$$

No R, uma versão mais completa dessa técnica está implementada na função *density*, com diversos *kernels* acessíveis pelo parâmetro *kernel*.

Transformações

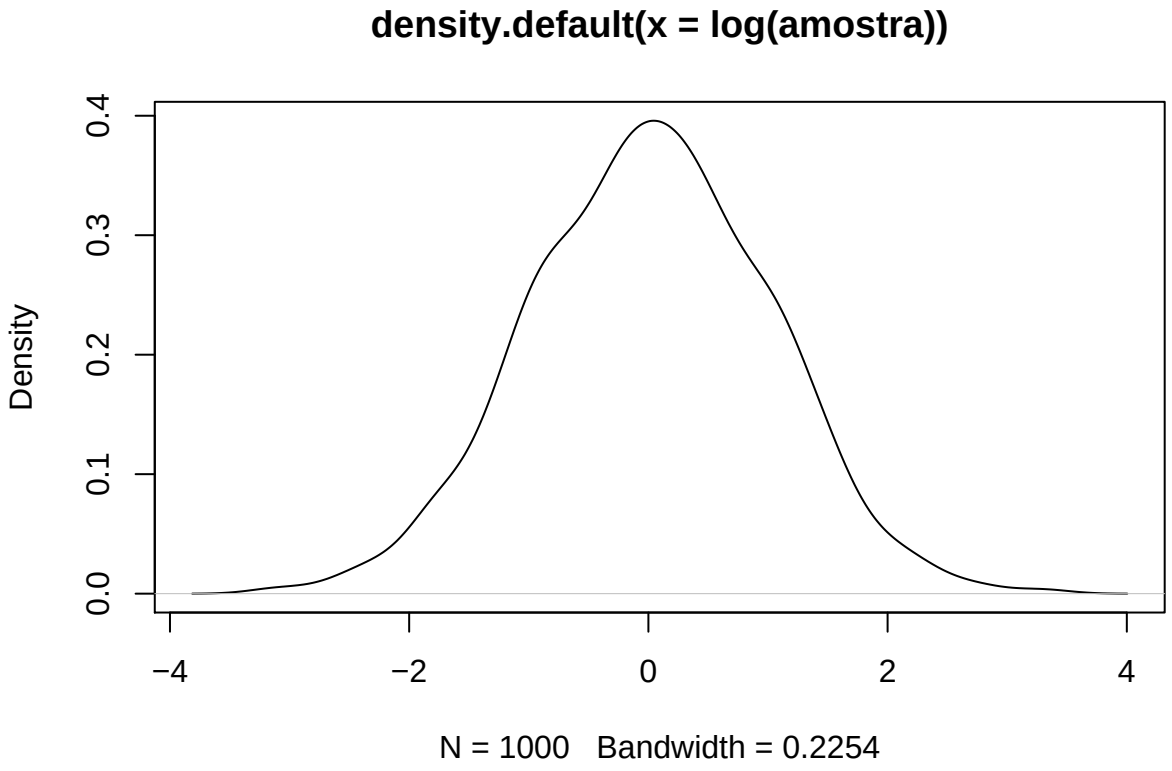
Muitas vezes podemos transformar as variáveis originais em outras variáveis com características interessantes para o modelo a ser utilizado. Um exemplo é quando os dados apresentam uma distribuição assimétrica similar à uma distribuição log-normal:

```
amostra <- rlnorm(1000)
plot(density(amostra))
```



Nesse caso, podemos transformar os dados para se aproximarem de uma distribuição normal aplicando o logaritmo:

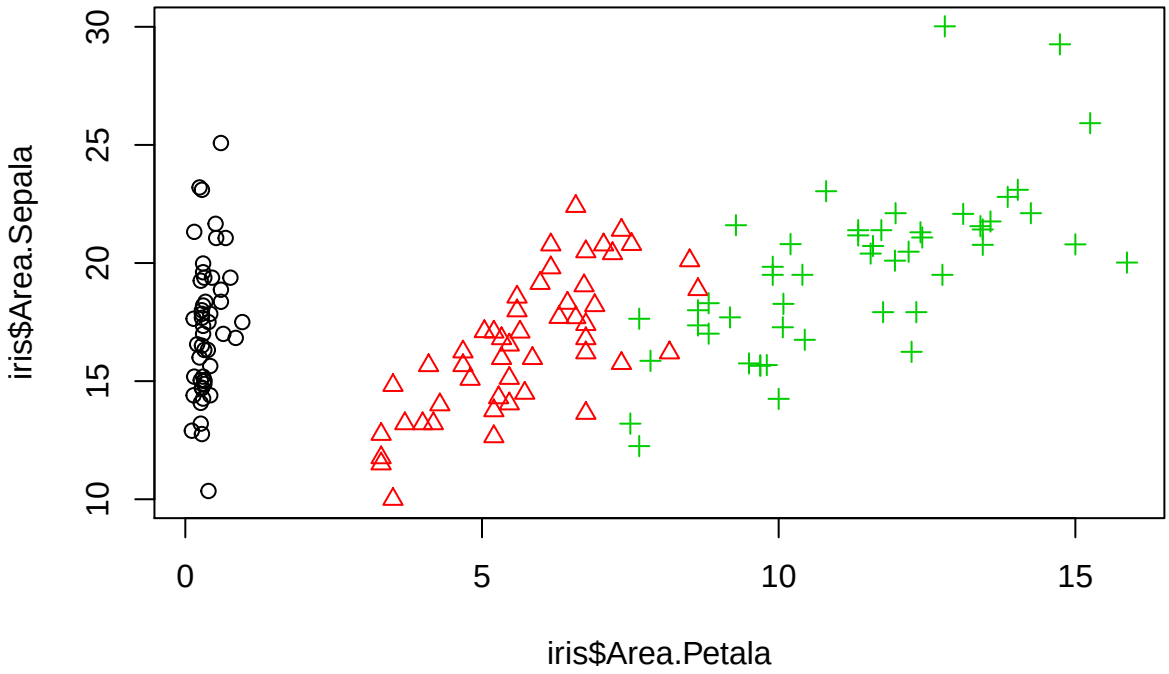
```
plot(density(log(amostra)))
```



Diversas transformações costumam ser utilizadas: x^2, $1/x$. Além desse tipo de transformação, também

podemos gerar variáveis por meio da combinação de variáveis originais. Por exemplo, podemos *simplificar* a base de dados Íris aproximando a área da pétala e da sépala, conforme abaixo:

```
iris$Area.Petala <- iris$Petal.Length*iris$Petal.Width
iris$Area.Sepala <- iris$Sepal.Length*iris$Sepal.Width
plot(iris$Area.Petala, iris$Area.Sepala, pch=as.numeric(iris$Species),
     col=as.numeric(iris$Species))
```



Esse processo de buscar por atributos informativos e que sejam mais apropriados para o modelo a ser utilizado é conhecido como *engenharia de atributos*. Muitas vezes o sucesso de uma aplicação não está na escolha do modelo, mas sim na escolha de uma boa representação dos dados.