- Tutorial
- Tutorial II
- Exercícios
- Apostila

5. Tutoriais para Criação e Edição de Gráficos no R

Orientações Gerais

Aqui são apenas dados os códigos e funções, para você executar e entender como funcionam. Para entender as funções, leia antes o capítulo correspondente na apostila. Experimente também mudar mais argumentos e funções além das apresentadas nos exercícios.

Criando Gráficos

Há duas maneiras de se especificar as variáveis em gráficos:

- Cartesiana: plot(x,y)
- Fórmula estatística: plot(y~x)

Experimente as duas para criar alguns gráficos simples:

```
riqueza <- c(15,18,22,24,25,30,31,34,37,39,41,45)
area <- c(2,4.5,6,10,30,34,50,56,60,77.5,80,85)
area.cate <- rep(c("pequeno", "grande"), each=6)

plot(riqueza~area)
plot(area,riqueza) # o mesmo que o anterior
boxplot(riqueza~area.cate)
```

Editando Gráficos

Parâmetros Globais e locais

Experimente o comando `par` para modificar alguns parâmetros gráficos. Por exemplo, coloque as legendas dos eixos na vertical com:

```
par(las=1)
plot(riqueza~area)
```

Para aumentar o tamanho dos elementos do gráfico, modifique o parâmetro `cex`:

```
par(cex=2)
plot(riqueza~area)
```

Em ambos casos, os parâmetro gráficos foram modificados no dispositivo gráfico aberto, ou seja, ficaram retidos como padrão até a modificação do `par` ou fechamento do dispositivo.

```


boxplot(riqueza~area.cate)
```

O fechamento dos dispositivos pode ser feito de várias maneiras, fechando a janela gráfica com o mouse, fechando o dispositivo com a função `dev.off()` ou fechando todos os dispositivos abertos com a função `graphics.off()` . Abaixo, fechamos todos os dispositivos e refazemos o gráfico com o padrão básico do R.

```
graphics.off()
plot(riqueza~area)
```

Alguns parâmetros gráficos só podem ser modificados através dos parâmetros do dispositivo usando o `par()` , outros, podem ser modificados também nas funções de alto nível, aquelas que abrem o dispositivo e iniciam o gráfico como `plot(), hist(), boxplot(), image()`

```
plot(riqueza~area, cex = 2, las = 1)
```

Note que pode haver diferenças entre modificar os parâmetros gráficos globalmente ou na função de alto nível. Nesse caso, o `cex` só modificou o tamanho do símbolo, enquanto nos parâmetros globais modificou todos os elementos do gráfico (legenda, eixos, títulos, ...). Nesse último caso, o dispositivo não leva para o próximo gráfico os parâmetros modificados.

```


boxplot(riqueza~area.cate)
```

Alguns outros parâmetros funcionam com alguns funções de alto nível e não com outras. Portanto, deve estar atento para quais parâmetros pode ser modificados localmente, veja o exemplo abaixo:

```


plot(riqueza~area)
plot(riqueza~area, bty="l", tcl=0.3)
```

Percebeu o que mudou?

Agora tente:

```
boxplot(riqueza~area.cate, bty="l", tcl=0.3)
```

O que aconteceu?

E agora?

```
par(bty="l")
par(tcl=0.3)
boxplot(riqueza~area.cate)
```

Gráficos Multiplos e Margens

O parâmetro `mfrow` permite que o dispositivo gráfico seja repartido em painéis que recebem diferentes gráficos. Esse é um exemplo de parâmetro que só pode ser modificado pela função `par()`. Por exemplo, para colocar dois gráficos em um mesmo dispositivo de tela, usamos:

```
par(mfrow=c(2,1))
plot(riqueza~area)
boxplot(riqueza~area.cate)


par(mfrow=c(1,2))
plot(riqueza~area)
boxplot(riqueza~area.cate)
```

Que pode ser combinado com `mar` para mudar o tamanho das margens também:

```
par(mfrow=c(2,1), mar=c(4,14,2,6))
plot(riqueza~area)
boxplot(riqueza~area.cate)


par(mfrow=c(1,2))
par(mar=c(14,4,8,2))
plot(riqueza~area)
boxplot(riqueza~area.cate)

par(mfrow=c(1,2))
par(mar=c(8,4,8,1))
plot(riqueza~area)
par(mar=c(14,2,4,0.5))
boxplot(riqueza~area.cate)
```

Inserindo mais Informações em Gráficos

Dentre as várias funções existentes para se inserir informações em gráficos, existem sete que são bastante úteis.

Usando as variáveis:

```
riqueza <- c(15,18,22,24,25,30,31,34,37,39,41,45)
area <- c(2,4.5,6,10,30,34,50,56,60,77.5,80,85)
abundancia <- rev(riqueza)
```

Crie gráficos inserindo os parâmetros abaixo.

lines()

Para inserir linhas retas ou curvas não-paramétricas (como `lowess`, `loess`, `gam`, etc).

```
 plot(riqueza~area)
 lines(lowess(area, riqueza))
```

Um exemplo já conhecido: sobrepondo uma curva de densidade probabilística empírica a um histograma:

```
##Duracao em minutos  das erupções do Faithful Geiser, Yellowstone
## Objeto faithful do pacote datasets

hist(faithful$eruptions,prob=TRUE)
lines(density(faithful$eruptions),col="blue")
```

abline()

Insira uma linha com intercepto e inclinação dados por números com

```
##100 numeros de uma distribuicao uniforme entre 1 e 10
x <- runif(100, 1, 10)
## Os mesmos numeros somados a um ruído normal de media zero
## e desvio-padrao um:
y <- x+rnorm(100)
##Scatterplot e linha teorica esperada
plot(y~x)
abline(0,1, col="blue")
```

Agora veja o que acontece se o argumento da função `abline` é o objeto resultante do ajuste de uma regressão linear simples, obtido com a função `lm`[1)]:

```
modelo <-  lm(riqueza~area)
plot(riqueza~area)
abline(modelo)
```

E há ainda os argumentos h e v para linhas horizontais e verticais. Aqui traçamos as linhas que passam pelas médias de cada variável em um gráfico de dispersão.

```
plot(riqueza~area)
abline(v=mean(area))
```

```
abline(h=mean(riqueza))
```

Você sabia? A reta da regressão linear simples sempre passa pelo ponto que é a interseção destas duas linhas.

text()

Use esta função para inserir texto dentro do gráfico. O texto pode ser uma letra, um símbolo (muito usado para mostrar diferenciar classes no gráfico), uma palavra ou até mesmo uma frase:

```
plot(riqueza~area)
text(x=20,y=40,"texto")
```

Você pode usar esta função para identificar identificar pontos em seu gráfico, ou plotar rótulos ao invés de pontos:

```
require(MASS)# para objeto Animals
head(Animals)# consulte o help para entender
plot(brain~body, data=Animals, log="xy", type="n")
text(y=Animals$brain,x=Animals$body,labels=rownames(Animals))
```

mtext()

Esta função acrescenta texto nas margens do gráfico ou da janela gráfica. Consulte a página de ajuda para entender seus argumentos

```
plot(riqueza~area)
mtext("legenda no lado errado", side=4, line=0.9, at=20,cex=2,
family="serif")
```

par(new=TRUE)

Comando para sobrepor um novo gráfico a um gráfico já existente. Execute-o e compare com o obtido com o comando `par(mfrow=c(2,2))`

```
plot(riqueza~area)
par(new=TRUE)
plot(abundancia~area)
```

axis()

Para se inserir um eixo novo. Esta função é bastante usada nos casos em que se deseja ter dois gráfico dentro de uma mesma figura (ver `par(new=TRUE)`), ou então se deseja controlar muitos dos parâmetros dos eixos (como em `mtext()`).

```
plot(riqueza~area)
par(new=TRUE)
plot(abundancia~area, axes=FALSE, ann=FALSE) ## começa com gráfico sem eixos
axis(4) ## adiciona eixo
```

arrows(), rect(), polygon()

Descubra o efeito destas funções, por exemplo:

```
plot(riqueza~area)
rect(20,20,40,40)
```

Salvando Gráficos

Abra um dispositivo *jpg* para armazenar um gráfico, com a função `jpeg`, execute os comandos para criar o gráfico e fecehe o dispositivo:

```
jpeg(filename = "Rplotaula.jpg", width = 480, height = 480,
     units = "px", pointsize = 12, quality = 100,
     bg = "white",  res = NA)

par(mfrow=c(1,2))
par(mar=c(14,4,8,2))
plot(riqueza~area)
boxplot(riqueza~area.cate)

dev.off() ## fecha o dispositivo jpg
```

Verifique em seu diretório de trabalho se há agora uma figura *jpg* com o nome “Rplotaula.jpg”.

Agora abra um dispositivo gráfico cria arquivos com numeração sequencial, envie para ele dois gráficos e feche o dispositivo:

```
png("meugrafico%02d.png")
plot(riqueza~area)
boxplot(riqueza~area.cate)
dev.off()
```

Qual a diferença no resultado deste comando e do anterior?

Manipulação de dispositivos gráficos

Feche toda as janelas de gráficos do R e execute um comando `plot`, o que irá abrir um dispositivo gráfico de tela:

```
plot(riqueza~area)
```

Agora abra um novo dispositivo gráfico de tela:

```
> x11()
```

Uma nova janela gráfica em branco se abrirá, na margem superior da qual você verá a indicação de que agora este é o dispositivo ativo, *e.g.*:

Verifique que agora o dispositivo ativo é o de número 3, e ative o dispositivo 2:

```
dev.cur()
dev.set(which=2)
```

Agora um dispositivo gráfico png e outro pdf:

```
png("figura%02d.png")
pdf("figura%02d.pdf")
```

Quantos dispositivos estão abertos? Obtenha a lista com

```
dev.list()
```

Mas apenas umestá ativo, isto é, recebe a saída dos comandos gráficos. Descubra qual é com

```
dev.cur()
```

E altere para o dispositivo 3 com

```
dev.set(3)
dev.cur() #verificando
```

Agora experimente fechar dispositivos com o comando `dev.off`:

```
dev.list()
dev.cur()
dev.off() # quem foi fechado?
dev.list() # verificando
dev.cur() # e quem é o ativo agora?
```

Próximos passos

Para mais informações sobre a edição de gráficos siga para a 05a-graficos. Nesse wiki focamos no uso das ferramentas básicas do R e nesse tutorial no pacote `graphics` carregado por padrão na sessão do R. O pacote para elaboração de gráficos chamado `ggplot2` vem se tornando muito popular nos últimos anos, mas apresenta uma sintaxe muito diferente da usual no R. Por essa razão preferimos

deixá-lo de fora do nosso material. Existem muitos bons tutoriais sobre o `ggplot2`, inclusive um ótimo feito pelo colaborador da disciplina Gustavo Burin Ferreira, caso tenha interesse acesse:

- https://blog.gburin.com/tutorial-de-ggplot2

[1)]

detalhes na unidade sobre modelos lineares