7. Visualização de dados: gráficos em R

Neste capítulo, abordaremos algumas das principais funções existentes no R ao nível da visualização de dados através de gráficos. Serão abordadas as funções do package **graphics**, um dos packages base do R. Outros packages especializados como o **lattice** ou o **ggplot2** podem ser alternativas, não abordadas neste texto.

7.1 Gráficos de dispersão

A função mais usada e flexível para construir gráficos é a função **plot,** que permite construir gráficos de dispersão (ou *scatter plots*) representando uma (ou várias) variáveis numéricas (ditas dependentes) no eixo dos *yy* em função de outra no eixo dos *xx* (dita independente).

A função **plot** recebe como argumentos um vector para o eixo dos *xx* e um outro vector para o eixo dos yy. O primeiro destes vetores (*xx*) pode ser omitido assumindo o R que se trata de um vector com a sequência de números inteiros começada em 1 e com comprimento igual ao vector definido para o eixo dos *yy*.

O exemplo seguinte ilustra esta função:

```
> x<-seq(0.2,2,0.01)
> f <- function(x) (log(x))^2-x*exp(-x^3)
> plot(x, f(x))
```

A função plot pode receber um número considerável de argumentos opcionais, entre os quais podemos destacar:

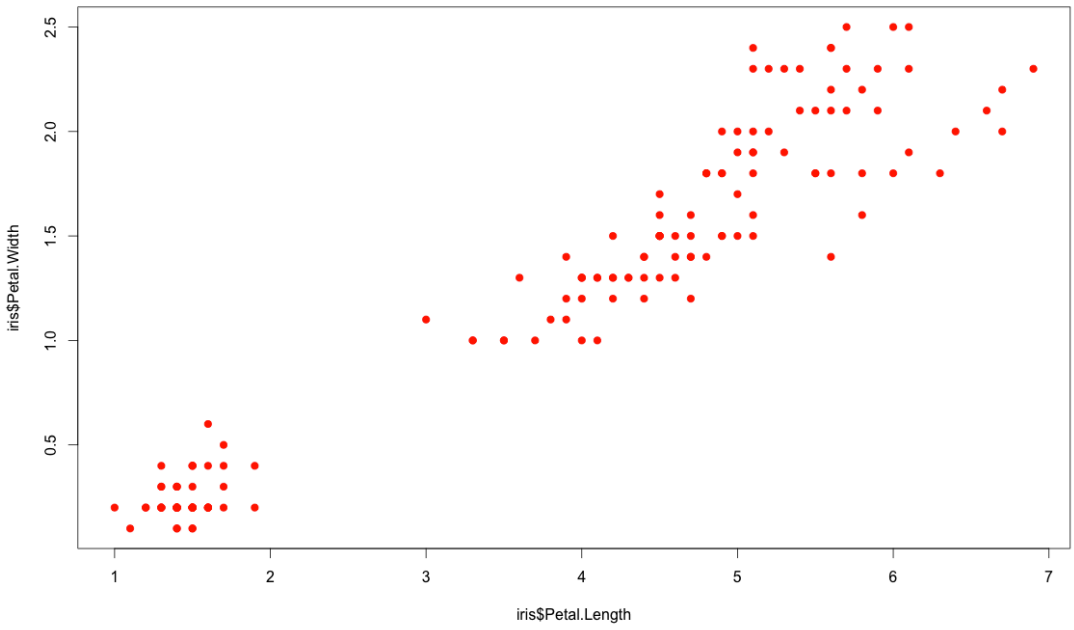
- *type* – tipo do gráfico (linhas – "l", pontos – "p", ...)
- *main* – título do gráfico
- *xlab*, *ylab* – legendas dos eixos x, y
- *col* – cor(es) dos pontos ou linhas
- *cex* – tamanho dos pontos
- pch – caracter do ponto

Caso se queira desenhar uma linha em vez de representar pontos individuais pode usar-se:

```
> plot(x, f(x), type="l")
```

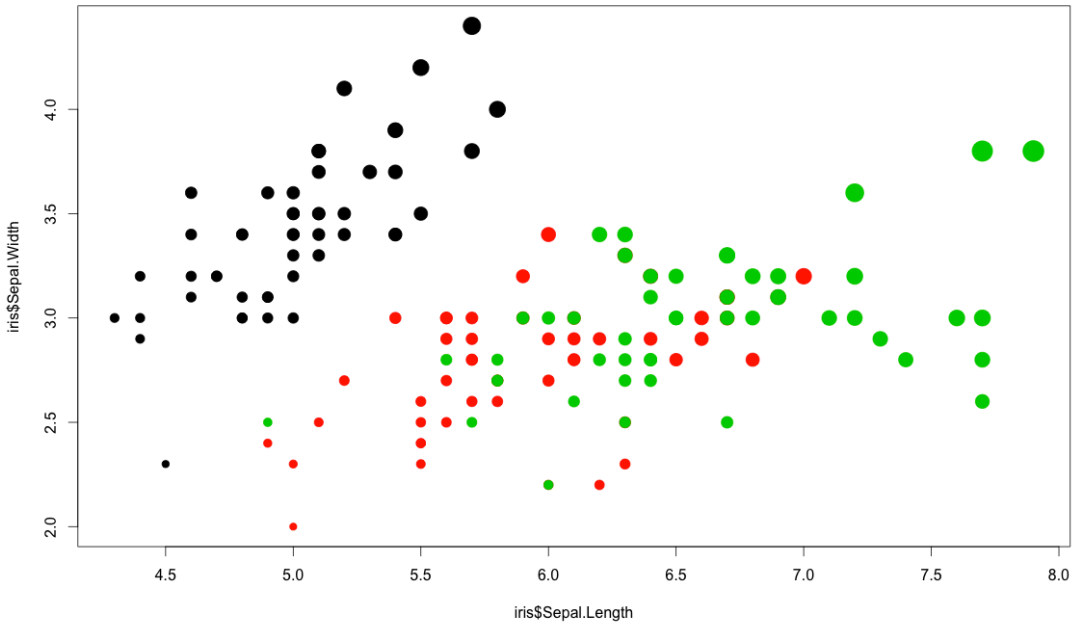
Vejamos um exemplo usando o data frame iris onde se usam os argumentos *col* e *pch*:

```
> plot(iris$Petal.Length, iris$Petal.Width, col="red", pch = 19)
```



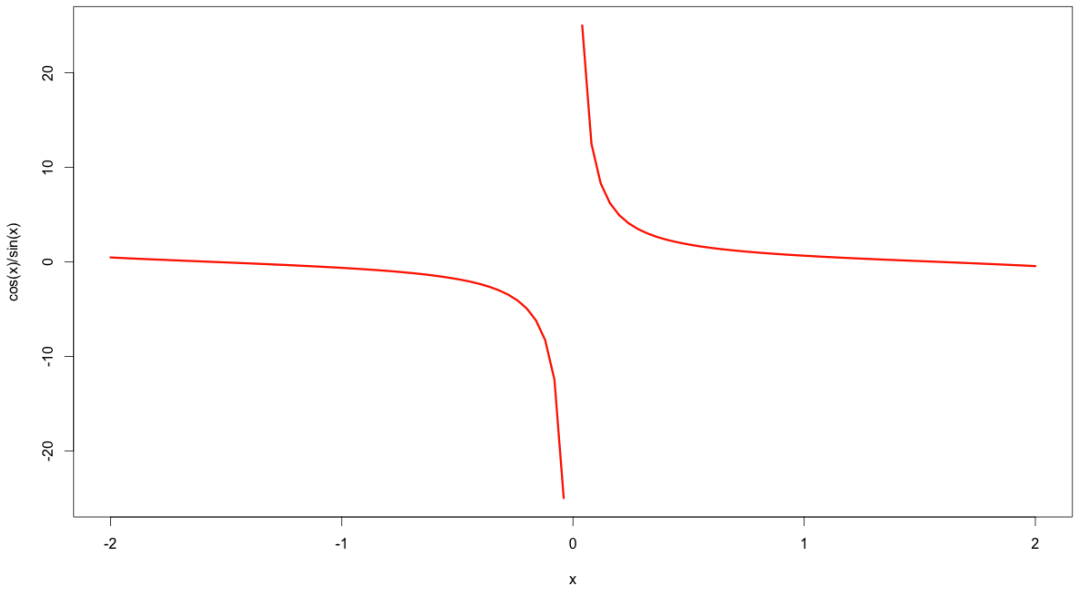
Vejamos um outro exemplo onde se definem as cores dos pontos e o seu tamanho:

```
> plot(iris$Sepal.Length, iris$Sepal.Width, col=iris$Species, pch =
19, cex = iris$Sepal.Length*iris$Sepal.Width*0.1)
```



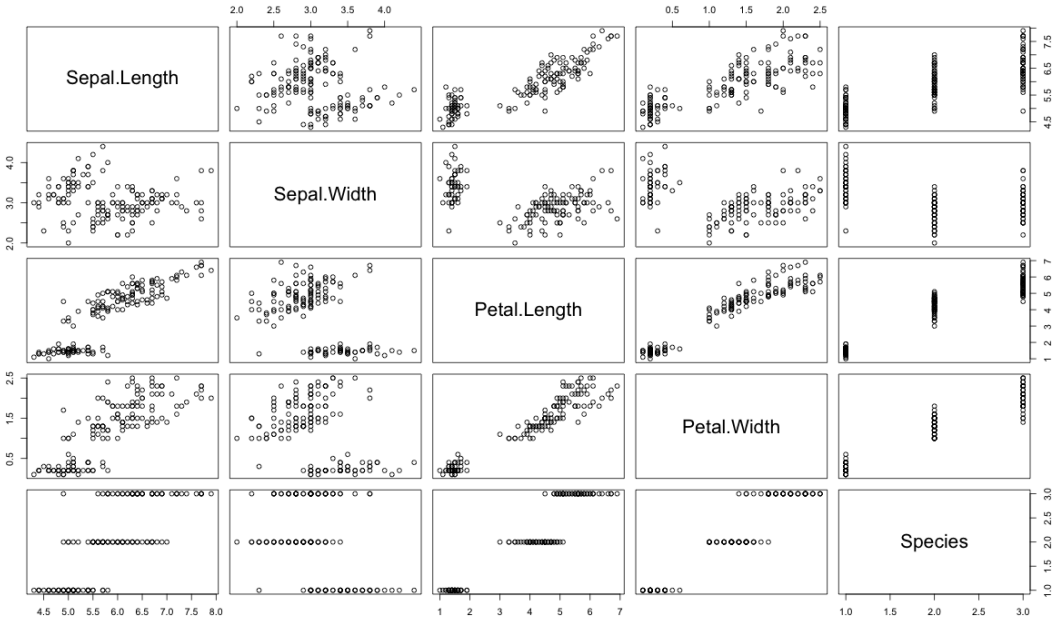
A função **curve** pode ser usada para representar gráficos de funções de uma forma mais simples. Vejamos um exemplo simples:

```
> curve(cos(x)/sin(x), -2, 2, col="red")
```



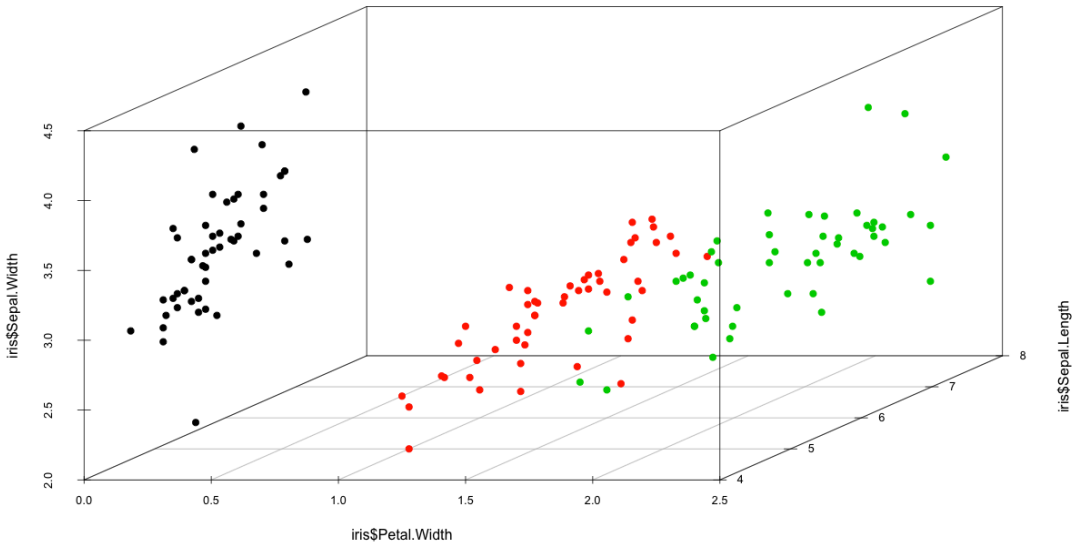
A função **pairs** permite representar gráficos de dispersão múltiplos entre todos os campos de um data frame, permitindo visualizar as relações entre as variáveis (e.g. quais são mais correlacionadas). Vejamos um exemplo da sua aplicação:

```
> pairs(iris)
```



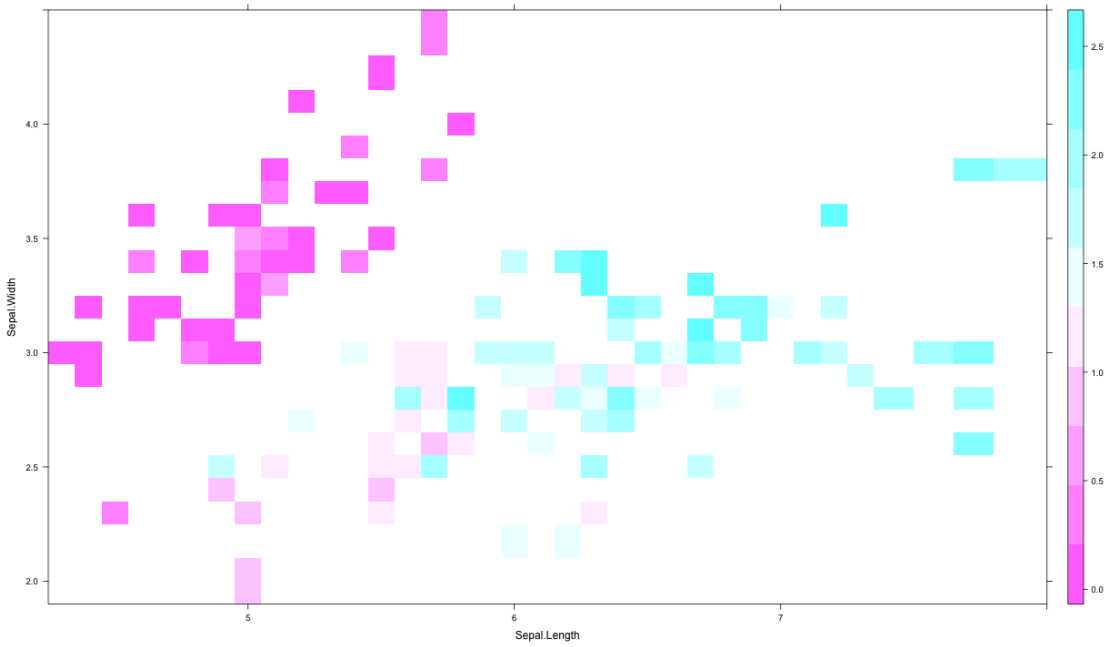
Adicionalmente, o R tem alguns packages que permitem a representação de três variáveis. Uma das hipótese é o uso de gráficos de dispersão 3D providenciados pelo package **scatterplot3d**:

```
> library(scatterplot3d)
>  scatterplot3d(iris$Petal.Width,iris$Sepal.Length,iris$Sepal.Width,
pch=19, color=as.integer(iris$Species))
```



Uma alternativa distinta são os chamados *levelplots*, que permitem representar três variáveis num gráfico de duas dimensões, onde a côr dos pontos é usada como forma de representar a 3ª variável, conforme o exemplo seguinte:

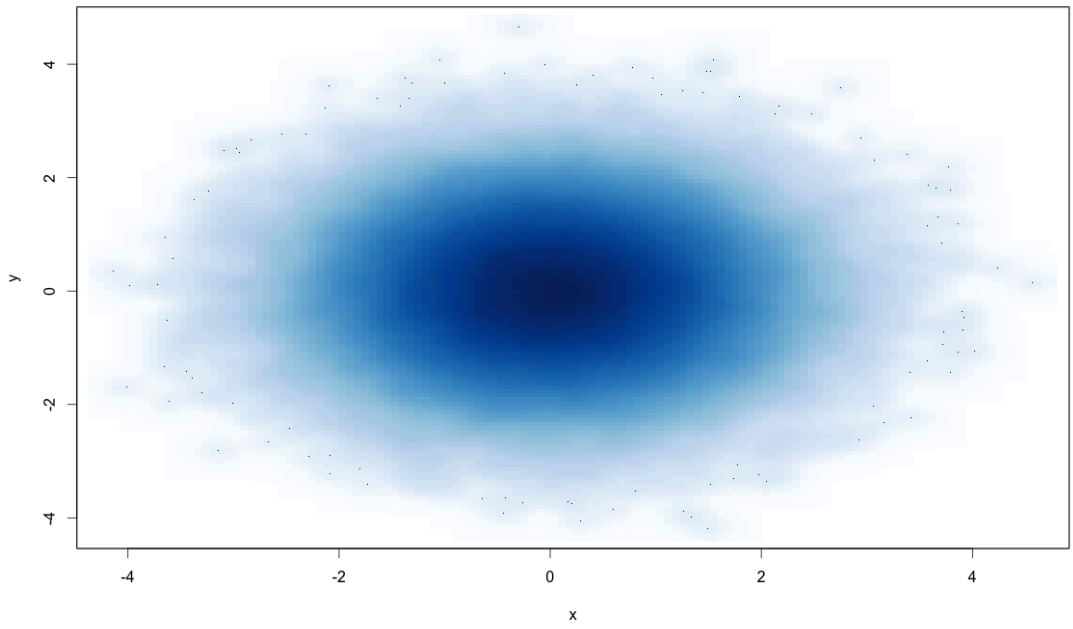
```
> library("lattice")
> levelplot(Petal.Width~Sepal.Length*Sepal.Width, iris)
```



Uma tarefa razoavelmente comum na análise de dados passa pela necessidade de visualizar grandes quantidades de dados. Neste caso, os pontos individuais de um gráfico de dispersão podem ser uma forma confusa, dada a aglomeração de pontos. Pode-se abordar este problema através de métodos de amostragem, i.e. considerando apenas uma parte dos pontos (por exemplo usando a função **sample** em R, tratada no capítulo 10).

Outra abordagem é o uso de outros tipos de gráficos que representam conjuntos de pontos com densidade de cores. Neste âmbito, destaca-se a função **smoothScatter**, que se ilustra no exemplo seguinte:

```
> x <- rnorm(100000)
> y = rnorm(100000)
> smoothScatter(x,y)
```



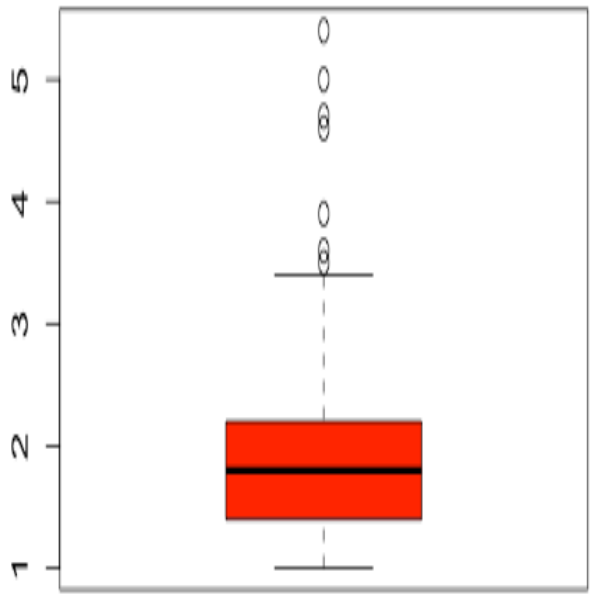
7.2 Boxplots

Os *boxplots* servem para visualizar a distribuição de valores de uma variável numérica mostrando algumas medidas de estatística descritiva de forma gráfica. Num boxplot típico, a mediana é dada por um traço horizontal central, uma zona rectangular

cujo lado superior é dado pelo terceiro quartil (Q3) e o inferior pelo primeiro quartil (Q1). Os traços superior e inferior (os chamados *bigodes*) mostram o valor máximo e mínimo, respetivamente, ou um outro valor limite. No caso da linha superior o seu valor é dado pelo menor dos seguintes valores: o máximo ou o Q3 somado de 1,5 vezes o intervalo inter-quartis (Q3-Q1). Se existirem valores superiores, estes são mostrados por pontos individuais (possíveis *outliers*). Algo de simétrico acontece na linha inferior.

Este tipo de gráficos pode ser construído usando a funçãoo **boxplot** que permite uma configuração alargada. Veja-se o exemplo seguinte:

```
> boxplot(eData$Magnitude, col="red")
```



Os cálculos seguintes complementam o gráfico anterior:

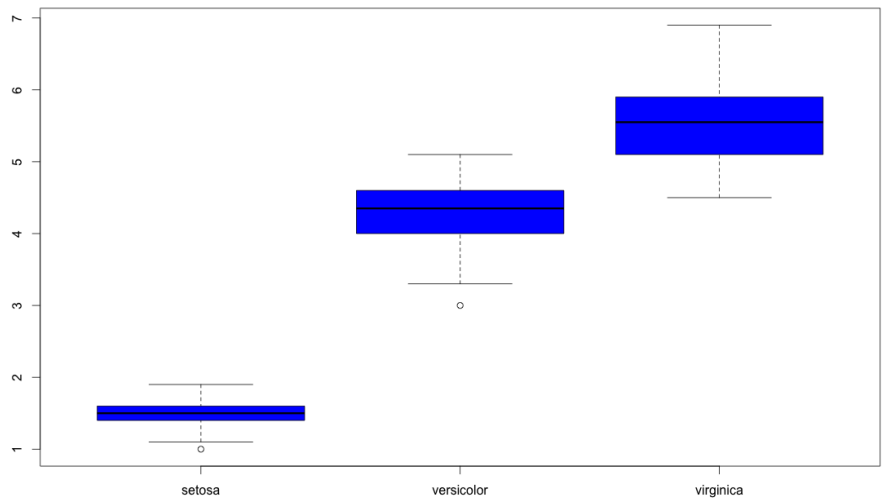
```
> boxplot(eData$Magnitude, col="red")
> range(eData$Magnitude)
[1] 1.0 5.4
> mean(eData$Magnitude)
[1] 1.860278
> mean(eData$Magnitude, trim = 0.1)
[1] 1.785764
> median(eData$Magnitude)
[1] 1.8
> quantile(eData$Magnitude, c(0.25, 0.75))
25% 75%
```

```
1.4 2.2
> quantile(eData$Magnitude, 0.75)+IQR(eData$Magnitude)*1.5
3.4
> quantile(eData$Magnitude, 0.25)-IQR(eData$Magnitude)*1.5
25%
0.2
```

Note-se que neste caso o bigode superior corresponde a Q3 + 1.5 IQR existindo alguns outliers (valores acima deste valor), enquanto o bigode inferior ficará no valor mínimo da gama de valores.

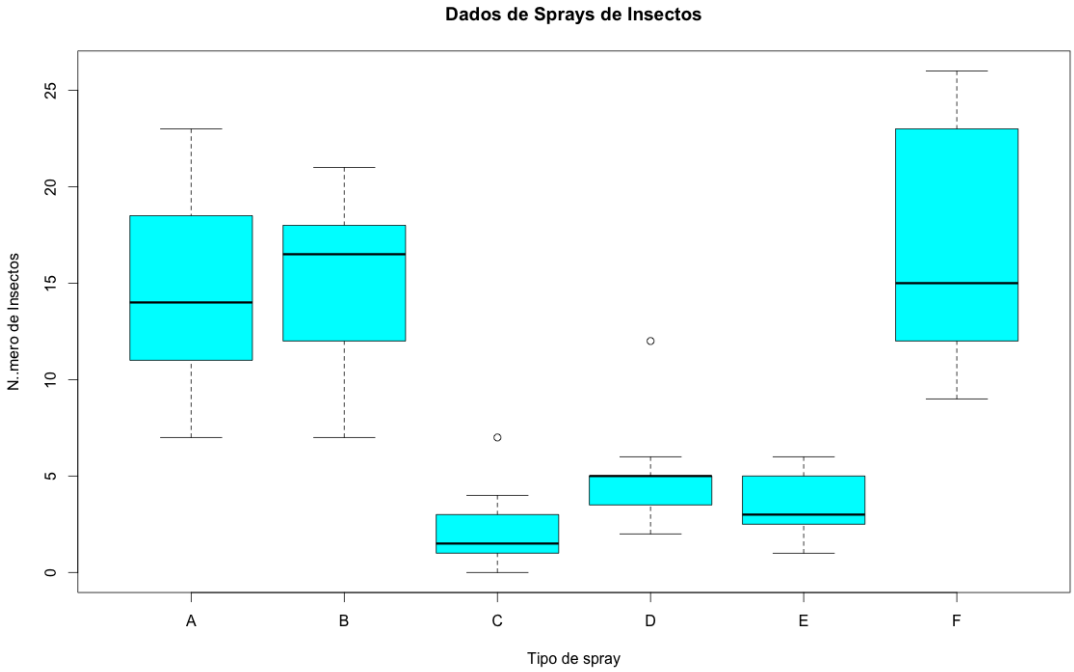
Estes gráficos são sobretudo úteis para comparar distribuições de valores em situações distintas. Podem, por exemplo, ilustrar a distribuição de uma variável dependente de um factor, ajudando a perceber a sua relação. No exemplo seguinte, comparamos a distribuição do comprimento das pétalas em três tipos de flores (usa-se o conjunto de dados *iris*):

```
> boxplot(iris$Petal.Length ~iris$Species, col = "blue")
```



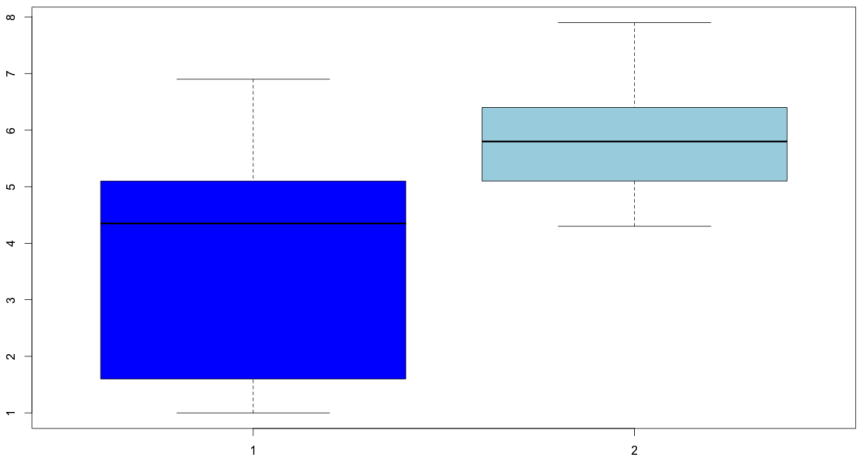
Vamos fazer o mesmo para outra tabela que já vem com o R. Esta trata da utilização de inseticidas:

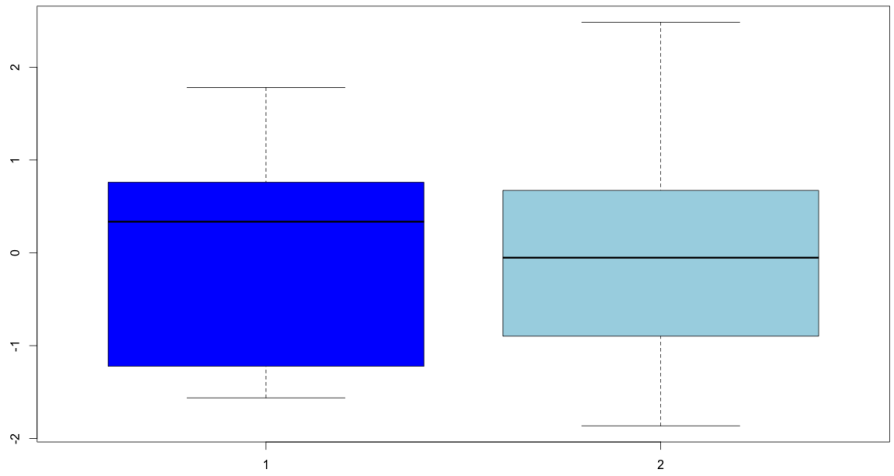
```
> boxplot(count ~ spray, data = InsectSprays, xlab = "Tipo de spray", ylab = "Número de Insectos", main = "Dados de Sprays de Insectos", col="cyan")
```



Podemos usar boxplots para ilustrar os resultados de uma standardização de valores.

```
> boxplot( iris$Petal.Length, iris$Sepal.Length, col= c ("blue",
"lightblue"))
> sl = as.vector(scale(iris$Sepal.Length))
> pl = as.vector(scale(iris$Petal.Length))
```





7.3 Histogramas

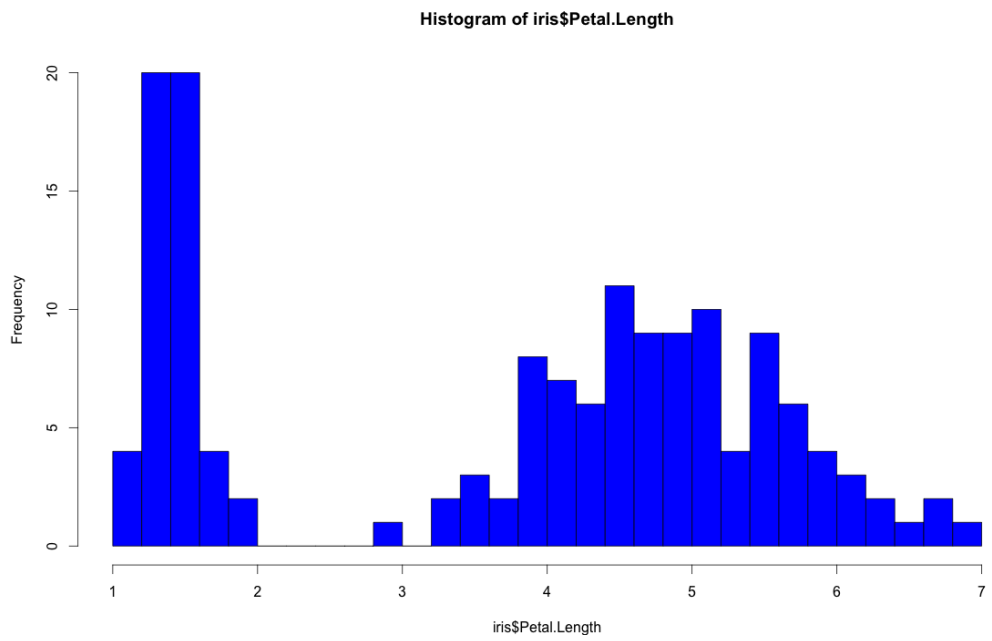
Os histogramas são gráficos bastante simples que permitem caracterizar a distribuição de frequências de valores de uma variável. Nos histogramas, a altura de cada rectângulo é proporcional ao número de observações do intervalo de valores na base. Os histogramas permitem ter uma primeira ideia da distribuição de dados. Por exemplo, pode verificar-se se os dados seguem uma curva "em forma de sino" que indicie uma distribuição normal ou se há enviesamento nos valores.

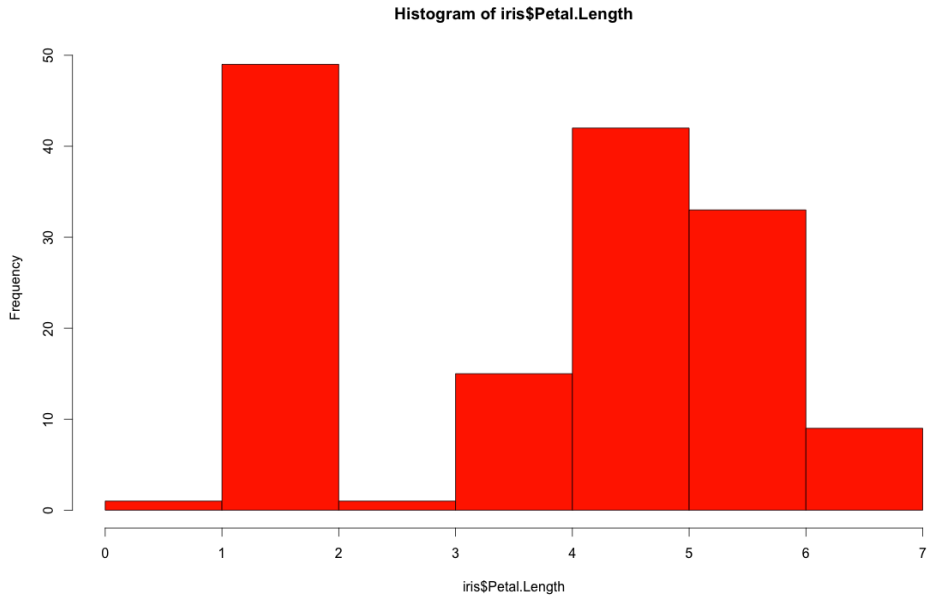
Em R, a construção de histogramas pode ser conseguida através do uso da função **hist**. Nesta função, os principais parâmetros são o vetor de valores da variável a representar e o número de posições de quebra (*breaks*). As posições de quebra também podem ser indicadas explicitamente por um vetor.

No exemplo seguinte são construídos dois histogramas usando um dos campos do data frame *iris*, já usado anteriormente.

```
> data(iris)
> hist(iris$Petal.Length, breaks= 30, col="blue")
> hist(iris$Petal.Length, breaks= 0:7, col = "red")
```

O resultado é mostrado nas figuras seguintes:

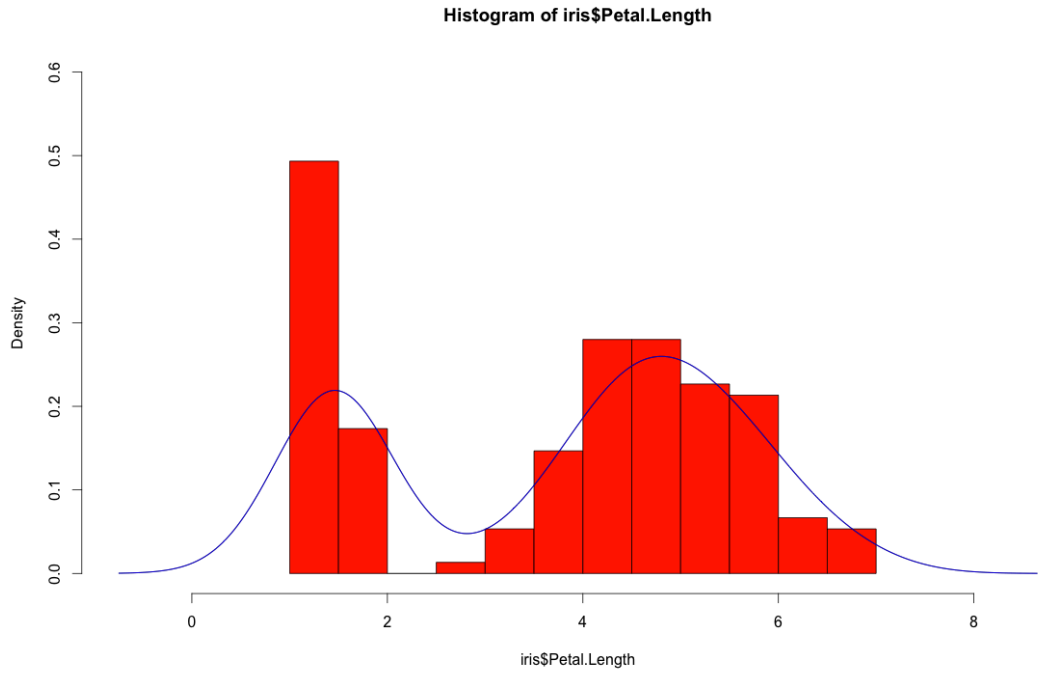




O exemplo seguinte mostra como podemos calcular uma função de densidade e sobrepor esta, na forma de uma linha a um histograma, usando a mesma variável dos exemplos anteriores. Note-se o uso do parâmetro *probability* que define que os valores do histograma serão definidos de forma a que a área total das suas barras some 1. Esta alteração é necessária para normalizar os valores, sendo assim comparáveis com os gerados pela funçãoo dens. O papel da função lines será explicado em mais detalhe na secção 7.6.

```
> dens = density(iris$Petal.Length)
```

```
> hist(iris$Petal.Length, breaks= 10, xlim=range(dens$x),
   ylim =c(0,0.6), probability = T, col = “red”)
> lines(dens, col = “blue”)
```

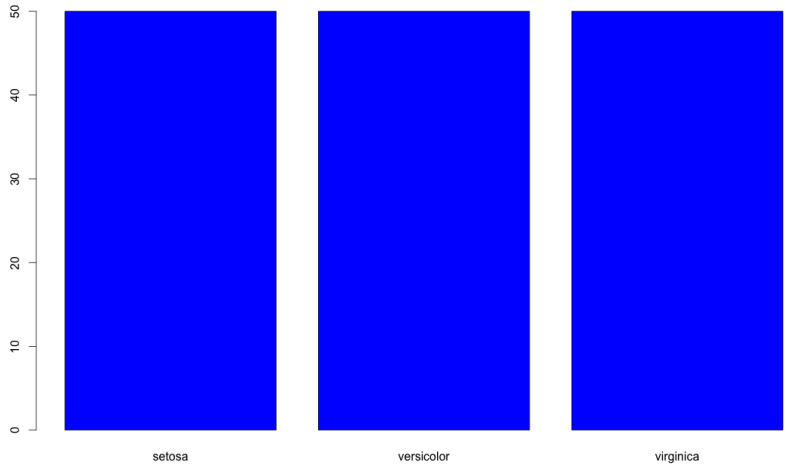


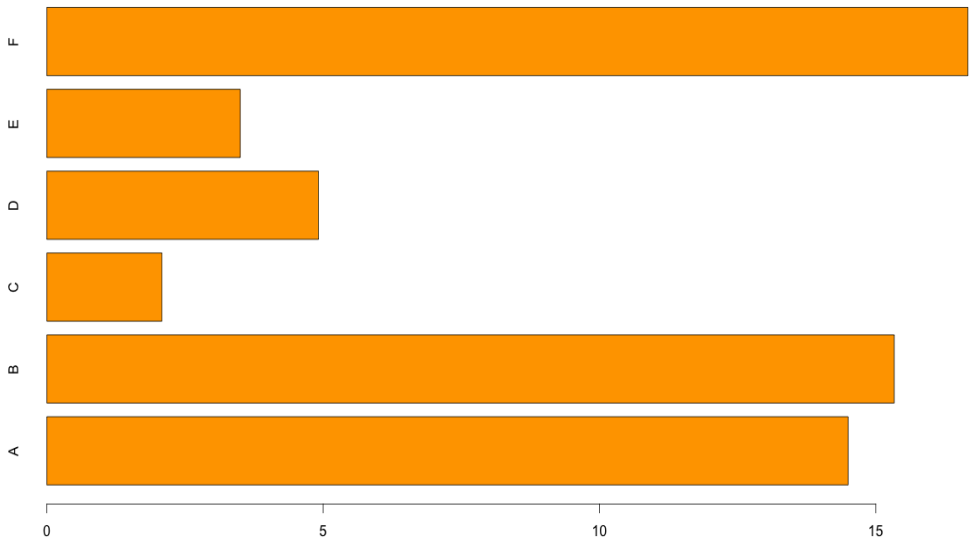
7.4 Gráficos de barras

Os gráficos de barras são uma forma de representar valores numéricos associados a categorias. Podem ser usados para representar frequências de fatores ou para sumariar métricas numéricas em função de categorias. A função barplot permite construir em R gráficos de barras, quer verticais quer horizontais, definindo-se para o efeito o parâmetro *horiz*..

Alguns exemplos usando conjunto de dados já anteriormente abordados:

```
> barplot(table(iris$Species), col = "blue")
> barplot(tapply(InsectSprays$count, InsectSprays$spray, mean),
col = "orange”, horiz = T)
```

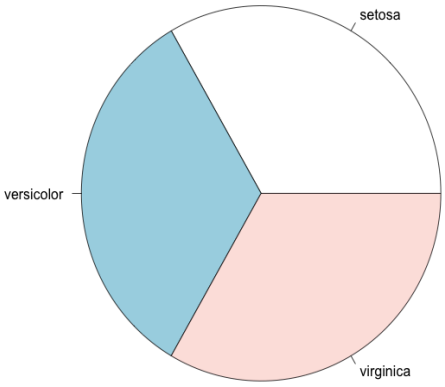




7.5 Gráficos do tipo pie-chart

Os gráficos do tipo pie-chart permitem representar frequências de ocorrência de valores distintos de uma variável nominal. Vejamos um exemplo:

```
> pie(table(iris$Species))
```



7.6 Sobreposição de objetos gráficos

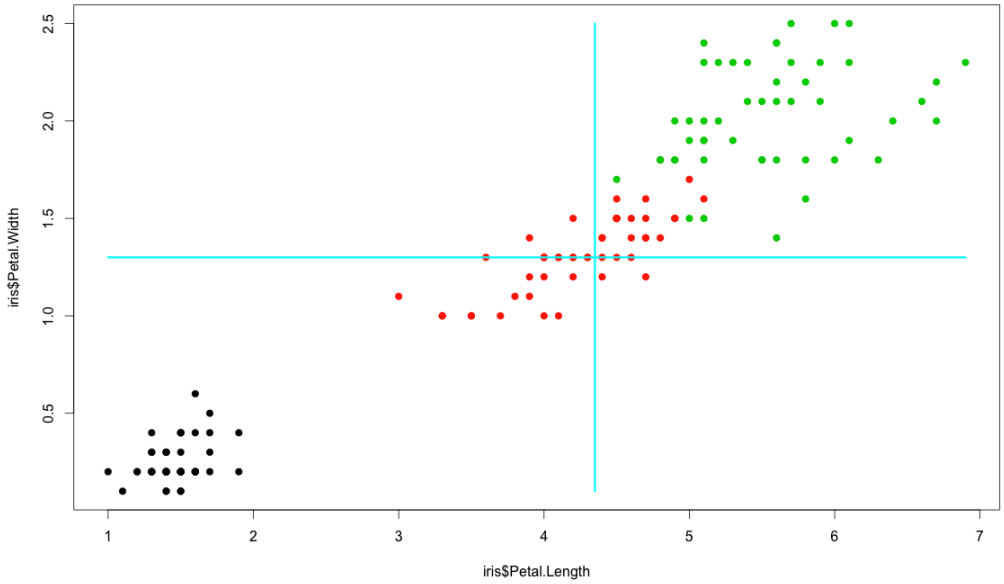
O R permite que num único gráfico possam ser sobrepostos vários elementos. Existem algumas funções cuja operação sobrepõe elementos sobre um gráfico pré-existente, construído com a função **plot**, ou outras funções básicas. Entre estas funções podem destacar-se as seguintes:

- *lines* – sobrepõe linhas conectando um conjunto de pontos dado por segmentos de reta
- *abline* – sobrepõe uma linha definida pelo declive e ordenada na origem
- *points* – sobrepõe um conjunto de pontos
- *legend* – permite colocar legendas no gráfico
- *text* – sobrepõe campos de texto
- *symbols* – sobrepõe símbolos matemáticos
- *axis* – inclui textos nos eixos
- *polygon* – inclui áreas "pintadas"
- *curve* (usando a opção *add = T*)

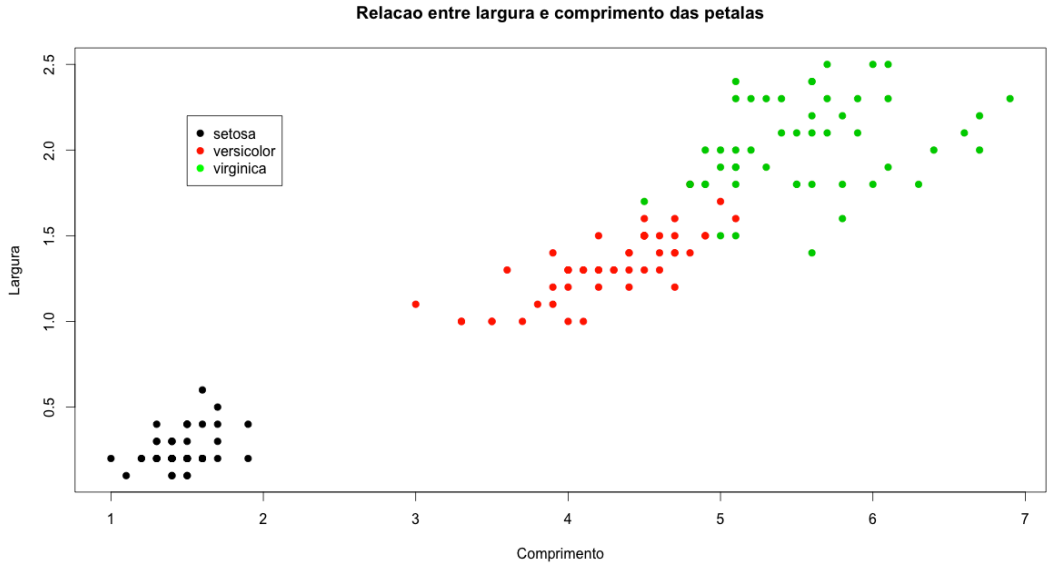
Vejamos dois exemplos simples:

```
> plot(iris$Petal.Length, iris$Petal.Width, col=iris$Species, pch =
19)
> lines(rep(median(iris$Petal.Length),2), c(min(iris$Petal.Width),
```

```
max(iris$Petal.Width)), col = "cyan", lwd=3)
> lines(c(min(iris$Petal.Length), max(iris$Petal.Length)),
rep(median(iris$Petal.Width),2), col= "cyan", lwd=3)
```

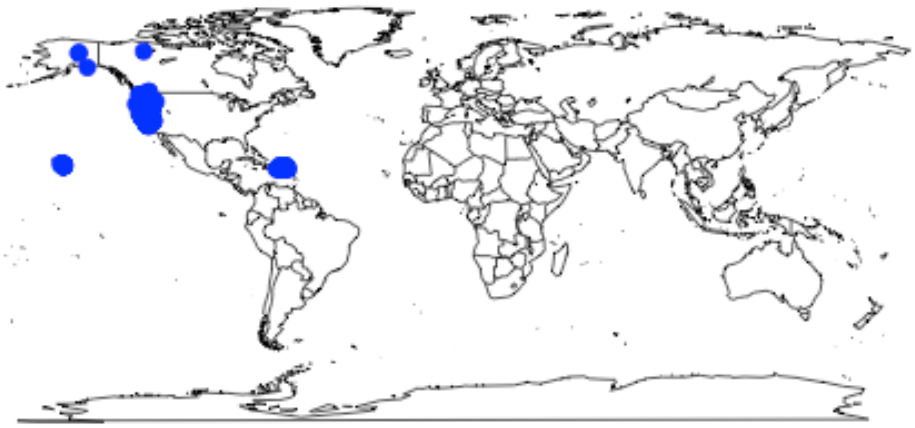



```
> plot(iris$Petal.Length, iris$Petal.Width, col=iris$Species, pch =
19, main = "Relacao entre largura e comprimento das petalas",
ylab="Largura", xlab="Comprimento")
>        legend(1.5,        2.2,        legend=levels(iris$Species),
col=c("black","red","green"), pch=c(19,19,19))
```



Um último exemplo mostra-nos como podemos usar estas potencialidades para sobrepor informação sobre mapas, neste caso um mapa mundial interno do R.

```
> library("maps")
> map("world")
> points(eData$Lon, eData$Lat, pch=19, col="blue")
```



7.7 Formatação e exportação de gráficos

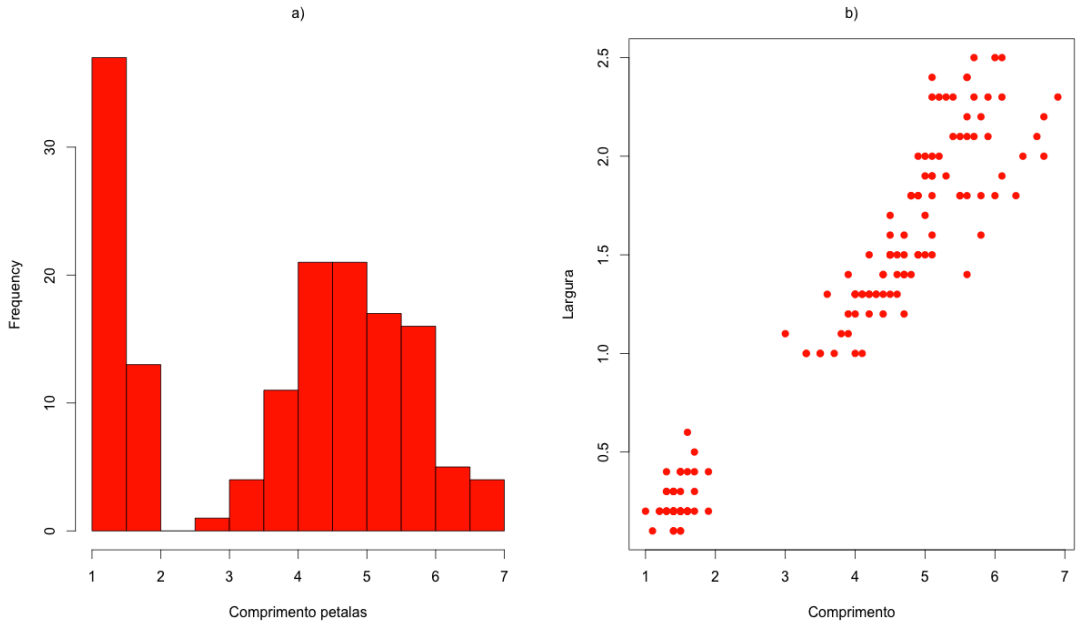
O R permite que se construam painéis com vários gráficos individuais organizados horizontal e/ ou verticalmente. Esta definição é realizada através da função **par**. Esta função é usada para definir uma grande quantidade de parâmetros gráficos.

Neste caso, usaremos o argumento *mfrow* para definir o número de linhas e colunas para organizar os dados. Por exemplo, se definirmos *mfrow = c(2.3)*, estaremos a organizar um painel com 6 gráficos em duas linhas e três colunas. Esta definição é realizada antes de se definirem os gráficos definindo a sua organização. Os gráficos serão então colocados posição a posição até completarem os espaços disponíveis.

Veja-se o exemplo seguinte, onde a função **mtext** é usada para colocar algum texto acima de cada um dos gráficos individuais:

```
> par(mfrow=c(1,2))
>  hist(iris$Petal.Length, breaks= 10, col = "red", xlab =
"Comprimento petalas", main="")
> mtext(text="a)", side=3, line=1)
```

```
> plot(iris$Petal.Length, iris$Petal.Width, col="red", pch = 19,
xlab="Comprimento", ylab="Largura", main="")
> mtext(text="b)", side=3, line=1)
```



Uma outra funcionalidade importante do R passa pela possibilidade de gravar os conteúdos de um gráfico (ou um painel de gráficos). Por exemplo, se quisermos guardar as nossas figuras num formato PDF, usa-se a função **pdf**. Esta é chamada antes de se gerar o gráfico, definindo como argumentos o nome do ficheiro e as dimensões do gráfico (em polegadas). Todos os comandos subsequentes irão gerar o gráfico respetivo sendo o processo terminado com o comando **dev.off()**.

Como exemplo, se quisermos guardar num ficheiro pdf a figura anterior deveríamos executar os seguintes comandos:

```
> pdf(file = “grafico.pdf”, height=4, width=8)
> par(mfrow=c(1,2))
> hist(iris$Petal.Length, breaks= 10, col = "red", xlab =
"Comprimento petalas", main="")
> mtext(text="a)", side=3, line=1)
> plot(iris$Petal.Length, iris$Petal.Width, col="red", pch = 19,
xlab="Comprimento", ylab="Largura", main="")
> mtext(text="b)", side=3, line=1)
> dev.off()
```

Se quiséssemos usar os formatos PNG, BMP, TIFF ou JPEG, o processo seria semelhante, usando-se as funções **png**, **bmp**, **tiff** e **jpeg**, respetivamente. Note que nestes casos a unidade usada para as dimensões da imagem serão pixels.

Exercícios resolvidos

1. Tendo em consideração os dado airquality, crie um diagrama de caixa (boxplot) para analisar a variação da Radiação solar entre os vários meses.
Titulo: "Diagrama de caixas - Airquality";
Cor das caixas : verde;
Texto dos eixos de X e Y : "Meses" e "Rad. Solar (inL angleys)";
Alterar nomes de meses para "Maio","Jun","Jul", "Agos", "Set".

```
> boxplot(airquality$Solar.R~airquality$Month, col='green',
        main ="Diagrama de caixas - Airquality",
        names=c("Maio","Jun","Jul", "Agos", "Set"),
        xlab="Meses", ylab="Rad. Solar (inL angleys)")
```

2. Crie um gráficos de dispersão que represente a relação entre as variáveis Ozone e Solar.R. Deve filtrar os dados em falta.
Titulo: "Ozono vs Radiação Solar";
Eixos: cor azul com números a laranja;
Texto dos eixos : modificar cor e tamanho;
Usar para o desenho dos pontos "pch =19" explorar outros símbolos;
Insira uma linha no ponto médio dos valores de Radiação Solar usando a função ;
Desafio: criar um degradé de verde para vermelho, onde a cor verde fica perto da origem. A atribuição de cor deve seguir uma distribuição circular, ou seja, a cor nos pontos (0,a) e (0,a) deve ser a mesma.

```
getColor<- function(x,y){
  maximo = max(c(x,y))+1 #+1 devido a arredondamentos
  print(maximo)
  colors = "c("
    for(i in 1:length(x)){
      #calculo do raio
      r = sqrt(x[i]^2+y[i]^2)
      color = r/maximo
```

```
      actualCor      =      eval(parse(text=paste("rgb(",color,",",1-
color,",",0,")")))
      colors = paste(colors,"'",actualCor,"',",sep="")
    }
  colors = paste(substr(colors,1,nchar(colors)-1),")",sep="")
  colors
}

> dados <- airquality[complete.cases(airquality), ]
> plot(dados$Ozone, dados$Solar.R,
     col=eval(parse(text=getColor(dados$Ozone, dados$Solar.R))),
     pch =19,
     main = "Ozono vs Radiacao Solar",
     xlab = "Ozono",
     ylab = "",
     axes=FALSE
     )

> abline(mean(dados$Solar.R),0, col='red')
>   axis(1,   col="blue",   col.ticks="green",   col.axis="orange",
cex.axis=0.8)
>   axis(2,   col="blue",   col.ticks="green",   col.axis="orange",
cex.axis=0.8)
> mtext("Radiacao Solar", side=2, line=3, col="purple", cex=1.5)
```

3. Coloque na mesma janela, lado a lado, dois gráficos que representem as funções **cos** e **sin** no intervalo [–2 , 2].

```
par(mfcol=c(1,2))
curve(cos(x),-2,2, col ='green')
curve(sin(x),-2,2, col ='red')
```