

Uso de R para análise dos dados da World Values Survey

Tiago Vier, UFRGS Daniel Capistrano, UCD Jéssica Duarte, UFRGS
Luana Isabelle Beal, UFRGS Carla Mendonça, ISCTE
Cristiana Verônica Mueller, UNISC

Coletivo WVSR, última atualização em 28/07/2020

Sumário

Bem-Vindo(a)!	5
1 Introdução	7
1.1 R e RStudio	7
1.2 World Values Survey (WVS)	7
2 Instalando o R e o RStudio	9
2.1 LINUX	9
2.2 Mac OS X	10
2.3 Windows	11
2.4 Identificando o RStudio	12
2.5 Criando um projeto no RStudio	13
3 Bibliotecas	15
3.1 Instalando bibliotecas adicionais	15
3.2 Carregando bibliotecas	16
4 Importando bases de dados	17
4.1 Com o uso das ferramentas da aba “Environment”	17
4.2 Com o uso de linhas de comando	18
4.3 Gerando um dicionário de variáveis	19
5 Descrevendo e analisando os dados	21
5.1 Descrevendo os dados	21
5.2 Estatísticas descritivas	23
5.3 Análises bi e multivariadas	25
6 Visualizar os dados	27
7 Glossário	31

Bem-Vindo(a)!

Este tutorial é resultado de um trabalho conjunto do Coletivo de Análises da World Values Survey (WVS), ou Pesquisa Mundial de Valores, em Português, sobre o uso da linguagem R e ambiente de programação RStudio.

Trata-se de um documento evolutivo e introdutório ao uso dos softwares livres R e RStudio nas análises dos dados da base estruturada da WVS.

Este tutorial foi preparado em Rmarkdown [2020] com a biblioteca bookdown [2020].

Um agradecimento aos demais colaboradores do grupo do WVS Brasil.

Capítulo 1

Introdução

1.1 R e RStudio

R é uma linguagem de programação com diversas características: tem como foco uma programação funcional, flexível, dinâmica e, principalmente, direcionada à manipulação, análise e visualização de dados.

É muito útil, por exemplo, para estudos de bases de dados complexas e pesquisas estatísticas. Essa funcionalidade também pode ser chamada de mineração de dados: o processo de exploração de dados em grande escala com o objetivo de buscar padrões, associações, relacionamentos e sistematização de variáveis.

O R e o RStudio são softwares livres e são compatíveis com os sistemas operacionais Windows, Linux e macOS, além de possuir boa integração com outros programas, como planilhas (por exemplo, Microsoft Excel) e outros softwares de análise de dados (por exemplo, SPSS).

O que justifica seu uso nas Ciências Sociais? Existe um uso crescente de métodos quantitativos nas Ciências Sociais acompanhado de uma maior disponibilidade de dados quantitativos. No entanto, cientistas sociais enfrentam diversas dificuldades de acesso às ferramentas tradicionais de análise de dados que são caras e oferecem pouco suporte para o aprendizado de estudantes e jovens pesquisadores. Além de contribuir para democratizar o uso de métodos quantitativos em Ciências Sociais, o R também tem o potencial de impulsionar a colaboração entre cientistas sociais e a reprodutibilidade de estudos na área.

1.2 World Values Survey (WVS)

A World Values Survey (WVS) investiga mudanças culturais em diversas sociedades e possibilita a comparação das características dessas mudanças, contribuindo, entre outros campos, para o debate sobre a relação entre desenvolvimento socio-econômico e mudanças culturais, para o acompanhamento longitudinal das mudanças em curso e para a ampliação do conhecimento de diferentes áreas do planeta – muitas delas eram antes de acesso limitado a determinadas regiões.



O embasamento teórico do WVS está associada à teoria da modernização e do pós-materialismo elaborada por Ronald Inglehart, que sugere que fenômenos como o crescimento do setor de serviços, a melhoria na

qualidade de vida e o aumento das oportunidades educacionais nas sociedades industriais avançadas ou pós-industriais têm levado a uma gradual transformação nos valores e atitudes sociais.

A pesquisa teve início em 1981 e consiste em inquéritos representativos em nível nacional realizados em quase cem países, abrangendo quase 90% da população mundial. Utiliza um questionário comum de 180 variáveis e inclui atualmente entrevistas com quase 400 mil indivíduos.

A tese articula duas hipóteses para explicar essa mudança:

- a) a *hipótese da escassez*: defende que as prioridades da ação humana são resultado do ambiente sócio-econômico vigente, no qual valoriza-se subjetivamente coisas e aspectos da realidade que são escassos; e
- b) *hipótese da socialização*: defende que grande parte dos valores básicos de um indivíduo derivam das condições presentes em seu período de formação, anterior à idade adulta.

Dessa forma, o WVS se desenvolve a partir de um referencial teórico específico mas possibilita uma ampla gama de análises. A pesquisa utiliza como hipótese central que *“as mudanças nos sistemas de crenças de massas têm consequências sociais, políticas e econômicas importantes. Mas, ao mesmo tempo, [...] esta pesquisa proporciona outras análises a partir de seus resultados, haja vista a qualidade e a diversidade das dimensões e perguntas presentes no questionário.”* [de Castro et al., 2015]



O Brasil participou de cinco das sete ondas da WVS realizadas até agora (1990-1994; 1995-1999; 2005-2009; 2010-2014; e 2017-2020). O professor Henrique Castro coordena a pesquisa no país desde a quinta onda que foi baseada na Universidade de Brasília, entre 2005 e 2014, e a partir da Universidade Federal do Rio Grande do Sul (UFRGS), desde 2017, com financiamento da Capes/MEC.

O WVS no Brasil utiliza uma amostra probabilística completa em três etapas: a) seleção aleatória e estabelecimento de 150 clusters (baseada nos setores censitários do país, incluindo zonas urbanas e rurais); b) seleção aleatória de habitações em cada setor censitário; e c) escolha do entrevistado na habitação, considerando que seja alfabetizado, tenha 16 anos ou mais de idade e data de aniversário mais próxima do dia da primeira visita bem-sucedida.

Atualmente, o grupo de pesquisa WVS Brasil é integrado por discentes e docentes do Programa de Pós-Graduação em Ciência Política e discentes do Departamento de Economia e Relações Internacionais, ambos da UFRGS, e por pesquisadores de diversas universidades do Brasil, como Universidade de Brasília, Unisinos e Universidade de Caxias do Sul, e do Exterior, como a University College Dublin, integrando uma equipe de cerca de 60 pesquisadores.

Os dados gerados por todos os países integrantes da WVS ficam disponíveis para livre acesso na internet, por meio da página www.worldvaluessurvey.org.

Capítulo 2

Instalando o R e o RStudio

Para trabalhar com R, é necessário instalar dois softwares: R-base e RStudio. Para utilizar o RStudio, é necessário primeiro instalar o R-base.

Siga os passos para instalação de acordo com o sistema operacional do seu computador.

2.1 LINUX

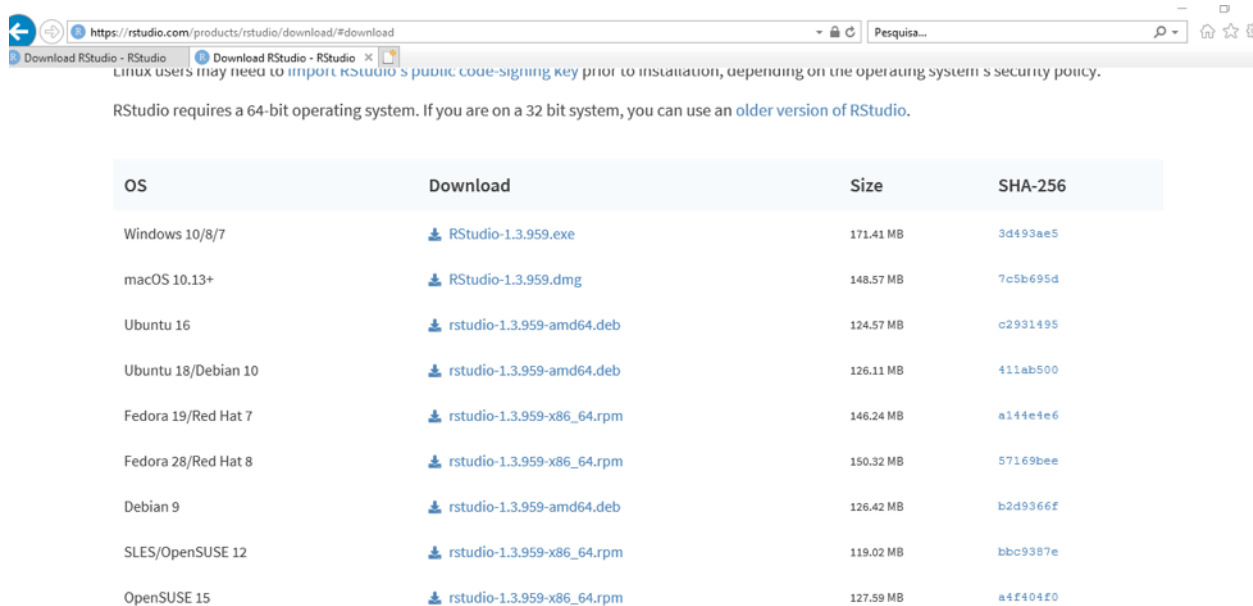
2.1.1 Instalar o R

- Passo 1: Abra o terminal. Se utilizar distribuição Fedora, pressione as teclas Super + T, e no Ubuntu Ctrl + Alt + t;
- Passo 2: Com o terminal aberto digite a seguinte linha de comando:
 - Fedora: `sudo dnf install R`
 - Ubuntu: `sudo apt-get install r-base r-base-core`
- Passo 3: Pressione a tecla Enter para confirmar;
- Passo 4: Digite a senha do usuário;
- Passo 5: Clique em Confirmar. Pronto! O R estará instalado e pode ser acessado.

Link para eventual consulta: <http://cran-r.c3sl.ufpr.br/bin/linux/>

2.1.2 Instalar o RStudio

- Passo 1: Acesse o site <https://rstudio.com/products/rstudio/download/>
- Passo 2: Encontre na página o local de download gratuito conforme figura abaixo:



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

RStudio requires a 64-bit operating system. If you are on a 32 bit system, you can use an [older version of RStudio](#).

OS	Download	Size	SHA-256
Windows 10/8/7	RStudio-1.3.959.exe	171.41 MB	3d493ae5
macOS 10.13+	RStudio-1.3.959.dmg	148.57 MB	7c5b695d
Ubuntu 16	rstudio-1.3.959-amd64.deb	124.57 MB	c2931495
Ubuntu 18/Debian 10	rstudio-1.3.959-amd64.deb	126.11 MB	411ab500
Fedora 19/Red Hat 7	rstudio-1.3.959-x86_64.rpm	146.24 MB	a144e4e6
Fedora 28/Red Hat 8	rstudio-1.3.959-x86_64.rpm	150.32 MB	57169bee
Debian 9	rstudio-1.3.959-amd64.deb	126.42 MB	b2d9366f
SLES/OpenSUSE 12	rstudio-1.3.959-x86_64.rpm	119.02 MB	bbc9387e
OpenSUSE 15	rstudio-1.3.959-x86_64.rpm	127.59 MB	a4f404f0

- Passo 3: Encontre o sistema operacional do seu computador (Ubuntu, Fedora, Debian ou OpenSUSE) e faça download.
- Passo 4: Acesse o terminal na pasta onde foi feito o download e siga as instruções abaixo usando Fedora ou Ubuntu:
 - Fedora: `sudo dnf install nomedo_arquivo_baixado.rpm`; Ex: `sudo dnf install rstudio-1.3.959-x86_64.rpm`
 - Ubuntu: `sudo dpkg -i nomedo_arquivo_baixado.deb`
- Passo 5: O RStudio está instalado e pronto para uso!

2.2 Mac OS X

2.2.1 Instalar o R

- Passo 1: Abra o site CRAN - <https://cran.r-project.org/>
- Passo 2: Clique em “Download de R for (Mac) OS X”.
- Passo 3: Clique duas vezes no arquivo depois de baixado. Pronto! O R está instalado.

2.2.2 Instalar o RStudio

- Passo 1: Acesse o site <https://rstudio.com/products/rstudio/download/>
- Passo 2: Encontre na página o local de download gratuito conforme figura abaixo:

OS	Download	Size	SHA-256
Windows 10/8/7	RStudio-1.3.959.exe	171.41 MB	3d493ae5
macOS 10.13+	RStudio-1.3.959.dmg	148.57 MB	7c5b695d
Ubuntu 16	rstudio-1.3.959-amd64.deb	124.57 MB	c2931495
Ubuntu 18/Debian 10	rstudio-1.3.959-amd64.deb	126.11 MB	411ab500
Fedora 19/Red Hat 7	rstudio-1.3.959-x86_64.rpm	146.24 MB	a144e4e6
Fedora 28/Red Hat 8	rstudio-1.3.959-x86_64.rpm	150.32 MB	57169bee
Debian 9	rstudio-1.3.959-amd64.deb	126.42 MB	b2d9366f
SLES/OpenSUSE 12	rstudio-1.3.959-x86_64.rpm	119.02 MB	bbc9387e
OpenSUSE 15	rstudio-1.3.959-x86_64.rpm	127.59 MB	a4f404f0

- Passo 3: Encontre o sistema operacional do seu computador (Mac OS) e faça download.
- Passo 4: Depois de baixado, clique duas vezes no arquivo para instalá-lo. O RStudio está instalado e pronto para uso!

2.3 Windows

2.3.1 Instalar o R

- Passo 1: Clique no seguinte link <https://cran.r-project.org/bin/windows/base/>
- Passo 2: Clique em Download R for Windows (os números que aparecem nesse arquivo de download correspondem à versão do R disponível):

R-4.0.2 for Windows (32/64 bit)

[Download R 4.0.2 for Windows](#) (84 megabytes, 32/64 bit)

[Installation and other instructions](#)

[New features in this version](#)

If you want to double-check that the package you have downloaded matches the package distributed by CRAN, you can compare the [md5sum](#) of the .exe to the [fingerprint](#) on the master server. You will need a version of md5sum for windows: both [graphical](#) and [command line versions](#) are available.

Frequently asked questions

- [Does R run under my version of Windows?](#)
- [How do I update packages in my previous version of R?](#)
- [Should I run 32-bit or 64-bit R?](#)

Please see the [R FAQ](#) for general information about R and the [R Windows FAQ](#) for Windows-specific information.

Other builds

- Patches to this release are incorporated in the [r-patched snapshot build](#).
- A build of the development version (which will eventually become the next major release of R) is available in the [r-devel snapshot build](#).
- [Previous releases](#)

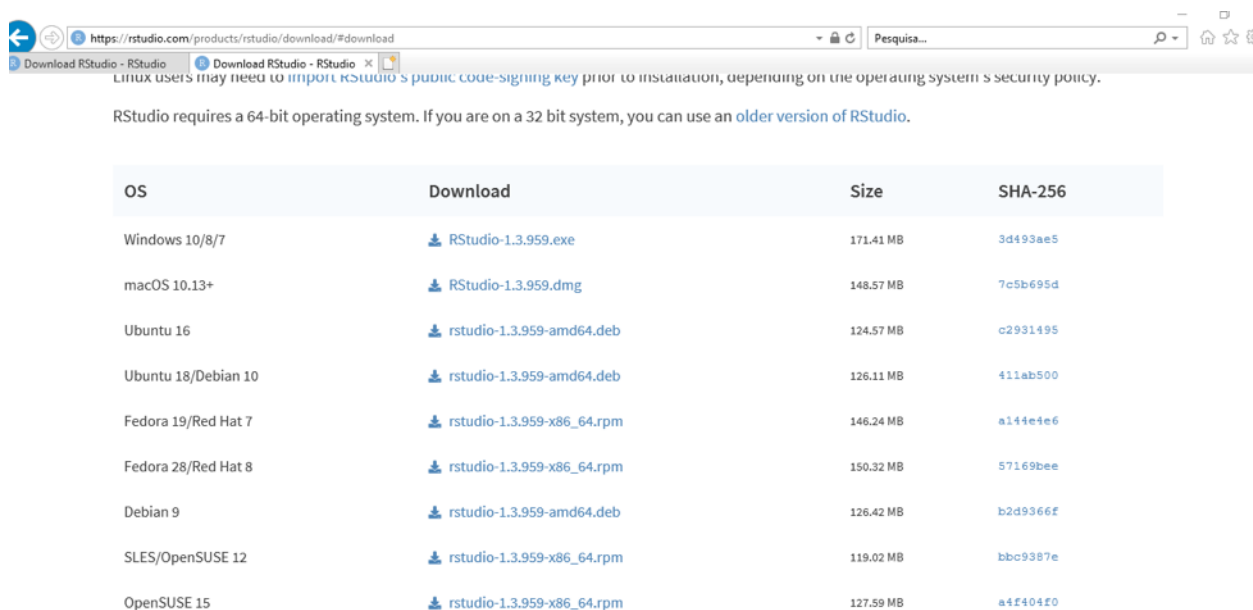
Note to webmasters: A stable link which will redirect to the current Windows binary release is CRAN.MIRROR>/bin/windows/base/release.html.

Last change: 2020-06-22

- Passo 3: Clique duas vezes no arquivo depois de baixado. Clique em Avançar até finalizar a instalação. Pronto! O R está instalado.

2.3.2 Instalar o RStudio

- Passo 1: Acesse o site <https://rstudio.com/products/rstudio/download/>
- Passo 2: Encontre na página o local de download gratuito conforme figura abaixo:



OS	Download	Size	SHA-256
Windows 10/8/7	RStudio-1.3.959.exe	171.41 MB	3d493ae5
macOS 10.13+	RStudio-1.3.959.dmg	148.57 MB	7c5b695d
Ubuntu 16	rstudio-1.3.959-amd64.deb	124.57 MB	c2931495
Ubuntu 18/Debian 10	rstudio-1.3.959-amd64.deb	126.11 MB	411ab500
Fedora 19/Red Hat 7	rstudio-1.3.959-x86_64.rpm	146.24 MB	a144e4e6
Fedora 28/Red Hat 8	rstudio-1.3.959-x86_64.rpm	150.32 MB	57169bee
Debian 9	rstudio-1.3.959-amd64.deb	126.42 MB	b2d9366f
SLES/OpenSUSE 12	rstudio-1.3.959-x86_64.rpm	119.02 MB	bbc9387e
OpenSUSE 15	rstudio-1.3.959-x86_64.rpm	127.59 MB	a4f404f0

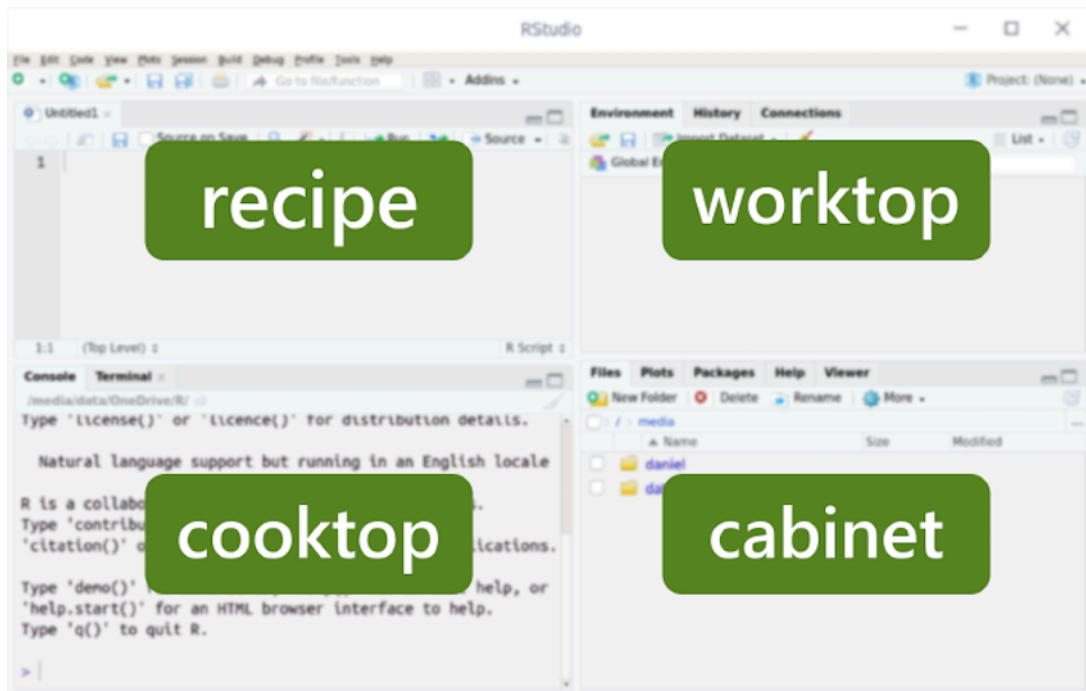
- Passo 3: Encontre o sistema operacional do seu computador (Windows) e faça download.
- Passo 4: Depois de baixado, clique duas vezes no arquivo para instalá-lo. O RStudio está instalado e pronto para uso!

2.4 Identificando o RStudio

O RStudio tem quatro interfaces, espaços distribuídos na sua tela, fáceis de identificar e de trabalhar.

São elas:


- Source: local onde são inseridos os códigos-fonte. Fica no canto superior esquerdo.
- Console: também chamado de terminal, onde são mostrados os resultados dos comandos executados pelos códigos na Source. Fica no canto inferior esquerdo.
- Environment: também chamado de ambiente, onde são mostrados os elementos (por exemplo, vetores, bases de dados etc) que foram criados. Fica no canto superior direito.
- Viewer: também chamado files, onde são mostrados os arquivos, pacotes, pastas, entre outros. Fica no canto inferior direito.



2.5 Criando um projeto no RStudio

Uma forma prática de iniciar o trabalho com o RStudio é criando projetos.

Para iniciar um novo projeto:

- Acesse o menu “File > New Project”, ou
- então o ícone  na parte superior direita do RStudio.

O RStudio vai perguntar se você quer criar um novo diretório para o projeto, usar um diretório existente, ou ainda clonar um repositório distante (GitHub, por exemplo).

Para criar um novo diretório:

- Clique em “New Directory”, escolha entre as opções de tipos de projeto oferecidas e dê um nome ao seu novo diretório.

Ali, você poderá salvar todos os documentos do projeto.

Ao fechar o RStudio, todos os documentos e os objetos do ambiente serão salvos no projeto, e você pode retomar o trabalho de onde parou quando quiser.

Lembre-se: as bibliotecas devem ser recarregadas cada vez que você iniciar uma sessão do R.

Capítulo 3

Bibliotecas

Quando instalamos o R, instalamos junto uma grande quantidade de comandos e funções. A lista de funções do R-base pode ser vista aqui.

Para além desses recursos, o R dispõe de uma grande quantidade de bibliotecas (ou pacotes). No momento em que escrevemos este tutorial, a plataforma de contribuições CRAN lista 16,045 bibliotecas disponíveis. As diferentes contribuições são agrupadas por tópicos. É possível, por exemplo, conhecer os pacotes disponíveis para as Ciências Sociais.

Neste tutorial, utilizamos as funções do R-base e sugerimos bibliotecas adicionais que facilitam algumas operações e são suficientes para a maior parte das necessidades de um pesquisador interessado em analisar os dados do WVS.

As bibliotecas que são usadas neste tutorial fazem parte do grupo de pacotes tidyverse [2019]. Entre elas, estão os pacotes dplyr, haven e ggplot2. Mais detalhes sobre esse tema podem ser obtidos no livro acessível neste link. Além do tidyverse, usamos a biblioteca codebook para ter acesso ao metadados disponíveis nos arquivos de SPSS e Stata.

3.1 Instalando bibliotecas adicionais

Como mencionamos anteriormente, quando instalamos o R-base já instalamos uma série de funções e comandos.

Entretanto, você pode usar um comando específico para instalar novas bibliotecas do repositório CRAN.

Para instalar bibliotecas, use o comando: `install.packages("nome_da_biblioteca")`.

Outra maneira de instalar novas bibliotecas é usar a aba “Packages”.

Essa aba permite visualizar as bibliotecas já instaladas e instalar novas usando o botão “Install”.

Para instalar a biblioteca tidyverse digite:

```
# para instalar as bibliotecas da tidyverse
install.packages("tidyverse")
```

Para instalar a biblioteca codebook de visualização de metadados (ver no capítulo seguinte) digite:

```
# para instalar a biblioteca codebook para visualizar os metadados
# (ver no capítulo seguinte)
install.packages("codebook")
```

Quando rodamos o comando, usando **Ctrl + Enter** ou **Run** no script, ou **Enter** se você está digitando diretamente no console, o R vai verificar se existem dependências e instalar todas as que forem necessárias para o seu funcionamento.

É importante que se instale todas as dependências indicadas pelo R. Verifique no Console e, se o R perguntar se você quer fazer a instalação das dependências, selecione “Yes”.

3.2 Carregando bibliotecas

É preciso carregar as bibliotecas sempre que se inicia uma nova sessão.

O comando para carregar as bibliotecas é `library("nome_da_biblioteca")`.

Para carregar as bibliotecas que vamos usar, use os comandos abaixo:

```
# carregar o tidyverse  
library(tidyverse)  
  
# carregar a biblioteca haven,  
# usada para importar o arquivo no formato do SPSS  
library(haven)  
  
# carregar o codebook  
library(codebook)
```

Sempre que abrir o RStudio, é preciso recarregar as bibliotecas, pois o RStudio reinicia a sessão somente com os comandos do R-base.

É recomendado que as bibliotecas que serão usadas nas análises sejam indicadas no início do código R onde são feitas as análises. Dessa forma, qualquer pesquisador poderá identificar os requisitos necessários para reproduzir as suas análises.

Uma vez carregadas as bibliotecas, nós podemos continuar para o próximo passo que é importar os dados que vamos analisar.

Capítulo 4

Importando bases de dados

Utilizando o RStudio, pode-se abrir arquivos salvos no formato do próprio R (.RData) e de variados outros softwares, como Excel(.xlsx), SPSS (.sav) e Stata (.dta), e em diversos outros formatos, incluindo texto e csv.

Há duas formas de abrir arquivos de dados no RStudio:

- Com o uso do painel “Environment”, ou
- Com o uso de linhas de comando escritas diretamente no Console ou em um script

4.1 Com o uso das ferramentas da aba “Environment”

Por essa forma, você usa as ferramentas do R do mesmo jeito que faz com qualquer outro software, como os do pacote Microsoft Office, por exemplo.

4.1.1 Arquivos em formato R

Para arquivos formato R (formato .RData):

- Na aba “Environment”, clique sobre o ícone “Abrir Arquivos”.
- Procure o seu arquivo nos locais do seu computador.
- Clique no arquivo e clique em Open ou clique duas vezes sobre o arquivo.

Pronto! O seu arquivo vai aparecer no Environment.

4.1.2 Arquivos em outros formatos

Para importar os dados em outros formatos, é preciso ter bibliotecas específicas para esse fim. Neste tutorial, sugerimos que você tenha instalada e carregada a biblioteca haven, da gramática Tidyverse (veja em bibliotecas).

- Clique sobre o ícone Import Dataset.
- Escolha e clique sobre o formato em que o seu arquivo foi previamente salvo.
- Clique em Browse.
- Procure o seu arquivo nos locais do seu computador.
- Clique no arquivo e clique em Open ou clique duas vezes sobre o arquivo.

Pronto! O seu arquivo vai aparecer no Environment.

4.2 Com o uso de linhas de comando

4.2.1 Arquivos em formato R

Para carregar arquivos no formato nativo do R usando linhas de comando:

- Abra um novo script R e digite: `load("nome_do_arquivo.rdata")`.
 - Dica: Deixe o cursor entre as aspas e tecle TAB. O R vai lhe mostrar os diretórios do seu computador. Se seu arquivo está em alguma pasta ou subpasta o R também vai lhe mostrar esses locais.
- Clique no arquivo que você quer importar
- Digite Control + Enter, ou clique em Run (disponível na aba superior do Source Code do seu R).

Pronto! O seu arquivo vai aparecer no Environment.

4.2.2 Arquivos em outros formatos

Para importar os dados em outros formatos, é preciso ter bibliotecas específicas para esse fim. Neste tutorial, sugerimos que você tenha instalada e carregada a biblioteca **haven** da gramática Tidyverse (veja em bibliotecas).

Cada tipo de arquivo tem um comando específico na biblioteca **haven**:

Arquivos SPSS: função `read_sav()`

Arquivos Excel: função `read_csv()`

Arquivos Stata: função `read_dta()`

Arquivos SAS: função `read_sas()`

Para que a base se transforme em um objeto no Environment do R, é preciso que se atribua um nome a ele. Veja este exemplo com a extensão do SPSS (arquivos em formato `.sav`):

```
# carregar a biblioteca haven
library(haven)

# abrir usando a função e criar um objeto no ambiente R
base_do_wvs <- read_sav("diretório_do_projeto/nome_do_arquivo.sav")
```

Como dito na dica anteriormente, deixe o cursor entre as aspas e tecle TAB. O R vai lhe mostrar os diretórios do seu computador. Em seguida, é só clicar sobre o arquivo que você procura.

- Tecle Control + Enter, ou
- Clique em Run (disponível na aba superior do Source Code do seu R).

Pronto! O seu arquivo vai aparecer no Environment.

Com outras extensões você usará, por exemplo, `read_csv()` ou `read_dta()`. Para mais informações acesse a ajuda da biblioteca **haven**

4.3 Gerando um dicionário de variáveis

Para navegar na base de dados recém importada e reconhecer as variáveis importadas e suas características, existem diversas ferramentas disponíveis.

Uma delas é o pacote `codebook`, que gera um dicionário de variáveis. Nos exemplos abaixo, mostramos como gerar um dicionário de variáveis estático na aba “Preview” e também na forma de um objeto que pode ser visualizado como uma tabela.

Para carregar a biblioteca `codebook` e gerar um dicionário de variáveis estático navegável na aba “Preview”, gerar uma tabela e visualisá-la use os comandos abaixo:

```
# carregar a biblioteca codebook
library(codebook)

# gerar um dicionário de variáveis estático navegável na aba "Preview"
label_browser_static(base_do_wvs)

# gerar uma tabela com todas as variáveis
dicionario <- codebook_table(base_do_wvs)

# visualizar na forma de tabela junto aos scripts R
view(dicionario)
```

Feita a importação da base de dados e com o dicionário em mãos, podemos passar a descrição, análise e visualização dos dados.

Capítulo 5

Descrevendo e analisando os dados

Após instalar R e RStudio, reconhecer seu ambiente de trabalho, instalar e carregar as bibliotecas e importar os bancos de dados, começa a verdadeira diversão: finalmente, você vai trabalhar com os dados!

O R permite que se faça desde as operações mais simples, como uma calculadora, passando pela organização e limpeza dos dados, até análises estatísticas mais avançadas.

Um excelente começo dessa empreitada é a descrição dos dados que não só garante o divertimento do pesquisador, como também permite que se façam excelentes análises e conclusões.

A partir da estatística descritiva, obtém-se um conjunto de ferramentas que permitem:

- a) organizar os dados - agrupando, selecionando, filtrando, ordenando, criando variáveis;
- b) descrever e analisar os dados - resumindo-os com base em sua média, moda, mediana, cruzando e contando as variáveis, criando gráficos e tabelas.

A principal função da estatística descritiva é resumir os dados e facilitar a assimilação da informação [Agresti et al., 2012].

Além da descrição dos dados, o R também oferece suporte para que sejam feitos testes estatísticos mais avançados bi e multivariados, entre eles, correlação e regressão linear múltipla que iremos discutir mais adiante.

Nesta seção, você pode aprender os comandos do R que permitem criar anotações, criar variáveis e vetores, conectar sentenças do código de programação, visualizar os valores máximo e mínimo de uma variável, obter a média, somar os valores de uma variável, filtrar, selecionar, ordenar, contar, cruzar, criar nova coluna, resumir os dados a partir de alguma operação, excluir valores da análise, visualizar variáveis, fazer gráficos, fazer testes de correlação e análises de regressão multivariada.

Isso é tudo o que se precisa para, em ordem de importância: ter adoráveis momentos com o seu banco de dados e construir conhecimento a partir de análises robustas.

5.1 Descrevendo os dados

5.1.1 Criar variáveis e vetores

Para criar uma variável, escolha um nome para ela e atribua através do sinal `<-` um valor para a mesma, lembrando sempre de que o R não aceita caracteres do tipo espaço, então se o nome de sua variável tiver mais de um termo é necessário usar o conector underline (`_`) entre as palavras. Lembre-se também que o R faz a diferença entre letras maiúsculas e minúsculas.

Para criar vetores, segue-se a mesma lógica, mas os valores devem ser inseridos entre os parênteses do código: `c()`, o qual indica que há um conjunto de valores (esses valores devem ser separados por vírgulas

e entre aspas no caso de o valor ser um caractere, ou *string*). Os valores atribuídos podem ser números simples, podem ser operações como soma, subtração, multiplicação.

Após criados, variáveis e vetores, podem também ter seus valores submetidos a algum tipo de operação. Veja estes exemplos:

```
# Criar um vetor chamado "Meu_resultado" que é o resultado da operação 1 + 5
Meu_resultado <- 1 + 5

# Criar um vetor chamado "Brasil" e multiplicá-lo por 3
Brasil <- c(1, 3, 10, 12, 19, 21, 28, 30, 37)
Brasil * 3

# Criar um vetor chamado Países com strings/textos
Países <- c("Argentina", "Brasil")

# Cálculo da idade com base no ano de nascimento
Ano_Nasc <- c(1990, 1984, 1961)

# Calcular as idades
2020 - Ano_Nasc

# Armazenar como um vetor na aba Environment
Idade <- 2020 - Ano_Nasc
```

5.1.2 Funções básicas e comandos úteis

Após carregar seu banco de dados ou criar suas próprias variáveis, é possível começar a conhecer e operar os dados observando seus valores máximos, mínimos, média e efetuando operações a partir dos valores dentro da variável. Para executar tais funções, alguns comandos do R-base para a variável “Idade”, por exemplo, são os seguintes:

- Valor máximo: `max(Idade)`
- Valor mínimo: `min(Idade)`
- Média: `mean(Idade)`
- Soma: `sum(Idade)`

Na gramática do tidyverse, para executar mais de um comando você pode inserir entre eles o conector `%>%` (que pode ser lido como “e então”, em uma sequência de operações)

Além disso, pode ser útil acrescentar anotações que não sejam lidas pelo R como uma linha de código. Para tanto basta inserir no início da frase o caractere cerquilha, ou jogo da velha (`#`).

Veja o exemplo:

```
# Criar anotações: #
# Conectar as linhas do código: %>%

# Exemplos:
# Abrir a base de dados do WVS, e então sumarizar pela média,
# e então efetuar a contagem das variáveis S003 e X001
# Variáveis do WVS usadas: X003 = Idade; S003 = País; X001 = Sexo
base_do_wvs %>%
  summarise(media = mean(X003, na.rm = TRUE)) %>%
  count(S003, X001)
```

5.2 Estatísticas descritivas

5.2.1 Filtrar e selecionar

Partindo para funções mais sofisticadas, é possível organizar o banco de dados de maneira sistemática e fazer análises resumidas das variáveis. O comando `filter()` filtra as observações (linhas) baseadas em uma condição determinada pelo pesquisador, como, por exemplo, selecionar determinados países, gênero feminino ou masculino, faixas de renda, idade ou escolaridade.

```
# Neste exemplo, filtrar observações baseadas na condição  
# País (S003) igual a Brasil (código 76).  
# O resultado dessa operação é salvo como uma nova base de dados chamada "base_brasil".  
  
base_brasil <-  
  base_do_wvs %>%  
  filter(S003 == 76)
```

Com o comando `select()` é possível escolher as variáveis específicas (colunas) com que se deseja trabalhar.

```
# Selecionar somente variáveis país e sexo e salvar como uma nova base:  
  
base_duas_colunas <-  
  base_do_wvs %>%  
  select(S003, X001)
```

5.2.2 Criar uma variável

O comando `mutate()` cria uma nova variável (coluna) ou modifica uma já existente a partir de manipulações dos dados. No exemplo abaixo, nós criamos uma variável de idade a partir da variável de ano de nascimento:

```
# Criar uma variável com a idade do respondente a partir do ano de nascimento (X002):  
  
base_do_wvs <-  
  base_do_wvs %>%  
  mutate(idade = 2020 - X002)
```

A função `if_else()` divide os valores em dois grupos: os que se encaixam na condição e os que não se encaixam. No exemplo a seguir, utilizamos esse comando para criar uma nova variável. Uma variável que assume dois valores: os que têm idade inferior a 30 ($X003 < 30$) são classificados como “Jovem”, os que não possuem são classificados como “Não Jovem”.

```
# Criar uma variável de faixas etárias.  
# O if_else() precisa de três instruções:  
# if_else("condição", "valor se sim", "valor se não")  
  
base_do_wvs <-  
  base_do_wvs %>%  
  mutate(faixa_etaria = if_else(idade < 30, "Jovem", "Não Jovem"))
```

5.2.3 Contar

A função `count()` permite que se faça análises dos dados ao apresentar sua contagem (frequência) e cruzar variáveis.

```
#Neste exemplo contamos o número de observações de cada país:
base_do_wvs %>%
  count (S003)

# Aqui podemos combinar o comando filter (filtrar somente para o Brasil) e o
# comando count (contar o numero de observacoes para cada país)
base_do_wvs %>%
  filter(S003 == 76) %>%
  count (S003)
```

5.2.4 Sumarizar

Outra forma de manipular e avaliar os dados é utilizando o comando `summarise()` para resumir os dados em função de alguma operação (média/mean e mediana/median). Usando o critério `na.rm = TRUE`, em alguns comandos, as não-respostas (*missing values*) são removidas do cálculo.

```
# summarise(), exemplo resumir os dados pela média e mediana de idade
base_do_wvs %>%
  summarise(media = mean(X003))

# para excluir missing values: na.rm = TRUE
base_do_wvs %>%
  summarise(media = mean(X003, na.rm = TRUE),
            mediana = median(X003, na.rm = TRUE))
```

5.2.5 Agrupar

Para mostrar os resultados agrupados por uma variável específica, utilizamos o comando `group_by`. Nesse exemplo abaixo, fazemos a mesma média do exemplo anterior para cada um dos países:

```
# Mesma operação de média de idade do exemplo anterior
# Só que adicionamos uma nova linha (linha 2) para agrupar os resultados por país
# O resultado apresenta a média de idade para cada um dos países

base_do_wvs %>%
  group_by(S003) %>%
  summarise(media = mean(X003, na.rm = TRUE))
```

5.2.6 Fator de ponderação (peso)

Na maioria das pesquisas de tipo *survey* existe uma variável que devemos utilizar para que os resultados das nossas operações sejam mais precisos. Nessas pesquisas, a amostra planejada nem sempre consegue ser executada perfeitamente, ou seja, nem todas as pessoas selecionadas na amostra são entrevistadas.

Para corrigir os possíveis vieses de participação decorrente disso, o WVS possui um fator de ponderação (peso/*weight*) que deve ser utilizado quando estamos produzindo estatísticas descritivas. Para isso, nós vamos incluir uma opção de peso (*wt / w*) assinalando a variável de peso do WVS: S018.

```
# Nesse exemplo contamos o número de observações de cada país usando o peso (S018)
base_do_wvs %>%
  count(S003, wt = S018)

# Nesse exemplo temos a média de idade por país usando o peso (S018)
# Note que o comando também é diferente "weighted.mean" em vez de "mean"
```



```
base_do_wvs %>%
  group_by(S003) %>%
  summarise(media = weighted.mean(X003, w = S018, na.rm = TRUE))
```

5.3 Análises bi e multivariadas

As análises descritivas oferecem diversas possibilidades para conhecer comportamentos, distribuição e padrões das diferentes variáveis. Contudo, quando o objetivo é estabelecer relação entre as mesmas é necessário adotar testes mais avançados bi e multivariados como correlação e regressão linear múltipla.

A correlação tem por função avaliar o comportamento de duas variáveis em função uma da outra, mensurando a interdependência das mesmas e o seu grau de associação. Logo, a correlação é apropriada se o seu propósito é analisar se há relação entre a variação de duas dimensões e qual a força dessa relação, como, por exemplo, nas clássicas premissas: nível educacional e renda variam de maneira diretamente proporcional ou apoio à democracia varia de maneira inversamente proporcional a valores autoritários?

A regressão linear multivariada permite estabelecer valores estimativos de uma variável dependente em relação a um conjunto de outras variáveis independentes. Resulta no nível de previsibilidade de uma variável em função de outras variáveis explicativas. Esse tipo de teste se aplica aos casos em que se almeja avaliar e estabelecer potencial previsibilidade de uma variável em função de outras, e não apenas comparar seus comportamentos, por exemplo, prevendo a taxa de desconfiança em relação aos partidos com base na quantidade de pessoas que estão satisfeitas com a democracia.

5.3.1 Correlação

Para fazer o teste de correlação basta selecionar o par de variáveis em questão e adicionar o comando `cor()` utilizando todas as observações, como no exemplo abaixo:

```
# Selecionar duas variáveis e extrair a correlação
base_do_wvs %>%
  select(X002, X003) %>%
  cor(use = "complete.obs")
```

5.3.2 Regressão linear múltipla

Para efetuar o teste de regressão linear múltipla basta atribuir a uma variável resultado `<-` a função `lm()`.

Veja o exemplo:

```
lm(variavel_resposta ~ variável_independente_1 + variável_independente_2 + variavel_independente_n
data = .)
```

É necessário sinalizar qual é o banco de dados utilizado antes para o programa pelo nome do objeto e o recorte através das funções `group_by()` ou `filter()`. Nos exemplos a seguir é possível visualizar melhor a estrutura do comando:

Veja estes exemplos de regressão linear. No Exemplo 1 abaixo, criamos um modelo de regressão usando como variável resposta a renda (X047), idade (X002) e educação (X025R) como variáveis explicativas. Neste exemplo o “” indica que a base de dados utilizada na função `lm()` é a que está referenciada antes do “e então” (`%>%`) na primeira linha.

```
# Exemplos de regressão linear

# Exemplo 1:

modelo_wvs <- base_do_wvs %>%
  lm(X047 ~ X025R + X002, data = .)
```

No Exemplo 2 abaixo, o modelo é gerado a partir do filtro feito na linha 2. Ou seja, neste caso, o modelo utiliza somente os dados do país (S003) de código 68 (Bolívia).

Exemplo 2:

```
modelo_bolivia <- base_do_wvs %>%  
  filter(S003 == 68) %>%  
  lm(X047 ~ X025R + X002, data = .)
```

Para verificar os coeficientes e estatísticas dos modelos de regressão gerados, basta utilizar a função `summary()`:

Resultados do Exemplo 1:

```
summary(modelo_wvs)
```

Resultados do Exemplo 2:

```
summary(modelo_bolivia)
```

Capítulo 6

Visualizar os dados

Para visualizar os dados, usamos a biblioteca `ggplot2()` também da gramática Tidyverse.

Após organizar e manipular os dados e fazer análises descritivas, é uma boa estratégia apresentar as informações em gráficos para evidenciar padrões, tendências e comparações. Conforme visto anteriormente, a biblioteca para plotar gráficos é a `ggplot()`. A biblioteca `ggthemes()` permite aplicar alguns temas já pré-estabelecidos.

O `ggplot()` é uma biblioteca que constrói os gráficos em camadas. Primeiramente, define-se a “folha” com uma base de dados que será usada; em seguida, a geometria e os demais parâmetros dos gráficos.

O comando básico do `ggplot()` é: `ggplot(data = dados, aes(x = Explicativa, y = Resposta)) + geoma + ...`

Tipos de geometrias disponíveis:

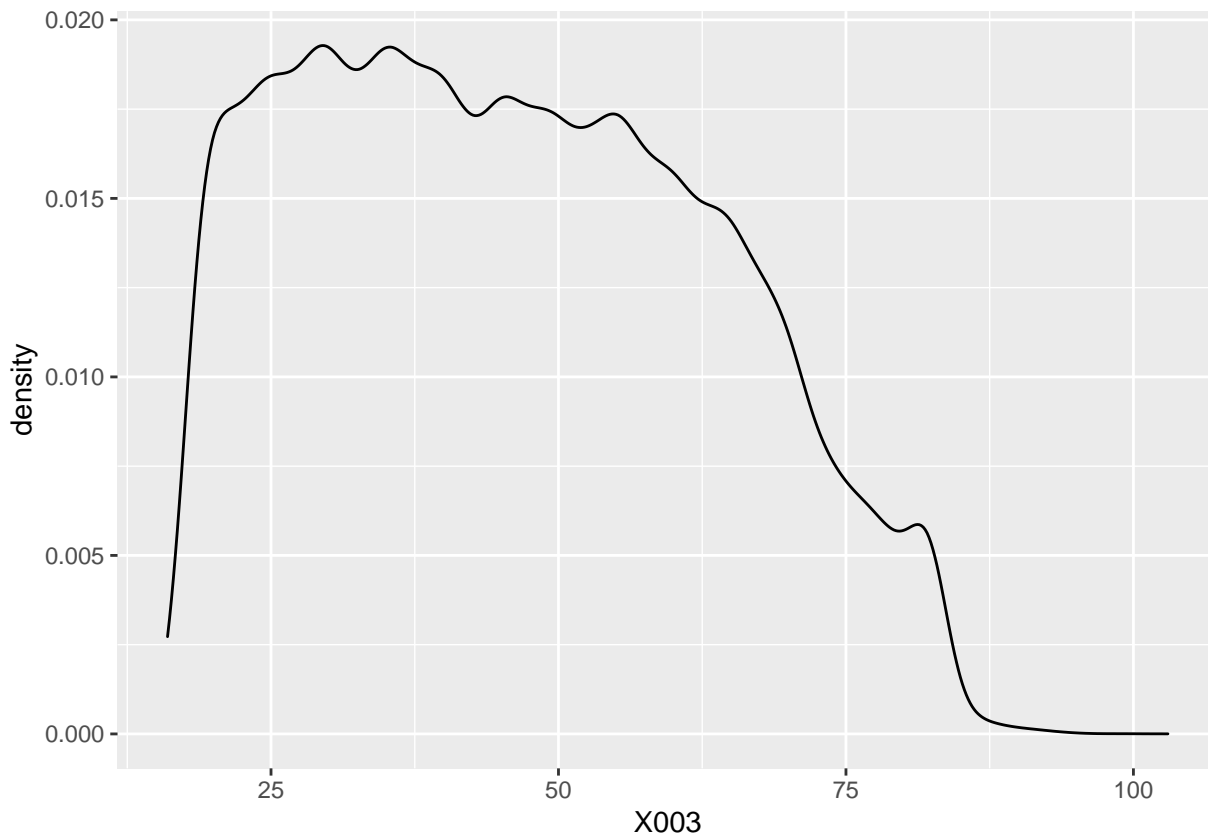
- Histograma: `geom_hist()`
- Barras: `geom_bar()`
- Pontos: `geom_point()`
- Diagrama de caixa (boxplot): `geom_boxplot()`
- Linhas: `geom_line()`

O guia do `ggplot()` pode ser encontrado aqui. Nesta galeria, podem ser encontrados diversos exemplos de gráficos para se inspirar.

Veja uns exemplos abaixo:

Exemplo 1: plotar um gráfico de densidade da variável idade:

```
# plotar um gráfico de densidade da variável idade  
ggplot(df_wvs7, aes(x = X003)) + geom_density()
```



Exemplo 2: aqui, reproduzimos o mapa cultural de Inglehart e Welzel [2005] para alguns países. Antes, é preciso fazer algumas manipulações com os dados com o que aprendemos nos capítulos anteriores.

```
# Criar uma lista com os países que se quer analisar
# 76=Brasil, 32=Chile, 152=Argentina, 170=Colombia
países <- c(76, 32, 152, 170)

# Extrair e retrabalhar as variáveis que precisamos
# select() para selecionar as variáveis de interesse
# filter() para filtrar usando o vetor 'países' criado acima
# group_by() para agrupar por onda-país, onda e país
# summarise() para agregar os dados pela média
df_y001 <- df_wvs7 %>%
  select(S024, S020, S003, tradrat5, survself) %>%
  filter(S003 %in% países) %>%
  group_by(S024, S020, S003) %>%
  summarise(tradrat5 = mean(tradrat5, na.rm = T),
            survself = mean(survself, na.rm = T))
```

Feito isso podemos criar a figura:

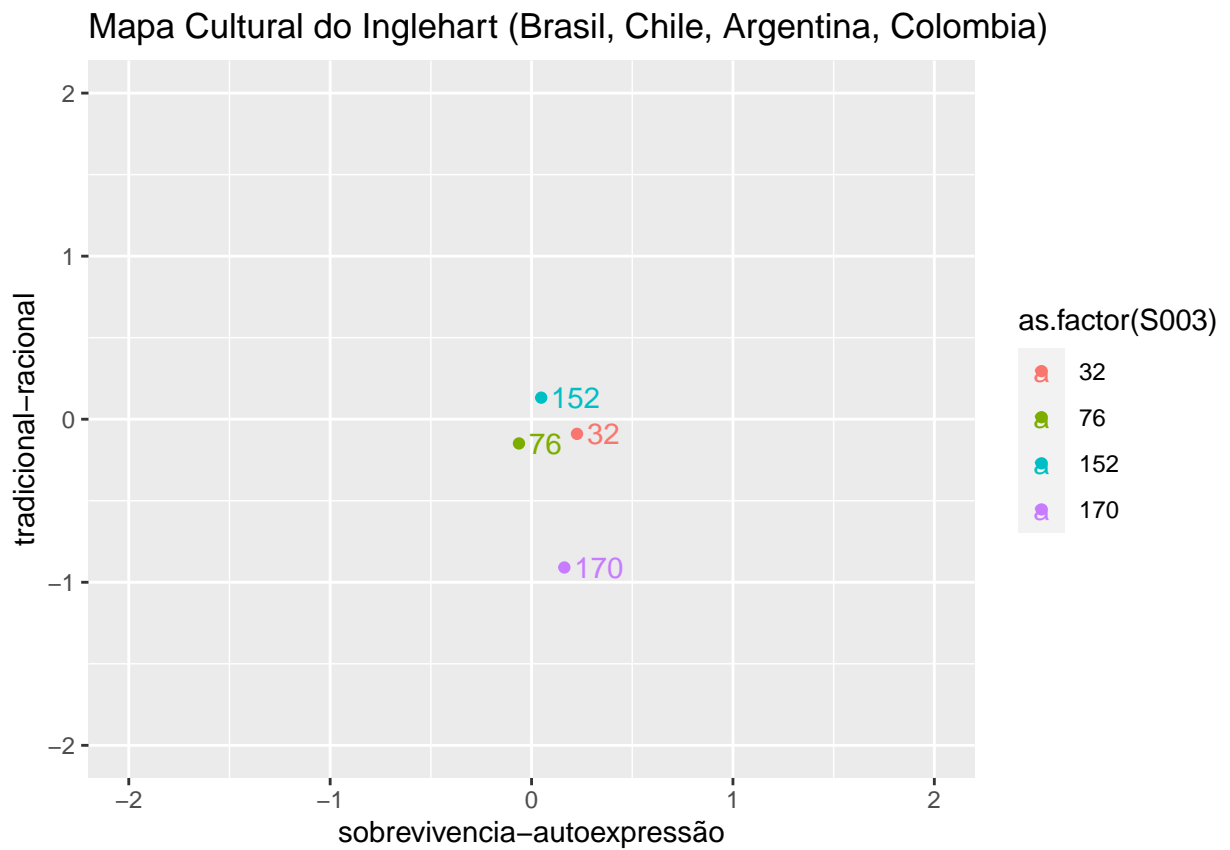
Lembre-se que o ggplot usa o símbolo + para adicionar cada um dos elementos da figura.

```
# lembre se de carregar a variável ggplot se ela não tiver sido carregada
library(ggplot2)

# Reproduzir o Mapa Cultural Inglehart

ggplot(df_y001, aes(x = survself,
                    y = tradrat5,
                    color = as.factor(S003),
                    label = as.factor(S003))) +
```

```
# definir os dados e as dimensões
geom_point() + # geometria pontos
geom_text(hjust = 0, nudge_x = 0.05, check_overlap = TRUE) + # rótulos dos pontos
scale_x_continuous(limits = c(-2, 2)) + # restringe eixo x e y entre -2 e 2
scale_y_continuous(limits = c(-2, 2)) +
labs(x = "sobrevivencia-autoexpressão",
     y = "tradicional-racional",
     title = "Mapa Cultural do Inglehart (Brasil, Chile, Argentina, Colombia)") # rótulos do gráfico
```



Capítulo 7

Glossário

- Programação: alguns conceitos relevantes para aprender uma linguagem de programação.
 - Variável: representa um local de armazenamento na memória, com um nome e um valor, mas que não é o valor em si, porque esse pode ser constantemente alterado. Cada variável possui diferentes propriedades, por exemplo, pode ser numérica (*integral* or *double*) ou uma cadeia de caracteres (*string*). Inserindo uma metáfora: cada variável é uma caixinha que pode ter diferentes características (grande, pequena, quadrada, redonda) e que guarda alguma coisa. `x <- 2` #cria variável "x" e atribuir valor 2
 - Objeto: cada objeto é único, específico, e pode ser referenciado. Seu conteúdo pode ser atribuído diretamente ou ser o resultado de uma operação. Cada objeto pode ser armazenado em uma variável. Na metáfora, um objeto pode ser uma carta, uma moeda, um lápis-de-cor. `y <- 5` #criar objeto com nome y e atribuir valor 5
 - Vetor: uma sequência ordenada de elementos. Importante notar que um vetor armazena apenas um tipo de informação, por exemplo, só números, o que se chama de unidimensional. Como se fosse uma caixinha com várias cartas (uma de fulano, outra de beltrano etc) ou uma caixinha com várias moedas (1 euro, 50 centavos de dólar, 25 centavos de real etc.), uma caixinha com diferentes lápis-de-cor (rosa, azul, verde), ou ainda uma caixinha com moedas e lápis-de-cor. `z <- c(1, 2, 3)` #criar vetor z e atribuir os valores 1, 2 e 3
 - Lista: é como um vetor avançado, pois também é uma sequência de elementos, mas pode guardar diferentes tipos de dados (números, letras etc), ou seja, é multidimensional. Assim, pode ser uma caixinha que guarda tudo junto: cartas, moedas e lápis-de-cor. `lista <- list(idade = 20, "oi", 35, "tchau")` #criar "lista" e atribuir diferentes conteúdos
- Interfaces: o RStudio possui quatro interfaces, que são as “pequenas telas” ou “janelinhas” que aparecem ao iniciar o programa. Pensando em uma cozinha, o Source seria a receita, o passo-a-passo; o Console seria o fogão, onde tudo está acontecendo, onde se forma o resultado; e o Environment seria uma mesa, onde estão os ingredientes disponíveis. Os pacotes e arquivos ficam no armário, o Viewer, com os utensílios de cozinha e ingredientes disponíveis.
- Funções: diferentes “verbos”, tanto do R-base quanto na gramática tidyverse. Realizam diferentes tipos de funções.
 - Count: conta quantas vezes aparece aquela variável, por exemplo, numa base de dados, e mostra seu valor. `count(x)` #contar a quantidade da variável x
 - Filter: filtrar observações (“cadastros”) baseadas em uma condição. `filter(x == 1)` #filtra todos os valores de x e deixar apenas os cadastros em que x tem o valor 1
 - Select: selecionar variáveis - no plural -, ou seja, escolher mais de uma variável. `select(x, y, z)` #selecionar as variáveis x, y e z
 - Group_by: agrupar variáveis, por exemplo, em uma base de dados grande, “juntar” a partir de um determinado quesito. `group_by(país)` #juntar os dados com base nos diferentes valores da variável país
 - View: visualiza a operação realizada, normalmente, o R cria uma tabela com o(s) conteúdo(s), ficando mais “simples” de enxergar os dados. `view(objeto)`

- Summarise: resumir, sintetizar, os dados baseados em uma operação, ou seja, criando uma espécie de atalho. `summarize(media = mean(idade))` #tem a média das idades das pessoas como critério para resumir ou sintetizar os dados
- Mutate: criar nova informação com base em outras. Visualmente, é como se criasse uma nova “coluna”. `view(a = b + c)` #a ‘coluna’ a vai ter o valor da soma de b e c
- Códigos-chave:
 - # para fazer comentários, ou seja, escrever anotações sem que elas sejam executadas como códigos.
 - %>% para criar um comando no sentido de “e então”, ou seja, especificar que seja realizado o próximo passo do código em seguida. Dica: clicar **Ctrl + Shift + M**.
 - **Ctrl + enter**, ou clicar no botão **Run**, para “rodar”, ou seja, executar o código programado/escrito.
- Estatística:
 - Variáveis numéricas: quantitativas, podem ser discretas (contagem, ex.: número de filhos) ou contínuas (mensuração, ex.: peso, altura)
 - Variáveis categóricas: qualitativas, podem ser ordinais (quando há ordem, por exemplo, grau de instrução) ou nominais (classificação, por exemplo, sexo, raça)

Referências Bibliográficas

Alan Agresti, Barbara Finlay, and Lori Viali. *Métodos Estatísticos para as Ciências Sociais*. Penso, Porto Alegre, edição: 4 edition, February 2012. ISBN 978-85-63899-57-6.

JJ Allaire, Yihui Xie, Jonathan McPherson, Javier Luraschi, Kevin Ushey, Aron Atkins, Hadley Wickham, Joe Cheng, Winston Chang, and Richard Iannone. Rmarkdown: Dynamic documents for r, 2020.

Henrique C. de O. de Castro, Sonia Ranincheski, and Daniel Capistrano. O conteúdo da globalização para os latino-americanos: Uma análise a partir da Pesquisa Mundial de Valores–WVS. 2015.

Ronald Inglehart and Christian Welzel. *Modernization, Cultural Change, and Democracy: The Human Development Sequence*. Cambridge University Press, Cambridge, UK ; New York, August 2005. ISBN 978-0-521-60971-5.

Hadley Wickham. Tidyverse: Easily install and load the 'tidyverse', 2019.

Yihui Xie. Bookdown: Authoring books and technical documents with r markdown, 2020.