R sem medo

Um guia amigável de análise de dados para quem nunca programou

```
>
> dados ← ler.csv("dados.csv")
> summary(dados)
> plot(dados$valor ~ dados$tempo)
> lm_modelo ← lm(valor ~ tempo, data = dados)
> summary(lm_modelo)
>
```

Iara Teixeira

R sem medo

Um guia amigável de análise de dados para quem nunca programou

Iara Teixeira

R sem medo

Um guia amigável de análise de dados para quem nunca programou

Iara Teixeira

Primeira edição

ISBN: 9798184895062

Material complementar (toolbox), de acesso aberto:
https://doi.org/10.17605/OSF.IO/J4HQU

Prefácio

Aprender a usar R não é difícil. O que é difícil é começar quando tudo parece um pouco hostil ao mesmo tempo: o programa, a sintaxe, a interface, os erros que aparecem em inglês, a sensação de que toda explicação na internet já assume um conhecimento que ninguém te ensinou. Este livro foi pensado para uma pessoa que se reconhece nesse cenário.

A proposta aqui é simples. Cobrir um percurso curto e organizado, desde abrir o RStudio pela primeira vez até rodar uma análise completa, interpretar o resultado e salvar o trabalho. O foco está em fazer a leitora chegar ao final de cada capítulo com algo concreto: um banco carregado, uma descritiva conferida, um teste rodado, um gráfico salvo.

O livro vem acompanhado de uma toolbox: um conjunto pequeno de scripts em R, um banco simulado para praticar e um dicionário das variáveis. A toolbox cobre o mesmo percurso do livro. Onde o livro explica, a toolbox executa. As duas peças foram pensadas para serem usadas juntas, embora possam funcionar separadas. As instruções de acesso estão no fim do volume.

Boa leitura.

Sumário

Fundamentos

Como abrir o RStudio, entender como o R pensa e conhecer o banco que vamos usar.

Capítulo 1 — Como usar este livro

Neste capítulo

- Sumário do percurso
- O que este livro cobre
- Como usar este livro
- A promessa deste livro

1.1 Sumário do percurso

Este livro foi pensado para uma pessoa que abre o R pela primeira vez e sente que entrou em um território estranho. A proposta não é transformar você em programadora profissional. A proposta é muito mais prática: ajudar você a carregar um banco de dados, entender o que está fazendo, rodar análises básicas, interpretar os resultados e salvar o trabalho de forma organizada.

1.2 O que este livro cobre

Este volume cobre o percurso básico e suficiente para uma primeira análise quantitativa em R:

abrir o RStudio sem medo;
entender scripts, console, objetos e funções;
carregar e conferir um banco de dados;
conhecer as variáveis antes de testar hipóteses;
produzir descritivas;
comparar grupos;
fazer regressões simples;
criar gráficos básicos;
salvar resultados;
revisar erros comuns.

1.3 Como usar este livro

Leia com o RStudio aberto. Não leia como se fosse um romance. Leia como se fosse uma oficina. A cada trecho de código, copie, rode, observe o que aconteceu e volte ao texto. R é aprendido pelo uso, não pela memorização.

Ao longo dos capítulos, você verá exemplos pequenos de código. Eles foram mantidos no livro porque ajudam a entender a lógica. Os scripts completos ficam na toolbox. Assim, o livro continua legível, e a prática fica organizada em arquivos .R.

1.4 A promessa deste livro

Você não precisa entender tudo de primeira. Você só precisa entender o suficiente para dar o próximo passo. O livro vai repetir algumas ideias de propósito, porque repetição é parte da aprendizagem. A meta é autonomia, não velocidade.

Capítulo 2 — Primeiros passos: abrindo o R sem medo

Neste capítulo

- O que você vai aprender neste capítulo
- R e RStudio não são a mesma coisa
- O que é um projeto do RStudio
- Como criar um projeto
- A tela do RStudio
- Script e console: a diferença mais importante
- Como rodar código
- A primeira experiência com erro
- O que salvar e o que não salvar
- Uma estrutura simples de pasta
- Mini exercício
- Erros comuns deste capítulo
- Onde está o script completo
- A mentalidade de trabalho reprodutível
- Como saber onde o R está procurando arquivos
- Nomeando arquivos sem criar confusão
- O que fazer na primeira semana usando R

2.1 O que você vai aprender neste capítulo

Neste capítulo, você vai entender o que são R, RStudio, projeto, script, console e pasta de trabalho. Parece básico, mas é aqui que muitas pessoas se perdem. Quando alguém diz “meu código não encontra o banco”, “não sei onde o resultado foi salvo” ou “rodei ontem e hoje não funciona”, quase sempre o problema começou na organização inicial.

Ao final deste capítulo, você deve conseguir abrir o RStudio, criar um projeto, localizar a pasta correta, criar um script e rodar comandos linha por linha.

2.2 R e RStudio não são a mesma coisa

R é a linguagem. É o motor que faz os cálculos. RStudio é o ambiente de trabalho. É a tela onde você escreve, organiza, salva e executa o código. Uma comparação simples: R é o motor do carro; RStudio é o painel, o volante, o banco e o câmbio. Você até poderia usar R sem RStudio, mas para quem está começando, isso só dificultaria a vida.

Por isso, instale os dois. Primeiro o R, depois o RStudio. A ordem importa porque o RStudio precisa encontrar o R instalado no computador.

Os dois programas são gratuitos. O R é baixado no site do CRAN, e o RStudio Desktop, no site da Posit:

Exemplo

```
https://cran.r-project.org
https://posit.co/download/rstudio-desktop
```

Em cada site, escolha a versão do seu sistema (Windows, Mac ou Linux), baixe o instalador e aceite as opções padrão. No Mac, há duas versões do R: uma para processador Apple Silicon e outra para Intel. Para saber qual é o seu, abra o menu Apple e clique em Sobre este Mac.

Dois tropeços comuns nessa etapa: baixar o instalador do sistema errado e, depois de instalar, abrir o programa errado. O R instala um aplicativo próprio, mas não é ele que você vai usar no dia a dia. Abra sempre o RStudio.

2.3 O que é um projeto do RStudio

Um projeto é uma pasta organizada para uma análise. Ele guarda os scripts, os dados, os resultados e tudo que pertence àquele trabalho. A vantagem é simples: quando você abre o projeto, o RStudio entende que aquela pasta é o “mundo” da sua análise.

Sem projeto, você pode cair no erro de escrever caminhos enormes, como:

Exemplo

```
read.csv("/Users/seu_nome/Downloads/pasta_antiga/
dados.csv")
```

Esse tipo de caminho funciona hoje e quebra amanhã. Funciona no seu computador e quebra no computador de outra pessoa. Com projeto, você trabalha com caminhos simples:

Código em R

```
read.csv("dados/dados_simulados.csv")
```

O R entende que a pasta dados está dentro do seu projeto.

2.4 Como criar um projeto

No RStudio, vá em File > New Project > New Directory > New Project. Escolha um nome simples, sem acentos e sem espaços. Por exemplo:

Exemplo

```
analise_burnout
r_sem_medo
artigo_1_descritivas
```

Evite nomes como:

Exemplo

```
Análise final da Iara versão 2 corrigida NOVA
```

Não é proibido, mas nomes longos, acentos e espaços aumentam a chance de pequenos problemas. Em programação, simplicidade ajuda.

2.5 A tela do RStudio

Quando você abre o RStudio, normalmente aparecem quatro áreas.

A parte superior esquerda é o **script**. É onde você escreve o código que será salvo. Pense no script como o caderno da análise. Tudo que importa deve estar nele.

A parte inferior esquerda é o **console**. É onde o R executa comandos. O console mostra respostas, erros e mensagens. Você pode digitar direto ali, mas para uma análise reprodutível, prefira escrever no script e enviar para o console.

A parte superior direita é o **Environment**. Ele mostra os objetos que você criou: bancos de dados, resultados, modelos e vetores. Se você carrega um banco chamado dados, ele aparece ali.

A parte inferior direita reúne abas como **Files**, **Plots**, **Packages** e **Help**. A aba Files mostra os arquivos da pasta. A aba Plots mostra gráficos. A aba Packages mostra pacotes instalados. A aba Help mostra a documentação das funções.

2.6 Script e console: a diferença mais importante

O console é imediato, mas passageiro. O script é organizado e permanente. Se você digita um comando no console e fecha o RStudio, aquele comando não fica salvo. Se você escreve no script e salva, pode abrir de novo amanhã.

A regra é: use o console para testes rápidos, mas coloque no script tudo que faz parte da análise.

Um script simples pode começar assim:

Código em R

```
# Meu primeiro script em R

2 + 2

nome <- "Iara"
idade <- 35

nome
idade
```

A primeira linha começa com #. Isso é um comentário. O R ignora comentários. Eles servem para você explicar o que está fazendo.

2.7 Como rodar código

No RStudio, coloque o cursor na linha que quer executar e pressione Ctrl + Enter no Windows/Linux ou Cmd + Enter no Mac. O comando será enviado ao console. Outro atalho que vale aprender cedo: Alt + - no Windows/Linux, ou Option + - no Mac, insere o símbolo <- já com os espaços corretos.

Você pode selecionar várias linhas e usar o mesmo atalho. Evite rodar o script inteiro logo no começo. Para aprender, rode devagar. Linha por linha. Observe o que muda.

2.8 A primeira experiência com erro

R mostra erros. Isso não significa que você falhou. Significa que o R não entendeu algo.

Por exemplo:

Código em R

```
media(1, 2, 3)
```

O R provavelmente responderá algo como:

Saída do R

```
Error: could not find function "media"
```

Ele está dizendo: “não conheço uma função chamada media”. A função correta em R é mean():

Código em R

```
mean(c(1, 2, 3))
```

O erro não é um julgamento. É uma pista. Aprender R é aprender a ler pistas.

2.9 O que salvar e o que não salvar

Salve o script. Não salve o workspace.

Quando você fecha o RStudio, ele pode perguntar se quer salvar o workspace. O workspace guarda objetos criados na sessão. Parece útil, mas cria confusão. Você abre o R no dia seguinte e vê objetos que existem, mas não lembra como foram criados.

Prefira esta lógica: se um objeto é importante, o código que cria esse objeto deve estar no script. Assim, você sempre consegue recriar a análise do zero.

No RStudio, recomendo configurar:

Exemplo

```
Tools > Global Options > General
```

Desmarque a opção de restaurar .RData ao iniciar e escolha Never para salvar workspace ao sair.

2.10 Uma estrutura simples de pasta

A toolbox deste livro já vem organizada, mas é útil entender a lógica:

Exemplo

```
meu_projeto/
  dados/
  scripts/
  resultados/
  resultados/figuras/
```

A pasta dados guarda o banco. A pasta scripts guarda código. A pasta resultados guarda tabelas, saídas e arquivos salvos. A pasta figuras guarda gráficos.

Essa organização parece detalhe, mas ela transforma uma análise bagunçada em uma análise auditável.

2.11 Mini exercício

Abra o RStudio, crie um projeto chamado treino_r_sem_medo e crie dentro dele as pastas:

Exemplo

```
dados
scripts
resultados
```

Depois, crie um script chamado 01_treino.R e escreva:

Código em R

```r
# Treino do Capítulo 2

2 + 2

mensagem <- "Eu consegui rodar meu primeiro script"
mensagem
```

Rode linha por linha. Salve o script.

2.12 Erros comuns deste capítulo

O erro mais comum é abrir o RStudio fora do projeto. Se você abriu o RStudio, mas não abriu o arquivo .Rproj, talvez o R esteja procurando arquivos na pasta errada.

Outro erro comum é escrever código no console e não salvar no script. No momento parece rápido, mas depois você perde o caminho da análise.

O terceiro erro comum é tentar entender tudo antes de começar. Não faça isso. Abra, rode, observe. A compreensão vem com repetição.

2.13 Onde está o script completo

Na toolbox, este capítulo conversa principalmente com:

Exemplo

```
scripts/00_setup_auto.R
scripts/01_carregar_banco.R
```

Você não precisa abrir esses scripts agora. Apenas entenda que eles existem para automatizar parte da preparação quando chegar a hora de rodar as análises.

2.14 A mentalidade de trabalho reprodutível

Uma análise reprodutível é uma análise que pode ser refeita. Isso não significa apenas que o código “roda”. Significa que outra pessoa, ou você daqui a seis meses, consegue abrir a pasta, entender a ordem dos scripts, encontrar o banco e reproduzir os resultados.

Para uma iniciante, reprodutibilidade começa com hábitos simples:

não trabalhar diretamente na pasta Downloads;

não deixar arquivos espalhados pela área de trabalho;

não salvar resultados apenas no console;

não depender de cliques que você não sabe repetir;

não alterar dados brutos sem guardar uma cópia.

Pense no script como uma receita. Se a receita está clara, o prato pode ser feito de novo. Se você cozinhou “no olho” e não anotou nada, pode até ter dado certo uma vez, mas fica difícil repetir.

2.15 Como saber onde o R está procurando arquivos

Quando o R tenta abrir um arquivo, ele procura a partir de uma pasta atual. Para ver essa pasta, rode:

Código em R

```
getwd()
```

getwd() significa “get working directory”, ou seja, “mostre a pasta de trabalho atual”. Se essa pasta não for a pasta da toolbox ou do seu projeto, o R pode não encontrar o banco.

Para listar arquivos da pasta atual:

Código em R

```
list.files()
```

Para listar arquivos dentro da pasta dados:

Código em R

```
list.files("dados")
```

Essas três funções ajudam muito quando aparece erro de caminho. Antes de mudar o código inteiro, verifique onde o R está.

2.16 Nomeando arquivos sem criar confusão

Evite nomes como:

Exemplo

```
analise final FINAL corrigida 2.R
```

Prefira nomes numerados:

Exemplo

```
01_carregar_banco.R
02_descritivas.R
03_comparar_grupos.R
04_regressao.R
05_graficos.R
```

A numeração mostra a ordem. Isso ajuda você, ajuda orientadores, ajuda coautores e ajuda qualquer pessoa que precise revisar o trabalho.

2.17 O que fazer na primeira semana usando R

Na primeira semana, o objetivo não é dominar estatística. O objetivo é ficar confortável com o ambiente.

Uma rotina realista:

abrir o projeto;
abrir um script;
rodar cinco linhas;
observar o Environment;
salvar o script;
fechar e abrir de novo;
rodar novamente.

Se você consegue repetir esse ciclo sem se perder, já avançou muito.

Capítulo 3 — Como o R pensa: objetos, funções e pequenos comandos

Neste capítulo

- O que você vai aprender neste capítulo
- O que é um objeto
- Nomes de objetos
- O que é uma função
- Argumentos: as opções da função
- Comentários: o código conversando com você
- O símbolo $
- O símbolo ==
- O que é uma tabela para o R
- Vetores: uma coluna é uma lista de valores
- Tipos de variáveis
- Valores ausentes: o famoso NA
- Uma linha de código bem lida
- Mini exercício
- Erros comuns deste capítulo
- Onde está o script completo
- Como ler uma função desconhecida
- A diferença entre salvar e imprimir
- O que é uma fórmula no R
- Pequenas regras que evitam grandes erros
- Filtrando linhas com comparações

3.1 O que você vai aprender neste capítulo

Este capítulo explica a lógica mínima do R. A maior dificuldade de quem começa não é estatística. É entender a gramática do código. O R parece falar uma língua cheia de símbolos. Depois que você entende a lógica desses símbolos, o medo diminui muito.

Vamos trabalhar com quatro ideias: objeto, função, argumento e comentário. Se você entender essas quatro palavras, já consegue ler a maior parte dos scripts básicos deste livro.

3.2 O que é um objeto

Um objeto é qualquer coisa que você guarda com um nome. Pode ser um número, um texto, uma tabela, um resultado ou um modelo.

Código em R

```
idade <- 35
```

Essa linha significa: guarde o valor 35 em um objeto chamado idade.

O símbolo <- é chamado de atribuição. Ele pode ser lido como “recebe”. A frase acima pode ser lida assim:

Exemplo

```
idade recebe 35
```

Agora, se você digitar:

Exemplo

```
idade
```

O R mostra o valor guardado.

Objetos são a memória da sua análise. Você cria objetos para não precisar repetir tudo o tempo todo.

3.3 Nomes de objetos

Escolha nomes simples e descritivos. Bons nomes:

Código em R

```
idade_media <- 41.2
banco_principal <- read.csv("dados/dados_simulados.csv")
modelo_regressao <- lm(burnout ~ demanda, data =
banco_principal)
```

Nomes ruins:

Código em R

```
x <- 41.2
aaa <- read.csv("dados.csv")
coisa <- lm(burnout ~ demanda, data = aaa)
```

Nomes curtos demais funcionam, mas deixam o script difícil de ler. A pessoa mais importante que vai ler seu script daqui a alguns meses é você mesma.

3.4 O que é uma função

Uma função é uma ação que o R sabe fazer. Funções têm nome e parênteses.

Código em R

```
mean(c(1, 2, 3))
```

Aqui, mean() é a função que calcula média. O que está dentro dos parênteses é a entrada da função.

A função pode ser lida como uma pergunta:

Exemplo

```
R, calcule a média destes números: 1, 2 e 3.
```

O R responde:

Exemplo

```
2
```

3.5 Argumentos: as opções da função

Algumas funções precisam de informações extras. Essas informações são chamadas de argumentos.

Código em R

```
mean(c(1, 2, NA, 3), na.rm = TRUE)
```

O argumento na.rm = TRUE diz ao R para remover valores ausentes antes de calcular a média. Sem isso, a média vira NA.

A tradução é:

Exemplo

```
calcule a média destes valores e remova os NAs antes.
```

Argumentos tornam funções flexíveis. A mesma função pode fazer coisas diferentes dependendo dos argumentos que você passa.

3.6 Comentários: o código conversando com você

Tudo que vem depois de # é ignorado pelo R.

Código em R

```
# Calcular a média de idade
mean(dados$idade, na.rm = TRUE)
```

Comentários são fundamentais em scripts para iniciantes. Não pense neles como “enfeite”. Pense como uma explicação para o seu eu do futuro.

Um script sem comentários pode até funcionar, mas é difícil de revisar. Um script com comentários vira material de estudo.

3.7 O símbolo $

Quando você tem uma tabela chamada dados, as colunas ficam dentro dela. Para pegar uma coluna específica, use $.

Código em R

```
dados$idade
```

Isso significa: dentro de dados, pegue a coluna idade.

Se você quiser calcular a média da idade:

Código em R

```
mean(dados$idade, na.rm = TRUE)
```

Essa linha parece curta, mas tem muita coisa acontecendo:

dados é a tabela;
$idade escolhe a coluna idade;
mean() calcula a média;
na.rm = TRUE ignora valores ausentes.

3.8 O símbolo ==

Em R, = e == não são a mesma coisa. O = pode ser usado dentro de funções para definir argumentos. O == é usado para comparar.

Código em R

```
dados$genero == "Mulher"
```

Essa linha pergunta, para cada pessoa, se o gênero é igual a “Mulher”. O resultado é uma sequência de TRUE e FALSE.

Erro comum:

Código em R

```
dados$genero = "Mulher"
```

Essa linha não compara. Ela tenta atribuir. E o pior: roda sem mensagem de erro e substitui a coluna inteira pelo texto Mulher. Se isso acontecer, recarregue o banco com read.csv() para recuperar os dados originais. Por isso, em comparações, use ==. Na seção 3.21, essa comparação vai servir para escolher linhas do banco.

3.9 O que é uma tabela para o R

O banco de dados que usamos neste livro é uma tabela. Em R, uma tabela geralmente é chamada de data.frame. Ela tem linhas e colunas.

As linhas representam pessoas, casos ou observações. As colunas representam variáveis.

Se a base tem 1200 linhas e 13 colunas, isso significa que há 1200 participantes e 13 variáveis.

Você pode conferir com:

Código em R

```
nrow(dados)
ncol(dados)
dim(dados)
```

dim(dados) devolve duas informações: número de linhas e número de colunas.

3.10 Vetores: uma coluna é uma lista de valores

Quando você pega uma coluna com dados$idade, o resultado é um vetor. Um vetor é uma sequência de valores.

Código em R

```
idades <- c(22, 35, 41, 58)
```

A função c() significa “combine”. Ela junta valores em um vetor.

Você pode calcular:

Código em R

```
mean(idades)
sd(idades)
min(idades)
max(idades)
```

Essas funções aparecem de novo nos capítulos de descritivas.

3.11 Tipos de variáveis

O R trata variáveis de formas diferentes. Algumas são numéricas. Outras são texto. Outras são categorias.

Código em R

```
class(dados$idade)
class(dados$genero)
class(dados$burnout)
```

Uma variável numérica pode ter média. Uma variável categórica não. Faz sentido calcular média de idade, mas não faz sentido calcular média de genero.

Essa distinção é essencial porque cada tipo de variável pede uma análise diferente.

3.12 Valores ausentes: o famoso NA

NA significa valor ausente. Não é zero. Não é branco. Não é erro. É ausência de informação.

Se uma pessoa não respondeu uma pergunta, esse valor pode aparecer como NA.

Para contar quantos valores ausentes existem em uma variável:

Código em R

```
sum(is.na(dados$idade))
```

Para contar no banco inteiro:

Código em R

```
sum(is.na(dados))
```

Muitos erros de iniciantes acontecem porque o R, por padrão, não ignora NA em algumas funções.

Código em R

```
mean(dados$demanda)
```

Se demanda tiver algum NA, o resultado será NA. Use:

Código em R

```
mean(dados$demanda, na.rm = TRUE)
```

3.13 Uma linha de código bem lida

Veja esta linha:

Código em R

```
media_burnout <- mean(dados$burnout, na.rm = TRUE)
```

Leia assim:

Exemplo

```
Crie um objeto chamado media_burnout.
Esse objeto recebe a média da coluna burnout dentro do
banco dados.
Ao calcular, ignore valores ausentes.
```

A maior mudança para aprender R é parar de olhar para o código como símbolo e começar a ler como frase.

3.14 Mini exercício

Crie um script e rode:

Código em R

```r
notas <- c(7, 8, 9, NA, 10)

mean(notas)
mean(notas, na.rm = TRUE)

media_notas <- mean(notas, na.rm = TRUE)
media_notas
```

Depois responda: por que a primeira média deu NA e a segunda não?

3.15 Erros comuns deste capítulo

O primeiro erro comum é esquecer parênteses em funções. mean é o nome da função. mean() chama a função.

O segundo é escrever nomes diferentes sem perceber. Para o R, Dados, dados e DADOS são três nomes diferentes.

O terceiro é usar vírgula decimal. Em R, número decimal usa ponto:

Exemplo

```
3.5
```

Não use:

Exemplo

```
3,5
```

No R, vírgula separa argumentos dentro de funções.

3.16 Onde está o script completo

Este capítulo conversa com a lógica de todos os scripts. Para praticar com o banco da toolbox, use:

Exemplo

```
scripts/01_carregar_banco.R
```

3.17 Como ler uma função desconhecida

Você não precisa conhecer todas as funções. Precisa saber investigar uma função. Quando encontrar algo como:

Código em R

```
lm(burnout ~ demanda, data = dados)
```

faça três perguntas:

Qual é o nome da função?

O que está dentro dos parênteses?

O resultado foi salvo em algum objeto?

No exemplo, a função é lm. Dentro dos parênteses há uma fórmula e o argumento data. Se a linha for:

Código em R

```
modelo <- lm(burnout ~ demanda, data = dados)
```

O resultado foi salvo em modelo.

Para pedir ajuda:

Exemplo

```
?lm
```

A documentação pode parecer difícil no começo, mas procure exemplos no final da página de ajuda. Muitas vezes, eles são mais úteis do que a definição formal.

3.18 A diferença entre salvar e imprimir

Quando você roda:

Código em R

```
mean(dados$idade, na.rm = TRUE)
```

O R calcula e imprime a média no console. Mas o resultado não fica salvo com nome.

Quando você roda:

Código em R

```
media_idade <- mean(dados$idade, na.rm = TRUE)
```

O R calcula e guarda o resultado no objeto media_idade. Ele pode não imprimir nada no console, mas o valor está salvo.

Para ver:

Exemplo

```
media_idade
```

Isso é importante porque muitos comandos no R só mostram algo se você pedir. Silêncio não significa erro. Às vezes significa que o objeto foi criado com sucesso.

3.19 O que é uma fórmula no R

A fórmula aparece em testes e modelos:

Exemplo

```
burnout ~ demanda
burnout ~ grupo
burnout ~ demanda + apoio_social
```

Leia o ~ como “em função de”.

Exemplo

```
burnout em função de demanda
burnout em função de grupo
burnout em função de demanda e apoio social
```

Essa fórmula aparece em teste t, ANOVA e regressão. Aprender a lê-la resolve uma parte grande da sintaxe estatística do R.

3.20 Pequenas regras que evitam grandes erros

Uma boa prática é rodar o código em blocos pequenos. Se você seleciona 80 linhas e aparece erro, pode ser difícil saber onde aconteceu. Se roda de 5 em 5 linhas, encontra o problema mais rápido.

Outra prática é conferir o resultado de cada criação. Criou dados? Rode head(dados). Criou modelo? Rode summary(modelo). Criou media_burnout? Rode media_burnout.

Essas checagens parecem lentas no começo, mas economizam tempo.

3.21 Filtrando linhas com comparações

A comparação da seção 3.8 produz uma sequência de TRUE e FALSE. Essa sequência pode ser usada para escolher linhas do banco. É assim que você analisa apenas um subgrupo.

Código em R

```
dados_mulheres <- dados[dados$genero == "Mulher", ]
nrow(dados_mulheres)
```

A leitura é: dentro de dados, pegue as linhas em que genero é igual a Mulher, e todas as colunas. A vírgula antes do colchete de fechamento é obrigatória. Antes dela ficam as linhas; depois dela, as colunas. Deixar essa parte vazia significa pegar todas as colunas.

Sempre confira o resultado com nrow(). Se o novo banco tem zero linhas, provavelmente a categoria foi escrita de forma diferente do que está no banco. Confira as categorias com table(dados$genero).

Outra forma de escrever o mesmo filtro é a função subset():

Código em R

```
dados_mulheres <- subset(dados, genero == "Mulher")
```

As duas formas produzem o mesmo resultado. Use a que parecer mais clara para você.

Capítulo 4 — O banco simulado: nossa base de treino

Neste capítulo

- Por que usar um banco simulado
- O que existe no banco
- Linha, coluna e célula
- Variáveis contínuas e categóricas
- O que significa cada variável
- Carregando o banco
- O que observar depois de carregar
- Cuidado: banco carregado não significa banco correto
- Missing no banco simulado
- O banco simulado não é evidência científica
- Mini exercício
- Erros comuns deste capítulo
- Onde está o script completo
- Dicionário de dados
- Como comparar o banco simulado com seu banco real
- Criando uma variável simples
- Não altere dados brutos sem necessidade
- Quando o CSV vem do Excel em português

4.1 Por que usar um banco simulado

Um livro para iniciantes precisa de um banco que funcione sempre. Se cada capítulo usa um exemplo diferente, a leitora aprende uma análise,

mas não aprende o fluxo. O banco simulado resolve isso: é uma base simulada, sem dados reais de pessoas, criada apenas para treino.

Usar banco simulado também reduz ansiedade. Você pode errar sem medo. Pode apagar, recriar, modificar, rodar de novo. Nada disso afeta dados reais.

4.2 O que existe no banco

A toolbox iniciante usa um banco principal chamado:

Exemplo

```
dados/dados_simulados.csv
```

Ele tem variáveis suficientes para as análises básicas deste livro:

Exemplo

```
id
genero
idade
escolaridade
grupo
demanda
apoio_social
recursos
burnout
burnout_caso
burnout_caso_bin
saude_geral
intencao_sair
```

Essas variáveis foram escolhidas porque representam situações comuns em pesquisa aplicada: características demográficas, variáveis psicossociais, desfechos contínuos e um desfecho binário.

4.3 Linha, coluna e célula

Em um banco de dados, cada linha representa uma observação. Neste livro, pense em cada linha como uma pessoa. Cada coluna representa uma variável. Uma célula é o cruzamento entre uma pessoa e uma variável.

Por exemplo: a idade da pessoa 17 é uma célula. O gênero de todas as pessoas é uma coluna. Todas as informações da pessoa 17 formam uma linha.

Essa distinção parece simples, mas ajuda muito quando você interpreta erros. Se o R diz que a coluna burnout não foi encontrada, ele está procurando uma variável. Se ele diz que há 1200 linhas, está falando do número de observações.

4.4 Variáveis contínuas e categóricas

No banco, algumas variáveis são contínuas. Elas têm valores numéricos com muitas possibilidades, como idade, demanda, apoio_social, recursos, burnout e intencao_sair.

Outras variáveis são categóricas. Elas representam grupos ou categorias, como genero, escolaridade, grupo e burnout_caso.

Essa diferença determina a análise. Para uma variável contínua, você pode calcular média e desvio-padrão. Para uma variável categórica, você calcula frequência e porcentagem.

4.5 O que significa cada variável

id é o identificador da pessoa. Ele não deve ser usado como variável de análise. Serve para diferenciar uma linha da outra.

genero representa uma categoria simulada. Não deve ser interpretado como descrição de uma população real.

idade é a idade simulada em anos.

escolaridade é o nível educacional simulado.

grupo é uma variável de grupo para comparação. Pode representar uma condição, setor ou categoria de trabalho.

demanda representa demandas do trabalho. Valores maiores indicam maior demanda.

apoio_social representa apoio recebido no contexto de trabalho. Valores maiores indicam maior apoio.

recursos representa recursos disponíveis para lidar com o trabalho.

burnout é um escore contínuo simulado. Valores maiores indicam mais burnout.

burnout_caso é uma versão categórica: caso ou não caso.

burnout_caso_bin é a mesma ideia em formato 0/1, útil para regressão logística.

saude_geral é um escore simples de saúde geral.

intencao_sair representa intenção de sair do emprego.

4.6 Carregando o banco

O código básico é:

Código em R

```
dados <- read.csv("dados/dados_simulados.csv")
```

Depois de carregar, confira:

Código em R

```
dim(dados)
names(dados)
head(dados)
```

Essas três linhas devem virar hábito. Carregar o banco não basta. Você precisa verificar se ele entrou do jeito esperado.

4.7 O que observar depois de carregar

Quando você roda:

Código em R

```
dim(dados)
```

O R mostra algo como:

Exemplo

```
1200   13
```

Isso significa 1200 linhas e 13 colunas.

Quando você roda:

Código em R

```
names(dados)
```

O R mostra os nomes das variáveis. Confira se eles batem com o que você espera. Erros de digitação em nome de coluna são uma das causas mais comuns de problemas.

Quando você roda:

Código em R

```
head(dados)
```

O R mostra as primeiras linhas. É uma inspeção rápida: os valores parecem plausíveis? Idade parece idade? Burnout parece número? Gênero parece categoria?

4.8 Cuidado: banco carregado não significa banco correto

Um banco pode carregar sem erro e ainda estar errado. Por exemplo: a variável idade pode ter sido lida como texto. Uma coluna pode ter vindo com nome estranho. O separador decimal pode ter sido interpretado de forma incorreta. A primeira linha pode ter sido lida como dado, não como cabeçalho.

Por isso, o primeiro passo de qualquer análise é sempre checar.

Código em R

```
str(dados)
summary(dados)
```

str() mostra a estrutura do banco. summary() mostra resumos rápidos das variáveis.

4.9 Missing no banco simulado

O banco simulado pode conter alguns valores ausentes. Isso é proposital. Bancos reais quase sempre têm missing. Aprender com uma base perfeitamente limpa criaria uma expectativa irreal.

Para contar missing:

Código em R

```
sum(is.na(dados))
```

Para contar por variável:

Código em R

```
colSums(is.na(dados))
```

Não entre em pânico quando aparecer missing. Primeiro descreva. Depois decida o que fazer.

4.10 O banco simulado não é evidência científica

Este ponto é importante: os resultados gerados com o banco simulado não dizem nada sobre o mundo real. Eles servem para aprender o procedimento. Não interprete diferenças, regressões ou gráficos do banco simulado como achados empíricos.

A pergunta não é “o que este banco prova?”. A pergunta é “eu entendi como rodar, interpretar e salvar a análise?”.

4.11 Mini exercício

Carregue o banco e responda:

Código em R

```
dados <- read.csv("dados/dados_simulados.csv")

dim(dados)
names(dados)
head(dados)
summary(dados)
```

Depois anote:

- Quantas linhas o banco tem?
- Quantas colunas?
- Quais variáveis são numéricas?
- Quais variáveis são categóricas?
- Há valores ausentes?

4.12 Erros comuns deste capítulo

O erro mais comum é o arquivo estar na pasta errada. Se o R diz que não conseguiu abrir o arquivo, confira se você abriu o projeto certo e se a pasta dados está no lugar certo.

Outro erro comum é copiar o nome do arquivo com diferença mínima. Para o R, dados_simulados.csv e dados_simulado.csv são dois nomes diferentes. Uma letra a menos basta para o arquivo não ser encontrado.

Também é comum tentar analisar antes de conferir. Não pule a inspeção inicial.

4.13 Onde está o script completo

Na toolbox:

Exemplo

```
scripts/01_carregar_banco.R
```

4.14 Dicionário de dados

Um dicionário de dados é uma tabela que explica o significado de cada variável. Ele deve responder perguntas como: o que a variável mede? Qual é o tipo? Quais valores são possíveis? Há código especial para missing?

Para o banco simulado, uma versão simples seria:

Exemplo

```
variável          significado
id                identificador da pessoa
genero            categoria simulada de gênero
idade             idade em anos
demanda           demanda percebida no trabalho
apoio_social      apoio social percebido
recursos          recursos disponíveis
burnout           escore contínuo de burnout
burnout_caso_bin  classificação binária de burnout
```

Em um projeto real, o dicionário de dados é indispensável. Sem ele, você corre o risco de interpretar uma variável ao contrário, usar escala errada ou confundir códigos.

4.15 Como comparar o banco simulado com seu banco real

Quando for adaptar a toolbox para dados reais, não tente trocar tudo de uma vez. Faça uma correspondência.

Exemplo:

Exemplo

```
banco simulado      meu banco real
idade           age
genero          sex
burnout         burnout_total
demanda         job_demands
apoio_social    supervisor_support
```

Depois, você pode adaptar o script com calma. O importante é entender que o script não “sabe” o significado das variáveis. Ele só reconhece nomes. Se o nome muda, o código precisa mudar.

4.16 Criando uma variável simples

Às vezes você precisa criar uma variável derivada. Por exemplo, uma classificação de idade:

Código em R

```
dados$idade_grupo <- ifelse(dados$idade < 40, "menos_40",
"40_mais")
```

Essa linha cria uma nova coluna chamada idade_grupo. Se idade for menor que 40, recebe menos_40. Caso contrário, recebe 40_mais.

Depois confira:

Código em R

```
table(dados$idade_grupo)
```

Sempre confira variáveis criadas. Erro em variável derivada é comum e pode contaminar análises posteriores.

4.17 Não altere dados brutos sem necessidade

Em projetos reais, separe dados brutos e dados processados. Dados brutos são como documento original. Não devem ser editados manualmente sem registro.

Uma estrutura segura:

Exemplo

```
dados/brutos/
dados/processados/
```

O script lê o bruto, cria variáveis, limpa, e salva um processado. Assim, qualquer mudança pode ser rastreada.

Para este livro, a toolbox é mais simples, mas o princípio continua: não perca a origem dos dados.

4.18 Quando o CSV vem do Excel em português

O read.csv() espera um arquivo separado por vírgulas, com ponto como separador decimal. O Excel em português salva CSV de outro jeito: separa as colunas com ponto e vírgula e usa vírgula como decimal. É o formato mais comum em bancos reais no Brasil e em Portugal.

O sintoma é fácil de reconhecer. Depois de carregar, o banco aparece com uma única coluna gigante, ou os números viram texto. Se isso acontecer, use read.csv2(), que foi feita exatamente para esse formato:

Código em R

```
dados <- read.csv2("dados/meu_banco.csv")
```

Também é possível indicar os separadores manualmente:

Código em R

```
dados <- read.csv("dados/meu_banco.csv", sep = ";", dec =
",")
```

Depois de carregar, rode str(dados) e confira se as variáveis numéricas entraram como número. Se o seu banco estiver em outro formato, como planilha do Excel (.xlsx) ou arquivo do SPSS (.sav), existem pacotes para isso: readxl para Excel e haven para SPSS. A instalação de pacotes está explicada no Capítulo 11.

Análise

Descritivas, comparações de grupos, regressão, gráficos e um exemplo de análise completa de ponta a ponta.

Capítulo 5 — Descritivas: conhecendo os dados antes de testar hipóteses

Neste capítulo

5.1 Por que descritivas importam

Descritivas são a primeira leitura do banco. Antes de perguntar se um grupo difere de outro, antes de rodar regressão, antes de fazer gráfico bonito, você precisa saber como os dados se comportam.

Muita gente trata descritivas como uma tabela burocrática. Isso é um erro. Descritivas respondem perguntas essenciais: quantas pessoas há no banco? Qual é a idade média? Quantas pessoas estão em cada grupo? Há muitos valores ausentes? Os escores parecem plausíveis? Alguma variável tem valores impossíveis?

Descritiva é diagnóstico. É onde você descobre problemas antes que eles estraguem análises posteriores.

5.2 Comece sempre pelo tamanho do banco

Código em R

```r
nrow(dados)
ncol(dados)
dim(dados)
```

Essas três linhas mostram o tamanho da base. Elas parecem simples, mas ajudam a detectar perda de casos. Se você esperava 1200 pessoas e o banco tem 1170, alguma coisa aconteceu.

5.3 Descritivas para variáveis numéricas

Para uma variável numérica, os resumos mais comuns são média, desvio-padrão, mediana, mínimo e máximo.

Código em R

```r
mean(dados$burnout, na.rm = TRUE)
sd(dados$burnout, na.rm = TRUE)
median(dados$burnout, na.rm = TRUE)
min(dados$burnout, na.rm = TRUE)
max(dados$burnout, na.rm = TRUE)
```

Essas funções são o núcleo das descritivas. Cada uma responde uma pergunta diferente.

A média é o valor médio. O desvio-padrão mostra o quanto as pessoas variam em torno da média. A mediana é o valor central, menos sensível a valores extremos. O mínimo e o máximo mostram os limites observados.

5.4 Média e mediana não dizem a mesma coisa

Imagine duas distribuições.

Na primeira, os valores são:

Exemplo

```
3, 3, 4, 4, 5
```

Média e mediana serão parecidas.

Na segunda:

Exemplo

```
3, 3, 4, 4, 40
```

A média aumenta muito por causa do valor 40. A mediana continua representando melhor o centro típico.

Por isso, quando há valores extremos, a mediana pode ser mais informativa. Em pesquisa aplicada, a média ainda é muito usada, mas a interpretação deve considerar a distribuição.

5.5 Descritivas para várias variáveis

Você pode repetir as funções para cada variável, mas isso fica cansativo. Um caminho simples em R base é usar summary():

Código em R

```
summary(dados)
```

Ele mostra resumos rápidos de todas as variáveis. Para começar, é excelente.

Para selecionar apenas algumas variáveis:

Código em R

```
summary(dados[c("idade", "demanda", "apoio_social",
"burnout")])
```

Esse comando pega apenas as colunas indicadas.

5.6 Descritivas para variáveis categóricas

Variáveis categóricas pedem frequência e porcentagem.

Código em R

```
table(dados$genero)
```

Para porcentagens:

Código em R

```
prop.table(table(dados$genero)) * 100
```

O mesmo vale para grupo, escolaridade e burnout_caso.

Código em R

```
table(dados$grupo)
prop.table(table(dados$grupo)) * 100
```

Frequência mostra quantas pessoas há em cada categoria. Porcentagem mostra o tamanho relativo de cada categoria.

5.7 Frequência com valores ausentes

Por padrão, table() pode esconder valores ausentes. Para mostrar missing:

Código em R

```
table(dados$genero, useNA = "ifany")
```

Use isso sempre pelo menos uma vez. Missing escondido é um problema silencioso.

5.8 Descritivas por grupo

Muitas vezes queremos saber a média de burnout em cada grupo.

Um jeito simples é:

Código em R

```
tapply(dados$burnout, dados$grupo, mean, na.rm = TRUE)
```

A tradução é: calcule a média de burnout separadamente para cada valor de grupo.

Para desvio-padrão:

Código em R

```
tapply(dados$burnout, dados$grupo, sd, na.rm = TRUE)
```

Esse tipo de resultado prepara o terreno para comparação de grupos. Antes de testar se há diferença, veja as médias.

5.9 Missing: o que contar

Conte missing no banco inteiro:

Código em R

```
sum(is.na(dados))
```

Conte missing por variável:

Código em R

```
colSums(is.na(dados))
```

E porcentagem por variável:

Código em R

```
colMeans(is.na(dados)) * 100
```

A porcentagem é importante porque 10 valores ausentes podem ser pouco em um banco de 5000 pessoas e muito em um banco de 50.

5.10 Valores impossíveis

Descritivas ajudam a encontrar valores impossíveis. Por exemplo, se idade mínima for 3 em uma amostra de trabalhadores, há problema. Se burnout varia de 0 a 100, mas aparece 999, há erro de codificação.

Sempre pergunte:

- O mínimo faz sentido?
- O máximo faz sentido?
- A média parece plausível?
- Há categorias estranhas?
- Há missing em variáveis essenciais?

5.11 Como escrever uma interpretação simples

Depois de rodar descritivas, não basta olhar números. Escreva uma frase.

Exemplo:

Exemplo

```
A base possui 1200 participantes. A idade média foi de
aproximadamente 41 anos. O escore de burnout apresentou
média mais alta em comparação às variáveis de apoio social
e recursos. A variável gênero não apresentou valores
ausentes, mas algumas variáveis contínuas tiveram pequena
quantidade de missing.
```

A frase não precisa ser sofisticada. Ela precisa mostrar que você entendeu o banco.

5.12 Como reportar descritivas em um texto

Um relatório simples pode dizer:

Exemplo

```
A amostra simulada incluiu 1200 participantes. A idade
média foi de M = 41,2 anos (DP = 9,9). A distribuição por
gênero e grupo foi descrita por frequências e porcentagens.
Os escores de demanda, apoio social, recursos e burnout
foram descritos por média, desvio-padrão, mínimo e máximo.
Valores ausentes foram inspecionados antes das análises
inferenciais.
```

Em um artigo, os detalhes costumam ir em uma tabela. Neste livro, o foco é entender o processo.

5.13 Mini exercício

Rode:

Exemplo

```
mean(dados$idade, na.rm = TRUE)
sd(dados$idade, na.rm = TRUE)

table(dados$grupo)
prop.table(table(dados$grupo)) * 100

colSums(is.na(dados))
```

Depois escreva um parágrafo curto descrevendo o banco.

5.14 Erros comuns deste capítulo

O primeiro erro é esquecer na.rm = TRUE. Se uma variável tem missing, a média pode retornar NA.

O segundo é calcular média de variável categórica. Se a variável é texto ou categoria, use frequência, não média.

O terceiro é interpretar o banco simulado como resultado real. Lembre-se: aqui o objetivo é aprender o fluxo.

5.15 Onde está o script completo

Na toolbox:

Exemplo

```
scripts/02_descritivas.R
```

5.16 Outliers: valores extremos

Um outlier é um valor muito distante dos outros. Nem todo outlier é erro. Uma pessoa pode realmente ter escore muito alto. O problema é que valores extremos podem influenciar média, desvio-padrão e regressão.

Para olhar visualmente:

Código em R

```r
boxplot(dados$burnout)
```

Para ver os maiores valores:

Código em R

```r
sort(dados$burnout, decreasing = TRUE)[1:10]
```

Para ver os menores:

Código em R

```r
sort(dados$burnout)[1:10]
```

Se aparecer valor impossível, como 999, investigue. Se for valor real, não exclua automaticamente.

5.17 Média ou mediana: como decidir

Use média e desvio-padrão quando a distribuição é aproximadamente simétrica e sem extremos muito fortes. Use mediana e intervalo interquartil quando a distribuição é muito assimétrica.

Na prática, muitos artigos em psicologia e saúde reportam média e desvio-padrão mesmo com alguma assimetria. Isso não é necessariamente errado, mas a decisão deve ser consciente.

Uma regra simples para iniciantes:

Exemplo

```
Se a diferença entre média e mediana for pequena (algo como
menos de 10% do desvio-padrão), a média costuma ser
suficiente.
Se a diferença for grande, olhe o histograma e considere
reportar mediana e intervalo interquartil.
```

5.18 Descritivas não são só números

Quando você descreve um banco, está contando uma história inicial sobre a amostra. Por exemplo:

Exemplo

```
A amostra é majoritariamente adulta, com distribuição
equilibrada entre grupos e níveis moderados de burnout. A
variável demanda apresenta variação suficiente para ser
usada em regressão. O missing é baixo e distribuído em
poucas variáveis.
```

Essa descrição mostra que você entendeu a base. Sem essa etapa, a análise vira execução mecânica.

5.19 Uma tabela descritiva manual

Você pode criar uma pequena tabela manual em R base:

Código em R

```
tabela_desc <- data.frame(
  variavel = c("idade", "burnout", "demanda"),
  media = c(
    mean(dados$idade, na.rm = TRUE),
    mean(dados$burnout, na.rm = TRUE),
    mean(dados$demanda, na.rm = TRUE)
  ),
  dp = c(
    sd(dados$idade, na.rm = TRUE),
    sd(dados$burnout, na.rm = TRUE),
    sd(dados$demanda, na.rm = TRUE)
  )
)

tabela_desc
```

Depois, você pode salvar:

Código em R

```
write.csv(tabela_desc, "resultados/descritivas.csv",
row.names = FALSE)
```

Esse é um exemplo de como transformar resultado do console em arquivo.

5.20 Checklist antes de sair das descritivas

Antes de ir para teste ou regressão, responda:

- O banco tem o número esperado de linhas?
- As variáveis principais têm missing?
- As médias fazem sentido?
- Os mínimos e máximos são plausíveis?
- As categorias estão escritas corretamente?
- Há grupos muito pequenos?
- Os gráficos mostram algo estranho?

Se uma dessas respostas preocupar, resolva antes de avançar.

Capítulo 6 — Comparando grupos sem complicar

Neste capítulo

- Exemplo de interpretação mais completa

6.1 A pergunta central

Comparar grupos significa perguntar se pessoas em grupos diferentes apresentam resultados diferentes. Por exemplo: o burnout médio é diferente entre grupos? A proporção de casos de burnout muda conforme o gênero? A intenção de sair varia entre três grupos de trabalho?

A comparação de grupos é uma das portas de entrada para estatística inferencial. Ela parece simples, mas exige clareza sobre dois pontos: qual é a variável de grupo e qual é a variável de resultado.

6.2 Duas perguntas antes de escolher o teste

Antes de rodar qualquer teste, responda:

A variável de resultado é numérica ou categórica?

A variável de grupo tem dois níveis ou mais de dois?

Essas duas respostas guiam o teste.

Se o resultado é numérico e há dois grupos, o teste mais comum é o teste t. Se o resultado é numérico e há três ou mais grupos, usamos ANOVA. Se as duas variáveis são categóricas, usamos qui-quadrado.

6.3 O teste t: duas médias

O teste t compara a média de uma variável numérica entre dois grupos.

Exemplo:

Código em R

```
t.test(burnout ~ genero, data = dados)
```

Essa fórmula pode ser lida assim:

Exemplo

```
Compare burnout entre os grupos de genero, usando o banco
dados.
```

O símbolo ~ separa resultado e grupo. À esquerda fica a variável que queremos comparar. À direita fica a variável que define os grupos.

6.4 Lendo o resultado do teste t

O R mostra várias informações. Para uma leitora iniciante, as mais importantes são:

as médias dos grupos;

o valor de t;

os graus de liberdade (df);

o valor de p;

o intervalo de confiança da diferença.

O p ajuda a decidir se a diferença observada é compatível com variação aleatória. Se p < .05, geralmente dizemos que há diferença estatisticamente significativa.

Mas não pare no p. Olhe as médias. Uma diferença pode ser significativa e pequena. Outra pode ser não significativa, mas relevante se a amostra for pequena.

6.5 Significativo não significa importante

Este é um ponto essencial. Significância estatística não é importância prática. Em amostras grandes, diferenças pequenas podem ficar significativas. Em amostras pequenas, diferenças grandes podem não ficar significativas.

Por isso, sempre observe a direção e o tamanho da diferença.

Pergunte:

Exemplo

```
Qual grupo teve média maior?
A diferença é pequena ou grande?
Faz sentido teoricamente?
```

6.6 Qui-quadrado: duas variáveis categóricas

Quando as duas variáveis são categóricas, usamos uma tabela de contingência.

Código em R

```
tabela <- table(dados$genero, dados$burnout_caso)
tabela
```

Depois:

Código em R

```
chisq.test(tabela)
```

O qui-quadrado pergunta se a distribuição de uma variável muda conforme a outra. Por exemplo: a proporção de casos de burnout é diferente entre categorias de gênero?

6.7 Lendo porcentagens no qui-quadrado

Antes de interpretar o teste, veja porcentagens:

Código em R

```
prop.table(tabela, margin = 1) * 100
```

margin = 1 calcula porcentagens por linha. Isso ajuda a responder: dentro de cada grupo, qual é a proporção de casos?

Sem porcentagens, a tabela de frequências pode enganar, especialmente quando os grupos têm tamanhos diferentes.

6.8 ANOVA: três ou mais médias

Quando há três ou mais grupos e o resultado é numérico, usamos ANOVA.

Código em R

```
modelo_anova <- aov(burnout ~ grupo, data = dados)
summary(modelo_anova)
```

A ANOVA pergunta se pelo menos uma média difere das outras. Ela não diz, sozinha, qual par difere.

Antes de rodar, olhe as médias:

Código em R

```
tapply(dados$burnout, dados$grupo, mean, na.rm = TRUE)
```

Depois rode a ANOVA.

6.9 O que fazer se a ANOVA for significativa

Se a ANOVA tem p < .05, você sabe que há alguma diferença entre grupos, mas não sabe onde. Para comparar pares, use um teste pós-hoc, como Tukey:

Código em R

```
TukeyHSD(modelo_anova)
```

O Tukey compara os pares de grupos e ajusta o valor de p para múltiplas comparações.

6.10 Quando usar testes não-paramétricos

Testes não-paramétricos são alternativas quando a variável numérica é muito assimétrica, há muitos outliers ou a amostra é pequena.

Para dois grupos:

Código em R

```
wilcox.test(burnout ~ genero, data = dados)
```

Para três ou mais grupos:

Código em R

```
kruskal.test(burnout ~ grupo, data = dados)
```

Mas cuidado: não troque automaticamente para não-paramétrico só porque a distribuição não é perfeitamente normal. Em amostras moderadas e grandes, testes paramétricos costumam ser robustos.

6.11 Antes do teste: sempre faça descritiva

Não rode comparação de grupos “no escuro”. Antes, veja:

Código em R

```
tapply(dados$burnout, dados$grupo, mean, na.rm = TRUE)
tapply(dados$burnout, dados$grupo, sd, na.rm = TRUE)
table(dados$grupo)
```

Essas linhas mostram média, desvio-padrão e tamanho dos grupos. Se um grupo tem apenas 5 pessoas e outro tem 800, a interpretação muda.

6.12 Como escrever o resultado do teste t

Um modelo simples:

Exemplo

> Foi realizado um teste t para comparar o escore de burnout entre os grupos. O grupo A apresentou média maior que o grupo B. A diferença foi estatisticamente significativa, **t**(gl) = valor, p = valor. Esse resultado sugere que, neste banco de treino, os grupos diferem quanto ao escore de burnout.

Substitua com os números do seu resultado. Não escreva apenas “deu significativo”. Explique a direção.

6.13 Como escrever o resultado do qui-quadrado

Exemplo

> Foi realizado um teste qui-quadrado para avaliar a associação entre gênero e classificação de burnout. As porcentagens indicaram maior proporção de casos em um dos grupos. O teste indicou associação estatisticamente significativa, $\chi^2(gl) = valor$, $p = valor$.

A porcentagem é tão importante quanto o teste.

6.14 Como escrever o resultado da ANOVA

Exemplo

> Foi realizada uma ANOVA para comparar o escore de burnout entre os grupos. O resultado indicou diferença estatisticamente significativa entre pelo menos dois grupos, **F**(gl1, gl2) = valor, p = valor. Comparações pós-hoc de Tukey foram usadas para identificar quais grupos diferiram.

6.15 Mini exercício

Rode:

Exemplo

```
t.test(burnout ~ genero, data = dados)

tabela <- table(dados$genero, dados$burnout_caso)
chisq.test(tabela)
prop.table(tabela, margin = 1) * 100

modelo_anova <- aov(burnout ~ grupo, data = dados)
summary(modelo_anova)
TukeyHSD(modelo_anova)
```

Depois escreva uma frase para cada análise.

6.16 Erros comuns deste capítulo

O primeiro erro é inverter a fórmula. Em burnout ~ grupo, burnout é o resultado e grupo é a variável de comparação. Não escreva grupo ~ burnout para teste t.

O segundo é olhar só o p-valor e esquecer as médias.

O terceiro é fazer muitos testes sem pensar. Cada teste deve responder uma pergunta.

6.17 Onde está o script completo

Na toolbox:

Exemplo

```
scripts/03_comparar_grupos.R
```

6.18 Pressupostos sem pânico

Muitos livros apresentam pressupostos de forma intimidadora. Para uma iniciante, o essencial é entender que testes estatísticos fazem algumas suposições sobre os dados. Quando essas suposições são muito violadas, o resultado pode ficar menos confiável.

Para o teste t e ANOVA, as preocupações básicas são:

os grupos são independentes?

a variável de resultado é numérica?

os grupos têm tamanho razoável?

a distribuição não é absurdamente assimétrica?

as variâncias são muito diferentes?

Você não precisa resolver tudo com testes formais. Comece com descritivas e gráficos.

6.19 Teste t de Welch como padrão

O R usa, por padrão, uma versão do teste t chamada Welch. Ela não exige que os grupos tenham variâncias iguais. Isso é bom. Em geral, aceite o padrão do R:

Código em R

```
t.test(burnout ~ genero, data = dados)
```

Não precisa acrescentar var.equal = TRUE a menos que você tenha motivo específico.

6.20 Tamanho de efeito em linguagem simples

O p-valor responde se a diferença é estatisticamente detectável. O tamanho de efeito responde se a diferença é pequena ou grande.

Não vamos aprofundar o cálculo de tamanho de efeito, mas a ideia precisa ficar clara: uma diferença pode ser estatisticamente significativa e pequena. Por isso, sempre olhe as médias.

Se um grupo tem média 51 e outro 52, a diferença é pequena, mesmo que o p-valor seja baixo em amostra enorme.

Se um grupo tem média 40 e outro 60, a diferença é grande, mesmo que uma amostra pequena gere p não significativo.

A toolbox calcula três medidas de tamanho de efeito e as salva no arquivo resultados/03_comparacao_grupos.csv. Vale saber ler cada uma.

O d de Cohen acompanha o teste t. Ele expressa a diferença entre as médias em unidades de desvio-padrão. Como referência aproximada, valores em torno de 0,2 indicam efeito pequeno; 0,5, médio; e 0,8, grande.

O phi acompanha o qui-quadrado de uma tabela 2 por 2. Ele varia de 0 a 1 e indica a força da associação entre as duas variáveis categóricas.

O eta² acompanha a ANOVA. Ele indica a proporção da variação do resultado explicada pela variável de grupo. Um eta² de 0,02 significa 2%.

6.21 Grupos independentes e grupos pareados

Um detalhe importante: o teste t que usamos até aqui assume grupos independentes. Isso significa que cada pessoa aparece em apenas um grupo.

Se as mesmas pessoas foram medidas antes e depois, o teste é pareado:

Código em R

```r
t.test(dados$burnout_pre, dados$burnout_pos, paired = TRUE)
```

As variáveis burnout_pre e burnout_pos não existem no banco simulado; o exemplo serve apenas para você reconhecer a sintaxe quando encontrar dados pareados. Não confunda. Comparar pessoas diferentes é uma coisa. Comparar a mesma pessoa em dois momentos é outra.

6.22 Comparações múltiplas

Quando você compara muitos grupos ou faz muitos testes, aumenta a chance de encontrar diferença por acaso. Por isso, pós-testes como Tukey ajustam comparações.

Para iniciante, a regra prática é:

Exemplo

```
Se a ANOVA tem três ou mais grupos e você vai comparar
pares, use TukeyHSD.
```

Não faça dezenas de testes t separadamente sem pensar. Isso aumenta o risco de conclusão falsa.

6.23 Como decidir entre teste e gráfico

Você não precisa escolher. Use os dois. O gráfico mostra padrão. O teste ajuda a avaliar evidência estatística.

Fluxo recomendado:

Exemplo

```
1. olhar médias por grupo
2. fazer boxplot
3. rodar teste
4. escrever interpretação juntando os três
```

Um resultado forte costuma ser coerente nessas três camadas.

6.24 Exemplo de interpretação mais completa

Em vez de:

Exemplo

```
O teste t foi significativo.
```

Escreva:

Exemplo

```
O grupo A apresentou média de burnout maior que o grupo B.
O teste t de Welch indicou diferença estatisticamente
significativa entre os grupos. Visualmente, o boxplot
também sugeriu maior concentração de valores altos no grupo
A. Como se trata de banco simulado, o resultado serve
apenas para treino de interpretação.
```

Isso mostra direção, teste, visualização e cautela.

Capítulo 7 — Regressão: entendendo relações entre variáveis

Neste capítulo

- A ideia central
- Resultado e preditor
- Regressão linear simples
- A equação da regressão
- Lendo o summary do modelo
- Coeficiente positivo e negativo
- Regressão não prova causalidade
- Regressão múltipla
- Por que controlar variáveis
- R^2: quanto o modelo explica
- Regressão com variável categórica
- Regressão logística: quando o resultado é 0/1
- Não precisa dominar logística de primeira
- Diagnóstico mínimo
- Como escrever um resultado de regressão linear
- Mini exercício
- Erros comuns deste capítulo
- Onde está o script completo
- Correlação antes da regressão
- Diferença entre associação bruta e ajustada
- Intercepto sem drama
- Predição não é explicação completa
- Variáveis demais em modelo pequeno

- Como comparar dois modelos de forma simples
- Interpretação cuidadosa da logística
- Um parágrafo de regressão mais completo
- Fatores e categoria de referência

7.1 A ideia central

Regressão é uma forma de estudar relação entre variáveis. Ela pergunta: quanto uma variável de resultado muda quando uma ou mais variáveis preditoras mudam?

Se comparação de grupos pergunta “as médias são diferentes?”, regressão pergunta “uma variável ajuda a explicar outra?”.

Exemplo: a demanda no trabalho está associada ao burnout? Apoio social reduz burnout? Idade, demanda e recursos juntos ajudam a prever burnout?

7.2 Resultado e preditor

Em regressão, a variável que queremos explicar é o resultado, também chamada de desfecho. As variáveis que usamos para explicar são preditores.

No exemplo:

Código em R

```
modelo <- lm(burnout ~ demanda, data = dados)
```

burnout é o resultado. demanda é o preditor.

A leitura é:

Exemplo

```
Modele burnout como função de demanda, usando o banco
dados.
```

7.3 Regressão linear simples

A regressão linear simples usa um preditor numérico para explicar um resultado numérico.

Código em R

```
modelo <- lm(burnout ~ demanda, data = dados)
summary(modelo)
```

A função lm() significa linear model. Ela estima uma reta que resume a relação entre demanda e burnout.

7.4 A equação da regressão

A ideia matemática é simples:

Exemplo

```
burnout previsto = intercepto + coeficiente * demanda
```

O intercepto é o valor previsto de burnout quando demanda é zero. Nem sempre tem interpretação substantiva, especialmente se zero não é um valor realista.

O coeficiente de demanda indica quanto o burnout muda quando a demanda aumenta uma unidade.

Se o coeficiente for 10, isso significa: para cada unidade a mais em demanda, o burnout previsto aumenta 10 pontos. Em regressão múltipla, essa leitura assume que as demais variáveis do modelo são mantidas constantes.

7.5 Lendo o summary do modelo

O summary(modelo) mostra muita coisa. Para começar, foque em:

Estimate: estimativa do coeficiente;

Std. Error: erro-padrão;

t value: estatística t;

Pr(>|t|): p-valor;

Multiple R-squared: proporção de variância explicada.

O Estimate é a informação substantiva principal. O p-valor diz se o coeficiente é estatisticamente diferente de zero. O R^2 diz quanto o modelo explica.

7.6 Coeficiente positivo e negativo

Se o coeficiente é positivo, quando o preditor aumenta, o resultado tende a aumentar.

Se o coeficiente é negativo, quando o preditor aumenta, o resultado tende a diminuir.

Exemplo:

Exemplo

```
Coeficiente de demanda = positivo
```

Maior demanda está associada a maior burnout.

Exemplo

```
Coeficiente de apoio_social = negativo
```

Maior apoio social está associado a menor burnout.

7.7 Regressão não prova causalidade

Este ponto precisa ser repetido. Regressão mostra associação, não prova causa. Se demanda está associada a burnout, isso não significa automaticamente que demanda causou burnout. Pode haver outras variáveis envolvidas.

Para falar em causalidade, você precisa de desenho de estudo, teoria, temporalidade e controle adequado de confundidores. Neste livro, use regressão como ferramenta de associação e predição básica.

7.8 Regressão múltipla

A regressão múltipla inclui mais de um preditor.

Código em R

```
modelo_multiplo <- lm(
  burnout ~ demanda + apoio_social + recursos + idade,
  data = dados
)

summary(modelo_multiplo)
```

Agora, cada coeficiente é interpretado “controlando pelos outros”.

Por exemplo, o coeficiente de demanda representa a associação entre demanda e burnout mantendo apoio social, recursos e idade constantes.

Essa frase é importante. Em regressão múltipla, o efeito de uma variável é ajustado pelas demais.

7.9 Por que controlar variáveis

Controlar variáveis ajuda a separar associações. Imagine que demanda e recursos estejam relacionados. Pessoas com alta demanda podem ter menos recursos. Se você olha apenas demanda, talvez misture os dois efeitos.

Ao incluir ambos no modelo, você pergunta: demanda ainda está associada a burnout depois de considerar recursos?

Isso não resolve todos os problemas causais, mas torna a análise mais informativa.

7.10 R^2: quanto o modelo explica

O R^2 varia de 0 a 1. Um R^2 de 0,25 significa que o modelo explica 25% da variação do resultado.

Em ciências sociais e saúde ocupacional, R^2 muito altos são raros. Um R^2 de 0,20 pode ser bastante relevante, dependendo da área.

Não interprete R^2 fora de contexto. Pergunte: para este tipo de fenômeno, esse nível de explicação é plausível?

7.11 Regressão com variável categórica

Você pode incluir variáveis categóricas como preditores:

Código em R

```
modelo <- lm(burnout ~ demanda + genero, data = dados)
summary(modelo)
```

O R cria comparações com uma categoria de referência. Se genero tem dois grupos, o coeficiente de uma categoria indica a diferença em relação à referência.

Por isso, a interpretação depende de qual categoria é referência.

7.12 Regressão logística: quando o resultado é 0/1

Se o resultado é binário, como burnout_caso_bin, não usamos lm(). Usamos glm() com família binomial.

Código em R

```
modelo_log <- glm(
  burnout_caso_bin ~ demanda + apoio_social + idade,
  data = dados,
  family = binomial
)

summary(modelo_log)
```

A regressão logística modela a probabilidade de um evento. O resultado bruto vem em log-odds, que não é intuitivo. Por isso, muitas vezes transformamos em odds ratio.

Código em R

```
exp(coef(modelo_log))
```

Um odds ratio maior que 1 indica aumento nas chances do evento. Menor que 1 indica redução.

7.13 Não precisa dominar logística de primeira

Para uma pessoa iniciante, o mais importante é saber quando usar logística: quando o resultado é binário. A interpretação detalhada pode vir com a prática.

Neste livro, a logística aparece como extensão básica, não como análise avançada.

7.14 Diagnóstico mínimo

Antes de confiar em uma regressão, faça perguntas simples:

- Os nomes das variáveis estão corretos?
- O número de casos usado no modelo faz sentido?
- Os sinais dos coeficientes fazem sentido?
- Algum coeficiente parece absurdo?
- Há muitos valores ausentes?

Você pode ver quantos casos entraram no modelo com:

Código em R

```
nobs(modelo_multiplo)
```

Se o número for muito menor que o esperado, provavelmente há missing nas variáveis usadas.

7.15 Como escrever um resultado de regressão linear

Modelo simples:

Exemplo

```
Foi ajustado um modelo de regressão linear para avaliar a
associação entre demanda e burnout. A demanda apresentou
associação positiva com burnout, indicando que maiores
níveis de demanda estão associados a maiores escores de
burnout. O modelo explicou aproximadamente X% da variação
no burnout.
```

Para modelo múltiplo:

Exemplo

```
Em um modelo com demanda, apoio social, recursos e idade,
demanda permaneceu positivamente associada ao burnout,
enquanto apoio social e recursos apresentaram associações
negativas. Isso sugere que a demanda se relaciona ao
burnout mesmo após ajuste pelas demais variáveis incluídas
no modelo.
```

7.16 Mini exercício

Rode:

Exemplo

```
modelo1 <- lm(burnout ~ demanda, data = dados)
summary(modelo1)

modelo2 <- lm(burnout ~ demanda + apoio_social + recursos +
idade,
              data = dados)
summary(modelo2)

modelo_log <- glm(burnout_caso_bin ~ demanda + apoio_social
+ idade,
                  data = dados,
                  family = binomial)
summary(modelo_log)
exp(coef(modelo_log))
```

Depois responda:

- O coeficiente de demanda é positivo ou negativo?
- O que acontece com burnout quando demanda aumenta?
- O modelo múltiplo explica mais que o simples?
- Na logística, algum odds ratio é maior que 1?

7.17 Erros comuns deste capítulo

O primeiro erro é interpretar coeficiente como causalidade. Escreva “associado a”, não “causa”, a menos que o desenho permita causalidade.

O segundo é esquecer que modelos com missing usam menos casos.

O terceiro é interpretar odds ratio como se fosse média. Odds ratio fala de chances, não de pontos no escore.

7.18 Onde está o script completo

Na toolbox:

Exemplo

```
scripts/04_regressao.R
```

7.19 Correlação antes da regressão

Antes de rodar uma regressão simples, muitas vezes é útil olhar a correlação entre duas variáveis numéricas.

Código em R

```
cor(dados$demanda, dados$burnout, use = "complete.obs")
```

A correlação varia de -1 a 1. Valores positivos indicam que as variáveis tendem a aumentar juntas. Valores negativos indicam que uma tende a aumentar quando a outra diminui. Valores próximos de zero indicam pouca relação linear.

Correlação e regressão simples estão relacionadas. A regressão dá uma equação de predição; a correlação resume a força e direção da relação.

7.20 Diferença entre associação bruta e ajustada

Uma regressão simples mostra associação bruta:

Código em R

```
lm(burnout ~ demanda, data = dados)
```

Uma regressão múltipla mostra associação ajustada:

Código em R

```
lm(burnout ~ demanda + apoio_social + recursos, data =
dados)
```

A palavra “ajustada” significa que a associação de demanda com burnout é estimada considerando as outras variáveis no modelo.

Se o coeficiente muda muito entre o modelo simples e o múltiplo, isso indica que as outras variáveis estavam relacionadas à associação inicial.

7.21 Intercepto sem drama

Muita gente se confunde com o intercepto. Em muitos modelos, ele não é substantivamente importante. Se o modelo é:

Exemplo

```
burnout ~ demanda
```

O intercepto é o burnout previsto quando demanda é zero. Se demanda varia de 1 a 5, demanda zero nem existe no banco. Nesse caso, o intercepto é só parte da equação.

Foque primeiro nos coeficientes dos preditores.

7.22 Predição não é explicação completa

A regressão pode prever valores, mas previsão não é explicação total. Um modelo pode prever razoavelmente bem e ainda não capturar todos os mecanismos. Em pesquisa, regressão deve ser guiada por pergunta teórica.

Não coloque variáveis no modelo apenas porque elas existem. Coloque porque fazem sentido para a pergunta.

7.23 Variáveis demais em modelo pequeno

Um modelo com muitos preditores e poucos casos pode ficar instável. Regra simples para iniciantes: não coloque dez preditores se você não consegue explicar por que cada um está ali.

Comece com modelos pequenos:

Exemplo

```
burnout ~ demanda
burnout ~ demanda + apoio_social
burnout ~ demanda + apoio_social + recursos
```

Veja como os resultados mudam.

7.24 Como comparar dois modelos de forma simples

Você pode olhar o R^2 de dois modelos:

Código em R

```r
modelo1 <- lm(burnout ~ demanda, data = dados)
modelo2 <- lm(burnout ~ demanda + apoio_social + recursos,
data = dados)

summary(modelo1)$r.squared
summary(modelo2)$r.squared
```

Se o segundo R^2 é maior, o modelo com mais variáveis explica mais variação. Mas não adicione variáveis apenas para aumentar R^2. O modelo precisa continuar interpretável.

7.25 Interpretação cuidadosa da logística

Na regressão logística, exp(coef(modelo)) gera odds ratios. Uma forma simples de ler:

Exemplo

```
OR > 1: aumenta as chances do evento
OR < 1: diminui as chances do evento
OR = 1: não altera as chances
```

Se demanda tem OR = 2, isso sugere que cada aumento de uma unidade em demanda está associado a aproximadamente o dobro das chances do evento, mantendo as outras variáveis constantes.

Mas cuidado: odds (chances) e probabilidade não são a mesma coisa. Chances são a razão entre o que acontece e o que não acontece (por exemplo, 3 para 1). Probabilidade é a fração do total (por exemplo, 0,75). Para iniciantes, basta entender direção e ideia geral: se OR > 1, o preditor aumenta as chances do evento; se OR < 1, diminui. A conversão exata para probabilidade depende da probabilidade de base e exige cálculo adicional.

7.26 Um parágrafo de regressão mais completo

Exemplo

```
Foi ajustado um modelo de regressão linear com burnout como
desfecho e demanda, apoio social e recursos como
preditores. A demanda apresentou associação positiva com
burnout, enquanto apoio social e recursos apresentaram
associações negativas. Isso indica que, no banco de treino,
maior demanda se relaciona a maior burnout, enquanto maior
apoio e mais recursos se relacionam a menor burnout. O
modelo deve ser interpretado como associação, não como
prova causal.
```

Esse parágrafo é melhor do que uma lista de coeficientes soltos.

7.27 Fatores e categoria de referência

Quando uma variável categórica entra em um modelo, o R a trata como fator. Um fator é uma variável de categorias com níveis definidos. Para transformar uma coluna de texto em fator:

Código em R

```r
dados$genero <- factor(dados$genero)
```

Para ver os níveis e a ordem deles:

Código em R

```r
levels(dados$genero)
```

O primeiro nível é a categoria de referência. Por padrão, o R ordena os níveis em ordem alfabética. No banco simulado, Homem vem antes de

Mulher, então o coeficiente de genero no modelo representa a diferença de Mulher em relação a Homem.

Para trocar a referência, use relevel():

Código em R

```
dados$genero <- relevel(dados$genero, ref = "Mulher")
```

O script scripts/04_regressao.R da toolbox faz a conversão com factor() antes de ajustar os modelos. A substância do resultado não muda quando a referência muda; muda apenas o ponto de comparação dos coeficientes.

Capítulo 8 — Gráficos básicos: vendo os dados antes de concluir

Neste capítulo

- Por que fazer gráficos
- Três gráficos suficientes para começar
- Histograma
- Como interpretar um histograma
- Boxplot
- Boxplot não mostra tudo
- Gráfico de dispersão
- Adicionando uma reta de regressão
- Gráfico bom tem rótulo claro
- Gráfico para explorar e gráfico para publicar
- Salvando gráficos
- Quando usar ggplot2
- Como comentar um gráfico
- Mini exercício
- Erros comuns deste capítulo
- Onde está o script completo
- A pergunta que o gráfico responde
- Gráficos também revelam erro de dados
- Cores: use com moderação
- Salvando com nomes informativos
- Interpretando visualmente sem exagerar

- Quando um gráfico contradiz o teste

8.1 Por que fazer gráficos

Gráficos ajudam a ver padrões que tabelas escondem. Uma média pode ser igual em dois grupos, mas a distribuição ser completamente diferente. Uma regressão pode ter coeficiente significativo, mas a relação pode ser curva. Um valor extremo pode distorcer uma conclusão sem aparecer claramente na tabela.

Gráfico é ferramenta de diagnóstico e comunicação, não decoração.

8.2 Três gráficos suficientes para começar

Para um primeiro livro de R, você não precisa dominar todos os tipos de visualização. Três gráficos já resolvem muita coisa:

histograma;

boxplot;

gráfico de dispersão.

O histograma mostra a distribuição de uma variável numérica. O boxplot compara distribuição entre grupos. O gráfico de dispersão mostra relação entre duas variáveis numéricas.

8.3 Histograma

Um histograma mostra quantas pessoas estão em faixas de valores.

Exemplo

```
hist(dados$burnout)
```

Esse é o gráfico mais simples em R base. Para melhorar um pouco:

Código em R

```
hist(
  dados$burnout,
  main = "Distribuição do burnout",
  xlab = "Burnout",
  ylab = "Frequência"
)
```

Observe o formato. A distribuição é simétrica? Tem cauda longa? Há valores extremos?

8.4 Como interpretar um histograma

Se o histograma tem um pico central e caudas semelhantes, a distribuição é aproximadamente simétrica. Se a cauda vai muito para a direita, há assimetria positiva. Se vai para a esquerda, assimetria negativa.

Se aparecem valores muito distantes dos demais, investigue. Pode ser um caso real extremo ou um erro de entrada.

8.5 Boxplot

O boxplot resume a distribuição por grupo.

Código em R

```
boxplot(burnout ~ grupo, data = dados)
```

A fórmula segue a mesma lógica de comparação de grupos: burnout por grupo.

Para melhorar:

Código em R

```
boxplot(
  burnout ~ grupo,
  data = dados,
  main = "Burnout por grupo",
  xlab = "Grupo",
  ylab = "Burnout"
)
```

A linha dentro da caixa é a mediana. A caixa mostra a região central dos dados. Pontos fora podem indicar outliers.

8.6 Boxplot não mostra tudo

Boxplot é útil, mas resumido. Ele não mostra exatamente quantas pessoas há em cada grupo nem a forma completa da distribuição. Por isso, use junto com descritivas e tamanhos de grupo.

Antes de interpretar visualmente:

Código em R

```
table(dados$grupo)
```

Um boxplot de um grupo com 20 pessoas e outro com 600 precisa ser lido com cuidado.

8.7 Gráfico de dispersão

O gráfico de dispersão mostra a relação entre duas variáveis numéricas.

Código em R

```
plot(dados$demanda, dados$burnout)
```

Melhorando rótulos:

Código em R

```
plot(
  dados$demanda,
  dados$burnout,
  xlab = "Demanda",
  ylab = "Burnout",
  main = "Relação entre demanda e burnout"
)
```

Se os pontos sobem da esquerda para a direita, a relação é positiva. Se descem, é negativa. Se estão espalhados sem padrão, a relação é fraca.

8.8 Adicionando uma reta de regressão

Código em R

```
modelo <- lm(burnout ~ demanda, data = dados)
plot(dados$demanda, dados$burnout)
abline(modelo)
```

A reta resume a tendência média. Ela não substitui a análise, mas ajuda a visualizar.

8.9 Gráfico bom tem rótulo claro

Um erro comum é deixar eixos com nomes técnicos ou abreviados. Um gráfico com eixo x e y não comunica bem. Use nomes compreensíveis.

Ruim:

Exemplo

```
x = demanda
y = burnout
```

Melhor:

Exemplo

```
Demanda no trabalho
Escore de burnout
```

O gráfico deve ser compreensível para uma pessoa que não abriu o script.

8.10 Gráfico para explorar e gráfico para publicar

Há dois tipos de gráfico: o gráfico de exploração e o gráfico de publicação.

O gráfico de exploração é rápido. Serve para você olhar os dados. Pode ser simples, sem muita estética.

O gráfico de publicação precisa ser limpo, legível e bem rotulado. Deve ter tamanho adequado, fonte legível e, quando necessário, legenda clara.

Neste livro, o foco é começar pelos gráficos de exploração e salvar versões simples para relatório.

8.11 Salvando gráficos

Em R base, você pode salvar uma figura abrindo um arquivo, desenhando e fechando:

Código em R

```
png("resultados/figuras/histograma_burnout.png",
    width = 1200, height = 800)

hist(
  dados$burnout,
  main = "Distribuição do burnout",
  xlab = "Burnout",
  ylab = "Frequência"
)

dev.off()
```

A função png() abre um arquivo. O gráfico é desenhado dentro dele. dev.off() fecha e salva.

Esquecer dev.off() é erro comum. Se o arquivo não aparece direito, confira isso.

8.12 Quando usar ggplot2

O ggplot2 é excelente e muito usado, mas adiciona uma camada de sintaxe. Para um primeiro contato, R base é suficiente. Depois que a lógica dos gráficos estiver clara, ggplot2 se torna uma ótima próxima etapa. Quando chegar essa hora, o caminho de instalação de pacotes está descrito no Capítulo 11.

A toolbox iniciante prioriza simplicidade e usa gráficos básicos. Isso reduz a chance de erro de pacote e deixa a aprendizagem mais tranquila.

8.13 Como comentar um gráfico

Não diga apenas “o gráfico mostra a distribuição”. Seja específico:

Exemplo

```
O histograma sugere que o escore de burnout está
aproximadamente concentrado em torno de valores
intermediários, com alguns valores mais altos. O boxplot
indica diferenças visuais entre grupos, mas a interpretação
formal deve ser feita junto com as médias e os testes
estatísticos.
```

Gráfico sem comentário vira imagem solta. Comentário transforma gráfico em evidência.

8.14 Mini exercício

Rode:

Código em R

```r
hist(dados$burnout)
boxplot(burnout ~ grupo, data = dados)
plot(dados$demanda, dados$burnout)
modelo <- lm(burnout ~ demanda, data = dados)
abline(modelo)
```

Depois responda:

O histograma parece simétrico?

Algum grupo parece ter burnout mais alto?

A relação entre demanda e burnout parece positiva, negativa ou fraca?

A reta ajuda a enxergar o padrão?

8.15 Erros comuns deste capítulo

O primeiro erro é interpretar gráfico sem olhar o tamanho dos grupos.

O segundo é salvar gráfico sem conferir se o arquivo foi criado.

O terceiro é exagerar na estética antes de entender a mensagem. Primeiro clareza, depois beleza.

8.16 Onde está o script completo

Na toolbox:

Exemplo

```
scripts/05_graficos_basicos.R
```

8.17 A pergunta que o gráfico responde

Antes de fazer gráfico, pergunte: o que quero mostrar?

Se quer mostrar distribuição de uma variável, use histograma. Se quer comparar grupos, use boxplot. Se quer mostrar relação entre duas variáveis numéricas, use dispersão.

Evite escolher gráfico porque parece bonito. Escolha porque responde uma pergunta.

8.18 Gráficos também revelam erro de dados

Um histograma pode revelar um valor impossível. Um boxplot pode revelar um grupo com pouquíssimas observações. Um scatterplot pode mostrar que uma relação não é linear.

Por isso, gráficos devem aparecer cedo, antes da conclusão.

8.19 Cores: use com moderação

Cor deve ter função. Se há dois grupos, cor pode ajudar. Se o eixo já mostra os grupos, talvez a cor seja redundante.

Em gráficos de iniciante, menos é mais. Prefira rótulos claros, eixos legíveis e poucos elementos.

8.20 Salvando com nomes informativos

Nomes bons para figuras:

Exemplo

```
histograma_burnout.png
boxplot_burnout_por_grupo.png
scatter_demanda_burnout.png
```

Nomes ruins:

Exemplo

```
grafico1.png
novo.png
final_final.png
```

O nome do arquivo deve dizer o que há na figura.

8.21 Interpretando visualmente sem exagerar

Use linguagem cautelosa:

Exemplo

```
O gráfico sugere...
O padrão visual indica...
A distribuição parece...
```

Não escreva que o gráfico “prova” uma relação. Ele ajuda a observar padrões. A interpretação formal vem junto com descritivas e testes.

8.22 Quando um gráfico contradiz o teste

Às vezes o p-valor é significativo, mas o gráfico mostra grupos muito parecidos. Isso pode acontecer com amostra grande. Às vezes o gráfico mostra diferença visual, mas o teste não é significativo. Isso pode acontecer com amostra pequena ou alta variabilidade.

Nesses casos, não escolha o resultado que você prefere. Descreva a tensão:

Exemplo

> Apesar de uma diferença visual entre os grupos, o teste não indicou diferença estatisticamente significativa.

ou:

Exemplo

> Embora o teste tenha sido significativo, a diferença visual entre os grupos parece pequena.

Esse tipo de frase mostra maturidade analítica.

Capítulo 9 — Rodando uma análise completa do começo ao fim

Neste capítulo

- Por que este capítulo existe
- A ordem básica
- Um roteiro mínimo em R
- Por que conferir antes de analisar
- O que salvar
- Console não é relatório
- Como organizar o script
- Como saber se o resultado faz sentido
- Quando um resultado contradiz sua expectativa
- O script mestre da toolbox
- Mini exercício
- Erros comuns deste capítulo
- Onde está o script completo
- Criando um rastro de auditoria
- Rodar do zero
- A diferença entre análise exploratória e análise final
- Uma rotina de revisão

9.1 Por que este capítulo existe

Até aqui, vimos partes separadas: carregar dados, descrever, comparar grupos, regressão e gráficos. Na prática, uma análise real não acontece em pedaços isolados. Você precisa juntar tudo em um fluxo.

Este capítulo mostra a ordem recomendada. O objetivo é que você saiba o que fazer quando abre um banco pela primeira vez.

9.2 A ordem básica

Um fluxo simples pode seguir esta sequência:

abrir o projeto;
carregar o banco;
conferir linhas, colunas e nomes;
verificar tipos de variáveis;
contar missing;
fazer descritivas;
fazer gráficos exploratórios;
rodar testes ou modelos;
salvar resultados;
escrever interpretação.

Essa ordem evita muitos erros. O erro mais comum é pular direto para o teste.

9.3 Um roteiro mínimo em R

Código em R

```r
# 1. Carregar banco
dados <- read.csv("dados/dados_simulados.csv")

# 2. Conferir estrutura
dim(dados)
names(dados)
head(dados)
summary(dados)

# 3. Missing
colSums(is.na(dados))

# 4. Descritivas
mean(dados$burnout, na.rm = TRUE)
sd(dados$burnout, na.rm = TRUE)
table(dados$grupo)

# 5. Comparação de grupos
t.test(burnout ~ genero, data = dados)

# 6. Regressão
modelo <- lm(burnout ~ demanda + apoio_social + recursos,
             data = dados)
summary(modelo)

# 7. Gráfico
plot(dados$demanda, dados$burnout)
abline(lm(burnout ~ demanda, data = dados))
```

Esse não é o script completo da toolbox. É um mapa mental do fluxo.

9.4 Por que conferir antes de analisar

Imagine que a variável burnout veio como texto. A regressão pode falhar. Imagine que o grupo tem uma categoria escrita como Controle e outra como controle. O R vai entender como duas categorias diferentes. Imagine que uma variável tem muitos missing. O modelo pode usar menos casos do que você esperava.

Esses problemas são detectados antes, não depois.

9.5 O que salvar

Nem todo resultado precisa ser salvo em arquivo. Mas alguns devem ser:

- resumo do banco;
- descritivas principais;
- resultados dos testes;
- coeficientes da regressão;
- gráficos.

A toolbox salva arquivos dentro de resultados/. Isso é importante porque permite conferir depois, sem depender do console.

9.6 Console não é relatório

O console mostra resultados, mas não é o local final da análise. Se um resultado é importante, salve ou copie para um documento de relatório.

Uma boa prática é manter três camadas:

Exemplo

```
script -> resultado salvo -> interpretação escrita
```

O script mostra como você fez. O resultado salvo guarda os números. A interpretação explica o que os números significam.

9.7 Como organizar o script

Um script iniciante pode ter blocos com comentários:

Exemplo

```
#
------------------------------------------------------------
-
# 1. Carregar dados
#
------------------------------------------------------------
-

#
------------------------------------------------------------
-
# 2. Conferir estrutura
#
------------------------------------------------------------
-

#
------------------------------------------------------------
-
# 3. Descritivas
#
------------------------------------------------------------
-

#
------------------------------------------------------------
-
# 4. Análises
#
------------------------------------------------------------
-

#
------------------------------------------------------------
-
# 5. Salvar resultados
#
------------------------------------------------------------
-
```

Isso deixa o script navegável. Quando der erro, você encontra o bloco mais rápido.

9.8 Como saber se o resultado faz sentido

Antes de aceitar qualquer resultado, faça perguntas de sanidade:

O número de casos bate com o esperado?

As médias estão em uma faixa possível?

As frequências somam o total?

O sinal do coeficiente faz sentido?

O gráfico confirma ou contradiz a tabela?

Essas perguntas não exigem conhecimento avançado. Exigem atenção.

9.9 Quando um resultado contradiz sua expectativa

Resultados inesperados acontecem. Não corrija o código para “dar certo”. Primeiro investigue.

Pergunte:

A variável está codificada corretamente?

A categoria de referência está correta?

Há missing reduzindo a amostra?

Há valores extremos?

A hipótese estava clara?

Às vezes o resultado inesperado é real. Às vezes é erro. O papel da análise é descobrir qual dos dois.

9.10 O script mestre da toolbox

A toolbox tem um script que roda todos os testes essenciais:

Código em R

```
source("scripts/99_rodar_todos_os_testes.R")
```

Ele é útil para verificar se tudo está funcionando. Mas para aprender, não use apenas o script mestre. Abra cada script individual, leia os comentários e rode em partes.

O script mestre é teste de funcionamento. Os scripts individuais são material de aprendizagem.

9.11 Mini exercício

Crie um script chamado meu_fluxo_completo.R. Nele, escreva os blocos:

Exemplo

```
# 1. Carregar dados
# 2. Conferir estrutura
# 3. Descritivas
# 4. Comparação de grupos
# 5. Regressão
# 6. Gráfico
# 7. Interpretação anotada em comentários
```

Em cada bloco, coloque pelo menos uma linha de código. Rode do começo ao fim.

9.12 Erros comuns deste capítulo

O primeiro erro é rodar o script fora de ordem. Se você usa dados antes de criar dados, o R dará erro.

O segundo é limpar o ambiente e esquecer de carregar de novo o banco.

O terceiro é deixar resultados importantes apenas no console.

9.13 Onde está o script completo

Na toolbox:

Exemplo

```
scripts/99_rodar_todos_os_testes.R
```

9.14 Criando um rastro de auditoria

Um rastro de auditoria é a trilha que mostra como você chegou ao resultado. Em análises simples, isso pode ser apenas:

Exemplo

```
script usado
banco usado
resultados salvos
data da análise
```

Na prática, crie uma pequena nota no começo do script:

Exemplo

```
# Projeto: R sem medo - treino
# Banco: dados_simulados.csv
# Objetivo: testar fluxo básico de análise
# Data: 2026-05-22
```

Isso parece formalidade, mas ajuda quando há várias versões.

9.15 Rodar do zero

Um bom teste de qualidade é fechar o RStudio, abrir o projeto novamente e rodar o script do começo ao fim. Se funciona, o script está reprodutível.

Se só funciona porque objetos antigos ficaram no Environment, há problema. Por isso, evite depender de objetos que não são criados pelo script.

9.16 A diferença entre análise exploratória e análise final

Na exploração, você testa, olha, muda, aprende. Na análise final, você organiza e deixa apenas o fluxo necessário.

Não há problema em explorar. O problema é confundir exploração com resultado final. Depois de explorar, limpe o script, comente e deixe a versão final clara.

9.17 Uma rotina de revisão

Antes de considerar a análise pronta:

rode do zero;
confira se os arquivos aparecem em resultados;
abra os gráficos;
leia os resultados principais;
compare com sua interpretação escrita;
veja se os nomes de variáveis estão corretos;
salve uma versão final do script.

Essa rotina reduz muito a chance de erro bobo.

Comunicação e suporte

Escrever resultados sem copiar o console, identificar erros comuns, usar a toolbox e fechar o aprendizado com calma.

Capítulo 10 — Escrevendo resultados sem copiar o console

Neste capítulo

- O problema do console colado
- Três camadas de resultado
- Reportando descritivas
- Reportando comparação de grupos
- Reportando regressão
- Como evitar linguagem causal indevida
- Tabelas simples
- Arredondamento
- O que fazer quando o resultado não é significativo
- Comentários no script ajudam a escrever
- Mini exercício
- Erros comuns deste capítulo
- Onde está o script completo
- Métodos e resultados não são a mesma coisa
- Tabelas não substituem texto
- Cuidado com excesso de certeza
- Revisando uma frase de resultado
- Como lidar com muitos números

10.1 O problema do console colado

Muitos relatórios iniciantes parecem uma sequência de outputs colados do R. Isso não é uma boa escrita científica. Quem lê não quer decifrar o

console. Quer entender o que foi feito, quais foram os números principais e o que eles significam.

O console é para você. O texto é para quem lê.

10.2 Três camadas de resultado

Para cada análise, pense em três camadas:

procedimento: qual análise foi feita;

resultado numérico: estatística, p-valor, média, coeficiente;

interpretação: o que isso significa em linguagem clara.

Exemplo ruim:

Exemplo

```
O teste deu p = 0.003.
```

Exemplo melhor:

Exemplo

```
Foi realizado um teste t para comparar o burnout entre os
grupos. O grupo A apresentou média maior que o grupo B, e a
diferença foi estatisticamente significativa (p = 0.003).
```

O segundo exemplo informa o teste, a direção e a conclusão.

10.3 Reportando descritivas

Para descritivas, inclua N, média e desvio-padrão quando a variável for contínua.

Exemplo

```
A idade média foi de 41,2 anos (DP = 9,9). O escore médio
de burnout foi de 47,7 pontos (DP = 14,5).
```

Para categóricas, inclua frequência e porcentagem.

Exemplo

```
A categoria mais frequente foi o grupo Comparacao  (n =
443; 36,9%).
```

Não reporte todos os números no texto se eles já estão em uma tabela. Use o texto para destacar o essencial.

10.4 Reportando comparação de grupos

Para teste t:

Exemplo

> Foi realizado um teste t para comparar o burnout entre os grupos de gênero. O grupo X apresentou média maior que o grupo Y. A diferença foi estatisticamente significativa, **t**(gl) = valor, p = valor.

Para qui-quadrado:

Exemplo

> Foi realizado um teste qui-quadrado para avaliar a associação entre gênero e classificação de burnout. A proporção de casos foi maior no grupo X. O teste indicou associação estatisticamente significativa, χ^2(gl) = valor, p = valor.

Para ANOVA:

Exemplo

> Foi realizada uma ANOVA para comparar o burnout entre três grupos. O resultado indicou diferença estatisticamente significativa entre grupos, **F**(gl1, gl2) = valor, p = valor. Comparações pós-hoc foram usadas para identificar os pares que diferiram.

10.5 Reportando regressão

Para regressão, diga qual foi o resultado, quais preditores foram incluídos e como interpretar os coeficientes principais.

Exemplo

> Foi ajustado um modelo de regressão linear para avaliar se demanda, apoio social e recursos estavam associados ao burnout. A demanda apresentou associação positiva com burnout, enquanto apoio social e recursos apresentaram associações negativas. O modelo explicou X% da variação do burnout.

Se for regressão logística:

Exemplo

```
Foi ajustado um modelo de regressão logística para predizer
a classificação de burnout. Valores mais altos de demanda
estiveram associados a maiores chances de classificação
como caso de burnout.
```

10.6 Como evitar linguagem causal indevida

Não escreva:

Exemplo

```
A demanda causou burnout.
```

Escreva:

Exemplo

```
A demanda esteve associada ao burnout.
```

Ou:

Exemplo

```
Participantes com maior demanda apresentaram maior burnout.
```

A menos que o estudo tenha desenho causal adequado, prefira linguagem de associação.

10.7 Tabelas simples

Uma tabela de descritivas pode ter colunas como:

Exemplo

```
Variável | Média | DP | Mínimo | Máximo
```

Uma tabela de regressão pode ter:

Exemplo

```
Preditor | Coeficiente | Erro-padrão | p
```

No começo, não complique. O objetivo é apresentar os números essenciais com clareza.

10.8 Arredondamento

Use arredondamento consistente. Em geral:

médias e desvios-padrão: uma ou duas casas decimais;

p-valores: três casas decimais;

se p for muito pequeno: p < .001.

Evite:

Exemplo

```
M = 41.24432891
```

Prefira:

Exemplo

```
M = 41,2
```

Números excessivamente precisos dão aparência técnica, mas atrapalham a leitura.

10.9 O que fazer quando o resultado não é significativo

Resultado não significativo também é resultado. Não esconda.

Evite:

Exemplo

```
Não houve nada.
```

Prefira:

Exemplo

```
A comparação entre grupos não indicou diferença
estatisticamente significativa no escore de burnout (p
= .230). As médias foram próximas entre os grupos.
```

Se o p não é significativo, olhe também o tamanho da diferença. Pode haver ausência real de diferença ou falta de poder estatístico.

10.10 Comentários no script ajudam a escrever

Durante a análise, escreva comentários interpretativos no script:

Exemplo

```
# Interpretação:
# Demanda tem coeficiente positivo.
# Isso sugere que maior demanda está associada a maior
burnout.
```

Esses comentários viram rascunho do relatório. Você não precisa escrever tudo do zero no final.

10.11 Mini exercício

Escolha um resultado do teste t, um da ANOVA ou um da regressão. Escreva três versões:

uma versão muito curta;

uma versão adequada para relatório;

uma versão explicativa para alguém que não sabe R.

Compare as três. A versão final deve ser clara sem ser longa demais.

10.12 Erros comuns deste capítulo

O primeiro erro é copiar o output inteiro do R.

O segundo é reportar p-valor sem direção do efeito.

O terceiro é usar linguagem causal sem desenho causal.

O quarto é arredondar de formas diferentes em cada linha.

10.13 Onde está o script completo

Este capítulo não tem um script específico. Ele usa os resultados produzidos pelos scripts:

Exemplo

```
scripts/02_descritivas.R
scripts/03_comparar_grupos.R
scripts/04_regressao.R
```

10.14 Métodos e resultados não são a mesma coisa

Na seção de Métodos, você diz o que fez. Na seção de Resultados, você diz o que encontrou.

Métodos:

Exemplo

```
As variáveis contínuas foram descritas por média e desvio-padrão. Diferenças entre dois grupos foram avaliadas com teste t de Welch. Associações entre variáveis contínuas foram avaliadas por regressão linear.
```

Resultados:

Exemplo

```
O grupo A apresentou maior média de burnout que o grupo B. A regressão indicou associação positiva entre demanda e burnout.
```

Não misture demais. Quem lê precisa entender primeiro o procedimento e depois o achado.

10.15 Tabelas não substituem texto

Uma tabela apresenta números. O texto guia a leitura. Em geral, use o texto para destacar o padrão principal, não para repetir cada número.

Exemplo:

Exemplo

```
Como mostrado na Tabela 1, os grupos foram semelhantes em idade, mas diferiram no escore de burnout.
```

Depois, a tabela mostra detalhes.

10.16 Cuidado com excesso de certeza

Evite frases como:

Exemplo

```
Isso comprova que...
```

Prefira:

Exemplo

```
Esses resultados sugerem que...
Os achados são compatíveis com...
Observou-se associação entre...
```

Essa linguagem é mais adequada para análises observacionais e bancos de treino.

10.17 Revisando uma frase de resultado

Frase inicial:

Exemplo

```
Demanda deu significativo.
```

Versão melhor:

Exemplo

```
A demanda apresentou associação positiva e estatisticamente
significativa com burnout.
```

Versão ainda melhor:

Exemplo

```
A demanda apresentou associação positiva e estatisticamente
significativa com burnout, indicando que participantes com
maior demanda tendem a apresentar maiores escores de
burnout.
```

A boa escrita estatística explica o número em linguagem humana.

10.18 Como lidar com muitos números

Se o texto fica cheio de números, talvez eles pertençam a uma tabela. O texto deve ter os números necessários para sustentar a frase, mas não precisa carregar todos os detalhes.

Uma regra prática:

Exemplo

```
Texto: padrão principal.
Tabela: detalhes numéricos.
Figura: padrão visual.
```

Capítulo 11 — Erros comuns: como não travar quando o R reclama

Neste capítulo

- Erro não é fracasso
- Leia a mensagem inteira
- Object not found
- Could not find function
- Cannot open file
- Argument is not numeric
- NAs introduced by coercion
- Subscript out of bounds
- Erros de maiúsculas e minúsculas
- Erros de aspas
- Erros de vírgula
- Estratégia de depuração para iniciantes
- Use comentários para marcar problemas
- Mini exercício
- Onde está o script completo
- O método do menor exemplo
- Conferindo nomes com copiar e colar
- Recomeçar faz parte
- Quando pedir ajuda

11.1 Erro não é fracasso

Todo mundo que usa R recebe erros. Pessoas experientes não são pessoas que nunca erram. São pessoas que sabem investigar o erro com calma.

Quando o R mostra uma mensagem, ele está dizendo onde algo não funcionou. Às vezes a mensagem é clara. Às vezes é estranha. Mas quase sempre há uma pista.

11.2 Leia a mensagem inteira

O primeiro impulso é entrar em pânico quando aparece texto vermelho. Respire e leia. Procure palavras como:

Exemplo

```
object not found
could not find function
cannot open file
argument is not numeric
```

Essas frases indicam problemas comuns.

11.3 Object not found

Mensagem típica:

Saída do R

```
Error: object 'dados' not found
```

Significa que o R não encontrou um objeto chamado dados.

Possíveis causas:

- você não carregou o banco;
- você escreveu dado em uma linha e dados em outra;
- você abriu uma nova sessão e não rodou o script desde o começo;
- você apagou o objeto sem perceber.

Solução:

Código em R

```
dados <- read.csv("dados/dados_simulados.csv")
```

Depois confira se apareceu no Environment.

11.4 Could not find function

Mensagem típica:

Saída do R

```
Error: could not find function "read_csv"
```

Significa que o R não conhece essa função na sessão atual. Muitas vezes a função pertence a um pacote que não foi carregado.

Neste livro, usamos principalmente R base para reduzir esse problema. Mas se você usar funções de pacotes, precisa carregar o pacote com library().

Pacotes precisam de dois passos. O primeiro é instalar, uma única vez por computador, com install.packages(). O segundo é carregar, em toda sessão em que o pacote for usado, com library().

Código em R

```
install.packages("readr")
library(readr)
```

As mensagens de erro são diferentes e ajudam a saber qual passo faltou. Could not find function indica que o pacote não foi carregado na sessão. There is no package called indica que ele nem está instalado. A instalação exige conexão com a internet; o carregamento, não.

Exemplo:

Código em R

```
library(readr)
read_csv("dados.csv")
```

Ou use a função base:

Código em R

```
read.csv("dados.csv")
```

11.5 Cannot open file

Mensagem típica:

Saída do R

```
Error in file(file, "rt") : cannot open the connection
```

Geralmente significa que o R não encontrou o arquivo.

Verifique:

Código em R

```
getwd()
list.files()
list.files("dados")
```

getwd() mostra a pasta atual. list.files() mostra arquivos. Se o arquivo não aparece, o caminho está errado.

11.6 Argument is not numeric

Mensagem típica:

Saída do R

```
Error in mean.default(...) : argument is not numeric or
logical
```

Você tentou calcular média de algo que não é numérico.

Confira:

Código em R

```
class(dados$burnout)
```

Se veio como texto, talvez precise converter:

Código em R

```
dados$burnout <- as.numeric(dados$burnout)
```

Mas cuidado: se houver textos misturados, a conversão pode gerar NA. Sempre confira depois.

11.7 NAs introduced by coercion

Mensagem típica:

Exemplo

```
NAs introduced by coercion
```

Significa que o R tentou converter algo para número, mas alguns valores não puderam ser convertidos.

Exemplo:

Código em R

```
as.numeric(c("1", "2", "abc"))
```

O valor abc não vira número. O R transforma em NA.

11.8 Subscript out of bounds

Essa mensagem aparece quando você tenta acessar algo que não existe.

Por exemplo, pedir a coluna 20 de um banco que só tem 13 colunas.

Exemplo

```
dados[, 20]
```

Solução: confira dimensões e nomes.

Código em R

```
dim(dados)
names(dados)
```

11.9 Erros de maiúsculas e minúsculas

R diferencia maiúsculas e minúsculas.

Exemplo

```
dados
Dados
DADOS
```

São três nomes diferentes. Mantenha padrão. Recomendo nomes em minúsculas, com _ para separar palavras:

Exemplo

```
media_burnout
apoio_social
modelo_regressao
```

11.10 Erros de aspas

Texto precisa estar entre aspas.

Correto:

Código em R

```
dados$genero == "Mulher"
```

Incorreto:

Código em R

```
dados$genero == Mulher
```

Sem aspas, o R entende Mulher como nome de objeto, não como texto.

11.11 Erros de vírgula

Em R, vírgula separa argumentos. Decimal usa ponto.

Correto:

Exemplo

```
3.5
```

Incorreto:

Exemplo

```
3,5
```

A segunda forma não significa três vírgula cinco para o R.

11.12 Estratégia de depuração para iniciantes

Quando der erro:

leia a mensagem;
identifique a linha que falhou;
rode a linha anterior;

confira se os objetos existem;
confira nomes de colunas;
confira tipos de variáveis;
reduza o código ao menor exemplo possível.

Não tente corrigir tudo ao mesmo tempo. Mude uma coisa por vez.

11.13 Use comentários para marcar problemas

Quando encontrar um erro, escreva no script:

Código em R

```
# Erro encontrado:
# object 'dados' not found
# Causa: eu não tinha carregado o banco.
# Solução: rodei read.csv() antes.
```

Isso transforma erro em aprendizado.

11.14 Mini exercício

Crie erros de propósito:

Código em R

```
mean(dados$variavel_que_nao_existe)
media(dados$idade)
read.csv("arquivo_inexistente.csv")
```

Leia as mensagens. Tente explicar cada uma em português simples.

11.15 Onde está o script completo

A toolbox foi escrita para reduzir erros, mas os erros fazem parte do aprendizado. Use:

Exemplo

```
scripts/99_rodar_todos_os_testes.R
```

Se ele falhar, leia o arquivo:

Exemplo

```
resultados/00_status_testes.csv
```

11.16 O método do menor exemplo

Quando um código grande dá erro, reduza. Em vez de tentar consertar 50 linhas, encontre a menor linha que reproduz o erro.

Se a regressão falhou, teste primeiro:

Código em R

```
names(dados)
class(dados$burnout)
class(dados$demanda)
summary(dados$burnout)
summary(dados$demanda)
```

Depois rode o modelo simples:

Código em R

```
lm(burnout ~ demanda, data = dados)
```

Se isso funciona, o problema está em alguma variável adicional.

11.17 Conferindo nomes com copiar e colar

Se um nome de variável é longo, copie de names(dados) em vez de digitar de memória. Erros de uma letra são difíceis de enxergar.

Código em R

```
names(dados)
```

Depois copie exatamente o nome.

11.18 Recomeçar faz parte

Às vezes a sessão fica confusa. Você testou muitas coisas, criou muitos objetos, esqueceu o que mudou. Nesses casos, salve o script, feche o RStudio, abra de novo e rode desde o começo.

Se o script está bem escrito, ele recria tudo. Se não recria, descobrimos o que falta.

11.19 Quando pedir ajuda

Ao pedir ajuda, envie:

a mensagem de erro completa;

a linha que gerou erro;

o resultado de names(dados) se o erro envolve variável;

o resultado de str(dados) se o erro envolve tipo de variável;

o que você esperava que acontecesse.

Dizer apenas “não funciona” dificulta. Dizer “ao rodar esta linha, apareceu esta mensagem” ajuda muito.

Também ajuda saber onde procurar. A primeira parada é a ajuda do próprio R, com ?funcao. A segunda é a documentação oficial no site do CRAN. Depois vêm as comunidades: fóruns como o Posit Community e o Stack Overflow concentram respostas para a maioria dos erros comuns, e grupos como o R-Ladies, com capítulos no Brasil e em Portugal, oferecem encontros e materiais abertos em português.

Capítulo 12 — A toolbox: como usar sem se perder

Neste capítulo

- O papel da toolbox
- O que há na toolbox
- Scripts principais
- Como abrir corretamente
- Quando usar o script 99
- Como adaptar para seu banco real
- Não modifique o original sem cópia
- O que fazer se o script não roda
- O que os resultados salvos significam
- Mini exercício
- Erros comuns deste capítulo
- A toolbox não precisa crescer para ser útil
- O que significa autodetectar a pasta
- Como estudar um script da toolbox
- Transformando a toolbox em modelo para seus projetos
- Conclusão

12.1 O papel da toolbox

A toolbox é o lugar dos scripts completos. O livro explica. A toolbox executa. Essa separação deixa o livro mais legível e torna a prática mais organizada.

A versão iniciante da toolbox foi mantida enxuta de propósito. Ela não contém análises complexas. A ideia é que a primeira experiência com R seja estável, não assustadora.

12.2 O que há na toolbox

A estrutura principal é:

Exemplo

```
R_sem_medo_toolbox/
  dados/
  scripts/
  resultados/
  resultados/figuras/
```

A pasta dados contém o banco simulado. A pasta scripts contém os scripts completos. A pasta resultados recebe os arquivos gerados. As pastas resultados e figuras não vêm no arquivo baixado: elas são criadas automaticamente na primeira execução.

12.3 Scripts principais

Exemplo

```
00_setup_auto.R
01_carregar_banco.R
02_descritivas.R
03_comparar_grupos.R
04_regressao.R
05_graficos_basicos.R
99_rodar_todos_os_testes.R
```

O script 00_setup_auto.R ajuda a detectar a pasta da toolbox. O script 01_carregar_banco.R carrega e confere o banco. Os scripts 02 a 05 rodam as análises básicas. O script 99 roda tudo.

12.4 Como abrir corretamente

Abra o arquivo .Rproj da toolbox. Não abra apenas um script solto. O projeto garante que o RStudio reconheça a pasta correta.

Depois, no RStudio, abra:

Exemplo

```
scripts/01_carregar_banco.R
```

Rode devagar. Leia os comentários.

12.5 Quando usar o script 99

Use o script mestre quando quiser verificar se tudo funciona:

Código em R

```
source("scripts/99_rodar_todos_os_testes.R")
```

Ele roda os blocos principais e salva resultados. Mas não use apenas ele para aprender. Se você só roda o script mestre, vê o resultado, mas não entende o caminho.

A melhor ordem de estudo é:

abrir 01_carregar_banco.R;
rodar linha por linha;
abrir 02_descritivas.R;
repetir;
continuar até 05_graficos_basicos.R;
só depois rodar 99_rodar_todos_os_testes.R.

12.6 Como adaptar para seu banco real

Quando você tiver seu banco real, não substitua imediatamente tudo. Primeiro, compare.

Pergunte:

- Meu banco tem uma coluna de identificação?
- Os nomes das variáveis são parecidos?
- Quais variáveis são contínuas?
- Quais são categóricas?
- Há desfecho binário?
- Há missing?

Depois, adapte nomes aos poucos.

Se a toolbox usa:

Código em R

```
dados$burnout
```

E seu banco usa:

Código em R

```
dados$exaustao_total
```

Você precisa trocar o nome no script.

12.7 Não modifique o original sem cópia

Antes de adaptar, faça uma cópia da toolbox ou do script. Por exemplo:

Exemplo

```
04_regressao_meu_banco.R
```

Assim, se algo quebrar, você ainda tem a versão original funcionando.

12.8 O que fazer se o script não roda

Primeiro, veja se você abriu o .Rproj. Depois, confira se a pasta dados existe e se o banco está lá.

Rode:

Código em R

```
list.files()
list.files("dados")
```

Se o arquivo aparece, o caminho está certo. Se não aparece, você está na pasta errada ou o arquivo não está onde deveria.

12.9 O que os resultados salvos significam

A pasta resultados guarda arquivos gerados pelos scripts. Eles servem para revisão. Por exemplo, uma tabela de descritivas salva em CSV pode ser aberta depois sem rodar tudo de novo.

A pasta figuras guarda os gráficos. Verifique visualmente se eles fazem sentido. Não assuma que um gráfico salvo está correto apenas porque o script rodou.

12.10 Mini exercício

Abra a toolbox e rode:

Código em R

```
source("scripts/01_carregar_banco.R")
source("scripts/02_descritivas.R")
```

Depois veja a pasta resultados. Quais arquivos foram criados? Abra pelo menos um.

12.11 Erros comuns deste capítulo

O primeiro erro é abrir o script sem abrir o projeto.

O segundo é mudar o nome do banco e esquecer de atualizar o script.

O terceiro é editar o script original sem cópia.

O quarto é rodar o script mestre sem entender os scripts individuais.

12.12 A toolbox não precisa crescer para ser útil

Uma toolbox iniciante boa é aquela que funciona, é pequena e pode ser entendida. Se ela inclui muitas análises avançadas, vira um novo problema.

A decisão editorial desta versão é manter a toolbox enxuta. Isso não empobrece o livro. Pelo contrário: protege o percurso da leitora.

12.13 O que significa autodetectar a pasta

O setup automático tenta descobrir onde a toolbox está no computador. Isso evita caminhos fixos como:

Exemplo

```
/Users/nome/Downloads/toolbox/...
```

Caminhos fixos quebram quando a pasta muda de lugar. A autodetecção torna a toolbox mais portátil.

Mesmo assim, o jeito mais seguro continua sendo abrir o .Rproj antes de rodar os scripts.

12.14 Como estudar um script da toolbox

Não leia o script inteiro de uma vez. Leia em camadas:

títulos e comentários;
blocos principais;
comandos dentro de cada bloco;
resultados salvos.

Depois rode um bloco por vez. Essa abordagem é mais lenta, mas ensina.

12.15 Transformando a toolbox em modelo para seus projetos

Quando você começar um projeto real, copie a estrutura:

Exemplo

```
dados/
scripts/
resultados/
resultados/figuras/
```

Depois copie a lógica dos scripts, não necessariamente as mesmas variáveis. A toolbox é um molde.

12.16 Conclusão

A toolbox é sua oficina. O livro é seu guia. Use os dois juntos: leia, rode, observe, salve, revise. Esse ciclo é o caminho para autonomia.

Capítulo 13 — Fechamento: o que você já consegue fazer

Neste capítulo

- O ponto de chegada deste volume
- O que ainda não precisa dominar
- Como continuar praticando
- Quando avançar para temas mais complexos
- Último conselho
- Uma frase para guardar
- O sinal de que você está aprendendo
- O que fazer com insegurança
- Uma meta prática
- Para continuar aprendendo

13.1 O ponto de chegada deste volume

Ao terminar este livro, você não precisa saber “programar em R” em sentido amplo. Essa não era a meta. A meta era conseguir realizar um fluxo básico de análise quantitativa com autonomia inicial.

Você deve conseguir:

abrir um projeto no RStudio;
carregar um banco;
conferir estrutura e nomes de variáveis;
identificar variáveis numéricas e categóricas;
calcular descritivas;
comparar grupos;
rodar regressões simples;
fazer gráficos básicos;

salvar resultados;

ler erros comuns.

Isso já é muito. Para uma pessoa que nunca usou R, esse conjunto representa uma mudança grande.

13.2 O que ainda não precisa dominar

Você não precisa memorizar todas as funções. Não precisa decorar todos os argumentos. Não precisa entender cada detalhe matemático de cada teste antes de começar a usar com responsabilidade.

Você precisa saber consultar, revisar e interpretar.

R é uma ferramenta de trabalho. Ninguém domina uma ferramenta apenas lendo instruções. Você domina usando em situações reais.

13.3 Como continuar praticando

Uma boa sequência de prática é:

rode todos os scripts da toolbox com o banco simulado;

mude uma variável de cada vez;

crie pequenas perguntas novas;

adapte um script para seu banco real;

compare resultados com expectativas;

escreva interpretações curtas.

Exemplo de pergunta nova:

Exemplo

```
Apoio social está associado à intenção de sair?
```

Tente responder com regressão simples:

Código em R

```
modelo <- lm(intencao_sair ~ apoio_social, data = dados)
summary(modelo)
```

Depois escreva uma frase interpretando.

13.4 Quando avançar para temas mais complexos

Você estará pronta para temas mais complexos quando conseguir explicar, sem olhar, o que fazem estas linhas:

Código em R

```
dados <- read.csv("dados/dados_simulados.csv")
summary(dados)
t.test(burnout ~ genero, data = dados)
modelo <- lm(burnout ~ demanda + apoio_social, data =
dados)
summary(modelo)
```

Se essas linhas ainda parecem misteriosas, continue no básico. Não há vergonha nisso. O básico bem feito é melhor que análise avançada mal compreendida.

13.5 Último conselho

Sempre que sentir que está perdida, volte para as perguntas básicas:

- Qual é meu banco?
- Qual é minha variável de resultado?
- Qual é meu preditor ou grupo?
- A variável é numérica ou categórica?
- Qual análise responde minha pergunta?
- O resultado faz sentido?

Essas perguntas são mais importantes do que qualquer pacote.

13.6 Uma frase para guardar

Não é preciso escrever código perfeito. É preciso escrever código claro, salvo, comentado e revisável.

Esse é o começo da autonomia.

13.7 O sinal de que você está aprendendo

Você está aprendendo quando consegue explicar o que uma linha faz, mesmo que ainda precise consultar a sintaxe.

Por exemplo, se você olha para:

Código em R

```
t.test(burnout ~ genero, data = dados)
```

E consegue dizer “isso compara burnout entre grupos de gênero”, você está avançando.

Se olha para:

Código em R

```
modelo <- lm(burnout ~ demanda + apoio_social, data =
dados)
```

E consegue dizer “isso cria um modelo de regressão para burnout usando demanda e apoio social”, você está no caminho certo.

13.8 O que fazer com insegurança

Insegurança é normal. R tem muitas funções, muitos pacotes e muitas formas de fazer a mesma coisa. Para não se perder, volte ao básico:

Exemplo

```
carregar -> conferir -> descrever -> analisar ->
interpretar -> salvar
```

Esse fluxo vale para projetos pequenos e grandes.

13.9 Uma meta prática

Antes de avançar para qualquer análise complexa, tente fazer isto sem ajuda:

abrir a toolbox;
carregar o banco;
gerar descritivas;
comparar dois grupos;
rodar uma regressão;
salvar um gráfico;
escrever um parágrafo de resultado.

Quando conseguir fazer isso com calma, você já terá uma base sólida.

13.10 Para continuar aprendendo

Quando o conteúdo deste livro estiver confortável, três caminhos gratuitos ajudam a seguir adiante.

O primeiro é o livro R for Data Science, de Hadley Wickham, Mine Çetinkaya-Rundel e Garrett Grolemund, disponível gratuitamente online e com tradução comunitária para o português em pt.r4ds.hadley.nz. Ele apresenta o tidyverse, incluindo o ggplot2 citado no Capítulo 8.

O segundo é a documentação oficial do R, no site do CRAN, que reúne manuais introdutórios e a página de ajuda de cada função.

O terceiro são as comunidades: os grupos e fóruns citados no Capítulo 11 também funcionam como espaço de aprendizagem contínua.

Capítulo 14 —
Glossário amigável

Análise descritiva. Resumo inicial dos dados. Inclui média, desvio-padrão, mediana, frequência, porcentagem, mínimo, máximo e missing.

ANOVA. Teste usado para comparar médias de três ou mais grupos.

Argumento. Informação passada para uma função. Em mean(x, na.rm = TRUE), na.rm = TRUE é um argumento.

Banco de dados. Tabela com linhas e colunas. Neste livro, linhas representam pessoas e colunas representam variáveis.

Boxplot. Gráfico que resume a distribuição de uma variável, geralmente comparando grupos.

Coeficiente. Número estimado em uma regressão. Indica direção e magnitude da associação entre preditor e resultado.

Console. Área do RStudio onde o R executa comandos e mostra respostas.

Data frame. Tipo de objeto em R usado para tabelas de dados.

Desfecho. Variável de resultado que queremos explicar ou comparar.

Desvio-padrão. Medida de variação em torno da média.

Função. Ação que o R sabe executar. Exemplos: mean(), sd(), lm().

Histograma. Gráfico que mostra a distribuição de uma variável numérica.

Missing. Valor ausente. Em R, aparece como NA.

Objeto. Algo guardado com um nome no R. Pode ser número, texto, banco, modelo ou resultado.

p-valor. Número usado para avaliar compatibilidade dos dados com uma hipótese nula. Não mede importância prática.

Preditor. Variável usada para explicar ou prever um resultado.

Projeto. Pasta organizada do RStudio para guardar dados, scripts e resultados.

Regressão linear. Modelo que avalia associação entre uma variável numérica de resultado e um ou mais preditores.

Regressão logística. Modelo usado quando o resultado é binário, como 0/1 ou caso/não caso.

Script. Arquivo onde você escreve e salva código R.

Variável categórica. Variável formada por grupos ou categorias, como gênero, grupo ou escolaridade.

Variável contínua. Variável numérica com muitos valores possíveis, como idade, demanda ou burnout.

Apêndice — Respostas dos mini exercícios

As respostas a seguir cobrem os mini exercícios de cada capítulo. Os valores numéricos foram calculados com o banco simulado distribuído na toolbox. Se o seu console mostrar números diferentes, confira se o banco carregado é o dados_simulados.csv original.

Capítulo 2. O objetivo é o processo, não o resultado. Ao rodar linha por linha, 2 + 2 devolve 4 no console e o objeto mensagem aparece no Environment. Digitar mensagem em uma linha própria imprime o texto guardado.

Capítulo 3. A primeira média é NA porque o vetor contém um valor ausente e, por padrão, mean() não o ignora. Com na.rm = TRUE, a média dos valores válidos é 8,5.

Capítulo 4. O banco tem 1200 linhas e 13 colunas. São numéricas: idade, demanda, apoio_social, recursos, burnout, saude_geral e intencao_sair, além de burnout_caso_bin (0/1) e id, que serve apenas de identificador. São categóricas: genero, escolaridade, grupo e burnout_caso. Há valores ausentes apenas em demanda (12), apoio_social (21) e recursos (18).

Capítulo 5. A idade tem média de 41,2 anos e desvio-padrão de 9,9. Os grupos têm tamanhos parecidos: Comparacao com 443 pessoas (36,9%), Controle com 379 (31,6%) e Intervencao com 378 (31,5%). Um parágrafo adequado menciona o número de participantes, uma medida de tendência central, a distribuição dos grupos e o missing.

Capítulo 6. O teste t indica diferença entre os gêneros: as mulheres apresentam média de burnout maior que os homens (49,3 contra 45,8; $p < 0,001$). O qui-quadrado indica associação entre gênero e classificação de burnout (36,3% de casos entre mulheres e 25,5% entre homens; $p < 0,001$). A ANOVA indica diferença entre grupos ($p < 0,001$), e o Tukey mostra que o grupo Intervencao difere dos outros dois, enquanto Controle e Comparacao não diferem entre si.

Capítulo 7. O coeficiente de demanda é positivo: quando a demanda aumenta, o burnout previsto aumenta. O modelo múltiplo explica mais que o simples (R^2 de 0,53 contra 0,36). Na logística, o odds ratio de demanda é maior que 1 e o de apoio social é menor que 1.

Capítulo 8. O histograma do burnout é aproximadamente simétrico, com um pico central. No boxplot, o grupo Intervencao aparece com valores mais baixos. A dispersão entre demanda e burnout sobe da esquerda para a direita, e a reta ajuda a enxergar essa tendência positiva.

Capítulo 9. Não há uma única resposta certa. O critério de qualidade é o script rodar do começo ao fim em uma sessão nova, com um bloco para cada etapa e comentários de interpretação no final.

Capítulo 10. Compare as três versões pelo que cada uma comunica. A versão curta informa só o número. A versão de relatório informa teste, direção e conclusão. A versão explicativa traduz o resultado para quem não conhece estatística. A do meio costuma ser a mais útil em textos acadêmicos.

Capítulo 11. A primeira linha não gera erro, e sim um aviso, argument is not numeric or logical, com resultado NA: a coluna pedida não existe e o R devolve um objeto vazio. A segunda gera could not find function, porque media não é uma função do R; a correta é mean(). A terceira gera cannot open the connection, porque o arquivo não existe na pasta.

Capítulo 12. Os dois scripts criam quatro arquivos em resultados: 01_resumo_banco.csv, 01_variaveis_obrigatorias.csv, 02_descritivas_continuas.csv e 02_frequencias_categoricas.csv.

Sobre a toolbox

Acesso à toolbox

Aponte a câmera do celular para o código abaixo ou acesse o endereço indicado para obter a toolbox (scripts, dados e materiais de apoio), em acesso aberto.

https://doi.org/10.17605/OSF.IO/J4HQU

Este livro foi escrito em par com a toolbox **R sem medo — toolbox iniciante**, um conjunto pequeno de scripts em R, um banco simulado para praticar e um dicionário das variáveis. A toolbox cobre o mesmo percurso do livro: carregar dados, descrever, comparar grupos, ajustar uma regressão simples e gerar gráficos básicos.

Cada script da toolbox corresponde a um trecho do livro. A leitora pode acompanhar o livro lendo o script ao lado, ou abrir um script primeiro e voltar ao capítulo quando precisar de contexto.

Para acessar, descompacte o arquivo da toolbox em uma pasta do seu computador, mantendo a estrutura interna. Dentro dela, abra o arquivo R_sem_medo_iniciante.Rproj com um duplo clique: o RStudio abre já apontando para a pasta certa. Para conferir se tudo funciona, rode no console:

Código em R

```
source("scripts/99_rodar_todos_os_testes.R")
```

Na primeira execução, as pastas resultados e resultados/figuras são criadas automaticamente. Não mova os scripts para fora da pasta nem renomeie a pasta dados.

Sobre a autora

Iara Teixeira é psicóloga e doutora em Psicologia Aplicada pela Universidade do Minho, em Portugal. Atualmente, é pesquisadora no Intersectionality Lab – European Research Center. Sua pesquisa situa-se no cruzamento entre saúde ocupacional, riscos psicossociais, gênero e interseccionalidade, e combina métodos qualitativos e quantitativos.

R sem medo nasce dessa experiência de pesquisa e ensino: a convicção de que qualquer pessoa pode aprender a analisar dados em R, sem jargão e sem medo, a partir de um percurso curto e prático.