Introdução ao R

Aplicações em Finanças e Economia

Marcelo S. Perlin (marcelo.perlin@ufrgs.br)

15/03/2024

Introdução ao R - Aplicações em Finanças e Economia

por **Marcelo S. Perlin**

Publicação Independente. Impresso sob demanda por Amazon.com.
Versão Online disponível em *https://www.msperlin.com/introR/*

Revisão de texto: Diversos
Capa: Rubens Lima - http://capista.com.br/
ISBN (paperback): 9798393912772
ISBN (hardcover): 9798394656910

Histórico de edições:
Primeira edição: 01/XX/202X

ÍNDICE

PREFÁCIO

Este livro surge como um *spin off* de minha outra obra "Análise de Dados Financeiros com o R". O conteúdo foi adaptado … TODO Este livro introduz o leitor ao uso do R como ferramenta de computação e análise de dados. Ao final deste livro você irá aprender como utilizar o R para importar e manipular dados e, por fim, reportar tabelas e figuras de uma pesquisa em um relatório técnico.

Sou professor da Universidade Federal do Rio Grande do Sul, onde leciono, na graduação e pós-graduação, disciplinas relacionadas ao uso do R em análise de dados. Aprendi na prático onde alunos mais erram em termos de aprendizado, e compilei um conteúdo abrangente que irá sanar a maioria das dúvidas. Seja voce um pesquisador acadêmico, ou analistas de dados, este livro é um projeto pessoal para disseminar conhecimento sobre a ferramenta para um público maior e mais diversificado.

Outra motivação que tive para escrever o livro foi minha experiência na utilização de códigos disponibilizados por outros pesquisadores em pesquisas específicas. Na maioria das vezes, esses códigos são desorganizados, pouco claros e, possivelmente, funcionam apenas no computador do pesquisador que os escreveu! Surpreendentemente, devido a desorganização, o trabalho de desvendar o código de outros professores tende a levar mais tempo do que desenvolver eu mesmo o procedimento, por mais complexo que ele fosse. Esses casos, aliás, ferem a ciência, pois um dos princípios básicos da pesquisa é a **replicabilidade** - isto é, uma rotina de pesquisa mal escrita irá reduzir a possibilidade de outras pessoas a utilizarem.

Assim como se espera que um artigo científico esteja bem escrito, também se deve esperar que o código por trás da respectiva pesquisa seja de qualidade. Porém, esse não é o caso na grande maioria das vezes. Com este livro, irei atacar esse problema, formalizando uma estrutura de código voltada à reprodutibilidade científica, focando em organização e usabilidade. Nesse sentido, espero que as futuras gerações de pesquisadores estejam mais bem preparadas para compartilhar o seu trabalho.

Antes de prosseguir, um aviso. Não iremos trabalhar usos avançados do R. O conteúdo será limitado a exemplos simples e práticos de utilização do *software* para a construção de pesquisa baseada em dados financeiros e econômicos. De fato, um dos desafios na escrita deste livro foi definir o limite entre o material introdutório e o avançado. Procurei, sempre que possível, dosar gradualmente o nível de complexidade. Para leitores interessados em conhecer funções avançadas do programa e o seu funcionamento interno, sugiro a leitura do manual oficial do R (**teetor2011r?**) e de (**wickham2019advanced?**).

Com este livro irás aprender os seguinte tópicos:

Usar o R e RStudio O capítulo 01 apresenta, discute e justifica o uso do R como uma plataforma de programação desenhada para resolver problemas relacionados a dados. No capítulo 02 exploraremos os comandos básicos R e os recursos do RStudio, incluindo criação de objetos, execução de *scripts*, interação com o disco rígido do computador, e muito mais.

Importação de dados financeiros e econômicos Nos capítulos 04 e 05 vamos aprender a importar dados de arquivos locais, tal como uma planilha do Excel, ou então da internet. Aprenderemos a tirar proveito da modularidade do R e, quando necessário, instalar pacotes que permitam o download de dados diversos tal como preços de ações, índices econômicos, curva de juros, dados financeiros de empresas e muito mais.

Limpar, estruturar e analisar dados Nos capítulos 06 e 07 iremos estudar o ecossistema de objetos do R. Aprenderemos a manipular objetos tal como vetores numéricos, datas e tabelas inteiras. Nos capítulo 08 e 09 vamos estudar o uso de ferramentas de programação para resolver problemas com dados incluindo limpeza, reestruturação e também análise.

Visualização de dados No capítulo 10 aprenderemos a usar o pacote `ggplot2` para criar visualizações do conjunto de dados, incluindo os casos mais comuns em finanças e economia, séries temporais e gráficos estatísticos.

Analisar dados com econometria No capítulo 11 aprenderemos a usar os modelos econométricos mais populares em finanças e economia, incluindo o modelo linear, GLM, Arima e outros. Isto inclui simulação, teste de hipóteses e estimação de modelos para diversas séries de dados reais.

Reporte de resultados No capítulo 12 veremos como reportar os resultados de sua pesquisa para um documento externo com a exportação fácil e reproduzível de tabelas e figuras. O conteúdo também inclui uma seção sobre a inovadora tecnologia *RMarkdown*, a qual permite que código e texto sejam compilados conjuntamente.

Melhorando o seu código No último capítulo do livro vamos discutir as melhores práticas de programação, incluindo análise do perfil de execução do código R, destacando pontos de gargalo e melhoria do tempo de execução com estratégias de cacheamento local, uso de código `C++` e processamento paralelo.

Material Suplementar

Todo o material usado no livro, incluindo exemplos de código separados por capítulos, está publicamente disponível na Internet e distribuído com um pacote R denominado `adfeR`. Este inclui arquivos de dados, código em si, e várias funções que irão facilitar a execução dos exemplos do livro. Se você planeja, como sugerido, escrever código enquanto lê o livro, este pacote ajudará muito em sua jornada.

Para instalar este pacote no seu computador, basta executar algumas linhas de comando no R. Veja o código destacado a seguir e copie e cole o mesmo no prompt do RStudio (canto inferior esquerdo da tela, com um sinal ">") e pressione Enter para cada comando. Esteja ciente de que você precisará do R e RStudio instalados em seu computador (consulte a seção @ref(instalacao)).

```
# install devtools dependency
install.packages('devtools')

# install book package
devtools::install_github('msperlin/adfeR')
```

O que este código fará é instalar o pacote `devtools`, uma dependência necessária para instalar código do Github – um repositório de pacotes onde o livro está hospedado. Depois disso, uma chamada para

`install_github('msperlin/adfeR')` irá instalar o pacote em seu computador.

Depois da instalação, todos os arquivos do livro estarão disponíveis localmente, salvos em uma pasta do seu computador. Iremos usar todos estes arquivos futuramente. Opcionalmente, caso quiser olhar os arquivos, podes copiar todo conteúdo para outra pasta com o código a seguir:

```
adfeR::copy_book_files(path_to_copy = '~')
```

Veja que o tilda (~) é um atalho para o diretório "Documentos" no Windows (ou "home" no Linux/Mac). Assim, o código anterior descompactará o arquivo do livro na pasta "Documentos/adfeR-files". O pacote também inclui várias outras funções que serão usadas ao longo do livro. Se você preferir a maneira antiga e consagrada de baixar o arquivo e descompactar manualmente, podes encontrar uma cópia no site do livro[1].

Conteúdo para Instrutores

Se você for um instrutor de R, aqui encontrarás muito material para usar em suas aulas:

Exercícios estáticos na internet Cada capítulo deste livro inclui exercícios que seus alunos podem praticar. Todas as soluções estão disponíveis na versão online do livro, disponível em https://www.msperlin.com/adfeR/.

Exercícios exportáveis para pdf ou plataformas de *e-learning* Todos exercícios do livro estão no formato `exams` (Zeileis et al. 2022) e são exportáveis para arquivos em pdf ou então para plataformas de *e-learning* tal como o *Moodle* ou *Blackboard*. Veja este post no blog[2] para maiores detalhes.

Acesso ao livro na internet Na versão web do livro, o conteúdo integral está liberado até o capítulo 7, o qual é mais que suficiente para um curso introdutório sobre a plataforma.

Espero que goste deste livro. O conteúdo tem sido compilado por um longo período de tempo, a base de muito suor e, literalmente, litros de café por parte do autor.

[1] https://www.msperlin.com/files/adfer-files/adfeR-code-and-data.zip
[2] https://www.msperlin.com/post/2021-02-18-dynamic-exercises-adfer/

Boa leitura!

Marcelo S. Perlin

AGRADECIMENTOS

Este livro não seria possível sem a devida autonomia do meu cargo de professor universitário. A liberdade de meu cargo acadêmico foi um dos facilitadores de todos meus projetos literários, os quais me dedico com muita paixão. Assim, deixo aqui o meu agradecimento a UFRGS (Universidade Federal do Rio Grande do Sul), por possibilitar e incentivar este empreendimento no mercado literário. Meus dias mais produtivos de trabalhos foram aqueles onde pude escrever meus livros.

Adicionalmente, não posso também deixar de agradecer a toda a comunidade do R. Em especial, agradeço os autores do pacote **{quarto}** (Allaire 2023), sem o qual não seria possível compilar este livro de uma forma tão fácil. Adicionalmente, abaixo destaco os respectivos pacotes disponíveis no CRAN utilizados na produção do livro e suas devidas referências. A lista foi gerada automaticamente e está em ordem alfabética.

{base} (R Core Team 2023a), **{dplyr}** (Wickham, François, et al. 2023), **{forcats}** (Wickham 2023a), **{fs}** (Hester, Wickham, e Csárdi 2023), **{ggplot2}** (Wickham, Chang, et al. 2023), **{glue}** (Hester e Bryan 2022), **{gt}** (Iannone et al. 2023), **{knitr}** (Xie 2023), **{purrr}** (Wickham e Henry 2023), **{quarto}** (Allaire 2023), **{readr}** (Wickham, Hester, e Bryan 2023), **{renv}** (Ushey e Wickham 2023), **{reticulate}** (Ushey, Allaire, e Tang 2023), **{rmarkdown}** (Allaire et al. 2023), **{stats}** (R Core Team 2023b), **{tibble}** (Müller e Wickham 2023), **{tidyr}** (Wickham, Vaughan, e Girlich 2023), **{tidyverse}** (Wickham 2023b)

CAPÍTULO 1

INTRODUÇÃO AO RSTUDIO

Após instalar os dois programas, R e RStudio, procure o ícone do RStudio na área de trabalho ou via menu *Iniciar*. Note que a instalação do R inclui um programa de interface e isso muitas vezes gera confusão. Verifique que estás utilizado o *software* correto. A janela resultante deve ser igual a figura Figura 1.1, apresentada a seguir.

Observe que o RStudio automaticamente detectou a instalação do R e inicializou a sua tela no lado esquerdo. Caso não visualizar uma tela parecida ou chegar em uma mensagem de erro indicando que o R não foi encontrado, repita os passos de instalação do capítulo anterior (seção @ref(instalacao)).

> **Alternativa ao RStudio**
>
> Este capítulo inteiro é dedicado ao uso do RStudio, o qual entendo ser um dos melhores ambientes de desenvolvimento para o R. Alternativamente, um grande concorrente é o vscode, o qual oferece uma plataforma agnóstica a linguagem de programação, funcionando para R, Python ou qualquer outra lingaguem. Em projeto que envolve múltiplas linguagens, o vscode é ótimo e fácil de usar.

Como um primeiro exercício, clique em *File*, *New File* e *R Script*. Após, um editor de texto deve aparecer no lado esquerdo da tela do RStudio. É

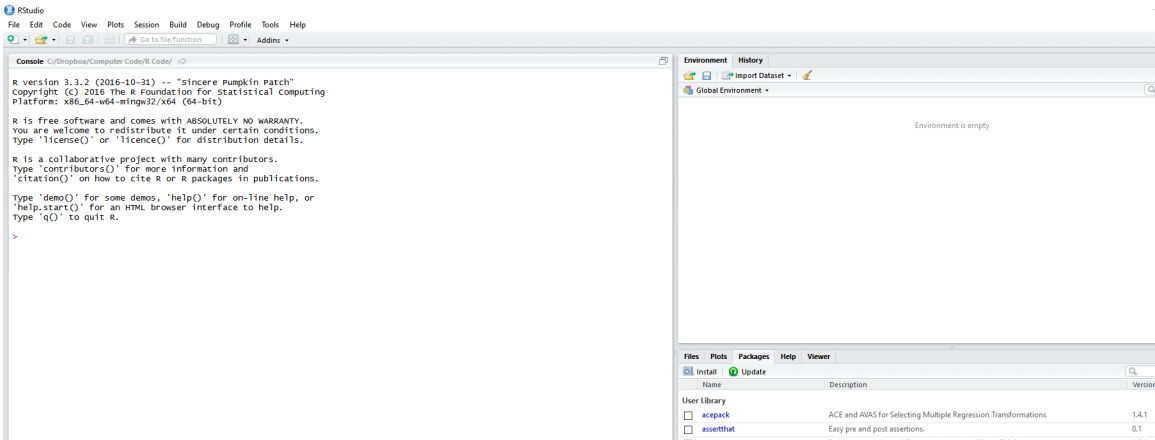

Figura 1.1: A tela do RStudio

nesse editor que iremos inserir os nossos comandos, os quais são executados de cima para baixo, na mesma direção em que normalmente o lemos. Note que essa direção de execução introduz uma dinâmica de recursividade: cada comando depende do comando executado nas linhas anteriores. Após realizar os passos definidos anteriormente, a tela resultante deve ser semelhante à apresentada na figura Figura 1.2.

> **Importante**
>
> Uma sugestão importante aqui é modificar o esquema de cores do RStudio para uma configuração de **tela escura**. Não é somente uma questão estética mas sim de prevenção e melhoria de sua saúde física. Possivelmente irás passar demasiado tempo na frente do computador. Assim, vale a pena modificar as cores da interface para aliviar seus olhos do constante brilho da tela. Dessa forma, conseguirás trabalhar por mais tempo, sem forçar a sua visão. Podes configurar o esquema de cores do RStudio indo na opção *Tools*, *Global Options* e então em *Appearance*. Um esquema de cores escuras que pessoalmente gosto e sugiro é o *Ambience*.

Após os passos anteriores, a tela do RStudio deve estar semelhante a Fi-

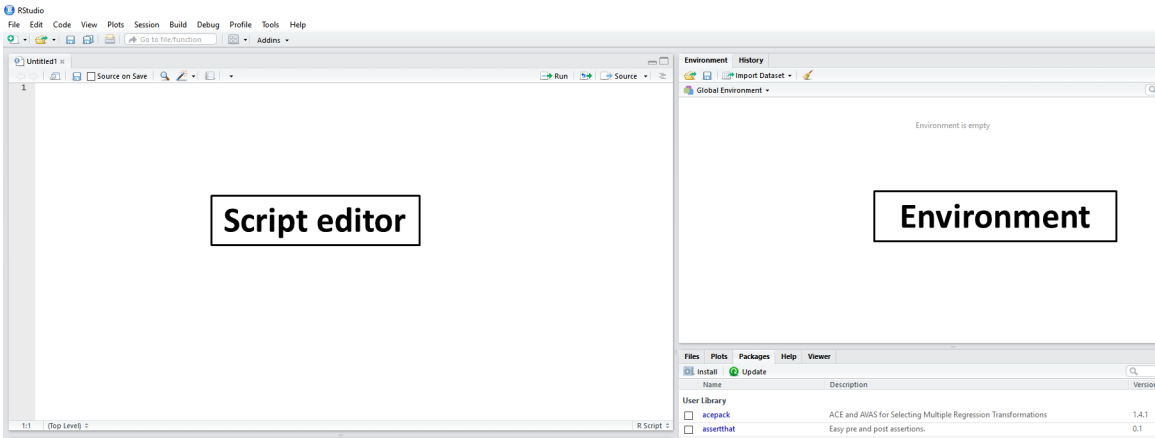

Figura 1.2: Explicando a tela do RStudio

gura Figura 1.2, com os seguintes itens/painéis:

Editor de scripts (*Script editor*): localizado no lado esquerdo e acima da tela. Esse painel é utilizado para escrever código e é onde passaremos a maior parte do tempo.

Console do R (*R prompt*): localizado no lado esquerdo e abaixo do editor de *scripts*. Apresenta o *prompt* do R, o qual também pode ser utilizado para executar comandos. A principal função do *prompt* é testar código e apresentar os resultados dos comandos inseridos no editor de *scripts*.

Área de trabalho (*Environment*): localizado no lado direito e superior da tela. Mostra todos os objetos, incluindo variáveis e funções atualmente disponíveis para o usuário. Observe também a presença do painel *History*, o qual mostra o histórico dos comandos já executados.

Pacotes (*Panel Packages*): mostra os pacotes instalados e carregados pelo R. Um pacote é nada mais que um módulo no R, cada qual com sua finalidade específica. Observe a presença de quatro abas: *Files*, para carregar e visualizar arquivos do sistema; *Plots*, para visualizar figuras; *Help*, para acessar o sistema de ajuda do R e *Viewer*, para mostrar resultados dinâmicos e interativos, tal como uma página da internet.

Como um exercício introdutório, vamos inicializar duas variáveis. Dentro do console do R (lado esquerdo inferior), digite os seguintes comandos e aperte *enter* ao final de cada linha. O símbolo <- é nada mais que a junção de < com -. O símbolo ' representa uma aspa simples e sua localização no teclado Brasileiro é no botão abaixo do *escape* (*esc*), lado esquerdo superior do teclado.

```
# set x and y
x <- 1
y <- 'my text'
```

Após a execução, dois objetos devem aparecer no painel *Environment*, um chamado `x` com o valor 1, e outro chamado `y` com o conjunto de caracteres `'my text'`. O histórico de comandos na aba *History* também foi atualizado com os comandos utilizados anteriormente.

Agora, vamos mostrar na tela os valores de `x`. Para isso, digite o seguinte comando no *prompt* e aperte *enter* novamente:

```
# print x
print(x)
```

```
R> [1] 1
```

A função **print()** é uma das principais funções para mostrarmos valores no *prompt* do R. O texto apresentado como `[1]` indica o índice do primeiro número da linha. Para verificar isso, digite o seguinte comando, o qual irá mostrar vários números na tela:

```
# print vector from 50 to 100
print(50:100)
```

```
R>  [1]  50  51  52  53  54  55  56  57  58  59  60  61  62  63
R> [15]  64  65  66  67  68  69  70  71  72  73  74  75  76  77
R> [29]  78  79  80  81  82  83  84  85  86  87  88  89  90  91
R> [43]  92  93  94  95  96  97  98  99 100
```

Nesse caso, utilizamos o símbolo : em `50:100` para criar uma sequência iniciando em 50 e terminando em 100. Observe que temos valores encapsulados por colchetes ([]) no lado esquerda da tela. Esses representam os índices do primeiro elemento apresentado na linha. Por exemplo, o décimo quinto elemento do vetor criado é o valor 64.

1.1 Executando Códigos em um *Script*

Agora, vamos juntar todos os códigos digitados anteriormente e colar na tela do editor (lado esquerdo superior), assim como mostrado a seguir:

```
# set objects
x <- 1
y <- 'my text'

# print it
print(x)
print(1:50)
```

Após colar todos os comandos no editor, salve o arquivo *.R* em alguma pasta pessoal. Esse arquivo, o qual no momento não faz nada de especial, registrou os passos de um algoritmo simples que cria dois objetos e mostra os seus valores. Futuramente esse irá ter mais forma, com a importação de dados, manipulação e modelagem dos mesmos e saída de tabelas e figuras.

No RStudio existem alguns atalhos predefinidos para executar códigos que economizam bastante tempo. Para executar um *script* inteiro, basta apertar `control + shift + s`. Esse é o comando *source*. Com o RStudio aberto, sugiro testar essa combinação de teclas e verificar como o código digitado anteriormente é executado, mostrando os valores no *prompt* do R. Visualmente, o resultado deve ser próximo ao apresentado na figura Figura 1.3.

Outro comando muito útil é a execução por linha. Nesse caso não é executado todo o arquivo, mas somente a linha em que o cursor do *mouse* se encontra. Para isto, basta apertar `control+enter`. Esse atalho é bastante útil no desenvolvimento de rotinas pois permite que cada linha seja testada antes de executar o programa inteiro. Como um exemplo de uso, aponte o cursor para a linha `print(x)` e pressione `control + enter`. Verás que o valor de `x` é mostrado na tela do *prompt*. A seguir destaco esses e outros atalhos do RStudio, os quais também são muito úteis.

- **control+shift+s** executa o arquivo atual do RStudio, sem mostrar comandos no *prompt* (sem eco – somente saída);
- **control+shift+enter**: executa o arquivo atual, mostrando comandos na tela (com eco – código e saída);
- **control+enter**: executa a linha selecionada, mostrando comandos na tela;

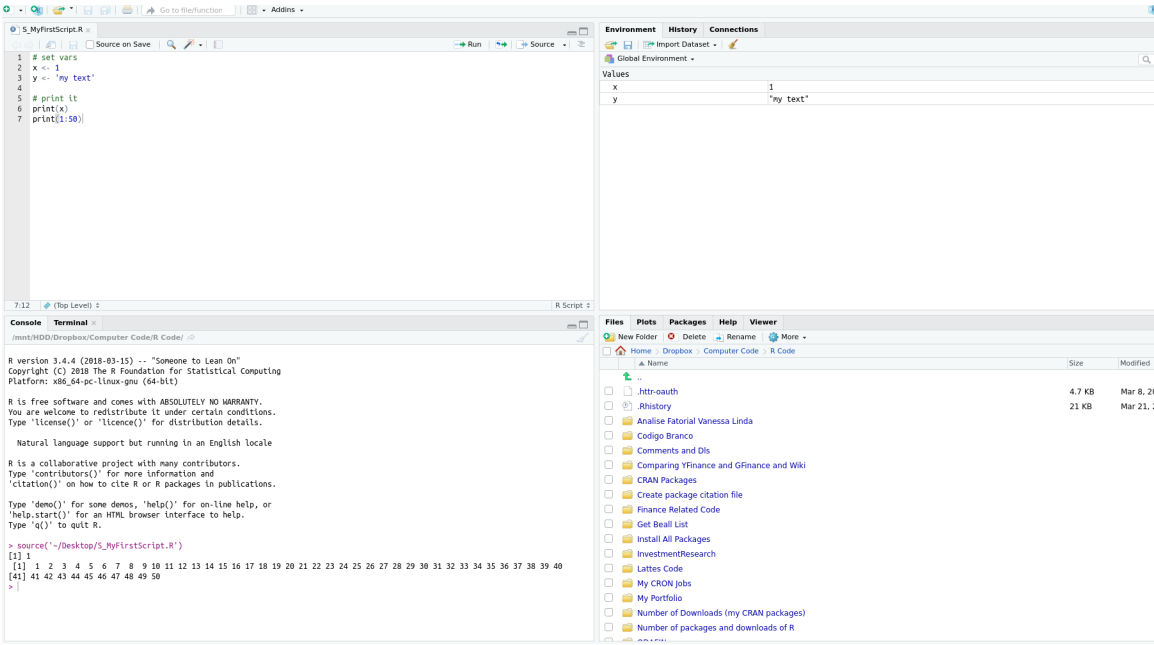

Figura 1.3: Exemplo de Rotina no R

- **control+shift+b**: executa os códigos do início do arquivo até a linha atual onde o cursor se encontra;
- **control+shift+e**: executa os códigos da linha onde o cursor se encontra até o final do arquivo.

Sugere-se que esses atalhos sejam memorizados e utilizados. Isso facilita bastante o uso do programa. Para aqueles que gostam de utilizar o *mouse*, uma maneira alternativa para rodar o código do *script* é apertar o botão *source*, localizado no canto direito superior do editor de rotinas. Isto é equivalente ao atalho `control+shift+s`.

Porém, no mundo real de programação, poucos são os casos em que uma análise de dados é realizada por um *script* apenas. Como uma forma de organizar o código, pode-se dividir o trabalho em N *scripts* diferentes, onde um deles é o "mestre", responsável por rodar os demais.

Neste caso, para executar os *scripts* em sequência, basta chamá-los no *script* mestre com o comando **source()** , como no código a seguir:

```
# Import all data
source('01-import-data.R')

```

```
# Clean up
source('02-clean-data.R')

# Build tables
source('03-build-table.R')
```

Nesse caso, o código anterior é equivalente a abrirmos e executarmos (*control + shift + s*) cada um dos *scripts* sequencialmente.

Como podemos ver, existem diversas maneiras de executar uma rotina de pesquisa. Na prática, porém, iras centralizar o uso em dois comandos apenas: `control+shift+s` para rodar o *script* inteiro e `control+enter` para rodar por linha.

1.2 Tipos de Arquivos

Assim como outros programas, o R e o RStudio possuem um ecossistema de arquivos e cada extensão tem uma finalidade diferente. A seguir apresenta-se uma descrição de diversas extensões de arquivos exclusivos ao R e RStudio. Os itens da lista estão ordenados por ordem de importância e uso.

Arquivos com extensão *.R*: Representam arquivos do tipo texto contendo diversas instruções para o R. Esses são os arquivos que conterão o código da pesquisa e onde passaremos a maior parte do tempo. Também pode ser chamado de um *script* ou rotina de pesquisa. Como sugestão, pode-se dividir toda uma pesquisa em etapas e arquivos numerados. Exemplos: *01-Get-Data.R*, *02-Clean-data.R*, *03-Estimate-Models.R*.

Arquivos com extensão *.RData* e *.rds*: armazenam dados nativos do R. Esses arquivos servem para salvar (ou congelar) objetos do R em um arquivo no disco rígido do computador para, em sessão futura, serem novamente carregados. Por exemplo, podes guardar o resultado de uma pesquisa em uma tabela, a qual é salva em um arquivo com extensão *.RData* ou *.rds*. Exemplos: *Raw-Data.RData*, *Table-Results.rds*.

Arquivos com extensão *.Rmd*, *.Rnw* e *.qmd*: São arquivos relacionados a tecnologia *Rmarkdown* e *Quarto*. O uso desses arquivos permite a criação de documentos onde texto e código são integrados.

Arquivos com extensão *.Rproj*: Contém informações para a edição de projetos no RStudio. O sistema de projetos do RStudio permite a configuração customizada do projeto e também facilita a utilização de ferramentas de controle de código, tal como controle de versões. O seu uso, porém, não é essencial. Para aqueles com interesse em conhecer esta funcionalidade, sugiro a leitura do manual do RStudio. Uma maneira simples de entender os tipos de projetos disponíveis é, no RStudio, clicar em "File", "New project", "New Folder" e assim deve aparecer uma tela com todos os tipos possíveis de projetos no RStudio. Exemplo: *My-Dissertation-Project.Rproj*.

1.3 Testando Código

O desenvolvimento de códigos em R segue um conjunto de etapas. Primeiro você escreverá uma nova linha de comando em uma rotina. Essa linha será testada com o atalho `control + enter`, verificando-se a ocorrência de erros e as saídas na tela. Caso não houver erro e o resultado for igual ao esperado, parte-se para a próxima linha de código.

Um ciclo de trabalho fica claro, a escrita do código da linha atual é seguida pela execução, seguido da verificação de resultados, modificação caso necessário e assim por diante. Esse é um processo normal e esperado. Dado que uma rotina é lida e executada de cima para baixo, você precisa ter certeza de que cada linha de código está corretamente especificada antes de passar para a próxima.

Quando você está tentando encontrar um erro em um *script* preexistente, o R oferece algumas ferramentas para controlar e avaliar sua execução. Isso é especialmente útil quando você possui um código longo e complicado. A ferramenta de teste mais simples e fácil de utilizar que o RStudio oferece é o ponto de interrupção do código. No RStudio, você pode clicar no lado esquerdo do editor e aparecerá um círculo vermelho, como na Figura Figura 1.4.

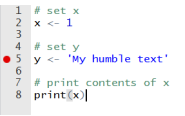

Figura 1.4: Exemplo de debug

O círculo vermelho indica um ponto de interrupção do código que forçará o R a pausar a execução nessa linha. Quando a execução atinge o ponto de

interrupção, o *prompt* mudará para `browser[1]>` e você poderá verificar o conteúdo dos objetos. No console, você tem a opção de continuar a execução para o próximo ponto de interrupção ou interrompê-la. O mesmo resultado pode ser alcançado usando a função **browser()** . Dê uma olhada:

```
# set x
x <- 1

# set y
browser()
y <- 'My humble text'

# print contents of x
print(x)
```

O resultado prático do código anterior é o mesmo que utilizar o círculo vermelho do RStudio, figura Figura 1.4. Porém, o uso do **browser()** permite mais controle sobre onde a execução deve ser pausada. Como um teste, copie e cole o código anterior no RStudio, salve em um novo *script* e execute com *Control + Shift + S*. Para sair do ambiente de depuramento (*debug*), aperte *enter* no *prompt* do RStudio.

1.4 Cancelando a Execução de um Código

Toda vez que o R estiver executando algum código, uma sinalização visual no formato de um pequeno círculo vermelho no canto direito do *prompt* irá aparecer. Caso conseguir ler (o símbolo é pequeno em monitores modernos), o texto indica o termo *stop*. Esse símbolo não somente indica que o programa ainda está rodando mas também pode ser utilizado para cancelar a execução de um código. Para isso, basta clicar no referido botão. Outra maneira de cancelar uma execução é apontar o mouse no *prompt* e pressionar a tecla *Esc* no teclado.

Para testar o cancelamento de código, copie e cole o código a seguir em um *script* do RStudio. Após salvar, rode o mesmo com `control+shift+s`.

```
for (i in 1:100) {
  message('\nRunning code (please make it stop by hitting esc!)')
  Sys.sleep(1)
}
```

O código anterior usa um comando especial do tipo `for` para mostrar a mensagem a cada segundo. Neste caso, o código demorará 100 segundos para rodar. Caso não desejes esperar, aperte `esc` para cancelar a execução. Por enquanto, não se preocupe com as funções utilizadas no exemplo. Iremos discutir o uso do comando `for` no capítulo **?@sec-programacao**.

1.5 Procurando Ajuda

Uma tarefa muito comum no uso do R é procurar ajuda. A quantidade de funções disponíveis para o R é gigantesca e memorizar todas peculariedades é quase impossível. Assim, até mesmo usuários avançados comumente procuram ajuda sobre tarefas específicas no programa, seja para entender detalhes sobre algumas funções ou estudar um novo procedimento.

É possível buscar ajuda utilizando tanto o painel de *help* do RStudio como diretamente do *prompt*. Para isso, basta digitar o ponto de interrogação junto ao objeto sobre o qual se deseja ajuda, tal como em `?mean`. Nesse caso, o objeto `mean` é uma função e o uso do comando irá abrir o painel de ajuda sobre ela.

No R, toda tela de ajuda de uma função é igual, conforme se vê na Figura 1.5 apresentada a seguir. Esta mostra uma descrição da função `mean`, seus argumentos de entrada explicados e também o seu objeto de saída. A tela de ajuda segue com referências e sugestões para outras funções relacionadas. Mais importante, os **exemplos de uso da função** aparecem por último e podem ser copiados e colados para acelerar o aprendizado no uso da função.

Caso quiséssemos procurar um termo nos arquivos de ajuda, bastaria utilizar o comando `??"standard deviation"`. Essa operação irá procurar a ocorrência do termo em todos os pacotes do R e é muito útil para aprender como realizar alguma operação, nesse caso o cálculo de desvio padrão.

Como sugestão, o ponto inicial e mais direto para aprender uma nova função é observando o seu exemplo de uso, localizada no final da página de ajuda. Com isto, podes verificar quais tipos de objetos de entrada a mesma aceita e qual o formato e o tipo de objeto na sua saída. Após isso, leia atentamente a tela de ajuda para entender se a mesma faz exatamente o que esperas e quais são as suas opções de uso nas respectivas entradas. Caso a função realizar o procedimento desejado, podes copiar e colar o exemplo de uso para o teu próprio *script*, ajustando onde for necessário.

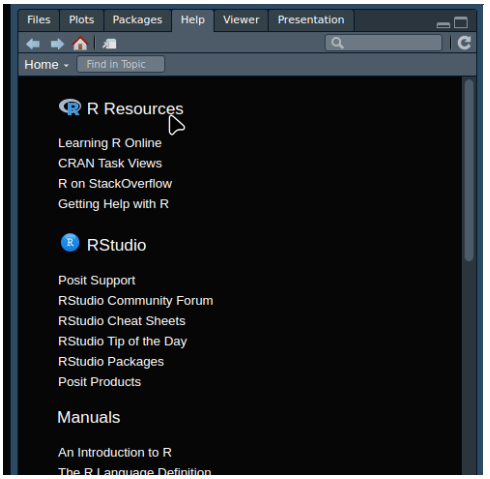

Figura 1.5: Usando o sistema de ajuda do RStudio

Outra fonte muito importante de ajuda é a própria internet. Sites como *stackoverflow.com* e *mailing lists* específicos do R, cujo conteúdo também está na internet, são fontes preciosas de informação. Havendo alguma dúvida que não foi possível solucionar via leitura dos arquivos de ajuda do R, vale o esforço de procurar uma solução via mecanismo de busca na internet. Em muitas situações, o seu problema, por mais específico que seja, já ocorreu e já foi solucionado por outros usuários.

Caso estiver recebendo uma mensagem de erro enigmática, outra dica é copiar e colar a mesma para uma pesquisa no Google. Aqui apresenta-se outro benefício do uso da língua inglesa. É mais provável que encontres a solução se o erro for escrito em inglês, dado o maior número de usuários na comunidade global. Caso não encontrar uma solução desta forma, podes inserir uma pergunta no stackoverflow ou no grupo Brasileiro do R no Facebook.

> **Cuidado**
>
> Toda vez que for pedir ajuda na internet, procure sempre 1) descrever claramente o seu problema e 2) adicionar um código reproduzível do seu problema. Assim, o leitor pode facilmente verificar o que está acontecendo ao rodar o exemplo no seu computador. Não tenho dúvida que, se respeitar ambas regras, logo uma pessoa caridosa lhe ajudará com o seu problema.

1.6 Utilizando *Code Completion* com a Tecla *tab*

Um dos recursos mais úteis do RStudio é o preenchimento automático de código (*code completion*). Essa é uma ferramenta de edição que facilita o encontro de nomes de objetos, nome de pacotes, nome de arquivos e nomes de entradas em funções. O seu uso é muito simples. Após digitar um texto qualquer, basta apertar a tecla *tab* e uma série de opções aparecerá. Veja a Figura @ref(fig:autocomplete) apresentada a seguir, em que, após digitar a letra *f* e apertar *tab*, aparece uma janela com uma lista de objetos que iniciam com a respectiva letra. Figura Figura 1.6

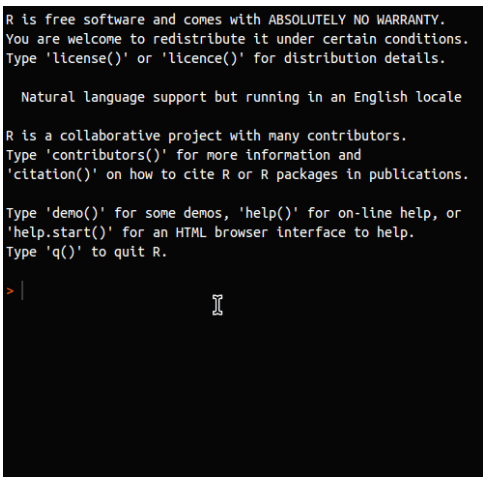

Figura 1.6: Usando o sistema de ajuda do RStudio

Essa ferramenta também funciona para pacotes. Para verificar, digite `library(r)` no *prompt* ou no editor, coloque o cursor entre os parênteses e aperte *tab*. O resultado deve ser algo parecido com a Figura 1.7.

Observe que uma descrição do pacote ou objeto também é oferecida. Isso facilita bastante o dia a dia, pois a memorização das funcionalidades e dos nomes dos pacotes e os objetos do R não é uma tarefa fácil. O uso do *tab* diminui o tempo de investigação dos nomes e evita possíveis erros de digitação na definição destes.

O uso dessa ferramenta torna-se ainda mais benéfico quando os objetos são nomeados com algum tipo de padrão. No restante do livro observarás que os objetos tendem a ser nomeados com o prefixo *my*, como em `my_x`, `my_num`, `my_char`. O uso desse padrão facilita o encontro futuro do nome dos objetos, pois basta digitar *my*, apertar *tab* e uma lista de todos os objetos criados pelo usuário aparecerá.

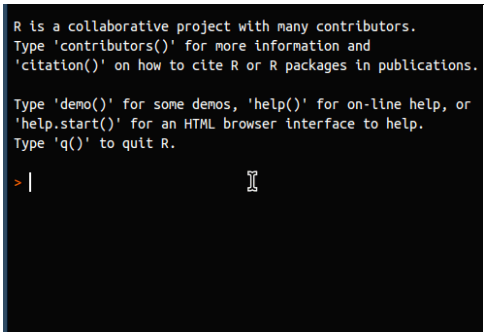

Figura 1.7: Usando o autocomplete para procurar pacotes

Outro uso do *tab* é no encontro de arquivos e pastas no computador. Basta criar uma variável como `my_file <- " "`, apontar o cursor para o meio das aspas e apertar a tecla *tab*. Uma tela com os arquivos e pastas do diretório atual de trabalho aparecerá, conforme mostrado na Figura 1.8. Nesse caso específico, o R estava direcionado para a minha pasta de códigos, em que é possível enxergar diversos trabalhos realizados no passado.

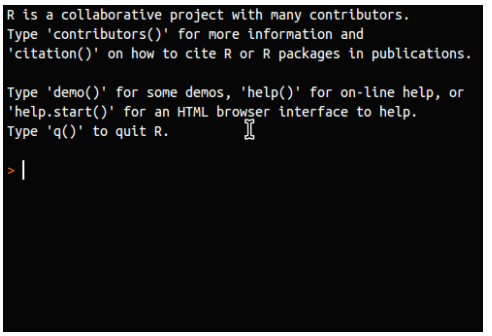

Figura 1.8: Usando o autocomplete para navegar diretórios

Uma dica aqui é utilizar o *tab* com a raiz do computador. Assumindo que o disco do seu computador está alocado para `C:/`, digite `my_file <- "C:/"` e pressione *tab* após o símbolo `/`. Uma tela com os arquivos da raiz do

computador aparecerá no RStudio. Podes facilmente navegar o sistema de arquivos utilizando as setas e *enter*.

O *autocomplete* também funciona para encontrar e definir as entradas de uma função. Veja um exemplo na Figura 1.9.

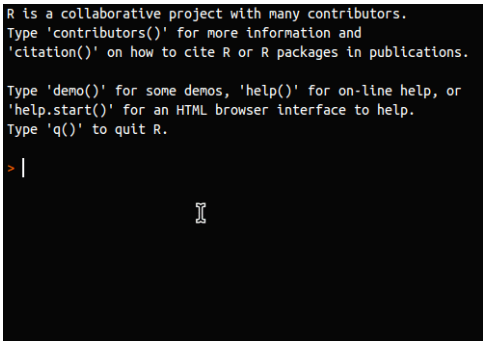

Figura 1.9: Usando o autocomplete para navegar argumentos de funções

> **Importante**
>
> O *autocomplete* é uma das ferramentas mais importantes do RStudio, funcionando para encontro de objetos, locais no disco rígido, pacotes e funções. Acostume-se a utilizar a tecla *tab* o quanto antes e logo verá como fica mais fácil escrever código rapidamente, e sem erros de digitação.

REFERÊNCIAS BIBLIOGRÁFICAS

Allaire, JJ. 2023. *quarto: R Interface to Quarto Markdown Publishing System*. https://github.com/quarto-dev/quarto-r.

Allaire, JJ, Yihui Xie, Christophe Dervieux, Jonathan McPherson, Javier Luraschi, Kevin Ushey, Aron Atkins, et al. 2023. *rmarkdown: Dynamic Documents for R*. https://github.com/rstudio/rmarkdown.

Hester, Jim, e Jennifer Bryan. 2022. *glue: Interpreted String Literals*. https://github.com/tidyverse/glue.

Hester, Jim, Hadley Wickham, e Gábor Csárdi. 2023. *fs: Cross-Platform File System Operations Based on libuv*. https://fs.r-lib.org.

Iannone, Richard, Joe Cheng, Barret Schloerke, Ellis Hughes, Alexandra Lauer, e JooYoung Seo. 2023. *gt: Easily Create Presentation-Ready Display Tables*. https://gt.rstudio.com.

Müller, Kirill, e Hadley Wickham. 2023. *tibble: Simple Data Frames*. https://tibble.tidyverse.org/.

R Core Team. 2023b. *R: A Language and Environment for Statistical Computing*. Vienna, Austria: R Foundation for Statistical Computing. https://www.R-project.org/.

———. 2023a. *R: A Language and Environment for Statistical Computing*. Vienna, Austria: R Foundation for Statistical Computing. https://www.R-project.org/.

Ushey, Kevin, JJ Allaire, e Yuan Tang. 2023. *reticulate: Interface to Python*. https://rstudio.github.io/reticulate/.

Ushey, Kevin, e Hadley Wickham. 2023. *renv: Project Environments*. https://rstudio.github.io/renv/.

Wickham, Hadley. 2023a. *forcats: Tools for Working with Categorical Variables (Factors)*. https://forcats.tidyverse.org/.

———. 2023b. *tidyverse: Easily Install and Load the Tidyverse*. https://tidyverse.tidyverse.org.

Wickham, Hadley, Winston Chang, Lionel Henry, Thomas Lin Pedersen, Kohske Takahashi, Claus Wilke, Kara Woo, Hiroaki Yutani, e Dewey Dunnington. 2023. *ggplot2: Create Elegant Data Visualisations Using the Grammar of Graphics*. https://ggplot2.tidyverse.org.

Wickham, Hadley, Romain François, Lionel Henry, Kirill Müller, e Davis Vaughan. 2023. *dplyr: A Grammar of Data Manipulation*. https://dplyr.tidyverse.org.

Wickham, Hadley, e Lionel Henry. 2023. *purrr: Functional Programming Tools*. https://purrr.tidyverse.org/.

Wickham, Hadley, Jim Hester, e Jennifer Bryan. 2023. *readr: Read Rectangular Text Data*. https://readr.tidyverse.org.

Wickham, Hadley, Davis Vaughan, e Maximilian Girlich. 2023. *tidyr: Tidy Messy Data*. https://tidyr.tidyverse.org.

Xie, Yihui. 2023. *knitr: A General-Purpose Package for Dynamic Report Generation in R*. https://yihui.org/knitr/.

Zeileis, Achim, Bettina Gruen, Friedrich Leisch, e Nikolaus Umlauf. 2022. *exams: Automatic Generation of Exams in R*. https://www.R-exams.org/.

ÍNDICE