Introdução à Estatística com R para Biólogos

Fernando Andrade

Sumário

Prefácio

Essa apostila foi criada para ser um guia abrangente e conceitual para pesquisadores de mestrado e doutorado na área de Ciências Biológicas e Zootecnia, com um foco particular em estudos de aves. Reconhecendo que muitos biólogos possuem uma base sólida em suas áreas de especialidade mas podem não ter uma formação apropriada em programação e estatística teórica, este material busca preencher essa lacuna de maneira saudável. O objetivo não é apresentar um oceano de fórmulas matemáticas, mas sim construir uma compreensão intuitiva e rigorosa dos princípios estatísticos e de sua aplicação prática utilizando a linguagem de programação R, um forte e indispensável aliado nos tempos atuais para pesquisadores e cientistas de dados.

A estrutura da apostila foi projetada para seguir uma progressão lógica, começando com as habilidades fundamentais de programação e manipulação de dados em R, e avançando gradualmente para modelos estatísticos mais complexos que são frequentemente necessários para pesquisas biológicas, como os modelos lineares mistos e generalizados mistos. Cada capítulo combina explicações conceituais, exemplos práticos contextualizados na ornitologia e zootecnia, e blocos de códigos em R detalhadamente explicados.

O pressuposto é que a estatística não seja um obstáculo a ser superado, mas uma ferramenta poderosa para analisar criteriosamente os dados, planejar experimentos robustos e extrair conclusões válidas e significativas do trabalho de pesquisa. Ao final desta jornada, é esperado que o leitor estará equipado não apenas para analisar seus próprios dados com confiança, mas também para interpretar criticamente a literatura científica de sua área.

ZUUR et al. (2007) e (ZUUR et al., 2007)

Parte I

Ferramentas Essenciais: Dominando o R

Nesta primeira parte, o foco residirá no entendimento das ferramentas computacionais essenciais que fundamentam qualquer análise de dados moderna. Antes de mergulhar nos conceitos estatísticos propriamente ditos, é necessário desenvolver uma fluência básica no ambiente de programação que será utilizado.

A abordagem que será adotada aqui prioriza a eficiência e a intuição, introduzindo o início do ecossistema `tidyverse`, um conjunto de pacotes R projetados para trabalhar em harmonia, tornando a manipulação, exploração e visualização de dados um processo mais humano, lógico e simples.

Ao dominar essa ferramenta, o leitor transformará tarefas árduas de preparação de dados em uma parte integrada e poderosa do processo de descoberta científica.

1 Introdução

Para começar essa jornada, o primeiro passo é configurar o ambiente de trabalho. Isso envolve a instalação de dois *softwares* distintos, mas que trabalham juntos: R e RStudio. Compreender o funcionamento de cada um e como eles se interagem é fundamental para as próximas etapas.

1.1 O que são R e RStudio?

É comum que iniciantes confundam R e RStudio, mas esta distinção é crucial para o processo.

- **R** é a linguagem de programação e o ambiente de software para computação estatística e gráficos. Pode-se pensar que é o "motor" que executa todos os cálculos, análises e gera os gráficos. Além de tudo, é um projeto de código aberto, gratuito e mantido por uma vasta comunidade de desenvolvedores e estatísticos ao redor do mundo.
- **RStudio** é um Ambiente de Desenvolvimento Integrado (IDE, do inglês *Integrated Development Environment*). Se o R é o motor do carro, o RStudio é o painel, o volante, e todo o interior que torna a condução do carro uma experiência agradável e gerenciável. O RStudio fornece uma interface gráfica e amigável que organiza o trabalho em R, facilitando a escrita de *scripts* (arquivos de códigos), a visualização de gráficos, o gerenciamento de pacotes (bibliotecas) e muito mais. Embora seja possível utilizar o R sem o RStudio, a utilização do RStudio é fortemente recomendada, pois deixa o processo de análise muito mais interativo e organizado.

1.2 Instalação passo a passo

A instalação adequada dos programas é um pré-requisito crucial. A ordem de instalação é importante: **R deve ser instalado antes do RStudio**.

1. Instalando o R:

- Acesse o site do *Comprehensive R Archive Network* (CRAN), que é o repositório oficial para o R e seus pacotes.
- Na página inicial, selecione o link de download para o seu sistema operacional (Linux, macOS ou Windows).
- Siga as instruções para baixar a versão mais recente ("base"). É crucial baixar a versão diretamente do CRAN, pois os gerenciadores de pacotes de alguns sistemas operacionais (como o `get-apt` do Ubuntu) podem fornecer versões desatualizadas.
- Execute o arquivo de instalação baixado e siga as instruções padrão, aceitando as configurações padrão.

2. Instalando o RStudio:

- Após a instalação do R, acesse o site da Posit e clique para baixar a versão gratuita do RStudio Desktop.
- Baixe o instalador apropriado para o seu sistema operacional.
- Execute o arquivo de instalação. O RStudio detectará automaticamente a instalação do R existente.

1.3 Navegando na interface do RStudio

Ao abrir o RStudio pela primeira vez, a interface se apresenta dividida em quatro painéis ou quadrantes principais, cada um com uma função específica:

1. **Editor de *scripts*** (Superior esquerdo): Este é o seu principal espaço de trabalho. Aqui, você escreverá e salvará seus *scripts* R (arquivos com extensão `.R`). Trabalhar em um *script*, em vez de digitar comandos diretamente no console, é a base da ciência reprodutível, pois permite salvar, comentar e reutilizar seu código.
2. **Console** (Inferior esquerdo): O console é o código R é efetivamente executado. Você pode digitar os comandos diretamente nele para testes rápidos ou executar linhas de códigos do seu *script* (utilizando o atalho `Ctrl+Enter`). A saída dos comandos também aparecerá aqui.
3. **Ambiente e Histórico** (Superior direito): A aba *Environment* mostra todos os objetos (como *datasets*, variáveis, etc.) que foram criadas na sessão atual do R. Já a aba *History* mantém um registro de todos os comandos utilizados.
4. **Arquivos, Gráficos, Pacotes e Ajuda** (Inferior direita): Este painel multifuncional permite navegar pelos arquivos do seu computador (*Files*) , visualizar gráficos gerados (*Plots*), gerenciar pacotes instalados (*Packages*), e acessar documentações de ajuda do R (*Help*).

É importante salientar que o RStudio permite customizações, como a alteração das posições dos painéis.

1.4 O conceito de pacotes

A grande força do R reside em seu ecossistema de pacotes. Um pacote é a coleção de funções, dados e documentação que estende as capacidades iniciais do R. Para qualquer tarefa estatística ou de manipulação de dados que se possa imaginar, provavelmente existe algum pacote que a facilita.

1.4.1 Instalando e Carregando Pacotes Essenciais

Existe uma distinção básica a ser realizada entre instalar e carregar um pacote.

- **Instalação**: É o ato de baixar o pacote do CRAN e instalá-lo no computador. Isso é realizado apenas uma vez para cada pacote.
- **Carregamento**: É o ato de carregar o pacote instalado em sua sessão do R de forma que as funções adicionais fiquem disponíveis para uso. Isso precisa ser feito toda vez que uma sessão no R é iniciada.

Para este material, os pacotes centrais são: `tidyverse`, `lme4`, `lmerTest` e `nlme`. Um dos métodos para instalar pacotes R no computador é por meio da função `install.packages()`:

```
# Instala o pacote tidyverse, que inclui dplyr, ggplot2 e outros
install.packages("tidyverse")

# Instala o pacote para modelos lineares mistos
install.packages("lme4")

# Instala outros pacotes para modelos mistos
install.packages("lmerTest")
install.packages("nlme")
```

Após a instalação, para usar as funções de um pacote, é preciso carregá-lo com a função `library()`:

```
library(tidyverse)
```

1.5 Diretório de Trabalho e Projetos RStudio

O diretório de trabalho é a pasta no seu computador onde o R irá procurar por arquivos para ler e onde, também, salvará os arquivos criados (como gráficos, *scripts* e *datasets* modificados). É

possível identificar o diretório atual através do comando `getwd()` e, embora também seja possível defini-la manualmente com a função `setwd("caminho/para/sua/pasta")`, essa prática não é aconselhável, visto que o uso de caminhos de arquivos absolutos torna o código não portável; ou seja, ele não irá funcionar se você mover a pasta do projeto ou tentá-la executá-lo em outro computador.

A solução moderna e robusta para esse problema é a utilização de **Projetos RStudio**. Um projeto RStudio (extensão `.Rproj`) é um arquivo que você cria dentro de uma pasta do seu projeto de pesquisa. Ao abrir um projeto, o RStudio automaticamente define o diretório de trabalho para aquela pasta. Isso garante que todos os caminhos de arquivo do seu código possam ser relativos à raiz do projeto, tornando sua análise totalmente reprodutível e compartilhável de forma eficaz. Outra maneira de criar projetos é através do próprio RStudio, através das seguintes instruções `File > New Project > New Directory > New Project` e nesta última etapa, você escolherá um nome para o projeto e a pasta de sua pesquisa, finalizando em `Create Project`. A criação de um projeto para cada análise de pesquisa é uma prática fundamental para a organização e a reprodutibilidade científica.

1.6 R Básico

A leitura desta sessão é aconselhada para o leitor que nunca teve contato com o R. Os tópicos introduzidos são especiais para a compreensão do que é um *dataframe*, a estrutura dos *datasets* dentro do R, e quais operações estarão sendo realizadas quando estivermos efetuando filtragens e modificações de suas colunas. Também são importantes para a compreensão do que é uma função no R.

1.6.1 Operadores Matemáticos

Os operadores matemáticos, também conhecidos por operadores binários, dentro do ambiente R soam como familiares. A Tabela 1.1 exibe os operadores mais básicos utilizados.

Para exemplificar como efetuar cálculos de expressões matemáticas no R, suponha que desenhamos calcular o valor de:

$$2 \times 2 + \frac{4+4}{2}.$$

Para isso, escrevemos `2*2 + (4+4)/2` no console para determinarmos o resultado

```
2*2 + (4+4)/2
```

```
[1] 8
```

Tabela 1.1: Operadores matemáticos básicos.

Operadores	Descrição
+	Adição
-	Subtração
*	Multiplicação
/	Divisão
^	Exponenciação

1.6.2 Objetos e funções

O R permite guardar valores dentro de um **objeto**. Um objeto é simplesmente um nome que guarda uma determinada informação na memória do computador, que é criado por meio do operador `<-`. Veja que no código a seguir

```
x <- 10 # Salvando "10" em "x"
x       # Avaliando o objeto "x"
```

```
[1] 10
```

foi salvo que a informação que `x` carrega é o valor `10`. Portanto, toda vez que o objeto `x` for avaliado, o R irá devolver o valor `10`.

É importante ressaltar que há regras para a nomeação dos objetos, dentre elas, não começar com números. Assim, todos os seguintes exemplos são permitidos: `x <- 1`, `x1 <- 1`, `meu_objeto <- 1`, `meu.objeto <- 1`. Ainda, o R diferencia letras minúsculas de maiúsculas, então objetos como `y` e `Y` são diferentes.

Enquanto que os objetos são nomes que salvam informações de valores, **funções** são nomes que guardam informações de um código R, retornando algum resultado programado. A sintaxe básica de uma função é `nome_funcao(arg1, arg2, ...)`. Os valores dentro dos parênteses são chamados por **argumentos**, que são informações necessárias para o bom funcionamento de uma função. Às vezes, uma função não necessita do fornecimento de argumentos específicos.

Uma função simples, porém útil, é a `sum()`. Ela consiste em somar os valores passados em seu argumento. Suponha que desejamos somar `1+2+3+4+5`. Assim,

```
sum(1,2,3,4,5)
```

```
[1] 15
```

é possível reparar que o resultado é 15.

A classe de um objeto é muito importante na programação em R. É a partir disso que as funções e operadores conseguem entender o que fazer com cada objeto. Há uma infinidade de classes, dentre as mais conhecidas são: `numeric`, `character`, `data.frame`, `logical` e `factor`. Para averiguar o tipo de classe, a função `class()` retorna exatamente a classe do objeto.

```
class("a")
```

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

```
class(1)
```

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

```
class(mtcars)
```

```
[1] "data.frame"
```

```
class(TRUE)
```

```
[1] "logical"
```

1.6.3 Vetores e *Data frames*

Vetores são uma estrutura fundamental dentro do R, em especial, é a partir deles que os *data frames* são construídos. Por definição, são conjuntos indexados de valores e para criá-los, basta utilizar a função `c()` com valores separados por vírgula (ex.: `c(1,2,4,10)`). Para acessar um valor dentro de um determinado vetor, utiliza-se os colchetes `[]`:

```
vetor <- c("a", "b", "c")

# Acessando valor "b"
vetor[2]
```

```
[1] "b"
```

Um vetor só pode guardar um tipo de objeto e ele terá sempre a mesma classe dos objetos que guarda. Caso tentarmos misturar duas classes, o R vai apresentar o comportamento conhecido como **coerção**.

```r
class(c(1,2,3))
```

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

```r
class(vetor)
```

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

```r
class(c(1,2,"a","b"))
```

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

Neste caso, todos os elementos do vetor se transformaram em texto.

Assim, também, *data frames* são de extrema importância no R, visto que são os objetos que guardam os dados e são equivalentes a uma planilha do Excel. A principal característica é possuir linha e colunas. Em geral, as colunas são vetores de mesmo tamanho (ou dimensão). Um valor específico de um *data frame* pode ser acessado, também, via colchetes `[]`:

```r
class(mtcars)
```

```
[1] "data.frame"
```

```r
mtcars[1,2]
```

```
[1] 6
```

`mtcars` é um conjunto de dados muito conhecido na comunidade R.

1.6.4 Fatores

Fatores são uma classe de objetos no R criada para representar variáveis categóricas numericamente. A característica que define essa classe é o atributo `levels`, que representam as possíveis categorias de uma variável categórica.

A título de exemplificação, considere o objeto `sexo` que contém as informações do sexo de uma pessoa. As possibilidades são: `F` (feminino) e `M` (masculino). Por padrão, o R interpreta essa variável como texto (*character*), no entanto, é possível transformá-la em fator por meio da função `as.factor()`.

```r
sexo <- c("F", "F", "M", "M", "F")
class(sexo)
```

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

```r
# Transformando em fator
class(as.factor(sexo))
```

```
[1] "factor"
```

```r
as.factor(sexo)
```

```
[1] F F M M F
Levels: F M
```

Observa-se que a linha adicional `Levels: F M` indicam as categorias. Por padrão, o R ordena esses níveis em ordem alfabética. Para facilitar os cálculos e análises, o R interpreta os níveis categóricos como sendo números distintos, sendo assim, dentro do nosso exemplo `F` representaria o número `0` e `M` representaria o `1`.

1.6.5 Valores especiais

Valores como `NA`, `NaN`, `Inf` e `NULL` ocorrem frequentemente dentro do mundo da programação estatística no R. Em resumo:

- `NA` representa a Ausência de Informação. Suponha que o vetor `idades` que representa a idade de três pessoas. Uma situação que pode ocorrer é `idades <- c(10, NA, NA)`. Portanto, não é sabido a idade das pessoas 2 e 3.
- `NaN` representa indefinições matemáticas. Um exemplo típico é o valor $\log -1$, do qual $x = -1$ não pertence aos possíveis valores de saída da função logarítmica, gerando um `NaN` (*Not a number*).

```r
log(-1)
```

```
Warning in log(-1): NaNs produzidos
```

```
[1] NaN
```

- `Inf` representa um número muito grande ou um limite matemático. Exemplos:

```
# Número muito grande
10^510
```

```
[1] Inf
```

```
# Limite matemático
1/0
```

```
[1] Inf
```

- `NULL` representa a ausência de um objeto. Muitas vezes define-se um objeto como nulo para dizer ao R que não desejamos atribuir valores a ele.

2 Tidyverse

In summary, this book has no content whatsoever.

```
1 + 1
```

```
[1] 2
```

3 Inferência

texto teste $f(x) = ax^2$.

Parte II

Fundamentos do Pensamento Estatístico

4 A Lógica da Inferência Estatística

Sejam os testes de hipóteses H_0 *vs.* H_A.

5 Delineamento de Experimentos

Um delineamento é…

Parte III

Modelagem Estatística de Dados Biológicos

6 Modelos Lineares – A Base da Modelagem

Um modelo linear é definido por:

$$Y = X\beta + \epsilon$$

7 Entendendo os Modelos Mistos

Os efeitos aleatórios são…

8 Modelos Lineares Mistos com `lme4`

O pacote `lme4` nos permite modelar…

9 Modelos Lineares Generalizados Mistos

Um GLMM é definino como sendo…

10 Validação e Interpretação de Modelos Mistos

Após ajustar os modelos, é boa prática validá-los através de técnicas…

Parte IV

Aplicações Práticas e Recursos

Referências

ZUUR, Alain F. *et al.* **Analysing Ecological Data**. New York, NY: Springer, 2007.