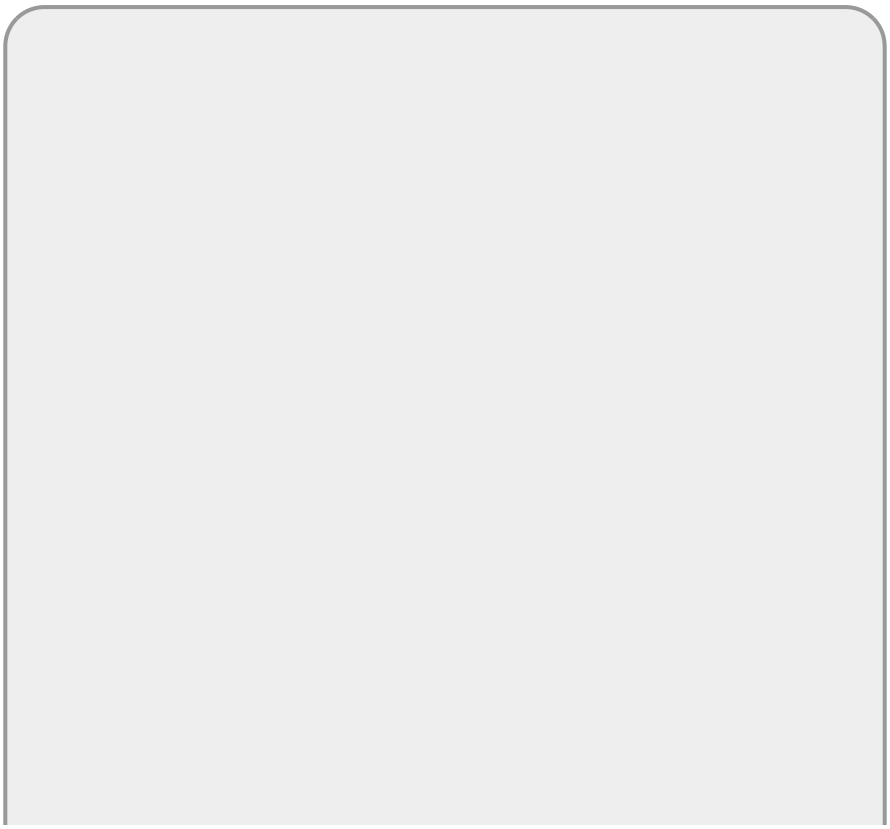
- Tutorial
- Exercícios
- Apostila
- Apostila-Avançado

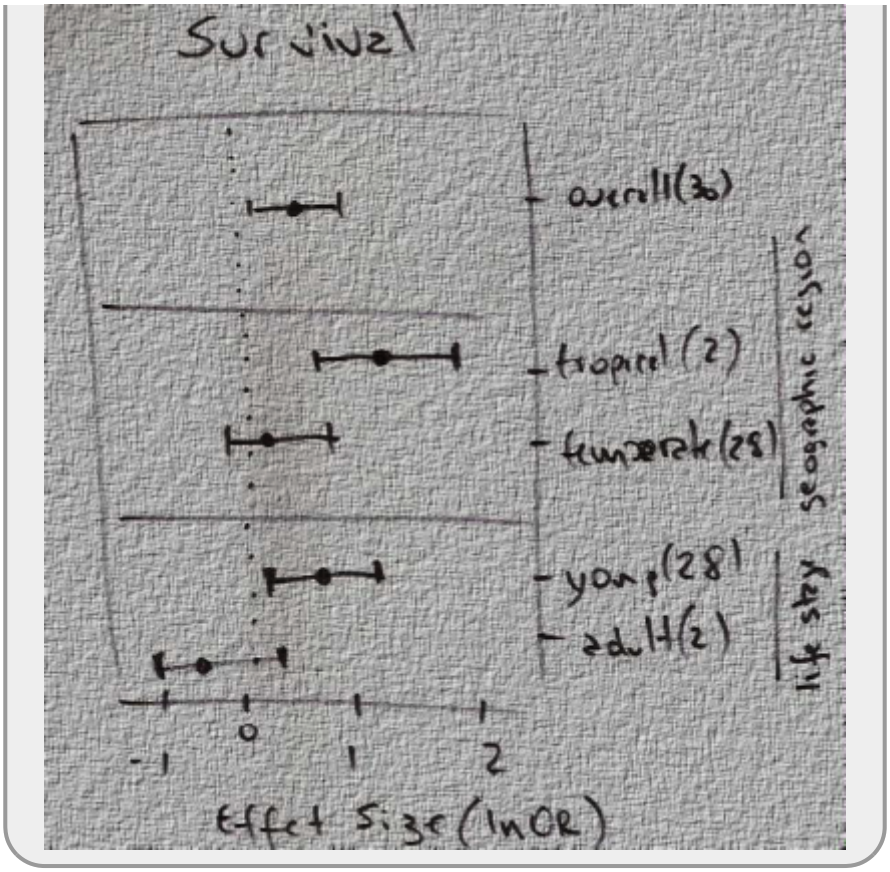
5b. Gráficos Avançados

Existem várias bibliotecas publicadas no repositório do R para auxiliá-los na construção de gráficos[1)]. Esses pacotes usam um dos sistemas de maquinaria básica do R para a construção gráfica, os pacotes: ***graphics*** ou ***grid***. Por exemplo, os pacotes *lattice* e *ggplot2* usam o sistema ***grid***, enquanto o pacote *maps* o sistema ***graphics***. Nesse capítulo utilizaremos apenas as funções básicas já instaladas e carregadas por padrão na sessão R, providas pelo sistema ***graphics***, em outro capítulos abordaremos o sistema ***grid***. Nesse capítulo utilizaremos um exemplo para ilustrar um procedimento básico para a elaboração de gráficos mais complexos.

Procedimento

A parte mais importante e difícil da elaboração de uma boa representação gráfica está na sua concepção, que precede o procedimento que iremos descrever nesse capítulo. Partiremos da premissa que já foi decidido qual o melhor gráfico e como os elementos que o compõem estarão distribuídos e organizados. Uma boa dica é construir um esquema do gráfico a mão antes de iniciar o código para montá-lo no R. No nosso caso, desenhamos, em um guardanapo de papel na mesa de um bar, um esboço do gráfico que iremos construir nesse capítulo. Os dados provém do resultado de uma metanálise que já havíamos explorados usando técnicas que apresentamos no capítulo de Análise Exploratória de Dados.





O gráfico que esboçamos, apesar de aparentemente simples, apresenta desafios na sua construção e será utilizado para apresentar o procedimento padrão que utilizamos para a elaboração de gráficos mais avançados, baseado apenas nas funções básicas do R.

Layout

A primeira parte na construção do gráfico é definir seu layout. Aprendemos no capítulo introdutório de gráficos que podemos dividir a janela gráfica modificando os parâmetros *mfrow* ou *mfcol* do dispositivo gráfico, através da função `par()`. Uma forma mais versátil e flexível de dividir a janela gráfica é utilizar a função `layout()`. O primeiro argumento dessa função é uma matriz com o número de linhas e colunas correspondente às divisões do dispositivo. Os valores nessa matriz correspondem à ordem que cada área do dispositivo será plotada. Veja um exemplo dividindo o dispositivo gráfico em nove áreas:

```
mtplot <- matrix(1:9, ncol=3, nrow=3, byrow=TRUE)
layout(mtplot)
layout.show(9)
```

Além disso, é possível controlar a proporção que cada linha e coluna terá no layout final utilizando os argumentos `widths` e `heights`.

```
mtplot <- matrix(1:9, ncol=3, nrow=3, byrow=TRUE)
layout(mtplot, widths=c(0.1, 0.6, 0.4), heights=c(1,4,1))
layout.show(9)
```

Note que os valores referentes ao tamanho das colunas ou linhas não necessita somar um; eles significam apenas uma proporção relativa.

Exercício

- Formate um layout com:
  - seis áreas de plotagem, três colunas e duas linhas
  - a coluna central é o dobro de largura das outras duas
  - a linha inicial e cinco vezes menor que a outra linha

O nosso gráfico tem um layout bem simples. Uma região principal onde vamos fazer o gráfico propriamente dito e uma coluna mais estreita onde vamos colocar a legenda da borda direita (“life stage” e “geographic region”) e as linhas associadas.

```
layout (matrix(c(1,2),ncol=2, nrow=1), width=c(8,2))
layout.show(2)
```

Região Principal

Antes de plotar algo na primeira área do gráfico, vamos primeiro ajustar as bordas usando a função `par()` e seu argumento `mar`. Em seguida iniciamos o gráfico sem nenhum valor, apenas definindo as coordenadas da região gráfica. Lembre-se, queremos colocar cada elemento do gráfico separadamente e ter controle total sobre o que será desenhado. Os valores do eixo *x* correspondem ao intervalo dos dados e do eixo *y* foi definido pelo número de elementos que serão desenhados e o espaçamento entre eles. Nesse último caso a escala é arbitrária. Para definir tanto o espaçamento das bordas quanto a escala de *y* utilizamos um processo iterativo de ajuste, plotando o gráfico e reajustando os valores.

```
par(mar=c(5,4,4,3.5))
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="", main="SURVIVAL")
```

Agora vamos colocar as legendas e os eixos do lado direito do gráficos usando a função `axis()` e as linhas vertical e horizontais utilizando a função `abline()`:

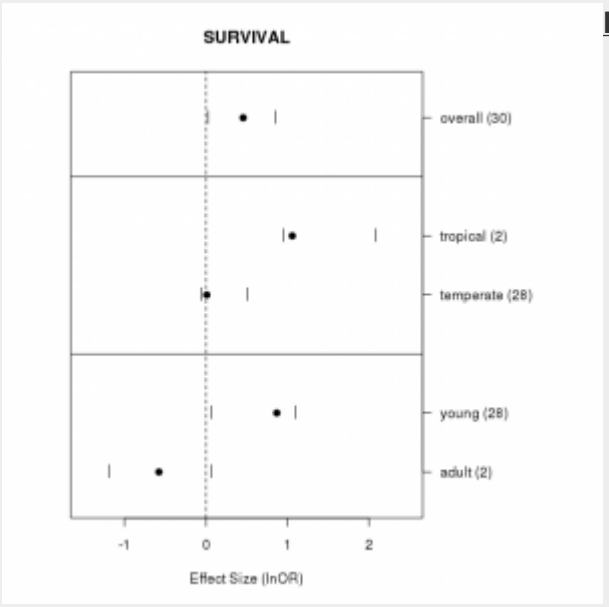
```
axis(side=4, at=c(1,2,4,5,7), labels=c("adult (2)", "young (28)",
                  "temperate (28)", "tropical (2)", "overall (30)"),
                  las=2 )
abline (v=0, lty=2)
abline (h=c(3,6))
```

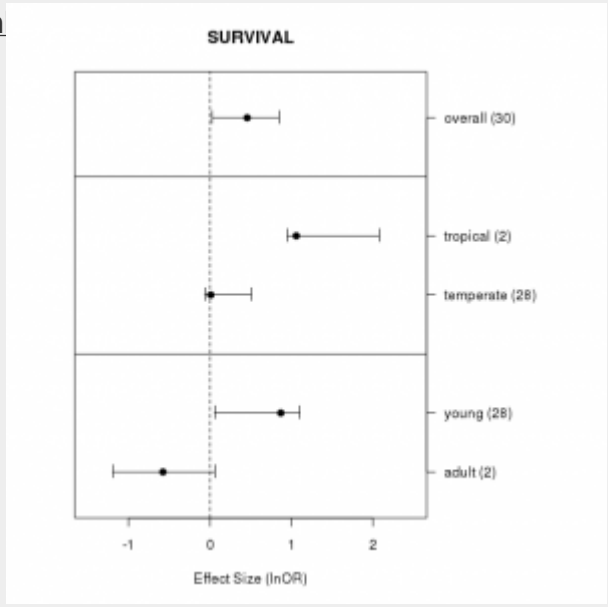
Agora vamos colocar os pontos e as barras referentes aos valores médios e intervalo de confiança usando a função `points()` e os símbolos `pch=19` (circulo preenchido) e `pch="_"` para a barra [2]. Em

seguida desenhamos as linhas usando a função `segments()` que tem como argumento as coordenadas do início (*x0* e *y0*) e do final da linha (*x1* e *y1*). Note que poderíamos desenhar isoladamente cada ponto e cada segmento, mas fazemos todos juntos concatenando as coordenadas em vetores em que as posições são equivalentes.

```
points (x=c(-0.577, 0.87, 0.01, 1.06,0.457),
                y=c(1,2, 4, 5, 7), pch=19 ) #medias
points (x=c(-1.2, 0.05, 0.05, 1.1, -0.07, 0.5, .946,2.073, 0.025,0.847),
              y=rep(c(1,2, 4, 5, 7),each=2), pch= "|" )
segments(x0=c(-1.2,  0.05, -0.07, .946, 0.025),
         y0=c(1,2, 4, 5, 7), x1=c( 0.05, 1.1, 0.5,2.073,0.847),
         y1=c(1,2, 4, 5, 7))
```

Desen





Segunda região gráfica

Vamos iniciar a segunda região gráfica da mesma maneira que a primeira, ajustando as margens e definindo as coordenadas cartesianas da região que nesse caso são totalmente arbitrárias, mas devem ter correspondência com o eixo *y* da primeira área gráfica. Note os argumentos do `plot()` nesse caso são necessários para que nada seja desenhado na região, e estamos apenas definindo a escala das coordenadas *x* e *y*.

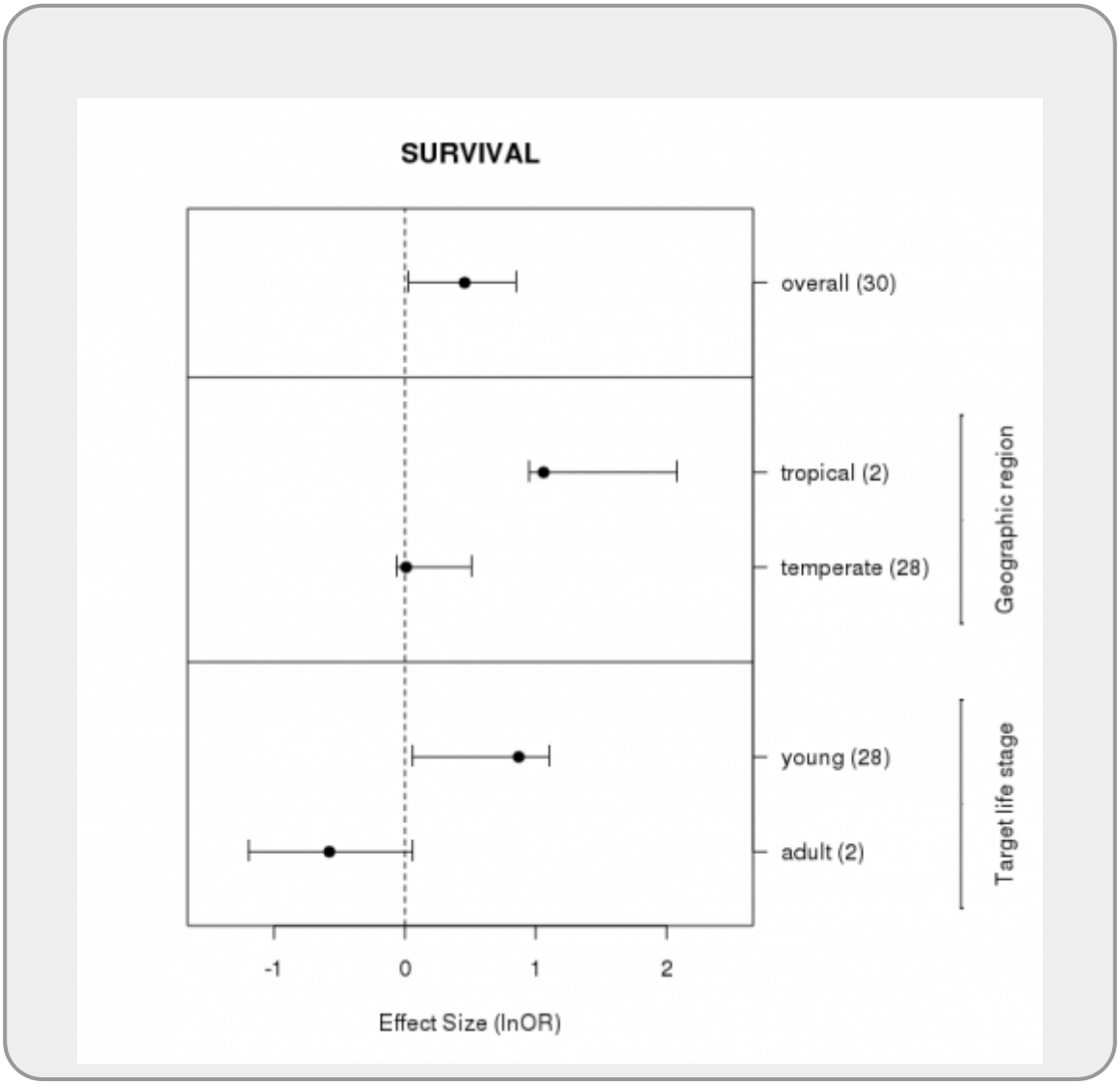
```
par (mar=c(5,4,4,2.9))
plot(x=NULL,y=NULL, xlim=c(0,2), ylim=c(0.5,7.5),type="n",
     xaxt="n", yaxt="n",xlab="", ylab="", bty="n")
```

- `type` tipo de gráfico, no caso “n” significa “não plotar valores”
- `xaxt` e `yaxt`: tipo de eixo “n” significa “não desenhar o eixo”.
- `xlim` e `ylim`: define a escala dos eixos e deve ter os valores iniciais e finais concatenados
- `xlab` e `ylab`: legendas dos eixos
- `bty`: tipo de caixa ao redor do gráfico

Agora só precisamos colocar os segmentos e as legendas para finalizar o nosso gráfico:

```
points(x=rep(c(0.5),4), y=c(0.4, 2.6, 3.4, 5.6), pch="-")
segments(x0=c(0.5, 0.5), y0=c(0.4, 3.4), x1=c(0.5,0.5), y1=c(2.6, 5.6))
axis(side=4, at=1.5, labels= "Target life stage", lwd.ticks=0)
axis(side=4, at=4.5, labels= "Geographic region", lwd.ticks=0)
```

Pronto, nosso gráfico planejado na mesa do bar e executado no R!



Salvando o Gráfico

Explicamos os procedimentos gerais para salvar gráfico no capítulo anterior, no tópico Salvando Gráficos. Vamos recordar alguns pontos importantes. O nosso gráfico foi construído no dispositivo de tela e pode ser salvo na resolução padrão com a função `savePlot()` que permite salvar em formatos *jpg*, *png*, *tiff* e *bmp*. Veja o help da função para mais informações. Para salvar uma cópia com resolução e tamanho definidos pelo usuário é necessário usar o dispositivo de arquivo. O R tem vários dispositivos de arquivos que são abertos com as funções com o nome do formato do arquivo: `bmp()`, `jpeg()`, `png()` e `tiff()`.

O procedimento para usar o dispositivo de arquivo é o seguinte:

- abra o dispositivo usando a função correspondente, nomeando o arquivo que será construído e definindo os parâmetros de tamanho da imagem e unidade a ser utilizada. Por exemplo: `width = 480, height = 480, units = "px"`
- faça o gráfico utilizando o código construído anteriormente no dispositivo de janela. Note que não irá aparecer o gráfico pois ele está sendo montado diretamente no arquivo nomeado
- ao finalizar o gráfico, feche o dispositivo gráfico que abriu utilizando a função `dev.off()`. Nesse momento o arquivo é fechado e gravado no seu diretório de trabalho `getwd()`.

Veja exemplo abaixo para a construção de um arquivo *png* com fundo transparente:

```
png(filename = "metaPlot.png",
     width = 900, height = 900, units = "px", pointsize = 12,
      bg = "transparent")

      ...# início do código do gráfico
      ...
      ...# fim do código do gráfico

dev.off()
```

1)

veja o taskview http://cran.r-project.org/web/views/Graphics.html

2)

veja o help e o exemplo da função points para os códigos dos símbolos