- Tutorial
- Exercícios
- Apostila
- TesteTutorial

1. Tutoriais de Introdução ao R

Testando o rdrr interativo

```
<iframe width='100%' height='400'
src='https://rdrr.io/snippets/embed/?code=print(%22Hello%2C%20world!%22)'
frameborder='0'></iframe>
```

Testando o IPython Notebook

```
<head>     <!-- Load require.js. Delete this if your page already loads
require.js --> <script
src="https://cdnjs.cloudflare.com/ajax/libs/require.js/2.3.4/require.min.js"
integrity="sha256-Ae2Vz/4ePdIu6ZyI/5ZGsYnb+m0JlOmKPjt6XZ9JJkA="
crossorigin="anonymous"></script> <script
src="https://unpkg.com/@jupyter-widgets/html-manager@*/dist/embed-amd.js"
crossorigin="anonymous"></script> <script
type="application/vnd.jupyter.widget-state+json"> { "version_major": 2,
"version_minor": 0, "state": {} } </script> </head> <body>     </body>
```

O Código é Tudo!

Um dos primeiros hábitos que você deve adquirir para trabalhar com o R é **não digitar os comandos diretamente no R**, e sim em um arquivo texto, que chamamos de *script* ou *código*. Na interface gráfica de usuário (R-GUI) do Windows e do MAC há um editor de códigos, que você pode manter aberto, juntamente com a janela do R.

Usuário Linux

A instalação do R no LINUX não inclui uma interface como a do Windows ou do MAC, para que o usuário escolha a sua.

Há várias opções de editores e interfaces de desenvolvimento (IDEs), veja dicas e linques na seção de material de apoio.

Se você usa LINUX, pode seguir este tutorial com qualquer editor ou IDE.

A figura abaixo é uma captura de tela do R-GUI do Windows, mas no MAC o editor é similar, e você

pode manter a mesma lógica. Deixe sempre uma janela de código aberta acima da janela do R:

Na figura acima há duas janelas com funcionamentos e objetivos muito diferente;

1. a janela da parte superior apresenta um arquivo de texto puro que pode ser editado.
2. a janela na parte inferior é o prompt do R, ou seja o programa propriamente dito. Essa janela recebe informações que são processadas pelo R e apresenta o resultado final desse processamento[1)].

Para evitar confusão e perda de trabalho é importante digitar as informações que serão transmitidas ao R (linhas de código) no arquivo texto e ir passando esses comandos ao R. Tenha certeza que entendeu a diferença entre essas duas janelas, mais a frente iremos falar sobre a importância dessa distinção e de não se trabalhar diretamente na janela do R. É imprescindível aprender a se organizar dentro da lógica do ambiente de programação, com o risco de perder trabalho ou ficar completamente perdido entre as tarefas que executa.

- Clique com o botão da direita do mouse no link do script e,
- Salve o arquivo em uma pasta [2)] conhecida do seu computador

Execute o R e abra o *script* com a opção do menu “Arquivo/Abrir script”:

Vá para a janela do *script*, coloque o cursor na primeira linha e tecle `Ctrl-r`. O comando é enviado ao R. Faça o mesmo com as duas linhas seguintes.

Para Usuários de MAC

Para enviar comandos do editor de código do R-GUI para o R utilize *Command+Enter* ou *Command+Return*.

Veja o material
https://cran.r-project.org/bin/macosx/RMacOSX-FAQ.html

Selecione as cinco primeiras linhas com o cursor e tecle `Ctrl-r`. Os comandos são enviados ao R.

Agora modifique os dados, alterando a segunda e terceira linhas do *script* para:

```
area <- c(300, 350, 961, 295, 332, 47,  122, 11, 53, 2749)
riqueza <- c(1, 7, 20, 7, 8, 4, 8, 3, 5, 23)
```

Salve o *script* com estas modificações.

Rodando o script

Além de poder enviar as linhas de comando é possível submeter o script integralmente para o R. Você pode fazer isto de duas maneiras:

- 1. Na janela do editor de código, selecione com o mouse ou marque todo o *script* com “Ctrl-a” e depois envie-o para o R com “Ctrl-r”
- **OU**
- 2. Na janela do R digite o comando:

```
source("regressao.r")
```

OBS: Este comando só funcionará se o arquivo *regressao.r* estiver no diretório de trabalho, que é o nome técnico da pasta para onde o R está direcionado no seu computador. Caso não esteja, aparecerá uma mensagem de erro dizendo que não é possível abrir a conexão solicitada. A seguir veremos com podemos mudar o diretório de trabalho após abrir uma sessão do R.
Por padrão, o `source` não mostra na tela os comandos executados nem seus resultados, mas todos os objetos são criados. Verifique os objetos criados por estes comandos no *workspace* do R digitando:

```
ls()
```

Se você quiser ver todos os comandos e resultados use:

```
source( "regressao.r", echo=TRUE, print.eval=TRUE )
```

Consulte a ajuda do comando `source` para entender os argumentos `echo` e `print.eval`.

Agora vamos simular a perda dos objetos: saia do R, respondendo “NÃO” à pergunta “Salvar Área de Trabalho”[3)].

Abra o R de novo. Tudo perdido? Não! Com o código salvo (script) você pode executá-lo novamente, e recuperar todo o trabalho. Repita o procedimento novamente de abrir o arquivo de script e rodá-lo  .

Escolha seu Editor de Código

O editor de códigos do R-GUI no Windows e no Mac é bastante simples e costuma ser uma boa opção inicial para usuários deste sistema. Para esta disciplina ele é suficiente, mas logo que você se acostume com o R é bom buscar um editor ou ambiente integrado de programação com mais recursos.

No Linux não uma há uma GUI padrão para o R, e esta escolha já tem que ser feita no início.

Ou seja, mais cedo ou mais tarde, terá que avaliar as opções de editores de código ou ambientes de programação e escolher uma que lhe agrade.

Na página de material de apoio há uma seção com várias dicas sobre interfaces para o R para lhe ajudar.

A Melhor Maneira de Executar o R no Windows

Antes de começar um novo projeto de análise, crie um diretório para ele, com o menu *"Arquivo/Novo/Pasta"* do windows explorer.

Em seguida execute o R a partir do atalho na área de trabalho ou na barra de ferramentas.

Verifique qual é o diretório de trabalho que o R está usando, com o comando:

```
getwd()
```

E você verá que ao abrir o R desta maneira ele sempre começará com um mesmo diretório de trabalho, possivelmente em "Meus Documentos", e.g.:

```
[1] "C:/Documents and Settings/Administrador/Meus documentos"
```

Para mudar o diretório de trabalho use:

```
setwd("C:/Documents and Settings/Administrador/Meus documentos/temp1")
```

IMPORTANTE: as barras devem ser no padrão Linux, ou seja, o inverso do usado em Windows.

Verifique se mudança funcionou, com um novo comando `getwd`:

```
getwd()
```

Se seu diretório de trabalho é o desejado, verifique que está vazio, com o comando:

```
dir()
```

E também verifique se o *workspace* está vazio com

```
ls()
```

Agora baixe o arquivo letras.rdata para o diretório de trabalho, e carregue-o no *workspace* do R com o comando:

```
load("letras.rdata")
```

Verifique agora seu workspace, e salve-o:

```
ls()
save.image()
```

Crie alguns outros objetos em seu *workspace*:

```
pares <- c(2,4,6,8)
impares <- c(1,3,5,7,9)
todos.os.numeros <- c(pares,impares)
```

Agora saia do R, tomando o cuidado de salvar de novo seu *workspace*.

Para trabalhar novamente no mesmo projeto, abra o diretório correspondente com o Windows Explorer e clique no arquivo `.RData`:

IMPORTANTE: certifique-se de que o diretório está com a opção de exibir arquivos ocultos, ou você não verá o arquivo `.RData`:

Verifique se todos os objetos da última seção estão em seu *workspace*:

```
ls()
```

Outra solução é criar um atalho para o projeto, indicando o diretório de trabalho na caixa de propriedades do atalho:

Você pode manter um atalho para cada projeto em andamento em sua área de trabalho.

Para Usuários de LINUX

Em LINUX não há estes problemas, pois basta executar o R na linha de comando (*shell*) a partir do diretório de trabalho (veja na apostila).

Com o uso de editores de código fica ainda mais fácil, consulte o guia que há em nosso material de apoio.

Criando Objetos

Os três operadores de atribuição `<-`, `=` e `->` podem ser usados de várias maneiras para criar objetos. Por exemplo, estes comandos:

```
a <- 1
b <- a
```

São equivalentes a este:

```
b <- a <- 1
```

Ou a este:

```
a = 1 -> b
```

Experimente!

Listando e Removendo Objetos

Várias funções retornam resultados mesmo sem que você forneça argumentos. Nestes casos, basta não escrever entre os parênteses. No caso da função `ls`, por exemplo, você irá obter a lista de todos os objetos em sua área de trabalho:

```
A1 <- c(1,2,3)
A2 <- c(10,20,30)
b <- c(A1,A2)
ls()
```

Consulte a página de ajuda da função ls:

```
help(ls)
```

Onde você verá a explicação para o argumento `pattern`. Execute, então, este comando:

```
ls(pattern="A")
```

Para mudar os nomes de objetos e apagar os antigos, experimente:

```
a.1 <- A1
a.2 <- A2
ls()
rm( list=c("A1","A2") )
ls()
```

Que tem o mesmo efeito de:

```
rm(list=ls(pattern="A"))
```

Ou de

```
rm(A1,A2)
```

Verifique!

Classes Date

Crie objetos com a datas do tri e tetracampeonatos mundiais do Brasil[4]:

```
copa.70 <- "21/06/70"
copa.94 <- "17/07/94"
```

Qual a diferença em dias entre estas datas? A subtração retorna um erro (verifique):

```
copa.94 - copa.70
```

Isto acontece porque os objetos são caracteres, uma classe que obviamente não permite operações aritméticas:

```
class(copa.70)
class(copa.94)
```

Mas o R tem uma classe para datas, que é `Date`. Faça a coerção dos objetos para esta classe, verifique se a coerção foi bem sucedida, e repita a subtração:

```
copa.70 <- as.Date(copa.70,format="%d/%m/%y")
copa.94 <- as.Date(copa.94,format="%d/%m/%y")
class(copa.70)
class(copa.94)
copa.94 - copa.70
```

NOTA: o argumento `format` da função `as.Date` informa o formato em que está o conjunto de caracteres que deve ser transformado em data, no caso dia/mês/ano (`%d/%m/%y`), todos com dois algarismos. Veja a ajuda da função para outros formatos.

Níveis de Fatores

Imagine uma escala de herbivoria com os níveis “alto”, “médio”, “baixo” e “nulo”. Vamos criar um objeto que representa o valor desta medida de herbivoria em uma amostra de 14 plantas:

```
herb <- c("A","M","M","A","A","M","M","B","A","A","A","A","B","A")
```

E então vamos criar um objeto da classe fator com estes valores:

```
herb.f <- factor(herb)
```

Usamos a função `table` para contar o número de observações em cada nível do fator, cujo resultado atribuímos a um outro objeto. Os valores são exibidos se digitamos o nome do objeto.

```
(herb.t <- table(herb.f))
```

A função para gerar gráficos `plot` pode ser aplicada diretamente ao objeto desta tabela:

```
plot(herb.t)
```

Há dois problemas aqui: na tabela e na figura os níveis não estão ordenados, e falta o nível de herbivoria nula. Isto acontece porque, ao criar uma variável de fator a partir de um vetor de valores, o R cria níveis apenas para os valores presentes, e ordena estes níveis alfabeticamente.

Isto pode ser mudado explicitando os níveis e sua ordem com o argumento `levels` da função fator:

```
herb.f <- factor(herb, levels=c("N","B","M","A"))
herb.t <- table(herb.f)
herb.t
plot(herb.t)
```

NOTA: há uma classe para fatores ordenados que poderia se aplicar aqui, mas seu uso tem implicações importantes nos resultados de algumas análises, que no momento não vêm ao caso. Mais informações a respeito na ajuda da função factor.

Carregando Pacotes

Pacotes são conjuntos de funções específicas do R, distribuídos em conjunto. No repositório do R estão armazenadas uma quantidade muito grande de pacotes que geralmente tem funções para um certo conjunto de tarefas associadas (p.ex: análise de padrões espaciais de pontos). Para usar um pacote é necessário entender a diferença entre baixar o pacote (download) do repositório e carregar o pacote na sua área de trabalho. Veja a apostila para mais detalhes se houver ainda dúvidas sobre como usar pacotes

Quais pacotes estão disponíveis na sua instalação de R? Você pode verificar isto com o comando:

```
library()
```

Outra maneira é iniciar a interface hipertexto de ajuda com:

```
help.start()
```

E escolher o link *"Packages"*, que você terá a lista do pacotes já instalados. Na interface hipertexto clique no nome de um dos pacotes. Você verá a lista de todos os objetos que este pacote contém.

Quais pacotes estão carregados? Uma maneira simples de descobrir é com o comando:

```
search()
```

Agora vamos gerar 15 números sorteados de uma distribuição normal, com média 1 e desvio-padrão=3, e guardar o resultado no onbjeto `x1`:

```
x1 <- rnorm(n=15, mean=1, sd=3)
```

Para fazer um histograma deste valores, há a função `hist`, do pacote *graphics*:

```
hist(x1)
```

Mas há também a função `truehist`, do pacote *MASS*:

```
truehist(x1)
```

Este comando retornará uma mensagem de erro, que avisa que o R não encontrou o objeto `truehist`. Para que isso não aconteça, é preciso carregar o pacote *MASS*, que já está instalado na distribuição básica do R:

```
search()
library(MASS)
search()
truehist(x1)
```

1)
quando a tarefa solicitada é a representação de um gráfico, uma nova janela é aberta, um dispositivo gráfico
2)
diretório de trabalho é o nome técnico desta pasta para o R
3)
não faça isto normalmente!!!, detalhes na apostila
4)
fonte: FIFA