Proposta de guia prático para análise dos dados do Protocolo Campestre-Savânico do programa MONITORA

Autor: Ronald Sodré Martins[1]

Co-autores: Celso Santos[1], Gislene Martins[1]

[1] Instituto Chico Mendes de Conservação da Biodiversidade (ICMBio) / Núcleo de Gestão Integrado Cautário-Guaporé

Data de criação do documento: 21 de junho de 2024

Última atualização do documento: 21 de agosto de 2024

Lattes, GitHub

Resumo

Proposta de guia prático para importação, manipulação e análise dos dados no formato XML, obtidos pelo *software* ODK, e planilhas XLSX utilizados no protocolo Campestre-Savânico dentro do programa MONITORA.

Introdução

A eficácia das ações de conservação da biodiversidade precisa ser monitorada, tanto para avaliar o sucesso dos programas de manejo quanto para prestar contas à sociedade. No Cerrado brasileiro, as áreas protegidas sofrem um conjunto de ameaças que resultam em forte ocupação por espécies exóticas invasoras, regimes inadequados de fogo e rápida degradação de habitat. O Programa MONITORA (Programa Nacional de Monitoramento da Biodiversidade), coordenado pelo ICMBio, foi desenvolvido para lidar simultaneamente com diversas dessas questões. Ele é participativo tanto em sua concepção quanto na execução, visando à ampla participação da sociedade e espera-se que seja executado por não especialistas em biodiversidade.

O Programa tem protocolos que permitem respostas a múltiplos fatores de pressão e é implementado ao mesmo tempo em numerosas Unidades de Conservação, visando à obtenção de informações e entendimentos em diversas escalas. Esses protocolos são bastante simples e, no caso do Componente Campestre-Savânico, podem ser obtidas informações valiosas a partir do levantamento da proporção entre diferentes formas de vida das plantas, mesmo sem sua identificação específica.

Quanto à gestão dos dados produzidos, o ICMBio tem desenvolvido sistemas de informação de biodiversidade para apoiar diversos processos, em alguns casos fornecendo serviços e disponibilizando ferramentas aos gestores, aos tomadores de decisão e à sociedade como um todo, todos lastreados em políticas de dados

construídas coletivamente, à luz da Lei de Acesso à Informação, e devidamente publicadas.

No âmbito do Monitora, está sendo desenvolvido um banco de dados estruturado para recepcionar os dados dos alvos globais e complementares dos Subprogramas Terrestre e Aquático Continental e parte dos alvos do Marinho-Costeiro. O Sistema de Gestão de Dados do Programa Monitora - SISMonitora – já tem sido útil, entre outras funções, para recepcionar, armazenar e disponibilizar os dados do Monitora.

A orientação geral é de disponibilização dos dados à sociedade no menor tempo possível. Em alguns casos, o período de carência dos dados brutos pode ser de até cinco anos. Esta opção é dada aos projetos de pesquisa cadastrados no Sisbio e aos envolvidos nos protocolos avançados do MONITORA, mas com estímulo à liberação em tempo menor, que vem sendo adotada amplamente. Além disso, de forma agregada, pode-se usar os dados ainda em carência para análises necessárias à gestão.

Este guia foi criado para auxiliar na elaboração de análises preliminares dos resultados do protocolo Campestre-Savânico do programa MONITORA. Atualmente, o Programa MONITORA está implementando o uso de dispositivos móveis com o *software* Open Data Kit (ODK) que, por sua vez, adota formulários no formato XML para coletar dados em campo.

Nosso objetivo é permitir as análises dos dados coletados do protocolo Campestre-Savânico do programa MONITORA possam ser realizadas localmente e de forma independente, ou seja, por qualquer servidor ou colaborador com acesso a um computador, sem a necessidade de especialistas em análise de dados. É importante destacar que este guia não tem a intenção de substituir o SISMonitora, mas sim permitir que outros gestores de Unidades de Conservação realizem análises preliminares dos resultados, incluindo dados não validados, apresentando-os para a comunidade e permitindo as primeiras discussões.

O guia contém caixas de códigos que podem ser utilizados para realizar as análises preliminares. Esse conjunto de códigos permite importar, manipular e analisar dois tipos de dados: 1) arquivos no formato XML, salvos nos dispositivos (celulares ou tablets) utilizados em campo durante as expedições do programa MONITORA que registraram os dados amostrais pelo *software* ODK, e 2) planilhas antigas no formato XLSX (formato padrão do Excel), adotadas anteriormente à implementação do ODK. Após a importação desses arquivos, os dados são tratados para que as análises preliminares sejam realizadas.

Vale ressaltar que este guia pode ser adaptado para diferentes protocolos dentro do Programa MONITORA. Além disso, o código atual pode ser implementado futuramente no SISMonitora. A linguagem R foi adotada por ser relativamente simples, robusta e apresentar bom suporte para análise de dados.

Justificativa

Os dados coletados pelo programa MONITORA, dentro da plataforma SISMonitora, podem ser baixados nos formatos CSV ou XLS (Figura 01). Ao baixar os pontos de amostragem no formato CSV, é possível abrir o arquivo em um leitor de planilhas.

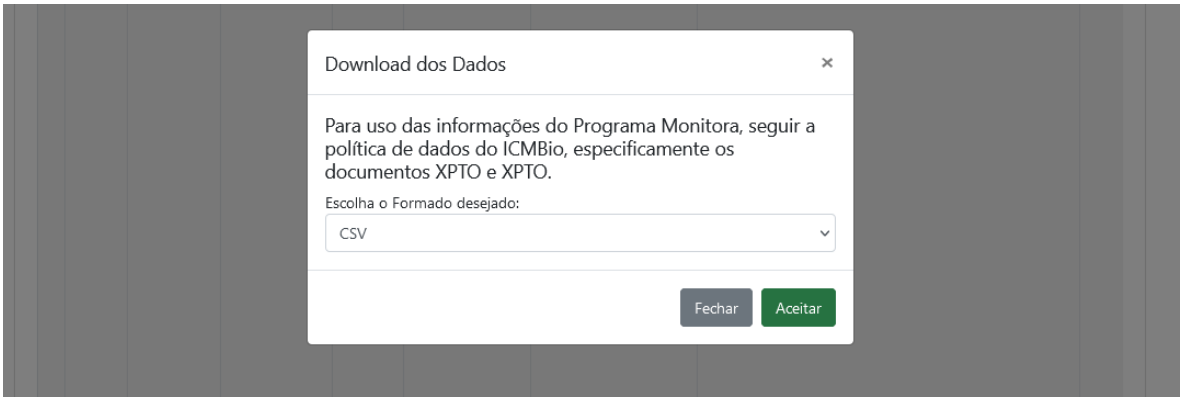


Figura 01: imagem capturada ao realizar o download de uma unidade amostral na plataforma SISMonitora.

Como exemplo, baixamos a unidade amostral 1101 (número da plaqueta) da Reserva Biológica do Guaporé, coletada durante a expedição de 2023 do Programa MONITORA, componente Campestre-Savânico, e abrimos o arquivo no *software* Excel (Figura 02).

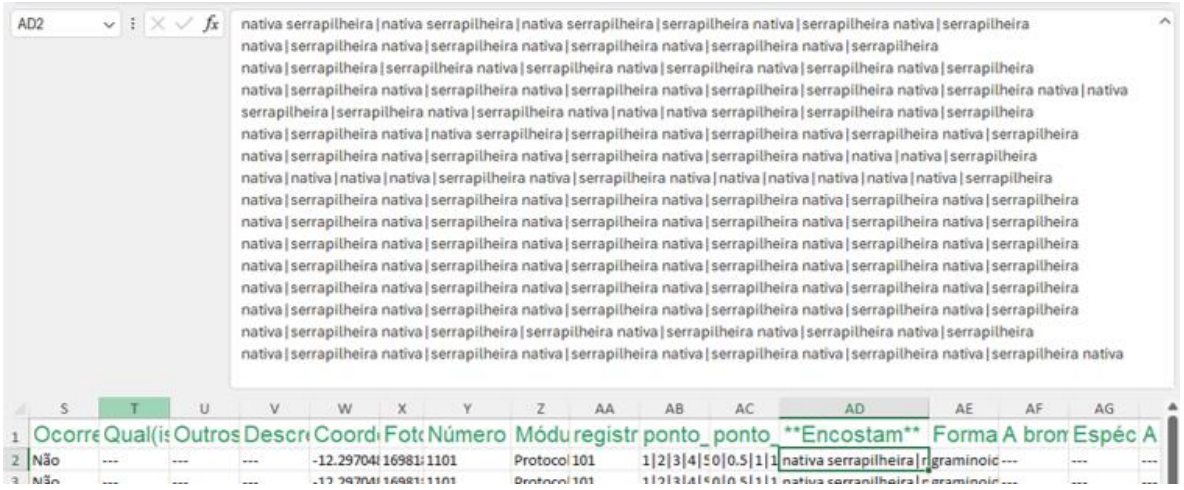


Figura 02: imagem das informações do que encosta na vareta no ponto 1 do arquivo da Unidade amostral 1101 coletada na Reserva Biológica do Guaporé na expedição 2023 dentro do programa MONITORA aberto no Excel.

Como podemos observar no nosso exemplo (figura 02), dentro da unidade amostral 1101, todos os elementos que encostam na vareta de todos os pontos amostrais estão repetidos dentro da mesma célula, utilizando o separador "|". Além disso, por um motivo desconhecido no SISMonitora, todo o conteúdo dentro das linhas da planilha está repetido. Ainda no nosso exemplo, observamos que todas as formas de vida no ponto 1 (figura 03) e no ponto 2 (figura 04) também estão repetidas dentro da mesma célula, adotando o separador "|". Entretanto, o último elemento diverge: no ponto 1

(figura 03), o último elemento é “graminoide arbusto_acima”, enquanto no ponto 2 (figura 04), o último elemento é “graminoide”.

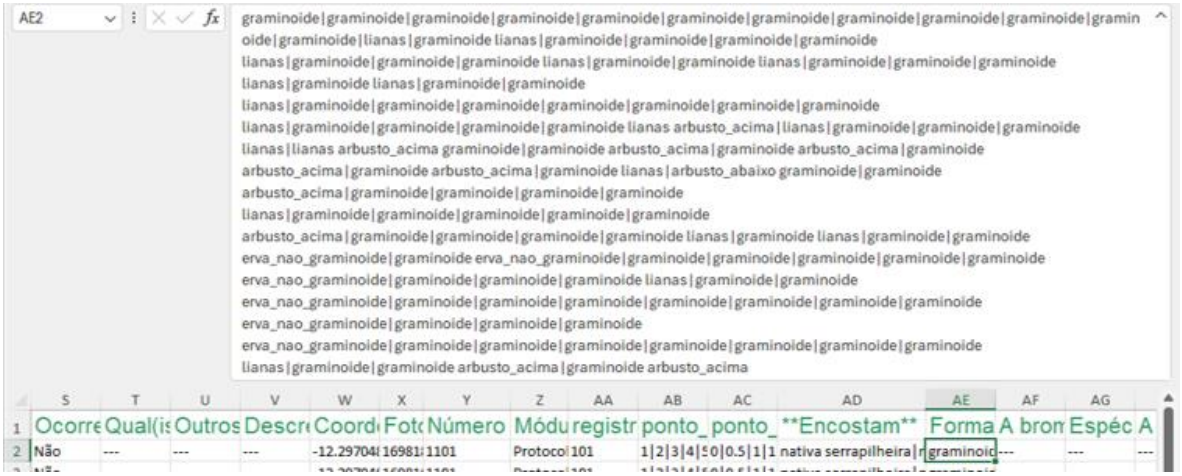


Figura 03: imagem das informações das formas de vidas nativas na primeira linha da unidade amostral 1101 coletada na REBIO do Guaporé na expedição 2023 dentro do programa MONITORA aberto no Excel.

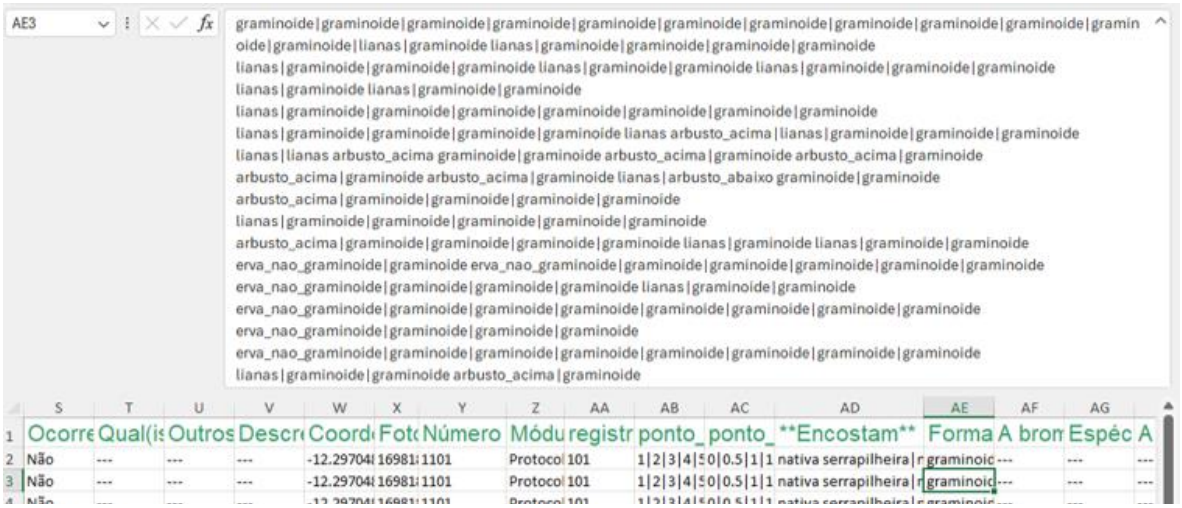


Figura 04: imagem das informações das formas de vidas nativas na segunda linha da unidade amostral 1101 coletada na REBIO do Guaporé na expedição 2023 dentro do programa MONITORA aberto no Excel.

Após analisar outras unidades amostrais, observamos que essas repetições do último elemento ocorrem de forma decrescente. No nosso exemplo da unidade amostral 1101 da Reserva Biológica do Guaporé, coletada em 2023, no ponto 1 ocorre repetição do último ponto de amostragem, repetindo o item “graminoide arbusto_acima”. Já no ponto 2, ocorre repetição do penúltimo ponto amostral, no caso “graminoide”.

Apesar das observações descritas acima, não podemos confirmar que isso ocorra de forma padronizada em todas as unidades amostrais, nem em todos os dados coletados em diferentes expedições e Unidades de Conservação. Nesse contexto, torna-se inviável confiar no formato atual dos dados disponíveis no SISMonitora para a análise preliminar pretendida neste trabalho.

Diante desse cenário, enquanto o SISMonitora não está em pleno funcionamento, identificou-se a necessidade de propor um guia prático que demonstre como converter os arquivos oriundos do ODK, no formato XML, para o formato tabular (formato de tabela) para serem analisados pelos gestores. Aproveitamos para confeccionar esse guia de modo também a aceitar as planilhas no formato XLSX, utilizadas antes da implementação do ODK ao Programa MONITORA. Além disso, também apresentamos alguns comandos básicos para analisar esses dados.

Portanto, esse guia foi pensado em um fluxo prático e direcionado contendo comandos básicos na linguagem R, permitindo importação dos dados coletados pelo protocolo Campestre-Savânico no Programa MONITORA e suas análises por outros gestores de Unidades de Conservação sem depender de especialistas.

Metodologia

Este guia contém: como encontrar os arquivos XML nos dispositivos utilizados nas coletas de dados em campo; a instalação do R e RStudio em um computador; como importar os dados do ODK (formato XML) e das planilhas no formato XLSX para dentro do ambiente do RStudio; alguns comandos básicos na linguagem R e como utilizar o RStudio; algumas funções em R para converter os arquivos ODK e XLSX para formato tabular; como salvar os arquivos tabulares em uma planilha editável no computador local; e como analisar os dados através de comandos em linguagem R.

Com este guia que estamos disponibilizando, qualquer gestor com pouca experiência com computadores, mesmo sem experiência prévia com a linguagem R, poderá utilizar o R dentro da interface do RStudio para analisar os dados do protocolo Campestre-Savânico do programa MONITORA.

Encontrando os arquivos XML

A seguir, iremos demonstrar uma maneira de encontrar os arquivos no formato XML do programa ODK dentro dos dispositivos utilizados durante as expedições do protocolo Campestre-Savânico.

Para ilustrar, utilizaremos um computador com Windows e um dispositivo tablet Android utilizado na coleta de dados. Primeiramente, conecte o dispositivo (celular ou tablet usado no monitoramento) ao computador onde será realizada a análise dos dados, via cabo usb e abra o explorador de arquivo `Windows+e` (figura 05).

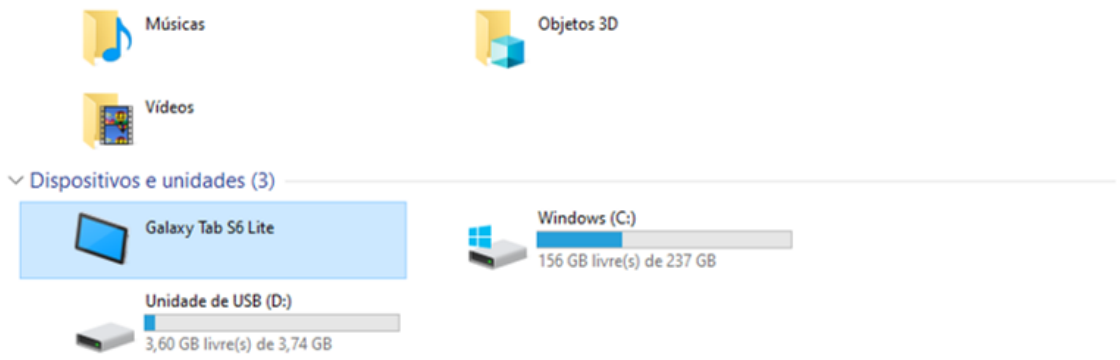


Figura 05: Ilustração do dispositivo utilizado no protocolo Campestre-Savânico conectado ao computador local.

Em seguida, entre no armazenamento interno do dispositivo (figura 06).

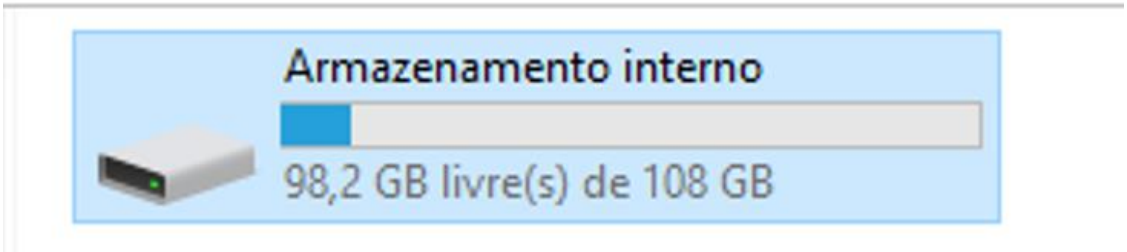


Figura 06: Ilustração do armazenamento interno do dispositivo.

Dentro do armazenamento interno, utilize a ferramenta de busca localizada na parte superior direita, digitando "ODK" (figura 07). A pasta contendo todas as informações do programa ODK deverá aparecer com o nome "org.odk.collect.android", caso o dispositivo seja android. Em seguida, entre na subpasta "files", em seguida, nas subpasta "projects" (figura 08), que armazena todos os protocolos salvos. O nome de cada protocolo é um conjunto de caracteres e não segue nenhum um padrão visível (figura 09).

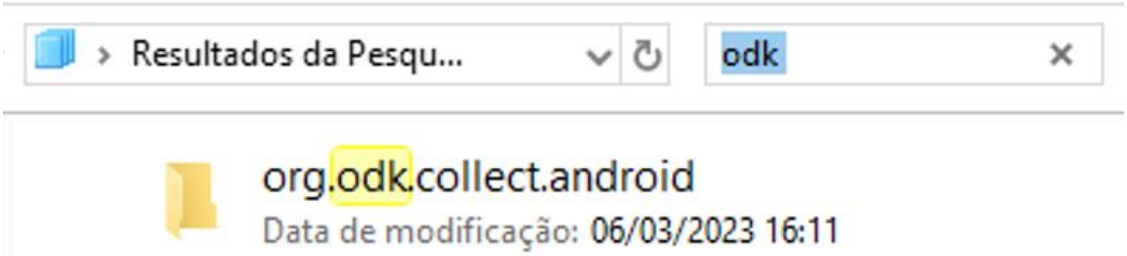


Figura 07: Ilustração do resultado de busca "odk".

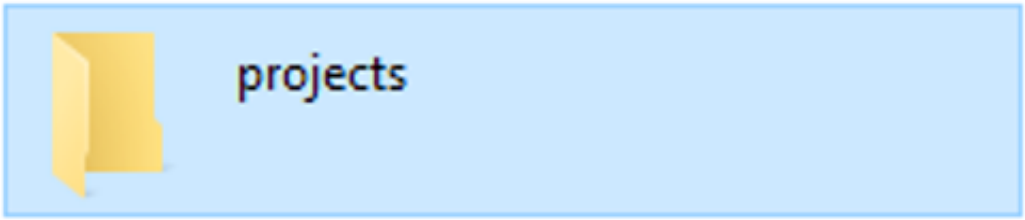


Figura 08: Ilustração da subspasta "projects" dentro da pasta "org.odk.collect.android".

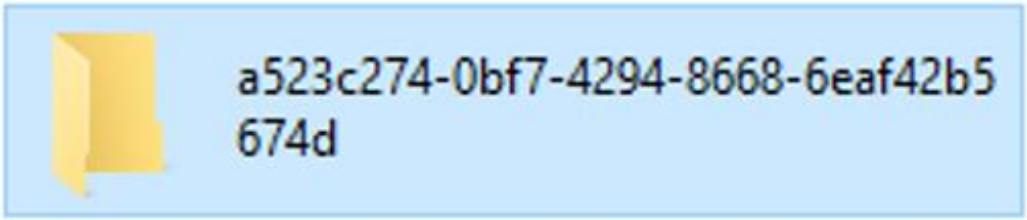


Figura 09: Ilustração da subpasta onde armazena o protocolo no programa ODK.

Na pasta do protocolo de interesse, existe a subpasta "forms" onde está armazenado o formulário do protocolo e a subpasta "instances" com os dados coletados durante a expedição (figura 10). Na pasta "instances", cada subpasta representa uma unidade amostral, e dentro dessas subpastas, estão as imagens coletadas e o arquivo no formato XML. (figura 11).

Vale destacar que o Sistema Operacional Windows costuma relacionar os arquivos XML como HTML, o que pode confundir os usuários.

Uma outra obsevação que fazemos é que o Windows costuma não permitir copiar e colar as pastas das unidades amostrais do programa ODK, porém, permite copiar os arquivos XML individualmente.

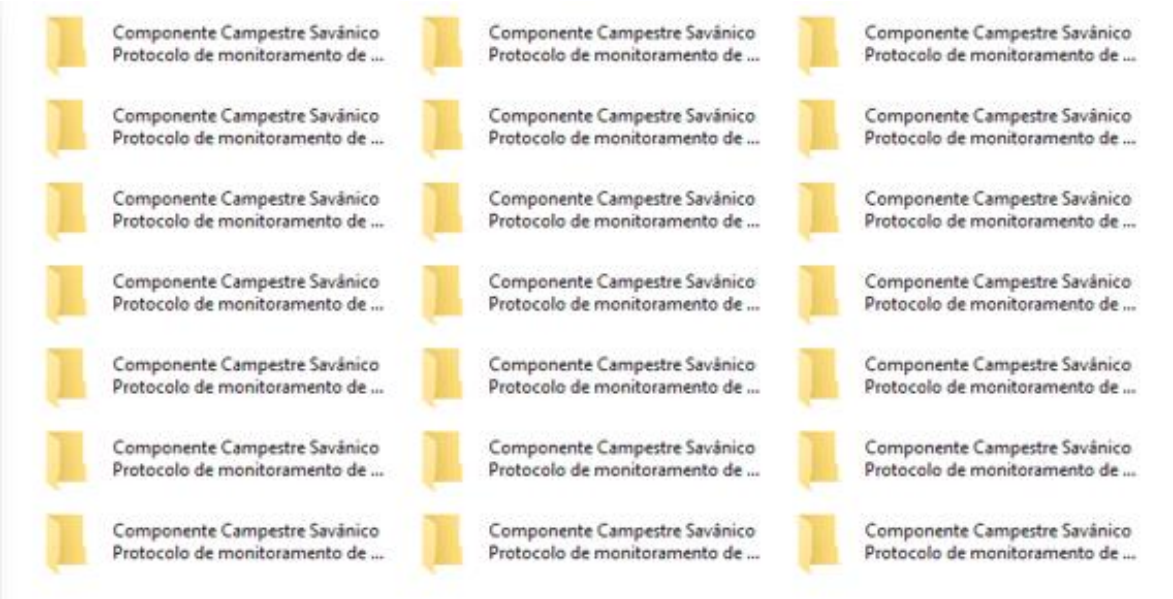


Figura 10: Ilustração do interior da subpasta "instances" que contém os dados coletados de todas as unidades amostrais.

Figura 11: Ilustração de um exemplo da subpasta de uma unidade amostral.

Indicamos que seja criada uma pasta dentro do computador local onde serão feitas as análises dos dados, e nessa nova pasta sejam armazenados todos os arquivos XML do protocolo. Não é necessário remover os arquivos do dispositivo, basta copiá-los e colá-los na nova pasta.

R e RStudio

A linguagem R é amplamente utilizada em análise de dados, oferecendo uma ampla gama de pacotes estatísticos e gráficos. O RStudio é um programa que utiliza a linguagem R, possui uma interface amigável e robusta, e facilita a organização e execução de scripts, a visualização de resultados e a gestão de projetos. A combinação do R e do RStudio proporciona uma plataforma poderosa e eficiente para realizar análises complexas de forma prática e intuitiva. Ressaltamos que o RStudio é um programa que possui uma versão gratuita oficial.

A seguir, disponibilizamos um resumo sobre como baixar e instalar o R, baixar e instalar o RStudio, criar um script, abrir um editor de código e adicionar códigos no RStudio.

Instalação do R e RStudio

Primeiramente, vamos baixar o R no link https://cran.rstudio.com/. Escolha o seu sistema operacional (Windows, macOS ou Linux) e siga as instruções para baixar o instalador adequado. Após baixar o instalador, execute o arquivo e siga as instruções até o fim da instalação do R. Em seguida, acesse o site do RStudio em

https://posit.co/products/open-source/rstudio/. Selecione a versão gratuita "Open Source Edition" e escolha o link correspondente ao seu sistema operacional. Baixe o instalador e execute o arquivo, seguindo as instruções do instalador até que a instalação esteja concluída. Importante destacar que é aconselhável a instalação do R antes do RStudio.

Comandos básicos no RStudio

A seguir, apresentaremos um breve resumo de alguns comandos no RStudio. Após a instalação do R e do RStudio, abra o RStudio clicando no ícone correspondente no seu desktop ou menu de aplicativos. Para criar um *script*, clique em **`File`**, depois em **`New File`** e selecione **`R Script`**. Isso abrirá um novo editor de código onde você pode escrever seus scripts em R.

Se você já está no RStudio mas não vê o editor de código, pode abri-lo clicando em **`File`**, depois em **`New File`** e selecionando **`R Script`**. Isso abrirá uma nova aba no editor de código.

Para adicionar algum código ao RStudio, copie um código que você deseja. Os códigos estão disponíveis em capítulos posteriores nesse guia. Vá para o editor de código no RStudio e cole o código copiado. Você pode fazer isso clicando com o botão direito do mouse e selecionando **`Paste`** ou pressionando `Ctrl+V` (Windows) ou `Cmd+V` (Mac).

Para executar o código, selecione o trecho que deseja executar e pressione `Ctrl+Enter` (Windows) ou `Cmd+Enter` (Mac), ou clique no botão **`Run`** no canto superior direito do editor de código.

Por fim, caso queira salvar o script, clique em **`File`** e em **`Save`**, ou pressionando `Ctrl+S` (Windows) ou `Cmd+S` (Mac). Dê um nome ao seu arquivo e escolha um local para salvá-lo.

Códigos básicos

Disponibilizamos alguns códigos básicos em R para auxiliar o usuário. A maioria dos exemplos é genérica, servindo basicamente como exemplos para oferecer ao gestor uma breve noção do funcionamento da linguagem R. Caso haja dúvidas quanto aos códigos, podem entrar em contato, pois estamos nos disponibilizando a ajudar.

Instalação de pacotes

Durante a utilização da linguagem R, é necessário instalar pacotes que servem como extensões ou ferramentas extras que adicionam funcionalidades ao R. Mesmo após a instalação do pacote pela primeira vez, é necessário importá-lo para o ambiente R todas as vezes que for abrir o R. Em um capítulo posterior, iremos demonstrar todos os pacotes necessários para executar todas as funções nesse guia. Por ora, apresentaremos apenas um exemplo de como instalar um pacote.

```
if (!require(readxl)) install.packages("readxl")
```

Importar arquivos

Durante análises de dados, é necessário importar arquivos para dentro do ambiente RStudio. No caso deste guia, iremos importar os dados coletados do protocolo Campestre-Savânico em dois formatos: XML do programa ODK e as planilhas no formato XLSX. A importação dos dados pode ser feita de diferentes maneiras, adotando diferentes pacotes, dependendo principalmente do formato do arquivo. Como dito anteriormente, é necessário instalar e importar pacotes antes de importar os arquivos.

No exemplo a seguir, instalamos o pacote **readxl** e usamos sua função **read_excel** para importar uma planilha no formato XLSX. Entre aspas, é indicado o caminho até a planilha. A função **sheet** serve para especificar o nome ou número da aba da planilha. No nosso exemplo, importamos a planilha localizada em "caminho/para/seu_arquivo.xlsx" e nomeamos de "df_xlsx". Dentro do ambiente R, utilizaremos o nome "df_xlsx" para se referir a essa planilha. O ambiente R é *case-sensitive*, ou seja, existem diferenças entre letras minúsculas e maiúsculas.

```
if (!require(readxl)) install.packages("readxl")
df_xlsx <- read_excel("caminho/para/seu_arquivo.xlsx", sheet = 1)
```

No próximo exemplo, apresentamos o código para importar arquivos no formato CSV, que é amplamente utilizado em análises de dados. No caso dos arquivos CSV, não é necessário importar nenhum pacote específico, pois esse formato é suportado pelo ambiente R.

```
df_csv <- read.csv("caminho/para/seu_arquivo.csv")
```

Visualizando o arquivo

Vamos aprender como visualizar as primeiras linhas do arquivo importado adotando a função **head**. Vamos usar o arquivo fictício importado como exemplo.

```
head(df_csv)
```

Apagando objetos

Caso seja necessário, é possível remover os objetos importados, dataframes e funções criadas durante as análises de dentro do RStudio. Para isso, basta usar a função **rm**. A função **rm** permite apagar mais de um objeto; basta adicionar seus nomes dentro dos parênteses, separando-os por vírgulas.

```
rm(df_xlsx)
```

Salvando dataframes no formato CSV

Durante análises de dados, é comum modificar ou criar dataframes, e pode ser interessante salvá-los para posterior análise ou compartilhamento dos dados. Vamos fornecer um exemplo genérico de como salvar nosso arquivo fictício no formato CSV.

O formato CSV é amplamente utilizado em análises de dados e pode ser aberto em diferentes programas além da linguagem R, como o Excel.

```
write.csv(df_csv, "caminho/salvar_seu_dataframe.csv", row.names = FALSE)
```

Códigos em R para análise dos dados

Instalação dos pacotes necessários

Antes de realizar as análises de dados, é necessário instalar alguns pacotes do R. Os pacotes em R são coleções de funções, dados e documentações que ampliam as capacidades do R. A seguir, apresentamos um bloco de código em R com a lista de pacotes necessários que devem ser instalados para o funcionamento dos códigos desse guia.

Ressalto que, exclusivamente durante a primeira instalação dos pacotes, é necessário selecionar e executá-los mais uma vez.

```
if (!require(dplyr)) install.packages("dplyr")
if (!require(tidyr)) install.packages("tidyr")
if (!require(splitstackshape)) install.packages("splitstackshape")
if (!require(reshape2)) install.packages("reshape2")
if (!require(ggplot2)) install.packages("ggplot2")
if (!require(stringr)) install.packages("stringr")
if (!require(patchwork)) install.packages("patchwork")
if (!require(forcats)) install.packages("forcats")
if (!require(gt)) install.packages("gt")
if (!require(gplots)) install.packages("gplots")
if (!require(readxl)) install.packages("readxl")
if (!require(xml2)) install.packages("xml2")
if (!require(purrr)) install.packages("purrr")
if (!require(RColorBrewer)) install.packages("RColorBrewer")
if (!require(viridis)) install.packages("viridis")
if (!require(flextable)) install.packages("flextable")
```

Importação e tratamento dos dados

Arquivos do software ODK (XML)

Atualmente, alguns monitores estão coletando os dados do programa Campestre-Savânico utilizando o *software* ODK em campo. O ODK salva seus arquivos no formato XML. Para analisar esses dados, é necessário processá-los em um formato tabular. A seguir, apresentamos um bloco de código contendo uma função que realiza a conversão dos arquivos XML para um formato tabular.

```
process_xml_file <- function(file_path) {
  doc <- read_xml(file_path)

  registros <- xml_find_all(doc, ".//registro")
```

```r
  registros_list <- vector("list", length(registros))

  for (i in seq_along(registros)) {
    ponto_amostral <- ifelse(length(xml_find_all(registros[[i]],
".//ponto_amostral")) > 0,

xml_text(xml_find_all(registros[[i]], ".//ponto_amostral")),
                                                      NA)
    ponto_metro <- ifelse(length(xml_find_all(registros[[i]],
".//ponto_metro")) > 0,

xml_text(xml_find_all(registros[[i]], ".//ponto_metro")),
                                                    NA)
    X <- ifelse(length(xml_find_all(registros[[i]],
".//tipo_forma_vida")) > 0,
                xml_text(xml_find_all(registros[[i]],
".//tipo_forma_vida")),
                NA)
    nativa <- ifelse(length(xml_find_all(registros[[i]],
".//forma_vida_nativa")) > 0,
                         xml_text(xml_find_all(registros[[i]],
".//forma_vida_nativa")),
                         NA)
    seca_morta <- ifelse(length(xml_find_all(registros[[i]],
".//forma_vida_seca_morta")) > 0,
                        xml_text(xml_find_all(registros[[i]],
".//forma_vida_seca_morta")),
                        NA)

    Ocorreram_impacto <- xml_text(xml_find_first(doc,
"//impacto_manejo_uso"))

    Cobertura = xml_text(xml_find_first(doc, "//form_veg"))

    CICLO <- substr(xml_text(xml_find_first(doc, "//data")), 1, 4)

    Plaqueta = xml_text(xml_find_first(doc, "//num_placa"))

    registros_list[[i]] <- data.frame(
      CICLO = CICLO,
      Plaqueta = Plaqueta,
      Cobertura = Cobertura,
      Ocorreram_impacto = Ocorreram_impacto,
      ponto_amostral = as.numeric(ponto_amostral),
      ponto_metro = as.numeric(ponto_metro),
      X = X,
      nativa = nativa,
      seca_morta = seca_morta
    )
```

```
  }

  df <- bind_rows(registros_list)

  df <- df %>%
    mutate(
      Ocorreram_impacto = str_replace_all(Ocorreram_impacto, "não",
"Não"),
      Ocorreram_impacto = str_replace_all(Ocorreram_impacto, "sim",
"Sim"),
      Cobertura = str_replace_all(Cobertura, "campestre", "Campestre"),
      Cobertura = str_replace_all(Cobertura, "savanica", "Savânica")
    )

  df <- df %>%
  separate_rows(seca_morta, sep = " ") %>%
  separate_rows(nativa, sep = " ") %>%
  separate_rows(X, sep = " ") %>%
  mutate(nativa = ifelse(X == "nativa", nativa, ifelse(X == "seca_morta",
NA, NA)),
         seca_morta = ifelse(X == "seca_morta", seca_morta, ifelse(X ==
"nativa", NA, NA))) %>%
  unique()

  return(df)
}
```

Uma sugestão é copiar os arquivos no formato XML do programa ODK de dentro dos dispositivos (celulares ou tablets) e colá-los em uma pasta no computador local onde serão realizadas as análises dos dados.

Outra dica é garantir que essa pasta no computador local contenha apenas os arquivos XML copiados do ODK.

Após copiar a função anterior no RStudio, vamos definir o caminho até a pasta com os arquivos no formato XML. No nosso exemplo, os arquivos estão salvos na pasta chamada “dados_odk”. Substitua o caminho entre aspas pelo local correto da pasta.

Caso não saiba identificar o caminho da pasta, encontre a pasta no explorador de arquivo, clique com o botão direito do mouse na pasta, clique em “Propriedades” e selecione “Local”. Observe que, no ambiente R, as barras do caminho até a pasta dvem estar no sentido “/” e não no “\”.

```
directory_path <- "dados_odk/"
```

Em seguida, utilizamos uma função que identifica todos os arquivos no formato XML na pasta definida. Vale ressaltar que, no caso do programa ODK, cada arquivo XML corresponde a uma unidade amostral.

```
file_paths <- list.files(directory_path, pattern = "\\.xml$", full.names
= TRUE, recursive = TRUE)
```

Em seguida, todos os arquivos XML são processados pela função criada anteriormente (process_xml_file). Os dataframes (arquivos tabulares) resultantes foram combinados em um único dataframe com todas as informações relevantes para análise. Neste exemplo, nomeamos o dataframe como "df.odk".

```
df.odk <- map_df(file_paths, process_xml_file)
```

Como o objetivo deste guia é oferecer uma maneira alternativa de analisar os dados do protocolo Campestre-Savânico do programa MONITORA, algumas informações não foram incluídas nos códigos. Entretanto, posteriormente, caso seja necessário e de interesse de outros gestores ou outras instâncias, os códigos podem ser facilmente ajustados para incluir essas informações.

Se caso a pasta selecionada contiver arquivos XML de mais de uma expedição, o dataframe "df.odk" irá importar todas as expedições, e será fácil distinguí-las pelo ano da expedição na coluna CICLO.

O dataframe contém as seguintes colunas: CICLO (referente ao ano que foi realizada a expedição), Plaqueta (número da plaqueta da unidade amostral), Cobertura (Campestre ou Savânica), Ocorreram_impacto (dicotômica - sim ou não), ponto_amostral (número do ponto amostral dentro da unidade amostral), ponto_metro (distância em metros do ponto amostral dentro da unidade amostral), X (o que tocou na varinha), nativa (forma de vida nativa que tocou na varinha) e seca_morta (forma de vida seca ou morta que tocou na varinha). De acordo com a estrutura do dataframe, se no mesmo ponto amostral mais de um item tocar na vareta, ocorre a repetição da linha do ponto amostral.

Vejamos o exemplo da tabela 01 referente ao resultado da expedição 2023, que ocorreu na REBIO do Guaporé na unidade amostral 1113 (número da plaqueta). No ponto amostral 01 (0 metros), a vareta tocou em "serrapilheira" e "nativa" (coluna X), portanto, existem duas linhas que representam esse ponto amostral. Como "serrapilheira" não representa nenhuma forma de vida, tanto a coluna "Formas_vida", quanto "seca_morta" estão vazias. Por outro lado, a coluna "nativa" está preenchida com "graminoide", na linha onde a coluna "X" está preenchida com "nativa". Se houvesse mais uma planta nativa nesse ponto amostral, haveria mais uma linha representando esse ponto amostral.

CICLO	Plaqueta	Cobertura	Ocorreram_impacto	ponto_amostral	ponto_metro	X	nativa	seca_morta
2023	1113	Campestre	Não	1	0.0	serrapilheira		
2023	1113	Campestre	Não	1	0.0	nativa	graminoide	

CICLO	Plaqueta	Cobertura	Ocorreram_impacto	ponto_amostral	ponto_metro	X	nativa	seca_morta
2023	1113	Campestre	Não	2	0.5	serrapilheira		
2023	1113	Campestre	Não	2	0.5	nativa	graminoide	
2023	1113	Campestre	Não	3	1.0	serrapilheira		
2023	1113	Campestre	Não	3	1.0	nativa	graminoide	
2023	1113	Campestre	Não	4	1.5	serrapilheira		
2023	1113	Campestre	Não	4	1.5	nativa	graminoide	
2023	1113	Campestre	Não	5	2.0	serrapilheira		
2023	1113	Campestre	Não	5	2.0	nativa	graminoide	

Tabela 01: Visualização das dez primeiras linhas do dataframe após processamento da função descrita utilizando a unidade amostral 1113 na expedição 2023 da REBIO do Guaporé.

Arquivos de planilhas (XLSX)

Como supracitado, antes da implementação do ODK na coleta de dados no Programa MONITORA, eram utilizadas planilhas no formato XLSX. Cada planilha representa uma expedição (ciclo) do protocolo, e nessas planilhas, os coletores anotavam as informações de cada unidade amostral em uma aba específica. No bloco de código a seguir, apresentamos uma função que chamamos de **process_xlsx_file** que ajusta os dados a partir do formato XLSX. Essa função é capaz de ler cada aba do arquivo e interpretá-la como se fosse uma unidade amostral.

```
process_xlsx_file <- function(file_path) {

  dfs <- list()

  for (sheet_name in excel_sheets(file_path)) {
    df <- read_excel(file_path, sheet = sheet_name, skip = 4)
    df <- df[1:(nrow(df) -3), -((ncol(df) - 1):ncol(df))]

    df <- df %>%
      rename(ponto_metro = `TRENA (m)`,
             solo_nu = `Solo exposto / rochas`,
             serrapilheira = `Serapilheira / folhiço`,
             graminoide = Gramíneas,
```

```
            arbusto_acima = `Arbustos maior 0,5m`,
            erva_nao_graminoide = Ervas,
            arbusto_abaixo = `Arbustos menor 0,5m`,
            lianas = Cipós,
            arvore_acima = `Árvores c/ diâmetro maior que 5cm`,
            arvore_abaixo = `Árvores c/ diâmetro menor que 5cm`,
            seca_morta = `Plantas secas`,
            exotica = Exóticas
            )

    df$nativa <- apply(df, 1, function(row) {
      nativas <- c("graminoide", "arbusto_abaixo", "lianas",
"erva_nao_graminoide", "arbusto_acima", "arvore_acima", "arvore_abaixo")
      valid_nativas <- nativas[!is.na(row[nativas]) & row[nativas] ==
"X"]
      paste(valid_nativas, collapse = " ")
    })

    df$ponto_metro <- as.numeric(df$ponto_metro)
    df <- df %>% arrange(as.numeric(ponto_metro))
    df$Plaqueta <- gsub("\\D", "", readxl::read_excel(file_path,sheet =
sheet_name, range = "K2:U2", col_names = FALSE)[1,1])
    df$ponto_amostral <- as.integer(as.numeric(df$ponto_metro) / 0.5) + 1
    df$CICLO <- str_extract(gsub("\\D", "", readxl::read_excel(file_path,
sheet = sheet_name, range = "A3:J3", col_names = FALSE)[1,1]), "\\d{4}$")

    df$X <- apply(df, 1, function(row) {
      columns_of_interest <- c("nativa", "solo_nu", "exotica",
"seca_morta", "serrapilheira")
      paste(columns_of_interest[sapply(columns_of_interest, function(col)
!is.na(row[col]) && row[col] != "")], collapse = " ")
    })

    dfs[[sheet_name]] <- df
  }

  df.merged <- bind_rows(dfs, .id = "aba") %>% select(-aba) %>%
select(CICLO, Plaqueta, solo_nu, ponto_amostral, ponto_metro, X, nativa,
seca_morta)

  df.final <- df.merged %>% separate_rows(X, sep = " ")

  df.final <- df.final %>% mutate(nativa = ifelse(X != "nativa", NA,
nativa))

  df.final <- df.final %>% separate_rows(nativa, sep = " ")

  return(df.final)
}
```

As planilhas das expedições 2019 e 2022 da REBIO do Guaporé foram utilizadas como exemplos. Antes de converter os dados, é necessário indicar o caminho das pastas até cada planilha usando o comando a seguir. Diferente do bloco de código que utilizamos para indicar o caminho até os arquivos XML, aqui, precisamos indicar um caminho para cada arquivo XLSX separadamente. Recomenda-se que ajustes sejam feitos para adequar a cada situação.

```
file_path.2019 <- "dados2019/MACS.Vegetação - REBIOGUAPORÉ.2019.xlsx"
file_path.2022 <- "dados2022/MACS.Vegetação -
REBIOGUAPORÉ.2022revisada.xlsx"
```

Em seguida, aplicamos a função **process_xlsx_file** em cada arquivo separadamente para gerar os arquivos tabulares prontos para as análises de dados. No nosso exemplo, nomeamos os dados da expedição de 2019 como "df.xlsx.2019" e os dados da expedição de 2022 como "df.xlsx.2022".

```
df.xlsx.2019 <- process_xlsx_file(file_path.2019)
df.xlsx.2022 <- process_xlsx_file(file_path.2022)
```

Uma informação importante a ser notada é a ausência de indicação da cobertura vegetal (campestre ou savânica) nas planilhas XLSX para cada unidade amostral. Para sanar esse problema, em nosso exemplo, definimos manualmente a cobertura vegetal para cada unidade amostral. No exemplo da REBIO do Guaporé, verificamos as plaquetas de cada unidade amostral dentro do projeto de amostragem da UC para identificar suas respectivas coberturas. Cada gestor terá que consultar o projeto de amostragem de sua UC para definir a cobertura vegetal para cada unidade amostral.

```
plaquetas_campestre <- c("1160", "1120", "1121", "1119", "1164", "1137",
"1144", "1143", "1159", "1118", "1156", "1123", "1158", "1157", "1148",
"1117", "1122", "1124", "1129", "1113", "1116", "1163", "1165", "1135",
"1115", "1154", "1114", "1166")
plaquetas_savanica <- c("1128", "1126", "1105", "1101", "1125", "1104",
"1127", "1132", "1136", "1103", "1133", "1134", "1130", "1131", "1151",
"1153", "1140", "1107", "1141", "1112", "1108", "1142", "1111", "1109",
"1110", "1155", "1152", "1102", "1106")
```

Por fim, adicionamos uma coluna chamada "Cobertura" para indicar a cobertura vegetal das unidades amostrais.

```
df.xlsx.2019 <- df.xlsx.2019 %>%
  mutate(Cobertura = case_when(
    Plaqueta %in% plaquetas_campestre ~ "Campestre",
    Plaqueta %in% plaquetas_savanica ~ "Savânica"))
df.xlsx.2022 <- df.xlsx.2022 %>%
  mutate(Cobertura = case_when(
    Plaqueta %in% plaquetas_campestre ~ "Campestre",
    Plaqueta %in% plaquetas_savanica ~ "Savânica"))
```

Uma etapa adicional que realizamos foi a reordenação das colunas para seguir a mesma ordem dos dados do ODK, permitindo comparações entre os dois tipos de arquivos.

```
df.xlsx.2019 <- df.xlsx.2019 %>%
    select(CICLO, Plaqueta, Cobertura, solo_nu, ponto_amostral,
ponto_metro, X, nativa, seca_morta)
df.xlsx.2022 <- df.xlsx.2022 %>%
    select(CICLO, Plaqueta, Cobertura, solo_nu, ponto_amostral,
ponto_metro, X, nativa, seca_morta)
```

Observando a tabela 02 e tabela 03 a seguir, os dataframes contêm as seguintes colunas: CICLO, Plaqueta, solo_nu (solo exposto), ponto_amostral, ponto_metro, X, nativa e seca_morta. Diferente dos arquivos copiados diretamente do ODK, os arquivos XLSX não possuem identificação de quais unidades amostrais ocorreram impactos.

CICLO	Plaqueta	Cobertura	solo_nu	ponto_amostral	ponto_metro	X	nativa	seca_morta
2019	1101	Savânica		1	0.0	nativa	graminoide	
2019	1101	Savânica		1	0.0	nativa	erva_nao_graminoide	
2019	1101	Savânica		1	0.0	serrapilheira		
2019	1101	Savânica		2	0.5	nativa	graminoide	
2019	1101	Savânica		2	0.5	nativa	lianas	
2019	1101	Savânica		2	0.5	serrapilheira		
2019	1101	Savânica		3	1.0	nativa	graminoide	
2019	1101	Savânica		3	1.0	serrapilheira		
2019	1101	Savânica		4	1.5	nativa	graminoide	
2019	1101	Savânica		4	1.5	serrapilheira		

Tabela 02: Visualização das dez primeiras linhas do dataframe após processamento da função descrita utilizando a unidade amostral 1101 na expedição 2019 da REBIO do Guaporé.

CICLO	Plaqueta	Cobertura	solo_nu	ponto_amostral	ponto_metro	X	nativa	seca_morta
2022	1101	Savânica		1	0.0	nativa	graminoide	
2022	1101	Savânica		2	0.5	nativa	graminoide	
2022	1101	Savânica		2	0.5	serrapilheira		

CICLO	Plaqueta	Cobertura	solo_nu	ponto_amostral	ponto_metro	X	nativa	seca_morta
2022	1101	Savânica		3	1.0	nativa	graminoide	
2022	1101	Savânica		3	1.0	serrapilheira		
2022	1101	Savânica		4	1.5	nativa	graminoide	
2022	1101	Savânica		5	2.0	nativa	graminoide	
2022	1101	Savânica		5	2.0	serrapilheira		
2022	1101	Savânica		6	2.5	nativa	graminoide	
2022	1101	Savânica		7	3.0	nativa	graminoide	

Tabela 03: Visualização das dez primeiras linhas do dataframe após processamento da função descrita utilizando a unidade amostral 1101 na expedição 2022 da REBIO do Guaporé.

Salvando os arquivos

Por fim, recomendamos salvar os dataframes para análises posteriores e compartilhamento dos dados. Os blocos de código seguintes contêm exemplos de como salvar o dataframe usando os dados do ODK e o dataframe utilizando os dados das planilhas no formato XLSX. Utilizamos a função **write.csv** para salvar um arquivo no formato CSV. Nós escolhemos o formato CSV porque é amplamente utilizado em análise de dados e pode ser lido em editores de planilhas, como o Excel.

Você pode escolher o nome que deseja dar para a planilha "ajustada" que será salva no computador; basta modificar o nome que está entre aspas. No entanto, certifique-se de que o nome do arquivo termine com a extensão ".csv". No Windows, geralmente os arquivos salvos são automaticamente alocados na pasta "Documentos" do computador.

```
write.csv(df.odk, "resultado_2023_REBIO_Guapore_ODK.csv", row.names =
FALSE, quote = F)

write.csv(df.xlsx.2022, "resultado_2022_REBIO_Guapore_xlsx.csv",
row.names = FALSE, quote = F)

write.csv(df.xlsx.2019, "resultado_2019_REBIO_Guapore_xlsx.csv",
row.names = FALSE, quote = F)
```

Visualização dos dados

Após a importação dos arquivos, o processamento dos dados e seus respectivos salvamentos, propomos algumas análises utilizando a linguagem R nos resultados prévios desses dados. Destacamos que, após as etapas anteriores, já é possível realizar algumas análises em outros programas, como o Excel. No entanto, para especialistas em análises de dados, a linguagem R é uma ferramenta robusta que oferece diversas

análises que seriam altamente complexas ou impossíveis de realizar em programas como o Excel.

Por isso, aproveitamos este guia para adicionar exemplos de blocos de código com algumas análises e visualizações dos dados do protocolo Campestre-Savânico que consideramos relevantes. Utilizamos os dataframes processados nas etapas anteriores, coletados nas expedições da REBIO do Guaporé, como exemplos.

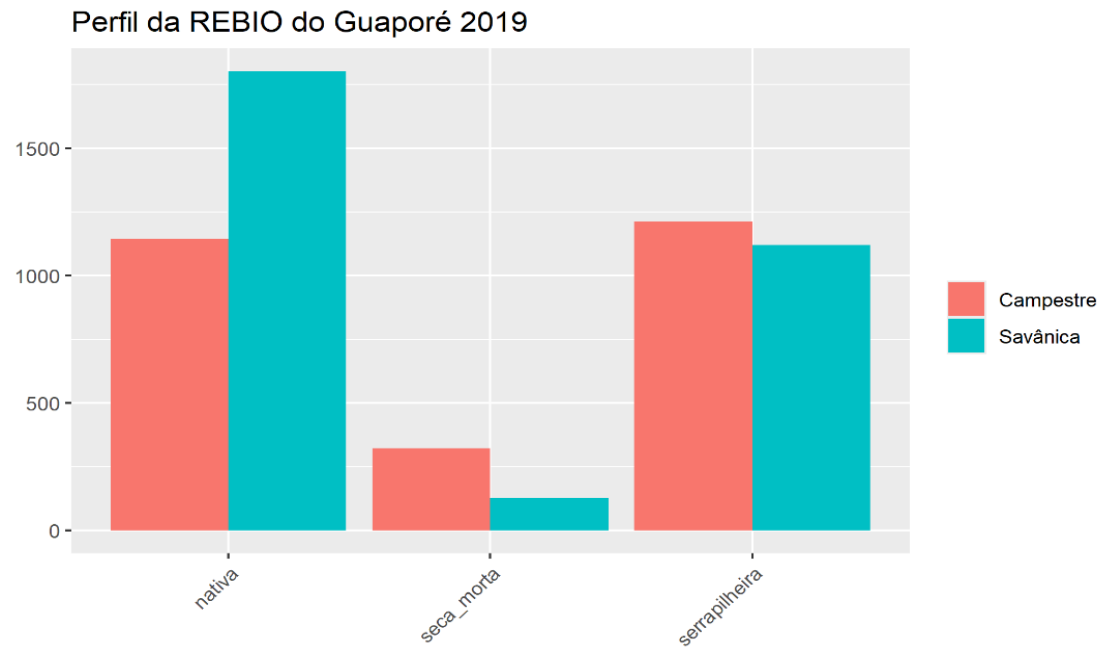
Em capítulo posterior, ensinaremos como salvar os gráficos em alta definição.

Perfil da cobertura da vegetação para uma única expedição

A seguir, apresentamos alguns blocos de código em R que utilizamos para analisar os diferentes perfis da cobertura da vegetação em uma única expedição, juntamente com os gráficos resultantes de cada código para ilustração. Utilizamos o dataframe "df.xlsx.2019" (expedição de 2019) como exemplo. No entanto, qualquer dataframe processado através das etapas anteriores pode ser utilizado, bastando substituir "df.xlsx.2019" pelo nome do dataframe desejado.

Na análise a seguir, as barras representam o quantitativo de formas de vida que tocaram na vareta (coluna X do dataframe) para as duas coberturas vegetais.

```
ggplot(df.xlsx.2019 %>%
         group_by(Cobertura, X) %>%
         summarise(count = n(), .groups = 'drop'),
       aes(x = X, y = count, fill = Cobertura)) +
  geom_bar(stat = "identity", position = "dodge") +
  labs(fill = "",
       title = "Perfil da REBIO do Guaporé 2019", x = "", y = "") +
  theme(axis.text.x = element_text(angle = 45, hjust = 1))
```

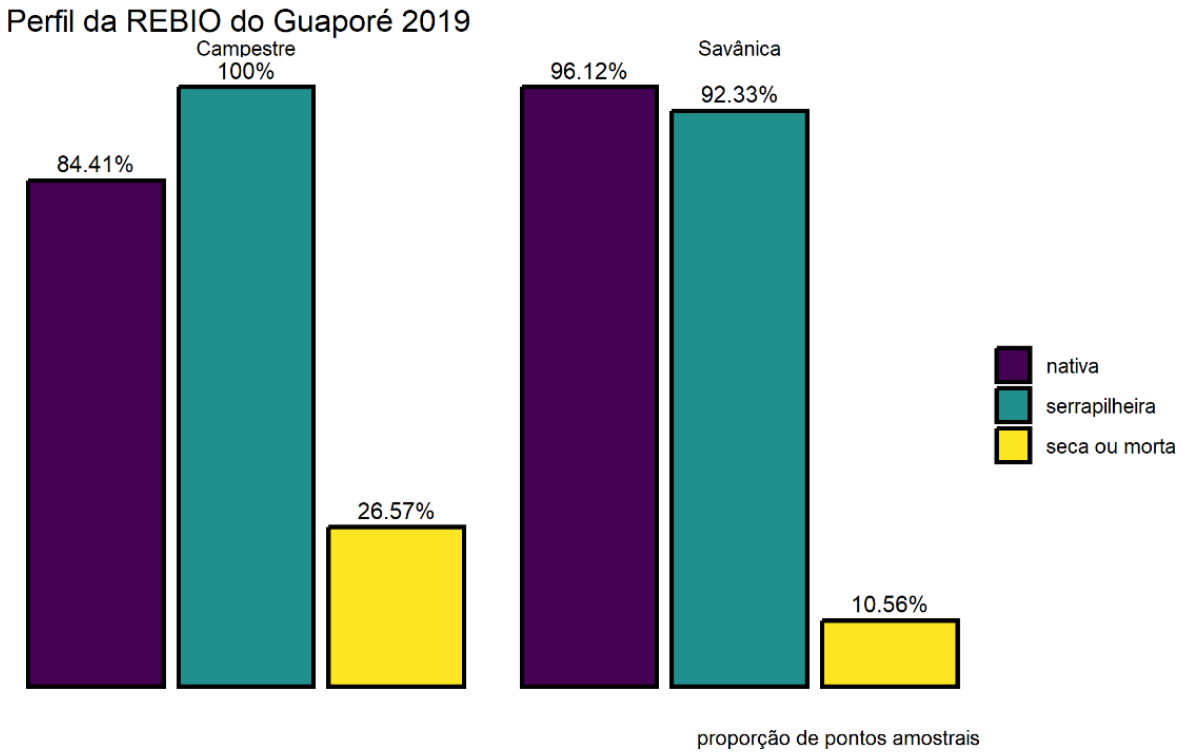


No exemplo da expedição de 2019 na REBIO do Guaporé, observamos que foram detectadas mais plantas nativas na cobertura savânica do que na cobertura campestre. No entanto, a cobertura campestre apresentou mais formas de vida seca ou morta, enquanto as duas coberturas apresentaram números bem próximos de serrapilheiras.

Vale esclarecer que, na análise acima, nos pontos amostrais onde são identificadas mais de uma forma de vida, é contabilizado o número total de formas de vida que tocam na vareta. Ou seja, se em um dado ponto amostral duas formas de vida nativa diferentes tocam na vareta, ambas são contabilizadas no gráfico. Portanto, na prática, é totalmente plausível e esperado que o número de formas de vida nativas seja maior que o número total de pontos amostrais.

Aproveitando que estamos trabalhando com o R, apresentamos uma outra versão da análise anterior, com um bloco de código mais complexo e detalhado.

```
ggplot(merge(df.xlsx.2019 %>%
               mutate(X = factor(X, levels = df.xlsx.2019 %>%
filter(Cobertura == "Savânica") %>% count(X) %>% arrange(desc(n)) %>%
pull(X))) %>%
               group_by(Cobertura, X) %>%
               summarise(n = n_distinct(Plaqueta, ponto_amostral, X)),
             df.xlsx.2019 %>%
               mutate(X = factor(X, levels = df.xlsx.2019 %>%
filter(Cobertura == "Savânica") %>% count(X) %>% arrange(desc(n)) %>%
pull(X))) %>%
               group_by(Cobertura, Plaqueta) %>%
               summarise(pontos = n_distinct(ponto_amostral)) %>%
               group_by(Cobertura) %>%
               summarise(total = sum(pontos)),
             by = "Cobertura") %>%
         mutate(proporcao = n / total * 100),
       aes(x = X, y = n, fill = X)) +
  geom_bar(stat = "identity", position = "dodge", linewidth = 1, size =
1, color = "black") +
  facet_wrap(~ Cobertura, scales = "free") +
  geom_text(aes(label = paste0(round(proporcao, 2), "%"),
                vjust = -0.5), size = 3.5) +
  scale_fill_viridis_d(option = "viridis",
                       labels = c("seca_morta" = "seca ou morta",
                                  "solo_nu" = "solo exposto",
                                  "exotica" = "exótica")) +
  labs(fill = "",
       title = "Perfil da REBIO do Guaporé 2019", x = "", y = "",
       caption = "proporção de pontos amostrais") +
  theme_void()
```



No gráfico acima, as barras representam as proporções de pontos amostrais da cobertura vegetal que possuem a respectiva forma de vida que tocou na vareta. As barras estão agrupadas de acordo com a cobertura vegetal: campestre à esquerda e savânica à direita. As barras da cobertura savânica estão obrigatoriamente em ordem decrescente, e as barras da cobertura campestre seguem a mesma ordem da outra cobertura. Além disso, foi utilizada uma paleta de cores específica.

Através do gráfico elaborado acima é fácil tirar algumas conclusões. No exemplo da expedição de 2019 na REBIO do Guaporé, observamos que 100% dos pontos amostrais da cobertura campestre e 92% da cobertura savânica contabilizaram serrapilheira, enquanto 84% e 96% dos pontos amostrais apresentaram formas de vida nativas nas coberturas campestre e savânica, respectivamente. Notamos que 26,57% dos pontos amostrais da cobertura campestre apresentaram formas de vida seca ou morta, enquanto na cobertura savânica esse percentual foi de 10,56%.

Em relação ao código que gerou o gráfico acima, vamos apresentar brevemente a funcionalidade de cada função utilizadas. A função **ggplot** é responsável pela elaboração do gráfico em si. A função **merge** combina dois dataframes modificados derivados do dataframe original (df.xlsx.2019). Para transformar esses dados, usamos **mutate** para modificar a coluna X em um tipo especial chamado “fator”, que é uma maneira ordenada de categorizar dados. Para isso, usamos a função **fator**, e os níveis desse fator são determinados após filtrar o dataframe com a função **filter** para incluir apenas dados odne a coluna Cobertura é “Savânica”. Contamos as ocorrências de cada valor na coluna X pela função **count** e ordenamos esses valores em ordem decrescente pela função **arrange**. Em seguida, agrupamos os dados através da função **group_by** pelas colunas Cobertura e X, ou pelas colunas Cobertura e Plaqueta, e depois

resumimos esses grupos para obter o número de combinações distintas que nomeamos de "n" (função **n_distinct**). Isso nos dá contagens de pontos amostrais únicos. Depois, juntamos os dois conjuntos de dados resultantes em um único dataframe usando a função **merge**, comentada anteriormente. Calculamos a proporção desses pontos em relação ao total usando novamente a função **mutate**. No gráfico, **aes** define como os dados são mapeados para os elementos visuais, especificando que a coluna X está no eixo x, os dados "n" no eixo y e a coluna X também define a cor de preenchimento das barras. A função **geom_bar** adiciona barras ao gráfico, configurando-as para mostrar valores exatos através do parâmetro **stat = "identity"** e ajustando sua posição, largura e cor. Para mostrar cada tipo de cobertura separadamente, usamos a função **facet_wrap** para criar subgráficos para cada cobertura vegetal. Adicionamos textos às barras usando **geom_text**, que mostra a proporção de cada barra acima dela. Para as cores das barras, usamos **scale_fill_viridis_d** para aplicar uma paleta de cores específica e renomear alguns valores para torná-los mais compreensíveis. Finalmente, **labs** define os títulos e rótulos do gráfico, e **theme_void** remove elementos visuais padrão, como linhas de grade e eixos, deixando um fundo limpo.

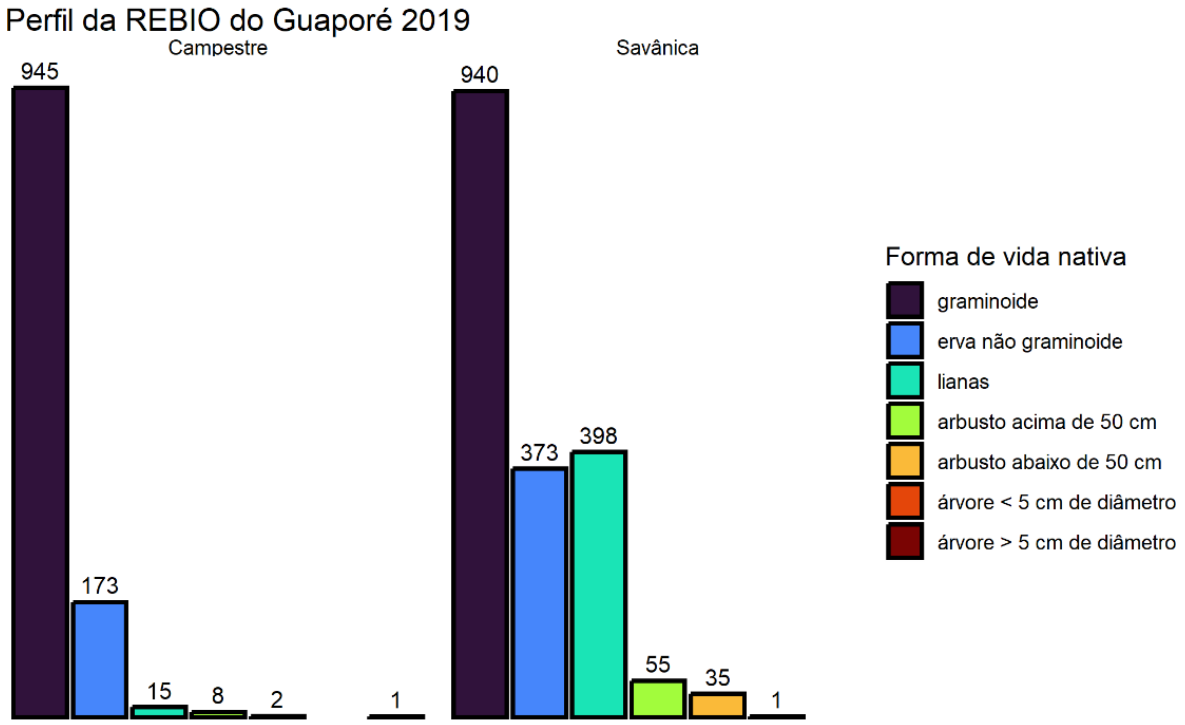
Essa série de transformações e funções juntas criaram o gráfico de barras personalizado acima. Vale destacar que existem diversas outras modificações disponíveis que não adotamos. A escolha das funções foi totalmente subjetiva e várias funções podem ser acrescentadas, modificadas ou removidas. Por exemplo, a função **arrange** que ordenou os gráficos em ordem decrescente é estético, não alterando a interpretação dos resultados, porém foi adicionada ao código para facilitar a interpretar visual do gráfico. Neste guia, não pretendemos ensinar todas as funções do R na elaboração de gráficos e análises de dados. Apenas aproveitamos a oportunidade para demonstrar a robustez e o leque de possibilidades que a linguagem R oferece.

A seguir, apresentamos outra análise do perfil da cobertura da vegetação, observando as formas de vida nativas em cada cobertura vegetal. Antes de elaborar o gráfico com a função **ggplot**, nós criamos um novo dataframe chamado "df.xlsx.2019.nativas", filtrando apenas as formas de vida nativas (função **filter**), e depois escolhemos manualmente a ordem da cobertura da vegetação no gráfico (campestre à esquerda), e selecionamos a ordem das barras de forma decrescente, através da função **factor**.

```
df.xlsx.2019.nativas <- df.xlsx.2019 %>%
  filter(X == "nativa")
df.xlsx.2019.nativas$Cobertura <- factor(df.xlsx.2019.nativas$Cobertura,
levels = c("Campestre", "Savânica"))
df.xlsx.2019.nativas$nativa <- factor(df.xlsx.2019.nativas$nativa,
       levels = names(sort(table(df.xlsx.2019.nativas$nativa), decreasing
= TRUE)))
```

Após as modificações acima, nós elaboramos o gráfico a seguir. Destacamos que as cores das barras foram selecionadas manualmente, adotando a função **scale_fill_manual**.

```
ggplot(df.xlsx.2019.nativas,
       aes(x = nativa, y = ..count.., fill = nativa)) +
  geom_bar(linewidth = 1, size = 1, color = "black") +
  scale_fill_viridis_d(option = "H",
                       labels = c("erva_nao_graminoide" = "erva não
graminoide", "arbusto_acima" = "arbusto acima de 50 cm",
                                  "arbusto_abaixo" = "arbusto abaixo de
50 cm","arvore_acima" = "árvore > 5 cm de diâmetro",
                                  "arvore_abaixo" = "árvore < 5 cm de
diâmetro")) +
  labs(title = "Perfil da REBIO do Guaporé 2019", fill = "Forma de vida
nativa" ,x = "", y = "") +
  theme_void() +
  geom_text(stat = "count", aes(label = ..count..), vjust = -0.5, size =
3.5) +
  facet_wrap(~ Cobertura, drop = TRUE, ncol = 2)
```



Uma outra análise que propomos é a observação da proporção de formas de vidas nativas herbáceas e lenhosas para cada cobertura vegetal. Primeiro, preparamos um novo dataframe, chamado "df.xlsx.2019.categoria", listando quais são as formas de vida nativas que são lenhosas ou herbáceas. Indicamos arbustos e árvores como formas de vida lenhosas, e todas as outras como herbáceas.

Na expedição na REBIO do Guaporé em 2019, não foram identificadas outras formas de vidas lenhosas além de arbustos e árvores, como palmeiras, por exemplo. Portanto, não incluímos essa última forma de vida nativa no nosso exemplo. Vale ressaltar que

outras formas de vidas lenhosas podem ser adicionadas no código, bastando colocá-las entre aspas logo após “árvore_abaixo”.

```
df.xlsx.2019.categoria <- df.xlsx.2019 %>%
  filter(X == "nativa")
df.xlsx.2019.categoria$Cobertura <-
factor(df.xlsx.2019.categoria$Cobertura, levels = c("Campestre",
"Savânica"))
df.xlsx.2019.categoria <- df.xlsx.2019.categoria %>%
  mutate(Tipo = if_else(nativa %in% c("arbusto_acima", "arbusto_abaixo",
"arvore_acima", "arvore_abaixo"),"Lenhosa", "Herbácea")) %>%
  group_by(CICLO, Cobertura, Tipo) %>%
  summarise(Contagem = n()) %>%
  ungroup()
df.xlsx.2019.categoria$Tipo <- factor(df.xlsx.2019.categoria$Tipo, levels
= c("Lenhosa", "Herbácea"))
```

Em seguida, criamos um dataframe calculando a proporção de formas de vida lenhosas e herbáceas para cada cobertura. Foram contabilizadas todas as formas de vidas nativas que tocaram na vareta durante a expedição.

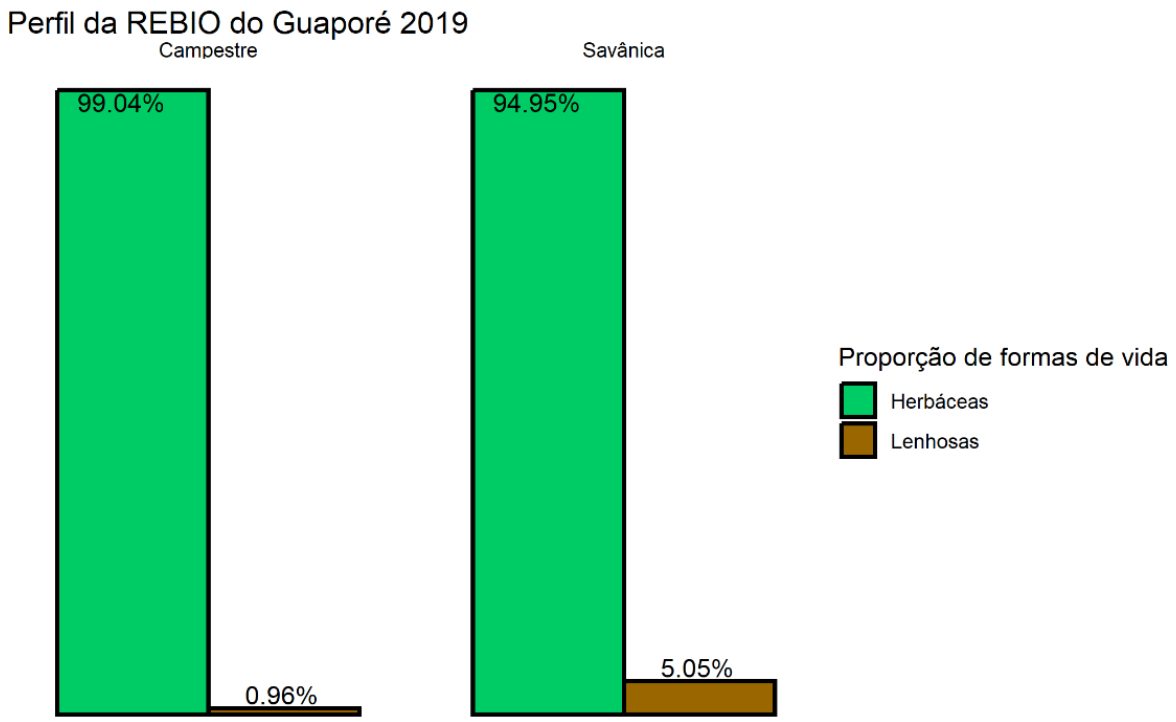
```
df.xlsx.2019.prop <- df.xlsx.2019.categoria %>%
  group_by(Cobertura) %>%
  summarize(prop_lenhosas = sum(Contagem[Tipo == "Lenhosa"]) /
sum(Contagem) * 100,
            prop_herbaceas = sum(Contagem[Tipo == "Herbácea"]) /
sum(Contagem) * 100)
df.xlsx.2019.categoria <- left_join(df.xlsx.2019.categoria,
df.xlsx.2019.prop, by = c("Cobertura"))
```

Para uma melhor visualização, calculamos a normalização dos valores, para que as barras ficassem proporcionais no gráfico. Além disso, escolhemos manualmente as cores das barras através da função **scale_fill_manual**.

```
df.xlsx.2019.prop <- pivot_longer(df.xlsx.2019.prop, cols =
c(prop_lenhosas, prop_herbaceas),
                          names_to = "Tipo_Vegetacao", values_to =
"Proporcao")

ggplot(df.xlsx.2019.prop, aes(x = Cobertura, y = Proporcao, fill =
Tipo_Vegetacao)) +
  geom_bar(stat = "identity", position = "dodge", linewidth = 1, size =
1, color = "black") +
  scale_fill_manual(values = c("prop_lenhosas" = "#996600",
"prop_herbaceas" = "#00CC66"),
                    name = "Proporção de formas de vida",
                    labels = c("prop_lenhosas" = "Lenhosas",
                               "prop_herbaceas" = "Herbáceas")) +
  geom_text(data = df.xlsx.2019.prop[df.xlsx.2019.prop$Tipo_Vegetacao ==
"prop_lenhosas", ],
            aes(label = paste0(round(Proporcao, digits = 2), "%"), vjust
```

```
= -0.25, hjust = -0.5)) +
  geom_text(data = df.xlsx.2019.prop[df.xlsx.2019.prop$Tipo_Vegetacao ==
"prop_herbaceas", ],
            aes(label = paste0(round(Proporcao, digits = 2), "%"), vjust
= 1.15, hjust = 1.5)) +
  theme_void() +
  labs(title = "Perfil da REBIO do Guaporé 2019") +
  facet_wrap(~Cobertura, scales = "free", ncol = 2)
```



Observamos que, na expedição 2019, foram identificados 99,04% de formas de vida nativas herbáceas e 0,96% de lenhosas para a cobertura campestre, enquanto 94,95% de herbáceas e 5,05% de lenhosas definiram o perfil da cobertura savânica.

Perfil da cobertura da vegetação para mais de uma expedição

A próxima sequência de blocos de código propõe as mesmas análises de perfis da cobertura vegetal descritas acima, porém permitindo analisar mais de uma expedição. Utilizamos as expedições 2022 e 2023 da REBIO do Guaporé como exemplos, utilizando um dataframe oriundo das planilhas no formato XLSX e outro dataframe do *software* ODK.

Assim como na análise de apenas uma expedição, é necessário seguir os primeiros passos deste guia: importar os dados e processá-los pelas respectivas funções (**process_xml_file** ou **process_xlsx_file**). Após essas etapas preliminares, é interesse juntar os dataframes em apenas um único dataframe, que batizamos de "df.UC". Adotamos a função **bind_rows**, que anexa um dataframe embaixo do outro. Entretanto, há diferenças entre as estruturas dos dataframes oriundos das planilhas

XLSX e do programa ODK, como, por exemplo, a ausência da coluna "Ocorreram impactos" na planilha XLSX. Portanto, selecionamos as colunas em comum entre os dois tipos de dados, através da função **select**.

```
df.UC <- bind_rows(df.odk %>% select(c(CICLO, Plaqueta, Cobertura,
ponto_amostral, ponto_metro, X, nativa, seca_morta)),
                   df.xlsx.2022 %>% select(c(CICLO, Plaqueta, Cobertura,
ponto_amostral, ponto_metro, X, nativa, seca_morta)))
```

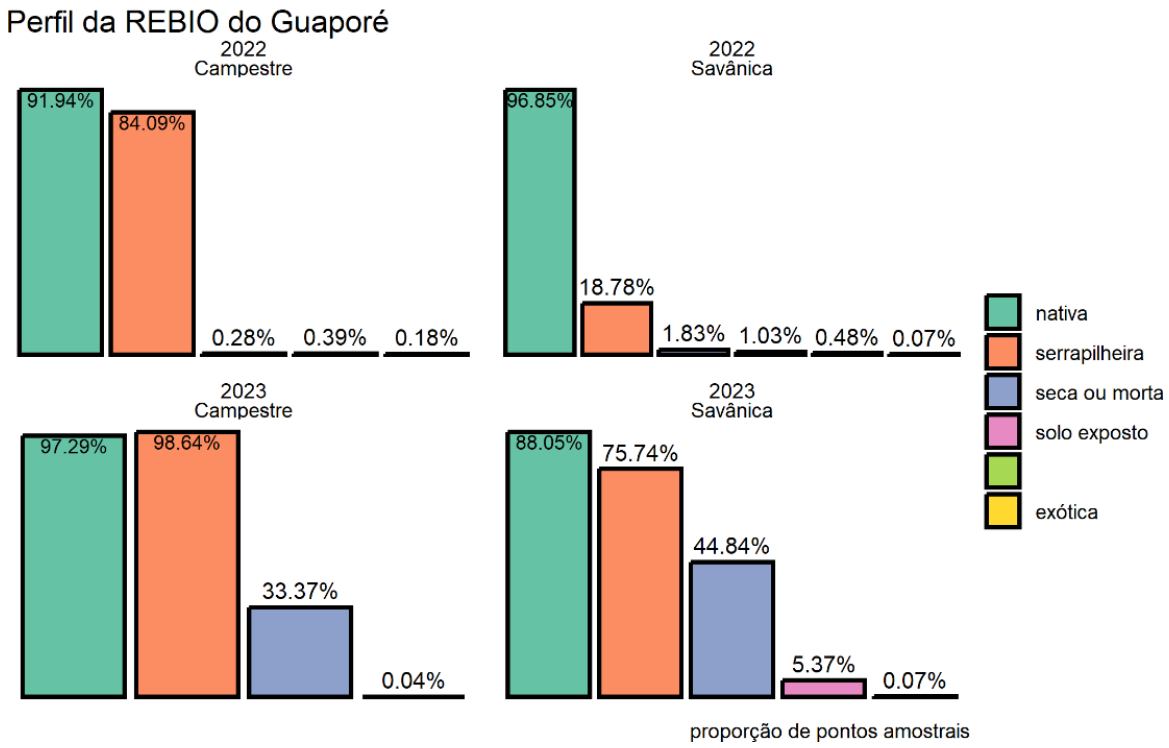
Em nossa primeira análise com diferentes expedições, a partir do dataframe "df.UC", criamos um novo dataframe chamado "df.UC.cober". Nesse novo dataframe, calculamos o número de pontos amostrais que detectaram cada elemento que tocou na vareta e a sua proporção dentro da cobertura vegetal da respectiva expedição.

```
df.UC.cober <- merge(df.UC %>%
                       mutate(X = factor(X, levels = df.UC %>% count(X)
%>% arrange(desc(n)) %>% pull(X))) %>%
                       group_by(CICLO, Cobertura, X) %>%
                       summarise(n = n_distinct(Plaqueta, ponto_amostral,
X)),
                     df.UC %>%
                       mutate(X = factor(X, levels = df.UC %>% count(X)
%>% arrange(desc(n)) %>% pull(X))) %>%
                       group_by(CICLO, Cobertura, Plaqueta) %>%
                       summarise(pontos = n_distinct(ponto_amostral)) %>%
                       group_by(CICLO, Cobertura) %>%
                       summarise(total = sum(pontos)),
                     by = c("CICLO","Cobertura")) %>%
  mutate(proporcao = n / total * 100)
```

Apresentamos o código para elaboração do gráfico. Por uma questão estética da própria função **ggplot**, nos casos em que as proporções eram superiores ou iguais a 80%, deslocamos os valores para dentro das respectivas barras para ficarem visíveis. Na função **facet_wrap**, selecionamos dois subgráficos por linha (**ncol = 2**). Além disso, adotamos outra função para gerar cores, o **scale_fill_brewer**.

```
ggplot(df.UC.cober,
       aes(x = X, y = n, fill = X)) +
  geom_bar(stat = "identity", position = "dodge", linewidth = 1, size =
1, color = "black") +
  facet_wrap(~ CICLO + Cobertura, ncol = 2, scales = "free") +
  geom_text(data = df.UC.cober[df.UC.cober$proporcao >= 80, ],
            aes(label = paste0(round(proporcao, 2), "%"),
                vjust = 1.15), size = 2.5) +
  geom_text(data = df.UC.cober[df.UC.cober$proporcao < 80, ],
            aes(label = paste0(round(proporcao, 2), "%"),
                vjust = -0.5), size = 2.5) +
  scale_fill_brewer(palette = "Set2",
                    labels = c("seca_morta" = "seca ou morta",
                               "solo_nu" = "solo exposto",
```

```
                                "exotica" = "exótica")) +
  labs(fill = "",
       title = "Perfil da REBIO do Guaporé", x = "", y = "",
       caption = "proporção de pontos amostrais") +
  theme_void()
```

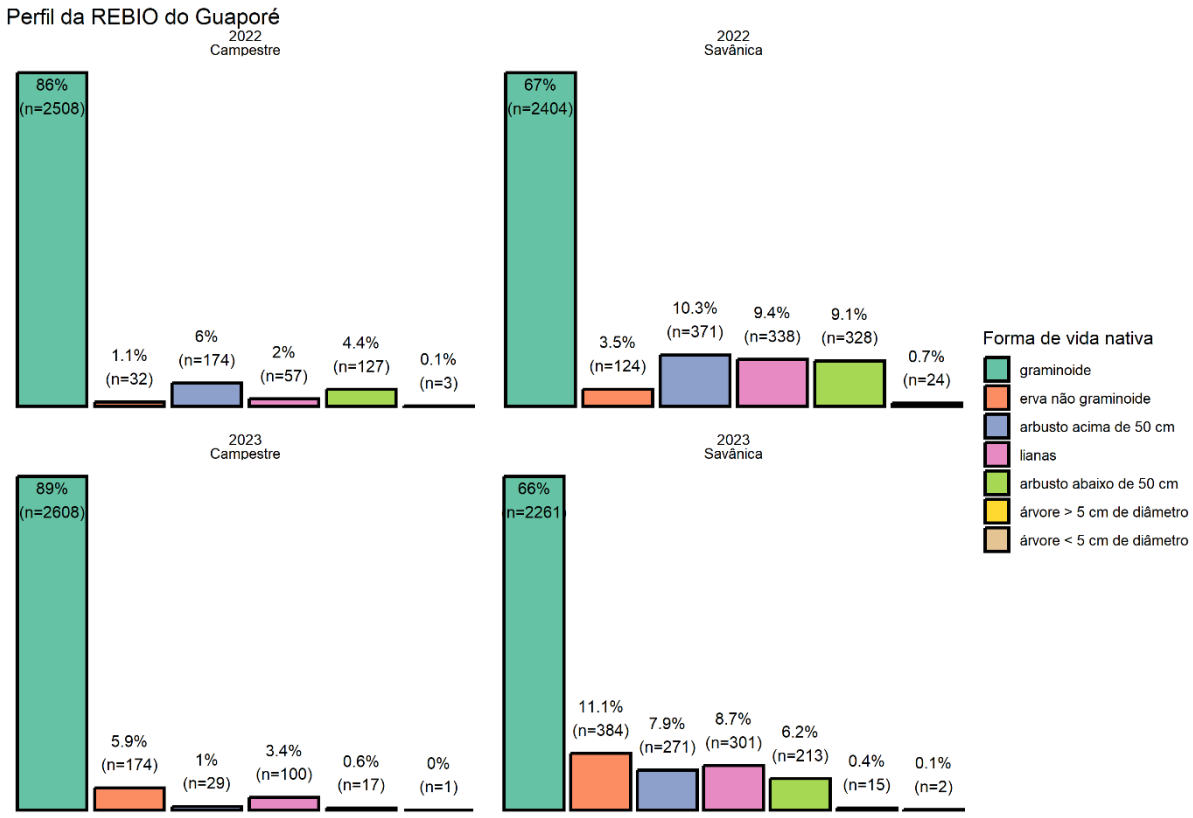


Em seguida, criamos um novo dataframe chamado "df.UC.nativas" para realizar a análise dos perfis das formas de vidas nativas.

```
df.UC.nativas <- df.UC %>%
  filter(X == "nativa") %>%
  group_by(CICLO, Cobertura, nativa) %>%
  summarise(Contagem = n(), .groups = 'drop') %>%
  group_by(CICLO, Cobertura) %>%
  mutate(Proporcao = Contagem / sum(Contagem) * 100) %>%
  ungroup()
df.UC.nativas$Cobertura <- factor(df.UC.nativas$Cobertura, levels =
c("Campestre", "Savânica"))
df.UC.nativas$nativa <- factor(df.UC.nativas$nativa,
                               levels =
unique(df.UC.nativas$nativa[order(df.UC.nativas$Proporcao, decreasing =
TRUE)]))
```

O gráfico foi elaborado seguindo a mesma estratégia da análise de apenas uma expedição. É importante ressaltar que, devido aos valores e ao número de barras, pode ser necessário ajustar as posições e os tamanhos dos textos das proporções utilizando as funções **vjust**, **hjust** e **size**.

```
ggplot(df.UC.nativas, aes(x = nativa, y = Proporcao, fill = nativa)) +
  geom_bar(linewidth = 1, stat = "identity", position = "dodge", size =
1, color = "black") +
  facet_wrap(~ CICLO + Cobertura, ncol = 2, scales = "free") +
  geom_text(data = df.UC.nativas[df.UC.nativas$Proporcao >= 50, ],
            aes(label = paste0(round(Proporcao, 0), "%"),
                vjust = 1.15), size = 3.5) +
  geom_text(data = df.UC.nativas[df.UC.nativas$Proporcao < 50, ],
            aes(label = paste0(round(Proporcao, 1), "%"),
                vjust = -0.5), size = 3.5) +
  scale_fill_brewer(palette = "Set2",
                    labels = c("erva_nao_graminoide" = "erva não
graminoide", "arbusto_acima" = "arbusto acima de 50 cm",
                               "arbusto_abaixo" = "arbusto abaixo de 50
cm","arvore_acima" = "árvore > 5 cm de diâmetro",
                               "arvore_abaixo" = "árvore < 5 cm de
diâmetro")) +
  labs(title = "Perfil da REBIO do Guaporé", fill = "Forma de vida
nativa", x = "", y = "") +
    theme_void()
```



Por fim, propomos analisar o perfil das formações herbáceas e lenhosas de cada cobertura vegetal em cada expedição. Primeiramente, criamos um novo dataframe chamado “df.UC.categoria”, que lista todos as formas de vidas nativas como herbáceas ou lenhosas.

```
df.UC.categoria <- df.UC %>%
  filter(X == "nativa")
df.UC.categoria$Cobertura <- factor(df.UC.categoria$Cobertura, levels =
c("Campestre", "Savânica"))
df.UC.categoria <- df.UC.categoria %>%
  mutate(Tipo = if_else(nativa %in% c("arbusto_acima", "arbusto_abaixo",
"arvore_acima", "arvore_abaixo"),"Lenhosa", "Herbácea")) %>%
  group_by(CICLO, Cobertura, Tipo) %>%
  summarise(Contagem = n()) %>%
  ungroup()
df.UC.categoria$Tipo <- factor(df.UC.categoria$Tipo, levels =
c("Lenhosa", "Herbácea"))
```

Posteriormente, criamos um dataframe chamado “df.UC.prop” que calcula a proporção de cada cobertura vegetal dentro da sua respectiva expedição.

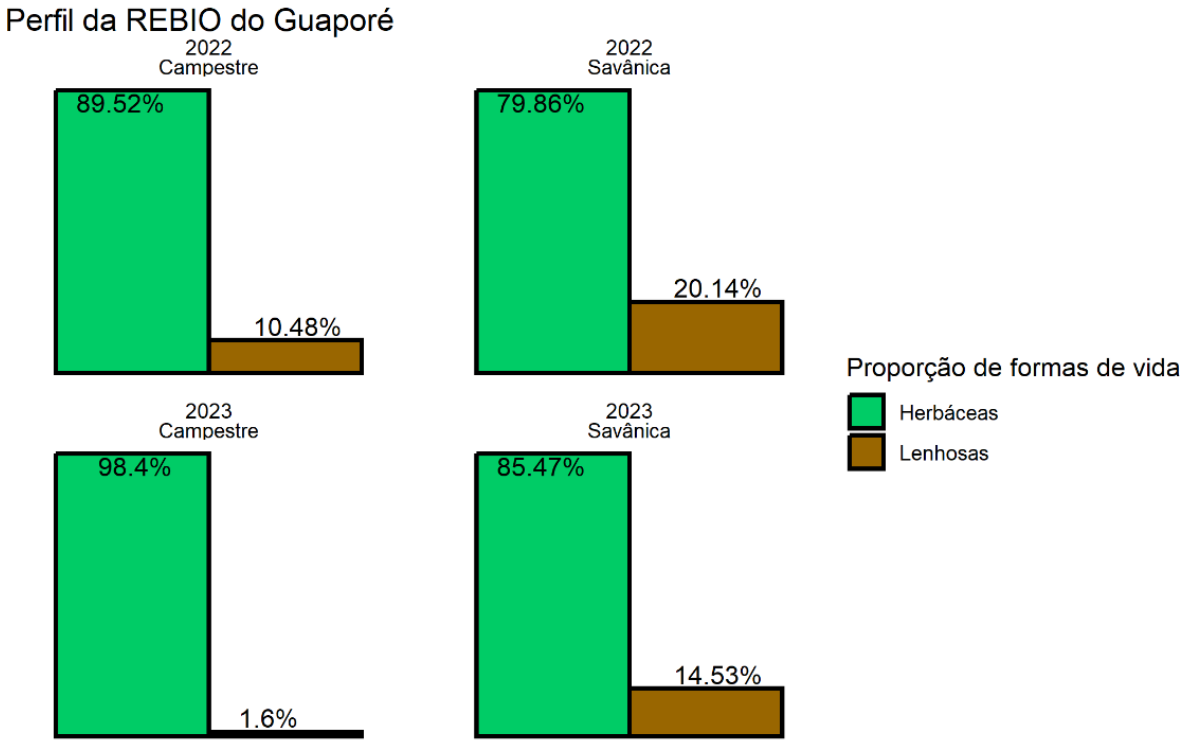
```
df.UC.prop <- df.UC.categoria %>%
  group_by(CICLO, Cobertura) %>%
  summarize(prop_lenhosas = sum(Contagem[Tipo == "Lenhosa"]) /
sum(Contagem) * 100,
            prop_herbaceas = sum(Contagem[Tipo == "Herbácea"]) /
sum(Contagem) * 100)
df.UC.categoria <- left_join(df.UC.categoria, df.UC.prop, by = c("CICLO",
"Cobertura"))


df.UC.prop <- pivot_longer(df.UC.prop, cols = c(prop_lenhosas,
prop_herbaceas),
                                  names_to = "Tipo_Vegetacao", values_to
= "Proporcao")
```

E, finalmente, elaboramos o gráfico.

```
ggplot(df.UC.prop, aes(x = Cobertura, y = Proporcao, fill =
Tipo_Vegetacao)) +
  geom_bar(stat = "identity", position = "dodge", linewidth = 1, size =
1, color = "black") +
  scale_fill_manual(values = c("prop_lenhosas" = "#996600",
"prop_herbaceas" = "#00CC66"),
                    name = "Proporção de formas de vida",
                    labels = c("prop_lenhosas" = "Lenhosas",
                               "prop_herbaceas" = "Herbáceas")) +
  geom_text(data = df.UC.prop[df.UC.prop$Tipo_Vegetacao ==
"prop_lenhosas", ],
            aes(label = paste0(round(Proporcao, digits = 2), "%"), vjust
= -0.25, hjust = -0.1)) +
  geom_text(data = df.UC.prop[df.UC.prop$Tipo_Vegetacao ==
"prop_herbaceas", ],
            aes(label = paste0(round(Proporcao, digits = 2), "%"), vjust
= 1.15, hjust = 1)) +
  theme_void() +
```

```
  labs(title = "Perfil da REBIO do Guaporé") +
  facet_wrap(~ CICLO + Cobertura, scales = "free", ncol = 2)
```



Nesse gráfico, podemos observar uma discrepância nas formas de vidas herbáceas e campestre nas duas coberturas vegetais. Entretanto, na cobertura campestre, houve uma variação entre ciclos: 10% de lenhosas no ano de 2022 e 1,6% de lenhosas no ano de 2023.

A partir da análise desses dados, é possível levantar hipóteses e começar observar a dinâmica e ciclagem da biomassa. Esses resultados podem auxiliar os gestores na tomada de decisões sobre as políticas adotadas na UC. Além disso, esses resultados são importantes para os gestores, permitindo que sejam apresentados à sociedade e aos comunitários das UC.

Por fim, a análise desses dados ressaltou a importância da coleta contínua de informações, destacando as vantagens do uso do ODK e sua facilidade.

Salvando gráficos em alta definição

Atualmente, existem várias formas de elaborar e salvar os gráficos usando a linguagem R. Vamos apresentar um exemplo de salvamento no formato PNG com 300 dpi (alta definição). Aqui, para fins de ilustração, iremos salvar um dos gráficos gerados neste guia. Como demonstrado no código a seguir, primeiramente é necessário nomear o gráfico. No nosso exemplo, o nomeamos de “perfil” usando a setinha “<-” antes da função **ggplot**.

```
perfil <- ggplot(df.xlsx.2019.prop, aes(x = Cobertura, y = Proporcao,
fill = Tipo_Vegetacao)) +
  geom_bar(stat = "identity", position = "dodge", linewidth = 1, size =
1, color = "black") +
  scale_fill_manual(values = c("prop_lenhosas" = "#996600",
"prop_herbaceas" = "#00CC66"),
                    name = "Proporção de formas de vida",
                    labels = c("prop_lenhosas" = "Lenhosas",
                               "prop_herbaceas" = "Herbáceas")) +
  geom_text(data = df.xlsx.2019.prop[df.xlsx.2019.prop$Tipo_Vegetacao ==
"prop_lenhosas", ],
            aes(label = paste0(round(Proporcao, digits = 2), "%"), vjust
= -0.25, hjust = -0.5)) +
  geom_text(data = df.xlsx.2019.prop[df.xlsx.2019.prop$Tipo_Vegetacao ==
"prop_herbaceas", ],
            aes(label = paste0(round(Proporcao, digits = 2), "%"), vjust
= 1.15, hjust = 1.5)) +
  theme_void() +
  labs(title = "Perfil da REBIO do Guaporé 2019") +
  facet_wrap(~Cobertura, scales = "free", ncol = 2)
```

Em seguida, usamos a função **ggsave** para salvar o gráfico no formato PNG. O nome do arquivo PNG pode ser alterado dentro das aspas. Repare que é necessário adicionar a extensão ".png" ao final do nome para garantir que o arquivo seja salvo no formato PNG. Além disso, é importante colocar o nome do gráfico (no nosso exemplo, "perfil") no parâmetro **plot**.

```
ggsave("rebio.png", plot = perfil, dpi = 300)
```

Perspectivas

Como mencionado em outros capítulos, o intuito deste guia não é substituir o SISMonitora, mas sim auxiliar os gestores a analisarem os dados preliminares coletados no protocolo Campestre-Savânico do programa MONITORA.

Após a elaboração deste guia, traçamos algumas perspectivas. A primeira perspectiva é analisar os dados após a sua validação, o que permitirá uma maior confiabilidade nos resultados.

Caso haja interesse da COMOB e dos Centros de Pesquisa, a segunda perspectiva é a implementação das funções descritas neste guia prático dentro do SISMonitora, principalmente na conversão dos formatos dos arquivos e nas análises dos dados.

E por fim, a terceira perspectiva é auxiliar os gestores de UC nas análises de outros protocolos do programa MONITORA, caso haja interesse da COMOB e dos Centros de Pesquisa, especialmente aqueles que lidam com grandes volumes de dados e que utilizam o *software* ODK nas coletas.