Nivelamento de Programação em R - Aula 1 - Parte 1

Umberto Mignozzetti

Fevereiro 2020

Introdução Ao Curso

Bem-vindos ao nosso curso de Nivelamento de Programação em R. Como esta é nossa primeira interação, aproveitaremos este espaço para apresentar um pouco mais a dinâmica do curso. Ao fim deste script teremos já alguns exercícios a serem feitos, então não deixe de ler este documento até o fim.

Ao fim de cada aula de nosso curso será requisitado a vocês fazer alguns rápidos exercícios para retenção de conhecimento. Caso vocês sintam alguma dificuldade, não hesitem em contactar a mim ou o nosso monitor. Recomendamos fortemente também que vocês tentem procurar a solução por conta própria, seja através de ferramentas de busca como o Google, ou mesmo através de sites ou fóruns especializados como o StackOverflow. O trabalho do cientista de dados requer a constante atualização de nossos conhecimentos, então não se sintam acanhados caso não conseguirem resolver algum problema!

Trabalharemos com três grandes tipos de exercícios em nosso curso: "Praticando", "Exercícios Gerais", "Fix the Code" e "Desafios". **Praticando** são exercícios introduzidos ao longo de nossos tutoriais. São exercícios mais simples, destinados a fazer com que vocês entendam melhor como funcionam as técnicas que estamos lhes ensinando. Já com os exercícios do tipo **Fix the code** apresentaremos alguns códigos que propositadamente estão incorretos. O trabalho de vocês será arrumar tais códigos, para que eles funcionem tal como deveria. Por fim, os **Desafios** são os exercícios requerirão um pouco mais de vocês. O objetivo de tais exercícios é para desenvolver sua autonomia enquanto pesquisadores. Caso não consigam resolvê-los, que escrevam a linha de raciocínio de vocês, ou mesmo apresentem um código, ainda que incompleto.

Ao final de cada aula, pediremos que enviem seus scripts (bem como eventuais arquivos complementares) ao monitor de nossa aula. Vocês podem contactá-lo em: **lucasmingardi@gmail.com**. Não se esqueçam de escrever o Assunto (Subject) do email como "Nivelamento de Programação em R - [seu nome]" (troque [seu nome] pelo seu próprio nome). Exercícios enviados sem o campo do Assunto preenchido corretamente serão sumariamente desconsiderados. Serão aceitos apenas um envio por aluno.

Damos liberdade aos alunos em escrever seus códigos tanto como scripts (formato ".R") como documentos RMarkdown (formato ".Rmd"). O importante é que o código funcione. Atenção: serão descontados pontos de códigos que apresentarem algum tipo de erro.

R e RStudio

Abrir o R puro pode ser uma experiência um pouco amedrontadora de início. A única coisa que vemos lá é uma caixa de comando pouco intuitiva. O R padrão é uma linguagem interpretada, logo é comum que ao abrirmos vejamos apenas um console de comandos.

O RStudio veio para facilitar a utilização do programador em R. O Rstudio se trata de um ambiente de desenvolvimento integrado (Integrated Development Environment, ou IDE), isto é, um frontend que permite maior produtividade ao criar uma interface mais amigável. Apesar de não ser o único IDE voltado para o R, o RStudio certamente é o mais popular entre os usuários.

O RStudio é dividido em quatro setores. O quadrante inferior esquerdo é o "console", que já tínhamos visto no R padrão. É lá que o resultado dos comandos são reportados. Podemos também escrever diretamente nossos comandos lá. Há outras abas neste console, mas elas são pouco utilizadas. O terminal, como o nome diz, é o terminal de seu computador. Ao invés de abrir o terminal diretamente, podemos acessá-lo através do R. Já a aba Jobs é a utilizada para criar scripts que rodam no background.

O segundo quadrante, superior esquerdo, é o "script". Trata-se de um documento de texto, utilizado para facilitar a execução e reprodução de nossos códigos. Por vezes um script não é automaticamente carregado no RStudio. Se este for o caso, podemos criar um novo script indo em File > New File > e escolhermos R Script/R Markdown/R Notebook. Trataremos da diferença destes modelos de anotação na seção "Fazendo anotações no R - Scripts, RMarkdown e RNotebooks". Note que múltiplos scripts podem ser carregados em um único ambiente do RStudio. Aprenderemos mais sobre pacotes na seção "acotes (packages)"

O terceiro quadrante, superior direito, é o "environment". Lá ficam guardados os objetos que criamos para fazer nossas análises. Objetos podem assumir diversas formas: vetores, listas, bancos de dadosser qualquer tipo de coisa: vetores, listas, valores, entre outras formas. Exploraremos na aula seguinte como guardamos objetos no ambiente do R, bem como as diversas características que objetos podem ter. Outras abas desse quadrante são a "history" e "connections". A aba history mostra o histórico de todas os comandos utilizados pelo usuário em dada sessão. Já a aba connections mostra as conexões feitas a servidores externos como, por exemplo, SQLservers ou Spark. Novamente, nem utilizaremos essa aba no presente curso.

Por fim, o quarto e último quadrante, superior direito, é uma aba de suporte. A aba que será mais utilizada aqui é a aba "Plots". Nela são impressas as imagens que criamos através do R. Outra aba útil é a aba "Help", um mecanismo de ajuda integrado do R. Trataremos mais dessa aba na seção "Pedindo ajuda no R". A aba "packages" mostra os pacotes instalados pelo usuário. A aba "Files" mostra os arquivos e pastas do diretório que o R está operando (working directory). No caso específico do Windows, esta pasta costuma ser a pasta "Documentos". Aprenderemos como mudar o working directory futuramente. Por fim, a aba Viewer é utilizada para visualizar conteúdos da web local. O exemplo típico são os Shiny Apps, que não veremos neste curso introdutório.

Operações básicas com R

Uma forma de olharmos para o R é como sendo uma grande calculadora, apesar de ser mais robusta. Assim como toda linguagem de programação, qualquer função ou operação que executamos, não importa quão complexa seja, pode ser reduzida a um certo conjunto de operações matemáticas e/ou lógicas. Antes de aprendermos funções mais avançadas, portanto, devemos entender melhor como funcionam as operações mais simples do R: operadores matemáticos. A seguir vemos uma lista de operadores:

- Soma: +
- Subtração: -
- Multiplicação: *
- Divisão: /
- Exponenciação: ^
- logaritmo: log()
- Resto da divisão: %%

Há também os operadores de comparação:

- Igualdade: ==
- Diferença !=
- Maior: >
- Maior ou igual: >=
- Menor: <
- Menor ou igual: <
- Subconjunto: %in%
- Negação: !

Também temos operadores lógicos:

- Verdadeiro: TRUE, ou simplesmente T
- Falso: FALSE, ou simplesmente F

Finalmente, temos algumas definições especiais, tais como:

- Pi: pi
- Infinito: Inf
- Infinito Negativo: -Inf
- Valor Indefinido (Not a Number): NaN
- Valor omisso (Missing Data): NA

Praticando

1.1) Faça algumas operações matemáticas com os operadores que vimos até agora. Escolha pelo menos três operadores distintos.

Pacotes (packages)

Uma das grandes vantagens de se usar o R são seus pacotes adicionais, chamados em inglês de *packages*. Pacotes são conjuntos de ferramentas adicionais, que fornecem funções inteiramente novas ao R. Como o R é uma linguagem aberta, é extremamente simples para usuários finais criarem seus próprios pacotes e compartilharem com a comunidade. Como a comunidade do R é extremamente diversa é possível de se encontrar pacotes dos mais variados tópicos, desde análises gráficas e análises textuais, a pacotes mais lúdicos como bancos de dados de partidos de futebol inglês. Pacotes aprovados pela comunidade do R são hospedados no CRAN (acrônimo para "Comprehensive R Archive Network"). É possível, no entanto, instalar pacotes de outras fontes, como por exemplo da plataforma GitHub.

O pacote que mais usaremos no presente curso é o *tidyverse*. Na verdade, o tidyverse não é só um pacote, mas um conjunto de pacotes, criados todos seguindo a mesma filosofia: o *tidy data* ("dados arrumados", em uma tradução um tanto livre). Originalmente, o tidyverse tinha sido criado por apenas uma pessoa, Hadley Wickam. Com o tempo, contudo, mais pessoas se uniram ao projeto, e hoje em dia vários autores integram o conjunto de pacotes tidyverse. O tidyverse veio para revolucionar a linguagem R, simplificando muito da forma que trabalhamos. Veremos as diferenças do R padrão (costumeiramente apelidado de "R base") do tidyverse ao longo de nossas aulas.

Para instalar um pacote no R é muito simples, basta utilizarmos a função install.packages(). Por exemplo, para instalarmos o tidyverse (o que todos deveríamos fazer, visto que utilizaremos muito este pacote!) devemos escrever: install.packages("tidyverse"). Note o uso de aspas. Cuidado também com o uso de maiúsculas ou minúsculas, pois o R é *case sensitive*.

Toda vez que quisermos utilizar uma função que está dentro de um pacote do R, devemos carregar seu pacote de origem primeiro. Para tal, usamos a função library(). Continuando a usar o exemplo do tidyverse, podemos carregar este conjunto de pacotes usando a expressão library(tidyverse). Note que, desta vez, não usamos aspas em nossa função.

Praticando

1.2) Mencionamos antes que é possível de se instalar pacotes diretamente através do GitHub. Fazemos isso através do pacote *devtools*. Instale este pacote.

1.3) ggplot2 é um pacote de recursos gráficos do conjunto de pacotes do Tidyverse. Podemos instalar a versão experimental deste pacote indo diretamente em seu repositório GitHub. Utilize a função install_github("tidyverse/ggplot2") para tal. Certifique-se que o pacote devtools esteja corretamente carregado em seu sistema antes de usar esta função.

Pedindo ajuda no R

Como cientistas de dados vocês verão que muitas vezes há situações completamente novas que vocês simplesmente não sabem lidar. Ou, então, é possível que vocês não se lembram exatamente como funciona alguma função específica. Para tais casos, o R conta com algumas funções de ajuda que podem ser bem úteis.

Para entendermos como funciona o mecanismo de ajuda do R utilizaremos como exemplo a função sum(). Tal função é uma forma alternativa de se fazer somas no R. Para entendermos como funciona tal função podemos escrever help(sum). Alternativamente, o mesmo resultado pode ser obtido simplesmente digitando ?sum(). Para funções não-estandardizadas como *for*, *if* ou operadores matemáticos, devemos incluir aspas em nossa busca. Por exemplo, help(+) e ?+ retornarão um erro. Já help("+") e ?"+" trarão a informação que desejamos.

Caso você não saiba o nome de uma função, é possível utilizar a função help.search(). Por exemplo, caso queiramos aprender como fazer uma regressão linear no R, podemos escrever help.search("regression"). Alternativamente, podemos escrever ??"regression" para obtermos o mesmo resultado. Novamente, atente para o uso das aspas.

Outro comando útil é a função apropos(). Assim como o nome dessa função diz, ela busca funções "a propósito", isto é, que diz respeito a algo. Suponha, por exemplo, que você se lembre que há alguma função no R, mas não se recorde do nome exato dela. É para isto que a função apropos() serve, você escreve uma sequência de caracteres, e a função retorna o nome de todas as funções que contêm tal sequência. Por exemplo, caso queiramos saber o nome da função que faça uma correlação entre variáveis, podemos escrever apropos("cor"). Olhando para os resultados que o apropos nos fornece, descobrimos que esta é a função cor.test().

Praticando

1.4) separate() é uma função do pacote "tidyr", que faz parte do conjunto de pacotes "tidyverse". Use o comando de ajuda para aprender mais sobre esta função.

1.5) Busque, dentro do R, como fazer uma "logistic regression".

1.6) Use a função apropos() para descobrir as funções que contêm o termo "test".

Pedindo ajuda fora do R

Por vezes, as informações que obtemos do próprio R não são suficientes. Para isso, sempre podemos contar com fontes externas. Muitas vezes simplesmente usar o Google para pesquisar o que queremos salva horas de dores de cabeça. Para não perdermos tempo caçando resultados no Google, todavia, podemos utilizar alguns sites de confiança.

O melhor exemplo seria o StackOverflow. O StackOverflow é um grande fórum dos assuntos mais variados, cobrindo tópicos que vão desde áreas mais "duras" (programação, matemática, estatística, ciência), até a temas mais comuns como tecnologias, arte, estilos de vida e cultura. A grande vantagem do StackOverflow é que é uma comunidade grande e ativa, sendo bem respeitada por seus pares. Não é incomum ver certas "celebridades" do campo acadêmico darem as caras por lá! Hadley Wickham, o grande "guru" do tidyverse, por exemplo, é um frequentador assíduo do site!

Outro site útil é o RBloggers. Como este não é um fórum como o StackOverflow, não é possível de se fazer perguntas diretas por lá. o Rbloggers, contudo, é um blog que apresenta novidades da comunidade do R. Não é raro também de se encontrar tutoriais por lá, de assuntos mais elementares aos mais complexos.

Por fim, o GitHub é uma plataforma feita para desenvolvedores. Lá é possível encontrar projetos dos mais variados, não só relativos ao R. Não é incomum, contudo, de se achar pacotes do R sendo hospedados por lá. A grande vantagem do GitHub é o espírito de compartilhamento e de cooperação. Após fazer uma conta, é possível "clonar" projetos, fazendo alterações que você acha conveniente. Caso você queira compartilhar tais alterações com o dono original do repositório, você pode "publicar" (publish) seu clone. Caso o dono original do projeto goste de sua alteração, ele pode incorporar ao projeto original dele. É neste ambiente cooperativo que muitos pacotes do R são criados e aperfeiçoados.

Praticando

1.7) Para a presente tarefa pedimos que vocês se familiarizem com o ambiente do StackOverflow. Navegue um pouco pelo site. Em seguida, **escolha uma função** que foi aprendida neste módulo e busque no site uma dúvida acerca dela. Com o tempo você perceberá que, caso você tenha alguma dúvida acerca do R, muito provavelmente alguém mais teve a mesma dúvida no passado. **Copie aqui o link** da discussão acerca de seu tema escolhido. **Resuma (em não mais que 3 ou 4 frases)** a dúvida da pessoa, bem como se a(s) solução(ões) propostas pelos outros usuários é(são) ou não suficiente(s) para responder à pergunta do postador original.

Fazendo anotações no R - Scripts, RMarkdown e RNotebooks

Embora seja possível fazer um trabalho inteiro no R usando unicamente o Console, esta não seria das decisões mais sensatas. A principal desvantagem de se usar unicamente o Console é que é difícil de rastrear cada uma das funções que utilizamos. Pior ainda é o cenário em que acabamos utilizando erroneamente uma função, tendo de refazer todo nosso trabalho. Para não cairmos em tais tipos de problemas, fazemos anotamos todas as operações que fazemos. Anotar suas operações, no entanto, não é útil somente em situações de uso próprio. Anotações de código são essenciais para uma questão de ética e transparência. Somente podemos ter confiança na pesquisa de uma pessoa se ela disponibiliza um código minimamente reproduzível.

O R apresenta diversas formas para anotarmos nossos códigos. Ensinaremos aqui as três mais usuais delas: **scripts**, **RMarkdown** e **RNotebooks**.

Scripts são textos padrões de R. Apesar de serem guardados em um formato único (.R), scripts nada mais são do que documentos simples de texto. De fato, é possível abrir arquivos .R no próprio Bloco de Notas (Notepad) do Windows. Tudo que for escrito em tais códigos será reproduzido no console. Caso queiramos rodar o código por inteiro, basta clicar no botão "Run", no canto superior direito da aba do script no RStudio. Caso queiramos rodar apenas parte de um código, podemos selecionar a parte desejada e clicar em "Run". Alternativamente, podemos selecionar o código e apertar Ctrl e Enter (Cmd e Enter para Mac).

Talvez mais importante do que registrar os comandos que utilizamos é **registrar comentários acerca de nossos códigos**. Por vezes, olhar para um conjunto de códigos sem explicação nenhuma pode parecer algo um tanto esotérico. Para isso, adicionamos comentários explicando o que cada função faz. Novamente, isto que faz um script verdadeiramente transparente. Pouco importa se o cientista de dados compartilha um punhado de códigos se ele não se dá o trabalho de minimamente explicar o que cada comando que ele utiliza faz. Mas talvez tão importante quanto a transparência que temos com os outros é a transparência que temos consigo mesmos. Scripts são, em certa medida, um pacto que temos com o nosso "eu" de hoje e o "eu" do futuro. Alguns minutos a mais que gastamos comentando nossas operações podem salvar horas de revisão de código no futuro!

Para incluir comentários em scripts é bem simples, basta utilizarmos a cerquilha (#, também conhecido como "tralha", "jogo-da-velha", ou "cancela"). É importante que escrevamos comentários com a cerquilha para que o R não o rode como código (ou seja, que o comentário seja "cancelado", como diriam os programadores de velha-guarda).

```
#Comentando o meu script

#isso aqui é uma soma com vários números
1 + 2 + 3 + 4
```

```
## [1] 10
```

```
5/6 #também podemos comentar à direita do código - isso aqui é uma divisão....
```

```
## [1] 0.8333333
```

```
#mas se comentamos à esquerda, o código fica como se fosse comentário! 3*4
```

Já o **RMarkdown** funciona relativamente como o inverso de scripts comuns. O RMarkdown é um formato de arquivo próprio (.RMd) capaz de criar documentos dinâmicos no R, através da sintaxe Markdown. O RMarkdown é escrito em texto simples (plain-text), podendo, posteriormente, ser transformado em outros formatos, como PDF, HTML e DOCX. Como já mencionamos, o RMarkdown funciona um pouco como o inverso do R: seu propósito principal é criar anotações. Portanto, tudo o que escrevemos no RMarkdown sai como texto comum, e não como código. Para rodarmos códigos dentro do RMarkdown precisamos adicionar "chunks" ("pedaços" em uma tradução literal). Para criarmos um chunk, abrimos uma caixa fazemos uma começando com três sinais graves ("sinal de crase") seguida de {r} e terminando também com três sinais graves. Alternativamente, podemos simplesmente usar o comando "Ctrl + Alt + i" para criar um novo chunk. Caso desejem, também é possível criar comentários (através de #) dentro dos chunks de códigos do RMarkdown.

Por fim, o **RNotebook** é um tipo específico de documento RMarkdown. A grande vantagem deste é que os chunks podem ser utilizados independentemente uns dos outros. O resultado do código é impresso logo abaixo do chunk do código.

Praticando

1.8) Crie um novo script de R. Escreva, como comentário de código, seu nome e a data de hoje. Salve o documento como "doc_1_[seu nome].R". Não esqueça de substituir "[seu nome]" por seu próprio nome.

1.9) Abra o documento de RMarkdown que criamos para gerar este pdf que vocês estão lendo. Veja como foram usados os comandos para criarmos o presente documento.

1.10) Crie um novo documento RMarkdown. Escreva seu nome como o autor do documento, faça com que o título seja "Nivelamento de Programação em R - [seu nome]". Deixe o *output* do documento como *html_document*. Crie um novo *chunk* de código e escreva "2 + 2" (sem aspas) dentro deste chunk. Use o botão "Knit" para rodar seu documento.

1.11) Abra o documento RMarkdown criado no exercício anterior. Altere o *output* para *pdf_document*. Use o botão Knit para criar este seu novo documento.

Fix the Code

2.1)

```
"2 + 2"
```

2.2)

```
2 + "pi"
```

2.3)

```
1 = TRUE
```

2.4)

```
4 \ 3
```

2.5)

```
Help(cor.test)
```

2.6)

```
???cor.test
```

2.7)

```
help(if)
```

2.8)

```
?help("cor.test")
```

2.9)

```
esta é a minha soma

\# 1 + 2
```

Desafio

3.1) “separate()” é uma função do pacote “tidyr”, que faz parte do conjunto de pacotes “tidyverse”. Utilize corretamente o comando de ajuda do R para visualizar o comando “separate()” SEM que o pacote “tidyr” ou o “tidyverse” estejam carregados em seu *environment*. Dica: certifique-se que o pacote tidyverse esteja instalado eu seu computador. Dica 2: busque como se escreve a notação para se referir a uma função dentro de um pacote.