Estatística Computacional I

Lupércio França Bessegato
Dep. de Estatística/UFJF

Gráficos em R

Roteiro do Módulo

3. Gráficos em R:
   a) Introdução
   b) Gráficos e argumentos padrão
   c) Personalização de gráficos tradicionais
   d) Controle de aparência dos gráficos
   e) Anotações em gráficos
   f) Criação de novos gráficos
   g) Gráficos dinâmicos
   h) Referências

Anotações em Gráficos

Anotação na Região do Gráfico

- Funções que acrescentam elementos gráficos em plot ativo:
  - √ Em geral, utiliza sistema de coordenadas da região de gráfico ativa
  - √ Elementos gráficos básicos:
    - Linhas, retângulos, texto, etc.
  - √ Funções:
    - `text()`, `points()`, `lines()`, `rect()`, `symbols()`, etc.

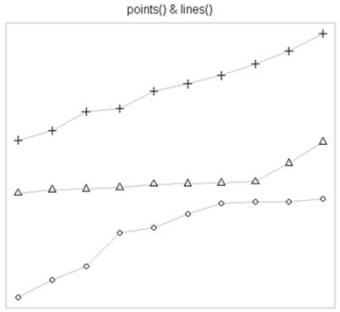
260

- √ Construção de gráficos sequenciais:
  - Funções: `points()` e `lines()`.

```
> # -----  Figura 3.16a
>
>   # conjuntos de dados
> x <- 1:10
> y <- matrix(sort(rnorm(30)), ncol = 3)
>
>   # gráfico do 1° conjunto de y
> plot(x, y[, 1], ylim = range(y), ann = FALSE, axes = FALSE, type = "l",
+ col = "grey")
> points(x, y[,1])
> box(col = "grey")
>
>   # adicionando 2° conjunto de dados de y
> lines(x, y[,2], col = "grey")
> points(x, y[,2], pch = 2)
>
>   # acrescentando 3° conjunto de dados de y
> lines(x, y[,3], col = "grey")
> points(x, y[,3], pch = 3)
>
>   # título usando mtext()
> mtext("points() & lines()", side = 3, line = 0.5)
```

261

- Gráfico final



262

- Comentários:
  - √ Função `lines()`:
    - Valores de `NA` entre os valores das coordenadas criarão quebras nas linhas
  - √ Função `segments()`:
    - Desenha linhas diferentes entre pares de coordenadas
    - Sintaxe:
      - `segments(x0, y0, x1, y1)`

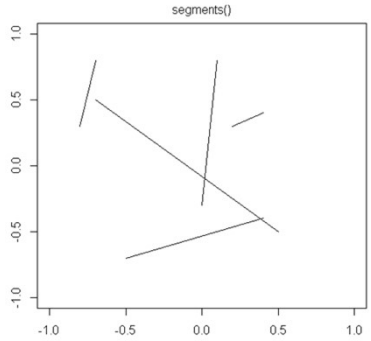
263

√ Construção de segmentos de reta:

– Função: `segments()`.

```
> # função segments()
>
>   # vetores com as coordenadas das extremidades
> varios <- data.frame(x0 = c(0.1, 0.2, - 0.7, 0.4, - 0.8),
+                      y0 = c(0.8, 0.3, 0.5, - 0.4, 0.3),
+                      x1 = c(0, 0.4, 0.5, - 0.5, - 0.7),
+                      y1 = c(- 0.3, 0.4, - 0.5, - 0.7, 0.8))
>
>   # Criando plot vazio
> plot(0, 0, col = "white", xlab = "", ylab = "")
>
>    # Desenhando múltiplos sementps
>  segments(x0 = varios$x0,
+           y0 = varios$y0,
+           x1 = varios$x1,
+           y1 = varios$y1)
>
>    # título usando mtext()
> mtext("segments()", side = 3, line = 0.5)
```


– Gráfico final



- Comentários:

√ Função `text()`:

– Anota texto nas coordenadas `(x, y)`.

– Argumento `pos` auxilia para evitar sobreposição do texto com os símbolos dos dados.

√ Diagrama do uso da função `text()`.

```
> # -----  Figura 3.16b
>
>   # coordenadas dos pontos
> x <- c(4, 5, 2, 1)
> y <- x
>
>   # plot das coordenadas
> plot(x, y, ann = FALSE, axes = FALSE, col = "darkgrey", pch = 3)
> points(3, 3, col = "darkgrey", pch = 16)
> box(col = "darkgrey")
>
>   # anotação do texto
> text(x, y, c("embaixo", "esquerda", "acima", "direita"), pos = 1:4)
> text(3, 3, "sobreposto")
>
>   # título
> mtext("text()", side = 3, line = 0.5)
```

– Gráfico final

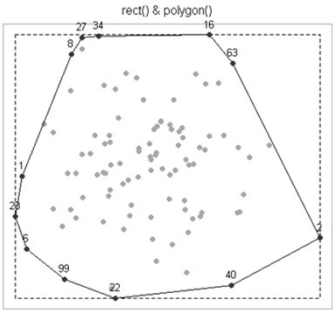
- Comentários:
  - √ Função `rect()`:
    - Desenha retângulos
    - Para múltiplos retângulos, argumentos podem ser vetores
  - √ Função `polygon()`:
    - Desenha polígonos
    - Desenho de múltiplos polígonos com `NA` entre cada conjunto de vértices
  - √ Em ambas funções, argumento `col` especifica cor de preenchimento do interior das formas.

√ Construção de poligonal circunscrita:

– Funções: `rect()` e `polygon()`.

```
> # -----  Figura 3.16c
>   # conjunto de dados
> x <- rnorm(100)
> y <- rnorm(100)
>
>   # diagrama de dispersão dos pontos
> par(ann = FALSE, xpd = TRUE)
> plot(x, y, ann = FALSE, axes = FALSE, col = "darkgrey", pch = 16)
> box(col = "darkgrey")
>
>   # retângulo circunscrito aos dados
> rect(min(x), min(y), max(x), max(y), lty = "dashed")
>
>   # polígono envelopando os dados
>
>   # linhas dos pontos do polígono
> envelope <- chull(x, y)
> polygon(x[envelope], y[envelope])
>   # pontos da poligonal da envoltória
> points(x[envelope], y[envelope], pch = 19, col = "red")
>   # linhas das coordenadas da poligonal
> text(x[envelope], y[envelope], envelope, cex = 0.8, pos = 3)
>
>   # título
> mtext("rect() & polygon()", side = 3, line = 0.5)
```


– Gráfico final

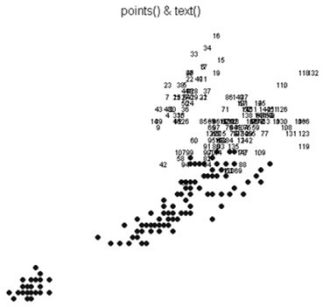


- Comentários:
  - √ Funções `points()`, `lines()` e `text()`:
    - São genéricas
      - Tem interfaces flexíveis na especificação das coordenadas oi produzem saídas diferentes, dependendo da classe do objeto
      - `lines()` atua de maneira diferente com objeto da classe `ts` (*time series*)
    - Aceitam fórmula na especificação das coordenadas

- √ Especificação de coordenadas por fórmula:
  - Funções: `points()` e `text()`.

```
>   # Gráfico length vs. Width
>
>   # limites dos eixos
> xlim <- range(iris[, c(1, 3)])
> ylim <- range(iris[, c(2, 4)])
>
>   # plot em branco
> plot(iris$Sepal.L, iris$Sepal.W, type = "n",  ann = FALSE, axes = FALSE,
+ xlim = xlim, ylim = ylim)
>   # pontos das dimensões das pétalas
> points(Petal.Width ~ Petal.Length, data = iris, col = "blue", pch = 19)
>   # textos nas posições das sépalas
> text(Sepal.Width ~ Sepal.Length, data = iris, rownames(iris), cex = 0.55)
>
>   # título
> mtext("points() & text()", side = 3, line = 0.5)
```


- Gráfico final



- √ Função `grid()`:
  - Adiciona linhas de grade em um plot
  - Objetivo:
    - Auxiliar visualmente, sem interferir com os símbolos dos dados principais
  - Sintaxe:
    - `grid(nx=NULL, ny=nx, lty = "dotted", col = "lightgray", lwd = par("lwd"), equilogs = TRUE)`

√ Linhas de grade – painel e comando externo:

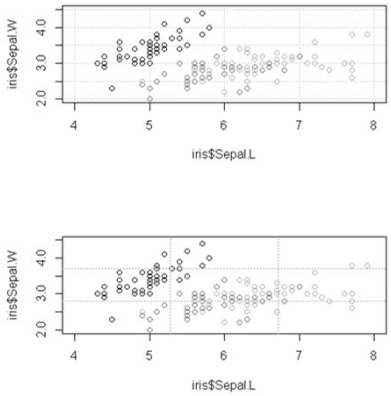
– Função: `grid()`.

```
>   # Função grid()
>
> par(mfrow = c(2,1) )
>   # função grid executada internamente
> plot(iris$Sepal.L, iris$Sepal.W, col = rep(1:3, rep(50, 3)),
+      xlim = c(4, 8), ylim = c(2, 4.5), panel.first = grid())
>
>   # função grid executada externamente
> plot(iris$Sepal.L, iris$Sepal.W, col = rep(1:3, rep(50, 3)),
+      xlim = c(4, 8), ylim = c(2, 4.5))
> grid(nx = 3, col = "darkgray")
```

276

Estatística Computacional I - 2020

– Gráfico final



277

Estatística Computacional I - 2020

√ Função `abline()`:

– Adiciona linhas ou linhas em um plot

– Especificação da linha:

- Inclinação e intercepto
- Coeficientes de regressão linear (objeto `lm`)
- Vetor de valores de x (linhas verticais)
- Vetor de valores de y (linhas horizontais)

– Maneira simples para adicionar reta ajustada de regressão em diagrama de dispersão

√ Função `arrows()`:

– `arrows(x0, y0, x1=x0, y1=y0, length=0.25, angle=30, code=2, col=par("fg"), lty=par("lty"), lwd=par("lwd"), ...)`

278

Estatística Computacional I - 2020

√ Diagrama de dispersão e reta de MQO:

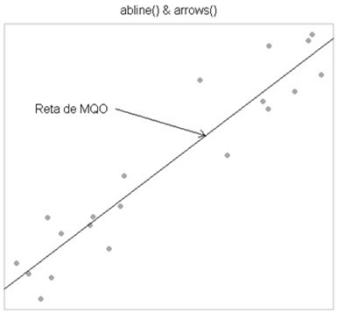
– Função: `lines()` e `arrows()`.

```
> # -----  Figura 3.17a
>
>  # geração dos dados
> x <- runif(20, 1, 10)
> y <- x + rnorm(20)
>
>
> plot(x, y, ann = FALSE, axes = FALSE, col = "darkgray", pch = 16)
> box(col = "grey")
>
>   # ajuste de MQO
> ajuste.ml <- lm(y ~ x)
>   # reta de MQO
> abline(ajuste.ml)
>   # anotação no gráfico
> arrows(5, 8, 7, predict(ajuste.ml, data.frame(x = 7)), length = 0.1)
> text(5, 8, "Reta de MQO", pos = 2)
>
>   # título
> mtext("abline() & arrows()", side = 3, line = 0.5)
```

279

Estatística Computacional I - 2020

– Gráfico final



√ Função `box()`:

– Desenha moldura em torno de regiões da página gráfica:

– Argumento `which`:

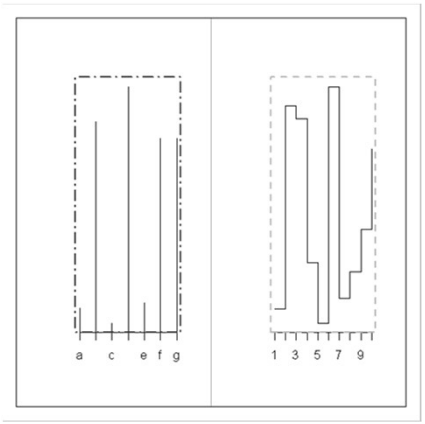
- Possibilita escolher a região a ser limitada
- `"plot"`, `"figure"`, `"inner"` e `"outer"`.

– Maneira simples para adicionar reta ajustada de regressão em diagrama de dispersão.

√ Função `box()`.

```
>   # Funcionamento da função box()
>
> par(mfrow = c(1, 2), oma = c(1, 1, 1, 1))
>
>   # Figura 1
> plot(1:7, abs(stats::rnorm(7)), type = "h", axes = FALSE, ann = F)
> axis(1, at = 1:7, labels = letters[1:7])
> box(which = "figure", col = "darkgrey")
> box(lty = '1373', col = 'red', lwd = 2)
>
>   # Figura 2
> plot(1:10, abs(stats::rnorm(10)), type = "s", axes = FALSE, ann = F)
> axis(1, at = 1:10, labels = 1:10)
> box(which = "figure", col = "darkgrey")
> box(lty = 2, col = "green", lwd = 2)
>
>   # margem externa - limite interior
> box(which = "inner", col = "blue")
>   # margem externa - limite exterior
> box(which = "outer", col = "gray", lwd = 2
```


– Gráfico final



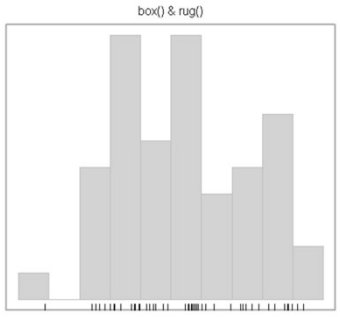
√ Função `rug()`:

- Produz marcas em um dos eixos, representando os valores das coordenadas

√ Função `rug()` e `box()`.

```
> # -----  Figura 3.17b
>
>   # dados
> y <- rnorm(50)
>   # histograma
> hist(y, main = "", ann = F, axes = FALSE, border = "grey",
+ col = "light grey")
>   # moldura do gráfico
> box(col = "darkgrey", lwd = 2)
>   # coordenadas dos dados - univariados
> rug(y, ticksize = 0.02)
>
>   # título
> mtext("box() & rug()", side = 3, line = 0.5)
```


- Gráfico final



Valores Especiais

- `NA` e valores não finitos (`NaN` e `Inf`)
  - √ Maioria das funções permite coordenadas com valores especiais
  - √ R não desenha as coordenadas em questão
    - Símbolos e pontos: não são desenhados
    - Linhas: falha nas coordenadas em questão
    - Retângulos: não será desenhado
  - √ Polígonos:
    - Vértice com valor especial interpretado com final de um polígono e início de outro

√ Desenho de dodecágono:

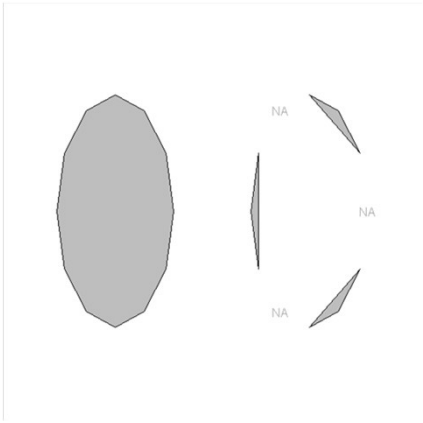
– Polígono regular com 12 lados

```
> #  Figura 3.18 - Drawing polygons
>   # construção de polígono regular com 13 lados
> angle <- seq(0, 2*pi, length = 13)[-13]
> x <- 0.15*cos(angle)
> y <- 0.5 + 0.3*sin(angle)
>
>   # parâmetros
> par(mar = rep(0, 4))
> plot.new()
> box("outer", col = "grey")
>
> # -----  Figura 3.18a
>   # grafico do polígono
> polygon(0.25 + x, y, col = "grey")
>
> # -----  Figura 3.18b
>   # anotação nas coordenadas que serão retirados (NA)
> text(0.75 + x[c(1, 5, 9)], y[c(1, 5, 9)], "NA", col = "darkgrey", lwd = 2)
>
>   # vértice com NA
> x[c(1, 5, 9)] <- NA; y[c(1, 5, 9)] <- NA
>
>   # gráfico do polígono
> polygon(0.75 + x, y, col="grey")
```

288

Estatística Computacional I - 2020

– Gráfico final



289

Estatística Computacional I - 2020

Anotação nas Margens

- Função `mtext()`:
  - √ Anota texto em qualquer posição das margens externas e da figura.
    - – Argumento `outer`:
      - Controla se saída atinge margens externas (se disponível)
      - Default: `FALSE`.

290

Estatística Computacional I - 2020

- É possível anotar nas margens usando funções gráficas [p.ex. `lines()`]
  - √ Em geral, toda saída gráfica é estabelecida apenas na área de plotagem.
  - √ Argumento `xpd`
    - – `NA`: todo dispositivo como região de clipping
    - – `FALSE`: clipping apenas na área de plotagem (*default*)
    - – TRUE: permite atuação da função gráfica na região da figura

291

Estatística Computacional I - 2020

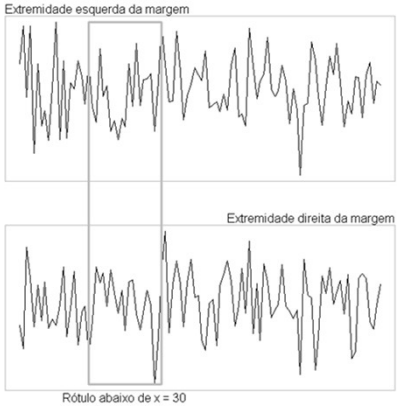
√ Anotações de texto e desenhos nas margens:

– Funções: `mtext()` e `lines()`.

```
> # -----  Figura 3.19
>   # Dados
> set.seed(666)
> y1 <- rnorm(100)
> y2 <- rnorm(100)
>
>   # configurações dispositivo
> par(mar=c(2, 1, 1, 1))
> par(mfrow = c(2, 1), xpd = NA)
>
>   # 1º gráfico
> plot(y1, type = "l", axes = FALSE, xlab = "", ylab = "", main = "")
> box(col = "grey")
> mtext("Extremidade esquerda da margem", adj = 0, side = 3)
> lines(x = c(20, 20, 40, 40), y = c(-9, max(y1), max(y1), -9),
+     lwd = 3, col = "grey")
>   # 2º gráfico
> plot(y2, type = "l", axes = FALSE, xlab = "", ylab = "", main = "")
> box(col = "grey")
> mtext("Extremidade direita da margem", adj = 1, side = 3)
> mtext("Rótulo abaixo de x = 30", at = 30, side = 1)
> lines(x = c(20, 20, 40, 40), y = c(8, min(y2), min(y2), 8),
+     lwd = 3, col = "grey")
```

292

– Gráfico final



293

Legendas

- Função `legend()`:
  - √ Usualmente posicionada na área de plotagem
  - √ Posição de acordo com o sistema de coordenadas do usuário
  - √ Argumentos permitem grande flexibilidade de especificações de conteúdo e formato

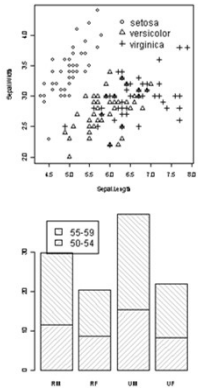
294

√ Gráfico com usos típicos da função `legend()`.

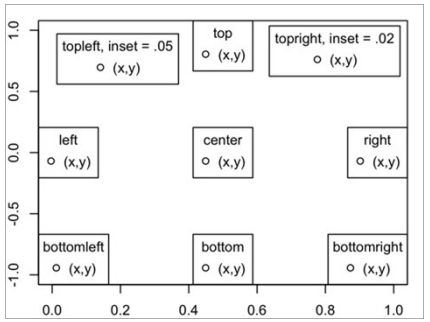
```
> # ===================================================================
> #  Figure 3.20 - Some simple legends
> # ===================================================================
>
>
>
> par(mfrow = c(2, 1), mar = c(5, 3, 2, 1), cex = 0.5, pty = "s")
>
> # -----  Figura 3.20a
>
>
>      plot(Sepal.Width ~ Sepal.Length, data = iris, pch = as.numeric(Species),
+ cex=1.2)
> legend(6.1, 4.4, levels(iris$Species), cex = 1.5, pch = 1:3, bty = "n")
>
> # -----  Figura 3.20b
>
> barplot(VADeaths[1:2,], angle = c(45, 135), density = 20,
+         col = "grey", names = c("RM", "RF", "UM", "UF"))
> legend(0.4, 38, c("55-59", "50-54"), cex = 1.5,
+        angle = c(135, 45), density = 20, fill = "grey")
```

295

– Gráfico final



- Posicionamento pode ser especificado por:
  - √ `"topleft"`, `"top"`, `"topright"`, `"left"`, `"right"`, `"center"`, `"bottomright"`, `"bottomleft"`, `"bottom"`,



- Comentários:
  - √ Usuário deve assegurar-se que a legenda corresponde ao plot
  - √ Não há verificação automática entre os símbolos da legenda e do plot ou entre os rótulos na legenda e os dados
  - √ Algumas funções high-level desenham legenda específica
    - Ex.: `filled.contour()`

Eixos

- Em geral, são gerados automaticamente
  - √ Ex.:
    - Rotulados com nomes dos níveis de fator
- Função `par()` permite modificação da aparência default dos eixos

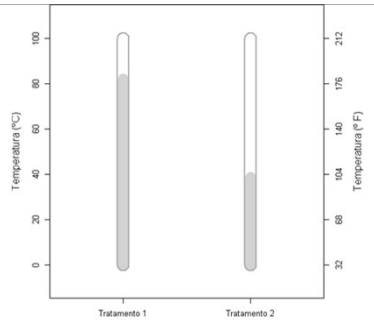
- Maioria da funções gráficas permite inibição de geração automática dos eixos:
  - √ Todos os eixos: `axes=F`
  - √ Um dos eixos: `xaxt="n"` (ou `yaxt="n"`)
- Função `axis()`:
  - √ `side`: lado do plot para posicionamento eixo.
  - √ `at`: locação das marcas.
  - √ `labels`: texto dos rótulos

√ Exemplo de personalização dos eixos.

```
> #  ------------------  Figura 3.21 - Customizing axes -----------
>
>   # dados de temperatura
> x <- 1:2
> y <- runif(2, 0, 100)
>
>   # plot vazio
> par(cex = 0.8, mar = c(4, 4, 2, 4))
> plot(x, y, type = "n", xlim = c(0.5, 2.5), ylim = c(-10, 110),
+      axes = FALSE, ann = FALSE)
>
>   # eixo y - lado esquerdo (side = 2)
> axis(2, at = seq(0, 100, 20))
> mtext("Temperatura (°C)", side = 2, line = 3)
>
>   # eixo x - embaixo (side = 1)
> axis(1, at = 1:2, labels = c("Tratamento 1", "Tratamento 2"))
>
>   # eixo y - lado direito (side = 3)
> axis(4, at = seq(0, 100, 20), labels = seq(0, 100, 20)*9/5 + 32)
> mtext("Temperatura (° F)", side = 4, line = 3)
>
>   # moldura em torno da região do plot
> box()
> ...
```


– Continuação do código:

```
...
>   # representação do termômetros
> segments(x, 0, x, 100, lwd = 20, col = "dark grey")
> segments(x, 0, x, 100, lwd = 16, col = "white")
> segments(x, 0, x, y, lwd = 16, col = "light grey")
>
```

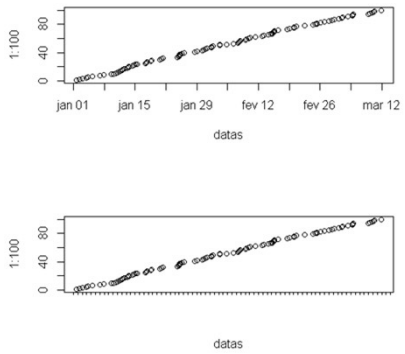


- Função para plotar dados relacionados com tempo [`axis.Date()`]
  - √ Usam objeto contendo datas
  - √ Produz eixo com rótulos apropriados para representar tempo (dias, meses e anos)
- Função `axTicks()` pode ser usada para determinar as posições das marcas de escala:
  - √ Pode ser útil para forçar um ajuste dado por `xaxp` (ou `yaxp`)

√ Exemplo da função `axis.Date()`:

```
> # ------ Funções axis.Date() ----
>
>   # 100 datas aleatórias em um período de 10 semanas
> datas <- as.Date("2021/1/1") + 70*sort(runif(100))
>
> par(mfrow = c(2, 1))
>   # plot das datas com eixo x em semanas
> plot(datas, 1:100, xaxt = "n")
> axis.Date(1, at = seq(as.Date("2021/1/1"), max(datas) + 6, "week"))
>
>   # plot das datas com eixo x em dias
> plot(datas, 1:100, xaxt = "n")
> axis.Date(1, at = seq(as.Date("2021/1/1"), max(datas) + 6, "days"),
+       labels = FALSE,
+ tcl = -0.2)
>
> axis.Date(1, at = seq(as.Date("2021/1/1"), max(datas) + 6, "days"),
+       labels = FALSE, tcl = -0.2)
```


– Gráfico final



Anotação Matemática

- Funções anotam texto e expressões matemática:
  - √ Função `expression()`:
    - – Saída é expressão matemática
  - √ Descrição completa:
    - – `help(plotmath)`
    - – `demo(plotmath)`
  - √ R interpreta certos nomes como símbolos matemáticos especiais

√ Operadores e símbolos para anotação matemática (1):

Arithmetic Operators		Lists	
x + y	x + y	list(x, y, z)	x, y, z
x - y	x − y	**Relations**	
x * y	xy	x == y	x = y
x/y	x/y	x != y	x ≠ y
x %+-% y	x ± y	x < y	x < y
x%/%y	x ÷ y	x <= y	x ≤ y
x %*% y	x × y	x > y	x > y
x %.% y	x · y	x >= y	x ≥ y
-x	−x	x %~~% y	x ≈ y
+x	+x	x %=~% y	x ≅ y
Sub/Superscripts		x %==% y	x ≡ y
x[i]	x_i	x %prop% y	x ∝ y
x^2	x^2	x %~% y	x ~ y
Juxtaposition		**Typeface**	
x * y	xy	plain(x)	x
paste(x, y, z)	xyz	italic(x)	*x*
Radicals		bold(x)	**x**
sqrt(x)	$\sqrt{x}$	bolditalic(x)	***x***
sqrt(x, y)	$\sqrt[y]{x}$	underline(x)	$\underline{x}$

√ Operadores e símbolos para anotação matemática (2):

Ellipsis		Arrows	
list(x[1], ..., x[n])	$x_1, \ldots, x_n$	x %<->% y	$x \leftrightarrow y$
x[1] + ... + x[n]	$x_1 + \cdots + x_n$	x %->% y	$x \rightarrow y$
list(x[1], cdots, x[n])	$x_1, \cdots, x_n$	x %<-% y	$x \leftarrow y$
x[1] + ldots + x[n]	$x_1 + \ldots + x_n$	x %up% y	$x \uparrow y$
Set Relations		x %down% y	$x \downarrow y$
x %subset% y	$x \subset y$	x %<=>% y	$x \Leftrightarrow y$
x %subseteq% y	$x \subseteq y$	x %=>% y	$x \Rightarrow y$
x %supset% y	$x \supset y$	x %<=% y	$x \Leftarrow y$
x %supseteq% y	$x \supseteq y$	x %dblup% y	$x \Uparrow y$
x %notsubset% y	$x \not\subset y$	x %dbldown% y	$x \Downarrow y$
x %in% y	$x \in y$	**Symbolic Names**	
x %notin% y	$x \notin y$	Alpha - Omega	$A - \Omega$
Accents		alpha - omega	$\alpha - \omega$
hat(x)	$\hat{x}$	phi1 + sigma1	$\varphi + \varsigma$
tilde(x)	$\tilde{x}$	Upsilon1	Υ
ring(x)	$\mathring{x}$	infinity	∞
bar(xy)	$\overline{xy}$	32 * degree	32°
widehat(xy)	$\widehat{xy}$	60 * minute	$60'$
widetilde(xy)	$\widetilde{xy}$	30 * second	$30''$

√ Operadores e símbolos para anotação matemática (3):

Style	
displaystyle(x)	x
textstyle(x)	x
scriptstyle(x)	x
scriptscriptstyle(x)	x
Spacing	
x ~ ~y	$x\ \ y$

x + phantom(0) + y	$x + \ \ + y$
x + over(1, phantom(0))	$x + \frac{1}{\ }$
Fractions	
frac(x, y)	$\frac{x}{y}$
over(x, y)	$\frac{x}{y}$
atop(x, y)	$\begin{matrix} x \\ y \end{matrix}$

√ Operadores e símbolos para anotação matemática (4):

Big Operators	
sum(x[i], i = 1, n)	$\sum_{1}^{n} x_i$
prod(plain(P)(X == x), x)	$\prod_{x} P(X = x)$
integral(f(x) * dx, a, b)	$\int_{a}^{b} f(x)dx$
union(A[i], i == 1, n)	$\bigcup_{i=1}^{n} A_i$
intersect(A[i], i == 1, n)	$\bigcap_{i=1}^{n} A_i$
lim(f(x), x %->% 0)	$\lim_{x \to 0} f(x)$
min(g(x), x >= 0)	$\min_{x \geq 0} g(x)$
inf(S)	$\inf S$
sup(S)	$\sup S$

√ Operadores e símbolos para anotação matemática (5):

Grouping	
(x + y) * z	$(x+y)z$
x^y + z	$x^y + z$
x^(y + z)	$x^{(y+z)}$
x^{y + z}	x^{y+z}
group("(", list(a, b), "]")	$(a, b]$
bgroup("(", atop(x, y), ")")	$\binom{x}{y}$
group(lceil, x, rceil)	$\lceil x \rceil$
group(lfloor, x, rfloor)	$\lfloor x \rfloor$
group("|", x, "|")	$\lvert x \rvert$

√ Algumas expressões matemáticas em plot:

Temperatura (°C) em 2021

expression(paste("Temperatura (", degree, "C) em 2021"))

$$\bar{x} = \sum_{i=1}^{n} \frac{x_i}{n}$$

expression(bar(x) == sum(frac(x[i], n), i == 1, n))

$$\hat{\beta} = (X^t X)^{-1} X^t y$$

hat(beta) == (X^t * X)^{-1} * X^t * y

$$z_i = \sqrt{x_i^2 + y_i^2}$$

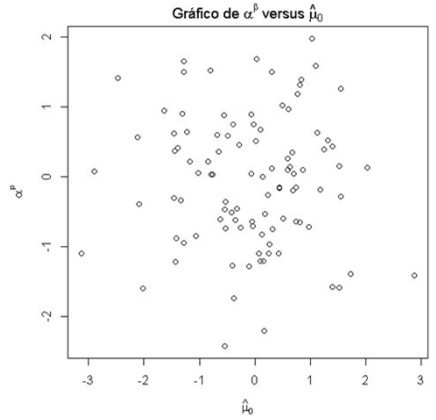
expression(z[i] == sqrt(x[i]^2 + y[i]^2))

√ Combinando expressões e texto:

- Função `paste()` dentro de `expression()`:

```
> # mesclando texto e expressão matemática
>
> par(mar = c(4, 4, 2, 0.1))
> plot(rnorm(100), rnorm(100),
+   xlab = expression(hat(mu)[0]), ylab = expression(alpha^beta),
+   main = expression(paste("Gráfico de ", alpha^beta, " versus ", hat(mu)[0])))
```


√ Combinação *string* e expressão matemática:



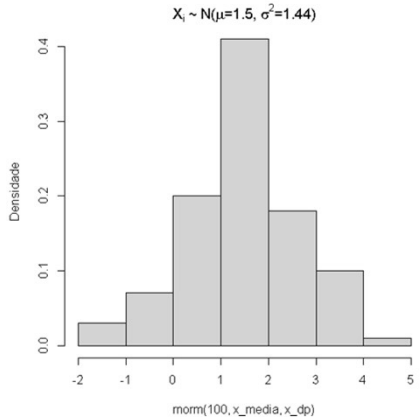
√ Fórmula matemática usando valores armazenados em variáveis

- Nome da variável será tratado como string
- Função `substitute()`:

√ Combinando expressões e variáveis:
- Função `substitute()`:

```
> # uso de substitute - mesclando expressão matemática e variáveis
>
> par(mar = c(4, 4, 2, 0.1))
>
>   # variaveis
> x_media <- 1.5
> x_dp <- 1.2
>
>   # plot
> hist(rnorm(100, x_media, x_dp),freq = FALSE, ylab = "Densidade",
+   main = substitute(
+     paste(X[i], " ~ N(", mu, "=", m, ", ", sigma^2, "=", s2, ")"),
+     list(m = x_media, s2 = x_dp^2)
+ )
+ )
```


√ Combinação expressão matemática e variável:



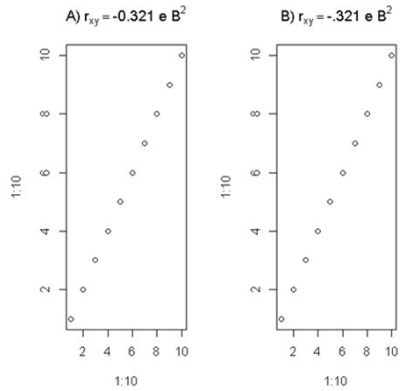
√ Fórmula matemática com valores de variáveis:
- Função `bquote()`:

√ Regras para sua utilização:
- *Strings*: entre aspas com separador til
  - (`"texto"~`)
- Expressões matemáticas:
  - Sem aspas, de acordo com operadores e símbolos
- Números:
  - Sem aspas, quando fizerem parte da anotação matemática
- Variáveis:
  - Usar `.()` (*string* ou numérica)

√ Combinando expressões e variáveis:
- Função `bquote()`:
- `cor`: armazena número; `cor2`: armazena *string*.

```
> # uso de bquote - mesclando expressão matemática e variáveis
>
>
>  # variável numérica
> cor <- -.321
>   # variável de string
> cor2 <- "-.321"
>
> par(mfrow = c(1, 2))
> plot(1:10, 1:10, main = bquote("A)" ~ r[xy] == .(cor) ~ "e" ~ B^2))
> plot(1:10, 1:10, main = bquote("B)" ~ r[xy] == .(cor2) ~ "e" ~ B^2))
```

√ Combinação expressão matemática e variável:



Sistema de Coordenadas

- Sistema gráfico tradicional:
  - √ Área de plotagem:
    - Posicionamento de acordo com escalas dos eixos coordenados
  - √ Margens da figura:
    - Posicionamento em termos de linhas de texto
  - √ Possível posicionar saídas de acordo com outro sistema de coordenadas
    - Calculado a partir do sistema existente

- Argumentos da função `par()`:
  - √ `"din"`: dimensões atuais do dispositivo gráfico, em polegadas [`c(width, height)`].
  - √ `"pin"`: dimensões atuais da área de plotagem, em polegadas [`c(width, height)`].
  - √ `"fin"`: dimensões atuais da região da figura, em polegadas [`c(width, height)`].

- `par("usr")`:
  - √ Fornece a amplitude em uso dos eixos x e y (sistema de coordenadas)
  - √ `c(xmin, xmax, ymin, ymax)`

√ Uso de coordenadas personalizadas :

```
> # -----  Figura 3.23 - Sistema personalizado de coordenadas
>
> dev.new(width = 5, height = 1.5, unit="in")
>
> par(mar = rep(1, 4))
> plot(0:1, 0:1, type = "n", axes = FALSE, ann = FALSE)
> usr <- par("usr")
> pin <- par("pin")
> xcm <- diff(usr[1:2])/(pin[1]*2.54)
> ycm <- diff(usr[3:4])/(pin[2]*2.54)
>
> par(xpd = NA)
> rect(0 + 0.2*xcm, 0 - 0.2*ycm,
+      1 + 0.2*xcm, 1 - 0.2*ycm,
+      col = "grey", border = NA)
>
> rect(0, 0, 1, 1, col = "white")
> segments(seq(1, 8, 0.1)*xcm, 0,
+         seq(1, 8, 0.1)*xcm,
+         c(rep(c(0.5, rep(0.25, 4),
+               0.35, rep(0.25, 4)),
+             7), 0.5)*ycm)
> text(1:8*xcm, 0.6*ycm, 0:7, adj = c(0.5, 0))
> text(8.2*xcm, 0.6*ycm, "cm", adj = c(0, 0))
```

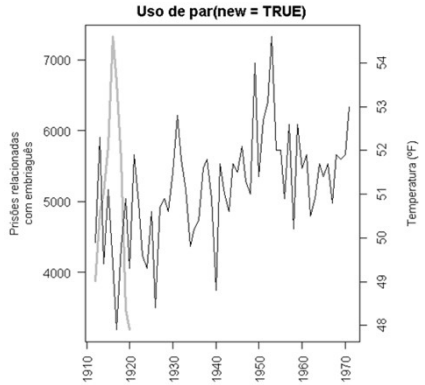
- Comentários sobre o código:
  - √ `par(xpda=NA):`
    - permite retângulo cinza ser desenhado também nas margens da figura
  - √ `rect():`
    - desenha retângulos (extremos da régua)
  - √ `segments():`
    - desenha marcas de escala
  - √ `text():`
    - escreve valores da escala.

- Sobreposição de saídas – Exemplo A:
  - √ Período: 1912 à 1971
  - √ Prisões relacionadas com embriaguez, em New Connecticut
    - Dados disponíveis apenas nos 9 primeiros anos
  - √ Temperatura média anual em New Haven
    - Conjunto de dados: `nhtemp`
  - √ Plot da mesma variável `x` (período), com escalas `y` diferentes (prisões e temperatura).

√ 1ª. Abordagem: `par(new = TRUE)`
- Permite sobrepor dois gráficos distintos

```
> # ------ Figura 3.24a - Embriaguês & Temperatura  ----
>
>   # Dados de prisões relacionadas com embriaguês
> embriagues <- ts(c(3875, 4846, 5128, 5773, 7327, 6688, 5582, 3473, 3186,
+                    rep(NA, 51)), start=1912, end=1971)
>
>    # 1a. abordagem - par(new = TRUE)
>
>   # 1o. plot
> par(mar = c(5, 6, 2, 4))
> plot(embriagues, lwd = 3, col = "grey", ann = FALSE, las = 2)
> mtext("Prisões relacionadas\ncom embriaguês", side = 2, line = 3.5, cex = 0.9)
>
>   # 2o. plot
> par(new = TRUE)
> plot(nhtemp, ann = FALSE, axes = FALSE)
> mtext("Temperatura (°F)", side = 4, line = 3, cex = 0.9)
> title("Uso de par(new = TRUE)")
> axis(4)
```

– Gráfico final

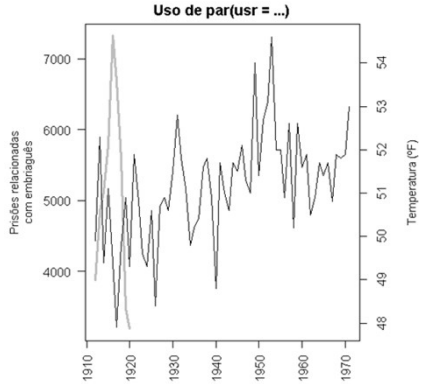


√ 2ª. Abordagem: `par("usr")`

– Redefinir o estado `usr` antes de plotar o segundo conjunto de dados

```
> # ------ Figura 3.24a - Embriaguês & Temperatura
>
>   # 2a. abordagem - par("usr")
>
>   # 1o. plot
> par(mar = c(5, 6, 2, 4))
> plot(embriagues, lwd = 3, col = "grey", ann = FALSE, las = 2)
> mtext("Prisões relacionadas\ncom embriaguês", side = 2, line = 3.5, cex = 0.9)
>
>   # 2o. plot
> usr <- par("usr")
> par(usr = c(usr[1:2], 47.6, 54.9))
> lines(nhtemp)
> mtext("Temperatura (°F)", side = 4, line = 3, cex = 0.9)
> title("Uso de par(usr = ...)")
> axis(4)
```


– Gráfico final



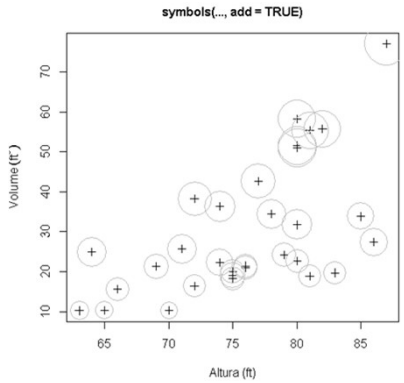
- Sobreposição de saídas – Exemplo B:
  - √ Dimensões de cerejeiras
  - √ Variáveis:
    - `Girth`: diâmetro da árvore, em polegadas
    - `Height`: altura, em pés
    - `Volume`: volume de madeira, em pés cúbicos
  - √ Conjunto de dados: `trees`
  - √ Gráfico de bolhas:
    - `Volume x Height x Girth`.

√ 1ª. Abordagem: `par(new = TRUE)`

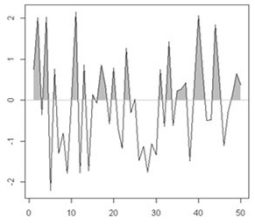
– Permite sobrepor dois gráficos distintos

```
> # ---- Fig 3.24b - Gráfico de bolhas trees  ----
>
>
>   # comando do 2o. gráfico com add = TRUE
>
>   # 1o. plot - scatterplot
>
> par(mar=c(5, 4, 4, 2))
> plot(Volume ~ Height, data = trees, pch=3, xlab = "Altura (ft)",
+      ylab = expression(paste("Volume ", (ft^3))))
>
>   # 2o. plot - bubles
>
> with(trees, {
+        symbols(Volume ~ Height, circles = Girth/12, fg = "grey",
+        inches = FALSE, add = TRUE)
+      })
> mtext("symbols(..., add = TRUE)", font = 2, side = 3, line = 1)
>
```


– Gráfico final



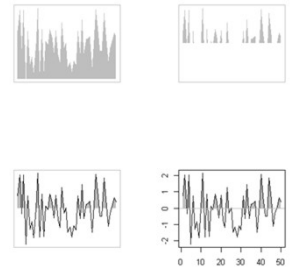
- Sobreposição de saídas – Exemplo C:
  - √ Gráfico sequencial qualquer
  - √ Variáveis:
    - `xx`: ordem dos valores da série, de 1 a 50
    - `yy`: valores independentes e identicamente distribuídos de acordo com normal padrão
  - √ Gráfico desejado:



√ Criação do conjunto de dados:

```
>   # geração do conjunto de dados e valores
> set.seed(666)
> xx <- c(1:50)
> yy <- rnorm(50)
> n <- 50
> hline <- 0
```

√ Construção em etapas:



√ Construção do gráfico em etapas:

- 1ª etapa:
  - Polígono cinza abaixo dos valores de y
  - Plot sem os pontos e uso da função `polygon()`
- 2ª etapa:
  - Sobreposição retângulo branco abaixo de 0
  - Uso da função `rect()` para plotar área abaixo de 0 com mesma cor de fundo.
  - Comando `par("usr")` para obter escalas de x e y
- 3ª etapa:
  - Linha conectando pontos plotados
  - Comando `lines()`.
- 4ª etapa:
  - Linha cinza em y = 0
  - Construção dos eixos
  - Caixa nos limites da área de plotagem

336

Estatística Computacional I - 2020

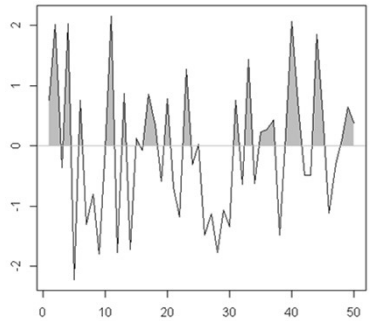
√ Sobreposição de saídas gráficas:

```
> #  Figura 3.25 - Sobreposição de saídas gráficas
> # ================================================================
>
>   # Plot com configurações dos pontos
> plot (yy ~ xx, type = "n", axes = FALSE, ann = FALSE)
>
>   # 1ª etapa - polígono cinza abaixo dos valores de y
> polygon(c(xx[1], xx, xx[n]), c(min(yy), yy, min(yy)),
+         col = "grey", border = NA)
>
>   # 2ª etapa - sobreposição de retângulo branco abaixo de 0
> usr <- par("usr")
> rect(usr[1], usr[3], usr[2], hline, col = "white", border = NA)
>
>   # 3ª etapa - linha conectando os pontos
> lines(xx, yy)
>
>   # 4ª etapa - conclusão da figura
> abline(h = hline, col = "grey")
> box()
> axis(1)
> axis(2)
```

337

Estatística Computacional I - 2020

– Gráfico final



338

Estatística Computacional I - 2020

Casos Especiais

- Algumas funções *high-level* podem dificultar a anotação em gráficos:
  - √ Não é óbvia a área de plotagem estabelecida pela função
  - √ Área de plotagem indisponível após execução da função

339

Estatística Computacional I - 2020

- Escala confusa:
  - √ A escala de eixo da variável categórica não é óbvia no sistema gráfico tradicional.
    - – Ex.: `barplot()`; `boxplot()`
  - √ Como adicionar anotação extra?

340

- Anotação em `barplot()`:
  - √ Escala numérica não é óbvia:
    - – Por *default*, escala do eixo do fator não é todo rotulado
  - √ Função `barplot()`: saída fornece coordenada dos pontos médios de cada barra.

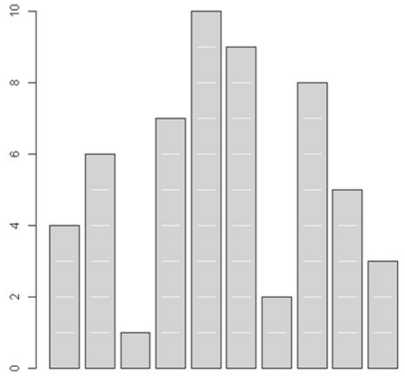
341

- √ Linhas horizontais adicionais em *barplot*:
  - – Variável médios: extremidades segmentos
  - – Função `segments()`: anotação dos segmentos

```
> # -----  Figura 3.27a
>   # Dados
> y <- sample(1:10)
> par(mar = c(3, 2, 1, 1))
> (medios <- barplot(y, col = "light grey"))
      [,1]
 [1,]  0.7
 [2,]  1.9
 [3,]  3.1
 [4,]  4.3
 [5,]  5.5
 [6,]  6.7
 [7,]  7.9
 [8,]  9.1
 [9,] 10.3
[10,] 11.5
> largura <- diff(medios[1:2])/4
> esquerda <- rep(medios, y - 1) - largura
> direita <- rep(medios, y - 1) + largura
> alturas <- unlist(apply(matrix(y, ncol = 10), 2, seq))[-cumsum(y)]
> segments(esquerda,alturas, direita, alturas, col = "white")
```

342

- – Gráfico final



343

- Anotação em `boxplot()`:
  - √ Ajuste na escala do fator é mais intuitiva
    - Boxplots individuais são desenhados nas coordenadas `1:n`
    - `n`: quantidade de boxplots

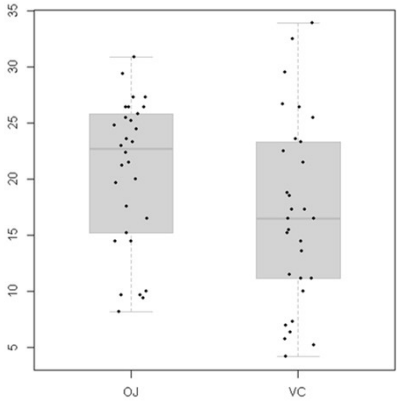
- √ Anotação de símbolos dos pontos dos dados:
  - Posição na escala `x` com ruído
  - Conjunto de dados: `ToothGrowth`

```
> str(ToothGrowth)
data.frame':   60 obs. of  3 variables:
 $ len : num  4.2 11.5 7.3 5.8 6.4 10 11.2 11.2 5.2 7 ...
 $ supp: Factor w/ 2 levels "OJ","VC": 2 2 2 2 2 2 2 2 2 2 ...
 $ dose: num  0.5 0.5 0.5 0.5 0.5 0.5 0.5 0.5 0.5 0.5 ...
> table(ToothGrowth$supp)

OJ VC
30 30
> # -----  Figura 3.27b
>
>   # boxplot
> par(mar = c(3, 3, 1, 1))
> boxplot(len ~ supp, data = ToothGrowth, border = "grey", col = "light grey",
+   boxwex = 0.5)
>   # pontos dos dados
> with(ToothGrowth,
+ {
+       points(jitter(rep(1:2, each = 30), 0.5),
+              unlist(split(len, supp)),
+              cex = 0.5, pch = 16)
+      })
```


- Função `jitter()`:
  - √ Adiciona ruído a um vetor numérico
  - √ Níveis `OJ` e `VC` do fator `supp`:
    - Coordenadas `x`: 1 e 2 (30 valores em cada nível)

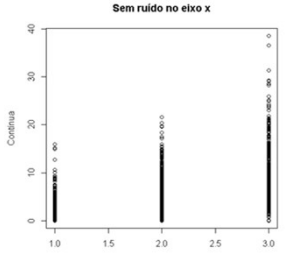
- √ Gráfico final:



- √ Boxplot é um gráfico de 5 números.
- √ Dados com ruído centrados nas posições 1 e 2
- √ Tipo de visualização útil para mostrar padrões que podem estar ofuscados pelo boxplot.

- Visualização de pontos concentrados:
  - √ 3 grupos de 500 valores de normais ao quadrado, com médias diferentes

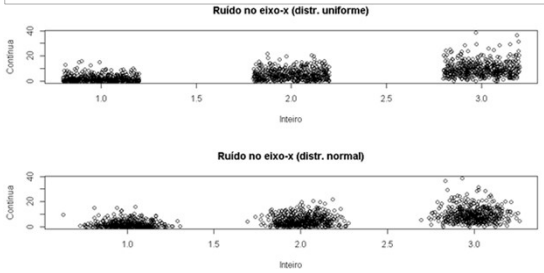
```
>   # Geração dos dados
>
> n <- 500
> dados <- data.frame(inteiro = rep(1:3, each=n),
+ continua = c(rnorm(n, mean = 1), rnorm(n, mean = 2), rnorm(n, mean =3))^2)
>
>   # plot sem ruído
> plot(dados, main = "Sem ruído no eixo x", xlab = "Inteiro", ylab = "Contínua"
```



- √ Plots com ruído uniforme e ruído branco:

```
> par(mfrow = c(2, 1))
>   # Plot com ruído uniforme
> plot(jitter(dados[, 1]), dados[, 2], main = "Ruído no eixo-x (distr.
uniforme)",
+ xlab = "Inteiro", ylab = "Contínua")
>   # plot com ruído branco
> plot(dados[, 1] + rnorm(3*n, sd = 0.1), dados[, 2],
+ main = "Ruído no eixo-x (distr. normal)", xlab = "Inteiro",
+ ylab = "Contínua")
```

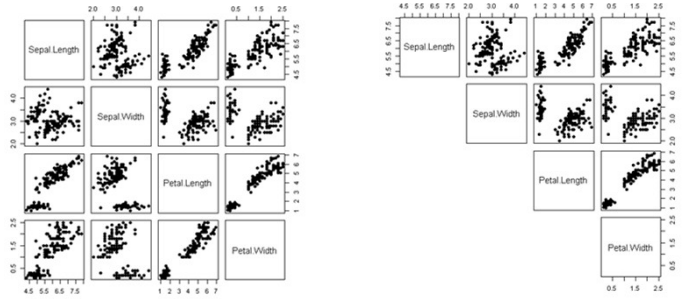
Espalhamento controlado pelo desvio padrão



- Funções que desenham vários plots:
  - √ Função `pairs()`:
    - Salva estado gráfico atual
    - Aciona `par(mfrow)` ou `layout()`
    - Desesenha todos os gráficos
    - Restaura o estado gráfico
  - √ Não é possível anotar em qualquer dos *scatterplots* após a execução de `pairs()`.
  - √ Funções paanel anotam a saída dessa função:
    - `diag.panel`, `upper.panel`, `lower.panel` e `text.panel`

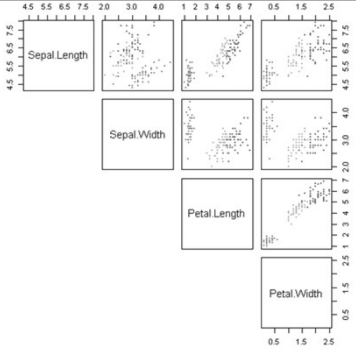
- Visualização das variáveis de `iris`:

```
> # -----  plot básico
>
> pairs(iris[,1:4], pch = 19)
>
> # -----  plot apenas com o painel superior
>
> pairs(iris[,1:4], pch = 19, lower.panel = NULL)
```



√ Símbolos coloridos de acordo a `Species`:

```
> # -----  plot com símbolos dos dados coloridos de acordo com o fator Species
> cores <- c("#00AFBB", "#E7B800", "#FC4E07")
> pairs(iris[,1:4], pch = 19,  cex = 0.5, col = cores[iris$Species],
+       lower.panel=NULL)
>
```

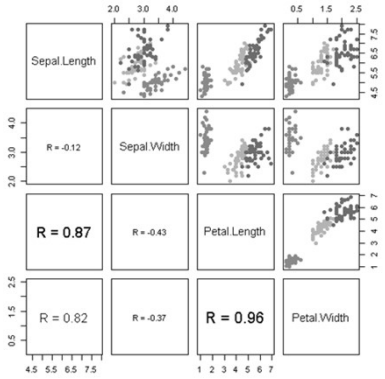


√ Personalização:

- Inferior: texto proporcional à correlação
- Superior: Cores de acordo com `Species`

```
> # -----  plot com correlações nos painéis inferiores
>
>   # Valor da correlação em painéis
> panel.cor <- function(x, y){
+     usr <- par("usr"); on.exit(par(usr))
+     par(usr = c(0, 1, 0, 1))
+     r <- round(cor(x, y), digits = 2)
+     txt <- paste0("R = ", r)
+     cex.cor <- 0.8/strwidth(txt)
+     text(0.5, 0.5, txt, cex = cex.cor * r)
+ }
>
>   # Símbolos coloridos de acordo com Species
> upper.panel<-function(x, y){
+   points(x, y, pch = 19, col = cores[iris$Species])
+ }
>
>   # Scatterplots e textos proporcionais às correlações
> pairs(iris[,1:4],
+       lower.panel = panel.cor,
+       upper.panel = upper.panel)
>
```


- Gráfico final:

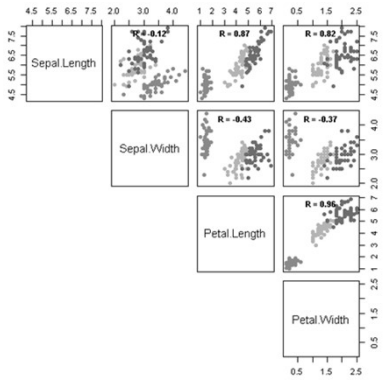


√ Personalização:

- Inferior: NULL
- Superior: Cores de acordo com Species e valores das correlações em cada *scatterplot*

```
> # -----  plot com correlações em cada scatter plot
>
>
>   # Customização do painel superior
> upper.panel<-function(x, y){
+   points(x, y, pch = 19, col = cores[iris$Species])
+   r <- round(cor(x, y), digits = 2)
+   txt <- paste0("R = ", r)
+   usr <- par("usr"); on.exit(par(usr))
+   par(usr = c(0, 1, 0, 1))
+   text(0.5, 0.9, txt, font = 2)
+ }
>
>   # Scatterplots com correlações
> pairs(iris[,1:4], lower.panel = NULL,
+       upper.panel = upper.panel)
>
```

– Gráfico final:



√ Função `coplot()` :

– Conjunto de dados: quakes{maps}

– `lat` e `long`, condicionados a `depth`

– Função painel:

▪ Adiciona mapas de Fiji e de N.Z. em cada saída

```
> #  Figura 3.28 - A panel function example
> library(maps)
> coplot(lat ~ long | depth, data = quakes, number = 4,
+        panel=function(x, y, ...) {
+          usr <- par("usr")
+          rect(usr[1], usr[3], usr[2], usr[4], col = "white")
+          map("world2", regions = c("New Zealand", "Fiji"),
+              add = TRUE, lwd = 0.1, fill = TRUE, col = "grey")
+          text(180, -13, "Fiji", adj = 1, cex = 0.7)
+          text(170, -35, "NZ", cex = 0.7)
+          points(x, y, pch = ".")
+        })
>
```


• Comentários sobre o código do painel:

√ Função `rect()`:

– Sobrepõe quadro branco e oculta linhas de grade default

√ Caractere muito pequeno

– `pch = "."`

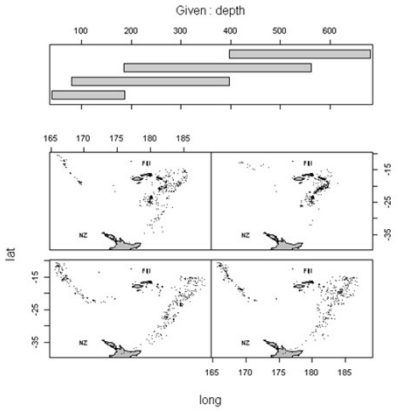
√ Função `map()`{`maps`}

– Desenha mapas de Fiji e da parte superior de North Island (N. Zelândia)

√ Função `text()`:

– Anota texto dos países

– Gráfico final:



- Gráficos em 3D:
  - √ Função `persp()`:
    - É possível anotar, mas é mais difícil
    - Necessário acessar a saída da matriz de transformação para transformar locações 3D em 2D.
      - Para uso em funções do tipo `lines()` ou `text()`
    - Possui argumento add
      - Permite a sobreposição de múltiplas superfícies

360

Estatística Computacional I - 2020

Referências

Bibliografia Recomendada

- ALBERT, J.; RIZZO, M. *R by Example*. Springer, 2012.
- CHRISTIAN, N. *Basic Programming*, Lecture Notes
- DALGAARD, P. *Introductory statistics with R*. Springer, 2008.
- MURRELL, P. *R Graphics*. Chapman & Hall, 2006.

393

Estatística Computacional I - 2020