R para iniciantes

Boa parte dos roteiros deste site usam o ambiente de programação estatística R. Você não precisa saber programar em R para seguir os roteiros, mas é recomendável que se familiarize com a filosofia do R, e o modo de executar comandos.

Este guia foi preparado por Marcel Caritá Vaz e Vitor Rios. É uma introdução voltada para quem nunca teve contato com o ambiente R.

Prazer, meu nome é Gui, R Gui

Em nossas práticas iremos utilizar um programa que interage com você por meio de linhas de comando, o que significa que ele tem uma linguagem própria que vocês podem não conhecer. À primeira vista vai parecer assustador, mas vocês verão que não é tão ruim assim. Este roteiro atuará como seu guia e tradutor. Sigam-no e ninguém sairá ferido. *Bon voyage!*

Primeiro vamos abrir o tal programa e nos familiarizar um pouco com ele.

Se você usa Windows ou Mac procure no desktop, dock menu ou menu iniciar do seu computador um *R* azul e estiloso como o da figura acima. Para Linux, recomendamos que você use o Rstudio[224)].

Abram o programa e vocês verão uma tela árida e sem vida (Lembrem-se de que a Terra era assim também há alguns bilhões de anos atrás e veja no que ela se tornou). Note que há algumas poucas opções de comandos no menu superior. Esse é o GUI (Graphic User Interface) que apesar de básico irá ajudá-lo em algumas tarefas mais simples.

Sem desanimar, vamos utilizar o *R* para fazer a coisa mais simples que ele sabe: operações aritméticas. Note que os comandos que devem ser inseridos no programa estarão sempre no formato de texto *code*, com um fundo azulado.

Durante esse roteiro terá duas opções: (1) copiar os códigos que aparecem no roteiro em um fundo azul claro e colar no console do R ou (2) digitar diretamente no console do *R* os comando que estão no roteiro. Não esqueça de que devem dar um *ENTER* depois de escrever cada linha de comando. Veja o exemplo de linhas de comando abaixo e teste-as no *R*.

```
2+2
5-1
2*3
4/5
```

Notem que há um sinal de > em vermelho no canto esquerdo. Isso mostra o início da linha de

comando. As linhas de comando, ou as ordens dadas ao programa, vão sempre ficar em vermelho, enquanto que as respostas que ele nos der vão sempre ficar em azul. Repare também que sempre, antes do resultado, aparece um **[1]**. Isso nada mais é do que um contador, quer ver? Vamos pedir agora uma seqüência de 1 a 100.

```
1:100
```

Note que o contador mostra a posição do primeiro valor de cada linha de resultado.

Um pouco mais de R

Continuando, uma coisa que o *R* gosta bastante de fazer é criar objetos. Para tanto basta escolher um nome e atribuir-lhe algum valor ou conjunto de valores ou mesmo palavras. Se você quiser que seu objeto tenha mais de um elemento basta escrever **c(elemento 1,elemento 2,etc)**. Vamos tentar?

```
valor <- 5
pares <- c(0,2,4,6,8)
cores <- c("azul","vermelho","amarelo")
```

Quando usamos o símbolo "=" ou "<" seguido de "-" estamos criando objetos com um nome que aparece a esquerda e que contém alguns elementos (o que vem a direita do símbolo). Quando criamos um objeto, ele fica guardado na memória do *R* até que você feche o programa. Porém, os objetos criados ficam ocultos. Para ver a lista de arquivos ocultos basta dar o seguinte comando:

```
ls()
```

Já para ver o conteúdo de cada objeto basta chamá-lo pelo nome:

```
valor
pares
cores
```

Muito bem. Espero que até aqui ninguém tenha morrido **ainda**. Agora vamos fazer um dos truques mais bacanas do *R*: que comece a jogatina! Você sabia que podemos jogar dados com esse programa? Quer apostar?

```
moeda <- c("cara","coroa")
moeda
sample(moeda,1)
sample(moeda,1)
sample(moeda,1)
sample(moeda,1)
```

A cada vez que você roda *sample(dado,1)* o *R* faz um sorteio. Vamos jogar uns dados agora.

```
dado <- c(1:6)
dado
sample(dado,1)
```

```
sample(dado,1)
sample(dado,1)
```

Gostaram? Mas não vão ficar viciados, hein?! 

Finalmente, vamos às brincadeiras de gente grande. Em algumas de nossas práticas vamos trabalhar com matrizes. Felizmente é muito fácil transformar um conjunto de dados em uma matriz utilizando o comando *matrix()*.

```
1:100
matrix(1:100,nrow=20,ncol=5)
```

Note que os argumentos *nrow* define o número de linhas (20) e *ncol* o de colunas (5).

Mas podemos ir além e criar um objeto tridimensional do tipo array:

```
array(1:100,dim=c(5,5,4))
```

Note que o argumento agora é *dim* deve conter os valores de cada dimensão c('linha', 'colunas', 'matriz'). O resultado aparece como fatias de um bolo, onde cada camada é uma matriz. Conseguiu visualizar?

Outra coisa bacana é que podemos criar objetos vazios. (*NA* quer dizer não se aplica, ou seja, aquela posição não está sendo ocupada por nenhum valor.)

```
a=rep(NA,10)
a

b=matrix(NA,nrow=3,ncol=3)
b

c=array(NA,dim=c(3,3,3))
c
```

Mas qual a graça de se fazer isso? A graça é que depois podemos preencher os lugares vazios com os valores que quisermos.

```
a[1]=2
a
a[1]

b[3,3]="oi"
b
b[2,2]="oi"
b
b[1,1]="oi"
b
b[3,3]

c[3,3,1]="oi"
c
```

```
c[3,3,2]="ai"
c
c[3,3,3]="ui"
c
c[3,3,3]
```

Pronto, agora sem querer querendo vocês já sabem indexar! O que é isso? É simplesmente dizer a posição de um objeto que você quer. Se for um objeto simples do tipo linear, como é o caso de *a*, basta escrever o nome do objeto e em seguida um valor dentro dos colchetes (**nome[posição]**). Se for uma matriz como o *b*, você tem que dizer **nome[número da linha, número da coluna]**. Finalmente, se for um objeto tridimensional como o *c* você deve dizer **nome[número da linha, número da coluna, número da fatia do bolo]**.

Gráficos

Outro forte do *R* é a criação de gráficos. Veja alguns exemplos extremamente simples:

```
plot(1:10,10:1)
plot(1:10,10:1,pch=2)
plot(1:10,10:1,pch=2,col=2)
plot(1:10,10:1,pch=2,xlab="eixo x")
plot(1:10,10:1,pch=2,xlab="eixo x",ylab="eixo y")


hist(rnorm(1000),col=33)
arrows(2,100,2,70,col=2)

boxplot(iris$Sepal.Length~iris$Species)
```

Funções no R

Agora, o golpe de misericórdia: como criar uma função. Criar uma função nada mais é do que programar o computador para fazer uma tarefa em seu lugar. Soou interessante? Pois é, mas isso exige um pouco mais do nosso esforço. Vamos ver um exemplo bem simples. Vou ensinar o programa a calcular a média de um conjunto de valores.

```
media=function(objeto){sum(objeto)/length(objeto)}
media(c(1,2,3))

alturas=c(1.4,1.7,2.0,1.6,1.8)
media(alturas)
```

Note que entre parênteses estão os valores ou o objeto dos quais você quer a média. Entre chaves está o comando que calcula a média. Note também que para calcular a média usei duas funções pré-existentes no *R*: *sum* e *length*. Na verdade também já existe uma função pronta que calcula a média: *mean*.

```
sum(alturas)    # soma dos elementos do objeto "alturas"
length(alturas) # número de elementos do objeto "alturas"
mean(alturas)   # média dos elementos do objeto "alturas"
```

O "#" em cada linha de comando significa que o *R* deve desconsiderar o texto. Isso é bom: quer dizer que o programa não se mete na conversa dos outros.

Bom, isso é tudo, ou quase tudo, o que precisávamos saber sobre a lógica do *R* para fazermos a prática. Não se preocupem com detalhes sobre a linguagem. Contem com seu guia: o roteiro a seguir. E se ele falhar, por favor não hesitem em clamar pela ajuda dos professores e monitores. Boa sorte.

Para você que quer mais!

Se você se interessou pelo programa não se sinta mal. Ele realmente é incrível, pois te permite fazer quase tudo que quiser. Visite o site da disciplina do R, oferecida para os alunos de pós graduação da Ecologia da USP. O começo, como em toda linguagem, é uma subida árdua. Mas com o tempo a coisa chega a ser até prazerosa. *Marcel*

Carregando Arquivos no R

O R também permite ler arquivo de dados, para que você não tenha de digitar tudo à mão sempre que for fazer suas análises.

Geralmente estamos acostumados a salvar nossas planilhas de dados no Excel (formato .xls e .xlsx), e fazer as análises e transformações de dados diretamente na planilha, porém este não é o melhor procedimento para o uso do R. A melhor prática é salvar os dados originais em um arquivo, e fazer as análises diretamente no R, salvando os comandos utilizados para a análise em outro arquivo, com a extensão .R (que chamamos de "script"), e os resultados da análise em um terceiro arquivo. Parece mais prático ter tudo em um arquivo só, mas a filosofia do R é a reciclagem dos comandos: se você salvar os comandos separado dos dados originais, pode usar esse arquivo para analisar outros dados semelhantes, sem correr risco de sobrescrever ou misturar os dados de dois experimentos diferentes. Pense quantas vezes você já teve de fazer uma ANOVA no Statistica ou SPSS ou transformar seus dados? Usar um script para automatizar sua análise salva tempo, dor de cabeça e lesões por esforço repetitivo.

Definindo o diretório de trabalho

Tudo que você fizer no R será salvo ou carregado a partir do seu diretório de trabalho. Para isso, é necessário indicar ao R em qual diretório você está trabalhando, usando os comandos

```
getwd()
```

e

```
setwd()
```

. Primeiro, verifique em que diretório o r está digitando getwd(). por padrão, no windows o diretório é

a sua pasta documentos. para alterar isso, utilize o comando setwd()

```
setwd("C:/meu_drive/meu_diretorio")
```

Note que o endereço do diretório tem de estar entre aspas, e as barras (“\”) tem de ser trocadas por barras invertidas(“/”). Após usar o comando acima, confira se funcionou usando o getwd(). você também pode alterar o diretório utilizando o menu Arquivo → Mudar dir

Formatação dos arquivos

O R é capaz de ler arquivos de texto em uma variedade de formatos. Sugerimos que você digite seus dados em um planilha no Excel, OpenOffice ou LibreOffice, e salve os dados em um formato especial chamado “.csv” (No Excel, vá em Arquivo → Salvar Como → outros formatos, e escolha CSV (separado por virgulas)) Isso transforma sua planilha em um arquivo de texto, em que cada célula é separada das outras por uma virgula.

> Muita calma nessa hora!
>
> O R segue o padrão americano de formatação de números, portanto os decimais devem ser separados por pontos e não por vírgulas. Revise bem seus dados antes de começar suas análises! Sugerimos definir o seu editor de planilhas para sempre usar o padrão americano, para evitar confusões. Busque na ajuda do seu editor como fazer isso (o procedimento é diferente para cada programa)

Mas porque tanta firula quanto a salvar os dados em um formato especial? Bem, primeiro de tudo, o formato .csv nada mais é que um arquivo de texto puro, que pode ser lido em qualquer computador, independente de ser Mac, Windows ou Linux, e de ter o MS Office instalado ou não. Além disso, as empresas mudam frequente os formatos dos seus arquivos, gerando incompatibilidade e forçando você a obter a versão mais recente do programa (duvida? tente abrir uma planilha do Excel 2013 em um PC com Excel 2003). Salvar em um formato padronizado e aberto garante que daqui a dez anos você ainda vai poder ter acesso a seus dados e colaborar com pessoas que usem outros programas.

Com seus dados devidamente arrumados e salvos, podemos prosseguir para a leitura dos arquivos no R

Lendo os arquivos

Agora basta dizer ao R qual arquivo você quer que ele leia, e como esse arquivo está formatado. para isso usamos a função `read.table()` e guardamos o resultado em uma tabela

```
tabela = <- read.table(file="DadosDoMeuDoutorado.csv",header=T,sep=",")
```

O que isso quer dizer? esse comando diz ao R e os dados do meu doutorado estão no arquivo “DadosDoMeuDoutorado.csv”, que possui uma linha de cabeçalho com o nome das minhas colunas (“`header=T`”), e que minhas colunas são separadas por vígulas (`sep=“,”`). Se seu arquivo nao tiver cabeçalho, use `header=F`, e se suas colunas estiverem separador por outro símbolo, por exemplo ponto-e-vírgula, coloque o símbolo dentro das aspas do parâmetro `sep` (`sep=“;”`).

Após rodas essa linha de comando, meus dados estão guardados dentro do objeto `tabela`, e agora posso realizar minhas análises sem me preocupar em sobrescrever os dados originais, que estão são e salvos no arquivo DadosDoMeuDoutorado.csv, no meu diretório de trabalho (por falar nisso, você já fez o backup dos seus dados hoje?)

Para se aprofundar

- Impatient R: introdução muito didática e bem humorada.
- R em 4668 palavras: uma introdução mais extensa ao R, que inclui os principais procedimentos estatísticos. Boa apresentação também da filosofia de programação com dados e código aberto do R.
- Disciplina de R da PG-Ecologia USP: tutoriais, exercícios e muitos links para outros recursos.
- Learn R: excelente curso, do USGS. Tutoriais, exercícios e vídeo-aulas.

224)

também disponível para outros sistemas operacionais