Aula 4 - Visualização de dados

Vitor Rios

11 de novembro de 2017

Visualizar seus dados pode ser a parte mais importante da sua análise

Fazer gráficos no R pode dar (muito) trabalho, mas os resultados valem a pena

Gráficos no R são baseados em dispositivos. O dipositivo padrão é a tela, mas podem ser arquivos, cada tipo de arquivo gráfico sendo um dispositivo diferente

No RStudio, gráficos são plotados na aba plots, mas podem ser abertos em janelas separadas, usando a função `x11()` (Linux e Windows), `windows()` (Windows somente) ou `quartz()` (Macintosh)

Os tipos padrão de gráficos disponíveis são `postscript()` para arquivo postscript, extensão `.ps` ou `.eps`
`pdf()` salva o gráfico num arquivo `.pdf`
`pictex()` salva o gráfico no formato LaTeX/PicTeX
`xfig()` salva o gráfico no formato `.XFIG`
`bmp()`, `bitmap()` salva um arquivo `.bitmap` (precisa de GhostScript instalado).
`X11()` plota numa janela via sistema gráfico X11
`png()` salva o gráfico no formato `.PNG` `tiff()` salva o gráfico no formato `.tiff`
`jpeg()` salva o gráfico no formato `.jpg` `svg()` salva o gráfico no formato `.svg`

Gravando o gráfico num arquivo

```
# Chame o tipo de arquivo desejado, passando informações de tamanho, resolu
png("SocialNetworkExample.png",width = 900,height = 900)

#plote seu gráfico normalmente, ele será escrito no arquivo e não irá apare
plot(graph.subset,
     layout= layout_nicely(graph=graph.subset),
     vertex.label=NA,
     vertex.color=colors[membership(graph.subset.modulos)],
     vertex.size = 5,
     )
#feche o dispositivo, usando dev.off(), senão o arquivo ficará corrompido
dev.off()
```

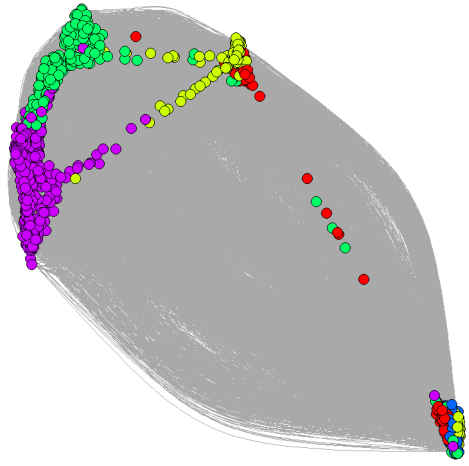
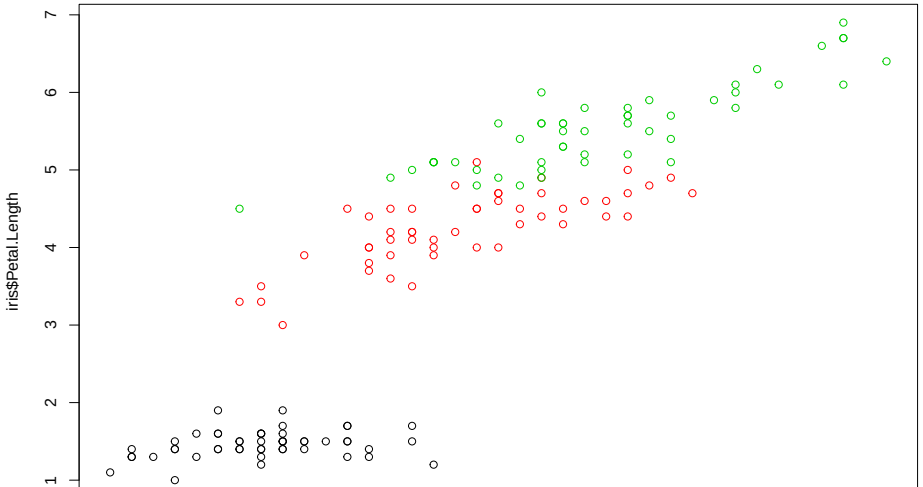


Figure 1: Grafo

Função básica e poderosa (e feia): `plot()`

```
plot(x= iris$Sepal.Length, y=iris$Petal.Length,col=iris$Species)
```



A função `plot()` tem dezenas de opções para controlar cada detalhe mínimo do gráfico

Principais: - x - y - type - main - xlab - ylab - xlim - ylim xaxt bty pch cex

Parâmetro	Controla	valores
x	valores a serem plotados em x	
y	valores a serem plotados em x	
type	tipo de gráfico (ponto, linhas, etc)	“p”,“l”,“b”...
lty	tipo de linha (pontilhada, tracejada, etc)	“blank”, “solid”, “dashed”, “dotted”, “dotdash”, “longdash”

Parâmetro	Controla	valores
main	título do gráfico	“texto entre aspas”
xlab	rótulos do eixo x	“texto entre aspas”
ylab	rótulos do eixo y	“texto entre aspas”
cex	tamanho das letras, em proporção	cex = 1 -> 100%, cex = 0.5 ->50%, etc
cex.axis, cex.lab, cex.main	eixos, rotulos dos eixos, título	cex.axis = 1 -> 100%, cex.axis = 0.5 ->50%, etc
bty	linhas de contorno do gráfico	“o”, “l”, “7”, “c”, “u”, “]”, “n”

Função par() e função axis()

`par()` estabelece os parâmetros para todos os gráficos que se seguem a ela exceto parâmetros de intervalo de dados a serem plotado. Usada para definir gráficos com vários painéis, margens e área do gráfico. Usada antes de qualquer `plot()`

`axis()` define todos os parâmetros para os eixos, usada depois de `plot(dados, xaxt = "n")` (para eixo x) ou `plot(dados, yaxt = "n")` (para eixo y)

```
par(mfrow=c(1,2))       #linhas, colunas

plot(x=iris$Sepal.Length, y=iris$Petal.Length,          #primeiro gráfico
     col = iris$Species,
     main = "Sepal.Length x Petal.Length")

plot(x=iris$Sepal.Length, y=iris$Sepal.Width,           #segundo gráfico
     col = iris$Species,
     main = "Sepal.Length x Sepal.Width in Iris")
```

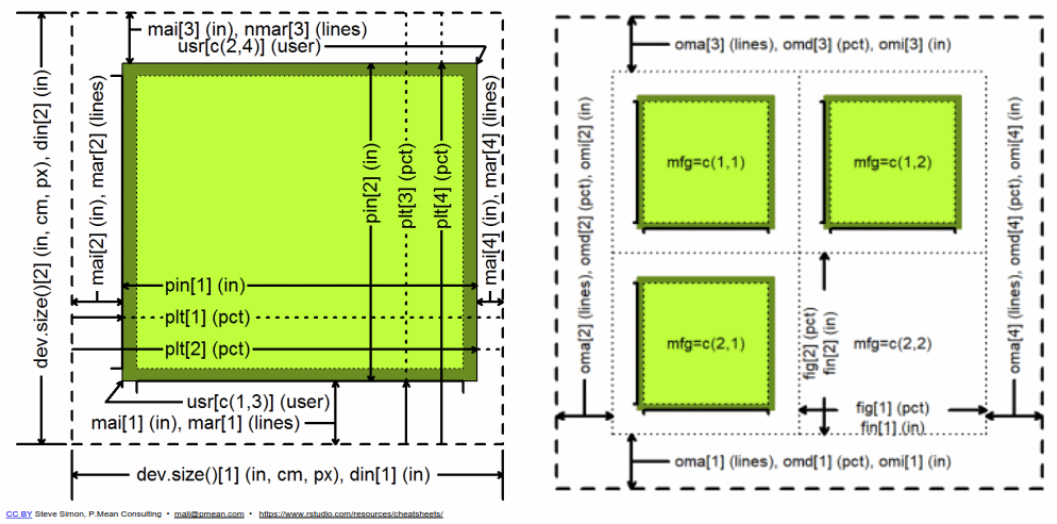


Figure 2: fonte:https://www.rstudio.com/resources/cheatsheets/

Funções que adicionam elemento a gráficos

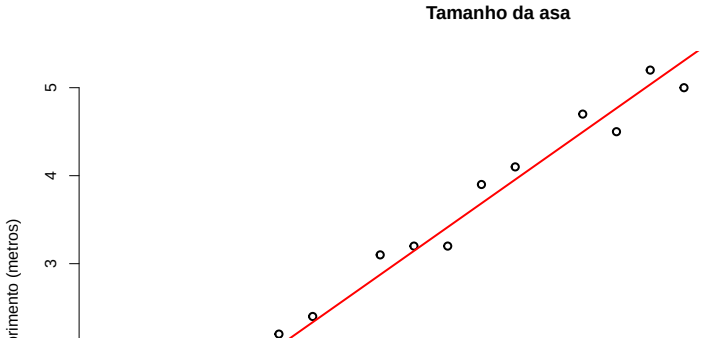
`abline()` adiciona uma linha, definida por intercepto e inclinação, ou por valores de posição horizontal (h) e vertical (v)

`segments((x0, y0, x1, y1)` adiciona um segmento indo de (x0,y0) até (x1,y1)

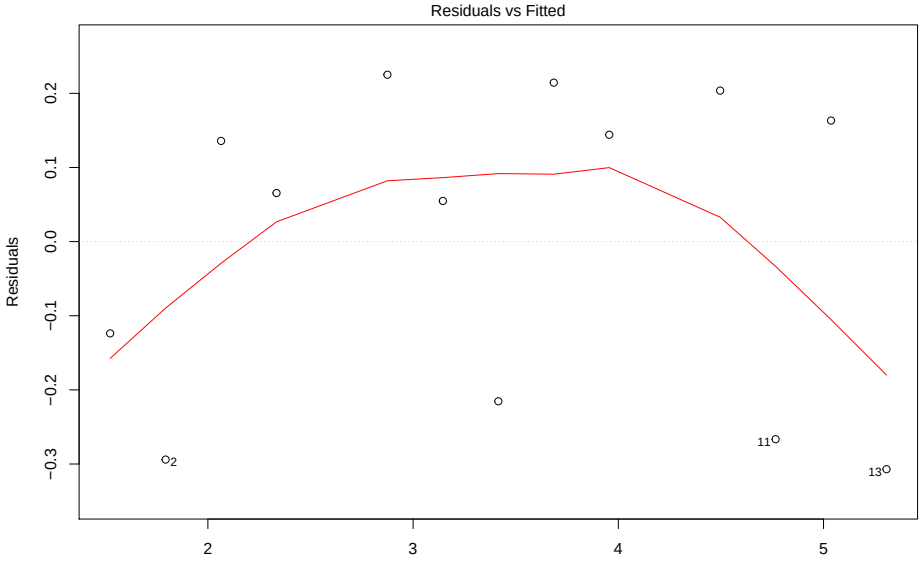
`arrows(x0, y0, x1 , y1 , length, angle = 30)` adiciona uma seta a um gráfico. `lengthé` o comprimento da seta e `angle` é o angulo da ponta da seta

`points(x, y, pch)` adiciona pontos nas coordenadas x e y. `pch` é o tipo de simbolo utilizado

```r
plot(TamanhoDaAsa ~ Idade, xlab = "Idade (décadas)",
     ylab = "Comprimento (metros)",
     main = "Tamanho da asa"
     , bty = "n"
     , xlim = c(0, 23), ylim = c(0, max(TamanhoDaAsa))
     , lwd = 2
)
abline(modelo, col = "red", lwd = 2)
```



```
plot(modelo)
```

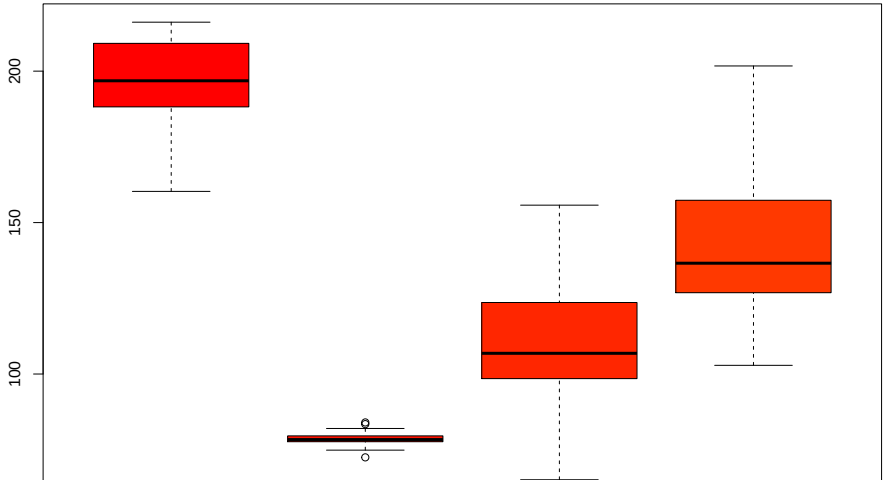


Para plotar médias com barras de desvio padrão

```
m1=mean(dragoes$y[dragoes$x=="aventureiros"]);m2=mean(dragoes$y[dragoes$x=
m3=mean(dragoes$y[dragoes$x=="vacas"]);m4=mean(dragoes$y[dragoes$x=="virge
s1=sd(dragoes$y[dragoes$x=="aventureiros"]);s2=sd(dragoes$y[dragoes$x=="fa
s3=sd(dragoes$y[dragoes$x=="vacas"]);s4=sd(dragoes$y[dragoes$x=="virgens"]
avg=c(m1,m2,m3,m4)
sdev=c(s1,s2,s3,s4)
plot(x= 1:4, y= avg,cex=1.5,pch=16, col=1:4,ylim=range(c(avg-sdev, avg+sde
arrows(x0=1:4, y0=avg-sdev, x1=1:4, y1=avg+sdev, length=0.05, angle=90, co
abline(h=mean(avg), col="red")#media total
```

Boxplot

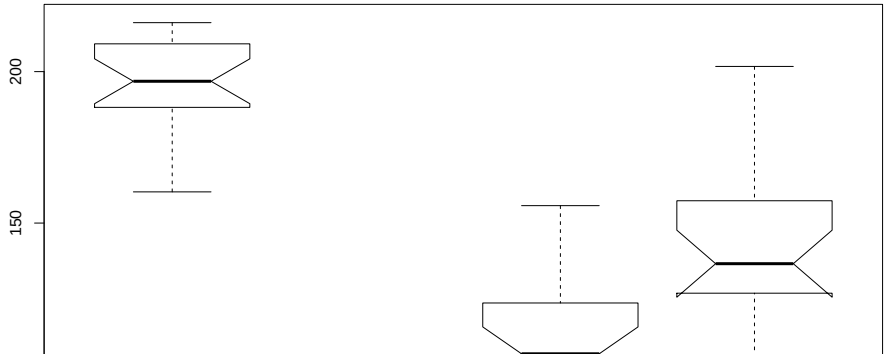
```
boxplot(dragoes$y ~ dragoes$x, col=rainbow(length(dragoes$x)))
```



A função boxplot permite comparar os intervalos de confiança 95% das medianas: se os entalhes não se sobrepuserem, há forte evidência de que as medianas são diferentes

```
boxplot(dragoes$y ~ dragoes$x, notch=T)
```

```
## Warning in bxp(structure(list(stats = structure(c(160.2968, 188.2027,
## 196.819, : some notches went outside hinges ('box'): maybe set notch=FA
```



Função aggregate()

A função summary nos dá muitas informações sobre a estrutura dos nossos dados, mas tudo. Em particular, desvio padrão, erro padrão da média, curtose, e assimetria devem ser calculados à mão, principalmente se quisermos essas estatísticas agrupadas por níveis de um fator. Para isso usamos a função aggregate()

`aggregate(x, by, FUN, ...)`
x é a coluna do dataframe que ser quer agrupar
by é a coluna que contém os níveis dos grupos que se quer usar
FUN é a função que se quer aplicar nos grupos, sem parênteses
... são os argumentos que serão passados para a função FUN

```
media.por.comida = aggregate(x=dragoes$y,by = list(dragoes$x), FUN = mean,
media.por.comida
```

```
##        Group.1        x
## 1 aventureiros 195.3093
## 2  fazendeiros  78.6809
## 3        vacas 110.3509
```

```
files_to_read=list.files(pattern = reg.exp)
if (length(files_to_read)!=0)
{
  for (r in 1:(length(files_to_read)))
  {temp_valores_replicas[,r] <- read.table(files_to_read[r],  header = FALS
  graphtype=NA#creating plot names
  if(type=="g"){graphtype=" with general recognition "}
  if (type=="i") {graphtype=" with Individual Recognition "}
  if (type=="nr") {graphtype=" with No Recognition "}
  plotname =paste(metrica,graphtype, ", mundo = ", world_size, ", mem leng
  if("plots"%in%dir()==FALSE) dir.create("plots") #se não existir, cria di
  #C:/Users/Vitor/Desktop/100replicas/resultados
  filename = file.path("C:", "Users", "Vitor","Desktop", "100replicas", "r

  tiff(filename ,width = 7.5 ,height = 5,units = "in"
       ,res=300, compression = "lzw",type="cairo", antialias = "default"
       )
  y=max(temp_valores_replicas,na.rm = TRUE)+1
  par(mar=c(4.5,1.2)+0.1)
```