Ecologia de Paisagens com R

Darren Norris: darren.norris@unifap.br

03 março, 2024

SumÃ¡rio

I ApresentaÃ§Ã£o **6**

Bem-vindos **6**
Agradecimentos . . . 6

PrefÃ¡cio Ã primeira ediÃ§Ã£o **7**

IntroduÃ§Ã£o **8**
O que vocÃª vai aprender . . . 8
Como este livro estÃ¡ organizado . . . 8
O que vocÃª nÃ£o vai aprender . . . 8
Prerequisites . . . 8
R . . . 8
RStudio . . . 9
O universo arrumado - tidyverse . . . 9
Outros pacotes . . . 9
Codigo no livro . . . 9
OrganizaÃ§Ã£o do codigo no livro . . . 11

II Escala e padrÃµes **12**

1 Escala **13**
1.1 ApresentaÃ§Ã£o . . . 13
1.2 Escala: breve definaÃ§Ã£o . . . 13
1.3 Pacotes e dados . . . 15
1.3.1 Pacotes . . . 15
1.3.2 Dados . . . 16
1.4 Alterando a resoluÃ§Ã£o . . . 19
1.5 Escala espacial e desenho amostral . . . 23
1.5.1 Obter e carregar dados (vectores) . . . 23
1.5.2 Visualizar os arquivos (camadas vector) . . . 24
1.5.3 Obter e carregar dados (raster) . . . 25
1.5.4 Visualizar os arquivos (camadas raster e vector) . . . 25
1.5.5 ReclassificaÃ§Ã£o . . . 27
1.6 ComparaÃ§Ã£o multiescala . . . 29
1.7 PrÃ³ximos passos: repetindo para muitas amostras. . . . 32

2 MÃ©tricas da paisagem **34**
2.1 ApresentaÃ§Ã£o . . . 34

2.2 MÃ©tricas da paisagem e pacote “landscapemetrics” . . . 35
2.2.1 Pacotes . . . 35
2.3 Dados . . . 37
2.3.1 Exibir dados raster e sobreposiÃ§Ã£o com locais de amostragem . . . 38
2.4 Calculo de mÃ©tricas . . . 41
2.4.1 Uso de pontos de amostragem na ecologia da paisagem . . . 41
2.4.2 Ponto Ãºnico, raio Ãºnico, mÃ©trica Ãºnica . . . 42
2.4.3 Ponto Ãºnico, distÃ¢ncias variados, mÃ©trica Ãºnica . . . 44
2.4.4 Ponto Ãºnico, distÃ¢ncias variados, mÃ©tricas variadas . . . 55
2.5 Links Ãºteis . . . 59

3 MudanÃ§as na cobertura vegetal **61**
3.1 ApresentaÃ§Ã£o . . . 61
3.2 Ã rea de estudo . . . 61
3.3 Pacotes e Fontes de dados . . . 61
3.3.1 Pacotes . . . 61
3.3.2 Fontes de dados . . . 62
3.4 MudanÃ§as na paisagem: indo alÃ©m de desmatamento . . . 64
3.4.1 DefinaÃ§Ã£o . . . 65
3.4.2 Suavizar a variaÃ§Ã£o nas estimativas anuais . . . 65
3.4.3 AnÃ¡lise exploratÃ³ria . . . 67
3.4.4 ReclassificaÃ§Ã£o . . . 69
3.4.5 MudanÃ§as . . . 72
3.4.6 Mapeamento de mudanÃ§as . . . 73
3.4.7 ConclusÃ£o . . . 74
3.5 Metodologia Espacialmente ExplÃcita para Quantificar os Impactos A Jusante da HidrelÃ©trica Ferreira Gomes na Cobertura Vegetal Nativa . . . 76
3.5.1 Dados: . . . 76
3.5.2 Metodologia: . . . 77
3.6 Links uteis . . . 78

4 Respostas multiescala de espÃ©cies **79**
4.1 ApresentaÃ§Ã£o . . . 79
4.2 Escala de efeito - definiÃ§Ã£o . . . 81
4.2.1 DefiniÃ§Ãµes . . . 81
4.2.2 AplicaÃ§Ã£o . . . 81
4.2.3 Como identificar a escala de efeito? . . . 82
4.3 Pacotes e dados . . . 82
4.3.1 Pacotes . . . 82
4.3.2 Dados necessarios . . . 82
4.3.3 O paisagem do estudo . . . 84
4.4 Buffers . . . 86
4.4.1 Floresta . . . 86
4.4.2 Floresta e a ocorrÃªncia de espÃ©cies . . . 86
4.5 Kernels . . . 99
4.6 Links Ãºteis . . . 99

III Exemplos de caso: padrÃµes de paisagem **102**

5 Garimpo do LourenÃ§o **103**
5.1 ApresentaÃ§Ã£o . . . 103
5.2 Pacotes necessarios: . . . 104
5.3 Ã rea de estudo . . . 104

5.4 Dados . . . 104
5.4.1 Ponto de referÃªncia (EPSG: 4326) . . . 104
5.4.2 Ponto de referÃªncia (EPSG: 31976) . . . 105
5.4.3 Verificar com mapa de base (OpenStreetMap). . . . 105
5.4.4 Dados: MapBiomas cobertura da terra . . . 106
5.5 Calculo de mÃ©tricas . . . 107
5.5.1 MÃ©tricas para a paisagem . . . 108
5.5.2 MÃ©tricas para as classes . . . 108
5.5.3 MÃ©tricas para as manchas . . . 109
5.5.4 Quais mÃ©tricas devo escolher? . . . 110
5.5.5 MÃ©tricas por classe de mineraÃ§Ã£o . . . 110
5.5.6 Exportar as mÃ©tricas . . . 112
5.6 Apresentando os resultados . . . 112
5.6.1 Preparando os resultados . . . 112
5.6.2 Uma tabela versatil . . . 115
5.6.3 ReorganizaÃ§Ã£o . . . 115
5.6.4 Uma figura elegante . . . 116
5.7 ComparaÃ§Ã£o entre anos . . . 118
5.8 ComparaÃ§Ã£o entre anos e paisagens . . . 122
5.9 ConclusÃµes e prÃ³ximos passos . . . 127

IV R e RStudio: Codigo com certeza 128

6 PrÃ©-requisitos 129
6.1 IntroduÃ§Ã£o . . . 129
6.1.1 VersÃ£o do R . . . 129
6.2 InstalaÃ§Ã£o do R . . . 129
6.3 InstalaÃ§Ã£o do Rtools . . . 129
6.4 InstalaÃ§Ã£o do RStudio . . . 129
6.5 Pacotes . . . 130
6.5.1 Pacotes do GitHUb . . . 131
6.6 Dados . . . 131

7 IntroduÃ§Ã£o ao R 132
PrÃ©-requisitos do capÃtulo . . . 132
7.1 ContextualizaÃ§Ã£o . . . 132
7.2 R e RStudio . . . 132
7.3 Funcionamento da linguagem R . . . 134
7.3.1 Console . . . 135
7.3.2 Scripts . . . 135
7.3.3 Operadores . . . 136
7.3.4 Objetos . . . 138
7.3.5 FunÃ§Ãµes . . . 139
7.3.6 Pacotes . . . 140
7.3.7 Ajuda (*Help*) . . . 143
7.3.8 Ambiente (*Environment*) . . . 144
7.3.9 CitaÃ§Ãµes . . . 145
7.3.10 Principais erros de iniciantes . . . 147
7.4 Estrutura e manipulaÃ§Ã£o de objetos . . . 149
7.4.1 Atributo dos objetos . . . 149
7.4.2 ManipulaÃ§Ã£o de objetos unidimensionais . . . 158
7.4.3 ManipulaÃ§Ã£o de objetos multidimensionais . . . 161
7.4.4 Valores faltantes e especiais . . . 165

7.4.5 DiretÃ³rio de trabalho . . . 166
7.4.6 Importar dados . . . 166
7.4.7 ConferÃªncia dos dados importados . . . 168
7.4.8 Exportar dados . . . 169
7.5 Para se aprofundar . . . 170
7.5.1 Livros . . . 170
7.5.2 Links . . . 170
7.6 ExercÃcios . . . 170

8 Tidyverse **172**
PrÃ©-requisitos do capÃtulo . . . 172
8.1 ContextualizaÃ§Ã£o . . . 172
8.2 *tidyverse* . . . 173
8.3 readr, readxl e writexl . . . 175
8.4 tibble . . . 176
8.5 pipe: base (|>) e magrittr (%>%) . . . 176
8.6 tidyr . . . 178
8.6.1 palmerpenguins . . . 178
8.6.2 glimpse() . . . 179
8.6.3 unite() . . . 180
8.6.4 separate() . . . 180
8.6.5 drop_na() e replace_na() . . . 181
8.6.6 pivot_longer() e pivot_wider() . . . 182
8.7 dplyr . . . 184
8.7.1 GramÃ¡tica . . . 184
8.7.2 Sintaxe . . . 184
8.7.3 palmerpenguins . . . 185
8.7.4 relocate() . . . 185
8.7.5 rename() . . . 186
8.7.6 select() . . . 186
8.7.7 pull() . . . 187
8.7.8 mutate() . . . 187
8.7.9 arrange() . . . 188
8.7.10 filter() . . . 189
8.7.11 slice() . . . 191
8.7.12 distinct() . . . 193
8.7.13 count() . . . 194
8.7.14 group_by() . . . 194
8.7.15 summarise() . . . 195
8.7.16 bind_rows() e bind_cols() . . . 196
8.7.17 *_join() . . . 197
8.7.18 OperaÃ§Ãµes de conjuntos e comparaÃ§Ã£o de dados . . . 198
8.8 stringr . . . 198
8.9 forcats . . . 200
8.10 lubridate . . . 202
8.11 purrr . . . 209
8.12 Para se aprofundar . . . 213
8.12.1 Livros . . . 213
8.12.2 Links . . . 213
8.13 ExercÃcios . . . 213

9 Primeiros passos com uma raster **215**
9.0.1 Obter e carregar dados (raster) . . . 216
9.0.2 ReclassificaÃ§Ã£o . . . 217

10 Primeiros passos com vector **219**
10.1 Obter e carregar dados (vectores) 219
10.2 Visualizar os arquivos (camadas vector) 222

V Encerramento **223**

Bibliografia **224**

GlossÃ¡rio **225**

A Annexo 1 **233**
A.1 Download mapbiomas cover 233
A.2 Crop mapbiomas cover 234
A.3 Plot with legend 236

Part I

ApresentaÃ§Ã£o

Bem-vindos

Este Ã© um trabalho em andamento do 1Âª ediÃ§Ã£o: **"Ecologia de Paisagens com R"**.

Versoes:

- web: https://darrennorris.github.io/epr/
- pdf: https://github.com/darrennorris/epr/blob/main/docs/epr.pdf

Este material introdutÃ³rio Ã© direcionado principalmente a estudantes de graduaÃ§Ã£o e pÃ³s-graduaÃ§Ã£o em ecologia e Ã¡reas afins. Ele estÃ¡ sendo utilizado como interface para os materiais do curso de Ecologia da Paisagem, do Bacharelado em CiÃªncias Ambientais da Universidade Federal do AmapÃ¡, Brasil.

O objetivo Ã© de apresentar os capacidades e opÃ§Ãµes para desenvolver e integrar pesquisas na Ecologia da Paisagem no ambiente estatÃstica de R.

Esperamos que ele seja utilizado tanto por quem quer se aprofundar em anÃ¡lises comumente utilizadas em Ecologia da Paisagem, mesmo por quem tem poucas habilidades quantitativas.

O conteudo e organizaÃ§Ã£o dos capÃtulos esta separado em partes.

- O primeiro parte Ã© "ApresentaÃ§Ã£o".
  o objetivo dessa parte Ã© inclui os aspectos mais gerais sobre os componentes e da estrutura do livro e seus objetivos.
- O segundo Ã© "Escala e PadrÃµes".
  Aqui, os capitulos inclui um breve introduÃ§Ã£o usando R para explorar alguns componentes chaves da Ecologia da Paisagem.
- A terceira Ã© "Exemplos de caso".
  Aqui, os capÃtulos apresentam a aplicaÃ§Ã£o de anÃ¡lises especÃficas e atualmente utilizadas em Ecologia da Paisagem.
- A quarta parte do livro Ã© chamada de "R e RStudio".
  Esta parte trata o funcionamento da linguagem R. Aqui, nÃ³s aprendemos desde como instalar o R e o RStudio atÃ© a manipulaÃ§Ã£o e visualizaÃ§Ã£o de dados geoespaciais no R.
- A parte final Ã© "Encerramento".
  Aqui os capÃtulos apresentam a bibliografia e alguns apÃªndices (exemplos de cÃ³digo).

Agradecimentos

O material deste livro foi desenvolvido a partir de cursos de graduaÃ§Ã£o ministrados pelos autores. Agradecemos a todos os alunos por fornecerem um suprimento ilimitado de perguntas que melhoraram o conteÃºdo. Este livro nÃ£o Ã© apenas o resultado dos autores. Mas Ã© o resultado de muitas pessoas na comunidade R e Ecologia da Paisagem no Brasil. Agradecimentos especiais ao Dr. Alexandre Martensen, cujos exemplos inspiraram meu desenvolvimento de materiais de aprendizagem on-line e aos autores do livro AnÃ¡lises EcolÃ³gicas no R cuja dedicaÃ§Ã£o me salvou anos de trabalho. Muito obrigado!

PrefÃ¡cio Ã primeira ediÃ§Ã£o

HÃ¡ quem diga que a velocidade com que a tecnologia e a ciÃªncia avanÃ§am tende a tornar livros e manuais sobre mÃ©todos rapidamente obsoletos. A evoluÃ§Ã£o dos computadores pessoais e a ampliaÃ§Ã£o do acesso a estes e Ã Internet tÃªm transformado o jeito como aprendemos e ensinamos. As pessoas estÃ£o cada vez mais familiarizados com a tecnologia e esperam que o ensino seja dinÃ¢mico e interativo.

Portanto, um livro aberto e disponÃvel livremente, que as pessoas possam compartilhar e contribuir torna-se uma opÃ§Ã£o cada vez mais relevante. A intenÃ§Ã£o Ã© para o livro seja utilizado como:

- Material de referÃªncia - fornecendo informaÃ§Ãµes atualizadas.
- Ferramenta de aprendizagem - apresentando diferentes ideias e abordagens.
- Projeto de colaboraÃ§Ã£o - permitindo que pessoas trabalhem juntos para criar um recurso educacional valioso.

IntroduÃ§Ã£o

Ecologia da Paisagem Ã© uma disciplina empolgante que permite transformar dados brutos em compreensÃ£o, insight e conhecimento.

O que vocÃª vai aprender

Ecologia da Paisagem Ã© um campo vasto e nÃ£o hÃ¡ como dominar tudo lendo um Ãºnico livro. Este livro visa fornecer uma base sÃ³lida nas ferramentas mais importantes e conhecimento suficiente para encontrar os recursos para aprender mais quando necessÃ¡rio. Um modelo das etapas de um projeto tÃpico de Ecologia da Paisagem se parece como…..

Six key steps for functional landscape analyses of habitat change https://doi.org/10.1007/s10980-020-01048-y - Acknowledge ecological theory and conceptual paradigms - Evaluate the fit of available data - Assess the three facets of the scale concept - Recognize different sampling designs - Use proper conceptual models - Measure meaningful raster characteristics

Como este livro estÃ¡ organizado

A descriÃ§Ã£o anterior das ferramentas da Ecologia da Paisagem Ã© organizada aproximadamente de acordo com a ordem em que vocÃª as usa em uma anÃ¡lise (embora, Ã© claro, vocÃª as itere vÃ¡rias vezes). Em nossa experiÃªncia, aprender os detalhes do linguagem R e programaÃ§Ã£o primeiro nÃ£o Ã© o ideal, porque 80% do tempo Ã© rotineiro e chato e, nos outros 20% do tempo, Ã© estranho e frustrante. Esse Ã© um pÃ©ssimo lugar para comeÃ§ar a aprender um novo assunto! Em vez disso, os primeiros capÃtulos apresentam alguns dos conceitos centrais da Ecologia da Paisagem, apoiados em exemplos com R com dados que jÃ¡ foram importados e organizados. Dessa forma, quando vocÃª ingerir e organizar seus prÃ³prios dados, sua motivaÃ§Ã£o permanecerÃ¡ alta porque vocÃª sabe que a dor vale o esforÃ§o.

Dentro de cada capÃtulo, tentamos aderir a um padrÃ£o consistente: comece com alguns exemplos motivadores para que vocÃª possa ver o quadro geral e depois mergulhe nos detalhes. Cada seÃ§Ã£o do livro Ã© combinada com exercÃcios para ajudÃ¡-lo a praticar o que aprendeu. Embora possa ser tentador pular os exercÃcios, nÃ£o hÃ¡ melhor maneira de aprender do que praticar em problemas reais.

O que vocÃª nÃ£o vai aprender

Base teorica pra tras os conceitos e calculos. O foco Ã© sobre a aplicaÃ§Ã£o, e isso nÃ£o preciso avanÃ§os teoricas. Incluimos referencias para que Ã© possivel obter maior detalhes sobre os conceitos e teorias ecologicos e caluclos usados na Ecologia da Paisagem.

Prerequisites

Fizemos algumas suposiÃ§Ãµes sobre o que vocÃª jÃ¡ sabe para aproveitar ao mÃ¡ximo este livro. VocÃª deve ser geralmente alfabetizado numericamente, e com conhecimento previa de ecologia, geoprocessamento e uso de sistemas de informaÃ§Ã£o geografica.

VocÃª precisa de quatro coisas para executar o cÃ³digo deste livro: R, RStudio, RTools e uma coleÃ§Ã£o de pacotes. Os pacotes sÃ£o as unidades fundamentais do cÃ³digo R reproduzÃvel. Eles incluem funÃ§Ãµes reutilizÃ¡veis, documentaÃ§Ã£o que descreve como usÃ¡-los e dados de amostra.

R

Para fazer o download do R, acesse CRAN, a **c**omprehensive **R** **a**rchive **n**network, https://cloud.r-project.org. Uma nova versÃ£o principal do R Ã© lanÃ§ada uma vez por ano e hÃ¡ 2 a 3 versÃµes secundÃ¡rias a cada ano. Ã uma boa ideia atualizar regularmente. A atualizaÃ§Ã£o pode ser um pouco complicada, especialmente para as versÃµes principais que exigem a reinstalaÃ§Ã£o de todos os seus pacotes, mas adiar sÃ³ piora as coisas. Para aqueles que sÃ£o novos no uso do R, evite atualizaÃ§Ãµes no meio do projeto ou quando hÃ¡ prazos se aproximando. Recomendamos R 4.2.0 ou posterior para este livro.

RStudio

RStudio Ã© um ambiente de desenvolvimento integrado, ou IDE, para programaÃ§Ã£o R, que vocÃª pode baixar em https://posit.co/download/rstudio-desktop/. O RStudio Ã© atualizado algumas vezes por ano e avisa automaticamente quando uma nova versÃ£o Ã© lanÃ§ada, para que nÃ£o haja necessidade de verificar novamente. Ã uma boa ideia atualizar regularmente para aproveitar os melhores e mais recentes recursos. Para este livro, certifique-se de ter pelo menos o RStudio 2022.02.0.

O universo arrumado - tidyverse

VocÃª tambÃ©m precisarÃ¡ instalar alguns pacotes do R. Um **pacote** do R Ã© uma coleÃ§Ã£o de funÃ§Ãµes, dados e documentaÃ§Ã£o que estende os recursos do R base. O uso de pacotes Ã© a chave para o uso bem-sucedido do R. A muitos dos pacotes que vocÃª aprenderÃ¡ neste livro faz parte do chamado tidyverse: https://www.tidyverse.org/. Todos os pacotes no tidyverse compartilham uma filosofia comum de programaÃ§Ã£o de dados e R e sÃ£o projetados para trabalhar juntos. A anÃ¡lise de dados ecolÃ³gicos tende a ser repetitiva e exige cuidado e atenÃ§Ã£o para garantir que os resultados sejam corretos, cientificamente vÃ¡lidos e possam ser verificados. os pacotes no tidyverse torna o trabalho com dados ecolÃ³gicos mais rÃ¡pido e fÃ¡cil.

VocÃª pode instalar o tidyverse completo com uma Ãºnica linha de cÃ³digo:

```
install.packages("tidyverse")
```

No seu computador, digite essa linha de cÃ³digo no console e pressione enter para executÃ¡-lo. R irÃ¡ baixar os pacotes do CRAN e instalÃ¡-los em seu computador.

VocÃª nÃ£o poderÃ¡ usar as funÃ§Ãµes, objetos ou arquivos de ajuda em um pacote atÃ© carregÃ¡-lo com `library()`. Depois de instalar um pacote, vocÃª pode carregÃ¡-lo usando a funÃ§Ã£o `library()`:

```
library(tidyverse)
```

Isso diz a vocÃª que o tidyverse carrega nove pacotes: dplyr, forcats, ggplot2, lubridate, purrr, readr, stringr, tibble, alignr. Eles sÃ£o considerados o **nÃºcleo** do tidyverse porque vocÃª os usarÃ¡ em quase todas as anÃ¡lises.

Os pacotes no tidyverse mudam com bastante frequÃªncia. VocÃª pode ver se hÃ¡ atualizaÃ§Ãµes disponÃveis executando `tidyverse_update()`.

Outros pacotes

Existem muitos outros pacotes excelentes que nÃ£o fazem parte do tidyverse porque resolvem problemas em um domÃnio diferente ou sÃ£o projetados com um conjunto diferente de princÃpios subjacentes. Isso nÃ£o os torna melhores ou piores, apenas diferentes. Em outras palavras, o complemento do tidyverse nÃ£o Ã© o universo bagunÃ§ado, mas muitos outros universos de pacotes inter-relacionados. Ao lidar com mais projetos de Ecologia da Paisagem com R, vocÃª aprenderÃ¡ novos pacotes e novas formas de pensar sobre os dados.

Usaremos outras pacotes de fora do tidyverse neste livro. Por exemplo, usaremos os seguintes pacotes porque eles fornecem conjuntos de funcÃµes e dados interessantes para trabalharmos no processo de aprendizado de Ecologia da paisagem e de R:

```
install.packages(c("sp", "sf","raster", "mapview","tmap",
                   "terra", "kableExtra", "landscapemetrics", "siland"))
```

Codigo no livro

- Objetivo nÃ£o Ã© de apresentar detalhes sobre os cÃ¡lculos/mÃ©todos estatÃsticas ou os funÃ§Ãµes no R. Existem diversos exemplos disponÃveis "CiÃªncia de Dados com R–IntroduÃ§Ã£o……": e com

google “r cran introduÃ§Ã£o tutorial”……. Alem disso, existem grupos de ajuda, como por exemplo: R Brasil e Stack Overflow em PortuguÃªs

- O objetivo Ã© de apresentar um livro mostrado os capacidades e opÃ§Ãµes para desenvolver e integrar pesquisas na ecologia da paisagem no ambiente estatÃstica de R

Obviamente, todas as tarefas de geoprocessamento podem ser desenvolvidas anteriormente em um SIG (QGIS). EntÃ£o porque use R? R tem a capacidade (baseada em codigo) para alternar entre tarefas de processamento, modelagem e visualizaÃ§Ã£o de dados geogrÃ¡ficos e nÃ£o geogrÃ¡ficos. Alem disso, como Ã© possÃvel importar, modificar, analisar e visualizar dados espaciais no mesmo ambiente com script/codigo, o R permite fluxos de trabalho transparentes e reproduzÃveis (A CiÃªncia Aberta).

AliÃ¡s, atualmente a grande maioria dos artigos cientÃficos publicados na revista Landscape Ecology incluir anÃ¡lises usando R.

OrganizaÃ§Ã£o do codigo no livro

O capitulo estÃ¡ organizado em etapas de processamento, com blocos de cÃ³digo em caixas cinzas:

```
codigo de R para executar
```

Para segue os passos, os blocos de cÃ³digo precisam ser executados em sequÃªncia. Se vocÃª pular uma etapa, ou rodar fora de sequÃªncia o prÃ³ximo bloco de cÃ³digo provavelmente nÃ£o funcionarÃ¡.

As linhas de codigo de R dentro de cada caixa tambem preciso ser executados em sequÃªncia. O simbolo # Ã© usado para incluir comentarios sobre os passos no codgio (ou seja, linhas comeÃ§ando com # nÃ£o Ã© codigo de executar).

```
# Passo 1
codigo de R passo 1 # texto e numeros tem cores diferentes
# Passo 2
codigo de R passo 2
# Passo 3
codigo de R passo 3
```

Alem disso, os simbolos #> e/ou **[1]** no inÃcio de uma linha indica o resultado que vocÃª verÃ¡ no console de R.

```
# Passo 1
1+1
```

[1] 2

```
# Passo 2
x <- 1 + 1
# Passo 3
x
```

[1] 2

```
# Passo 4
x + 1
```

[1] 3

Part II

Escala e padrÃµes

1 Escala

1.1 ApresentaÃ§Ã£o

Nesta capitulo vamos entender a importÃ¢ncia de escala na ecologia da paisagem atravÃ©s cÃ¡lculos com a proporÃ§Ã£o de floresta. Durante o capitulo vocÃª aprenderÃ¡ a

1. Alterar escala (resoluÃ§Ã£o e extensÃ£o espacial),

2. Calcular a Ã¡rea de uma classe de habitat,

3. Desenvolve uma comparaÃ§Ã£o multiescala.

Ã muito importante ficar claro para vocÃª o que Ã© escala (e o que nÃ£o Ã©!), e qual a importÃ¢ncia desse conceito na elaboraÃ§Ã£o do desenho amostral, na coleta de dados, nas anÃ¡lises e na tomada de decisÃ£o. Nesse tutorial usaremos conteudo baseado no CapÃtulo 2 do livro Spatial Ecology and Conservation Modeling [@fletcher2018] e "Tutorial Escala" do Dr. Alexandre Martensen.

Para ajudar a acompanhar e entender os exemplos no capÃtulo, vocÃª deve ler os seguintes artigos:

- Best, J. 2019.
  Anthropogenic stresses on the worldâ s big rivers. Nature Geoscience 12, 7â "21. https://doi.org/10.1038/s41561-018-0262-x
- Raffo DCD, Norris D, Hartz SM, Michalski F. 2022.
  Anthropogenic influences on the distribution of a threatened apex-predator around sustainable-use reserves following hydropower dam installation. PeerJ 10:e14287 https://doi.org/10.7717/peerj.14287
- BÃ¡rcenasâ GarcÃa, A., Michalski, F., Gibbs, J. P., & Norris, D. (2022). Amazonian runâ ofâ river dam reservoir impacts underestimated: Evidence from a beforeâ "after controlâ "impact study of freshwater turtle nesting areas. Aquatic Conservation: Marine and Freshwater Ecosystems, 32(3), 508-522. https://doi.org/10.1002/aqc.3775
- SÃ¡â Oliveira JC, Hawes JE, Isaacâ Nahum VJ, Peres C 2015.
  Upstream and downstream responses of fish assemblages to an eastern Amazonian hydroelectric dam. Freshwater biology. Oct;60(10):2037-50. https://doi.org/10.1111/fwb.12628

1.2 Escala: breve definaÃ§Ã£o

Todos os processos e padrÃµes ecolÃ³gicos tÃªm uma dimensÃ£o temporal e espacial. Assim sendo, o conceito de escala nÃ£o somente representar essas dimensÃµes, mas tambÃ©m, ajudar nos apresentÃ¡-los de uma forma que facilite o entendimento sobre os processos e padrÃµes sendo estudados.

Na ecologia o termo escala refere-se Ã dimensÃ£o ou domÃnio espaÃ§o-temporal de um processo ou padrÃ£o. Na ecologia da paisagem, a escala Ã© frequentemente descrita por sua componentes: resoluÃ§Ã£o e extensÃ£o.

- **ResoluÃ§Ã£o**: menor unidade espacial de medida para um padrÃ£o ou processo.
- **ExtensÃ£o**: descreve o comprimento ou tamanho de Ã¡rea sob investigaÃ§Ã£o.

ResoluÃ§Ã£o e extensÃ£o tendem a covariar – estudos com maior extensÃ£o tendem a ter resoluÃ§Ã£o maiores tambÃ©m. Parte dessa covariÃ¢ncia Ã© prÃ¡tica: Ã© difÃcil trabalhar em grandes extensÃµes com dados coletados em tamanhos de resoluÃ§Ã£o finos. No entanto, parte dessa covariÃ¢ncia tambÃ©m Ã© conceitual: muitas vezes em grandes extensÃµes, podemos esperar que processos operando em resoluÃ§Ã£o muito finos forneÃ§am somente "ruÃdo" e nÃ£o dados/informaÃ§Ãµes relevantes sobre os sistemas. Como os desafios computacionais diminuÃram e a disponibilidade de dados de alta resoluÃ§Ã£o aumentou, a covariÃ¢ncia entre resoluÃ§Ã£o e extensÃ£o nas investigaÃ§Ãµes diminuiu.

Lembrando, na primeira aula, vimos que a escala espacial pode ser interpretada com base em trÃªs dimensoes:

- no fenÃ´meno de interesse;

- na amostragem que ocorre;

e/ou

- na anÃ¡lise

Para que a Ecologia da Paisagem gere evidÃªncias cientÃficas robustas e Ãºteis, a escala nas trÃªs dimensÃµes deve ser consistente e apropriada para o estudo. Incompatibilidade entre a escala de anÃ¡lise (definida por o pesquisador) e a escala de efeito (definida pelo verdadeiro relacionamento) representa uma versÃ£o do Problema de Unidade Areal ModificÃ¡vel ("Modifiable Areal Unit Problem" [@gehlke1934certain]), em que os resultados e a inferÃªncia torna-se um artefato do esquema de amostragem espacial em vez de um processo ecolÃ³gico [@jelinski1996modifiable; @WU2006].

Aqui nos concentramos na dimensÃ£o â anÃ¡liseâ , e aprendemos como a escala espacial pode ser alterada e representada em modelos ecolÃ³gicos.

1.3 Pacotes e dados

Em geral Ã© necessÃ¡rio baixar alguns pacotes para que possamos fazer as nossas anÃ¡lises. Neste caso precisamos os seguintes pacotes, que deve esta instalado antes:

- tidyverse,
- sf,
- terra,
- mapview,
- tmap.

1.3.1 Pacotes

No R, carregar os pacotes necessarios com o codigo:

```
library(tidyverse)
library(sf)
library(terra)
library(mapview)
library(tmap)
library(eprdados)
```

Caso os pacotes nÃ£o tenham sido instalados, o R vai avisar atraves um mensagem tipo: Error in library(nomepacote) there is no package called nomepacote. Neste caso, para instalÃ¡-los consulte o CapÃtulo 6 ou os capitulos aqui Capitulo 4 instalaÃ§Ã£o de pacotes e aqui Capitulo 4 pacotes . Prestando atenÃ§Ã£o nas diferenÃ§as para instalar um pacote como `eprdados` que vem do GitHub.

1.3.2 Dados

Vamos olhar um exemplo do mundo real. Uma pequena amostra do Rio Araguari, perto de Porto Grande. O ponto central Ã© de longitude: -51.406312 latitude: 0.726236. Para visualizar o ponto no Google Earth: https://earthengine.google.com/timelapse#v=0.72154,-51.41543,11.8,latLng&t=2.24&ps=25&bt=19840101&et=20201231&startDwell=0&endDwell=0 .

Vammos trabalhar com os dados de MapBiomas, que produz mapeamento anual da cobertura e uso da terra no Brasil desde 1985. Os dados de MapBiomas vem no formato de raster, que tem uma classificaÃ§Ã£o da terra feito a partir da classificaÃ§Ã£o pixel a pixel de imagens das satÃ©lites Landsat. Todo processo Ã© feito com algoritmos de aprendizagem de mÃ¡quina (machine learning) atravÃ©s da plataforma Google Earth Engine, que oferece imensa capacidade de processamento na nuvem. Mais detalhes sobre a metodologia aqui: Metodologia MapBiomas.

Para carregar um arquivo raster trabalhamos com o pacote terra. O pacote tem varios funÃ§oes para a anÃ¡lise e modelagem de dados geogrÃ¡ficos. NÃ³s podemos ler os dados de cobertura da terra no arquivo ".tif" com a funÃ§Ã£o `rast`.

```
# arquivo no pacote "eprdados"
arquivo <- system.file("raster/amostra_mapbiomas_2020.tif",
                       package = "eprdados")
# carregar
ramostra <- rast(arquivo)
```

Plotar para verificar.

```
plot(ramostra)
```

Figura 1.1: Mapbiomas 2020. Uma pequena amostra do Rio Araguari, perto de Porto Grande.

Podemos tambÃ©m verificar informaÃ§Ãµes sobre o raster (metadados) - simplesmente rodar o nome do objeto:

```
ramostra
```

```
## class       : SpatRaster
## dimensions  : 8, 9, 1  (nrow, ncol, nlyr)
## resolution  : 29.89281, 29.89281  (x, y)
## extent      : 454659.8, 454928.9, 80160.06, 80399.2  (xmin, xmax, ymin, ymax)
## coord. ref. : SIRGAS 2000 / UTM zone 22N (EPSG:31976)
## source      : amostra_mapbiomas_2020.tif
## name        : mapbiomas_2020
```

```
## min value   :              3
## max value   :             33
```

Isso nos mostra informaÃ§Ãµes sobre os atributos da raster no objeto ramostra. As informaÃ§Ãµes como escala espacial (resoluÃ§Ã£o e extensÃ£o) e a sistema de coordenadas (SIRGAS 2000 / UTM zone 22N , EPSG:31976) sÃ£o apresentados e representem os componentes conforme a proxima figura.

Figura 1.2: Componentes de uma raster e suas atributos no pacote terra.

AlÃ©m disso Ã© possÃvel obter informaÃ§Ãµes especÃficas atravÃ©s de funÃ§Ãµes especÃficas.

```
# Obter informaÃ§Ãµes sobre escala espacial
# resoluÃ§Ã£o, comprimento e largura do pixel em metros
res(ramostra)
# numero de colunas
ncol(ramostra)
# numero de linhas
nrow(ramostra)
```

1.3.2.1 Pergunta 1 **Sobre o objeto ramostra. Com base nos resultados obtidos, qual o Ã¡rea do pixel em metros quadrados? Qual o Ã¡rea total da paisagem em hectares e quilometros quadrados?**

Olhando a mapa (Figura 1.1), existem trÃªs classes com valores de 3, 12 e 33. O objetivo principal nÃ£o Ã© de fazer mapas, mas, a visualizaÃ§Ã£o dos dados Ã© um passo importante para verificar e entender os padrÃµes. Portanto, segue exemplo mostrando uma forma de visualizar o arquivo de raster como mapa.

Para entender o que os valores (3, 12, 33) representam no mundo real precisamos de uma referÃªncia (legenda). Para a MapBiomas ColeÃ§Ã£o 6, arquivo: Cod_Class_legenda_Col6_MapBiomas_BR.pdf. Existe tambÃ©m arquivos para fazer as mapas com cores corretas em QGIS ou ArcGIS.

Olhando a legenda (Cod_Class_legenda_Col6_MapBiomas_BR.pdf), sabemos que "3", "12" e "33" representem cobertura de "FormaÃ§Ã£o Florestal", "FormaÃ§Ã£o Campestre", e "Rio, Lago e Oceano". EntÃ£o podemos fazer um mapa mostrando tais informaÃ§Ãµes.

Daqui pra frente vamos aproveitar uma forma mais elegante de apresentar mapas e grÃ¡ficos. Isso seria atraves funÃ§oes em 2 pacotes:

- tmap Ã© utilizado para gerar mapas temÃ¡ticos
- ggplot2, que faz parte do "tidyverse", e Ã© utilizado para produÃ§Ã£o de grÃ¡ficos, e pode representar dados geoespacias.

Exemplos : Pacotes ggplot2 e tmap

Mais exemplos sobre o uso de ggplot2 no R cookbook : http://www.cookbook-r.com/Graphs/ .

E com mais exemplos de mapas e dados espaciais no R: sf e ggplot2 : https://www.r-spatial.org/r/2018/10/25/ggplot2-sf.html

Capitulo 9 no livro Geocomputation with R : https://geocompr.robinlovelace.net/adv-map.html

Primeiramente precisamos incluir as informaÃ§Ãµes relevantes da legenda. Ou seja, incluir os nomes para cada valor de classe.

```
# legenda e cores na sequencia correta
classe_valor <- c(3, 12, 33)
classe_legenda <- c("FormaÃ§Ã£o Florestal",
                    "FormaÃ§Ã£o Campestre", "Rio, Lago e Oceano")
classe_cores <- c("#006400", "#B8AF4F", "#0000FF")
```

Agora podemos fazer o mapa com as classes e os cores seguindo o padrÃ£o recomendado pela MapBiomas para ColeÃ§Ã£o 6.

```
tm_shape(ramostra) +
  tm_raster(style = "cat",
            palette = c("3" = "#006400", "12" ="#B8AF4F",
                        "33"= "#0000FF"), legend.show = FALSE) +
  tm_grid(labels.format = list(big.mark = "")) +
  tm_add_legend(type = "fill", labels = classe_legenda,
    col = classe_cores, title = "Classe") +
tm_compass(position = c("right", "bottom")) +
tm_scale_bar(breaks = c(0, 0.05, 0.1), text.size = 1,
             text.color = "white", position=c("right", "bottom")) +
tm_layout(legend.position = c("right","top"),legend.bg.color = "white")
```

Figura 1.3: Paisagem com valores e classes de cobertura da terra. Mapbiomas 2020. Uma pequena amostra do Rio Araguari, perto de Porto Grande.

1.4 Alterando a resoluÃ§Ã£o

Alterando a resoluÃ§Ã£o serve como exemplo mostrando como os passos/etapas/cÃ¡lculos mude dependendo o tipo de dados. Ou seja, Ã© preciso adotar metodologias diferentes para dados categÃ³ricos (por exemplo classificaÃ§Ã£o de cobertura da terra) e dados contÃnuos (por exemplo distÃ¢ncia atÃ© rio).

Alterando a resoluÃ§Ã£o Ã s vezes seria necessÃ¡rio, por exemplo, quando preciso padronizar dados/imagens oriundos de fontes diferentes com resoluÃ§Ãµes diferentes e/ou para reduzir a complexidade da modelagem. Lembrando - em cada nÃvel de resoluÃ§Ã£o, sÃ£o observÃ¡veis processos e padrÃµes que nÃ£o podem necessariamente ser inferidos daqueles abaixo ou acima.

A principal razÃ£o para tornar o mapa mais grosseiro Ã© traduzi-lo para uma resoluÃ§Ã£o de dados coletados em campo que estamos usando para fazer inferÃªncias. Por exemplo, se os dados foram coletados ao longo de dois transectos de 200 x 100 m dentro de manchas florestais. Se desejarmos fazer previsÃµes das relaÃ§Ãµes espÃ©cie-ambiente, podemos querer que o grÃ£o do nosso mapa reflita o grÃ£o da amostragem. Consequentemente, gostarÃamos que o mapa tivesse um grÃ£o aproximado de 200 x 200 m.

Ao escolher um tamanho de pixel apropriado, equilibre a resoluÃ§Ã£o espacial desejada â com base na unidade mÃnima de mapeamento das variaveis que vocÃª precisa analisar â com requisitos prÃ¡ticos para exibiÃ§Ã£o rÃ¡pida, tempo de processamento e armazenamento. Essencialmente, num SIG, os resultados sÃ£o tÃ£o precisos quanto o conjunto de dados menos preciso. Se vocÃª estiver usando um conjunto de dados classificados derivado de imagens de satÃ©lite com resoluÃ§Ã£o de 30 metros, o uso de um modelo digital de elevaÃ§Ã£o ou outros dados auxiliares com resoluÃ§Ã£o mais alta, como 10 metros, pode ser desnecessÃ¡rio. Quanto mais homogÃªnea for uma Ã¡rea para variÃ¡veis crÃticas, como topografia

Figura 1.4: ComparaÃ§Ã£o de tamanhos de pixels pequenos e grandes. Mostrando diferenÃ§as causadas pela agregaÃ§Ã£o de 4 pixels em 1.

e cobertura, maior poderÃ¡ ser o tamanho da cÃ©lula sem afetar a precisÃ£o.

Agora iremos degradar a resoluÃ§Ã£o desses dados, ou seja, iremos alterar o tamanho dos pixels. Como exemplo, iremos juntar (agregar) 3 pixels em um Ãºnico pixel. Como vocÃª acha que podemos fazer isso? Quais valores esse pixel que vai substituir os 3 originais deve ter? Existem diversas maneiras de se fazer isso, como por exemplo atravÃ©s a mÃ©dia ou valor modal (valor mais comum). O valor mais comum da Ã¡rea, Ã© particularmente adequado quando temos um mapa categÃ³rico, como por exemplo a classificaÃ§Ã£o do MapBiomas. Segue exemplo de codigo para agregar com a mÃ©dia e o valor mais frequente (modal).

```
# MÃ©dia
ramostra_media <- aggregate(ramostra, fact=3, fun="mean")
ramostra_media <- resample(ramostra, ramostra_media)

# Modal
ramostra_modal <- aggregate(ramostra, fact=3, fun="modal")
ramostra_modal <- resample(ramostra, ramostra_modal, method="near")
```

1.4.0.1 Pergunta 2 **Utilizando as funÃ§Ãµes disponÃveis no pacote tmap, crie mapas temÃ¡ticos dos objetos ramostra_media e ramostra_modal. Inclua cÃ³pias do seu cÃ³digo e mapas na sua resposta. VocÃª pode usar o printscreen para mostrar o RStudio com seu cÃ³digo e mapas.**

Visualizar os resultados apresentados em Figura 1.4. Os valores calculados pela mÃ©dia nÃ£o fazem sentido para uma classificaÃ§Ã£o categÃ³rica. Os valores calculados pela modal sÃ£o consistentes com o original e fazem sentido.

Em cada nÃvel de resoluÃ§Ã£o, sÃ£o observÃ¡veis processos e padrÃµes que nÃ£o podem necessariamente ser inferidos daqueles abaixo ou acima. Aqui por exemplo, mudamos a proporÃ§Ã£o de cobertura florestal em nossa pequeno paisagem quando juntamos 3 pixels em um Ãºnico: a proporÃ§Ã£o de floresta moudou de 4% (3/72) para 11% (1/9). Ou seja, com cada passo mudamos a representÃ§Ã£o do mundo.

Figura 1.5: MudanÃ§as causadas pela agregaÃ§Ã£o.

1.4.0.2 Pergunta 3 **Confira o codigo e os resultados obtidos anteriormente, quando mudamos a resoluÃ§Ã£o da raster ramostra (por exemplo figura 1.4). Explique o que aconteceu. Como e porque moudou os valores em cada caso (mÃ©dia e modal)?**

1.5 Escala espacial e desenho amostral

Figura 1.6: A cobertura florestal ao redor de um ponto de amostragem pode variar em escalas diferentes. Para entender essas variaÃ§Ãµes, podemos criar buffers circulares de diferentes extensÃµes ao redor dos pontos de amostragem. Esses buffers representam Ã¡reas de diferentes tamanhos ao redor de cada ponto. Quantificando a quantidade de floresta que ocorre em cada buffer, podemos obter uma visÃ£o geral da escala em que a cobertura florestal muda ao redor dos pontos de amostragem. Por exemplo, podemos descobrir que a cobertura florestal Ã© mais alta dentro de um buffer de 5 km do que em um buffer de 10 km.

Dado o papel que a escala pode desempenhar em nossa compreensÃ£o dos padrÃµes e processos ecolÃ³gicos, como escala deve ser considerada no desenho do estudo? Claramente, a resposta a esta pergunta irÃ¡ variar dependendo dos fenÃ´menos de interesse, mas ecologistas e estatÃsticos tÃªm forneceu algumas orientaÃ§Ãµes importantes. As questÃµes-chave incluem o tamanho da unidade de amostragem (resoluÃ§Ã£o), o tipo de unidade de amostra e localizaÃ§Ãµes da unidade de amostra, incluindo o espacamento entre as amostras (distÃ¢ncia entre as amostras) e o tamanho da Ã¡rea de estudo.

Com a disponibilidade de imagens de satÃ©lite Ã© possÃvel responder questÃµes importantes relacionadas ao desenho do estudo antes de qualquer trabalho de campo. Uma tecnica de geoprocessamento (zona tampÃ£o - Buffers) Ã© um dos mais frequentemente adotados para quantificar escala espacial na ecologia da paisagem.

O objetivo Ã© criar buffers circulares de diferentes extensÃµes ao redor dos sitios de amostragem (pontos, pixels, manchas, transetos lineares etc). Aqui, vamos entender a escala em que a cobertura de floresta muda ao redor dos rios. Para isso, quantificamos a quantidade de floresta que ocorre em vÃ¡rias distÃ¢ncias em pontos ao longo dos rios a montante das hidrelÃ©tricas no Rio Araguari. Para ilustrar esta abordagem geral, usamos o banco de dados MapBiomas ColeÃ§Ã£o 6 de 2020, e vincule esses dados de cobertura da terra aos pontos de amostragem em rios.

1.5.1 Obter e carregar dados (vectores)

Antes de quantificar a quantidade de floresta, precisamos carregar os dados de rios e pontos de amostragem. O formato de vector Ã© diferente de "tif" (raster), portanto o processo de importaÃ§Ã£o Ã© diferente. Aqui, nÃ³s sÃ³ precisamos de duas dessas camadas, ambos do pacote 'eprdados': "rio_linhacentral" e "rio_pontos". A primeira camada de dados contÃ©m o eixo central de 260 km de rios a montante da Barragem Cachoeira CaldeirÃ£o. Os dados foram obtidos a partir de registros de trajetÃ³ria GPS durante levantamentos de barco. Os rios foram divididos em 52 seÃ§Ãµes, cada uma com aproximadamente 5 km de extensÃ£o. A segunda camada de dados, "rio_pontos", contÃ©m 52 pontos espaÃ§ados regularmente ao longo dos rios. Cada ponto estÃ¡ localizado a aproximadamente 5 km de distÃ¢ncia do ponto anterior.

1.5.2 Visualizar os arquivos (camadas vector)

Visualizar para verificar. Mapa com ambos a linha central e pontos de rios em trechos de 5km.

```
ggplot(rio_linhacentral) +
  geom_sf(aes(color=rio)) +
  geom_sf(data = rio_pontos, shape=21, aes(fill=zone))
```

Figura 1.7: Pontos ao longo dos rios a montante das hidrelÃ©tricas no Rio Araguari.

Mapa interativo (funcione somente com internet) Mostrando agora com fundo de mapas "base" (OpenStreetMap/ESRI etc)

```
#
mapview(rio_linhacentral, zcol = "rio")
```

Figura 1.8: Linhas dos rios a montante das hidrelÃ©tricas no Rio Araguari.

1.5.3 Obter e carregar dados (raster)

Mais uma vez vamos aproveitar os dados de MapBiomas. Agora um arquivo raster com cobertura de terra no entorno dos rios em 2020, (formato ".tif", tamanho 1.3 MB).O cÃ³digo abaixo vai carregar os dados e criar o objeto "mapbiomas_2020":

```
meuSIGr <- system.file("raster/utm_cover_AP_rio_2020.tif",
                       package = "eprdados")
# carregar
mapbiomas_2020 <- rast(meuSIGr)
```

1.5.4 Visualizar os arquivos (camadas raster e vector)

Visualizar para verificar. Ã possÃvel de visualizar varios camadas de raster e vetor juntos com funcÃµes no pacote tmap (https://r-tmap.github.io/tmap-book/index.html).

```
# Passo necessario para agilizar o processamento
mapbiomas_2020_modal <- aggregate(mapbiomas_2020,
                                  fact = 10,
                                  fun = "modal")
# Plot
tm_shape(mapbiomas_2020_modal) +
  tm_raster(title = "Classe", style = "cat", palette = "Set3") +
tm_shape(rio_linhacentral) +
  tm_lines(col="blue") +
tm_shape(rio_pontos) +
  tm_dots(size = 0.2, col = "yellow") +
tm_compass(position=c("left", "top")) +
```

```
tm_scale_bar(breaks = c(0, 25, 50), text.size = 1,
             position=c("left", "bottom")) +
tm_layout(legend.position = c("right","top"), legend.bg.color="white")
```

Figura 1.9: Cobertura da terra ao redor do Rio Araguari em 2020. Mostrando os pontos de amostragem (pontos amarelas) cada 5 quilÃ´metros ao longo do rio.

1.5.5 ReclassificaÃ§Ã£o

Para simplificar nossa avaliaÃ§Ã£o de escala, reclassificamos a camada mapbiomas_2020 em uma camada binÃ¡ria de floresta/nÃ£o-floresta. Essa tarefa de geoprocessamento pode ser realizada anteriormente usando SIG (QGIS). Aqui vamos reclassificar as categorias de cobertura da terra (agrupando diferentes Ã¡reas de cobertura florestal tipos) usando alguns comandos genÃ©ricos do R para criar uma nova camada com a cobertura de floresta em toda a regiÃ£o de estudo. Para isso, criamos um mapa do mesmo resoluÃ§Ã£o e extensÃ£o, e entÃ£o podemos redefinir os valores do mapa. Neste caso, queremos agrupar a cobertura da terra categorias 3 e 4 (FormaÃ§Ã£o Florestal e FormaÃ§Ã£o SavÃ¢nica, respectivamente).

```
# criar uma nova camada de floresta
floresta_2020 <- mapbiomas_2020
# Com valor de 0
values(floresta_2020) <- 0
# Atualizar categorias florestais agrupados com valor de 1
floresta_2020[mapbiomas_2020 == 3 | mapbiomas_2020 == 4] <- 1
```

Vizualizar para verificar.

```
# Passo necessario para agilizar o processamento
floresta_2020_modal <- aggregate(floresta_2020,
                                 fact=10,
                                 fun="modal")
# Plot
tm_shape(floresta_2020_modal) +
  tm_raster(style = "cat",
            palette = c("0" = "#E974ED", "1" ="#129912"), legend.show = FALSE) +
  tm_add_legend(type = "fill", labels = c("nÃ£o-floresta", "floresta"),
    col = c("#E974ED", "#129912"), title = "Classe") +
tm_shape(rio_linhacentral) +
  tm_lines(col="blue") +
tm_shape(rio_pontos) +
  tm_dots(size = 0.2, col = "yellow") +
tm_scale_bar(breaks = c(0, 25, 50), text.size = 1,
             text.color = "white", position=c("left", "bottom")) +
tm_layout(legend.position = c("right","top"),legend.bg.color = "white")
```

Figura 1.10: Floresta ao redor do Rio Araguari. MapBiomas 2020 reclassificado em floresta e nÃ£o-floresta. Mostrando os pontos de amostragem (pontos amarelas) cada 5 quilÃ´metros ao longo do rio.

1.6 ComparaÃ§Ã£o multiescala

Em seguida, com as coordenadas dos pontos de amostragem, podemos calcular a quantidade de floresta que circunda cada local de amostragem em diferentes extensÃµes. Primeiramente, vamos fazer so para um ponto, assim para entender o processo e os passos melhor.

```
rio_pontos_31976 <- st_transform(rio_pontos, 31976)
# Buffer
rio_pontos_31976_b1000 <- st_buffer(rio_pontos_31976[1, ], dist = 1000)

# Recorte com buffer de 1000 metros (mudando a extensÃ£o).
# MÃ¡scara (mask), assimos pixels fora do polÃgono sejam nulos.
buffer.forest1.1km <- terra::crop(floresta_2020, rio_pontos_31976_b1000,
                          snap="out", mask = TRUE)
names(buffer.forest1.1km) <- "forest_2020_1km"
```

Vizualizar para verificar.

```
# Plot
tm_shape(buffer.forest1.1km) +
  tm_raster(style = "cat",
            palette = c("0" = "#E974ED",
                        "1" ="#129912"), legend.show = FALSE) +
tm_shape(rio_pontos_31976[1, ]) +
  tm_symbols(shape =21, col = "yellow",
            border.col = "black", border.lwd = 0.2, size=0.5) +
tm_shape(rio_pontos_31976_b1000) +
  tm_borders(col = "black", lwd = 4, lty = "dashed") +
tm_add_legend(type = "fill",  labels = c("nÃ£o-floresta", "floresta"),
    col = c("#E974ED", "#129912"), title = "Classe") +
tm_compass(position=c("left", "top")) +
tm_scale_bar(breaks = c(0, 0.5, 1), text.size = 1,
             position=c("left", "bottom")) +
tm_layout(legend.position = c("right","top"), legend.bg.color = "white")
```

Figura 1.11: IlustraÃ§Ã£o da determinaÃ§Ã£o da quantidade de habitat ao redor de um ponto. Para um determinada extensÃ£o, o habitat de interesse Ã© isolado. Um buffer (linha tracejada) Ã© colocado ao redor de um ponto (amarela) e o nÃºmero de cÃ©lulas (pixels) que contÃ©m o habitat Ã© somado e multiplicado pela Ã¡rea de cada pixel.

1.6.0.1 Pergunta 4 **Qual Ã© a extensÃ£o em nÃºmero de pixels desse recorte (buffer.forest1.1km)?**

Temos valores de 0 (nÃ£o-floresta) e 1 (floresta). EntÃ£o, para saber a arÃ©a de floresta podemos somar o nÃºmero de cÃ©lulas (pixels) que contÃ©m o habitat e multiplica pela Ã¡rea de cada pixel conforme o codigo:

```
# 1) SomatÃ³rio.
# No caso igual o numero de pixels de floresta.
# Para todo a paisagem, somatorio "global".
# NÃ£o deve incluir pixels nulos, entÃ£o use "na.rm = TRUE".
soma_floresta <- global(buffer.forest1.1km, "sum", na.rm = TRUE)
soma_floresta
```

```
##                 sum
## forest_2020_1km 939
```

```
# 2) Ã rea de cada pixel.
# Sabemos o sistema de coordenadas (EPSG = 31976).
# EPSG 31976 Ã© uma sistema projetado com unidade em metros.
buffer.forest1.1km
```

```
## class       : SpatRaster
## dimensions  : 68, 68, 1  (nrow, ncol, nlyr)
## resolution  : 29.89281, 29.89281  (x, y)
## extent      : 465959.3, 467992, 90921.47, 92954.18  (xmin, xmax, ymin, ymax)
## coord. ref. : SIRGAS 2000 / UTM zone 22N (EPSG:31976)
## source(s)   : memory
## varname     : utm_cover_AP_rio_2020
## name        : forest_2020_1km
## min value   :               0
## max value   :               1
```

```
# Portanto, o tamanho de cada pixel Ã© igual.
area_pixel_m2 <- 29.89281 * 29.89281
area_pixel_m2
```

```
## [1] 893.5801
```

```
# 3) Calculos de arÃ©a.
# ArÃ©a de floresta m2
area_floresta_m2 <- soma_floresta * area_pixel_m2
area_floresta_m2
```

```
##                      sum
## forest_2020_1km 839071.7
```

```
# ArÃ©a de floresta hectares
area_floresta_ha <- area_floresta_m2 / 10000
area_floresta_ha
```

```
##                      sum
## forest_2020_1km 83.90717
```

Para uma comparaÃ§Ã£o multiescala, vamos repetir o mesmo processo, mas agora com distancias de 250, 500, 1000, 2000 e 4000 metros, doubrando a escala (extensÃ£o) em cada passo.

Figura 1.12: Cobertura florestal em extensÃµes diferentes ao redor de um local de amostragem.

Aspectos quantitativos das paisagens mudam fundamentalmente com a escala. Por exemplo, nesse caso, parece que a proporÃ§Ã£o de floresta aumenta Ã medida que a extensÃ£o aumenta de 500 para 4000 metros. Esta percepÃ§Ã£o visual Ã© confirmada pelos valores calculados, onde as Ã¡reas sÃ£o:

- raio 250 m = 0 hectares de floresta
- raio 500 m = 6,3 hectares de floresta
- raio 1000 m = 84,3 hectares de floresta
- raio 2000 m = 502.6 hectares de floresta
- raio 4000 m = 3351.0 hectares de floresta

1.6.0.2 Pergunta 5 **Usando os valores listadas acima de raio e Ã¡rea de floresta para os diferentes buffers circulares, calcule a proporÃ§Ã£o de floresta em cada uma das diferentes extensÃµes de buffer. Apresente 1) os resultados incluindo cÃ¡lculos. 2) um grÃ¡fico com valores de extensÃ£o no eixo x e proporÃ§Ã£o da floresta no eixo y. 3) Em menos de 200 palavras apresente a sua interpretaÃ§Ã£o do grÃ¡fico.**

1.6.0.3 Pergunta 6 **A modelagem multiescala quantifica as condiÃ§Ãµes do ambiente em mÃºltiplas escalas alterando o resoluÃ§Ã£o ou a extensÃ£o da anÃ¡lise e, em seguida, avaliando qual das escalas consideradas explica melhor um padrÃ£o ou processo. Escolha 1 espÃ©cie aquÃ¡tico e 1 espÃ©cie terrestre que ocorram na regiÃ£o a montante das hidrelÃ©tricas no Rio Araguari (especificando nome comum junto com o nome cientÃfico para ambos). Com base nas diferenÃ§as entre extensÃµes (indicados no exemplo anterior) e as caracterÃsticas funcionais das espÃ©cies (por exemplo Ã¡rea de vida), escolher as extensÃµes mais adequadas para um estudo multiescala de cada espÃ©cie.**

Essa foi a Ãºltima pergunta.

SoluÃ§Ãµes dos exercÃcios: https://darrennorris.github.io/eprsol/solutions01_escala.html

1.7 PrÃ³ximos passos: repetindo para muitas amostras.

Neste exemplo comparamos a Ã¡rea de floresta em torno de um Ãºnico ponto de amostragem. Para calcular o mesmo para todos os 52 pontos, seriam necessÃ¡rias varias repetiÃ§Ãµes (52 pontos x 5 extensÃµes = 260 repetiÃ§Ãµes).

PoderÃamos escrever cÃ³digo para executar esse processo automaticamente. Felizmente, alguÃ©m jÃ¡ escreveu funÃ§Ãµes para fazer isso e muito mais. O prÃ³ximo tutorial sobre mÃ©tricas de paisagem mostrarÃ¡ exemplos usando o pacote "landscapemetrics" (https://r-spatialecology.github.io/landscapemetrics/).

2 MÃ©tricas da paisagem

2.1 ApresentaÃ§Ã£o

As mÃ©tricas da paisagem nos ajudam a entender as mudanÃ§as na paisagem de diferentes perspectivas (visual, ecolÃ³gica, cultural).

Asssim sendo, anÃ¡lises com mÃ©tricas de paisagem Ã© um atividade fundamental na ecologia da paisagem. Nesta capitulo aprenderemos sobre como analisar a cobertura da terra com mÃ©tricas de paisagem em R. o objetivo principal Ã© mostrar a importÃ¢ncia de selecionar a mÃ©trica â certaâ para o problema â certoâ usando R. As tecnicas serÃ¡ ilustrada atravÃ©s cÃ¡lculos usando a cobertura florestal ao redor do Rio Araguari, AmapÃ¡, Brasil.

Ao longo do caminho, revisaremos modelos lineares e nÃ£o lineares, aprenderemos sobre manipulaÃ§Ã£o de dados em R e aprenderemos como criar grÃ¡ficos com o pacote ggplot2. No capitulo vocÃª aprenderÃ¡ a:

- Importar e plotar dados raster em R e mapear locais de amostragem com os pacotes terra, sf e tmap.
- Calcular mÃ©tricas de paisagem com o pacote landscapemetrics.
- Calcular mÃ©tricas de paisagem em locais de amostragem e dentro de um buffer ao redor deles (comparaÃ§Ã£o multiescala).
- Construir grÃ¡ficos com o pacote ggplot2.
- ComparaÃ§Ã£o de padrÃµes lineares e nÃ£o-lineares.

Para ajudar a acompanhar e entender os exemplos no capÃtulo, use como base as aulas ("MÃ©tricas da paisagem: Modelo mancha-corredor-matriz" e "Ã ndices de Paisagem e AnÃ¡lises de PadrÃµes Espaciais") e vocÃª deve ler os seguintes artigos:

- Tischendorf, L. 2001.
  Can landscape indices predict ecological processes consistently?. Landscape Ecology 16, 235â "254. https://doi.org/10.1023/A:1011112719782
- Casimiro PC. 2009.
  Estrutura, composiÃ§Ã£o e configuraÃ§Ã£o da paisagem conceitos e princÃpios para a sua quantificaÃ§Ã£o no Ã¢mbito da ecologia da paisagem. Revista portuguesa de estudos regionais. (20):75-99. https://www.redalyc.org/pdf/5143/514351895006.pdf
- Pereira JL, et. al. 2001.
  MÃ©tricas da paisagem na caracterizaÃ§Ã£o da evoluÃ§Ã£o da ocupaÃ§Ã£o da AmazÃ´nia. Geografia. 2001:59-90. https://www.periodicos.rc.biblioteca.unesp.br/index.php/ageteo/article/view/1907
- McGarigal K. 2017.
  Landscape metrics for categorical map patterns. Lecture notes. https://opencourses.ionio.gr/modules/document/file.php/TFP122/%CE%98%CE%95%CE%A9%CE%A1%CE%99%CE%91/%CE%94%CE%95%CE%99%CE%9A%CE%A4%CE%95%CE%A3%20%CE%A4%CE%9F%CE%A0%CE%99%CE%9F%CE%A5/Landscape%20metrics.pdf

CÃ³pias dos artigos estÃ£o no Google Classroom e tambÃ©m podem ser baixadas aqui: https://drive.google.com/drive/folders/1n6KrIiqu8IPvKLdhcKqwpI5o3MB_4gEu?usp=sharing

2.2 MÃ©tricas da paisagem e pacote "landscapemetrics"

As mÃ©tricas de paisagem sÃ£o a forma que os ecÃ³logos de paisagem usam para descrever os padrÃµes espaciais de paisagens para depois avaliar a influÃªncia destes padrÃµes espaciais nos padrÃµes e processos ecolÃ³gicos.

A paisagem objeto de anÃ¡lise Ã© definida pelo usuÃ¡rio e pode representar qualquer fenÃ´meno espacial. As mÃ©tricas simplesmente quantifica a heterogeneidade espacial da paisagem; cabe ao usuÃ¡rio estabelecer uma base sÃ³lida para definir e dimensionar a paisagem em termos de conteÃºdo temÃ¡tico e resoluÃ§Ã£o e extensÃ£o espacial.

Ã importante ressaltar que o resultado das mÃ©tricas sÃ³ Ã© relevante se a paisagem definida for adequado em relaÃ§Ã£o ao fenÃ´meno em consideraÃ§Ã£o.

O uso comum do termo "mÃ©tricas de paisagem" refere-se exclusivamente a Ãndices desenvolvidos para padrÃµes de mapas categÃ³ricos. MÃ©tricas de paisagem sÃ£o algoritmos que quantificam caracterÃsticas espaciais especÃficas de manchas, classes de manchas ou mosaicos inteiros de paisagem. Uma infinidade de mÃ©tricas foi desenvolvida para quantificar padrÃµes de mapas categÃ³ricos. Estas mÃ©tricas enquadram-se em duas categorias gerais: aquelas que quantificam a composiÃ§Ã£o do mapa sem referÃªncia a atributos espaciais, e aquelas que quantificam a configuraÃ§Ã£o espacial do mapa, necessitando de informaÃ§Ã£o espacial para o seu cÃ¡lculo McGarigal & Marks 1995, https://doi.org/10.2737/PNW-GTR-351.

O pacote landscapemetrics tem funÃ§Ãµes para calcular mÃ©tricas de paisagem em paisagems categÃ³ricos (onde tem uma classificaÃ§Ã£o de cobertura de terra/habitat - modelo mancha-corredor-matriz), em um fluxo de trabalho organizado [@Hesselbarth2019]. Existem diversos tutoriais e exemplos no site do proprio pacote: https://r-spatialecology.github.io/landscapemetrics/ .

O pacote landscapemetrics pode ser usado como um substituto do FRAGSTATS (McGarigal & Marks 1995, https://doi.org/10.2737/PNW-GTR-351), pois oferece um fluxo de trabalho reproduzÃvel para anÃ¡lise de paisagem em um Ãºnico ambiente. FRAGSTATS Ã© um programa de anÃ¡lise de padrÃµes espaciais para mapas categÃ³ricos disponÃvel apenas para a sistema operacional Windows. Pode obter FRAGSTATS aqui: https://fragstats.org/ . AlÃ©m de mÃ©tricas padrÃ£o landscapemetrics tambÃ©m permite cÃ¡lculos de quatro mÃ©tricas teÃ³ricas de complexidade da paisagem: entropia marginal, entropia condicional, entropia conjunta e informaÃ§Ã£o mÃºtua (Nowosad e Stepinski 2019 https://doi.org/10.1007/s10980-019-00830-x).

2.2.1 Pacotes

AlÃ©m do "landscapemetrics", precisamos carregar alguns pacotes a mais para facilitar a organizaÃ§Ã£o e apresentaÃ§Ã£o de dados espaciais (vector e raster) e os resultados.

Carregar pacotes (que deve esta instalado antes):

```
library(tidyverse)
library(sf)
library(terra)
library(tmap)
library(mgcv)
library(eprdados)
```

Caso os pacotes nÃ£o tenham sido instalados, o R vai avisar atraves um mensagem tipo: Error in library(nomepacote) there is no package called nomepacote. Neste caso, para instalÃ¡-los consulte o CapÃtulo 6 ou os capitulos aqui Capitulo 4 instalaÃ§Ã£o de pacotes e aqui Capitulo 4 pacotes . Prestando atenÃ§Ã£o nas diferenÃ§as para instalar um pacote como `eprdados` que vem do GitHub.

2.2.1.1 landscapemetrics Agora, vamos para o pacote principal landscapemetrics. Digite o cÃ³digo abaixo e veja o resultado. Leia com atenÃ§Ã£o e preste particular atenÃ§Ã£o na organizaÃ§Ã£o da pÃ¡gina de ajuda.

```r
library(landscapemetrics)
# Olhar a pagina de ajuda
?landscapemetrics
```

No final da pÃ¡gina vocÃª vai encontrar a palavra "Index". Clique nela e vocÃª verÃ¡ todas as funÃ§Ãµes do pacote. Desca atÃ© as lsm_. . . e clique em algumas delas ali. Explorar! Para listar todas as mÃ©tricas disponÃveis, vocÃª pode usar a funÃ§Ã£o `list_lsm()` . A funÃ§Ã£o tambÃ©m permite mostrar mÃ©tricas filtradas por nÃvel, tipo ou nome da mÃ©trica. Para obter mais informaÃ§Ãµes sobre as mÃ©tricas, consulte os arquivos de ajuda correspondentes ou https://r-spatialecology.github.io/landscapemetrics.

Digite o cÃ³digo abaixo e veja o resultados, mostrando exemplos das mÃ©tricas diferentes disponÃveis no pacote.

```r
# mÃ©tricas de agregaÃ§Ã£o, nÃvel de fragmento
landscapemetrics::list_lsm(level = "patch", type = "aggregation metric")
# mÃ©tricas de agregaÃ§Ã£o, nÃvel de classe
landscapemetrics::list_lsm(level = "class", type = "aggregation metric")
#
landscapemetrics::list_lsm(metric = "area")
# ajudar com opÃ§Ãµes da funÃ§Ã£o
?landscapemetrics::list_lsm
```

Nesse pacote o formato geral para uma funÃ§Ã£o Ã© em trÃ©s partes "lsm_nÃvel_mÃ©trica". A primeira parte Ã© sempre lsm_ ("landscapemetric"), seguinda do â nÃvel_â e por fim a â mÃ©tricaâ :

1. Ou seja, todas as funÃ§Ãµes que calculam mÃ©tricas comeÃ§am com lsm_ …….
2. DaÃ vocÃª deve incluir o nÃvel da anÃ¡lise: â pâ , "c" ou "l".
   Sendo, â pâ para patch (ou seja, para a mancha/fragmento), â câ para classe e â lâ para landscape ou seja, mÃ©tricas para a paisagem como um todo.
3. E daÃ existem inÃºmeras mÃ©tricas.
   Como por exemplo a `cpland`, que Ã© o percentual de Ã¡rea central - â core areaâ na paisagem, como vimos na aula teÃ³rica. Assim sendo, a funÃ§Ã£o `lsm_c_cpland` vai calacular a mÃ©trica porcentagem da Ã¡rea central em cada classe. Lembrando existem metricas que podem se calculados nos trÃ©s niveis, e metricas que so pode se calculados somente para um nivel espcifco. Ja sabendo o nome da funÃ§Ã£o, podemos buscar ajudar para entender mais detalhes. Digite o cÃ³digo abaixo e veja o resultado, mostrando um exemplo para uma mÃ©trica.

```r
# ajudar com opÃ§Ãµes para uma funÃ§Ã£o especifica
?landscapemetrics::lsm_c_cpland
```

2.2.1.2 Pergunta 1 **Descreva brevemente 2 mÃ©tricas de cada nÃvel (patch, class, landscape) usando ajudar (usando ? e/ou list_lsm), aulas (MÃ©tricas da paisagem: Modelo mancha-corredor-matriz e "Ã ndices de Paisagem e AnÃ¡lises de PadrÃµes Espaciais") e/ou a leitura disponivel no Google Classroom (Base teÃ³rica 4 Dados, mÃ©tricas, analises). Incluindo na descriÃ§Ã£o - o nome, porque serve, unidades de medida, e relevÃ¢ncia ecolÃ³gica.**

2.3 Dados

Vamos continuar as analises que comeÃ§o no capÃtulo "Escala", trabalhando com os mesmos bancos de dados. EntÃ£o, primeiramente carregar os dados de cobertura da terra com a funÃ§Ã£o rast. E, em seguida implementar uma reclassificaÃ§Ã£o para gerar uma camada binaria de floresta (valor de "1") e nÃ£o-floresta (valor de "0").

```
# Carregar
mapbiomas_2020 <- rast(utm_cover_AP_rio_2020)

# ReclassificaÃ§Ã£o -
# Criar uma nova camada de floresta (novo objeto de raster copiando mapbiomas_2020,
# assim para ter os mesmos coordenados, resoluÃ§Ã£o e extensÃ£o)
floresta_2020 <- mapbiomas_2020
# Todos os pixels com valor de 0
values(floresta_2020) <- 0
# Atualizar com valor de 1 quando pixels originais sÃ£o de floresta (classe 3 e 4)
floresta_2020[mapbiomas_2020==3 | mapbiomas_2020==4] <- 1
```

Plotar para verificar, incluindo nomes e os cores para classes de floresta (valor = 1) e nÃ£o-floresta (valor = 0).

```
# Passo necessario para agilizar o processamento
floresta_2020_modal <- aggregate(floresta_2020,
                                 fact=10,
                                 fun="modal")
# Plot
tm_shape(floresta_2020_modal) +
  tm_raster(style = "cat",
            palette = c("0" = "#E974ED", "1" ="#129912"), legend.show = FALSE) +
  tm_add_legend(type = "fill", labels = c("nÃ£o-floresta", "floresta"),
    col = c("#E974ED", "#129912"), title = "Classe") +
tm_layout(legend.bg.color = "white")
```

Se esta todo certo, voces devem ter uma imagem assim:

Figura 2.1: Floresta ao redor do Rio Araguari. MapBiomas 2020 reclassificado em floresta e nÃ£o-floresta.

2.3.1 Exibir dados raster e sobreposiÃ§Ã£o com locais de amostragem

Agora temos a paisagem, precisamos tambem os pontos de amostra. Por isso, precisamos carregar os dados de rios e pontos de amostragem que usamos no tutorial Escala. Vamos carregar as camadas, e no mesmo

tempo implementar uma reprojeÃ§Ã£o, assim para que as sistemas de coordenados ficam iguais para todas as camadas - tanto de vector quanto raster.

No codigo abaixo, usamos |>, que estabelece a ligaÃ§Ã£o entre os passos do processo. Ou seja, |> passa o objeto anterior automaticamente para a prÃ³xima funÃ§Ã£o como primeiro argumento. Primeiro, carregamos os dados. Em seguida, usamos a funÃ§Ã£o `st_transform` para converter as coordenadas para o mesmo sistema de referÃªncia do arquivo raster. As duas etapas ficam ligados atravÃ©s a funÃ§Ã£o |>.

```
# pontos cada 5 km
rio_pontos_31976 <- rio_pontos |>
  st_transform(31976)
# linha central de rios
rio_linhacentral_31976 <- rio_linhacentral |>
  st_transform(31976)
```

Visualizer para verificar.

```
# Passo necessario para agilizar o processamento
floresta_2020_modal <- aggregate(floresta_2020,
                                 fact=10,
                                 fun="modal")
# Mapa
tm_shape(floresta_2020_modal) +
  tm_raster(style = "cat",
            palette = c("0" = "#E974ED", "1" ="#129912"), legend.show = FALSE) +
  tm_add_legend(type = "fill", labels = c("nÃ£o-floresta", "floresta"),
    col = c("#E974ED", "#129912"), title = "Classe") +
tm_shape(rio_linhacentral_31976) +
  tm_lines(col="blue") +
tm_shape(rio_pontos_31976) +
  tm_dots(size = 0.2, col = "yellow") +
tm_layout(legend.bg.color="white")
```

Depois de executar ("run") o cÃ³digo acima, vocÃª deverÃ¡ ver a figura a seguir.

Figura 2.2: Cobertura da terra ao redor do Rio Araguari em 2020. Mostrando os pontos de amostragem (pontos amarelas) cada 5 quilÃ´metros ao longo do rio (linha azul).

2.4 Calculo de mÃ©tricas

Agora com todos as camadas organizados e verifcados podemos calcular as mÃ©tricas de paisagem. Para ilustrar como rodar as funÃ§Ãµes e cÃ¡lculos com landscapemetrics, vamos calcular o percentual de Ã¡rea central na paisagem (`cpland`). Vamos estudar uma classe (floresta), portanto vamos incluir as mÃ©tricas para nÃvel de classe. No geral, as mÃ©tricas de paisagem em nÃvel de classe sÃ£o mais eficazes na definiÃ§Ã£o de processos ecolÃ³gicos (Tischendorf, L. Can landscape indices predict ecological processes consistently?. Landscape Ecology 16, 235â "254 (2001). https://doi.org/10.1023/A:1011112719782.). No entanto, por vezes, os Ãndices de manchas podem ser importantes e informativos nas investigaÃ§Ãµes ao nÃvel da paisagem. Por exemplo, muitos vertebrados requerem manchas de habitat maiores do que um tamanho mÃnimo, por isso seria Ãºtil saber tambÃ©m o tamanho de cada mancha na paisagem.

2.4.1 Uso de pontos de amostragem na ecologia da paisagem

Na ecologia da paisagem, os pontos de amostragem sÃ£o ferramentas cruciais para a coleta de dados e a compreensÃ£o dos padrÃµes e processos espaciais que ocorrem em diferentes paisagens. Eles atuam como janelas para o complexo mosaico de ecossistemas, fornecendo informaÃ§Ãµes valiosas sobre fatores como:

- ComposiÃ§Ã£o e distribuiÃ§Ã£o de espÃ©cies:
  Os pontos de amostragem podem ser usados para registrar a presenÃ§a e abundÃ¢ncia de diferentes espÃ©cies em locais especÃficos, permitindo aos pesquisadores mapear padrÃµes de biodiversidade e identificar Ã¡reas com alto valor de conservaÃ§Ã£o.

- CaracterÃsticas do habitat:
  Ao medir vÃ¡rias variÃ¡veis ambientais em pontos de amostragem, como cobertura vegetal, umidade do solo e caracterÃsticas topogrÃ¡ficas, os pesquisadores podem caracterizar os diferentes tipos de habitat dentro de uma paisagem e compreender como eles influenciam a distribuiÃ§Ã£o de espÃ©cies e os processos ecolÃ³gicos.

- Conectividade da paisagem:
  Os pontos de amostragem podem ser usados para avaliar a conectividade entre diferentes manchas de habitat, o que Ã© crucial para o movimento e dispersÃ£o de espÃ©cies. Essas informaÃ§Ãµes podem informar estratÃ©gias de conservaÃ§Ã£o e manejo destinadas a manter paisagens saudÃ¡veis e funcionais.

- MudanÃ§a da paisagem:
  A amostragem repetida dos mesmos pontos ao longo do tempo pode fornecer dados valiosos sobre como as paisagens estÃ£o mudando. Essas informaÃ§Ãµes podem ser usadas para monitorar os impactos das atividades humanas, perturbaÃ§Ãµes naturais e mudanÃ§as climÃ¡ticas nas comunidades ecolÃ³gicas.

Os mÃ©todos especÃficos usados para amostragem de pontos em ecologia da paisagem variam dependendo da questÃ£o de pesquisa e do tipo de dados coletados. Alguns mÃ©todos comuns incluem:

- Amostragem aleatÃ³ria:
  Os pontos sÃ£o distribuÃdos aleatoriamente por toda a paisagem para garantir uma representaÃ§Ã£o imparcial de toda a Ã¡rea.

- Amostragem estratificada:
  Os pontos sÃ£o localizados em diferentes tipos de habitat ou outros estratos da paisagem para garantir que todos os componentes da paisagem sejam amostrados adequadamente.

- Amostragem sistemÃ¡tica:
  Os pontos sÃ£o localizados em intervalos regulares ao longo de transectos ou grades dispostas sobre a paisagem. Amostragens sistemÃ¡ticas sÃ£o adotados como padrÃ£o dentro do Programa de Pesquisa em Biodiversidade: https://ppbio.inpa.gov.br/ .

A escolha do mÃ©todo de amostragem e do nÃºmero de pontos de amostragem necessÃ¡rios dependerÃ¡ do projeto especÃfico do estudo e do nÃvel de precisÃ£o desejado.

Neste capitulo, vamos calcular as mÃ©tricas de paisagem dentro de um certo buffer em torno de pontos de amostra. Ou seja, "buffers" de diferentes tamanhos serÃ¡ usado para medir a paisagem circundante aos locais de amostragem. Usando o pacote landscapemetrics, podemos calcular as mÃ©tricas de paisagem dentro de um certo buffer em torno de pontos de amostra, com a funÃ§Ã£o `sample_lsm()`. AtravÃ©s da funÃ§Ã£o `sample_lsm()` podemos calcular mais de 50 mÃ©tricas da paisagem, dentro de buffers (raios/distancias) diferentes.

Aqui vamos calcular as mÃ©tricas em trÃªs exemplos de caso:

1. Ponto Ãºnico, raio Ãºnico, mÃ©trica Ãºnica.
2. Ponto Ãºnico, distÃ¢ncias variados, mÃ©trica Ãºnica.
3. Ponto Ãºnico, distÃ¢ncias variados, mÃ©tricas variadas.

2.4.2 Ponto Ãºnico, raio Ãºnico, mÃ©trica Ãºnica

MÃ©tricas de Ã¡rea central ("core area") sÃ£o consideradas medidas da qualidade de hÃ¡bitat, uma vez que indica quanto existe realmente de Ã¡rea efetiva de um fragmento/classe, apÃ³s descontar-se o efeito de borda. Vamos calcular a percentual de Ã¡rea central ("core area") no entorno de um ponto de amostragem. Isso seria, a percentual de Ã¡reas centrais (excluÃdas as bordas de 30 m) de cada classe em relaÃ§Ã£o Ã Ã¡rea total da paisagem.

Para a funÃ§Ã£o `sample_lsm()` funcionar, precisamos informar (i) a paisagem (arquivo de raster), (ii) ponto de amostragem (arquivo vector), (iii) raio do buffer desejada, (iv) forma do buffer (cÃrculo ou quadrado) e por final (v) a mÃ©trica desejada. Cada opÃ§Ã£o tem especificaÃ§Ãµes particulares, assim para que a funÃ§Ã£o pode receber dados em formatos diferentes e produzir resultados conforme as necessidades de diversos casos.

Para este primeiro exemplo trabalharemos com apenas um ponto. Fazemos isso selecionando a primeira linha (primeiro ponto) do objeto com os 52 pontos de amostragem ("rio_pontos_31976") usando colchetes: `[1, ]`.

```
# com a paisagem "floresta_2020",
# fazer uma amostragem de um ponto "rio_pontos_31976[1, ]"
# dentro de uma buffer com raio de 1000 metros e forma circular
# e calcular a mÃ©trica "cpland"

minha_amostra_1000 <- sample_lsm(landscape = floresta_2020,
                                 y = rio_pontos_31976[1, ],
                                size = 1000,
                                shape = "circle",
                                metric = "cpland",
                                edge_depth = 1)
```

Depois que executar ("run"), podemos olhar os dados com o codigo a seguir. Rodando o nome do obejto, como no proximo bloco de codigo podemos verifcar os resultados.

```
minha_amostra_1000
```

Os dados deve ter os valores (coluna "value") da mÃ©trica (coluna "metric") de cada classe (coluna "class") conforme a proxima tabela:

layer	level	class	id	metric	value	plot_id	percentage_inside
1	class	0	NA	cpland	66.95102	1	103.3533
1	class	1	NA	cpland	19.59274	1	103.3533

2.4.2.1 Pergunta 2 **O modelo mancha-corredor-matriz Ã© frequentemente adotado na ecologia da paisagem. Com base nas aulas teÃ³ricas e usando os valores no objeto minha_amostra_1000 apresentados na tabela acima, identificar qual classe representar a matriz na paisagem. HÃ¡ alguma informaÃ§Ã£o faltando que limita a sua capacidade de identificar qual classe representar a matriz? Se sim, o que precisa ser adicionado? Justifique as suas respostas de forma clara e concisa.**

2.4.3 Ponto Ãºnico, distÃ¢ncias variados, mÃ©trica Ãºnica

Figura 2.3: Cobertura florestal em extensÃµes diferentes ao redor de um ponto de amostragem.

Para uma comparaÃ§Ã£o multiescala, vamos calcular a mesma mÃ©trica, no mesmo ponto, mas agora com extensÃµes diferentes. Continuando o exemplo no tutorial anteriio (Escala), vamos repetir o mesmo processo, mas agora com raios de 250, 500, 1000, 2000, 4000, 8000 e 16000 metros, doubrando a escala (extensÃ£o) em cada passo.

Para obter resultados com extensÃµes diferentes, precisamos primeiramente repetir o cÃ³digo, ajustando para cada extensÃ£o, e depois juntar os resultados. O cÃ³digo a seguir calcularÃ¡ a mesma mÃ©trica para as diferentes distÃ¢ncias. No exemplo, usamos |>, que estabelece a ligaÃ§Ã£o entre os passos do processo. Neste caso, usamos a funÃ§Ã£o `mutate()` para incluir uma coluna nova ("raio") para manter o valor das diferentes distÃ¢ncias. `mutate()` Ã© uma funÃ§Ã£o muito versÃ¡til que pode ser usada para criar, modificar e excluir colunas em um quadro de dados (data frame). Mais exemplos com o uso de `mutate()` aqui: https://analises-ecologicas.com/cap5#mutate.

Uso de `mutate()` seguir com a sintaxe: o quadro de dados serÃ¡ sempre o primeiro argumento, seguido de um operador pipe (`|>`) e pelo nome da funÃ§Ã£o `mutate()`. Isso permite o encadeamento de vÃ¡rias operaÃ§Ãµes consecutivas mantendo a estrutura do dado original e acrescentando mudanÃ§as num encadeamento lÃ³gico.

Sendo assim, `mutate()` nÃ£o precisa modificar necessariamente os dados originais, sendo que as operaÃ§Ãµes de manipulaÃ§Ãµes podem e devem ser atribuÃdas a um novo objeto.

```
# raio 250 metros
sample_lsm(floresta_2020, y = rio_pontos_31976[1, ],
           size = 250, shape = "circle",
           metric = "cpland") |>
  mutate(raio = 250) -> minha_amostra_250
# raio 500 metros
sample_lsm(floresta_2020, y = rio_pontos_31976[1, ],
           size = 500, shape = "circle",
           metric = "cpland")  |>
  mutate(raio = 500) -> minha_amostra_500
# raio 1 km (1000 metros)
sample_lsm(floresta_2020, y = rio_pontos_31976[1, ],
           size = 1000, shape = "circle",
           metric = "cpland")  |>
  mutate(raio = 1000) -> minha_amostra_1000
# raio 2 km
sample_lsm(floresta_2020, y = rio_pontos_31976[1, ],
           size = 2000, shape = "circle",
           metric = "cpland") |>
  mutate(raio = 2000) -> minha_amostra_2000
# raio 4 km
sample_lsm(floresta_2020, y = rio_pontos_31976[1, ],
           size = 4000, shape = "circle",
           metric = "cpland")  |>
  mutate(raio = 4000) -> minha_amostra_4000
# raio 8 km
sample_lsm(floresta_2020, y = rio_pontos_31976[1, ],
           size = 8000, shape = "circle",
           metric = "cpland")  |>
  mutate(raio = 8000) -> minha_amostra_8000
# raio 16 km
sample_lsm(floresta_2020, y = rio_pontos_31976[1, ],
           size = 16000, shape = "circle",
           metric = "cpland")  |>
  mutate(raio = 16000) -> minha_amostra_16000
```

E agora, o cÃ³digo a seguir juntarÃ¡ os resultados das diferentes extensÃµes. Para combinar as linhas de duas ou mais tabelas de dados podemos utilizar a funÃ§Ã£o `bind_rows()`. Para esta funÃ§Ã£o, precisamos incluir os nomes de todos os dados que queremos combinar, separados por vÃrgulas. Aqui combinamos as mÃ©tricas de sete buffers separados e criamos um novo objeto "amostras_metrica".

```
bind_rows(minha_amostra_250,
          minha_amostra_500,
          minha_amostra_1000,
          minha_amostra_2000,
          minha_amostra_4000,
          minha_amostra_8000,
          minha_amostra_16000) -> amostras_metrica
```

Depois que executar ("run"), podemos olhar os dados "amostras_metrica" com o codigo a seguir.

```
amostras_metrica
```

Os dados deve ter os valores (coluna value) da mÃ©trica (coluna metric) de cada classe (coluna class) para cada distÃ¢ncia (coluna raio):

layer	level	class	id	metric	value	plot_id	percentage_inside	raio
1	class	0	NA	cpland	80.3	1	113	250
1	class	0	NA	cpland	86.4	1	106	500
1	class	1	NA	cpland	0.7	1	106	500
1	class	0	NA	cpland	67.0	1	103	1000
1	class	1	NA	cpland	19.6	1	103	1000
1	class	0	NA	cpland	59.1	1	102	2000
1	class	1	NA	cpland	27.9	1	102	2000
1	class	0	NA	cpland	41.9	1	101	4000
1	class	1	NA	cpland	44.2	1	101	4000
1	class	0	NA	cpland	39.6	1	100	8000
1	class	1	NA	cpland	48.5	1	100	8000
1	class	0	NA	cpland	39.2	1	100	16000
1	class	1	NA	cpland	50.7	1	100	16000

2.4.3.1 FaÃ§a um grÃ¡fico

Uma imagem vale mais que mil palavras. Portanto, grÃ¡ficos/figuras/imagens sÃ£o uma das mais importantes formas de comunicar a ciÃªncia. Os dados apresentados em uma tabela podem ser difÃceis de entender. Portanto, a primeira pergunta que vocÃª deve se fazer Ã© se vocÃª pode transformar aquela tabela (chata e feia) em algum tipo de grÃ¡fico. Lembrando, sempre pode incluir a tabela como anexo.

Aqui, vamos fazer um grÃ¡fico com os dados amostras_metrica, usando o pacote ggplot2 para verificar como a paisagem muda com a escala.

O ggplot2 faz parte do conjunto de pacotes tidyverse, e Ã© um pacote de visualizaÃ§Ã£o de dados. "gg" se refere a uma gramÃ¡tica de grÃ¡ficos. A ideia principal Ã© criar um grÃ¡fico como se fosse uma frase, onde cada elemento do grÃ¡fico seria uma palavra, organizados em uma sequencia logica para construir uma frase completo (grÃ¡fico final). VocÃª fornece os dados, informa ao ggplot2 como mapear variÃ¡veis para estÃ©tica, quais tipos/formatos grÃ¡ficas usar e ele cuida dos detalhes.

Isto nos permite construir grÃ¡ficos tÃ£o complexos quanto quisermos. Os grÃ¡ficos criados com ggplot2 sÃ£o, em geral, mais elegantes do que os grÃ¡ficos tradicionais do R. Para mais exemplos e tutoriais com mais detalhes veja os capitulos sobre ggplot2 nos livros:

- CiÃªncia de Dados com R
- AnÃ¡lises EcolÃ³gicas no R
- No livro em ingles R Graphics Cookbook .
- E sempre pode buscar exemplos no Google, por exemplo digitando: ggplot2 grÃ¡fico de barra no Google, tem mais de 50 mil resultados com paginas de imagens, codigo pronto e exemplos no YouTube.

O ggplot2 exige que os dados a serem plotados estejam em um "data frame" (quadro de dados). Ou seja, sempre teremos que transformar os dados para um data frame ou construir um data frame com os dados que possuÃmos. Data frame Ã© um formato comum e fÃ¡cil de trabalhar - parecido com uma planilha de dados com linhas e colunas. Todo que pode fazer com uma planilha de dados pode fazer com um data frame. Por exemplo, se vocÃª importar uma planilha de dados, o resultado seria como data frame (para mais detalhes veja Estrutura e manipulaÃ§Ã£o de objetos e lendo dados).

AlÃ©m disso, o mais importante Ã© que o resultado dos funÃ§Ãµes de landscapemetrics Ã© sempre um data frame, ou seja os resultados da funÃ§Ã£o `sample_lsm()` sÃ£o prontos para um grÃ¡fico elegante com ggplot2.

O principal funÃ§Ã£o a ser utilizado Ã© ggplot() . Para `ggplot()`, precisamos os dados (data frame), e depois cria o â mapeamentoâ das variÃ¡veis, normalmente usando aes() (de aesthetics). Ou seja, vocÃª especifica quais sÃ£o as variÃ¡veis dos eixos x e y dentro de aes() . AtravÃ©s dele vamos definir qual Ã© a variÃ¡vel preditora/explanatora/independente (eixo x) e qual Ã© a variÃ¡vel resposta/dependente (eixo y) em nosso conjunto de dados. Depois da funÃ§Ã£o ggplot() , na sequencia no codigo nÃ³s especificamos qual tipo de grÃ¡fico com um "geom". Por exemplo, geom_point() para plotar pontos, geom_boxplot() para um boxplot, etc. Para a lista completa de geoms e todas as outras opÃ§Ãµes do pacote, visite a pÃ¡gina do projeto ggplot2 https://ggplot2.tidyverse.org/index.html .

Aqui vamos fazer um grÃ¡fico com valores de extensÃ£o no eixo x (calculado a partir da coluna raio) e proporÃ§Ã£o da floresta central no eixo y (coluna value). Assim sendo, com o codigo a segir, vamos informar (i) os dados, selecionando classe de floresta atraves de um filtro com a funÃ§Ã£o `filter()` e acresentando com `mutate()` uma coluna nova ("ext_m") com a extensÃ£o em metros, (ii) as colunas para os eixos x e y, (iii) tipo de grÃ¡fico (grÃ¡fico de pontos - geom_point() e grÃ¡fico de linha - geom_line()), (iv) nomes para os eixos. No exemplo, usamos |>, que estabelece a ligaÃ§Ã£o entre os passos do processo, ligando os dados (amostras_metrica) e o grÃ¡fico ggplot . Note que no codigo a seguir, adicionamos um geom com um â +â . No ggplot2, nÃ³s criamos grÃ¡ficos em camadas, e adicionamos camada a camada com um â +â . Assim, Ã© posivel ajustar qualquer elemento do grÃ¡fico.

```
# arrumar os dados
amostras_metrica |>
  filter(class==1) |>
  mutate(ext_m = 2*raio) |>
# fazer o grÃ¡fico
  ggplot(aes(x = ext_m, y = value)) +
  geom_point() +
  geom_line() +
  labs(x = "ExtensÃ£o (metros)",
       y = "Ã rea central de floresta (porcentagem da paisagem)")
```

Depois de executar (â runâ) o cÃ³digo acima, vocÃª deverÃ¡ ver o grÃ¡fico a seguir.

2.4.3.2 Pergunta 3 **Em vez de extensÃ£o, vocÃª preciso incluir o tamanho (Ã¡rea do circulo) correspondente a cada raio. Incluir uma cÃ³pia do cÃ³digo ajustado para produzir uma figura com tamanho (Ã¡rea em quilÃ´metros quadrados) no eixo x.**

2.4.3.3 Pergunta 4 **Por que Ã© melhor usar mÃ©tricas como proporÃ§Ã£o da Ã¡rea central (proportion of core area) para comparaÃ§Ãµes multiescala na ecologia da paisagem? Inclua uma comparaÃ§Ã£o com outras mÃ©tricas, como Ã¡rea de mancha (patch area), para ilustrar por que as mÃ©tricas expressas como proporÃ§Ã£o sÃ£o mais apropriadas para comparaÃ§Ãµes multiescala. Justifique a sua resposta de forma clara e concisa (menos de 500 palavras). Apoie sua resposta com exemplos da literatura cientÃfica.**

2.4.3.4 FaÃ§a um grÃ¡fico elegante Podemos ajustar qualquer elemento do grÃ¡fico com ggplot2. Agora, vamos mudar as unidades de metros para quilometros, aumentar o tamanho dos pontos, incluir uma linha reta para ilustrar a tendÃªncia geral usando a funÃ§Ã£o `stat_smooth()`, colocar o titulo longo do eixo y em duas linhas, e aumentar o tamanho da fonte para o texto ficar mais claro. O objetivo principal de `stat_smooth()` Ã© ajudar o olho a ver padrÃµes nos dados plotados, ou seja, `stat_smooth()` sobrepÃµe uma curva ajustada aos dados.

Esta funÃ§Ã£o pode traÃ§ar curvas de uma variedade de modelos diferentes, incluindo regressÃ£o linear (`method = "lm"`).

```
# arrumar os dados
amostras_metrica |>
  filter(class==1) |>
  mutate(ext_m = 2*raio,
         ext_km = (2*raio)/1000) |>
# fazer o grÃ¡fico
  ggplot(aes(x = ext_km, y = value)) +
  geom_point(size = 4) +
  geom_line() +
  stat_smooth(method = "lm", se = FALSE, color = "green",
              linetype = "dashed") +
  labs(x = "ExtensÃ£o (quilÃ´metros)",
       y = "Ã rea central de floresta\n(porcentagem da paisagem)") +
  theme(text = element_text(size = 18))
```

```
## `geom_smooth()` using formula = 'y ~ x'
```

Figura 2.4: ComparaÃ§Ã£o da Ã¡rea central de floresta em diferentes extensÃµes.

2.4.3.5 Pergunta 5 **Em menos de 200 palavras apresente a sua interpretaÃ§Ã£o do grÃ¡fico em figura 2.4. EsclareÃ§a em sua resposta por que hÃ¡ apenas seis pontos mostrados quando sete buffers foram usados.**

2.4.3.6 Modelos linear e nÃ£o linear Um dos desafios mais frequentes Ã© como melhor representar os dados observados para gerar evidÃªncias cientÃficas robustas e informaÃ§Ãµes confiÃ¡veis. NÃ³s vimos que as mudancas na mÃ©trica porcentagem de Ã¡rea central de floresta nÃ£o segue uma linha reta em relaÃ§Ã£o de escala (extensÃ£o). Para ir alÃ©m de uma descriÃ§Ã£o simplista dos padrÃµes observados, na ecologia da paisagem uma variedade de modelos estatÃsticos sÃ£o usados.NÃ£o vamos rodar modelos (ainda), mas Ã© importante entender algumas das opÃ§Ãµes disponÃveis ao interpretar os grÃ¡ficos.

Por exemplo, modelos de regressÃ£o sÃ£o amplamente usados em diversas aplicaÃ§Ãµes para descrever a relaÃ§Ã£o entre uma variÃ¡vel resposta Y e uma variÃ¡vel explicativa x. Os modelos lineares sÃ£o uma generalizaÃ§Ã£o dos testes de hipÃ³tese clÃ¡ssicos mais simples (para mais detalhes veja Modelos linear e Modelos lineares). Uma regressÃ£o linear sÃ³ pode ser aplicada para dados em que tanto a variÃ¡vel preditora quanto a resposta sÃ£o contÃnuas, enquanto uma anÃ¡lise de variÃ¢ncia Ã© utilizada quando a variÃ¡vel preditora/explicativa Ã© categÃ³rica. Os modelos lineares generalizados nÃ£o tÃªm essa limitaÃ§Ã£o, podemos usar variÃ¡veis contÃnuas ou categÃ³ricas indistintamente (para mais detalhes e exemplos ecolÃ³gicos veja Modelos Lineares Generalizados).

Mas, no caso de padroes ecologicas, sera que um modelo linear Ã© o melhor modelo para representar a relaÃ§Ã£o que explica â yâ em funÃ§Ã£o de â xâ ? Um nÃºmero crescente de pesquisadores compartilham o sentimento de que as relaÃ§Ãµes entre variÃ¡veis biolÃ³gicas/ecologicas sÃ£o melhores descritas por funÃ§Ãµes nÃ£o lineares. Os sistemas ecolÃ³gicos sÃ£o inerentemente complexos, com interaÃ§Ãµes multifacetadas entre organismos e seu ambiente. Capturar essas complexidades muitas vezes requer ir alÃ©m da simplicidade das relaÃ§Ãµes lineares. Ã aÃ que os modelos nÃ£o lineares se tornam ferramentas valiosas na pesquisa ecolÃ³gica.

Ao contrÃ¡rio dos modelos lineares, que assumem uma relaÃ§Ã£o linear entre as variÃ¡veis, os modelos nÃ£o lineares abraÃ§am a possibilidade de curvas, limiares e outros padrÃµes intrincados. Eles alcanÃ§am isso por meio de funÃ§Ãµes matemÃ¡ticas que permitem diversas interaÃ§Ãµes entre variÃ¡veis. A principal vantagem dos modelos nÃ£o lineares sobre os lineares Ã© que eles sÃ£o baseados em conhecimento prÃ©vio sobre a relaÃ§Ã£o a ser modelada e geralmente apresentam interpretaÃ§Ã£o prÃ¡tica para os parÃ¢metros. Portanto, modelos nÃ£o lineares captura relaÃ§Ãµes complexas como:

- Efeitos de limiar:
  MudanÃ§as repentinas na resposta alÃ©m de um ponto crÃtico, como o colapso das populaÃ§Ãµes de plantas apÃ³s exceder uma pressÃ£o de pastejo especÃfica.
- Loops de feedback:
  InteraÃ§Ãµes positivas ou negativas entre variÃ¡veis, como ciclos predador-presa ou crescimento populacional exponencial.
- Respostas saturadas:
  DiminuiÃ§Ã£o ou nivelamento da resposta Ã medida que uma variÃ¡vel explicativa aumenta, como a absorÃ§Ã£o de nutrientes se aproximando de um limite mÃ¡ximo.

Em modelos nÃ£o lineares, os dados observados de uma variÃ¡vel resposta sÃ£o descritos por uma funÃ§Ã£o que Ã© nÃ£o linear em seus parÃ¢metros. Assim como nos modelos lineares, o objetivo Ã© identificar e estabelecer a relaÃ§Ã£o entre as variÃ¡veis explicativas e resposta. No entanto, enquanto os modelos lineares geralmente sÃ£o utilizados para descrever relaÃ§Ãµes empÃricas, os modelos nÃ£o lineares sÃ£o geralmente motivados pelo conhecimento do tipo de relaÃ§Ã£o entre as variÃ¡veis. Assim sendo, os modelos nÃ£o lineares podem se ajustar melhor e prever padrÃµes complexos nos dados ecolÃ³gicos, levando a previsÃµes mais precisas da dinÃ¢mica populacional, impactos ambientais e estratÃ©gias de manejo de recursos.

Como resultado, os modelos nÃ£o lineares sÃ£o amplamente utilizados em diversas Ã¡reas, como fÃsica, biologia, quÃmica, fisiologia, etc., onde as relaÃ§Ãµes entre as variÃ¡veis sÃ£o descritas por funÃ§Ãµes nÃ£o lineares.

Como as mudanÃ§as na estrutura da paisagem caracterizam-se por serem nÃ£o-lineares, para desenvolver anÃ¡lises estatÃsticas robustos pode (i) aplicar uma transformaÃ§Ã£o (por exemplo, “log”) ou (ii) adotar modelos nÃ£o-lineares.

Nesta anÃ¡lise, exploramos os dados `amostras_metrica` atravÃ©s das lentes de trÃªs modelos distintos:

1. Modelo Linear (Dados Originais):
   Este modelo de linha de base assume uma relaÃ§Ã£o linear entre as variÃ¡veis independentes e dependentes nos dados originais nÃ£o transformados. Podemos interpretar os coeficientes do modelo para compreender o impacto direto de cada preditor na variÃ¡vel resposta. No entanto, esta abordagem pode nÃ£o capturar padrÃµes nÃ£o lineares ou heterocedasticidade (variÃ¢ncia desigual) nos dados.

2. Modelo Linear (dados transformados em log):
   Reconhecendo a potencial nÃ£o linearidade, transformamos logaritmo da variÃ¡vel dependente. Isto pode estabilizar a variÃ¢ncia e potencialmente linearizar a relaÃ§Ã£o, melhorando o ajuste de um modelo linear. No entanto, a interpretaÃ§Ã£o dos coeficientes torna-se menos simples, exigindo transformaÃ§Ã£o reversa e exponenciaÃ§Ã£o para retornar Ã s unidades originais.

3. Modelo NÃ£o Linear (Dados Originais):
   Para contabilizar a potencial nÃ£o linearidade sem transformar os dados, empregamos um modelo nÃ£o linear, como uma regressÃ£o polinomial. Isto permite relaÃ§Ãµes mais flexÃveis entre as variÃ¡veis mas pode levar a coeficientes menos interpretÃ¡veis.

Visualizando o ajuste:

Para comparar a eficÃ¡cia destes modelos, empregamos ferramentas de diagnÃ³stico visual:

- Valores previstos versus pontos de dados:
  sobrepomos os valores previstos (linhas tracejadas) gerados por cada modelo nos pontos de dados originais (pontos pretos). Quanto mais prÃ³ximas as linhas previstas estiverem dos pontos de dados, melhor o modelo captura as tendÃªncias subjacentes.

- Intervalos de confianÃ§a:
  visualizamos os intervalos de confianÃ§a (sombreamento cinza) associados Ã s previsÃµes de cada modelo. Intervalos mais estreitos indicam maior precisÃ£o e menor incerteza nas estimativas. A comparaÃ§Ã£o das larguras dos intervalos entre os modelos pode revelar qual deles fornece previsÃµes mais confiÃ¡veis.

Podemos mostrar os intervalos de confianÃ§a no grÃ¡fico com a Ã¡rea central de floresta em diferentes extensÃµes com um ajuste simples no codigo. Os intervalos de confianÃ§a sÃ£o feitos com `stat_smooth()` pela opÃ§Ã£o `se = TRUE`, usando o seguinte cÃ³digo:

```
# arrumar os dados
amostras_metrica |>
  filter(class==1) |>
  mutate(ext_m = 2*raio,
         ext_km = (2*raio)/1000) |>
# fazer o grÃ¡fico
  ggplot(aes(x = ext_km, y = value)) +
  geom_point(size = 4) +
  geom_line() +
  stat_smooth(method = "lm",
              se = TRUE,
              color = "green",
              linetype = "dashed") +
  labs(x = "ExtensÃ£o (quilÃ´metros)",
       y = "Ã rea central de floresta\n(porcentagem da paisagem)") +
  theme_bw() +
  theme(text = element_text(size = 18))
```

```
## `geom_smooth()` using formula = 'y ~ x'
```

Figura 2.5: ComparaÃ§Ã£o da Ã¡rea central de floresta em diferentes extensÃµes. Mostrando os valores previstos do modelo linear (linha tracejada), pontos de dados originais (pontos pretos) e o intervalo de confianÃ§a do modelo (sombreamento cinza).

Para fornecer evidÃªncias cientÃficas robustas, precisarÃamos ir alÃ©m de uma avaliaÃ§Ã£o visual subjetiva e adicionar anÃ¡lises estatÃsticas para apoiar as nossas conclusÃµes. Ao analisar esses recursos visuais juntamente com mÃ©tricas estatÃsticas como R-quadrado (por enquanto nÃ£o precisamos saber os detalhes, apenas que existem tÃ©cnicas), podemos tirar conclusÃµes informadas sobre qual modelo melhor se ajusta aos dados amostras_metrica. Esta abordagem comparativa proporciona uma compreensÃ£o mais profunda das relaÃ§Ãµes dentro dos dados e permite-nos escolher o modelo mais apropriado para anÃ¡lise e previsÃ£o subsequentes.

Vamos comparar os trÃªs modelos modelo linear (dados Originais), modelo Linear (dados transformados em log) e modelo NÃ£o Linear (dados Originais).

Figura 2.6: ComparaÃ§Ã£o de padrÃµes lineares e nÃ£o-lineares. Comparamos visualmente o ajuste dos modelos comparando o quÃ£o prÃ³ximos os valores previstos (linhas tracejadas) estÃ£o dos pontos de dados originais (pontos pretos) e as diferenÃ§as nos intervalos de confianÃ§a do modelo (sombreamento cinza).

2.4.3.7 Pergunta 6 **Comparar os resultados apresentados nas figuras com modelos lineares e nÃ£o-lineares. Como podemos estabelecer qual seria o melhor modelo? Qual modelo seria mais adequado para identificar limiares no padrÃ£o de Ã¡rea central de floresta?**

2.4.3.8 Pergunta 7 **Aumentar o tamanho amostral na analise. Usando os exemplos de codigo anteriores, aumentar o numero de buffers entre 125 m e 16 km. i) Calcular a metrica cpland para 12 distÃ¢ncias de bufferes e montar dados novas (isso Ã© sete distÃ¢ncias originais e as cinco distÃ¢ncias novas) em um objeto com nome de "amostras_metrica_nova". ii) usando ggplot2 fazer um grÃ¡fico com os novos dados mostrando valores de extensÃ£o no eixo x e proporÃ§Ã£o da floresta central no eixo y. Ajustar o codigo para incluir intervalos de confianÃ§a junto com o modelo linear. Comparar os resultados apresentados nas figuras com modelos lineares sem transformaÃ§Ã£o. O que aconteceu com o ajuste do modelo e os intervalos de confianÃ§Ã£? Foi Ãºtil incluir mais distÃ¢ncias? Justifique as suas respostas de forma clara e concisa. Inclua cÃ³pias do seu cÃ³digo e grÃ¡ficos na sua resposta. VocÃª pode usar o printscreen para mostrar o RStudio com seu cÃ³digo e grÃ¡ficos.**

2.4.4 Ponto Ãºnico, distÃ¢ncias variados, mÃ©tricas variadas

No exemplo anterior comparamos uma mÃ©trica da paisagem em torno de um Ãºnico ponto de amostragem. Mas sabemos que uma combinaÃ§Ã£o de vÃ¡rias mÃ©tricas Ã© necessÃ¡ria para entender os padrÃµes na paisagem. Aqui mostraremos como incluir cÃ¡lculos de diferentes mÃ©tricas de paisagem ao mesmo tempo.

Figura 2.7: IlustraÃ§Ã£o da determinaÃ§Ã£o de mÃ©tricas da paisagem diferentes ao redor de um ponto. Exemplo com a estrutura da paisagem representado com trÃ©s caracteristicas (A) Ã rea central, (B) Borda e (C) Vizinho mais prÃ³ximo. O habitat de interesse (classe) Ã© isolado. Um buffer (linha tracejada) Ã© colocado ao redor de um ponto (amarela) e as mÃ©tricas calculadas. E em seguida o processo Ã© repetido em diferentes extensÃµes.

NÃ£o deve calcular todas as mÃ©tricas disponÃveis, mas sim, escolher aquelas que podem ser realmente adequadas para sua pergunta de pesquisa.

Calculando todas as mÃ©tricas se chama um "tiro no escuro", algo cujo resultado se desconhece ou Ã© imprevisÃvel. Isso nÃ£o Ã© recomendado. Para fazer uma escolha melhor (mais robusta), seguindo princÃpios bÃ¡sicos da ciÃªncia, precisamos ler os estudos anteriores (artigos) para obter as mÃ©tricas mais relevantes para nosso objetivo, pergunta e/ou a hipÃ³tese a ser testada.

A mÃ©trica mais simples e mais importante Ã© o tamanho da mancha, que representa um atributo fundamental do carÃ¡ter espacial de uma mancha. A maioria das mÃ©tricas de paisagem incorpora diretamente informaÃ§Ãµes sobre o tamanho da mancha ou Ã© afetada pelo tamanho da mancha.

Aqui, como exemplo illustrativa vamos calcular alguns das mÃ©tricas mais comums. Mas, isso nÃ£o representa necessariamente as mÃ©tricas mais adequados ou recommendadas. Lembre-se que algumas mÃ©tricas possuem uma combinaÃ§Ã£o de estatÃsticas (mÃ©dia, coeficiente de variaÃ§Ã£o e desvio padrÃ£o). Ã importante incluir medidas de variaÃ§Ã£o (coeficiente de variaÃ§Ã£o e desvio padrÃ£o), bem como a mÃ©dia, para que possamos fornecer evidÃªncias cientÃficas robustas. **Nunca apresente uma mÃ©dia sem uma medida da variaÃ§Ã£o**.

- MÃ©tricas de Ã¡rea e borda (area and edge metrics). Quantificam a composiÃ§Ã£o da paisagem:
  - `pland` = percentage of landscape. Percentagem da paisagem. Porcentagem de cobertura da classe na paisagem.
  - `ed` = edge density . Densidade de borda que Ã© igual Ã soma dos comprimentos (m) de todos os segmentos de borda que envolvem o fragmento, dividida pela Ã¡rea total da paisagem (mÂ²), sendo posteriormente convertido em hectares.
  - `cpland` = core area percentage of landscape. Percentual de Ã¡rea central ("core") na paisagem. Percentual de Ã¡reas centrais (excluÃdas as bordas de 30 m) em relaÃ§Ã£o Ã Ã¡rea total da paisagem. O termo "Core area" foi traduzido como Ã¡rea central ou Ã¡rea nÃºcleo. Aqui vamos adotar Ã¡rea central. Os Ãndices de Ã¡rea central integram em uma Ãºnica medida os efeitos da Ã¡rea do patch, formato do patch e distÃ¢ncia do efeito de borda. Todos os Ãndices de Ã¡rea central sÃ£o afetados pela interaÃ§Ã£o do tamanho do patch, formato do patch e profundidades de borda especificadas. Portanto, embora quantifiquem a composiÃ§Ã£o da paisagem, sÃ£o afetados pela configuraÃ§Ã£o da paisagem. Por exemplo, se a Ã¡rea central for pequena, indica que hÃ¡ muito pouca Ã¡rea central disponÃvel, mas nÃ£o discrimina entre uma Ã¡rea pequena (efeito de Ã¡rea) e uma Ã¡rea grande com formato complexo (efeito de configuraÃ§Ã£o).
- MÃ©tricas de agregaÃ§Ã£o. Quantificam a configuraÃ§Ã£o da paisagem:
  - `enn` = euclidian nearest neighbour distance. DistÃ¢ncia euclidiana do vizinho mais prÃ³ximo.
  - `enn_cv` = Coefficient of variation of euclidean nearest-neighbor distance. Coeficiente de variaÃ§Ã£o da distÃ¢ncia euclidiana do vizinho mais prÃ³ximo. A mÃ©trica resume cada classe como o Coeficiente de variaÃ§Ã£o das distÃ¢ncias euclidianas do vizinho mais prÃ³ximo entre as manchas pertencentes Ã classe. O valor de enn_cv = 0 se a distÃ¢ncia euclidiana do vizinho mais prÃ³ximo for idÃªntica para todas as manchas. Aumenta, sem limite, Ã medida que a variaÃ§Ã£o do ENN aumenta.
  - `enn_sd` = Standard deviation of euclidean nearest-neighbor distance. Desvio padrÃ£o da distÃ¢ncia euclidiana do vizinho mais prÃ³ximo.
  - `pd` = Patch density. Densidade das manchas.
  - `cohesion` = Cohesion index. Ã ndice de coesÃ£o das manchas.

Para incluir cÃ¡lculos de diferentes mÃ©tricas de paisagem ao mesmo tempo, precisamos acrescentar somente uma nova linha de codigo. Uma nova linha, que cria um objeto com os nomes das funÃ§Ãµes para as mÃ©tricas que queremos calcular….. TambÃ©m precisamos usar a opÃ§Ã£o "what" na funÃ§Ã£o para aceitar os nomes das funÃ§Ãµes .

```r
# Objeto com os nomes das funÃ§Ãµes para calcular as mÃ©tricas desejadas.
# 6 mÃ©tricas,
# um (enn) com 3 estatisticas (mn = media,
#                              sd = desvio padrÃ£o,
#                              cv = coeficiente de variaÃ§Ã£o)
minhas_metricas <- c("lsm_c_pland", "lsm_c_ed", "lsm_c_cpland",
                     "lsm_c_enn_mn", "lsm_c_enn_sd", "lsm_c_enn_cv",
                     "lsm_c_pd","lsm_c_cohesion")

# 6 MÃ©tricas calculadas para cada extensÃ£o
# raio 250 metros
sample_lsm(floresta_2020, y = rio_pontos_31976[1, ],
           size = 250, shape = "circle",
           what = minhas_metricas) |>
  mutate(raio = 250) -> metricas_amostra_250
# raio 500 metros
sample_lsm(floresta_2020, y = rio_pontos_31976[1, ],
           size = 500, shape = "circle",
           what = minhas_metricas)  |>
   mutate(raio = 500) -> metricas_amostra_500
# raio 1 km (1000 metros)
sample_lsm(floresta_2020, y = rio_pontos_31976[1, ],
           size = 1000, shape = "circle",
           what = minhas_metricas)  |>
   mutate(raio = 1000) -> metricas_amostra_1000
# raio 2 km (2000 metros)
sample_lsm(floresta_2020, y = rio_pontos_31976[1, ],
           size = 2000, shape = "circle",
           what = minhas_metricas) |>
  mutate(raio = 2000) -> metricas_amostra_2000
# raio 4 km (4000 metros)
sample_lsm(floresta_2020, y = rio_pontos_31976[1, ],
           size = 4000, shape = "circle",
           what = minhas_metricas)  |>
  mutate(raio = 4000)  -> metricas_amostra_4000
# raio 8 km (8000 metros)
sample_lsm(floresta_2020, y = rio_pontos_31976[1, ],
           size = 8000, shape = "circle",
           what = minhas_metricas)  |>
  mutate(raio = 8000) -> metricas_amostra_8000
# raio 16 km (16000 metros)
sample_lsm(floresta_2020, y = rio_pontos_31976[1, ],
           size = 16000, shape = "circle",
           what = minhas_metricas)  |>
  mutate(raio = 16000) -> metricas_amostra_16000
```

E agora, o cÃ³digo a seguir juntarÃ¡ os resultados das diferentes extensÃµes.

```r
bind_rows(metricas_amostra_250,
          metricas_amostra_500,
          metricas_amostra_1000,
          metricas_amostra_2000,
          metricas_amostra_4000,
          metricas_amostra_8000,
```

```
          metricas_amostra_16000) -> amostras_metricas
```

Depois que executar ("run"), podemos olhar os dados com o codigo a seguir.

```
amostras_metricas
```

Os dados deve ter os valores (coluna "value") das mÃ©tricas (coluna metric) de cada classe (coluna "class") para cada distÃ¢ncia (coluna "raio"):

layer	level	class	id	metric	value	plot_id	percentage_inside	raio
1	class	0	NA	cohesion	100.00	1	113.3	250
1	class	0	NA	cpland	80.32	1	113.3	250
1	class	0	NA	ed	0.00	1	113.3	250
1	class	NA	NA	enn_cv	NA	1	113.3	250
1	class	NA	NA	enn_mn	NA	1	113.3	250
1	class	NA	NA	enn_sd	NA	1	113.3	250
1	class	0	NA	pd	4.49	1	113.3	250
1	class	0	NA	pland	100.00	1	113.3	250
1	class	0	NA	cohesion	99.95	1	106.4	500
1	class	1	NA	cohesion	76.69	1	106.4	500

Agora, vamos fazer um grÃ¡fico com os dados amostras_metricas, usando o pacote ggplot2. Para ajudar na visualizaÃ§Ã£o incluÃmos quais mÃ©tricas sÃ£o para composiÃ§Ã£o e configuraÃ§Ã£o, e nomes que sÃ£o mais fÃ¡ceis de entender. A funÃ§Ã£o `mutate()` Ã© usado para incluir novas colunas.

```
metricas_composicao <- c("pland", "ed", "cpland")
# arrumar dados
amostras_metricas |>
  filter(class==1) |>
  mutate(ext_km = (2*raio)/1000,
         met_cat = if_else(metric %in% metricas_composicao,
                           "composiÃ§Ã£o", "configuraÃ§Ã£o")) |>
# fazer grÃ¡fico
  ggplot(aes(x=ext_km, y=value)) +
  geom_point() +
  geom_line() +
  facet_wrap(met_cat~metric, scales = "free_y") +
  labs(title = "ComparaÃ§Ã£o multiescala de vÃ¡rias mÃ©tricas",
    x = "extensÃ£o (quilÃ´metros)",
    y = "metric value")
```

2.4.4.1 Pergunta 8 **Com base nos resultados apresentados (figura e tabela) caracterisar as mudanÃ§Ã£s na paisagem em funÃ§Ã£o de extensÃµes diferentes. Olhando os grÃ¡ficos prever como seria o padrÃ£o para extensÃµes maiores (lembrando que valores sÃ£o doubrados - por exemplo raio de 250 metros gerar uma extensÃ£o de 500 metros). Seria relevante repetir incluindo calculos para extensÃµes maiores (por exemplo 64 km e 128 km)? Justifique sua caracterizaÃ§Ã£o e previsÃµes de forma clara e concisa, apoie sua escolha com exemplos da literatura cientÃfica.**

2.4.4.2 Pergunta 9 **Usando como base o conteudo das aulas, leitura disponivel no Google Classroom (Base teÃ³rica 4 Dados, mÃ©tricas, analises), e/ou exemplos apresentados aqui no tutorial, selecione pelo menos oito mÃ©tricas da paisagem de nÃvel classe para caracterizar a paisagem de estudo e objectivos da sua projeto em grupo. Justifique sua seleÃ§Ã£o de forma clara e concisa, apoie sua escolha com exemplos da literatura cientÃfica.**

Essa foi a Ãºltima pergunta.

SoluÃ§Ãµes dos exercÃcios: https://darrennorris.github.io/eprsol/solutions02_metricas.html

2.5 Links Ãºteis

Spatial Data Science with R: https://rspatial.org/

McGarigal K., Cushman SA., & Ene, E. 2023.
Spatial Pattern Analysis. Concepts and applications: https://www.fragstats.org/index.php/documentation

Wagner, H. 2022.
Landscape Genetic Data Analysis with R: https://bookdown.org/hhwagner1/LandGenCourse_book/ Helene Wagner and Max Hesselbarth. Spatial data landscape analysis: https://bookdown.org/hhwagner1/LandGenCourse_book/WE_2.html

Lovelace, R. 2023.
Geocomputation with R: https://geocompr.robinlovelace.net/index.html

3 MudanÃ§as na cobertura vegetal

3.1 ApresentaÃ§Ã£o

Este capÃtulo apresenta uma metodologia para calcular mudanÃ§as na cobertura da terra usando o software R.

A metodologia consiste em aplicar mÃ©todos de geoprocessamento para elaborar mapas de perda de formaÃ§Ãµes naturais, com base em mapas de cobertura da terra de uma regiÃ£o do AmapÃ¡, nos anos 1986 e 2021. A metodologia Ã© uma ferramenta importante para o monitoramento da cobertura da terra e para a avaliaÃ§Ã£o dos impactos das atividades humanas sobre o meio ambiente.

As principais etapas da metodologia sÃ£o as seguintes:

1. ObtenÃ§Ã£o dos dados: Os mapas de cobertura da terra de 1986 e 2021 sÃ£o obtidos do projeto MapBiomas.
2. Recorte dos dados: Os mapas sÃ£o recortados para a Ã¡rea de estudo no AmapÃ¡.
3. ClassificaÃ§Ã£o dos dados: As classes de cobertura da terra dos mapas sÃ£o reclassificadas para facilitar a anÃ¡lise.
4. CÃ¡lculo das mudanÃ§as: As mudanÃ§as na cobertura da terra sÃ£o calculadas usando mÃ©todos de geoprocessamento, como o cÃ¡lculo de Ã¡reas e o cÃ¡lculo de proporÃ§Ãµes.
5. ElaboraÃ§Ã£o dos mapas: Os mapas de perda de formaÃ§Ãµes naturais sÃ£o elaborados a partir dos resultados do cÃ¡lculo das mudanÃ§as.

A metodologia apresentada Ã© uma ferramenta Ãºtil para a anÃ¡lise de mudanÃ§as na cobertura da terra. Ela permite identificar as formaÃ§Ãµes naturais que foram perdidas ou que sofreram alteraÃ§Ãµes significativas ao longo do tempo. AlÃ©m das etapas apresentadas, a metodologia pode ser complementada com outros procedimentos, como:

- AnÃ¡lise da qualidade dos dados:
  Ã importante avaliar a qualidade dos dados de cobertura da terra antes de realizar o cÃ¡lculo das mudanÃ§as.
- ValidaÃ§Ã£o dos resultados:
  Os resultados do cÃ¡lculo das mudanÃ§as podem ser validados comparando-os com dados de campo ou com informaÃ§Ãµes de outras fontes.

3.2 Ã rea de estudo

A paisagem de estudo Ã© a Ã¡rea de um raio de 32 km ao redor do reservatÃ³rio da Usina HidrelÃ©trica Ferreira Gomes. A UHE Ferreira Gomes Ã© uma usina hidrelÃ©trica localizada no AmapÃ¡, com capacidade instalada de 252 MW. Coordenadas geogrÃ¡ficas (longitude / latitude): 51Âº11â 41,071ÂºW / 00Âº51â 20,126ÂºN (https://ferreiragomesenergia.com.br/usina/). A usina foi inaugurada em etapas, com a primeira turbina entrando em operaÃ§Ã£o comercial em novembro de 2014 e a terceira turbina em maio de 2015.

Link para serie temporal (Landsat timelapse): https://earthengine.google.com/timelapse#v=0.89999,-51.42342,8.79,latLng&t=3.53&ps=50&bt=19840101&et=20221231

Link para mudanÃ§as na superfice de agua: https://global-surface-water.appspot.com/map#v=0.85826,-51.1794,10,latLng&t=0.00&l=wc

3.3 Pacotes e Fontes de dados

3.3.1 Pacotes

Carregar pacotes (que deve esta instalado antes):

```r
library(tidyverse)
library(sf)
library(terra)
library(mapview)
library(tmap)
library(eprdados)
library(landscapemetrics)
library(tidyterra)
library(patchwork)
```

Caso os pacotes nÃ£o tenham sido instalados, o R vai avisar atraves um mensagem tipo: Error in library(nomepacote) there is no package called nomepacote. Neste caso, para instalÃ¡-los consulte o CapÃtulo 6 ou os capitulos aqui Capitulo 4 instalaÃ§Ã£o de pacotes e aqui Capitulo 4 pacotes . Prestando atenÃ§Ã£o nas requisitos diferenciados para instalar um pacote como `eprdados` que vem do GitHub.

3.3.2 Fontes de dados

As etapas da metodologia requer um conjunto de dados espaciais incluindo camadas de vetor e raster.

3.3.2.1 Arquivos de vetor Os dados necessÃ¡rios estÃ£o disponÃveis no pacote `eprdados`. As fontes originais dos dados vetoriais foram:

- Reservatorios
  AgÃªncia Nacional de Ã guas e Saneamento BÃ¡sico (ANA) Massas d'Ã gua: https://metadados.snirh.gov.br/files/7d054e5a-8cc9-403c-9f1a-085fd933610c/geoft_bho_massa_dagua_v2019.zip
- Barragens
  AgÃªncia Nacional de Energia ElÃ©trica (ANEEL SIGEL) Aproveitamento HidrelÃ©tricos - AHE e reservatorios (bacia 1): https://sigel.aneel.gov.br/portal/home/index.html

Agora carregar as camadas de pontos (barragem), linhas (rios) e poligonos (reservatorio e Ã¡rea de estudo).

```r
FG_ponto <- st_read(system.file("vector/UHE_Ferreira_Gomes.gpkg",
                                package="eprdados"),
                    layer ="FG_ponto")
FG_araguari <- st_read(system.file("vector/UHE_Ferreira_Gomes.gpkg",
                                package="eprdados"),
                    layer ="FG_curso_araguari")
FG_rios <- st_read(system.file("vector/UHE_Ferreira_Gomes.gpkg",
                                package="eprdados"),
                    layer ="FG_curso_agua")
FG_reservatorio <- st_read(system.file("vector/UHE_Ferreira_Gomes.gpkg",
                                       package="eprdados"),
                           layer ="FG_reservatorio")
FG_extensao <- st_read(system.file("vector/UHE_Ferreira_Gomes.gpkg",
                                       package="eprdados"),
                           layer ="fg_extensao")
# Outros barragens no Rio Araguari
res_cn <- st_read(system.file("vector/barragens.gpkg",
                              package="eprdados"),
                  layer ="CN_reservatorio")
res_cc <- st_read(system.file("vector/barragens.gpkg",
                              package="eprdados"),
                  layer ="CC_reservatorio")
# trÃªs juntos
res_ara <- st_read(system.file("vector/barragens.gpkg",
```

```
                                package="eprdados"),
                    layer ="AP_reservatorios") |>
  filter(nmoriginal != "UHE Santo AntÃ´nio do Jari")
```

3.3.2.2 Arquivos de raster

Mapbiomas ColeÃ§Ã£o 8. - Cobertura da terra
A ColeÃ§Ã£o 8 do MapBiomas de mapas anuais de cobertura e uso da terra do Brasil. Dados anuais para o perÃodo de 1985 a 2022: https://brasil.mapbiomas.org/

Os dados do mapbiomas de 1985 a 2022 foram baixados e recortados para a Ã¡rea de estudo no AmapÃ¡. O arquivo "utm_cover_UHEFerreiraGomes_85a22.tif" tem uma classificaÃ§Ã£o de terra dentro de um raio de 32 km ao redor do reservatorio. Dentro do arquivo tem 38 camadas, um para cada ano entre 1985 e 2022.

Baixar arquivo com os dados (formato ".tif"), link: https://github.com/darrennorris/epr/blob/main/data/raster/mapbiomas/utm_cover_UHEFerreiraGomes_85a22.tif . Lembrando-se de salvar o arquivo ("utm_cover_UHEFerreiraGomes_85a22.tif") em um local conhecido no seu computador.

Para carregar o arquivo trabalhamos com o pacote terra. O pacote tem varios funÃ§oes para a anÃ¡lise e modelagem de dados geogrÃ¡ficos. NÃ³s podemos ler os dados de cobertura da terra no arquivo ".tif" com a funÃ§Ã£o `rast()`. IncluÃmos outra funÃ§Ã£o `file.choose()` que abrirÃ¡ uma nova janela (explorador de Windows). Assim, vocÃª encontre o local onde o arquivo baixado foi salvo e selecione o arquivo "utm_cover_UHEFerreiraGomes_85a22.tif".

```
#
r85a22 <- rast(file.choose())
```

Verifique algumas camadas usando um mapa:

```
# Selecione um ano e agregue para acelerar o processamento.
r22 <- subset(r85a22, 38)
r22_modal <- aggregate(r22, fact=10, fun="modal")

# Mapa
mapview(FG_reservatorio) +
  mapview(FG_ponto, col.regions = "yellow") +
  mapview(as.factor(r22_modal))
```

3.4 MudanÃ§as na paisagem: indo alÃ©m de desmatamento

Precisamos entender o uso da terra na paisagem. Para isso vamos olhar as mudancas durante 36 anos, comparando a percentual das diferentes classes em 1986 com 2021. A metodologia possui as seguintes etapas:

1. Suavizar a variaÃ§Ã£o nas estimativas de dados anuais:
   Essa abordagem visa reduzir flutuaÃ§Ãµes temporais abruptas (por exemplo erros na classificaÃ§Ã£o) e realÃ§ar padrÃµes de mudanÃ§a de cobertura terrestre, proporcionando uma visÃ£o mais estÃ¡vel e representativa ao longo do tempo.

2. AnÃ¡lise exploratÃ³ria:
   As mudanÃ§as na cobertura da terra sÃ£o calculadas usando mÃ©tricas da paisagem como Ã¡reas e proporÃ§Ãµes de classes diferentes.

3. ClassificaÃ§Ã£o dos dados:
   Com base nas anÃ¡lises exploratÃ³rias, as classes de cobertura da terra sÃ£o reclassificadas para facilitar a anÃ¡lise.

4. CÃ¡lculo das mudanÃ§as:
   As mudanÃ§as na cobertura da terra sÃ£o calculadas usando mÃ©todos de geoprocessamento para obter Ã¡reas e taxas de mudanÃ§a nas classes diferentes.

5. ElaboraÃ§Ã£o dos mapas:
   Os mapas de perda de formaÃ§Ãµes naturais sÃ£o elaborados a partir dos resultados do cÃ¡lculo das mudanÃ§as.

3.4.1 DefinaÃ§Ã£o

A perda florestal e o desmatamento sÃ£o dois conceitos relacionados, mas que possuem significados diferentes. A perda florestal Ã© um conceito mais amplo que o desmatamento, e inclui todos os fatores que podem levar Ã reduÃ§Ã£o da cobertura florestal. O desmatamento Ã© um "tipo" especÃfico de perda florestal que ocorre quando a cobertura florestal Ã© removida para uso em outros fins por seres humanos, como agricultura, pecuÃ¡ria, mineraÃ§Ã£o ou urbanizaÃ§Ã£o. Ã importante lembrar que tanto a perda florestal, quanto o desmatamento nÃ£o inclui alteraÃ§Ãµes em habitats nÃ£o florestais como formaÃ§oes campestres ou campos alagados.

3.4.2 Suavizar a variaÃ§Ã£o nas estimativas anuais

Aqui usamos a modal de trÃªs anos para suavizar a variaÃ§Ã£o nas estimativas de dados anuais da classificaÃ§Ã£o de cobertura terrestre do MapBiomas. Para os trÃªs anos usamos:

- i) o ano atual,
- ii) o ano anterior e
- iii) o ano seguinte.

Ou seja, o primerio ano seria 1986, com cada cÃ©lula o valor modal dos anos 1985, 1986 e 1987. Embora isso reduza o nÃºmero de anos de 38 para 36, ainda temos 7 anos apÃ³s a barragem entrou em operaÃ§Ã£o (ano da primeira turbina foi 2014). A utilizaÃ§Ã£o da modal de trÃªs anos para suavizar a variaÃ§Ã£o nas estimativas anuais oferece vÃ¡rios benefÃcios para anÃ¡lise e interpretaÃ§Ã£o:

- ReduÃ§Ã£o do ruÃdo e das flutuaÃ§Ãµes:
  As estimativas anuais de sensoriamento remoto podem ser suscetÃveis a ruÃdo e flutuaÃ§Ãµes de curto prazo devido a diversos fatores, como:
  - CondiÃ§Ãµes climÃ¡ticas:
    Cobertura de nuvens, precipitaÃ§Ã£o e variaÃ§Ãµes sazonais podem afetar a qualidade e a interpretaÃ§Ã£o das imagens de satÃ©lite.
  - VariaÃ§Ãµes do sensor:
    Diferentes satÃ©lites e sensores possuem suas prÃ³prias caracterÃsticas de calibraÃ§Ã£o e sensibilidade, o que leva a discrepÃ¢ncias sutis entre os anos.
  - Erros de classificaÃ§Ã£o:
    Erros menores nos algoritmos de classificaÃ§Ã£o automatizados podem levar a flutuaÃ§Ãµes em categorias especÃficas de cobertura da terra.

A utilizaÃ§Ã£o da modal de trÃªs anos atua como um filtro simples, mas eficaz, atenuando o impacto dessas flutuaÃ§Ãµes e fornecendo uma estimativa mais robusta de mudanÃ§a na cobertura da terra. Isso resulta em:

- Melhora na detecÃ§Ã£o de tendÃªncias:
  Ao suavizar o ruÃdo, torna-se mais fÃ¡cil identificar tendÃªncias e padrÃµes verdadeiros na mudanÃ§a da cobertura da terra ao longo do tempo. Isso Ã© crucial para entender as dinÃ¢micas ecolÃ³gicas de longo prazo e o impacto de atividades humanas, como desmatamento ou projetos de infraestrutura.
- Aumento da precisÃ£o:
  Embora nÃ£o esteja provado estatisticamente que seja definitivamente mais preciso, o suavizaÃ§Ã£o pode reduzir o potencial de outliers e flutuaÃ§Ãµes aleatÃ³rias distorcerem a imagem geral da distribuiÃ§Ã£o da cobertura da terra.

Passos para obter o modal de trÃªs anos para suavizar a variaÃ§Ã£o nas estimativas: selecionar os trÃªs anos com `subset()` , aggregar pixels para reduzir tempo de processamento `aggregate()` e calcular o valor modal de cada pixel para os trÃªs anos `modal()`.

```r
# 85 a 87
r86 <- subset(r85a22, c(2, 1, 3))
r86_agg10 <- aggregate(r86, fact=10, fun="modal")
r86_agg10_mod <- modal(r86_agg10, ties = "first")
names(r86_agg10_mod) <- "modal_85a87"
# 2020 a 2022
r21 <- subset(r85a22, c(37, 38, 36))
r21_agg10 <- aggregate(r21, fact=10, fun="modal")
r21_agg10_mod <- modal(r21_agg10, ties = "first")
names(r21_agg10_mod) <- "modal_20a22"

r86a21 <- c(r86_agg10_mod, r21_agg10_mod)
```

FaÃ§a um mapa.

```r
classes_fg86a21 <- c(3, 4, 6, 9, 11, 12, 15, 24, 30, 33, 39, 41)
mapbiomas_fg <- mapbiomas_legend |>
  filter(Class_ID %in% classes_fg86a21) |>
  arrange(Class_ID)
mybreaks <- as.character(c(3, 4, 6, 9, 11, 12, 15, 24, 30, 33, 39, 41))
# PLot
fig_mapa <- ggplot() +
  geom_spatraster(data = as.factor(r86a21)) +
  scale_fill_manual("Classe", breaks= mybreaks,
                     labels= mapbiomas_fg$Descricao,
                     values = mapbiomas_fg$Color) +
    scale_x_continuous(breaks =
                      c(-51.5, -51.3, -51.1, -50.9)) +
  facet_wrap(~lyr, ncol = 2) +
       theme(legend.position = "top")

png("figures/fig_fg_mapa.png", width=10, height=5,
    units="in", res = 600)
fig_mapa
invisible(dev.off())
knitr::include_graphics("figures/fig_fg_mapa.png")
```

Podemos ver nos mapas que as maiores mudanÃ§as ocorreram ao sul do rio Araguari e envolveram plantaÃ§Ãµes florestais (silvicultura).

3.4.3 AnÃ¡lise exploratÃ³ria

Para isso vamos comparar a Ã¡rea total e percentual das diferentes classes em 1986 com 2021.

```
res_m2 <- xres(r86a21) * yres(r86a21)
pixels_total <- ncell(r86a21)
area_m2_agg10 <- res_m2 * pixels_total
area_ha_agg10 <- (res_m2 * pixels_total) / 10000
area_km2_agg10 <- (res_m2 * pixels_total) / 1000000

valores <- values(r86a21)
classe_frequencia_85a87 <- table(valores[ ,'modal_85a87'])
df_classe_frequencia_85a87 <- data.frame(classe_frequencia_85a87)
df_classe_frequencia_85a87 <- df_classe_frequencia_85a87 |>
  mutate(ano = 1986,
         Classe = as.numeric(as.character(Var1)),
         Classe_area_m2 = Freq * res_m2) |>
  mutate(Classe_prop = Classe_area_m2 / area_m2_agg10) |>
  left_join(mapbiomas_legend,
            by = c("Classe" = "Class_ID"))
```

Repetir o mesmo processo com valores de 2021.

```
classe_frequencia_20a22 <- table(valores[ ,'modal_20a22'])
df_classe_frequencia_20a22 <- data.frame(classe_frequencia_20a22)
df_classe_frequencia_20a22 <- df_classe_frequencia_20a22 |>
  mutate(ano = 2021,
         Classe = as.numeric(as.character(Var1)),
         Classe_area_m2 = Freq * res_m2) |>
  mutate(Classe_prop = Classe_area_m2 / area_m2_agg10) |>
  left_join(mapbiomas_legend,
            by = c("Classe" = "Class_ID"))
```

Combinar os dados de ambos os anos em um Ãºnico dataframe.

```
df_classe_frequencia_anos <- bind_rows(df_classe_frequencia_85a87,
                                       df_classe_frequencia_20a22)
```

Podemos visualizar os resultados em um grafico:

```
fig_anos_simples <- ggplot(df_classe_frequencia_anos,
       aes(x = fct_reorder(Descricao, Classe_prop, mean),
           y = Classe_prop)) +
  geom_col(aes(fill=factor(ano)), position = "dodge") +
  coord_flip()  +
  scale_fill_discrete("Ano") +
  labs(x = "",
       y = "cobertura (proporÃ§Ã£o da paisagem)") +
  theme(legend.position = c(0.85, 0.2))

png("figures/fig_anos_simples.png", width=5, height=3,
    units="in", res = 600)
fig_anos_simples
invisible(dev.off())
knitr::include_graphics("figures/fig_anos_simples.png")
```

A paisagem tem uma Ã¡rea total de cerca 544401 hectares ou 5444 quilÃ´metros quadrados. Assim sendo, 1% da paisagem de estudo Ã© 5444 hectares ou 54 quilÃ´metros quadrados.

E agora o mesmo processo com o pacote landscapemetrics:

```
mymetrics <- c("lsm_c_pland", "lsm_c_ca")
lmet <- landscapemetrics::calculate_lsm(r86a21,
                                        what = mymetrics) |>
  mutate(ano = case_when(layer == 1 ~ "1986",
                         layer == 2 ~ "2021")) |>
  left_join(mapbiomas_legend, by = c("class" = "Class_ID"))
```

Usando as mÃ©tricas calculados podemos comparar as mudanÃ§as na paisagem entre 1986 e 2021.

Podemos apresentar os mapas juntamente com grÃ¡ficos resumindo as mÃ©tricas da paisagem em uma figura composta usando o pacote patchwork. Uma figura composta permite-nos ver a distribuiÃ§Ã£o espacial das mudanÃ§as (mapas) juntamente com os valores associados (grÃ¡ficos). Uma Ãºnica figura composta pode transmitir as mesmas informaÃ§Ãµes que vÃ¡rias figuras individuais, economizando espaÃ§o e reduzindo a confusÃ£o visual. Ao combinar dados relacionados em uma Ãºnica figura, os leitores podem facilmente comparar e contrastar tendÃªncias e relaÃ§Ãµes entre diferentes variÃ¡veis. Por exemplo, as figuras compostas permitem uma entendimento mais completa e unificada com seus dados. Em vez de pular entre figuras separadas, os leitores podem compreender a narrativa geral de suas descobertas em uma Ãºnica imagem. Isso pode melhorar o fluxo e a coerÃªncia do seu texto, tornando-o mais envolvente e impactante. Isto pode ser especialmente importante em artigos com espaÃ§o limitado para figuras ou conjuntos de dados complexos.

Atualmente (2021) a paisagem de estudo Ã© composta principalmente por classes florestais (48.9 %) e campestres (30.3 %). Ao longo de trÃªs dÃ©cadas, a criaÃ§Ã£o de plantaÃ§Ãµes florestais gerou as maiores mudanÃ§as, principalmente na perda das Ã¡reas campestres. Ao longo deste perÃodo, as plantaÃ§Ãµes florestais aumentaram de 0.02 para 13.5 por cento da paisagem, enquanto as campestres foram reduzidas em 12.3 por cento da paisagem (de 42.6 em 1986 para 30.3 em 2021), ou seja uma perda de 67014.2 hectares.

3.4.4 ReclassificaÃ§Ã£o

Identificaremos Ã¡reas onde as formaÃ§Ãµes naturais foram convertidas para uso humano da terra. Como a anÃ¡lise exploratÃ³ria mostrou que em 1986 a paisagem era composta principalmente por dois classes de cobertura, analisaremos separadamente a floresta e as outras formaÃ§Ãµes naturais.

3.4.4.1 Dados de reservatorios. Primeiramente atualizamos a classificaÃ§Ã£o de MapBiomas com uma nova valor de "35" para representar os reservatorios na Ã¡rea de estudo.

Atualizar 1986 com o reservatorio de Coracy Nunes (operacional desde 1975):

```
rnew <- r86_agg10_mod
v_cn <- vect(res_cn |> mutate(reservoir = 100))
z <- rasterize(v_cn, rnew, "reservoir")
z[is.na(z)] <- 0
xrc <- r86_agg10_mod + z

rcxrc <- classify(xrc, cbind(100, 170, 35),
           include.lowest=TRUE, right=TRUE)
```

Atualizar 2021 com os trÃªs reservatorios:

```
rnew <- r21_agg10_mod
v_ara <- vect(res_ara |> mutate(reservoir = 100))
z <- rasterize(v_ara, rnew, "reservoir")
z[is.na(z)] <- 0
xrc <- r21_agg10_mod + z

rcxrc21 <- classify(xrc, cbind(100, 170, 35),
```

Figura 3.1: MudanÃ§as na paisagem de 1986 atÃ© 2021. ComparaÃ§Ã£o de cobertura da terra de 1986 a 2021 no entorno da barragem Ferreira Gomes. (A) Mapeamento da classificaÃ§Ã£o de cobertura da terra do MapBiomas ColeÃ§Ã£o 8 usada para quantificar as diferenÃ§as na cobertura da terra entre 1986 e 2021. (B) ComparaÃ§Ã£o da percentagem das diferentes classes de cobertura da terra na paisagem. (C) DiferenÃ§as na cobertura de cada classe entre 1986 e 2021.

```
           include.lowest=TRUE, right=TRUE)
rcxrc86a21 <- c(rcxrc, rcxrc21)
```

Update legend.

```
mapbiomas_legend_new <- mapbiomas_legend |>
                          add_row(Class_ID = 35, Level = 2,
                                  Description = "Reservoir",
                                  Descricao = "Reservatorio",
                                  Color = "#00FFFF",
                                  Descricao_novo = NA)
```

Agora vamos reclassificar para trÃªs categorias: florestal, outras formaÃ§Ãµes e nÃ£o natural. Em seguida, combinamos os mapas de 1986 e 2021 para gerar um mapa das Ã¡reas de mudanÃ§a durante este perÃodo.

```
human_class <- c(14, 15, 18, 19, 39, 20, 40, 62, 41,
                 36, 46, 47, 35, 48, 9, 21, 22, 24, 25, 30, 31, 35)
forest_class <- c(3, 4, 5, 6)
other_natural <- c(11, 12, 32, 29, 50, 13)
water <- 33
old_class <- c(human_class, forest_class,
               other_natural, water)
new_class <- c(rep(0, length(human_class)),
               rep(1, length(forest_class)),
               rep(2, length(other_natural)),
               rep(3, length(water)))
reclass_m <- as.matrix(data.frame(old = old_class,
                                  new_value = new_class))

reclass_m
```

```
##       old new_value
##  [1,]  14         0
##  [2,]  15         0
##  [3,]  18         0
##  [4,]  19         0
##  [5,]  39         0
##  [6,]  20         0
##  [7,]  40         0
##  [8,]  62         0
##  [9,]  41         0
## [10,]  36         0
## [11,]  46         0
## [12,]  47         0
## [13,]  35         0
## [14,]  48         0
## [15,]   9         0
## [16,]  21         0
## [17,]  22         0
## [18,]  24         0
## [19,]  25         0
## [20,]  30         0
## [21,]  31         0
## [22,]  35         0
## [23,]   3         1
```

```
## [24,]   4         1
## [25,]   5         1
## [26,]   6         1
## [27,]  11         2
## [28,]  12         2
## [29,]  32         2
## [30,]  29         2
## [31,]  50         2
## [32,]  13         2
## [33,]  33         3
```

Agora reclassificar usando a funÃ§Ã£o `classify()`.

```
r86a21res <- classify(rcxrc86a21, reclass_m)
f86 <- classify(rcxrc, reclass_m)
f21 <- classify(rcxrc21, reclass_m)
```

3.4.5 MudanÃ§as

3.4.5.1 DistribuiÃ§Ã£o espacial de mudanÃ§as Agora, vamos combinar os mapas reclassificadas de 1986 e 2021 para gerar um mapa das Ã¡reas onde a vegetaÃ§Ã£o natural foi perdida/convertida para uso humano durante este perÃodo. Aqui usamos uma operaÃ§Ã£o lÃ³gica de Ã¡lgebra para identificar pixels com perda: onde as Ã¡reas com vegetaÃ§Ã£o natural em 1986 (codificadas 1 ou 2 no mapa) ficaram sem cobertura natural em 2021 (codificadas 0 no mapa). Para isso precisamos escrever uma funÃ§Ã£o, incluindo a funÃ§Ã£o `ifelse()`, que contÃ©m uma condiÃ§Ã£o e os valores de saÃda no caso da condiÃ§Ã£o se cumplir ou Ã±ao, no caso de floresta: `ifelse(x == 1 & y == 0, 1, 0)`, permite obter um mapa binÃ¡rio das Ã¡reas desmatadas (valor de 1) entre 1986 e 2021. Para executar isso para objetos raster, usamos a funÃ§Ã£o `lapp()`, que aplica uma funÃ§Ã£o a raster, usando as camadas como argumentos.

```
# 1) FunÃ§Ã£o para duas camadas,
# A primeira camada Ã© x a segunda camada Ã© y.
fv1 <- function(x,y) {ifelse(x == 1 & y == 0, 1, 0)}
fv2 <- function(x,y) {ifelse(x == 2 & y == 0, 1, 0)}

# 2) testar a funÃ§Ã£o
dados <- list(c(1:4, NA, NA), c(0, 0, 1, NA, 4, NA))
do.call(fv1, dados)
```

```
## [1]  1  0  0  0  0 NA
```

```
do.call(fv2, dados)
```

```
## [1]  0  1  0  0  0 NA
```

```
# 3) Rodar a funÃ§Ã£o usando objeto que tem as 2 camadas
# 3.1) Desmatamento.
def <- lapp(r86a21res, fun =  fv1)
# 3.2) Perda de outra vegetaÃ§Ã£o natural.
gloss <- lapp(r86a21res, fun =  fv2)

# 4) Junte-se aos dois novos arquivos
cover_loss <- c(def, gloss)
names(cover_loss) <- c("desmatamento",
                       "outras formaÃ§Ãµes naturais")
```

3.4.5.2 Taxa de mudanÃ§as Taxa de desmatamento (Ã¡rea desmatada por km^2 por ano). Devido ao fato da resoluÃ§Ã£o espacial do mapa binÃ¡rio de desmatamento ser de 298.8 metros, as cÃ©lulas

correspondem a 8.9 ha. Assim, Ã© necessÃ¡rio multiplicar a soma das cÃ©lulas por 8.9, para obtenÃ§Ã£o de um mapa indicativo da quantidade de hectares desmatados por km^2. E depois dividir por 36 para obter a taxa anual.

Para fazer isso fazemos uma grade poligonal com pixels de 1 km. Em seguida, resumimos a perda de vegetaÃ§Ã£o em cada uma das â zonasâ da grade de 1 km.

```
# 1)
area_m2 <- xres(r86a21res) * yres(r86a21res)
area_ha <- area_m2 / 10000

# 2) PolÃgono - grade de 1 km
v <- st_as_sf(as.polygons(f21))
zones_1km <- st_make_grid(v, cellsize = 1000) |>
  st_sf()
zones_1km <- zones_1km |>
  mutate(aid = 1:nrow(zones_1km))

# 3) soma de mudanÃ§as hectares por km2
# Calcular estatÃsticas zonais, ou seja, resumir os valores da raster
# para cada "zona" definida por um polÃgono.
# 3.1) desmatamento
def_sum_r <- (zonal(def, vect(zones_1km),
                "sum", na.rm=TRUE,
                as.raster = TRUE) * area_ha) / 36
# 3.2) outras formaÃ§oes naturais
gloss_sum_r <- (zonal(gloss, vect(zones_1km),
                "sum", na.rm=TRUE,
                as.raster = TRUE) * area_ha) / 36

# 4) Junte-se aos dois novos arquivos
cover_loss_rate <- c(def_sum_r, gloss_sum_r)
names(cover_loss_rate) <- c("desmatamento",
                       "outras formaÃ§Ãµes naturais")
```

3.4.6 Mapeamento de mudanÃ§as

Por fim, apresente os resultados em um mapa.

```
# Plot with facets
# Deforestation
fig_perda <- ggplot() +
  geom_spatraster(data = as.factor(cover_loss)) +
    geom_sf(data = FG_araguari, color = "blue",
          linewidth = 1) +
  geom_sf(data = FG_ponto, color = "#00FFFF", fill = NA,
          size = 2) +
  scale_x_continuous(breaks =
                       c(-51.5, -51.3, -51.1, -50.9)) +
  scale_fill_manual(values = c("grey90","orange2"),
                    labels = c("nÃ£o", "sim")) +
  facet_wrap(~lyr, ncol = 2) +
  labs(title = "(A)",
       fill = "Perda") +
  theme(plot.title.position = "plot",
```

```
        legend.position="bottom")

# rate of natural cover loss
fig_perda_taxa <- ggplot() +
  geom_spatraster(data = cover_loss_rate) +
  facet_wrap(~lyr, ncol = 2) +
    geom_sf(data = FG_araguari, color = "blue",
          linewidth = 1) +
  geom_sf(data = FG_ponto, color = "#00FFFF", fill = NA,
          size = 2) +
  scale_fill_gradientn(colours = terrain.colors(5),
                       trans = "reverse",
                       na.value = "white") +
  scale_x_continuous(breaks =
                       c(-51.5, -51.3, -51.1, -50.9)) +
  labs(title = "(B) ",
       fill = "Perda\n(ha por km2)") +
  theme(plot.title.position = "plot",
        legend.position="bottom")

fig_taxa <- fig_perda / fig_perda_taxa
png("figures/fig_taxa.png", width=5, height=8,
    units="in", res = 600)
fig_taxa
invisible(dev.off())
knitr::include_graphics("figures/fig_taxa.png")
```

3.4.7 ConclusÃ£o

Se tivÃ©ssemos simplesmente nos concentrado no desmatamento, terÃamos ignorado muitas das mudanÃ§as na paisagem ao longo de trÃªs dÃ©cadas. Na escala da paisagem a maior parte das mudanÃ§as ocorreu ao sul (perda de vegetaÃ§Ã£o natural) e leste/amontante (desmatamento) da UHE Ferreira Gomes. Olhando para estes resultados preliminares, fica claro que serÃ¡ necessÃ¡ria uma anÃ¡lise mais refinada para quantificar os impactos a jusante da UHE Ferreira Gomes na cobertura vegetal nativa. Isto envolve o estabelecimento de escalas espaciais e temporais mais detalhadas e a inclusÃ£o de tÃ©cnicas que nos permitem levar em consideraÃ§Ã£o a natureza linear e gradientes longitudinais associados com os rios.

- Natureza linear e gradientes longitudinais:
  Ao contrÃ¡rio das florestas, os rios sÃ£o sistemas lineares com fortes gradientes longitudinais (por exemplo, mudanÃ§as na velocidade do fluxo, concentraÃ§Ãµes de nutrientes). A escolha da escala deve levar em conta esses gradientes para evitar interpretaÃ§Ãµes errÃ´neas. ExtensÃµes menores podem perder impactos importantes a jusante, enquanto extensÃµes maiores podem mascarar variaÃ§Ãµes localizadas.

- Escala e conectividade:
  A conectividade fluvial para organismos aquÃ¡ticos como peixes Ã© altamente dependente da escala. A anÃ¡lise das barreiras Ã circulaÃ§Ã£o pode exigir a concentraÃ§Ã£o nos obstÃ¡culos locais (escala precisa), enquanto a avaliaÃ§Ã£o da dinÃ¢mica populacional pode exigir a consideraÃ§Ã£o dos alcances a montante e a jusante (escala ampla).

- Escala e qualidade da Ã¡gua:
  Os parÃmetros de qualidade da Ã¡gua, como temperatura e oxigÃªnio dissolvido, podem variar dramaticamente ao longo dos rios. A escolha da escala apropriada para amostragem e anÃ¡lise garante uma representaÃ§Ã£o precisa desses padrÃµes dinÃmicos.

Figura 3.2: DistribuiÃ§Ã£o e taxa de mudanÃ§a na cobertura da terra. ComparaÃ§Ã£o da mudanÃ§a de cobertura da terra de 1986 a 2021 no entorno da barragem Ferreira Gomes (ponto azul claro) ao longo do rio Araguari (linha azul) AmapÃ¡, Brasil. ClassificaÃ§Ã£o de cobertura da terra do MapBiomas ColeÃ§Ã£o 8 usada para identificar (A) a distribuiÃ§Ã£o e (B) a taxa de mudanÃ§a de cobertura da terra entre 1986 e 2021. MudanÃ§a identificada como a transiÃ§Ã£o das classes de cobertura natural para antrÃ³picas.

Mas lembre-se que estas anÃ¡lises baseados nos dados obtidos de sensorimento remoto nÃ£o consideram impactos como alteraÃ§Ãµes no vazÃ£o do rio e interacÃ§Ãµes sinÃ©rgicas que podem ir alÃ©m dos dados disponÃveis.

3.5 Metodologia Espacialmente ExplÃcita para Quantificar os Impactos A Jusante da HidrelÃ©trica Ferreira Gomes na Cobertura Vegetal Nativa

Considerando as padrÃµes gerais identificadas, para compreender o impacto da barragem hidroelÃ©ctrica de Ferreira Gomes, precisamos olhar para escalas espaciais mais detalhadas. Adotaremos uma anÃ¡lise utilizando autocorrelaÃ§Ã£o espacial para examinar os impactos da barragem nos padrÃµes de mudanÃ§a da paisagem. Esta metodologia envolve a extraÃ§Ã£o de informaÃ§Ãµes de cobertura da terra ao longo dos anos, o cÃ¡lculo de gradientes e, em seguida, a realizaÃ§Ã£o de uma anÃ¡lise de autocorrelaÃ§Ã£o espacial usando semivariogramas. Os mapas de superfÃcie anisotrÃ³pica resultantes fornecem informaÃ§Ãµes sobre os padrÃµes espaciais de mudanÃ§as na cobertura vegetal nativa a jusante da barragem hidroelÃ©ctrica de Ferreira Gomes.

Esta metodologia descreve uma abordagem espacialmente explÃcita para analisar os impactos a jusante da hidrelÃ©trica Ferreira Gomes na cobertura vegetal nativa usando mapas anuais de cobertura da terra com resoluÃ§Ã£o de 30 metros do MapBiomas.

Recomenda-se que vocÃª leia os capÃtulos anteriores antes de iniciar este. Para ajudar a acompanhar e entender os exemplos no capÃtulo, vocÃª deve ler os seguintes artigos:

- ErÅ s, T., Lowe, W.H. (2019).
  The Landscape Ecology of Rivers: from Patch-Based to Spatial Network Analyses. Current Landscape Ecology Reports 4, 103â “112: https://doi.org/10.1007/s40823-019-00044-6
- Bishopâ Taylor, R., Tulbure, M. G., & Broich, M. (2017)
  Surfaceâ water dynamics and land use influence landscape connectivity across a major dryland region. Ecological Applications, 27(4), 1124-1137: https://doi.org/10.1002/eap.1507
- Bishop-Taylor, R., Tulbure, M. G., & Broich, M. (2018).
  Evaluating static and dynamic landscape connectivity modelling using a 25-year remote sensing time series. Landscape Ecology, 33, 625-640: https://doi.org/10.1007/s10980-018-0624-1
- Huang, J., & Gergel, S. E. (2023).
  Effectiveness of landscape indicators for explaining the variability of benthic macro-invertebrates in urban streams. Journal of Environmental Management, 342, 118130: https://doi.org/10.1016/j.jenvman.2023.118130

CÃ³pias dos artigos estÃ£o no Google Classroom e tambÃ©m podem ser baixadas aqui: https://drive.google.com/drive/folders/1oMQKn3Y1Wy3-p5L9sndBjdzCJ3adpKUw?usp=sharing

3.5.1 Dados:

- Mapas anuais de cobertura da terra do MapBiomas (resoluÃ§Ã£o de 30 metros):
  Esses mapas fornecem informaÃ§Ãµes espaciais explÃcitas sobre as mudanÃ§as na cobertura da terra ao longo do tempo.
- Mapas anuais de cobertura de agua da Global surface water:
  https://global-surface-water.appspot.com/ . Baixar os dados diretamente no R atraves o pacote ecochange. Artigo: ecochange: An R-package to derive ecosystem change indicators from freely available earth observation products :https://doi.org/10.1111/2041-210X.13986
- Dados operacionais da hidrelÃ©trica Ferreira Gomes:
  Datas de construÃ§Ã£o, enchimento do reservatÃ³rio e quaisquer mudanÃ§as significativas no fluxo de Ã¡gua ou transporte de sedimentos. Inclusive:

  - Ã rea Diretamente Afetada (ADA): Ã¡rea na qual o empreendimento serÃ¡ instalado. No caso de hidroelÃ©tricas a ADA inclui o reservatÃ³rio, a barragem, o vertedouro e a Ã rea de PreservaÃ§Ã£o Permanente (APPs) que envolve o reservatÃ³rio.
  - Ã rea de InfluÃªncia Direta (AID): engloba a Ã rea Diretamente Afetada e Ã© o local em que os impactos gerados afetam diretamente o meio ambiente e a sociedade. Tipicamente compreende alÃ©m da Ã¡rea do reservatÃ³rio, uma faixa marginal que variou de 1 Km a 7 Km [@carvalho2018criterios].

- Imagens de alta resoluÃ§Ã£o (opcional):
  Imagens de satÃ©lite com resoluÃ§Ã£o mais fina (por exemplo, Sentinel-2) podem ser usadas para validar as classificaÃ§Ãµes e analisar Ã¡reas especÃficas de interesse - dados anuais desde 2016 versÃ£o beta aqui: https://brasil.mapbiomas.org/mapbiomas-cobertura-10m/.

- DegradaÃ§Ã£o (opcional):
  Camadas de desmatamento e vegetaÃ§Ã£o secundÃ¢ria: https://brasil.mapbiomas.org/colecoes-mapbiomas/

3.5.2 Metodologia:

1. Definir a Ã¡rea de estudo:

Delinear a Ã¡rea a jusante potencialmente afetada pela barragem, considerando fatores como:

- Fluxo do rio e morfologia da bacia.
- MudanÃ§as esperadas nos nÃveis de Ã¡gua e transporte de sedimentos.
- DistribuiÃ§Ã£o da cobertura vegetal existente.

2. PrÃ©-processamento dos dados do MapBiomas:

Baixe os mapas anuais de cobertura da terra do MapBiomas para os anos antes, durante e apÃ³s a construÃ§Ã£o da barragem (idealmente, vÃ¡rios anos antes e depois). Aplique filtros espaciais para suavizar erros de classificaÃ§Ã£o e melhorar a qualidade dos dados.

3. AnÃ¡lise de gradientes:

- Calcule as mÃ©tricas de paisagem relevantes.
- Calcule as mudanÃ§as. Isso depende do objetivo do estudo, mas pode incluir:
  - Porcentagem de perda de cobertura vegetal nativa em cada ano em relaÃ§Ã£o ao ano base (prÃ©-barragem).
  - Use mÃ©tricas como matrizes de transiÃ§Ã£o de cobertura para analisar tipos especÃficos de perda de vegetaÃ§Ã£o (por exemplo, floresta para pastagem).
- Visualize a distribuiÃ§Ã£o espacial das mudanÃ§as na cobertura vegetal e sua relaÃ§Ã£o com a distÃ¢ncia da barragem.

4. AnÃ¡lise estatÃstica:

Realize testes estatÃsticos para avaliar a significÃ¢ncia das mudanÃ§as observadas na cobertura vegetal em diferentes classes de distÃ¢ncia e ao longo do tempo. Isso depende do objetivo do estudo, mas pode incluir:

- Use semivariogramas para criar mapas de superfÃcie anisotrÃ³pica.
  - Os semivariogramas revelam a autocorrelaÃ§Ã£o espacial nos gradientes de mudanÃ§a. A escolha de um modelo adequado proporciona insights sobre a estrutura e alcance da dependÃªncia espacial.
  - Krigagem para Mapas de SuperfÃcie AnisotrÃ³pica: A krigagem Ã© utilizada para interpolar os gradientes de mudanÃ§a, criando um mapa de superfÃcie anisotrÃ³pica. Esse mapa visualiza como as mudanÃ§as se propagam espacialmente, identificando Ã¡reas com impactos significativos.

- Empregue modelos de regressÃ£o para identificar possÃveis relaÃ§Ãµes entre a distÃ¢ncia da barragem (superfÃcie anisotrÃ³pica), e fatores especÃficos de impacto (por exemplo, inundaÃ§Ã£o) e a magnitude da perda de cobertura vegetal.

6. ValidaÃ§Ã£o e refinamento:

Se disponÃvel, use imagens de alta resoluÃ§Ã£o para validar as classificaÃ§Ãµes do MapBiomas, particularmente em Ã¡reas de mudanÃ§as significativas. Considere incorporar fontes de dados adicionais, como levantamentos de campo ou conhecimento da comunidade local, para refinar ainda mais a anÃ¡lise e a interpretaÃ§Ã£o dos resultados.

7. Resultados e implicaÃ§Ãµes:

Desenvolva mapas e figuras que representem os padrÃµes espaciais e temporais das mudanÃ§as na cobertura vegetal a jusante da barragem. Quantifique o impacto geral da barragem na perda de cobertura vegetal nativa e identifique as Ã¡reas mais afetadas. Analise as possÃveis consequÃªncias ecolÃ³gicas da perda de cobertura vegetal, como declÃnio da biodiversidade, erosÃ£o do solo e alteraÃ§Ã£o dos serviÃ§os ecossistÃªmicos. Use as descobertas para informar as prÃ¡ticas de gestÃ£o ambiental e estratÃ©gias de mitigaÃ§Ã£o para reduzir os impactos negativos da barragem na cobertura vegetal nativa.

ConsideraÃ§Ãµes adicionais:

Considere fatores de confusÃ£o que possam influenciar as mudanÃ§as na cobertura vegetal, como os drivers de desmatamento nÃ£o relacionados Ã barragem (por exemplo, agricultura, extraÃ§Ã£o de madeira, silvicultura). Considere incorporar variÃ¡veis de paisagem, como topografia e tipos de solo, para melhorar a anÃ¡lise dos gradientes de impacto. Esta metodologia pode ser adaptada para estudar os impactos a jusante de outros tipos de projetos de infraestrutura em diferentes ecossistemas.

3.6 Links uteis

Mas, J. F., Horta, M. B., de Vasconcelos, R. N., & Cambui, E. C. B. AnÃ¡lise espacial com R. 2019. UEFS Editora. https://www.academia.edu/download/59341189/LivroR_completo20190521-63855-yz3c44.pdf

4 Respostas multiescala de espÃ©cies

4.1 ApresentaÃ§Ã£o

Este capitulo pretende ilustrar a abordagem geral de identificaÃ§Ã£o de escalas de efeitos nas relaÃ§Ãµes ambientais usando R. Usaremos conteudo baseado no CapÃtulo 2 do livro Spatial Ecology and Conservation Modeling [@fletcher2018] . Aqui interpretamos a escala em que as espÃ©cies podem responder ao habitat, como a cobertura florestal. Para fazer isso, quantificamos a quantidade de floresta que ocorre a vÃ¡rias distÃ¢ncias dos pontos de amostragem e, em seguida, determinamos a escala em que a cobertura florestal melhor prevÃª a ocorrÃªncia de espÃ©cies.

Durante o capitulo vocÃª aprenderÃ¡:

1. Sobre conceitos de escalas caracterÃsticas e escalas de efeito,

2. Quantificar e avaliar as correlaÃ§Ãµes entre diferentes escalas,

3. Determinar a escala em que a floresta melhor prevÃª a ocorrÃªncia de espÃ©cies,

4. Compreender a relevÃ¢ncia dos conceitos para a interpretaÃ§Ã£o e aplicaÃ§Ã£o de resultados dos estudos de ecologia da paisagem.

Recomenda-se que vocÃª leia os capÃtulos anteriores antes de iniciar este. Para ajudar a acompanhar e entender os exemplos no capÃtulo, vocÃª deve ler os seguintes artigos:

- da Silva, C. R., et. al. (2013).
  Mammals of AmapÃ¡ State, Eastern Brazilian Amazonia: a revised taxonomic list with comments on species distributions. Mammalia, 77(4), 409-424. https://doi.org/10.1515/mammalia-2012-0121

- Ferreira, A. S., Peres, C. A., Dodonov, P., & Cassano, C. R. (2020).
  Multi-scale mammal responses to agroforestry landscapes in the Brazilian Atlantic Forest: the conservation value of forest and traditional shade plantations. Agroforestry Systems, 94, 2331-2341. https://doi.org/10.1007/s10457-020-00553-y

- Lyra-Jorge, M. C., Ribeiro, M. C., Ciocheti, G., Tambosi, L. R., & Pivello, V. R. (2010).
  Influence of multi-scale landscape structure on the occurrence of carnivorous mammals in a human-modified savanna, Brazil. European Journal of Wildlife Research, 56, 359-368. https://doi.org/10.1007/s10344-009-0324-x

- Michalski, F., Boulhosa, R. L. P., Nascimento, Y. N. D., & Norris, D. (2020).
  Rural wage-earnersâ attitudes towards diverse wildlife groups differ between tropical ecoregions: implications for forest and savanna conservation in the Brazilian Amazon. Tropical Conservation Science, 13, https://doi.org/10.1177/1940082920971747

- Michalski, F., & Peres, C. A. (2007).
  Disturbanceâ mediated mammal persistence and abundanceâ area relationships in Amazonian forest fragments. Conservation Biology, 21(6), 1626-1640. https://doi.org/10.1111/j.1523-1739.2007.00797.x

- Michalski, F., & Peres, C. A. (2005).
  Anthropogenic determinants of primate and carnivore local extinctions in a fragmented forest landscape of southern Amazonia. Biological Conservation, 124(3), 383-396. https://doi.org/10.1016/j.biocon.2005.01.045

- Regolin, et. al. (2017).
  Forest cover influences occurrence of mammalian carnivores within Brazilian Atlantic Forest. Journal of Mammalogy, 98(6), 1721-1731. https://doi.org/10.1093/jmammal/gyx103

CÃ³pias dos artigos estÃ£o no Google Classroom e tambÃ©m podem ser baixadas aqui: https://drive.google.com/drive/folders/13Bv_k4-fZiCNo-RlDfKDOT3qW-gJdJ6M?usp=sharing

Para quem estÃ¡ comeÃ§ando com o R, este capÃtulo pode ser desafiador. Ele requer um entendimento de conceitos como modelagem, estruturas de dados e iteraÃ§Ã£o. Se vocÃª nÃ£o tiver esse conhecimento, nÃ£o se preocupe em entender completamente o cÃ³digo de R. Em vez disso, concentre-se nos resultados apresentados, que mostram como as escalas caracterÃsticas e as escalas de efeito podem ser aplicadas na Ecologia da Paisagem.

4.2 Escala de efeito - definiÃ§Ã£o

Quantificando a escla de efeito Ã© um problema cada vez mais considerado na ecologia aplicada [@jackson2015ecologists; @dattilo2023spatial; @mcgarigal2016multi]. A escala de efeito refere-se Ã extensÃ£o espacial em que um elemento ou processo da paisagem influencia mais fortemente uma resposta ecolÃ³gica [@fletcher2018]. Em termos mais simples, Ã© o â escala idealâ , onde a relaÃ§Ã£o entre as caracterÃsticas da paisagem e os padrÃµes ecolÃ³gicos Ã© mais significativa. Este conceito Ã© crucial para compreender como a heterogeneidade da paisagem molda os processos ecolÃ³gicos e informa as estratÃ©gias de conservaÃ§Ã£o.

4.2.1 DefiniÃ§Ãµes

VÃ¡rios ecologistas definiram a escala do efeito:

- @MartinFahrig2012: "A extensÃ£o espacial, ou gama de extensÃµes, dentro da qual uma variÃ¡vel da paisagem tem seu(s) efeito(s) mais forte(s) na resposta de uma determinada espÃ©cie."

- Wu & Li (2006): "As escalas espaciais, temporais e hierÃ¡rquicas relevantes para processos e fenÃ´menos ecolÃ³gicos." No mesmo trabalho @WU2006 identificou duas conceitos de escala distintas, mas inter-relacionadas: escalas caracterÃsticas e escalas de efeito.

1. escalas caracterÃsticas
   O conceito de escala caracterÃstica implica que muitos, se nÃ£o a maioria, dos fenÃ´menos naturais tÃªm suas prÃ³prias escalas distintas (ou faixas de escalas) que caracterizam seu comportamento (por exemplo, extensÃ£o espacial tÃpica ou frequÃªncia do evento). As escalas caracterÃsticas sÃ£o intrÃnsecas aos fenÃ´menos de interesse, mas as escalas caracterÃsticas detectadas com o envolvimento do observador podem ser matizadas com subjetividade. Os padrÃµes e processos ecolÃ³gicos tÃªm escalas caracterÃsticas distintas, nas quais sua dinÃ¢mica pode ser estudada de forma mais eficaz. Assim sendo, identificando escalas caracterÃsticas fornece uma chave para uma compreensÃ£o profunda na Ecologia da Paisagem [@wu2004effects].

2. escalas de efeito
   As escala de efeito geralmente se referem Ã s mudanÃ§as no resultado de um estudo devido a uma mudanÃ§a na escala em que o estudo Ã© conduzido. Os efeitos da mudanÃ§a de escala no desenho experimental, na amostragem, nas anÃ¡lises estatÃsticas e na modelagem foram bem documentados em ecologia. As escalas de efeito podem ser explicados em termos de multiplicidade de escala, escalas caracterÃsticas e hierarquia. Mas escalas de efeito tambÃ©m podem ser artefatos devido a erros na amostragem e mediÃ§Ãµes, distorÃ§Ãµes na reamostragem de dados e falhas na anÃ¡lise estatÃstica e modelagem [@wu2004effects]. As escalas caracterÃsticas e as escalas de efeito estÃ£o inerentemente relacionados Ã questÃ£o da escala. Embora as escalas caracterÃsticas forneÃ§am uma base conceitual e diretrizes prÃ¡ticas para o dimensionamento, as descriÃ§Ãµes quantitativas das escalas de efeito podem levar diretamente a relaÃ§Ãµes de escala [@wu2004effects].

Estas definiÃ§Ãµes destacam a natureza dinÃ¢mica e dependente do contexto da escala de efeito. Ou seja - o que Ã© mais importante para uma espÃ©cie ou processo pode ser irrelevante numa escala diferente ou para outro organismo.

4.2.2 AplicaÃ§Ã£o

Existem mutias exemplos de uso onde a escala de efeito foi analisado como:

- OcorrÃªncia de espÃ©cies:
  Estudos que investigam a relaÃ§Ã£o entre a cobertura florestal e a diversidade de aves frequentemente encontram diferentes escalas de efeito para diferentes espÃ©cies [@miguet2016determines]. Por exemplo, alguns especialistas florestais podem apresentar a resposta mais forte num raio de 1 km de manchas florestais, enquanto os generalistas podem ser influenciados pela cobertura florestal em escalas maiores (por exemplo, 5 km).

- DispersÃ£o das sementes:
  A distÃ¢ncia percorrida pelas sementes antes da germinaÃ§Ã£o pode variar dependendo do agente dispersor, variÃ¡veis de resposta e preditor de paisagens [@san2019scale]. Por exemplo, as sementes dispersas pelo vento podem ter efeitos em escalas maiores, enquanto as sementes dispersas pelas formigas podem ser limitadas a paisagens menores.

- DinÃ¢mica metapopulacional:
  A persistÃªncia de populaÃ§Ãµes pequenas e fragmentadas pode depender da conectividade de manchas de habitat adequadas dentro de uma determinada escala de paisagem [@frey2012rel; @foster2016planning; @saade2023landscape]. Portanto, compreender a escala do efeito da dispersÃ£o e movimento entre manchas Ã© crucial para a concepÃ§Ã£o de estratÃ©gias de conservaÃ§Ã£o eficazes.

4.2.3 Como identificar a escala de efeito?

Consideramos como podemos relacionar diferenÃ§as de escala (extensÃ£o) com a ocorrÃªncia de espÃ©cies para ajudar a identificar a escala caracterÃstica (ou escala de efeito) da cobertura florestal na ocorrÃªncia de espÃ©cies. Existem vÃ¡rias maneiras de quantificar a escala de efeito. Uma abordagem Ã© usar zonas tampÃ£o (daqui adiante "buffers") de diferentes tamanhos para medir a paisagem circundante aos locais de amostragem. Essa abordagem foi popularizada por @pearson1993spatial e Ã© amplamente utilizada por outros pesquisadores [@jackson2015ecologists; @GalanAcedo2018; @Moll2020].

Um mÃ©todo alternativo emprega uma funÃ§Ã£o de suavizaÃ§Ã£o ponderada pela distÃ¢ncia - Kernels espaciais. Os kernels podem ser usados para ponderar os dados com base na distÃ¢ncia do local de amostreagem. Isso ajuda a capturar melhor os efeitos da vizinhanÃ§a, ponderando mais os locais prÃ³ximos do que os distantes. Chandler e Hepinstall-Cymerman [-@chandler2016estimating] mostraram como os kernels espaciais podem ser usados para selecionar a escala de efeito sem recorrer a binning a priori (como por exemplo, os buffers no capÃtulo "MÃ©tricas").

Neste capitulo, primeiramente ilustramos o uso de buffers de diferentes distancias e depois ilustramos o uso de kernels espaciais.

4.3 Pacotes e dados

4.3.1 Pacotes

Carregar pacotes (que deve esta instalado antes):

```
library(mapview)
library(eprdados)
library(tidyverse)
library(tidymodels)
library(GGally)
```

Caso os pacotes nÃ£o tenham sido instalados, o R vai avisar atraves um mensagem tipo: Error in library(nomepacote) there is no package called nomepacote. Neste caso, para instalÃ¡-los consulte o CapÃtulo 6 ou os capitulos aqui Capitulo 4 instalaÃ§Ã£o de pacotes e aqui Capitulo 4 pacotes . Prestando atenÃ§Ã£o nas diferenÃ§as para instalar um pacote como `eprdados` que vem do GitHub.

4.3.2 Dados necessarios

Aqui usaremos os conjuntos de dados entrevistas e carnivoros. Os dados estÃ£o no pacote eprdados. Carregar os dados com o codigo seguinte:

```
entrevistas <- eprdados::entrevistas
carnivoros <- eprdados::carnivoros
```

Uma vez importados os dados para o R, geralmente antes de iniciarmos qualquer manipulaÃ§Ã£o, visualizaÃ§Ã£o ou anÃ¡lise de dados, fazemos a conferÃªncia desses dados. Para isso, podemos utilizar as funÃ§Ãµes listadas na Tabela "FunÃ§Ãµes para verificaÃ§Ã£o e resumo de dados multidimensionais." no CapÃtulo 7.

Exemplos de alguns funÃ§Ãµes e os resultados sÃ£o no codigo seguinte:

```
## tipo de objeto
class(carnivoros)

## Primeiras linhas
head(carnivoros)

## NÃºmero de linhas e colunas
nrow(carnivoros)
#> [1] 2544
ncol(carnivoros)
#> [1] 13
dim(carnivoros)
#> [1] 2544   13

## Estrutura dos dados
str(carnivoros)

## Ajudar para dados que vem de uma pacote
?eprdados::carnivoros
```

4.3.2.1 Pergunta 1 **Verificar os dados entrevistas e carnivoros, usando funÃ§Ãµes "class()"; "str()" e ajudar "?". Descreva o conteÃºdo e compare as diferenÃ§as nesses dois conjuntos de dados.**

Os dados entrevistas tem a localizaÃ§Ã£o de 106 propriedades rurais no AmapÃ¡ [@Michalski2020int].

Os dados carnivoros consistem em:

- AbundÃ¢ncias e ausÃªncia-presenÃ§a de trÃªs espÃ©cies de carnÃvoros simuladas no entorno de 106 propriedades rurais no AmapÃ¡, Brasil [@Michalski2020int]. A distribuiÃ§Ã£o das espÃ©cies foi simulada usando o pacote virtualspecies.

- Uma mÃ©trica de paisagem cpland foi calculada no entorno das propriedades seguindo a metodologia no capÃtulo 2 usando o pacote landscapemetrics.

As trÃªs espÃ©cies tÃªm respostas contrastantes Ã cobertura florestal. Os padrÃµes de resposta seguem aqueles conhecidos para espÃ©cies de carnÃvoros [@MICHALSKI2005383; @MICHALSKI2007conbio; @regolin2017forest].

Os dados incluem valores de uma mÃ©trica de paisagem (cpland) calculada para uma Ãºnica classe de cobertura do solo - cobertura florestal (MapBiomas Collection 8). A cobertura florestal foi reclassificada das classes originais, e inclui formaÃ§Ãµes florestais e savÃ¢nicas. A mÃ©trica cpland Ã© o percentual da Ã¡rea central da classe em relaÃ§Ã£o Ã Ã¡rea total da paisagem. A mÃ©trica cpland foi calculada em torno de cada propriedade rural em buffers com raios de 0,125, 0,25, 0,5, 1, 2, 4, 8 e 16 km. Para reduzir a influÃªncia da qualidade da imagem de satÃ©lite, os valores foram calculados como medianas de trÃªs anos: o ano de estudo, o ano anterior e o ano posterior.

4.3.3 O paisagem do estudo

Para compreender o contexto da paisagem de estudo, examinaremos primeiro a Ã¡rea de estudo e a localizaÃ§Ã£o das propriedades rurais. Para visualizar as mudanÃ§as na area de estudo (onde foi feito as entrevistas) nos ultimos decadas podemos olhar com Google Earth Timelapse (com coordenados graus decimais 0.50979, -52.12023): https://earthengine.google.com/timelapse#v=0.50979,-52.12023,7.512,latLng&t=0.49&ps=50&bt=19840101&et=20221231 .

Alem disso, no R podemos olhar os dados atraves de uma mapa interativa com o pacote mapview : https://r-spatial.github.io/mapview/ . mapview serve para visualizaÃ§Ã£o exploratÃ³ria de dados espacias â gerando mapas Ãºteis com muito pouco cÃ³digo. Uma captura de tela do mapa padrÃ£o Ã© mostrada abaixo, incluindo uma tabela pop-up que aparece se vocÃª clicar em um feiÃ§Ã£o (ponto, linha etc) e o texto que aparece se vocÃª passar o mouse sobre uma camada.

Figura 4.1: mapview: Componentes e funÃ§Ãµes disponÃveis no mapa interativo padrÃ£o.

A maneira mais fÃ¡cil de criar um mapa no mapview Ã© usar a funÃ§Ã£o `mapview()` e passar um conjunto de dados espacias para ele. Podemos visualizar os locais das entrevistas com o seguinte cÃ³digo, que tem a mesma estrutura de sempre "pacote::funÃ§Ã£o(dados)":

```
mapview::mapview(entrevistas)
```

Com o codigo `mapview::mapview(entrevistas)` vocÃª deverÃ¡ ver um mapa semelhante ao mapa da prÃ³xima figura. Com este mapa interativo vocÃª pode aproximar diferentes pontos e alterar o mapa base para outros, incluindo OpenStreetMap (com cidades, ruas etca) e ESRI WorldView (com imagem de satÃ©lite). Essas funcionalidades permitem ver padrÃµes e heterogeneidade na cobertura da paisagem em diferentes escalas.

4.3.3.1 Pergunta 2 **Na aula "OrganizaÃ§Ã£o de paisagens" vimos que as paisagens contemporÃ¢neas sÃ£o uma resultado de i) variabilidade em condiÃ§Ãµes abiÃ³ticos; ii) interaÃ§Ãµes biÃ³ticas; iii) distÃºrbios naturais e sucessÃ£o; e iv) padrÃµes antrÃ³picos no uso da terra. Usando Google Earth Timelapse e a forma interativa da mapview, verficar a paisagem acerca das propriadades. Com base na sua interpretaÃ§Ã£o das mapas, as aulas, e literatura citado atÃ© agora (como Michalski et. al. 2020, Michalski & Peres 2005, Michalski & Peres 2007), apresentar trÃ©s fatores que poderia explicar a distribuiÃ§Ã£o espacial de especies de mamiferos na paisagem. Para cada fator inclui uma hipotese valida (e nÃ£o trivial) junto com uma previsÃ£o que pode se testado com um estudo de Ecologia da Paisagem.**

4.4 Buffers

Agora investigamos a escala em que as espÃ©cies podem responder ao habitat, como a cobertura florestal. Para fazer isso, quantificamos a quantidade de floresta que ocorre em vÃ¡rias distÃ¢ncias e, em seguida, determinamos a escala em que a cobertura florestal melhor explica a ocorrÃªncia das espÃ©cies.

4.4.1 Floresta

Primeiramente, olhamos a proporÃ§Ã£o de cobertura florestal em diferentes escalas (distÃ¢ncias de buffer) para todos os pontos (locais de 106 entrevistas). Esses dados ja foram calculados seguindo os passos no capitulo "MÃ©tricas". Para nos auxilia Ã ter uma visÃ£o geral do conjunto de dados e de suas interrelaÃ§Ãµes podemos apresentar como um grÃ¡fico pareado. Esse grÃ¡fico tambÃ©m Ã© chamado de "pairs plot" ou "correlograma". A funÃ§Ã£o `ggpairs()` do pacote GGally permite criar mÃºltiplos grÃ¡ficos pareados comparando as variÃ¡veis contÃnuas no seu conjunto de dados. Ou seja, com `ggpairs()` podemos comparar cada distÃ¢ncia do buffer com todas as outras distÃ¢ncias do buffer.

Existem trÃªs resultados diferentes apresentados pela funÃ§Ã£o:

1. GrÃ¡ficos de dispersÃ£o de cada par de variÃ¡veis (lado esquerdo abaixo da diagonal).
2. GrÃ¡ficos de distribuiÃ§Ã£o de densidade de cada variÃ¡vel individualmente (no diagonal).
3. Os valores de correlaÃ§Ã£o entre os pares analisados (lado direito acima da diagonal).

Para mais detalhes sobre a funÃ§Ã£o `ggpairs()` veja CapÃtulo **??**.

Ao fazÃª-lo, descobrimos que a percentagem de cobertura florestal em diferentes escalas tende a ser altamente correlacionada. As correlaÃ§Ãµes mais fortes sÃ£o entre as pares de escalas similares (125 e 250 metros por exemplo) e escalas maiores (acima de 1 km). Isto nÃ£o Ã© surpreendente, dado que os cÃ¡lculos com um tamanho de buffer maior incluem a Ã¡rea em tamanhos de buffer menores.

4.4.2 Floresta e a ocorrÃªncia de espÃ©cies

Para entender como a cobertura florestal afeta a ocorrÃªncia de espÃ©cies, usamos um modelo de regressÃ£o logÃstica. Esse modelo Ã© usado para prever a probabilidade de um evento ocorrer, como a presenÃ§a de uma espÃ©cie. No caso da regressÃ£o logÃstica, o evento Ã© a presenÃ§a de uma espÃ©cie (1) ou sua ausÃªncia (0).

A regressÃ£o logÃstica Ã© um dos mÃ©todos estatÃsticos mais comumente usados em ecologia. Ã aplicado a uma ampla gama de questÃµes, incluindo:

- DistribuiÃ§Ã£o de espÃ©cies e adequaÃ§Ã£o de habitat:
  Modelagem da probabilidade de uma espÃ©cie ocorrer em um habitat especÃfico com base em fatores ambientais [@pearce2000evaluation].

- DinÃ¢mica populacional:
  AnÃ¡lise de fatores que afetam o crescimento ou declÃnio populacional [@garcia2014rethinking].

- Biologia da conservaÃ§Ã£o:
  Prevendo o impacto das atividades humanas nas espÃ©cies ameaÃ§adas [@MICHALSKI2005383].

- Ecologia comunitÃ¡ria:
  Investigando as relaÃ§Ãµes entre as espÃ©cies e seu ambiente [@warton2015so].

A regressÃ£o logÃstica funciona de forma semelhante Ã regressÃ£o linear, mas Ã© adaptada para dados binÃ¡rios (0 e 1). Na regressÃ£o linear, o modelo prevÃª um valor contÃnuo, como o tamanho do corpo de um animal. Na regressÃ£o logÃstica, o modelo prevÃª a probabilidade de um evento ocorrer, como em nossa caso a presenÃ§a de uma espÃ©cie. O modelo de regressÃ£o logÃstica que usamos Ã© simples, e nÃ£o

Figura 4.2: Cobertura florestal em torno dos locais de amostragem, calculada em diferentes escalas. SÃ£o mostrados grÃ¡ficos de dispersÃ£o da porcentagem de cobertura florestal para cada combinaÃ§Ã£o de escalas em pares. Observe o alto grau de correlaÃ§Ã£o entre as escalas similares (125 e 250 metros por exemplo) e escalas maiores (acima de 1 km).

leva em conta todos os fatores que podem afetar a ocorrÃªncia de espÃ©cies. No entanto, Ã© Ãºtil para ilustrar como efeitos de escala na cobertura florestal pode afetar nossa entendimento sobre a ocorrÃªncia de espÃ©cies.

Comparamos diferentes modelos com base em mediÃ§Ãµes de cobertura florestal em diferentes escalas - extensÃµes locais (distÃ¢ncias de buffer). Para comparar os modelos com buffers diferentes, podemos usar medidas, como medidas de ajuste (verossimilhanÃ§a), a variaÃ§Ã£o explicada e/ou o sucesso preditivo. Uma avaliaÃ§Ã£o de modelo completa incorpora vÃ¡rios critÃ©rios para garantir que sua regressÃ£o logÃstica forneÃ§a insights confiÃ¡veis sobre seus dados ecolÃ³gicos.

Primeiramente vamos rodar as analises para uma especie. Isso seria um exemplo mostrando os passos.

4.4.2.1 terminologia Um detalhe importante sobre a regressÃ£o logÃstica Ã© que este modelo se enquadra na classe de modelos lineares generalizados (Generalized Linear Models - GLM). Logo, no R, este modelo pode ser estimado a partir da funÃ§Ã£o `glm()`, escolhendo a famÃlia binomial no argumento family. Para obter mais detalhes e exemplos mais aprofundados de modelos lineares e modelos lineares generalizados em R, consulte:

- CiÃªncia de Dados com R - Modelos: http://sillasgonzaga.com/material/cdr/modelos

- AnÃ¡lises EcolÃ³gicas no R - Modelos Lineares Generalizados: https://analises-ecologicas.com/cap8

Usando os dados carnivoros, vamos nos referir a:

- A coluna `sp_pa` como variÃ¡vel dependente/resposta, com o valor observado (1 ou 0) denominado respostas.

As outras variÃ¡veis sÃ£o referidas como variÃ¡veis independentes, mas classificaremos adicionalmente:

- A coluna `buff_dist_km` como variÃ¡vel de desenho (isto foi controlada pelo experimentador).
- A coluna `value_median` como variÃ¡vel independente/explanatoria; um covariÃ¡vel especÃfica do ponto de amostragem (proprieadade rural) e distÃ¢ncia de buffer.

A forma estrutural do modelo de regressÃ£o logÃstica Ã©:

$logit(E[Y_i|X_i]) = logit(p_i) = ln(\frac{p_i}{1-p_i}) = \beta_0 + \beta_1 X_{1i} + \epsilon$

A notaÃ§Ã£o tÃpica da regressÃ£o linear inclui:

- Y_i denota a variÃ¡vel de resposta (ou dependente) Y para cada linha i; na regressÃ£o logÃstica esta Ã© uma variÃ¡vel binÃ¡ria
- X_{1i} denota a variÃ¡vel independente (X_1) para cada linha i
- β_0 beta zero denota a interceptaÃ§Ã£o Y quando X Ã© zero; isso nÃ£o Ã© informativo para modelos de regressÃ£o logÃstica
- β_1 beta denota a declive, isso Ã© a mudanÃ§a em Y com uma mudanÃ§a de 1 unidade em X
- p_i denota a probabilidade do evento ocorrer
- ϵ denota os erros

O modelo de regressÃ£o logÃstica Ã© um modelo preditivo para dados binÃ¡rios. ConseqÃ¼entemente, o modelo de regressÃ£o logÃstica pode gerar probabilidades de que uma amostra terÃ¡ o resultado discreto dada(s) uma(s) variÃ¡vel(Ãµes) de entrada. O modelo de regressÃ£o logÃstica usa estimativa de mÃ¡xima verossimilhanÃ§a (Maximum Likelihood Estimation - MLE), que Ã© uma probabilidade condicional que classifica o resultado se um determinado limite for atingido (por exemplo, > 0,50). Portanto, o intervalo de probabilidade de um modelo de regressÃ£o logÃstica estÃ¡ entre 0 e 1.

Para comparar o ajuste de modelos diferentes, usamos a log-verossimilhanÃ§a, que tem um forte base filosÃ³fica em estatÃstica. log-verossimilhanÃ§as sÃ£o baseadas no conceito de mÃ¡xima verossimilhanÃ§a, que Ã© uma tÃ©cnica estatÃstica para estimar os parÃ¢metros de um modelo a partir dos dados. Neste estudo, comparamos modelos que possuem o mesmo nÃºmero de parÃ¢metros. Isso Ã© importante porque permite que os resultados sejam interpretados com mais confianÃ§a. Se os modelos tivessem diferentes nÃºmeros de parÃ¢metros, seria difÃcil dizer se as diferenÃ§as nos resultados eram devido Ã s diferenÃ§as nas escalas espaciais ou ao nÃºmero de parÃ¢metros. Assim sendo, neste caso, o uso de alternativos como critÃ©rios de seleÃ§Ã£o de modelos, como o CritÃ©rio de InformaÃ§Ã£o de Akaike, que penaliza a log-verossimilhanÃ§a com base no nÃºmero de parÃ¢metros [@burnham2002], forneceria resultados idÃªnticos.

4.4.2.2 ConstruÃ§Ã£o de modelos

Lembre-se de que todos os modelos estão errados; a questão prática é quão errados eles devem ser para não serem úteis.

fonte:http://clubes.obmep.org.br/blog/sala-de-estudos-probabilidade-sala-4

Todos os modelos estÃ£o errados, alguns sÃ£o Ãºteis (â All models are wrong, some are useful.â - George E. P. Box : https://en.wikipedia.org/wiki/All_models_are_wrong).

Devemos ter cuidado durante o processo utilizado para construir modelos, assim, nÃ£o somente para fornecer uma representaÃ§Ã£o robusta e Ãºtil do mundo real, mas tambÃ©m para representar fielmente os erros/incertezas associadas. Cada escolha feita mudarÃ¡ a representaÃ§Ã£o dos padrÃµes espaciais e dos processos ecolÃ³gicos. Portanto, para gerar evidÃªncias cientÃficas robustas, todas as etapas devem ser descritas detalhadamente. Um modelo Ã© uma simplificaÃ§Ã£o ou aproximaÃ§Ã£o da realidade e, portanto, nÃ£o refletirÃ¡ toda a realidade. Embora um modelo nunca possa ser â verdadeiroâ , um modelo pode ser classificado de muito Ãºtil, a Ãºtil, a algo Ãºtil e, finalmente, essencialmente inÃºtil [@burnham2002]. Uma avaliaÃ§Ã£o de modelo completa incorpora vÃ¡rios critÃ©rios para garantir que sua regressÃ£o logÃstica forneÃ§a insights confiÃ¡veis sobre seus dados ecolÃ³gicos.

Neste exemplo, analisamos a ocorrÃªncia da espÃ©cie "sppD" em relaÃ§Ã£o Ã cobertura florestal. Antes de construir e executar os modelos, Ã© importante executar uma anÃ¡lise exploratÃ³ria de dados. Aqui usamos boxplots para uma anÃ¡lise exploratÃ³ria preliminar. Existem muitos tipos de anÃ¡lise exploratÃ³ria de dados, que variam dependendo dos dados e do tipo de modelo. No caso de modelos de regressÃ£o logÃstica, para as variÃ¡veis independentes contÃnuas como proporÃ§Ã£o de floresta (`value_median`), podemos tentar grÃ¡ficos de caixa (boxplots) de cada variÃ¡vel independente no exio "y" com a variÃ¡vel dependente (presenÃ§a-ausencia) tratada como um fator de agrupamento no eixo "x".

O boxplot mostra a tendÃªncia central e a variabilidade dos dados, mais sobre boxplot no capÃtlo de visualizaÃ§Ã£o **??**. Aqui, os boxplots mostram que, onde a espÃ©cie estÃ¡ presente, a cobertura florestal Ã©, em mÃ©dia (mediana), mais elevada do que onde a espÃ©cie estÃ¡ ausente. No entanto, hÃ¡ uma variaÃ§Ã£o substancial na cobertura florestal nos locais onde a espÃ©cie estÃ¡ ausente. Essa variaÃ§Ã£o sugere que, embora a cobertura florestal seja um fator importante que influencia a ocorrÃªncia da espÃ©cie, outros fatores tambÃ©m podem desempenhar um papel. Por exemplo, a espÃ©cie pode estar sujeita a outros fatores de pressÃ£o, que podem incluir a caÃ§a, limitaÃ§Ã£o de recursos (comida, abrigo, etc.) e/ou a competiÃ§Ã£o com outras espÃ©cies.

Figura 4.3: GrÃ¡fico para explorar as relaÃ§Ãµes entre as variÃ¡veis. Boxplot comparando Ã¡rea central de floresta entre a presenÃ§a-ausenÃ§a de uma espÃ©cie em escalas diferentes (distÃ¢ncias de 0.125 a 4 km).

A previsÃ£o da ausÃªncia de espÃ©cies na ecologia apresenta vÃ¡rios desafios que a tornam mais difÃcil do que a previsÃ£o da sua presenÃ§a. Aqui estÃ£o alguns motivos principais:

1. DetecÃ§Ã£o imperfeita:
   Detectar de forma conclusiva a ausÃªncia de uma espÃ©cie Ã© muitas vezes mais difÃcil do que confirmar a sua presenÃ§a. A falta de detecÃ§Ã£o pode ser devido a:
   - Baixas taxas de encontro: Algumas espÃ©cies podem ser naturalmente raras ou secretas, tornando-as difÃceis de encontrar mesmo quando estÃ£o presentes.
   - Vieses de detecÃ§Ã£o: Os mÃ©todos de amostragem podem ser tendenciosos em relaÃ§Ã£o a determinadas espÃ©cies ou habitats, levando a falsos negativos para outros.
   - Fatores ambientais: As condiÃ§Ãµes ambientais, como o clima ou a cobertura vegetal, podem influenciar a detectabilidade, tornando mais difÃcil inferir a verdadeira ausÃªncia.
2. Nichos ecolÃ³gicos:
   Os nichos ecolÃ³gicos sÃ£o complexos e multifacetados. Prever a ausÃªncia requer a compreensÃ£o de toda a gama de condiÃ§Ãµes e recursos ambientais que uma espÃ©cie pode tolerar, o que pode ser um desafio devido a:
   - Conhecimento limitado: A nossa compreensÃ£o das necessidades ecolÃ³gicas de muitas espÃ©cies ainda Ã© incompleta, especialmente para tÃ¡xons menos estudados.
   - Variabilidade espacial e temporal: A adequaÃ§Ã£o do habitat pode variar no espaÃ§o e no tempo, tornando difÃcil prever a ausÃªncia de uma espÃ©cie em todos os locais potenciais.
   - PreferÃªncias de microhabitat: Algumas espÃ©cies podem ter preferÃªncias especÃficas de microhabitat em Ã¡reas aparentemente adequadas, tornando a sua ausÃªncia mais difÃcil de identificar.
3. Efeitos de limiar:
   As respostas das espÃ©cies aos factores ambientais envolvem frequentemente limiares, onde estÃ£o presentes acima de um determinado valor, mas ausentes abaixo dele. Identificar esses limites com precisÃ£o pode ser difÃcil, levando a incertezas na previsÃ£o de Ã¡reas de ausÃªncia.

4.4.2.3 Pergunta 3 **Identifique a espÃ©cie "sppD". Com base nos resultados dos boxplots e os artigos (Michalski & Peres 2005; Michalski & Peres 2007; Regolin et. al. 2017), sugerem trÃªs espÃ©cies de carnÃvoros que poderiam apresentar os mesmos padrÃµes do sppD? Que informaÃ§Ãµes/resultados adicionais o ajudariam a identificar a espÃ©cie sppD com maior certeza? Justifique suas escolhas de forma clara e concisa.**

Os boxplots sÃ£o Ãºteis para anÃ¡lises exploratÃ³rias, mas nÃ£o fornecem muitas informaÃ§Ãµes para a compreensÃ£o da escala do efeito.

Agora plotamos os dados, ajustando uma regressÃ£o logÃstica para obter uma representaÃ§Ã£o visual. Ã Ãºtil executar isso (representaÃ§Ã£o visual) antes de executar os modelos, para que tenhamos uma ideia dos padrÃµes que existem nos dados. Como no capÃtulo anterior usamos a funÃ§Ã£o `stat_smooth()` para sobrepÃµe uma curva da regressÃ£o logÃstica ajustada aos dados.

Os graficos mostrarm a mesma padrao que os boxplots, mas com informaÃ§Ãµes adicionais. Como vimos nos boxplots a probabilidade de ocorrÃªncia aumenta com a cobertura florestal. Agora podemos ver mais claramente como a forma e a forÃ§a da relaÃ§Ã£o com a cobertura florestal mudam em diferentes escalas. Por exemplo, observe as diferenÃ§as dos valores da cobertura florestal com 50% de probabilidade (linhas horizontais tracejadas).

Figura 4.4: GrÃ¡fico para explorar as relaÃ§Ãµes entre as variÃ¡veis. Comparando a resposta de presenÃ§a-ausenÃ§a de uma espÃ©cie em escalas diferentes (distÃ¢ncias de 0.125 a 4 km). Pontos mostram presenÃ§a-ausÃªncia da espÃ©cie em 106 propriedades rurais. A linha azul Ã© o ajuste de uma regressÃ£o logÃstica. A linha horizontal tracejada Ã© a probabilidade de ocorrÃªncia de 50%.

Uma impressÃ£o visual Ã© Ãºtil. Mas devemos utilizar modelos estatÃsticos para fornecer evidÃªncias mais robustas e objetivas. Usamos cÃ³digo que executarÃ¡ modelos simultaneamente para todas as distÃ¢ncias de buffer. Podemos fazer a mesma tarefa com modelos e resultados separadamente para cada distÃ¢ncia de buffer. Mas, seria pelo menos 50 linhas de codigo para rodar os modelos separadamente e obter os resultados que precisamos. Rodando os modelos seperadamente seria viavel se precisÃ¡ssemos de apenas um ou dois modelos. Mas torna-se muito ineficiente e muito propenso a gerar erros quando precisamos executar vÃ¡rios modelos diferentes, por exemplo, em distÃ¢ncias diferentes e para espÃ©cies diferentes. Aqui usamos funÃ§Ãµes oriundos dos pacotes tidyverse e tidymodels para rodar os modelos e obter os resumos que precisamos em menos de 10 linhas - fazendo mÃºltiplos modelos de regressÃ£o simultaneamente no R. Vamos fazer mÃºltiplos modelos simultaneamente usando funÃ§Ãµes de:

- listas aninhadas `tidyr::nest`,
- iteraÃ§Ã£o `purr::map` e
- extraÃ§Ã£o de resultados arrumados `broom::glance` e `broom::tidy`.

Embora possa parecer complicado, este Ã© um processo poderoso que pode ser aplicado a grandes conjuntos de dados e funciona para muitos dos diferentes tipos de modelos que vocÃª pode executar em R (incluindo aprendizado de mÃ¡quina). O processo inclui dois princÃpios bÃ¡sicos:

1. Usando colunas de lista para armazenar estruturas de dados em um data frame. Por exemplo, isso permitirÃ¡ que vocÃª tenha uma coluna que contenha modelos lineares generalizados como uma regressÃ£o logistica.
2. Usando o pacote broom, para transformar modelos em dados organizados.
   Essa Ã© uma tÃ©cnica poderosa para trabalhar com um grande nÃºmero de modelos porque, depois de ter dados organizados, vocÃª poderÃ¡ aplicar muitas opÃ§oes para anÃ¡lise e visualizaÃ§Ãµes rapidamente e sem erros.

Em menos de 10 linhas, incluimos funÃ§oes para obter um conjunto de 8 modelos diferentes (um modelo para cada distÃ¢ncia de buffer) e resultados associados dentro do objeto final "modelos_sppD". Para isso o processo tem quatro etapas conectadas com o operador pipe `|>` . ComeÃ§ando com o banco de dados carnivoros, na sequencia:

1. aplicar um filtro para selecionar o especie "sppD" `dplyr::filter`
2. selecionar as colunas que precisamos no modelos `dplyr::select`
3. divida os dados em partes que serÃ£o modeladas separadamente (listas aninhadas para 8 modelos, um para cada distÃ¢ncia do buffer) `tidyr::nest`
4. aplique os modelos e salve os resultados `dplyr::mutate` .

   A Ãºltima etapa inclui a criaÃ§Ã£o de trÃªs novos itens usando a funÃ§Ã£o `map` :

   - fit: o modelo de regressÃ£o logÃstica a ser aplicado `glm`
   - glanced: resumo do modelo (informaÃ§Ãµes e ajustes dos modelos)
   - tidied: resumo das variÃ¡veis (estimativas, intervalos de confianÃ§a etc)

```
# Modelos para sppD e 8 distÃ¢ncias de buffer
modelos_sppD <- carnivoros |>
# 1. filtro a especie desejada
  dplyr::filter(sp_name == "sppD") |>
# 2. selecionar as colunas que faz parte dos modelos
  dplyr::select(sp_name, buff_dist_km, sp_pa, value_median) |>
# 3. agrupar para rodar modelos diferentes para especie
# e escala (distÃ¢ncia de buffer)
  tidyr::nest(.by = c(sp_name, buff_dist_km)) |>
# 4. especificar modelo e os resumos desejados
  dplyr::mutate(
    fit = purrr::map(data, ~ glm(sp_pa ~ value_median,
                          data = .x, family = "binomial")
              ),
    glanced = purrr::map(fit, broom::glance),
    tidied = purrr::map(fit, broom::tidy, conf.int = TRUE, conf.level = 0.95)
  )
```

Olhar os resultados. Se tudo correr bem, haverÃ¡ um novo objeto de dados "modelos_sppD" no ambiente. Existem 8 linhas (um modelo para cada distÃ¢ncia do buffer) e 6 colunas. As colunas incluem os dados ("data") e resumos para cada modelo ("glanced") e para cada variavel em cada modelo ("tidied")

```
modelos_sppD
```

```
## # A tibble: 8 x 6
##   sp_name buff_dist_km data               fit    glanced          tidied
##   <chr>          <dbl> <list>             <list> <list>           <list>
## 1 sppD           0.125 <tibble [106 x 2]> <glm>  <tibble [1 x 8]> <tibble [2 x 7]>
## 2 sppD           0.25  <tibble [106 x 2]> <glm>  <tibble [1 x 8]> <tibble [2 x 7]>
## 3 sppD           0.5   <tibble [106 x 2]> <glm>  <tibble [1 x 8]> <tibble [2 x 7]>
## 4 sppD           1     <tibble [106 x 2]> <glm>  <tibble [1 x 8]> <tibble [2 x 7]>
## 5 sppD          16     <tibble [106 x 2]> <glm>  <tibble [1 x 8]> <tibble [2 x 7]>
## 6 sppD           2     <tibble [106 x 2]> <glm>  <tibble [1 x 8]> <tibble [2 x 7]>
## 7 sppD           4     <tibble [106 x 2]> <glm>  <tibble [1 x 8]> <tibble [2 x 7]>
## 8 sppD           8     <tibble [106 x 2]> <glm>  <tibble [1 x 8]> <tibble [2 x 7]>
```

Olhar resumos dos modelos. Para verificar o conteudo de uma coluna com lista aninhada feita com `tidyr::nest` , precisa desembrulhar com `tidyr::unnest` . O prÃ³ximo cÃ³digo mostrarÃ¡ os valores de resumo para cada modelo. Isso inclui as colunas "logLik" (log-verossimilhanÃ§a) e "AIC" (CritÃ©rio de

InformaÃ§Ã£o de Akaike) que usaremos para compare e identifique a escala do efeito.

```
modelos_sppD |>
  tidyr::unnest(glanced)
```

```
## # A tibble: 8 x 13
##   sp_name buff_dist_km data     fit    null.deviance df.null logLik   AIC   BIC
##   <chr>          <dbl> <list>   <list>         <dbl>   <int>  <dbl> <dbl> <dbl>
## 1 sppD           0.125 <tibble> <glm>           143.     105  -63.8 132.  137.
## 2 sppD           0.25  <tibble> <glm>           143.     105  -54.6 113.  119.
## 3 sppD           0.5   <tibble> <glm>           143.     105  -47.5  99.1 104.
## 4 sppD           1     <tibble> <glm>           143.     105  -44.6  93.2  98.5
## 5 sppD          16     <tibble> <glm>           143.     105  -50.2 104.  110.
## 6 sppD           2     <tibble> <glm>           143.     105  -46.8  97.7 103.
## 7 sppD           4     <tibble> <glm>           143.     105  -49.1 102.  108.
## 8 sppD           8     <tibble> <glm>           143.     105  -48.8 102.  107.
## # i 4 more variables: deviance <dbl>, df.residual <int>, nobs <int>, tidied <list>
```

Olhar resumos das variÃ¡veis, ou seja valores que resume as "descobertas" estatÃsticas do modelo, como as estimativas, erro padrÃ£o ("std.error"), intervalos de confianÃ§a (colunas "conf.low" e "conf.high"), e valores de p (coluna "p.value"). A coluna "estimate" (termos beta) tem os valores quantificando a relaÃ§Ã£o entre a ocorrÃªncia de espÃ©cie sppD e a cobertura florestal. Quando os valores na coluna "estimate" sÃ£o positivos existe uma relaÃ§Ã£o positiva com a variavel na coluna "term", e quando sÃ£o negativos existe uma relaÃ§Ã£o negativa. Para saber se esta relaÃ§Ã£o Ã© significativa podemos observar o valor p juntamente com os intervalos de confianÃ§a. Para serem significativos, os intervalos de confianÃ§a nÃ£o devem se sobrepor a zero.

```
modelos_sppD |>
  tidyr::unnest(tidied)
```

```
## # A tibble: 16 x 12
##    sp_name buff_dist_km data     fit    glanced  term   estimate std.error statistic
##    <chr>          <dbl> <list>   <list> <list>   <chr>     <dbl>     <dbl>     <dbl>
##  1 sppD           0.125 <tibble> <glm>  <tibble> (Inte~  -1.08      0.285      -3.81
##  2 sppD           0.125 <tibble> <glm>  <tibble> value~   0.0419    0.0114      3.68
##  3 sppD           0.25  <tibble> <glm>  <tibble> (Inte~  -2.46      0.514      -4.79
##  4 sppD           0.25  <tibble> <glm>  <tibble> value~   0.0662    0.0139      4.76
##  5 sppD           0.5   <tibble> <glm>  <tibble> (Inte~  -4.66      1.03       -4.51
##  6 sppD           0.5   <tibble> <glm>  <tibble> value~   0.0832    0.0177      4.70
##  7 sppD           1     <tibble> <glm>  <tibble> (Inte~  -5.89      1.35       -4.36
##  8 sppD           1     <tibble> <glm>  <tibble> value~   0.0853    0.0185      4.62
##  9 sppD          16     <tibble> <glm>  <tibble> (Inte~  -4.29      0.873      -4.91
## 10 sppD          16     <tibble> <glm>  <tibble> value~   0.0543    0.0106      5.13
## 11 sppD           2     <tibble> <glm>  <tibble> (Inte~  -5.45      1.22       -4.48
## 12 sppD           2     <tibble> <glm>  <tibble> value~   0.0726    0.0154      4.72
## 13 sppD           4     <tibble> <glm>  <tibble> (Inte~  -4.88      1.07       -4.58
## 14 sppD           4     <tibble> <glm>  <tibble> value~   0.0624    0.0130      4.80
## 15 sppD           8     <tibble> <glm>  <tibble> (Inte~  -4.55      0.958      -4.74
## 16 sppD           8     <tibble> <glm>  <tibble> value~   0.0575    0.0115      4.99
## # i 3 more variables: p.value <dbl>, conf.low <dbl>, conf.high <dbl>
```

4.4.2.4 Compare e identifique a escala do efeito Agora vamos usar os resultados em "modelos_sppD" para responder duas perguntas:

1. Qual escala melhor se ajusta aos dados de presenÃ§a-ausencia de sppD?
2. A relaÃ§Ã£o entre a ocorrÃªncia de espÃ©cie sppD e a cobertura florestal Ã© a mesma em diferentes

escalas?

Qual escala melhor se ajusta aos dados de presenÃ§a-ausencia de sppD?.

Ajustamos modelos de regressÃ£o logÃstica aos dados, usando a cobertura florestal como variÃ¡vel independente. Para identificar a escala do efeito da cobertura florestal na ocorrÃªncia de sppD, plotamos as log-verossimilhanÃ§as, desvio explicado e AIC de diferentes modelos de regressÃ£o logÃstica ajustados aos dados em funÃ§Ã£o da cobertura florestal calculada em diferentes escalas, variando de 0.125 m a 16 km.

- Log-verossimilhanÃ§a: uma medida da probabilidade dos dados serem observados sob o modelo.
  A log-verossimilhanÃ§a mede quÃ£o bem as probabilidades previstas do modelo correspondem aos dados observados. Valores mais altos de log-verossimilhanÃ§a indicam um melhor ajuste. InterpretaÃ§Ã£o:
  - NÃ£o Ã© diretamente interpretÃ¡vel por si sÃ³, mas Ã© usado para calcular outras medidas, como desvio e AIC.
  - Comparar log-verossimilhanÃ§as de diferentes modelos (aninhados ou com o mesmo nÃºmero de parÃ¢metros) pode indicar qual deles se ajusta melhor aos dados.
- Desvio explicado: uma medida da proporÃ§Ã£o da variaÃ§Ã£o na ocorrÃªncia que Ã© explicada pelo modelo.
  O desvio explicado Ã© a proporÃ§Ã£o do desvio total (desvio nulo) que Ã© explicado pelo modelo. InterpretaÃ§Ã£o:
  - Representa a capacidade do modelo de capturar a variabilidade na variÃ¡vel resposta.
  - Os valores variam de 0 a 1, com valores mais altos indicando melhor ajuste.
  - Pode ser interpretado de forma semelhante ao R-quadrado na regressÃ£o linear.
  - NÃ£o existe um Ãºnico valor â bomâ para o desvio explicado em uma regressÃ£o logÃstica em ecologia, pois depende de vÃ¡rios fatores, incluindo o tamanho da amostra e complexidade do modelo:

  - Tamanho da amostra:
    Amostras maiores: Geralmente, com amostras maiores, vocÃª verÃ¡ valores explicados de desvio mais altos devido ao aumento do poder estatÃstico para detectar efeitos menores. Um valor acima de 0,5 pode ser considerado bom em amostras grandes.
    Amostras menores: Valores em torno de 0,2-0,3 podem ser considerados decentes para amostras menores devido Ã s limitaÃ§Ãµes na detecÃ§Ã£o de efeitos sutis.

  - Complexidade do modelo:
    Modelos mais simples: com menos variÃ¡veis explicativas, vocÃª provavelmente verÃ¡ valores explicados de desvio mais baixos, mesmo que o modelo capture os padrÃµes principais. Valores acima de 0,3 podem ser bons para esses modelos.
    Modelos mais complexos: Esses modelos tendem a ter valores explicados de desvio mais altos, mas isso nÃ£o significa necessariamente que sejam melhores. Valores altos tambÃ©m podem indicar ajuste excessivo aos dados especÃficos. Valores prÃ³ximos a 0,8 ou acima podem exigir um exame cuidadoso para overfitting.
- AIC: Ãndice de informaÃ§Ã£o de Akaike.
  O AIC equilibra a qualidade do ajuste (medida pela Log-verossimilhanÃ§a) com a complexidade do modelo (nÃºmero de parÃ¢metros). Penaliza os modelos pela sua complexidade, recompensando modelos mais simples que se ajustam quase tÃ£o bem aos dados. Isso significa que um modelo com uma valor de AIC mais baixa Ã© geralmente considerado mais preferÃvel, pois provavelmente representa bem os dados, evitando o sobreajuste. InterpretaÃ§Ã£o:
  - Valores mais baixos de AIC indicam um melhor equilÃbrio entre ajuste e parcimÃ´nia. Ou seja, valores mais baixos sÃ£o melhores, ao contrÃ¡rio de medidas como desvio explicado, onde valores mais altos sÃ£o melhores.
  - Usado para seleÃ§Ã£o de modelo, sendo geralmente preferido o modelo com o AIC mais baixo.

EsperarÃamos que a escala que melhor se ajustasse aos dados fosse a que apresentasse os maiores valores de log-verossimilhanÃ§a, desvio explicado e valor de AIC mais baixo. Nesse caso, descobrimos que a cobertura

florestal dentro de 1 km Ã© a que melhor se ajusta aos dados.

A relaÃ§Ã£o entre a ocorrÃªncia de espÃ©cie sppD e a cobertura florestal Ã© a mesma em diferentes escalas?.

Vimos nos grÃ¡ficos comparando a resposta de presenÃ§a-ausenÃ§a de sppD em escalas diferentes, que existem diferenÃ§as no padrÃ£o (50% de probabilidade aumenta de 25% para 75% de cobertura florestal de 0,125 para 1 km). Ou seja, em escalas maiores, sÃ£o necessÃ¡rias maiores proporÃ§Ãµes de floresta para que a sppD ocorra. Mas existe algum suporte estatÃstico para mostrar que as diferenÃ§as entre escalas sÃ£o relevantes?

Para identificar se ha diferenÃ§as na relaÃ§Ã£o entre a ocorrÃªncia de espÃ©cie sppD e a cobertura florestal, plotamos os valores estimados (coluna "estimate") para a variavel independente (value_median). Inclumos intervalos de confianÃ§a de diferentes modelos de regressÃ£o logÃstica ajustados aos dados em funÃ§Ã£o da cobertura florestal calculada em diferentes escalas, variando de 0.125 m a 16 km. Ao representar graficamente os termos beta em diferentes escalas, descobrimos que SÃ£o todos significativamente positivos (os intervalos de confianÃ§a nÃ£o se sobrepÃµem a zero). A forÃ§a da relaÃ§Ã£o com a floresta dobra entre 0,125 e 1 km, mas depois diminui. Mas hÃ¡ uma grande sobreposiÃ§Ã£o nos intervalos de confianÃ§a. Esta incerteza Ã© esperada porque sabemos que os modelos explicam 30 por cento ou menos do desvio global.

- ConclusÃ£o:
  HÃ¡ evidÃªncias consistentes de que a ocorrÃªncia de sppD aumenta com a cobertura florestal na paisagem (com base na cobertura medida em 1 km e em escalas maiores).

4.4.2.5 Pergunta 4 **O que sabemos sobre a espÃ©cie "sppD"? Com base nos resultados obtidos: boxplots, graficos com a relaÃ§Ã£o entre occorencia e proporÃ§Ã£o de floresta, resultados sobre escala do efeito e os artigos (Michalski & Peres 2005; Michalski & Peres 2007; Regolin et. al. 2017), forneÃ§a uma caracterizaÃ§Ã£o da espÃ©cie (por exemplo, especialista/generalista/sensÃvel/resiliente). Justifique a sua caracterizaÃ§Ã£o e interpretaÃ§Ãµes com resultados especÃficos, destacando os resultados com maior relevÃ¢ncia. Que informaÃ§Ãµes/resultados adicionais o ajudariam a caracterizar a espÃ©cie sppD com maior certeza? Justifique suas escolhas de forma clara e concisa.**

4.4.2.6 MÃºltiplas espÃ©cies

Vamos agora repetir o processo para todas as trÃªs espÃ©cies nos dados carnÃvoros. Indo na mesma sequÃªncia: boxplots, grÃ¡ficos de regressÃ£o logÃstica e, em seguida, escala de efeito pela plotagem de resumos de modelos de regressÃ£o logÃstica e termos beta.

Primeiramente os boxplots:

Agora plotamos os dados, ajustando uma regressÃ£o logÃstica para obter uma representaÃ§Ã£o visual. Ã Ãºtil executar isso (representaÃ§Ã£o visual) antes de executar os modelos, para que tenhamos uma ideia dos padrÃµes que existem nos dados. Como no capÃtulo anterior usamos a funÃ§Ã£o `stat_smooth()` para sobrepÃµe uma curva da regressÃ£o logÃstica ajustada aos dados.

Para identificar a escala do efeito da cobertura florestal na ocorrÃªncia de espÃ©cies, plotamos as log-verossimilhanÃ§as, desvio explicado e AIC de diferentes modelos de regressÃ£o logÃstica ajustados aos dados em funÃ§Ã£o da cobertura florestal calculada em diferentes escalas, variando de 0.125 m a 16 km. EsperarÃamos que a escala que melhor se ajustasse aos dados fosse a que apresentasse os maiores valores de log-verossimilhanÃ§a, desvio explicado e valor de AIC mais baixo:

Para identificar se ha diferenÃ§as na relaÃ§Ã£o entre a ocorrÃªncia de espÃ©cies e a cobertura florestal, plotamos os valores estimados (coluna "estimate") para a variavel independente (value_median). Inclumos intervalos de confianÃ§a de diferentes modelos de regressÃ£o logÃstica ajustados aos dados em funÃ§Ã£o da cobertura florestal calculada em diferentes escalas, variando de 0.125 m a 16 km:

Figura 4.5: Escala do efeito da cobertura florestal na ocorrÃªncia do sppD, com base em uma anÃ¡lise de buffer. A probabilidade de ocorrÃªncia foi modelada como uma funÃ§Ã£o da porcentagem de floresta, calculada em diferentes escalas usando um buffer circular em torno de pontos (0,125â 16 km). Escala do efeito identificada usando trÃªs medidas de ajuste do modelo (i) CritÃ©rio de informaÃ§Ã£o de Akaike (AIC). Um AIC mais baixo sugere um modelo mais Ãºtil. (ii) a proporÃ§Ã£o de desvio explicado para os modelos. Um maior desvio explicado sugere um melhor ajuste do modelo aos dados. (iii) A verossimilhanÃ§a logarÃtmica (logLik) para modelos de regressÃ£o logÃstica. Uma log-verossimilhanÃ§a mais alta sugere um melhor ajuste do modelo aos dados. A linha azul vertical tracejada mostra a melhor escala para cada medida de ajuste do modelo.

Figura 4.6: As estimativas do parÃ¢metro para o efeito da cobertura florestal (termo beta) na probabilidade de ocorrÃªncia de sppD usando buffers de diferentes tamanhos. A probabilidade de ocorrÃªncia foi modelada em funÃ§Ã£o da porcentagem de floresta, calculada em diferentes escalas usando um buffer circular em torno de pontos (0,125â 16 km). Os pontos sÃ£o as estimativas mÃ©dias com linhas verticais mostrando intervalos de confianÃ§a de 95%. A linha azul vertical tracejada mostra a melhor escala para cada medida de ajuste do modelo.

4.4.2.7 Pergunta 5

O que sabemos sobre as trÃªs espÃ©cies de carnÃvoros? Com base nos resultados obtidos: boxplots, graficos com a relaÃ§Ã£o entre occorencia e proporÃ§Ã£o de floresta, resultados sobre escala do efeito e os artigos (Michalski & Peres 2005; Michalski & Peres 2007; Regolin et. al. 2017), forneÃ§a uma identificaÃ§Ã£o (nome cientÃfico e nome comum) e caracterizaÃ§Ã£o de cada espÃ©cie (por exemplo, especialista/generalista/sensÃvel/resiliente). Justifique a sua identificaÃ§Ã£o, caracterizaÃ§Ã£o e interpretaÃ§Ãµes com resultados especÃficos, destacando os resultados com maior relevÃ¢ncia. Que informaÃ§Ãµes/resultados adicionais o ajudariam a caracterizar as espÃ©cies com maior certeza? Justifique suas escolhas de forma clara e concisa.

4.4.2.8 Pergunta 6

Vimos como a incerteza quanto Ã escala do efeito Ã© provavelmente, pelo menos em parte, uma consequÃªncia da correlaÃ§Ã£o entre escalas. TambÃ©m Ã© possÃvel que, alÃ©m da cobertura florestal, existam outros fatores que afetem a ocorrÃªncia das trÃªs espÃ©cies na paisagem estudada. Usando como base o conteudo das aulas, leitura disponivel no Google Classroom (Base teÃ³rica 4 Dados, mÃ©tricas, analises), e/ou exemplos apresentados aqui no capitulo, apresentam algumas hipÃ³teses alternativas que poderiam explicar a incerteza nos modelos para as trÃªs espÃ©cies. ForneÃ§a recomendaÃ§Ãµes para variÃ¡veis adicionais que poderiam ser incluÃdas nos modelos para testar suas hipÃ³teses. Justifique sua seleÃ§Ã£o de forma clara e concisa, apoie sua escolha com exemplos da literatura cientÃfica.

Figura 4.7: Boxplot comparando Ã¡rea central de floresta entre a presenÃ§a-ausenÃ§a de espÃ©cies em escalas diferentes (distÃ¢ncias de 0.125 a 16 km).

Essa foi a Ãºltima pergunta. O que se segue Ã© um exemplo que mostra uma abordagem alternativa.

SoluÃ§Ãµes dos exercÃcios: https://darrennorris.github.io/eprsol/solutions03_especies.html

Figura 4.8: GrÃ¡fico para explorar as relaÃ§Ãµes entre as variÃ¡veis. Comparando a resposta de presenÃ§a-ausenÃ§a de trÃªs espÃ©cies de carnÃvoros em escalas diferentes (distÃ¢ncias de 0.125 a 16 km). Pontos mostram presenÃ§a-ausÃªncia da espÃ©cie em 106 propriedades rurais. A linha azul Ã© o ajuste de uma regressÃ£o logÃstica. A linha horizontal tracejada Ã© a probabilidade de ocorrÃªncia de 50%.

4.5 Kernels

Nesta seÃ§Ã£o usaremos funÃ§Ãµes no pacote `siland` para executar uma anÃ¡lise de escala de efeito usando Kernels espaciais.

A abordagem de Kernels fornece uma maneira rigorosa de estimar as escalas nas quais as variÃ¡veis da paisagem afetam respostas ecologicas, e pode ser incorporado na maioria das classes de modelos estatÃsticos [@chandler2016estimating]. Esta abordagem pode reduzir o viÃ©s do modelo e levar a interpretaÃ§Ãµes mais precisas das relaÃ§Ãµes espÃ©cie-paisagem [@Aue2012; @chandler2016estimating; @Miguet2017dwm]. A funÃ§Ã£o de suavizaÃ§Ã£o Kernel assume que o efeito de uma variÃ¡vel da paisagem Ã© mais forte perto do local focal e decai em funÃ§Ã£o da distÃ¢ncia. Assim fornecendo uma representaÃ§Ã£o mais realista de muitos processos ecolÃ³gicos como por exemplo, dispersÃ£o de sementes [@bullock2017] e movimentos de forrageadoras de lugar central (Orians e Pearson 1979).

4.6 Links Ãºteis

Software R curso avanÃ§ado: https://smolski.github.io/livroavancado/
CapÃtulo 7 RegressÃ£o LogÃstica: https://smolski.github.io/livroavancado/reglog.html

AnÃ¡lises EcolÃ³gicas no R Modelos Lineares Generalizados: https://analises-ecologicas.com/cap8

EstatÃstica com R: https://storopoli.io/Estatistica/
RegressÃ£o LogÃstica : https://storopoli.io/Estatistica/7-Regressao_Logistica.html

Tidy Modeling with R: https://www.tmwr.org/

Figura 4.9: Escala do efeito da cobertura florestal na ocorrÃªncia de trÃªs espÃ©cies de carnÃvoros, com base em uma anÃ¡lise de buffer. A probabilidade de ocorrÃªncia foi modelada como uma funÃ§Ã£o da porcentagem de floresta, calculada em diferentes escalas usando um buffer circular em torno de pontos (0,125â 16 km). Escala do efeito identificada usando trÃªs medidas de ajuste do modelo (i) CritÃ©rio de informaÃ§Ã£o de Akaike (AIC). Um AIC mais baixo sugere um modelo mais Ãºtil. (ii) a proporÃ§Ã£o de desvio explicado para os modelos. Um maior desvio explicado sugere um melhor ajuste do modelo aos dados. (iii) A verossimilhanÃ§a logarÃtmica (logLik) para modelos de regressÃ£o logÃstica. Uma log-verossimilhanÃ§a mais alta sugere um melhor ajuste do modelo aos dados. A linha azul vertical tracejada mostra a melhor escala para cada medida de ajuste do modelo.

models: https://www.tmwr.org/models workflow: https://www.tmwr.org/workflows

R for Data Science Many Models: https://r4ds.had.co.nz/many-models.html

Astonishingly easy mapping in R with mapview: https://www.infoworld.com/article/3644848/astonishingly-easy-mapping-in-r-with-mapview.html

Video - Fazer mÃºltiplos modelos de regressÃ£o simultaneamente no R: https://www.youtube.com/watch?v=Wukf7v3_u4I

Figura 4.10: Estimativas para o efeito da cobertura florestal (termo beta) na ocorrÃªncia de trÃªs espÃ©cies usando buffers de diferentes tamanhos. As estimativas do parÃ¢metro para o efeito da cobertura florestal (termo beta) na probabilidade de ocorrÃªncia das trÃªs espÃ©cies. A probabilidade de ocorrÃªncia foi modelada em funÃ§Ã£o da porcentagem de floresta, calculada em diferentes escalas usando um buffer circular em torno de pontos (0,125â 16 km). Os pontos sÃ£o as estimativas mÃ©dias com linhas verticais mostrando intervalos de confianÃ§a de 95%. A linha azul vertical tracejada mostra a melhor escala para cada medida de ajuste do modelo.

Figura 4.11: Esquema ilustrativo demonstrando diferentes tipos de kernels espaciais: (A) diagrama representando a estrutura geral de um filtro espacial baseado em kernel; (B) os cinco tipos mais comuns de distribuiÃ§Ãµes espaciais de kernel; (C) nÃºcleos espaciais de largura de banda de distÃ¢ncia fixa (estÃ¡tica); e (D) nÃºcleos espaciais de distÃ¢ncia adaptativa. Observe que a figura Ã© baseada nas seguintes fontes: Figura 3.4 na pÃ¡gina 86 de Bailey e Gatrell (1995); Fotheringham et al. (2002) Figuras 2.11 e 2.13 nas pÃ¡ginas 45 e 47 respectivamente; Figura 3.2 na pÃ¡gina 37 de Wang (2006); pÃ¡ginas 67-68 de Smith e Bruce (2008); e Figura 4-47 na pÃ¡gina 177 de de Smith et al. (2009).

Part III

Exemplos de caso: padrÃµes de paisagem

5 Garimpo do LourenÃ§o

5.1 ApresentaÃ§Ã£o

MudanÃ§as na paisagem ao redor do Garimpo do LourenÃ§o. Changes in the landscape surrounding the LourenÃ§o gold mine.

CÃ³digo de R e dados para calcular mÃ©tricas de paisagem associadas com a exploraÃ§Ã£o de recursos minerÃ¡rios.

O objetivo Ã© calcular mÃ©tricas de paisagem e descrever a composiÃ§Ã£o e a configuraÃ§Ã£o da paisagem no entorno do Garimpo do LourenÃ§o.

As mÃ©tricas de paisagem sÃ£o a forma que os ecÃ³logos de paisagem usam para descrever os padrÃµes espaciais de paisagens para depois avaliar a influÃªncia destes padrÃµes espaciais nos padrÃµes e processos ecolÃ³gicos.

Nesta exemplo (https://rpubs.com/darren75/lourenco) aprenderemos sobre como analisar a cobertura da terra com mÃ©tricas de paisagem em R.

Este exemplo tem como base teÃ³rica o modelo "mancha-corredor-matriz" - uma representaÃ§Ã£o da paisagem em manchas de habitat (fragmentos).

5.2 Pacotes necessarios:

```
library(tidyverse)
library(readxl)
library(terra)
library(sf)
library(landscapemetrics)
library(mapview)
library(knitr)
library(gridExtra)
library(eprdados)
```

5.3 Ã rea de estudo

Para alcanÃ§ar o objetivo de caracterizar a paisagem no entorno do Garimpo do LourenÃ§o, precisamos estabelecer a extensÃ£o da Ã¡rea de estudo. Isso seria estabelicida com base nos objetivos e estudos anteriores. Sabemos que atividades asociados com a mineraÃ§Ã£o pode aumentar a perda da floresta atÃ© 70 km alÃ©m dos limites do processo de mineraÃ§Ã£o: Sonter et. al. 2017. Mining drives extensive deforestation in the Brazilian Amazon https://www.nature.com/articles/s41467-017-00557-w

Para visualizar um exemplo com a ExtraÃ§Ã£o de bauxita na Flona SaracÃ¡-Taquera: https://earthengine.google.com/timelapse/#v=-1.70085,-56.45017,8.939,latLng&t=2.70

E aqui com o Garimpo do LourenÃ§o: https://earthengine.google.com/timelapse#v=2.2994,-51.68423,11.382,latLng&t=0.03

5.4 Dados

5.4.1 Ponto de referÃªncia (EPSG: 4326)

Aqui vamos incluir um raio de 20 km alÃ©m do ponto de acesso para o Garimpo do LourenÃ§o em 1985. Isso representa uma Ã¡rea quadrada de 40 x 40 km (1600 km2).

```
# Tabela de dados com coordenados de acesso em 1985.
acesso <- data.frame(nome = "garimpo do LourenÃ§o",
           coord_x = -51.630871,
           coord_y = 2.318514)

# Converter para objeto espacial, com sistema de coordenados geogrÃ¡fica.
sf_acesso <- st_as_sf(acesso,
               coords = c("coord_x", "coord_y"),
            crs = 4326)
```

Visualizar para verificar.

```
#
plot(sf_acesso) # teste basica
mapview(sf_acesso) #verificar com mapa de base (OpenStreetMap)
```

5.4.2 Ponto de referÃªncia (EPSG: 31976)

As anÃ¡lises da paisagem com o modelo "mancha-corredor-matriz" depende de uma classificaÃ§Ã£o categÃ³rica. Portanto, deve optar para uma sistema de coordenados projetados, com pixels de Ã¡rea igual e com unidade em metros. Temos um raio de 20 km, que Ã© um area geografica onde o retÃ¢ngulo envolvente Ã© menor que um fuso UTM. Assim sendo, vamos adotar a sistema de coordenados projetados de datum SIRGAS 2000, especificamente EPSG:31976 (SIRGAS 2000/UTM zone 22N).

Precisamos entÃ£o reprojetar o objeto original (em coordenados geogrÃ¡ficas) para a sistema de coordenados projetados. Em seguida, vamos produzir um polÃgono com raio de 20 km no entorno do ponto.

```
# Reprojetar o ponto.
sf_acesso_utm <- st_transform(sf_acesso, crs = 31976)
# PolÃgono com raio de 500 metros no entorno do ponto.
sf_acesso_500m <- st_buffer(sf_acesso_utm, dist=500) |>
  mutate(raio_km = 0.5)
# PolÃgono com raio de 1 km no entorno do ponto.
sf_acesso_1km <- st_buffer(sf_acesso_utm, dist=1000) |>
  mutate(raio_km = 1)
# PolÃgono com raio de 2 km no entorno do ponto.
sf_acesso_2km <- st_buffer(sf_acesso_utm, dist=2000) |>
  mutate(raio_km = 2)
# PolÃgono com raio de 4 km no entorno do ponto.
sf_acesso_4km <- st_buffer(sf_acesso_utm, dist=4000) |>
  mutate(raio_km = 4)
# PolÃgono com raio de 20 km no entorno do ponto.
sf_acesso_20km <- st_buffer(sf_acesso_utm, dist=20000)

acesso_buffers <- bind_rows(sf_acesso_500m, sf_acesso_1km,
                            sf_acesso_2km, sf_acesso_4km)
```

5.4.3 Verificar com mapa de base (OpenStreetMap).

```
#
mapview(sf_acesso_20km) +
  mapview(sf_acesso_utm, color = "black", col.regions = "yellow")
```

5.4.4 Dados: MapBiomas cobertura da terra

Agora vamos olhar cobertura e uso da terra no espaco que preciso (Ã¡rea de estudo). Para isso, vamos utilizar um arquivo de raster do projeto MapBiomas com cobertura de terra ao redor do Garimpo do LourenÃ§o em 1985. Este arquivo no formato raster, tem apenas valores inteiros, em que cada cÃ©lula/pixel representa uma Ã¡rea considerada homogÃªnea, como uso do solo ou tipo de vegetaÃ§Ã£o. Arquivo ".tif" disponÃvel aqui: utm_cover_AP_lorenco_1985.tif

NÃ£o vamos construir mapas, portanto os cores nas visualizaÃ§Ãµes nÃ£o corresponde ao mundo real (por exemplo, verde nÃ£o Ã© floresta). Para visualizar em QGIS preciso baixar um arquivo com a legenda e cores para ColeÃ§Ã£oÂ¨6 (https://mapbiomas.org/codigos-de-legenda) e segue tutoriais: https://www.youtube.com/watch?v=WtyotodHK8E .

Este vez, a entrada de dados espaciais seria atraves a importaÃ§Ã£o de um raster (arquivo de .tif). Lembre-se, para facilitar, os arquivos deve ficar no mesmo diretÃ³rio do seu cÃ³digo (verifique com getwd()). Como nÃ³s jÃ¡ sabemos a sistema de coordenados desejadas, o geoprocessamento da raster foi concluÃdo antes de comeÃ§ar com as anÃ¡lises da paisagem.

```
r1985 <- rast("utm_cover_AP_lorenco_1985.tif")
r1985

#class       : SpatRaster
#dimensions  : 1341, 1341, 1  (nrow, ncol, nlyr)
#resolution  : 29.87713, 29.87713  (x, y)
#extent      : 409829.5, 449894.7, 236241.1, 276306.3  (xmin, xmax, ymin, ymax)
#coord. ref. : SIRGAS 2000 / UTM zone 22N (EPSG:31976)
#source      : utm_cover_AP_lorenco_1985.tif
#name        : classification_1985
#min value   :                    3
#max value   :                   33
```

Ou use o funÃ§Ã£o file.choose(), que faz a busca para arquivos.

```
r1985 <- rast(file.choose())
r1985
```

```
##            used  (Mb) gc trigger  (Mb)  max used   (Mb)
## Ncells  7987540 426.6   15368022 820.8  15368022  820.8
## Vcells 34260577 261.4  108312594 826.4 135390740 1033.0
```

Agora que o arquivo foi importado, podemos visualizÃ¡- lo.

```
# Visualizar para verificar
# Gradiente de cores padrÃ£o nÃ£o corresponde
# ao mundo real (por exemplo verde nÃ£o Ã© floresta)
plot(r1985, type="classes")
plot(sf_acesso_20km, add = TRUE, lty ="dashed", color = "black")
plot(sf_acesso_utm, add = TRUE, cex = 2, pch = 19, color = "black")
```

5.5 Calculo de mÃ©tricas

Vamos olhar alguns exemplos de mÃ©tricas para cada nÃvel da anÃ¡lise:

- landscape (mÃ©tricas para a paisagem como um todo).
- class (mÃ©tricas por classe ou tipo de habiat).
- patch (para a mancha ou fragmento).

Primeiro, pecisamos verificar se o raster estÃ¡ no formato correto.

```
check_landscape(r1985)
```

```
##   layer       crs units   class n_classes OK
## 1     1 projected     m integer         7  v
```

```
#  layer crs    units   class n_classes OK
#  1  projected   m   integer         7  v
```

Tudo certo (veja a coluna do "OK")!

5.5.1 MÃ©tricas para a paisagem

Vamos comeÃ§ar avaliando a Ã¡rea total da paisagem (Ã¡rea) de estudo.

```
area.total <- lsm_l_ta(r1985)
area.total #160264 Hectares
```

```
## # A tibble: 1 x 6
##   layer level     class    id metric   value
##   <int> <chr>     <int> <int> <chr>    <dbl>
## 1     1 landscape    NA    NA ta     160264.
```

Agora vamos ver a distÃ¢ncia total de borda (te= â total edgeâ).

```
##            used  (Mb) gc trigger  (Mb)  max used   (Mb)
## Ncells  7988859 426.7   15368022 820.8  15368022  820.8
## Vcells 34260460 261.4  108312594 826.4 135390740 1033.0
```

```
te <- lsm_l_te(r1985)
te # 547140 metros
```

```
## # A tibble: 1 x 6
##   layer level     class    id metric   value
##   <int> <chr>     <int> <int> <chr>    <dbl>
## 1     1 landscape    NA    NA te     547140.
```

Total de borda mede a configuraÃ§Ã£o da paisagem porque uma paisagem altamente fragmentada terÃ¡ muitas bordas. No entanto, a borda total Ã© uma medida absoluta, dificultando comparaÃ§Ãµes entre paisagens com
Ã¡reas totais diferentes. Mas pode ser aplicado para comparar a configuraÃ§Ã£o na mesma paisagem em anos diferentes.

Agora vamos ver a densidade de Borda (â Edge Densityâ). Densidade de Borda mede a configuraÃ§Ã£o da paisagem porque uma paisagem altamente fragmentada terÃ¡ valores mais altas. "Densidade" Ã© uma medida adequado para comparacoes de paisagens com Ã¡reas totais diferentes.

```
ed <- lsm_l_ed(r1985)
ed #3.41 metros por hectare
```

```
## # A tibble: 1 x 6
##   layer level     class    id metric value
##   <int> <chr>     <int> <int> <chr>  <dbl>
## 1     1 landscape    NA    NA ed      3.41
```

5.5.2 MÃ©tricas para as classes

Area de cada classe em hectares.

```
lsm_c_ca(r1985)
```

```
## # A tibble: 7 x 6
##   layer level class    id metric     value
##   <int> <chr> <int> <int> <chr>      <dbl>
## 1     1 class     3    NA ca     158582.
```

```
## 2     1 class     4    NA ca          1.70
## 3     1 class    11    NA ca          1.79
## 4     1 class    12    NA ca        548.
## 5     1 class    15    NA ca        563.
## 6     1 class    30    NA ca        526.
## 7     1 class    33    NA ca         41.4
```

Como tem varios classes Ã© dificil de interpretar os resultados porque os numeros (3, 4, 11…..) nÃ£o tem uma referncia do mundo real. Para entender os resultados, precisamos acrescentar nomes para os valores. Ou seja incluir uma coluna de legenda com os nomes para cada classe. Para isso precisamos outro arquivo com os nomes.

Baixar a arquivo de legenda: mapbiomas_6_legend.xlsx.

Agora carregar o arquivo com o codigo a seguir.

```
class_nomes <- read_excel(file.choose())
```

Agora rodar de novo, com os resultados juntos com a legenda de cada classe. Nos resultados acima, os valores na coluna "class" sÃ£o as mesmas que tem na coluna "aid" no objeto "class_nomes", onde tambÃ©m tem os nomes . Assim, podemos repetir, mas agora incluindo os nomes para cada valor de class, com base na ligaÃ§Ã£o (join) entre as colunas.

```
# Ã rea de cada classe em hectares, incluindo os nomes para cada classe
lsm_c_ca(r1985) |>
  left_join(class_nomes, by = c("class" = "aid"))

# Numero de fragmentos (manchas)
lsm_c_np(r1985) |>
  left_join(class_nomes, by = c("class" = "aid"))

# Maior numero de manchas em classes de cobertura classificadas como
# pasto (pasture) e formaÃ§Ã£o campestre (grassland).

#  layer level class    id metric value class_description  group_description
#     1 class     3    NA np        28 Forest Formation     Natural forest
#     1 class     4    NA np         2 Savanna Formation    Natural forest
#     1 class    11    NA np         7 Wetlands             Natural non fore
#     1 class    12    NA np       246 Grassland            Natural non fore.
#     1 class    15    NA np       262 Pasture              Farming
#     1 class    30    NA np        35 Mining               Non vegetated
#     1 class    33    NA np        50 River,Lake and Ocean Water
```

5.5.3 MÃ©tricas para as manchas

Vamos calcular o tamanho de cada mancha agora.

```
mancha_area <- lsm_p_area(r1985) # 630 manchas
mancha_area
```

Agora queremos saber o tamanho da maior mancha em cada class, e portanto o tamanho da maior mancha de mineraÃ§Ã£o.

```
mancha_area |>
group_by(class) |>
summarise(max_ha = max(value))
# 30.8 hectares (class 15 = mineraÃ§Ã£o)
```

5.5.4 Quais mÃ©tricas devo escolher?

A decisÃ£o deve ser tomada com base em uma combinaÃ§Ã£o de fatores. Incluindo tais fatores como: base teÃ³rica, consideraÃ§Ãµes estatÃsticas, relevÃncia para o objetivo/hipÃ³tese e a escala e heterogeneidade na paisagem de estudo.

Queremos caracterizar Ã¡reas de mineraÃ§Ã£o na paisagem, e aqui vamos olhar somente uma paisagem, em um momento do tempo. EntÃ£o as mÃ©tricas para a paisagem como todo nÃ£o tem relevÃncia.

Estamos olhando uma classe (mineraÃ§Ã£o), portanto vamos incluir as mÃ©tricas para classes. Alem disso, as mÃ©tricas de paisagem em nÃvel de classe sÃ£o mais eficazes na definiÃ§Ã£o de processos ecolÃ³gicos (Tischendorf, L. Can landscape indices predict ecological processes consistently?. Landscape Ecology 16, 235â "254 (2001). https://doi.org/10.1023/A:1011112719782.).

```
# mÃ©tricas de composiÃ§Ã£o para a paisagem por classes
list_lsm(level = "class", type = "area and edge metric")

# mÃ©tricas de configuraÃ§Ã£o para a paisagem por classes
list_lsm(level = "class", type = "aggregation metric")
```

5.5.5 MÃ©tricas por classe de mineraÃ§Ã£o

Aqui vamos calcular todos as mÃ©tricas por classe (funÃ§Ã£o calculate_lsm()).

```
# mÃ©tricas de composiÃ§Ã£o para a paisagem por classes
metrics_comp <- calculate_lsm(r1985, level = "class", type = "area and edge metric")

# mÃ©tricas de configuraÃ§Ã£o para a paisagem por classes
metrics_config <- calculate_lsm(r1985, level = "class", type = "aggregation metric")
```

E aqui, calcular correlaÃ§Ãµes entre todos as mÃ©tricas por classe (funÃ§Ã£o show_correlation()).

```
show_correlation(data = metrics_comp, method = "pearson", labels = TRUE)
```

```
show_correlation(data = metrics_config, method = "pearson", labels = TRUE)
```

Temos muitos valores e muitas mÃ©tricas. Este se chama um "tiro no escuro", algo cujo resultado se desconhece ou Ã© imprevisÃvel. Isso nÃ£o Ã© recomendado. Para fazer uma escolha melhor (mais robusta), seguindo princÃpios bÃ¡sicos da ciÃªncia, precisamos ler os estudos anteriores (artigos) para obter as mÃ©tricas mais relevantes para nosso objetivo e a hipÃ³tese a ser testada. Com base em os estudos anteriores e os objetivos vamos incluir 8 mÃ©tricas nos resultados.

5.5.6 Exportar as mÃ©tricas

O prÃ³ximo passo Ã© comunicar os resultados obtidos. Para isso precisamos resumir e apresentar as mÃ©tricas selecionadas em tabelas e figuras. Agora jÃ¡ fizemos os cÃ¡lculos, as tabelas e figuras podem ser feitas no R (figuras), tanto quanto em aplicativos diferentes (por exemplo tabelas atraves ["tabelas dinamicas"] no Microsoft Excel ou LibreOffice calc). Mas por isso, primeiramente precisamos exportar os resultados (veja mais exemplos aqui: IntroduÃ§Ã£o ao R import-export.

O arquivo vai sai no mesmo diretÃ³rio do seu cÃ³digo (verifique com getwd()).

```
bind_rows(metrics_comp, metrics_config) -> metricas_1985
write.csv2(metricas_1985, "metricas_lourenco_1985.csv", row.names=FALSE)
```

5.6 Apresentando os resultados

5.6.1 Preparando os resultados

A entrada de dados seria com as mÃ©tricas da paisagem calculados anteriormente.

VocÃªs devem baixar o arquivo de Excel metricas_lourenco_1985.xlsx. Lembre-se, para facilitar, os dados deve ficar no mesmo diretÃ³rio do seu cÃ³digo (verifique com getwd()).

No caso de um arquivo de Excel simples, a importaÃ§Ã£o poderia ser feita atravÃ©s menu de "Import Dataset" na janela/panel "Environment" de Rstudio. Ou com linhas de cÃ³digo:

```
metricas_1985 <- read_excel("metricas_lourenco_1985.xlsx")
metricas_1985

#  layer level class id    metric     value
#   <dbl> <chr> <dbl> <chr> <chr>      <dbl>
#     1 class     3 NA    area_cv  529.
#     1 class     4 NA    area_cv   22.3
#     1 class    11 NA    area_cv   71.2
```

Ou use o funÃ§Ã£o file.choose(), que faz a busca para arquivos.

```
metricas_1985 <- read_excel(file.choose())
metricas_1985
```

```
## # A tibble: 182 x 6
##    layer level class id    metric     value
##    <dbl> <chr> <dbl> <chr> <chr>      <dbl>
##  1     1 class     3 NA    area_cv  529.
##  2     1 class     4 NA    area_cv   22.3
##  3     1 class    11 NA    area_cv   71.2
##  4     1 class    12 NA    area_cv  169.
##  5     1 class    15 NA    area_cv  175.
##  6     1 class    30 NA    area_cv  276.
##  7     1 class    33 NA    area_cv   74.2
##  8     1 class     3 NA    area_mn 5664.
##  9     1 class     4 NA    area_mn    0.848
## 10     1 class    11 NA    area_mn    0.255
## # i 172 more rows
```

Os dados sÃ£o padronizados ("tidy"), mas ainda nÃ£o parece adequados para apresentaÃ§Ã£o em tabelas ou figuras. Temos muitos valores e muitas mÃ©tricas (listadas na coluna "metric"). Com base em os estudos anteriores e os objetivos vamos incluir 8 mÃ©tricas (4 de composiÃ§Ã£o e 4 de configuraÃ§Ã£o).

MÃ©tricas de composiÃ§Ã£o:

- mean patch area (lsm_c_area_mn) Ã rea mÃ©dio das manchas por classe.
- SD patch area (lsm_c_area_sd) Desvio padrÃ£o das Ã¡reas dos manchas por classe.
- class area percentage of landscape (lsm_c_pland) Porcentagem de Ã¡rea na paisagem por classe.
- largest patch index (lsm_c_lpi) Ã ndice de maior mancha (proporÃ§Ã£o da paisagem).

MÃ©tricas de configuraÃ§Ã£o:

- aggregation index (lsm_c_ai) Ã ndice de agregaÃ§Ã£o.
- patch cohesion index (lsm_c_cohesion) Ã ndice de coesÃ£o das manchas.
- number of patches (lsm_c_np) NÃºmero de manchas.
- patch density (lsm_c_pd) Densidade de manchas.

Escolheremos (atraves um filtro) as mÃ©tricas que queremos para obter uma tabela de dados. Mantendo os dados originais, assim sendo para acresentar mais mÃ©tricas nos resultados, preciso somente acrescentar mais no codigo.

```
# Arquivo com os nomes das classes
class_in <- "C:\\Users\\user\\Documents\\Articles\\gis_layers\\gisdata\\inst\\raster\\mapbiomas_cover_l
class_nomes <- read_excel(class_in)

# Especificar mÃ©tricas desejados
```

```r
met_comp <- c("pland", "lpi", "area_mn", "area_sd")
met_conf <- c("ai", "cohesion", "np", "pd")
met_todos <- c(met_comp, met_conf)

# Escholer mÃ©tricas desejados do conjunto completo
metricas_1985 |>
filter(metric %in% met_todos) |>
left_join(class_nomes, by = c("class" = "aid")) -> metricas_nomes
```

5.6.2 Uma tabela versatil

Mas, ainda nÃ£o tem uma coluna com os nomes das mÃ©tricas. Portanto, soluÃ§Ã£o simples Ã© de exportar no formato de .csv e finalizar/editar no Excel / calc.

Outra opÃ§Ã£o que pode facilitar, particularmente quando pode hÃ¡ mudanÃ§as e revisÃµes, Ã© produzir a tabela no R. Aqui vamos repetir no R os passos que vocÃªs conhecem com as ferramentas de Excel (arraste e solte, copiar-colar, filtro, tabela dinÃ¢mica).

5.6.3 ReorganizaÃ§Ã£o

Escolhendo as colunas desejadas (select), reorganizando para as mÃ©tricas ficam nas colunas (pivot_wider) e colocando as colunas novas na sequÃªncia desejada (select).

```
metricas_nomes |>
# Escholer mÃ©tricas desejados do conjunto completo de mÃ©tricas.
dplyr::select(c(type_class, classe_descricao, hexadecimal_code,
metric, value)) |>
# reorganizando
pivot_wider(names_from = metric, values_from = value) -> metricas_tab
```

5.6.4 Uma figura elegante

Uma imagem vale mais que mil palavras. Portanto, grÃ¡ficos/figuras/imagens sÃ£o uma das mais importantes formas de comunicar a ciÃªncia.

Como exemplo ilustrativo, aqui vamos produzir grÃ¡ficos comparando mÃ©tricas de composiÃ§Ã£o e configuraÃ§Ã£o da paisagem ao redor do Garimpo do LourenÃ§o.

Ã uma boa ideia gastar bastante tempo para tornar figuras cientÃficas as mais informativas e atraentes possÃveis. Escusado serÃ¡ dizer que a precisÃ£o empÃrica Ã© primordial. E por isso, o que fica excluÃdo/omitido Ã© tÃ£o importante quanto o que foi incluÃdo. Para ajudar, vocÃª deve se perguntar o seguinte ao criar uma figura: eu apresentaria essa figura em uma apresentaÃ§Ã£o para um grupo de colegas? Eu o apresentaria a um pÃºblico de nÃ£o especialistas? Eu gostaria que essa figura aparecesse em um artigo de notÃcias sobre meu trabalho? Ã claro que todos esses locais exigem diferentes graus de precisÃ£o, complexidade e estÃ©tica, mas uma boa figura deve servir para educar simultaneamente pÃºblicos muito diferentes.

Tabelas versus grÃ¡ficos â A primeira pergunta que vocÃª deve se fazer Ã© se vocÃª pode transformar aquela tabela (chata e feia) em algum tipo de grÃ¡fico. VocÃª realmente precisa dessa tabela no texto principal? VocÃª nÃ£o pode simplesmente traduzir as entradas das cÃ©lulas em um grÃ¡fico de barras/colunas/xy? Se vocÃª pode, vocÃª deve. Quando uma tabela nÃ£o pode ser facilmente traduzida em uma figura, na maioria das vezes a provavelmente pertence Ã s InformaÃ§Ãµes Suplementares/Anexos/ApÃªndices.

5.6.4.1 GrÃ¡fico de barra

Primeiramente, vamos produzir uma grÃ¡fico de barra comparando a proporÃ§Ã£o que cada classe representa na paisagem.

```
# Inclundo cores conforme legenda da Mapbiomas ColeÃ§Ã£o 6
# Legenda nomes ordem alfabetica
classe_cores <- c("Campo Alagado e Ã rea Pantanosa" = "#45C2A5",
"FormaÃ§Ã£o Campestre" = "#B8AF4F",
"FormaÃ§Ã£o Florestal" = "#006400",
"FormaÃ§Ã£o SavÃ¢nica" = "#00ff00",
"MineraÃ§Ã£o" = "#af2a2a",
"Pastagem" = "#FFD966",
"Rio, Lago e Oceano" = "#0000FF")
```

E agora o grafico…….

```
# Grafico de barra basica
metricas_tab |>
mutate(class_prop = pland) |>
ggplot(aes(x = classe_descricao, y = class_prop)) +
geom_col()


# Agora com ajustes
# Agrupando por tipo (natural e antropico)
# Com cores conforme legenda da Mapbiomas ColeÃ§Ã£o 6
# Corrigindo texto dos eixos.
# Mudar posiÃ§Ã£o da leganda para o texto com nomes longas encaixar.
metricas_tab |>
mutate(class_prop = pland) |>
ggplot(aes(x = type_class, y = class_prop,
fill = classe_descricao)) +
scale_fill_manual("classe", values = classe_cores) +
geom_col(position = position_dodge2(width = 1)) +
```

```
coord_flip() +
labs(title = "MapBiomas cobertura da terra",
subtitle = "Entorno do Garimpo do LorenÃ§o 1985",
y = "ProporÃ§Ã£o da paisagem (%)",
x = "") +
theme(legend.position="bottom") +
guides(fill = guide_legend(nrow = 4))
```

Uma imagem vale mais que mil palavras:

Mas existe uma separaÃ§Ã£o grande na faixa de valores e ainda Ã© difÃcil de ver todas as classes. Temos uma distribuiÃ§Ã£o com valores muito mais altos comparada com os outros. extremos. Uma soluÃ§Ã£o seria uma transformaÃ§Ã£o (por exemplo "log"), assim os valores ficarem mais proximos.

5.6.4.2 GrÃ¡fico de boxplot

Agora com uma mÃ©trica de configuraÃ§Ã£o:

```
# MÃ©trica de configuraÃ§Ã£o: Densidade de manchas (coluna "pd").
# Agrupando por tipo (natural e antrÃ³pico)
# Incluindo boxplot indicando tendÃªncia central (mediano)
# Com cores conforme legenda da Mapbiomas ColeÃ§Ã£o 6
# Tamanho dos pontos proporcional o numero de manchas
# Corrigindo texto dos eixos.
# Mudar posiÃ§Ã£o da leganda para o texto com nomes longas encaixar.
metricas_tab |>
ggplot(aes(x = type_class, y = pd)) +
geom_boxplot(colour = "grey50") +
geom_point(aes(size = np, colour = classe_descricao)) +
scale_color_manual("classe", values = classe_cores) +
scale_size(guide = "none") +
coord_flip() +
labs(title = "MapBiomas cobertura da terra",
subtitle = "Entorno do Garimpo do LorenÃ§o 1985",
y = "Densidade de manchas (nÃºmero por 100 hectares)",
x = "") +
theme(legend.position="bottom") +
guides(col = guide_legend(nrow = 4))
```

5.7 ComparaÃ§Ã£o entre anos

Calcular as mÃ©tricas para 3 anos.

```
# metricas desejados
what_metricas <- c("lsm_c_pland", "lsm_c_lpi", "lsm_c_area_mn", "lsm_c_area_sd",
                   "lsm_c_ai", "lsm_c_cohesion", "lsm_c_np", "lsm_c_pd")
# rodar
metricas_anos <- sample_lsm(landscape = mapbiomas_85a20,
                            y = acesso_buffers,
                           plot_id = data.frame(acesso_buffers)[, 'raio_km'],
                             what = what_metricas,
                           edge_depth = 1)

# Organizar dados
# Dados rferentes os buffers
resultados_anos <- acesso_buffers |>
  left_join(metricas_anos |>
              dplyr::mutate(value = round(value,2),
         ano = case_when(layer==1 ~1985,
                         layer==2~2003,
                         layer==3~2020)) |>
              dplyr::select(ano, plot_id, class, metric, value),
            by=c("raio_km"="plot_id"))
# Dados referentes os classes
resultados_anos <- resultados_anos |>
```

```
left_join(class_nomes, by = c("class" = "aid"))
```

Agora grafico de barra com varios anos

```
# It's recommended to use a named vector
# Legenda nomes ordem alfabetica
classe_cores <- c("Campo Alagado e Ã rea Pantanosa" = "#45C2A5",
"FormaÃ§Ã£o Campestre" = "#B8AF4F",
"FormaÃ§Ã£o Florestal" = "#006400",
"FormaÃ§Ã£o SavÃ¢nica" = "#00ff00",
"MineraÃ§Ã£o" = "#af2a2a",
"Pastagem" = "#FFD966",
"Rio, Lago e Oceano" = "#0000FF",
"Outras Lavouras TemporÃ¡rias" = "#e787f8")

resultados_anos |>
  mutate(rcor = paste("#", hexadecimal_code, sep="")) |>
  filter(metric=="pland") |>
  ggplot(aes(x=ano, y=value)) +
  geom_col(position="stack", aes(fill=classe_descricao)) +
  scale_fill_manual(values = classe_cores) +
  facet_grid(raio_km~type_class)
```

Agora com "pland" e densidade de manchas juntos.

```
resultados_anos |>
  mutate(rcor = paste("#", hexadecimal_code, sep="")) |>
```

```r
  filter(metric=="pland") |>
  ggplot(aes(x=ano, y=value)) +
  geom_col(position="stack", aes(fill=classe_descricao)) +
  scale_fill_manual("classe", values = classe_cores) +
  scale_y_continuous(breaks=c(0,50,100)) +
  scale_x_continuous(breaks=c(1985,2003, 2020)) +
  facet_grid(raio_km~type_class) +
  labs(title = "MapBiomas cobertura da terra",
subtitle = "Entorno do Garimpo do LorenÃ§o 1985-2020",
y = "Porcentagem da paisagem",
x = "Ano") +
theme(legend.position="bottom",
      legend.title = element_text(size = 3),
              legend.text  = element_text(size = 3),
              legend.key.size = unit(0.1, "lines")) +
guides(fill = guide_legend(nrow = 4)) -> fig_pland

# Densidade de manchas
resultados_anos |>
  mutate(rcor = paste("#", hexadecimal_code, sep="")) |>
  filter(metric=="pd") |>
ggplot(aes(x = factor(ano), y = value)) +
geom_boxplot(colour = "grey50") +
geom_point(aes(colour = classe_descricao)) +
scale_color_manual("classe", values = classe_cores) +
scale_size(guide = "none") +
facet_grid(raio_km~type_class) +
labs(title = "MapBiomas cobertura da terra",
subtitle = "Entorno do Garimpo do LorenÃ§o 1985-2020",
y = "Densidade de manchas (nÃºmero por 100 hectares)",
x = "Ano") +
theme(legend.position="bottom",
      legend.title = element_text(size = 3),
              legend.text  = element_text(size = 3),
              legend.key.size = unit(0.1, "lines")) +
guides(fill = guide_legend(nrow = 4)) -> fig_pd

grid.arrange(fig_pland, fig_pd, nrow=1)
```

MapBiomas cobertura da terra
Entorno do Garimpo do Lorenço 1985−2
antropic
natural
Porcentagem da paisagem
100
50
0
0.5
1
2
4
1985
2003
2020
Ano
classe
Campo Alagado e Área Pantanosa
Formação Campestre
Formação Florestal
Outras Lavouras Temporárias
Pastagem
Rio, Lago e Oceano

MapBiomas cobertura da terra
Entorno do Garimpo do Lorenço 1985−20
antropic
natural
Densidade de manchas (número por 100 hectares)
6
4
2
0
0.5
1
2
4
1985
2003
2020
Ano
classe
Campo Alagado e Área Pantanosa
Formação Campestre
Formação Florestal
Outras Lavouras Temporárias
Pastagem
Rio, Lago e Oceano

5.8 ComparaÃ§Ã£o entre anos e paisagens

Aqui compararemos as mudanÃ§as na cobertura e uso da terra entre dois garimpos: Garimpo do LourenÃ§o e Garimpo do Vila Nova. Como eles estÃ£o separados por aproximadamente 200 quilÃ´metros em linha reta, iremos processar os dois separadamente e depois combinar os resultados.

Para Vila Nova, vamos incluir buffers com os mesmos raios do que fizemos para Garimpo do LourenÃ§o.

```
# Tabela de dados com coordenados.
acesso_vn <- data.frame(nome = "garimpo do Vila Nova",
           coord_x = -51.735512,
           coord_y = 0.402381)

# Converter para objeto espacial, com sistema de coordenados geogrÃ¡fica.
sf_acesso_vn <- st_as_sf(acesso_vn,
               coords = c("coord_x", "coord_y"),
            crs = 4326)

# Agora fazer os buffers
# Reprojetar o ponto.
sf_acesso_vn_utm <- st_transform(sf_acesso_vn, crs = 31976)
# PolÃgono com raio de 500 metros no entorno do ponto.
sf_acesso_vn_500m <- st_buffer(sf_acesso_vn_utm, dist=500) |>
  mutate(raio_km = 0.5)
# PolÃgono com raio de 1 km no entorno do ponto.
sf_acesso_vn_1km <- st_buffer(sf_acesso_vn_utm, dist=1000) |>
  mutate(raio_km = 1)
# PolÃgono com raio de 2 km no entorno do ponto.
sf_acesso_vn_2km <- st_buffer(sf_acesso_vn_utm, dist=2000) |>
  mutate(raio_km = 2)
# PolÃgono com raio de 4 km no entorno do ponto.
sf_acesso_vn_4km <- st_buffer(sf_acesso_vn_utm, dist=4000) |>
  mutate(raio_km = 4)
# PolÃgono com raio de 20 km no entorno do ponto.
sf_acesso_vn_20km <- st_buffer(sf_acesso_vn_utm,
                               dist=20000) |>
  mutate(raio_km = 20)

acesso_buffers_vn <- bind_rows(sf_acesso_vn_500m,
                               sf_acesso_vn_1km,
                            sf_acesso_vn_2km,
                            sf_acesso_vn_4km)
```

Verificar:

```
mapview(sf_acesso_vn) +
  mapview(sf_acesso_vn_20km)
```

Agora dados de Mapbiomas. CÃ³pias sÃ£o fornecidas no pacote eprdados. Um arquivo para cada garimpo. Cada arquivo Ã© um raster que inclui mÃºltiplas camadas, uma para cada ano.

```
# arquivo no pacote "eprdados"
arquivo_l <- system.file("raster/lorenco_85a22.tif",
                         package = "eprdados")
arquivo_vn <- system.file("raster/vilanova_85a22.tif",
                         package = "eprdados")
# carregar
r85a22_l <- rast(arquivo_l)
r85a22_vn <- rast(arquivo_vn)
```

A expectativa Ã© que se houvesse mudanÃ§as na paisagem (por exemplo, novas classes que poderiam incluir usos humanos, como mineraÃ§Ã£o e Ã¡reas urbanas), entÃ£o mÃ©tricas como a diversidade da paisagem deveriam aumentar significativamente.

Tome cuidado, pois se for executado em grandes paisagens, isso pode levar algum tempo (horas). Para este exemplo, primeiro cortamos o raster para uma extensÃ£o menor de quatro quilÃ´metros usando o buffer que fizemos anteriormente.

```
# Cortar
r85a22_l_c <- crop(r85a22_l, sf_acesso_4km, snap="out")
r85a22_vn_c <- crop(r85a22_vn, sf_acesso_vn_4km, snap="out")
# MÃ©tricas
# lsm_l_joinent = Complexity of a landscape pattern.
# An overall spatio-thematic complexity metric.
# lsm_l_ent = diversity
# (thematic complexity) of landscape classes.
```

```
# lsm_l_condent = geometric intricacy
# (configurational complexity) of a landscape pattern.
# lsm_l_shdi = Shannon diversity
mymetrics <- c("lsm_l_joinent", "lsm_l_ent",
          "lsm_l_condent", "lsm_l_shdi")
# MÃ©tricas para toda a paisagem de 4 quilometros
# acerca do Garimpo do LourenÃ§o.
# LourenÃ§o
metricas_lourenco <- calculate_lsm(r85a22_l_c,
                                   what = mymetrics)
# Vila Nova
metricas_vilanova <- calculate_lsm(r85a22_vn_c,
                                   what = mymetrics)
```

Acabamos de calcular quatro mÃ©tricas de paisagem diferentes ao longo de 38 anos. Assim sendo, os resultados (no novo objeto metricas_lourenco) deve ter 152 linhas (4 mÃ©tricas X 38 anos = 152).

Aora vamos arrumar os dados para facilitar apresentaÃ§Ã£o e communicaÃ§Ã£o dos resutlados:

```
resultados_anos_lourenco <- metricas_lourenco |>
              dplyr::mutate(value = round(value,2),
                            onde = "Garimpo do LourenÃ§o",
                            ano = case_when(layer==1~1985,
                         layer==2~1986,
                         layer==3~1987,
                         layer==4~1988,
                        layer==5~1989,
                        layer==6~1990,
                         layer==7~1991,
                         layer==8~1992,
                         layer==9~1993,
                         layer==10~1994,
                         layer==11~1995,
                         layer==12~1996,
                         layer == 13 ~ 1997,
                         layer == 14 ~ 1998,
                         layer == 15 ~ 1999,
                         layer == 16 ~ 2000,
                         layer == 17 ~ 2001,
                         layer == 18 ~ 2002,
                         layer == 19 ~ 2003,
                         layer == 20 ~ 2004,
                         layer == 21 ~ 2005,
                         layer == 22 ~ 2006,
                         layer == 23 ~ 2007,
                         layer == 24 ~ 2008,
                       layer == 25 ~ 2009,
                        layer == 26 ~ 2010,
                       layer == 27 ~ 2011,
                       layer == 28 ~ 2012,
                       layer == 29 ~ 2013,
                        layer == 30 ~ 2014,
                        layer == 31 ~ 2015,
                         layer == 32 ~ 2016,
                       layer == 33 ~ 2017,
```

```
                        layer == 34 ~ 2018,
                       layer == 35 ~ 2019,
                         layer == 36 ~ 2020,
                        layer == 37 ~ 2021,
                        layer == 38 ~ 2022))
```

E agora para Vila Nova.

```
resultados_anos_vilanova <- metricas_vilanova |>
               dplyr::mutate(value = round(value,2),
                             onde = "Garimpo do Vila Nova",
                             ano = case_when(
                        layer==1 ~ 1985,
                        layer==2 ~ 1986,
                        layer==3 ~ 1987,
                        layer==4 ~ 1988,
                        layer==5 ~ 1989,
                        layer==6 ~ 1990,
                        layer==7 ~ 1991,
                        layer==8 ~ 1992,
                        layer==9 ~ 1993,
                        layer==10 ~ 1994,
                        layer==11 ~ 1995,
                        layer==12 ~ 1996,
                        layer == 13 ~ 1997,
                        layer == 14 ~ 1998,
                        layer == 15 ~ 1999,
                        layer == 16 ~ 2000,
                        layer == 17 ~ 2001,
                        layer == 18 ~ 2002,
                        layer == 19 ~ 2003,
                        layer == 20 ~ 2004,
                        layer == 21 ~ 2005,
                        layer == 22 ~ 2006,
                        layer == 23 ~ 2007,
                        layer == 24 ~ 2008,
                        layer == 25 ~ 2009,
                        layer == 26 ~ 2010,
                        layer == 27 ~ 2011,
                        layer == 28 ~ 2012,
                        layer == 29 ~ 2013,
                        layer == 30 ~ 2014,
                        layer == 31 ~ 2015,
                        layer == 32 ~ 2016,
                        layer == 33 ~ 2017,
                        layer == 34 ~ 2018,
                        layer == 35 ~ 2019,
                        layer == 36 ~ 2020,
                        layer == 37 ~ 2021,
                        layer == 38 ~ 2022
                        )
                         )
```

Apresentar os resultados em um grÃ¡fico.

```
## `geom_smooth()` using formula = 'y ~ s(x, bs = "cs")'
```

Figura 5.1: ComparaÃ§Ã£o da diversidade da paisagem em torno de dois garimpos. CComparaÃ§Ã£o de mudanÃ§as dentro de quatro quilÃ´metros de Garimpo do LourenÃ§o e garimpo do Vila Nova entre 1985 a 2022. ClassificaÃ§Ã£o de cobertura da terra do MapBiomas ColeÃ§Ã£o 8 usada para quantificar a diversidade representado por quatro mÃ©tricas de paisagem (condent: Entropia condicional; ent: Entropia marginal; joinent: Entropia conjunta; shdi: Ã ndice de diversidade de Shannon).

Esses resultados mostram mudanÃ§as na diversidade da paisagem ao longo do tempo que diferem entre os garimpos. Como podemos ver que os valores de diversidade diferem (LourenÃ§o tende a ter maior diversidade desde o inÃcio) seria Ãºtil comparar a dinÃ¢mica calculada como taxas de mudanÃ§as ao longo do tempo. AlÃ©m disso, agora que sabemos que houve mudanÃ§as, faz sentido investigar mais a fundo. Para compreender as mudanÃ§as, Ã© necessÃ¡ria uma anÃ¡lise mais aprofundada que compare as mÃ©tricas de nÃvel de classe em diferentes escalas. Isto permitir-nos-Ã¡ compreender exactamente que tipos de cobertura do solo mudaram e como essas mudanÃ§as estÃ£o relacionadas com a mineraÃ§Ã£o e outros usos, como o desenvolvimento urbano.

5.9 ConclusÃµes e prÃ³ximos passos

Para entender melhor os padores, precisamos:

- Verficar padrÃµes nas outras metricas calculados
- Verficar padrÃµes usando poligonos/pontos dos processos de mineraÃ§Ã£o (dados no SIGMINE https://www.gov.br/anm/pt-br e https://app.anm.gov.br/dadosabertos/SIGMINE/PROCESSOS_MINERARIOS/)

Alem disso, para aumentar a precisÃ£o dos resultados seria relevante buscar complementar os resultados obtidos com:

1. Imagens com uma classificaÃ§Ã£o do satÃ©lite Sentinel-2 com resoluÃ§Ã£o espacial de 10 metros. A ColeÃ§Ã£o BETA do MapBiomas 10 metros inclui mapas anuais de cobertura e uso da terra para o perÃodo de 2016 a 2022 (perÃodo de disponibilidade de imagens do satÃ©lite Sentinel-2): https://brasil.mapbiomas.org/mapbiomas-cobertura-10m/ .

2. Imagens com uma classificaÃ§Ã£o supervisionado usando imagens de Sentinel-2 (exemplo com QGIS Semi-Automatic Classification plugin aqui: https://fromgistors.blogspot.com/2016/09/basic-tutorial-2.html).

Uma forma alternativa para visualizaÃ§Ã£o as mudanÃ§as entre anos seria um diagrama "Sankey"/"Alluvial". Como exemplo, veja figura 3 no artigo "Rapid land use conversion in the Cerrado has affected water transparency in a hotspot of ecotourism, Bonito, Brazil" https://doi.org/10.1177/19400829221127087 .

Figura 5.2: Diagrama Sankey mostrando a mudanÃ§a de uso da terra na bacia do rio Prata entre 2010 (lado esquerdo) e 2020 (lado direto). Fonte: Figura 3. Chiaravalloti et. al. 2022. Tropical Conservation Science. doi:10.1177/19400829221127087

No R pode fazer com o pacote "networkD3" segue tutoriais:

- https://www.displayr.com/sankey-diagrams-r/
- https://epirhandbook.com/en/diagrams-and-charts.html#alluvialsankey-diagrams
- https://rpubs.com/droach/CPP526-codethrough

Part IV

R e RStudio: Codigo com certeza

6 PrÃ©-requisitos

Conteudo copiado, com pequenas alteraÃ§Ãµes e atualizaÃ§Ãµes de CapÃtulo 3 PrÃ©-requisitos, do livro AnÃ¡lises EcolÃ³gicas no R . O livro estÃ¡ licenciado sob a licenÃ§a BY-NC-ND 4.0 : https://creativecommons.org/licenses/by-nc-nd/4.0/.

6.1 IntroduÃ§Ã£o

O objetivo deste capÃtulo Ã© informar como fazer a instalaÃ§Ã£o dos Programas R e RStudio, alÃ©m de descrever os pacotes e dados necessÃ¡rios para reproduzir os exemplos do livro.

6.1.1 VersÃ£o do R

Todas os cÃ³digos, pacotes e anÃ¡lises disponibilizados no livro foram realizados no Programa R versÃ£o 4.3.2 (31/10/2023).

6.2 InstalaÃ§Ã£o do R

Abaixo descrevemos os passos necessÃ¡rios para a instalaÃ§Ã£o do programa R no seu computador. Para Windows, use link https://cran.r-project.org/bin/windows/base/ e vai para passo 6 :

1. Para comeÃ§armos a trabalhar com o R Ã© necessÃ¡rio baixÃ¡-lo na pÃ¡gina do R Project. EntÃ£o, acesse esse site http://www.r-project.org
2. Clique no link **download R**
3. Na pÃ¡gina *CRAN Mirros (Comprehensive R Archive Network)*, escolha "0-Cloud" ou uma das pÃ¡ginas espelho do Brasil mais prÃ³xima de vocÃª para baixar o programa
4. Escolha agora o sistema operacional do seu computador (passos adicionais existem para diferentes distribuiÃ§Ãµes Linux ou MacOS). Aqui faremos o exemplo com o Windows
5. Clique em **base** para finalmente chegar Ã pÃ¡gina de download com a versÃ£o mais recente do R
6. Clique no arquivo **Download R (versÃ£o mais recente) for Windows** que serÃ¡ instalado no seu computador
7. Abra o arquivo que foi baixado no seu computador e siga os passos indicados para finalizar a instalaÃ§Ã£o do programa R

Importante
Para o Sistema Operacional (SO) Windows, alguns pacotes, por exemplo `eprdados`, sÃ£o dependentes da instalaÃ§Ã£o separada do Rtools. Ou seja antes de instalar `eprdados` vocÃª deve instalar Rtools. Da mesma forma, GNU/Linux e MacOS tambÃ©m possuem dependÃªncias de outras bibliotecas para pacotes especÃficos, mas que nÃ£o abordaremos aqui. Essas informaÃ§Ãµes de dependÃªncia geralmente sÃ£o retornadas como erros e vocÃª pode procurar ajuda em fÃ³runs especÃficos.

6.3 InstalaÃ§Ã£o do Rtools

Abaixo descrevemos os passos necessÃ¡rios para a instalaÃ§Ã£o do Rtools no seu computador:

1. Para fazer o download do Rtools, acessamos o site https://cran.rstudio.com/bin/windows/Rtools/ ,
2. clique no link com o mesmo nÃºmero de versÃ£o de R.
   Por exemplo, com R versÃ£o 4.3.2 vocÃª precisa RTools 43.
3. Clique em **Rtools43 installer** ,
4. Abra o arquivo que foi baixado no seu computador e siga os passos indicados para finalizar a instalaÃ§Ã£o do programa RTools.
   Recomenda-se usar os padrÃµes, incluindo o local de instalaÃ§Ã£o padrÃ£o "C:/rtools43".

6.4 InstalaÃ§Ã£o do RStudio

O RStudio possui algumas caracterÃsticas que o tornam popular: vÃ¡rias janelas de visualizaÃ§Ã£o, marcaÃ§Ã£o e preenchimento automÃ¡tico do script, integraÃ§Ã£o com controle de versÃ£o, dentre outras

Fonte das figuras:
imagem computador https://pt.wikipedia.org/wiki/Computador_pessoal
imagem da lupa https://openclipart.org/detail/185356/magnifier

Figura 6.1: Esquema ilustrativo demonstrando os passos necessÃ¡rios para instalaÃ§Ã£o do programa R no computador.

funcionalidades.

Abaixo descrevemos os passos necessÃ¡rios para a instalaÃ§Ã£o do RStudio no seu computador:

1. Para fazer o download do RStudio, acessamos o site https://posit.co/download/rstudio-desktop/ ,
2. Clique em **download...**,
3. Abra o arquivo que foi baixado no seu computador e siga os passos indicados para finalizar a instalaÃ§Ã£o do programa RStudio.

6.5 Pacotes

Descrevemos no CapÃtulo 7 o que sÃ£o e como instalar os pacotes para realizar as anÃ¡lises estatÃsticas no R.

Importante
Criamos o pacote `eprdados` que contÃ©m todas as informaÃ§Ãµes e dados utilizados neste livro. Assim, recomendamos que vocÃª instale e carregue este pacote no inÃcio de cada capÃtulo para ter acesso aos dados necessÃ¡rios para executar as funÃ§Ãµes no R.

Para a instalaÃ§Ã£o do pacote `eprdados` no macOS, vocÃª precisarÃ¡ ter instalado o programa XCode que pode ser baixado aqui. Este programa Ã© disponibilizado gratuitamente pela Apple e Ã© necessÃ¡rio para compilar quaisquer programas distribuÃdos em cÃ³digo fonte (ou seja, sem um binÃ¡rio). ApÃ³s instalar esse programa e o pacote `devtools`, vocÃª poderÃ¡ instalar o eprdados utilizando as instruÃ§Ãµes abaixo.

Abaixo, listamos todos os pacotes utilizados no livro. VocÃª pode instalar os pacotes agora ou esperar para instalÃ¡-los quando ler o CapÃtulo 7 e entender o que sÃ£o as funÃ§Ãµes `install.packages()`, `library()` e `remotes::install_github()`. Para fazer a instalaÃ§Ã£o, vocÃª vai precisar estar conectado Ã internet.

```
install.packages(c("ade4", "adespatial", "ape", "bbmle", "betapart", "BiodiversityR", "car", "cati", "da
```

Fonte das figuras: imagem computador https://pt.wikipedia.org/wiki/Computador_pessoal

Figura 6.2: Esquema ilustrativo demonstrando os passos necessÃ¡rios para instalaÃ§Ã£o do programa RStudio no computador.

6.5.1 Pacotes do GitHUb

Diferente dos pacotes anteriores que sÃ£o baixados do CRAN, alguns pacotes sÃ£o baixados do GitHub dos pesquisadores responsÃ¡veis pelos pacotes. GitHub Ã© um repositÃ³rio remoto de cÃ³digos que permite controle de versÃ£o, muito utilizado por desenvolvedores e programadores. Nestes casos, precisamos carregar o pacote `remotes` para acessar a funÃ§Ã£o `remotes::install_github()`. Durante as instalaÃ§Ãµes destes pacotes, algumas vezes o R irÃ¡ pedir para vocÃª digitar um nÃºmero indicando os pacotes que vocÃª deseja fazer update. Neste caso, digite 1 para indicar que ele deve atualizar os pacotes dependentes antes de instalar os pacotes requeridos.

```
library(remotes)
remotes::install_github("darrennorris/eprdados")
remotes::install_github("ropensci/rnaturalearthhires")
```

6.6 Dados

A maioria dos exemplos do livro utilizam dados reais extraÃdos de artigos cientÃficos que jÃ¡ foram publicados ou dados que foram coletados por um dos autores deste livro. Todos os dados, publicados ou simulados, estÃ£o disponÃveis no pacote `eprdados`. AlÃ©m disso, em cada capÃtulo fazemos uma breve descriÃ§Ã£o dos dados para facilitar a compreensÃ£o sobre como essas variÃ¡veis estÃ£o relacionadas com as perguntas do exemplo.

7 IntroduÃ§Ã£o ao R

Conteudo copiado, com pequenas alteraÃ§Ãµes e atualizaÃ§Ãµes de CapÃtulo 4 IntroduÃ§Ã£o ao R, do livro AnÃ¡lises EcolÃ³gicas no R . O livro estÃ¡ licenciado sob a licenÃ§a BY-NC-ND 4.0 : https://creativecommons.org/licenses/by-nc-nd/4.0/.

PrÃ©-requisitos do capÃtulo

Pacotes e dados que serÃ£o utilizados neste capÃtulo.

```
## Pacotes
library(palmerpenguins)

## Dados necessÃ¡rios
pinguins <- palmerpenguins::penguins
```

7.1 ContextualizaÃ§Ã£o

O objetivo deste capÃtulo Ã© apresentar os aspectos bÃ¡sicos da linguagem R para a realizaÃ§Ã£o dos principais passos para a manipulaÃ§Ã£o, visualizaÃ§Ã£o e anÃ¡lise de dados. Abordaremos aqui as questÃµes bÃ¡sicas sobre a linguagem R, como: i) R e RStudio, ii) funcionamento da linguagem, iii) estrutura e manipulaÃ§Ã£o de objetos, iv) exercÃcios e v) principais livros e material para se aprofundar nos seus estudos.

Todo processo de aprendizagem torna-se mais efetivo quando a teoria Ã© combinada com a prÃ¡tica. Assim, recomendamos fortemente que vocÃª, leitor(a) acompanhe os cÃ³digos e exercÃcios deste livro, ao mesmo tempo que os executa em seu computador e nÃ£o sÃ³ os leia passivamente. AlÃ©m disso, se vocÃª tiver seus prÃ³prios dados Ã© muito importante tentar executar e/ou replicar as anÃ¡lises e/ou grÃ¡ficos. Por motivos de espaÃ§o, nÃ£o abordaremos todas as questÃµes relacionadas ao uso da linguagem R neste capÃtulo. Logo, aconselhamos que vocÃª consulte o material sugerido no final do capÃtulo para se aprofundar.

Este capÃtulo, na maioria das vezes, pode desestimular as pessoas que estÃ£o iniciando, uma vez que o mesmo nÃ£o apresenta os cÃ³digos para realizar as anÃ¡lises estatÃsticas. Contudo, ele Ã© essencial para o entendimento e interpretaÃ§Ã£o do que estÃ¡ sendo informado nas linhas de cÃ³digo, alÃ©m de facilitar a manipulaÃ§Ã£o dos dados antes de realizar as anÃ¡lises estatÃsticas. VocÃª perceberÃ¡ que nÃ£o usarÃ¡ este capÃtulo para fazer as anÃ¡lises, mas voltarÃ¡ aqui diversas vezes para relembrar qual Ã© o cÃ³digo ou o que significa determinada expressÃ£o ou funÃ§Ã£o usada nos prÃ³ximos capÃtulos.

7.2 R e RStudio

Com o R Ã© possÃvel manipular, analisar e visualizar dados, alÃ©m de escrever desde pequenas linhas de cÃ³digos atÃ© programas inteiros. O R Ã© a versÃ£o em cÃ³digo aberto de uma linguagem de programaÃ§Ã£o chamada de S, criada por John M. Chambers (Stanford University, CA, EUA) nos anos 1980 no Bell Labs. No final dos anos 1990, Robert Gentleman e Ross Ihaka (ambos da Universidade de Auckland, Nova ZelÃ¢ndia), iniciaram o desenvolvimento da versÃ£o livre da linguagem S, a linguagem R, com o seguinte histÃ³rico: Desenvolvimento (1997-2000), VersÃ£o 1 (2000-2004), VersÃ£o 2 (2004-2013), VersÃ£o 3 (2013-2020) e VersÃ£o 4 (2020). Atualmente a linguagem R Ã© mantida por uma rede de colaboradores denominada *R Core Team*. A origem do nome R Ã© desconhecida, mas reza a lenda que ao lanÃ§arem o nome da linguagem os autores se valeram da letra que vinha antes do S, uma vez que a linguagem R foi baseada nela e utilizaram a letra "R". Outra histÃ³ria conta que pelo fato do nome dos dois autores iniciarem por "R", batizaram a linguagem com essa letra, vai saber.

Um aspecto digno de nota Ã© que a linguagem R Ã© uma linguagem de programaÃ§Ã£o interpretada, assim como o Python. Isso a faz ser mais fÃ¡cil de ser utilizada, pois processa linhas de cÃ³digo e as transforma em linguagem de mÃ¡quina (cÃ³digo binÃ¡rio que o computador efetivamente lÃª), apesar desse fato diminuir a velocidade de processamento.

Para comeÃ§armos a trabalhar com o R Ã© necessÃ¡rio baixÃ¡-lo na pÃ¡gina do **R Project**. Os detalhes de instalaÃ§Ã£o sÃ£o apresentados no CapÃtulo 6. Reserve um tempo para explorar esta pÃ¡gina do R-Project (https://www.r-project.org/). Existem vÃ¡rios livros dedicados a diversos assuntos baseados no R. AlÃ©m disso, estÃ£o disponÃveis manuais em diversas lÃnguas para serem baixados gratuitamente.

Como o R Ã© um software livre, nÃ£o existe a possibilidade de o usuÃ¡rio entrar em contato com um serviÃ§o de suporte de usuÃ¡rios. Ao invÃ©s disso, existem vÃ¡rias listas de e-mails que fornecem suporte Ã comunidade de usuÃ¡rios. NÃ³s, particularmente, recomendamos o ingresso nas seguintes listas: R-help, R-sig-ecolog, R-br e discourse.curso-r. Os dois Ãºltimos grupos reÃºnem pessoas usuÃ¡rias brasileiras do programa R.

Apesar de podermos utilizar o R com o IDE (Ambiente de Desenvolvimento Integrado - *Integrated Development Environment*) RGui que vem com a instalaÃ§Ã£o da linguagem R para usuÃ¡rios Windows ou no prÃ³prio terminal para usuÃ¡rios Linux e MacOS, existem alguns IDEs especÃficos para facilitar nosso uso dessa linguagem.

Figura 7.1: Interface do RGui. Os nÃºmeros indicam: (1) R Script, (2) R Console, e (3) R Graphics.

Dessa forma, nÃ³s utilizamos o IDE RStudio, e assumimos que vocÃª que estÃ¡ lendo farÃ¡ o mesmo.

O RStudio permite diversas personalizaÃ§Ãµes, grande parte delas contidas em `Tools > Global options`. Incentivamos as leitoras e leitores a "fuÃ§ar" com certa dose de cuidado, nas opÃ§Ãµes para personalizaÃ§Ã£o. Dentre essas mudanÃ§as, destacamos trÃªs:

1. `Tools > Global options > Appearance > Editor theme`: para escolher um tema para seu RStudio
2. `Tools > Global options > Code > [X] Soft-wrap R source files`: com essa opÃ§Ã£o habilitada, quando escrevemos comentÃ¡rios longos ou mudamos a largura da janela que estamos trabalhando, todo o texto e o cÃ³digo se ajustam a janela automaticamente
3. `Tools > Global options > Code > Display > [X Show Margis] e Margin column (80)`: com essa opÃ§Ã£o habilitada e para esse valor (80), uma linha vertical irÃ¡ aparecer no script marcando 80 caracteres, um comprimento mÃ¡ximo recomendado para padronizaÃ§Ã£o dos scripts

Importante
Para evitar possÃveis erros Ã© importante instalar primeiro o software da linguagem R e depois o IDE RStudio.

O RStudio permite tambÃ©m trabalhar com projetos. **Projeto do RStudio** Ã© uma forma de organizar os arquivos de scripts e dados dentro de um diretÃ³rio, facilitando o compartilhamento de fluxo de anÃ¡lises

de dados e aumentando assim a reprodutibilidade. Podemos criar um Projeto do RStudio indo em `File > New Project` ou no Ãcone de cubo azul escuro que possui um R dentro com um um cÃrculo verde com um sinal de `+` na parte superior esquerda ou ainda no canto superior direito que possui cubo azul escrito `Project` que serve para gerenciar os projetos e depois em `New Project`. Depois de escolher uma dessas opÃ§Ãµes, uma janela se abrirÃ¡ onde escolhemos uma das trÃªs opÃ§Ãµes: i) New Directory (para criar um diretÃ³rio novo com diversas opÃ§Ãµes), ii) Existing Directory (para escolher um diretÃ³rio jÃ¡ existente) e iii) Version Control (para criar um projeto que serÃ¡ versionado pelo git ou Subversion).

7.3 Funcionamento da linguagem R

Nesta seÃ§Ã£o, veremos os principais conceitos para entender como a linguagem R funciona ou como geralmente utilizamos o IDE RStudio no dia a dia, para executar nossas rotinas utilizando a linguagem R. Veremos entÃ£o: i) console, ii) script, iii) operadores, iv) objetos, v) funÃ§Ãµes, vi) pacotes, vii) ajuda (*help*), viii) ambiente (*environment/workspace*), ix) citaÃ§Ãµes e x) principais erros.

Antes de iniciarmos o uso do R pelo RStudio Ã© fundamental entendermos alguns pontos sobre as janelas e o funcionamento delas no RStudio.

Figura 7.2: Interface do RStudio. Os nÃºmeros indicam: (1) janela com abas de Script, R Markdown, dentre outras; (2) janela com abas de Console, Terminal e Jobs; (3) janela com abas de Environment, History, Conections e Tutorial; e (4) janela com abas de Files, Plots, Packages, Help e Viewer.

Detalhando algumas dessas janelas e abas, temos:

- **Console**: painel onde os cÃ³digos sÃ£o rodados e vemos as saÃdas
- **Editor/Script**: painel onde escrevemos nossos cÃ³digos em R, R Markdown ou outro formato
- **Environment**: painel com todos os objetos criados na sessÃ£o
- **History**: painel com o histÃ³rico dos cÃ³digos rodados
- **Files**: painel que mostra os arquivos no diretÃ³rio de trabalho
- **Plots**: painel onde os grÃ¡ficos sÃ£o apresentados
- **Packages**: painel que lista os pacotes
- **Help**: painel onde a documentaÃ§Ã£o das funÃ§Ãµes Ã© exibida

No RStudio, alguns atalhos sÃ£o fundamentais para aumentar nossa produtividade:

- **F1**: abre o painel de *Help* quando digitado em cima do nome de uma funÃ§Ã£o
- **Ctrl + Enter**: roda a linha de cÃ³digo selecionada no script

- **Ctrl + Shift + N**: abre um novo script
- **Ctrl + S**: salva um script
- **Ctrl + Z**: desfaz uma operaÃ§Ã£o
- **Ctrl + Shift + Z**: refaz uma operaÃ§Ã£o
- **Alt + -**: insere um sinal de atribuiÃ§Ã£o (<-)
- **Ctrl + Shift + M**: insere um operador pipe (%>%)
- **Ctrl + Shift + C**: comenta uma linha no script - insere um (#)
- **Ctrl + I**: indenta (recuo inicial das linhas) as linhas
- **Ctrl + Shift + A**: reformata o cÃ³digo
- **Ctrl + Shift + R**: insere uma sessÃ£o (# ——————————-)
- **Ctrl + Shift + H**: abre uma janela para selecionar o diretÃ³rio de trabalho
- **Ctrl + Shift + F10**: reinicia o console
- **Ctrl + L**: limpa os cÃ³digos do console
- **Alt + Shift + K**: abre uma janela com todos os atalhos disponÃveis

7.3.1 Console

O console Ã© onde a versÃ£o da linguagem R instalada Ã© carregada para executar os cÃ³digos da linguagem R (Figura 7.2 janela 2). Na janela do console aparecerÃ¡ o sÃmbolo `>`, seguido de uma barra vertical `|` que fica piscando (cursor), onde digitamos ou enviamos nossos cÃ³digos do script. Podemos fazer um pequeno exercÃcio: vamos digitar `10 + 2`, seguido da tecla `Enter` para que essa operaÃ§Ã£o seja executada.

```
10 + 2
```

```
## [1] 12
```

O resultado retorna o valor `12`, precedido de um valor entre colchetes. Esses colchetes demonstram a posiÃ§Ã£o do elemento numa sequÃªncia de valores. Se fizermos essa outra operaÃ§Ã£o `1:42`, o R vai criar uma sequÃªncia unitÃ¡ria de valores de 1 a 42. A depender da largura da janela do console, vai aparecer um nÃºmero diferente entre colchetes indicando sua posiÃ§Ã£o na sequÃªncia: antes do nÃºmero 1 vai aparecer o `[1]`, depois quando a sequÃªncia for quebrada, vai aparecer o nÃºmero correspondente da posiÃ§Ã£o do elemento, por exemplo, `[37]`.

```
1:42
```

```
##  [1]  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26
## [27] 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42
```

Podemos ver o histÃ³rico dos cÃ³digos executados no console na aba **History** (Figura 7.2 janela 3).

7.3.2 Scripts

Scripts sÃ£o arquivos de texto simples, criados com a extensÃ£o (terminaÃ§Ã£o) `.R` (Figura 7.2 janela 1). Para criar um script, basta ir em `File > New File > R Script`, ou clicar no Ãcone com uma folha branca e um cÃrculo verde com um sinal de `+`, logo abaixo de `File`, ou ainda usando o atalho `Ctrl + Shift + N`.

Uma vez escrito os cÃ³digos no script podemos rodar esses cÃ³digos de duas formas: i) todo o script de uma vez, clicando em *Source* (que fina no canto superior direito da aba script) ou usando o atalho `Ctrl + Shift + Enter`; ou ii) apenas a linha onde o cursor estiver posicionado, independentemente de sua posiÃ§Ã£o naquela linha, clicando em **Run** ou usando o atalho `Ctrl + Enter`.

Devemos sempre salvar nossos scripts, tomando por via de regra: primeiro criar o arquivo e depois ir salvando nesse mesmo arquivo a cada passo de desenvolvimento das anÃ¡lises (nÃ£o Ã© raro o RStudio fechar sozinho e vocÃª perder algum tempo de trabalho). HÃ¡ diversos motivos para se criar um script: continuar o desenvolvimento desse script em outro momento ou em outro computador, preservar trabalhos passados, ou ainda compartilhar seus cÃ³digos com outras pessoas. Para criar ou salvar um script basta ir em `File > Save`, escolher um diretÃ³rio e nome para o script e salvÃ¡-lo. Podemos ainda utilizar o atalho `Ctrl + S`.

Em relaÃ§Ã£o aos scripts, hÃ¡ ainda os comentÃ¡rios, representados pelos sÃmbolos `#` (hash), `#'` (hash-linha) e `#>` (hash-maior). A diferenÃ§a entre eles Ã© que para o segundo e terceiro, quando pressionamos a tecla `Enter` o comentÃ¡rio `#'` e `#>` sÃ£o inseridos automaticamente na linha seguinte. Linhas de cÃ³digos do script contendo comentÃ¡rios em seu inÃcio nÃ£o sÃ£o lidos pelo console do R. Se o comentÃ¡rio estiver no final da linha, essa linha de cÃ³digo ainda serÃ¡ lida. Os comentÃ¡rios sÃ£o utilizados geralmente para: i) descrever informaÃ§Ãµes sobre dados ou funÃ§Ãµes e/ou ii) suprimir linhas de cÃ³digo.

Ã interessante ter no inÃcio de cada script um cabeÃ§alho identificando o objetivo ou anÃ¡lise, autor e data para facilitar o compartilhamento e reprodutibilidade. Os comentÃ¡rios podem ser inseridos ou retirados das linhas com o atalho `Ctrl + Shift + C`.

```
#' ---
#' TÃtulo: CapÃtulo 04 - IntroduÃ§Ã£o ao R
#' Autor: MaurÃcio Vancine
#' Data: 11-11-2021
#' ---
```

AlÃ©m disso, podemos usar comentÃ¡rios para adicionar informaÃ§Ãµes sobre os cÃ³digos.

```
## ComentÃ¡rios
# O R nÃ£o lÃª a linha do cÃ³digo depois do # (hash).
42 # Essas palavras nÃ£o sÃ£o executadas, apenas o 42, a resposta para questÃ£o fundamental da vida, o
```

```
## [1] 42
```

Por fim, outro ponto fundamental Ã© ter boas prÃ¡ticas de estilo de cÃ³digo. Quanto mais organizado e padronizado estiver seus scripts, mais fÃ¡cil de entendÃª-los e de procurar possÃveis erros. Existem dois guias de boas prÃ¡ticas para adequar seus scripts: Hadley Wickham e Google. Para simplificar a vida temos o pacote styler (https://styler.r-lib.org/), que serve para adequar o codigo.

Ainda em relaÃ§Ã£o aos scripts, temos os *Code Snippets* (Fragmentos de cÃ³digo), que sÃ£o macros de texto usadas para inserir rapidamente fragmentos comuns de cÃ³digo. Por exemplo, o snippet `fun` insere uma definiÃ§Ã£o de funÃ§Ã£o R. Para mais detalhes, ler o artigo do RStudio Code Snippets.

```
# fun {snippet}
fun
name <- function(variables) {

}
```

Uma aplicaÃ§Ã£o bem interessante dos *Code Snippets* no script Ã© o `ts`. Basta digitar esse cÃ³digo e em seguida pressionar a tecla `Tab` para inserir rapidamente a data e horÃ¡rio atuais no script em forma de comentÃ¡rio.

```
# ts {snippet}
# Thu Nov 11 18:19:26 2021 -------------------------------
```

7.3.3 Operadores

No R, podemos agrupar os operadores em cinco tipos: aritmÃ©ticos, relacionais, lÃ³gicos, atribuiÃ§Ã£o e diversos. Grande parte deles sÃ£o descritos na Tabela "Principais operadores no R.".

Como exemplo, podemos fazer operaÃ§Ãµes simples usando os operadores aritmÃ©ticos.

```
## OperaÃ§Ãµes aritmÃ©ticas
10 + 2 # adiÃ§Ã£o
```

```
## [1] 12
```

Table 1: Principais operadores no R.

Operador	Tipo	DescriÃ§Ã£o
+	AritmÃ©tico	AdiÃ§Ã£o
-	AritmÃ©tico	SubtraÃ§Ã£o
*	AritmÃ©tico	MultiplicaÃ§Ã£o
/	AritmÃ©tico	DivisÃ£o
%%	AritmÃ©tico	Resto da divisÃ£o
%/%	AritmÃ©tico	DivisÃ£o inteira
^ ou **	AritmÃ©tico	Expoente
>	Relacional	Maior
<	Relacional	Menor
>=	Relacional	Maior ou igual
<=	Relacional	Menor ou igual
==	Relacional	Igualdade
!=	Relacional	DiferenÃ§a
!	LÃ³gico	LÃ³gico NÃ O
&	LÃ³gico	LÃ³gico elementar E
|	LÃ³gico	LÃ³gico elementar OU
&&	LÃ³gico	LÃ³gico E
||	LÃ³gico	LÃ³gico OU
<- ou =	AtribuiÃ§Ã£o	AtribuiÃ§Ã£o Ã esquerda
«-	AtribuiÃ§Ã£o	Super atribuiÃ§Ã£o Ã esquerda
->	AtribuiÃ§Ã£o	AtribuiÃ§Ã£o Ã direita
-»	AtribuiÃ§Ã£o	Super atribuiÃ§Ã£o Ã direita
:	Diversos	SequÃªncia unitÃ¡ria
%in%	Diversos	Elementos que pertencem a um vetor
%*%	Diversos	Multiplicar matriz com sua transposta
%>%	Diversos	Pipe (pacote magrittr)
|>	Diversos	Pipe (R base nativo)
%–%	Diversos	Intervalo de datas (pacote lubridate)

```
10 * 2 # multiplicaÃ§Ã£o
```

```
## [1] 20
```

Precisamos ficar atentos Ã prioridade dos operadores aritmÃ©ticos:

> **PRIORITÃ RIO () > ^ > * ou / > + ou - NÃ O PRIORITÃ RIO**

Veja no exemplo abaixo como o uso dos parÃªnteses muda o resultado.

```
## Sem especificar a ordem
# Segue a ordem dos operadores.
1 * 2 + 2 / 2 ^ 2
```

```
## [1] 2.5
```

```
## Especificando a ordem
# Segue a ordem dos parenteses.
((1 * 2) + (2 / 2)) ^ 2
```

```
## [1] 9
```

7.3.4 Objetos

Objetos sÃ£o palavras Ã s quais sÃ£o atribuÃdos dados. A atribuiÃ§Ã£o possibilita a manipulaÃ§Ã£o de dados ou armazenamento dos resultados de anÃ¡lises. Utilizaremos os sÃmbolos `<` (menor), seguido de `-` (menos), sem espaÃ§o, dessa forma `<-`. TambÃ©m podemos utilizar o sÃmbolo de igual (`=`), mas nÃ£o recomendamos, por nÃ£o fazer parte das boas prÃ¡ticas de escrita de cÃ³digos em R. Podemos inserir essa combinaÃ§Ã£o de sÃmbolos com o atalho `Alt + -`. Para demonstrar, vamos atribuir o valor `10` Ã palavra `obj_10`, e chamar esse objeto novamente para verificar seu conteÃºdo.

```
## AtribuiÃ§Ã£o - sÃmbolo (<-)
obj_10 <- 10
obj_10
```

```
## [1] 10
```

Importante
Recomendamos sempre verificar o conteÃºdo dos objetos chamando-os novamente para confirmar se a atribuiÃ§Ã£o foi realizada corretamente e se o conteÃºdo corresponde Ã operaÃ§Ã£o realizada.

Todos os objetos criados numa sessÃ£o do R ficam listados na aba **Environment** . AlÃ©m disso, o RStudio possui a funÃ§Ã£o *autocomplete*, ou seja, podemos digitar as primeiras letras de um objeto (ou funÃ§Ã£o) e em seguida apertar `Tab` para que o RStudio liste tudo que comeÃ§ar com essas letras.

Dois pontos importantes sobre atribuiÃ§Ãµes: primeiro, o R sobrescreve os valores dos objetos com o mesmo nome, deixando o objeto com o valor da Ãºltima atribuiÃ§Ã£o.

```
## Sobrescreve o valor dos objetos
obj <- 100
obj
```

```
## [1] 100
```

```
## O objeto 'obj' agora vale 2
obj <- 2
obj
```

```
## [1] 2
```

Segundo, o R tem limitaÃ§Ãµes ao nomear objetos:

- nome de objetos sÃ³ podem comeÃ§ar por letras (`a-z` ou `A-Z`) ou pontos (`.`)
- nome de objetos sÃ³ podem conter letras (`a-z` ou `A-Z`), nÃºmeros (`0-9`), underscores (`_`) ou pontos (`.`)
- R Ã© *case-sensitive*, i.e., ele reconhece letras maiÃºsculas como diferentes de letras minÃºsculas. Assim, um objeto chamado "resposta" Ã© diferente do objeto "RESPOSTA"
- devemos evitar acentos ou cedilha (`ÃS`) para facilitar a memorizaÃ§Ã£o dos objetos e tambÃ©m para evitar erros de codificaÃ§Ã£o (*encoding*) de caracteres
- nomes de objetos nÃ£o podem ser iguais a nomes especiais, reservados para programaÃ§Ã£o (`break`, `else`, `FALSE`, `for`, `function`, `if`, `Inf`, `NA`, `NaN`, `next`, `repeat`, `return`, `TRUE`, `while`)

Podemos ainda utilizar objetos para fazer operaÃ§Ãµes e criar objetos. Isso pode parecer um pouco confuso para os iniciantes, mas Ã© fundamental aprender essa lÃ³gica para passar para os prÃ³ximos passos.

```
## Definir dois objetos
va1 <- 10
va2 <- 2

## OperaÃ§Ãµes com objetos e atribuicÃ£o
adi <- va1 + va2
adi
```

```
## [1] 12
```

7.3.5 FunÃ§Ãµes

FunÃ§Ãµes sÃ£o cÃ³digos preparados para realizar uma tarefa especÃfica de modo simples. Outra forma de entender uma funÃ§Ã£o Ã©: cÃ³digos que realizam operaÃ§Ãµes em argumentos. Devemos retomar ao conceito do ensino mÃ©dio de funÃ§Ãµes: os dados de entrada sÃ£o argumentos e a funÃ§Ã£o realizarÃ¡ alguma operaÃ§Ã£o para modificar esses dados de entrada. A estrutura de uma funÃ§Ã£o Ã© muito similar Ã sintaxe usada em planilhas eletrÃ´nicas, sendo composta por:

nome_da_funÃ§Ã£o(argumento1, argumento2, ...)

1. **Nome da funÃ§Ã£o**: remete ao que ela faz
2. **ParÃªnteses**: limitam a funÃ§Ã£o
3. **Argumentos**: valores, parÃ¢metros ou expressÃµes onde a funÃ§Ã£o atuarÃ¡
4. **VÃrgulas**: separam os argumentos

Os argumentos de uma funÃ§Ã£o podem ser de dois tipos:

1. **Valores ou objetos**: a funÃ§Ã£o alterarÃ¡ os valores em si ou os valores atribuÃdos aos objetos
2. **ParÃ¢metros**: valores fixos que informam um mÃ©todo ou a realizaÃ§Ã£o de uma operaÃ§Ã£o. Informa-se o nome desse argumento, seguido de "=" e um nÃºmero, texto ou TRUE ou FALSE

Alguns exemplos de argumentos como valores ou objetos.

```
## FunÃ§Ãµes - argumentos como valores
sum(10, 2)
```

```
## [1] 12
```

```
## FunÃ§Ãµes - argumentos como objetos
sum(va1, va2)
```

```
## [1] 12
```

Vamos ver agora alguns exemplos de argumentos usados como parÃ¢metros. Note que apesar do valor do argumento ser o mesmo (10), seu efeito no resultado da funÃ§Ã£o `rep()` muda drasticamente. Aqui tambÃ©m Ã© importante destacar um ponto: i) podemos informar os argumentos sequencialmente, sem explicitar seus nomes, ou ii) independente da ordem, mas explicitando seus nomes. Entretanto, como no exemplo abaixo, devemos informar o nome do argumento (i.e., parÃ¢metro), para que seu efeito seja o que desejamos.

```
## FunÃ§Ãµes - argumentos como parÃ¢metros
## RepetiÃ§Ã£o - repete todos os elementos
rep(x = 1:5, times = 10)
```

```
##  [1] 1 2 3 4 5 1 2 3 4 5 1 2 3 4 5 1 2 3 4 5 1 2 3 4 5 1 2 3 4 5 1 2 3 4 5 1 2 3 4 5
## [41] 1 2 3 4 5 1 2 3 4 5
```

```
## RepetiÃ§Ã£o - repete cada um dos elementos
rep(x = 1:5, each = 10)
```

```
##  [1] 1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2 2 2 3 3 3 3 3 3 3 3 3 3 4 4 4 4 4 4 4 4 4 4
## [41] 5 5 5 5 5 5 5 5 5 5
```

Um ponto fundamental e que deve ser entendido Ã© o fluxo de atribuiÃ§Ãµes do resultado da operaÃ§Ã£o de funÃ§Ãµes a novos objetos. No desenvolvimento de qualquer script na linguagem R, grande parte da estrutura do mesmo serÃ¡ dessa forma: atribuiÃ§Ã£o de dados a objetos > operaÃ§Ãµes com funÃ§Ãµes > atribuiÃ§Ã£o dos resultados a novos objetos > operaÃ§Ãµes com funÃ§Ãµes desses novos objetos > atribuiÃ§Ã£o dos resultados a novos objetos. Ao entender esse funcionamento, comeÃ§amos a entender como devemos pensar na organizaÃ§Ã£o do nosso script para montar as anÃ¡lises que precisamos.

```r
## AtribuicÃ£o dos resultados
## RepetiÃ§Ã£o
rep_times <- rep(1:5, times = 10)
rep_times
```

```
##  [1] 1 2 3 4 5 1 2 3 4 5 1 2 3 4 5 1 2 3 4 5 1 2 3 4 5 1 2 3 4 5 1 2 3 4 5 1 2 3 4 5
## [41] 1 2 3 4 5 1 2 3 4 5
```

```r
## Somar e atribuir
rep_times_soma <- sum(rep_times)
rep_times_soma
```

```
## [1] 150
```

```r
## Raiz e atribuir
rep_times_soma_raiz <- sqrt(rep_times_soma)
rep_times_soma_raiz
```

```
## [1] 12.24745
```

Por fim, Ã© fundamental tambÃ©m entender a origem das funÃ§Ãµes que usamos no R. Todas as funÃ§Ãµes sÃ£o advindas de pacotes. Esses pacotes possuem duas origens.

1. pacotes jÃ¡ instalados por padrÃ£o e que sÃ£o carregados quando abrimos o R (*R Base*)
2. pacotes que instalamos e carregamos com funÃ§Ãµes

7.3.6 Pacotes

Pacotes sÃ£o conjuntos extras de funÃ§Ãµes para executar tarefas especÃficas, alÃ©m dos pacotes instalados no *R Base*. Existe literalmente milhares de pacotes (~19,000 enquanto estamos escrevendo esse livro) para as mais diversas tarefas: estatÃstica, ecologia, geografia, sensoriamento remoto, econometria, ciÃªncias sociais, grÃ¡ficos, *machine learning*, etc. Podemos verificar este vasto conjunto de pacotes pelo link que lista por nome os pacotes oficiais, ou seja, que passaram pelo crivo do **CRAN**. Existem ainda muito mais pacotes em desenvolvimento, geralmente disponibilizados em repositÃ³rios do **GitHub** ou **GitLab**.

Podemos listar esses pacotes disponÃveis no **CRAN** com esse cÃ³digo.

```r
## NÃºmero atual de pacotes no CRAN: 20192 em 15/12/2023
nrow(available.packages())
```

Primeiramente, com uma sessÃ£o do R sem carregar nenhum pacote extra, podemos verificar pacotes carregados pelo *R Base* utilizando a funÃ§Ã£o `search()`.

```r
## Verificar pacotes carregados
search()
```

Podemos ainda verificar todos pacotes instalados em nosso computador com a funÃ§Ã£o `library()`.

```r
## Verificar pacotes instalados
library()
```

No R, quando tratamos de pacotes, devemos destacar a diferenÃ§a de dois conceitos: instalar um pacote e carregar um pacote. A instalaÃ§Ã£o de pacotes possui algumas caracterÃsticas:

- Instala-se um pacote apenas uma vez
- Precisamos estar conectados Ã internet
- O nome do pacote precisa estar entre aspas na funÃ§Ã£o de instalaÃ§Ã£o
- FunÃ§Ã£o (CRAN): `install.packages()`

Vamos instalar o pacote `vegan` diretamente do CRAN, que possui funÃ§Ãµes para realizar uma sÃ©rie de anÃ¡lise em ecologia. Para isso, podemos ir em `Tools > Install Packages...`, ou ir na aba **Packages** , procurar o pacote e simplesmente clicar em "Install". Podemos ainda utilizar a funÃ§Ã£o `install.packages()`.

```
## Instalar pacotes
install.packages("vegan")
```

Podemos conferir em que diretÃ³rios um pacote serÃ¡ instalado com a funÃ§Ã£o `.libPaths()`.

```
## DiretÃ³rios de intalaÃ§Ã£o dos pacotes
.libPaths()
```

```
## [1] "C:/Users/user/AppData/Local/R/win-library/4.3"
## [2] "C:/Program Files/R/R-4.3.2/library"
```

Importante
Uma vez instalado um pacote, nÃ£o hÃ¡ necessidade de instalÃ¡-lo novamente. Entretanto, todas Ã s vezes que iniciarmos uma sessÃ£o no R, precisamos carregar os pacotes com as funÃ§Ãµes que precisamos utilizar.

O carregamento de pacotes possui algumas caracterÃsticas:

- Carrega-se o pacote toda vez que se abre uma nova sessÃ£o do R
- NÃ£o precisamos estar conectados Ã internet
- O nome do pacote nÃ£o precisa estar entre aspas na funÃ§Ã£o de carregamento
- FunÃ§Ãµes: `library()` ou `require()`

Vamos carregar o pacote `vegan` que instalamos anteriormente. Podemos ir na aba **Packages** e assinalar o pacote que queremos carregar ou utilizar a funÃ§Ã£o `library()`.

```
## Carregar pacotes
library(vegan)
```

Como dissemos, alguns pacotes em desenvolvimento encontram-se disponÃveis em repositÃ³rios como por exemplo: GitHub, GitLab e Bioconductor. Para instalar pacotes do GitHub, por exemplo, precisamos instalar e carregar o pacote `remotes`. Para funcionar, deve instalar "RTools" antes (https://cran.r-project.org/bin/windows/Rtools/) .

```
## Instalar pacote remotes
install.packages("remotes")

## Carregar pacote remotes
library(remotes)
```

Uma vez instalado e carregado esse pacote, podemos instalar o pacote do GitHub, utilizando a funÃ§Ã£o `remotes::install_github()`. Precisamos atentar para usar essa forma "nome_usuario/nome_repositorio", retirados do link do repositÃ³rio de interesse. Como exemplo, podemos instalar o pacote `eprdados` do repositÃ³rio do GitHub darrennorris/eprdados e depois utilizar a funÃ§Ã£o `library()` para carregÃ¡-lo. Para funcionar, deve instalar "RTools" antes (https://cran.r-project.org/bin/windows/Rtools/) .

```
## Instalar pacote do github
remotes::install_github("darrennorris/eprdados")

## Carregar pacote do github
library("eprdados")
```

Podemos ver a descriÃ§Ã£o de um pacote com a funÃ§Ã£o `packageDescription()`.

```
## DescriÃ§Ã£o de um pacote
packageDescription("vegan")
```

```
## Package: vegan
## Title: Community Ecology Package
## Version: 2.6-4
## Authors@R: c(person("Jari", "Oksanen", role=c("aut","cre"),
##         email="jhoksane@gmail.com"), person("Gavin L.", "Simpson",
##         role="aut", email="ucfagls@gmail.com"), person("F. Guillaume",
##         "Blanchet", role="aut"), person("Roeland", "Kindt", role="aut"),
##         person("Pierre", "Legendre", role="aut"), person("Peter R.",
##         "Minchin", role="aut"), person("R.B.", "O'Hara", role="aut"),
##         person("Peter", "Solymos", role="aut"), person("M. Henry H.",
##         "Stevens", role="aut"), person("Eduard", "Szoecs", role="aut"),
##         person("Helene", "Wagner", role="aut"), person("Matt", "Barbour",
##         role="aut"), person("Michael", "Bedward", role="aut"),
##         person("Ben", "Bolker", role="aut"), person("Daniel", "Borcard",
##         role="aut"), person("Gustavo", "Carvalho", role="aut"),
##         person("Michael", "Chirico", role="aut"), person("Miquel", "De
##         Caceres", role="aut"), person("Sebastien", "Durand", role="aut"),
##         person("Heloisa Beatriz Antoniazi", "Evangelista", role="aut"),
##         person("Rich", "FitzJohn", role="aut"), person("Michael",
##         "Friendly", role="aut"), person("Brendan","Furneaux", role="aut"),
##         person("Geoffrey", "Hannigan", role="aut"), person("Mark O.",
##         "Hill", role="aut"), person("Leo", "Lahti", role="aut"),
##         person("Dan", "McGlinn", role="aut"), person("Marie-Helene",
##         "Ouellette", role="aut"), person("Eduardo", "Ribeiro Cunha",
##         role="aut"), person("Tyler", "Smith", role="aut"),
##         person("Adrian", "Stier", role="aut"), person("Cajo J.F.", "Ter
##         Braak", role="aut"), person("James", "Weedon", role="aut"))
## Depends: permute (>= 0.9-0), lattice, R (>= 3.4.0)
## Suggests: parallel, tcltk, knitr, markdown
## Imports: MASS, cluster, mgcv
## VignetteBuilder: utils, knitr
## Description: Ordination methods, diversity analysis and other functions
##         for community and vegetation ecologists.
## License: GPL-2
## BugReports: https://github.com/vegandevs/vegan/issues
## URL: https://github.com/vegandevs/vegan
## NeedsCompilation: yes
## Packaged: 2022-10-11 08:36:07 UTC; jarioksa
## Author: Jari Oksanen [aut, cre], Gavin L. Simpson [aut], F. Guillaume
##         Blanchet [aut], Roeland Kindt [aut], Pierre Legendre [aut], Peter
##         R. Minchin [aut], R.B. O'Hara [aut], Peter Solymos [aut], M. Henry
##         H. Stevens [aut], Eduard Szoecs [aut], Helene Wagner [aut], Matt
##         Barbour [aut], Michael Bedward [aut], Ben Bolker [aut], Daniel
##         Borcard [aut], Gustavo Carvalho [aut], Michael Chirico [aut],
##         Miquel De Caceres [aut], Sebastien Durand [aut], Heloisa Beatriz
##         Antoniazi Evangelista [aut], Rich FitzJohn [aut], Michael Friendly
##         [aut], Brendan Furneaux [aut], Geoffrey Hannigan [aut], Mark O.
##         Hill [aut], Leo Lahti [aut], Dan McGlinn [aut], Marie-Helene
##         Ouellette [aut], Eduardo Ribeiro Cunha [aut], Tyler Smith [aut],
##         Adrian Stier [aut], Cajo J.F. Ter Braak [aut], James Weedon [aut]
## Maintainer: Jari Oksanen <jhoksane@gmail.com>
## Repository: CRAN
## Date/Publication: 2022-10-11 12:40:02 UTC
## Built: R 4.3.2; x86_64-w64-mingw32; 2023-11-02 03:07:22 UTC; windows
```

```
## Archs: x64
##
## -- File: C:/Users/user/AppData/Local/R/win-library/4.3/vegan/Meta/package.rds
```

A maioria dos pacotes possui conjuntos de dados que podem ser acessados pela funÃ§Ã£o `data()`. Esses conjuntos de dados podem ser usados para testar as funÃ§Ãµes do pacote. Se estiver com dÃºvida na maneira como vocÃª deve preparar a planilha para realizar uma anÃ¡lise especÃfica, entre na Ajuda (*Help*) da funÃ§Ã£o e veja os conjuntos de dados que estÃ£o no exemplo desta funÃ§Ã£o. Como exemplo, vamos carregar os dados `dune` do pacote `vegan`, que sÃ£o dados de observaÃ§Ãµes de 30 espÃ©cies vegetais em 20 locais.

```r
## Carregar dados de um pacote
library(vegan)
data(dune)
dune[1:6, 1:6]
```

```
##   Achimill Agrostol Airaprae Alopgeni Anthodor Bellpere
## 1        1        0        0        0        0        0
## 2        3        0        0        2        0        3
## 3        0        4        0        7        0        2
## 4        0        8        0        2        0        2
## 5        2        0        0        0        4        2
## 6        2        0        0        0        3        0
```

E um Ãºltimo ponto fundamental sobre pacotes, diz respeito Ã atualizaÃ§Ã£o dos mesmos. Os pacotes sÃ£o atualizados com frequÃªncia, e infelizmente (ou felizmente, pois as atualizaÃ§Ãµes podem oferecer algumas quebras entre pacotes), nÃ£o se atualizam sozinhos. Muitas vezes, a instalaÃ§Ã£o de um pacote pode depender da versÃ£o dos pacotes dependentes, e geralmente uma janela com diversas opÃ§Ãµes numÃ©ricas se abre perguntando se vocÃª quer que todos os pacotes dependentes sejam atualizados. Podemos ir na aba **Packages** e clicar em "Update" ou usar a funÃ§Ã£o `update.packages(checkBuilt = TRUE, ask = FALSE)` para atualizÃ¡-los, entretanto, essa Ã© uma funÃ§Ã£o que costuma demorar muito para terminar de ser executada.

```r
## AtualizaÃ§Ã£o dos pacotes
update.packages(checkBuilt = TRUE, ask = FALSE)
```

Destacamos e incentivamos ainda uma prÃ¡tica que achamos interessante para aumentar a reprodutibilidade de nossos cÃ³digos e scripts: a de chamar as funÃ§Ãµes de pacotes carregados dessa forma `pacote::funÃ§Ã£o()`. Com o uso dessa prÃ¡tica, deixamos claro o pacote em que a funÃ§Ã£o estÃ¡ implementada. Esta prÃ¡tica Ã© importante por que com frequÃªncia pacotes diferentes criam funÃ§Ãµes com mesmo nome, mas com caracterÃsticas internas (argumentos) diferentes. Assim, nÃ£o expressar o pacote de interesse pode gerar erros na execuÃ§Ã£o de suas anÃ¡lises. Destacamos aqui o exemplo de como instalar pacotes do GitHub do pacote `remotes`.

```r
## Pacote seguido da funÃ§Ã£o implementada daquele pacote
remotes::install_github()
```

7.3.7 Ajuda (*Help*)

Um importante passo para melhorar a usabilidade e ter mais familiaridade com a linguagem R Ã© aprender a usar a ajuda (*help*) de cada funÃ§Ã£o. Para tanto, podemos utilizar a funÃ§Ã£o `help()` ou o operador `?`, depois de ter carregado o pacote. O arquivo de ajuda do R possui tÃ³picos, que nos auxiliam muito no entendimento dos dados de entrada, argumentos e que operaÃ§Ãµes estÃ£o sendo realizadas.Abaixo descrevemos esses tÃ³picos:

- **Description**: resumo da funÃ§Ã£o
- **Usage**: como utilizar a funÃ§Ã£o e quais os seus argumentos

- **Arguments**: detalha os argumentos e como os mesmos devem ser especificados
- **Details**: detalhes importantes para se usar a funÃ§Ã£o
- **Value**: mostra como interpretar a saÃda (*output*) da funÃ§Ã£o (os resultados)
- **Note**: notas gerais sobre a funÃ§Ã£o
- **Authors**: autores da funÃ§Ã£o
- **References**: referÃªncias bibliogrÃ¡ficas para os mÃ©todos usados para construÃ§Ã£o da funÃ§Ã£o
- **See also**: funÃ§Ãµes relacionadas
- **Examples**: exemplos do uso da funÃ§Ã£o. Ã s vezes pode ser Ãºtil copiar esse trecho e colar no R para ver como funciona e como usar a funÃ§Ã£o.

Vamos realizar um exemplo, buscando o `help` da funÃ§Ã£o `aov()`, que realiza uma anÃ¡lise de variÃância.

```
## Ajuda
help(aov)
?aov
```

AlÃ©m das funÃ§Ãµes, podemos buscar detalhes de um pacote especÃfico, para uma pÃ¡gina simples do `help` utilizando a funÃ§Ã£o `help()` ou o operador `?`. Entretanto, para uma opÃ§Ã£o que ofereÃ§a uma descriÃ§Ã£o detalhada e um Ãndice de todas as funÃ§Ãµes do pacote, podemos utilizar a funÃ§Ã£o `library()`, mas agora utilizando o argumento `help`, indicando o pacote de interesse entre aspas.

```
## Ajuda do pacote
help(vegan)
?vegan

## Help detalhado
library(help = "vegan")
```

Podemos ainda procurar o nome de uma funÃ§Ã£o para realizar uma anÃ¡lise especÃfica utilizando a funÃ§Ã£o `help.search()` com o termo que queremos em inglÃªs e entre aspas.

```
## Procurar por funÃ§Ãµes que realizam modelos lineares
help.search("linear models")
```

Outra ferramenta de busca Ã© a pÃ¡gina rseek, na qual Ã© possÃvel buscar por um termo nÃ£o sÃ³ nos pacotes do R, mas tambÃ©m em listas de emails, manuais, pÃ¡ginas na internet e livros sobre o programa.

7.3.8 Ambiente (*Environment*)

O ambiente (*environment*), como vimos, Ã© onde os objetos criados sÃ£o armazenados. Ã fundamental entender que um objeto Ã© uma alocaÃ§Ã£o de um pequeno espaÃ§o na memÃ³ria RAM do nosso computador, onde o R armazenarÃ¡ um valor ou o resultado de uma funÃ§Ã£o, utilizando o nome dos objetos que definimos na atribuiÃ§Ã£o. Sendo assim, se fizermos a atribuiÃ§Ã£o de um objeto maior que o tamanho da memÃ³ria RAM do nosso computador, esse objeto nÃ£o serÃ¡ alocado, e a atribuiÃ§Ã£o nÃ£o funcionarÃ¡, retornando um erro. Existem opÃ§Ãµes para contornar esse tipo de limitaÃ§Ã£o, mas nÃ£o a abordaremos aqui. Entretanto, podemos utilizar a funÃ§Ã£o `object.size()` para saber quanto espaÃ§o nosso objeto criado estÃ¡ alocando de memÃ³ria RAM.

```
## Tamanho de um objeto
object.size(adi)
```

```
## 56 bytes
```

Podemos listar todos os objetos criados com a funÃ§Ã£o `ls()` ou `objects()`.

```
## Listar todos os objetos
ls()
```

Podemos ainda remover todos os objetos criados com a funÃ§Ã£o `rm()` ou `remove()`. Ou ainda fazer uma

funÃ§Ã£o composta para remover todos os objetos do *Environment*.

```
## Remover um objeto
rm(adi)

## Remover todos os objetos criados
rm(list = ls())
```

Quando usamos a funÃ§Ã£o `ls()` agora, nenhum objeto Ã© listado.

```
## Listar todos os objetos
ls()
```

```
## character(0)
```

Toda a vez que fechamos o R os objetos criados sÃ£o apagados do **Environment**. Dessa forma, em algumas ocasiÃµes, por exemplo, anÃ¡lises estatÃsticas que demoram um grande tempo para serem realizadas, pode ser interessante exportar alguns ou todos os objetos criados.

Para salvar todos os objetos, ou seja, todo o *Workspace*, podemos ir em `Session -> Save Workspace As...` e escolher o nome do arquivo do *Workspace*, por exemplo, "meu_workspace.RData". Podemos ainda utilizar funÃ§Ãµes para essas tarefas. A funÃ§Ã£o `save.image()` salva todo *Workspace* com a extensÃ£o `.RData`.

```
## Salvar todo o workspace
save.image(file = "meu_workspace.RData")
```

Depois disso, podemos fechar o RStudio tranquilamente e quando formos trabalhar novamente, podemos carregar os objetos criados indo em `Session -> Load Workspace...` ou utilizando a funÃ§Ã£o `load()`.

```
## Carregar todo o workspace
load("meu_workspace.RData")
```

Entretanto, em algumas ocasiÃµes, nÃ£o precisamos salvar todos os objetos. Dessa forma, podemos salvar apenas alguns objetos especÃficos usando a funÃ§Ã£o `save()`, tambÃ©m com a extensÃ£o `.RData`.

```
## Salvar apenas um objeto
save(obj1, file = "meu_obj.RData")

## Salvar apenas um objeto
save(obj1, obj2, file = "meus_objs.RData")

## Carregar os objetos
load("meus_objs.RData")
```

Ou ainda, podemos salvar apenas um objeto com a extensÃ£o `.rds`. Para isso, usamos as funÃ§Ãµes `saveRDS()` e `readRDS()`, para exportar e importar esses dados, respectivamente. Ã importante ressaltar que nesse formato `.rds`, apenas um objeto Ã© salvo por arquivo criado e que para que o objeto seja criado no *Workspace* do R, ele precisa ser lido e atribuÃdo Ã um objeto.

```
## Salvar um objeto para um arquivo
saveRDS(obj, file = "meu_obj.rds")

## Carregar esse objeto
obj <- readRDS(file = "meu_obj.rds")
```

7.3.9 CitaÃ§Ãµes

Ao utilizar o R para realizar alguma anÃ¡lise em nossos estudos, Ã© fundamental a citaÃ§Ã£o do mesmo. Para saber como citar o R em artigos, existe uma funÃ§Ã£o denominada `citation()`, que provÃª um formato genÃ©rico de citaÃ§Ã£o e um BibTeX para arquivos LaTeX e R Markdown.

```r
## CitaÃ§Ã£o do R
citation()
```

```
## To cite R in publications use:
##
##   R Core Team (2023). _R: A Language and Environment for Statistical
##   Computing_. R Foundation for Statistical Computing, Vienna, Austria.
##   <https://www.R-project.org/>.
##
## A BibTeX entry for LaTeX users is
##
##   @Manual{,
##     title = {R: A Language and Environment for Statistical Computing},
##     author = {{R Core Team}},
##     organization = {R Foundation for Statistical Computing},
##     address = {Vienna, Austria},
##     year = {2023},
##     url = {https://www.R-project.org/},
##   }
##
## We have invested a lot of time and effort in creating R, please cite it
## when using it for data analysis. See also 'citation("pkgname")' for citing
## R packages.
```

No resultado dessa funÃ§Ã£o, hÃ¡ uma mensagem muito interessante: "See also 'citation("pkgname")' for citing R packages.". Dessa forma, aconselhamos, sempre que possÃvel, claro, citar tambÃ©m os pacotes utilizados nas anÃ¡lises para dar os devidos crÃ©ditos aos desenvolvedores e desenvolvedoras das funÃ§Ãµes implementadas nos pacotes. Como exemplo, vamos ver como fica a citaÃ§Ã£o do pacote `vegan`.

```r
## CitaÃ§Ã£o do pacote vegan
citation("vegan")
```

```
## To cite package 'vegan' in publications use:
##
##   Oksanen J, Simpson G, Blanchet F, Kindt R, Legendre P, Minchin P, O'Hara
##   R, Solymos P, Stevens M, Szoecs E, Wagner H, Barbour M, Bedward M,
##   Bolker B, Borcard D, Carvalho G, Chirico M, De Caceres M, Durand S,
##   Evangelista H, FitzJohn R, Friendly M, Furneaux B, Hannigan G, Hill M,
##   Lahti L, McGlinn D, Ouellette M, Ribeiro Cunha E, Smith T, Stier A, Ter
##   Braak C, Weedon J (2022). _vegan: Community Ecology Package_. R package
##   version 2.6-4, <https://CRAN.R-project.org/package=vegan>.
##
## A BibTeX entry for LaTeX users is
##
##   @Manual{,
##     title = {vegan: Community Ecology Package},
##     author = {Jari Oksanen and Gavin L. Simpson and F. Guillaume Blanchet and Roeland Kindt and Pier
##     year = {2022},
##     note = {R package version 2.6-4},
##     url = {https://CRAN.R-project.org/package=vegan},
##   }
```

Podemos ainda utilizar a funÃ§Ã£o `write_bib()` do pacote `knitr` para exportar a citaÃ§Ã£o do pacote no formato `.bib`.

```r
## Exportar uma citaÃ§Ã£o em formato .bib
knitr::write_bib("vegan", file = "vegan_ex.bib")
```

7.3.10 Principais erros de iniciantes

Errar quando se estÃ¡ começÃ§ando a usar o R Ã© muito comum e faz parte do aprendizado. Entretanto, os erros nunca devem ser encarados como uma forma de desestÃmulo, mas sim como um desafio para continuar tentando. Todos nÃ³s, autores deste livro inclusive, e provavelmente usuÃ¡rios mais ou menos experientes, jÃ¡ passaram por um momento em que se quer desistir de tudo. Jovem aprendiz de R, a Ãºnica diferenÃ§a entre vocÃª que estÃ¡ iniciando agora e nÃ³s que usamos o R hÃ¡ mais tempo sÃ£o as horas a mais de uso (e Ã³dio). O que temos a mais Ã© experiÃªncia para olhar o erro, lÃª-lo e conseguir interpretar o que estÃ¡ errado e saber buscar ajuda.

Dessa forma, o ponto mais importante de quem estÃ¡ iniciando Ã© ter paciÃªncia, calma, bom humor, ler e entender as mensagens de erros. Recomendamos uma prÃ¡tica que pode ajudar: caso nÃ£o esteja conseguindo resolver alguma parte do seu cÃ³digo, deixe ele de lado um tempo, descanse, faÃ§a uma caminhada, tome um banho, converse com seus animais de estimaÃ§Ã£o ou plantas, tenha um pato de borracha ou outro objeto inanimado (um dos autores tem um sapinho de madeira), explique esse cÃ³digo para esse pato (processo conhecido como Debug com Pato de Borracha), logo a soluÃ§Ã£o deve aparecer.

Listaremos aqui o que consideramos os principais erros dos iniciantes no R.

1. Esquecer de completar uma funÃ§Ã£o ou bloco de cÃ³digos

Esquecer de completar uma funÃ§Ã£o ou bloco de cÃ³digos Ã© algo bem comum. Geralmente esquecemos de fechar aspas `""` ou parÃªnteses `()`, mas geralmente o R nos informa isso, indicando um sÃmbolo de `+` no console. Se vocÃª cometeu esse erro, lembre-se de apertar a tecla esc do seu computador clicando antes com o cursor do mouse no console do R.

```r
sum(1, 2
  +
```

```
## Error: <text>:3:0: unexpected end of input
## 1: sum(1, 2
## 2:   +
##   ^
```

2. Esquecer de vÃrgulas dentro de funÃ§Ãµes

Outro erro bastante comum Ã© esquecer de acrescentar a vÃrgula `,` para separar argumentos dentro de uma funÃ§Ã£o, principalmente se estamos compondo vÃ¡rias funÃ§Ãµes acopladas, i.e., uma funÃ§Ã£o dentro da outra.

```r
sum(1 2)
```

```
## Error: <text>:1:7: unexpected numeric constant
## 1: sum(1 2
##            ^
```

3. Chamar um objeto pelo nome errado

Pode parecer simples, mas esse Ã© de longe o erro mais comum que pessoas iniciantes comentem. Quando temos um script longo, Ã© de se esperar que tenhamos atribuÃdo diversos objetos e em algum momento atribuÃmos um nome do qual nÃ£o lembramos. Dessa forma, quando chamamos o objeto ele nÃ£o existe e o console informa um erro. Entretanto, esse tipo de erro pode ser facilmente identificado, como o exemplo abaixo.

```
obj <- 10
OBJ
```

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

4. Esquecer de carregar um pacote

Esse tambÃ©m Ã© um erro recorrente, mesmo para usuÃ¡rios mais experientes. Em scripts de anÃ¡lises complexas, que requerem vÃ¡rios pacotes, geralmente esquecemos de um ou outro pacote. A melhor forma de evitar esse tipo de erro Ã© listar os pacotes que vamos precisar usar logo no inÃcio do script.

```
## Carregar dados
data(dune)
```

```
## Warning in data(dune): data set 'dune' not found
```

```
## FunÃ§Ã£o do pacote vegan
decostand(dune[1:6, 1:6], "hell")
```

```
## Error in decostand(dune[1:6, 1:6], "hell"): could not find function "decostand"
```

Geralmente a mensagem de erro serÃ¡ de que a funÃ§Ã£o nÃ£o foi encontrada ou algo nesse sentido. Carregando o pacote, esse erro Ã© contornado.

```
## Carregar o pacote
library(vegan)
```

```
## This is vegan 2.6-4
```

```
##
## Attaching package: 'vegan'
```

```
## The following object is masked from 'package:parsnip':
##
##     nullmodel
```

```
## Carregar dados
data(dune)

## FunÃ§Ã£o do pacote vegan
decostand(dune[1:6, 1:6], "hell")
```

```
##    Achimill  Agrostol Airaprae  Alopgeni  Anthodor  Bellpere
## 1 1.0000000 0.0000000        0 0.0000000 0.0000000 0.0000000
## 2 0.6123724 0.0000000        0 0.5000000 0.0000000 0.6123724
## 3 0.0000000 0.5547002        0 0.7337994 0.0000000 0.3922323
## 4 0.0000000 0.8164966        0 0.4082483 0.0000000 0.4082483
## 5 0.5000000 0.0000000        0 0.0000000 0.7071068 0.5000000
## 6 0.6324555 0.0000000        0 0.0000000 0.7745967 0.0000000
```

5. Usar o nome da funÃ§Ã£o de forma errÃ´nea

Esse erro nÃ£o Ã© tÃ£o comum, mas pode ser incÃ´modo Ã s vezes. Algumas funÃ§Ãµes possuem nomes no padrÃ£o "Camel Case", i.e., com letras maiÃºsculas no meio do nome da funÃ§Ã£o. Isso Ã s vezes pode

confundir, ou ainda, as funÃ§Ãµes podem ou nÃ£o ser separadas com `.`, como `row.names()` e `rownames()`.

```
## Soma das colunas
colsums(dune)
```

```
## Error in colsums(dune): could not find function "colsums"
```

```
## Soma das colunas
colSums(dune)
```

```
## Achimill Agrostol Airaprae Alopgeni Anthodor Bellpere Bromhord Chenalbu Cirsarve
##       16       48        5       36       21       13       15        1        2
## Comapalu Eleopalu Elymrepe Empenigr Hyporadi Juncarti Juncbufo Lolipere Planlanc
##        4       25       26        2        9       18       13       58       26
##  Poaprat  Poatriv Ranuflam Rumeacet Sagiproc Salirepe Scorautu Trifprat Trifrepe
##       48       63       14       18       20       11       54        9       47
## Vicilath Bracruta Callcusp
##        4       49       10
```

6. Atentar para o diretÃ³rio correto

Muitas vezes o erro Ã© simplesmente porque o usuÃ¡rio(a) nÃ£o definiu o diretÃ³rio correto onde estÃ¡ o arquivo a ser importado ou exportado. Por isso Ã© fundamental sempre verificar se o diretÃ³rio foi definido corretamente, geralmente usando as funÃ§Ãµes `dir()` ou `list.files()` para listar no console a lista de arquivos no diretÃ³rio. Podemos ainda usar o argumento `pattern` para listar arquivos por um padrÃ£o textual.

```
## Listar os arquivos do diretÃ³rio definido
dir()
list.files()

## Listar os arquivos do diretÃ³rio definido por um padrÃ£o
dir(pattern = ".csv")
```

AlÃ©m disso, Ã© fundamental ressaltar a importÃ¢ncia de verificar se o nome do arquivo que importaremos foi digitado corretamente, atentando-se tambÃ©m para a extensÃ£o: `.csv`, `.txt`, `.xlsx`, etc.

7.4 Estrutura e manipulaÃ§Ã£o de objetos

O conhecimento sobre a estrutura e manipulaÃ§Ã£o de objetos Ã© fundamental para ter domÃnio e entendimento do funcionamento da linguagem R. Nesta seÃ§Ã£o, trataremos da estrutura e manipulaÃ§Ã£o de dados no R, no que ficou conhecido como modo *R Base*, em contrapartida ao *tidyverse*, tÃ³pico tratado no CapÃtulo 8. Abordaremos aqui temas chaves, como: i) atributos de objetos, ii) manipulaÃ§Ã£o de objetos unidimensionais e multidimensionais, iii) valores faltantes e especiais, iv) diretÃ³rio de trabalho e v) importar, conferir e exportar dados tabulares.

7.4.1 Atributo dos objetos

Quando fazemos atribuiÃ§Ãµes de dados no R (`<-`), os objetos gerados possuem trÃªs caracterÃsticas.

1. **Nome**: palavra que o R reconhece os dados atribuÃdos
2. **ConteÃºdo**: dados em si
3. **Atributos**: modos (*natureza*) e estruturas (*organizaÃ§Ã£o*) dos elementos

Vamos explorar mais a fundo os **modos** e **estruturas** dos objetos. Vale ressaltar que isso Ã© uma simplificaÃ§Ã£o, pois hÃ¡ muitas classes de objetos, como funÃ§Ãµes e saÃdas de funÃ§Ãµes que possuem outros atributos.

Podemos verificar os atributos dos objetos com a funÃ§Ã£o `attributes()`.

```r
## Atributos
attributes(dune)
```

```
## $names
##  [1] "Achimill" "Agrostol" "Airaprae" "Alopgeni" "Anthodor" "Bellpere" "Bromhord"
##  [8] "Chenalbu" "Cirsarve" "Comapalu" "Eleopalu" "Elymrepe" "Empenigr" "Hyporadi"
## [15] "Juncarti" "Juncbufo" "Lolipere" "Planlanc" "Poaprat"  "Poatriv"  "Ranuflam"
## [22] "Rumeacet" "Sagiproc" "Salirepe" "Scorautu" "Trifprat" "Trifrepe" "Vicilath"
## [29] "Bracruta" "Callcusp"
##
## $row.names
##  [1] "1"  "2"  "3"  "4"  "5"  "6"  "7"  "8"  "9"  "10" "11" "12" "13" "14" "15" "16"
## [17] "17" "18" "19" "20"
##
## $class
## [1] "data.frame"
```

Modo dos objetos

A depender da natureza dos elementos que compÃµem os dados e que foram atribuÃdos aos objetos, esses objetos podem ser, de forma simples um dos cinco modos: numÃ©rico do tipo inteiro (*integer*), numÃ©rico do tipo flutuante (*double*), texto (*character*), lÃ³gico (*logical*) ou complexo (*complex*).

A atribuiÃ§Ã£o de nÃºmeros no R pode gerar dois tipos de modos: **integer** para nÃºmeros inteiros e **double** para nÃºmeros flutuantes ou com decimais.

```r
## NumÃ©rico double
obj_numerico_double <- 1

## Modo
mode(obj_numerico_double)
```

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

```r
## Tipo
typeof(obj_numerico_double)
```

```
## [1] "double"
```

A tÃtulo de praticidade, ambos sÃ£o incorporados como o modo numeric, com o tipo double, a menos que especifiquemos que seja inteiro com a letra `L` depois do nÃºmero, representando a palavra *Larger*, geralmente usando para armazenar nÃºmeros muito grandes.

```r
## NumÃ©rico integer
obj_numerico_inteiro <- 1L

## Modo
mode(obj_numerico_inteiro)
```

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

```r
## Tipo
typeof(obj_numerico_inteiro)
```

```
## [1] "integer"
```

AlÃ©m de nÃºmeros, podemos atribuir textos, utilizando para isso aspas `""`.

```r
## Caracter ou string
obj_caracter <- "a" # atencao para as aspas
```

```r
## Modo
mode(obj_caracter)
```

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

Em algumas situaÃ§Ãµes, precisamos indicar a ocorrÃªncia ou nÃ£o de um evento ou uma operaÃ§Ã£o. Para isso, utilizamos as palavras reservadas (`TRUE` e `FALSE`), chamadas de variÃ¡veis booleanas, pois assumem apenas duas possibilidades: falso (0) ou verdadeiro (1). Devemos nos ater para o fato dessas palavras serem escritas com letras maiÃºsculas e sem aspas.

```r
## LÃ³gico
obj_logico <- TRUE # maiusculas e sem aspas

## Modo
mode(obj_logico)
```

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

Por fim, existe um modo pouco utilizado que cria nÃºmeros complexos (raiz de nÃºmeros negativos).

```r
## Complexo
obj_complexo <- 1+1i

## Modo
mode(obj_complexo)
```

```
## [1] "complex"
```

Podemos verificar o modo dos objetos ou fazer a conversÃ£o entre esses modos com diversas funÃ§Ãµes.

```r
## Verificar o modo dos objetos
is.numeric()
is.integer()
is.character()
is.logical()
is.complex()

## ConversÃµes entre modos
as.numeric()
as.integer()
as.character()
as.logical()
as.complex()

## Exemplo
num <- 1:5
num
mode(num)

cha <- as.character(num)
cha
mode(cha)
```

Estrutura dos objetos

Uma vez entendido a natureza dos modos dos elementos dos objetos no R, podemos passar para o passo seguinte e entender como esses elementos sÃ£o estruturados dentro dos objetos.

Essa estruturaÃ§Ã£o irÃ¡ nos contar sobre a organizaÃ§Ã£o dos elementos, com relaÃ§Ã£o aos modos e dimensionalidade da disposiÃ§Ã£o desses elementos. De modo bem simples, os elementos podem ser estruturados em cinco tipos:

1. **Vetores e fatores**: homogÃªneo (*um modo*) e unidimensional (*uma dimensÃ£o*). Um tipo especial de vetor sÃ£o os fatores, usados para designar variÃ¡veis categÃ³ricas
2. **Matrizes**: homogÃªneo (*um modo*) e bidimensional (*duas dimensÃµes*)
3. **Arrays**: homogÃªneo (*um modo*) e multidimensional (*mais de duas dimensÃµes*)
4. **Data frames**: heterogÃªneo (*mais de um modo*) e bidimensional (*duas dimensÃµes*)
5. **Listas**: heterogÃªneo (*mais de um modo*) e unidimensional (*uma dimensÃ£o*)

Figura 7.3: Estruturas de dados mais comuns no R: vetores, matrizes, arrays, listas e data frames. Adaptado de: R for Data Science (2e) .

Vetor

Vetores representam o encadeamento de elementos numa sequÃªncia unidimensional. Dessa forma, no R, essa estrutura de dados pode ser traduzida como medidas de uma variÃ¡vel numÃ©rica (discretas ou contÃnuas), variÃ¡vel binÃ¡ria (booleana - TRUE e FALSE) ou descriÃ§Ã£o (informaÃ§Ãµes em texto).

HÃ¡ diversas formas de se criar um vetor no R:

1. Concatenando elementos com a funÃ§Ã£o `c()`
2. Criando sequÃªncias unitÃ¡rias `:` ou com a funÃ§Ã£o `seq()`
3. Criando repetiÃ§Ãµes com a funÃ§Ã£o `rep()`
4. "Colar" palavras com uma sequÃªncia numÃ©rica com a funÃ§Ã£o `paste()` ou `paste0()`
5. Amostrando aleatoriamente elementos com a funÃ§Ã£o `sample()`

```
## Concatenar elementos numÃ©ricos
concatenar <- c(15, 18, 20, 22, 18)
concatenar
```

```
## [1] 15 18 20 22 18
```

```r
## SequÃªncia unitÃ¡ria (x1:x2)
sequencia <- 1:10
sequencia
```

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

```r
## SequÃªncia com diferentes espaÃ§amentos
sequencia_esp <- seq(from = 0, to = 100, by = 10)
sequencia_esp
```

```
##  [1]   0  10  20  30  40  50  60  70  80  90 100
```

```r
## RepetiÃ§Ã£o
repeticao <- rep(x = c(TRUE, FALSE), times = 5)
repeticao
```

```
##  [1]  TRUE FALSE  TRUE FALSE  TRUE FALSE  TRUE FALSE  TRUE FALSE
```

```r
## Cola palavra e sequÃªncia numÃ©rica
colar <- paste("amostra", 1:5)
colar
```

```
## [1] "amostra 1" "amostra 2" "amostra 3" "amostra 4" "amostra 5"
```

```r
## Amostragem aleatÃ³ria
amostragem <- sample(x = 1:100, size = 10)
amostragem
```

```
##  [1] 66 28 31 98  8 60 17 14  2 15
```

Como os vetores sÃ£o homogÃªneos, i.e., sÃ³ comportam um modo, quando combinamos mais de um modo no mesmo objeto ocorre uma dominÃ¢ncia de modos. Existe, dessa forma, uma **coerÃ§Ã£o** dos elementos combinados para que todos fiquem iguais. Essa dominÃ¢ncia segue essa ordem:

> `DOMINANTE` character > double > integer > logical `RECESSIVO`

AlÃ©m disso, podemos utilizar as conversÃµes listadas anteriormente para alterar os modos. Vamos exemplificar combinando os vetores criados anteriormente e convertendo-os.

```r
## CoerÃ§Ã£o
c(colar, amostragem)
```

```
##  [1] "amostra 1" "amostra 2" "amostra 3" "amostra 4" "amostra 5" "66"
##  [7] "28"        "31"        "98"        "8"         "60"        "17"
## [13] "14"        "2"         "15"
```

```r
## ConversÃ£o
as.numeric(repeticao)
```

```
##  [1] 1 0 1 0 1 0 1 0 1 0
```

Fator

O fator representa medidas de uma variÃ¡vel categÃ³rica, podendo ser nominal ou ordinal. Ã fundamental destacar que fatores no R devem ser entendidos como um vetor de **integer**, i.e., ele Ã© composto por nÃºmeros inteiros representando os nÃveis da variÃ¡vel categÃ³rica.

Para criar um fator no R usamos uma funÃ§Ã£o especÃfica `factor()`, na qual podemos especificar os **nÃveis** com o argumento `level`, ou fazemos uma conversÃ£o usando a funÃ§Ã£o `as.factor()`. Trabalhar com fatores no *R Base* nÃ£o Ã© das tarefas mais agradÃ¡veis, sendo assim, no CapÃtulo 8 usamos a versÃ£o *tidyverse* usando o pacote `forcats`. Destacamos ainda a existÃªncia de fatores nominais para variÃ¡veis

categÃ³ricas nominais e fatores ordinais para variÃ¡veis categÃ³ricas ordinais, quando hÃ¡ ordenamento entre os nÃveis, como dias da semana ou classes de altura.

```
## Fator nominal
fator_nominal <- factor(x = sample(x = c("floresta", "pastagem", "cerrado"),
                                   size = 20, replace = TRUE),
                        levels = c("floresta", "pastagem", "cerrado"))
fator_nominal
```

```
##  [1] pastagem cerrado  floresta cerrado  pastagem cerrado  pastagem cerrado
##  [9] pastagem floresta cerrado  cerrado  pastagem cerrado  cerrado  pastagem
## [17] floresta pastagem floresta cerrado
## Levels: floresta pastagem cerrado
```

```
## Fator ordinal
fator_ordinal <- factor(x = sample(x = c("baixa", "media", "alta"),
                                   size = 20, replace = TRUE),
                        levels = c("baixa", "media", "alta"), ordered = TRUE)
fator_ordinal
```

```
##  [1] alta  baixa baixa baixa media alta  baixa alta  baixa baixa alta  media media
## [14] baixa media media alta  media media alta
## Levels: baixa < media < alta
```

```
## ConversÃ£o
fator <- as.factor(x = sample(x = c("floresta", "pastagem", "cerrado"),
                              size = 20, replace = TRUE))
fator
```

```
##  [1] cerrado  floresta floresta floresta cerrado  pastagem floresta pastagem
##  [9] pastagem pastagem floresta cerrado  pastagem pastagem pastagem pastagem
## [17] cerrado  pastagem cerrado  floresta
## Levels: cerrado floresta pastagem
```

Matriz

A matriz representa dados no formato de tabela, com linhas e colunas. As linhas geralmente representam unidades amostrais (locais, transectos, parcelas) e as colunas representam variÃ¡veis numÃ©ricas (discretas ou contÃnuas), variÃ¡veis binÃ¡rias (TRUE ou FALSE) ou descriÃ§Ãµes (informaÃ§Ãµes em texto).

Podemos criar matrizes no R de duas formas. A primeira delas dispondo elementos de um vetor em um certo nÃºmero de linhas e colunas com a funÃ§Ã£o `matrix()`, podendo preencher essa matriz com os elementos do vetor por linhas ou por colunas alterando o argumento `byrow`.

```
## Vetor
ve <- 1:12

## Matrix - preenchimento por linhas - horizontal
ma_row <- matrix(data = ve, nrow = 4, ncol = 3, byrow = TRUE)
ma_row
```

```
##      [,1] [,2] [,3]
## [1,]    1    2    3
## [2,]    4    5    6
## [3,]    7    8    9
## [4,]   10   11   12
```

```
## Matrix - preenchimento por colunas - vertical
ma_col <- matrix(data = ve, nrow = 4, ncol = 3, byrow = FALSE)
```

```r
ma_col
```

```
##      [,1] [,2] [,3]
## [1,]    1    5    9
## [2,]    2    6   10
## [3,]    3    7   11
## [4,]    4    8   12
```

A segundo forma, podemos combinar vetores, utilizando a funÃ§Ã£o `rbind()` para combinar vetores por linha, i.e., um vetor embaixo do outro, e `cbind()` para combinar vetores por coluna, i.e., um vetor ao lado do outro.

```r
## Criar dois vetores
vec_1 <- c(1, 2, 3)
vec_2 <- c(4, 5, 6)

## Combinar por linhas - vertical - um embaixo do outro
ma_rbind <- rbind(vec_1, vec_2)
ma_rbind
```

```
##       [,1] [,2] [,3]
## vec_1    1    2    3
## vec_2    4    5    6
```

```r
## Combinar por colunas - horizontal - um ao lado do outro
ma_cbind <- cbind(vec_1, vec_2)
ma_cbind
```

```
##      vec_1 vec_2
## [1,]     1     4
## [2,]     2     5
## [3,]     3     6
```

Array

O array representa combinaÃ§Ã£o de tabelas, com linhas, colunas e dimensÃµes. Essa combinaÃ§Ã£o pode ser feita em mÃºltiplas dimensÃµes, mas apesar disso, geralmente Ã© mais comum o uso em Ecologia para trÃªs dimensÃµes, por exemplo: linhas (unidades amostrais), colunas (espÃ©cies) e dimensÃ£o (tempo). Isso gera um "cubo mÃ¡gico" ou "cartas de um baralho", onde podemos comparar, nesse caso, comunidades ao longo do tempo. AlÃ©m disso, arrays tambÃ©m sÃ£o muito comuns em morfometria geomÃ©trica ou sensoriamento remoto.

Podemos criar arrays no R dispondo elementos de um vetor em um certo nÃºmero de linhas, colunas e dimensÃµes com a funÃ§Ã£o `array()`. Em nosso exemplo, vamos compor cinco comunidades de cinco espÃ©cies ao longo de trÃªs perÃodos.

```r
## Array
ar <- array(data = sample(x = c(0, 1), size = 75, rep = TRUE),
            dim = c(5, 5, 3))
ar
```

```
## , , 1
##
##      [,1] [,2] [,3] [,4] [,5]
## [1,]    1    1    0    1    0
## [2,]    1    0    1    0    1
## [3,]    0    0    0    1    1
## [4,]    0    1    1    0    1
```

```
## [5,]    0    1    0    0    0
##
## , , 2
##
##      [,1] [,2] [,3] [,4] [,5]
## [1,]    1    0    0    1    0
## [2,]    0    0    0    1    0
## [3,]    0    1    1    0    1
## [4,]    1    0    1    0    0
## [5,]    1    1    1    1    0
##
## , , 3
##
##      [,1] [,2] [,3] [,4] [,5]
## [1,]    0    0    0    1    1
## [2,]    0    0    1    1    1
## [3,]    0    1    1    0    0
## [4,]    1    0    0    0    1
## [5,]    1    0    1    0    0
```

Data frame

O data frame tambÃ©m representa dados no formato de tabela, com linhas e colunas, muito semelhante Ã matriz. Mas diferentemente das matrizes, os data frames comportam mais de um modo em suas colunas. Dessa forma, as linhas do data frame ainda representam unidades amostrais (locais, transectos, parcelas), mas as colunas agora podem representar descriÃ§Ãµes (informaÃ§Ãµes em texto), variÃ¡veis numÃ©ricas (discretas ou contÃnuas), variÃ¡veis binÃ¡rias (TRUE ou FALSE) e variÃ¡veis categÃ³ricas (nominais ou ordinais).

A forma mais simples de se criar data frames no R Ã© atravÃ©s da combinaÃ§Ã£o de vetores. Essa combinaÃ§Ã£o Ã© feita com a funÃ§Ã£o `data.frame()` e ocorre de forma horizontal, semelhante Ã funÃ§Ã£o `cbind()`. Sendo assim, todos os vetores precisam ter o mesmo nÃºmero de elementos, ou seja, o mesmo comprimento. Podemos ainda nomear as colunas de cada vetor. Outra forma, seria converter uma matriz em um data frame, utilizando a funÃ§Ã£o `as.data.frame()`.

```r
## Criar trÃªs vetores
vec_ch <- c("sp1", "sp2", "sp3")
vec_nu <- c(4, 5, 6)
vec_fa <- factor(c("campo", "floresta", "floresta"))

## Data frame - combinar por colunas - horizontal - um ao lado do outro
df <- data.frame(vec_ch, vec_nu, vec_fa)

df
```

```
##   vec_ch vec_nu   vec_fa
## 1    sp1      4    campo
## 2    sp2      5 floresta
## 3    sp3      6 floresta
```

```r
## Data frame - nomear as colunas
df <- data.frame(especies = vec_ch,
                 abundancia = vec_nu,
                 vegetacao = vec_fa)
df
```

```
##   especies abundancia vegetacao
```

```
## 1      sp1          4     campo
## 2      sp2          5  floresta
## 3      sp3          6  floresta
```

```
## Data frame - converter uma matriz
ma <- matrix(data = ve, nrow = 4, ncol = 3, byrow = TRUE)
ma
```

```
##      [,1] [,2] [,3]
## [1,]    1    2    3
## [2,]    4    5    6
## [3,]    7    8    9
## [4,]   10   11   12
```

```
df_ma <- as.data.frame(ma)
df_ma
```

```
##   V1 V2 V3
## 1  1  2  3
## 2  4  5  6
## 3  7  8  9
## 4 10 11 12
```

Lista

A lista Ã© um tipo especial de vetor que aceita objetos como elementos. Ela Ã© a estrutura de dados utilizada para agrupar objetos, e Ã© geralmente a saÃda de muitas funÃ§Ãµes.

Podemos criar listas atravÃ©s da funÃ§Ã£o `list()`. Essa funÃ§Ã£o funciona de forma semelhante Ã funÃ§Ã£o `c()` para a criaÃ§Ã£o de vetores, mas agora estamos concatenando objetos. Podemos ainda nomear os elementos (objetos) que estamos combinando.

Um ponto interessante para entender data frames, Ã© que eles sÃ£o listas, em que todos os elementos (colunas) possuem o mesmo nÃºmero de elementos, ou seja, mesmo comprimento.

```
## Lista
lista <- list(rep(1, 20), # vector
              factor(1, 1), # factor
              cbind(c(1, 2), c(1, 2))) # matrix
lista
```

```
## [[1]]
##  [1] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
##
## [[2]]
## [1] 1
## Levels: 1
##
## [[3]]
##      [,1] [,2]
## [1,]    1    1
## [2,]    2    2
```

```
## Lista - nomear os elementos
lista_nome <- list(vector = rep(1, 20), # vector
              factor = factor(1, 1), # factor
              matrix = cbind(c(1, 2), c(1, 2))) # matrix
lista_nome
```

```
## $vector
##  [1] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
##
## $factor
## [1] 1
## Levels: 1
##
## $matrix
##      [,1] [,2]
## [1,]    1    1
## [2,]    2    2
```

FunÃ§Ãµes

Uma Ãºltima estrutura de objetos criados no R sÃ£o as funÃ§Ãµes. Elas sÃ£o objetos criados pelo usuÃ¡rio e reutilizados para fazer operaÃ§Ãµes especÃficas. A criaÃ§Ã£o de funÃ§Ãµes geralmente Ã© um tÃ³pico tratado num segundo momento, quando o usuÃ¡rio de R adquire certo conhecimento da linguagem. Aqui abordaremos apenas seu funcionamento bÃ¡sico, diferenciando sua estrutura para entendimento e sua diferenciaÃ§Ã£o das demais estruturas.

Vamos criar uma funÃ§Ã£o simples que retorna a multiplicaÃ§Ã£o de dois termos. Criaremos a funÃ§Ã£o com o nome `multi`, Ã qual serÃ¡ atribuÃda uma funÃ§Ã£o com o nome `function()`, com dois argumentos `x` e `y`. Depois disso abrimos chaves `{}`, que Ã© onde iremos incluir nosso bloco de cÃ³digo. Nosso bloco de cÃ³digo Ã© composto por duas linhas, a primeira contendo a operaÃ§Ã£o de multiplicaÃ§Ã£o dos argumento com a atribuiÃ§Ã£o ao objeto `mu` e a segunda contendo a funÃ§Ã£o `return()` para retornar o valor da multiplicaÃ§Ã£o.

```
## Criar uma funÃ§Ã£o
multi <- function(x, y){

  mu <- (x * y)
  return(mu)

}
multi
```

```
## function(x, y){
##
##   mu <- (x * y)
##   return(mu)
##
## }
```

```
## Uso da funÃ§Ã£o
multi(42, 23)
```

```
## [1] 966
```

7.4.2 ManipulaÃ§Ã£o de objetos unidimensionais

Vamos agora explorar formas de manipular elementos de objetos unidimensionais, ou seja, vetores, fatores e listas.

A primeira forma de manipulaÃ§Ã£o Ã© atravÃ©s da **indexaÃ§Ã£o**, utilizando os operadores `[]`. Com a indexaÃ§Ã£o podemos acessar elementos de vetores e fatores por sua posiÃ§Ã£o. Utilizaremos nÃºmeros, sequÃªncia de nÃºmeros ou operaÃ§Ãµes booleanas para retornar partes dos vetores ou fatores. Podemos ainda retirar elementos dessas estruturas com o operador aritmÃ©tico `-`.

No exemplo a seguir, iremos fixar o ponto de partida da amostragem da funÃ§Ã£o `sample()`, utilizando a funÃ§Ã£o `set.seed(42)` (usamos 42 porque Ã© a resposta para a vida, o universo e tudo mais - O Guia do Mochileiro das GalÃ¡xias, mas poderia ser outro nÃºmero qualquer). Isso permite que o resultado da amostragem aleatÃ³ria seja igual em diferentes computadores.

```
## Fixar a amostragem
set.seed(42)

## Amostrar 10 elementos de uma sequÃªncia
ve <- sample(x = seq(0, 2, .05), size = 10)
ve
```

```
##  [1] 1.80 0.00 1.20 0.45 1.75 0.85 1.15 0.30 1.90 0.20
```

```
## Seleciona o quinto elemento
ve[5]
```

```
## [1] 1.75
```

```
## Seleciona os elementos de 1 a 5
ve[1:5]
```

```
## [1] 1.80 0.00 1.20 0.45 1.75
```

```
## Retira o dÃ©cimo elemento
ve[-10]
```

```
## [1] 1.80 0.00 1.20 0.45 1.75 0.85 1.15 0.30 1.90
```

```
## Retira os elementos 2 a 9
ve[-(2:9)]
```

```
## [1] 1.8 0.2
```

Podemos ainda fazer uma seleÃ§Ã£o condicional do vetor. Ao utilizarmos operadores relacionais, teremos como resposta um vetor lÃ³gico. Esse vetor lÃ³gico pode ser utilizado dentro da indexaÃ§Ã£o para seleÃ§Ã£o de elementos.

```
## Quais valores sao maiores que 1?
ve > 1
```

```
##  [1]  TRUE FALSE  TRUE FALSE  TRUE FALSE  TRUE FALSE  TRUE FALSE
```

```
## Selecionar os valores acima de 1 no vetor ve
ve[ve > 1]
```

```
## [1] 1.80 1.20 1.75 1.15 1.90
```

AlÃ©m da indexaÃ§Ã£o, temos algumas funÃ§Ãµes que nos auxiliam em algumas operaÃ§Ãµes com objetos unidimensionais.

Table 2: FunÃ§Ãµes para verificaÃ§Ã£o e resumo de dados unidimensionais.

FunÃ§Ã£o	DescriÃ§Ã£o
`max()`	Valor mÃ¡ximo
`min()`	Valor mÃnimo
`range()`	Amplitude
`length()`	Comprimento
`sum()`	Soma
`cumsum()`	Soma cumulativa

FunÃ§Ã£o	DescriÃ§Ã£o
`prod()`	Produto
`sqrt()`	Raiz quadrada
`abs()`	Valor absoluto
`exp()`	Expoente
`log()`	Logaritmo natural
`log1p()`	Logaritmo natural mais 1 *log(x + 1)*
`log2()`	Logaritmo base 2
`log10()`	Logaritmo base 10
`mean()`	MÃ©dia
`mean.weighted()`	MÃ©dia ponderada
`var()`	VariÃ¢ncia
`sd()`	Desvio PadrÃ£o
`mediam()`	Mediana
`quantile()`	Quantil
`quarters()`	Quartil
`IQR()`	Amplitude interquartil
`round()`	Arredondamento
`sort()`	OrdenaÃ§Ã£o
`order()`	PosiÃ§Ã£o ordenada
`rev()`	Reverso
`unique()`	Ã nicos
`summary()`	Resumo estatÃstico
`cut()`	Divide variÃ¡vel contÃnua em fator
`pretty()`	Divide variÃ¡vel contÃnua em intervalos
`scale()`	PadronizaÃ§Ã£o e centralizaÃ§Ã£o
`sub()`	Substitui caracteres
`grep()`	PosiÃ§Ã£o de caracteres
`any()`	Algum valor?
`all()`	Todos os valores?
`which()`	Quais valores?
`subset()`	Subconjunto
`ifelse()`	OperaÃ§Ã£o condicional

Para listas, tambÃ©m podemos usar a indexaÃ§Ã£o `[]` para acessar ou retirar elementos.

```
## Lista
li <- list(elem1 = 1, elem2 = 2, elem3 = 3)

## Acessar o primeiro elemento
li[1]
```

```
## $elem1
## [1] 1
```

```
## Retirar o primeiro elemento
li[-1]
```

```
## $elem2
## [1] 2
##
## $elem3
## [1] 3
```

Podemos ainda usar a indexaÃ§Ã£o dupla `[[]]` para acessar os valores desses elementos.

```r
## Acessar o valor do primeiro elemento
li[[1]]
```

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

```r
## Acessar o valor do segundo elemento
li[[2]]
```

```
## [1] 2
```

Para listas nomeadas, podemos ainda utilizar o operador `$` para acessar elementos pelo seu nome.

```r
## Acessar o primeiro elemento
li$elem1
```

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

E ainda podemos utilizar funÃ§Ãµes para medir o comprimento dessa lista, listar os nomes dos elementos ou ainda renomear os elementos: `length()` e `names()`.

```r
## Comprimento
length(li)
```

```
## [1] 3
```

```r
## Nomes
names(li)
```

```
## [1] "elem1" "elem2" "elem3"
```

```r
## Renomear
names(li) <- paste0("elemento0", 1:3)
li
```

```
## $elemento01
## [1] 1
##
## $elemento02
## [1] 2
##
## $elemento03
## [1] 3
```

7.4.3 ManipulaÃ§Ã£o de objetos multidimensionais

Da mesma forma que para objetos unidimensionais, podemos manipular elementos de objetos multidimensionais, ou seja, matrizes, data frames e arrays.

Novamente, a primeira forma de manipulaÃ§Ã£o Ã© atravÃ©s da indexaÃ§Ã£o, utilizando os operadores `[]`. Com a indexaÃ§Ã£o podemos acessar elementos de matrizes, data frames e arrays por sua posiÃ§Ã£o. Podemos ainda retirar elementos dessas estruturas com o operador aritmÃ©tico `-`.

Entretanto, agora temos mais de uma dimensÃ£o na estruturaÃ§Ã£o dos elementos dentro dos objetos. Assim, utilizamos nÃºmeros, sequÃªncia de nÃºmeros ou operaÃ§Ã£o booleanas para retornar partes desses objetos, mas as dimensÃµes tÃªm de ser explicitadas e separadas por **vÃrgulas** para acessar linhas e colunas. Essa indexaÃ§Ã£o funciona para matrizes e data frames. Para arrays, especificamos tambÃ©m as dimensÃµes, tambÃ©m separadas por vÃrgulas para acessar essas dimensÃµes.

```r
## Matriz
ma <- matrix(1:12, 4, 3)
ma
```

```
##      [,1] [,2] [,3]
## [1,]    1    5    9
## [2,]    2    6   10
## [3,]    3    7   11
## [4,]    4    8   12
```

```
## IndexaÃ§Ã£o
ma[3, ] # linha 3
```

```
## [1]  3  7 11
```

```
ma[, 2] # coluna 2
```

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

```
ma[1, 2] # elemento da linha 1 e coluna 2
```

```
## [1] 5
```

```
ma[1, 1:2] # elementos da linha 1 e coluna 1 e 2
```

```
## [1] 1 5
```

```
ma[1, c(1, 3)] # elementos da linha 1 e coluna 1 e 3
```

```
## [1] 1 9
```

```
ma[-1, ] # retirar a linha 1
```

```
##      [,1] [,2] [,3]
## [1,]    2    6   10
## [2,]    3    7   11
## [3,]    4    8   12
```

```
ma[, -3] # retirar a coluna 3
```

```
##      [,1] [,2]
## [1,]    1    5
## [2,]    2    6
## [3,]    3    7
## [4,]    4    8
```

Para data frames, alÃ©m de utilizar nÃºmeros e/ou sequÃªncias de nÃºmeros dentro do operador `[]` simples, podemos utilizar o operador `[[]]` duplo para retornar apenas os valores de uma linha ou uma coluna. Se as colunas estiverem nomeadas, podemos utilizar o nome da coluna de interesse entre aspas dentro dos operadores `[]` (retornar coluna) e `[[]]` (retornar apenas os valores), assim como ainda podemos utilizar o operador `$` para data frames. Essas Ãºltimas operaÃ§Ãµes retornam um vetor, para o qual podemos fazer operaÃ§Ãµes de vetores ou ainda atualizar o valor dessa coluna selecionada ou adicionar outra coluna.

```
## Criar trÃªs vetores
sp <- paste("sp", 1:10, sep = "")
abu <- 1:10
flo <- factor(rep(c("campo", "floresta"), each = 5))

## data frame
df <- data.frame(sp, abu, flo)
df
```

```
##      sp abu      flo
## 1   sp1   1    campo
## 2   sp2   2    campo
```

```
## 3   sp3   3    campo
## 4   sp4   4    campo
## 5   sp5   5    campo
## 6   sp6   6 floresta
## 7   sp7   7 floresta
## 8   sp8   8 floresta
## 9   sp9   9 floresta
## 10 sp10  10 floresta
```

```r
## [] - nÃºmeros
df[, 1]
```

```
##  [1] "sp1"  "sp2"  "sp3"  "sp4"  "sp5"  "sp6"  "sp7"  "sp8"  "sp9"  "sp10"
```

```r
## [] - nome das colunas - retorna coluna
df["flo"]
```

```
##         flo
## 1     campo
## 2     campo
## 3     campo
## 4     campo
## 5     campo
## 6  floresta
## 7  floresta
## 8  floresta
## 9  floresta
## 10 floresta
```

```r
## [[]] - nome das colunas - retorna apenas os valores
df[["flo"]]
```

```
##  [1] campo    campo    campo    campo    campo    floresta floresta floresta
##  [9] floresta floresta
## Levels: campo floresta
```

```r
## $ funciona apenas para data frame
df$sp
```

```
##  [1] "sp1"  "sp2"  "sp3"  "sp4"  "sp5"  "sp6"  "sp7"  "sp8"  "sp9"  "sp10"
```

```r
## OperaÃ§Ã£o de vetors
length(df$abu)
```

```
## [1] 10
```

```r
## Converter colunas
df$abu <- as.character(df$abu)
mode(df$abu)
```

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

```r
## Adicionar ou mudar colunas
set.seed(42)
df$abu2 <- sample(x = 0:1, size = nrow(df), rep = TRUE)
df
```

```
##      sp abu      flo abu2
## 1   sp1   1    campo    0
## 2   sp2   2    campo    0
```

```
## 3   sp3   3    campo    0
## 4   sp4   4    campo    0
## 5   sp5   5    campo    1
## 6   sp6   6 floresta    1
## 7   sp7   7 floresta    1
## 8   sp8   8 floresta    1
## 9   sp9   9 floresta    0
## 10 sp10  10 floresta    1
```

Podemos ainda fazer seleÃ§Ãµes condicionais para retornar linhas com valores que temos interesse, semelhante ao uso de filtro de uma planilha eletrÃ´nica.

```
## Selecionar linhas de uma matriz ou data frame
df[df$abu > 4, ]
```

```
##    sp abu      flo abu2
## 5 sp5   5    campo    1
## 6 sp6   6 floresta    1
## 7 sp7   7 floresta    1
## 8 sp8   8 floresta    1
## 9 sp9   9 floresta    0
```

```
df[df$flo == "floresta", ]
```

```
##      sp abu      flo abu2
## 6   sp6   6 floresta    1
## 7   sp7   7 floresta    1
## 8   sp8   8 floresta    1
## 9   sp9   9 floresta    0
## 10 sp10  10 floresta    1
```

AlÃ©m disso, hÃ¡ uma sÃ©rie de funÃ§Ãµes para conferÃªncia e manipulaÃ§Ã£o de dados que listamos na Tabela seguinte:

Table 3: FunÃ§Ãµes para verificaÃ§Ã£o e resumo de dados multidimensionais.

FunÃ§Ã£o	DescriÃ§Ã£o
‘head()‘	Mostra as primeiras 6 linhas
‘tail()‘	Mostra as Ãºltimas 6 linhas
‘nrow()‘	Mostra o nÃºmero de linhas
‘ncol()‘	Mostra o nÃºmero de colunas
‘dim()‘	Mostra o nÃºmero de linhas e de colunas
‘rownames()‘	Mostra os nomes das linhas (locais)
‘colnames()‘	Mostra os nomes das colunas (variÃ¡veis)
‘str()‘	Mostra as classes de cada coluna (estrutura)
‘summary()‘	Mostra um resumo dos valores de cada coluna
‘rowSums()‘	Calcula a soma das linhas (horizontal)
‘colSums()‘	Calcula a soma das colunas (vertical)
‘rowMeans()‘	Calcula a mÃ©dia das linhas (horizontal)
‘colMeans()‘	Calcula a mÃ©dia das colunas (vertical)
‘table()‘	TabulaÃ§Ã£o cruzada
‘t()‘	Matriz ou data frame transposto

7.4.4 Valores faltantes e especiais

Valores faltantes e especiais sÃ£o valores reservados que representam dados faltantes, indefiniÃ§Ãµes matemÃ¡ticas, infinitos e objetos nulos.

1. **NA (*Not Available*)**: significa dado faltante ou indisponÃvel
2. **NaN (*Not a Number*)**: representa indefiniÃ§Ãµes matemÃ¡ticas
3. **Inf (*Infinito*)**: Ã© um nÃºmero muito grande ou um limite matemÃ¡tico
4. **NULL *(Nulo)***: representa um objeto nulo, sendo Ãºtil para preenchimento em aplicaÃ§Ãµes de programaÃ§Ã£o

```r
## Data frame com elemento NA
df <- data.frame(var1 = c(1, 4, 2, NA), var2 = c(1, 4, 5, 2))
df
```

```
##   var1 var2
## 1    1    1
## 2    4    4
## 3    2    5
## 4   NA    2
```

```r
## Resposta booleana para elementos NA
is.na(df)
```

```
##       var1  var2
## [1,] FALSE FALSE
## [2,] FALSE FALSE
## [3,] FALSE FALSE
## [4,]  TRUE FALSE
```

```r
## Algum elemento Ã© NA?
any(is.na(df))
```

```
## [1] TRUE
```

```r
## Remover as linhas com NAs
df_sem_na <- na.omit(df)
df_sem_na
```

```
##   var1 var2
## 1    1    1
## 2    4    4
## 3    2    5
```

```r
## Substituir NAs por 0
df[is.na(df)] <- 0
df
```

```
##   var1 var2
## 1    1    1
## 2    4    4
## 3    2    5
## 4    0    2
```

```r
## Desconsiderar os NAs em funÃ§Ãµes com o argumento rm.na = TRUE
sum(1, 2, 3, 4, NA, na.rm = TRUE)
```

```
## [1] 10
```

```r
## NaN - not a number
0/0
```

```
## [1] NaN
```

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

```
## Warning in log(-1): NaNs produced
```

```
## [1] NaN
```

```r
## Limite matemÃ¡tico
1/0
```

```
## [1] Inf
```

```r
## NÃºmero grande
10^310
```

```
## [1] Inf
```

```r
## Objeto nulo
nulo <- NULL
nulo
```

```
## NULL
```

7.4.5 DiretÃ³rio de trabalho

O diretÃ³rio de trabalho Ã© o endereÃ§o da pasta (ou diretÃ³rio) de onde o R importarÃ¡ ou exportar nossos dados.

Podemos utilizar o prÃ³prio RStudio para tal tarefa, indo em `Session > Set Work Directory > Choose Directory...` ou simplesmente utilizar o atalho `Ctrl + Shift + H`.

Podemos ainda utilizar funÃ§Ãµes do R para definir o diretÃ³rio. Para tanto, podemos navegar com o aplicativo de gerenciador de arquivos (e.g., Windows Explorer) atÃ© nosso diretÃ³rio de interesse e copiar o endereÃ§o na barra superior. Voltamos para o R e colamos esse endereÃ§o entre aspas como argumento da funÃ§Ã£o `setwd()`. Ã fundamental destacar que no Windows Ã© necessÃ¡rio inverter as barras (\ por / ou duplicar elas \\).

Aconselhamos ainda utilizar as funÃ§Ãµes `getwd()` para retornar o diretÃ³rio definido na sessÃ£o do R, assim como as funÃ§Ãµes `dir()` ou `list.files()` para listagem dos arquivos no diretÃ³rio, ambas medidas de conferÃªncia do diretÃ³rio correto.

```r
## Definir o diretÃ³rio de trabalho
setwd("/home/mude/data/github/livro_aer/dados")

## Verificar o diretÃ³rio
getwd()

## Listar os arquivos no diretÃ³rio
dir()
list.files()
```

7.4.6 Importar dados

Uma das operaÃ§Ãµes mais corriqueiras do R, antes de realizar alguma anÃ¡lise ou plotar um grÃ¡fico, Ã© a de importar dados que foram tabulados numa planilha eletrÃ´nica e salvos no formato .csv, .txt ou .xlsx. Ao importar esse tipo de dado para o R, o formato que o mesmo assume, se nenhum parÃ¢metro for

especificado, Ã© o da classe `data frame`, prevendo que a planilha de dados possua colunas com diferentes modos.

Existem diversas formas de importar dados para o R. Podemos importar utilizando o RStudio, indo na janela **Environment** e clicar em "Importar Dataset".

Entretanto, aconselhamos o uso de funÃ§Ãµes que fiquem salvas em um script para aumentar a reprodutibilidade do mesmo. Dessa forma, as trÃªs principais funÃ§Ãµes para importar os arquivos nos trÃªs principais extensÃµes (.csv, .txt ou .xlsx) sÃ£o, respectivamente: `read.csv()`, `read.table()` e `openxlsx::read.xlsx()`, sendo o Ãºltimo do pacote `openxlsx`.

Para exemplificar como importar dados no R, vamos usar os dados de pinguins no Palmer Station, Antarctica Palmer Penguins. Faremos o download diretamente do site da fonte dos dados.

Vamos antes escolher um diretÃ³rio de trabalho com a funÃ§Ã£o `setwd()`, e em seguida criar um diretÃ³rio com a funÃ§Ã£o `dir.create()` chamado "dados". Em seguida, vamos mudar nosso diretÃ³rio para essa pasta e criar mais um diretÃ³rio chamado "tabelas", e por fim, definir esse diretÃ³rio para que o conteÃºdo do download seja armazenado ali.

```
## Escolher um diretÃ³rio
setwd("/home/mude/data/github/livro_aer")

## Criar um diretÃ³rio 'dados'
dir.create("dados")

## Escolher diretÃ³rio 'dados'
setwd("dados")

## Criar um diretÃ³rio 'tabelas'
dir.create("tabelas")

## Escolher diretÃ³rio 'tabelas'
setwd("tabelas")
```

Agora podemos fazer o download do arquivo `.csv` e importar a tabela de dados.

```
## Download
download.file(url = "https://github.com/allisonhorst/palmerpenguins/blob/main/inst/extdata/penguins_raw
              destfile = "penguins_raw.csv", method = "curl")
```

Agora podemos importar a tabela de dados com a funÃ§Ã£o `read.csv()`, atribuindo ao objeto `penguins_raw`.

```
## Importar a tabela
pinguins_raw <- read.csv("dados/tabelas/penguins_raw.csv")
```

Esse arquivo foi criado com separador de decimais sendo `.` e separador de colunas sendo `,`. Caso tivesse sido criado com separador de decimais sendo `,` e separador de colunas sendo `;`, usarÃamos a funÃ§Ã£o `read.csv2()`.

Para outros formatos, basta usar as outras funÃ§Ãµes apresentadas, atentando-se para os argumentos especÃficos de cada funÃ§Ã£o.

Caso o download nÃ£o funcione ou haja problemas com a importaÃ§Ã£o, os dados estao tambÃ©m no pacote `palmerpenguins`.

```
## Importar os dados pelo pacote palmerpenguins
pinguins_raw <- palmerpenguins::penguins_raw
head(pinguins_raw)
```

```
## # A tibble: 6 x 17
##   studyName `Sample Number` Species              Region Island Stage `Individual ID`
##   <chr>               <dbl> <chr>                <chr>  <chr>  <chr> <chr>
## 1 PAL0708                 1 Adelie Penguin (Pyg~ Anvers Torge~ Adul~ N1A1
## 2 PAL0708                 2 Adelie Penguin (Pyg~ Anvers Torge~ Adul~ N1A2
## 3 PAL0708                 3 Adelie Penguin (Pyg~ Anvers Torge~ Adul~ N2A1
## 4 PAL0708                 4 Adelie Penguin (Pyg~ Anvers Torge~ Adul~ N2A2
## 5 PAL0708                 5 Adelie Penguin (Pyg~ Anvers Torge~ Adul~ N3A1
## 6 PAL0708                 6 Adelie Penguin (Pyg~ Anvers Torge~ Adul~ N3A2
## # i 10 more variables: `Clutch Completion` <chr>, `Date Egg` <date>,
## #   `Culmen Length (mm)` <dbl>, `Culmen Depth (mm)` <dbl>,
## #   `Flipper Length (mm)` <dbl>, `Body Mass (g)` <dbl>, Sex <chr>,
## #   `Delta 15 N (o/oo)` <dbl>, `Delta 13 C (o/oo)` <dbl>, Comments <chr>
```

7.4.7 ConferÃªncia dos dados importados

Uma vez importados os dados para o R, geralmente antes de iniciarmos qualquer manipulaÃ§Ã£o, visualizaÃ§Ã£o ou anÃ¡lise de dados, fazemos a conferÃªncia desses dados. Dentre todas as funÃ§Ãµes de verificaÃ§Ã£o, destacamos a importÃ¢ncia destas funÃ§Ãµes apresentadas abaixo para saber se as variÃ¡veis foram importadas e interpretadas corretamente e reconhecer erros de digitaÃ§Ã£o, por exemplo:

```r
pinguins <- palmerpenguins::penguins
## Primeiras linhas
head(pinguins)
```

```
## # A tibble: 6 x 8
##   species island    bill_length_mm bill_depth_mm flipper_length_mm body_mass_g sex
##   <fct>   <fct>              <dbl>         <dbl>             <int>       <int> <fct>
## 1 Adelie  Torgersen           39.1          18.7               181        3750 male
## 2 Adelie  Torgersen           39.5          17.4               186        3800 fema~
## 3 Adelie  Torgersen           40.3          18                 195        3250 fema~
## 4 Adelie  Torgersen           NA            NA                  NA          NA <NA>
## 5 Adelie  Torgersen           36.7          19.3               193        3450 fema~
## 6 Adelie  Torgersen           39.3          20.6               190        3650 male
## # i 1 more variable: year <int>
```

```r
## Ãltimas linhas
tail(pinguins)
```

```
## # A tibble: 6 x 8
##   species   island bill_length_mm bill_depth_mm flipper_length_mm body_mass_g sex
##   <fct>     <fct>           <dbl>         <dbl>             <int>       <int> <fct>
## 1 Chinstrap Dream            45.7          17                 195        3650 female
## 2 Chinstrap Dream            55.8          19.8               207        4000 male
## 3 Chinstrap Dream            43.5          18.1               202        3400 female
## 4 Chinstrap Dream            49.6          18.2               193        3775 male
## 5 Chinstrap Dream            50.8          19                 210        4100 male
## 6 Chinstrap Dream            50.2          18.7               198        3775 female
## # i 1 more variable: year <int>
```

```r
## NÃºmero de linhas e colunas
nrow(pinguins)
```

```
## [1] 344
```

```r
ncol(pinguins)
```

```
## [1] 8
```

```r
dim(pinguins)
```

```
## [1] 344   8
```

```r
## Nome das linhas e colunas
rownames(pinguins)
colnames(pinguins)

## Estrutura dos dados
str(pinguins)

## Resumo dos dados
summary(pinguins)

## Verificar NAs
any(is.na(pinguins))
which(is.na(pinguins))

## Remover as linhas com NAs
pinguins_na <- na.omit(pinguins)
```

Importante
A funÃ§Ã£o `na.omit()` retira a **linha inteira** que possui algum `NA`, inclusive as colunas que possuem dados que vocÃª nÃ£o tem interesse em excluir. Dessa forma, tenha em mente quais dados vocÃª realmente quer remover da sua tabela.

AlÃ©m das funÃ§Ãµes apresentadas, recomendamos olhar os seguintes pacotes que ajudam na conferÃªncia dos dados importados: `Hmisc`, `skimr` e `inspectDF`.

7.4.8 Exportar dados

Uma vez realizado as operaÃ§Ãµes de manipulaÃ§Ã£o ou tendo dados que foram analisados e armazenados num objeto no formato de data frame ou matriz, podemos exportar esses dados do R para o diretÃ³rio que definimos anteriormente.

Para tanto, podemos utilizar funÃ§Ãµes de escrita de dados, como `write.csv()`, `write.table()` e `openxlsx::write.xlsx()`. Dois pontos sÃ£o fundamentais: i) o nome do arquivo tem de estar entre aspas e no final dele deve constar a extensÃ£o que pretendemos que o arquivo tenha, e ii) Ã© interessante utilizar os argumentos **row.names = FALSE** e **quote=FALSE**, para que o arquivo escrito nÃ£o tenha o nome das linhas ou aspas em todas as cÃ©lulas, respectivamente.

```r
## Exportar dados na extensÃ£o .csv
write.csv(pinguins_na, "pinguins_na.csv",
          row.names = FALSE, quote = FALSE)

## Exportar dados na extensÃ£o .txt
write.table(pinguins_na, "pinguins_na.txt",
            row.names = FALSE, quote = FALSE)

## Exportar dados na extensÃ£o .xlsx
openxlsx::write.xlsx(pinguins_na, "pinguins_na.xlsx",
                     row.names = FALSE, quote = FALSE)
```

7.5 Para se aprofundar

Listamos a seguir livros e links com material que recomendamos para seguir com sua aprendizagem em *R Base*.

7.5.1 Livros

Recomendamos aos (Ã s) interessados(as) os livros: i) Crawley The R Book, ii) Davies The Book of R: A First Course in Programming and Statistics, iii) Gillespie e Lovelace Efficient R programming, iv) Holmes e Huber Modern Statistics for Modern Biology, v) Irizarry e Love Data Analysis for the Life Sciences with R, vi) James e colaboradores An Introduction to Statistical Learning: with Applications in R, vii) Kabacoff R in Action: Data analysis and graphics with R, viii) Matloff The Art of R Programming: A Tour of Statistical Software Design, ix) Long e Teetor R Cookbook e x) Wickham Advanced R.

7.5.2 Links

Existem centenas de ferramentas online para aprender e explorar o R. Dentre elas, indicamos os seguintes links (em portuguÃªs e inglÃªs):

IntroduÃ§Ã£o ao R

- An Introduction to R - Douglas A, Roos D, Mancini F, Couto A, Lusseau D
- A (very) shortintroduction to R - Paul Torfs & Claudia Brauer
- R for Beginners - Emmanuel Paradis

CiÃªncia de dados - CiÃªncia de Dados em R - Curso-R - Data Science for Ecologists and Environmental Scientists - Coding Club

EstatÃstica - EstatÃstica Computacional com R - Mayer F. P., Bonat W. H., Zeviani W. M., Krainski E. T., Ribeiro Jr. P. J - Data Analysis and Visualization in R for Ecologists - Data Carpentry

MiscelÃ¢nea

- Materiais sobre R - Beatriz Milz
- R resources (free courses, books, tutorials, & cheat sheets) - Paul van der Laken

7.6 ExercÃcios

4.1 Use o R para verificar o resultado da operaÃ§Ã£o `7 + 7 Ã· 7 + 7 x 7 - 7`.

4.2 Verifique atravÃ©s do R se `3x2Ã³` Ã© maior que `2x3Ã²`.

4.3 Crie dois objetos (qualquer nome) com os valores 100 e 300. Multiplique esses objetos (funÃ§Ã£o `prod()`) e atribuam ao objeto `mult`. FaÃ§a o logaritmo natural (funÃ§Ã£o `log()`) do objeto `mult` e atribuam ao objeto `ln`.

4.4 Quantos pacotes existem no CRAN nesse momento? Execute essa combinaÃ§Ã£o no Console: `nrow(available.packages(repos = "http://cran.r-project.org"))`.

4.5 Instale o pacote `tidyverse` do CRAN.

4.6 Escolha nÃºmeros para jogar na mega-sena usando o R, nomeando o objeto como `mega`. Lembrando: sÃ£o 6 valores de 1 a 60 e atribuam a um objeto.

4.7 Crie um fator chamado `tr`, com dois nÃveis ("cont" e "trat") para descrever 30 locais de amostragem, 15 de cada tratamento. O fator deve ser dessa forma `cont, cont, cont, ...., cont, trat, trat, ....,` `trat`.

4.8 Crie uma matriz chamada `ma`, resultante da disposiÃ§Ã£o de um vetor composto por 300 valores aleatÃ³rios entre 0 e 10. A matriz deve conter 30 linhas e ser disposta por colunas.

4.9 Crie um data frame chamado `df`, resultante da composiÃ§Ã£o desses vetores:

- `id: 1:30`
- `sp: sp01, sp02, ..., sp29, sp30`
- `ab: 30 valores aleatÃ³rios entre 0 a 5`

4.10 Crie uma lista com os objetos criados anteriormente: `mega`, `tr`, `ma` e `df`.

4.11 Selecione os elementos Ãmpares do objeto `tr` e atribua ao objeto `tr_impar`.

4.12 Selecione as linhas com ids pares do objeto `df` e atribua ao objeto `df_ids_par`.

4.13 FaÃ§a uma amostragem de 10 linhas do objeto `df` e atribua ao objeto `df_amos10`. Use a funÃ§Ã£o `set.seed()` para fixar a amostragem.

4.14 Amostre 10 linhas do objeto `ma`, mas utilizando as linhas amostradas do `df_amos10` e atribua ao objeto `ma_amos10`.

4.15 Una as colunas dos objetos `df_amos10` e `ma_amos10` e atribua ao objeto `dados_amos10`.

SoluÃ§Ãµes dos exercÃcios.

8 Tidyverse

Conteudo copiado, com pequenas alteraÃ§Ãµes e atualizaÃ§Ãµes de CapÃtulo 5 Tidyverse, do livro AnÃ¡lises EcolÃ³gicas no R . O livro estÃ¡ licenciado sob a licenÃ§a BY-NC-ND 4.0 : https://creativecommons.org/licenses/by-nc-nd/4.0/.

PrÃ©-requisitos do capÃtulo

Pacotes e dados que serÃ£o utilizados neste capÃtulo.

```
## Pacotes
library(tidyverse)
library(ggplot2)
library(purrr)
library(tibble)
library(dplyr)
library(tidyr)
library(stringr)
library(readr)
library(forcats)
library(palmerpenguins)
library(lubridate)

## Dados
penguins <- palmerpenguins::penguins
penguins_raw <- palmerpenguins::penguins_raw
#tidy_anfibios_locais <- ecodados::tidy_anfibios_locais
```

8.1 ContextualizaÃ§Ã£o

Como todo idioma, a linguagem R tem "dialetos" e "sotaques" diversos. Ademais, R vem passando por transformaÃ§Ãµes nos Ãºltimos anos. Grande parte dessas mudanÃ§as estÃ£o dentro do paradigma de CiÃªncia de Dados (*Data Science*), uma nova Ã¡rea de conhecimento que vem se moldando a partir do desenvolvimento da sociedade em torno da era digital e da grande quantidade de dados gerados e disponÃveis pela internet, de onde advÃ©m os pilares das inovaÃ§Ãµes tecnolÃ³gicas: *Big Data*, *Machine Learning* e *Internet of Things*. A grande necessidade de computaÃ§Ã£o para desenvolver esse novo paradigma colocaram linguagens de programaÃ§Ã£o interpretadacomo como R R como as principais linguagens frente a esses novos desafios. Linguagens de programaÃ§Ã£o de cÃ³digo aberto sÃ£o uma Ã³tima escolha para diversos propÃ³sitos, pois sÃ£o gratuitas, possuem grandes comunidades de desenvolvedores, e sÃ£o relativamente fÃ¡ceis de aprender e aplicar.

As expansÃµes na utilizaÃ§Ã£o da linguagem R para a CiÃªncia de Dados comeÃ§aram a ser implementadas principalmente devido a um pesquisador: Hadley Wickham, que iniciou sua contribuiÃ§Ã£o Ã comunidade R com o desenvolvimento do pacote `ggplot2` [@wickham2016] para a composiÃ§Ã£o de grÃ¡ficos no R, baseado na gramÃ¡tica de grÃ¡ficos [@wilkinson2005]. Depois disso, Wickham dedicou-se ao desenvolvimento do pensamento de uma nova abordagem dentro da manipulaÃ§Ã£o de dados, denominada **Tidy Data** (Dados organizados) [@wickham2014], na qual focou na limpeza e organizaÃ§Ã£o dos mesmos. A ideia postula que dados estÃ£o `tidy` quando: i) variÃ¡veis estÃ£o nas colunas, ii) observaÃ§Ãµes estÃ£o nas linhas e iii) valores estÃ£o nas cÃ©lulas, sendo que para esse Ãºltimo, nÃ£o deve haver mais de um valor por cÃ©lula.

A partir dessas ideias, o dialeto *tidyverse* foi operacionalizado no R como uma coleÃ§Ã£o de pacotes que atuam no fluxo de trabalho comum da ciÃªncia de dados: importaÃ§Ã£o, manipulaÃ§Ã£o, exploraÃ§Ã£o, visualizaÃ§Ã£o, anÃ¡lise e comunicaÃ§Ã£o de dados e anÃ¡lises [@tidyverse2019] . O principal objetivo do *tidyverse* Ã© aproximar a "linguagem R" para melhorar a interaÃ§Ã£o entre ser humano e computador sobre dados, de modo que os pacotes compartilham uma filosofia de design de alto nÃvel e gramÃ¡tica,

alÃ©m da estrutura de dados de baixo nÃvel [@tidyverse2019]. As principais leituras sobre o tema no R sÃ£o os artigos "Tidy Data" [@wickham2014] e "Welcome to the Tidyverse" [@tidyverse2019], e o livro "R for Data Science" [@wickham2017], alÃ©m do Tidyverse que possui muito mais informaÃ§Ãµes.

Figura 8.1: Modelo das ferramentas necessÃ¡rias em um projeto tÃpico de ciÃªncia de dados: importar, organizar, entender (transformar, visualizar, modelar) e comunicar, envolto Ã essas ferramentas estÃ¡ a programaÃ§Ã£o.

8.2 *tidyverse*

Uma vez instalado e carregado, o pacote `tidyverse` disponibiliza um conjunto de ferramentas atravÃ©s de vÃ¡rios pacotes. Esses pacotes compartilham uma filosofia de design, gramÃ¡tica e estruturas. Podemos entender o *tidyverse* como um "dialeto novo" para a linguagem R, onde *tidy* quer dizer organizado, arrumado, ordenado, e *verse* Ã© universo. A seguir, listamos os principais pacotes e suas funcionalidades.

- `readr`: importa dados tabulares (e.g. `.csv` e `.txt`)
- `tibble`: implementa a classe `tibble`
- `tidyr`: transformaÃ§Ã£o de dados para `tidy`
- `dplyr`: manipulaÃ§Ã£o de dados
- `stringr`: manipulaÃ§Ã£o de caracteres
- `forcats`: manipulaÃ§Ã£o de fatores
- `ggplot2`: possibilita a visualizaÃ§Ã£o de dados
- `purrr`: disponibiliza ferramentas para programaÃ§Ã£o funcional

AlÃ©m dos pacotes principais, fazemos tambÃ©m menÃ§Ã£o a outros pacotes que estÃ£o dentro dessa abordagem e que trataremos ainda neste capÃtulo, em outro momento do livro, ou que vocÃª leitor(a) deve se familiarizar. Alguns pacotes compÃµem o tidyverse outros sÃ£o mais gerais, entretanto, todos estÃ£o envolvidos de alguma forma com ciÃªncia de dados.

- `readxl` e `writexl`: importa e exporta dados tabulares (.xlsx)
- `janitor`: examina e limpa dados sujos
- `DBI`: interface de banco de dados R
- `haven`: importa e exporta dados do SPSS, Stata e SAS
- `httr`: ferramentas para trabalhar com URLs e HTTP
- `rvest`: coleta facilmente (raspagem de dados) pÃ¡ginas da web
- `xml2`: trabalha com arquivos XML
- `jsonlite`: um analisador e gerador JSON simples e robusto para R
- `hms`: hora do dia
- `lubridate`: facilita o tratamento de datas
- `magrittr`: provÃª os operadores pipe (`%>%`, `%$%`, `%<>%`)
- `glue`: facilita a combinaÃ§Ã£o de dados e caracteres
- `rmarkdown`: cria documentos de anÃ¡lise dinÃ¢mica que combinam cÃ³digo, saÃda renderizada (como figuras) e texto

- `knitr`: projetado para ser um mecanismo transparente para geraÃ§Ã£o de relatÃ³rios dinÃ¢micos com R
- `shiny`: framework de aplicativo Web para R
- `flexdashboard`: painÃ©is interativos para R
- `here`: facilita a definiÃ§Ã£o de diretÃ³rios
- `usethis`: automatiza tarefas durante a configuraÃ§Ã£o e desenvolvimento de projetos (Git, 'GitHub' e Projetos RStudio)
- `data.table`: pacote que fornece uma versÃ£o de alto desempenho do `data.frame` (importar, manipular e expotar)
- `reticulate`: pacote que fornece ferramentas para integrar Python e R
- `sparklyr`: interface R para Apache Spark
- `broom`: converte objetos estatÃsticos em tibbles organizados
- `modelr`: funÃ§Ãµes de modelagem que funcionam com o pipe
- `tidymodels`: coleÃ§Ã£o de pacotes para modelagem e aprendizado de mÃ¡quina usando os princÃpios do tidyverse

Destacamos a grande expansÃ£o e aplicabilidade dos pacotes rmarkdown, knitr e bookdown, que permitiram a escrita deste livro usando essas ferramentas e linguagem de marcaÃ§Ã£o, chamada Markdown.

Para instalar os principais pacotes que integram o *tidyverse* podemos instalar o pacote `tidyverse`.

```
## Instalar o pacote tidyverse
install.packages("tidyverse")
```

Quando carregamos o pacote `tidyverse` podemos notar uma mensagem indicando quais pacotes foram carregados, suas respectivas versÃµes e os conflitos com outros pacotes.

```
## Carregar o pacote tidyverse
library(tidyverse)
```

Podemos ainda listar todos os pacotes do *tidyverse* com a funÃ§Ã£o `tidyverse::tidyverse_packages()`.

```
## Listar todos os pacotes do tidyverse
tidyverse::tidyverse_packages()
```

```
##  [1] "broom"         "conflicted"    "cli"           "dbplyr"        "dplyr"
##  [6] "dtplyr"        "forcats"       "ggplot2"       "googledrive"   "googlesheets4"
## [11] "haven"         "hms"           "httr"          "jsonlite"      "lubridate"
## [16] "magrittr"      "modelr"        "pillar"        "purrr"         "ragg"
## [21] "readr"         "readxl"        "reprex"        "rlang"         "rstudioapi"
## [26] "rvest"         "stringr"       "tibble"        "tidyr"         "xml2"
## [31] "tidyverse"
```

TambÃ©m podemos verificar se os pacotes estÃ£o atualizados, senÃ£o, podemos atualizÃ¡-los com a funÃ§Ã£o `tidyverse::tidyverse_update()`.

```
## Verificar e atualizar os pacotes do tidyverse
tidyverse::tidyverse_update(repos = "http://cran.us.r-project.org")
```

Todas as funÃ§Ãµes dos pacotes tidyverse usam **fonte minÃºscula** e **_ (*underscore*)** para separar os nomes internos das funÃ§Ãµes, seguindo a mesma sintaxe do Python ("Snake Case"). Neste sentido de padronizaÃ§Ã£o, Ã© importante destacar ainda que existe um guia prÃ³prio para que os scripts sigam a recomendaÃ§Ã£o de padronizaÃ§Ã£o, o The tidyverse style guide, criado pelo prÃ³prio Hadley Wickham. Para pessoas que desenvolvem funÃ§Ãµes e pacotes existe o Tidyverse design guide criado pelo *Tidyverse team*.

```
## FunÃ§Ãµes no formato snake case
read_csv()
read_tsv()
```

```
as_tibble()
left_join()
group_by()
```

Por fim, para evitar possÃveis conflitos de funÃ§Ãµes com o mesmo nome entre pacotes, recomendamos fortemente o hÃ¡bito de usar as funÃ§Ãµes precedidas do operador `::` e o respectivo pacote. Assim, garante-se que a funÃ§Ã£o utilizada Ã© referente ao pacote daquela funÃ§Ã£o. Segue um exemplo com as funÃ§Ãµes apresentadas anteriormente.

```
## FunÃ§Ãµes seguidas de seus respectivos pacotes
readr::read_csv()
readr::read_tsv()
tibble::as_tibble()
dplyr::left_join()
dplyr::group_by()
```

Seguindo essas ideias do novo paradigma da **CiÃªncia de Dados**, outro conjunto de pacotes foi desenvolvido, chamado de `tidymodels` que atuam no fluxo de trabalho da anÃ¡lise de dados em ciÃªncia de dados: separaÃ§Ã£o e reamostragem, prÃ©-processamento, ajuste de modelos e mÃ©tricas de performasse de ajustes. Por razÃµes de espaÃ§o e especificidade, nÃ£o entraremos em detalhes desses pacotes.

Um projeto de ciÃªncia de dados envolve uma sequÃªncia de passos:

1. Importar os dados
2. Organizar os dados
3. Entender os dados (transformar, visualizar, modelar)
4. Comunicar os resultados Nas prÃ³ximas seÃ§Ãµes, veremos como cada um desses passos pode ser realizado com funÃ§Ãµes de pacotes especÃficos.

8.3 readr, readxl e writexl

Dado que possuÃmos um conjunto de dados e que geralmente esse conjunto de dados estarÃ¡ no formato tabular com umas das extensÃµes: .csv, .txt ou .xlsx, usaremos o pacote `readr` ou `readxl` para importar esses dados para o R. Esses pacotes leem e escrevem grandes arquivos de forma mais rÃ¡pida, alÃ©m de fornecerem medidores de progresso de importaÃ§Ã£o e exportaÃ§Ã£o, e imprimir a informaÃ§Ã£o dos modos das colunas no momento da importaÃ§Ã£o. Outro ponto bastante positivo Ã© que tambÃ©m classificam automaticamente o modo dos dados de cada coluna, i.e., se uma coluna possui dados numÃ©ricos ou apenas texto, essa informaÃ§Ã£o serÃ¡ considerada para classificar o modo da coluna toda. A classe do objeto atribuÃdo quando lido por esses pacotes Ã© automaticamente um `tibble`, que veremos melhor na seÃ§Ã£o seguinte. Todas as funÃ§Ãµes deste pacote sÃ£o listadas na pÃ¡gina de referÃªncia do pacote.

Usamos as funÃ§Ãµes `readr::read_csv()` e `readr::write_csv()` para importar e exportar arquivos .csv do R, respectivamente. Para dados com a extensÃ£o .txt, podemos utilizar as funÃ§Ãµes `readr::read_tsv()` ou ainda `readr::read_delim()`. Para arquivos tabulares com a extensÃ£o .xlsx, temos de instalar e carregar dois pacotes adicionais: `readxl` e `writexl`, dos quais usaremos as funÃ§Ã£o `readxl::read_excel()`, para importar dados, atentado para o fato de podermos indicar a aba/planilha com os dados com o argumento `sheet`, e `writexl::write_xlsx()` para exportar.

Se o arquivo .csv foi criado com separador de decimais sendo `.` e separador de colunas sendo `,`, usamos as funÃ§Ãµes listadas acima normalmente. Caso seja criado com separador de decimais sendo `,` e separador de colunas sendo `;`, devemos usar a funÃ§Ã£o `readr::read_csv2()` para importar e `readr::write_csv2()` para exportar nesse formato, que Ã© mais comum no Brasil.

Para se aprofundar no tema, recomendamos a leitura do CapÃtulo 11 Data import de Wickham & Grolemund [-@wickham2017].

8.4 tibble

O `tibble` (`tbl_sf`) Ã© uma versÃ£o aprimorada do data frame (`data.frame`). Ele Ã© a classe aconselhada para que as funÃ§Ãµes do tidyverse funcionem melhor sobre conjuntos de dados tabulares importados para o R.

Geralmente, quando utilizamos funÃ§Ãµes tidyverse para importar dados para o R, Ã© essa classe que os dados adquirem depois de serem importados. AlÃ©m da importaÃ§Ã£o de dados, podemos criar um tibble no R usando a funÃ§Ã£o `tibble::tibble()`, semelhante ao uso da funÃ§Ã£o `data.frame()`. Podemos ainda converter um `data.frame` para um `tibble` usando a funÃ§Ã£o `tibble::as_tibble()`. Entretanto, em alguns momentos precisaremos da classe `data.frame` para algumas funÃ§Ãµes especÃficas, e podemos converter um `tibble` para `data.frame` usando a funÃ§Ã£o `tibble::as_data_frame()`.

Existem duas diferenÃ§as principais no uso do `tibble` e do `data.frame`: impressÃ£o e subconjunto. Objetos da classe `tibbles` possuem um mÃ©todo de impressÃ£o que mostra a contagem do nÃºmero de linhas e colunas, e apenas as primeiras 10 linhas e todas as colunas que couberem na tela no console, alÃ©m dos modos ou tipos das colunas. Dessa forma, cada coluna ou variÃ¡vel, pode ser do modo numbers (`int` ou `dbl`), character (`chr`), logical (`lgl`), factor (`fctr`), date + time (`dttm`) e date (`date`), alÃ©m de outras inÃºmeras possibilidades.

Todas as funÃ§Ãµes deste pacote sÃ£o listadas na pÃ¡gina de referÃªncia do pacote.

```
## Tibble - impressÃ£o
tidy_anfibios_locais
```

Para o subconjunto, como vimos no CapÃtulo 7, para selecionar colunas e linhas de objetos bidimensionais podemos utilizar os operadores `[]` ou `[[]]`, associado com nÃºmeros separados por vÃrgulas ou o nome da coluna entre aspas, e o operador `$` para extrair uma coluna pelo seu nome. Comparando um `data.frame` a um `tibble`, o Ãºltimo Ã© mais rÃgido na seleÃ§Ã£o das colunas: ele nunca faz correspondÃªncia parcial e gera um aviso se a coluna que vocÃª estÃ¡ tentando acessar nÃ£o existe.

```
## Tibble - subconjunto
tidy_anfibios_locais$ref
```

Por fim, podemos "espiar" os dados utilizando a funÃ§Ã£o `tibble::glimpse()` para ter uma noÃ§Ã£o geral de nÃºmero de linhas, colunas, e conteÃºdo de todas as colunas. Essa Ã© a funÃ§Ã£o *tidyverse* da funÃ§Ã£o *R Base* `str()`.

```
## Espiar os dados
tibble::glimpse(tidy_anfibios_locais[, 1:10])
```

Para se aprofundar no tema, recomendamos a leitura do CapÃtulo 10 Tibbles de Wickham & Grolemund [-@wickham2017].

8.5 pipe: base (|>) e magrittr (%>%)

O operador pipe permite o encadeamento de vÃ¡rias funÃ§Ãµes, eliminando a necessidade de criar objetos para armazenar resultados intermediÃ¡rios. Dessa forma, pipes sÃ£o uma ferramenta poderosa para expressar uma sequÃªncia de mÃºltiplas operaÃ§Ãµes.

Aqui usamos a versÃ£o nativo (|>), que vem na instqallaÃ§Ã£o, que Ã© mais "simples". O operador pipe original `%>%` vem do pacote `magrittr`. R 4.1.0 introduziu um operador de pipe nativo "|>". O comportamento do pipe nativo Ã© em geral o mesmo do pipe %>% fornecido pelo pacote magrittr. Ambos os operadores (|> e %>%) permitem â canalizarâ um objeto para uma funÃ§Ã£o ou expressÃ£o de chamada, permitindo assim expressar uma sequÃªncia de operaÃ§Ãµes que transformam um objeto.

Felizmente, nÃ£o hÃ¡ necessidade de se comprometer inteiramente com um pipe ou outro - vocÃª pode usar o pipe nativo na maioria dos casos, e usar o pipe magrittr quando realmente precisar de seus recursos especiais. O pipe torna os cÃ³digos em R mais simples, pois podemos realizar mÃºltiplas operaÃ§Ãµes em

uma Ãºnica linha. Ele captura o resultado de uma declaraÃ§Ã£o e o torna a primeira entrada da prÃ³xima declaraÃ§Ã£o, entÃ£o podemos pensar como "EM SEGUIDA FAÃ A" ao final de cada linha de cÃ³digo.

A principal vantagem do uso dos pipes Ã© facilitar a depuraÃ§Ã£o (*debugging* - achar erros) nos cÃ³digos, porque seu uso torna a linguagem R mais prÃ³xima do que falamos e pensamos, uma vez que evita o uso de funÃ§Ãµes dentro de funÃ§Ãµes (funÃ§Ãµes compostas, lembra-se do fog e gof do ensino mÃ©dio? Evitamos eles aqui tambÃ©m).

Para deixar esse tÃ³pico menos estranho a quem possa ver essa operaÃ§Ã£o pela primeira vez, vamos fazer alguns exemplos.

```
## R Base - sem pipe
sqrt(sum(1:100))
```

```
## [1] 71.06335
```

```
## com pipe
1:100 |>
    sum() |>
    sqrt()
```

```
## [1] 71.06335
```

Essas operaÃ§Ãµes ainda estÃ£o simples, vamos tornÃ¡-las mais complexas com vÃ¡rias funÃ§Ãµes compostas. Ã nesses casos que a propriedade organizacional do uso do pipe emerge: podemos facilmente ver o encadeamento de operaÃ§Ãµes, onde cada funÃ§Ã£o Ã© disposta numa linha. Apenas um adendo: a funÃ§Ã£o `set.seed()` fixa a amostragem de funÃ§Ãµes que geram valores aleatÃ³rios, como Ã© o caso da funÃ§Ã£o `rpois()`.

```
## Fixar amostragem
set.seed(42)

## R Base - sem pipe
ve <- sum(sqrt(sin(log10(rpois(100, 10)))))
ve
```

```
## [1] 91.27018
```

```
## Fixar amostragem
set.seed(42)

## com pipe
ve <- rpois(100, 10) |>
    log10() |>
    sin() |>
    sqrt() |>
    sum()
ve
```

```
## [1] 91.27018
```

O uso do pipe vai se tornar especialmente Ãºtil quando seguirmos para os pacotes das prÃ³ximas duas seÃ§Ãµes: `tidyr` e `dplyr`. Com esses pacotes faremos operaÃ§Ãµes em linhas e colunas de nossos dados tabulares, entÃ£o podemos encadear uma sÃ©rie de funÃ§Ãµes para manipulaÃ§Ã£o, limpeza e anÃ¡lise de dados.

HÃ¡ ainda trÃªs outras variaÃ§Ãµes do pipe que podem ser Ãºteis em alguns momentos, mas que para funcionar precisam que o pacote `magrittr` esteja carregado:

- `%T>%`: retorna o lado esquerdo em vez do lado direito da operaÃ§Ã£o

- `%$%`: “explode” as variÃ¡veis em um quadro de dados
- `%<>%`: permite atribuiÃ§Ã£o usando pipes

Para se aprofundar no tema, recomendamos a leitura do CapÃtulo 18 Pipes de Wickham & Grolemund [-@wickham2017].

Importante
A partir da versÃ£o do R 4.1+ (18/05/2021), o operador pipe `|>` se tornou nativo do R. Podendo ser inserido com o mesmo atalho `Ctrl + Shift + M`, mas necessitando uma mudanÃ§a de opÃ§Ã£o em `Tools > Global Options > Code > [x] Use native pipe operator, |> (requires R 4.1+)`, requerendo que o RStudio esteja numa versÃ£o igual ou superior a 1.4.17+.

8.6 tidyr

Um conjunto de dados `tidy` (organizados) sÃ£o mais fÃ¡ceis de manipular, modelar e visualizar. Um conjunto de dados estÃ¡ no formato `tidy` ou nÃ£o, dependendo de como linhas, colunas e cÃ©lulas sÃ£o combinadas com observaÃ§Ãµes, variÃ¡veis e valores. Nos dados tidy, as variÃ¡veis estÃ£o nas colunas, observaÃ§Ãµes estÃ£o nas linhas e valores estÃ£o nas cÃ©lulas, sendo que para esse Ãºltimo, nÃ£o deve haver mais de um valor por cÃ©lula (Figura 8.2).

1. Cada variÃ¡vel em uma coluna
2. Cada observaÃ§Ã£o em uma linha
3. Cada valor como uma cÃ©lula

Figura 8.2: As trÃªs regras que tornam um conjunto de dados *tidy*.

Todas as funÃ§Ãµes deste pacote sÃ£o listadas na pÃ¡gina de referÃªncia do pacote.

Para realizar diversas transformaÃ§Ãµes nos dados, a fim de ajustÃ¡-los ao formato `tidy` existe uma sÃ©rie de funÃ§Ãµes para: unir colunas, separar colunas, lidar com valores faltantes (`NA`), transformar a base de dados de formato longo para largo (ou vice-e-versa), alÃ©m de outras funÃ§Ãµes especÃficas.

- `unite()`: junta dados de mÃºltiplas colunas em uma coluna
- `separate()`: separa caracteres em mÃºltiplas colunas
- `separate_rows()`: separa caracteres em mÃºltiplas colunas e linhas
- `drop_na()`: retira linhas com `NA` do conjunto de dados
- `replace_na()`: substitui `NA` do conjunto de dados
- `pivot_wider()`: transforma um conjunto de dados longo (*long*) para largo (*wide*)
- `pivot_longer()`: transforma um conjunto de dados largo (*wide*) para longo (*long*)

8.6.1 palmerpenguins

Para exemplificar o funcionamento dessas funÃ§Ãµes, usaremos os dados de medidas de pinguins chamados *palmerpenguins*, disponÃveis no pacote `palmerpenguins`.

```
## Instalar o pacote
install.packages("palmerpenguins")
```

Esses dados foram coletados e disponibilizados pela Dra. Kristen Gorman e pela Palmer Station, Antarctica LTER, membro da Long Term Ecological Research Network.

O pacote `palmerpenguins` contÃ©m dois conjuntos de dados. Um Ã© chamado de `penguins` e Ã© uma versÃ£o simplificada dos dados brutos. O segundo conjunto de dados Ã© `penguins_raw` e contÃ©m todas as variÃ¡veis e nomes originais. Ambos os conjuntos de dados contÃªm dados para 344 pinguins, de trÃªs espÃ©cies diferentes, coletados em trÃªs ilhas no arquipÃ©lago de Palmer, na AntÃ¡rtica. Destacamos tambÃ©m a versÃ£o traduzida desses dados para o portuguÃªs, disponÃvel no pacote `dados`.

Vamos utilizar principalmente o conjunto de dados `penguins_raw`, que Ã© a versÃ£o dos dados brutos.

```
## Carregar o pacote palmerpenguins
library(palmerpenguins)
```

Podemos ainda verificar os dados, pedindo uma ajuda de cada um dos objetos.

```
## Ajuda dos dados
?penguins
?penguins_raw
```

8.6.2 glimpse()

Primeiramente, vamos observar os dados e utilizar a funÃ§Ã£o `tibble::glimpse()` para ter uma noÃ§Ã£o geral dos dados.

```
## Visualizar os dados
penguins_raw
```

```
## # A tibble: 344 x 17
##    studyName `Sample Number` Species             Region Island Stage `Individual ID`
##    <chr>               <dbl> <chr>               <chr>  <chr>  <chr> <chr>
##  1 PAL0708                 1 Adelie Penguin (Py~ Anvers Torge~ Adul~ N1A1
##  2 PAL0708                 2 Adelie Penguin (Py~ Anvers Torge~ Adul~ N1A2
##  3 PAL0708                 3 Adelie Penguin (Py~ Anvers Torge~ Adul~ N2A1
##  4 PAL0708                 4 Adelie Penguin (Py~ Anvers Torge~ Adul~ N2A2
##  5 PAL0708                 5 Adelie Penguin (Py~ Anvers Torge~ Adul~ N3A1
##  6 PAL0708                 6 Adelie Penguin (Py~ Anvers Torge~ Adul~ N3A2
##  7 PAL0708                 7 Adelie Penguin (Py~ Anvers Torge~ Adul~ N4A1
##  8 PAL0708                 8 Adelie Penguin (Py~ Anvers Torge~ Adul~ N4A2
##  9 PAL0708                 9 Adelie Penguin (Py~ Anvers Torge~ Adul~ N5A1
## 10 PAL0708                10 Adelie Penguin (Py~ Anvers Torge~ Adul~ N5A2
## # i 334 more rows
## # i 10 more variables: `Clutch Completion` <chr>, `Date Egg` <date>,
## #   `Culmen Length (mm)` <dbl>, `Culmen Depth (mm)` <dbl>,
## #   `Flipper Length (mm)` <dbl>, `Body Mass (g)` <dbl>, Sex <chr>,
## #   `Delta 15 N (o/oo)` <dbl>, `Delta 13 C (o/oo)` <dbl>, Comments <chr>
```

```
## Espiar os dados
dplyr::glimpse(penguins_raw)
```

```
## Rows: 344
## Columns: 17
## $ studyName             <chr> "PAL0708", "PAL0708", "PAL0708", "PAL0708", "PAL0708~
## $ `Sample Number`       <dbl> 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 1~
## $ Species               <chr> "Adelie Penguin (Pygoscelis adeliae)", "Adelie Pengu~
## $ Region                <chr> "Anvers", "Anvers", "Anvers", "Anvers", "Anvers", "A~
## $ Island                <chr> "Torgersen", "Torgersen", "Torgersen", "Torgersen", ~
## $ Stage                 <chr> "Adult, 1 Egg Stage", "Adult, 1 Egg Stage", "Adult, ~
```

```
## $ `Individual ID`       <chr> "N1A1", "N1A2", "N2A1", "N2A2", "N3A1", "N3A2", "N4A~
## $ `Clutch Completion`   <chr> "Yes", "Yes", "Yes", "Yes", "Yes", "Yes", "No", "No"~
## $ `Date Egg`            <date> 2007-11-11, 2007-11-11, 2007-11-16, 2007-11-16, 200~
## $ `Culmen Length (mm)`  <dbl> 39.1, 39.5, 40.3, NA, 36.7, 39.3, 38.9, 39.2, 34.1, ~
## $ `Culmen Depth (mm)`   <dbl> 18.7, 17.4, 18.0, NA, 19.3, 20.6, 17.8, 19.6, 18.1, ~
## $ `Flipper Length (mm)` <dbl> 181, 186, 195, NA, 193, 190, 181, 195, 193, 190, 186~
## $ `Body Mass (g)`       <dbl> 3750, 3800, 3250, NA, 3450, 3650, 3625, 4675, 3475, ~
## $ Sex                   <chr> "MALE", "FEMALE", "FEMALE", NA, "FEMALE", "MALE", "F~
## $ `Delta 15 N (o/oo)`   <dbl> NA, 8.94956, 8.36821, NA, 8.76651, 8.66496, 9.18718,~
## $ `Delta 13 C (o/oo)`   <dbl> NA, -24.69454, -25.33302, NA, -25.32426, -25.29805, ~
## $ Comments              <chr> "Not enough blood for isotopes.", NA, NA, "Adult not~
```

8.6.3 unite()

Primeiramente, vamos exemplificar como juntar e separar colunas. Vamos utilizar a funÃ§Ã£o `tidyr::unite()` para unir as colunas. HÃ¡ diversos parÃ¢metros para alterar como esta funÃ§Ã£o funciona, entretanto, Ã© importante destacar trÃªs deles: `col` nome da coluna que vai receber as colunas unidas, `sep` indicando o caractere separador das colunas unidas, e `remove` para uma resposta lÃ³gica se as colunas unidas sÃ£o removidas ou nÃ£o. Vamos unir as colunas "Region" e "Island" na nova coluna "region_island".

```r
## Unir colunas
penguins_raw_unir <- tidyr::unite(data = penguins_raw,
                                  col = "region_island",
                                  Region:Island,
                                  sep = ", ",
                                  remove = FALSE)
head(penguins_raw_unir[, c("Region", "Island", "region_island")])
```

```
## # A tibble: 6 x 3
##   Region Island    region_island
##   <chr>  <chr>     <chr>
## 1 Anvers Torgersen Anvers, Torgersen
## 2 Anvers Torgersen Anvers, Torgersen
## 3 Anvers Torgersen Anvers, Torgersen
## 4 Anvers Torgersen Anvers, Torgersen
## 5 Anvers Torgersen Anvers, Torgersen
## 6 Anvers Torgersen Anvers, Torgersen
```

8.6.4 separate()

De forma contrÃ¡ria, podemos utilizar as funÃ§Ãµes `tidyr::separate()` e `tidyr::separate_rows()` para separar elementos de uma coluna em mais colunas. Respectivamente, a primeira funÃ§Ã£o separa uma coluna em novas colunas conforme a separaÃ§Ã£o, e a segunda funÃ§Ã£o separa uma coluna, distribuindo os elementos nas linhas. Novamente, hÃ¡ diversos parÃ¢metros para mudar o comportamento dessas funÃ§Ãµes, mas destacaremos aqui quatro deles: `col` coluna a ser separada, `into` os nomes das novas colunas, `sep` indicando o caractere separador das colunas, e `remove` para uma resposta lÃ³gica se as colunas separadas sÃ£o removidas ou nÃ£o. Vamos separar a coluna "Stage" nas colunas "stage" e "egg_stage".

```r
## Separar colunas
penguins_raw_separar <- tidyr::separate(data = penguins_raw,
                                        col = Stage,
                                        into = c("stage", "egg_stage"),
                                        sep = ", ",
                                        remove = FALSE)
head(penguins_raw_separar[, c("Stage", "stage", "egg_stage")])
```

```
## # A tibble: 6 x 3
##   Stage              stage egg_stage
##   <chr>              <chr> <chr>
## 1 Adult, 1 Egg Stage Adult 1 Egg Stage
## 2 Adult, 1 Egg Stage Adult 1 Egg Stage
## 3 Adult, 1 Egg Stage Adult 1 Egg Stage
## 4 Adult, 1 Egg Stage Adult 1 Egg Stage
## 5 Adult, 1 Egg Stage Adult 1 Egg Stage
## 6 Adult, 1 Egg Stage Adult 1 Egg Stage
```

```
## Separar colunas em novas linhas
penguins_raw_separar_linhas <- tidyr::separate_rows(data = penguins_raw,
                                                    Stage,
                                                    sep = ", ")
head(penguins_raw_separar_linhas[, c("studyName", "Sample Number", "Species",
                                     "Region", "Island", "Stage")])
```

```
## # A tibble: 6 x 6
##   studyName `Sample Number` Species                            Region Island  Stage
##   <chr>               <dbl> <chr>                              <chr>  <chr>   <chr>
## 1 PAL0708                 1 Adelie Penguin (Pygoscelis adeliae) Anvers Torger~ Adult
## 2 PAL0708                 1 Adelie Penguin (Pygoscelis adeliae) Anvers Torger~ 1 Eg~
## 3 PAL0708                 2 Adelie Penguin (Pygoscelis adeliae) Anvers Torger~ Adult
## 4 PAL0708                 2 Adelie Penguin (Pygoscelis adeliae) Anvers Torger~ 1 Eg~
## 5 PAL0708                 3 Adelie Penguin (Pygoscelis adeliae) Anvers Torger~ Adult
## 6 PAL0708                 3 Adelie Penguin (Pygoscelis adeliae) Anvers Torger~ 1 Eg~
```

8.6.5 drop_na() e replace_na()

Valor faltante (`NA`) Ã© um tipo especial de elemento que sÃ£o discutidos no CapÃtulo 7 e sÃ£o relativamente comuns em conjuntos de dados. Em *R Base*, vimos algumas formas de lidar com esse tipo de elemento. No formato `tidyverse`, existem tambÃ©m vÃ¡rias formas de lidar com eles, mas aqui focaremos nas funÃ§Ãµes `tidyr::drop_na()` e `tidyr::replace_na()`, para retirar linhas e substituir esses valores, respectivamente.

```
## Remover todas as linhas com NAs
penguins_raw_todas_na <- tidyr::drop_na(data = penguins_raw)
head(penguins_raw_todas_na)
```

```
## # A tibble: 6 x 17
##   studyName `Sample Number` Species              Region Island Stage `Individual ID`
##   <chr>               <dbl> <chr>                <chr>  <chr>  <chr> <chr>
## 1 PAL0708                 7 Adelie Penguin (Pyg~ Anvers Torge~ Adul~ N4A1
## 2 PAL0708                 8 Adelie Penguin (Pyg~ Anvers Torge~ Adul~ N4A2
## 3 PAL0708                29 Adelie Penguin (Pyg~ Anvers Biscoe Adul~ N18A1
## 4 PAL0708                30 Adelie Penguin (Pyg~ Anvers Biscoe Adul~ N18A2
## 5 PAL0708                39 Adelie Penguin (Pyg~ Anvers Dream  Adul~ N25A1
## 6 PAL0809                69 Adelie Penguin (Pyg~ Anvers Torge~ Adul~ N32A1
## # i 10 more variables: `Clutch Completion` <chr>, `Date Egg` <date>,
## #   `Culmen Length (mm)` <dbl>, `Culmen Depth (mm)` <dbl>,
## #   `Flipper Length (mm)` <dbl>, `Body Mass (g)` <dbl>, Sex <chr>,
## #   `Delta 15 N (o/oo)` <dbl>, `Delta 13 C (o/oo)` <dbl>, Comments <chr>
```

```
## Remover linhas de colunas especÃficas com NAs
penguins_raw_colunas_na <- tidyr::drop_na(data = penguins_raw,
                                          any_of("Comments"))
head(penguins_raw_colunas_na[, "Comments"])
```

```
## # A tibble: 6 x 1
##   Comments
##   <chr>
## 1 Not enough blood for isotopes.
## 2 Adult not sampled.
## 3 Nest never observed with full clutch.
## 4 Nest never observed with full clutch.
## 5 No blood sample obtained.
## 6 No blood sample obtained for sexing.
```

```
## Substituir NAs por outro valor
penguins_raw_subs_na <- tidyr::replace_na(data = penguins_raw,
                                          list(Comments = "Unknown"))
head(penguins_raw_subs_na[, "Comments"])
```

```
## # A tibble: 6 x 1
##   Comments
##   <chr>
## 1 Not enough blood for isotopes.
## 2 Unknown
## 3 Unknown
## 4 Adult not sampled.
## 5 Unknown
## 6 Unknown
```

8.6.6 pivot_longer() e pivot_wider()

Por fim, trataremos da pivotagem ou remodelagem de dados. Veremos como mudar o formato do nosso conjunto de dados de longo (*long*) para largo (*wide*) e vice-versa. Primeiramente, vamos ver como partir de um dado longo (*long*) e criar um dado largo (*wide*). Essa Ã© uma operaÃ§Ã£o semelhante Ã "Tabela DinÃ¢mica" das planilhas eletrÃ´nicas. Consiste em usar uma coluna para distribuir seus valores em outras colunas, de modo que os valores dos elementos sÃ£o preenchidos corretamente, reduzindo assim o nÃºmero de linhas e aumentando o nÃºmero de colunas. Essa operaÃ§Ã£o Ã© bastante comum em Ecologia de Comunidades, quando queremos transformar uma lista de espÃ©cies em uma matriz de comunidades, com vÃ¡rias espÃ©cies nas colunas. Para realizar essa operaÃ§Ã£o, usamos a funÃ§Ã£o `tidyr::pivot_wider()`. Dos diversos parÃ¢metros que podem compor essa funÃ§Ã£o, dois deles sÃ£o fundamentais: `names_from` que indica a coluna de onde os nomes serÃ£o usados e `values_from` que indica que indica a coluna com os valores.

```
## Selecionar colunas
penguins_raw_sel_col <- penguins_raw[, c(2, 3, 13)]
head(penguins_raw_sel_col)
```

```
## # A tibble: 6 x 3
##   `Sample Number` Species                              `Body Mass (g)`
##             <dbl> <chr>                                          <dbl>
## 1               1 Adelie Penguin (Pygoscelis adeliae)            3750
## 2               2 Adelie Penguin (Pygoscelis adeliae)            3800
## 3               3 Adelie Penguin (Pygoscelis adeliae)            3250
## 4               4 Adelie Penguin (Pygoscelis adeliae)              NA
## 5               5 Adelie Penguin (Pygoscelis adeliae)            3450
## 6               6 Adelie Penguin (Pygoscelis adeliae)            3650
```

```
## Pivotar para largo
penguins_raw_pivot_wider <- tidyr::pivot_wider(data = penguins_raw_sel_col,
                                               names_from = Species,
```

```
                                                values_from = `Body Mass (g)`)
head(penguins_raw_pivot_wider)
```

```
## # A tibble: 6 x 4
##   `Sample Number` `Adelie Penguin (Pygoscelis adeliae)` Gentoo penguin (Pygoscelis~1
##             <dbl>                                 <dbl>                        <dbl>
## 1               1                                  3750                         4500
## 2               2                                  3800                         5700
## 3               3                                  3250                         4450
## 4               4                                    NA                         5700
## 5               5                                  3450                         5400
## 6               6                                  3650                         4550
## # i abbreviated name: 1: `Gentoo penguin (Pygoscelis papua)`
## # i 1 more variable: `Chinstrap penguin (Pygoscelis antarctica)` <dbl>
```

De modo oposto, podemos partir de um conjunto de dados largo (*wide*), ou seja, com vÃ¡rias colunas, e queremos que essas colunas preencham uma Ãºnica coluna, e que os valores antes espalhados nessas vÃ¡rias colunas sejam adicionados um embaixo do outro, numa Ãºnica coluna, no formato longo (*long*). Para essa operaÃ§Ã£o, podemos utilizar a funÃ§Ã£o `tidyr::pivot_longer()`. Novamente, dos diversos parÃ¢metros que podem compor essa funÃ§Ã£o, trÃªs deles sÃ£o fundamentais: `cols` indicando as colunas que serÃ£o usadas para serem pivotadas, `names_to` que indica a coluna de onde os nomes serÃ£o usados e `values_to` que indica a coluna com os valores.

```
## Selecionar colunas
penguins_raw_sel_col <- penguins_raw[, c(2, 3, 10:13)]
head(penguins_raw_sel_col)
```

```
## # A tibble: 6 x 6
##   `Sample Number` Species                    `Culmen Length (mm)` `Culmen Depth (mm)`
##             <dbl> <chr>                                     <dbl>               <dbl>
## 1               1 Adelie Penguin (Pygoscel~                  39.1                18.7
## 2               2 Adelie Penguin (Pygoscel~                  39.5                17.4
## 3               3 Adelie Penguin (Pygoscel~                  40.3                18
## 4               4 Adelie Penguin (Pygoscel~                  NA                  NA
## 5               5 Adelie Penguin (Pygoscel~                  36.7                19.3
## 6               6 Adelie Penguin (Pygoscel~                  39.3                20.6
## # i 2 more variables: `Flipper Length (mm)` <dbl>, `Body Mass (g)` <dbl>
```

```
## Pivotar para largo
penguins_raw_pivot_longer <- tidyr::pivot_longer(data = penguins_raw_sel_col,
                                                 cols = `Culmen Length (mm)`:`Body Mass (g)`,
                                                 names_to = "medidas",
                                                 values_to = "valores")
head(penguins_raw_pivot_longer)
```

```
## # A tibble: 6 x 4
##   `Sample Number` Species                              medidas             valores
##             <dbl> <chr>                                <chr>                 <dbl>
## 1               1 Adelie Penguin (Pygoscelis adeliae) Culmen Length (mm)     39.1
## 2               1 Adelie Penguin (Pygoscelis adeliae) Culmen Depth (mm)      18.7
## 3               1 Adelie Penguin (Pygoscelis adeliae) Flipper Length (mm)   181
## 4               1 Adelie Penguin (Pygoscelis adeliae) Body Mass (g)        3750
## 5               2 Adelie Penguin (Pygoscelis adeliae) Culmen Length (mm)     39.5
## 6               2 Adelie Penguin (Pygoscelis adeliae) Culmen Depth (mm)      17.4
```

Para se aprofundar no tema, recomendamos a leitura do CapÃtulo 12 Tidy data de Wickham & Grolemund

[-@wickham2017].

8.7 dplyr

O `dplyr` Ã© um pacote que facilita a manipulaÃ§Ã£o de dados, com uma gramÃ¡tica simples e flexÃvel (por exemplo, como filtragem, reordenamento, seleÃ§Ã£o, entre outras). Ele foi construÃdo com o intuito de obter uma forma mais rÃ¡pida e expressiva de manipular dados tabulares. O `tibble` Ã© a versÃ£o de data frame mais conveniente para se usar com pacote `dplyr`.

Todas as funÃ§Ãµes deste pacote sÃ£o listadas na pÃ¡gina de referÃªncia do pacote.

8.7.1 GramÃ¡tica

Sua gramÃ¡tica simples contÃ©m funÃ§Ãµes verbais para manipulaÃ§Ã£o de dados, baseada em:

- Verbos: `mutate()`, `select()`, `filter()`, `arrange()`, `summarise()`, `slice()`, `rename()`, etc.
- ReplicaÃ§Ã£o: `across()`, `if_any()`, `if_all()`, `where()`, `starts_with()`, `ends_with()`, `contains()`, etc.
- Agrupamento: `group_by()` e `ungroup()`
- JunÃ§Ãµes: `inner_join()`, `full_join()`, `left_join()`, `right_join()`, etc.
- CombinaÃ§Ãµes: `bind_rows()` e `bind_cols()`
- Resumos, contagem e seleÃ§Ã£o: `n()`, `n_distinct()`, `first()`, `last()`, `nth()`, etc.

Existe uma sÃ©rie de funÃ§Ãµes para realizar a manipulaÃ§Ã£o dos dados, com diversas finalidades: manipulaÃ§Ã£o de uma tabela, manipulaÃ§Ã£o de duas tabelas, replicaÃ§Ã£o, agrupamento, funÃ§Ãµes de vetores, alÃ©m de muitas outras funÃ§Ãµes especÃficas.

- `relocate()`: muda a ordem das colunas
- `rename()`: muda o nome das colunas
- `select()`: seleciona colunas pelo nome ou posiÃ§Ã£o
- `pull()`: seleciona uma coluna como vetor
- `mutate()`: adiciona novas colunas ou resultados em colunas existentes
- `arrange()`: reordena as linhas com base nos valores de colunas
- `filter()`: seleciona linhas com base em valores de colunas
- `slice()`: seleciona linhas de diferente formas
- `distinct()`: remove linhas com valores repetidos com base nos valores de colunas
- `count()`: conta observaÃ§Ãµes para um grupo com base nos valores de colunas
- `group_by()`: agrupa linhas pelos valores das colunas
- `summarise()`: resume os dados atravÃ©s de funÃ§Ãµes considerando valores das colunas
- `*_join()`: funÃ§Ãµes que juntam dados de duas tabelas atravÃ©s de uma coluna chave

8.7.2 Sintaxe

As funÃ§Ãµes do `dplyr` podem seguir uma mesma sintaxe: o `tibble` serÃ¡ sempre o primeiro argumento dessas funÃ§Ãµes, seguido de um operador pipe (`|>`ou %>%‘) e pelo nome da funÃ§Ã£o que irÃ¡ fazer a manipulaÃ§Ã£o nesses dados. Isso permite o encadeamento de vÃ¡rias operaÃ§Ãµes consecutivas mantendo a estrutura do dado original e acrescentando mudanÃ§as num encadeamento lÃ³gico.

Sendo assim, as funÃ§Ãµes verbais nÃ£o precisam modificar necessariamente o tibble original, sendo que as operaÃ§Ãµes de manipulaÃ§Ãµes podem e devem ser atribuÃdas a um novo objeto.

```
## Sintaxe
tb_dplyr <- tb |>
    funcao_verbal1(argumento1, argumento2, ...) |>
    funcao_verbal2(argumento1, argumento2, ...) |>
    funcao_verbal3(argumento1, argumento2, ...)
```

AlÃ©m de `data.frames` e `tibbles`, a manipulaÃ§Ã£o pelo formato `dplyr` torna o trabalho com outros formatos de classes e dados acessÃveis e eficientes como `data.table`, SQL e Apache Spark, para os quais existem pacotes especÃficos.

- *dtplyr*: manipular conjuntos de dados `data.table`
- *dbplyr*: manipular conjuntos de dados SQL
- *sparklyr*: manipular conjuntos de dados no Apache Spark

8.7.3 palmerpenguins

Para nossos exemplos, vamos utilizar novamente os dados de pinguins *palmerpenguins*. Esses dados estÃ£o disponÃveis no pacote `palmerpenguins`. Vamos utilizar principalmente o conjunto de dados `penguins`, que Ã© a versÃ£o simplificada dos dados brutos `penguins_raw`.

```
## Carregar o pacote palmerpenguins
library(palmerpenguins)
```

8.7.4 relocate()

Primeiramente, vamos reordenar as colunas com a funÃ§Ã£o `dplyr::relocate()`, onde simplesmente listamos as colunas que queremos mudar de posiÃ§Ã£o e para onde elas devem ir. Para esse Ãºltimo passo hÃ¡ dois argumentos: `.before` que indica a coluna onde a coluna realocada deve se mover antes, e o argumento `.after` indicando onde deve se mover depois. Ambos podem ser informados com os nomes ou posiÃ§Ãµes dessas colunas com nÃºmeros.

```
## Reordenar colunas - nome
penguins_relocate_col <- penguins |>
    dplyr::relocate(sex, year, .after = island)
head(penguins_relocate_col)
```

```
## # A tibble: 6 x 8
##   species island    sex     year bill_length_mm bill_depth_mm flipper_length_mm
##   <fct>   <fct>     <fct>  <int>          <dbl>         <dbl>             <int>
## 1 Adelie  Torgersen male    2007           39.1          18.7               181
## 2 Adelie  Torgersen female  2007           39.5          17.4               186
## 3 Adelie  Torgersen female  2007           40.3          18                 195
## 4 Adelie  Torgersen <NA>    2007           NA            NA                  NA
## 5 Adelie  Torgersen female  2007           36.7          19.3               193
## 6 Adelie  Torgersen male    2007           39.3          20.6               190
## # i 1 more variable: body_mass_g <int>
```

```
## Reordenar colunas - posiÃ§Ã£o
penguins_relocate_ncol <- penguins |>
    dplyr::relocate(sex, year, .after = 2)
head(penguins_relocate_ncol)
```

```
## # A tibble: 6 x 8
##   species island    sex     year bill_length_mm bill_depth_mm flipper_length_mm
##   <fct>   <fct>     <fct>  <int>          <dbl>         <dbl>             <int>
## 1 Adelie  Torgersen male    2007           39.1          18.7               181
## 2 Adelie  Torgersen female  2007           39.5          17.4               186
## 3 Adelie  Torgersen female  2007           40.3          18                 195
## 4 Adelie  Torgersen <NA>    2007           NA            NA                  NA
## 5 Adelie  Torgersen female  2007           36.7          19.3               193
## 6 Adelie  Torgersen male    2007           39.3          20.6               190
## # i 1 more variable: body_mass_g <int>
```

8.7.5 rename()

Podemos renomear colunas facilmente com a funÃ§Ã£o `dplyr::rename()`, onde primeiramente informamos o nome que queremos que a coluna tenha, seguido do operador `=` e a coluna do nosso dado ("nova_coluna = antiga_coluna"). TambÃ©m podemos utilizar a funÃ§Ã£o `dplyr::rename_with()`, que faz a mudanÃ§a do nome em mÃºltiplas colunas, que pode depender ou nÃ£o de resultados booleanos.

```
## Renomear as colunas
penguins_rename <- penguins |>
    dplyr::rename(bill_length = bill_length_mm,
                  bill_depth = bill_depth_mm,
                  flipper_length = flipper_length_mm,
                  body_mass = body_mass_g)
head(penguins_rename)
```

```
## # A tibble: 6 x 8
##   species island    bill_length bill_depth flipper_length body_mass sex     year
##   <fct>   <fct>           <dbl>      <dbl>          <int>     <int> <fct>  <int>
## 1 Adelie  Torgersen        39.1       18.7            181      3750 male    2007
## 2 Adelie  Torgersen        39.5       17.4            186      3800 female  2007
## 3 Adelie  Torgersen        40.3       18              195      3250 female  2007
## 4 Adelie  Torgersen        NA         NA               NA        NA <NA>    2007
## 5 Adelie  Torgersen        36.7       19.3            193      3450 female  2007
## 6 Adelie  Torgersen        39.3       20.6            190      3650 male    2007
```

```
## mudar o nome de todas as colunas
penguins_rename_with <- penguins |>
    dplyr::rename_with(toupper)
head(penguins_rename_with)
```

```
## # A tibble: 6 x 8
##   SPECIES ISLAND    BILL_LENGTH_MM BILL_DEPTH_MM FLIPPER_LENGTH_MM BODY_MASS_G SEX
##   <fct>   <fct>              <dbl>         <dbl>             <int>       <int> <fct>
## 1 Adelie  Torgersen           39.1          18.7               181        3750 male
## 2 Adelie  Torgersen           39.5          17.4               186        3800 fema~
## 3 Adelie  Torgersen           40.3          18                 195        3250 fema~
## 4 Adelie  Torgersen           NA            NA                  NA          NA <NA>
## 5 Adelie  Torgersen           36.7          19.3               193        3450 fema~
## 6 Adelie  Torgersen           39.3          20.6               190        3650 male
## # i 1 more variable: YEAR <int>
```

8.7.6 select()

Outra operaÃ§Ã£o bastante usual dentro da manipulaÃ§Ã£o de dados tabulares Ã© a seleÃ§Ã£o de colunas. Podemos fazer essa operaÃ§Ã£o com a funÃ§Ã£o `dplyr::select()`, que seleciona colunas pelo nome ou pela sua posiÃ§Ã£o. Aqui hÃ¡ uma sÃ©rie de possibilidades de seleÃ§Ã£o de colunas, desde utilizar operadores como `:` para selecionar intervalos de colunas, `!` para tomar o complemento (todas menos as listadas), alÃ©m de funÃ§Ãµes como `dplyr::starts_with()`, `dplyr::ends_with()`, `dplyr::contains()` para procurar colunas com um padrÃ£o de texto do nome da coluna.

```
## Selecionar colunas por posiÃ§Ã£o (3a atÃ© 6a)
penguins_select_position <- penguins |>
    dplyr::select(3:6)
head(penguins_select_position)
```

```
## # A tibble: 6 x 4
##   bill_length_mm bill_depth_mm flipper_length_mm body_mass_g
```

```
##            <dbl>         <dbl>             <int>       <int>
## 1           39.1          18.7               181        3750
## 2           39.5          17.4               186        3800
## 3           40.3          18                 195        3250
## 4           NA            NA                  NA          NA
## 5           36.7          19.3               193        3450
## 6           39.3          20.6               190        3650
```

```
## Selecionar colunas por nomes
penguins_select_names <- penguins |>
    dplyr::select(bill_length_mm, body_mass_g)
head(penguins_select_names)
```

```
## # A tibble: 6 x 2
##   bill_length_mm body_mass_g
##            <dbl>       <int>
## 1           39.1        3750
## 2           39.5        3800
## 3           40.3        3250
## 4           NA            NA
## 5           36.7        3450
## 6           39.3        3650
```

```
## Selecionar colunas por padrÃ£o
penguins_select_contains <- penguins |>
    dplyr::select(contains("_mm"))
head(penguins_select_contains)
```

```
## # A tibble: 6 x 3
##   bill_length_mm bill_depth_mm flipper_length_mm
##            <dbl>         <dbl>             <int>
## 1           39.1          18.7               181
## 2           39.5          17.4               186
## 3           40.3          18                 195
## 4           NA            NA                  NA
## 5           36.7          19.3               193
## 6           39.3          20.6               190
```

8.7.7 pull()

Quando usamos a funÃ§Ã£o `dplyr::select()`, mesmo que para apenas uma coluna, o retorno da funÃ§Ã£o Ã© sempre um `tibble`. Caso precisemos que essa coluna se torne um vetor dentro do encadeamento dos `pipes`, usamos a funÃ§Ã£o `dplyr::pull()`, que extrai uma Ãºnica coluna como vetor.

```
## Coluna como vetor
penguins_select_pull <- penguins |>
    dplyr::pull(bill_length_mm)
head(penguins_select_pull, 15)
```

```
##  [1] 39.1 39.5 40.3   NA 36.7 39.3 38.9 39.2 34.1 42.0 37.8 37.8 41.1 38.6 34.6
```

8.7.8 mutate()

Uma das operaÃ§Ãµes mais Ãºteis dentre as operaÃ§Ãµes para colunas Ã© adicionar ou atualizar os valores de colunas. Para essa operaÃ§Ã£o, usaremos a funÃ§Ã£o `dplyr::mutate()`. Podemos ainda usar os argumentos `.before` e `.after` para indicar onde a nova coluna deve ficar, alÃ©m do parÃ¢metro `.keep` com diversas possibilidades de manter colunas depois de usar a funÃ§Ã£o `dplyr::mutate()`. Por fim,

Ã© fundamental destacar o uso das funÃ§Ãµes de replicaÃ§Ã£o: `dplyr::across()`, `dplyr::if_any()` e `dplyr::if_all()`, para os quais a funÃ§Ã£o farÃ¡ alteraÃ§Ãµes em mÃºltiplas colunas de uma vez, dependendo de resultados booleanos.

```
## Adicionar colunas
penguins_mutate <- penguins |>
    dplyr::mutate(body_mass_kg = body_mass_g/1e3, .before = sex)
head(penguins_mutate)
```

```
## # A tibble: 6 x 9
##   species island    bill_length_mm bill_depth_mm flipper_length_mm body_mass_g
##   <fct>   <fct>              <dbl>         <dbl>             <int>       <int>
## 1 Adelie  Torgersen           39.1          18.7               181        3750
## 2 Adelie  Torgersen           39.5          17.4               186        3800
## 3 Adelie  Torgersen           40.3          18                 195        3250
## 4 Adelie  Torgersen           NA            NA                  NA          NA
## 5 Adelie  Torgersen           36.7          19.3               193        3450
## 6 Adelie  Torgersen           39.3          20.6               190        3650
## # i 3 more variables: body_mass_kg <dbl>, sex <fct>, year <int>
```

```
## Modificar vÃ¡rias colunas
penguins_mutate_across <- penguins |>
    dplyr::mutate(across(where(is.factor), as.character))
head(penguins_mutate_across)
```

```
## # A tibble: 6 x 8
##   species island    bill_length_mm bill_depth_mm flipper_length_mm body_mass_g sex
##   <chr>   <chr>              <dbl>         <dbl>             <int>       <int> <chr>
## 1 Adelie  Torgersen           39.1          18.7               181        3750 male
## 2 Adelie  Torgersen           39.5          17.4               186        3800 fema~
## 3 Adelie  Torgersen           40.3          18                 195        3250 fema~
## 4 Adelie  Torgersen           NA            NA                  NA          NA <NA>
## 5 Adelie  Torgersen           36.7          19.3               193        3450 fema~
## 6 Adelie  Torgersen           39.3          20.6               190        3650 male
## # i 1 more variable: year <int>
```

8.7.9 arrange()

AlÃ©m de operaÃ§Ãµes em colunas, podemos fazer operaÃ§Ãµes em linhas. Vamos comeÃ§ar com a reordenaÃ§Ã£o das linhas com base nos valores das colunas. Para essa operaÃ§Ã£o, usamos a funÃ§Ã£o `dplyr::arrange()`. Podemos reordenar por uma ou mais colunas de forma crescente ou decrescente usando a funÃ§Ã£o `desc()` ou o operador `-` antes da coluna de interesse. Da mesma forma que na funÃ§Ã£o `dplyr::mutate()`, podemos usar as funÃ§Ãµes de replicaÃ§Ã£o para ordenar as linhas para vÃ¡rias colunas de uma vez, dependendo de resultados booleanos.

```
## Reordenar linhas - crescente
penguins_arrange <- penguins |>
    dplyr::arrange(body_mass_g)
head(penguins_arrange)
```

```
## # A tibble: 6 x 8
##   species   island  bill_length_mm bill_depth_mm flipper_length_mm body_mass_g sex
##   <fct>     <fct>            <dbl>         <dbl>             <int>       <int> <fct>
## 1 Chinstrap Dream             46.9          16.6               192        2700 fema~
## 2 Adelie    Biscoe            36.5          16.6               181        2850 fema~
## 3 Adelie    Biscoe            36.4          17.1               184        2850 fema~
```

```
## 4 Adelie    Biscoe            34.5          18.1                187        2900 fema~
## 5 Adelie    Dream             33.1          16.1                178        2900 fema~
## 6 Adelie    Torger~           38.6          17                  188        2900 fema~
## # i 1 more variable: year <int>
```

```
## Reordenar linhas - decrescente
penguins_arrange_desc <- penguins |>
    dplyr::arrange(desc(body_mass_g))
head(penguins_arrange_desc)
```

```
## # A tibble: 6 x 8
##   species island bill_length_mm bill_depth_mm flipper_length_mm body_mass_g sex
##   <fct>   <fct>           <dbl>         <dbl>             <int>       <int> <fct>
## 1 Gentoo  Biscoe           49.2          15.2               221        6300 male
## 2 Gentoo  Biscoe           59.6          17                 230        6050 male
## 3 Gentoo  Biscoe           51.1          16.3               220        6000 male
## 4 Gentoo  Biscoe           48.8          16.2               222        6000 male
## 5 Gentoo  Biscoe           45.2          16.4               223        5950 male
## 6 Gentoo  Biscoe           49.8          15.9               229        5950 male
## # i 1 more variable: year <int>
```

```
## Reordenar linhas - decrescente
penguins_arrange_desc_m <- penguins |>
    dplyr::arrange(-body_mass_g)
head(penguins_arrange_desc_m)
```

```
## # A tibble: 6 x 8
##   species island bill_length_mm bill_depth_mm flipper_length_mm body_mass_g sex
##   <fct>   <fct>           <dbl>         <dbl>             <int>       <int> <fct>
## 1 Gentoo  Biscoe           49.2          15.2               221        6300 male
## 2 Gentoo  Biscoe           59.6          17                 230        6050 male
## 3 Gentoo  Biscoe           51.1          16.3               220        6000 male
## 4 Gentoo  Biscoe           48.8          16.2               222        6000 male
## 5 Gentoo  Biscoe           45.2          16.4               223        5950 male
## 6 Gentoo  Biscoe           49.8          15.9               229        5950 male
## # i 1 more variable: year <int>
```

```
## Reordenar linhas - multiplas colunas
penguins_arrange_across <- penguins |>
    dplyr::arrange(across(where(is.numeric)))
head(penguins_arrange_across)
```

```
## # A tibble: 6 x 8
##   species island    bill_length_mm bill_depth_mm flipper_length_mm body_mass_g sex
##   <fct>   <fct>              <dbl>         <dbl>             <int>       <int> <fct>
## 1 Adelie  Dream               32.1          15.5               188        3050 fema~
## 2 Adelie  Dream               33.1          16.1               178        2900 fema~
## 3 Adelie  Torgersen           33.5          19                 190        3600 fema~
## 4 Adelie  Dream               34            17.1               185        3400 fema~
## 5 Adelie  Torgersen           34.1          18.1               193        3475 <NA>
## 6 Adelie  Torgersen           34.4          18.4               184        3325 fema~
## # i 1 more variable: year <int>
```

8.7.10 filter()

Uma das principais e mais usuais operaÃ§Ãµes que podemos realizar em linhas Ã© a seleÃ§Ã£o de linhas atravÃ©s do filtro por valores de uma ou mais colunas, utilizando a funÃ§Ã£o `dplyr::filter()`. Para

realizar os filtros utilizaremos grande parte dos operadores relacionais e lÃ³gicos que listamos no CapÃtulo 7, especialmente os lÃ³gicos para combinaÃ§Ãµes de filtros em mais de uma coluna. AlÃ©m desses operadores, podemos utilizar a funÃ§Ã£o `is.na()` para filtros em elementos faltantes, e as funÃ§Ãµes `dplyr::between()` e `dplyr::near()` para filtros entre valores, e para valores prÃ³ximos com certa tolerÃ¢ncia, respectivamente. Por fim, podemos usar as funÃ§Ãµes de replicaÃ§Ã£o para filtro das linhas para mais de uma coluna, dependendo de resultados booleanos.

```r
## Filtrar linhas
penguins_filter <- penguins |>
    dplyr::filter(species == "Adelie")
head(penguins_filter)
```

```
## # A tibble: 6 x 8
##   species island    bill_length_mm bill_depth_mm flipper_length_mm body_mass_g sex
##   <fct>   <fct>              <dbl>         <dbl>             <int>       <int> <fct>
## 1 Adelie  Torgersen           39.1          18.7               181        3750 male
## 2 Adelie  Torgersen           39.5          17.4               186        3800 fema~
## 3 Adelie  Torgersen           40.3          18                 195        3250 fema~
## 4 Adelie  Torgersen           NA            NA                  NA          NA <NA>
## 5 Adelie  Torgersen           36.7          19.3               193        3450 fema~
## 6 Adelie  Torgersen           39.3          20.6               190        3650 male
## # i 1 more variable: year <int>
```

```r
## Filtrar linhas
penguins_filter_two <- penguins |>
    dplyr::filter(species == "Adelie" & sex == "female")
head(penguins_filter_two)
```

```
## # A tibble: 6 x 8
##   species island    bill_length_mm bill_depth_mm flipper_length_mm body_mass_g sex
##   <fct>   <fct>              <dbl>         <dbl>             <int>       <int> <fct>
## 1 Adelie  Torgersen           39.5          17.4               186        3800 fema~
## 2 Adelie  Torgersen           40.3          18                 195        3250 fema~
## 3 Adelie  Torgersen           36.7          19.3               193        3450 fema~
## 4 Adelie  Torgersen           38.9          17.8               181        3625 fema~
## 5 Adelie  Torgersen           41.1          17.6               182        3200 fema~
## 6 Adelie  Torgersen           36.6          17.8               185        3700 fema~
## # i 1 more variable: year <int>
```

```r
## Filtrar linhas
penguins_filter_in <- penguins |>
    dplyr::filter(species %in% c("Adelie", "Gentoo"),
                  sex == "female")
head(penguins_filter_in)
```

```
## # A tibble: 6 x 8
##   species island    bill_length_mm bill_depth_mm flipper_length_mm body_mass_g sex
##   <fct>   <fct>              <dbl>         <dbl>             <int>       <int> <fct>
## 1 Adelie  Torgersen           39.5          17.4               186        3800 fema~
## 2 Adelie  Torgersen           40.3          18                 195        3250 fema~
## 3 Adelie  Torgersen           36.7          19.3               193        3450 fema~
## 4 Adelie  Torgersen           38.9          17.8               181        3625 fema~
## 5 Adelie  Torgersen           41.1          17.6               182        3200 fema~
## 6 Adelie  Torgersen           36.6          17.8               185        3700 fema~
## # i 1 more variable: year <int>
```

```r
## Filtrar linhas - NA
penguins_filter_na <- penguins |>
    dplyr::filter(!is.na(sex) == TRUE)
head(penguins_filter_na)
```

```
## # A tibble: 6 x 8
##   species island    bill_length_mm bill_depth_mm flipper_length_mm body_mass_g sex
##   <fct>   <fct>              <dbl>         <dbl>             <int>       <int> <fct>
## 1 Adelie  Torgersen           39.1          18.7               181        3750 male
## 2 Adelie  Torgersen           39.5          17.4               186        3800 fema~
## 3 Adelie  Torgersen           40.3          18                 195        3250 fema~
## 4 Adelie  Torgersen           36.7          19.3               193        3450 fema~
## 5 Adelie  Torgersen           39.3          20.6               190        3650 male
## 6 Adelie  Torgersen           38.9          17.8               181        3625 fema~
## # i 1 more variable: year <int>
```

```r
## Filtrar linhas - intervalos
penguins_filter_between <- penguins |>
    dplyr::filter(between(body_mass_g, 3000, 4000))
head(penguins_filter_between)
```

```
## # A tibble: 6 x 8
##   species island    bill_length_mm bill_depth_mm flipper_length_mm body_mass_g sex
##   <fct>   <fct>              <dbl>         <dbl>             <int>       <int> <fct>
## 1 Adelie  Torgersen           39.1          18.7               181        3750 male
## 2 Adelie  Torgersen           39.5          17.4               186        3800 fema~
## 3 Adelie  Torgersen           40.3          18                 195        3250 fema~
## 4 Adelie  Torgersen           36.7          19.3               193        3450 fema~
## 5 Adelie  Torgersen           39.3          20.6               190        3650 male
## 6 Adelie  Torgersen           38.9          17.8               181        3625 fema~
## # i 1 more variable: year <int>
```

```r
## Filtrar linhas por vÃ¡rias colunas
penguins_filter_if <- penguins |>
    dplyr::filter(if_all(where(is.integer), ~ . > 200))
head(penguins_filter_if)
```

```
## # A tibble: 6 x 8
##   species island    bill_length_mm bill_depth_mm flipper_length_mm body_mass_g sex
##   <fct>   <fct>              <dbl>         <dbl>             <int>       <int> <fct>
## 1 Adelie  Dream               35.7          18                 202        3550 fema~
## 2 Adelie  Dream               41.1          18.1               205        4300 male
## 3 Adelie  Dream               40.8          18.9               208        4300 male
## 4 Adelie  Biscoe              41            20                 203        4725 male
## 5 Adelie  Torgersen           41.4          18.5               202        3875 male
## 6 Adelie  Torgersen           44.1          18                 210        4000 male
## # i 1 more variable: year <int>
```

8.7.11 slice()

AlÃ©m da seleÃ§Ã£o de linhas por filtros, podemos fazer a seleÃ§Ã£o das linhas por intervalos, indicando quais linhas desejamos, usando a funÃ§Ã£o `dplyr::slice()`, e informando o argumento `n` para o nÃºmero da linha ou intervalo das linhas. Essa funÃ§Ã£o possui variaÃ§Ãµes no sufixo muito interessantes: `dplyr::slice_head()` e `dplyr::slice_tail()` seleciona as primeiras e Ãºltimas linhas, `dplyr::slice_min()` e `dplyr::slice_max()` seleciona linhas com os maiores e menores valores de uma

coluna, e `dplyr::slice_sample()` seleciona linhas aleatoriamente.

```r
## Seleciona linhas 300 atÃ© final
penguins_slice <- penguins |>
    dplyr::slice(300:n())
head(penguins_slice)
```

```
## # A tibble: 6 x 8
##   species   island bill_length_mm bill_depth_mm flipper_length_mm body_mass_g sex
##   <fct>     <fct>           <dbl>         <dbl>             <int>       <int> <fct>
## 1 Chinstrap Dream            50.6          19.4               193        3800 male
## 2 Chinstrap Dream            46.7          17.9               195        3300 female
## 3 Chinstrap Dream            52            19                 197        4150 male
## 4 Chinstrap Dream            50.5          18.4               200        3400 female
## 5 Chinstrap Dream            49.5          19                 200        3800 male
## 6 Chinstrap Dream            46.4          17.8               191        3700 female
## # i 1 more variable: year <int>
```

```r
## Seleciona linhas - head
penguins_slice_head <- penguins |>
    dplyr::slice_head(n = 5)
head(penguins_slice_head)
```

```
## # A tibble: 5 x 8
##   species island    bill_length_mm bill_depth_mm flipper_length_mm body_mass_g sex
##   <fct>   <fct>              <dbl>         <dbl>             <int>       <int> <fct>
## 1 Adelie  Torgersen           39.1          18.7               181        3750 male
## 2 Adelie  Torgersen           39.5          17.4               186        3800 fema~
## 3 Adelie  Torgersen           40.3          18                 195        3250 fema~
## 4 Adelie  Torgersen           NA            NA                  NA          NA <NA>
## 5 Adelie  Torgersen           36.7          19.3               193        3450 fema~
## # i 1 more variable: year <int>
```

```r
## Seleciona linhas - max
penguins_slice_max <- penguins |>
    dplyr::slice_max(body_mass_g, n = 5)
head(penguins_slice_max)
```

```
## # A tibble: 6 x 8
##   species island bill_length_mm bill_depth_mm flipper_length_mm body_mass_g sex
##   <fct>   <fct>           <dbl>         <dbl>             <int>       <int> <fct>
## 1 Gentoo  Biscoe           49.2          15.2               221        6300 male
## 2 Gentoo  Biscoe           59.6          17                 230        6050 male
## 3 Gentoo  Biscoe           51.1          16.3               220        6000 male
## 4 Gentoo  Biscoe           48.8          16.2               222        6000 male
## 5 Gentoo  Biscoe           45.2          16.4               223        5950 male
## 6 Gentoo  Biscoe           49.8          15.9               229        5950 male
## # i 1 more variable: year <int>
```

```r
## Seleciona linhas - sample
penguins_slice_sample <- penguins |>
    dplyr::slice_sample(n = 30)
head(penguins_slice_sample)
```

```
## # A tibble: 6 x 8
##   species island    bill_length_mm bill_depth_mm flipper_length_mm body_mass_g sex
##   <fct>   <fct>              <dbl>         <dbl>             <int>       <int> <fct>
```

```
## 1 Adelie  Biscoe              41.3          21.1                195        4400 male
## 2 Gentoo  Biscoe              44.5          15.7                217        4875 <NA>
## 3 Adelie  Torgersen           41.4          18.5                202        3875 male
## 4 Adelie  Biscoe              37.6          17                  185        3600 fema~
## 5 Adelie  Dream               36            17.9                190        3450 fema~
## 6 Adelie  Biscoe              35.7          16.9                185        3150 fema~
## # i 1 more variable: year <int>
```

8.7.12 distinct()

A Ãºltima operaÃ§Ã£o que apresentaremos para linhas Ã© a retirada de linhas com valores repetidos com base nos valores de uma ou mais colunas, utilizando a funÃ§Ã£o `dplyr::distinct()`. Essa funÃ§Ã£o por padrÃ£o retorna apenas a(s) coluna(s) utilizada(s) para retirar as linhas com valores repetidos, sendo necessÃ¡rio acrescentar o argumento `.keep_all = TRUE` para retornar todas as colunas. Por fim, podemos usar as funÃ§Ãµes de replicaÃ§Ã£o para retirar linhas com valores repetidos para mais de uma coluna, dependendo de resultados booleanos.

```r
## Retirar linhas com valores repetidos
penguins_distinct <- penguins |>
    dplyr::distinct(body_mass_g)
head(penguins_distinct)
```

```
## # A tibble: 6 x 1
##   body_mass_g
##         <int>
## 1        3750
## 2        3800
## 3        3250
## 4          NA
## 5        3450
## 6        3650
```

```r
## Retirar linhas com valores repetidos - manter as outras colunas
penguins_distinct_keep_all <- penguins |>
    dplyr::distinct(body_mass_g, .keep_all = TRUE)
head(penguins_distinct_keep_all)
```

```
## # A tibble: 6 x 8
##   species island    bill_length_mm bill_depth_mm flipper_length_mm body_mass_g sex
##   <fct>   <fct>              <dbl>         <dbl>             <int>       <int> <fct>
## 1 Adelie  Torgersen           39.1          18.7               181        3750 male
## 2 Adelie  Torgersen           39.5          17.4               186        3800 fema~
## 3 Adelie  Torgersen           40.3          18                 195        3250 fema~
## 4 Adelie  Torgersen           NA            NA                  NA          NA <NA>
## 5 Adelie  Torgersen           36.7          19.3               193        3450 fema~
## 6 Adelie  Torgersen           39.3          20.6               190        3650 male
## # i 1 more variable: year <int>
```

```r
## Retirar linhas com valores repetidos para vÃ¡rias colunas
penguins_distinct_keep_all_across <- penguins |>
    dplyr::distinct(across(where(is.integer)), .keep_all = TRUE)
head(penguins_distinct_keep_all_across)
```

```
## # A tibble: 6 x 8
##   species island    bill_length_mm bill_depth_mm flipper_length_mm body_mass_g sex
##   <fct>   <fct>              <dbl>         <dbl>             <int>       <int> <fct>
```

```
## 1 Adelie  Torgersen           39.1          18.7                181        3750 male
## 2 Adelie  Torgersen           39.5          17.4                186        3800 fema~
## 3 Adelie  Torgersen           40.3          18                  195        3250 fema~
## 4 Adelie  Torgersen           NA            NA                   NA          NA <NA>
## 5 Adelie  Torgersen           36.7          19.3                193        3450 fema~
## 6 Adelie  Torgersen           39.3          20.6                190        3650 male
## # i 1 more variable: year <int>
```

8.7.13 count()

Agora entraremos no assunto de resumo das observaÃ§Ãµes. Podemos fazer contagens resumos dos nossos dados, utilizando para isso a funÃ§Ã£o `dplyr::count()`. Essa funÃ§Ã£o contarÃ¡ valores de uma ou mais colunas, geralmente para variÃ¡veis categÃ³ricas, semelhante Ã funÃ§Ã£o *R Base* `table()`, mas num contexto *tidyverse*.

```r
## Contagens de valores para uma coluna
penguins_count <- penguins |>
    dplyr::count(species)
penguins_count
```

```
## # A tibble: 3 x 2
##   species       n
##   <fct>     <int>
## 1 Adelie      152
## 2 Chinstrap    68
## 3 Gentoo      124
```

```r
## Contagens de valores para mais de uma coluna
penguins_count_two <- penguins |>
    dplyr::count(species, island)
penguins_count_two
```

```
## # A tibble: 5 x 3
##   species   island        n
##   <fct>     <fct>     <int>
## 1 Adelie    Biscoe       44
## 2 Adelie    Dream        56
## 3 Adelie    Torgersen    52
## 4 Chinstrap Dream        68
## 5 Gentoo    Biscoe      124
```

8.7.14 group_by()

Uma grande parte das operaÃ§Ãµes feitas nos dados sÃ£o realizadas em grupos definidos por valores de colunas com dados categÃ³ricas. A funÃ§Ã£o `dplyr::group_by()` transforma um `tibble` em um `tibble grouped`, onde as operaÃ§Ãµes sÃ£o realizadas "por grupo". Essa funÃ§Ã£o Ã© utilizada geralmente junto com a funÃ§Ã£o `dplyr::summarise()`, que veremos logo em seguida. O agrupamento nÃ£o altera a aparÃªncia dos dados (alÃ©m de informar como estÃ£o agrupados). A funÃ§Ã£o `dplyr::ungroup()` remove o agrupamento. Podemos ainda usar funÃ§Ãµes de replicaÃ§Ã£o para fazer os agrupamentos para mais de uma coluna, dependendo de resultados booleanos.

```r
## Agrupamento
penguins_group_by <- penguins |>
    dplyr::group_by(species)
head(penguins_group_by)
```

```
## # A tibble: 6 x 8
```

```
## # Groups:   species [1]
##   species island    bill_length_mm bill_depth_mm flipper_length_mm body_mass_g sex
##   <fct>   <fct>              <dbl>         <dbl>             <int>       <int> <fct>
## 1 Adelie  Torgersen           39.1          18.7               181        3750 male
## 2 Adelie  Torgersen           39.5          17.4               186        3800 fema~
## 3 Adelie  Torgersen           40.3          18                 195        3250 fema~
## 4 Adelie  Torgersen           NA            NA                  NA          NA <NA>
## 5 Adelie  Torgersen           36.7          19.3               193        3450 fema~
## 6 Adelie  Torgersen           39.3          20.6               190        3650 male
## # i 1 more variable: year <int>
```

```r
## Agrupamento de vÃ¡rias colunas
penguins_group_by_across <- penguins |>
    dplyr::group_by(across(where(is.factor)))
head(penguins_group_by_across)
```

```
## # A tibble: 6 x 8
## # Groups:   species, island, sex [3]
##   species island    bill_length_mm bill_depth_mm flipper_length_mm body_mass_g sex
##   <fct>   <fct>              <dbl>         <dbl>             <int>       <int> <fct>
## 1 Adelie  Torgersen           39.1          18.7               181        3750 male
## 2 Adelie  Torgersen           39.5          17.4               186        3800 fema~
## 3 Adelie  Torgersen           40.3          18                 195        3250 fema~
## 4 Adelie  Torgersen           NA            NA                  NA          NA <NA>
## 5 Adelie  Torgersen           36.7          19.3               193        3450 fema~
## 6 Adelie  Torgersen           39.3          20.6               190        3650 male
## # i 1 more variable: year <int>
```

8.7.15 summarise()

Como dissemos, muitas vezes queremos resumir nossos dados, principalmente para ter uma noÃ§Ã£o geral das variÃ¡veis (colunas) ou mesmo comeÃ§ar a anÃ¡lise exploratÃ³ria resumindo variÃ¡veis contÃnuas por grupos de variÃ¡veis categÃ³ricas. Dessa forma, ao utilizar a funÃ§Ã£o `dplyr::summarise()` teremos um novo `tibble` com os dados resumidos, que Ã© a agregaÃ§Ã£o ou resumo dos dados atravÃ©s de funÃ§Ãµes. Da mesma forma que outras funÃ§Ãµes, podemos usar funÃ§Ãµes de replicaÃ§Ã£o para resumir valores para mais de uma coluna, dependendo de resultados booleanos.

```r
## Resumo
penguins_summarise <- penguins |>
    dplyr::group_by(species) |>
    dplyr::summarize(body_mass_g_mean = mean(body_mass_g, na.rm = TRUE),
                     body_mass_g_sd = sd(body_mass_g, na.rm = TRUE))
penguins_summarise
```

```
## # A tibble: 3 x 3
##   species   body_mass_g_mean body_mass_g_sd
##   <fct>                <dbl>          <dbl>
## 1 Adelie               3701.           459.
## 2 Chinstrap            3733.           384.
## 3 Gentoo               5076.           504.
```

```r
## Resumo para vÃ¡rias colunas
penguins_summarise_across <- penguins |>
    dplyr::group_by(species) |>
    dplyr::summarize(across(where(is.numeric), ~ mean(.x, na.rm = TRUE)))
penguins_summarise_across
```

```
## # A tibble: 3 x 6
##   species   bill_length_mm bill_depth_mm flipper_length_mm body_mass_g  year
##   <fct>              <dbl>         <dbl>             <dbl>       <dbl> <dbl>
## 1 Adelie              38.8          18.3              190.       3701. 2008.
## 2 Chinstrap           48.8          18.4              196.       3733. 2008.
## 3 Gentoo              47.5          15.0              217.       5076. 2008.
```

8.7.16 bind_rows() e bind_cols()

Muitas vezes teremos de combinar duas ou mais tabelas de dados. Podemos utilizar as funÃ§Ãµes *R Base* `rbind()` e `cbind()`, como vimos no CapÃtulo 7. Entretanto, pode ser interessante avanÃ§ar para as funÃ§Ãµes `dplyr::bind_rows()` e `dplyr::bind_cols()` do formato *tidyverse*. A ideia Ã© muito semelhante: a primeira funÃ§Ã£o combina dados por linhas e a segunda por colunas. Entretanto, hÃ¡ algumas vantagens no uso dessas funÃ§Ãµes, como a identificaÃ§Ã£o das linhas pelo argumento `.id` para a primeira funÃ§Ã£o, e a conferÃªncia do nome das colunas pelo argumento `.name_repair` para a segunda funÃ§Ã£o.

```r
## Selecionar as linhas para dois tibbles
penguins_01 <- dplyr::slice(penguins, 1:5)
penguins_02 <- dplyr::slice(penguins, 51:55)

## Combinar as linhas
penguins_bind_rows <- dplyr::bind_rows(penguins_01, penguins_02, .id = "id")
head(penguins_bind_rows)
```

```
## # A tibble: 6 x 9
##   id    species island    bill_length_mm bill_depth_mm flipper_length_mm body_mass_g
##   <chr> <fct>   <fct>              <dbl>         <dbl>             <int>       <int>
## 1 1     Adelie  Torgersen           39.1          18.7               181        3750
## 2 1     Adelie  Torgersen           39.5          17.4               186        3800
## 3 1     Adelie  Torgersen           40.3          18                 195        3250
## 4 1     Adelie  Torgersen           NA            NA                  NA          NA
## 5 1     Adelie  Torgersen           36.7          19.3               193        3450
## 6 2     Adelie  Biscoe              39.6          17.7               186        3500
## # i 2 more variables: sex <fct>, year <int>
```

```r
## Combinar as colunas
penguins_bind_cols <- dplyr::bind_cols(penguins_01, penguins_02, .name_repair = "unique")
```

```
## New names:
## * `species` -> `species...1`
## * `island` -> `island...2`
## * `bill_length_mm` -> `bill_length_mm...3`
## * `bill_depth_mm` -> `bill_depth_mm...4`
## * `flipper_length_mm` -> `flipper_length_mm...5`
## * `body_mass_g` -> `body_mass_g...6`
## * `sex` -> `sex...7`
## * `year` -> `year...8`
## * `species` -> `species...9`
## * `island` -> `island...10`
## * `bill_length_mm` -> `bill_length_mm...11`
## * `bill_depth_mm` -> `bill_depth_mm...12`
## * `flipper_length_mm` -> `flipper_length_mm...13`
## * `body_mass_g` -> `body_mass_g...14`
## * `sex` -> `sex...15`
## * `year` -> `year...16`
```

```r
head(penguins_bind_cols)
```

```
## # A tibble: 5 x 16
##   species...1 island...2 bill_length_mm...3 bill_depth_mm...4 flipper_length_mm...5
##   <fct>       <fct>                   <dbl>             <dbl>                 <int>
## 1 Adelie      Torgersen                39.1              18.7                   181
## 2 Adelie      Torgersen                39.5              17.4                   186
## 3 Adelie      Torgersen                40.3              18                     195
## 4 Adelie      Torgersen                NA                NA                      NA
## 5 Adelie      Torgersen                36.7              19.3                   193
## # i 11 more variables: body_mass_g...6 <int>, sex...7 <fct>, year...8 <int>,
## #   species...9 <fct>, island...10 <fct>, bill_length_mm...11 <dbl>,
## #   bill_depth_mm...12 <dbl>, flipper_length_mm...13 <int>, body_mass_g...14 <int>,
## #   sex...15 <fct>, year...16 <int>
```

8.7.17 *_join()

Finalmente, veremos o Ãºltimo conjunto de funÃ§Ãµes do pacote `dplyr`, a junÃ§Ã£o de tabelas. Nessa operaÃ§Ã£o, fazemos a combinaÃ§Ã£o de pares de conjunto de dados tabulares por uma ou mais colunas chaves. HÃ¡ dois tipos de junÃ§Ãµes: junÃ§Ã£o de modificaÃ§Ã£o e junÃ§Ã£o de filtragem. A junÃ§Ã£o de modificaÃ§Ã£o primeiro combina as observaÃ§Ãµes por suas chaves e, em seguida, copia as variÃ¡veis (colunas) de uma tabela para a outra. Ã fundamental destacar a importÃ¢ncia da coluna chave, que Ã© indicada pelo argumento `by`. Essa coluna deve conter elementos que sejam comuns Ã s duas tabelas para que haja a combinaÃ§Ã£o dos elementos.

Existem quatro tipos de junÃ§Ãµes de modificaÃ§Ãµes, que sÃ£o realizadas pelas funÃ§Ãµes: `dplyr::inner_join()`, `dplyr::left_join()`, `dplyr::full_join()` e `dplyr::right_join()`, e que podem ser representadas na Figura 8.3.

Figura 8.3: Diferentes tipos de joins, representados com um diagrama de Venn.

Considerando a nomenclatura de duas tabelas de dados por `x` e `y`, temos:

- `inner_join(x, y)`: mantÃ©m apenas as observaÃ§Ãµes em `x` e em `y`
- `left_join(x, y)`: mantÃ©m todas as observaÃ§Ãµes em `x`
- `right_join(x, y)`: mantÃ©m todas as observaÃ§Ãµes em `y`
- `full_join(x, y)`: mantÃ©m todas as observaÃ§Ãµes em `x` e em `y`

Aqui, vamos demostrar apenas a funÃ§Ã£o `dplyr::left_join()`, combinando um `tibble` de coordenadas geogrÃ¡ficas das ilhas com o conjunto de dados do penguins.

```r
## Adicionar uma coluna chave de ids
penguin_islands <- tibble(
    island = c("Torgersen", "Biscoe", "Dream", "Alpha"),
    longitude = c(-64.083333, -63.775636, -64.233333, -63),
    latitude = c(-64.766667, -64.818569, -64.733333, -64.316667))
```

```r
## JunÃ§Ã£o - left
penguins_left_join <- dplyr::left_join(penguins, penguin_islands, by = "island")
head(penguins_left_join)
```

```
## # A tibble: 6 x 10
##   species island    bill_length_mm bill_depth_mm flipper_length_mm body_mass_g sex
##   <fct>   <chr>              <dbl>         <dbl>             <int>       <int> <fct>
## 1 Adelie  Torgersen           39.1          18.7               181        3750 male
## 2 Adelie  Torgersen           39.5          17.4               186        3800 fema~
## 3 Adelie  Torgersen           40.3          18                 195        3250 fema~
## 4 Adelie  Torgersen           NA            NA                  NA          NA <NA>
## 5 Adelie  Torgersen           36.7          19.3               193        3450 fema~
## 6 Adelie  Torgersen           39.3          20.6               190        3650 male
## # i 3 more variables: year <int>, longitude <dbl>, latitude <dbl>
```

JÃ¡ o segundo tipo de junÃ§Ã£o, a junÃ§Ã£o de filtragem combina as observaÃ§Ãµes da mesma maneira que as junÃ§Ãµes de modificaÃ§Ã£o, mas afetam as observaÃ§Ãµes (linhas), nÃ£o as variÃ¡veis (colunas). Existem dois tipos.

- `semi_join(x, y)`: mantÃ©m todas as observaÃ§Ãµes em `x` que tÃªm uma correspondÃªncia em `y`
- `anti_join(x, y)`: elimina todas as observaÃ§Ãµes em `x` que tÃªm uma correspondÃªncia em `y`

De forma geral, *semi-joins* sÃ£o Ãºteis para corresponder tabelas de resumo filtradas de volta Ã s linhas originais, removendo as linhas que nÃ£o estavam antes do join. JÃ¡ *anti-joins* sÃ£o Ãºteis para diagnosticar incompatibilidades de junÃ§Ã£o, por exemplo, ao verificar os elementos que nÃ£o combinam entre duas tabelas de dados.

8.7.18 OperaÃ§Ãµes de conjuntos e comparaÃ§Ã£o de dados

Temos ainda operaÃ§Ãµes de conjuntos e comparaÃ§Ã£o de dados.

- `union(x, y)`: retorna todas as linhas que aparecem em `x`, `y` ou mais dos conjuntos de dados
- `interesect(x, y)`: retorna apenas as linhas que aparecem em `x` e em `y`
- `setdiff(x, y)`: retorna as linhas que aparecem `x`, mas nÃ£o em `y`
- `setequal(x, y)`: retorna se `x` e `y` sÃ£o iguais e quais suas diferenÃ§as

Para se aprofundar no tema, recomendamos a leitura do CapÃtulo 13 Relational data de Wickham & Grolemund [-@wickham2017].

8.8 stringr

O pacote `stringr` fornece um conjunto de funÃ§Ãµes para a manipulaÃ§Ã£o de caracteres ou strings. O pacote concentra-se nas funÃ§Ãµes de manipulaÃ§Ã£o mais importantes e comumente usadas. Para funÃ§Ãµes mais especÃficas, recomenda-se usar o pacote `stringi`, que fornece um conjunto mais abrangente de funÃ§Ãµes. As funÃ§Ãµes do `stringr` podem ser agrupadas em algumas operaÃ§Ãµes para tarefas especÃficas como: i) correspondÃªncia de padrÃµes, ii) retirar e acrescentar espaÃ§os em branco, iii) mudar maiÃºsculas e minÃºsculas, alÃ©m de muitas outras operaÃ§Ãµes com caracteres.

Todas as funÃ§Ãµes deste pacote sÃ£o listadas na pÃ¡gina de referÃªncia do pacote.

Demonstraremos algumas funÃ§Ãµes para algumas operaÃ§Ãµes mais comuns, utilizando um vetor de um elemento, com o string "penguins".

Podemos explorar o comprimento de strings com a funÃ§Ã£o `stringr::str_length()`.

```r
## Comprimento
stringr::str_length(string = "penguins")
```

```
## [1] 8
```

Extrair um string por sua posiÃ§Ã£o usando a funÃ§Ã£o `stringr::str_sub()` ou por um padrÃ£o com `stringr::str_extract()`.

```r
## Extrair pela posiÃ§Ã£o
stringr::str_sub(string = "penguins", end = 3)
```

```
## [1] "pen"
```

```r
## Extrair por padrÃ£o
stringr::str_extract(string = "penguins", pattern = "p")
```

```
## [1] "p"
```

Substituir strings por outros strings com `stringr::str_replace()`.

```r
## Substituir
stringr::str_replace(string = "penguins", pattern = "i", replacement = "y")
```

```
## [1] "penguyns"
```

Separar strings por um padrÃ£o com a funÃ§Ã£o `stringr::str_split()`.

```r
## Separar
stringr::str_split(string = "p-e-n-g-u-i-n-s", pattern = "-", simplify = TRUE)
```

```
##      [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8]
## [1,] "p"  "e"  "n"  "g"  "u"  "i"  "n"  "s"
```

Inserir espaÃ§os em brancos pela esquerda, direita ou ambos com a funÃ§Ã£o `stringr::str_pad()`.

```r
## Inserir espacos em branco
stringr::str_pad(string = "penguins", width = 10, side = "left")
```

```
## [1] "  penguins"
```

```r
stringr::str_pad(string = "penguins", width = 10, side = "right")
```

```
## [1] "penguins  "
```

```r
stringr::str_pad(string = "penguins", width = 10, side = "both")
```

```
## [1] " penguins "
```

TambÃ©m podemos remover espaÃ§os em branco da esquerda, direita ou ambos, utilizando `stringr::str_trim()`.

```r
## Remover espacos em branco
stringr::str_trim(string = " penguins ", side = "left")
```

```
## [1] "penguins "
```

```r
stringr::str_trim(string = " penguins ", side = "right")
```

```
## [1] " penguins"
```

```r
stringr::str_trim(string = " penguins ", side = "both")
```

```
## [1] "penguins"
```

Podemos tambÃ©m alterar minÃºsculas e maiÃºsculas em diferentes posiÃ§Ãµes do string, com vÃ¡rias funÃ§Ãµes.

```r
## Alterar minÃºsculas e maiÃºsculas
stringr::str_to_lower(string = "Penguins")
```

```
## [1] "penguins"
```

```
stringr::str_to_upper(string = "penguins")
```

```
## [1] "PENGUINS"
```

```
stringr::str_to_sentence(string = "penGuins")
```

```
## [1] "Penguins"
```

```
stringr::str_to_title(string = "penGuins")
```

```
## [1] "Penguins"
```

Podemos ainda ordenar os elementos de um vetor por ordem alfabÃ©tica de forma crescente ou decrescente, usando `stringr::str_sort()`.

```
## Ordenar
stringr::str_sort(x = letters)
```

```
##  [1] "a" "b" "c" "d" "e" "f" "g" "h" "i" "j" "k" "l" "m" "n" "o" "p" "q" "r" "s" "t"
## [21] "u" "v" "w" "x" "y" "z"
```

```
stringr::str_sort(x = letters, dec = TRUE)
```

```
##  [1] "z" "y" "x" "w" "v" "u" "t" "s" "r" "q" "p" "o" "n" "m" "l" "k" "j" "i" "h" "g"
## [21] "f" "e" "d" "c" "b" "a"
```

Podemos ainda utilizar essas funÃ§Ãµes em complemento com o pacote `dplyr`, para alterar os strings de colunas ou nome das colunas.

```
## Alterar valores das colunas
penguins_stringr_valores <- penguins |>
    dplyr::mutate(species = stringr::str_to_lower(species))

## Alterar nome das colunas
penguins_stringr_nomes <- penguins |>
    dplyr::rename_with(stringr::str_to_title)
```

Para se aprofundar no tema, recomendamos a leitura do CapÃtulo 14 Strings de Wickham & Grolemund [-@wickham2017].

8.9 forcats

O pacote `forcats` fornece um conjunto de ferramentas Ãºteis para facilitar a manipulaÃ§Ã£o de fatores. Como dito no CapÃtulo 7, usamos fatores geralmente quando temos dados categÃ³ricos, que sÃ£o variÃ¡veis que possuem um conjunto de valores fixos e conhecidos. As funÃ§Ãµes sÃ£o utilizadas principalmente para: i) mudar a ordem dos nÃveis, ii) mudar os valores dos nÃveis, iii) adicionar e remover nÃveis, iv) combinar mÃºltiplos nÃveis, alÃ©m de outras operaÃ§Ãµes.

Todas as funÃ§Ãµes deste pacote sÃ£o listadas na pÃ¡gina de referÃªncia do pacote.

Vamos utilizar ainda os dados `penguins` e `penguins_raw` para exemplificar o uso do pacote `forcats`.

```
## Carregar o pacote palmerpenguins
library(palmerpenguins)
```

Primeiramente, vamos converter dados de string para fator, utilizando a funÃ§Ã£o `forcats::as_factor()`.

```r
## String
forcats::as_factor(penguins_raw$Species) |> head()
```

```
## [1] Adelie Penguin (Pygoscelis adeliae) Adelie Penguin (Pygoscelis adeliae)
## [3] Adelie Penguin (Pygoscelis adeliae) Adelie Penguin (Pygoscelis adeliae)
## [5] Adelie Penguin (Pygoscelis adeliae) Adelie Penguin (Pygoscelis adeliae)
## 3 Levels: Adelie Penguin (Pygoscelis adeliae) ...
```

Podemos facilmente mudar o nome dos nÃveis utilizando a funÃ§Ã£o `forcats::fct_recode()`.

```r
## Mudar o nome dos nÃveis
forcats::fct_recode(penguins$species, a = "Adelie", c = "Chinstrap", g = "Gentoo") |> head()
```

```
## [1] a a a a a a
## Levels: a c g
```

Para inverter os nÃveis, usamos a funÃ§Ã£o `forcats::fct_rev()`.

```r
## Inverter os nÃveis
forcats::fct_rev(penguins$species) |> head()
```

```
## [1] Adelie Adelie Adelie Adelie Adelie Adelie
## Levels: Gentoo Chinstrap Adelie
```

Uma operaÃ§Ã£o muito comum com fatores Ã© mudar a ordem dos nÃveis. Quando precisamos especificar a ordem dos nÃveis, podemos fazer essa operaÃ§Ã£o manualmente com a funÃ§Ã£o `forcats::fct_relevel()`.

```r
## Especificar a ordem dos nÃveis
forcats::fct_relevel(penguins$species, "Chinstrap", "Gentoo", "Adelie") |> head()
```

```
## [1] Adelie Adelie Adelie Adelie Adelie Adelie
## Levels: Chinstrap Gentoo Adelie
```

Como vimos, a reordenaÃ§Ã£o dos nÃveis pode ser feita manualmente. Mas existem outras formas automÃ¡ticas de reordenaÃ§Ã£o seguindo algumas regras, para as quais existem funÃ§Ãµes especÃficas.

- `forcats::fct_inorder()`: pela ordem em que aparecem pela primeira vez
- `forcats::fct_infreq()`: por nÃºmero de observaÃ§Ãµes com cada nÃvel (decrescente, i.e., o maior primeiro)
- `forcats::fct_inseq()`: pelo valor numÃ©rico do nÃvel

```r
## NÃveis pela ordem em que aparecem
forcats::fct_inorder(penguins$species) |> head()
```

```
## [1] Adelie Adelie Adelie Adelie Adelie Adelie
## Levels: Adelie Gentoo Chinstrap
```

```r
## Ordem (decrescente) de frequÃªncia
forcats::fct_infreq(penguins$species) |> head()
```

```
## [1] Adelie Adelie Adelie Adelie Adelie Adelie
## Levels: Adelie Gentoo Chinstrap
```

Por fim, podemos fazer a agregaÃ§Ã£o de nÃveis raros em um nÃvel utilizando a funÃ§Ã£o `forcats::fct_lump()`.

```r
## AgregaÃ§Ã£o de nÃveis raros em um nÃvel
forcats::fct_lump(penguins$species) |> head()
```

```
## [1] Adelie Adelie Adelie Adelie Adelie Adelie
## Levels: Adelie Gentoo Other
```

Podemos ainda utilizar essas funÃ§Ãµes em complemento com o pacote `dplyr` para fazer manipulaÃ§Ãµes de fatores nas colunas de `tibbles`.

```
## Transformar vÃ¡rias colunas em fator
penguins_raw_multi_factor <- penguins_raw |>
    dplyr::mutate(across(where(is.character), forcats::as_factor))
```

Para se aprofundar no tema, recomendamos a leitura do CapÃtulo 15 Factors de Wickham & Grolemund [-@wickham2017].

8.10 lubridate

O pacote `lubridate` fornece um conjunto de funÃ§Ãµes para a manipulaÃ§Ã£o de dados de data e horÃ¡rio. Dessa forma, esse pacote facilita a manipulaÃ§Ã£o dessa classe de dado no R, pois geralmente esses dados nÃ£o sÃ£o intuitivos e mudam dependendo do tipo de objeto de data e horÃ¡rio. AlÃ©m disso, os mÃ©todos que usam datas e horÃ¡rios devem levar em consideraÃ§Ã£o fusos horÃ¡rios, anos bissextos, horÃ¡rios de verÃ£o, alÃ©m de outras particularidades. Existem diversas funÃ§Ãµes nesse pacote, sendo as mesmas focadas em: i) transformaÃ§Ãµes de data/horÃ¡rio, ii) componentes, iii) arredondamentos, iv) duraÃ§Ãµes, v) perÃodos, vi) intervalos, alÃ©m de muitas outras funÃ§Ãµes especÃficas.

Todas as funÃ§Ãµes deste pacote sÃ£o listadas na pÃ¡gina de referÃªncia do pacote.

Apesar de estar inserido no escopo do *tidyverse*, este pacote nÃ£o Ã© carregado com os demais, requisitando seu carregamento solo.

```
## Carregar
library(lubridate)
```

Existem trÃªs tipos de dados data/horÃ¡rio:

- **Data**: tempo em dias, meses e anos `<date>`
- **HorÃ¡rio**: tempo dentro de um dia `<time>`
- **Data-horÃ¡rio**: tempo em um instante (data mais tempo) `<dttm>`

Para trabalhar exclusivamente com horÃ¡rios, podemos utilizar o pacote `hms`.

Ã fundamental tambÃ©m destacar que algumas letras terÃ£o um significado temporal, sendo abreviaÃ§Ãµes de diferentes perÃodos em inglÃªs: **y**ear (ano), **m**onth (mÃªs), **w**eak (semana), **d**ay (dia), **h**our (hora), **m**inute (minuto), e **s**econd (segundo).

Para acessar a informaÃ§Ã£o da data e horÃ¡rios atuais podemos utilizar as funÃ§Ãµes `lubridate::today()` e `lubridate::now()`.

```
## Extrair a data nesse instante
lubridate::today()
```

```
## [1] "2024-03-03"
```

```
## Extrair a data e tempo nesse instante
lubridate::now()
```

```
## [1] "2024-03-03 17:14:51 -03"
```

AlÃ©m dessas informaÃ§Ãµes instantÃ¢neas, existem trÃªs maneiras de criar um dado de data/horÃ¡rio.

- De um string
- De componentes individuais de data e horÃ¡rio
- De um objeto de data/horÃ¡rio existente

Os dados de data/horÃ¡rio geralmente estÃ£o no formato de strings. Podemos transformar os dados especificando a ordem dos seus componentes, ou seja, a ordem em que ano, mÃªs e dia aparecem no string, usando as letras `y` (ano), `m` (mÃªs) e `d` (dia) na mesma ordem, por exemplo, `lubridate::dmy()`.

```
## Strings e nÃºmeros para datas
lubridate::dmy("03-03-2021")
```

```
## [1] "2021-03-03"
```

Essas funÃ§Ãµes tambÃ©m aceitam nÃºmeros sem aspas, alÃ©m de serem muito versÃ¡teis e funcionarem em outros diversos formatos.

```
## Strings e nÃºmeros para datas
lubridate::dmy("03-Mar-2021")
lubridate::dmy(03032021)
lubridate::dmy("03032021")
lubridate::dmy("03/03/2021")
lubridate::dmy("03.03.2021")
```

AlÃ©m da data, podemos especificar horÃ¡rios atrelados a essas datas. Para criar uma data com horÃ¡rio adicionamos um underscore (`_`) e `h` (hora), `m` (minuto) e `s` (segundo) ao nome da funÃ§Ã£o, alÃ©m do argumento `tz` para especificar o *fuso horÃ¡rio* (tema tratado mais adiante nessa seÃ§Ã£o).

```
## Especificar horÃ¡rios e fuso horÃ¡rio
lubridate::dmy_h("03-03-2021 13")
```

```
## [1] "2021-03-03 13:00:00 UTC"
```

```
lubridate::dmy_hm("03-03-2021 13:32")
```

```
## [1] "2021-03-03 13:32:00 UTC"
```

```
lubridate::dmy_hms("03-03-2021 13:32:01")
```

```
## [1] "2021-03-03 13:32:01 UTC"
```

```
lubridate::dmy_hms("03-03-2021 13:32:01", tz = "America/Sao_Paulo")
```

```
## [1] "2021-03-03 13:32:01 -03"
```

Podemos ainda ter componentes individuais de data/horÃ¡rio em mÃºltiplas colunas. Para realizar essa transformaÃ§Ã£o, podemos usar as funÃ§Ãµes `lubridate::make_date()` e `lubridate::make_datetime()`.

```
## Dados com componentes individuais
dados <- tibble::tibble(
    ano = c(2021, 2021, 2021),
    mes = c(1, 2, 3),
    dia = c(12, 20, 31),
    hora = c(2, 14, 18),
    minuto = c(2, 44, 55))

## Data de componentes individuais
dados |>
    dplyr::mutate(data = lubridate::make_datetime(ano, mes, dia, hora, minuto))
```

```
## # A tibble: 3 x 6
##     ano   mes   dia  hora minuto data
##   <dbl> <dbl> <dbl> <dbl>  <dbl> <dttm>
## 1  2021     1    12     2      2 2021-01-12 02:02:00
## 2  2021     2    20    14     44 2021-02-20 14:44:00
```

```
## 3  2021     3    31    18     55 2021-03-31 18:55:00
```

Por fim, podemos criar datas modificando entre data/horÃ¡rio e data, utilizando as funÃ§Ãµes `lubridate::as_datetime()` e `lubridate::as_date()`.

```r
## Data para data-horÃ¡rio
lubridate::as_datetime(today())
```

```
## [1] "2024-03-03 UTC"
```

```r
## Data-horÃ¡rio para data
lubridate::as_date(now())
```

```
## [1] "2024-03-03"
```

Uma vez que entendemos como podemos criar dados de data/horÃ¡rio, podemos explorar funÃ§Ãµes para acessar e definir componentes individuais. Para essa tarefa existe uma grande quantidade de funÃ§Ãµes para acessar partes especÃficas de datas e horÃ¡rios.

- `year()`: acessa o ano
- `month()`: acessa o mÃªs
- `month()`: acessa o dia
- `yday()`: acessa o dia do ano
- `mday()`: acessa o dia do mÃªs
- `wday()`: acessa o dia da semana
- `hour()`: acessa as horas
- `minute()`: acessa os minutos
- `second()`: acessa os segundos

```r
## Extrair
agora <- lubridate::now()
lubridate::year(agora)
```

```
## [1] 2024
```

```r
lubridate::month(agora)
```

```
## [1] 3
```

```r
lubridate::day(agora)
```

```
## [1] 3
```

```r
lubridate::wday(agora)
```

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

```r
lubridate::second(agora)
```

```
## [1] 52.17942
```

AlÃ©m de acessar componentes de datas e horÃ¡rios, podemos usar essas funÃ§Ãµes para fazer a inclusÃ£o de informaÃ§Ãµes de datas e horÃ¡rios.

```r
## Data
data <- dmy_hms("04-03-2021 01:04:56")

## Incluir
lubridate::year(data) <- 2020
lubridate::month(data) <- 01
lubridate::hour(data) <- 13
```

Mais convenientemente, podemos utilizar a funÃ§Ã£o `update()` para alterar vÃ¡rios valores de uma vez.

```
## Incluir vÃ¡rios valores
update(data, year = 2020, month = 1, mday = 1, hour = 1)
```

```
## [1] "2020-01-01 01:04:56 UTC"
```

Muitas vezes precisamos fazer operaÃ§Ãµes com datas, como: adiÃ§Ã£o, subtraÃ§Ã£o, multiplicaÃ§Ã£o e divisÃ£o. Para tanto, Ã© preciso entender trÃªs classes importantes que representam intervalos de tempo.

- **DuraÃ§Ãµes**: representam um nÃºmero exato de segundos
- **PerÃodos**: representam unidades humanas como semanas e meses
- **Intervalos**: representam um ponto inicial e final

Quando fazemos uma subtraÃ§Ã£o de datas, criamos um objeto da classe `difftime`. Essa classe pode ser um pouco complicada de trabalhar, entÃ£o dentro do *lubridate* usamos funÃ§Ãµes que convertem essa classe em duraÃ§Ã£o, da classe `Duration`. As duraÃ§Ãµes sempre registram o intervalo de tempo em segundos, com alguma unidade de tempo maior entre parÃªnteses. HÃ¡ uma sÃ©rie de funÃ§Ãµes para tratar dessa classe.

- `duration()`: cria data em duraÃ§Ã£o
- `as.duration()`: converte datas em duraÃ§Ã£o
- `dyears()`: duraÃ§Ã£o de anos
- `dmonths()`: duraÃ§Ã£o de meses
- `dweeks()`: duraÃ§Ã£o de semanas
- `ddays()`: duraÃ§Ã£o de dias
- `dhours()`: duraÃ§Ã£o de horas
- `dminutes()`: duraÃ§Ã£o de minutos
- `dseconds()`: duraÃ§Ã£o de segundos

```
## SubtraÃ§Ã£o de datas
tempo_estudando_r <- lubridate::today() - lubridate::dmy("30-11-2011")

## ConversÃ£o para duraÃ§Ã£o
tempo_estudando_r_dur <- lubridate::as.duration(tempo_estudando_r)

## Criando duraÃ§Ãµes
lubridate::duration(90, "seconds")
```

```
## [1] "90s (~1.5 minutes)"
```

```
lubridate::duration(1.5, "minutes")
```

```
## [1] "90s (~1.5 minutes)"
```

```
lubridate::duration(1, "days")
```

```
## [1] "86400s (~1 days)"
```

```
## TransformaÃ§Ã£o da duraÃ§Ã£o
lubridate::dseconds(100)
```

```
## [1] "100s (~1.67 minutes)"
```

```
lubridate::dminutes(100)
```

```
## [1] "6000s (~1.67 hours)"
```

```
lubridate::dhours(100)
```

```
## [1] "360000s (~4.17 days)"
```

```r
lubridate::ddays(100)
```

```
## [1] "8640000s (~14.29 weeks)"
```

```r
lubridate::dweeks(100)
```

```
## [1] "60480000s (~1.92 years)"
```

```r
lubridate::dyears(100)
```

```
## [1] "3155760000s (~100 years)"
```

Podemos ainda utilizar as duraÃ§Ãµes para fazer operaÃ§Ãµes aritmÃ©ticas com datas como adiÃ§Ã£o, subtraÃ§Ã£o e multiplicaÃ§Ã£o.

```r
## Somando duraÃ§Ãµes a datas
lubridate::today() + lubridate::ddays(1)
```

```
## [1] "2024-03-04"
```

```r
## Subtraindo duraÃ§Ãµes de datas
lubridate::today() - lubridate::dyears(1)
```

```
## [1] "2023-03-03 18:00:00 UTC"
```

```r
## Multiplicando duraÃ§Ãµes
2 * dyears(2)
```

```
## [1] "126230400s (~4 years)"
```

AlÃ©m das duraÃ§Ãµes, podemos usar perÃodos, que sÃ£o extensÃµes de tempo nÃ£o fixados em segundos como as duraÃ§Ãµes, mas flexÃveis, com o tempo em dias, semanas, meses ou anos, permitindo uma interpretaÃ§Ã£o mais intuitiva das datas. Novamente, hÃ¡ uma sÃ©rie de funÃ§Ãµes para realizar essas operaÃ§Ãµes.

- `period()`: cria data em perÃodo
- `as.period()`: converte datas em perÃodo
- `seconds()`: perÃodo em segundos
- `minutes()`: perÃodo em minutos
- `hours()`: perÃodo em horas
- `days()`: perÃodo em dias
- `weeks()`: perÃodo em semanas
- `months()`: perÃodo em meses
- `years()`: perÃodo em anos

```r
## Criando perÃodos
period(c(90, 5), c("second", "minute"))
```

```
## [1] "5M 90S"
```

```r
period(c(3, 1, 2, 13, 1), c("second", "minute", "hour", "day", "week"))
```

```
## [1] "20d 2H 1M 3S"
```

```r
## TransformaÃ§Ã£o de perÃodos
lubridate::seconds(100)
```

```
## [1] "100S"
```

```r
lubridate::minutes(100)
```

```
## [1] "100M 0S"
```

```r
lubridate::hours(100)
```

```
## [1] "100H 0M 0S"
```

```r
lubridate::days(100)
```

```
## [1] "100d 0H 0M 0S"
```

```r
lubridate::weeks(100)
```

```
## [1] "700d 0H 0M 0S"
```

```r
lubridate::years(100)
```

```
## [1] "100y 0m 0d 0H 0M 0S"
```

AlÃ©m disso, podemos fazer operaÃ§Ãµes com os perÃodos, somando e subtraindo.

```r
## Somando datas
lubridate::today() + lubridate::weeks(10)
```

```
## [1] "2024-05-12"
```

```r
## Subtraindo datas
lubridate::today() - lubridate::weeks(10)
```

```
## [1] "2023-12-24"
```

```r
## Criando datas recorrentes
lubridate::today() + lubridate::weeks(0:10)
```

```
##  [1] "2024-03-03" "2024-03-10" "2024-03-17" "2024-03-24" "2024-03-31" "2024-04-07"
##  [7] "2024-04-14" "2024-04-21" "2024-04-28" "2024-05-05" "2024-05-12"
```

Por fim, intervalos sÃ£o perÃodos de tempo limitados por duas datas, possuindo uma duraÃ§Ã£o com um ponto de partida, que o faz preciso para determinar uma duraÃ§Ã£o. Intervalos sÃ£o objetos da classe `Interval`. Da mesma forma que para duraÃ§Ã£o e perÃodos, hÃ¡ uma sÃ©rie de funÃ§Ãµes para realizar essas operaÃ§Ãµes.

- `interval()`: cria data em intervalo
- `%--%`: cria data em intervalo
- `as.interval()`: converte datas em intervalo
- `int_start()`: acessa ou atribui data inicial de um intervalo
- `int_end()`: acessa ou atribui data final de um intervalo
- `int_length()`: comprimento de um intervalo em segundos
- `int_flip()`: inverte a ordem da data de inÃcio e da data de tÃ©rmino em um intervalo
- `int_shift()`: desloca as datas de inÃcio e tÃ©rmino de um intervalo
- `int_aligns()`: testa se dois intervalos compartilham um ponto final
- `int_standardize()`: garante que todos os intervalos sejam positivos
- `int_diff()`: retorna os intervalos que ocorrem entre os elementos de data/horÃ¡rio
- `int_overlaps()`: testa se dois intervalos se sobrepÃµem
- `%within%`: testa se o primeiro intervalo estÃ¡ contido no segundo

```r
## Criando duas datas - inÃcio de estudos do R e nascimento do meu filho
r_inicio <- lubridate::dmy("30-11-2011")
filho_nascimento <- lubridate::dmy("26-09-2013")
r_hoje <- lubridate::today()

## Criando intervalos - interval
r_intervalo <- lubridate::interval(r_inicio, r_hoje)
```

```r
## Criando intervalos - interval %--%
filho_intervalo <- filho_nascimento %--% lubridate::today()

## OperaÃ§Ãµes com intervalos
lubridate::int_start(r_intervalo)
```

```
## [1] "2011-11-30 UTC"
```

```r
lubridate::int_end(r_intervalo)
```

```
## [1] "2024-03-03 UTC"
```

```r
lubridate::int_length(r_intervalo)
```

```
## [1] 386812800
```

```r
lubridate::int_flip(r_intervalo)
```

```
## [1] 2024-03-03 UTC--2011-11-30 UTC
```

```r
lubridate::int_shift(r_intervalo, duration(days = 30))
```

```
## [1] 2011-12-30 UTC--2024-04-02 UTC
```

Uma operaÃ§Ã£o de destaque Ã© verificar a sobreposiÃ§Ã£o entre dois intervalos.

```r
## Verificar sobreposiÃ§Ã£o - int_overlaps
lubridate::int_overlaps(r_intervalo, filho_intervalo)
```

```
## [1] TRUE
```

```r
## Verificar se intervalo estÃ¡ contido
r_intervalo %within% filho_intervalo
```

```
## [1] FALSE
```

```r
filho_intervalo %within% r_intervalo
```

```
## [1] TRUE
```

Podemos ainda calcular quantos perÃodos existem dentro de um intervalo, utilizando as operaÃ§Ãµes de `/` e `%/%`.

```r
## PerÃodos dentro de um intervalo - anos
r_intervalo / lubridate::years()
```

```
## [1] 12.25683
```

```r
r_intervalo %/% lubridate::years()
```

```
## [1] 12
```

```r
## PerÃodos dentro de um intervalo - dias e semandas
filho_intervalo / lubridate::days()
```

```
## [1] 3811
```

```r
filho_intervalo / lubridate::weeks()
```

```
## [1] 544.4286
```

Ainda podemos fazer transformaÃ§Ãµes dos dados para perÃodos e ter todas as unidades de data e tempo que o intervalo compreende.

```r
## Tempo total estudando R
lubridate::as.period(r_intervalo)
```

```
## [1] "12y 3m 3d 0H 0M 0S"
```

```r
## Idade do meu filho
lubridate::as.period(filho_intervalo)
```

```
## [1] "10y 5m 6d 0H 0M 0S"
```

Por fim, fusos horÃ¡rios tendem a ser um fator complicador quando precisamos analisar informaÃ§Ãµes instantÃ¢neas de tempo (horÃ¡rio) de outras partes do planeta, ou mesmo fazer conversÃµes dos horÃ¡rios. No `lubridate` hÃ¡ funÃ§Ãµes para ajudar nesse sentido. Para isso, podemos utilizar a funÃ§Ã£o `lubridate::with_tz()` e no argumento `tzone` informar o fuso horÃ¡rio para a transformaÃ§Ã£o do horÃ¡rio.

Podemos descobrir o fuso horÃ¡rio que o R estÃ¡ considerando com a funÃ§Ã£o `Sys.timezone()`.

```r
## Fuso horÃ¡rio no R
Sys.timezone()
```

```
## [1] "America/Cayenne"
```

No R hÃ¡ uma listagem dos nomes dos fusos horÃ¡rios que podemos utilizar no argumento `tzone` para diferentes fusos horÃ¡rios.

```r
## Verificar os fuso horÃ¡rios
length(OlsonNames())
```

```
## [1] 596
```

```r
head(OlsonNames())
```

```
## [1] "Africa/Abidjan"     "Africa/Accra"       "Africa/Addis_Ababa"
## [4] "Africa/Algiers"     "Africa/Asmara"      "Africa/Asmera"
```

Podemos nos perguntar que horas sÃ£o em outra parte do globo ou fazer as conversÃµes facilmente no *lubridate*.

```r
## Que horas sÃ£o em...
lubridate::with_tz(lubridate::now(), tzone = "GMT")
```

```
## [1] "2024-03-03 20:14:52 GMT"
```

```r
lubridate::with_tz(lubridate::now(), tzone = "Europe/Berlin")
```

```
## [1] "2024-03-03 21:14:52 CET"
```

```r
## Altera o fuso sem mudar a hora
lubridate::force_tz(lubridate::now(), tzone = "GMT")
```

```
## [1] "2024-03-03 17:14:52 GMT"
```

Para se aprofundar no tema, recomendamos a leitura do CapÃtulo 16 Dates and times de Wickham & Grolemund [-@wickham2017].

8.11 purrr

O pacote `purrr` implementa a *ProgramaÃ§Ã£o Funcional* no R, fornecendo um conjunto completo e consistente de ferramentas para trabalhar com funÃ§Ãµes e vetores. A programaÃ§Ã£o funcional Ã© um

assunto bastante extenso, sendo mais conhecido no R pela famÃlia de funÃ§Ãµes `purrr::map()`, que permite substituir muitos *loops for* por um cÃ³digo mais sucinto e fÃ¡cil de ler. NÃ£o focaremos aqui nas outras funÃ§Ãµes, pois esse Ã© um assunto extremamente extenso.

Todas as funÃ§Ãµes deste pacote sÃ£o listadas na pÃ¡gina de referÃªncia do pacote.

Um *loop for* pode ser entendido como uma iteraÃ§Ã£o: um bloco de cÃ³digos Ã© repetido mudando um contador de uma lista de possibilidades. Vamos exemplificar com uma iteraÃ§Ã£o bem simples, onde imprimiremos no console os valores de 1 a 10, utilizando a funÃ§Ã£o `for()`, um contador `i` em um vetor de dez nÃºmeros `1:10` que serÃ¡ iterado, no bloco de cÃ³digos definido entre `{}`, usando a funÃ§Ã£o `print()` para imprimir os valores.

A ideia Ã© bastante simples: a funÃ§Ã£o `for()` vai atribuir o primeiro valor da lista ao contador `i`, esse contador serÃ¡ utilizado em todo o bloco de cÃ³digos. Quando o bloco terminar, o segundo valor Ã© atribuÃdo ao contador `i` e entra no bloco de cÃ³digos, repetindo esse processo atÃ© que todos os elementos da lista tenham sido atribuÃdos ao contador.

```
## Loop for
for(i in 1:10){
    print(i)
}
```

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

Com essa ideia em mente, a programaÃ§Ã£o funcional faz a mesma operaÃ§Ã£o utilizando a funÃ§Ã£o `purrr::map()`. O mesmo *loop for* ficaria dessa forma.

```
## Loop for com map
purrr::map(.x = 1:10, .f = print)
```

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

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

```
##
## [[4]]
## [1] 4
##
## [[5]]
## [1] 5
##
## [[6]]
## [1] 6
##
## [[7]]
## [1] 7
##
## [[8]]
## [1] 8
##
## [[9]]
## [1] 9
##
## [[10]]
## [1] 10
```

Nessa estrutura, temos:

`map(.x, .f)`

- `.x`: um vetor, lista ou data frame
- `.f`: uma funÃ§Ã£o

Num outro exemplo, aplicaremos a funÃ§Ã£o `sum()` para somar os valores de vÃ¡rios elementos de uma lista.

```r
## FunÃ§Ã£o map
x <- list(1:5, c(4, 5, 7), c(1, 1, 1), c(2, 2, 2, 2, 2))
purrr::map(x, sum)
```

```
## [[1]]
## [1] 15
##
## [[2]]
## [1] 16
##
## [[3]]
## [1] 3
##
## [[4]]
## [1] 10
```

HÃ¡ diferente tipos de retornos da famÃlia `purrr::map()`.

- `map()`: retorna uma lista
- `map_chr()`: retorna um vetor de strings
- `map_dbl()`: retorna um vetor numÃ©rico (double)
- `map_int()`: retorna um vetor numÃ©rico (integer)
- `map_lgl()`: retorna um vetor lÃ³gico
- `map_dfr()`: retorna um data frame (por linhas)
- `map_dfc()`: retorna um data frame (por colunas)

```r
## VariaÃ§Ãµes da funÃ§Ã£o map
purrr::map_dbl(x, sum)
```

```
## [1] 15 16  3 10
```

```r
purrr::map_chr(x, paste, collapse = " ")
```

```
## [1] "1 2 3 4 5" "4 5 7"     "1 1 1"     "2 2 2 2 2"
```

Essas funcionalidades jÃ¡ eram conhecidas no *R Base* pelas funÃ§Ãµes da *famÃlia* `apply()`: `apply()`, `lapply()`, `sapply()`, `vapply()`, `mapply()`, `rapply()` e `tapply()`. Essas funÃ§Ãµes formam a base de combinaÃ§Ãµes mais complexas e ajudam a realizar operaÃ§Ãµes com poucas linhas de cÃ³digo, para diferentes retornos.

Temos ainda duas variantes da funÃ§Ã£o `map()`: `purrr::map2()` e `purrr::pmap()`, para duas ou mais listas, respectivamente. Como vimos para a primeira funÃ§Ã£o, existem vÃ¡rias variaÃ§Ãµes do sufixo para modificar o retorno da funÃ§Ã£o.

```r
## Listas
x <- list(3, 5, 0, 1)
y <- list(3, 5, 0, 1)
z <- list(3, 5, 0, 1)

## FunÃ§Ã£o map2
purrr::map2_dbl(x, y, prod)
```

```
## [1]  9 25  0  1
```

```r
## FunÃ§Ã£o pmap
purrr::pmap_dbl(list(x, y, z), prod)
```

```
## [1]  27 125   0   1
```

Essas funÃ§Ãµes podem ser usadas em conjunto para implementar rotinas de manipulaÃ§Ã£o e anÃ¡lise de dados com poucas linhas de cÃ³digo, mas que nÃ£o exploraremos em sua completude aqui. Listamos dois exemplos simples.

```r
## Resumo dos dados
penguins |>
    dplyr::select(where(is.numeric)) |>
    tidyr::drop_na() |>
    purrr::map_dbl(mean)
```

```
##    bill_length_mm     bill_depth_mm flipper_length_mm       body_mass_g
##          43.92193          17.15117         200.91520        4201.75439
##              year
##        2008.02924
```

```r
## AnÃ¡lise dos dados
penguins |>
    dplyr::group_split(island, species) |>
    purrr::map(~ lm(bill_depth_mm ~ bill_length_mm, data = .x)) |>
    purrr::map(summary) |>
    purrr::map("r.squared")
```

```
## [[1]]
## [1] 0.2192052
##
## [[2]]
```

```
## [1] 0.4139429
##
## [[3]]
## [1] 0.2579242
##
## [[4]]
## [1] 0.4271096
##
## [[5]]
## [1] 0.06198376
```

Para se aprofundar no tema, recomendamos a leitura do CapÃtulo 21 Iteration de Wickham & Grolemund [-@wickham2017].

8.12 Para se aprofundar

Listamos a seguir livros que recomendamos para seguir com sua aprendizagem em R e tidyverse.

8.12.1 Livros

Recomendamos aos (Ã s) interessados(as) os livros: i) Oliveira e colaboradores [-@oliveira_ciencia_2018] CiÃªncia de dados com R, ii) Grolemund [-@williams_essentials_2018] The Essentials of Data Science: Knowledge Discovery Using R,iii) Holmes e Huber [-@holmes_modern_2019] Modern Statistics for Modern Biology, iv) Ismay e Kim [-@ismay_statistical_2020] Statistical Inference via Data Science: A ModernDive into R and the Tidyverse, v) Wickham e Grolemund [-@wickham2017] R for Data Science: Import, Tidy, Transform, Visualize, and Model Data, vi) Zumel e Mount [-@zumel_practical_2014] Practical Data Science with R Paperback, vii) Irizarry [-@irizarry_data_2017] Introduction to Data Science: Data Analysis and Prediction Algorithms with R, e viii) Irizarry [-@irizarry_introduction_2019] Introduction to Data Science.

8.12.2 Links

CiÃªncia de Dados em R

tidyverse

STHDA - Statistical tools for high-throughput data analysis

EstatÃstica Ã© com R! - tidyverse

Tidyverse Skills for Data Science in R

Getting Started with the Tidyverse

Tidyverse Basics

Manipulando Dados com dplyr e tidyr

8.13 ExercÃcios

5.1 Reescreva as operaÃ§Ãµes abaixo utilizando pipes `|>`.

- `log10(cumsum(1:100))`
- `sum(sqrt(abs(rnorm(100))))`
- `sum(sort(sample(1:10, 10000, rep = TRUE)))`

5.2 Use a funÃ§Ã£o `download.file()` e `unzip()` para baixar e extrair o arquivo do data paper de mÃ©dios e grandes mamÃferos: ATLANTIC MAMMALS. Em seguida, importe para o R, usando a funÃ§Ã£o `readr::read_csv()`.

5.3 Use a funÃ§Ã£o `tibble::glimpse()` para ter uma noÃ§Ã£o geral dos dados importados no item anterior.

5.4 Compare os dados de penguins (*palmerpenguins::penguins_raw* e *palmerpenguins::penguins*). Monte uma sÃ©rie de funÃ§Ãµes dos pacotes *tidyr* e `dplyr` para limpar os dados e fazer com que o primeiro dado seja igual ao segundo.

5.5 Usando os dados de penguins (*palmerpenguins::penguins*), calcule a correlaÃ§Ã£o de Pearson entre comprimento e profundidade do bico para cada espÃ©cie e para todas as espÃ©cies. Compare os Ãndices de correlaÃ§Ã£o para exemplificar o Paradoxo de Simpson.

5.6 Oficialmente a pandemia de COVID-19 comeÃ§ou no Brasil com o primeiro caso no dia 26 de fevereiro de 2020. Calcule quantos anos, meses e dias se passou desde entÃ£o. Calcule tambÃ©m quanto tempo se passou atÃ© vocÃª ser vacinado.

SoluÃ§Ãµes dos exercÃcios.

9 Primeiros passos com uma raster

Organize following: https://jakubnowosad.com/posts/2020-05-26-intro-to-landscape-ecology/ and https://github.com/r-spatialecology/ialena-2020

Focus on methods used to make the data and analysis in the chapters. Chapter on visualizations? https://www.wvview.org/os_sa/10_Maps_with_tmap.html

https://youtu.be/dCTOGDnDuvM at 28 minutes Attribute data operations - Vector attribute subsetting, aggrergatyion and joins - Creating new vector attributes - Raster subsetting - Summarizing raster objects

Spatial data operations - Spatial subsetting - Topological relations - Spatial joining - - zonal raster operations

Geometry operations - on vector - on raster - Vector-raster interactions

Coordinate reference systems - Understaning map projections - Reprojecting spatial data - Modifying map projections

Uma raster Ã© um matriz de valores com coordenadas geogrÃ¡ficos. Cada pixel de uma raster representa uma regiÃ£o geogrÃ¡fica, e o valor do pixel representa uma caracterÃstica dessa regiÃ£o (mais sobre dados raster).

Em geral Ã© necessÃ¡rio baixar alguns pacotes para que possamos fazer as nossas anÃ¡lises. Precisamos os seguintes pacotes, que deve esta instalado antes:

- tidyverse,
- terra,
- sf,
- mapview,
- tmap.

Carregar pacotes:

```
library(tidyverse)
library(terra)
library(sf)
library(mapview)
library(tmap)
```

Inicialmente iremos gerar uma raster representando uma paisagem bem simples, de 6 por 6 pixels. VocÃª jÃ¡ deve saber que pixel Ã© a unidade bÃ¡sica de uma imagem (lembra da camera do seu celular, 10Mb ou algo assim?!). VocÃªs devem ter visto sobre pixels e resoluÃ§Ã£o no mesmo em aulas de geoprocessamento. Aqui podemos tratar o pixel como a resoluÃ§Ã£o. Vamos dizer que temos um pixel de 10 metros (res=10 no bloco de cÃ³digo), ou seja, uma quadrado de 10 por 10m, sendo essa, a menor unidade mapeÃ¡vel. Assim sendo, a resoluÃ§Ã£o tambem tem ligaÃ§Ã£o com escala cartogrÃ¡fica!

Vamos gerar e plotar uma paisagem simples em 4 passos. Primeiramente, a funÃ§Ã£o `rast` cria um objeto do tipo raster. E depois a funÃ§Ã£o `values` atribui valores, e na sequencia vamos visualisar os valores com `plot` e `text` .

```
#FunÃ§Ã£o "rast" gera a paisagem virtual (paisagem simulado)
pai_sim <- rast(ncols=6, nrows=6,
                xmin=1, xmax=60,
                ymin=1, ymax=60,
                res=10)
#E essa atribui valores ("values") para os pixels criados acima
values(pai_sim) <- 1
plot(pai_sim) #Essa plota
text(pai_sim) #Essa coloca os valores dos pixels
```

Figura 9.1: Paisagem simples de 36 pixels.

9.0.1 Obter e carregar dados (raster)

Mais uma vez vamos aproveitar os dados de MapBiomas. Agora baixar arquivo raster com cobertura de terra no entorno dos rios em 2020, (formato ".tif", tamanho 3.3 MB). Link: https://github.com/darrennorris/gisdata/blob/master/inst/raster/mapbiomas_AP_utm_rio/utm_cover_AP_rio_2020.tif . Lembrando-se de salvar o arquivo ("utm_cover_AP_rio_2020.tif") em um local conhecido no seu computador. Agora avisar R sobre onde ficar o arquivo. O cÃ³digo abaixo vai abrir uma nova janela, e vocÃª deve buscar e selecionar o arquivo "utm_cover_AP_rio_2020.tif":

```
meuSIGr <- file.choose()
```

O cÃ³digo abaixo vai carregar os dados e criar o objeto "mapbiomas_2020".

```
mapbiomas_2020 <- rast(meuSIGr)
```

9.0.2 ReclassificaÃ§Ã£o

Para simplificar nossa avaliaÃ§Ã£o de escala, reclassificamos a camada mapbiomas_2020 em uma camada binÃ¡ria de floresta/nÃ£o-floresta. Essa tarefa de geoprocessamento pode ser realizada anteriormente usando SIG (QGIS). Aqui vamos reclassificar as categorias de cobertura da terra (agrupando diferentes Ã¡reas de cobertura florestal tipos) usando alguns comandos genÃ©ricos do R para criar uma nova camada com a cobertura de floresta em toda a regiÃ£o de estudo. Para isso, criamos um mapa do mesmo resoluÃ§Ã£o e extensÃ£o, e entÃ£o podemos redefinir os valores do mapa. Neste caso, queremos agrupar a cobertura da terra categorias 3 e 4 (FormaÃ§Ã£o Florestal e FormaÃ§Ã£o SavÃ¢nica, respectivamente).

```
# criar uma nova camada de floresta
floresta_2020 <- mapbiomas_2020
# Com valor de 0
values(floresta_2020) <- 0
# Atualizar categorias florestais agrupados com valor de 1
floresta_2020[mapbiomas_2020==3 | mapbiomas_2020==4] <- 1
```

Vizualizar para verificar.

```
# Passo necessario para agilizar o processamento
floresta_2020_modal<-aggregate(floresta_2020, fact=10, fun="modal")
# Plot
tm_shape(floresta_2020_modal) +
  tm_raster(style = "cat",
            palette = c("0" = "#E974ED", "1" ="#129912"),
            legend.show = FALSE) +
  tm_add_legend(type = "fill", labels = c("nÃ£o-floresta", "floresta"),
    col = c("#E974ED", "#129912"), title = "Classe") +
tm_scale_bar(breaks = c(0, 25, 50), text.size = 1,
             text.color = "white", position=c("left", "bottom")) +
tm_layout(legend.position = c("right","top"),legend.bg.color = "white")
```

Figura 9.2: Floresta ao redor do Rio Araguari. MapBiomas 2020 reclassificado em floresta e nÃ£o-floresta. Mostrando os pontos de amostragem (pontos amarelas) cada 5 quilÃ´metros ao longo do rio.

10 Primeiros passos com vector

Em geral Ã© necessÃ¡rio baixar alguns pacotes para que possamos fazer as nossas anÃ¡lises. Precisamos os seguintes pacotes, que deve esta instalado antes:

- tidyverse,
- sf,
- mapview,
- tmap.

Carregar pacotes:

```
library(tidyverse)
library(sf)
library(mapview)
library(tmap)
```

10.1 Obter e carregar dados (vectores)

Precisamos carregar os dados para rios e pontos de amostragem. Baixar arquivo (vector) com os dados (formato "GPKG", tamanho 54.9 MB). Mais sobre dados vetoriais. O formato aberto GeoPackage Ã© um contÃªiner que permite armazenar dados SIG (feiÃ§Ãµes/camadas) em um Ãºnico arquivo. Por exemplo, um arquivo GeoPackage pode conter vÃ¡rios dados (dados vetoriais e raster) em diferentes sistemas de coordenadas. Todos esses recursos permitem que vocÃª compartilhe dados facilmente e evite a duplicaÃ§Ã£o de arquivos.

Baixar o arquivo Link: https://github.com/darrennorris/gisdata/blob/master/inst/vector/rivers.gpkg . Lembrando-se de salvar o arquivo ("rivers.gpkg") em um local conhecido no seu computador.

O formato "GPKG" e diferente de "tif" (raster), o processo de importaÃ§Ã£o Ã©, portanto, diferente. Primeira, avisar R sobre onde ficar o arquivo. O cÃ³digo abaixo vai abrir uma nova janela, e vocÃª deve buscar e selecionar o arquivo "rivers.GPKG":

```
meuSIG <- file.choose()
```

Agora vamos olhar o que tem no arquivo. Depois que voces rodar o codigo **st_layers(meuSIG)**, o resultado mostra que o arquivo rivers.GPKG inclui camadas diferentes com pontos ("Point"), linhas ("Line String") e polÃgonos ("Polygon"). Alem disso, a coluna "crs_name" mostrar que a sistema de coordenadas Ã© geogrÃ¡fica (WGS84, (EPSG: 4326)

$$https://epsg.io/4326$$

, e Ã© diferente do arquivo raster:

```
sf::st_layers(meuSIG)
```

```
## Driver: GPKG
## Available layers:
##                    layer_name geometry_type features fields crs_name
## 1                  centerline   Line String       52     15   WGS 84
## 2                  forestloss         Point   276086     12   WGS 84
## 3                   canalpoly       Polygon        3      6   WGS 84
## 4              extentpoly50km       Polygon        1      0   WGS 84
## 5                   midpoints         Point       52     17   WGS 84
## 6            midpoints_hansen         Point       52     37   WGS 84
## 7         cachoeira_caldeirao         Point        1      2   WGS 84
## 8                porto_grande         Point        1      1   WGS 84
## 9                 icmbio_base         Point        1      1   WGS 84
## 10              direct_affect       Polygon        1      2   WGS 84
```

```
## 11 midpoints_hansen_distances         Point       52     43   WGS 84
## 12       midpoints_hansen_ffr         Point       52     82   WGS 84
## 13     midpoints_hansen_ffril         Point       52     91   WGS 84
## 14         direct_affect_line   Line String        1      2   WGS 84
```

NÃ³s sÃ³ precisamos de duas dessas camadas. O cÃ³digo abaixo vai carregar as camadas que precisamos e criar os objetos "rsm" e "rsl". Assim, agora temos dados com: pontos cada 5 km ao longo os rios ("rsm") e a linha central de rios ("rsl").

```r
# pontos cada 5 km
rsm <- sf::st_read(meuSIG, layer = "midpoints")
```

```
## Reading layer `midpoints' from data source
##   `C:\Users\user\Documents\Articles\gis_layers\gisdata\inst\vector\rivers.gpkg'
##   using driver `GPKG'
## Simple feature collection with 52 features and 17 fields
## Geometry type: POINT
## Dimension:     XY
## Bounding box:  xmin: -52.01259 ymin: 0.7175827 xmax: -51.29688 ymax: 1.330365
## Geodetic CRS:  WGS 84
```

```r
# linha central de rios
rsl <- sf::st_read(meuSIG, layer = "centerline")
```

```
## Reading layer `centerline' from data source
##   `C:\Users\user\Documents\Articles\gis_layers\gisdata\inst\vector\rivers.gpkg'
##   using driver `GPKG'
## Simple feature collection with 52 features and 15 fields
## Geometry type: LINESTRING
## Dimension:     XY
## Bounding box:  xmin: -52.01443 ymin: 0.7094595 xmax: -51.2924 ymax: 1.352094
## Geodetic CRS:  WGS 84
```

10.2 Visualizar os arquivos (camadas vector)

Visualizar para verificar. Mapa com linha central e pontos de rios em trechos de 5km.

```
ggplot(rsl) +
  geom_sf(aes(color=rio)) +
  geom_sf(data = rsm, shape=21, aes(fill=zone))
```

Figura 10.1: Pontos ao longo dos rios a montante das hidrelÃ©tricas no Rio Araguari.

Mapa interativo (funcione somente com internet). Mostrando agora com fundo de mapas "base" (OpenStreetMap/ESRI etc)

```
#
mapview(rsl, zcol = "rio")
```

Em andamento ……..

Figura 10.2: Linhas dos rios a montante das hidrelÃ©tricas no Rio Araguari.

Part V

Encerramento

Bibliografia

GlossÃ¡rio

Conteudo copiado, com pequenas alteraÃ§Ãµes e atualizaÃ§Ãµes de GlossÃ¡rio, do livro AnÃ¡lises EcolÃ³gicas no R .

- **Abundance-based Coverage Estimator (ACE)**: estimador de riqueza de espÃ©cies baseado na abundÃncia de espÃ©cies raras

- **AgregaÃ§Ã£o (raster)**: aumentar o tamanho dos pixels (diminuindo a resoluÃ§Ã£o) de um raster, agregando os valores dos pixels em um pixel maior
- **Agrupamento filogenÃ©tico**: espÃ©cies coexistindo nas comunidades sÃ£o mais aparentadas do que esperado pelo acaso
- **Alinhamento (raster)**: ajusta o tamanho do pixel, extensÃ£o, nÃºmero e origem dos pixels para vÃ¡rias camadas rasters
- **Ambiente ou *Environment* (RStudio)**: porÃ§Ã£o onde os objetos criados sÃ£o armazenados
- **Amostra**: subconjunto da populaÃ§Ã£o, selecionados por um processo adequado de amostragem, utilizado para estimar caracterÃsticas de toda a populaÃ§Ã£o
- **AnÃ¡lise de covariÃ¢ncia (ANCOVA)**: Ã© uma extensÃ£o da ANOVA com a adiÃ§Ã£o de uma covariÃ¡vel medida em todas as unidades amostrais
- **AnÃ¡lise de variÃ¢ncia (ANOVA)**: Ã© um teste estatÃstico que avalia se hÃ¡ diferenÃ§as entre as mÃ©dias de trÃªs ou mais grupos independentes. Existem uma variedade de delineamentos experimentais como - ANOVA de um fator, ANOVA de dois fatores, ANOVA em blocos aleatorizados, ANOVA de medidas repetidas e ANOVA *split-splot* - que diferem na maneira que o teste estatÃstico e os graus de liberdade sÃ£o calculados
- **AnÃ¡lise paramÃ©trica**: assume que os dados foram amostrados de uma distribuiÃ§Ã£o de forma conhecida (e.g., gaussiana, poisson, etc) e estima os parÃ¢metros (e.g., mÃ©dia, desvio padrÃ£o, etc) da distribuiÃ§Ã£o a partir dos dados
- **AnÃ¡lise nÃ£o paramÃ©trica**: nÃ£o pressupÃµe que os dados foram amostrados de uma distribuiÃ§Ã£o de forma conhecida (e.g., gaussiana, poisson, etc)
- **Array**: classe de objetos que representa elementos de um Ãºnico modo no formato de combinaÃ§Ã£o de tabelas, com linhas, colunas e dimensÃµes
- **Ã rvore filogenÃ©tica**: sÃ£o hipÃ³teses que representam a relaÃ§Ã£o de parentesco entre as espÃ©cies (pode ser tambÃ©m indivÃduos, genes, etc.) com informaÃ§Ãµes sobre quais espÃ©cies compartilham um ancestral comum e a distÃ¢ncia (tempo, genÃ©tica, ou diferenÃ§as nos caracteres) que as separam
- **Atributo funcional**: uma propriedade mensurÃ¡vel dos organismos (geralmente em nÃvel individual) que representa caracterÃsticas morfolÃ³gicas, fisiolÃ³gicas ou fenolÃ³gicas que afetam a aptidÃ£o alterando aspectos do crescimento, reproduÃ§Ã£o e sobrevivÃªncia
- **Atributos dos objetos**: sÃ£o o modo (natureza) e a estrutura (organizaÃ§Ã£o) dos elementos nos objetos
- **Autovalor(Eigenvalue)**: nÃºmero inteiro (escalar) que multiplica um vetor, sendo portanto mÃºltiplo deste. Esses valores representam a variÃ¢ncia dos eixos e, se convertidos em valores relativos, medem a porcentagem de variÃ¢ncia contida em cada eixo.

- **Autovetor (Eigenvector)**: vetor nÃ£o nulo que muda somente quando Ã© multiplicado por um escalar. O autovetor de uma matriz Ã© encontrado pelo resultado da multiplicaÃ§Ã£o de um vetor pela matriz, que Ã© igual a *lambda* vezes o vetor. Esse vetor do resultado passa a ser o autovetor, e o *lambda* o seu respectivo autovalor
- **Bloco (ANOVA)**: Ã© uma Ã¡rea ou perÃodo de tempo dentro do qual as condiÃ§Ãµes ambientais sÃ£o relativamente homogÃªneas. O objetivo do uso dos blocos Ã© controlar fontes de variaÃ§Ãµes indesejadas na variÃ¡vel dependente que nÃ£o sÃ£o de interesse do pesquisador
- **Bootstrap**: estimador de riqueza de espÃ©cies que estima os parÃ¢metros de uma populaÃ§Ã£o por reamostragens
- **Buffer (vetor)**: polÃgono que representa a Ã¡rea dentro de uma determinada distÃ¢ncia de um dado

vetorial, podendo ser a partir de um ponto, linha ou polÃgono

- **Camada (vetor ou raster)**: termo geral, mas geralmente associada Ã diferentes rasters reunidos num mesmo arquivo
- **Centroide (vetor ou raster)**: ponto central de uma feiÃ§Ã£o vetorial ou o centro do pixel do raster
- **CentrÃ³ide (multivariado)**: mÃ©dia ponderada de um conjunto multivariado, a menor distÃ¢ncia mÃ©dia de todos os objetos num espaÃ§o multivariado
- **Chao 1**: estimador de riqueza de espÃ©cies baseado na abundÃ¢ncia das espÃ©cies *singleton* e *doubletons* dentro de uma amostra
- **Chao 2**: estimador de riqueza de espÃ©cies baseado na incidÃªncia (presenÃ§a e ausÃªncia) das espÃ©cies *singleton* e *doubletons* dentro de uma amostra
- **Clado**: um grupo de espÃ©cies aparentadas descendendo de um Ãºnico nÃ³ na filogenia
- **Coeficiente de correlaÃ§Ã£o**: indica a forÃ§a da relaÃ§Ã£o linear entre as duas variÃ¡veis
- **CoerÃ§Ã£o (linguagem R)**: transformaÃ§Ã£o dos modos dos elementos seguindo uma hierarquia: character > double > integer > logical
- **CombinaÃ§Ã£o linear**: CombinaÃ§Ã£o de vÃ¡rias variÃ¡veis (vetores) que sÃ£o multiplicadas por constantes e adicionadas a outras variÃ¡veis. Por exemplo: combinaÃ§Ã£o linear de x, y, z pode ser escrita como ax+by+cz.
- **Community Mean Nearest Taxon Distance (COMDISTNT)**: mÃ©trica de diversidade beta que calcula a mÃ©dia da distÃ¢ncia filogenÃ©tica/funcional entre o tÃ¡xon mais prÃ³ximo das espÃ©cies de duas comunidades
- **Community Mean Pairwise Distance (COMDIST)**: mÃ©trica de diversidade beta que calcula a mÃ©dia da distÃ¢ncia filogenÃ©tica/funcional entre as espÃ©cies de duas comunidades
- **Console (RStudio)**: onde a versÃ£o da linguagem R instalada Ã© carregada para executar os cÃ³digos no RStudio
- **ConversÃ£o (linguagem R)**: transformaÃ§Ã£o dos modos ou estrutura dos elementos de um objeto a partir de funÃ§Ãµes especÃficas
- **ConversÃµes raster-vetor**: transformar dados vetoriais em rasters (rasterizaÃ§Ã£o) e transformar rasters em vetores (vetorizaÃ§Ã£o)
- **Convex Hull**: medida multivariada derivada da computaÃ§Ã£o geomÃ©trica que calcula o espaÃ§o dos atributos de uma espÃ©cie ou de vÃ¡rias espÃ©cies em uma comunidade
- **CorrelaÃ§Ã£o**: Ã© um teste que mede a forÃ§a relativa da relaÃ§Ã£o linear entre duas variÃ¡veis contÃnuas. A anÃ¡lise de correlaÃ§Ã£o nÃ£o assume que a variÃ¡vel X influencie a variÃ¡vel Y, ou que exista uma relaÃ§Ã£o de causa e efeito entre elas
- **Cortes e mÃ¡scaras (raster)**: ajusta o tamanho de um dado raster a uma Ã¡rea menor de interesse, geralmente definido por um dado vetorial
- **CovariÃ¡vel**: variÃ¡vel contÃnua que potencialmente afeta a variÃ¡vel resposta, mas nÃ£o Ã© necessariamente controlada ou manipulada pelo pesquisador
- **CritÃ©rio de InformaÃ§Ã£o de Akaike (*Akaike Information Criterion* - AIC)**: Ã© uma mÃ©trica para seleÃ§Ã£o de modelos. Ele Ã© usado para determinar qual entre os mÃºltiplos modelos Ã© o mais provÃ¡vel de prever os dados observados, ponderando pelo nÃºmero de parÃ¢metros dos modelos. Os menores valores de AIC representam modelos com melhores ajustes aos dados
- **Curva de dominÃ¢ncia**: veja Diagrama de Whittaker
- **Dados geoespaciais**: sÃ£o dados georreferenciados que expressÃ£o informaÃ§Ãµes espaciais, podendo ser no formato vetorial ou matricial (raster)
- **Dados matriciais (raster)**: consistem em uma matriz (com linhas e colunas) em que os elementos representam cÃ©lulas, geralmente igualmente espaÃ§adas (pixels)
- **Dados tidy**: dados organizados, com foco na limpeza e organizaÃ§Ã£o dos mesmos, de modo que os dados estÃ£o *tidy* quando: i) variÃ¡veis estÃ£o nas colunas, ii) observaÃ§Ãµes estÃ£o nas linhas e iii) valores estÃ£o nas cÃ©lulas, sendo que para esse Ãºltimo, nÃ£o deve haver mais de um valor por cÃ©lula
- **Dados vetoriais**: sÃ£o representaÃ§Ãµes geomÃ©tricas (pontos, linhas e polÃgonos) usadas para mapear fenÃ´menos ou objetos espacialmente explÃcitos que possuem localizaÃ§Ã£o ou dimensÃµes bem definidas, aos quais sÃ£o atribuÃdas informaÃ§Ãµes tabulares
- **Data frame**: classe de objetos que representa dados no formato de tabela, com linhas e colunas, mas

comportam mais de um modo em suas colunas

- **Data Science**: nova Ã¡rea de conhecimento que vem se moldando a partir do desenvolvimento da sociedade em torno da era digital e da grande quantidade de dados gerados e disponÃveis pela internet
- **Datum**: simplificadamente Ã© a relaÃ§Ã£o do sistema de coordenadas (geogrÃ¡fica ou projetada) com a superfÃcie da Terra
- **Dendrograma**: diagrama representando uma Ã¡rvore que organiza elementos, objetos e variÃ¡veis por suas semelhanÃ§as de maneira hierÃ¡rquica ascendente. Desse modo, objetos mais prÃ³ximos compartilham maior semelhanÃ§a do que objetos distantes no dendrograma.
- **DesagregaÃ§Ã£o (raster)**: diminui o tamanho dos pixels (aumentando a resoluÃ§Ã£o) de um raster, preenchendo com novos valores que depende do tipo de funÃ§Ã£o de preenchimento utilizada
- **Deviance**: Ã© um termo estatÃstico que mede o ajuste do modelo (*goodness of fit*). Quanto menor o valor de *deviance* melhor o modelo
- **Diagrama de Whittaker**: mÃ©todo que utiliza informaÃ§Ãµes visuais ao plotar as espÃ©cies ranqueadas no eixo X da mais abundante para a menos abundante, enquanto no eixo Y as abundÃ¢ncias relativas das espÃ©cies sÃ£o plotadas em escala logarÃtmica
- **DiretÃ³rio de trabalho**: endereÃ§o da pasta (ou diretÃ³rio) de onde o R importarÃ¡ ou exportar dados
- **DispersÃ£o filogenÃ©tica**: espÃ©cies coexistindo nas comunidades sÃ£o menos aparentadas do que esperado pelo acaso
- **DistribuiÃ§Ã£o de probabilidade**: Ã© uma funÃ§Ã£o estatÃstica que descreve todos os valores e probabilidades possÃveis que uma variÃ¡vel aleatÃ³ria pode assumir dentro de um determinado intervalo
- **DistribuiÃ§Ã£o Gaussiana**: veja distribuiÃ§Ã£o normal
- **DistribuiÃ§Ã£o normal**: Ã© uma distribuiÃ§Ã£o de probabilidade em formato de sino que Ã© simÃ©trica em torno da mÃ©dia, mostrando que dados prÃ³ximos Ã mÃ©dia sÃ£o mais frequentes que dados distantes da mÃ©dia.
- **Diversidade alfa**: Ã© um conceito caracterizado pela diversidade dentro do habitat ou unidade amostral
- **Diversidade beta**: Ã© um conceito caracterizado pela variaÃ§Ã£o na diversidade entre habitats ou unidades amostrais
- **Diversidade beta filogenÃ©tica**: engloba mÃ©tricas que utilizam dados de presenÃ§a e ausÃªncia ou abundÃ¢ncia das espÃ©cies para determinar um valor que representa a diferenÃ§a entre comunidades em relaÃ§Ã£o a histÃ³ria evolutiva das linhagens
- **Diversidade de espÃ©cies**: Ã© um conceito que representa o nÃºmero de espÃ©cies e a distribuiÃ§Ã£o de abundÃ¢ncia destas espÃ©cies em uma comunidade
- **Diversidade filogenÃ©tica**: engloba mÃ©tricas que capturam a ancestralidade compartilhada entre as espÃ©cies em termos de quantidade da histÃ³ria evolutiva e o grau de parentesco entre as espÃ©cies
- **Diversidade funcional**: Ã© um conceito que captura a variaÃ§Ã£o no grau de expressÃ£o de diferentes atributos funcionais entre diferentes populaÃ§Ãµes, comunidades ou ecossistemas
- **Diversidade gama**: Ã© um conceito caracterizado pela combinaÃ§Ã£o da diversidade alfa e beta ou definido como a diversidade regional englobando todos os habitat ou unidades amostrais
- **Doubletons**: nÃºmero de espÃ©cies observadas com abundÃ¢ncia de dois indivÃduos
- **Duplicate**: nÃºmero de espÃ©cies observadas em apenas duas amostras
- **Equitabilidade de Pielou**: Ã© uma mÃ©trica que descreve o padrÃ£o de distribuiÃ§Ã£o da abundÃ¢ncia relativa das espÃ©cies na comunidade
- **Erro do Tipo I**: rejeitar a hipÃ³tese nula de um teste estatÃstico quando ela Ã© verdadeira
- **Erro do Tipo II**: aceitar a hipÃ³tese nula de um teste estatÃstico quando ela Ã© falsa
- **Escalar**: nÃºmero inteiro com o qual geralmente se faz operaÃ§Ãµes com matrizes (e.g., multiplicaÃ§Ã£o ou adiÃ§Ã£o)
- **Escores**: posiÃ§Ã£o das unidades amostrais ao longo de um eixo de ordenaÃ§Ã£o. Pode se referir tanto a objetos quanto Ã variÃ¡veis. Escores sÃ£o fornecidos pela substituiÃ§Ã£o dos valores assumidos pelas variÃ¡veis originais nas combinaÃ§Ãµes lineares. SÃ£o utilizados para ordenar as unidades amostrais em um diagrama uni, bi ou tridimensional
- **ESRI Shapefile (vetor)**: principal formato de dados vetoriais, composto por pelo menos de qua-

tro arquivos: .shp (feiÃ§Ã£o), .dbf (tabela de atributos), .shx (ligaÃ§Ã£o entre .shp e .dbf) e .prj (projeÃ§Ã£o)

- **EstatÃstica frequentista**: sÃ£o anÃ¡lises paramÃ©tricas que estimam probabilidades das frequÃªncias observadas dos eventos e usam essas probabilidades como base para inferÃªncias.
- **Estrutura dos objetos**: diz respeito Ã organizaÃ§Ã£o dos elementos, com relaÃ§Ã£o aos modos e dimensionalidade da disposiÃ§Ã£o desses elementos. De modo bem simples, os elementos podem ser estruturados em seis tipos: i) vetor, ii) fator, iii) matriz, iv) array, v) data frame e vi) listas
- **ExtensÃ£o (vetor e raster)**: limites geogrÃ¡ficos de dados geoespaciais (vetor e raster), composto por dois pares de coordenadas de longitude e de latitude
- **ExtraÃ§Ã£o (vetor e raster)**: identifica e retorna valores associados de pixels de um raster com base em um objeto vetorial (ponto, linha e polÃgono)
- **ExtrapolaÃ§Ã£o**: Ã© o processo de estimar, alÃ©m do intervalo de observaÃ§Ã£o original, o valor de uma variÃ¡vel com base em sua relaÃ§Ã£o com outra variÃ¡vel
- **Fator**: classe de objetos que representa o encadeamento de elementos de um Ãºnico modo (integer) numa sequÃªncia unidimensional, representando medidas de uma variÃ¡vel categÃ³rica, podendo ser nominal ou ordinal
- **Fator aleatÃ³rio**: pesquisador amostra aleatoriamente os nÃveis de um fator na populaÃ§Ã£o
- **Fator fixo**: pesquisador controla todos os nÃveis do fator sobre os quais as inferÃªncias devem ser feitas
- **Fator de inflaÃ§Ã£o da variÃ¢ncia (VIF)**: Ã© um teste que quantifica quanto do erro padrÃ£o dos coeficientes estimados estÃ£o inflados devido Ã multicolinearidade
- **FeiÃ§Ã£o (vetor)**: um elemento do dado vetorial associado Ã cada linha da tabela de atributos
- **FenÃ´meno**: Um evento, entidade ou relaÃ§Ã£o observÃ¡vel
- **Forrageadoras de lugar central**: forrageiro viajando de uma base inicial ("casa"/"abrigo) para um local distante, em vez de simplesmente passar por uma Ã¡rea ou viajar aleatoriamente.
- **FunÃ§Ãµes**: classe de objetos que possui cÃ³digos preparados para realizar uma tarefa especÃfica de modo simples, realizando operaÃ§Ãµes em argumentos
- **Generalized Least Squares (GLS)**: Ã© um teste estatÃstico utilizado para estimar coeficientes desconhecidos de um modelo de regressÃ£o linear quando a variÃ¡vel independente Ã© correlacionada com os resÃduos
- **GeoPackage (vetor e raster)**: banco de dados geoespacial que armazena em apenas um arquivo, dados no formato vetorial, raster e tambÃ©m dados nÃ£o-espaciais (e.g., tabelas)
- **GeoTIFF (raster)**: principal formato para dados raster, composto geralmente de um arquivo TIFF contendo metadados geoespaciais adicionais
- **GitHub**: Ã© um repositÃ³rio de hospedagem de cÃ³digo-fonte e arquivos com controle de versÃµes de projetos abertos
- **Goodness of fit**: refere-se a um teste estatÃstico que determina quÃ£o bem os dados da amostra se ajustam a uma distribuiÃ§Ã£o de uma populaÃ§Ã£o
- **Grau de liberdade**: Ã© o nÃºmero de observaÃ§Ãµes nos dados que sÃ£o livres para variar quando estimamos os valores dos parÃ¢metros populacionais desconhecidos
- **HipÃ³tese**: AfirmaÃ§Ã£o testÃ¡vel derivada ou representando vÃ¡rios componentes de uma teoria
- **HipÃ³tese alternativa (H1)**: Ã© um conceito estatÃstico que sugere que diferenÃ§as entre grupos ou fenÃ´menos medidos sÃ£o maiores do que o esperado devido a variaÃ§Ãµes aleatÃ³rias. A H1 Ã© o oposto da hipÃ³tese nula.
- **HipÃ³tese nula (H0)**: Ã© um conceito estatÃstico que sugere que diferenÃ§as entre grupos ou fenÃ´menos medidos nÃ£o sÃ£o maiores do que o esperado devido a variaÃ§Ãµes aleatÃ³rias. Presume-se que a hipÃ³tese nula Ã© verdadeira atÃ© que evidÃªncias indiquem o contrÃ¡rio.
- **Homocedasticidade**: Ã© um dos pressupostos dos testes paramÃ©tricos como ANOVA, Teste T e regressÃ£o linear simples que a variÃ¢ncia da variÃ¡vel dependente deve ser constante para os valores das variÃ¡veis preditoras. SinÃ´nimo de homogeneidade da variÃ¢ncia
- **Homogeneidade da variÃ¢ncia**: veja homocedasticidade
- **IndependÃªncia estatÃstica**: dois eventos sÃ£o independentes se a ocorrÃªncia de um evento nÃ£o influenciar a probabilidade que o outro evento irÃ¡ ou nÃ£o ocorrer
- **IndexaÃ§Ã£o (linguagem R)**: acessa elementos de objetos por sua posiÃ§Ã£o utilizando os oper-

adores `[]` e `[[]]` ou por seu nome com o operador `$` depois do nome do objeto

- **Ã ndice de diversidade**: mÃ©tricas que calculam a riqueza de espÃ©cies e a distribuiÃ§Ã£o de abundÃ¢ncia para cada espÃ©cie dentro das comunidades.
- **Ã ndice de Gini-Simpson**: mÃ©trica que quantifica a probabilidade de dois indivÃduos retirados ao acaso da comunidade pertencerem a espÃ©cies diferentes. Ã o inverso do Ãndice de Simpson
- **Ã ndice de Margalef**: mÃ©trica que calcula a riqueza de espÃ©cies ponderando a abundÃ¢ncia total dentro de cada comunidade
- **Ã ndice de Menhinick**: mÃ©trica que calcula a riqueza de espÃ©cies ponderando a abundÃ¢ncia total dentro de cada comunidade
- **Ã ndice de Shannon-Wiener**: mÃ©trica de diversidade de espÃ©cies que quantifica a incerteza associada em predizer a identidade de uma espÃ©cie dado o nÃºmero de espÃ©cies e a distribuiÃ§Ã£o de abundÃ¢ncia para cada espÃ©cie
- **Ã ndice de Simpson**: mÃ©trica que quantifica a probabilidade de dois indivÃduos retirados ao acaso da comunidade pertencerem Ã mesma espÃ©cie
- **Inf (Infinito)**: Ã© um nÃºmero muito grande ou um limite matemÃ¡tico
- **InteraÃ§Ã£o raster-vetor**: operaÃ§Ãµes derivadas da interaÃ§Ã£o entre raster-vetor, como ajuste do tamanho do raster ou extraÃ§Ã£o dos valores dos pixels para dados vetoriais (pontos, linhas e polÃgonos)
- **InterpolaÃ§Ã£o**: Ã© o processo de estimar, dentro um domÃnio de valores conhecidos, um valor desconhecido com base em sua relaÃ§Ã£o com outra variÃ¡vel
- **Jackknife 1**: estimador de riqueza de espÃ©cies baseado no nÃºmero de espÃ©cies que ocorrem em somente uma amostra (*uniques*)
- **Jackknife 2**: estimador de riqueza de espÃ©cies baseado no nÃºmero de espÃ©cies que ocorrem em somente uma amostra (*uniques*) e no nÃºmero de espÃ©cies ocorrem em exatamente duas amostras (*duplicates*)
- **JunÃ§Ã£o de tabelas**: combinaÃ§Ã£o de pares de conjunto de dados tabulares por uma ou mais colunas chaves
- **Likelihood-ratio test (LRT)**: Ã© um teste estatÃstico que mede o grau do ajuste (*goodness-of-fit*) entre dois modelos aninhados. Um modelo relativamente mais complexo Ã© comparado a um modelo mais simples para ver se ele se ajusta significativamente melhor a um determinado conjunto de dados. Ele testa se hÃ¡ necessidade de se incluir uma variÃ¡vel extra no modelo para explicar os dados
- **Linguagem R**: ambiente de software livre para computaÃ§Ã£o estatÃstica e criaÃ§Ã£o de grÃficos
- **Lista**: classe de objetos que Ã© um tipo especial de vetor que aceita objetos como elementos
- **Loading**: CorrelaÃ§Ãµes de Pearson entre cada variÃ¡vel original com cada eixo. Os valores serÃ£o maiores (positiva ou negativamente) para aquelas variÃ¡veis que forem mais importantes na formaÃ§Ã£o de um dado eixo. Desse modo, representam o peso de uma variÃ¡vel para a construÃ§Ã£o de um eixo e variam de -1 a 1.
- **Loop for**: funÃ§Ã£o em que um bloco de cÃ³digos Ã© repetido mudando um contador de uma lista de possibilidades
- **Mapa**: representaÃ§Ã£o bidimensional de elementos geoespaciais em uma proporÃ§Ã£o menor, podendo representar dados vetoriais ou raster, com diversos elementos visuais e textuais que facilitam a interpretaÃ§Ã£o dos elementos geoespaciais mapeados
- **Mapas animados**: mapas que incorporam animaÃ§Ãµes para expressar mudanÃ§as nos padrÃµes espaciais ou ao longo do tempo
- **Mapas estÃ¡ticos**: mapas simples e fixos para visualizaÃ§Ã£o de dados, sendo o tipo mais comum de saÃda visual
- **Mapas interativos**: mapas que incorporam a capacidade de deslocar e ampliar qualquer parte de um conjunto de dados geoespaciais sobreposto em um â mapa da webâ
- **Matriz**: classe de objetos que representa elementos de um Ãºnico modo no formato de tabela, com linhas e colunas
- **MÃ¡xima VerossimilhanÃ§a**: Ã© um mÃ©todo que determina valores para os parÃ¢metros de um modelo. Os valores dos parÃ¢metros sÃ£o encontrados de tal forma que maximizam a probabilidade de que o processo descrito pelo modelo produza os dados que foram realmente observados
- **Mean Pairwise Distance (MPD)**: Ã© uma mÃ©trica que utiliza a matriz de distÃ¢ncia filo-

genÃ©tica para quantificar a distÃ¢ncia mÃ©dia do parentesco entre pares de espÃ©cies em uma comunidade

- **Mean Nearest Taxon Distance (MNTD)**: Ã© uma mÃ©trica que utiliza a matriz de distÃ¢ncia filogenÃ©tica para quantificar a mÃ©dia dos valores mÃnimos de parentesco entre pares de espÃ©cies em uma comunidade. Ou seja, qual o valor mÃ©dio da distÃ¢ncia para o vizinho mais prÃ³ximo
- **Mecanismo**: InteraÃ§Ã£o direta de uma relaÃ§Ã£o causal que resulta em um fenÃ´meno
- **Modo dos objetos**: diz respeito Ã natureza dos elementos que compÃµem os dados e que foram atribuÃdos aos objetos. Os modos geralmente sÃ£o: numÃ©rico do tipo inteiro (integer), numÃ©rico do tipo flutuante (double), texto (character), lÃ³gico (logical) ou complexo (complex)
- **Modelo misto**: pelo menos um fator no experimento Ã© fixo, e pelo menos um fator Ã© aleatÃ³rio.
- **Modelo nulo**: Ã© um procedimento estatÃstico que usa aleatorizaÃ§Ãµes para gerar distribuiÃ§Ãµes de valores para uma determinada variÃ¡vel de interesse na ausÃªncia do processo causal em questÃ£o
- **Multicolinearidade**: Ã© um conceito estatÃstico onde variÃ¡veis independentes em um modelo sÃ£o correlacionadas
- **NA (Not Available)**: significa dado faltante ou indisponÃvel
- **NaN (Not a Number)**: representa indefiniÃ§Ãµes matemÃ¡ticas
- **Nearest Relative Index (NRI)**: mÃ©trica que calcula o tamanho do efeito padronizado para a mÃ©trica *Mean Pairwise Distance*.Valores positivos de NRI indicam agrupamento filogenÃ©tico enquanto valores negativos de NRI indicam dispersÃ£o filogenÃ©tica
- **Nearest Taxon Index (NTI)**: mÃ©trica que calcula o tamanho do efeito padronizado para a mÃ©trica *Mean Nearest Taxon Distance*.Valores positivos de NTI indicam agrupamento filogenÃ©tico enquanto valores negativos de NTI indicam dispersÃ£o filogenÃ©tica
- **NetCDF (raster)**: *Network Common Data Form* Ã© um conjunto de bibliotecas de software e formatos de dados independentes que suportam a criaÃ§Ã£o, acesso e compartilhamento de dados cientÃficos orientados a arrays
- **NÃ³**: o ponto onde uma linhagem dÃ¡ origem a duas ou mais linhagens descendentes
- **Normalidade dos resÃduos**: Ã© um dos pressupostos dos testes paramÃ©tricos como ANOVA, Teste T e regressÃ£o linear simples que dependem que os resÃduos do modelo apresentem distribuiÃ§Ã£o normal ou gaussiana.
- **NULL (Nulo)**: representa um objeto nulo, sendo Ãºtil para preenchimento em aplicaÃ§Ãµes de programaÃ§Ã£o
- **NÃºmeros de Hill**: Ã© uma mÃ©trica que transforma a riqueza e a distribuiÃ§Ã£o da abundÃ¢ncia das espÃ©cies em nÃºmeros efetivos de espÃ©cies ou diversidade verdadeira
- **NÃºmero efetivo de espÃ©cies**: Ã© o nÃºmero de espÃ©cies igualmente abundantes (i.e., todas as espÃ©cies com a mesma abundÃ¢ncia) necessÃ¡rias para produzir o valor observado para um determinado Ãndice
- **Objetos**: palavras Ã s quais sÃ£o atribuÃdos dados atravÃ©s da atribuiÃ§Ã£o. A criaÃ§Ã£o de objetos possibilita a manipulaÃ§Ã£o de dados ou armazenar os resultados de anÃ¡lises. Em outras linguagens de programaÃ§Ã£o sÃ£o denominados variÃ¡veis
- **OperaÃ§Ãµes geoespaciais**: sÃ£o operaÃ§Ãµes para acessar ou alterar as propriedades nÃ£o-espaciais, espaciais e geomÃ©tricas dos dados geoespaciais, divididas em: operaÃ§Ãµes de atributos, operaÃ§Ãµes espaciais e operaÃ§Ãµes geomÃ©tricas
- **OperaÃ§Ãµes geoespaciais de atributos**: modificaÃ§Ã£o de objetos geoespaciais baseado em informaÃ§Ãµes nÃ£o espaciais associadas a dados geoespaciais, como a tabela de atributos ou valores das cÃ©lulas e nome das camadas dos rasters
- **OperaÃ§Ãµes geoespaciais espaciais**: modificaÃ§Ãµes de objetos geoespaciais baseado em informaÃ§Ãµes espaciais, como localizaÃ§Ã£o e formato
- **OperaÃ§Ãµes geoespaciais geomÃ©tricas**: modificaÃ§Ãµes em objetos geoespaciais baseado na geometria do vetor ou do raster e na interaÃ§Ã£o e conversÃ£o entre vetor-raster
- **Operadores**: conjuntos de caracteres que realiza operaÃ§Ãµes, agrupados em cinco tipos principais: aritmÃ©ticos, relacionais, lÃ³gicos, atribuiÃ§Ã£o e diversos
- **Pacotes (linguagem R)**: conjuntos extras de funÃ§Ãµes para executar tarefas especÃficas
- **PadrÃ£o**: Eventos repetidos, entidades recorrentes ou relaÃ§Ãµes replicadas no tempo ou no espaÃ§o
- **Phylogenetic Diversity (PD)**: Ã© uma mÃ©trica definida pela soma do comprimento dos ramos

conectando todas as espÃ©cies na comunidade

- **Phylogenetic Endemism (PE)**: Ã© uma mÃ©trica que calcula a fraÃ§Ã£o dos ramos restritos a regiÃµes especÃficas
- **Phylogenetic index of beta diversity (Phylosor)**: mÃ©trica de similaridade que determina o comprimento total dos ramos da filogenia que Ã© compartilhado entre pares de comunidades
- **Phylogenetic Species Richness (PSR)**: Ã© uma mÃ©trica diretamente comparÃ¡vel ao nÃºmero de espÃ©cies na comunidade, mas inclui o parentesco filogenÃ©tico entre as espÃ©cies
- **Phylogenetic Species Variability (PSV)**: Ã© uma mÃ©trica que estima a quantidade relativa dos comprimentos dos ramos nÃ£o compartilhados entre as comunidades
- **Pipe (|> e %>%)**: operador implementado por uma funÃ§Ã£o que faz com o que o resultado de uma funÃ§Ã£o seja o primeiro argumento da funÃ§Ã£o seguinte, permitindo o encadeamento de vÃ¡rias funÃ§Ãµes eliminando a necessidade de criar objetos para armazenar resultados intermediÃ¡rios
- **Pivotagem de dados**: transporte de dados que estÃ£o em linhas para colunas e vice-versa, fazendo a referÃªncia cruzada ou rotacionando os dados. Partirmos de dados no formato longo (*long*, muitas linhas e poucas colunas) e criamos dados no formato largo (*wide*, poucas linhas e muitas colunas) e vice-versa
- **Pixel (raster)**: tambÃ©m chamado cÃ©lula, Ã© a unidade geoespacial do raster, representando o elemento da matriz de dados
- **PolÃgono convexo**: operaÃ§Ã£o que liga os pontos externos de um conjunto de pontos e cria um polÃgono a partir deles
- **PolÃgono de Voronoi**: polÃgonos irregulares sÃ£o criados a partir da proximidade dos pontos, de modo a estimar uma Ã¡rea de abrangÃªncia no entorno dos mesmos
- **Politomia**: trÃªs ou mais linhagens descendendo de um Ãºnico nÃ³
- **PopulaÃ§Ã£o**: Ã© um conjunto de indivÃduos ou elementos semelhantes que interessa para alguma pergunta ou hipÃ³tese
- **PopulaÃ§Ã£o amostral**: Ã© um conjunto de indivÃduos ou elementos semelhantes que estÃ£o de fato acessÃvel para serem amostrados
- **PrediÃ§Ã£o espacial**: utiliza os coeficientes de um modelo ajustado para gerar um raster de prediÃ§Ã£o para todos os pixels considerando os dados preditores de entrada do modelo, extrapolando a prediÃ§Ã£o da resposta
- **PrediÃ§Ã£o**: uma declaraÃ§Ã£o de expectativa deduzida da estrutura lÃ³gica ou derivada da estrutura causal de uma teoria
- **Pressuposto**: condiÃ§Ãµes necessÃ¡rias para sustentar uma hipÃ³tese ou construÃ§Ã£o da teoria
- **Principais elementos de um mapa**: um mapa possui diversos elementos que facilitam sua interpretaÃ§Ã£o, dentre eles: i) mapa principal, ii) mapa secundÃ¡rio iii) tÃtulo, iv) legenda (apresentando as informaÃ§Ãµes detalhadas das classes ou escala de valores, v) barra de escala, vi) indicador de orientaÃ§Ã£o (Norte), vii) gride de coordenadas e viii) descriÃ§Ã£o do CRS
- **Processo**: um subconjunto de fenÃ´menos em que os eventos seguem uns aos outros no tempo ou espaÃ§o, que podem ou nÃ£o serem causalmente conectados. Ã a causa, mecanismo ou limitaÃ§Ã£o explicando um padrÃ£o
- **ProgramaÃ§Ã£o Funcional**: organizaÃ§Ã£o do cÃ³digo como funÃ§Ãµes e variÃ¡veis que trabalham de forma unificada para a resoluÃ§Ã£o de um problema
- **Projeto do RStudio**: arquivo no formato `.Rproj` que facilita o trabalho com o RStudio, pois define o diretÃ³rio automaticamente e permite o controle de versÃ£o
- **Pull request**: Ã© um mÃ©todo de submeter contribuiÃ§Ãµes que serÃ£o revisadas pelos responsÃ¡veis de um projeto de desenvolvimento aberto
- **Raiz**: representa o ancestral comum de todas as espÃ©cies na filogenia
- **Ramo**: uma linha orientada ao longo de um eixo terminais-raiz que conecta os nÃ³s na filogenia
- **RarefaÃ§Ã£o**: Ã© uma mÃ©trica usada para calcular o nÃºmero esperado de espÃ©cies em cada comunidade tendo como base comparativa um valor em que todas as amostras ou nÃºmero de indivÃduos atinjam um tamanho padrÃ£o entre as comunidades
- **RarefaÃ§Ã£o baseada nas amostras (*Sampled-based*)**: as comparaÃ§Ãµes sÃ£o padronizadas pela comunidade com menor nÃºmero de amostragens
- **RarefaÃ§Ã£o baseada na cobertura (*Coverage-based*)**: Ã© uma medida que determina a

proporÃ§Ã£o de amostras ou do nÃºmero de indivÃduos da comunidade que representa as espÃ©cies amostradas

- **RarefaÃ§Ã£o baseada nos indivÃduos (*Individual-based*)**: as comparaÃ§Ãµes sÃ£o feitas considerando a abundÃ¢ncia da comunidade padronizada pelo menor nÃºmero de indivÃduos
- **RasterizaÃ§Ã£o**: conversÃ£o de dados vetoriais para raster realizada de pontos, linhas ou polÃgonos para rasters
- **RegressÃ£o**: Ã© um teste usado para analisar a relaÃ§Ã£o entre uma ou mais variÃ¡veis preditoras (Xn) e uma variÃ¡vel resposta (Y). A regressÃ£o assume uma relaÃ§Ã£o de causa e efeito entre as variÃ¡veis
- **ReprojeÃ§Ã£o**: transformaÃ§Ã£o do Sistema de ReferÃªncia de Coordenadas (CRS) de um dado geoespacial, alterando o CRS original para outro. Na reprojeÃ§Ã£o alteramos tanto a unidade do dado geoespacial (unidades em â longitude/latitudeâ ou unidades de metros), quanto o Datum *datum*
- **ResÃduo**: Ã© a diferenÃ§a entre os valores preditos e observados dos dados. SÃ£o tambÃ©m chamados de erro
- **ResoluÃ§Ã£o (raster)**: termo amplo que pode ser usado em diversos contextos. Para dados raster, Ã© geralmente associado ao tamanho (dimensÃ£o) do pixel (i.e., altura e largura)
- **RStudio**: ambiente de desenvolvimento integrado (IDE) para R. Inclui um console, editor de realce de sintaxe que suporta execuÃ§Ã£o direta de cÃ³digo, bem como ferramentas para plotagem, histÃ³rico, depuraÃ§Ã£o e gerenciamento de espaÃ§o de trabalho
- **Script (RStudio)**: arquivos de texto simples, criados com a extensÃ£o (terminaÃ§Ã£o) `.R` onde os cÃ³digos sÃ£o escritos e salvos
- **SeleÃ§Ã£o de modelos**: Ã© um processo usado para comparar o valor relativo de diferentes modelos estatÃsticos e determinar qual deles Ã© o mais adequado para os dados observados
- **SÃ©rie de Hill**: veja NÃºmero de Hill
- **Singleton**: nÃºmero de espÃ©cies observadas com abundÃ¢ncia de um indivÃduo
- **Sistema de Coordenadas**: composto por dois sistemas, o Sistema de Coordenadas GeogrÃ¡ficas e o Sistema de Coordenadas Projetadas, que Ã¢ngulos e metros, respectivamente
- **Sistema de ReferÃªncia de Coordenadas (CRS)**: tambÃ©m chamada projeÃ§Ã£o, define a referÃªncia geoespacial dos dados vetor e raster na superfÃcie da Terra, composto pelo Sistema de Coordenadas e *Datum*
- **Tabela de atributos**: dado tabular que inclui dados geoespaciais e dados alfanumÃ©ricos, geralmente associada a dados vetoriais
- **Terminal (do inglÃªs tip)**: o final do ramo representando uma espÃ©cie atual ou extinta (pode tambÃ©m representar gÃªneros, indivÃduos, genes, etc.)
- **Teste de Levene**: Ã© um teste estatÃstico utilizado para verificar a homogeneidade da variÃ¢ncia entre dois ou mais grupos
- **Teste de Shapiro-Wilk**: Ã© um teste estatÃstico utilizado para verificar se os dados apresentam distribuiÃ§Ã£o normal
- **Teste T (de Student)**: Ã© um teste estatÃstico que segue uma distribuiÃ§Ã£o t de *Student* para rejeitar ou nÃ£o uma hipÃ³tese nula de mÃ©dias iguais entre dois grupos
- **Teste T pareado**: Ã© um teste estatÃstico que usa dados medidos duas vezes na mesma unidade amostral, resultando em pares de observaÃ§Ãµes para cada amostra (amostras pareadas). Ele determina se a diferenÃ§a da mÃ©dia entre duas observaÃ§Ãµes Ã© zero
- **Tibble**: classe de objetos que Ã© uma versÃ£o aprimorada do data frame. Ã a classe aconselhada para que as funÃ§Ãµes do *tidyverse* funcionem melhor sobre conjuntos de dados tabulares
- **tidyverse**: â dialeto novoâ para a linguagem R, onde *tidy* quer dizer organizado, arrumado, ordenado, e *verse* Ã© universo. Operacionalizado no R atravÃ©s de uma coleÃ§Ã£o de pacotes que atuam no fluxo de trabalho comum da ciÃªncia de dados: importaÃ§Ã£o, manipulaÃ§Ã£o, exploraÃ§Ã£o, visualizaÃ§Ã£o, anÃ¡lise e comunicaÃ§Ã£o de dados e anÃ¡lises. O principal objetivo do *tidyverse* Ã© aproximar a linguagem para melhorar a interaÃ§Ã£o entre ser humano e computador sobre dados, de modo que os pacotes compartilham uma filosofia de design de alto nÃvel e gramÃ¡tica, alÃ©m da estrutura de dados de baixo nÃvel
- **UltramÃ©trica**: a distÃ¢ncia de todos os terminais atÃ© a raiz sÃ£o idÃªnticas. CaracterÃstica requerida pela maioria dos Ãndices de diversidade filogenÃ©tica

- **Unidade amostral**: Ã© o indivÃduo (ou elemento) da populaÃ§Ã£o amostral sobre o qual a medida de interesse serÃ¡ observada
- **Unique**: nÃºmero de espÃ©cies observadas em apenas uma amostra
- **Unique Fraction metric (UniFrac)**: mÃ©trica de dissimilaridade que determina a fraÃ§Ã£o Ãºnica da filogenia contida em cada uma das duas comunidades
- **Valor de P**: probabilidade de um teste estatÃstico ser igual ou maior que o observado, dado que a hipÃ³tese nula Ã© verdadeira
- **Valores faltantes e especiais**: valores reservados que representam dados faltantes, indefiniÃ§Ãµes matemÃ¡ticas, infinitos e objetos nulos
- **Variance of Pairwise Distance (VPD)**: Ã© uma mÃ©trica que utiliza a matriz de distÃ¢ncia filogenÃ©tica para quantificar a variÃ¢ncia do parentesco entre pares de espÃ©cies em uma comunidade
- **VariÃ¡vel categÃ³rica**: sÃ£o variÃ¡veis que nÃ£o possuem valores quantitativos e sÃ£o definidas por categorias ou grupos distintos
- **VariÃ¡vel contÃnua**: sÃ£o variÃ¡veis numÃ©ricas que tÃªm um nÃºmero infinito de valores entre dois valores quaisquer. Neste caso, valores fracionais fazem sentido
- **VariÃ¡vel dependente**: Ã© uma variÃ¡vel mensurada ou observada de interesse do pesquisador que depende do valor de outra variÃ¡vel
- **VariÃ¡vel discreta**: Ã© uma variÃ¡vel numÃ©rica que tÃªm um nÃºmero contÃ¡vel de valores inteiros
- **VariÃ¡vel explicativa**: ver variÃ¡vel independente
- **VariÃ¡vel independente**: variÃ¡vel mensurada ou observada pelo pesquisador que prevÃª ou afeta a variÃ¡vel dependente
- **VariÃ¡vel nominal**: sÃ£o variÃ¡veis categÃ³ricas que nÃ£o apresentam ordenaÃ§Ã£o dentre as categorias (e.g., preto, branco e rosa)
- **VariÃ¡vel ordinal**: sÃ£o variÃ¡veis categÃ³ricas com ordenaÃ§Ã£o entre as categorias (e.g., pequeno, mÃ©dio e grande)
- **VariÃ¡vel preditora**: ver variÃ¡vel independente
- **VariÃ¡vel resposta**: ver variÃ¡vel dependente
- **Variograma**: Ã© uma descriÃ§Ã£o da continuidade espacial dos dados. Ele mede a variabilidade entre pares de pontos em vÃ¡rias distÃ¢ncias
- **Vetor**: classe de objetos que representa o encadeamento de elementos de um Ãºnico modo numa sequÃªncia unidimensional
- **Vetor (Multivariada)**: coluna de uma matriz
- **VetorizaÃ§Ã£o (vetor)**: conversÃ£o de dados rasters para vetor, sendo que esse vetor receberÃ¡ os valores dos pixels do raster, podendo ser convertido em pontos, isolinhas ou polÃgonos

A Annexo 1

Exemplos de cÃ³digo R usado para - baixar Mapbiomas, - recortar para uma regiÃ£o de interesse - plotar com legenda

```
library(plyr)
library(tidyverse)
library(terra)
library(sf)
library(readxl)
library(tmap)
```

A.1 Download mapbiomas cover

Site: https://brasil.mapbiomas.org/colecoes-mapbiomas/. Any zeros are NA (NODATA). http://forum.mapbiomas.ecostage.com.br/t/pixels-com-valor-zero/170/5 . And "27" â NÃ£o observadoâ . No exemplo os arquivos vai ficar no working directory.

```r
#Get mapbiomas data
#
url_text <- "https://storage.googleapis.com/mapbiomas-public/initiatives/brasil/collection_8/lclu/covera
layer_names <- paste("brasil_coverage_",1985:2022, sep="")
file_type <- ".tif"
filenames_mapbioma_8 <- paste(layer_names, file_type, sep="")
urlnames_mapbioma_8 <- paste(url_text, layer_names, file_type, sep="")

#set timeout to 20 minutes as big files and teeny tiny internet connection
options(timeout = max((60*20), getOption("timeout")))
# test with one year - 1985
res <- utils::download.file(url=urlnames_mapbioma_8[1],
                            destfile=filenames_mapbioma_8[1],
                            method="auto", quiet = FALSE, mode = "wb",
                            cacheOK = TRUE)
#make list
dflist <- data.frame(layer_names = layer_names,
                     urls = urlnames_mapbioma_8,
                     filenames = filenames_mapbioma_8)
alist <- split(dflist, f = dflist$layer_names)

#make function to automatically process list elements
get_mapbiomas <- function(x){
res <- utils::download.file(url=x$urls,
                            destfile=x$filenames,
                            method="auto", quiet = FALSE, mode = "wb",
                            cacheOK = TRUE)
res
}

#run function in blocks
lapply(alist[2], get_mapbiomas)
lapply(alist[3:5], get_mapbiomas)
lapply(alist[6:8], get_mapbiomas)
lapply(alist[9:20], get_mapbiomas)
lapply(alist[21:30], get_mapbiomas)
lapply(alist[31:37], get_mapbiomas)
```

A.2 Crop mapbiomas cover

Identify all tif files in a folder.

```r
#works one folder at a time
get_files <- function(folder_name = NA) {
  library(tidyverse)
  folder_location <- folder_name
  in_files <- list.files(folder_location,
                         pattern = ".tif", full.names = TRUE)
  data.frame(folder_id = folder_location, file_id = in_files) |>
    group_by(folder_id, file_id) |>
    dplyr::summarise(file_count = dplyr::n()) |>
    ungroup() -> df_muni_tif
  return(df_muni_tif)
}
```

```r
infolder <- "E:/mapbiomas"
df_muni_tif <- get_files(folder_name = infolder)
#update
df_muni_tif[1:38, ] |>
  dplyr::mutate(state_code = str_sub(folder_id, -2, -1),
         ) -> df_muni_tif
df_muni_tif$state_code <- "AP"
```

Area of interest - format extent for use by terra.

```r
#extent from 50 km arouund river upstream of cachoeira caldeirao
meuSIG <- file.choose()
# "C:\\Users\\user\\Documents\\CA\\CA_2023\\EP\\vectors\\AP_rodoviario\\br156.gpkg"
#"C:\\Users\\Darren\\Documents\\gisdata\\vector\\rivers.GPKG"
# "C:\\Users\\user\\Documents\\Articles\\gis_layers\\gisdata\\inst\\vector\\rivers.gpkg"
rsl <- st_read(meuSIG, layer = "centerline")
rsl_50km <- st_union(st_buffer(rsl, dist=50000))
br156 <- st_read(meuSIG, layer = "extent_br156")
myexent <- ext(vect(rsl_50km))
myexent_jari <- vect(br156[1, ])
myexent_portogrande <- vect(br156[2, ])
```

Now to crop and project to desired coordinate system (epsg:31976).

```r
# This function reads in file, crops and projects to new coordinate system.
# Then exports resutl as Geotiff (.tif).
crop_proj <- function(x, myextent = NA,
                      folder_name = NA, file_name = NA,
                      state_id = NA, sf_state = NA) {
  library(plyr)
  library(tidyverse)
  library(terra)
  library(sf)
  library(stringi)

  state_sigla <- x$state_code
  rin <- x$file_id
  #rin <- df_muni_tif$file_id[1]
  rbig <- terra::rast(rin)
  layer_name <- names(rbig)
  layer_year <- stri_sub(layer_name,-4,-1)
  out_name <- paste("mapbiomas_cover", layer_year, sep="_")
  e2 <- ext(project(myextent, rbig))

  #Crop
  rtest_mask <- crop(rbig, e2, snap="out")
  rm("rbig")

  #Project
  new_crs_utm <- "epsg:31976"
  rtest_mask <- project(rtest_mask, new_crs_utm, method = "near")
  names(rtest_mask) <- out_name

  #Export
  outfile_name <- paste(file_name,layer_year, ".tif", sep = "")
```

```
  f <- file.path(folder_name, outfile_name)
  writeRaster(rtest_mask, f, datatype = "INT1U", NAflag=27,
              gdal=c("COMPRESS=DEFLATE"), overwrite = TRUE)

  #clear temporary files
  tmpFiles(current =TRUE, remove = TRUE)

  endtime <- Sys.time()
  textout <- paste(outfile_name, ": ", endtime, sep="")
  print(textout)
}

#run br156 jari
plyr::a_ply(df_muni_tif[2:38, ], .margins = 1,
            .fun = crop_proj, myextent = myexent_jari,
            folder_name = "E:/mapbiomas/br156_jari",
            file_name = "br156_jari_")
#run br156 Porto Grande
plyr::a_ply(df_muni_tif[2:38, ], .margins = 1,
            .fun = crop_proj, myextent = myexent_portogrande,
            folder_name = "E:/mapbiomas/br156_portogrande",
            file_name = "br156_portogrande_")
```

A.3 Plot with legend

Combine all years into multi layer raster.

```
# check
rtest <- rast("E:/mapbiomas/br156_jari/br156_jari_1985.tif")
rtest <- rast("E:/mapbiomas/br156_portogrande/br156_portogrande_1985.tif")
plot(rtest)
infolder_rio <- "E:/mapbiomas/AP_utm_rio"
df_tif_rio <- get_files(folder_name = infolder_rio)
r85a22 <- rast(df_tif_rio$file_id)
writeRaster(r85a22, "E:/mapbiomas/utm_cover_AP_rio_85a22.tif",
            datatype = "INT1U", NAflag=27,
            gdal=c("COMPRESS=DEFLATE"), overwrite = TRUE)
# br156 jari
infolder_jari <- "E:/mapbiomas/br156_jari"
df_tif_jari <- get_files(folder_name = infolder_jari)
r85a22 <- rast(df_tif_jari$file_id)
writeRaster(r85a22, "E:/mapbiomas/br156_jari_85a22.tif",
            datatype = "INT1U", NAflag=27,
            gdal=c("COMPRESS=DEFLATE"), overwrite = TRUE)
# br156 Porto Grande
infolder_pg <- "E:/mapbiomas/br156_portogrande"
df_tif_pg <- get_files(folder_name = infolder_pg)
r85a22 <- rast(df_tif_pg$file_id)
writeRaster(r85a22, "br156_portogrande_85a22.tif",
            datatype = "INT1U", NAflag=27,
            gdal=c("COMPRESS=DEFLATE"), overwrite = TRUE)
# Garimpo Lorenco
infolder_lor <- "E:/mapbiomas/garimpo_lorenco"
df_tif_lor <- get_files(folder_name = infolder_lor)
```

```
r85a22 <- rast(df_tif_lor$file_id)
writeRaster(r85a22, "E:/mapbiomas/lorenco_85a22.tif",
            datatype = "INT1U", NAflag=27,
            gdal=c("COMPRESS=DEFLATE"), overwrite = TRUE)
# Garimpo Vila Nova
infolder_vn <- "E:/mapbiomas/garimpo_vilanova"
df_tif_vn <- get_files(folder_name = infolder_vn)
r85a22 <- rast(df_tif_vn$file_id)
writeRaster(r85a22, "E:/mapbiomas/vilanova_85a22.tif",
            datatype = "INT1U", NAflag=27,
            gdal=c("COMPRESS=DEFLATE"), overwrite = TRUE)
```

Legend.

```
mapbiomas8_legenda <- read.csv2("legend/Codigos-da-legenda-colecao-8.csv")
```

Plot.

```
class_vals <- mapbiomas8_legenda$Class_ID
class_color <- mapbiomas8_legenda$Color
names(class_color) <-  mapbiomas8_legenda$Class_ID
#labels
my_label <- mapbiomas8_legenda$Descricao
names(my_label) <- mapbiomas8_legenda$Class_ID

# Plot one year to check
tm_shape(subset(r85a22n_ag, c(1, 2, 37,38))) +
  tm_raster(style = "cat", palette = class_color, labels = my_label, title="") +
  tm_scale_bar(breaks = c(0, 10), text.size = 1,
               position=c("right", "bottom"))

# Now all years together.
tm_shape(r85a22n_ag) +
  tm_raster(style = "cat", palette = class_color, labels = my_label, title="") +
  tm_scale_bar(breaks = c(0, 10), text.size = 1,
               position=c("right", "bottom")) +
  tm_facets(ncol=5) +
  tm_layout(legend.bg.color="white", legend.outside = TRUE,
            legend.outside.position = "right",
            panel.labels = c('1985', '1986', '1987', '1988','1989', '1990',
                             '1991',  '1992', '1993', '1994', '1995','1996',
                             '1997', '1998',
                             '1999', '2000', '2001', '2002', '2003',
                             '2004', '2005', '2006', '2007',
                             '2008', '2009', '2010', '2011', '2012',
                             '2013', '2014', '2015',
                             '2016', '2017', '2018', '2019',
                             '2020', '2021', '2022')) -> figmap

png("figmap.png", width=6, height=9,
    units="in", res = 600)
figmap
invisible(dev.off())
```