

Ciência de Dados Aplicada à Saúde Materno-Infantil

Observatório Obstétrico Brasileiro

22/10/2022

Table of contents

Prefácio	4
Sumário	5
1 Introdução	6
1.1 Base de dados	6
Tutorial de R	8
Sobre o software R	8
2 Instalação R	9
2.0.1 Para Windows	9
2.0.2 Para MAC	16
2.0.3 Para Linux	19
3 Instalação RStudio	28
3.0.1 Para Windows	28
3.0.2 Para MAC	33
3.0.3 Para Linux	33
4 Primeiros passos no RStudio	38
4.0.1 Projetos	39
4.0.2 Boas práticas	42
5 Primeiros passos no R	44
5.1 R como calculadora	44
5.2 Atribuição	45
5.3 Objetos em R	46
5.3.1 Apagar objetos	47
5.4 Vetores	47
5.5 Matrizes	49
5.6 Fatores	52
5.7 Data frame	52
5.8 Operadores lógicos	55
5.9 Dados faltantes, infinitos e indefinições matemáticas	57
5.10 Condicionamento: If e else	58

5.11	Iterador for	59
5.12	Funções	60
5.13	Como obter ajuda no R	63
5.14	Pacotes	63
5.14.1	Instalação de pacotes	64
5.14.2	Carregar pacotes	65
5.15	Materiais complementares	65
Referências		67

Prefácio

Sumário

1 Introdução

1.1 Base de dados

Os dados que consideramos nas aplicações deste livro são baseados em gestantes e puérperas de 10 a 55 anos hospitalizadas com Síndrome Respiratória Aguda Grave (SRAG) por COVID-19, confirmada por teste de PCR, no período de março de 2020 a dezembro de 2021, disponíveis no Sistema de Informação da Vigilância Epidemiológica da Gripe (SIVEP-Gripe), sistema oficial para o registro dos casos e óbitos por SRAG disponibilizado pelo Ministério da Saúde.

Na base `dados_covid` estão contidas 11.485 observações e 50 variáveis. Assim, são as características observadas:

Variável	Descrição
<code>sem_pri</code>	Semana epidemiológica dos primeiros sintomas.
<code>idade_anos</code>	Idade, em anos, da gestante ou puérpera.
<code>sg_uf</code>	Sigla da Unidade Federativa de residência da gestante ou puérpera.
<code>id_mn_resi</code>	Município de residência da gestante ou puérpera.
<code>co_mun_res</code>	Código do município de residência da gestante ou puérpera.
<code>co_mu_inte</code>	Código do município onde está localizado a Unidade de Saúde onde a gestante ou puérpera internou.
<code>dt_sin_pri</code>	Data de primeiros sintomas do caso.
<code>dt_evoluca_2</code>	Data da alta ou do óbito da gestante ou puérpera.
<code>ano</code>	Ano da infecção pelo COVID-19.
<code>classi_gesta_puerp</code>	Tempo de gestacional da gestante e puerpério.
<code>raca</code>	Raça da gestante ou puérpera.
<code>escol</code>	Nível de escolaridade da gestante ou puérpera.
<code>mudou_muni</code>	Se gestante ou puérpera precisou se deslocar para outro município para realizar atendimento.
<code>zona</code>	Tipo de zona de residência da gestante ou puérpera.
<code>faixa_et</code>	Faixa etária da gestante ou puérpera.
<code>hospital</code>	Se gestante ou puérpera foi hospitalizada.
<code>hist_viagem</code>	Se gestante ou puérpera fez viagem internacional até 14 dias antes do início dos sintomas.
<code>sg_para_srag</code>	Se o caso é proveniente de síndrome gripal (SG) que evoluiu para síndrome respiratória aguda grave (SRAG).

Variável	Descrição
inf_inter	Se trata-se de caso nosocomial (infecção adquirida no hospital).
cont_ave_suino	Se a gestante ou puérpera trabalha ou tem contato direto com aves, suínos ou outros animais.
vacina	Se a gestante ou puérpera recebeu vacina contra <i>influenza</i> .
vacina_cov	Se a gestante ou puérpera recebeu vacina contra COVID-19.
antiviral	Se gestante ou puérpera usou antiviral para gripe e qual antiviral.
febre	Se gestante ou puérpera manifestou sintoma de febre.
tosse	Se gestante ou puérpera manifestou sintoma de tosse.
garganta	Se gestante ou puérpera manifestou sintoma de dor de garganta.
dispneia	Se gestante ou puérpera manifestou sintoma de dispneia.
desc_resp	Se gestante ou puérpera manifestou sintoma de desconforto respiratório.
saturacao	Se gestante ou puérpera manifestou sintoma de saturação.
diarreia	Se gestante ou puérpera manifestou sintoma de diarreia.
vomito	Se gestante ou puérpera manifestou sintoma de vômito.
dor_abd	Se gestante ou puérpera manifestou sintoma de dor abdominal.
fadiga	Se gestante ou puérpera manifestou sintoma de fadiga.
perd_olft	Se gestante ou puérpera manifestou sintoma de perda de olfato.
perd_pala	Se gestante ou puérpera manifestou sintoma de perda de paladar.
cardiopati	Se gestante ou puérpera tem doença cardiovascular crônica.
hematologi	Se gestante ou puérpera tem doença hematológica crônica.
hepatica	Se gestante ou puérpera tem doença hepática crônica.
asma	Se gestante ou puérpera tem asma.
diabetes	Se gestante ou puérpera tem diabetes <i>mellitus</i> .
neuro	Se gestante ou puérpera tem doença neurológica.
pneumopati	Se gestante ou puérpera tem outra pneumopatia crônica.
imunodepre	Se gestante ou puérpera tem imunodeficiência ou imunodepressão (diminuição da função do sistema imunológico).
renal	Se gestante ou puérpera tem doença renal crônica.
obesidade	Se gestante ou puérpera tem obesidade.
uti	Se gestante ou puérpera foi internada na UTI.
suport_ven	Se gestante ou puérpera precisou de ventilação mecânica; se sim, se foi invasiva ou não.
evolucao	Evolução do caso da gestante ou puérpera.
variante	Variante do vírus SARS-CoV-2 (vírus do COVID-19).

Tutorial de R

Sobre o software R

R é um ambiente computacional e uma linguagem de programação para manipulação, análise e visualização de dados. É considerado um dos melhores ambiente computacional para essa finalidade. O R é mantido pela [R Development Core Team](#) e está disponível para diferentes sistemas operacionais: Linux, Mac e Windows.

O software é livre, ou seja, gratuito, com código aberto em uma linguagem acessível. Nele, estão implementadas muitas metodologias estatísticas. Muitas dessas fazem parte do ambiente base de R e outras acompanham o ambiente sob a forma de pacotes, o que torna o R altamente expansível. Os pacotes são bibliotecas com dados e funções para diferentes áreas do conhecimento relacionados à estatística e áreas afins, devidamente documentados.

O R possui uma comunidade extremamente ativa, engajada desde o aprimoramento de ferramentas e desenvolvimento de novas bibliotecas, até o suporte aos usuários. Sobre o desenvolvimento de novas bibliotecas, um pesquisador em Estatística que desenvolve um novo modelo estatístico pode disponibilizá-lo em um pacote acessível a que se interessam pelo modelo.

Além disso, a disponibilidade e compartilhamento da pesquisa em um pacote no R é uma boa prática quando falamos de reprodutibilidade na Ciência. Ainda nesse ponto, realizar as análises de uma pesquisa aplicada em um programa livre e acessível a todos é um dos principais pontos para permitir reprodutibilidade.

Ao optar por programar em R também implica na escolha de uma IDE (Integrated Development Environment) que, na grande maioria dos casos, será o [RStudio](#). O RStudio é um conjunto de ferramentas integradas projetadas para editar e executar os códigos em R. Assim, quando for o interesse utilizar o R, só precisa abrir o RStudio (R é automaticamente carregado).

Para instalação do R e do RStudio, veja a Seção que segue.

2 Instalação R

Nessa Seção, vamos apresentar como instalar o R e o RStudio para os três sistemas operacionais: Windows, MAC e Linux, respectivamente.

2.0.1 Para Windows

Os passos para instalar o R quando o sistema operacional é Windows são os seguintes:

- 1) Entre neste [link](#) para acessar a página do R e clique em Download, como no link destacado em retângulo vermelho na Figura @ref(fig:windows1). Note que o 3.6.1 é o número da versão mais recente disponível no momento da construção desse material (5/7/19).

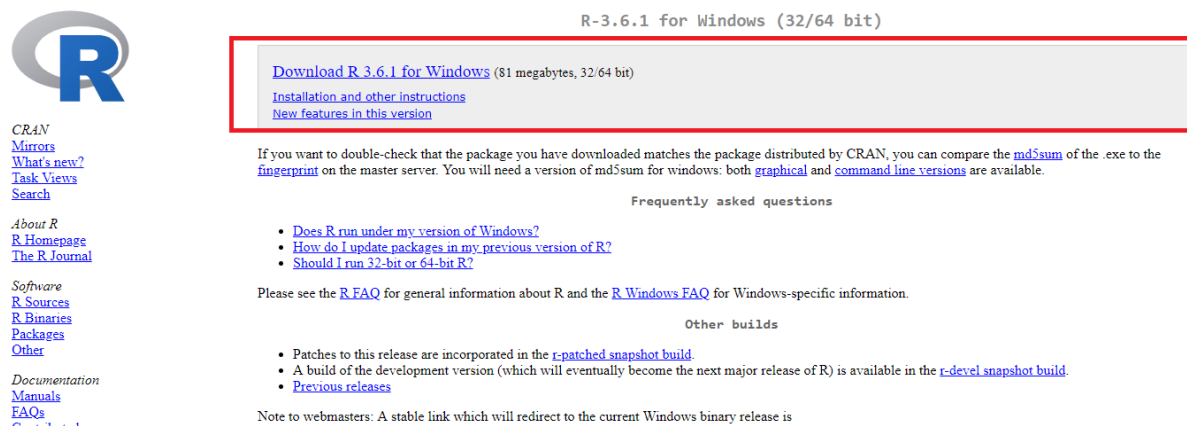


Figure 2.1: Download R para Windows

- 2) Salve o arquivo de instalação em algum caminho de interesse do seu computador. Por exemplo, na Figura @ref(fig:windows2) mostra que a pasta é “Downloads”.
- 3) Clique duas vezes com o botão esquerdo no instalador para iniciar a instalação. O próximo passo é escolher a língua para instalação. Na Figura @ref(fig:windows3) abaixo é português.
- 4) Clique em “Próximo” nas próximas janelas, como nas Figuras @ref(fig:windows4) a @ref(fig:windows9).

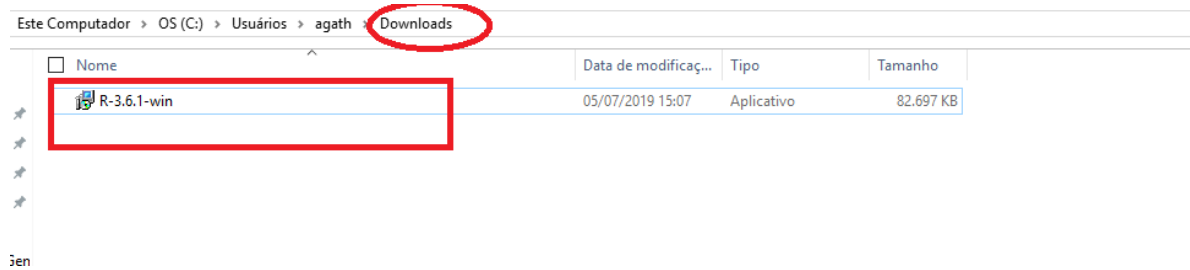


Figure 2.2: Instalador

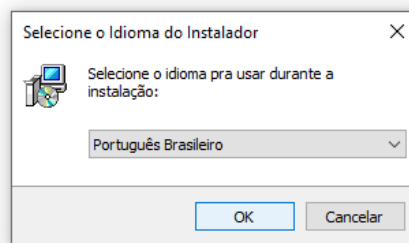
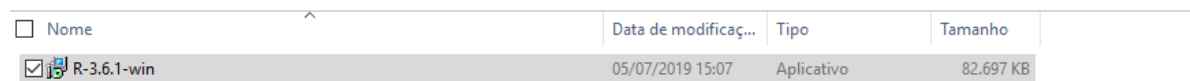


Figure 2.3: Escolha da lingua para instalação

<input type="checkbox"/> Nome	Data de modificaç...	Tipo	Tamanho
<input checked="" type="checkbox"/> R-3.6.1-win	05/07/2019 15:07	Aplicativo	82.697 KB

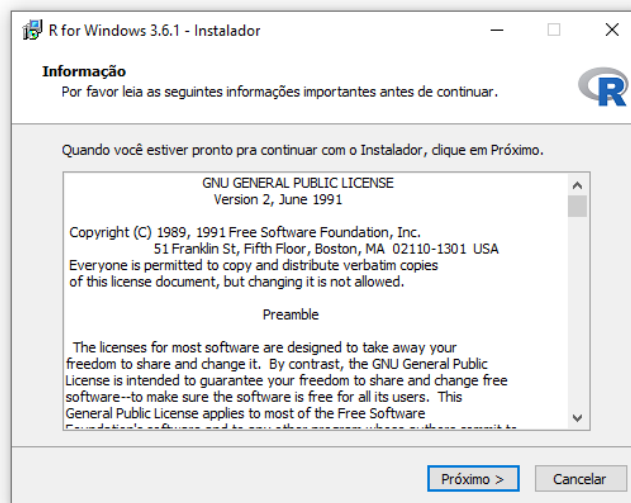


Figure 2.4: Próximo

<input type="checkbox"/> Nome	Data de modificaç...	Tipo	Tamanho
<input checked="" type="checkbox"/> R-3.6.1-win	05/07/2019 15:07	Aplicativo	82.697 KB

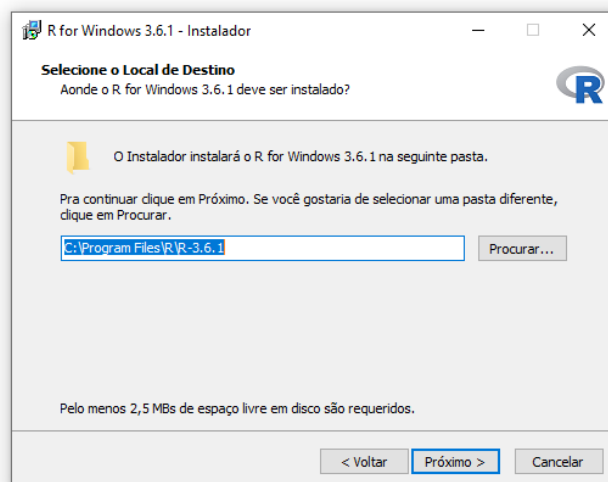


Figure 2.5: Próximo

<input type="checkbox"/> Nome	Data de modificaç...	Tipo	Tamanho
<input checked="" type="checkbox"/> R-3.6.1-win	05/07/2019 15:07	Aplicativo	82.697 KB

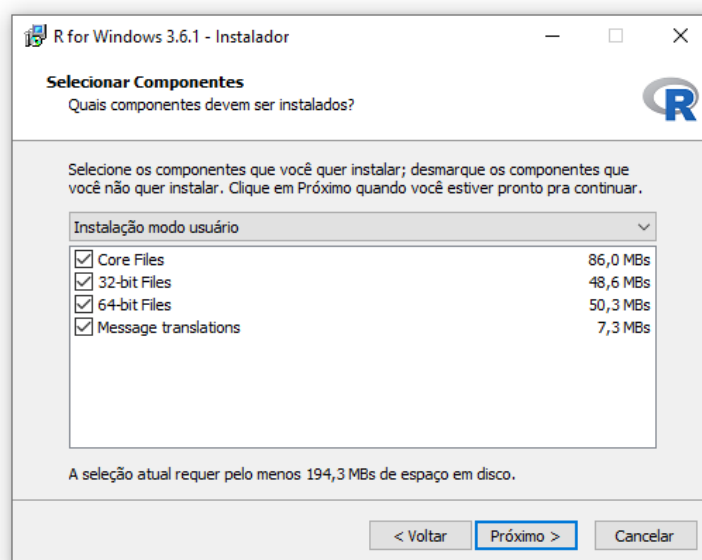


Figure 2.6: Próximo

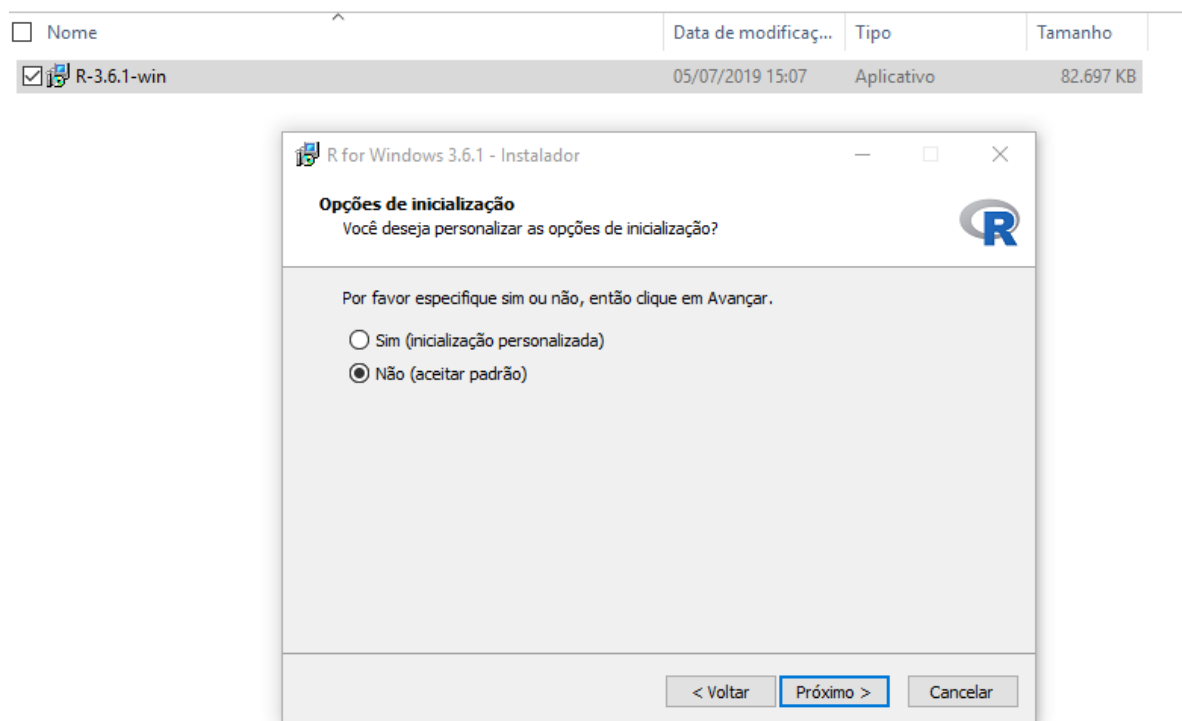



Figure 2.7: Próximo

<input type="checkbox"/> Nome	Data de modificaç...	Tipo	Tamanho
 R-3.6.1-win	05/07/2019 15:07	Aplicativo	82.697 KB

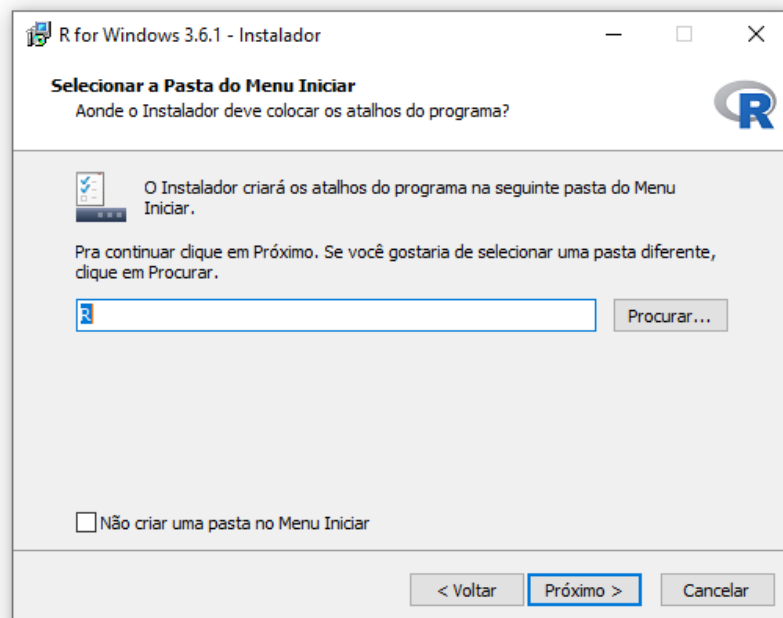



Figure 2.8: Próximo

<input type="checkbox"/> Nome	Data de modificaç...	Tipo	Tamanho
 R-3.6.1-win	05/07/2019 15:07	Aplicativo	82.697 KB

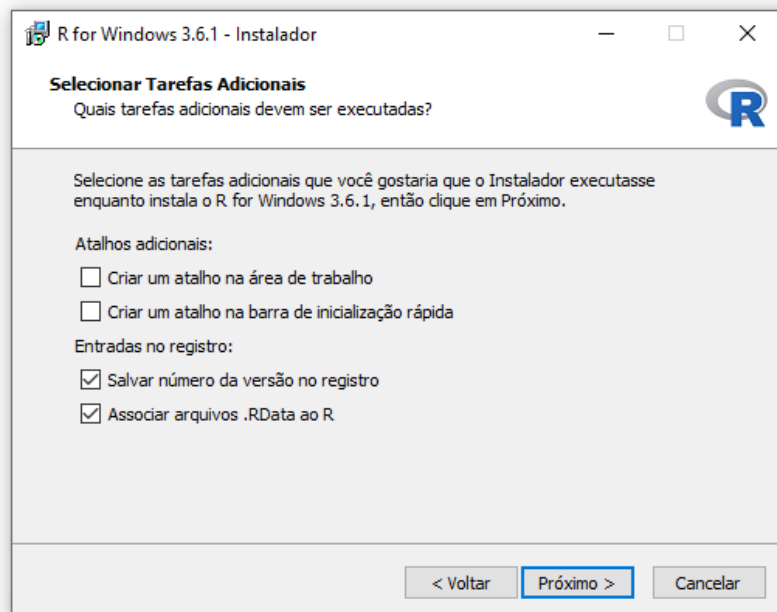


Figure 2.9: Próximo

- 5) Pronto, agora o software R será instalado, como na Figura @ref(fig:windows10). Quando terminar, aparecerá uma janela como apresentado na Figura @ref(fig:windows11).

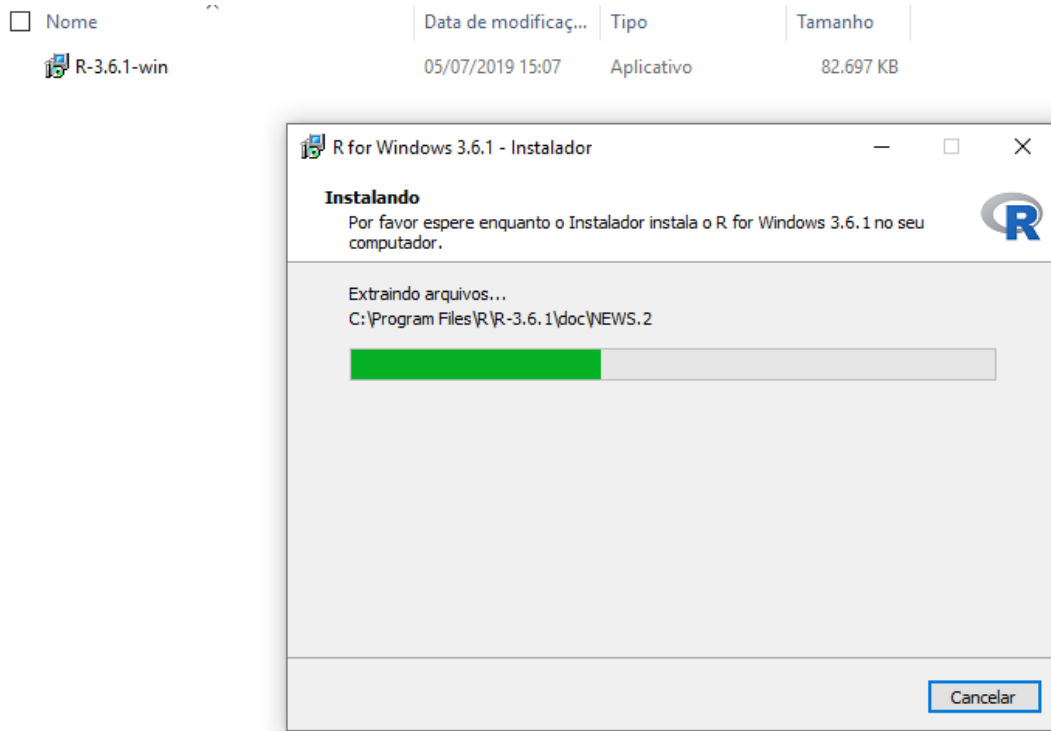


Figure 2.10: Instalação do R

2.0.2 Para MAC

Os passos para instalar o R quando o sistema operacional é OS X (Mac) são os seguintes:

- 1) Entre no [site](#) e clique em Download R for (MAC) OS X, conforme destacado abaixo em retângulo vermelho na Figura @ref(fig:mac1).
- 2) Baixe o pacote R-3.6.1.pkg clicando no link indicado no retângulo vermelho na Figura @ref(fig:mac2). Note que o 3.6.1 é o número da versão mais recente disponível no momento da confecção deste material.
- 3) Caso você não tenha configurado a pasta de downloads, o pacote será baixado na pasta “Downloads”, como mostrado na seguinte Figura @ref(fig:mac3). Observe que dois arquivos são baixados, clique duas vezes no arquivo “R-3.6.1.pkg” para abrir o assistente de instalação que o guiará durante o processo.

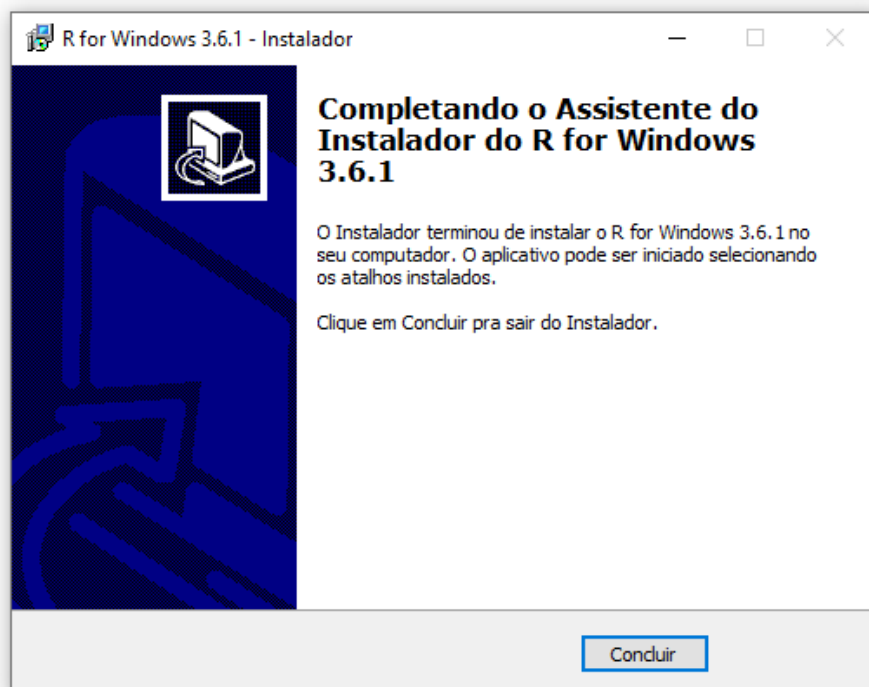


Figure 2.11: Pronto: R instalado



Figure 2.12: Download R para Mac



Important: this release uses Clang 7.0.0 and GNU Fortran 6.1, neither of which is supplied by Apple. If you wish to compile R packages from sources, you will need to download and install those tools - see the [tools](#) directory.

- 4) Acompanhe os passos indicados pelo instalador (Figura @ref(fig:mac4)).



Figure 2.15: Instalação

- 5) Deve concordar com os termos da licença, clique em “Agree” (Figura @ref(fig:mac5)).
- 6) Selecione o lugar onde instalará o programa, no caso de ter o disco particionado e assim desejar instalar em uma parte específica. Caso contrário, continue (Figura @ref(fig:mac6) e @ref(fig:mac7)).
- 7) Para finalizar a instalação, o assistente lhe pedirá nome de usuário e senha do seu notebook, como apresentado na Figura @ref(fig:mac8).
- 8) Pronto, agora o software R será instalado, como na Figura @ref(fig:mac9). Quando terminar, aparecerá uma janela como apresentado na Figura @ref(fig:mac10).

2.0.3 Para Linux

A instalação do R no Linux depende da distribuição utilizada. Entre neste [link](#) para acessar a página do R e clique em Download R for Linux, como no link destacado em retângulo vermelho

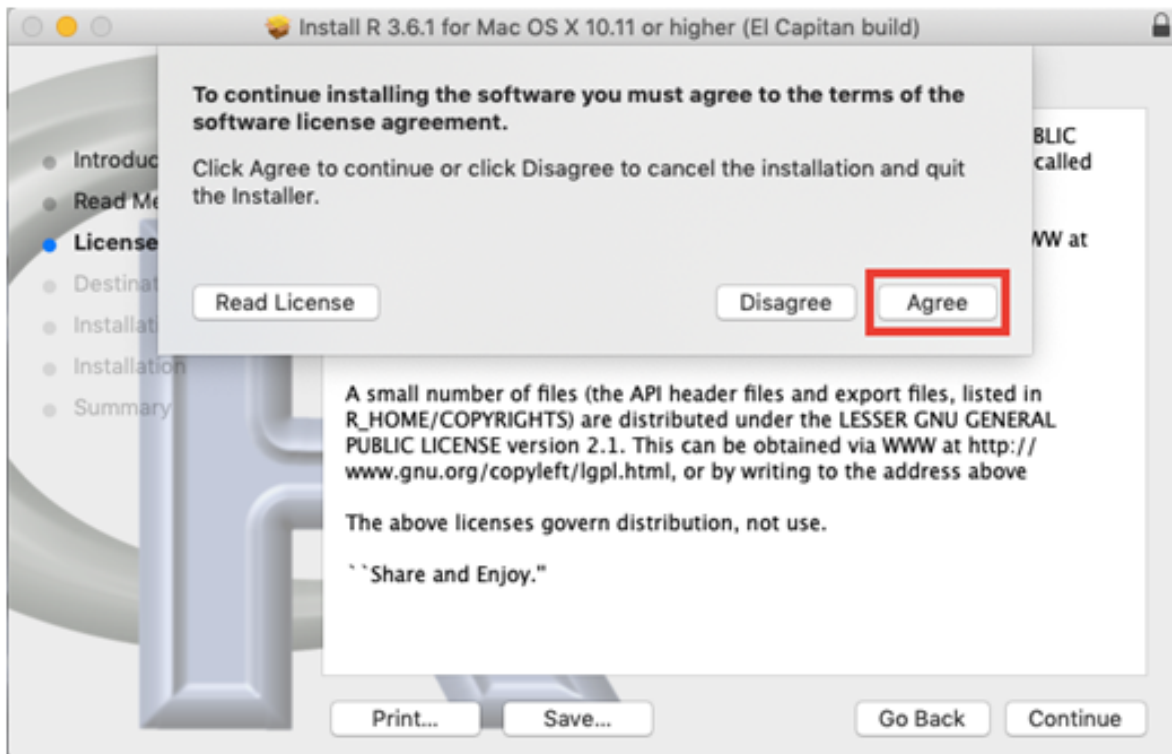


Figure 2.16: Instalação



Figure 2.17: Instalação

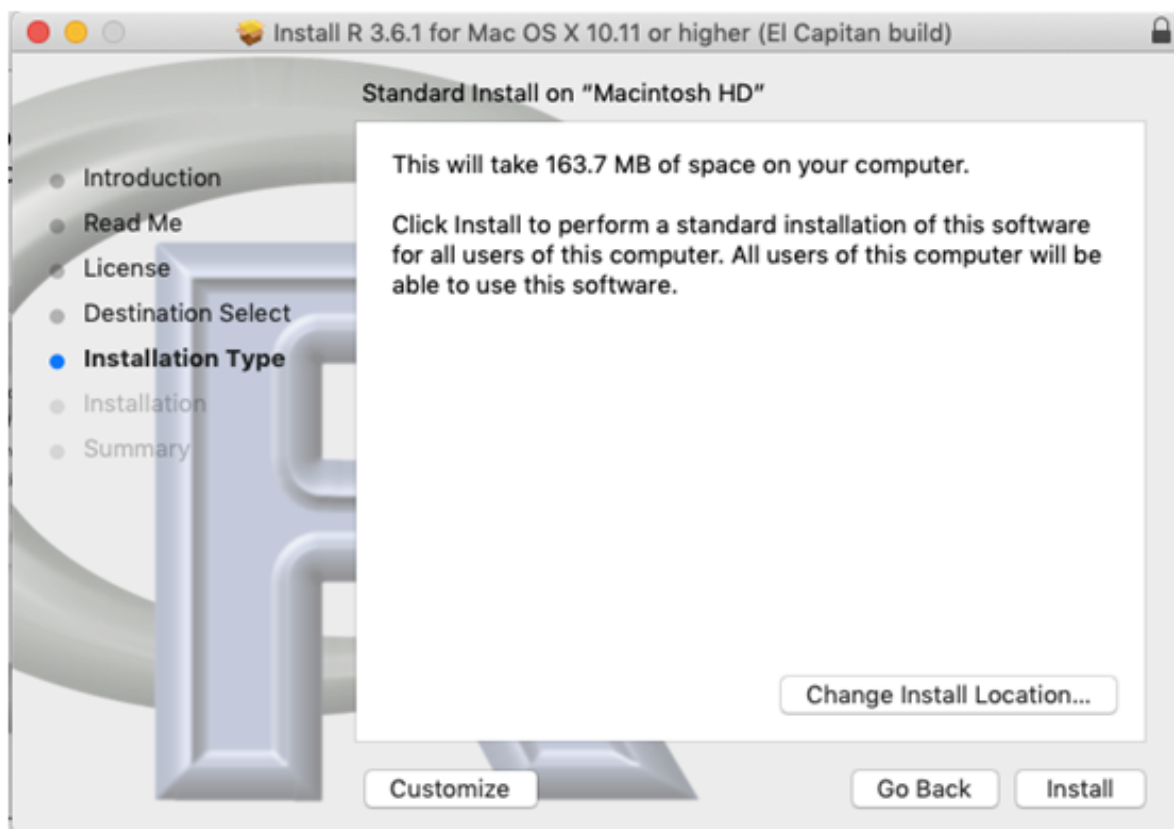


Figure 2.18: Instalação

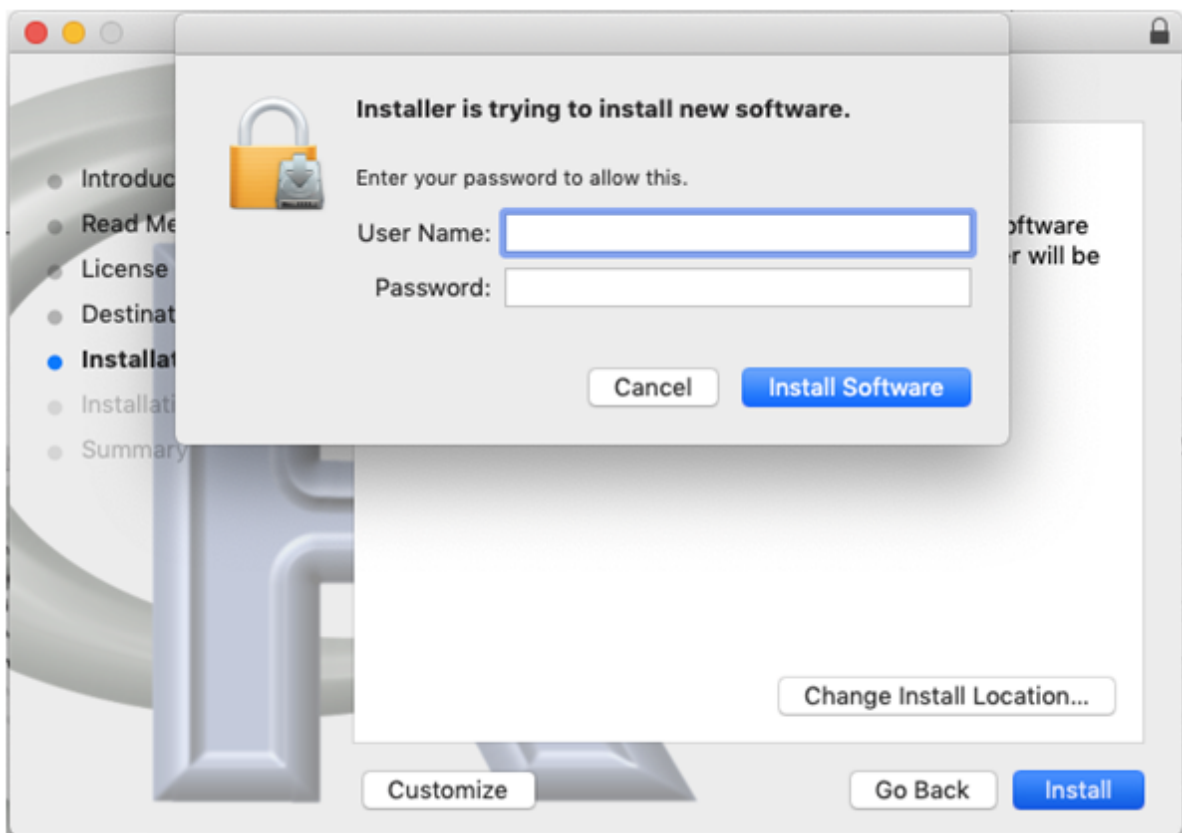


Figure 2.19: Instalação

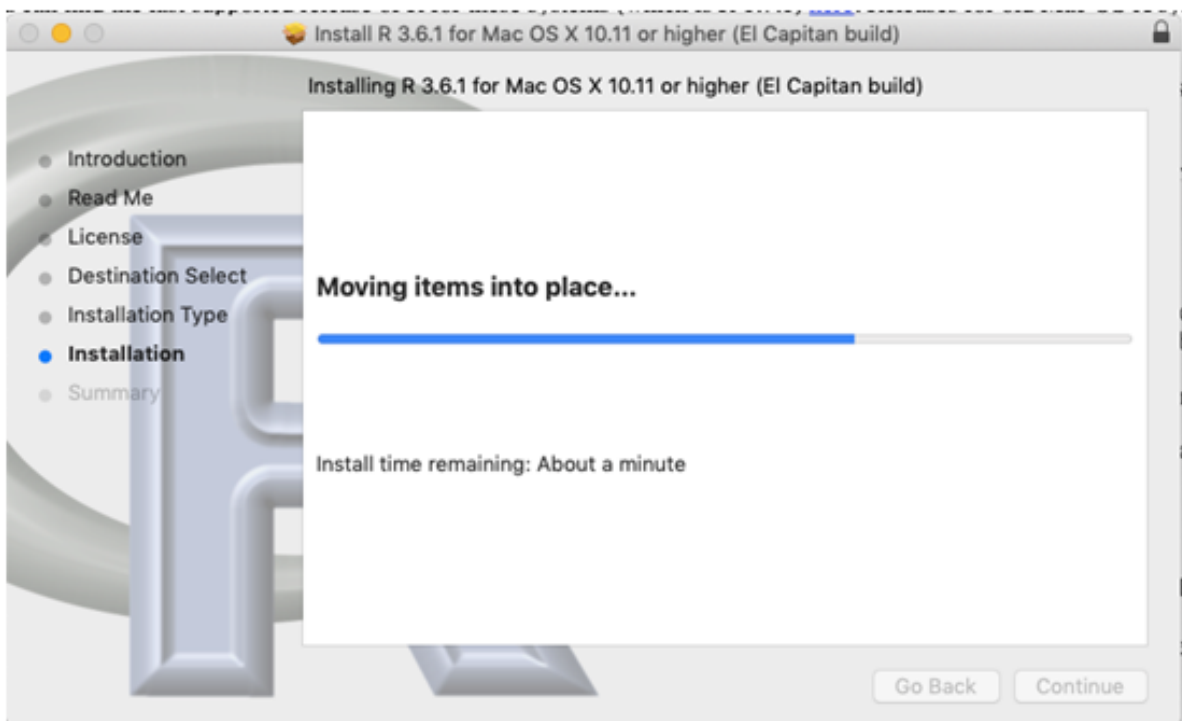


Figure 2.20: Instalação

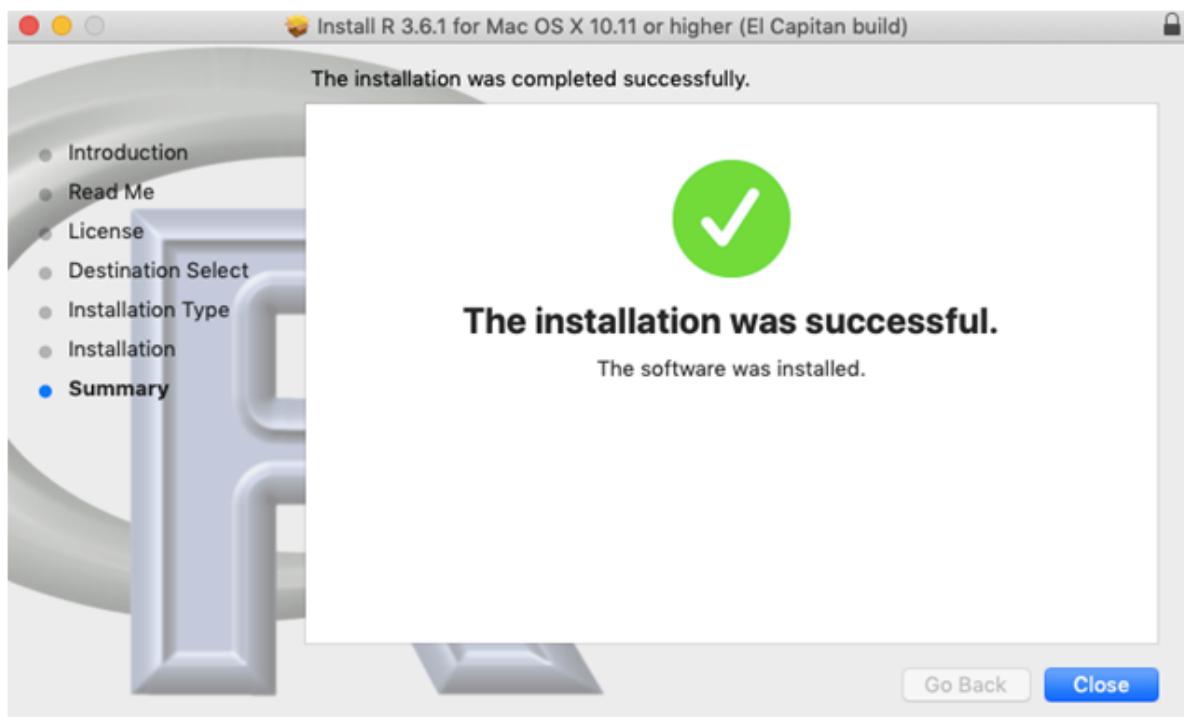


Figure 2.21: Instalação

na Figura @ref(fig:linux1). Em seguida, clique no link referente à distribuição utilizada (Figura @ref(fig:linux2)).

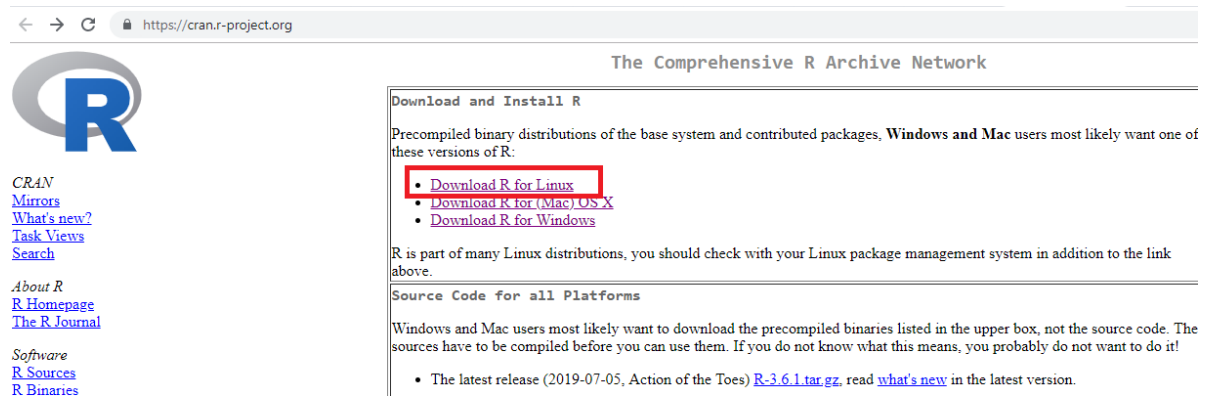



Figure 2.22: Download em Linux

← → ↻ <https://cran.r-project.org>








[CRAN](#)
[Mirrors](#)
[What's new?](#)
[Task Views](#)
[Search](#)

[About R](#)
[R Homepage](#)
[The R Journal](#)

[Software](#)
[R Sources](#)
[R Binaries](#)
[Packages](#)
[Other](#)

Index of /bin/linux

Name	Last modified	Size	Description
 Parent Directory		-	
 debian/	2019-06-26 06:55	-	
 redhat/	2014-07-27 21:12	-	
 suse/	2012-02-16 15:09	-	
 ubuntu/	2019-07-05 15:07	-	

Apache Server at cran.r-project.org Port 443

Figure 2.23: Download em Linux

3 Instalação RStudio

O RStudio é um conjunto de ferramentas integradas projetadas (IDE - Integrated Development Environment) da linguagem R para auxiliar na produtividade ao utilizar o R.

3.0.1 Para Windows

- 1) Entre neste [link](#) e clique em Download como em destaque na Figura @ref(fig:rswindows1).

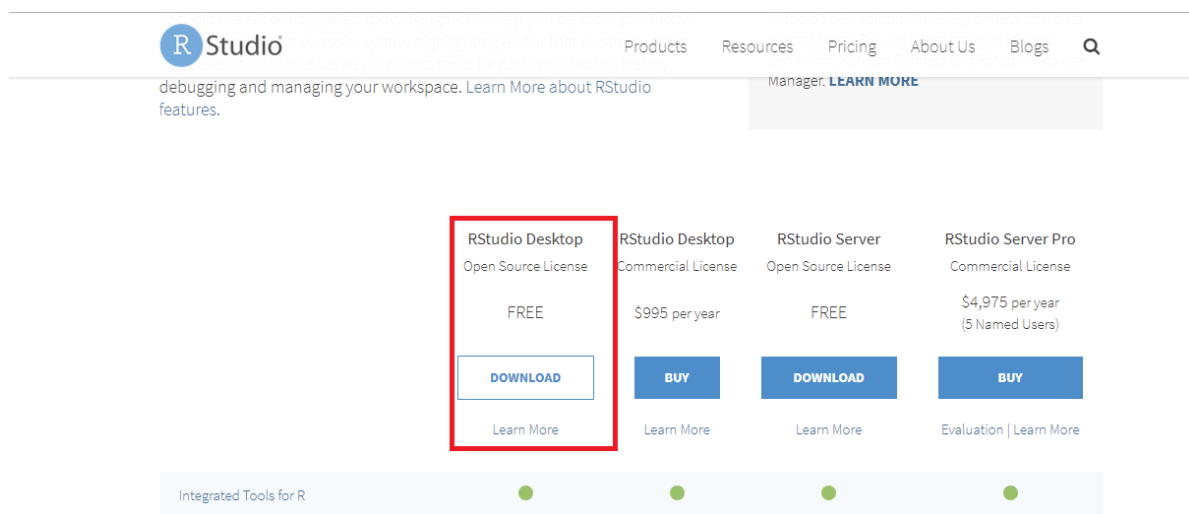


Figure 3.1: Site para download do RStudio

- 2) Clique no instalador em destaque na Figura @ref(fig:rswindows2).
- 3) Ao clicar no link, será feito o download do instalador e salvo na pasta de interesse. No caso da Figura @ref(fig:rswindows3), o instalador está na pasta Downloads. Dê dois cliques no botão esquerdo no arquivo para iniciar o download do arquivo.
- 4) Clique em “Próximo” nas próximas janelas e na última “Instalar”, como nas Figuras @ref(fig:rswindows4) a @ref(fig:rswindows6).
- 5) Pronto, a instalação será iniciada, como na Figura @ref(fig:rswindows7).

RStudio 1.2 requires a 64-bit operating system, and works exclusively with the 64 bit version of R. If you are on a 32 bit system or need the 32 bit version of R, you can use an older version of RStudio.

Installers for Supported Platforms

Installers	Size	Date	MD5
RStudio 1.2.1335 - Windows 7+ (64-bit)	126.9 MB	2019-04-08	d0e2470f1f8ef4cd35a669aa323a2136
RStudio 1.2.1335 - Mac OS X 10.12+ (64-bit)	121.1 MB	2019-04-08	6c570b0e2144583f7c48c284ce299eef
RStudio 1.2.1335 - Ubuntu 14/Debian 8 (64-bit)	92.2 MB	2019-04-08	c1b07d0511469abfe582919b183eee83
RStudio 1.2.1335 - Ubuntu 16 (64-bit)	99.3 MB	2019-04-08	c142d69c210257fb10d18c045fff13c7
RStudio 1.2.1335 - Ubuntu 18/Debian 10 (64-bit)	100.4 MB	2019-04-08	71a8d1990c0d97939804b46cfb0aea75
RStudio 1.2.1335 - Fedora 19/RedHat 7 (64-bit)	114.1 MB	2019-04-08	296b6ef88969a91297fab6545f256a7a
RStudio 1.2.1335 - Debian 9 (64-bit)	100.6 MB	2019-04-08	1e32d4d6f6e216f086a81ca82ef65a91
RStudio 1.2.1335 - OpenSUSE 15 (64-bit)	101.6 MB	2019-04-08	2795a63c7efd8e2aa2dae86ba09a81e5
RStudio 1.2.1335 - SLES/OpenSUSE 12 (64-bit)	94.4 MB	2019-04-08	c65424b06ef6737279d982db9eefcae1

Figure 3.2: Link para download do RStudio

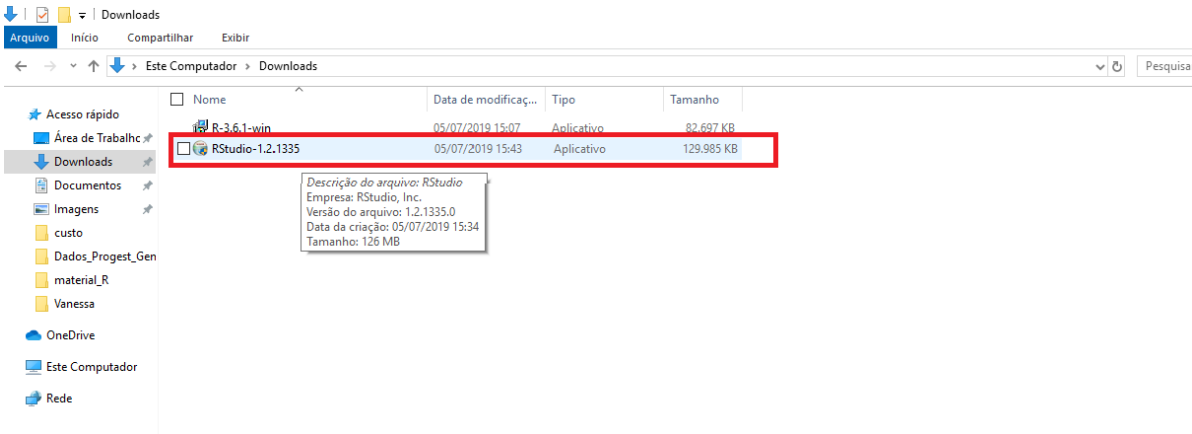


Figure 3.3: Instalador

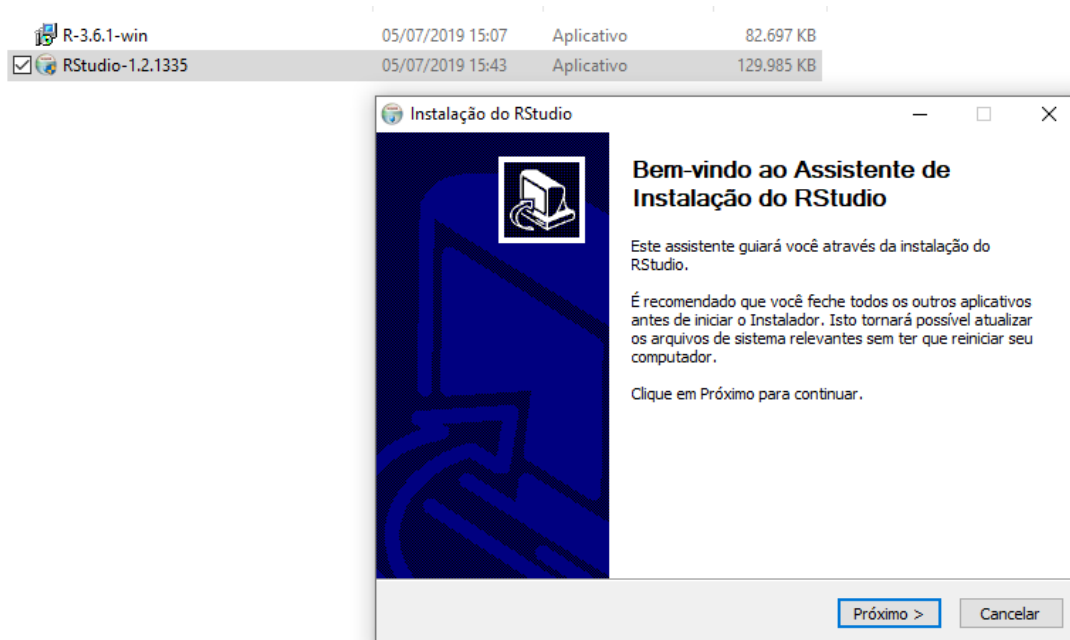


Figure 3.4: Instalação

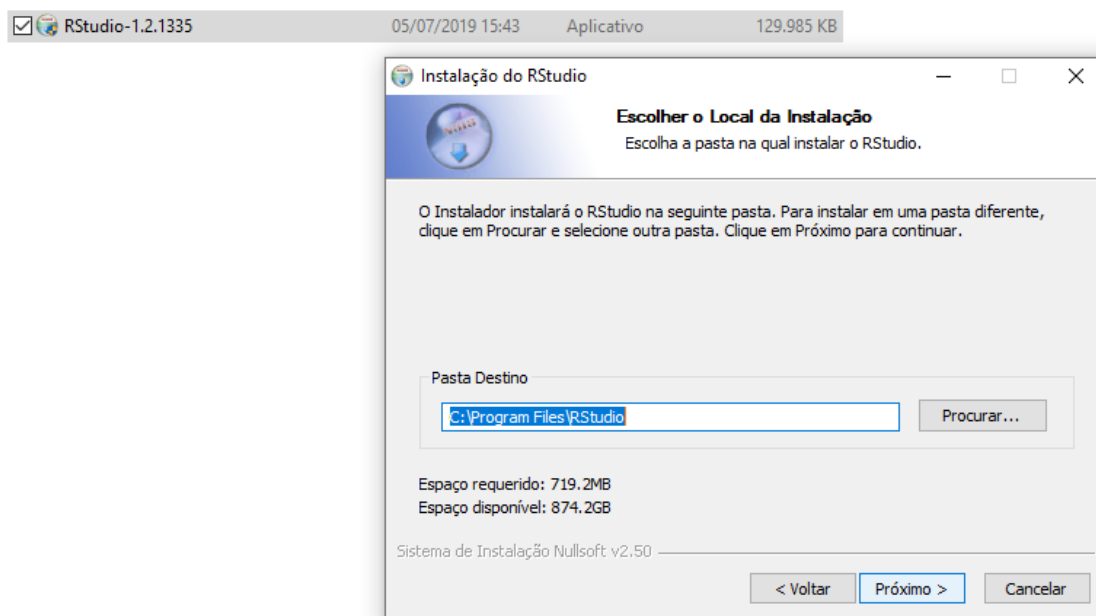


Figure 3.5: Instalação

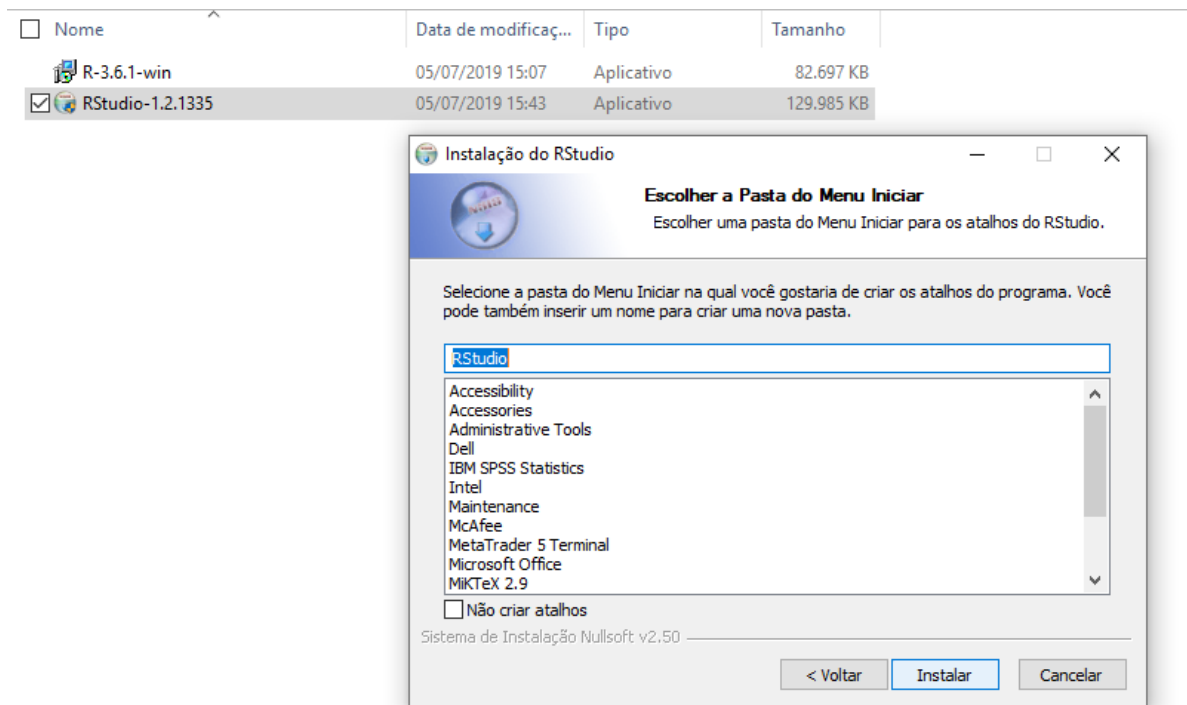


Figure 3.6: Instalação

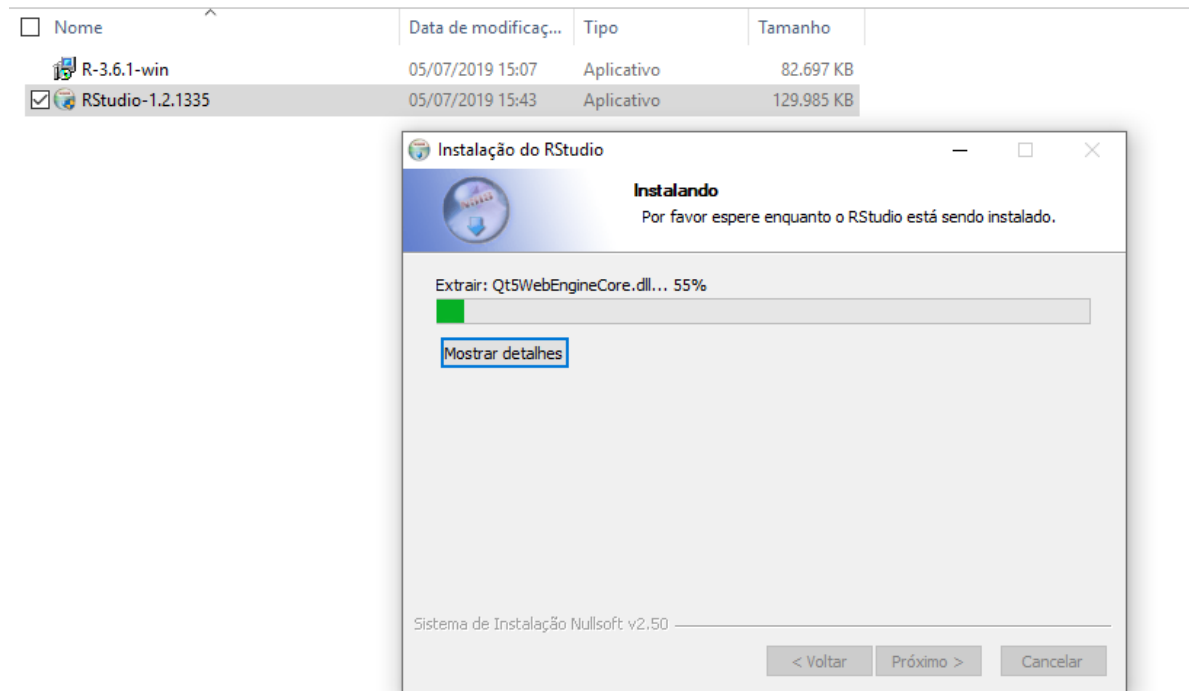


Figure 3.7: Instalação

3.0.2 Para MAC

- 1) Entre neste [link](#) e clique em Download como em destaque na Figura @ref(fig:rsmac1).

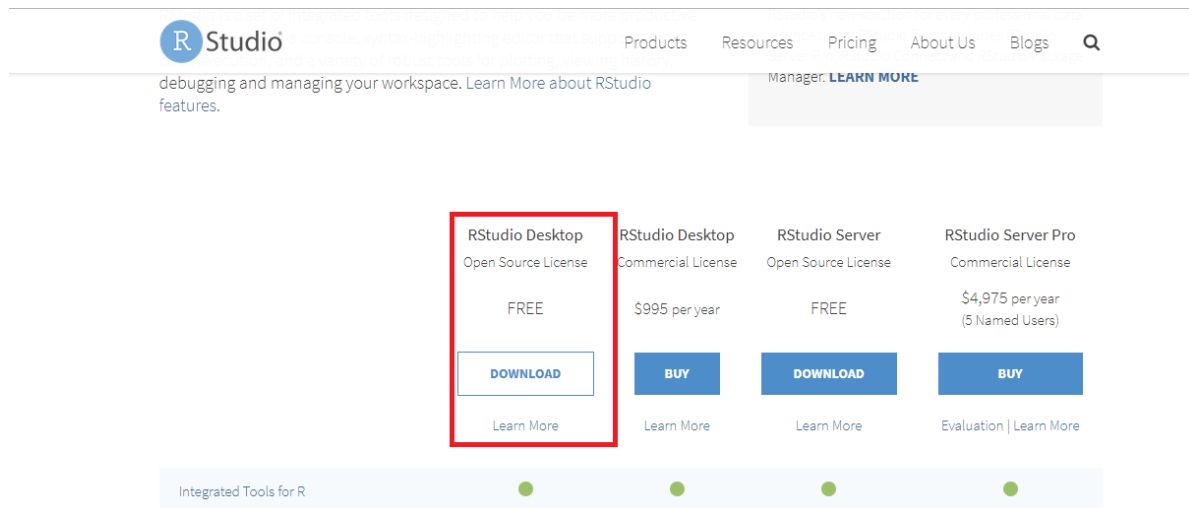




Figure 3.8: Site para download do RStudio

- 2) Clique no instalador como destacado na Figura @ref(fig:rsmac2).
- 3) Ao clicar no link, será feito o download do instalador e salvo na pasta de interesse. Caso você não tenha configurado a pasta de downloads, o instalador ficará na pasta “Downloads”, como na Figura @ref(fig:rsmac3).
- 4) Clicando duas vezes no arquivo “RStudio-1.2.1335.dmg” (versões mais atual do RStudio), será feita a descarga do mesmo abrindo a janela conforme na Figura @ref(fig:rsmac4). Clique no aplicativo de RStudio destacado em vermelho também na Figura @ref(fig:rsmac4).
- 5) O instalador pode perguntar se está seguro que o aplicativo será baixado da internet e clique em “Open” (Figura @ref(fig:rsmac5)).
- 6) Pronto! Imediatamente abre o RStudio, como na Figura @ref(fig:rsmac6), e você já pode utilizá-lo.

3.0.3 Para Linux

- 1) Entre neste [link](#) e clique em Download como em destaque na Figura @ref(fig:rslinux1).
- 2) Clique no link referente à distribuição utilizada (Figura @ref(fig:rslinux2)).



[Products](#)
[Resources](#)
[Pricing](#)
[About Us](#)
[Blogs](#)


RStudio Desktop 1.2.1335 — Release Notes

RStudio requires R 3.0.1+. If you don't already have R, download it [here](#).

Linux users may need to [import RStudio's public code-signing key](#) prior to installation, depending on the operating system's security policy.

RStudio 1.2 requires a 64-bit operating system, and works exclusively with the 64 bit version of R. If you are on a 32 bit system or need the 32 bit version of R, you can use an [older version of RStudio](#).

Installers for Supported Platforms

Installers	Size	Date	MD5
RStudio 1.2.1335 - Windows 7+ (64-bit)	126.9 MB	2019-04-08	d0e2470f1f8ef4cd35a669aa323a2136
RStudio 1.2.1335 - Mac OS X 10.12+ (64-bit)	121.1 MB	2019-04-08	6c570b0e2144583f7c48c284ce299eef
RStudio 1.2.1335 - Ubuntu 14/Debian 8 (64-bit)	92.2 MB	2019-04-08	e1b07d0511469abfe582919b183eee83
RStudio 1.2.1335 - Ubuntu 16 (64-bit)	99.3 MB	2019-04-08	e142d69c210257fb10d18c045fff13c7
RStudio 1.2.1335 - Ubuntu 18/Debian 10 (64-bit)	100.4 MB	2019-04-08	71a8d1990c0d97939804b46cfb0aea75
RStudio 1.2.1335 - Fedora 19/RedHat 7 (64-bit)	114.1 MB	2019-04-08	296b6ef88969a91297fab6545f256a7a
RStudio 1.2.1335 - Debian 9 (64-bit)	100.6 MB	2019-04-08	1e32d4d6f6e216f086a81ca82ef65a91
RStudio 1.2.1335 - OpenSUSE 15 (64-bit)	101.6 MB	2019-04-08	2795a63c7efd8e2aa2dae86ba09a81e5
RStudio 1.2.1335 - SLES/OpenSUSE 12 (64-bit)	94.4 MB	2019-04-08	c65424b06ef6737279d982db9eeefcae1

Figure 3.9: Site para download do RStudio para Mac

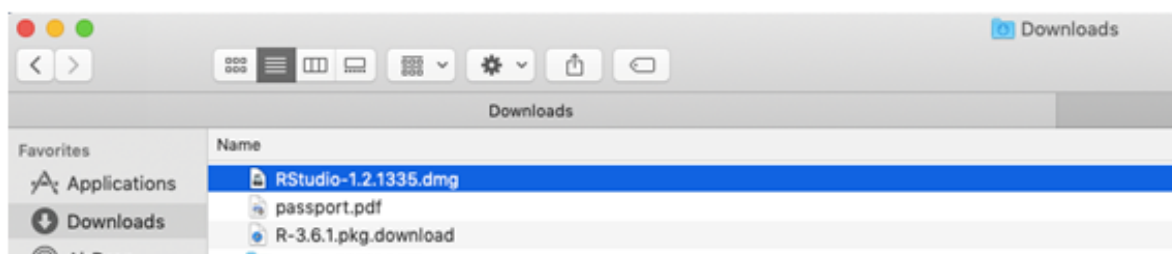


Figure 3.10: Instalador salvo em pasta



Figure 3.11: Instalação

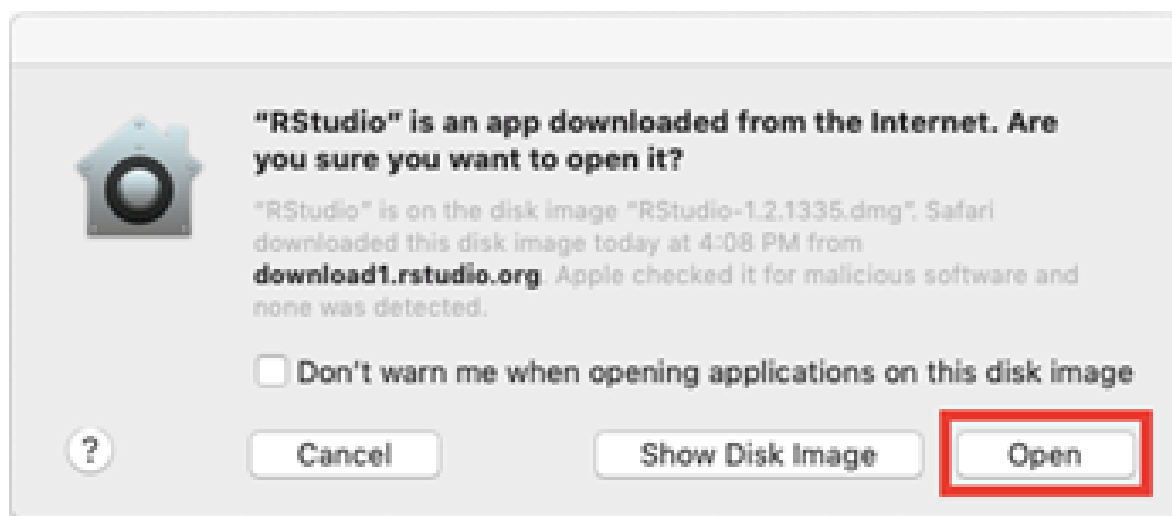


Figure 3.12: Instalação

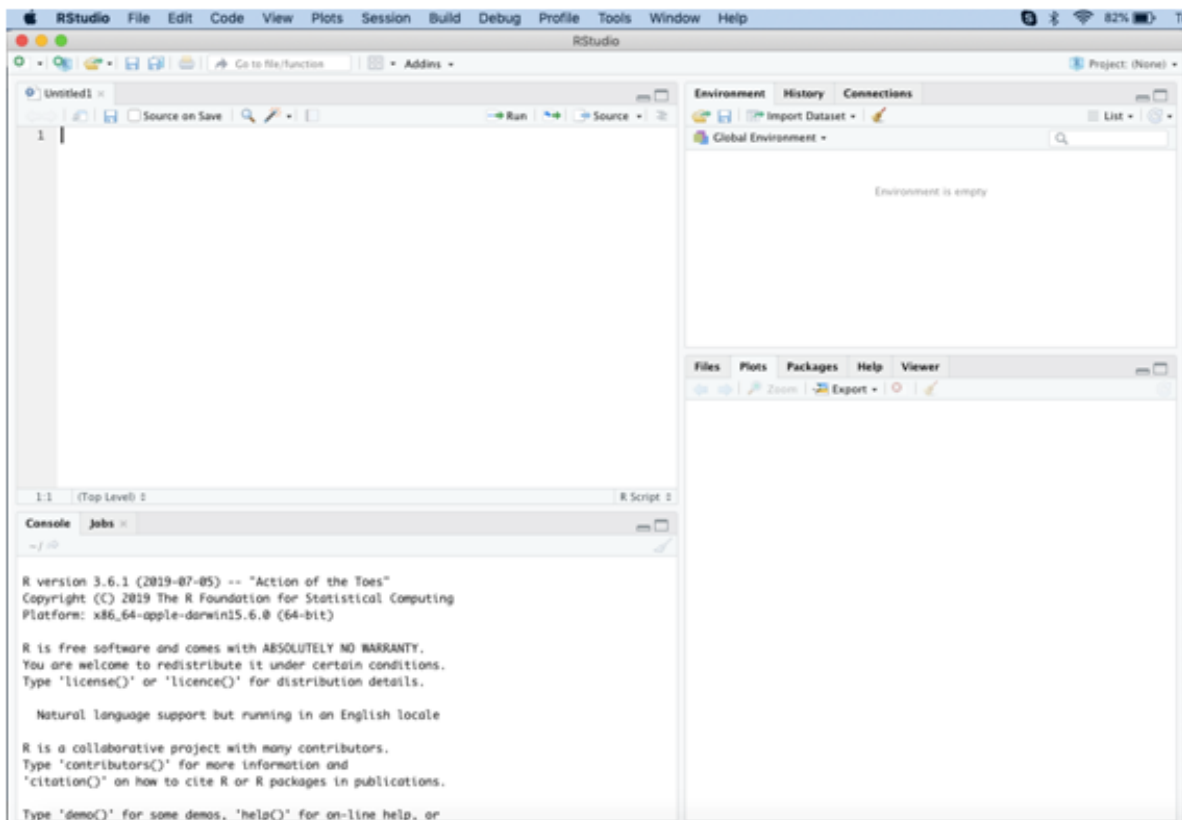


Figure 3.13: Instalação

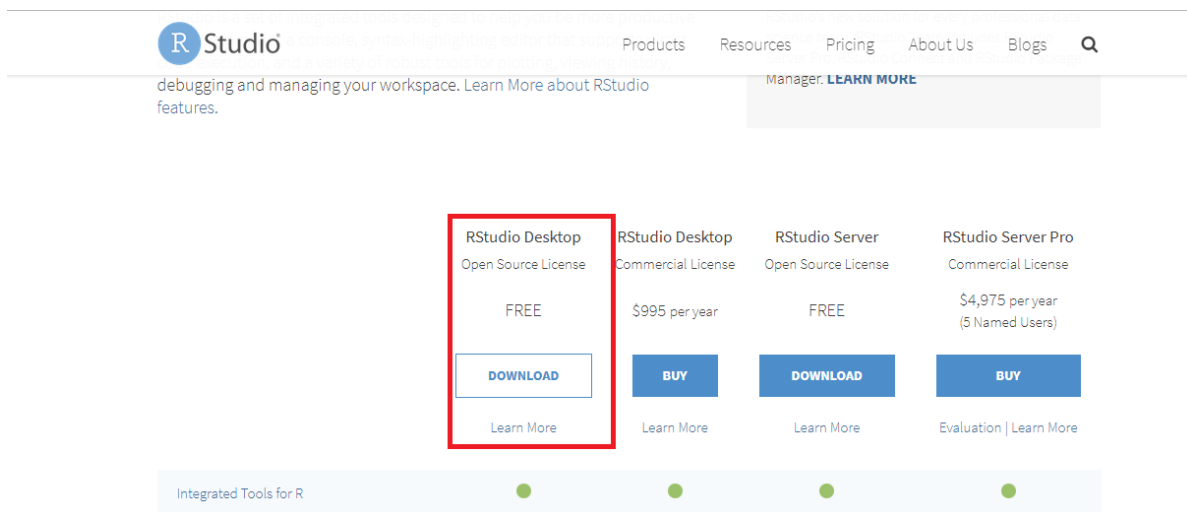



Figure 3.14: Site para download do RStudio



RStudio

RStudio is a 64-bit operating system, and works exclusively on 64-bit operating systems. If you are using a 32-bit version of R, you can use an older version of RStudio.

[Products](#)
[Resources](#)
[Pricing](#)
[About Us](#)
[Blogs](#)
[Search](#)

Installers for Supported Platforms

Installers	Size	Date	MD5
RStudio 1.2.1335 - Windows 7+ (64-bit)	126.9 MB	2019-04-08	d0e2470f1f8ef4cd35a669aa323a2136
RStudio 1.2.1335 - Mac OS X 10.12+ (64-bit)	121.1 MB	2019-04-08	6c570b0e2144583f7c48c284ce299eef
RStudio 1.2.1335 - Ubuntu 14/Debian 8 (64-bit)	92.2 MB	2019-04-08	c1b07d0511469abfe582919b183eee83
RStudio 1.2.1335 - Ubuntu 16 (64-bit)	99.3 MB	2019-04-08	c142d69c210257fb10d18c045fff13c7
RStudio 1.2.1335 - Ubuntu 18/Debian 10 (64-bit)	100.4 MB	2019-04-08	71a8d1990c0d97939804b46cfb0aea75
RStudio 1.2.1335 - Fedora 19/RedHat 7 (64-bit)	114.1 MB	2019-04-08	296b6ef88969a91297fab6545f256a7a
RStudio 1.2.1335 - Debian 9 (64-bit)	100.6 MB	2019-04-08	1e32d4d6f6e216f086a81ca82ef65a91
RStudio 1.2.1335 - OpenSUSE 15 (64-bit)	101.6 MB	2019-04-08	2795a63c7efd8e2aa2dae86ba09a81e5
RStudio 1.2.1335 - SLES/OpenSUSE 12 (64-bit)	94.4 MB	2019-04-08	c65424b06ef6737279d982db9eefcae1

Figure 3.15: Download do RStudio

4 Primeiros passos no RStudio

O RStudio é um conjunto de ferramentas integradas projetadas (IDE - Integrated Development Environment) da linguagem R para editar e executar os códigos em R.

Tem quatro áreas, conforme a Figura @ref(fig:telarstudio1).

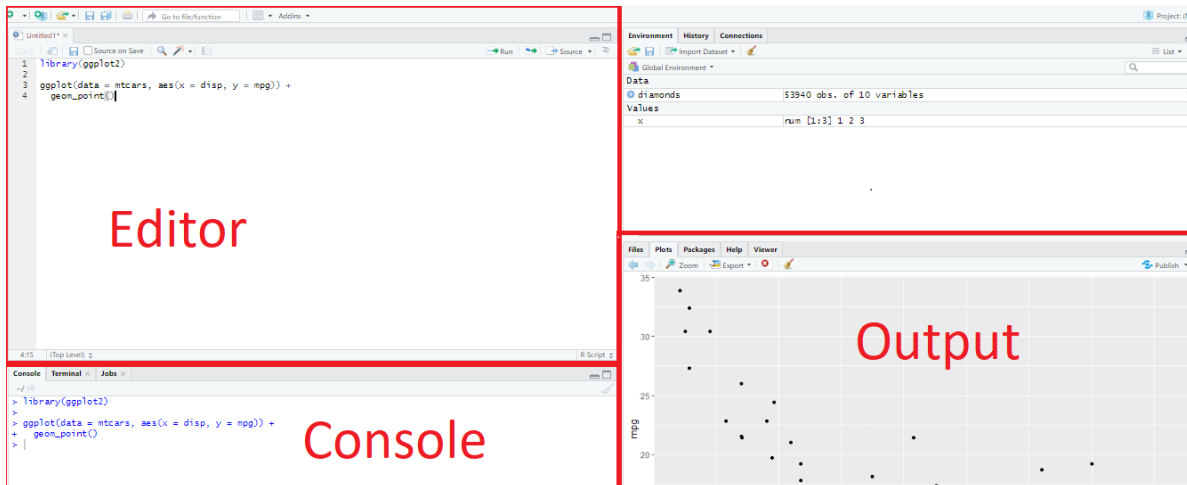


Figure 4.1: Visualização do RStudio

A seguir descrevemos melhor os painéis e abas do RStudio:

- Editor/Scripts: é onde escrever os códigos. Arquivos do tipo .R.
- Console: executar os comandos e ver os resultados.
- Environment: painel com todos os objetos criados.
- History: história dos comandos executados.
- Files: navegar em pastas e arquivos.
- Plots: onde os gráficos serão apresentados.
- Packages: pacotes instalados (sem ticar) e habilitados (ticados).
- Help: retorna o tutorial de ajuda do comando solicitado com `help()` ou `?comando`. Ver melhor como pedir ajuda no R no final deste capítulo.

O usuário pode alterar a aparência do RStudio, como fonte e cor. Como exemplo, as Figuras @ref(fig:telarstudio2) e @ref(fig:telarstudio3) apresentam os passos para mudar o tema do script. No exemplo, deixar com fundo preto.

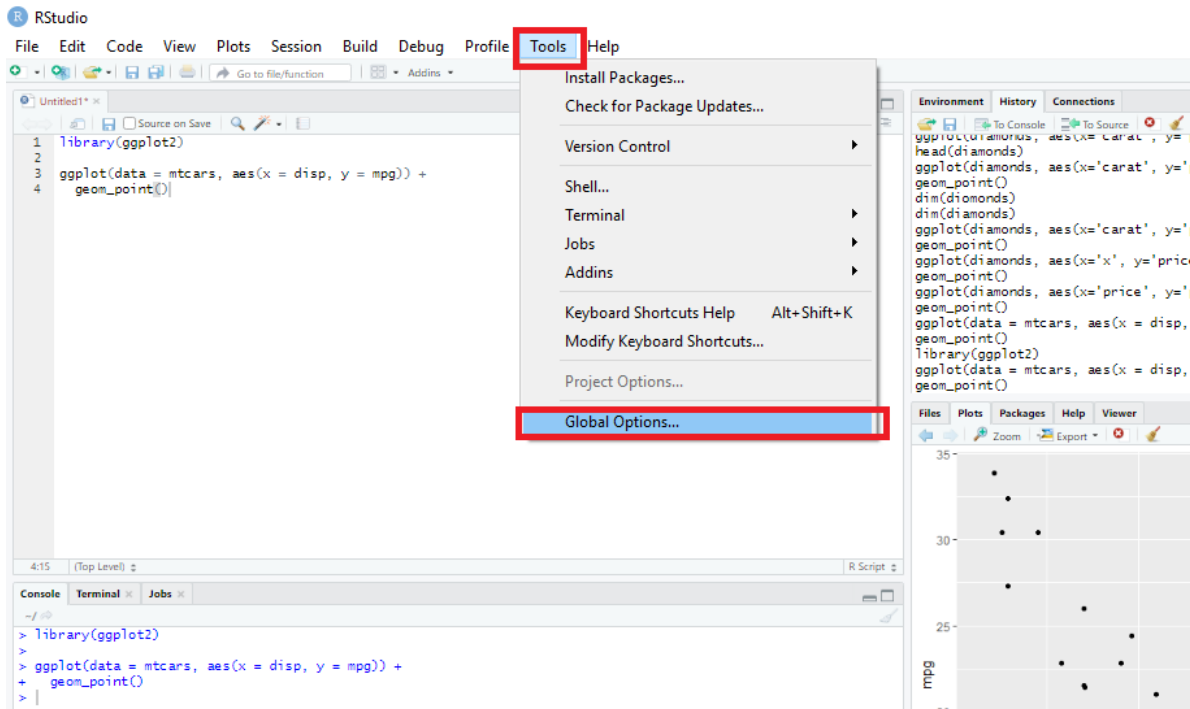


Figure 4.2: Ferramentas de aparência do RStudio

Ainda no menu Tools → Global Options → Pane Layout, o usuário pode organizar a ordem dos quadrantes do RStudio, como apresentado nas Figuras @ref(fig:telarstudio4), @ref(fig:telarstudio5) e @ref(fig:telarstudio6). No exemplo, o painel Console foi transferido para o lado do painel Script, o que facilita a visualização dos comandos rodados.

4.0.1 Projetos

Uma funcionalidade importante é a criação de projetos, permitindo dividir o trabalho em múltiplos ambientes, cada um com o seu diretório, documentos e workspace.

Para criar um projeto, os seguintes passos podem ser seguidos:

- 1) Clique na opção “File” do menu, e então em “New Project”.
- 2) Clique em “New Directory”.
- 3) Clique em “New Project”.

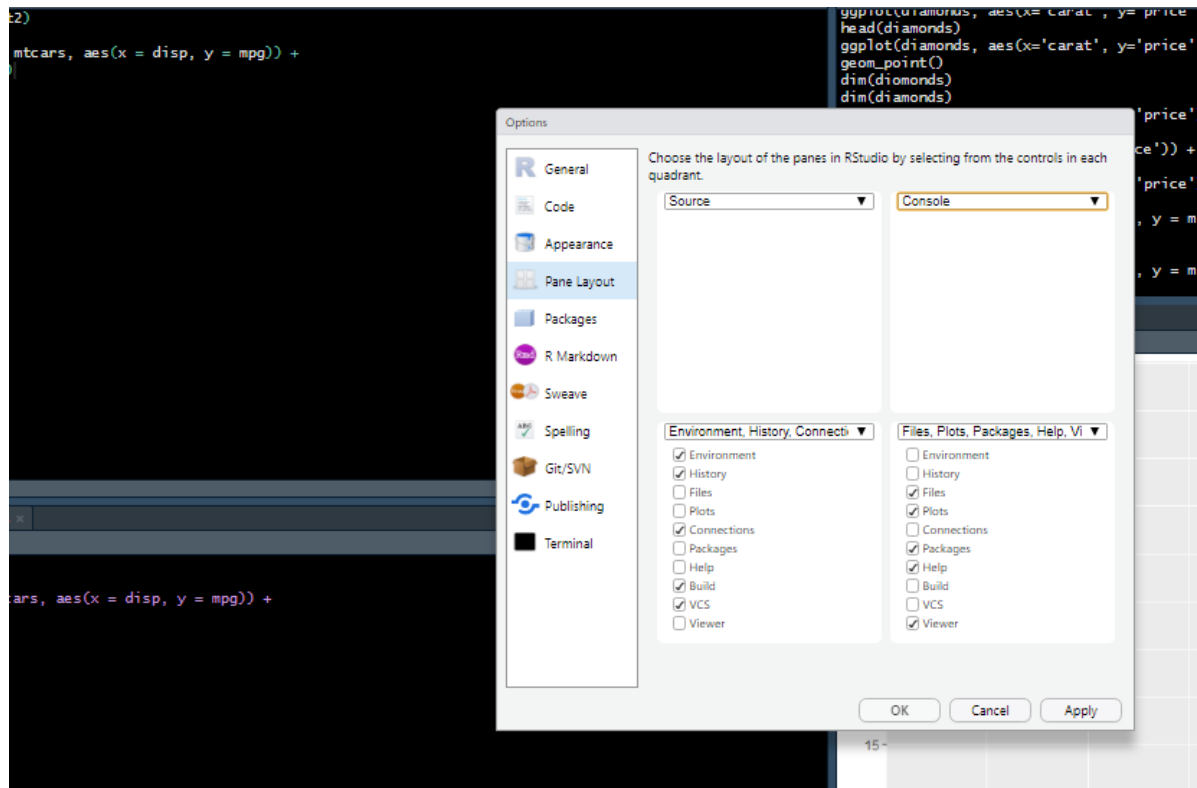


Figure 4.5: Ferramentas de aparência do RStudio

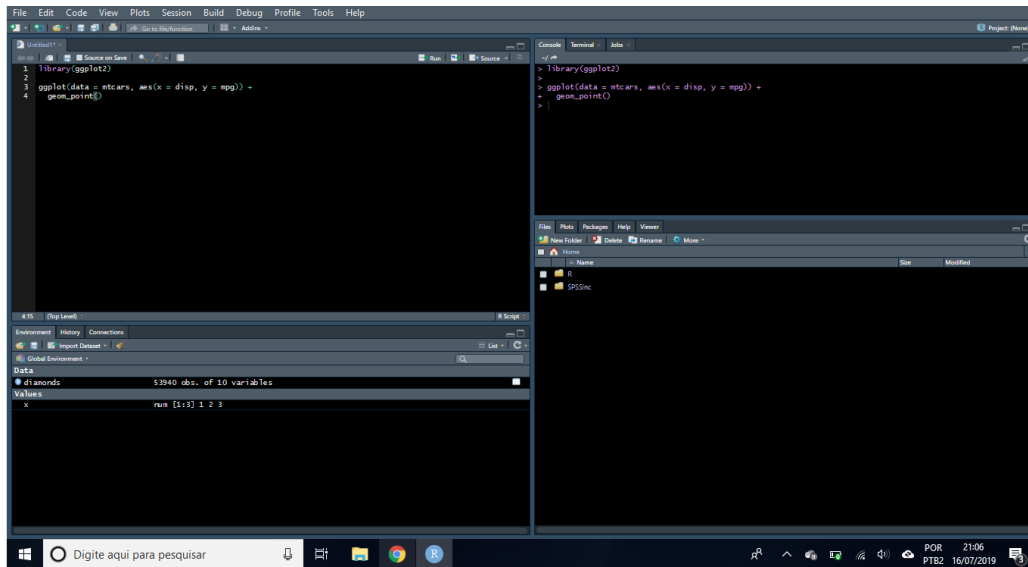


Figure 4.6: Ferramentas de aparência do RStudio

- 4) Escreva o nome do diretório (pasta) onde deseja manter seu projeto, exemplo: “my_project”.
- 5) Clique no botão “Create Project”.

Para criar um novo script para escrever os códigos, vá em File -> New File -> R Script

4.0.2 Boas práticas

Comente bem o seu código: é possível fazer comentários usando o símbolo ‘#’. É sempre bom explicar o que uma variável armazena, o que uma função faz, por que alguns parâmetros são passados para uma determinada função, qual é o objetivo de um trecho de código etc.

Evite linhas de código muito longas: usar linhas de código mais curtas ajuda na leitura do código.

Escreva um código organizado. Por exemplo, adote um padrão no uso de minúsculas e maiúsculas, uma lógica única na organização de pastas e arquivos, pode ser adotada uma breve descrição (como comentário) indicando o que um determinado script faz.

Carregue todos os pacotes que irá usar sempre no início do arquivo: quando alguém abrir o seu código será fácil identificar quais são os pacotes que devem ser instalados e quais dependências podem existir.

5 Primeiros passos no R

Posso escrever o código no Script e submetê-lo ao apertar o botão “Run” ou com o atalho no teclado Cmd/Ctrl+Enter.

5.1 R como calculadora

1) Operadores

```
#adição  
10+15
```

```
[1] 25
```

```
#subtração  
10-2
```

```
[1] 8
```

```
#multiplicação  
2*10
```

```
[1] 20
```

```
#divisão  
30/2
```

```
[1] 15
```

```
#raiz quadrada  
sqrt(4)
```

```
[1] 2
```

```
#potência  
2^2
```

```
[1] 4
```

Se você digitar um comando incompleto, como `10 *`, o R mostrará um `+`. Isso não tem a ver com a soma e apenas que o R está esperando você completar seu comando. Termine seu comando ou aperte `Esc` para recomeçar.

Vale também ressaltar que se você digitar um comando que o R não reconhece, ele retornará uma mensagem de erro e você pode digitar outro comando normalmente em seguida.

5.2 Atribuição

Podemos salvar valores dentro de um objeto, que é simplesmente um nome que guarda um valor, vetor, matriz, lista ou base de dados.

Para atribuir a um objeto, o sinal de atribuição é `=` ou `<-` (preferível).

Exemplos:

```
x <- 10/2  
x
```

```
[1] 5
```

```
X
```

```
Error in eval(expr, envir, enclos): object 'X' not found
```

Por que tivemos um erro acima?

O R é case sensitive, isto é, faz a diferenciação entre as letras minúsculas e maiúsculas. Portanto, `x` é diferente de `X`.

5.3 Objetos em R

Existem cinco classes básicas no R:

- character: “UAH!”
- numeric: 0.95 (números reais)
- integer: 100515 (inteiros)
- complex: $2 + 5i$ (números complexos, $a + bi$)
- logical: TRUE (booleanos, TRUE/FALSE)

Vamos atribuir a x a string banana.

```
x <- banana
```

Error in eval(expr, envir, enclos): object 'banana' not found

```
x <- "banana"
x
```

```
[1] "banana"
```

O primeiro caso (`x <- banana`) não deu certo, pois ele entendeu que estamos atribuindo a x outro objeto banana, que não foi declarado. Para atribuir o string banana a x, precisamos colocar entre aspas ou aspas simples. Uma string sem aspas é entendido como um objeto, veja abaixo:

```
banana <- 30
x <- banana
x
```

```
[1] 30
```

Para saber a classe de um objeto, use a função `class()`.

```
y <- "ola"
class(y)
```

```
[1] "character"
```

```
x <- 2.5  
class(x)
```

```
[1] "numeric"
```

5.3.1 Apagar objetos

E se eu quiser apagar um objeto?

```
x <- 20  
x
```

```
[1] 20
```

```
remove(x)  
x
```

Error in eval(expr, envir, enclos): object 'x' not found

E se eu quiser limpar o console - apaga todos os objetos atribuídos até aqui:

```
rm(list=ls())
```

5.4 Vetores

Como atribuir vários valores a um objeto? Para entrar com vários números (ou nomes, ou qualquer outro grupo de coisas), precisamos usar uma função para dizer ao programa que os valores serão combinados em um único vetor.

```
x <- c(2,3,4)  
x
```

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

```
y <- seq(1,10)  
y
```

```
[1] 1 2 3 4 5 6 7 8 9 10
```

```
z <- rep(1,10)
z
```

```
[1] 1 1 1 1 1 1 1 1 1 1
```

```
a <- 1:10
a
```

```
[1] 1 2 3 4 5 6 7 8 9 10
```

```
bicho <-c("macaco","pato","galinha","porco")
bicho
```

```
[1] "macaco" "pato" "galinha" "porco"
```

E se quisermos visualizar o conteúdo da posição 2 no vetor bicho?

```
bicho[2]
```

```
[1] "pato"
```

As operações vetoriais podem ser realizadas de maneira bastante intuitiva. Como exemplos:

```
x <- c(2,3,4)
x
```

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

```
ops <- x-1
ops
```

```
[1] 1 2 3
```



```
k <- x*2  
k
```

```
[1] 4 6 8
```

Vamos agora considerar um vetor de pesos em kg e altura em metros de 6 pessoas.

```
peso <- c(62, 70, 52, 98, 90, 70)  
peso
```

```
[1] 62 70 52 98 90 70
```

```
altura <- c(1.70, 1.82, 1.75, 1.94, 1.84, 1.61)  
altura
```

```
[1] 1.70 1.82 1.75 1.94 1.84 1.61
```

Vale mencionar que o separador de decimais no R é . (ponto)!

Como calcularia o IMC? Lembrando que o IMC é dado pelo peso (em kg) dividido pela altura (em metros) ao quadrado.

```
imc <- peso/(altura^2)  
imc
```

```
[1] 21.45329 21.13271 16.97959 26.03890 26.58318 27.00513
```

Para saber o tamanho do vetor, use a função `length()`.

```
length(imc)
```

```
[1] 6
```

5.5 Matrizes

Matrizes são vetores numéricos com duas dimensões, que são simplesmente a linha e a coluna às quais o elemento pertence.

```
x <- matrix(seq(1,16), nrow=4,ncol=4)
x
```

```
      [,1] [,2] [,3] [,4]
[1,]     1     5     9    13
[2,]     2     6    10    14
[3,]     3     7    11    15
[4,]     4     8    12    16
```

Note que os números de 1 a 16 foram dispostos na matriz coluna por coluna ou seja, preenchendo de cima para baixo e depois da esquerda para a direita.

Como sei qual elemento está na segunda linha e terceira coluna da matriz x?

```
x[2,3]
```

```
[1] 10
```

```
x[3, ] # seleciona a 3ª linha
```

```
[1] 3 7 11 15
```

```
x[, 2] # seleciona a 2ª coluna
```

```
[1] 5 6 7 8
```

```
x[1, 2] # seleciona o elemento da primeira linha e segunda coluna
```

```
[1] 5
```

E se eu quiser substituir a primeira linha por (13,15,19,30)?

```
x[1,] <- c(13,15,19,30)
```

```
x
```

	[,1]	[,2]	[,3]	[,4]
[1,]	13	15	19	30
[2,]	2	6	10	14
[3,]	3	7	11	15
[4,]	4	8	12	16

Seja o vetor d:

```
d <- c(128,124,213,234)
```

E se quisermos substituir a terceira coluna por d?

```
x[,3] <- d
```

Qual a dimensão da matriz x?

Vimos que para vetor usamos o comando `length()`. Serve para matriz também? Vamos testar!

```
length(x)
```

```
[1] 16
```

Note que retorna o número de colunas vezes o número de linhas ($4*4=16$). Mas o que quero saber é o numero de linhas e de colunas. Para isso, o comando é `dim()`.

```
dim(x)
```

```
[1] 4 4
```

Para concatenar linhas em uma matriz, podemos usar o comando `rbind()`:

```
vet <- c(2,20,12,34)
x2 <- rbind(x,vet)
x2
```

	[,1]	[,2]	[,3]	[,4]
	13	15	128	30
	2	6	124	14
	3	7	213	15
	4	8	234	16
vet	2	20	12	34

Para concatenar colunas em uma matriz, podemos usar o comando `cbind()`:

```
v2 <- c(25,10,15,4)
x3 <- cbind(x,v2)
x3
```

```
              v2
[1,] 13 15 128 30 25
[2,]  2  6 124 14 10
[3,]  3  7 213 15 15
[4,]  4  8 234 16  4
```

5.6 Fatores

Fatores podem ser vistos como vetores de inteiros que possuem rótulos (labels). Eles são úteis para representar uma variável categórica (nominal e ordinal).

```
sexo <- c("M", "H", "H", "H", "M", "M", "H")
sex <- as.factor(sexo)
sex
```

```
[1] M H H H M M H
Levels: H M
```

```
levels(sex)
```

```
[1] "H" "M"
```

5.7 Data frame

Trata-se de uma “tabela de dados” onde as colunas são as variáveis e as linhas são os registros. Essas colunas podem ser de classes diferentes.

Essa é a grande diferença entre `data.frame`'s e matrizes (matriz é só numérica).

Posso criar um data frame no R com os vetores, por exemplo:

```
ID <- seq(1,6)
pes <- c(62, 70, 52, 98, 90, 70)
alt <- c(1.70, 1.82, 1.75, 1.94, 1.84, 1.61)
imc <- pes/(alt^2)
dados <- data.frame(ID=ID,peso=pes,altura=alt, imc=imc)
dados
```

	ID	peso	altura	imc
1	1	62	1.70	21.45329
2	2	70	1.82	21.13271
3	3	52	1.75	16.97959
4	4	98	1.94	26.03890
5	5	90	1.84	26.58318
6	6	70	1.61	27.00513

Posso pensar que o data frame tem a mesma ideia de matriz. Quero olhar os dados de altura, que sei que está na coluna 3.

```
dados[,3]
```

```
[1] 1.70 1.82 1.75 1.94 1.84 1.61
```

Mas existe uma maneira mais fácil de selecionar a variável de interesse sem ter que saber em qual coluna ela está.

Por ser um data frame, posso usar \$ da seguinte maneira:

```
dados$altura
```

```
[1] 1.70 1.82 1.75 1.94 1.84 1.61
```

Putz, esqueci de colocar a variável de grupo no data frame. Tenho que criar tudo de novo? Não:

```
gr <- c(rep(1,3),rep(2,3))
dados$grupo <- gr
```

```
dados
```

	ID	peso	altura	imc	grupo
1	1	62	1.70	21.45329	1
2	2	70	1.82	21.13271	1
3	3	52	1.75	16.97959	1
4	4	98	1.94	26.03890	2
5	5	90	1.84	26.58318	2
6	6	70	1.61	27.00513	2

Veja que no “dados\$grupo” foi inserido o objeto “gr”. Se “gr” não tivesse o mesmo número de linhas do data frame retornaria um erro.

Funções úteis para data.frame:

Ainda não falamos com muito detalhes sobre funções no R, faremos isso mais adiante. Mas por enquanto, considere que sejam nomes já salvos no R e que, ao colocar o objeto da base de dados (no nosso exemplo é `dados`) dentro dos parênteses, retorna algumas informações úteis sobre a base de dados. São algumas delas:

- `head()` - Mostra as primeiras 6 linhas.
- `tail()` - Mostra as últimas 6 linhas.
- `dim()` - Número de linhas e de colunas.
- `names()` - Os nomes das colunas (variáveis).
- `str()` - Estrutura do data.frame. Mostra, entre outras coisas, as classes de cada coluna.

```
head(dados)
```

	ID	peso	altura	imc	grupo
1	1	62	1.70	21.45329	1
2	2	70	1.82	21.13271	1
3	3	52	1.75	16.97959	1
4	4	98	1.94	26.03890	2
5	5	90	1.84	26.58318	2
6	6	70	1.61	27.00513	2

```
dim(dados)
```

```
[1] 6 5
```

```
names(dados)
```

```
[1] "ID"      "peso"    "altura"  "imc"     "grupo"
```

```
str(dados)
```

```
'data.frame':  6 obs. of  5 variables:
 $ ID      : int  1 2 3 4 5 6
 $ peso   : num  62 70 52 98 90 70
 $ altura: num  1.7 1.82 1.75 1.94 1.84 1.61
 $ imc    : num  21.5 21.1 17 26 26.6 ...
 $ grupo  : num  1 1 1 2 2 2
```

5.8 Operadores lógicos

A operação lógica nada mais é do que um teste que retorna verdadeiro (**TRUE**) ou falso (**FALSE**). Esses dois valores recebem uma classe especial: **logical**.

- Igual a: **==**

Vamos testar se um valor é igual ao outro.

Exemplo:

```
10==11
```

```
[1] FALSE
```

```
11==11
```

```
[1] TRUE
```

No primeiro retornou **FALSE**, pois realmente 10 não é igual a 11 e no segundo caso acima retornou **TRUE**, pois realmente 11 é igual a 11.

De maneira análoga funciona para os operadores abaixo:

- Diferente de: **!=**

Exemplo:

```
10!=11
```

```
[1] TRUE
```

- Maior que: >
- Maior ou igual: >=
- Menor que: <
- Menor ou igual: <=

Exemplos:

```
10>5
```

```
[1] TRUE
```

```
10>=10
```

```
[1] TRUE
```

```
4<4
```

```
[1] FALSE
```

```
4<=4
```

```
[1] TRUE
```

- Um outro operador muito útil é o `%in%`. Com ele, podemos verificar se um valor está dentro de um vetor.

```
ex <- 1:15  
3 %in% ex
```

```
[1] TRUE
```

- E: `&` - será verdadeiro se os dois forem TRUE.

```
x <- 15  
x > 10 & x < 30
```

```
[1] TRUE
```



```
x < 10 & x < 30
```

```
[1] FALSE
```

- OU: | - será verdadeiro se um dos dois forem TRUE.

```
x <- 15  
x > 10 | x < 30
```

```
[1] TRUE
```

```
x < 10 | x < 30
```

```
[1] TRUE
```

- Negação: !

```
x <- 15  
!x < 30
```

```
[1] FALSE
```

5.9 Dados faltantes, infinitos e indefinições matemáticas

- NA (Not Available): dado faltante/indisponível. Exemplo:

```
x <- c(1,6,9)  
x[4]
```

```
[1] NA
```

Retornou NA porque não há elemento na posição 4 do vetor x.

- NaN (Not a Number): indefinições matemáticas. Como 0/0 e log(-1). Exemplo:

```
log(-10)
```

[1] NaN

- Inf (Infinito): número muito grande ou o limite matemático. Aceita sinal negativo (-Inf).
Exemplo:

```
10^14321
```

[1] Inf

5.10 Condicionamento: If e else

As estruturas *if* e *else* servem para executar um código apenas se uma condição (teste lógico) for satisfeita.

```
a <- 224
b <- 225
if (a==b) {
  v <- 10
} else {
  v <- 15
}
v
```

[1] 15

Veja que o R só executa o conteúdo das chaves {} se a expressão dentro dos parênteses () retornar TRUE.

Note que a condição de igualdade é representada por dois iguais (==). Como dito anteriormente, apenas um igual (=) é símbolo de atribuição (preferível <-).

Veja outro exemplo:

```
a <- 224
b <- 225
if (a==b) {
  v <- 10
} else if (a > b) {
  v <- 15
} else {
  v <- 25
}
```

```
}  
v
```

```
[1] 25
```

Veja que nesse exemplo gostaria de usar mais de duas condições, e por isso usamos a estrutura intermediária `else if`.

5.11 Iterador for

O *for* serve para repetir uma mesma tarefa para um conjunto de valores diferentes. Cada repetição é chamada de iteração.

Como exemplo, considere o vetor atribuído ao objeto `m` como segue:

```
m <- c(1,20,50,60,100)
```

Quero criar um novo vetor, `p` digamos, que seja formado por cada elemento de `m` dividido por sua posição.

```
p <- NULL  
for (i in 1: length(m)){  
  p[i] <- m[i]/i  
}  
p
```

```
[1] 1.00000 10.00000 16.66667 15.00000 20.00000
```

Note que primeiro definimos o objeto `p`, recebendo `NULL`. O `NULL` representa a ausência de um objeto e serve para já declarar algum objeto que receberá valor na sequência. No caso, ao rodar o `for`, o `p` é um vetor de tamanho 5 (tamanho do vetor `m`).

No exemplo, temos 5 iterações e para cada valor de `i`, correndo de 1 até 5 (tamanho de `m`), pegamos o valor de `m` na posição `i` e dividimos por sua posição. Assim, formamos o vetor `p`.

5.12 Funções

Funções no R são nomes que guardam um código de R. A ideia é que sempre que rodar a função com os seus argumentos, o código que ela guarda será executado e o resultado será retornado.

Já usamos anteriormente algumas funções que estão na base do R. Por exemplo, quando usamos `class()` para entender a classe do objeto que o R está entendendo. Colocamos um argumento dentro do parênteses e o R retornou qual a classe do objeto em questão. Relembre o que falamos ao perguntar ao R qual a classe do vetor `oi` criado:

```
oi <- c(10,20,2,1,0.5)
class(oi)
```

```
[1] "numeric"
```

Agora vamos conversar sobre outra função já criada e disponibilizada na base do R: `mean`. Essa função retorna a média do vetor que está em seu argumento. Vamos calcular a média dos valores do vetor `oi`:

```
mean(oi)
```

```
[1] 6.7
```

Considere que, por algum motivo, tenha no vetor `oi` uma observação faltante. No R, dado faltante é caracterizado por `NA`.

```
oi <- c(10,20,2,1,0.5,NA)
```

Perceba que, apesar de `NA` ser um texto, não coloquei entre aspas porque quero falar para o R que naquela posição não tem valor e o R entende isso ao ler `NA` (sem aspas). Se colocar entre aspas, ele entenderá como sendo um texto e não mais como valor faltante.

```
mean(oi)
```

```
[1] NA
```

Como não sabemos o valor do elemento na posição 6 do vetor `oi`, o R não teria como calcular a média de todos os 6 valores e por isso devolve `NA`. No entanto, queremos calcular a média dos elementos de `oi` ao retirar os valores faltantes, ou seja, queremos fazer: $(10+20+2+1+0.5)/5$.

Então devemos falar para o R o que queremos, e para isso podemos utilizar o argumento `na.rm = TRUE`:

```
mean(oi, na.rm = TRUE)
```

```
[1] 6.7
```

Importantes:

- 1) Se a função tiver mais de um argumento, eles são sempre separados por vírgulas;
- 2) Cada função tem os seus próprios argumentos. Para saber quais são e como usar os argumentos de uma função, basta acessar a sua documentação. Uma forma de fazer isso é pela função `help`, cujo argumento é o nome da função que precisa de ajuda:

```
help(mean)
```

Veja que abrirá a documentação sobre a função `mean` no menu “Help” do RStudio, e lá é possível ver os argumentos e exemplos de uso da função em questão.

Ainda sobre funções já presentes no R, vamos considerar agora a função `sample`. Veja a documentação dessa função para ver o que ela faz:

```
help(sample)
```

A função `sample` retorna uma amostra de um vetor com tamanho especificado em um de seus argumentos com ou sem reposição. Ela apresenta quatro argumentos: `sample(x, size, replace = FALSE, prob = NULL)`, em que: `x` é o vetor do qual será amostrado o número de elementos especificado no argumento `size`, seja com ou sem reposição (argumento `replace`) e com dadas probabilidades de seleção, especificadas em `prob`.

Quero usar essa função para amostrar do objeto `oi` (`x=oi`) dois elementos (`size=2`) em uma seleção com reposição (`replace = TRUE`) e que a probabilidade de seleção seja a mesma para todos os elementos do vetor `oi`. No caso da probabilidade, como podemos ver na documentação da função `sample`, o *default* (padrão se o usuário não mudar o argumento) é ser a mesma probabilidade de seleção para todos os elementos. Assim, se o usuário nada especificar para esse argumento, o R entenderá o seu *default*. O mesmo vale para o argumento `replace`: caso fosse o interesse fazer a seleção sem reposição, não precisaríamos colocar esse argumento porque seu *default* é `FALSE`.

```
sample(x=oi, size=2, replace=TRUE) #não colocamos argumento prob porque vamos usar o seu def
```

```
[1] 10 1
```

Também poderíamos usar a mesma função sem colocar o nome dos argumentos:

```
sample(oi,2,TRUE)
```

```
[1] 0.5 20.0
```

Mas, nesse caso, é importante que se respeite a ordem dos argumentos: o vetor tem que ser o primeiro, o segundo argumento é **size** e assim por diante.

Vale ressaltar que as duas últimas saídas não necessariamente serão as mesmas, porque é feito um sorteio aleatório de dois elementos de **oi** em cada uma delas.

Além de usar funções já prontas, podemos criar novas funções. Suponha que queremos criar uma função de dois argumentos que retorna o primeiro mais três vezes o segundo argumento. Criamos a função no que segue:

```
f_conta <- function(x,y) {  
  out <- x+3*y  
  return(out)  
}
```

A função acima tem:

- o nome: `f_conta`;
- os argumentos: `x` e `y`;
- o corpo `out`: `<- x+3*y`; e
- o que retorna: `return(out)`.

Suponha que eu queira fazer a conta $10+3*20$. Podemos fazer isso ao chamar a função criada `f_conta`.

```
f_conta(x=10,y=20)
```

```
[1] 70
```

Veja que o cálculo acima retorna exatamente o mesmo que o seguinte:

```
f_conta(y=20,x=10)
```

```
[1] 70
```

Isso acontece porque mudei a ordem dos argumentos, mas acompanhado com os nomes dos argumentos. Se eu não quiser colocar os nomes dos argumentos, precisa tomar cuidado para não errar a ordem deles. Pois:

```
f_conta(10,20)
```

```
[1] 70
```

é diferente de

```
f_conta(20,10)
```

```
[1] 50
```

5.13 Como obter ajuda no R

Listamos aqui 3 maneiras para buscar ajuda no R:

- Help/documentação do R (comandos `help(nome_da_funcao)` ou `?nome_da_funcao`). Como exemplo,

```
help(mean) #ou  
?mean
```

- Google Na Figura @ref(fig:help) está o exemplo de busca de ajuda no Google. Repare no ‘r’ no início da busca, isso pode ajudar.
- Comunidade O [Stack Overflow](#) e o [Stack Overflow em Português](#) são sites de Pergunta e Resposta amplamente utilizados por todas as linguagens de programação, e o R é uma delas.

5.14 Pacotes

Como dito quando falamos “Sobre o R”, o R apresenta funções na sua base e também em forma de pacotes (conjunto de funções bem documentado), que precisam ser instalados (uma vez no seu computador) e carregados na sessão de utilização do R (carregado em toda sessão aberta).

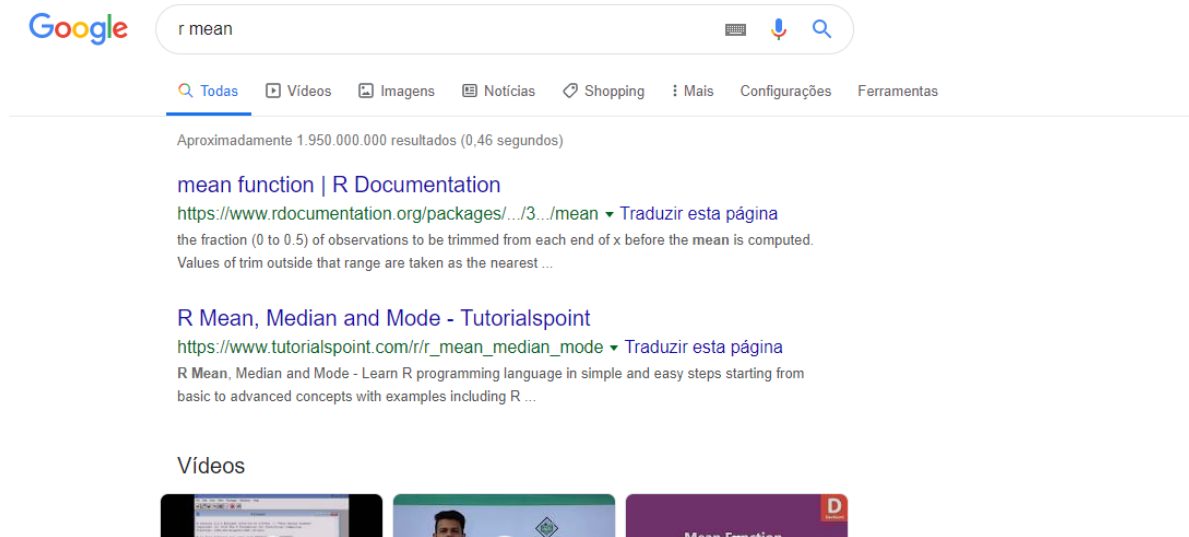


Figure 5.1: Pesquisa no Google

Difícilmente você vai fazer uma análise apenas com as funções básicas do R e dificilmente não vai existir um pacote com as funções que você precisa. Por esse motivo, falamos a seguir em como instalar e carregar pacotes.

5.14.1 Instalação de pacotes

- **Via CRAN:**

```
install.packages("nome-do-pacote")
```

Exemplo: Instalação do pacote `dplyr`.

```
install.packages("dplyr")
```

Note que o nome do pacote está entre aspas.

- **Via Github:** Para instalar via Github, precisa primeiramente instalar o pacote `devtools`.

```
devtools::install_github("nome-do-repo/nome-do-pacote")
```

Exemplo:


```
devtools::install_github("tidyverse/dplyr")
```

5.14.2 Carregar pacotes

Uma vez que um pacote de interesse está instalado em sua máquina, para carregá-lo na sessão atual do R é só rodar a seguinte linha de comando:

```
library(nome-do-pacote)
```

Veja que para carregar o pacote não se usa aspas.

Como exemplo, o carregamento do pacote `dplyr`:

```
library(dplyr)
```

Só é necessário instalar o pacote uma vez, mas é necessário carregá-lo toda vez que começar uma nova sessão.

Dado que o pacote está carregado ao rodar a função `library()`, todas as funções desse pacote podem ser usadas sem problemas.

Caso você não queira carregar o pacote e apenas usar uma função específica do pacote, você pode usar `nome-do-pacote::nome-da-funcao`. Por exemplo:

```
dplyr::distinct(...)
```

Se você tivesse carregado o pacote `dplyr` anteriormente (pela função `library()`), não seria necessário colocar `dplyr::` antes da função `distinct` do pacote.

5.15 Materiais complementares

- Critical Thinking in Clinical Research. Felipe Fregni & Ben M. W. Illigens. 2018.
- Sites:
 - <https://www.bmj.com/about-bmj/resources-readers/publications/statistics-square-one/1-data-display-and-summary>
 - <http://www.sthda.com/english/wiki/statistical-tests-and-assumptions>
- CHAPTER 3: Selecting the Study Population. In: Critical Thinking in Clinical Research by Felipe Fregni and Ben Illigens. Oxford University Press 2018.

- Fandino W. Formulating a good research question: Pearls and pitfalls. *Indian J Anaesth.* 2019;63(8):611–616. doi:10.4103/ija.IJA_198_19
- Riva JJ, Malik KM, Burnie SJ, Endicott AR, Busse JW. What is your research question? An introduction to the PICOT format for clinicians. *J Can Chiropr Assoc.* 2012;56(3):167–171.
- External validity, generalizability, and knowledge utilization. Ferguson L1. *J Nurs Scholarsh.* 2004;36(1):16-22.
- Peter M Rothwell; Commentary: External validity of results of randomized trials: disentangling a complex concept, *International Journal of Epidemiology*, Volume 39, Issue 1, 1 February 2010, Pages 94–96, <https://doi.org/10.1093/ije/dyp305>

Referências