Uma breve introdução ao R para Epidemiologia

Este material é uma tradução para o português de trechos do texto:
"**A short Introduction to R for Epidemiology**"

Michael Hills
Highgate, London

Martyn Plummer
International Agency for Research on Cancer, Lyon
plummer@iarc.fr

Bendix Carstensen
Steno Diabetes Center, Gentofte, Denmark
& Department of Biostatistics, University of Copenhagen
bxc@steno.dk
www.pubhealth.ku.dk/~bxc

Edition 2014 by Bendix Carstensen

Executando o R no seu computador

O que é R?

- R é um programa **gratuito** e de **código aberto** para análise de dados e gráficos. Ele contém todos os métodos estatísticos de última geração e tornou-se a ferramenta de análise preferida para a maioria dos estatísticos profissionais do mundo. Ele pode ser usado como uma calculadora simples e como um dispositivo de análise estatística muito especializado.
- O **aspecto especial** do R é que você insere comandos do teclado em uma janela do console, onde também vê os resultados. Essa é uma vantagem, pois você tem um script que pode ser usado para reproduzir suas análises - um requisito em qualquer empreendimento científico.
- A **desvantagem** é que você precisa descobrir de alguma forma o que digitar. Os exercícios mostrarão algumas dicas e, na maioria das vezes, você usará R como calculadora - digite uma expressão, pressione a tecla Enter e obtenha o resultado.

Obtendo R

- Você pode obter R, que é gratuito, no CRAN (a Rede de Arquivamento Compreensivo de R), em **http://cran.r-project.org/** Em "Download R for Windows", clique em "instalar R pela primeira vez" e depois em "Download R 3.6.1 for Windows", que é um instalador de extração automática. Isso significa que, se você salvá-lo em seu computador em algum lugar e clicar nele, ele instalará o R para você.
- Além do que arquivo você baixou, existem milhares de pacotes adicionais para o R, que lidam com todos os tipos de problemas, desde ecologia até finanças e, incidentalmente, à epidemiologia.
- Você deve fazer o download manualmente. Neste curso, precisaremos apenas do pacote **Epi**.

Iniciando R

- Você inicia o R clicando no ícone que o instalador colocou na sua área de trabalho.
- Depois de instalar o R, inicie-o e, na barra de menus, clique em **Pacotes → Instalar pacote(s)**..., escolha um espelho (este é apenas um servidor de onde você pode obter as coisas) e, em seguida, o pacote **Epi**.
- Depois que R lhe disser que foi instalado, você pode digitar:

```
> library(Epi)
```

  para iniciar o pacote Epi.

- Para obter um resumo das funções a conjuntos de dados no pacote:

```
> library(help=Epi)
```

Saindo do R

- Digite `q()` no console e responda “Não” quando for perguntado se deseja salvar a imagem da área de trabalho.

Trabalhando com o Editor de Script

- Se você clicar em **Arquivo → Novo script**, o R abrirá uma janela para você, que é um editor de texto muito parecido com o **Bloco de Notas**.
- Se você escrever um comando nele, poderá transferi-lo para o console R e **executá-lo** pressionando CTRL-r.
- Se nada estiver destacado, a linha onde está o cursor será transmitida ao console e o cursor se moverá para a próxima linha.
- Se uma parte da tela estiver destacada, a parte destacada será transmitida ao console.
- O destaque também pode ser usado para transmitir apenas uma parte de uma linha de código.

Experimente!

- Agora, abra um script em Arquivo → Novo script e digite (omita o ">" no início da linha)

  5+7

  pi

  1:10

  N <- c(27,33,81)

  N

- Execute as linhas, uma por uma, teclando CTRL-r
- Você também pode digitar os comandos diretamente no console. Se teclar **Arquivo → Salvar histórico** salvará tudo o que digitou no console.

Comandos básicos no R

- Para iniciar o **R**, clique no ícone 
- Para alterar seu diretório de trabalho, clique em ***Arquivo*** → ***Alterar dir ...*** e selecione o diretório em que deseja trabalhar.
- Como alternativa, você pode escrever:

  > setwd ("C:/onde/todos/meus/arquivos/estão")

- Para sair do **R**, clique no menu ***Arquivo*** e selecione ***Sair***, ou simplesmente, digite "***q()***".
- Você terá a chance de salvar o espaço de trabalho, mas, nesse estágio, saia sem salvar, inicie o **R** novamente e altere o diretório de trabalho, como antes.

Comandos básicos no R

- **R** diferencia maiúsculas de minúsculas, de modo que **A** é diferente de **a**.
- Os comandos em **R** são geralmente separados por uma nova linha, mas ponto e vírgula (**;**) também pode ser usado.
- Redigitação de comandos podem ser evitados: recupere os comandos anteriores usando a tecla de seta vertical (↑) e editando-os.

Usando R como calculadora

> *2+2*

> *2^3*

> *log(10)*

> *sqrt(25)*

- O resultado da operação pode ser armazenado num objeto

> *a <- 2+2*

> *2+2->a*

Objetos e Funções

- Todos os comandos em **R** são **funções** que atuam em **objetos**.
- Um tipo importante de objeto é um **vetor**, que é uma **coleção ordenada de números** ou uma **coleção ordenada de caracteres**.
- Exemplos de vetores são:
  - (4, 6, 1, 2.2), que é um vetor numérico com 4 componentes, e
  - ("Charles Darwin", "Alfred Wallace"), que é um vetor de caracteres com 2 componentes.
- A função de combinação ***c()***, juntamente com o operador de atribuição, é usada para criar vetores.

```
> v <- c(4,6,1,2.2)
```

```
> x<-c("Charles Darwin", "Alfred Wallace")
```

- Os componentes de um vetor devem ser do mesmo tipo (numéricos ou caracteres).

Objetos e Funções

- Coleções de componentes de diferentes tipos são chamadas de **listas** e são criadas com a função `list()`.

```
> m <- list(4,6,"name of company")
```

- Listas de vetores de mesmo comprimento são chamados de **data frames** e são criadas com a função `data.frame()`.

```
> n = c(2, 3, 5)
> s = c("aa", "bb", "cc")
> b = c(TRUE, FALSE, TRUE)
> df = data.frame(n, s, b)
```

df é um data frame contendo três vetores n, s, b.

Objetos e Funções

- Uma descrição da estrutura de qualquer objeto pode ser obtida usando a função **`str()`**.

```
> str(v)
num [1:4] 4 6 1 2.2
```

- Mostra que v é numérico com 4 componentes.
- Operações matemáticas podem ser feitas usando o objeto criado

```
> 3+v
> 3*v
```

Sequências

- Nem sempre é necessário digitar todos os componentes de um vetor.
- O vetor (15, 20, 25, ... ,85) pode ser criado com

```
> seq(15, 85, by=5)
```

- O vetor (5, 20, 25, ... ,85) pode ser criado com

```
> c(5,seq(20, 85, by=5))
```

Ajuda

- Para mais informações sobre funções digite **?** seguido do nome da função.

```
> ?seq
```

- fornece informação sobre a sintaxe e uso da função **seq()**.

Exercícios

1) Crie um vetor w com componentes 1, -1, 2, -2
2) Mostre esse vetor (na tela)
3) Obtenha uma descrição de w usando `str()`
4) Crie o vetor w+1, e mostre na tela
5) Crie o vetor (0, 1, 5, 10, 15, ... , 75) usando `c()` e `seq()`.

Respostas

```
> w<-c(1, -1, 2, -2)
> w
> str(w)
> w+1
>
c(0,1,5,10,15,20,25,30,35,40,45,50,55,60,65
,70,75)
> c(0,1,seq(5,75,by=5))
```

Conjunto de dados

Variable	Units or Coding	Type	Name
Subject number	–	categorical	id
Birth weight	grams	metric	bweight
Birth weight < 2500 g	1=yes, 0=no	categorical	lowbw
Gestational age	weeks	metric	gestwks
Gestational age < 37 weeks	1=yes, 0=no	categorical	preterm
Maternal age	years	metric	matage
Maternal hypertension	1=hypertensive, 0=normal	categorical	hyp
Sex of baby	1=male, 2=female	categorical	sex

Births Data

- O exemplo mais importante de um vetor em epidemiologia são dados de uma variável registrados para um grupo de indivíduos.
- Para introduzir o R usamos dados de nascimentos de 500 mães que tiveram partos únicos num grande hospital de Londres.
- Estes dados estão disponíveis como um objeto R chamado `births` no pacote `Epi`. Você pode inserí-los nos seu espaço de trabalho digitando:

```
> install.packages("Epi")
> library(Epi)
> data(births)
```

- A função

```
> str(births)
'data.frame': 500 obs. of  8 variables:
 $ id     : num  1 2 3 4 5 6 7 8 9 10 ...
 $ bweight: num  2974 3270 2620 3751 3200 ...
 $ lowbw  : num  0 0 0 0 0 0 0 0 0 0 ...
 $ gestwks: num  38.5 NA 38.2 39.8 38.9 ...
 $ preterm: num  0 NA 0 0 0 0 0 0 0 0 ...
 $ matage : num  34 30 35 31 33 33 29 37 36 39 ...
 $ hyp    : num  0 0 0 0 1 0 0 0 0 0 ...
 $ sex    : num  2 1 2 1 1 2 2 1 2 1 ...
```

  mostra que o objeto `births` é um data frame com 500 observações de 8 variáveis

- Os nomes e tipos de variáveis também são mostradas juntos com os primeiros 10 valores de cada variável

Exercícios

1) O data frame "diet" no pacote Epi contém dados de um estudo longitudinal com doença coronariana como desfecho. Carregue estes dados com:

```
> data(diet)
```

e imprima o conteúdo do data frame na tela.

2) Verifique se agora existem dois objetos - births e diet – na sua área de trabalho

3) Obter uma descrição do objeto diet.

4) Remove the object diet with the command

```
> rm(diet)
```

5) Verifique se restou somente o objeto births

Referenciando partes do data frame

- Digitando births retornará uma listagem do data frame inteiro – em geral não muito útil. Agora experimente

```
> births[1,"bweight"]
```

- Será mostrado o valor observado do primeiro indivíduo para a variável bweight.

```
> births[2,"bweight"]
```

- Mostrará o valor da variável bweight para o segundo indivíduo, e assim em diante.
- Para mostrar os dados dos primeiros 10 indivíduos para a variável bweight, experimente

```
> births[1:10,"bweight"]
```

  e para listar todos os dados para esta variável

```
> births[,"bweight"]
```

Exercícios

1) Imprima na tela o dado da variável `gestwks` para o indivíduo 7 do data frame `births`.
2) Imprima todos os dados do indivíduo 7.
3) Imprima todos os dados da variável `gestwks`.

Respostas

```
> births[7,"gestwks"]
> births[7,]
> births[,"gestwks"]
```

Resumos

- Uma boa maneira de iniciar uma análise é fazer um resumo dos dados

```
> summary(births)
```

- Para ver os nomes das variáveis no data frame

```
> names(births)
```

- Variáveis no data frame pode ser referidas pelo nome, mas para fazer isso também é necessário especificar o nome do data frame. Então

```
> births$gestwks
```

  refere-se à variável gestwks do data frame births.

- Para resumir a variável gestwks experimente

```
> summary(births$gestwks)
```

- Na maioria dos conjuntos de dados existirão alguns valores perdidos. Estes são em geral codificados usando espaços em branco para marcar os valores perdidos. **R** então codifica os valores perdidos usando o símbolo **NA (not available)**. O resumo mostra o número de valores perdidos para cada variável.

Convertendo uma variável num fator

- No R **variáveis categóricas** são conhecidas como **fatores**, e as diferentes **categorias** são chamadas de **níveis do fator**.
- Variáveis tais como hyp e sex foram originalmente codificadas usando códigos inteiros, e por default R interpretará estes códigos como valores numéricos.
- Para R reconhecer que códigos referem-se a categorias é necessário converter as variáveis para fatores e rotular os níveis.

```
> hyp <- factor(births$hyp)
> str(births)
> objects()
> str(hyp)
```

- Isso mostra que hyp está no seu espaço de trabalho (como um fator), e no data frame births (como uma variável numérica). **Confusão!!!**

Convertendo uma variável num fator

- É melhor usar a função de transformação no data frame

```
> births <- transform(births, hyp=factor(hyp))
```

```
> str(births)
```

- Veja que hyp, no data frame births, agora é um fator com dois níveis, rotulados "0" e "1" que são os valores originais da variável.
- É possível mudar os rótulos para "normal" e "hyper"

```
> births<-
transform(births,hyp=factor(hyp,labels=c("normal","hyper")))
```

```
> str(births)
```

Exercícios

1) Converta a variável sex num fator
2) Rotule os níveis do fator sex como "male" e "female".

Respostas

```
> library(Epi)
> data(births)
> births <- transform(births,
sex=factor(sex,labels=c("male","female")) )
```

Tabelas de frequência

- Ao começar a examinar qualquer novo data frame, o primeiro passo é verificar se os valores das variáveis fazem sentido e correspondem aos códigos definidos na programação de codificação.
- Para variáveis categóricas (fatores), isso pode ser feito observando tabelas de frequência unidimensionais e verificando se apenas os códigos (níveis) especificados ocorrem.
- A distribuição dos fatores hip e sexo pode ser visualizada digitando-se

```
> table(births$hyp)
> table(births$sex)
```

- A tabulação cruzada é obtida digitando-se

```
> table(births$hyp,births$sex)
```

Tabelas de frequência no Epi

- A função mais útil para criar tabelas é `stat.table` do pacote `Epi`
- Você precisará carregar este pacote primeiro

```
> library(Epi)
```

- A distribuição dos fatores hip e sexo pode ser visualizada digitando-se

```
> stat.table(hyp,data=births)
> stat.table(sex,data=births)
```

- A tabulação cruzada é obtida digitando-se

```
> stat.table(list(hyp,sex),data=births)
```

Agrupando os valores de uma variável numérica

- Para uma variável numérica como **matage**, é útil agrupar os valores e criar um novo fator que codifique os grupos.
- Por exemplo, podemos cortar os valores obtidos por maturidade nos grupos 20-29, 30-34, 35-39, 40-44 e, em seguida, criar um fator chamado **agegrp** com 4 níveis correspondentes aos quatro grupos.

```
> births <- transform(births,agegrp=cut(matage,
breaks=c(20,30,35,40,45),right=FALSE))

> stat.table(agegrp,data=births)
```

Agrupando os valores de uma variável numérica

- Por padrão, os níveis de fator são rotulados [20-25), [25-30), etc., onde [20-25) se refere ao intervalo que inclui a extremidade esquerda (20), mas não a extremidade direita (25). Esta é a razão para **`right = FALSE`**. Quando **`right = TRUE`** (que é o padrão), os intervalos incluem a extremidade direita, mas não a esquerda.
- É importante perceber que as observações que não estão dentro do intervalo especificado na parte `breaks()` do comando resultam em valores faltantes para o novo fator.

```
> births <- transform(births,agegrp=cut(matage,
breaks=c(20,30,35),right=FALSE))
```

```
> summary(births)
```

- Apenas observações de 20 até, mas não incluindo 35, estão incluídas. Quanto ao restante, o agegrp está codificado como ausente.
- Você pode especificar que deseja cortar uma variável em um determinado número de intervalos de igual comprimento, especificando o número de intervalos.

```
> births <- transform(births,agegrp=cut(matage,breaks=5,right=FALSE))
```

```
> stat.table(agegrp,data=births)
```

resulta em 5 intervalos de comprimento 4.

Exercícios

1) Resuma a variável numérica `gestwks`, que registra a duração da gestação do bebê e anote o intervalo de valores.
2) Crie um novo fator `gest4` que corte `gestwks` em 20, 35, 37, 39 e 45 semanas, incluindo a extremidade esquerda, mas não a direita. Faça uma tabela das frequências para os quatro níveis de `gest4`.
3) Crie um novo fator `gest5` que corte `gestwks` em 5 intervalos iguais e faça uma tabela de frequências.

Respostas

```
> summary(births$gestwks)

> births <- transform(births,
gest4=cut(gestwks,
breaks=c(20,35,37,39,45),right=FALSE))

> stat.table(gest4,data=births)

> births <- transform(births,
gest5=cut(gestwks, breaks=5,right=FALSE))

> stat.table(gest5,data=births)
```

Tabelas de médias e outras coisas

- Para obter a média de bweight por sex

```
> by(births$bweight,births$sex,mean) #base R
```

```
> stat.table(sex, mean(bweight), data=births) #Epi
```

- O cabeçalho da tabela pode ser melhorado com

```
> stat.table(sex,list("Mean birth
weight"=mean(bweight)),data=births)
```

- Para fazer uma tabela 2x2 do peso médio ao nascer por sex e hyp

```
> stat.table(list(sex,hyp),mean(bweight),data=births)
```

e tabular a contagem bem como as médias

```
> stat.table(list(sex,hyp),list(count(),mean(bweight)),data=births)
```

Tabelas de médias e outras coisas

- As funções disponíveis para as células da tabela são **`count`**, **`mean`**, **`weighted.mean`**, **`sum`**,**`min`**, **`max`**, **`quantil`**, **`mediana`**, **`IQR`** e **`ratio`**.
- O último deles é útil para taxas e odds. Por exemplo, para fazer uma tabela das chances de baixo peso ao nascer por hipertensão

```
> stat.table(hyp, list("odds"=ratio(lowbw,1-lowbw,100)),data=births)
```

- O fator de escala 100 faz as odds por 100.
- As margens podem ser adicionadas às tabelas, conforme necessário.

```
> stat.table(sex, mean(bweight),data=births,margins=TRUE)
```

- Para uma tabela de 1 entrada, e

```
> stat.table(list(sex,hyp),mean(bweight),data=births,margins=c(TRUE,FALSE))
> stat.table(list(sex,hyp), mean(bweight),data=births,margins=c(FALSE,TRUE))
> stat.table(list(sex,hyp), mean(bweight),data=births,margins=c(TRUE,TRUE))
```

para uma tabela de dupla entrada.

Exercícios

1) Faça uma tabela de peso médio ao nascer por sex.
2) Faça o mesmo para o tempo de gestação, mas inclua `count` como uma função a ser tabulada junto com a mediana. Observe que, quando há valores ausentes para a variável que está sendo resumida, a contagem se refere ao número de observações não ausentes para a variável de linha, não à variável resumida.
3) Crie uma tabela mostrando o tempo médio de gestação por hyp e lowbw, juntamente com margens para ambos.
4) Faça uma tabela mostrando as odds de hipertensão por sexo do bebê.

Respostas

```
> library(Epi)
> data(births)
> births <- transform(births,
sex=factor(sex,labels=c("male","female")) )
> stat.table(sex, list("mean birth
weight"=mean(bweight)), data=births)
> stat.table(sex, list("Freq"=count(),"mean gest
weeks"=mean(gestwks)), data=births)
> stat.table(list(sex,lowbw),
median(gestwks),data=births,margins=c(TRUE,TRUE))
> stat.table(sex, list("odds"=ratio(hyp,1-
hyp,100)),data=births)
```

Outras funções de tabulação

- Dê uma olhada nas páginas de Help das funções:
  - `table`
  - `ftable`
  - `xtabs`
  - `addmargins`
  - `array`
  - `tapply`
- Uma maneira de entender seu uso é usando o comando:

```
> example(table)
```

Gerando novas variáveis

- Novas variáveis podem ser produzidas usando atribuição juntamente com as operações e funções matemáticas usuais: `+ - * log exp ^ sqrt`
- O sinal ^ significa "ao poder de", log significa "logaritmo natural" e sqrt significa "raiz quadrada".
- A função `transform()` permite transformar ou gerar variáveis em um data frame.

```
> births <- transform(births,num1=1,num2=2,
logbw=log(bweight))
```

- A variável logbw é o logaritmo natural do peso ao nascer. Os logs base 10 são obtidos com `log10()`.

Variáveis Lógicas

- Variáveis lógicas recebem os valores TRUE ou FALSE e se comportam como fatores. Novas variáveis podem ser criadas, que são funções lógicas das variáveis existentes.

```
> births <- transform(births, low=bweight<2000)
> str(births)
```

  cria uma variável lógica low com os níveis TRUE e FALSE, de acordo com o peso ser menor que 2000 ou não.

- As expressões lógicas permitidas no R são

  == < <= > >= !=

- Um uso comum de variáveis lógicas é para restringir um comando a um subconjuntos dos dados.
- Por exemplo, para listas os valores da variável bweight para mulheres hipetensas:

```
> births$bweight[births$hyp=="hyper"]
```

- Data frame restrito às mulheres hipertensas:

```
> births[births$hyp=="hyper",]
```

- A função `subset()` também permite selecionar um subconjunto do data frame.

```
> subset(births, hyp=="hyper")
```

Exercícios

1) Crie uma variável lógica chamada `early`, dependendo se gestwks for menor que 30 ou não. Crie uma tabela de frequências de `early`.
2) Imprima o número de identificação de mulheres com gestwks com menos de 30 semanas.

Respostas

```
> births <- transform(births, early=gestwks<30)
> stat.table(early,data=births)
> births$id[births$early==TRUE]
```

Salvando o espaço de trabalho

- Ao sair do R, você terá a chance de salvar todos os objetos no seu espaço de trabalho
- Se você fizer isso, o espaço de trabalho será restabelecido na próxima vez que você iniciar o R.
- Pode ser útil, mas antes disso, vale a pena arrumar as coisas, porque o espaço de trabalho pode ser preenchido com objetos temporários e é fácil esquecer o que são quando você retoma a sessão.

Salvando saída em um arquivo

- Para salvar a saída de um comando R em um arquivo, para uso futuro, o comando `sink()` é usado. Por exemplo,

```
> sink("output.txt",append=TRUE)
> summary(births)
```

- primeiro instrui R para redirecionar a saída do terminal R para o arquivo "output.txt" e, em seguida, resume o data frame `births`, cuja saída vai para o output. A opção `append = TRUE` é usada para acrescentar linhas ao arquivo. Para fechar o arquivo, use

```
> sink()
```

Exercício

1) Envie a saída para um arquivo chamado "output1.txt"
2) Faça tabelas de frequência de hyp e sex
3) Faça uma tabela de peso médio ao nascer por sex
4) Feche o arquivo output1.txt
5) No Windows, dê uma olhada no arquivo output1.txt e verifique se ele contém a saída que você esperava

Respostas

```
> library(Epi)
> data(births)
> births <- transform(births,
hyp=factor(hyp,labels=c("normal","hyper")) )
> births <- transform(births,
sex=factor(sex,labels=c("male","female")) )
> sink("output1.txt",append=TRUE)
> stat.table(list(hyp, sex), data=births) #cross-
tabulation of the factors hyp and sex
> stat.table(sex, list("mean birth weight"=mean(bweight)),
data=births) #mean of bweight by sex
> sink()
```

Salvando objetos R num arquivo

- O comando `read.table()` é relativamente lento porque realiza bastante processamento enquanto lê os dados.
- Para evitar fazer isso mais de uma vez, você pode salvar o data frame, que inclui as informações de R, e ler esse arquivo salvo no futuro.

```
> save(births, file="births.Rdata")
```

  salvará o data frame `births` no arquivo `births.Rdata`.

- Por padrão, o data frame é salvo como um arquivo binário, mas a opção `ascii = TRUE` pode ser usada para salvá-lo como um arquivo texto.
- Para carregar o objeto do arquivo, use

```
> load("births.Rdata")
```

Exercício

1) Use `read.table()` para ler os dados do arquivo `diet.txt` num data frame chamado `diet`.
2) Salve este data frame num arquivo "`diet.Rdata`"
3) Remova o data frame
4) Carregue o data frame a partir do arquivo "`diet.Rdata`"

Respostas

```
diet<-read.table('diet.txt',sep=';',header=TRUE)
save(diet,file='diet.RData')
rm(diet)
load('diet.RData')
```

Usando um editor de texto com R

- Ao trabalhar com R, é melhor usar um editor de texto para preparar um script com comandos R e, em seguida, executá-los a partir do script.
- Para Windows, recomendamos o uso do editor de texto Tinn-R, mas você pode usar o seu editor de texto favorito, se preferir, e copiar e colar comandos dele no R-console.
- Como alternativa, você pode usar o editor de script do R: Clique em `Arquivo → Novo script` ou `Arquivo → Abrir script`, e usar um script antigo já salvo no seu computador. Você pode executar linha a linha do script no console usando `CTRL-R`. Se você destacou uma seção do script, a parte destacada será executada no console.
- Agora inicie o editor e adicione as linhas abaixo:

```
> births <- transform( births, lowbw = factor(lowbw,
labels=c("normal","low")), hyp = factor(hyp, labels=c("normal","hyper")),sex =
factor(sex, labels=c("male","female")) )
```

- Agora salve o script como mygetbirths.R e execute-o.
- Uma **grande vantagem** de executar todos os seus comandos R a partir de um script é que você termina com um registro exato do que fez, que **pode ser repetido** a qualquer momento.
- Isso também o ajudará a **refazer a análise** no evento (altamente provável) de que seus dados sejam alterados **antes da conclusão** de todas as análises.

Exercícios

1) Crie um script chamado `mytab.R` que inclua as linhas abaixo e execute apenas essas duas linhas.

```
> stat.table(hyp,data=births)
> stat.table(sex,data=births)
```

2) Edite o script para incluir as linhas

```
> stat.table(sex,mean(bweight),data=births)
> stat.table(hyp,mean(bweight),data=births)
```

e execute essas duas linhas.

3) Edite o script para criar um fator cortando `matage` em 20, 30, 35, 40, 45 anos e execute apenas essa parte do script.

4) Edite o script para criar um fator que corte `gestwks` em 20, 35, 37, 39, 45 semanas e execute apenas esta parte do script.

5) Salve e execute o script inteiro.

O caminho da pesquisa

- R organiza objetos em diferentes posições em um caminho de pesquisa. O comando abaixo mostra essas posições.

```
> search()
```

- O primeiro é o espaço de trabalho, o segundo é o pacote Epi, o terceiro é um pacote de comandos chamado métodos, o quarto é um pacote chamado estatísticas e assim por diante. Para ver o que há no espaço de trabalho, tente

```
> objects()
```

ou

```
> ls()
```

- Você deve ver apenas os objetos `births` e `diet`
- No pacote Epi existe uma função que fornece uma imagem mais detalhada, `lls()`
- Para ver o que há no pacote `Epi`

```
> ls(2)
```

Gráficos no R

- Existem três tipos de funções gráficas no R:
  - Funções que geram um novo gráfico
  - Funções que adicionam itens extras a um gráfico existente
  - Funções que permitem interagir com o gráfico
- O procedimento normal para criar um gráfico em R é fazer um gráfico inicial bastante simples e depois adicionar pontos, linhas, texto etc., de preferência em um script.

Gráfico simples na tela

- Histograma

  > hist(births$bweight, col="gray", border="white")

- Gráfico de dispersão

  > plot(births$gestwks, births$bweight)

- Podemos mudar os símbolos através da opção pch=.

  > plot(1:25, pch=1:25)

Exercícios

1) Faça um gráfico de bweight versus matage

```
> plot(births$matage, births$bweight)
```

2) Rotule os eixos com

```
> plot(births$matage, births$bweight,
xlab="Maternal age", ylab="Birth weight
(g)")
```

Adicionando ao gráfico

- Para iniciar com um gráfico vazio

```
> plot(births$gestwks, births$bweight, type="n")
```

- Adicionando pontos

```
> points(births$gestwks[births$sex==1],
births$bweight[births$sex==1], col="blue")

> points(births$gestwks[births$sex==2],
births$bweight[births$sex==2], col="red")
```

- Adicionando legendas

```
> legend("topleft", pch=1, legend=c("Boys","Girls"),
col=c("blue","red"))
```

- Adicionando título

```
> title("Birth weight vs gestational weeks in 500 singleton
births")
```

Interagindo com o gráfico

- A função locator() permite que você interaja com o gráfico usando o mouse. Ao digitar locator(1) você será levado para a janela de desenho e R aguardará um clique do botão esquerdo do mouse. Quando você clicar na tela, ele retornará as coordenadas correspondentes.
- Use a função locator() dentro de outras funções gráficas para posicionar elementos gráficos onde o usuário desejar.

```
> plot( births$gestwks, births$bweight, pch=c(16,3)[(births$matage>=40 )
+1], col=c("blue","red")[births$sex] )
```

- Adicione a legenda onde deseja que apareça

```
> legend(locator(1), pch=1, legend=c("Boys","Girls"), col=c("blue","red") )
```

- A função identify() permite encontrar quais registros correspondem aos pontos no gráfico.

```
> identify( births$gestwks, births$bweight )
```

- Quando você clica no botão esquerdo do mouse, um rótulo aparecerá no gráfico, identificando o número da linha do ponto mais próximo no data frame. Se não houver um ponto próximo, R imprimirá uma mensagem de aviso no console.
- Para finalizar a interação com a janela de desenho, clique com o botão direito do mouse: a função de identificação retorna um vetor de pontos identificados.

Salvando seus gráficos para uso em outros documentos

- Depois de ter um gráfico na tela, você pode clicar em `Arquivo → Salvar como` e escolher o formato em que deseja inserir o gráfico.
- Se você deseja controlar exatamente o tamanho do seu gráfico, pode iniciar um dispositivo gráfico antes de fazer o gráfico. Em vez de aparecer na tela, o gráfico será gravado diretamente em um arquivo.

```
> win.metafile(file="plot1.emf", height=3, width=4)
> plot(births$gestwks, births$bweight)
> dev.off()
```

Datas no R

- Estudos epidemiológicos geralmente contêm variáveis de data que assumem valores como 11/2/1962.
- Para converter um string de carateres para formato de data:

```
> as.Date( "14/07/1952", format="%d/%m/%Y" )
```

```
> as.numeric( as.Date( "14/07/1952", format="%d/%m/%Y" ) )
```

- O primeiro mostra o formato de data e o último o número de dias desde 1/1/1970, que é um número negativo para datas anteriores a 1/1/1970.
- Para outras possibilidades de formatos de datas, veja

```
?strftime
```