BMP Dashboard Manual

Ané Cloete

2023-10-12

Contents

Chapter 1

Hi :)

Everything you need to know to takeover the maintaining and building of the BMP dashboard is in this bookdown. The first chapter will outline the R project folder structure and contents and then in Chapter two I'll talk about how to get the biodiversity data form OneDrive and prepare the data for the dashboard. The remaining chapters will delve into the code itself.

Before starting, here is a short list of handy resources:

- R
- Shiny # 1
- Shiny # 2
- Github
- Bookdown (if you want to know how to make something like this).
- HTML and CSS

These are big and comprehensive resources that will aid you along the way. But remember google and stackoverflow are your best friends! And ChatGPT might become your closest and most frustrating colleague.

1.1 Owners

- Ané Cloete (2023)

If you are the new dashboard manager, add your name as well! All you need to do is open the bookdown project and navigate to the “index.Rmd” file and scroll down till you see this section. In the next chapter you’ll see where this folder is located.

Remember the dashboard is work in progress, so feel free to change and modify it as you like and most importantly have fun with it! Let’s begin!

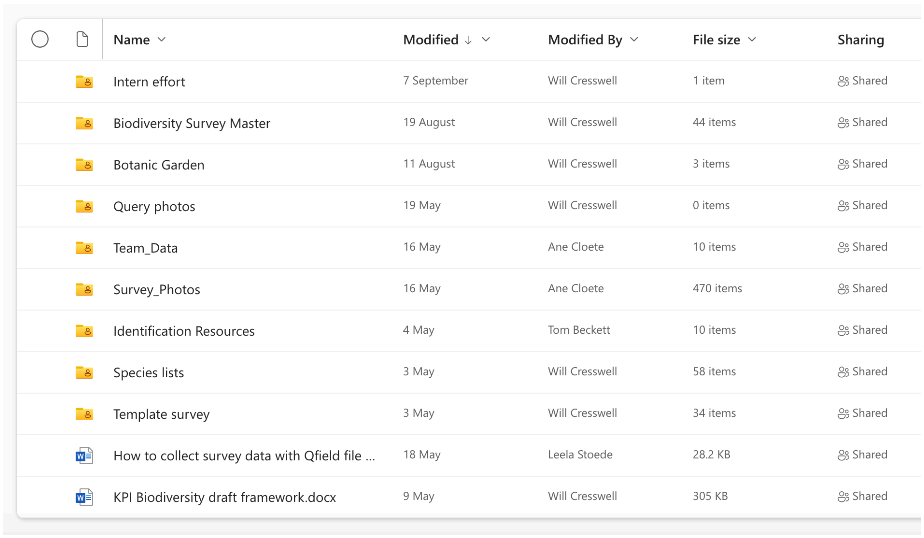
Chapter 2

Files within folders and folders within files and more files

Let’s go over all the files and folders you’ve now gotten access to starting with the OneDrive Folder.

2.1 Biodiversity KPI mapping Master

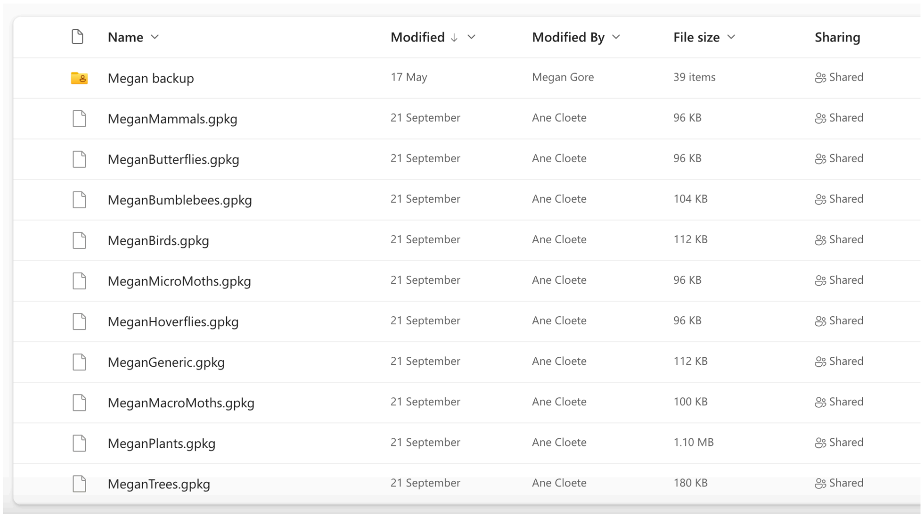
This is, as it’s name suggests, the master folder! And all the NB things are stored here, this is what you should see (unless more folders have been added since the creation of this bookdown). But don’t fret, you probably know about these folders already and the most important folders for the dashboard are: **Team_Data**, **Survey_Photos** and **Biodiversity Survey Master**.



Name	Modified ↓	Modified By	File size	Sharing
Intern effort	7 September	Will Cresswell	1 item	Shared
Biodiversity Survey Master	19 August	Will Cresswell	44 items	Shared
Botanic Garden	11 August	Will Cresswell	3 items	Shared
Query photos	19 May	Will Cresswell	0 items	Shared
Team_Data	16 May	Ane Cloete	10 items	Shared
Survey_Photos	16 May	Ane Cloete	470 items	Shared
Identification Resources	4 May	Tom Beckett	10 items	Shared
Species lists	3 May	Will Cresswell	58 items	Shared
Template survey	3 May	Will Cresswell	34 items	Shared
How to collect survey data with Qfield file ...	18 May	Leela Stoede	28.2 KB	Shared
KPI Biodiversity draft framework.docx	9 May	Will Cresswell	305 KB	Shared

2.1.1 Biodiversity Survey Master

Here is where all the students upload their biodiversity data. Each student has their own folder labelled with their names and within each folder all the geopackages for each taxa they have collected data on is stored (plus another folder containing backups), e.g.



Name	Modified ↓	Modified By	File size	Sharing
Megan backup	17 May	Megan Gore	39 items	Shared
MeganMammals.gpkg	21 September	Ane Cloete	96 KB	Shared
MeganButterflies.gpkg	21 September	Ane Cloete	96 KB	Shared
MeganBumblebees.gpkg	21 September	Ane Cloete	104 KB	Shared
MeganBirds.gpkg	21 September	Ane Cloete	112 KB	Shared
MeganMicroMoths.gpkg	21 September	Ane Cloete	96 KB	Shared
MeganHoverflies.gpkg	21 September	Ane Cloete	96 KB	Shared
MeganGeneric.gpkg	21 September	Ane Cloete	112 KB	Shared
MeganMacroMoths.gpkg	21 September	Ane Cloete	100 KB	Shared
MeganPlants.gpkg	21 September	Ane Cloete	1.10 MB	Shared
MeganTrees.gpkg	21 September	Ane Cloete	180 KB	Shared

If you don't know what a geopackage is, here's a quick description:

A GeoPackage is like a supercharged file for maps and location data. It's a clever way to bundle up all sorts of info—like where things are on a map, pictures, and details about those things. Think of it as a digital suitcase for geography. What makes it cool is that it works on different devices and software without any fuss.

2.1.2 Team_Data

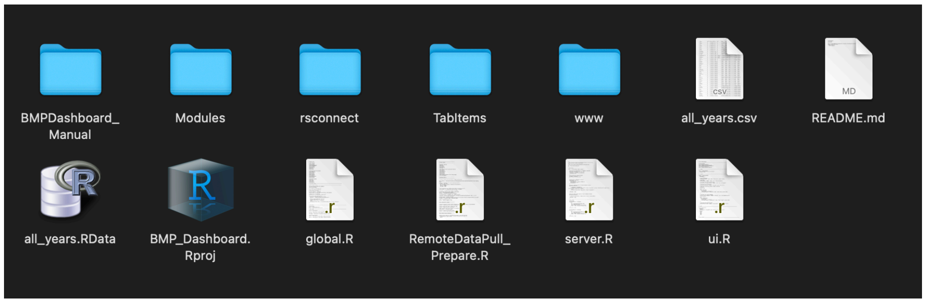
This folder contains photos of each student and a word document containing their “About Me” descriptions used in the dashboard. The nomenclature and file type is important here. The photo mus”t be saved with their name as the file name and they should be jpg’s! Then the word document should be called “student_aboutme” - you can change it if you want, but then you have to change it in the dashboard code as well. Within the document the student descriptions are paragraphs and within the first sentence the student introduces themselves with their names - this is NB. The code will will separate the text into paragraphs and then filter for the student by searching for their name. It’s your job to tell everyone to stick to this format and style!

2.1.3 Survey_Photos

All the photos taken while surveying are uploaded here with their unique photo ID as file name. The file name isn’t super important for the dashboard as long as it’s consisted between their records and the uploaded data. Ensure that all the photos are in the same format (i.e. jpg) - if not then there is a way to convert them all in one go which I’ll mention later.

2.2 BMPDashboard

Here is what you should see:



- BMPDashboard_Manual folder
- BMP_Dashboard.Rproj

- RemoteDataPullPrepare.R
- global.R
- server.R
- ui.R
- Modules folder
- rsconnect folder (you'll only see this once you've published the app to shinyapps.io)
- TabItems folder
- www folder
- README.md
- packages.R

If you're familiar with R projects and shiny apps then this should look familiar barring the obvious extras such the as the first folder (which contains the code for this bookdown). Let's quickly go over the rest.

2.2.1 RemoteDataPullPrepare.R

This contains the code for pulling the data from OneDrive and preparing it.

2.2.2 global.R, ui.R and server.R

These are the main shiny app files!

2.2.3 TabItems

Contains the ui code for each tab in the dashboard (Tab_About.R, Tab_KPI.R, Tab_Record_Finder.R, Tab_Student_Engagement.R and Tab_Taxa_Explorer.R). I've done this mostly to keep the ui file clean and comprehensible, but it's slightly annoying for work flow as when you make a change here, you have close the app and then run it again. You could put all the code into the ui script and then if you make changes you only have to reload the app. Up to you!

2.2.4 Modules

Contains the code for a custom valuebox which is used throughout the dashboard which I've turned into a shiny module. A Shiny module is like a neat toolkit for creating specific interactive parts of your app without making a mess of your code. It's like having a mini-app inside your bigger app. So, let's say you want a snazzy chart or a dynamic table—you can build that as a Shiny module. It keeps things organized and clean, making your web development life easier. It's like having building blocks for your website, and each block (or module) does a special job, making your site more interactive and user-friendly.

At the moment I've only converted the valuebox into a module, but there are other things that can be turned into modules too! Like the datatables or the bar graphs. Feel free to play around with this!

2.2.5 www

The **www** folder is like the storage room of your shiny app where you keep all the stuff that makes the outside look awesome. In the **www** folder, you put things like images, stylesheets (which control how things look), and JavaScript files (for extra functionality). In our **www** folder you'll find:

- all the survey photos
- a custom css stylesheet (custom.css)
- a copy of the Team_Data folder with additional objects: doc_parts.RData and student_text_sep.RData
- the allyears.RData object (this is the cleaned dataset which the dashboard uses)

2.2.6 README.md

This a plain text file usually written in a simple format called Markdown that contains the description you see on github.

2.2.7 packages.R

This r script contains all code to install and load all the packages required for the dashboard.

Chapter 3

The Data

3.1 Setting up OneDrive

The following instructions are for Mac users. If you are using windows, you should have OneDrive already installed on your computer.

1. **Download OneDrive:**
   - Go to the Mac App Store on your laptop.
   - Search for "OneDrive."
   - Click on "Get" or "Install" to download the OneDrive app.
2. **Install OneDrive:**
   - Once downloaded, open your Applications folder and locate the OneDrive app.
   - Drag the OneDrive app to your Dock for easier access (optional).
3. **Sign In:**
   - Open the OneDrive app.
   - Sign in with your university account.
   - Follow the on-screen prompts to set up OneDrive.
4. **Choose Folders to Sync:**
   - Once signed in, you'll have the option to choose which folders from your OneDrive cloud storage you want to sync with your Mac.
   - Select the folders you want or choose to sync everything.
5. **Set OneDrive Preferences:**
   - Click on the OneDrive icon in the Mac menu bar at the top of your screen.

- Click on the three dots (More) and select "Preferences."
- Here, you can adjust settings like:
  - Starting OneDrive automatically when you sign in to your Mac.
  - Choosing how files are downloaded or uploaded (e.g., over metered networks).
  - Setting up file on-demand (allows you to see all your files without having them downloaded).

6. **Accessing Your Files:**

- A OneDrive folder will now be present in your Mac's Finder. This folder will sync with your online OneDrive storage. Any files or folders you add to this folder will be automatically uploaded to the cloud, and any changes you make to files in this folder will be reflected in the cloud version. Read that again! It important that you know that if you change or delete any files in the OneDrive folder on your laptop it will affect the database online.

7. **To Unlink or Sign Out:**

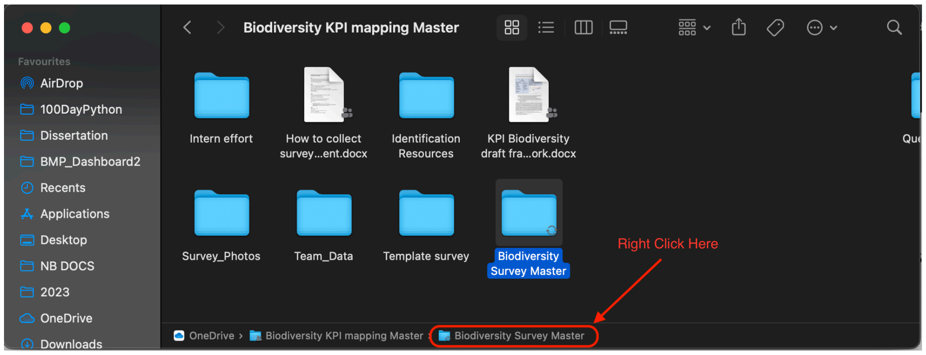
   - If you ever wish to unlink your account or sign out, click on the OneDrive icon in the Mac menu bar.
   - Click on the three dots (More) and select "Preferences."
   - Go to the "Account" tab and select "Unlink this Mac."

3.2 Getting the data

First, navigate to the R script called "packages" to check whether you have all the required packages, if you don't the script will also install and load them for you. Done!

Now, open the script "RemoteDataPull_Prepare".

The most important first step here is to modify the object "path_to_master" with the filepath to where ever you have the OneDrive folder on your device and specifically to the main survey folder "Biodiversity Survey Master". If you are using a Mac, navigate to the folder in Finder and then in the bottom panel (see below), right click the folder name and select "Copy"folder name" as Pathname".



Let's break down the code:

1. **Setting the Path to the Master Directory**:

```
  path_to_master <- "/Users/anecloete/Library/CloudStorage/OneDrive-UniversityofStAndrews/Biodi
```

This line sets a variable called `path_to_master` to the path of the main directory where the geopackage files are located. Paste the pathname you copied here.

2. **Obtaining All Geopackage Files**:

```
all_gpkgs <- list.files(
  path = paste0(path_to_master,"/Biodiversity Survey Master"),
  recursive = TRUE,
  pattern = "\\.gpkg$",
  full.names = TRUE
)
```

This code lists all files with the ".gpkg" extension in the specified directory and its subdirectories. - The `path` argument specifies where to look. - `recursive` set to `TRUE` means it will look in subdirectories as well. - `pattern` filters for files ending with ".gpkg". - `full.names` set to `TRUE` ensures the full path of each file is returned, not just its name.

3. **Filtering Geopackages with 'Habitat_Polygon' in Their Name**:

```
  survey_polygons <- all_gpkgs[grepl("Habitat_Polygon", all_gpkgs)]
```

This filters the `all_gpkgs` vector for filenames that contain the substring "Habitat_Polygon" and assigns the subset to `survey_polygons`.

4. **Excluding Geopackages Based on Certain Keywords**:

```r
pattern <- "Polygon|DESKTOP|PC|LAPTOP|Wills"
all_gpkgs <- all_gpkgs[!grepl(pattern, all_gpkgs)]
```

Here, a pattern is defined to exclude geopackages with certain keywords in their names. The `grepl` function checks for matches, and the `!` operator negates the condition to exclude matches.

5. **Exclude a Specific Problematic Geopackage**:

```r
problematic_gpkg <- paste0(path_to_master,"/Erica/Erica Backup/Ericabutterflies.gpk
all_gpkgs <- all_gpkgs[all_gpkgs != problematic_gpkg]
```

This code defines a specific geopackage file path that is "problematic" and then removes this file from the `all_gpkgs` vector.

6. **Reading All Geopackages into a List**:

```r
myfiles <- lapply(all_gpkgs, st_read)
```

The `lapply` function is used here to apply the `st_read` function to each file path in the `all_gpkgs` vector. The result is a list, with each element being the content of a geopackage file stored as an spatial features dataframe. The `st_read` function is part of the `sf` package in R and is used to read spatial data.

In summary, this code: - Sets a path to a master directory. - Lists all geopackage files from this directory and its subdirectories. - Filters and excludes certain geopackages based on keywords or specific filenames. - Reads the content of each remaining geopackage file into a list.

3.3 Data cleaning and preparation

Most of this is self explanatory or is a bit tedious to explain. I would recommended investigating and exploring the data before the cleaning and preparation so that some of the lines make more sense. But here are a few notable things:

- Each student have a backup folder embedded in their folder, these geopackages are removed in line 72
- There is separate geopackage for tree data collected, this is included in the dashboard so is cleaned separately and then joined to the rest.
- The Species column contains the name of the species recorded, whether this is the scientific name, the common name or anything else. So all the naming columns are coalesced into one column called Species.

- Line 135 was necessary because for some reason empty entries in the relevant columns were " " and not NA

Some of the code could be a little less sausage making-ish and could be simplified (somethings were added after the fact etc), feel free to make it more efficient!

SUMMARY:

1. **Coordinate Transformation**:
    - Transforms the coordinate reference system of spatial data to standard latitude and longitude (EPSG:4326).

```
# Transform coordinate reference system from WGS 84 / Pseudo-Mercator to standard lat long (EPSG:
taxa_dat <- lapply(myfiles, function(x) st_transform(x, 4326))
```

2. **Data Conversions**:
    - Converts the spatial data frames to regular data frames.
    - Names each data frame in the list based on the geopackage filename.
    - Removes duplicate data frames based on their names.

```
# Convert spatial data frames to regular data frames
taxa_dat <- lapply(taxa_dat, as.data.frame)

# Name each data frame in the list based on the geopackage filename
names(taxa_dat) <- basename(all_gpkgs)

# Remove any duplicate data frames by name
taxa_dat <- taxa_dat[!duplicated(names(taxa_dat))]
```

3. **Tree Entries Package Preparation**:
    - Modifies columns for a specific geopackage ('Tree Species Entries.gpkg') to ensure compatibility with subsequent operations.
    - Cleans and modifies species names.
    - Sets certain columns to specific values or `NA`.
    - Filters and selects specific columns.

```
# Modify certain columns for 'Tree Species Entries.gpkg' so that bind_rows works
taxa_dat$`Tree Species Entries.gpkg` <- taxa_dat$`Tree Species Entries.gpkg` %>%
  rename(Species = 1, Date = 2) %>%
  mutate(
    Species = case_when(
      Species %in% c("unknown/other", "Unknown/other") ~ comments.unlisted.species,
      Species == "Unknown young pine " ~ "Unknown young pine",
      Species == "Willow x10" ~ "Willow",
      TRUE ~ Species
    ),
    taxa = "Vascular Plants",
    Observer = NA,
    photoid = NA,
    Count = NA,
    Other = NA,
    Speciesful = NA
  ) %>%
  filter(Species != "") %>%
  select(Species, Date, taxa, Count)
```

4. **Date Processing**:
    - Converts, splits, and extracts components (year, month, day) of the 'Date' column.

```
# Date Processing: Convert, split and extract components of the 'Date' column
taxa_dat <- lapply(taxa_dat, function(df) {

  df$Date <- as.character(df$Date)

  df <- df %>%
    separate(Date, into = c("date", "time"), sep = " (?=[^ ]+$)") %>%
    mutate(
      date = ymd(gsub("/", "-", date)),
      year = year(date),
      month = month(date),
      day = day(date)
    )
  return(df)
})
```

5. **Tree Data Extraction**:
    - Removes the tree data frame from the list and stores it separately.

```
# Extract and remove the tree dataframe from the list
tree_data <- taxa_dat$`Tree Species Entries.gpkg`
taxa_dat$`Tree Species Entries.gpkg` <- NULL
```

6. **Combining Data**:

- Merges all data frames in the list into a single data frame.

```
# Combine all data frames in the list into a single data frame
taxa_comb <- bind_rows(taxa_dat)
```

7. **Data Cleaning and Transformation**:
    - Performs several operations to clean and transform the combined data, including:
        - Date conversions and modifications.
        - Handling missing values.
        - Excluding certain records.
        - Recoding values in various columns.
        - Selecting and renaming columns.

```
# Clean and transform taxa data for further analysis
taxa_clean <- taxa_comb %>%
  drop_na(taxa) %>%
  as.character(df$Date) %>%
  separate(Date, into = c("date", "time"), sep = " (?=[^ ]+$)") %>%
  mutate(
    date = ymd(gsub("/", "-", date)),
    year = year(date),
    month = month(date),
    day = day(date)
  ) %>%
  filter(!(taxa == "hoverfly" & Observer == "Erica")) %>% # remove Erica hoverfly entries
  unite(collapsed_species, specieslatin:seaweedlatin, sep = ",", na.rm = TRUE) %>%
  mutate_at(vars(Species, collapsed_species), na_if, "") %>%
  mutate(
    Species = coalesce(Species, SpeciesSci, Speciesfull, collapsed_species, species, Other),
    photoid = if_else(is.na(photoid), NA, paste0(photoid, ".jpg")),
    Count = ifelse(is.na(Count), 1, Count),
    taxa = recode(taxa, tree = "Vascular Plants"),
    year = recode(year, `2023` = "2022/2023", `2022` = "2022/2023")
  ) %>%
  filter(!str_detect(Species, "(?i)unknown"), taxa != "bee") %>%
  filter_at(vars(taxa, Observer), all_vars(!is.na(.))) %>%
  mutate(
    taxa = recode(taxa,
                  plant = "Vascular Plants",
                  bird = "Birds",
                  macromoth = "Macromoths",
                  micromoth = "Micromoths",
                  butterfly = "Butterflies",
                   dragonfly = "Dragonflies",
                   hoverfly = "Hoverflies",
                   bat = "Bats",
                   amphibian = "Amphibians",
                   reptileamphibian = "Amphibians",
                   bumblebee = "Bumblebee",
                   mammal = "Mammals",
                   ladybird = "Ladybirds",
                   tree = "Vascular Plants"),
    year = recode(year,
                   "2023" = "2022/2023",
                   "2022" = "2022/2023"),
    Observer = recode(Observer,
                       Other1 = "Cori")) %>%
  select(Species, SpeciesSci, Count, date, Observer, taxa, photoid, geometry, year, da
  rename(
    Date = date,
    Taxa = taxa,
    PhotoID = photoid
  )
```

8. **Merging Tree Data with Cleaned Data**:
   - Prepares the tree data to be merged with the cleaned data.
   - Merges both datasets.

```
# tree data prep for bind
tree_data <- tree_data %>%
  mutate(
    year = ifelse(year == "2023" | year == "2022", "2022/2023", year),
    Count = 1 # add column count
  ) %>%
  rename(
    Date = date,
    Taxa = taxa
  )

# Merge tree data with the cleaned data
all_years <- bind_rows(taxa_clean, tree_data)
```

9. **Geometry Processing**:
   - Splits the geometry column into separate latitude and longitude columns.

```r
# split into lat and long for mapping
all_years <- all_years %>% mutate(long = unlist(map(geometry,1)),
           lat = unlist(map(geometry,2)))
```

10. **Saving the Resulting Data**:
    - Saves the final data frame both as an RData object and as a CSV file.

```r
# Save the resulting dataframe
save(all_years, file = "all_years.RData")
write.csv(all_years, file = "all_years.csv")
```

3.4 Pulling photos and documents from OneDrive

3.4.1 Photos

1. **Reading Team Photos from a Directory**:

```r
   # read in all photo files within Survey_Photos in OneDrive Folder
   all_Tphotos <- list.files(
     path =   paste0(path_to_master,"/Team_data"),
     recursive = TRUE,
     pattern = "\\.jpg$",
     full.names = FALSE
   )
```

This is the same as the first few lines of code, except that the path is constructed by appending "/Team_data" to the `path_to_master`.

The result is stored in `all_Tphotos`, which will be a vector of filenames (with ".jpg" extension) from the specified directory and its subdirectories.

2. **Constructing a Destination Path for Team Photos**:

```r
   www_Tfolder_path <- file.path(getwd(), "www/Team_Data")
```

This line constructs a path by combining the current working directory (obtained using `getwd()`) with "www/Team_Data".

3. **Copying the Team Photos**:

```
photo_move <- file.copy(
  from = file.path(paste0(path_to_master,"/Team_data"), all_Tphotos),
  to = file.path(paste(www_Tfolder_path), all_Tphotos)
)
```

Here, `file.copy` is used to copy files. The `from` argument constructs the full paths of the source files, and the `to` argument constructs the full paths of the destination. The photos are copied from the source directory to the destination.

The same process as above is then done for the Survey Photos.

In summary, the code is designed to:

- Read all ".jpg" files from "Team_data" and "Survey_Photos" directories (including subdirectories).
- Copy those files to two new destinations under the "www" folder in the current working directory.

3.4.2 Documents

The code for downloading and saving the students' about me descriptions is very similar to the process above. The only addition is that the text of the "student_aboutme" word document is then extracted and saved into an object called "student_text". Then the text is split into paragraphs.

3.5 Things to think about

How will the next years data be included? In the same folder? New QGIS folder?

Chapter 4

UI

4.1 ui.R

Here we define the user interface (UI) for the dashboard. First we source the ui code for each of the tabs:

```r
# source tabItems
source("TabItems/Tab_KPI.R")
source("TabItems/Tab_Taxa_Explorer.R")
source("TabItems/Tab_Student_Engagement.R")
source("TabItems/Tab_About.R")
source("TabItems/Tab_Record_Finder.R")
```

1. **Header Section (`dashboardHeader`):**
    - Sets up the header of the dashboard with the title "Biodiversity Monitoring Programme."

```r
# 1. HEADER ------------------------------------------------------------------------------------------------

header <- dashboardHeader(
  title = "Biodiversity Monitoring Programme",
  titleWidth = 320
)
```

2. **Sidebar Section (`dashboardSidebar`):**
    - Creates a sidebar that contains a menu (`sidebarMenu`) with several items.
    - It includes a dropdown (`selectInput`) for choosing the survey year.

- Menu items include "About," "Key Performance Indices," "Taxa Explorer," "Student Engagement," and "Record Finder."
- Conditional panels (`conditionalPanel`) show/hide additional input fields based on the selected menu item. For example, when "Taxa Explorer" is selected, a dropdown to select a taxa is displayed; when "Student Engagement" is selected, a dropdown to select a student is displayed.

```r
# 2. SIDEBAR -------------------------------------------------------------------------

sidebar <- dashboardSidebar(
  sidebarMenu(
    id = "sidebar",

    # select survey year
    selectInput(inputId = "year", label = "Select year", choices = years),

    # About Page
    menuItem("About", tabName = "about", icon = icon("circle-info")),

    # first menu item: Key Performance Indices
    menuItem("Key Performance Indices", tabName = "kpi", icon = icon("gauge")),

    # second menu item: Taxa Explorer
    menuItem("Taxa Explorer", tabName = "taxa_expl", icon = icon("crow")),

    # Show panel only when taxa explorer sidebar is selected
    useShinyjs(),

    # select taxa
    div(
      id = "taxa_cond",
      conditionalPanel(
        "input.sidebar == 'taxa_expl'",
        selectInput(inputId = "taxa_select", label = "Select taxa", choices = NULL)
      )
    ),

    # third menu item: Student Engagement
    menuItem("Student Engagement", tabName = "stud_expl", icon = icon("graduation-cap"

    # select Observer
    div(
      id = "student_cond",
      conditionalPanel(
        "input.sidebar == 'stud_expl'",
        selectInput(inputId = "student_select", label = "Select Student", choices = NULL)
      )
    ),

    # fourth menu item: Record & Photos
    menuItem("Record Finder", tabName = "record_finder", icon = icon("camera-retro"))
  )
)
```

3. **Body Section (`dashboardBody`):**
   - Defines the main body of the dashboard.
   - Utilizes the `dashboardBody` function and includes additional styling using `tags$head` to link an external stylesheet (`custom.css`).
   - Uses the `tabItems` function to organize the content into different tabs (e.g., `Tab0`, `Tab1`, etc.).

```
# 3. BODY --------------------------------------------------------------------------------------

body <- dashboardBody(
  useShinyjs(),
  tags$head(
    tags$link(rel = "stylesheet", type = "text/css", href = "custom.css")
  ),
  tabItems(
    Tab0,
    Tab1,
    Tab2,
    Tab3,
    Tab4
  )
)
```

4. **UI Function (`dashboardPage`):**
   - Combines the header, sidebar, and body into a complete UI using the `dashboardPage` function.
   - This UI function essentially wraps up all the UI components, creating the structure of the Shiny app.

```
# 4. UI Function -------------------------------------------------------------------------------

ui <- dashboardPage(header, sidebar, body)
```

4.2 Tabs

4.2.1 Tab_About

This script defines the UI for the About tab. At the bottom of teh script is where the UI is put together:

```
# Tab Code ---------------------------------------------------------------------------

Tab0 <- tabItem(
  tabName = "about",
  tabBox(
    width = "100%",
    id = "tabset1",
    overview_panel,
    kpi_panel,
    taxa_explorer_panel,
    student_engagement_panel,
    record_finder_panel
  )
)
```

The tab itself consists of a tabBox (a box with tabs) and the content for each of the panels within the box are defined in the sections above this piece of code. For example, the tab named “Overview” is defined as:

```
# Overview Panel Content -------------------------------------------------------------

overview_panel <- tabPanel(
  title = "Overview",
  tags$div(
    class = "landing-wrapper",
    # child element 1: images
    tags$div(
      class = "landing-block background-content",
      # top left
      img(src = "group_photo2.jpeg"),

      # top right
      img(src = "TomButterfly020823HolBl.jpg"),

      # bottom left
      img(src = "TomMoth120823JulyHfly.jpg"),

      # bottom right
      img(src = "group_photo.jpeg")
    ),

    # child element 2: content
    tags$div(
      class = "landing-block foreground-content",
      tags$div(
        class = "foreground-text",
        # tags$h2("Biodiversity Monitoring Programme Data Dashboard"),
        tags$p("Welcome to the University of St Andrews Biodiversity Monitoring Program Data Dash
        tags$p("Explore the wonders of the diverse ecosystems that surround us and gain insights
        tags$p("On this dashboard, you will find a wealth of information, from species counts and
        tags$p("Together, we can make a difference in preserving and protecting the biodiversity
      )
    )
  )
)
```

The content of this tab is organized into two main elements within a div container:

Images Section:

Four images are displayed in a 2x2 grid (top left, top right, bottom left, bottom right). These images are loaded from files named "group_photo2.jpeg," "TomButterfly020823HolBl.jpg," "TomMoth120823JulyHfly.jpg," and "group_photo.jpeg."

Content Section:

Text content is provided in a "foreground-content" div.

The div elements are given class ids with the argument "class" because we refer to them in the custom.css file - where we style the elements. I used the code from this link to build the content: https://community.rstudio.com/t/background-images-in-shiny/12261.

4.2.2 Tab_KPI

Here we define the content for the tab: "Key Performance Indices".

Summary: - This tab is designed to display key performance indices related to biodiversity.

- It includes sections for the Annual Biodiversity Index, Benchmark Biodiversity Index, Change Biodiversity Index (conditionally displayed), and a chart showing the number of species per taxa.

- The layout is organized using `fluidRow` to arrange content horizontally and `box` to create collapsible boxes. Conditional rendering using `conditionalPanel` allows for dynamic display based on user input.

Let's break down the code to understand its structure and purpose:

```
# TAB: KPI
Tab1 <- tabItem(
  tabName = "kpi",
  h3("Annual Biodiversity Index"),
  fluidRow(
    valueBoxOutput("S_H"),
    valueBoxOutput("area"),
    valueBoxOutput("engagement")
  ),
  h3("Benchmark Biodiversity Index"),
  fluidRow(
    valueBoxOutput("MS"),
    valueBoxOutput("local"),
    valueBoxOutput("fife")
  ),

  div(
    conditionalPanel(
      "input.year !== '2022/2023'",
      h3("Change Biodiversity Index"),
      box(
        collapsible = TRUE, width = 12,
        plotlyOutput("example_plot"), style = "display:block; overflow-x: scroll;"
      )
    )
  ),

  # taxa chart
  fluidRow(
    box(
      collapsible = TRUE, width = 12, title = "Number of species per taxa",
      plotlyOutput("taxa_bar", height = "300px"), style = "display:block; overflow-x: s
    )
  )
)
```

Tab Structure:

- `tabItem`: This defines a tab within the Shiny application.

  - `tabName = "kpi"`: The tab is named "kpi" which corresponds to the tabName id we gave it in the ui.R script:

```r
# from ui.R:

# first menu item: Key Performance Indices
menuItem("Key Performance Indices", tabName = "kpi", icon = icon("gauge"))
```

Content Sections:

- **Annual Biodiversity Index Section:**
  - `h3("Annual Biodiversity Index")`: This adds a subheading indicating the section's title.
  - `fluidRow`: This creates a row to hold the content.
    - `valueBoxOutput`: These are placeholders for numeric values to be displayed. Three values ("S_H," "area," "engagement") are expected to be displayed in a row.
- **Benchmark Biodiversity Index Section:**
  - `h3("Benchmark Biodiversity Index")`: Another subheading for this section.
  - `fluidRow`: Another row to hold the content.
    - `valueBoxOutput`: Similar to the Annual Biodiversity Index section, three values ("MS," "local," "fife") are expected to be displayed in a row.
- **Change Biodiversity Index Section (Conditional):**
  - `div`: A div container.
    - `conditionalPanel`: This panel is conditionally displayed based on the input "year" not being equal to "2022/2023."
      - `h3("Change Biodiversity Index")`: A subheading for this section.
      - `box`: A collapsible box containing a Plotly plot (`example_plot`). This section seems to be about the change in biodiversity over time.
- **Taxa Chart Section:**
  - `fluidRow`: Another row.
    - `box`: A collapsible box with a title ("Number of species per taxa") and a Plotly plot (`taxa_bar`). This section likely displays a chart showing the number of species per taxa.

4.2.3 Tab_Taxa_Explorer

Summary - The “TAXA EXPLORER” tab is designed to explore biodiversity data related to taxa.

- It includes sections for summary charts, a Leaflet map, detailed lists of species, and records.
- The layout is organized using `tabBox` for grouping content into tabs and `box` for creating collapsible boxes.
- Various outputs such as plots and DataTables are defined for displaying data.

```
# TAB: TAXA EXPLORER
Tab2 <- tabItem(
  tabName = "taxa_expl",
  br(),
  tabBox(
    id = "box1", height = 500, width = 12,
    tabPanel("Summary",
      fluidRow(
        column(6,
          h5("Top 10 Species Composition"),
          plotOutput("species_pie")
        ),
        column(6, offset = 1.5,
          br(),
          br(),
          br(),
          fluidRow(
            valueBoxOutput("num_species_taxa", width = 10), style ="margin: auto;"
          ),
          fluidRow(
            valueBoxOutput("num_records_taxa", width = 11), style ="margin: auto;"
          ),
          fluidRow(
            valueBoxOutput("top_obs", width = 12), style ="margin: auto;"
          )
        )
      )
    ),
    tabPanel("Top 50 species", plotlyOutput("species_bar", inline = FALSE, width = "10
  ),
  box(
    id = "box1", height = "100%", width = 12,
```