

T4 – Visualizacion de imagenes, melhoria de contraste

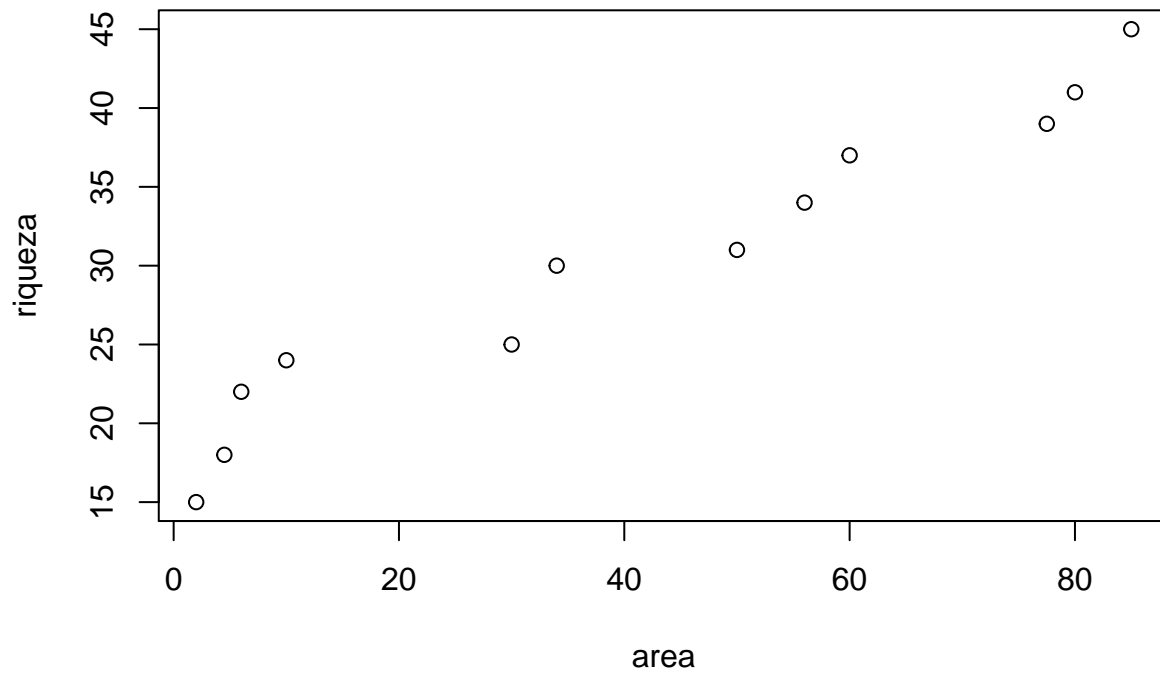
Criando Gráficos

Há duas maneiras de se especificar as variáveis em gráficos:

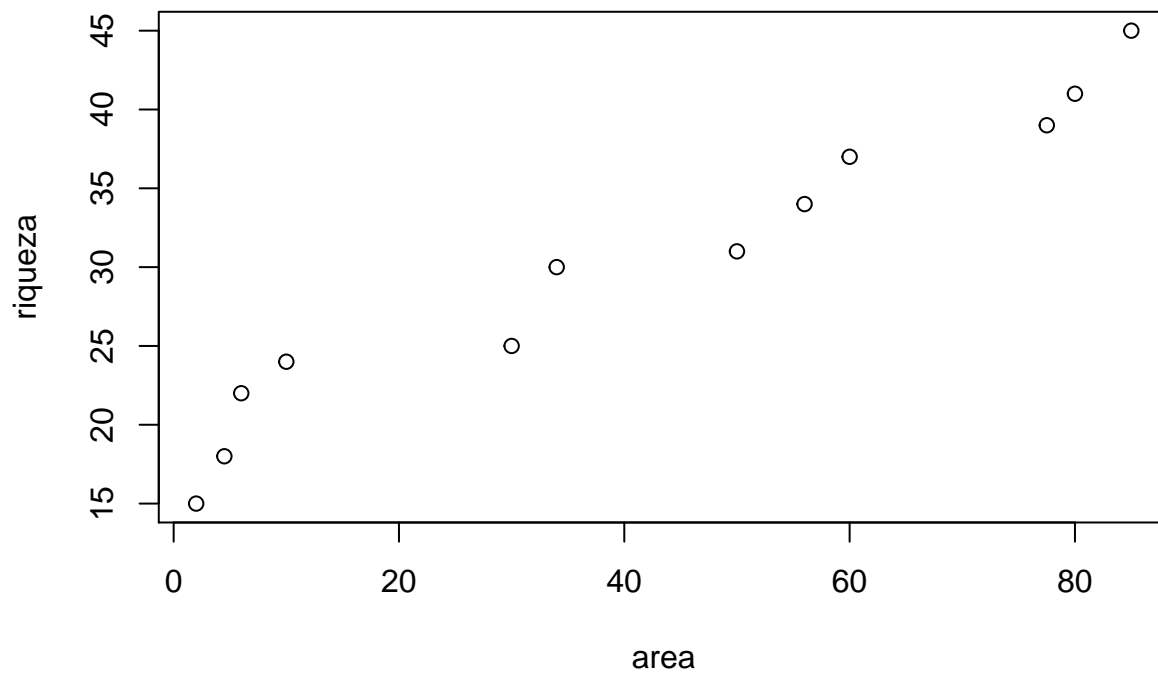
Cartesiana: `plot(x,y)` Fórmula estatística: `plot(y~x)` Experimente as duas para criar alguns gráficos simples:

```
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)
```

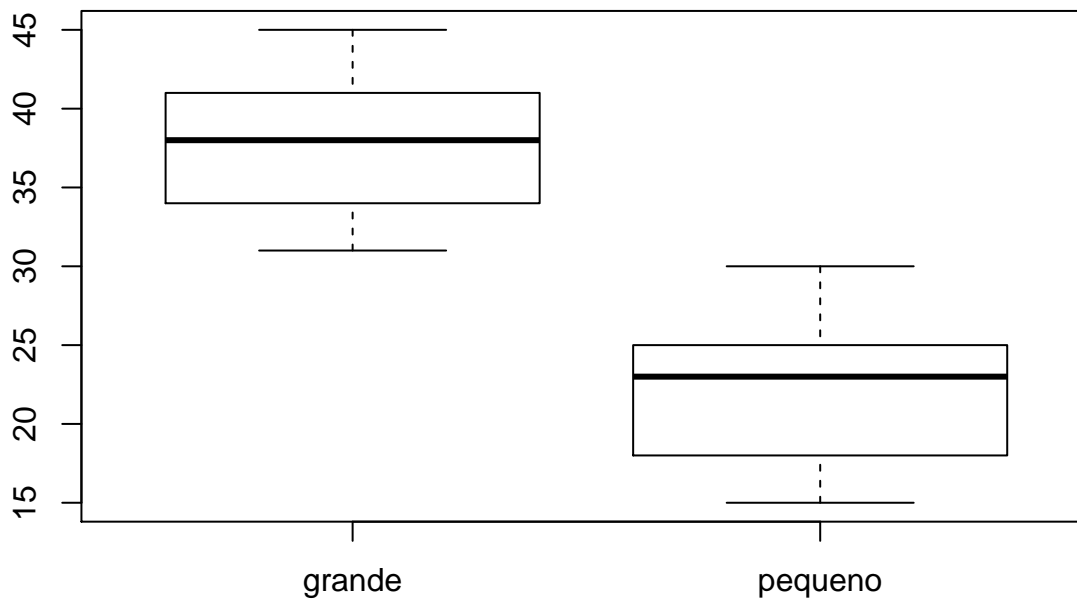
```
plot(riqueza~area)
```



```
plot(area,riqueza) # o mesmo que o anterior
```



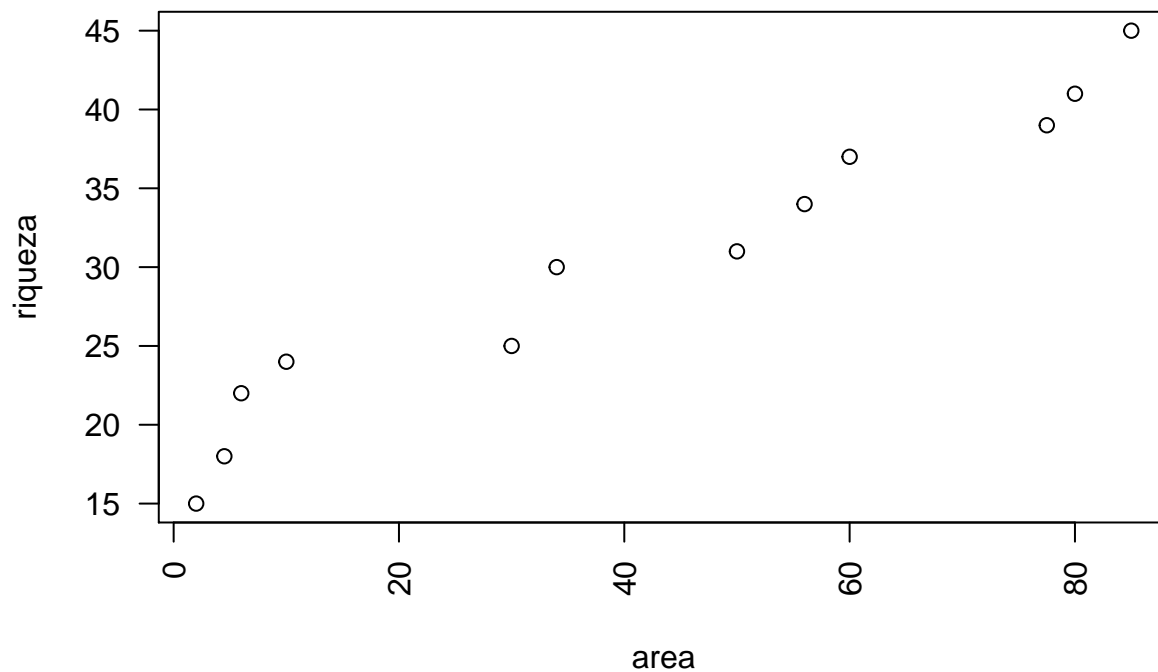
```
boxplot(riqueza~area.cate)
```



Editando Gráficos

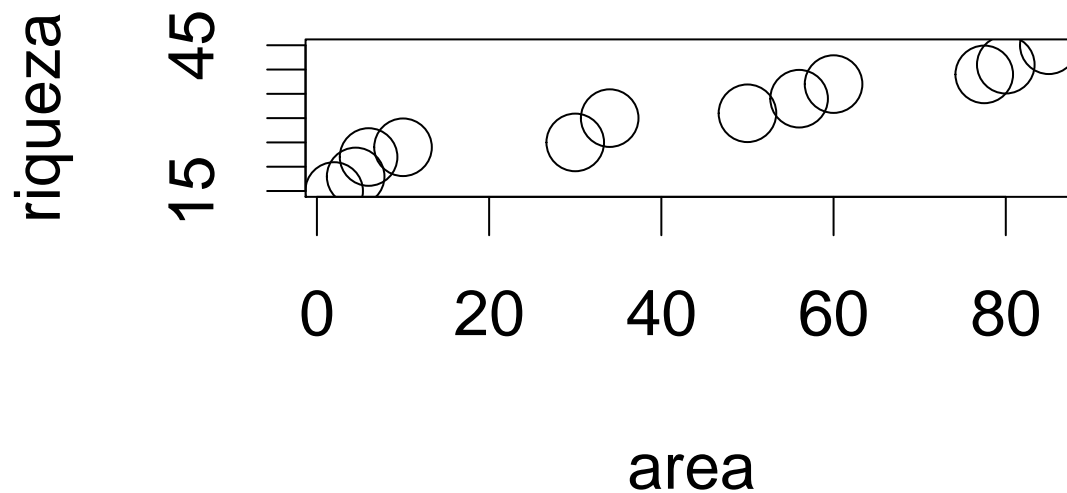
Experimente o comando `par` para modificar alguns parâmetros gráficos. Por exemplo, coloque as legendas dos eixos na vertical com:

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



Outro caso que é importante saber é a função cex. Veja o resultado do comando:

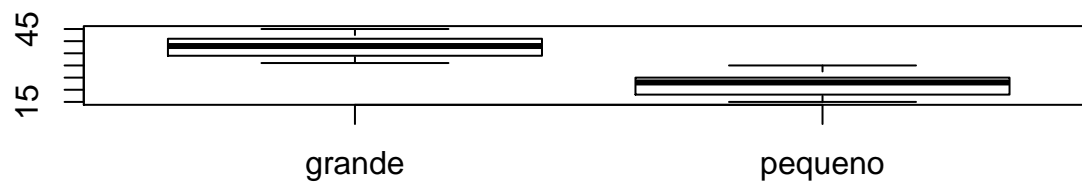
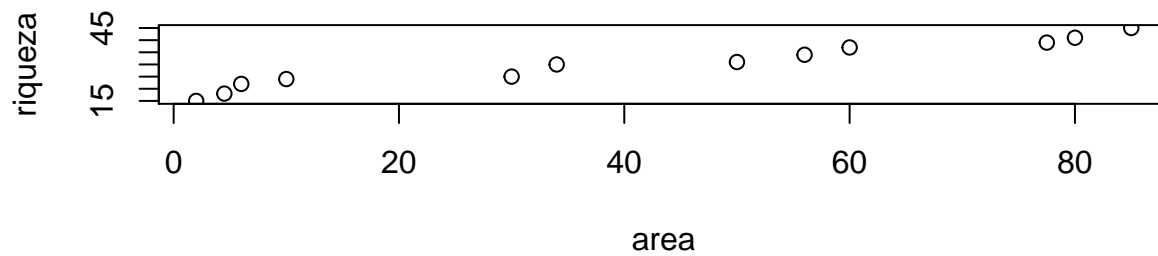
```
par(cex=2)
plot(riqueza~area, cex=2)
```



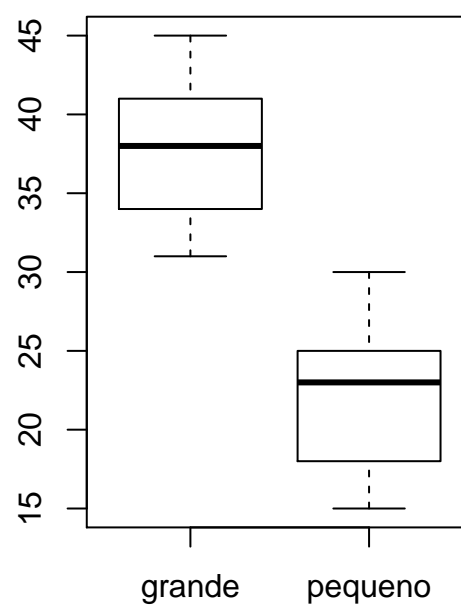
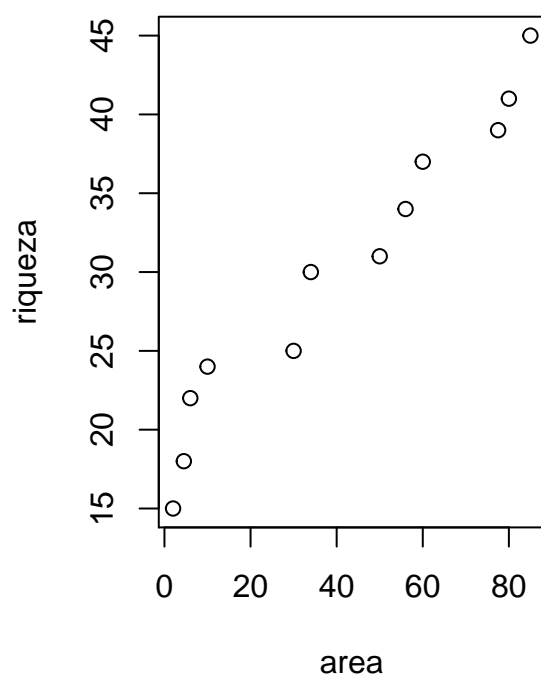
Gráficos Múltiplos e Margens

Use `par(mfrow=c())` para construir figuras com mais e um gráfico

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



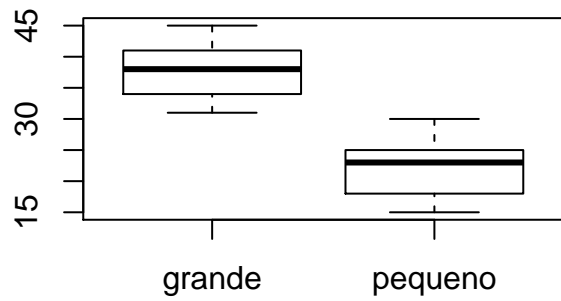
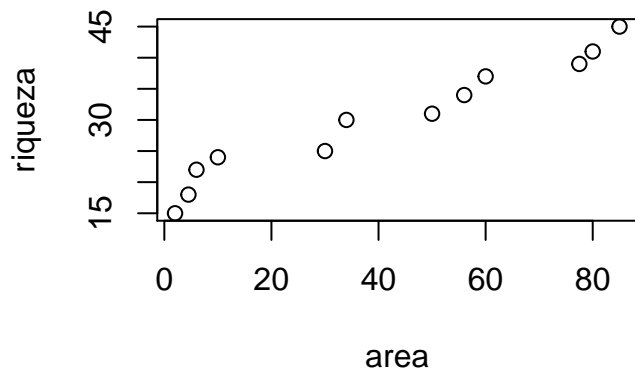
```
par(mfrow=c(1,2))
plot(riqueza~area)
boxplot(riqueza~area.cate)
```



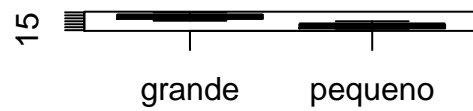
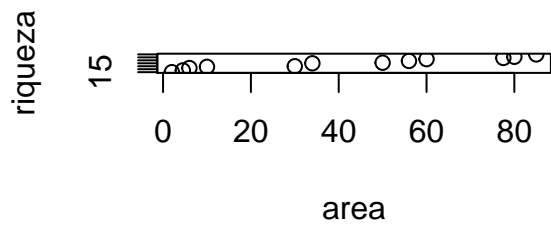
bine com 'par(mar=c()) para mudar as margens:

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

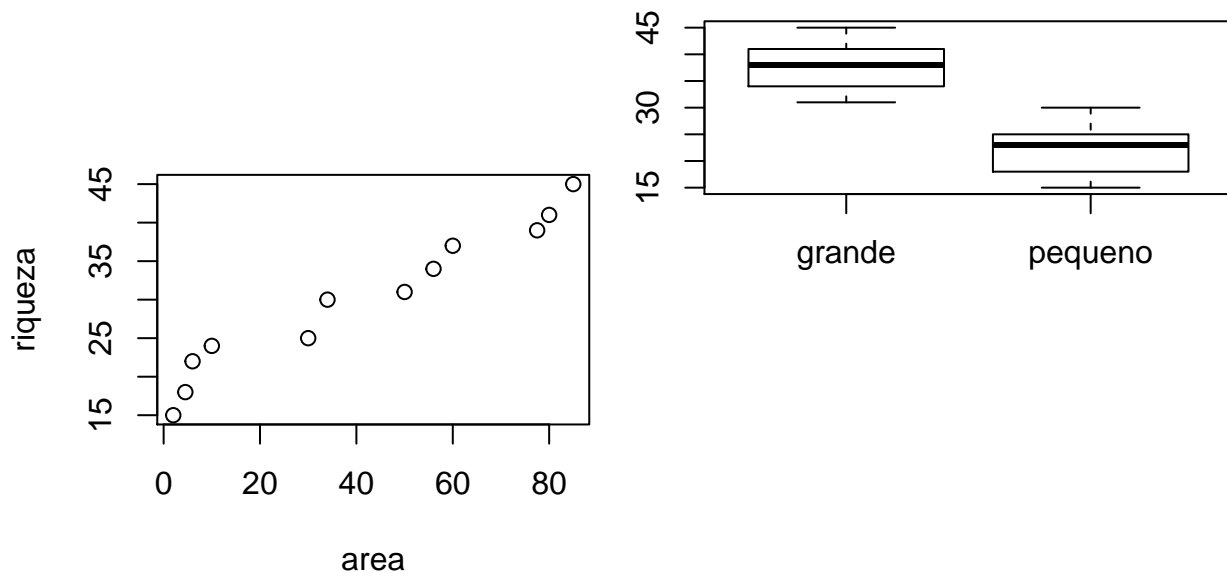
E com-



```
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)
```



Diferenças Entre Tipos De Gráfico

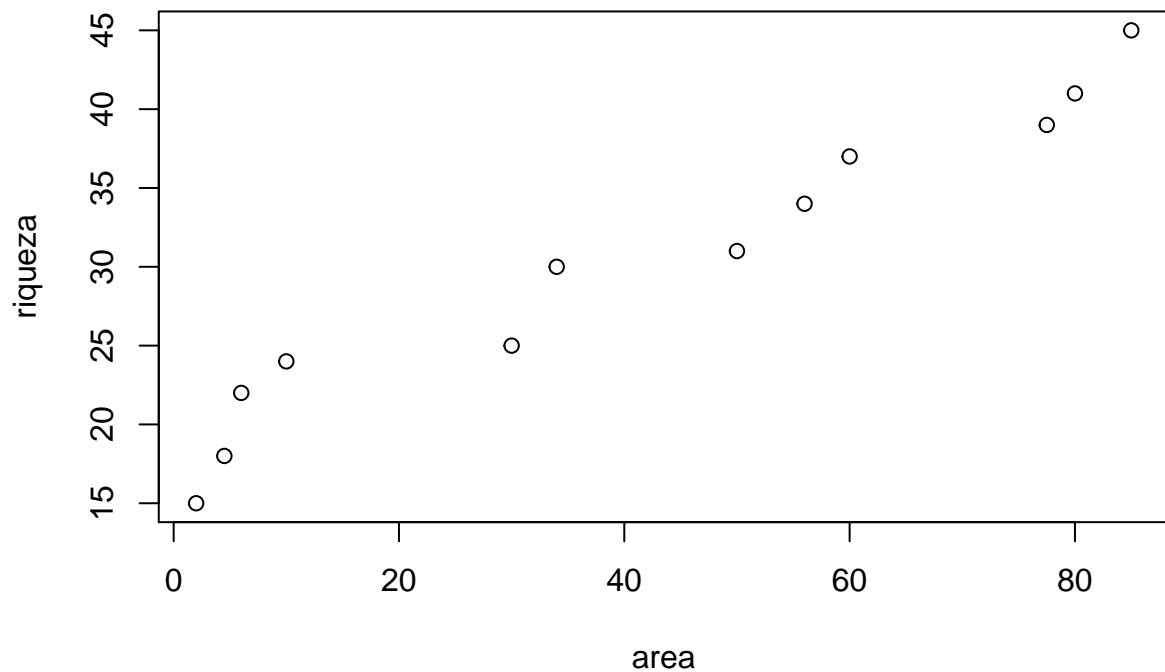
Alguns parâmetros gráficos são argumentos das funções gráficas, e outros só podem ser alterados com a função `par`, aplicada antes de iniciar o gráfico. Há ainda os que podem ser alterados das duas maneiras, veja abaixo alguns exemplos:

Com 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)
area.cate <- rep(c("pequeno", "grande"), each=6)
```

Crie:

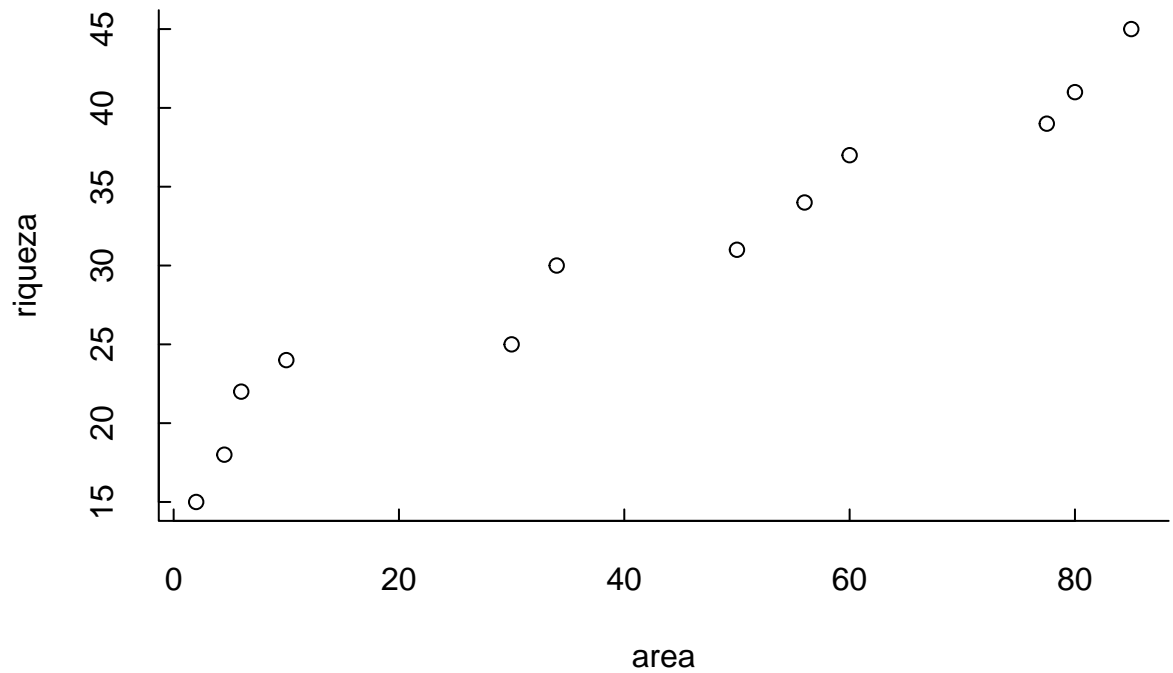
```
plot(riqueza~area)
```



E

agora:

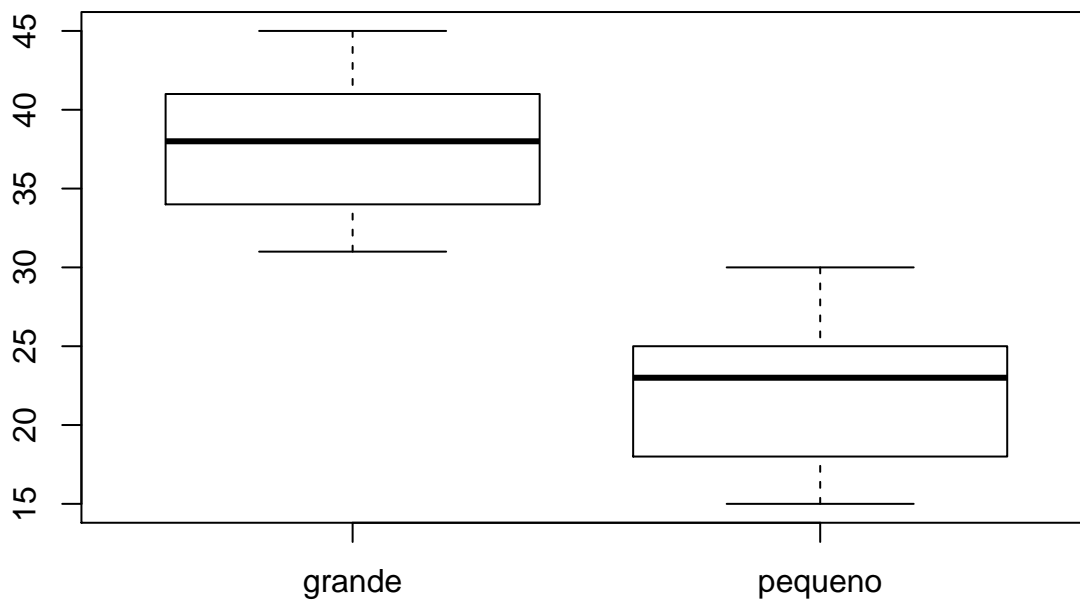
```
plot(riqueza~area, bty="l", tcl=0.3)
```



beu o que mudou?

Agora tente:

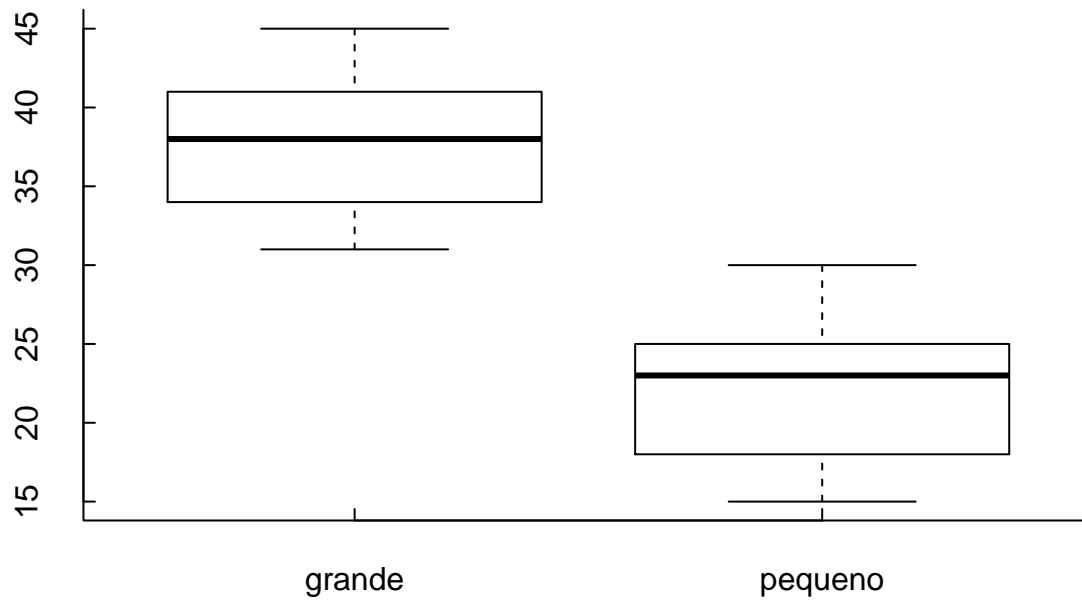
```
boxplot(riqueza~area.cate, bty="l", tcl=0.3)
```



ceu?

E agora?

```
par(bty="l")  
par(tcl=0.3)  
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:

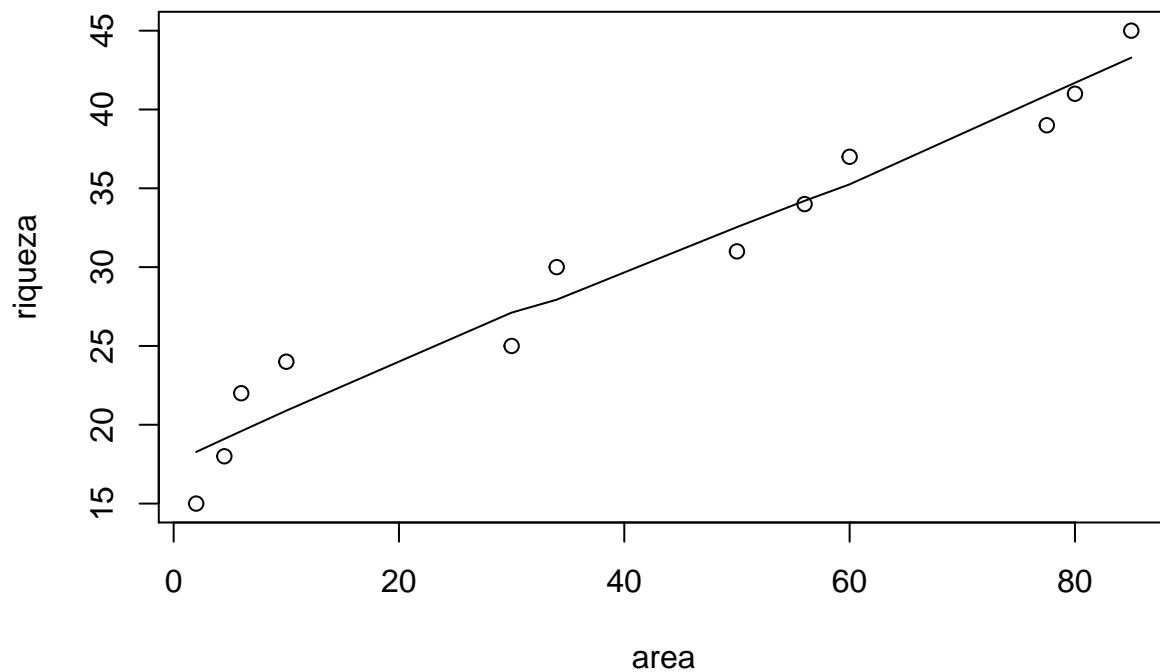
```
riqueza <- c(15,18,22,24,25,30,31,34,37,39,41,45)
area <- c(2,4.5,6,10,30,34,50,56,60,77.5,80,85)
abundancia <- rev(riqueza)
```

Crie gráficos inserindo os parâmetros abaixo.

lines()

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

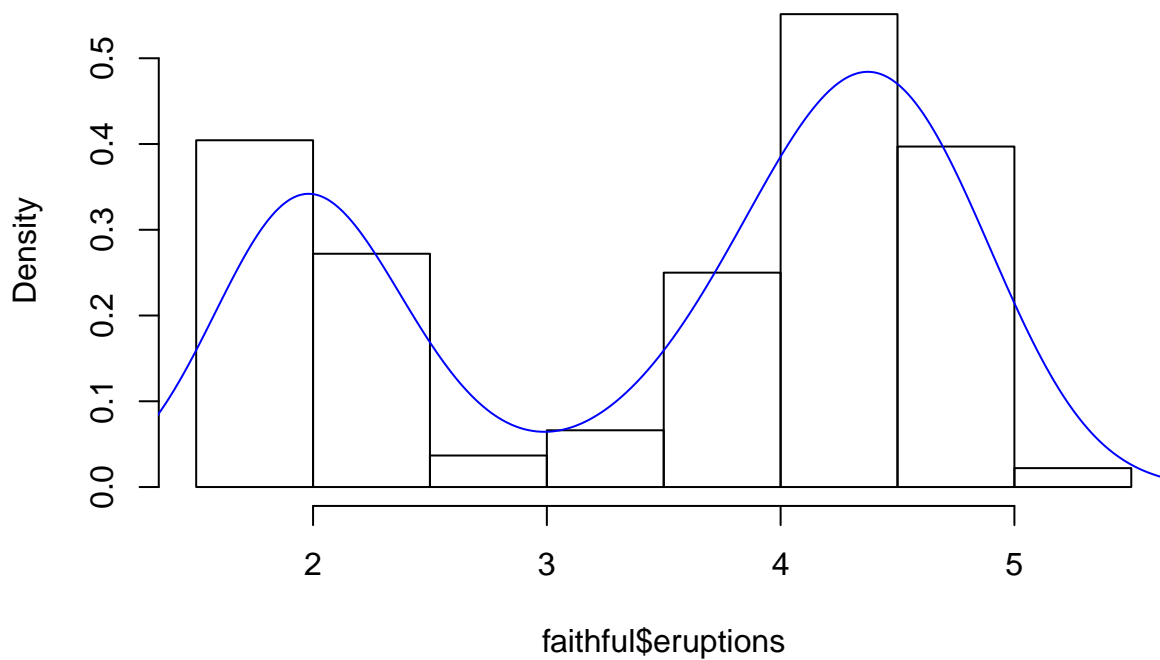
```
plot(riqueza~area)
lines(lowess(area, riqueza))
```

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

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

Histogram of faithful\$eruptions



Insira uma linha com intercepto e inclinação dados por números com 100 numeros de uma distribuicao uniforme entre 1 e 10

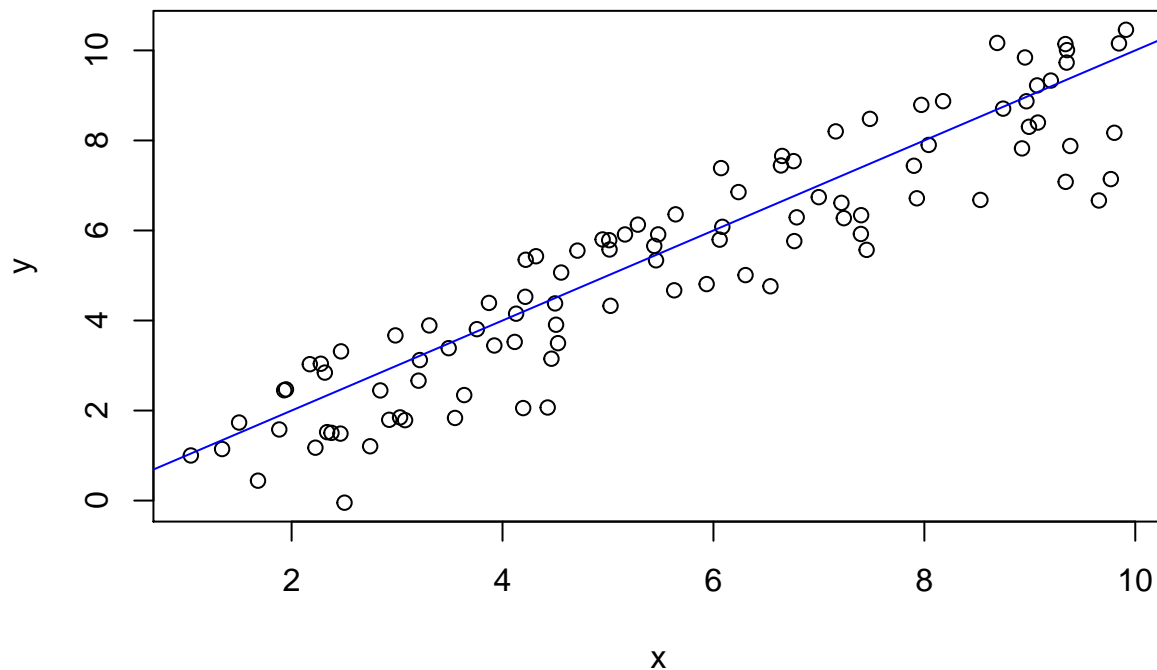
```
x <- runif(100, 1, 10)
```

Os mesmos numeros somados a um ruído normal de media zero e desvio-padrao um:

```
y <- x+rnorm(100)
```

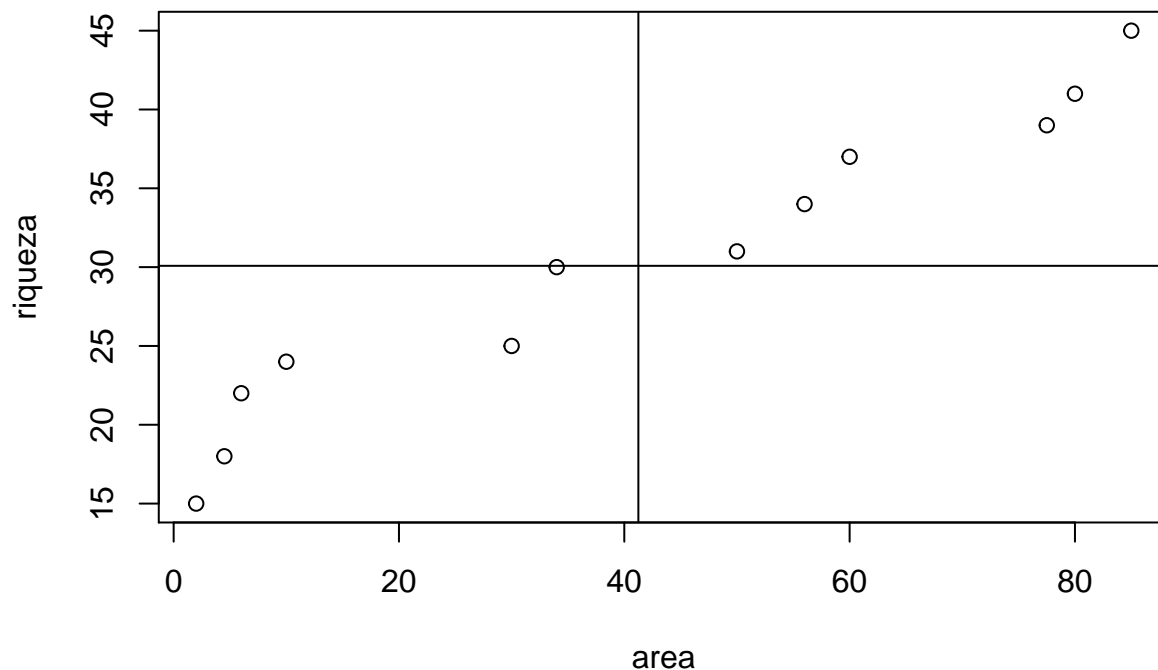
Scatterplot e linha teorica esperada

```
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 lm1): ‘{r} 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))
```

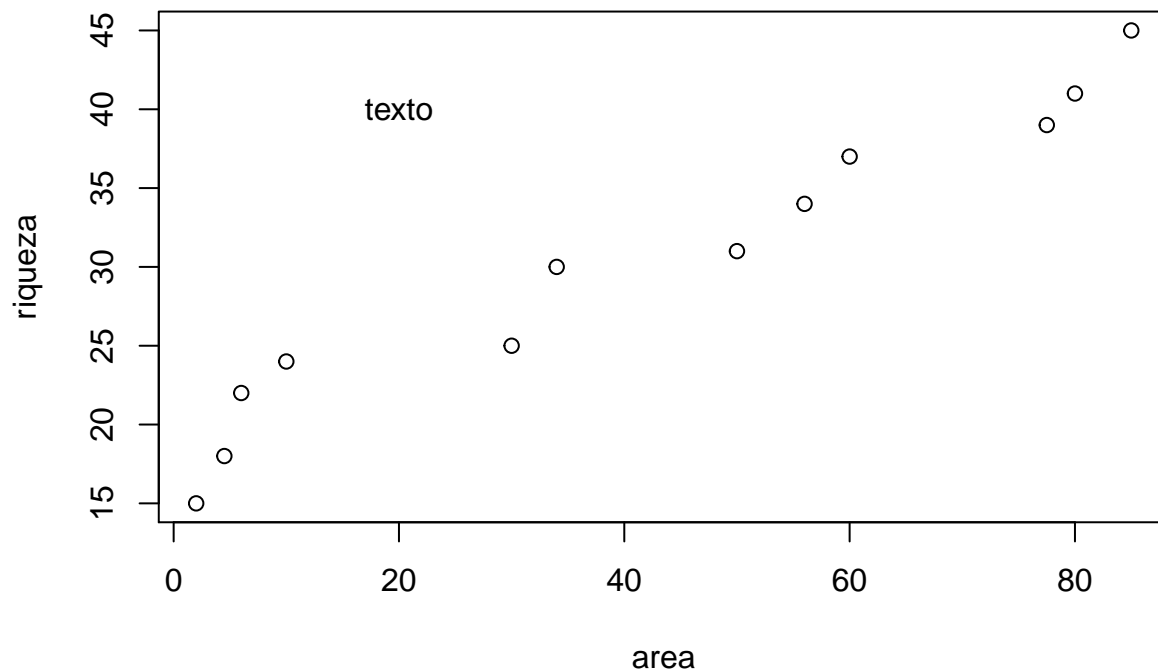


Você sabia? A reta da regressão linear simples sempre passa pelo ponto que é a interseção destas duas linhas.

`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 pontos em seu gráfico, ou plotar rótulos ao invés de pontos:

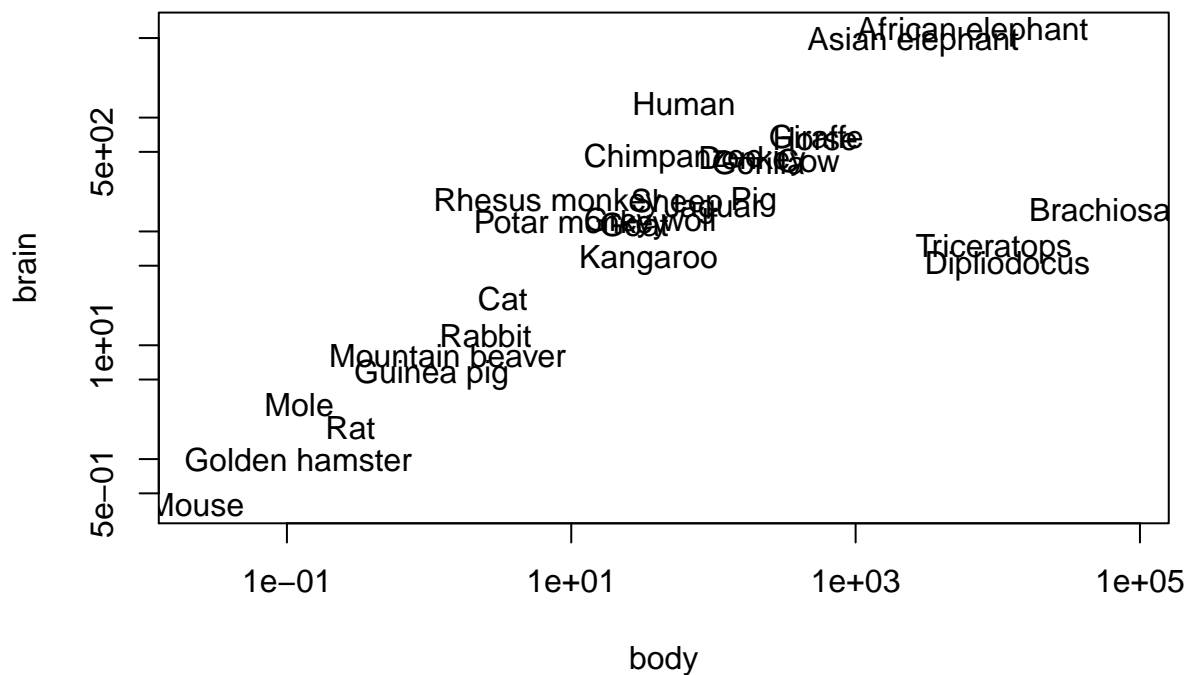
```
require(MASS) #para objeto Animals
```

```
## Loading required package: MASS
```

```
head(Animals) #consulte o help para entender
```

```
##           body brain
## Mountain beaver  1.35  8.1
## Cow              465.00 423.0
## Grey wolf        36.33 119.5
## Goat             27.66 115.0
## Guinea pig        1.04  5.5
## Dipliodocus      11700.00 50.0
```

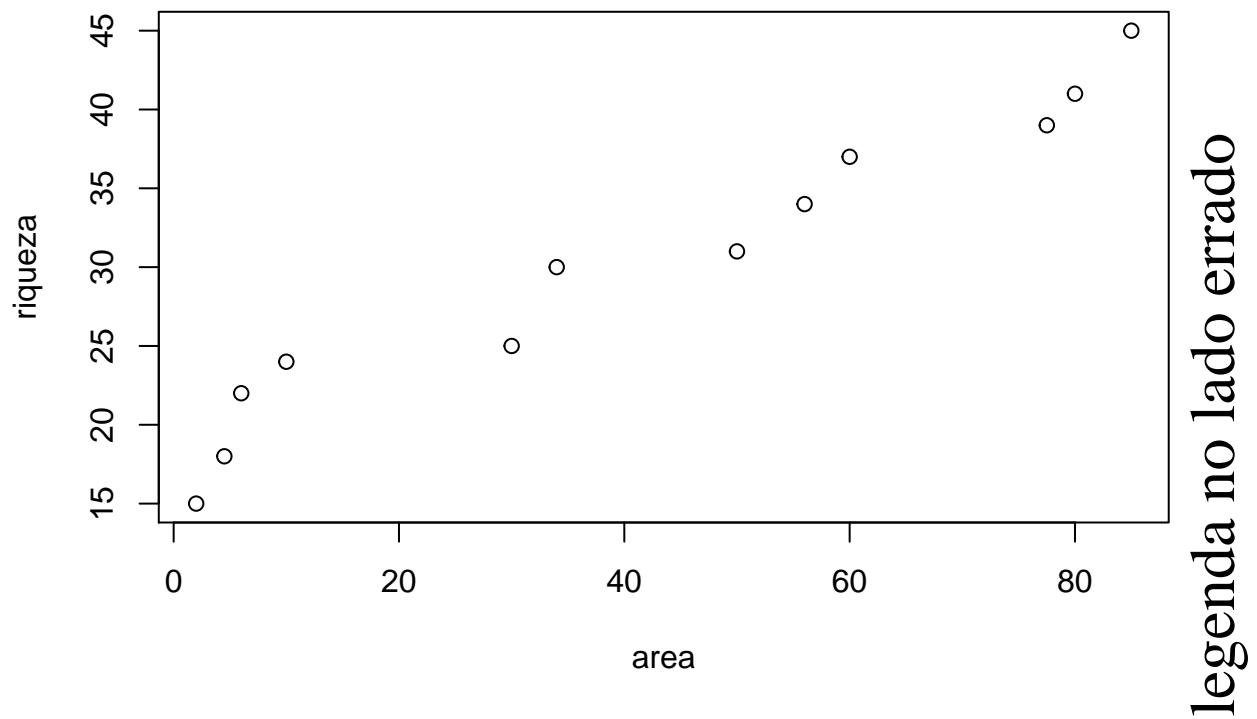
```
plot(brain~body, data=Animals, log="xy", type="n")
text(y=Animals$brain,x=Animals$body,labels=rownames(Animals))
```



```
##mtext()
```

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

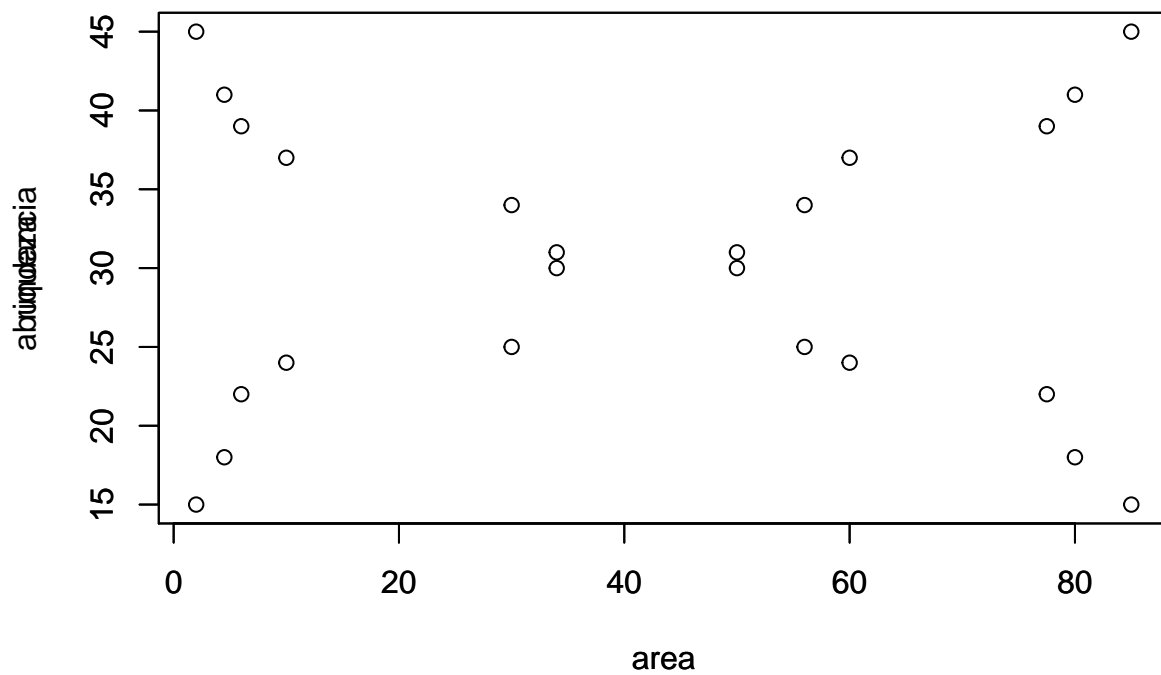
```
plot(riqueza~area)
mtext("legenda no lado errado", side=4, line=0.9, at=20,cex=2, family="serif")
```



`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))`

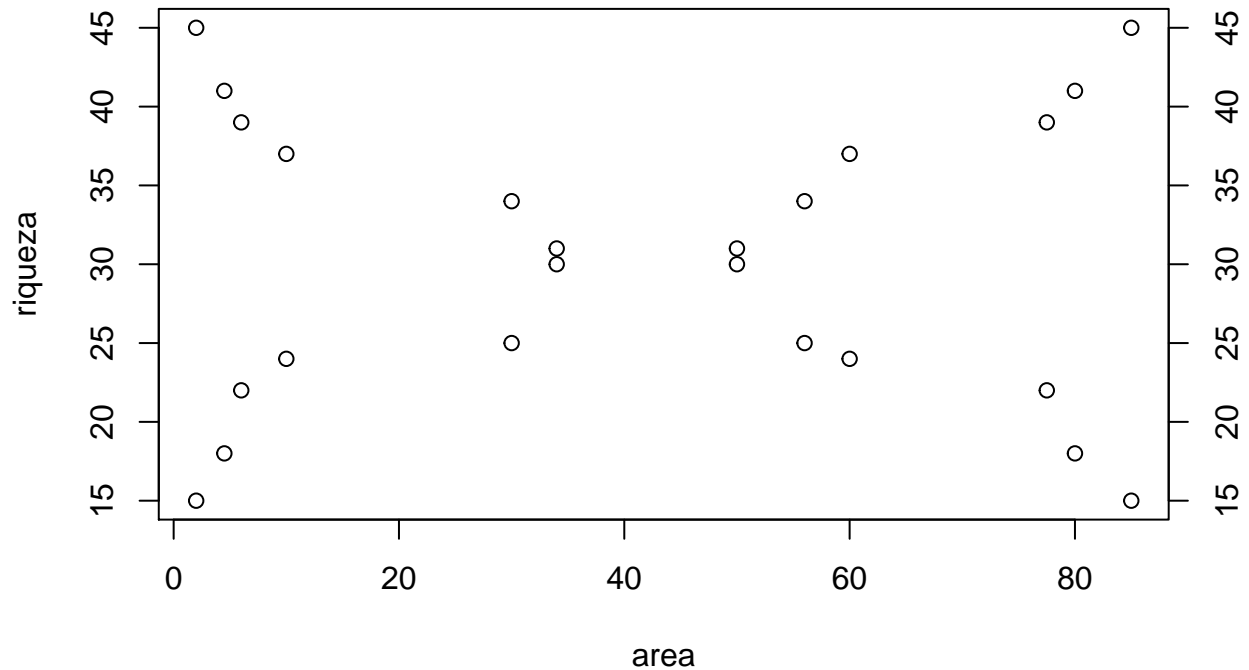
```
plot(riqueza~area)
par(new=TRUE)
plot(abundancia~area)
```



`###axis()`

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

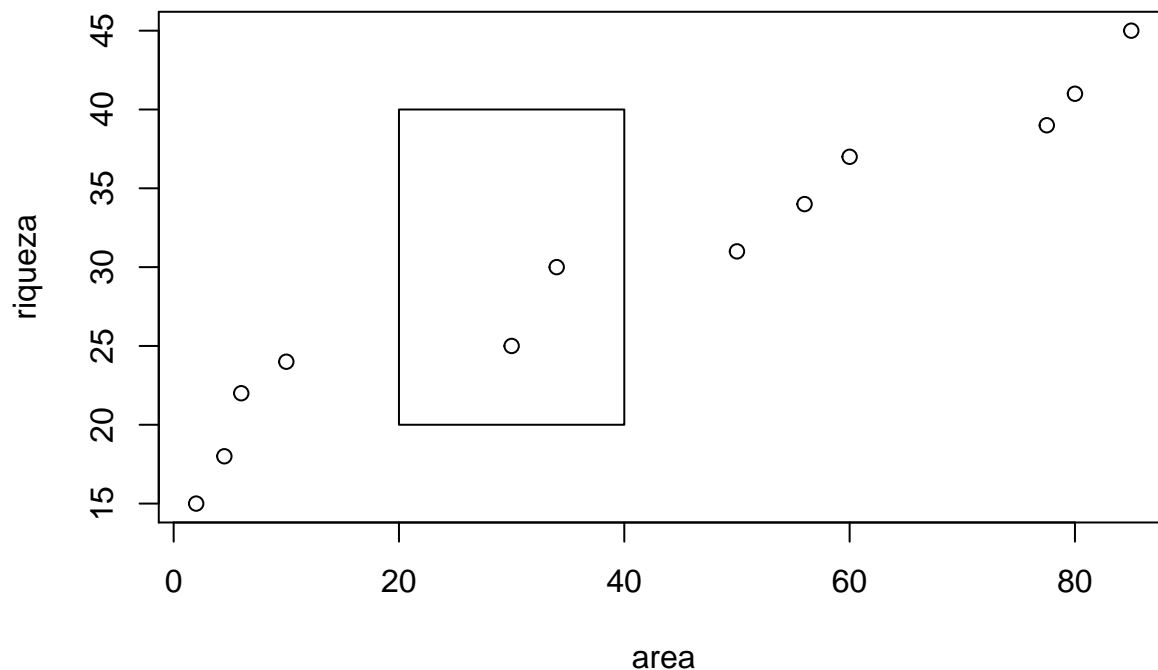
```
plot(riqueza~area)
par(new=TRUE)
plot(abundancia~area, axes=FALSE, ann=FALSE) ## começa com gráfico sem eixos
axis(4) #adiciona eixo
```



###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 feche 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
```

```
## pdf
## 2
```

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:

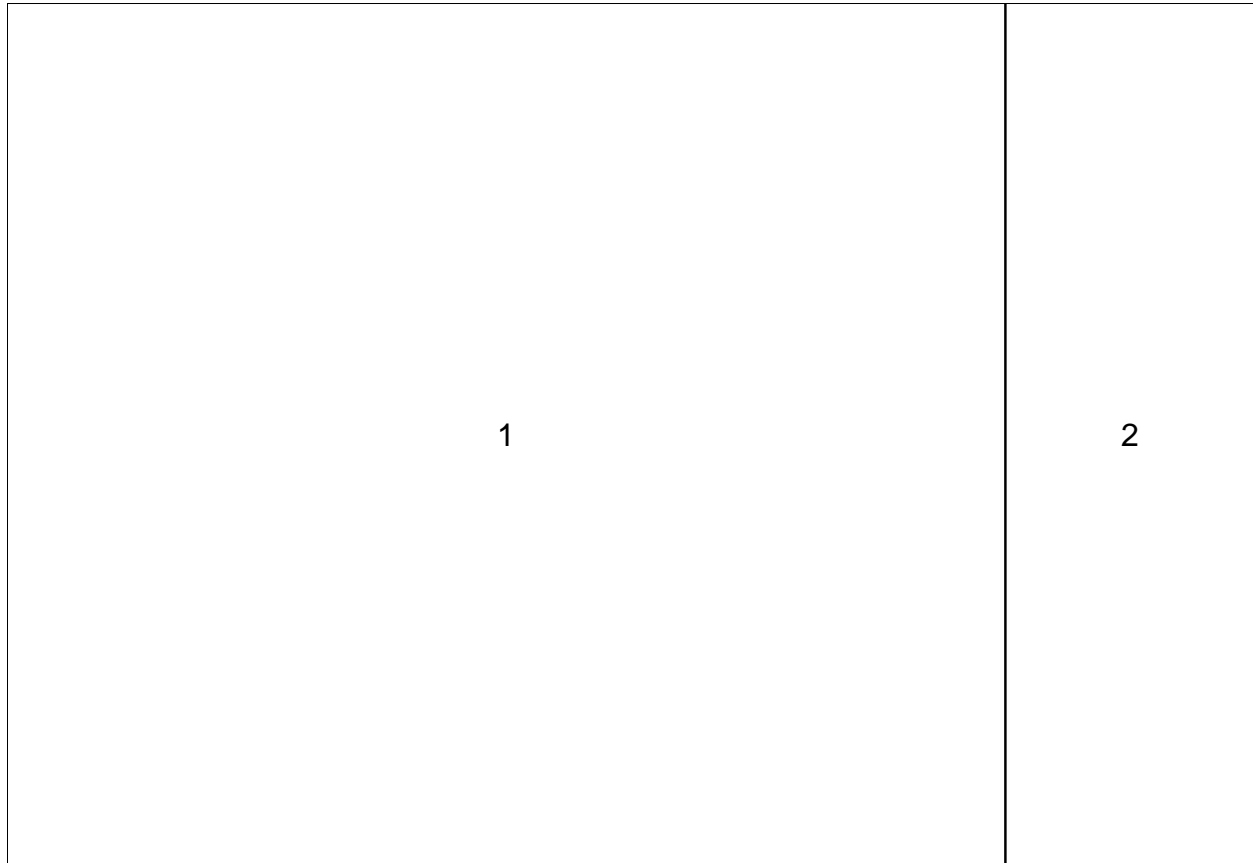
```
png("meugrafico%02d.png")
plot(riqueza~area)
boxplot(riqueza~area.cate)
dev.off()
```

```
## pdf
## 2
```

Criando o layout

A função `layout` cria painéis de diferentes tamanhos no dispositivo gráfico. No nosso caso vamos criar duas colunas, a direita com 80% da largura total. O primeiro argumento da função é uma matriz com a sequência com que os painéis irão ser desenhados.

```
layout(matrix(c(1,2),ncol=2, nrow=1), width=c(8,2))  
layout.show(2) #mostra o layout dos dois painéis
```



Iniciando o primeiro painel

Primeiro ajustamos os parâmetros gráficos do primeiro painel (no caso a margem), em seguida construímos o espaço de coordenadas sem nenhum elemento, apenas algumas legendas;

```
par (mar=c(5,4,4,3.5)) #controla tamanhos das margens  
plot(x=NULL,y=NULL, xlim=c(-1.5,2.5), ylim=c(0.5,7.5),type="n", yaxt="n", xlab="Effect Size (lnOR)", ylab="Effect Size (lnOR)",  
abline (v=0,lty=2) #desenha linhas de regressão (a+bx) ou v=vertical, h=horizontal  
abline (h=c(3,6))  
axis(side=4, at=c(1,2,4,5,7), labels=c("adult (2)", "young (28)", "temperate (28)", "tropical (2)", "overlapped (28)"))
```


SURVIVAL

