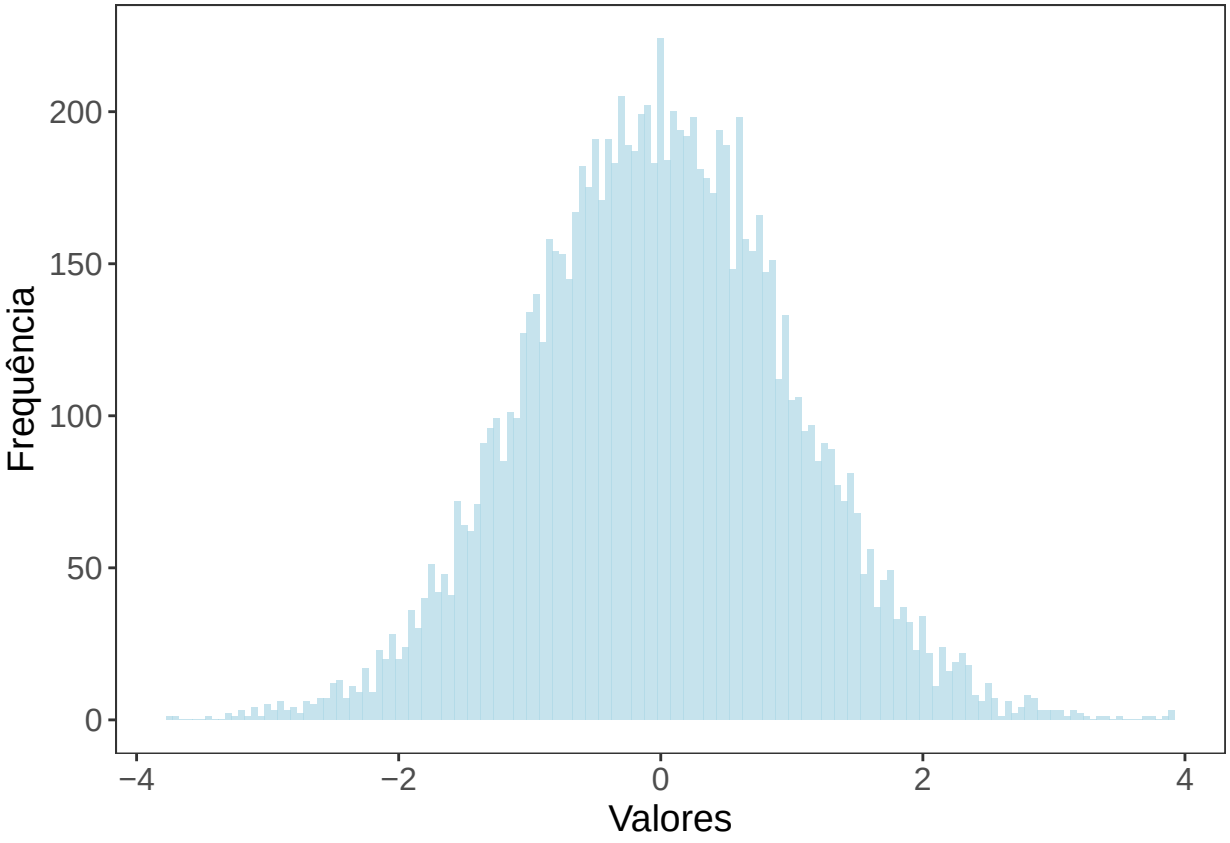
Introdução à linguagem R

Pablo Silva

Universidade Federal de Goiás



O que é programação?

Programação é um processo de escrita, testes e manutenção de programas de computadores. Basicamente é um sistema que automatiza as tarefas por meio de uma linguagem intemediária que transmite comandos desenvolvidos pelo programador e, com isso, são passíveis de serem executadas pela máquina.

A primeira pessoa a programar foi uma mulher conhecida como **Ada Lovelace**, que escreveu um código que possibilitou a utilização da máquina analítica de Charles Babbage, uma máquina robusta, de difícil comunicação, considerada a precursora dos computadores eletrônicos atuais.

A linguagem de programação é um método padronizado que permite comunicar instruções específicas para um computador. Existem várias linguagens, mas aqui adotaremos a linguagem R.

A programação envolve um conjunto de habilidades cognitivas, como o raciocínio lógico, habilidades matemáticas, capacidade de abstração e de lidar com diferentes tipos de dados. No entanto, quando aprendemos a programar, a primeira coisa que fazemos é criar um algoritmo, que funciona como uma receita. Nessa etapa descrevemos cada processo por meio de um pensamento lógico, do qual podemos utilizar diferentes abordagens para conseguir executar uma tarefa. Tudo aqui é feito baseado em tentativa e erro. Portanto, se você não conseguir executar o seu programa de primeira lembre-se que isso é normal e com o tempo você irá aperfeiçoar essas habilidades que funcionarão de forma quase que automática.

É importante ressaltar que esse aprendizado requer uma dedicação de tempo, explorando os recursos disponíveis e praticando diariamente. Vamos juntos descobrir os desafios que essa linguagem de programação pode nos proporcionar.

Essa apostila ficará disponibilizada na minha conta pessoal do github e você poderá acessar quando quiser: https://github.com/cientistacaotico/Introduction-to-R-programming

O software R

O R é uma linguagem bastante utilizada para análise de dados e como qualquer outra linguagem de programação você precisa saber o que está fazendo para que o comando faça sentido. A sua área de trabalho dentro do R possui uma interface interativa, no qual você escreve comandos e obtem o resultado. É uma linguagem funcional orientada a objetos. Isso significa que as análises e ações que você fizer no ambiente R são executadas por funções e essencialmente relacionadas a objetos.

O ambiente R é um sistema modular composto por pacotes que executam as tarefas. O pacote básico possui as funcionalidades necessárias para realizar os comandos matemáticos e estatísticos mais usuais. Além disso, o R é uma linguagem de programação em código aberto. Isso significa que qualquer um pode contribuir para essa comunidade, o que permite que novos pacotes possam surgir de acordo com a necessidade dos usuários.

Existem algumas comunidades onde podemos consultar sobre dúvidas e problemas que encontramos ao longo do caminho:

1. Blogs

- R-bloggers: https://www.r-bloggers.com/
- Revolution Analytics Blog: https://blog.revolutionanalytics.com/
- R-statistics: https://www.r-statistics.com/
- R Data Mining Blog:
- Simply Statistics: https://simplystatistics.org/ Embora não esteja diretamente vinculado ao R, este é um blog muito bem escrito sobre a aplicação prática das estatísticas. Big data e análises estatísticas são temas comuns encontradas nesse blog. Este é um blog que deve estar na sua lista de leitura.

2. Fóruns

- Nabble: https://r.789695.n4.nabble.com/
- Stack Overflow: https://stackoverflow.com/questions/tagged/r
- Cross Validated: https://stats.stackexchange.com/questions/tagged/r

- R-Help Mailing List: https://stat.ethz.ch/mailman/listinfo/r-help

3. Sites

- Crantastic: https://crantastic.org/
- Search the R Statistical Language: http://www.dangoldstein.com/search_r.html
- R-dir: https://r-dir.com/

Instalação R e Rstudio

É possível executar os seus comandos diretamente no software R ou utilizando uma interface gráfica (GUI, *Graphical User Interface*). Para instalar o software em seu computador basta acessar o link (https://cran.r-project.org/) e fazer o download de acordo com o seu sistema operacional. Para instalar, basta seguir as recomendações de instalação padrão. A interface gráfica que iremos utilizar aqui é o RStudio, que está disponível gratuitamente para Linux, Windows e Mac (http://www.rstudio.com/). O RStudio possui uma interface mais amigável em comparação com a interface básica do R e, por isso, pode ser interessante para quem está começando a programar agora. Lembre-se de instalar primeiramente o R e em seguida o RStudio.

Uso como calculadora

Uma das funcionalidades do software é o uso como calculadora. Aqui podemos escrever expressões matemáticas que são executadas facilmente. Vejamos alguns exemplos a seguir:

```
2+2 # Adição
```

```
## [1] 4
```

```
5-3 # Subtração
```

```
## [1] 2
```

```r
4*5 # Multiplicação
```

```
## [1] 20
```

```r
20/4 # Divisão
```

```
## [1] 5
```

```r
2^2 # Potência
```

```
## [1] 4
```

Como você pode perceber, a linguagem R permite que a gente crie comentários utilizando # para ajudar a entender o que foi realizado. Em muitos casos, criamos o nosso código e em algum momento precisamos retornar para corrigir algumas partes. Portanto, é indicado fazer anotações que ajude a identificar potenciais problemas nas suas linhas de comando.

```r
# As hashtags ajudam!!!
```

Note que ao executar uma linha de comando do script aparecerá o comando executado seguido do sinal de > e logo abaixo o resultado seguido de [1]. Por exemplo:

No script

2+2

No console

> 2+2

[1] 4

Observação: o número que vem dentro do [ ] representa a posição do seu resultado. Como no exemplo acima executamos uma expressão matemática que retorna apenas uma resposta,

então, temos [1]. Essa posição é mais importante quando temos um comando que retorna mais de um resultado ou quando estamos trabalhando com matrizes. Vejamos o seguinte exemplo:

```
2:8
```

```
## [1] 2 3 4 5 6 7 8
```

O comando acima cria uma sequência de números que variam de 2 a 8, e cada um deles representa uma posição no vetor. Nesse resultado temos sete itens, mas como todos couberam em apenas uma linha do console, então, o programa retorna a posição apenas do primeiro item. Caso contrário, se o comprimento do vetor extrapossasse a primeira linha do console, logo, apresentaria a posição do primeiro elemento da próxima linha. Por exemplo:

```
1:30
```

```
##  [1]  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25
## [26] 26 27 28 29 30
```

No próximo tópico veremos outras funcionalidades para as posições do vetor.

Atribuição (assign)

No tópico anterior, vimos que o resultado de um comando apareceu diretamente na tela, mas também podemos atribuir (gravar) esse resultado a um objeto. Para isso, utilizamos o símbolo de menor e hífen (com ou sem espaço antes do símbolo de menor e depois do hífen). Podemos também utilizar o atalho **Alt** + **-**

<-

O sinal **=** também funciona, mas para essa apostila adotaremos **<-** como o comando para atribuir uma informação a um objeto. Teste as duas opções e use a que mais te agradar.

Exemplo:

```
a <- 1:5
```

Com isso, atribuimos ao objeto **a** uma sequência que varia de 1 a 5 e, portanto, podemos retornar a esse objeto sempre que precisarmos. Basta escrever o nome do objeto.

```
a
```

```
## [1] 1 2 3 4 5
```

Podemos utilizar os objetos criados para executar operações matemáticas. Por exemplo:

```
a^2
```

```
## [1]  1  4  9 16 25
```

Nesse caso, elevamos todos os objetos do vetor à potência 2. Além disso, podemos gravar esse resultado em um novo objeto. Vamos nomear esse novo objeto como **b**.

```
b <- a^2
```

Agora nós temos dois vetores, **a** e **b**, e podemos brincar um pouco com as informações que estão contidos em cada um.

Temos que **b** é um vetor de comprimento igual a 5, e podemos alterar quaisquer itens utilizando a notação de posição do vetor [ ]. Por exemplo:

Valores originais:

```
b
```

```
## [1]  1  4  9 16 25
```

Substituindo o item que está na posição [3] pelo número 5:

```
b[3] <- 5
```

Esse é o nosso novo vetor **b**:

```
b
```

```
## [1]  1  4  5 16 25
```

Também podemos explorar outras possibilidades. Por exemplo: eu preciso que o item [5] do objeto **a** seja igual o item [1] do objeto **b**.

```
a
```

```
## [1] 1 2 3 4 5
```

```
b
```

```
## [1]  1  4  5 16 25
```

```
a[5] <- b[1]
a
```

```
## [1] 1 2 3 4 1
```

Além de alterar itens dentro do vetor, também podemos realizar operações matemáticas entre objetos. Por exemplo:

```
b-a
```

```
## [1]  0  2  2 12 24
```

Nesse caso, a subtração vai respeitar as posições dos vetores e, portanto, os vetores precisam ter o mesmo comprimento. Caso contrário, receberemos uma mensagem de alerta, e precisamos ter cuidado para que pequenos problemas não afetem o resultado do código. Faça o teste.

Agora você pode criar novas alterações utilizando outros objetos. Esse é o momento de você se divertir.

Tipos de objetos

Nos tópicos anteriores, vimos a palavra **vetor** sendo mencionada algumas vezes. Nessa sessão, iremos definir o que é um vetor e apresentar outros objetos frequentes na linguagem R. Esses objetos serão explorados no decorrer dos exemplos, mas para fins didáticos iremos conhecê-los agora.

1. **Vetores**: sequência composta por elementos numéricos ou caracteres (letras, palavras).
2. **Matrizes**: coleção de vetores organizados em linhas ou colunas (para esse tipo de objeto é fundamental que os elementos que compõem a matriz sejam da mesma classe, numérico ou categórico).
3. **Dataframe**: a estrutura desse objeto é semelhante a uma matriz, mas permite vetores com classes diferentes.
4. **Listas**: coleção ordenada de objetos. Em uma lista não há necessidade dos componentes serem da mesma classe ou comprimento. Por exemplo, podemos preencher uma lista com um vetor numérico, uma matriz, e até mesmo outra lista.
5. **Funções**: as funções criadas também são objetos no ambiente R. Veremos exemplos nas próximas sessões.

Utilizando funções existentes

Como já sabemos, o software R é um programa de código aberto (*Open Source*) e conta com o esforço colaborativo da comunidade de usuários, que podem criar e disponibilizar novas bibliotecas. Em cada biblioteca contém funções que executam uma determinada atividade e, por isso, esse tópico será dedicado para o uso dessas funções.

Ajuda