R NA PESQUISA ACADÊMICA

Jakson Alves de Aquino

Introdução

O R é um software livre de análise de dados quantitativos. Como todo software livre, é gratuito e tem seu código-fonte disponibilizado para ser inspecionado por curiosos ou colaboradores voluntários. R é também o nome da linguagem de programação executada pelo software R.E o que é uma linguagem de programação? É a linguagem usada para escrever um programa de computador, ou seja, uma sequência de comandos que, geralmente, tem o objetivo de executar tarefas repetitivas e longas.

Uma das dificuldades que existem para as pessoas da área das Ciências Sociais utilizarem o R é a não familiaridade com a lógica de programação. E o meu principal objetivo neste capítulo é ajudar aos que ainda não usam o R a decidirem se compensa investir tempo para aprender a usá-lo.[1] Não é meu propósito ensinar detalhadamente a linguagem R,[2] mas espero que, ao final do texto, os leitores tenham uma boa noção das vantagens e eventuais desvantagens de se usar o software R na pesquisa acadêmica. O público-alvo deste capítulo são leitores que já trabalham ou que planejam realizar pesquisas com metodologia quantitativa e estão em busca de um software para possibilitar essa tarefa.

[1] Sou grato a Luana Dias Motta – professora da UFSCar e integrante do Comitê de Mídia e Programação da Sociedade Brasileira de Sociologia – por ter me convidado para ministrar a Aula Aberta *A pesquisa com R em Sociologia* e, em seguida, ter proposto a conversão da aula em texto a ser publicado neste livro. A aula está disponível em: `https://www.youtube.com/live/qZiCA9aC_WM`. Agradeço também a Joana D'Arc da Silva pela revisão do texto.

[2] Em 2014, publiquei o livro *R para cientistas sociais*, um manual detalhado de como dominar a linguagem R. Entretanto, passados quase dez anos, o uso do R se tornou bem mais fácil, e vários capítulos precisariam ser reescritos.

O R está disponível em r-project.org, sendo preciso clicar em *CRAN* (abreviatura em inglês de *Rede Integral de Arquivos do R*) e escolher um dos servidores.[3] Usuários do Linux podem instalar o R a partir do repositório oficial de aplicativos da sua distribuição.

Nas próximas seções, veremos o que significa software livre, quais são algumas das características básicas da linguagem R, duas opções de interface de usuário, sugestões para trabalhar de modo mais produtivo e um breve comentário sobre pacotes que adicionam facilidades ou funcionalidades ao R. A conclusão é um convite para que você decida se irá usar ou não este software.

O R como software livre

Todo software livre tem seu código-fonte distribuído gratuitamente, permitindo a qualquer pessoa com conhecimento de programação modificar o código para atender às suas necessidades específicas, corrigir erros que eventualmente encontre ou reaproveitar o código em seus próprios projetos. Certamente, muitos leitores estão mais habituados com software proprietário, em que é preciso pagar por uma licença de uso, e devem se perguntar: como um software gratuito pode ser melhor do que um pago? Antes de responder esta pergunta, preciso explicar a diferença entre código-fonte e código binário e entre linguagem interpretada e compilada.

As linguagens de programação em uso atualmente são de alto nível, o que significa que estão relativamente próximas da linguagem natural humana, aproveitando, em geral, palavras da língua inglesa, tais como `for` e `while`, para instruir o computador a executar repetidamente alguns comandos, ou `if` e `else`, para executar um bloco de comandos somente se uma condição for atendida. Embora tenha uma "gramática" extremamente pobre e absolutamente rígida, qualquer pessoa com alguns minutos de treino consegue ter uma noção do que está escrito em um código-fonte. Mas o computador só executa comandos em linguagem binária, ou seja, contendo apenas os bits 0 e 1. Portanto, toda

[3] Para poupar os próprios servidores, o Projeto R permite que outras instituições mantenham cópias de seus arquivos. Muitas das instituições que espelham os arquivos são universidades, mas há também organizações sem fins lucrativos e empresas privadas que prestam o serviço voluntariamente.

linguagem de programação precisa, de alguma forma, ser convertida em linguagem binária para ser executada.

Algumas linguagens de programação precisam ser compiladas em linguagem binária antes de serem executadas. Nesse caso, os desenvolvedores de software têm a opção de distribuir somente o código binário e guardar para si o código-fonte. Assim, quem recebe ou compra um software consegue executá-lo porque ele já está pronto para ser processado pela máquina, mas não tem a oportunidade de inspecionar o código-fonte para ver como ele funciona e o que realmente faz. Um programa de computador, depois de compilado, é, para nós, uma quantidade enorme de bytes aleatórios. Assim são quase todos os programas que instalamos em nosso computador, como editores de texto, planilhas de cálculo, editor de imagens etc. Existem técnicas de engenharia reversa que tentam decifrar e descobrir quais são as instruções que estão sendo executadas, mas não é tarefa trivial escrever um código-fonte a partir de um código binário. Embora o software R, propriamente, tenha de ser compilado para ser executado, seu código-fonte é distribuído livremente. É possível inspecionar, alterar e recompilar o código. Então, essa é uma primeira vantagem do R.

Os programas de computador escritos em linguagens interpretadas são distribuídos na forma original – como código-fonte – e convertidos em linguagem binária no momento do uso por um interpretador. O R é um software binário (compilado a partir da linguagem C) que interpreta a linguagem R. Outro exemplo de interpretador é Python (que também é compilado a partir da linguagem C) e que interpreta a linguagem chamada de Python. Nas linguagens interpretadas, o código pode ser enviado para o interpretador, linha por linha, para vermos se cada linha, de fato, produz o resultado pretendido, o que facilita o trabalho de identificação e correção de erros.

Voltando à pergunta do início desta seção: software livre é melhor ou pior do que software proprietário? Depende de algumas coisas, principalmente do número, da motivação e da competência dos desenvolvedores do software. Os desenvolvedores do R são pesquisadores das mais diversas áreas – como agronomia, ciências sociais e medicina – com conhecimentos avançados de estatística e programação e que, muitas vezes, precisam desenvolver técnicas novas de análises de dados. Então, de maneira geral, as pessoas que contribuem para o desenvolvimento do R são pesquisadores da academia que desenvolvem novas

técnicas de análise de dados. Assim, o número de pesquisadores competentes usando o R é grande, sendo que milhares deles utilizam o R e outros milhares contribuem para o desenvolvimento de funções adicionais. É muito comum um conjunto de funções ser desenvolvido para resolver algum problema de pesquisa específico, e, depois, as funções serem reunidas em um pacote que é submetido à CRAN.

Mas o que motiva esses pesquisadores a disponibilizarem gratuitamente as funções que desenvolvem? A motivação pode ser uma combinação de obediência à ética da pesquisa e busca de prestígio. Todo pesquisador tem a obrigação de detalhar a metodologia da sua pesquisa para que outros possam tentar replicar seus resultados. Ao realizar uma pesquisa e chegar a alguns resultados, precisamos mostrar qual foi o método utilizado para que os nossos pares possam avaliar se a nossa pesquisa foi bem feita, se tem alguma falha ou se pode ser melhorada. Então, a promoção do conhecimento livre é própria da ética científica: os trabalhos têm de ser publicados e divulgados, de preferência, gratuitamente; o conhecimento tem de ser divulgado o máximo possível, com o mínimo de barreiras (idealmente, sem barreiras). Esse é o ideal da ciência, o conhecimento divulgado amplamente e com toda a metodologia utilizada sendo inspecionável, sendo, portanto, natural que os pesquisadores divulguem as funções que desenvolveram. Juntar todo o código desenvolvido em um pacote e submetê-lo à CRAN, além de ajudar outras pessoas, é uma forma de divulgar o próprio trabalho e angariar maior prestígio.

Esses mesmos pesquisadores também podem contribuir para o desenvolvimento de pacotes para outros programas, como SPSS ou Stata. Entretanto, acredito que as pessoas se sintam mais motivadas a contribuir gratuitamente para uma organização sem fins lucrativos (O Projeto R) e distribuir gratuitamente o seu produto (o programa R) do que a trabalhar gratuitamente para uma empresa privada. O resultado final nos levar a poder afirmar com segurança que o R, um exemplo de software livre, em geral, é melhor e mais completo do que seus concorrentes comerciais. Isso é ótimo para a ciência. Um software livre com pacotes e funções complementares, também livres, permite efetivar ao máximo procedimentos metodológicos inspecionáveis e resultados replicáveis.

A linguagem R

Observe o seguinte exemplo de código em linguagem R:

```
x <- c(23, 32, 41, 12, 33, 11, 13, 66, 10)
y <- c(2, 1, 2, 4, 5, 1, 1, 7, 1)
m <- lm("y ~ x")
```

Na linguagem R, criamos e manipulamos objetos, usando o símbolo `<-`. Objetos cujos nomes são seguidos de parênteses são funções, ou seja, um conjunto de instruções que executam tarefas específicas. As funções podem receber argumentos entre os parênteses. Os argumentos são objetos a serem processados pela função. No exemplo acima, a função `c( )` (abreviatura de *concatenate*) é usada para criar um objeto `x` e um objeto `y`, sendo ambos sequências de nove números. Em seguida, os nomes dos dois objetos são passados como parte de um texto para a função `lm( )` (abreviatura de linear model), que retorna um modelo de regressão, armazenado no objeto `m`. Note que passamos para a função `lm( )` um texto em que `y` e `x` estão separados por um `~`, símbolo que, neste caso, significa "explicado por". Ou seja, estamos ordenando ao R que faça uma análise de regressão em que `y` é a variável dependente e `x` a variável independente. No exemplo acima, usamos apenas as funções carregadas por padrão quando o R é iniciado. Para ver o resultado, basta digitar o nome do objeto com o modelo de regressão:

```
m

Call: lm(formula = "y ~ x")

Coefficients:

(Intercept) x
0.47041 0.08202
```

Para visualizar o mesmo modelo com uma formatação mais apropriada para um trabalho acadêmico, podemos usar a função `tab_model( )` do pacote `sjPlot`. Como se trata de um pacote que não vem instalado por padrão no R, será preciso primeiro instalá-lo e, uma vez instalado, carregá-lo usando a função `library( )`:

```
library("sjPlot")
tab_model(m, title = "Regressão linear: y
como variável dependente")
```

Regressão linear: y como variável dependente

	Dependent variable		
Predictors	Estimates	CI	p
(Intercept)	0.47	-1.93 – 2.87	657
x	0.08	0.01 – 0.16	36
Observations	9		
R2 / R2 adjusted	0.489 / 0.415		

Embora a seção de uso do R acima tenha sido brevíssima, é possível perceber uma diferença importante em relação à totalidade dos programas usados por não programadores: não selecionamos menus ou botões para executar comandos; temos de digitá-los manualmente. Isso significa que é necessário memorizar os nomes de dezenas ou centenas de funções e de quais argumentos elas recebem? Não exatamente. Embora seja preciso memorizar os nomes de algumas funções, atualmente existem interfaces para o R que facilitam muito a produção de código.

A interface do usuário

A interface para o R instalada por padrão (ver Figura 1) é perfeitamente funcional: os comandos digitados diretamente no console ou enviados do editor de textos embutido são executados e as análises são realizadas, os gráficos e tabelas produzidos etc. O sistema de consulta da documentação das funções também funciona perfeitamente. Mas a única ajuda para a escrita do código é a completação dos nomes de objetos no console do R após ser pressionada a tecla `Tab`.

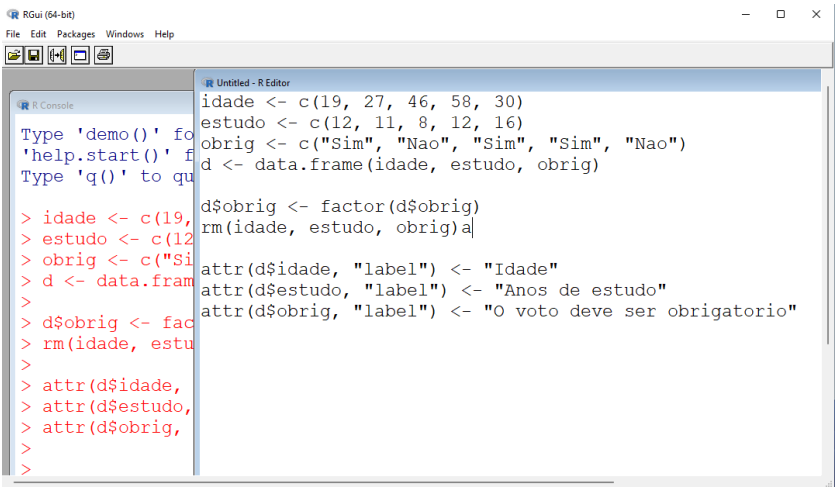


Figura 1 Rgui no Windows.

Interfaces alternativas não adicionam ao R nenhuma capacidade extra de manipular e analisar dados, mas trazem várias facilidades para a produção de código:

- Indentação automática: inserção de recuos no texto que facilitam a visualização de sua estrutura;
- Realce de sintaxe: atribuição de cores diferentes para os vários tipos de elementos presentes no código: funções, operadores lógicos, termos de controle de fluxo (como `for`, `while` e `if`) etc.;
- Visualização dos objetos presentes na memória do R;
- Completação automática dos nomes de objetos e argumentos de funções.

As duas primeiras funcionalidades facilitam a leitura do código e as outras duas liberam nossa própria memória para que possamos nos dedicar aos desafios lógicos relacionados à tarefa de elaborar a análise dos dados.

A interface mais usada é o *RStudio Desktop*, disponível gratuitamente em `https://posit.co`. Como poderá ser visto na Figura 2, a janela do RStudio apresenta quatro divisões: do lado esquerdo, no alto, o editor de texto e, embaixo, o console do R; do lado direito, no alto, abas

para visualizar os objetos carregados na memória do R, o histórico dos comandos e conexões e, embaixo, abas para visualizar gráficos, arquivos, ajuda de funções e para visualizar resultados.

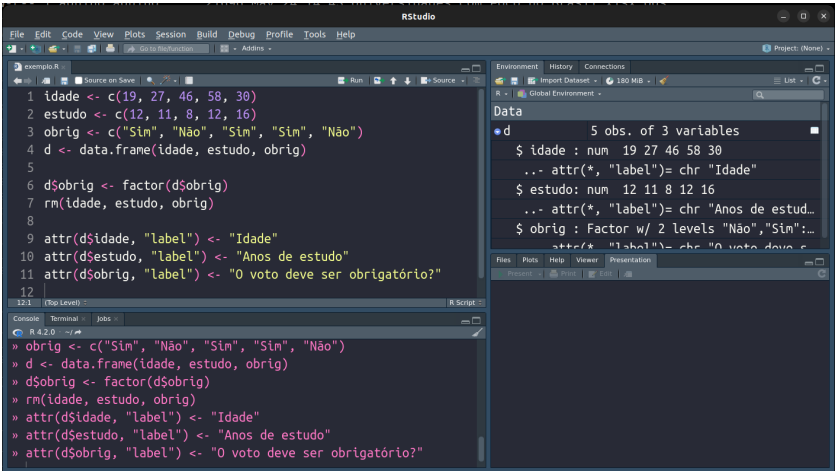


Figura 2 RStudio.

Outra interface para o R é o Nvim-R, um *plugin* para o editor de textos Neovim.[4] O Neovim funciona em um terminal, e, na Figura 3, está sendo executado em um terminal do Linux. Na imagem, podemos ver a janela dividida em três partes: na parte superior esquerda, o editor de texto; na parte inferior, o console do R; e, na parte superior direita, um painel para visualização dos objetos presentes na memória do R.

[4] Sou o principal autor e desenvolvedor do Nvim-R, disponível em: `https://github.com/jalvesaq/Nvim-R`.



Figura 3 Nvim-R.

O RStudio e o Nvim-R possuem sistemas semelhantes de completação automática de nomes de objetos, a ser visto na Figura 4. Mas o RStudio não precisa de nenhuma configuração para ser usado e seu editor de textos adota atalhos comuns, como `CTRL-C` para copiar e `CTRL-V` para colar, sendo possível usar o mouse para clicar em menus e botões. Ou seja, é um programa de interface familiar para todos os usuários de computador. O Neovim não possui menus e botões, sendo necessário se familiarizar com seu uso básico antes de conseguir editar algum texto. Além disso, é preciso instalar o *plugin* Nvim-R e memorizar alguns dos seus comandos antes de conseguir iniciar e usar o R. Apesar disso, o Neovim é considerado um editor de texto que permite uma maior produtividade em programação, e por isso é preferido por alguns desenvolvedores de software.[5]

[5] Interessados em usar o Nvim-R devem instalar o Neovim (`https://neovim.io/`) e, ao abri-lo pela primeira vez, digitar `:Tutor` para iniciar um minicurso de uso do editor.

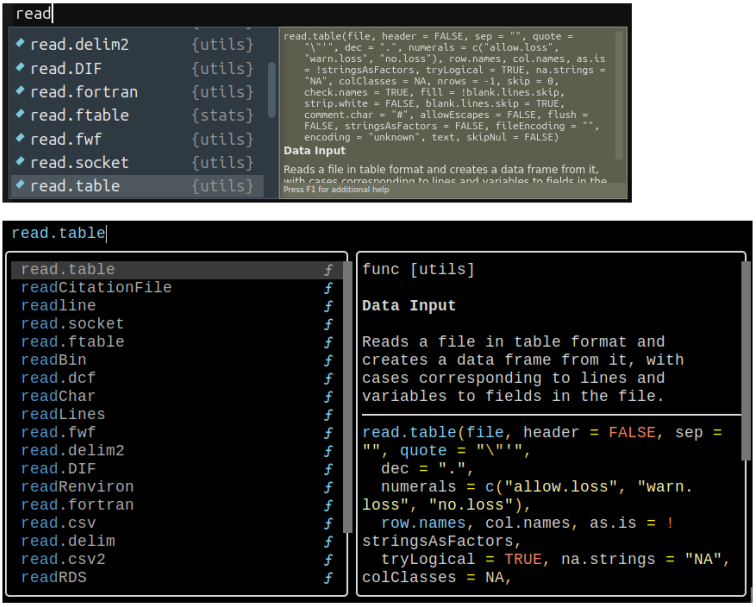


Figura 4 Completação automática no RStudio (acima) e no Nvim-R (abaixo).

Existem várias outras opções de interface não abordadas neste capítulo, como Emacs + ESS, Visual Studio Code e StatET. Assim como o Neovim, são editores de texto voltados para programadores e que incluem suporte para o R.

Como trabalhar com o R

Nesta seção, apresento três sugestões de como trabalhar com o R, independentemente da interface utilizada:

1. Desenvolver um *script* (e não digitar os comandos diretamente no console do R);
2. Produzir um *script* conciso, sem incluir comandos desnecessários para a realização da análise de dados;
3. Usar a linguagem `Quarto` para escrever relatórios e não apenas `scripts` simples do R.

O *script* é um texto sem formatação especial, como itálico ou negrito. O texto pode ser exibido pelo editor de textos em cores diferenciadas

para cada tipo de objeto, mas continua sendo apenas texto puro em que cada linha é um comando a ser executado pelo R. A minha sugestão é que, quando você for fazer uma pesquisa, desenvolva um *script* que registre todos os comandos necessários para análise, desde o carregamento do banco de dados até o último passo da análise de dados. Tudo deve ficar registrado no *script*. Por quê? Porque essa é uma vantagem e desvantagem do R. Em um programa como o SPSS, por exemplo, é possível fazer toda a análise dos dados apertando alguns botões e selecionando itens no menu. É mais fácil porque não precisa memorizar comandos, mas essa facilidade implica também uma enorme desvantagem: depois de alguns dias de trabalho, você não saberá mais qual caminho percorreu. Terá recodificado variáveis, computado novas variáveis, substituído valores omissos e, no final, dificilmente lembrará de todos os passos que seguiu. E será ainda pior quando interromper o trabalho por alguns dias ou quando, após decorridos alguns meses, resolver voltar a analisar o mesmo banco de dados. Sem um *script*, dificilmente alguém conseguirá replicar seus resultados. Possivelmente, nem mesmo você conseguirá replicar os próprios resultados, pois tantas decisões foram tomadas durante o processo de análise de dados que se torna pouco provável esses passos serem seguidos exatamente da mesma forma. O resultado será uma pesquisa não perfeitamente replicável. Ou seja, usar um software como o R, que requer a produção de *scripts*, é mais difícil, mas essa desvantagem é amplamente compensada por uma vantagem muito importante: a produção de resultados que podem ser replicados por outros pesquisadores, o que é fundamental para o avanço do conhecimento científico.

Além disso, se você notar um erro nos resultados, e esse erro tiver sido cometido vários passos atrás, bastará corrigir o comando errado e rodar novamente o *script* para, em poucos segundos, ter todos os resultados corrigidos. Se estiver clicando em menus e botões, terá de repetir todos os passos desde o início. E terá de fazer isso sempre que perceber algum erro.

Quando estamos fazendo análise de dados, precisamos de vários comandos que são auxiliares, por exemplo, listar os objetos que estão na memória do R, imprimir um objeto específico para ver se ele é um texto, se é numérico, para saber quais são os valores que ele possui para depois fazer alguma recodificação etc. Então, temos de fazer várias pequenas tarefas para conhecer melhor nossos dados antes de executar

algum comando que, propriamente, transformará os dados ou produzirá estatísticas úteis. Se tudo isso ficar no *script*, ele ficará com muita informação irrelevante e desordenada, dificultando a posterior leitura em busca de falhas ou de trechos que precisem de melhoria. A sugestão é que esses comandos auxiliares sejam feitos diretamente no console do R. Comandos que eventualmente estejam no *script* mas que tenham sido feitos apenas para conhecer melhor os dados devem ser eliminados. Você até pode deixar alguns comandos auxiliares se avaliar que, no futuro, eles lhe ajudarão a entender as escolhas que fez, mas a dica é manter o *script* limpo.

A produção de um *script* permite a realização de um trabalho mais organizado e, somado à distribuição dos dados, garante a replicabilidade dos resultados. A análise dos dados deve ser feita para ser inspecionada pelos pares. O recomendável é que, ao publicar um artigo ou um capítulo com análise quantitativa, o *script* que foi usado para fazer a análise dos dados também seja publicado. Assim, outros pesquisadores poderão avaliar seu trabalho, confirmar que não há nenhum erro, e poderão, também, aprender com seu *script*, que poderá servir de modelo, evitando que outros pesquisadores passem pelas dificuldades que você enfrentou. A ética do trabalho científico requer que facilitemos a superação do nosso trabalho por outros pesquisadores.

Mas escrever *scripts* em linguagem R ainda não é a melhor forma de aproveitar o potencial do software R. Podemos escrever o próprio texto do relatório da pesquisa em Quarto, um tipo de *script* que combina Markdown – linguagem de marcação de texto – com a linguagem R.[6] É o que estou usando para escrever este capítulo.

Na Figura 5 há um exemplo de documento Quarto e documento PDF resultante. É um exemplo mínimo, mas suficiente para ilustrar que a linguagem Markdown é extremamente simples. Com um investimento de poucos minutos no aprendizado da linguagem, já é possível começar a fazer relatórios em Quarto sem ter de manualmente copiar e colar figuras, tabelas e estatísticas, economizando tempo e evitando erros simples como a exibição de estatísticas que se tornaram desatualizadas após alguma revisão do código.

[6]Quarto é o sucessor do Rmd, com algumas melhorias e correções em relação a seu antecessor. Para um guia completo de como usar Quarto, acesse: `https://quarto.org/`.

````
---
title: "Pesquisa Social Brasileira --- 2002"
author: "Jakson Alves de Aquino"
lang: pt-BR
format: pdf
echo: false
---

```{r}
#| message: false

library("tidyverse")
load("pesb2002.RData")
p <- select(pesb,
            aborto = q90,
            escolaridade = q542) |>
    filter(! aborto %in% c("NS", "NR")) |>
    droplevels()
p$escolaridade <- factor(p$escolaridade,
                         levels = c("Analfabeto",
                                    "Até 4a. série",
                                    "De 5a. a 8a. série",
                                    "2o. grau",
                                    "Superior ou mais"))
```

Como mostra a fig-abXesc, em 2002, de um total de `r nrow(p)` de
entrevistados que responderam a questão sobre aborto e informaram
sua escolaridade, uma minoria considerava que o aborto era algo
sempre errado, sendo esta opinião mais comum entre as pessoas com
escolaridade mais baixa.

```{r}
#| fig-cap: "Opinião dos brasileiros sobre aborto em 2002"
#| label: fig-abXesc

ggplot(p, aes(aborto, fill = escolaridade)) +
    geom_bar(position="dodge") +
    ylab("Frequência") +
    xlab("Posição sobre o aborto") +
    guides(fill = guide_legend(title = "Escolaridade"))
```
````

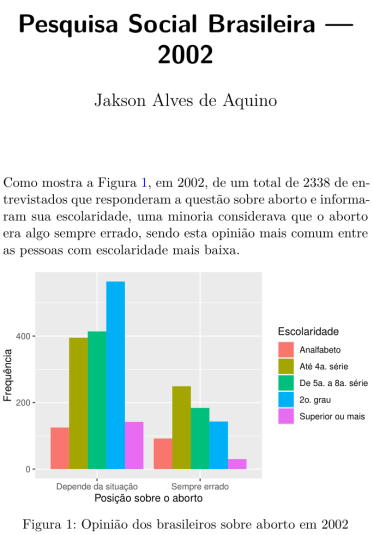


Figura 1: Opinião dos brasileiros sobre aborto em 2002

Figura 5 Exemplo de documento Quarto.

No exemplo, no cabeçalho delimitado por três hifens acima e abaixo, informamos qual é o título e quem é o autor do documento, que o idioma do documento é o português do Brasil (o que, neste exemplo, altera o prefixo da figura de "Figure" para "Figura" e garante a hifenização correta das palavras), que queremos um documento PDF (poderia ser HTML, ODT, DOCX, epub ou outros formatos) e que não queremos que o código R seja exibido (`echo: false`). No texto principal do documento, dois blocos de código R estão delimitados por ```` ```{r} ```` acima e ```` ``` ```` abaixo. O primeiro bloco carrega um banco de dados e prepara duas variáveis para um gráfico de barras: os casos em que os respondentes não responderam ("NR") ou disseram não saber a resposta ("NS") são eliminados e a variável escolaridade tem suas categorias reordenadas porque no banco original elas estavam em ordem alfabética, e o gráfico será de mais fácil interpretação se as categorias forem ordenadas da escolaridade menor para a maior. Em seguida, um texto comenta a figura que será produzida, fazendo uso de código R (`` `r nrow(p)` ``) para informar o número de linhas do banco de dados. Finalmente, temos um bloco de código do R que produz um gráfico. No cabeçalho do bloco, informamos qual é o título da figura (`fig-cap`) e o rótulo (label) a ser usado para fazer referência à figura.

Fica evidente que se fizermos qualquer alteração no código, gráficos e estatísticas serão atualizados automaticamente. Há o aumento de produtividade por não ser necessário trocar manualmente todas as figuras, tabelas e estatísticas do artigo após cada revisão da análise dos dados. Isso significa, por exemplo, que, se você estiver com prazo até às 23:59 para entregar um trabalho, pode continuar revisando sua análise de dados até às 23:57, gerar o documento final às 23:58 e entregar o trabalho no prazo. É possível que haja problemas de qualidade por você não ter conseguido se dedicar tanto quanto precisaria, mas não haverá motivo para recear ter alguma figura, tabela ou estatística desatualizada.

Alguns pacotes e funções

O R, ao ser carregado, tem mais de duas mil funções disponíveis, mas mesmo para muitas tarefas básicas existem funções alternativas que são mais fáceis de usar do que as que são carregadas por padrão. São particularmente úteis pacotes que trazem funções que unificam tarefas. É o caso do pacote `rio`, cuja função `import( )` permite carregar dados de vários formatos diferentes (SPSS, Excel, Stata, csv etc.), o que nos dispensa da necessidade de memorizar o nome de uma função diferente para cada tipo de arquivo. Também facilitam muito o uso do R os pacotes que contêm funções nomeadas de forma sistemática. Um exemplo é o pacote `ggplot2` (para produzir gráficos) e o pacote `stringr` (para manipular texto). No `ggplot2`, as 53 funções que adicionam elementos aos gráficos estão nomeadas com o prefixo `geom_`. Então, basta digitar `geom_`, aguardar uma fração de segundo para ser exibido o menu com os nomes das funções e escolher uma delas. No `stringr`, as 52 funções para manipulação de strings têm o prefixo `str_`.

Vários outros pacotes trazem funções que permitem fazer análises de dados comuns nas ciências sociais, tais como `lme4` (regressão multinível), `FactoMineR` (análise de correspondência múltipla), `psych` (análise de componentes principais), `igraph` (análise de redes sociais), `apc` para (análise de efeito idade-período-coorte) e `wnominate` (análise de votações nominais). A lista completa de pacotes disponíveis com uma breve descrição de cada um pode ser acessada na CRAN (procure por *Available CRAN Packages By Name*).[7]

[7] No dia 15 de maio de 2023, eram listados 19.511 pacotes.

Conclusão

Por ser software livre e por ter uma larga base de usuários competentes em programação, o R é o software de estatística mais completo e mais capaz de realizar tarefas complexas. O fato de ser necessário programar para usar o R tem, pelo menos, duas consequências relevantes para nossa discussão: 1) a análise de dados é efetivada com a elaboração de um *script*, o que torna os resultados replicáveis com exatidão, contribuindo para o atendimento de uma exigência da ética da pesquisa;2) é preciso se familiarizar com lógica de programação.

Espero que as informações apresentadas neste capítulo tenham sido suficientes para você decidir se usará o R ou outro software para análises estatísticas de dados, e acredito que as perguntas abaixo lhe ajudem na sua decisão:

1. Durante sua vida acadêmica, você terá de fazer análise estatística de dados com frequência?

2. Você se preocupa em produzir resultados replicáveis em suas pesquisas?

3. Na organização em que trabalha ou em que planeja trabalhar, você terá dificuldade para justificar a compra de software proprietário de análise estatística de dados?

4. Você tem disposição para aprender uma linguagem de programação?

Se respondeu "Sim" em todas as perguntas, então é alta a probabilidade de que usará o R em suas pesquisas.

Destaques e curiosidades

O R já teve um pacote – RQDA –, que contava com uma interface gráfica para análise de dados qualitativos. Esse pacote deixou de funcionar e foi removido da CRAN em 2020. Hoje, a alternativa é usar o QualCoder (`https://qualcoder.wordpress.com/`), que também é software livre e tem funcionalidade similar, mas não depende do R para funcionar.

Saiba mais

Curso-r:
`https://livro.curso-r.com/index.html`

Ciência de Dados com R:
`https://cdr.ibpad.com.br/`

R for Data Science (2ª ed.):
`https://r4ds.hadley.nz/`

Como citar este capítulo:

AQUINO, Jakson Alves de. R na pesquisa acadêmica. *In*: MOTTA, Luana Dias; ZANON, Breilla Valentina Barbosa. *Metodologia de pesquisa em sociologia: elaborando um projeto e realizando uma pesquisa*. São Carlos: EdUFSCar, 2024, p. 148–158. ISBN: 978-85-7600-633-6.