UNIVERSIDADE DE PASSO FUNDO
FACULDADE DE CIÊNCIAS ECONÔMICAS,
ADMINISTRATIVAS E CONTÁBEIS
CENTRO DE PESQUISA E EXTENSÃO DA FEAC
(www.upf.br/cepeac)

Texto para discussão

Texto para discussão Nº 03/2021

Introdução ao R-Studio

Andre da Silva Pereira
Luan Marca
Edson Jesus de Paiva Silva Filho

Passo Fundo - RS - Brasil

Introdução ao R-Studio

Andre da Silva Pereira
Luan Marca
Edson Jesus de Paiva Silva Filho

Universidade de Passo Fundo (UPF)
Programa de Pós-Graduação em Administração (**PPGAdm**)

Sumário

1. Introdução a Linguagem de Programação

Uma linguagem de computador e/ou programação, envolve uma sistematização de caracteres, símbolos e códigos. Esses, são utilizados como meios de sistematizar e estruturar a organização da rotina que se pretende analisar/calcular. Um Sistema eletrônico como tablet ou computador, possui tal aparato quando se faz qualquer que seja o uso.

1.1. Definição [1]

Sintática: Uma linguagem de programação é uma notação utilizada pelo programador para especificar ações a serem executadas por um computador.

Semântica: Uma linguagem de programação compreende um conjunto de conceitos que um programador usa para resolver problemas de programação.

1.2. Domínios de Programação [2]

O domínio da programação busca entender alguns passos fundamentais para sua organização e sistematização das rotinas que se pretende em sua formatação. São elas:

- Aplicações científicas
- Aplicações comerciais
- Inteligência artificial
- Sistemas básicos
- Aplicações Internet

2. Linguagem R [3]

O R é apresentado como sendo uma linguagem ambientada para análise estatística, para produzir gráficos, dentre outras funções. Foi desenvolvido pelos estatísticos Ross Ihaka e Robert Gentleman nos anos 90 quando da utilização e necessidade de estudos estatísticos. Como os programas mais utilizados são pagos e apresentam um elevado custo para sua aquisição, os mesmos buscaram o desenvolvimento de plataformas alternativas.

O R apresenta uma variedade de estatísticas (modelos lineares e não lineares, testes estatísticos, análise de séries de tempo, classificação, agrupamento, dentre outros) e tipos de análises gráficas que fornecem uma liberdade de escolha e de formatação das mesmas, em razão das rotinas apresentarem códigos abertos, ajustes e aperfeiçoamento das rotinas criadas. Ao contrário do Python que é utilizado em diversas áreas, a linguagem R é essencialmente estatística, hoje com crescimento do Big Data e do Machine Learning[4] essa linguagem vem se disseminando e por

---

[1] Escola Politécnica da Universidade de São Paulo
[2] Escola Politécnica da Universidade de São Paulo
[3] PERLIN, M.S. UFRGS - Disponível em (https://www.msperlin.com/padfeR/operacoes-basicas.html).
4 https://didatica.tech/curso-de-r-para-machine-learning-e-ciencia-de-dados-gratuito/

possuir uma praticidade na sua utilização a mesma constitui ferramenta perfeita para a análise de dados.

Ao utilizar o R, o usuário pode interagir com o programa pelo mouse ou através de comandos. O R e o RStudio, apresentam funcionalidades e praticidades pelo uso do mouse. Entretanto, como referendam estudiosos do programa, sua utilização é otimizada via inserção de comandos específicos. Quando um grupo de comandos é realizado, um script do R deve produzir algo importante no final de sua execução.

Em finanças e economia, isso pode ser o valor atualizado de um portfólio de investimento, uma análise estatística temporal de uma variável, o cálculo de um índice de atividade econômica, a performance histórica de uma estratégia de investimento, o resultado de uma pesquisa acadêmica, entre diversas outras possibilidades.

Além disso, o programa R possibilita ainda: exportar arquivos, inserção de figuras e até mesmo organizar arquivos de texto de maneiras distintas. Isso ocorre em razão da utilização e manuseio de outra ferramenta – denominada RMarkdown. Em XIE (2021)[5] a utilização do pacote *bookdown*, proporcionou a formatação do texto como se fosse um editor de texto comum utilizado comumente. Dessa forma, a editoração que estamos acostumados em outros editores de texto, podemos fazer a mesma tarefa agora baseada no RMarkdown.

O resultado de trabalhar e organizar seus trabalhos, estatísticos e de texto, no R e no RStudio, é o de criar e trabalhar com um script que vai produzir conteúdo para um relatório de dados. Para entender melhor o que foi dito acima, sugere-se, ao final da leitura, praticar os seguintes passos do script e rodar a rotina:

(1) instale R e RStudio no computador;

(2) copie o conteúdo de texto do link para um novo script ("File" -> "New File" -> "R Script");

(3) salve-o com um nome qualquer e,

(4) pressione control + shift + enter para executar o script inteiro.

3.1. Download e instalação do R (Windows)

Instalação Rápida:

- [Download R] (https://cran.r-project.org/bin/windows/base/);
- [Download R - Studio]; (https://download1.rstudio.org/desktop/windows/RStudio-1.4.1717.exe).

Importante! Instale nessa ordem: primeiro o R e só depois o RStudio. O RStudio é apenas uma interface para o R, e por isso, precisa encontrar a instalação do R para poder ser instalado.

Instalação Longa - R

---

[5] XIE, Y. **bookdown:** Authoring Books and Technical Documents with R Markdown. The R series. 2021.

- 1 - O primeiro passo é entrar na página do projeto em www.r-project.org;
- 2 - Do lado esquerdo da página clique sobre o link CRAN abaixo de Download, Packages;
- 3 - Uma nova página com uma série de links irá se abrir. Esses links são chamados de "espelhos" e servem para que você possa escolher o local mais próximo de onde você está para fazer o download do programa. Vamos selecionar o espelho da Universidade Federal do Paraná. Procure o [link] (http://cran-r.c3sl.ufpr.br) e clique sobre ele;
- 4 - Na seção Download and Install R, clique sobre o link Download R for Windows para baixar a versão para esse sistema;
- 5 - Clique sobre o link base;
- 6 - Clique sobre o link Download R 4.1.0 for Windows para fazer o download do arquivo R-4.1.0-win.exe;
- 7 - A instalação segue o formato padrão de instalação de programas no Windows, e, portanto, não são necessários maiores detalhes.

Realizada a instalação do R, abrir o programa requer clicar no ícone na sua área de trabalho ou até mesmo pelo menu de programas. Feito isso, você terá o primeiro contato com o sistema e visualização do R. A interface, de abertura, é o retrato inicial que o manipulador terá com R no que tange a linguagem de programação, aos recursos gráficos e aos blocos de manipulação do programa. Dessa forma, através dos códigos, que iremos elaborar via programação, teremos acesso para outras ferramentas de conteúdo do R.

Existem pessoas e grupos de estudo relacionados ao R que reestruturam suas interfaces. Objetiva-se com isso, aperfeiçoar/melhorar/facilitar ao programador um melhor controle e visualização do código que está escrevendo. Dentre tantas, disponíveis pela web, podemos citar a interface do Tinn-R. Recorrentemente, a plataforma R o do RStudio, disponibilizam novas interfaces e propriedades que facilitam e criam maneiras de programar e realizar as suas funções requeridas.

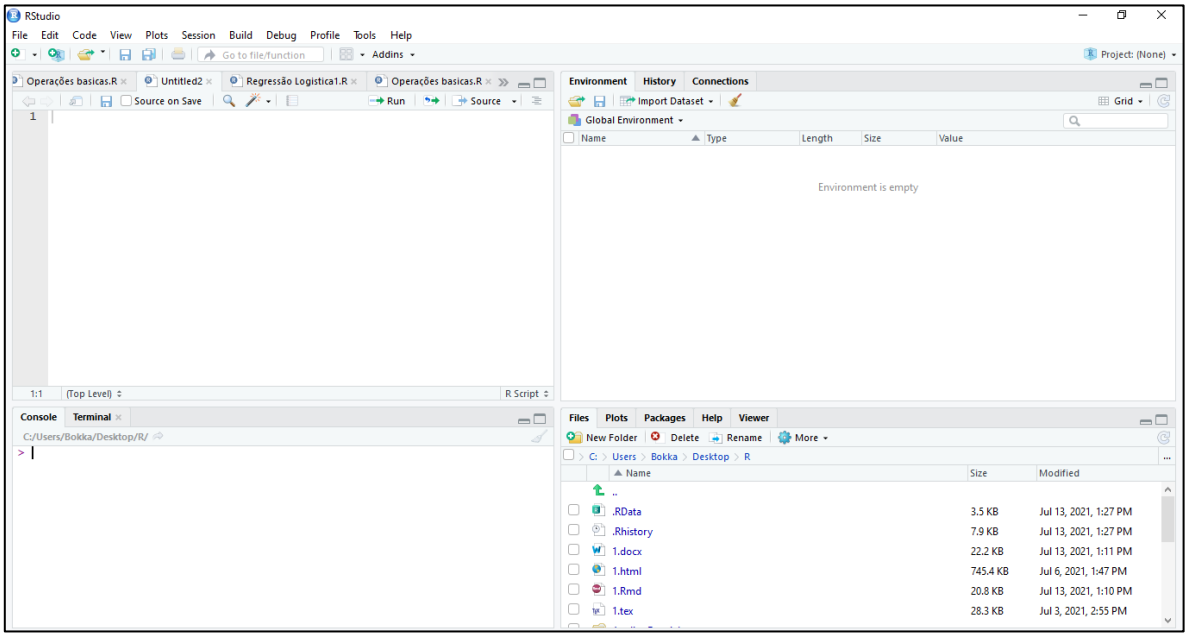
RStudio

- 1 - Para baixar o entre no [endereço] (www.rstudio.com);
- 2 - Clique no link Products > RStudio;
- 3 - Selecione a versão Desktop;
- 4 - Clique em DOWNLOAD RSTUDIO DESKTOP;
- 5 - Será exibida uma página com a recomendação para você baixar o RStudio 1.4.1717 – Windows (várias versões);
- 6 - Clicando nesse link, você irá baixar o arquivo RStudio- 1.4.1717.exe;
- 7 - Depois é só clicar e instalar da forma convencional do Windows.

Após a instalação, você pode abrir o programa pelo ícone. Ele estará pronto para ser utilizado.

3.2. Layout

Com relação às quatro janelas que são apresentadas no RStudio:



No topo a esquerda é onde você vai escrever seus códigos. Você pode ter várias janelas de códigos, para abrir mais uma basta ir ao sinal '+' em file. Depois que você escrever seu código, você deve selecionar a linha que gostaria de executar e clicar em ***run*** (executar a sua rotina criada).

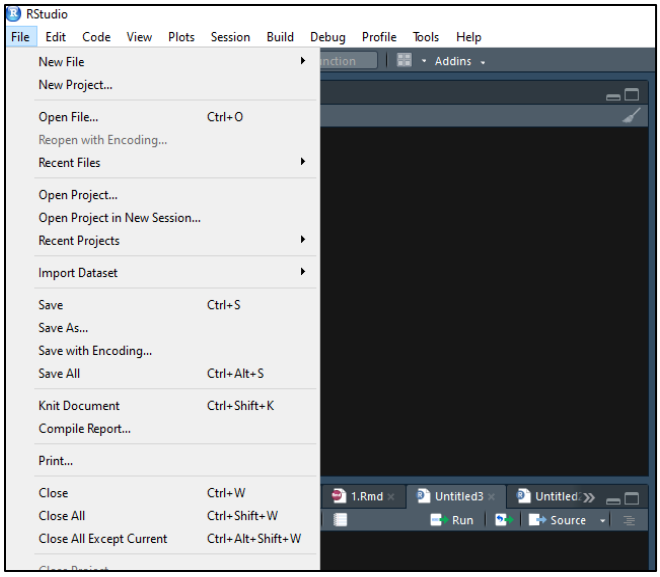
Ao executar o código (rotina) vai aparecer alterações no console. Na sua tela abaixo à esquerda. Pode-se escrever o código nessa tela também. Como sugestão dos programadores e dos usuários mais "profissionais", eles não recomendam manusear e alterar as rotinas aqui. O problema central é que o seu código pode ficar desorganizado.

Acima a direita você tem duas abas: environment e history. No caso de environment, recomendo que vocês não se preocupem muito. Basicamente, você vai ter seus objetos nessa aba, seus dados e variáveis criadas. Em history terá o histórico de seu script, ou seja, todo o código que for executando vai aparecer aqui. A utilidade desse fato, deve-se basicamente, para o caso de o RStudio fechar, poder salvar seu workplace (ao abrir não perderá o histórico). Sendo assim, ainda terá seu script, o que facilita entender o que estava trabalhando).

Abaixo a direita uma outra janela que vai facilitar muito sua vida. Existem cinco abas: files, plots, packages, help e viewer. Em files você terá o endereço do seu working directory. Basta executar getwd () que você saberá que endereço é aquele na aba. No r Studio você pode mover, renomear e deletar arquivos no seu computador. Esse diretório é onde você salva seus scripts e output no R. O recurso plots onde você vai visualizar os seus gráficos. Em packages você terá acesso para uma lista de pacotes (selecionar/instalar). Finalizando, o comando help ou o comando ? será o ponto de ajuda e de como manusear os vários pacotes e comandos do RStudio. Recurso esse muito interessante para os iniciantes e para aqueles que esquecem rápido.

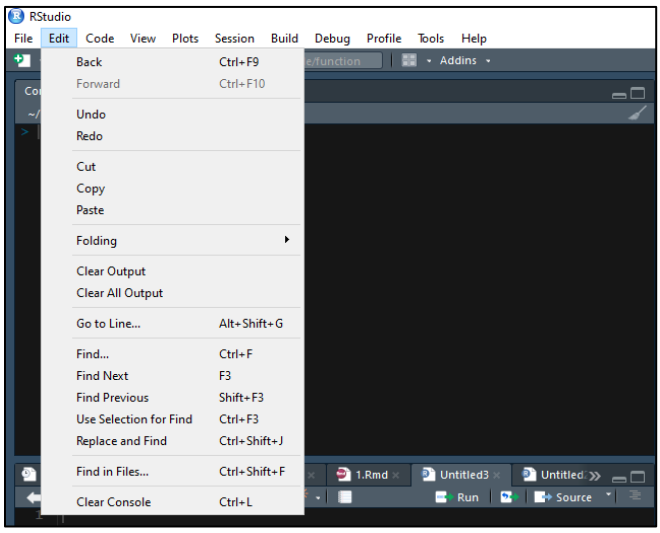
3.3. Menus

Sempre que usar o R-Studio, você verá a barra de menus na parte superior da tela. Composto por onze menus suspensos, a barra de menus o ajudará a executar os comandos adequados no R-Studio.



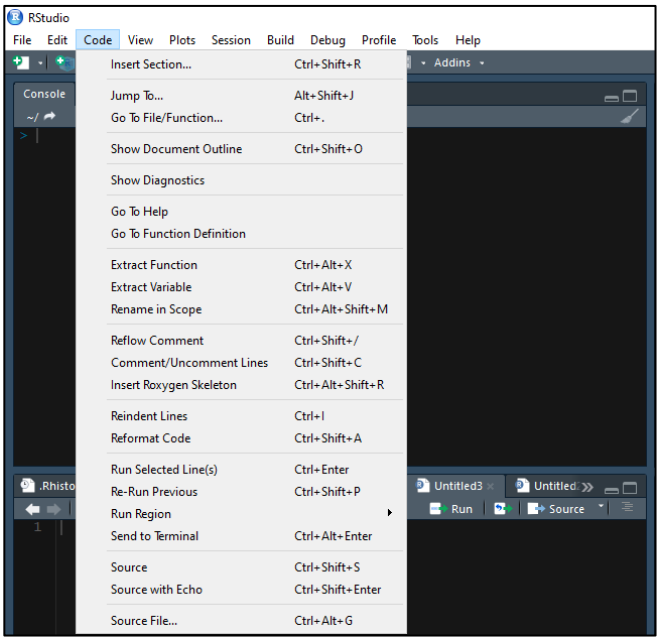
Menu Arquivo (*File*)

O menu file (Arquivo) do R-Studio contém muitas das funções padrão de um menu Arquivo de qualquer outro software ou programa - *New File* (novo arquivo), *Open File* (abrir arquivo), *Save* (salvar), *Print* (imprimir). Um recurso importante a ser mencionado no menu Arquivo do RStudio é o comando *Knit Document*. Esse comando converte seu arquivo R-Studio em um arquivo HTML, um documento PDF ou um documento do Microsoft Word. Isso torna o trabalho que você fez fácil de ser lido e compartilhado em uma variedade de configurações. O menu Arquivo também permite que os usuários importem conjuntos de dados de softwares ou programas externos.



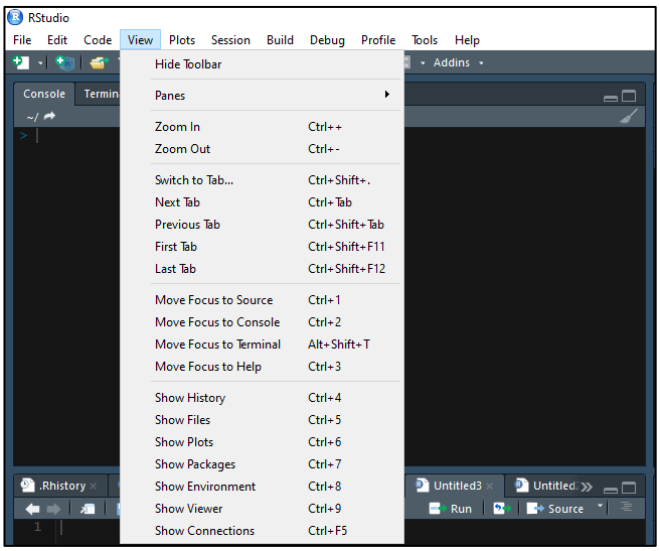
Menu Editar (*Edit*)

O menu Editar provavelmente será um menu usado com frequência. Aqui os usuários podem Recortar, Copiar, Colar, Desfazer e Refazer. O menu Editar também é muito útil para localizar códigos ou comandos usados anteriormente. Com os comandos *Go to Line…*, *Find…* e *Replace and Find*, os usuários podem editar ou substituir rapidamente o código RStudio. O menu Editar também possui o comando clean console, que permite aos usuários limpar o console.



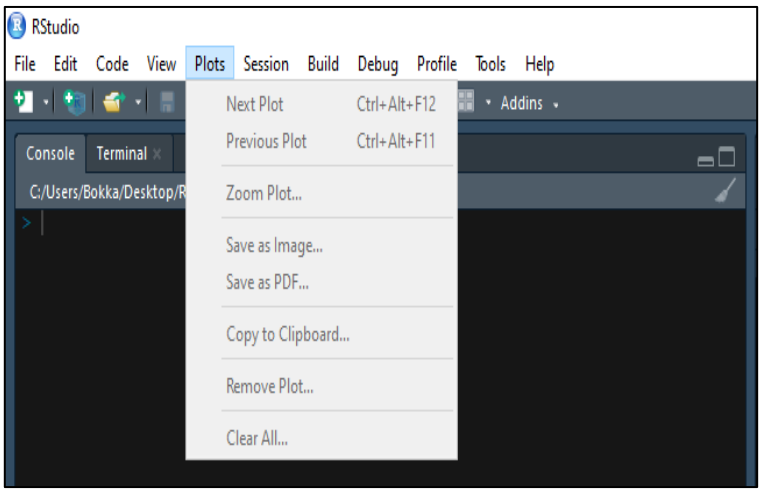
Menu Código (*Code*)

O menu Código possui comandos usados para trabalhar com código. Semelhante ao menu Editar, Code oferece *Jump To ...* para acesso rápido a um código específico. Aqui você pode retrabalhar a aparência do seu código com Reformat code e onde funções e variáveis podem ser removidas com *Extract Function* e *Extract Variable*. O menu Código também fornece comandos para executar o código com *Run select Line(s)*, *Re-Run Previous*, e *Run Region*. O código RStudio pode ser executado diretamente na guia *Source*, selecionando a seta verde com cada pedaço de código ou com o botão Executar no canto superior direito da guia Código-fonte, ou os usuários podem especificar o código que desejam executar com um dos comandos de o menu Código.



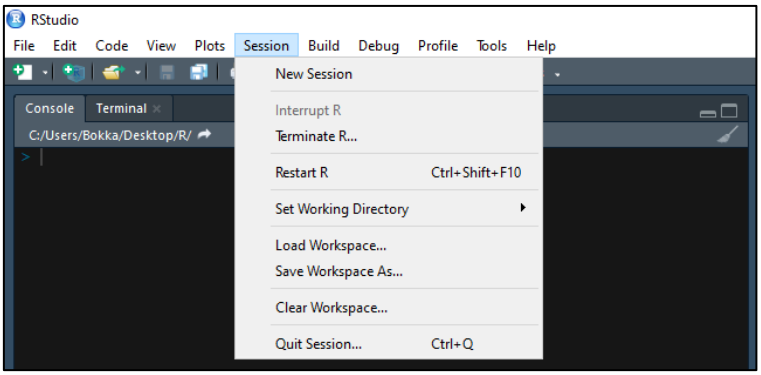
Menu Visualizar (*View*)

O menu *View* concentra-se em como o usuário vê sua área de trabalho RStudio. Este menu permite que você escolha qual guia deseja visualizar, bem como onde o mouse deve ser focado. Painéis concentra-se nas habilidades de zoom do RStudio e permite que os usuários ampliem uma guia específica.



Menu Gráficos (*Plots*)

O menu Plots funciona especificamente com plotagens feitas no RStudio. Este menu permite que você alterne rapidamente entre os gráficos e aplique zoom no gráfico para aumentar a clareza. É aqui que você pode escolher salvar o seu gráfico como uma imagem ou PDF, bem como onde você pode excluir os gráficos indesejados com *Remove Plot*.

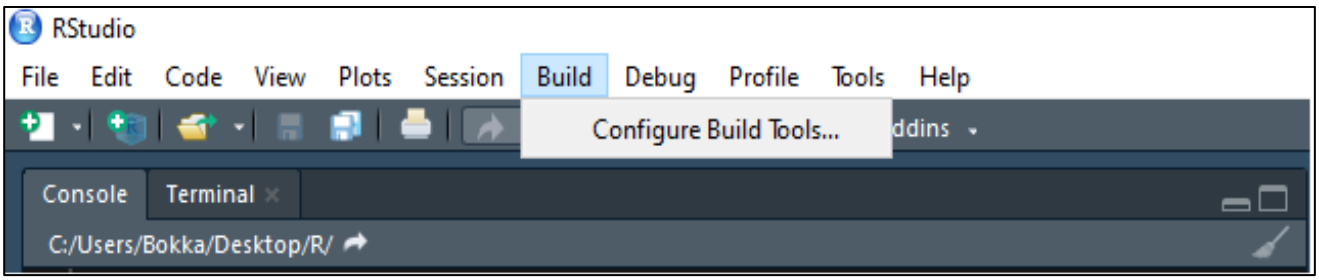


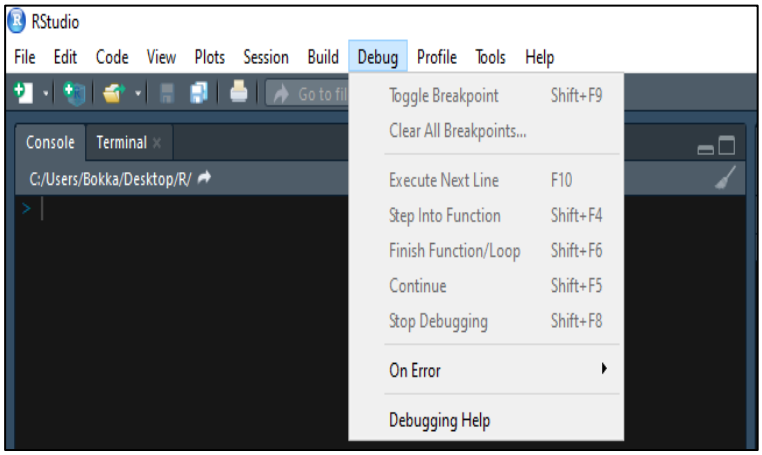
Menu Sessão (*Session*)

O menu *Session* permite que os usuários abram uma Nova Sessão… e Saiam da Sessão…. Aqui você também pode encerrar o R, ou reiniciar o R se uma atualização precisar ocorrer ou o programa precisar ser atualizado.

Menu Construir (*Build*)

O menu *Build* apresenta apenas um comando: *Configure Build Tools*. As ferramentas de construção só podem ser configuradas dentro de um projeto RStudio e são uma forma de empacotar e distribuir código R.



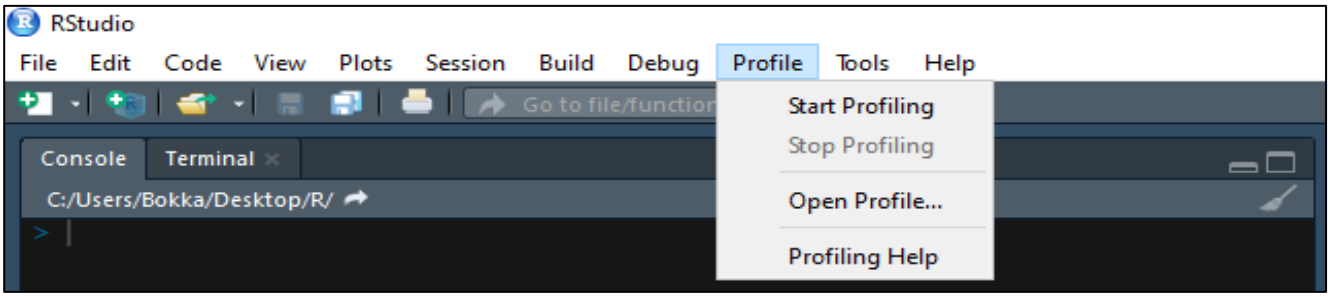


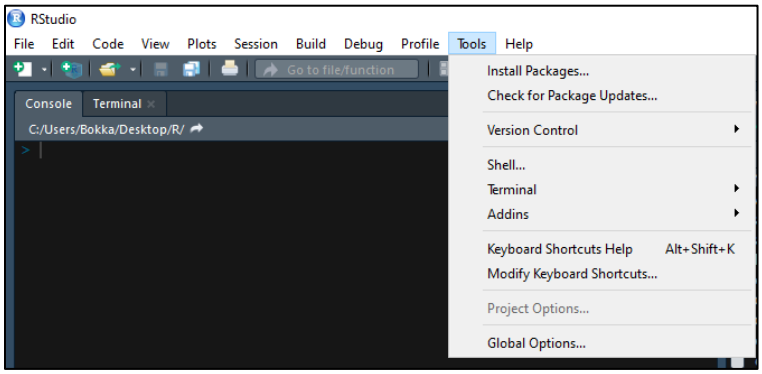
Menu *Debug*

O menu *Debug* é o que garante que seu código esteja funcionando corretamente. Quando houver um erro em sua codificação ou comandos, o menu *Debug* apontará os erros e permitirá que você decida se deseja continuar trabalhando com seu código RStudio. Você pode escolher como deseja ser notificado sobre um erro usando a opção *On error*. Se precisar de ajuda adicional para corrigir um erro, selecione *Debugging Help*.

Menu Perfil (*Profile)*

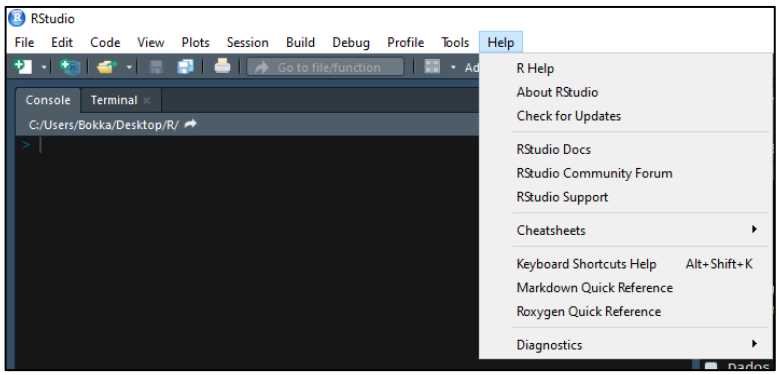
O menu *Profile* permite que os usuários entendam melhor o que exatamente o RStudio está fazendo. A criação de perfil fornece uma interface gráfica para que os usuários possam ver no que o RStudio está trabalhando nos momentos em que você espera que uma saída apareça. Ao criar o perfil, os usuários podem aprender o que está tornando seu código mais lento, para que possam ajustar as partes a fim de tornar o código executado mais rápido.





Menu Ferramentas (*Tools)*

O menu *Tools* fornece informações sobre a versão atual do RStudio sendo executado, bem como o local onde os Pacotes e Addins podem ser instalados. As ferramentas também auxiliam os usuários do RStudio com os atalhos do teclado e permitem que os usuários modifiquem os atalhos do teclado para atender às necessidades e preferências individuais.



Menu Ajuda (*Help*)

O menu *Help* fornece informações para ajudar os usuários a maximizar sua proficiência no RStudio. Links diretos para a Ajuda do RStudio são fornecidos, bem como Cheatsheets feitos por profissionais do RStudio e uma seção sobre diagnósticos para permitir que o usuário veja o que está ocorrendo no RStudio.

3.4. Atalhos

- **CTRL+ENTER:** avalia a linha selecionada no script. O atalho mais utilizado.
- **ALT-:** cria no script um sinal de atribuição (<-). Você o usará o tempo todo.
- **CTRL+SHIFT+M:** (%>%) operador pipe. Guarde esse atalho, você o usará bastante.
- **CTRL+1:** altera cursor para o script.
- **CTRL+2:** altera cursor para o console.
- **CTRL+ALT+I:** criar um chunk[6] no RMarkdown.
- **CTRL+SHIFT+K:** compilar um arquivo no RMarkdown.
- **ALT+SHIFT+K:** janela com todos os atalhos disponíveis.
- **CTRL+L:** Apagar a tela.

3.5. Projetos

Dentre as tarefas que o R e o RStudio podem efetuar, uma delas é a criação de projetos. Um projeto nada mais é do que uma pasta no seu computador. Nessa pasta, estarão todos os arquivos que você vai usar ou criar para a sua análise que pretende fazer.

O principal argumento para trabalhar com projetos é no que se refere quanto a sua organização. Com sua execução e organização, fica muito mais fácil importar bases de dados para dentro do R, criar análises e compartilhar os trabalhos pelo programador e entre grupos de estudo (compartilhamento de scripts).

---

[6] **Chunks** são os blocos de código que são executados quando o documento é compilado.

3.6. Repositórios CRAN

(Acessar aqui)

3. R Básico

A partir daqui começamos a ver algumas das principais funções do R-Studio.

4.1. Pedindo Ajuda (Help) ou (?)

A manipulação da linguagem R e, por conseguinte do RStudio, é bastante intuitiva. Ela abre possibilidades de se fazer muitas tarefas, principalmente para o programador iniciante. O processo de aprendizagem é evolutivo no sentido de que você pode ir galgando conhecimento e coragem para procurar aprender a manusear e utilizar as funções básicas e, que porventura, possam ser multiplicadas com outras funções ao mesmo tempo.

Tendo dificuldade no entendimento e manuseio de um comando, função, pacote e até mesmo script, o usuário pode interagir com a comunidade do R (nas suas mais variadas redes sociais) que é bastante colaborativa na busca de ajudar e compreender as dúvidas dos usuários do R. Vamos apresentar, mais adiante, algumas maneiras de interagir com essas comunidades.

Para pedir ajuda basta colocar o ponto de interrogação (?) ou do help antes do nome da função. Vamos para um exemplo prático.

Exemplo:

```
?6 * 9
?6 / 2
?6 - 1
```

Fazendo isso o R abrirá uma janela com todas as informações sobre a função. Além disso, devemos lembrar que os dados possuem sua tipificação, isto é, podem ser qualificados. Sendo assim os dados podem ser: NUMÉRICOS (podem ser feitos cálculos), CARACTERES (não possuem números) e LÓGICOS (podem ser verdadeiro ou falso).

Os dados numéricos o manipulador dos dados irá inserir normalmente os números nas linhas que o usuário estiver organizando seus dados logicamente. Entretanto, quando estiver manipulando dados – caracteres – uma aspa (“ “) será requerida ao programador. Por exemplo, (“ ? “) ou (‘ ? ’). No caso de os dados envolverem lógica, TRUE (T) ou FALSE (F) ou até mesmo (T) sendo TRUE (1) ou (F) FALSE sendo (0).

4.2. R como Calculadora

O papel do Console no R é executar os comandos que foram digitados pelo programador. Ele avalia o código que foi digitado e devolve a saída correspondente. Se der certo a programação, uma mensagem de acerto ou de erro aparecerá logo a seguir à função digitada. Podemos, agora, iniciar nossa programação no R, transformando o mesmo em apenas uma calculadora.

Na construção da hierarquia das operações aritméticas, os parênteses, a exponenciação, a multiplicação e a divisão e a adição e a subtração, irão envolver símbolos como utilizados na matemática e empregados agora pelo R.

QUADRO 1 - Símbolos

Símbolo matemático	Símbolo em R
=	= =
≠	!=
≻	≻
≺	≺
≥	≻=
≤	≺=

Fonte: autor.

As operações matemáticas simples são:

Soma

```
12 + 10
```

```
 [1] 22
```

Subtração

```
36 - 14
```

```
 [1] 22
```

Divisão

```
120 / 5
```

```
[1] 24
```

Multiplicação

```
45 * 20
```

```
 [1] 900
```

Potência

```
4**2
```

```
 [1] 16
```

Testes Lógicos

Poder fazer qualquer tipo de operação lógica é um dos motivos pelos quais programar nos deixar mais eficientes. Dê bastante atenção a elas, pois usaremos comparações lógicas o tempo todo.

Uma operação lógica nada mais é do que um teste que retorna verdadeiro (TRUE) ou falso (FALSE). No R e em outras linguagens de programação, esses dois valores recebem uma classe especial: logica.

O verdadeiro no R vai ser representado pelo valor TRUE e o falso pelo valor FALSE. Esses nomes no R são reservados, isto é, você não pode chamar nenhum objeto de TRUE ou FALSE. Ou seja, o TRUE sempre irá assumir o valor 1, enquanto o FALSE vai ser o valor 0.

Mais um exemplo para praticar:

Igual - Usar dois símbolos de =

```
5 == 4

 [1] FALSE

5 == 5

[1] TRUE
```

Diferente – Usar !=

```
5!= 4

 [1] TRUE

5!= 5

[1] FALSE
```

Menor - usar <

```
4 < 2

 [1] FALSE

4 < 6

 [1] TRUE
```

Maior - usar >

```
4 > 2

 [1] TRUE

4 > 6

 [1] FALSE
```

Menor ou Igual usar <=

```
4 <= 4
```

```
[1] TRUE
```

```
4 <= 3
```

```
 [1] FALSE
```

```
4 <= 5
```

```
 [1] TRUE
```

Maior ou Igual usar =>

```
4 >= 4
```

```
 [1] TRUE
```

```
4 >= 3
```

```
 [1] TRUE
```

```
4 >= 5
```

```
[1] FALSE
```

Lógica

```
6 == 6
```

```
 [1] TRUE
```

```
6 == 6 & 6 == 7
```

```
 [1] FALSE
```

```
5 > 4 & 10 < 11
```

```
[1] TRUE
```

4.3. Comentários

Comentários são inseridos no R para facilitar a organização dos códigos, para adicionar um comentário, basta usar # antes de começar a escrever. Tudo que estiver após # não será executado. Ou seja, ele não é parte do programa e sim um texto informativo como se fosse um guia.

4.4. Objetos

O R permite salvar valores dentro de um objeto. Um objeto é simplesmente um nome que guarda um valor. Para criar um objeto, utilizamos o operador **<-**.

Armazenamento de números

Exemplo:

```
a <- 2
x <- 5
y <- 185
idade <- 33
distancia <- 12
```

Nos exemplos acima, salvamos o valor **2** em **a**, **5** em **x**, **185** em **y**, **33** em **idade** e **12** em **distancia**[7]. Sempre que avaliarmos os objetos, o R vai devolver os respectivos valores.

Os objetos criados podem interagir, isso se dá da seguinte forma:

```
Exemplo1 <- a + x
Exemplo1

[1] 7

Exemplo2 <- y - x
Exemplo2

[1] 180

Exemplo3 <- y / idade
Exemplo3

[1] 5.606061

Exemplo4 <- x * distancia
Exemplo4

[1] 60
```

Armazenamento de caracteres

Além de números, objetos também podem guardar caracteres, possibilitando trabalhar com strings (letras, palavras, frases etc.). Para isso, é preciso usar aspas.

Exemplo:

```
b <- "Layne Staley"
d <- "Eddie Vedder"
nome <- "Phil Anselmo"
Nome2 <- "Joey Ramone"
```

[7] A palavra está sem acento porque um script no R não vai acentos.

Existem algumas regras para dar nomes aos objetos. A mais importante é: o nome deve começar com uma letra. O nome pode conter números, mas não pode começar com números. Você pode usar pontos. e underlines _ para separar palavras.

Permitido

```
z <- 1
x5 <- 15
Item <- 7
Meu_Item <- 33
Meu.Item <- 45
Salario <- 4540.59
Horas <- 220
Tempo_de_Empresa <- 15
```

Exemplo de interação entre os objetos, calculando o valor por hora de trabalho.

```
Valor_Hora <- Salario/Horas
Valor_Hora
```

```
## [1] 20.63905
```

Não Permitido

2x <- 6
_Item <- 4
Meu-Item <- 4

O R diferencia letras maiúsculas e minúsculas, isto é, **b** é considerado um objeto diferente de **B**.

Armazenamento Lógico

A variável salário é maior que a variável Tempo de Empresa?

```
Logico1 <- Salario > Tempo_de_Empresa
Logico1
```

```
 [1] TRUE
```

A variável Salário é menor que a variável Horas?

```
Logico2 <- Salario < Horas
Logico2
```

```
 [1] FALSE
```

4.5. Funções

Quase todas as funções do R seguem uma mesma estrutura. É preciso escrever o nome da função, abrir um parêntese e fornecer um ou mais argumentos para ela ser executada e fechar os parênteses.

Função c(): Combina todos os seus argumentos num só vetor R, que é parecido a uma matriz em linguagens de programação. 8

Exemplo 1: Função c()

```
Dados <- c(23 ,43 ,25 ,54 ,77)
c1 <- c(a, b, d)
Conjunto <- c(d, Item, Dados)
```

Utilizamos aqui a função **c**, esta é uma função genérica que combina seus argumentos.

Exemplo 2:

Função summary(): [9]Mostra o resumo. Geralmente um resumo estatístico, com média, mediana, min, max, etc...

```
summary(Dados)

    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
   23.0    25.0    43.0    44.4    54.0    77.0

summary(Conjunto)

   Length     Class      Mode

        7 character character
```

Essa função é usada para produzir resumos dos resultados de várias funções de ajuste de modelo. A função invoca métodos específicos que dependem da classe do primeiro argumento.

4.6. Vetores

Em especial, pensando em análise de dados, precisamos estudar vetores pois cada coluna de um dataframe será representada como um vetor.

Vetores no R são apenas conjuntos indexados de valores. Para criá-los, basta colocar os valores separados por vírgulas dentro de uma função **c()**.

Exemplo:

```
vetor1 <- c(1, 5, 3, -10)
vetor2 <- c("a", "b", "c")
Notas <- c(4.5, 5.7, 8.7 ,9.5, 5,3.1)
Peso <- c(85, 91.5, 72, 65, 105)
```

[8] R Documentation

[9] R Documentation

4.7. Listas

Listas são objetos muito importantes dentro do R. Primeiro porque todo data frame é uma lista. Segundo porque elas são bem parecidas com vetores, mas com uma diferença essencial: você pode misturar diferentes classes de objetos dentro dela

Para exemplificar criamos uma variável do tipo numérica, outra com caracteres e por fim usamos a função list para mesclar números com caracteres. A função str exibe de forma compacta a estrutura interna dos objetos.

Variável numérica:

```
Idade1 <- c(17, 25, 32, 45, 58, 65, 51, 85, 44)
str(Idade1)

 num [1:9] 17 25 32 45 58 65 51 85 44
```

Variável com caracteres:

```
Nome <- c("Jhonny", "joey", "Eddie", "Phil", "James", "Ozzy", "Dio", "Gen
e", "Halford")
str(Nome)

  chr [1:9] "Jhonny" "joey" "Eddie" "Phil" "James" "Ozzy" "Dio" "Gene"
```

Criando lista:

```
Caracteristicas <- list(17, 25, 32, "Jhonny", "Joey", "Eddie")
str(Caracteristicas)

 List of 6
  $ : num 17
  $ : num 25
  $ : num 32
  $ : chr "Jhonny"
  $ : chr "Joey"
  $ : chr "Eddie"
```

Conforme ficamos mais e mais proficientes na linguagem R, as listas passam a ficar cada vez mais frequentes.

4.8. Matriz

Toma um vetor e o transforma em matriz, dividindo os dados presentes no vetor em várias linhas da matriz.

Exemplo:

```
matrix(data = Idade1, nrow = 3, ncol = 3)

##      [,1] [,2] [,3]
## [1,]   17   45   51
## [2,]   25   58   85
## [3,]   32   65   44
```

- **data** é a fonte dos dados (por exemplo, um vetor)
- **nrow** é o número de linhas desejadas
- **ncol** é o número de colunas desejadas

Função rbind()

Conecta vetores formando linhas de uma matriz com cada vetor.

Exemplo:

Criando os vetores numéricos.

```
Minutos <- c(21, 52, 45, 20, 47)
Segundos <- c(14, 7, 54, 25, 32)
Horario <- c(14, 21, 8, 7, 14)
```

Chamando a função rbind

```
rbind(Minutos, Segundos, Horario)
```

```
          [,1] [,2] [,3] [,4] [,5]
   Minutos    21   52   45   20   47
   Segundos   14    7   54   25   32
   Horario    14   21    8    7   14
```

São mostrados os nomes dos vetores minutos, segundos e horário nas linhas do vetor criado.

Função cbind()

Conecta vetores formando colunas de uma matriz com cada vetor.

Exemplo:

```
cbind(Minutos, Segundos, Horario)
```

```
     Minutos Segundos Horario
[1,]      21       14      14
[2,]      52        7      21
[3,]      45       54       8
[4,]      20       25       7
[5,]      47       32      14
```

Os dados são dispostos em colunas, em vez de linhas, como ocorreu com a função rbind(). Em vez de serem mostrados os números de índice das colunas, são mostrados os nomes dos vetores empregados.

4.9. Data frames

Os data frames são de extrema importância no R, pois são os objetos que guardam os nossos dados. Eles são equivalentes a uma tabela do SQL ou uma planilha do Excel.

A principal característica de um dataframe é possuir linhas e colunas.

Veja o exemplo abaixo:

```
Item.2 <- data.frame(nome = c("João", "Maria", "José"),
                     sexo = c("M", "F", "M"),
                     salario = c(1000, 1200, 1300),
                     stringsAsFactors = FALSE)
```

```
Item.2
```

```
  nome sexo salário
1 João    M    1000
2 Maria   F    1200
3 José    M    1300
```

Selecionando Variáveis do data frame

Para selecionar uma coluna basta chamar o nome da data frame seguido do número da coluna.

Exemplo 1:

```
Item.2[1]
```

```
    nome
 1  Joao
 2 Maria
 3  Jose
```

Exemplo 2:

```
Item.2[2]
```

```
   sexo
 1    M
 2    F
 3    M
```

Exemplo 3:

```
Item.2[3]
```

```
   salario
 1    1000
 2    1200
 3    1300
```

Outra forma de selecionar uma coluna é usando **$.**

Exemplo 1:

```
Item.2$nome
```

```
 [1] "João"  "Maria" "José"
```

Exemplo 2:

```
Item.2$sexo

 [1] "M" "F" "M"
```

Exemplo 3:

```
Item.2$salario

 [1] 1000 1200 1300
```

Excluindo Colunas no data frame

Para excluir uma coluna basta chamar o dataframe, selecionar a coluna usando $, seguido da função **NULL.**

Exemplo: Excluindo a coluna nome

```
Item.2$nome <- NULL
Item.2

   sexo salario
 1    M    1000
 2    F    1200
 3    M    1300
```

Criando uma nova variável na data frame:

Para inserir uma nova coluna basta chamar o nome da data frame, seguido $, na sequência escreva o nome da nova coluna e dos itens que vão compor ela.

Exemplo:

```
Item.2$nome <- c("DeeDee Ramone", "Janis Joplin", "Axl")
Item.2

   sexo salário          nome
 1    M    1000 DeeDee Ramone

 2    F    1200 Janis Joplin
 3    M    1300           Axl
```

4.9. Filtros em Variáveis

Filtrando variáveis você pode obter apenas parte de seu conteúdo, podendo assim realizar a sua edição.

Exemplo:

Primeiramente criamos uma variável.

```
Vogais <- c("a", "e", "i", "o", "u")
```

Acessando um conteúdo específico dessa variável. Para isso deve-se chamar a variavel e usar colchetes [].

Exemplo:

```
Vogais[3]
```

```
[1] "i"
```

Acessando todos os item com exessão de um, para isso use o sinal de menos antes do numero do item dentro de colchetes [].

Exemplo:

```
Vogais[-3]
```

```
[1] "a" "e" "o" "u"
```

Filtrando os dados entre uma posição e outra, para isso use dois pontos entre as posições dentro de colchetes [].

Exemplo:

```
Vogais[3:5]
```

```
[1] "i" "o" "u"
```

Considerando o comprimento das variáveis, chamando a função length.

Exemplo:

```
length(Vogais)
```

```
[1] 5
```

```
Vogais[3: length(Vogais)]
```

```
[1] "i" "o" "u"
```

Filtrando uma variável considerando uma Terceira.

Exemplo:

```
o <- 3
u <- 5
 Vogais[o:u]
```

```
[1] "i" "o" "u"
```

Acessando retornos através de condições, perguntando:

Na variável Vogais eu tenho a letra e?

```
Vogais=="e"
```

```
[1] FALSE  TRUE FALSE FALSE FALSE
```

Na variável Vogais existem letras diferentes de e?

```
Vogais != "e"
```

```
[1]  TRUE FALSE  TRUE  TRUE  TRUE
```

Exemplo com números:

Criando a variável:

```
A <- c(1, 2, 3, 4, 5, 6)
```

O A é maior que 2?

```
A > 2
```

```
 [1] FALSE FALSE  TRUE  TRUE  TRUE  TRUE
```

O A é menor que 2?

```
A < 2
```

```
 [1]  TRUE FALSE FALSE FALSE FALSE FALSE
```

4.10. Filtros em data frames

Utilizaremos o dataframe criado anteriormente.

Item.2

```
   sexo salario          nome
1    M    1000 DeeDee Ramone
2    F    1200  Janis Joplin
3    M    1300           Axl
```

A ideia de filtros é similar a de matrizes, utilizamos colchetes para apontar a linha ou coluna a ser mostrada.

Chamando a terceira coluna do data frame, nome.

Exemplo:

```
Item.2[3]
```

```
            nome
  1 DeeDee Ramone
  2 Janis Joplin
  3          Axl
```

Agora chamaremos a terceira linha do data frame usando a mesma estrutura usada acima, porém, adicionando uma virgula.

Exemplo:

```
Item.2[3,]
```

```
   sexo salario nome
 3    M    1300  Axl
```

Para chamar mais de uma coluna do dataframe utilizamos o número das colunas.

Exemplo: Chamando a coluna 1 e 2 do data frame

```
Item.2[1:2]
```

```
   sexo salario
1    M    1000
2    F    1200

3    M    1300
```

A partir disso podemos ir elaborando mais os filtros Chamando as 2 primeiras linhas do dataframe nas colunas de 1 a 2.

Exemplo:

```
Item.2[1:2, 1:2]
```

```
   sexo salario
 1    M    1000
 2    F    1200
```

Filtrando Variàveis do data frame posição 2 da variável 3.

Exemplo 1: Utilizando a posição numérica.

```
Item.2[2,3]
```

```
 [1] "Janis Joplin"
```

Exemplo 2: Chamando a variável utilizando o cifrão ($).

```
Item.2$nome[1]
```

```
 [1] “DeeDee Ramone”
```

Exemplo 3: Chamando um conjunto de posições, item 1 e 2 da variável nome.

```
Item.2$nome[1:2]
```

```
 [1] "DeeDee Ramone" "Janis Joplin"
```

4.11. Condicionais na linguagem R

Usando If, loop FOR e loop WHILE para condicionar ações. Ou seja, consiste em enviar instruções ao R, essa instrução só será atendida se determinada condição for atendida.

Usando (if)

Exemplo 1: Se 5 for igual a 5 então some 6 + 6.

```
if (5 == 5) 6 + 6
```

```
 [1] 12
```

Exemplo 2: Caso a condição não seja atendida a frase "Condição não atendida" aparecerá. A estrutura do codigo deve ser feita usando parenteses () e chaves {}. O if executará quando a condição for verdadeira tudo que estiver dentro dos colchetes Se a condição (5 for igual a 5) então some {6 + 6}, caso não seja, else {Condição não atendida}.

```
if (5 == 4) {
   6 + 6
 } else {
  "Condição não atendida"
   }
```

```
## [1] "Condição não atendida"
```

Exemplo 3: Primeiro vamos criar 2 variáveis

```
IDADE <- c(33, 30)
NOMES <- c("Joey", "Kurt")
```

Criando Um dataframe com as variáveis criadas:

```
DF <- data.frame(IDADE, NOMES)
DF
```

```
   IDADE NOMES
 1    33  Joey
 2    30  Kurt
```

Usando a informação da coluna **IDADE** condicionada a variável **NOMES**. Quem é mais novo, Kurt ou Joey ?

```
if (DF$IDADE [DF$NOMES == "Joey"] > DF$IDADE[DF$NOMES == "Kurt"]) {
   "Mais novo: Kurt"
 } else { "Mais novo: Joey"
   }
```

```
 [1] "Mais novo: Kurt"
```

Usando (for)

Executando uma ação por determinado numero de vezes Primeiro vamos criar um data frame um pouco mais complexa.

```
IDADE.2 <- c(33,45,27,59,35,49)
 NOMES.2 <- c("Joey","Ozzy","James","Raulzito","Gene","Halford")
 DF2 <- data.frame(IDADE.2,NOMES.2)
 DF2
```

```
  IDADE.2  NOMES.2
1      33     Joey
2      45     Ozzy
3      27    James
4      59 Raulzito
5      35     Gene
6      49  Halford
```

Usando for para Executar uma ação por determinado numero de vezes Para cada item dentro da variável NOMES.2 será executado um Print(i), mostrando o conteudo do item.

```
for (i in IDADE.2) {
   print(i)
 }
```

```
[1] 33
[1] 45
[1] 27
[1] 59
[1] 35
[1] 49
```

Quem é mais velho? Para cada linha da variável IDADE.2 será executada uma ação (if). Rodando o loop o maior número vai sendo salvo na variável V.

```
V <- 0
 for (i in DF2$IDADE.2){
   if (i > V) {V <- i}
 }
 DF2$NOMES.2[DF2$IDADE.2 == V]
```

```
 [1] Raulzito
 Levels: Gene Halford James Joey Ozzy Raulzito

 [1] 59
```

Usando (While)

Executando uma ação enquanto algo for verdade. Imprimindo um valor de L por 10 vezes.

```
L <- 0
 while (L < 10) {
   print(L)
   L <- L+1
 }
```

```
[1] 0
[1] 1
[1] 2
[1] 3
[1] 4
[1] 5
[1] 6
[1] 7
[1] 8
[1] 9
```

Agora usando o dataframe criado anteriormente. Não permitir que a soma das idades seja maior que 100.

Criando as variáveis para carregar os valores:

```
Y <- 0
A <- 0
cont <- 0
Idades100 <- 0
```

Montando o Script:

```
while (Y < 100) {
   cont <- cont+1
   Idades100[cont] <- DF2$IDADE.2[cont]
   A <- A+DF2$IDADE.2[cont]
   Y <- A+ DF2$IDADE.2[cont+1]
 }
```

Resultado:

```
IDADE.2
```

```
## [1] 33 45 27 59 35 49
```

```
Idades100
```

```
## [1] 33 45
```

```
sum(Idades100)
```

```
## [1] 78
```

De posse dessas informações iniciais quanto ao entendimento do R e de como funcionam os passos iniciais, tanto da programação quanto das funções, podemos buscar uma maior interação com o R. Com os scripts, as funções, a estatística e os mais variados tipos de Gráficos, desafios maiores virão adiante. Faremos isso exatamente no Módulo 2. Convido vocês para continuar essa viagem pelo R.