ALA Data Cleaning

Table of contents

Welcome

Data cleaning is the exploration, detection and correction of data which may be errornous, unsuitably formatted, or otherwise unsuited for use in your project. The definition of 'clean' data therefore varies depending on the project and data sources, and as such there is no 'one-size-fits-all' approach. Nevertheless, there are some common processes and concepts that can be applied to many data cleaning workflows.

However, these processes typically require the use of a programming language, which can be a barrier of entry for many. Therefore, our goal in creating this resource is to assist researchers and decision makers that may have limited experience with cleaning geo-referenced biodiversity data. The language used will be R, which is commonly used in ecological projects, and has a rich ecosystem of packages for data cleaning.

In this book, we provide an overview of a typical data cleaning workflow - from acquisition, identifying potential errors, to correction. These processes are broken down into chapters, whithin each we include practical guidelines, example R code, and additional resources that may aid with each data cleaning step.

The content of this book was informed by the current state of biodiversity literature surrounding data preparation for species distribution modelling. For more details about how this was done, please refer to the Appendix. All resources that were used can be found in the References page.

How to contribute

We would like to preface that we are not experts in data cleaning, but felt there was need for a consolidated resource to guide

data cleaning decisions.

Contributions to this document are welcome. For any questions, feedback, or other issues, please open an issue on our GitHub repository. If you would like to make changes to the content of the website, you are welcome to submit a pull request containing your proposed changes. Please note, it is considered best practice to open an issue before working on a pull request, to allow discussion surrounding the proposed changes. Our contributing guidelines can be found here.

How to cite

You can cite this book as: Kar, F., Fenker, J., Schneider, M., Westgate, M. (2023). Cleaning biodiversity data in R. Retrieved Month dd, yyyy, from https://atlasoflivingaustralia.github.io/cleaning_data/

Acknowledgements

This book was inspired by an Australian Research Data Commons project where our team worked closely with research partners to streamline their data cleaning workflows. This book is a collaborative effort from the Science and Decision Support team at the Atlas of Living Australia (ALA)

Authors listed in alphabetic order: - Fonti Kar - Jessica Fenker - Margot Schneider - Martin Westgate

1 Introduction

> "Garbage in, garbage out" - George Fuechsel, IBM computer programmer

Data from different sources vary widely in their **structure, formatting and quality**. As a result, collating and combining them into something usable to answer a research question can be both challenging and time consuming.

The idea that inputting flawed or nonsensical data produces an output of similar quality is well known in many scientific disciplines. In ecology and biodiversity research, however, scientists are particularly at risk of unintentionally using poor data because they **often bring together lots of data from many sources to address their research questions.**

Data cleaning, the process of identifying and fixing incompatible, incorrect or doubtful data, is an essential step in ecology and biodiversity research. Data cleaning improves data quality and the validity of scientific findings (Rodrigues et al. 2022).

This book will guide you through how to acquire and clean open access biodiversity data in R. We will be working with point-based species occurrence data from online infrastructures such as Global Biodiversity Information Facility (GBIF) and the Atlas of Living Australia (ALA).

We will be working with the R package `galah` for accessing biodiversity data. If you have occurrence data you have personally collected, some parts of this book may still be relevant.

We have included code blocks throughout this book to show you how to execute a particular task.

```
library(r-package)
cleaning step |>
```

```
data
```

Any `R packages` that are needed for data cleaning or visualising will be noted at the beginning of the code block.

1.1 Outline: What you will learn

Cleaning open-access biodiversity data involves a few steps:

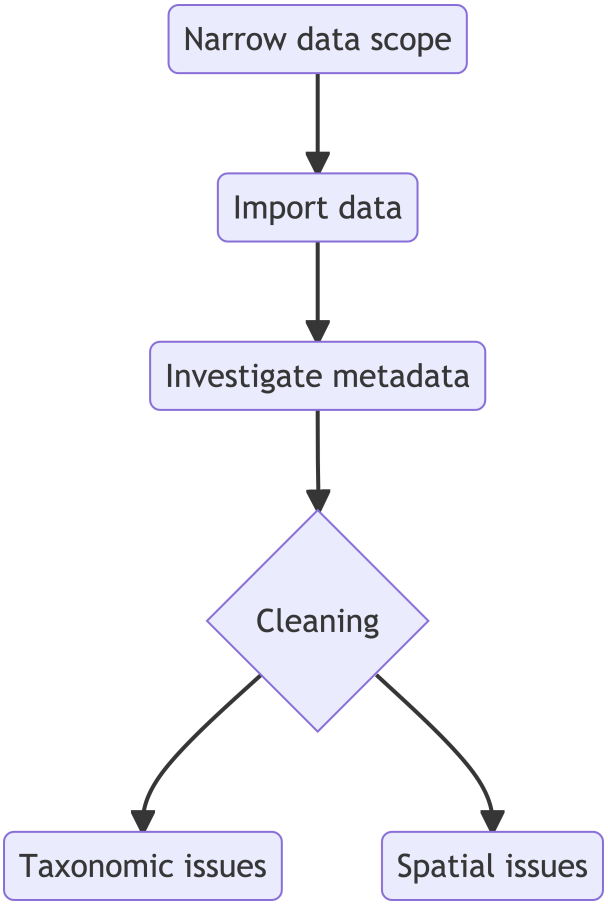


Figure 1.1: The cleaning process

Once data have been narrowed, imported and formatted correctly we can then start the major cleaning steps. We will focus on how to deal with the most common issues with biodiversity data:

Taxonomic standardisation problems, such as:

- **varied species name synonyms**
- **naming authorities and duplicates**

as well as spatial errors and issues, like:

- **suspicious outliers**
- **duplicates.**

> **!** **Important!**
>
> **No one size fits all in this workflow**.
> In the literature, we found that data cleaning steps are frequently completed in entirely different orders. Not all steps are relevant or possible depending on the nature of the study, or, the area of expertise possessed. Therefore this book does not necessarily have to be used in a linear fashion. *While some steps may logically come first, you may need to go back to them after completing another.*

1.2 What you won't learn

There are many important subject areas that this book will not cover. We won't be teaching you:

- How to clean environmental data that isn't occurrence / biodiversity data: e.g. trait data
- How to run a species distribution model
- Hypothesis testing or experimental design

1.3 Prerequisites

This data cleaning book assumes some basic knowledge using R and RStudio, but if you have never used R before there are a plethora of online and freely accessible resources to help you get started.

1.3.1 User accounts

To get data out of data infrastructures such as the Atlas of living Australia (ALA) or the Global Biodiversity Information (GBIF) you'll need to first create an account.

You'll want to sign up for an account with the relevant data infrastructure. This book will use ALA and GBIF data as examples.

You can do this for the Atlas of Living Australia here -

and for the Global Biodiversity Information Facility here.

1.3.2 R

Download R from CRAN for your operating system and install it on your device. Major updates for R come out yearly, with a few minor releases throughout the year - so make sure to update semi regularly.

1.3.3 RStudio

RStudio is an integrated development environment (IDE) for R programming. R operatees within R studio, so to code, you will need both R and RStudio. Download and install RStudio for your operating system here.

2 Data scope

Data scope refers to the type and breadth of data needed for your project. Defining your scope is an essential part of forming a research question, ultimately impacting what data you will use in your project. Depending on the data available, your scope and research question may change.

For example, you might have a question about several species in the same area. However, data for one or more of those species could be limited because observations are rare, surveying the area where it lives is difficult, or only several historical records exist.

Without narrowing your data scope, you might find yourself downloading more data than you need, which can needlessly increase how much time is spent processing data prior to analyses. Alternatively, you might find there isn't enough data to answer your question.

While there are workable methods to analyse small sets of biodiversity data (e.g. hulls), it's worth thinking critically about whether the amount of data available will allow you to sufficiently answer your research question.

To start, some initial questions you might ask are:

- What is the **temporal** unit relevant for your research question?

  Am I only interested in more recent data? Is there data that are too old to be relevant for my question?

- What is the **taxonomic** unit of your proposed research question?

  Is my question specific to one or more species in the same taxonomic group? Does it compare between higher taxonomic levels like genus, family or order?

- What is the **spatial** scale of your proposed research question?

  Is my question relevant at a global or national level, or is it specific to a region or ecosystem?

Questions like these will help you define what data is most relevant for your research question, and help you begin to think about how much evidence available, and the trade-offs you might make between the specificity of your question and the certainty of your answer.

2.1 An example question

As an example, let's imagine we were interested in understanding more about the distribution of several Jewel beetles in the genus *Lampromicra*. Let's see some ways we might investigate what data are available about them in Australia.

Curious what a Jewel beetle looks like? Here is a *Lampromicra senator* perched on a leaf by Matthew Connors CC-BY-NC 4.0 (Int)

2.2 Temporal scope

A good first step to understanding what data are available is to check how many observations there are over different time periods. We can check how many total observations there are in the Atlas of Living Australia (the largest biodiversity data aggregator in Australia) with the following query:

```
library(galah)
```

```
library(galah)

atlas_counts()
```

```
# A tibble: 1 x 1
      count
      <int>
1 130679056
```

Now let's check how many total insect records there are.

```
galah_call() |>
  galah_identify("insecta") |>
  atlas_counts()
```

```
# A tibble: 1 x 1
    count
    <int>
1 5762407
```

Now let's see how many insects have been observed each year over the last 10 years. We'll order this by descending year using `dplyr::arrange()` and `dplyr::desc()`.

```
library(dplyr)

galah_call() |>
  galah_identify("insecta") |>
  galah_filter(year >= 2013) |>
  galah_group_by(year) |>
  atlas_counts() |>
  dplyr::arrange(dplyr::desc(year))
```

```
# A tibble: 11 x 2
   year   count
   <chr>  <int>
 1 2023   332708
 2 2022   447952
 3 2021   356419
 4 2020   265366
 5 2019   186038
 6 2018   228845
 7 2017   194872
```

```
 8 2016  170011
 9 2015  124940
10 2014  126874
11 2013  116485
```

Now that we have an idea of the total amount of data in the Atlas of Living Australia, let's try checking how many observations exist of the genus *Lampromicra*. We'll first make sure the scientific name *Lampromicra* returns the taxon information we expect with `search_taxa()`.

```
search_taxa("Lampromicra")
```

```
# A tibble: 1 x 13
  search_term scientific_name scientific_name_authorship taxon_concept_id  rank
  <chr>       <chr>           <chr>                      <chr>             <chr>
1 Lampromicra Lampromicra     Stål, 1873                 https://biodiver~ genus
# i 8 more variables: match_type <chr>, kingdom <chr>, phylum <chr>,
#   class <chr>, order <chr>, family <chr>, genus <chr>, issues <chr>
```

This looks correct! Next let's see how many total observations there are of *Lampromicra* and how many observations there were in each of the last 10 years.

```
galah_call() |>
  galah_identify("Lampromicra") |>
  atlas_counts()
```

```
# A tibble: 1 x 1
  count
  <int>
1  1359
```

```
galah_call() |>
  galah_identify("Lampromicra") |>
  galah_filter(year >= 2013) |>
  galah_group_by(year) |>
  atlas_counts() |>
```

```
    dplyr::arrange(dplyr::desc(year))
```

```
# A tibble: 11 x 2
   year  count
   <chr> <int>
 1 2023    129
 2 2022    229
 3 2021    219
 4 2020    151
 5 2019     75
 6 2018     53
 7 2017     44
 8 2016     42
 9 2015     14
10 2014     14
11 2013      6
```

There are a growing number of observations from 2012 to 2023 of Jewel beetles in the Atlas of Living Australia, with fewer than 100 observations each year prior to 2020.

With this information, we might choose to combine data from all years in our analysis (rather than separating them by year). Alternatively, we might decide to only include data since 2020, which might be sufficient to represent where *Lampromicra* are found and be more relevant because they are more recent observations. These are decisions that might affect the granularity of the research question we ask.

2.3 Taxonomic scope

Taxonomy refers to ways by which we classify an organism and its relationship to all other organisms. Typically, your research question will be concerned with one or more taxonomic groups, ranging from a single species to an entire kingdom (e.g. *Plantae*).

In our example, we are interested in understanding more about Jewel beetles in the genus *Lampromicra*.

We first might want to know what species there are in the genus *Lampromicra* and view some additional taxonomic information about them. We can do this by adding `atlas_species()` to the end of a query in {galah}.

> **Note**
>
> You will need to add an email address registered with the Atlas of Living Australia in `galah_config()` to download species information.

```
galah_config(email = "oliviajane.t@hotmail.com")

galah_call() |>
  galah_identify("Lampromicra") |>
  atlas_species() |>
  select(family:species_guid)
```

```
# A tibble: 3 x 5
  family        genus       species              author            species_guid
  <chr>         <chr>       <chr>                <chr>             <chr>
1 Scutelleridae Lampromicra Lampromicra senator (Fabricius, 1803) https://biodi~
2 Scutelleridae Lampromicra Lampromicra aerea   (Distant, 1892)   https://biodi~
3 Scutelleridae Lampromicra Lampromicra regia   Bergroth, 1895    https://biodi~
```

Our query returned three species in the genus *Lampromicra*. We can enter the urls returned under `species_guid` in a web browser if we wished to know more information about any of them.

We might also wish to check the total number of observations of each of these species. We can check this by grouping our counts by species using `galah_group_by(species)`

```
galah_call() |>
  galah_identify("Lampromicra") |>
  galah_group_by(species) |>
  atlas_counts()
```

```
# A tibble: 3 x 2
```

```
  species             count
  <chr>               <int>
1 Lampromicra senator  1114
2 Lampromicra aerea     171
3 Lampromicra regia      14
```

Our result shows that the majority of observations of *Lampromicra* are of the species *Lampromicra senator*.

Given this information, we might consider whether our question needs to be made at the species level, or whether we might increase the taxonomic scope to the genus or family level to use more data. Ultimately, the taxonomic scope of your data will depend on how important it is to your question to compare specific taxonomic groups.

2.4 Spatial scope

It's useful to investigate the spatial range of available data for your taxonomic group(s) of interest. How specific your question can be may change depending on whether the majority of data is in only a few locations or evenly spread over the entire distribution of a species or taxonomic group.

For our example question about *Lampromicra*, we may wish to map where observations in Australia have been made. We can do this by using the {ozmaps} package to download a nice map of Australia, plot it with `sf::geom_sf()`, and add our observation points on top with `geom_point()`. We can separate the colours of our points by setting `colour = scientificName` within the `aes()` of `geom_point()`.

> **Note**
>
> You will need to add an email address registered with the Atlas of Living Australia in `galah_config()` to download species information.

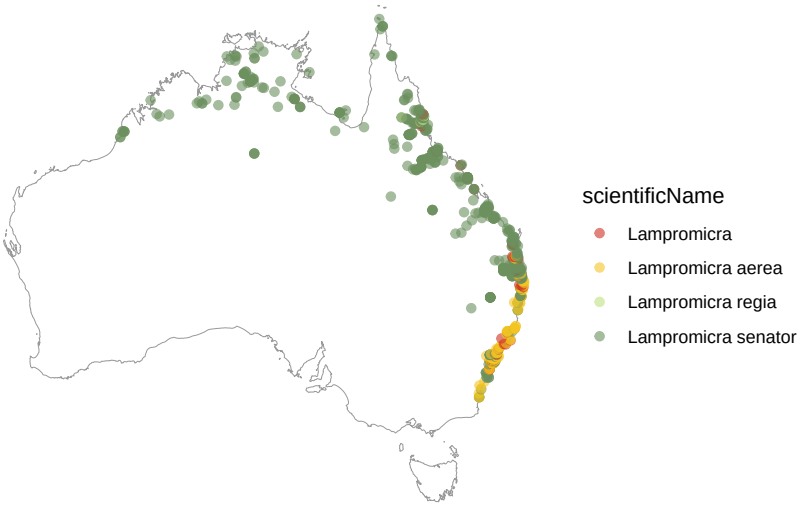
```r
# old fonti code. not sure why it's saved?
# beatles <- open_dataset("data/galah/lampromicra") |> collect()
library(ozmaps)
library(sf)
library(ggplot2)
library(paletteer) # colour palettes

# Download data
galah_config(email = "oliviajane.t@hotmail.com")

beetles <- galah_call() |>
  galah_identify("lampromicra") |>
  atlas_occurrences() |>
  tidyr::drop_na() # remove any NA values

# Get map of australia, set to correct projection for data
aus <- st_transform(ozmaps::ozmap_country, 4326)

# Map
ggplot() +
  geom_sf(data = aus,
          colour = "grey60",
          fill = "white",
          alpha = 0.2) +
  geom_point(data = beetles,
             mapping = aes(x = decimalLongitude,
                           y = decimalLatitude,
                           colour = scientificName),
             size = 1.8,
             alpha = 0.6) +
  scale_color_paletteer_d("feathers::eastern_rosella") +
  coord_sf(xlim = c(110, 155),
           ylim = c(-45, -10)) +
  theme_void()
```



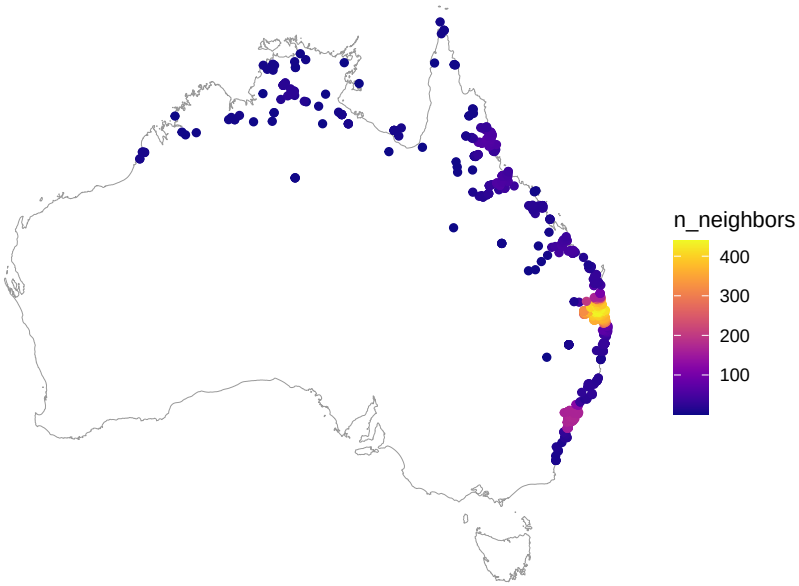
Plotting our points shows us that observations are spread along the northern and eastern coasts of Australia. We can also see that some observations are only identified to the genus level (e.g. *Lampromicra*), rather than to a specific species (e.g. *Lampromicra aerea*).

There are several places on the east coast of Australia where there are clumps of overlapping points. It's difficult to tell how many observations there really are in those areas. To investigate, we can recreate this into a point density plot using the {ggpointdensity package}.

```
library(ggpointdensity)

ggplot() +
  geom_sf(data = aus,
          colour = "grey60",
          fill = "white",
          alpha = 0.2) +
  geom_pointdensity(data = beetles,
                    mapping = aes(x = decimalLongitude,
                                  y = decimalLatitude)) +
  scale_color_paletteer_c("viridis::plasma") +
  coord_sf(xlim = c(110, 155),
           ylim = c(-45, -10)) +
```

```
  theme_void()
```



Adding the density of overlapping points to our map allows us to see that there is one area with many more observations—more than 400 observations are found in the light yellow area!

Using this information, we might decide to make our research question more specific to the region where there are the most records of *Lampromicra*.

Let's have a look at these records in the context of their IBRA bioregions (distinct areas defined on a common climate, geology, landform, native vegetation and species information).

To find out what region(s) the genus *Lampromicra* is most common, you can `group_by` the IBRA region field code in {galah} (use `search_fields` to see others).

```
ibra_counts <- galah_call() |>
  galah_identify("Lampromicra") |>
  galah_group_by("cl1048") |>    # IBRA regions
  atlas_counts()
```

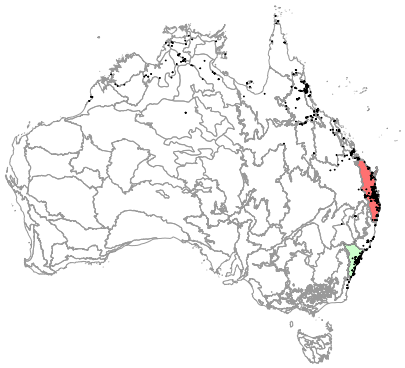
```
gt(head(ibra_counts))
```

cl1048	count
South Eastern Queensland	497
Sydney Basin	186
Victoria Bonaparte	81
Brigalow Belt North	74
Wet Tropics	71
Einasleigh Uplands	64

Looks like South Eastern Queensland has the most records followed by Sydney Basin.

This could be due to sampling bias in that Brisbane and Sydney are large metropolis areas. You might choose to investigate this bias further.

South Eastern Queensland (red), **Sydney Basin** (green)



```
shapefile <- st_read(here("data",
                          "shapefiles",
                          "IBRA7_regions",
                          "ibra7_regions.shp"),
                     quiet = TRUE) |>
  ms_simplify(keep = 0.1)


# South Eastern Queensland
ggplot() +
  geom_sf(data = shapefile %>% filter(REG_NAME_7 == "South Eastern Queensland"),
          aes(fill = "red"),
          colour = "grey60",
          alpha = 0.7) +
  geom_sf(data = shapefile %>% filter(REG_NAME_7 != "South Eastern Queensland"),
          aes(fill = "white"),
          colour = "grey60",
          alpha = 0.2) +
  geom_point(data = beetles,
             mapping = aes(x = decimalLongitude,
                           y = decimalLatitude),
             size = 0.5) +
```

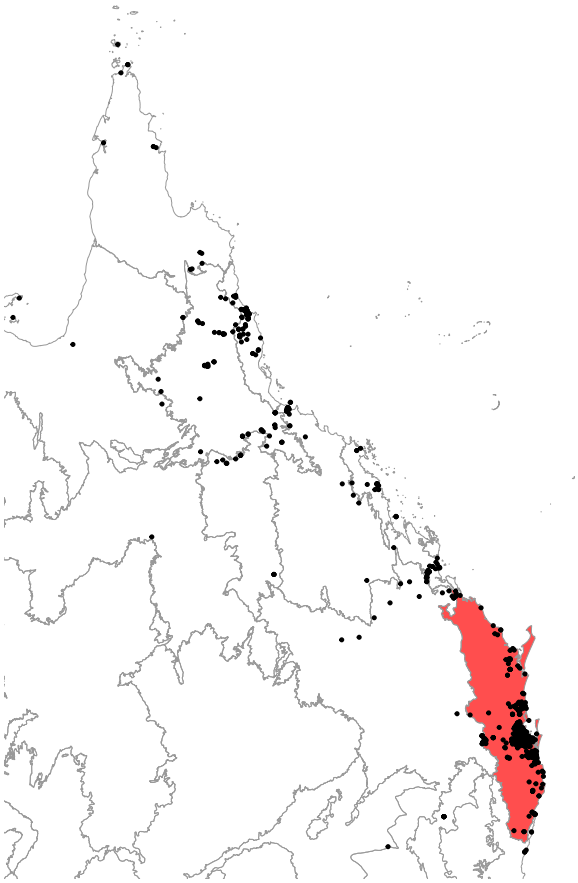
```
  coord_sf(xlim = c(140, 155),
           ylim = c(-30, -10)) +
  scale_fill_identity() +
  theme_void()
# Sydney Basin
ggplot() +
  geom_sf(data = aus,
          colour = "grey60",
          fill = "white",
          alpha = 0.2) +
  geom_sf(data = shapefile %>% filter(REG_NAME_7 == "Sydney Basin"),
          aes(fill = "green"),
          colour = "grey60",
          alpha = 0.7) +
  geom_sf(data = shapefile %>% filter(REG_NAME_7 != "Sydney Basin"),
          aes(fill = "white"),
          colour = "grey60",
          alpha = 0.2) +
  geom_point(data = beetles,
             mapping = aes(x = decimalLongitude,
                           y = decimalLatitude),
             size = 0.5) +
  coord_sf(xlim = c(145, 155),
           ylim = c(-37, -30)) +
  scale_fill_identity() +
  theme_void()
```

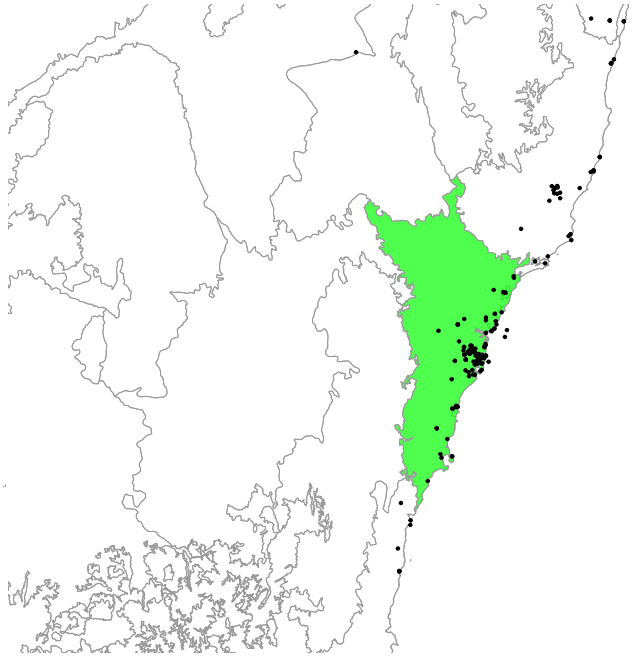
Alternatively, we might decide that there isn't enough data (or data of a good enough quality) to make accurate estimates about *Lampromicra*.

Depending on the spatial specificity of your question, you might have to adjust your data scope or your question accordingly.

2.5 Summary

This chapter has demonstrated some ways to initially investigate the data available to answer a research question using the {galah} package. This is just a small example of how you might





go about interogating the data to think critically about your data scope.

The next chapter will explain how you can download and save your data set to begin the first step of your analysis.

3 Download data

Once we have decided on our data scope, we can precede to download data. In this section, we will introduce a few common data infrastructures that offer open access biodiversity data and highlight some considerations when choosing one in the context of your data scope. We will then discuss some obstacles when consolidating data from multiple sources and the importance of metadata.

3.1 Where to get data from

3.1.1 Global

The Global Biodiversity Information Facility (GBIF), an international network and data infrastructure, stores global biodiversity data from many sources around the world. GBIF consists of a series of 'node' organisations who collate biodiversity data from their own regions and countries, with GBIF acting as an overarching organisation to store data from all nodes. Users that are interested in obtaining data that has global coverage may want to download directly from GBIF.

3.1.2 Regional

Alternatively, using a *regional node* may be more relevant if your project is at a smaller scale. For example, the Atlas of Living Australia (ALA) is the Australian node, Sistema de Informação sobre a Biodiversidade Brasileira (SiBBr) is the Brazilian node, and GBIF Sweden is the Swedish node. Living Atlases like the ALA ingest and aggregate data from a broad range of providers such as government monitoring programs, museums and herbaria, research projects and citizen science initiatives.

To see what national and regional nodes exist, check out this page.

Some regional nodes like the ALA and SiBBr use their own taxonomic backbone for classification, which might differ from the taxonomic backbone of other regional or global infrastructures like GBIF. These taxonomic classifications can be useful for classifying local flora and fauna, and accepted by experts of the region. It's worth being aware of what Section 3.2.1.1 are used to error caused by classification later on.

3.1.3 Data providers

If your project relates to data from a specific data provider, it might be best to download data directly from the source. For example, a common citizen science tool to collect species observations is iNaturalist. Downloading directly from the data provider may ensure you don't have any stray data from other sources.

3.2 How to download data

3.2.1 Using taxonomic information

3.2.1.1 Naming authorities

3.2.2 Choosing your taxonomic naming authority

Naming authorities are the people or organisations that create updated or revised lists of species, and where they fit on the taxonomic tree. Species names and taxonomic groups in these lists vary as there may be disagreements between lists about what distinguishes a new genus, species, subspecies etc. Taxonomy is hard!

Some data (especially from open source repositories) can contain data with conflicting taxonomies from different naming authorities. This difference can cause unexpected errors later in an analysis if you are unaware of which naming authorities were used in your data.

Choosing a naming authority is, then, one one way to make decisions surrounding taxonomic categorisations of biodiversity data.

Before starting your analysis, it's useful to:

- Be aware of which naming authority is accepted for your taxonomic group of interest.
- Check changes in the taxonomy of your focus species.
- Double check for updates, as most taxonomic society groups also release annual updates on taxonomy.

In the Atlas of Living Australia, the Australian Plant Name Index (APNI) is the primary naming authority for plants. With the Australian Faunal Directory (AFD) the main taxonomic catalog for animal species. For a full list, see the appendix.

Data that does not match the preferred naming authority or contains data from multiple naming authorities may need to be matched to a single taxonomic backbone .

Now that you've chosen a naming authority you can use it to ensure consistency across your data set, but also check you haven't missed any species data from your download.

3.2.2.1 Searching by classification

Include the original name that was recorded by the data provider in your download. This is often referred to as `verbatimScientificName`. or vernacularname

2. Include synonyms of the species names you're interested in your download

```
library(galah)

galah_call() |>
  galah_filter(year > 2019) |>
  atlas_counts()
```

```
# A tibble: 1 x 1
     count
     <int>
1 26671180
```

Many developers have created R packages to interact with each data infrastructures's API to aid access to biodiversity data. Here are a few examples, we recommend taking a look at each package's documentation to choose one that suits your project.

- `rgbif` an interface to GBIF
- `galah` an interface to a number living Atlases, as well as GBIF
- `rinat` an interface to iNaturalist observations
- `rebird` an interface with the eBird webservices.
- `spocc` a R package to query and collect species occurrence data from various sources including VertNet, iDigBio and others.

One benefit of using `galah` is that enables users to acquire not only species occurrence records but also taxonomic information, or associated media such as images or sounds. Below we have included some code blocks for downloading occurrence data with `galah` from GBIF and a Spain node.

3.2.2.2 GBIF data via `galah`

First, we have to configure `galah`. This is where you supply your account credentials and set the atlas to a particular region. See `?galah_config` for more configuration options. You can save these credentials in your `.Renviron` so you don't have to enter it explicitly in code.

We will be downloading all occurrences for the African Elephant from GBIF. This may take a while as it is around 12,000 records. Once downloaded, you can save the records locally in your desired format. For larger downloads, we recommend saving the data as `parquets` using `arrow::write_parquet`

```r
galah_config(email = Sys.getenv("ALA_EMAIL"),
             username = Sys.getenv("GBIF_USER"),
             password = Sys.getenv("GBIF_PWD"),
             atlas = "Global")

african_ele <- galah_call() %>%
  galah_identify("Loxodonta africana") %>%
  atlas_occurrences()

arrow::write_parquet(african_ele, "data/gbif/elephant")
```

```
# A tibble: 12,537 x 50
      gbifID datasetKey     occurrenceID kingdom phylum class order family genus
 *     <dbl> <chr>          <chr>        <chr>   <chr>  <chr> <chr> <chr>  <chr>
 1 924537719 95f132fe-f762~ 73e088f2-f8~ Animal~ Chord~ Mamm~ Prob~ Eleph~ Loxo~
 2 924537718 95f132fe-f762~ 73e0ad3c-f8~ Animal~ Chord~ Mamm~ Prob~ Eleph~ Loxo~
 3 924537717 95f132fe-f762~ 73f0f6ec-f8~ Animal~ Chord~ Mamm~ Prob~ Eleph~ Loxo~
 4 923926679 50c9509d-22c7~ http://www.~ Animal~ Chord~ Mamm~ Prob~ Eleph~ Loxo~
 5 923924124 50c9509d-22c7~ http://www.~ Animal~ Chord~ Mamm~ Prob~ Eleph~ Loxo~
 6 922237429 6ac3f774-d9fb~ <NA>         Animal~ Chord~ Mamm~ Prob~ Eleph~ Loxo~
 7 922237412 6ac3f774-d9fb~ <NA>         Animal~ Chord~ Mamm~ Prob~ Eleph~ Loxo~
 8 922237188 6ac3f774-d9fb~ <NA>         Animal~ Chord~ Mamm~ Prob~ Eleph~ Loxo~
 9 922237135 6ac3f774-d9fb~ <NA>         Animal~ Chord~ Mamm~ Prob~ Eleph~ Loxo~
10 922237121 6ac3f774-d9fb~ <NA>         Animal~ Chord~ Mamm~ Prob~ Eleph~ Loxo~
# i 12,527 more rows
# i 41 more variables: species <chr>, infraspecificEpithet <chr>,
#   taxonRank <chr>, scientificName <chr>, verbatimScientificName <chr>,
#   verbatimScientificNameAuthorship <chr>, countryCode <chr>, locality <chr>,
#   stateProvince <chr>, occurrenceStatus <chr>, individualCount <dbl>,
#   publishingOrgKey <chr>, decimalLatitude <dbl>, decimalLongitude <dbl>,
#   coordinateUncertaintyInMeters <dbl>, coordinatePrecision <dbl>, ...
```

3.2.2.3 Regional node via `galah`

In order to access data from the Australia node, we will need to reconfigure `galah` so that our query points to Australia. After that, we will download all records for the Pink Robin.

```r
galah_config(email = Sys.getenv("ALA_EMAIL"),
             atlas = "Australia")

pink_robin <- galah_call() %>%
  galah_identify("Petroica rodinogaster") %>%
  atlas_occurrences
```

```
# A tibble: 11,992 x 8
   decimalLatitude decimalLongitude eventDate           scientificName
 *           <dbl>            <dbl> <dttm>              <chr>
 1           -43.7             146. NA                  Petroica (Erythrodryas)~
 2           -43.7             146. NA                  Petroica (Erythrodryas)~
 3           -43.7             146. 1971-02-03 14:00:00 Petroica (Erythrodryas)~
 4           -43.7             146. 2020-10-07 02:39:00 Petroica (Erythrodryas)~
 5           -43.7             146. 2015-09-22 14:00:00 Petroica (Erythrodryas)~
 6           -43.6             146. 2002-01-03 13:00:00 Petroica (Erythrodryas)~
 7           -43.6             146. NA                  Petroica (Erythrodryas)~
 8           -43.6             146. 2020-12-27 13:00:00 Petroica (Erythrodryas)~
 9           -43.6             146. 2021-03-07 13:00:00 Petroica (Erythrodryas)~
10           -43.6             146. 2020-12-26 13:00:00 Petroica (Erythrodryas)~
# i 11,982 more rows
# i 4 more variables: taxonConceptID <chr>, recordID <chr>,
#   dataResourceName <chr>, occurrenceStatus <chr>
```

3.2.3 Using spatial information

3.2.3.1 Searching by region

One way to download data is by filtering to an area of interest using fields already in the ALA and {galah}.

3.2.3.2 Searching by bounding box

Another way to download data is by filtering to return observations within a bounding box.

This can be useful when you want to be sure to include records on the border or just outside the border of a defined region

(which might affect predictions of where species live in a model).

It can also be useful if you want to filter data yourself.

3.2.3.3 Searching with a shapefile

You can use a **shapefile** to filter observations. Shapefiles are [a file with points to make an outline, they can be simple or complex].

In R, using shapefiles requires using a package for handling spatial data like the {sf} package. [This is because the sf package helps transform & edit our shapefile to be plotted using longitude and latitude using a specified coordinate reference system - define.]

Here is a simple polygon we have constructed for a theoretical "site A". We use `st_as_sf()` and `st_set_crs()` from the convert our shapefile to a spatial object in R, then set its coordinate reference system. This allows us to plot this object with {ggplot2}.

Using shapefiles allows us to return data for very specific shapes, and are useful for long-term analyses of observations in specific regions or areas.

3.3 Choosing specific data columns

By default, `atlas_occurrences` will return a tibble with a selection of columns containing taxonomic and spatial data as well as other metadata. Alternatively, you can use `galah_select` to subset the columns that are relevant for your work. To see all available fields you can choose from:

```
show_all(fields)
```

```
# A tibble: 628 x 4
   id                  description                                      type  link
   <chr>               <chr>                                            <chr> <chr>
```

```
 1 _nest_parent_       <NA>                                              fiel~ <NA>
 2 _nest_path_         <NA>                                              fiel~ <NA>
 3 _root_              <NA>                                              fiel~ <NA>
 4 abcdTypeStatus      ABCD field in use by herbaria                     fiel~ <NA>
 5 acceptedNameUsage   http://rs.tdwg.org/dwc/terms/acceptedNameUsa~ fiel~ <NA>
 6 acceptedNameUsageID http://rs.tdwg.org/dwc/terms/acceptedNameUsa~ fiel~ <NA>
 7 accessRights        <NA>                                              fiel~ <NA>
 8 annotationsDoi      <NA>                                              fiel~ <NA>
 9 annotationsUid      <NA>                                              fiel~ <NA>
10 assertionUserId     User ID of the person who has made an assert~ fiel~ <NA>
# i 618 more rows
```

Here, we will choose a smaller subsets of 8 columns to download
for the Pink Robin

```
project_fields <- c("recordID",
                    "eventDate",
                    "year",
                    "basisOfRecord",
                    "occurrenceStatus",
                    "scientificName",
                    "decimalLatitude",
                    "decimalLongitude")

pink_robin_projfields <- galah_call() %>%
  galah_identify("Petroica rodinogaster") %>%
  galah_select(project_fields) %>%
  atlas_occurrences()
```

```
# A tibble: 11,992 x 8
   recordID             eventDate            year basisOfRecord occurrenceStatus
 * <chr>                <dttm>              <dbl> <chr>         <chr>
 1 000fc2ef-b696-4576-~ 1976-12-18 13:00:00  1976 HUMAN_OBSERV~ PRESENT
 2 001410fd-3a01-43aa-~ 1978-05-23 14:00:00  1978 HUMAN_OBSERV~ PRESENT
 3 00164a76-0b3a-4b44-~ 1977-12-15 13:00:00  1977 HUMAN_OBSERV~ PRESENT
 4 001bfca5-4197-40a6-~ 1940-12-31 14:00:00  1941 OBSERVATION   PRESENT
 5 001e33bd-bb15-4384-~ 2002-10-17 14:00:00  2002 HUMAN_OBSERV~ PRESENT
 6 0021cc7c-a371-4af6-~ 2016-04-09 14:00:00  2016 OCCURRENCE    PRESENT
 7 00287e24-87b8-429a-~ 2021-01-14 13:00:00  2021 HUMAN_OBSERV~ PRESENT
 8 002e05a7-b1bf-4eba-~ 2000-07-24 14:00:00  2000 OBSERVATION   PRESENT
```

```
 9 00327fc2-ae67-467b-~ 1983-11-23 13:00:00  1983 PRESERVED_SP~ PRESENT
10 0035016f-ae80-494a-~ NA                     NA HUMAN_OBSERV~ PRESENT
# i 11,982 more rows
# i 3 more variables: scientificName <chr>, decimalLatitude <dbl>,
#   decimalLongitude <dbl>
```

3.4 Refining your data download

Open access biodiversity data comes from many different data sources, (eg. government monitoring programs, museums, herbaria, research projects, citizen science apps). As such, data type and quality can vary considerably. For example, museums harbour older records that are associated with a preserved specimens. These data often contain lots of extra information (metadata) about a specific specimen and its location. Alternatively, data sourced from citizen science apps like iNaturalist or eBird are associated with images or sounds captured from a smart phone. These data might contain less metadata of each observation, but, thanks to phones' accurate location services, are quite accurate about the time and location of an observation.

Refining your download query ensures higher quality data and also reduces the download size as many data infrastructures impose constraints to download size. Below we have illustrated how you can refine your query a few quality measures using `galah_filter`.

3.4.0.1 By Year

Generally, old data records tend to be insufficient or less reliable as taxonomic knowledge and GPS tools were not readily available. For this reason, many users consider removing all occurrence records before a certain year to increase data precision (Gueta and Carmel 2016; J. R. Marsh et al. 2022) .

Choosing the year 'cut-off' is relatively arbitary, but the most commonly used year is 1945 (Zizka et al. 2020a; Führding-Potschkat, Kreft, and Ickert-Bond 2022), although some studies

discard all data collected before 1990 (Gueta and Carmel 2016; J. R. Marsh et al. 2022).

Here we will narrow the Pink Robin query from above to records after 1945 using `galah_filter`:

```
pink_robin_post1945 <- galah_call() %>%
  galah_identify("Petroica rodinogaster") %>%
  galah_filter(year > 1945) %>%
  atlas_occurrences()
```

3.4.0.2 Basis of record

Basis of record is a Darwin Core term that refers to the specific nature of the occurrence record. It can be used to refine your data download and ensure consistency when consolidating data from multiple organisations (Führding-Potschkat, Kreft, and Ickert-Bond 2022).

There are 6 different classes for basis of record:

- Living Specimen - a specimen that is alive, e.g. a living plant in a national park
- Preserved Specimen - a specimen that has been preserved, for example, a dried plant on an herbarium sheet
- Fossil Specimen - a preserved specimen that is a fossil
- Material Sample - a genetic or environmental sample
- Material Citation - A reference to, or citation of, a specimen in scholarly publications, e.g a citation of a physical specimen in a scientific journal
- Human Observation - an output of human observation process e.g. evidence of an occurrence taken from field notes or an occurrence without any physical evidence
- Machine Observation - An output of a machine observation process e.g. a photograph, a video, an audio recording, a remote sensing image or an occurrence record based on telemetry.

Depending on your data scope, it may be practical to limit data that can be traced to a physical specimen or observation (Godfree et al. 2021b), which we do for the Pink Robin below

```
tractable_records <- c("LIVING_SPECIMEN",
                       "PRESERVED_SPECIMEN",
                       "MATERIAL_SAMPLE",
                       "MACHINE_OBSERVATION")

pink_robin_tractable <- galah_call() %>%
  galah_identify("Petroica rodinogaster") %>%
  galah_filter(basisOfRecord == tractable_records) %>%
  atlas_occurrences()
```

3.4.0.3 Assertions

Data infrastructures use assertions to internally grade the quality, completeness and consistency of each occurrence record. Assertions take values of either 1 or 0, indicating the presence or absence of the data quality issue. Note that assertions may vary depending what atlas you have configured to. You can see the available assertions and their descriptions using:

```
show_all("assertions")
```

```
# A tibble: 114 x 4
   id                                description                 category type
   <chr>                             <chr>                       <chr>    <chr>
 1 AMBIGUOUS_COLLECTION              Ambiguous collection        Warning  asse~
 2 AMBIGUOUS_INSTITUTION             Ambiguous institution       Warning  asse~
 3 BASIS_OF_RECORD_INVALID           Basis of record badly form~ Warning  asse~
 4 biosecurityIssue                  Biosecurity issue           Error    asse~
 5 COLLECTION_MATCH_FUZZY            Collection match fuzzy      Warning  asse~
 6 COLLECTION_MATCH_NONE             Collection not matched      Warning  asse~
 7 CONTINENT_COUNTRY_MISMATCH        Continent country mismatch  Warning  asse~
 8 CONTINENT_DERIVED_FROM_COORDINATES Continent derived from coo~ Warning  asse~
 9 CONTINENT_INVALID                 Continent invalid           Warning  asse~
10 COORDINATE_INVALID                Coordinate invalid          Warning  asse~
# i 104 more rows
```

Once you have decided which assertions are important for your project you can further refine your download. To retrieve

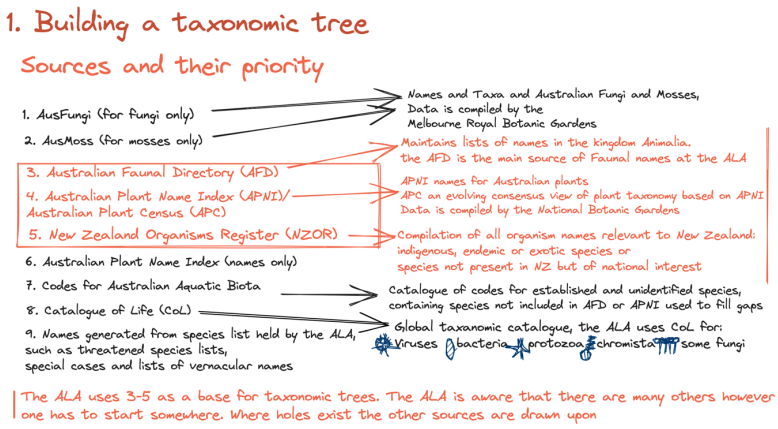
all the assertions for your query use `galah_select(group = "assertions")`

4 Appendix: Naming Authorities

Accurate species delimitation is also crucial for adequate conservation management and understanding evolutionary processes (Mace 2004). Species level lists are the foundation of many conservation decisions: such as the IUCN conservation status classification system. Taxonomic scope therefore has an effect on research application impacts.

Deciding what naming authority to use can be challenging. What you choose will depend on your own taxonomic research and evaluation, but also your research scope.

The Atlas of Living Australia uses the following priority list of naming authorities:



These authorities provide a list of accepted and authoritative names as a template. If you're unsure what naming authority to use and you're looking at Australian species, the APNI and the AFD are a good place to start, especially if the data you're investigating covers a wide range of taxa. If you're investigating specific taxa it's worth checking when the taxonomy was last

updated in the APNI or AFD, especially if you know there has been recent changes. If you want to investigate closer, we've provided some links to society groups, in some cases these can be more up to date that the APNI or AFD.

insert a mapped example of data changes when changing taxonomic naming authority?

5 Pre-cleaning

Pre-cleaning prepares the dataset(s) in a general manner to ensure logical and consistent formatting. It is a 'broad sweep' procedure that allows you to familiarise with the data, but it also makes the next stage of in-depth data cleaning proceed more smoothly (streamdna 2020). We will discuss some approaches on how to be curious with your data and how to detect and handle string inconsistencies or typography related errors, missing data, and outliers.

5.1 Metadata

Metadata describes your data set: it defines each variable and its contents. This might be describing what units a variable has been measured in, or, the climate the occurrence was collected in and whether it is a marked outlier. **Starting the process of pre-cleaning by briefing your metadata allows you to understand the kind of data you are working with and any potential biases that may limit what you can do with it**. Similarly, these biases may need to be of consideration when creating models or when used in your work more broadly.

Data infrastructures that use Darwin Core terms will have interoperable metadata. This makes it easier to consolidate across data sets. All Darwin Core term definitions can be found here, we suggest using `Ctrl/CMD F` and searching your variable name on the webpage. Don't hesitate to Google variable names if you are unsure what they represent.

It is also worth checking the metadata for the entire dataset to delineate if there is extra information about the data which may be relevant. You could Google the dataset name, or search the

dataset or institution on the ALA. The metadata on the ALA is submitted with the data, of which the ALA as a repository and not an owner cannot change. This means low-quality metadata cannot necessarily be vetoed.

An example of some good metadata is FrogID from the Australian Museum. From reading FrogID's metadata (Rowley and Callaghan 2020), you'll find:

1. The data is acoustic data, the majority of the species recorded are therefore male
2. Because this is citizen science data, it is especially biased towards populated areas
3. Audio is recorded via a smartphone app, the authors recommend if you require high coordinate precision to filter data to `geographic uncertainty of <3000m`
4. The data is presence only data

Metadata can also be useful for understanding the license that the data falls under. This is mostly relevant for using or republishing multimedia associated with the data.

5.2 Initial inspection

A great way to get an initial overview of your data is to use the R package `skimr`. Importantly `skimr` produces tables of descriptive statistics, such as amount of missing data, for every variable

The output is also grouped by data type (numeric, character, date) so you can also check for any inconsistencies. As you are looking through the output, ask yourself whether the data is in line with your expectations. For example:

If you requested data for a group of species, are they all represented?

Are the values for a variable reasonable? Looking at the quartiles will help you get the sense of the distribution of data.

These considerations will help you detect potential issues in the data.

```
library(skimr)

skim(african_ele)
```

Here is the `skimr` report for our African elephant dataset we downloaded earlier

Table 5.1: Data summary

Name	african_ele
Number of rows	12537
Number of columns	50
_____	
Column type frequency:	
character	31
logical	1
numeric	15
POSIXct	3
_____	
Group variables	None

Variable type: character

skim_variable	n_missing	complete_rate	min	max	empty	n_unique	whitespace
datasetKey	0	1.00	36	36	0	132	0
occurrenceID	708	0.94	1	81	0	11787	0
kingdom	0	1.00	8	8	0	1	0
phylum	0	1.00	8	8	0	1	0
class	0	1.00	8	8	0	1	0
order	0	1.00	11	11	0	1	0
family	0	1.00	12	12	0	1	0
genus	0	1.00	9	9	0	1	0
species	0	1.00	18	18	0	1	0
infraspecificEpithet	12410	0.01	8	13	0	2	0
taxonRank	0	1.00	7	10	0	3	0
scientificName	0	1.00	12	37	0	5	0
verbatimScientificName	3	1.00	12	53	0	32	0
verbatimScientificNameA	10964	0.13	2	31	0	255	0
countryCode	1461	0.88	2	2	0	44	0

skim_variable	n_missing	complete_rate	min	max	empty	n_unique	whitespace
locality	9211	0.27	3	254	0	686	0
stateProvince	3573	0.72	3	43	0	182	0
occurrenceStatus	0	1.00	6	7	0	2	0
publishingOrgKey	0	1.00	36	36	0	102	0
basisOfRecord	0	1.00	10	19	0	9	0
institutionCode	1325	0.89	2	76	0	103	0
collectionCode	1353	0.89	1	41	0	164	0
catalogNumber	1553	0.88	1	36	0	10866	0
recordNumber	12420	0.01	1	37	0	77	0
identifiedBy	4646	0.63	2	81	0	1464	0
license	0	1.00	7	12	0	3	0
rightsHolder	4240	0.66	2	56	0	1634	0
recordedBy	2138	0.83	1	160	0	1972	0
establishmentMeans	12249	0.02	6	6	0	1	0
mediaType	5021	0.60	5	16	0	4	0
issue	397	0.97	15	191	0	130	0

Variable type: logical

skim_variable	n_missing	complete_rate	mean	count
typeStatus	12537	0	NaN	:

Variable type: numeric

skim_variable	n_missing	complete_rate	mean	sd	p0	p25	p50	p75	p100	hist
gbifID	0	1.00	2535[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
individualCount	0226	0.18	4.81	14.03	0.00	1.00	1.00	2.00	[illegible]	[illegible]
decimalLatitude	1350	0.89	-10.01	14.67	-34.58	-24.06	-6.25	01	[illegible]	[illegible]
decimalLongitude	1350	0.89	25.56	12.35	-122.33	-22.97	31.08	3.48	[illegible]	[illegible]
coordinateUncertaintyInMeters	3966	0.68	4163	[illegible]	3.45	297	775.30	1580.30	[illegible]	[illegible]
coordinatePrecision	12538	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	[illegible]
elevation	12501	0.00	154.42	421.59	0.00	0.00	0.00	1.37	[illegible]	[illegible]
elevationAccuracy	12507	0.00	0.83	3.73	0.00	0.00	0.00	0.00	[illegible]	[illegible]
depth	12509	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	[illegible]

skim_variable	n_missing	complete_rate	mean	sd	p0	p25	p50	p75	p100	hist
depthAccuracy	12509	0.00	0.00	0.00	0.00	0.00	0.00	[illegible]	[illegible]	
day	1254	0.90	16.27	8.74	1.00	9.00	17.00	2.400000e+01	3.100000e+01	
month	1191	0.91	6.92	3.33	1.00	4.00	7.00	1.000000e+01	1.200000e+01	
year	997	0.92	2011.05	19.34	1799.00	2011.00	2016.00	2.019000e+03	2.023000e+03	
taxonKey	0	1.00	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	
speciesKey	0	1.00	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	

Variable type: POSIXct

skim_variable	n_missing	complete_rate	min	max	median	n_unique
eventDate	997	0.92	1799-01-01 00:00:00	2023-02-07 09:17:14	2016-02-07 12:00:00	8324
dateIdentified	4150	0.67	1783-01-01 00:00:00	2023-02-07 21:51:36	2020-04-16 19:35:25	7242
lastInterpreted	0	1.00	2023-01-24 14:57:46	2023-02-14 01:52:09	2023-02-13 16:01:27	10885

5.3 Structural inconsistencies

5.3.1 String inconsistencies and typographical errors

String inconsistencies include misspellings, capitalisation errors, misplaced punctuation or trailing white spaces. We will use the `janitor` R package to explore whether our data has any of these issues. The function `tabyl` will compute a counts and percent of total rows for each unique value.

We recommend `tabyl-ing` any character strings that are relevant to your project. For example, here is the `stateProvince` in alphabetical order.

```
library(janitor)
```

```r
african_ele %>%
  pull(stateProvince) %>%
  tabyl() %>%
  tibble() %>%
  print(n = 20)
```

```
# A tibble: 183 x 4
   .                     n   percent valid_percent
   <chr>             <int>     <dbl>         <dbl>
 1 Agadez                1 0.0000798      0.000112
 2 Al Qahirah            1 0.0000798      0.000112
 3 Alibori             601 0.0479         0.0670
 4 Arusha              231 0.0184         0.0258
 5 Arusha Region         1 0.0000798      0.000112
 6 Atacora             366 0.0292         0.0408
 7 Atakora             239 0.0191         0.0267
 8 Balaka                7 0.000558       0.000781
 9 Bassila               1 0.0000798      0.000112
10 Batha                 1 0.0000798      0.000112
11 Bauchi                3 0.000239       0.000335
12 Bengo                 3 0.000239       0.000335
13 Borgou                7 0.000558       0.000781
14 Budongo Forest        1 0.0000798      0.000112
15 Bushenyi             61 0.00487        0.00680
16 Cabo Delgado          2 0.000160       0.000223
17 Cape Prov.            2 0.000160       0.000223
18 Cape Province         1 0.0000798      0.000112
19 Central             113 0.00901        0.0126
20 Central Equatoria     2 0.000160       0.000223
# i 163 more rows
```

From the `tabyl` output, we can see there are few different variations of `Province`, `Prov.`, `Prov`. As an example, we will correct these with the `tidyverse` packages `stringr`, `dplyr`, `tidyr` as well as `glue`. If you are not very familiar with regular expressions, we highly recommend this cheatsheet

```r
library(tidyverse)
library(glue)
```

```
  # Create a regular expression to match Prov. and Prov
  # The pattern below means Prov that is NOT followed by any lowercase letters
  pattern = regex("Prov(?![:lower:])")

  # Use `str_subset` to pull out the cases that match our pattern
  # Confirm that these are the problematic ones
  # Assign these into an object
  str_subset(african_ele$stateProvince, pattern = pattern)
```

```
 [1] "Cape Prov."        "Cape Prov."        "West Nile Prov."
 [4] "Central Prov"      "Central Prov"      "Coastal Prov"
 [7] "Northeastern Prov" "Central Prov"      "Eastern Prov"
[10] "Coastal Prov"
```

```
  typos_provinces <- str_subset(african_ele$stateProvince, pattern = pattern)

  # Create a new variable `stateProvince_clean` using `mutate`, `if_else`, `str_detect` and `g
  # `str_detect` will evaluate values of `stateProvince` that matches our pattern we defined e
  # Matches will return TRUE, non-matches will return FALSE.
  # The `if_else` will then evaluate these logicals (TRUE/FALSE/NA)
  # for TRUE values, the `glue` function will take the first part of the province name enclose
  # for FALSE values , it will just take the corresponding value in stateProvince
  # Note that we are assigning these changes to a new object (`african_ele_2`)
  african_ele_2 <- african_ele %>%
    mutate(stateProvince_clean = if_else(str_detect(stateProvince, pattern = pattern),
                                         true = glue('{word(stateProvince, sep = " P")} Provinc
                                         false = stateProvince)
           )

  # Once we've made the correction we want to check we've done it correctly.
  # ALWAYS CHECK YOUR CORRECTIONS
  # Use the `select` function to isolate columns that `starts_with` "stateProvince"
  # Use the `filter` function to subset our the problematic provinces
  african_ele_2 %>%
    select(starts_with("stateProvince")) %>%
    filter(stateProvince %in% typos_provinces)
```

```
# A tibble: 10 x 2
```

```
   stateProvince     stateProvince_clean
   <chr>             <glue>
 1 Cape Prov.        Cape Province
 2 Cape Prov.        Cape Province
 3 West Nile Prov.   West Nile Province
 4 Central Prov      Central Province
 5 Central Prov      Central Province
 6 Coastal Prov      Coastal Province
 7 Northeastern Prov Northeastern Province
 8 Central Prov      Central Province
 9 Eastern Prov      Eastern Province
10 Coastal Prov      Coastal Province
```

```
  # Its good practice to check the other values were not affected by your corrections
  # Here we are removing the NA with `drop_na` and subsetting unique rows with `distinct`
  african_ele_2 %>%
    select(starts_with("stateProvince")) %>%
    drop_na() %>%
    distinct()
```

```
# A tibble: 182 x 2
   stateProvince    stateProvince_clean
   <chr>            <glue>
 1 Southern         Southern
 2 Taita Taveta     Taita Taveta
 3 Mara             Mara
 4 Arusha           Arusha
 5 Simiyu           Simiyu
 6 Morogoro         Morogoro
 7 Mashonaland West Mashonaland West
 8 Mpumalanga       Mpumalanga
 9 KwaZulu-Natal    KwaZulu-Natal
10 Manicaland       Manicaland
# i 172 more rows
```

```
  # Final check
  # Check with the original code that detected the issue
  african_ele_2 %>%
    pull(stateProvince_clean) %>%
```

```
    tabyl() %>%
    tibble() %>%
    print(n = 20)
```

```
# A tibble: 181 x 4
   .                     n   percent valid_percent
   <glue>            <int>     <dbl>         <dbl>
 1 Agadez                1 0.0000798      0.000112
 2 Al Qahirah            1 0.0000798      0.000112
 3 Alibori             601 0.0479         0.0670
 4 Arusha              231 0.0184         0.0258
 5 Arusha Region         1 0.0000798      0.000112
 6 Atacora             366 0.0292         0.0408
 7 Atakora             239 0.0191         0.0267
 8 Balaka                7 0.000558       0.000781
 9 Bassila               1 0.0000798      0.000112
10 Batha                 1 0.0000798      0.000112
11 Bauchi                3 0.000239       0.000335
12 Bengo                 3 0.000239       0.000335
13 Borgou                7 0.000558       0.000781
14 Budongo Forest        1 0.0000798      0.000112
15 Bushenyi             61 0.00487        0.00680
16 Cabo Delgado          2 0.000160       0.000223
17 Cape Province         3 0.000239       0.000335
18 Central             113 0.00901        0.0126
19 Central Equatoria     2 0.000160       0.000223
20 Central Province      4 0.000319       0.000446
# i 161 more rows
```

There are some other issues that can be corrected in a similar approach:

- `North West`, `North West District` and `North-Western`
- `Àfrica Central`, `Central Province` and `Central`
- `Atacora` and `Atakora`
- `Coastal Province` and `Coastal`

We recommend consulting reputable sources that can help delineate or consolidate similar values. Googling and looking at Wikipedia's sources are good places to find resources that you can verify accepted state and province names.

6 Taxonomy

Advances in taxonomy, especially in molecular biology has allowed researchers to describe new species more efficiently than ever before (Garraffoni et al. 2019). Modern approaches has also enabled reclassification of organisms that have been incorrectly described in the past. Unfortunately, multiple names (synonyms) for the same organism can arise when taxonomy is not unanimously agreed upon by researchers.

Harmonising taxonomic names is a prevalent and complex issue and so far, no unifying solution has been put forward — making research with biodiversity data challenging. A potential solution would require pulling together domain knowledge from experts and compiling a database for where taxonomic history for describe species is traceable and linked with published literature.

While there is no perfect solution, some tips, tricks and tools do exist. In this chapter we will go through some of these to clean taxonomic data and deal with synonyms.

6.1 Taxonomy preclean

Similar to what we did in the previous chapter, we will apply a broad sweep pre-clean to taxonomic data. This will make dealing with synonyms go as smoothly as possible.

The process is to first identify the issue, correct it, check it, and then document the changes. The goal is to standardise and correct as many errors issues before removing records.

6.1.1 Capitalisation

Normally higher taxonomy are capitalised e.g. Myrtaceae or Aves. Capitalisation errors are usually quick to spot when you print the data object. Alternatively you can try using `str_subset` on columns you expect to have capital letters.

The code below subsets out unique values for the variable `class` that have upper case letters. Notice that no matches are found

```
library(tidyverse)

str_subset(unique(bees$class), "[:upper:]")
```

```
character(0)
```

We can confirm that there are no upper case matches by subsetting unique values that have lower case letters to see what is going on. This shows us that Insecta is inputted entirely in lowercase.

```
str_subset(unique(bees$class), "[:lower:]")
```

```
[1] "insecta"
```

We can correct the lower case formatting as below, remember to check the fix before overwriting/removing the erroneous column(s)

```
bees |>
  mutate(class_corrected = str_to_sentence(class)) |>
  select(starts_with("class"))
```

```
# A tibble: 1,139 x 2
   class   class_corrected
   <chr>   <chr>
 1 insecta Insecta
```

```
 2 insecta Insecta
 3 insecta Insecta
 4 insecta Insecta
 5 insecta Insecta
 6 insecta Insecta
 7 insecta Insecta
 8 insecta Insecta
 9 insecta Insecta
10 insecta Insecta
# i 1,129 more rows
```

```
bees_corrected <- bees |>
  mutate(class_corrected = str_to_sentence(class)) |>
  select(-class) |> # Remove erroreous column
  rename(class = class_corrected) # Rename corrected column as the new 'class'
```

6.1.2 Seperators

In a taxonomic data, separators such as, spaces and underscore are found in scientific names and are used to delineate the genus and species name. While it is personal choice which separator you use, it is good practice to be consistent with your choice. Consistency ensures that unique values of scientific name truly reflects unique species and not due to inconsistencies.

Try `tabyl-ing` your taxonomic columns to check if you have any inconsistencies first

```
library(janitor)

plants |>
  pull(scientific_name) |>
  tabyl() |>
  tibble()
```

```
# A tibble: 623 x 3
   `pull(plants, scientific_name)`         n  percent
   <chr>                               <int>    <dbl>
 1 Acacia asparagoides                     2 0.000962
```

```
 2 Acacia barakulensis                       1 0.000481
 3 Acacia barringtonensis                    2 0.000962
 4 Acacia beadleana                          1 0.000481
 5 Acacia betchei                            6 0.00289
 6 Acacia blayana                            2 0.000962
 7 Acacia brunioides                         1 0.000481
 8 Acacia brunioides subsp. brunioides       6 0.00289
 9 Acacia brunioides subsp. granitica        1 0.000481
10 Acacia bulgaensis                         3 0.00144
# i 613 more rows
```

Consistent taxonomic formatting may not be an issue if you are downloading data from one single source such as the ALA where scientific names are already formatted consistently e.g. "Moloch horridus". This may not be the case when consolidating data from multiple sources.

Below is code to create an underscore scientific name from one that is separated with a space. Remember to check your changes

```
plants_updated <- plants |>
  mutate(scientific_name_undersc = str_replace_all(scientific_name, " ", "_"))

plants_updated |>
  pull(scientific_name_undersc) |>
  tabyl() |>
  tibble()
```

```
# A tibble: 623 x 3
   `pull(plants_updated, scientific_name_undersc)`     n  percent
   <chr>                                           <int>    <dbl>
 1 Acacia_asparagoides                                 2 0.000962
 2 Acacia_barakulensis                                 1 0.000481
 3 Acacia_barringtonensis                              2 0.000962
 4 Acacia_beadleana                                    1 0.000481
 5 Acacia_betchei                                      6 0.00289
 6 Acacia_blayana                                      2 0.000962
 7 Acacia_brunioides                                   1 0.000481
 8 Acacia_brunioides_subsp._brunioides                 6 0.00289
```

```
 9 Acacia_brunioides_subsp._granitica                    1 0.000481
10 Acacia_bulgaensis                                     3 0.00144
# i 613 more rows
```

6.1.3 Higher taxonomy

Higher taxonomy such as phylum and class may be used to group species for analysis or data visualisations. Its important to check the spelling and formatting of these columns. Its always good to start with some a useful table of counts for each taxonomic level. Keep an eye out for spelling errors, formatting issues and missing data. Note that `NA` in the output represents missing

As an example:

```
library(tidyverse)
library(janitor)

plants |>
  pull(class) |>
  tabyl()
```

```
pull(plants, class)    n     percent valid_percent
        Cycadopsida    4 0.001924002   0.001937046
      Equisetopsida  202 0.097162097   0.097820823
     Lycopodiopsida   10 0.004810005   0.004842615
      Magnoliopsida 1822 0.876382876   0.882324455
          Pinopsida    2 0.000962001   0.000968523
     Polypodiopsida   25 0.012025012   0.012106538
               <NA>   14 0.006734007            NA
```

```
plants |>
  pull(order) |>
  tabyl() |>
  head()
```

```
pull(plants, order)   n     percent valid_percent
        Alismatales   6 0.002886003   0.002939735
```

```
           Apiales  36 0.017316017   0.017638413
      Asparagales  67 0.032227032   0.032827046
        Asterales 158 0.075998076   0.077413033
  Austrobaileyales   6 0.002886003   0.002939735
       Canellales   5 0.002405002   0.002449780
```

```
plants |>
  pull(genus) |>
  tabyl() |>
  tail()
```

```
pull(plants, genus)  n     percent valid_percent
              Viola  5 0.002405002   0.002501251
          Westringia 12 0.005772006   0.006003002
           Xanthosia  9 0.004329004   0.004502251
               Xyris  4 0.001924002   0.002001001
              Zieria 14 0.006734007   0.007003502
                <NA> 80 0.038480038            NA
```

Missing higher taxonomy

If you noticed you have missing data in these columns, you can usually back fill this information using your chosen naming authority or retrieving this information from a living atlas such as the ALA.

The code below demonstrates how you can isolate the `scientific_names` of taxa with missing data and searching for taxonomic information from ALA

```
library(galah)

# Configure galah to point to Australia node
galah_config(atlas = "Australia",
             email = Sys.getenv("ALA_EMAIL"))

# These are the taxa missing `class` information
to_search <- inverts |>
  filter(is.na(class)) |>
```

```
  select(scientific_name) |>
  distinct()

# Reformat scientific_name to scientificName as the latter is the ALA format
backfilled_taxa <- to_search|>
  rename(scientificName = scientific_name) |>
  search_taxa(to_search) |> tibble()

backfilled_taxa
```

```
# A tibble: 818 x 15
   search_term     scientific_name scientific_name_auth~1 taxon_concept_id rank
   <chr>           <chr>           <chr>                  <chr>            <chr>
 1 Idiosoma manst~ Idiosoma manst~ (Pocock, 1897)         https://biodive~ spec~
 2 Holonuncia rec~ Holonuncia rec~ Hunt, 1992             https://biodive~ spec~
 3 Trachycosmus s~ Trachycosmus s~ Simon, 1893            https://biodive~ spec~
 4 Pseudotyrannoc~ Pseudotyrannoc~ Beier, 1971            https://biodive~ spec~
 5 Phryganoporus ~ Phryganoporus ~ (L. Koch, 1872)        https://biodive~ spec~
 6 Latrodectus ha~ Latrodectus ha~ Thorell, 1870          https://biodive~ spec~
 7 Ascoschoengast~ Ascoschoengast~ (Hirst, 1915)          https://biodive~ spec~
 8 Chrestobunus f~ Chrestobunus f~ Hickman, 1958          https://biodive~ spec~
 9 Supunna picta   Nyssus colorip~ Walckenaer, 1805       https://biodive~ spec~
10 Tasmanicosa le~ Tasmanicosa le~ (Thorell, 1870)        https://biodive~ spec~
# i 808 more rows
# i abbreviated name: 1: scientific_name_authorship
# i 10 more variables: match_type <chr>, kingdom <chr>, phylum <chr>,
#   class <chr>, order <chr>, family <chr>, genus <chr>, species <chr>,
#   issues <chr>, vernacular_name <chr>
```

Insufficient taxonomic rank

If a record is not identified down to the taxonomic level that needed for the study e.g. species, then the record should be removed.

During your data download, ensure you have requested for the column `taxonRank`, this variable tells us the lowest level of `scientificName`.

```r
library(galah)

galah_config(email = Sys.getenv("ALA_EMAIL"),
             atlas = "Australia")

honeyeaters <- galah_call() |>
  galah_identify("Meliphagidae") |>
  galah_filter(year == 2012 & stateProvince == "New South Wales") |>
  galah_select(group = "basic", taxonRank) |>
  atlas_occurrences()

honeyeaters$taxonRank |> unique()

honeyeaters |> filter(taxonRank == "species")
```

```r
library(arrow)
library(dplyr)

# honeyeaters <- galah_call() |>
#   galah_identify("Meliphagidae") |>
#   galah_filter(year == 2012 & stateProvince == "New South Wales") |>
#   galah_select(group = "basic", taxonRank) |>
#   atlas_occurrences()

# write_parquet(honeyeaters, "data/galah/honeyeater")

honeyeaters <- open_dataset("data/galah/honeyeater") |> collect()

honeyeaters$taxonRank |> unique()
```

```
[1] "species"    "genus"      "subgenus"   "subspecies" "family"
```

```r
honeyeaters |> filter(taxonRank == "species")
```

```
# A tibble: 43,684 x 9
   decimalLatitude decimalLongitude eventDate           scientificName
             <dbl>            <dbl> <dttm>              <chr>
 1           -37.4             150. 2012-09-26 14:00:00 Meliphaga (Meliphaga) l~
```

```
 2           -37.4             150. 2012-09-26 14:00:00 Acanthorhynchus tenuiro~
 3           -37.4             150. 2012-04-07 14:00:00 Acanthorhynchus tenuiro~
 4           -37.4             150. 2012-04-07 14:00:00 Nesoptilotis leucotis
 5           -37.4             150. 2012-04-07 14:00:00 Phylidonyris (Meliornis~
 6           -37.4             150. 2012-04-07 14:00:00 Phylidonyris (Phylidony~
 7           -37.4             150. 2012-04-07 14:00:00 Nesoptilotis leucotis
 8           -37.4             150. 2012-04-07 14:00:00 Phylidonyris (Meliornis~
 9           -37.4             150. 2012-04-07 14:00:00 Melithreptus (Melithrep~
10           -37.4             150. 2012-04-07 14:00:00 Acanthorhynchus tenuiro~
# i 43,674 more rows
# i 5 more variables: taxonConceptID <chr>, recordID <chr>,
#   dataResourceName <chr>, occurrenceStatus <chr>, taxonRank <chr>
```

Inconsistent higher taxonomy

A great approach to detect inconsistencies in your taxonomic data is to compute counts for each level of taxonomic rank. These counts act as a check for you to verify that the data is in line with your expectation. This is particularly important when combining data from different sources where their taxonomy might vary. If you have detected inconsistencies as we have done below, you will have to correct accordingly, either by consulting a taxonomic expert or a naming authority and ensure this is reported in your methods.

```
# Get counts for every species where they have more than 1 class
plants |>
  select(phylum:species, scientific_name) |>
  distinct() |>
  group_by(species) |>
  summarise(n_class = length(unique(class))) |>
  filter(n_class > 1)
```

```
# A tibble: 1 x 2
  species               n_class
  <chr>                   <int>
1 Allocasuarina distyla       2
```

```r
# Get the species that have more than 1 class
inconsistent_taxa <- plants |>
  select(phylum:species, scientific_name) |>
  distinct() |>
  group_by(species) |>
  summarise(n_class = length(unique(class))) |>
  filter(n_class > 1) |>
  pull(species)

# Filter species that have more than 1 class
plants |> filter(species %in% inconsistent_taxa) |>
  select(phylum:species, scientific_name) |>
  arrange(species) |>
  distinct()
```

```
# A tibble: 2 x 7
  phylum       class         order   family         genus species scientific_name
  <chr>        <chr>         <chr>   <chr>          <chr> <chr>   <chr>
1 Tracheophyta Magnoliopsida Fagales Casuarinaceae Allo~ Alloca~ Allocasuarina ~
2 Tracheophyta Equisetopsida Fagales Casuarinaceae Allo~ Alloca~ Allocasuarina ~
```

6.2 Synonyms

Synonyms is a complex issue when working with open source biodiversity data. Data infrastructures have their own taxonomic systems which may not align with researchers' view or consistent with your chosen naming authority.

Keeping in mind that there is no universal solution to synonymy. Best practice is to flag and correct synonyms in a clear and consistent manner. We recommend being explicit with your decisions about which names are retained and keeping a good record of the changes to aid transparency and reproducibility.

{worrms}

The {worrms} is the R interface to the World Register of Marine Species and has a ability to cross check synonyms with

their database for taxa that has an `AphiaID`. The function will return synonymous record(s) associated with another different AphiaID.

```
library(worrms)

marine_sp <- read_csv("data/worms/worms.csv")

marine_sp |>
  slice(7) |>
  pull(AphiaID) |>
  wm_synonyms()
```

```
# A tibble: 1 x 27
  AphiaID url   scientificname authority status unacceptreason taxonRankID rank
    <int> <chr> <chr>          <chr>     <chr>  <lgl>                <int> <chr>
1  453207 http~ Goniosoma ina~ Walker, ~ super~ NA                     220 Spec~
# i 19 more variables: valid_AphiaID <int>, valid_name <chr>,
#   valid_authority <chr>, parentNameUsageID <int>, kingdom <chr>,
#   phylum <chr>, class <chr>, order <chr>, family <chr>, genus <chr>,
#   citation <chr>, lsid <chr>, isMarine <int>, isBrackish <int>,
#   isFreshwater <int>, isTerrestrial <int>, isExtinct <int>, match_type <chr>,
#   modified <chr>
```

{taxize}

{taxize} allows users to search over many taxonomic data sources for species names (scientific and common) to resolve synonymy. The `gnr_resolve()` function matches your supplied list with up to 118 data sources including GBIF, Catalogue of Life, World Register of Marine Species and many more. The function scores how well matched your name is to these sources.

```
library(taxize)

# Read in a naming authority list
afd <- read_csv("data/naming/afd.csv")
unique(afd$VALID_NAME)
```

```
 [1] "Prosphaerosyllis battiri"         "Clavellopsis parasargi"
 [3] "Platypontonia hyotis"             "Palirhoeus eatoni"
 [5] "Diastylis kapalae"                "Xenobates chinai"
 [7] "Paratanais gaspodei"              "Paradexamine flindersi"
 [9] "Prostebbingia brevicornis"        "Cythere lactea"
[11] "Cythere melobesioides"            "Achelia transfugoides"
[13] "Halobates (Halobates) acherontis" "Quadraceps hopkinsi apophoretus"
[15] "Anabarhynchus striatus"           "Australocytheridea vandenboldi"
[17] "Enigmaplax littoralis"            "Hyphalus insularis"
[19] "Plesiopenaeus armatus"            "Uroptychus brucei"
[21] "Caligus dasyaticus"               "Coralliophila tetragona"
[23] "Triphora alveolata"               "Clavus obliquatus"
[25] "Naria beckii"                     "Pharaonella rostrata"
[27] "Lasaea australis"                 "Cadulus rudmani"
[29] "Bembicium flavescens"             "Mormula philippiana"
[31] "Turbonilla tiara"                 "Chlorodiloma crinita"
[33] "Mitrella merita"                  "Tritonoharpa antiquata"
[35] "Mauritia depressa dispersa"       "Laevidentalium zeidleri"
[37] "Conus (Harmoniconus) musicus"     "Marionia cyanobranchiata"
[39] "Tucetona flabellata"              "Neochromadora bilineata"
[41] "Desmoscolex membranosus"          "Echeneidocoelium indicum"
[43] "Indodidymozoon suttiei"           "Diploproctodaeum yosogi"
[45] "Pseudopecoelus japonicus"         "Pedibothrium lloydae"
[47] "Amphitethya stipitata"            "Pseudosuberites mollis"
[49] "Psammochela psammodes"
```

```
# Resolve names
resolved <- gnr_resolve(unique(afd$VALID_NAME), best_match_only = TRUE)
resolved |> print(n = 50)
```

```
# A tibble: 49 x 5
   user_supplied_name        submitted_name matched_name data_source_title score
 * <chr>                     <chr>          <chr>        <chr>              <dbl>
 1 Prosphaerosyllis battiri  Prosphaerosyl~ Prosphaeros~ National Center ~ 0.988
 2 Clavellopsis parasargi    Clavellopsis ~ Clavellopsi~ uBio NameBank     0.988
 3 Platypontonia hyotis      Platypontonia~ Platyponton~ National Center ~ 0.988
 4 Palirhoeus eatoni         Palirhoeus ea~ Palirhoeus ~ Wikispecies       0.988
 5 Diastylis kapalae         Diastylis kap~ Diastylis k~ Encyclopedia of ~ 0.988
 6 Xenobates chinai          Xenobates chi~ Xenobates c~ Encyclopedia of ~ 0.988
```

7	Paratanais gaspodei	Paratanais ga~	Paratanais ~	Wikispecies	0.988
8	Paradexamine flindersi	Paradexamine ~	Paradexamin~	Encyclopedia of ~	0.988
9	Prostebbingia brevicornis	Prostebbingia~	Prostebbing~	Encyclopedia of ~	0.988
10	Cythere lactea	Cythere lactea	Cythere lac~	Encyclopedia of ~	0.988
11	Cythere melobesioides	Cythere melob~	Cythere mel~	Encyclopedia of ~	0.988
12	Achelia transfugoides	Achelia trans~	Achelia tra~	National Center ~	0.988
13	Halobates (Halobates) ac~	Halobates (ha~	Halobates (~	CU*STAR	0.999
14	Quadraceps hopkinsi apop~	Quadraceps ho~	Quadraceps ~	Catalogue of Lif~	0.999
15	Anabarhynchus striatus	Anabarhynchus~	Anabarhynch~	Encyclopedia of ~	0.988
16	Australocytheridea vande~	Australocythe~	Australocyt~	Encyclopedia of ~	0.988
17	Enigmaplax littoralis	Enigmaplax li~	Enigmaplax ~	Encyclopedia of ~	0.988
18	Hyphalus insularis	Hyphalus insu~	Hyphalus in~	Wikispecies	0.988
19	Plesiopenaeus armatus	Plesiopenaeus~	Plesiopenae~	Wikispecies	0.988
20	Uroptychus brucei	Uroptychus br~	Uroptychus ~	Wikispecies	0.988
21	Caligus dasyaticus	Caligus dasya~	Caligus das~	Encyclopedia of ~	0.988
22	Coralliophila tetragona	Coralliophila~	Coralliophi~	Encyclopedia of ~	0.988
23	Triphora alveolata	Triphora alve~	Triphora al~	uBio NameBank	0.988
24	Clavus obliquatus	Clavus obliqu~	Clavus obli~	Encyclopedia of ~	0.988
25	Naria beckii	Naria beckii	Naria beckii	National Center ~	0.988
26	Pharaonella rostrata	Pharaonella r~	Pharaonella~	Arctos	0.988
27	Lasaea australis	Lasaea austra~	Lasaea aust~	National Center ~	0.988
28	Cadulus rudmani	Cadulus rudma~	Cadulus rud~	Encyclopedia of ~	0.988
29	Bembicium flavescens	Bembicium fla~	Bembicium f~	National Center ~	0.988
30	Mormula philippiana	Mormula phili~	Mormula phi~	Encyclopedia of ~	0.988
31	Turbonilla tiara	Turbonilla ti~	Turbonilla ~	Encyclopedia of ~	0.988
32	Chlorodiloma crinita	Chlorodiloma ~	Chlorodilom~	National Center ~	0.988
33	Mitrella merita	Mitrella meri~	Mitrella me~	Encyclopedia of ~	0.988
34	Tritonoharpa antiquata	Tritonoharpa ~	Tritonoharp~	National Center ~	0.988
35	Mauritia depressa disper~	Mauritia depr~	Mauritia de~	National Center ~	0.999
36	Laevidentalium zeidleri	Laevidentaliu~	Laevidental~	Encyclopedia of ~	0.988
37	Conus (Harmoniconus) mus~	Conus (harmon~	Conus Linna~	Catalogue of Lif~	0.75
38	Marionia cyanobranchiata	Marionia cyan~	Marionia cy~	National Center ~	0.988
39	Tucetona flabellata	Tucetona flab~	Tucetona fl~	Encyclopedia of ~	0.988
40	Neochromadora bilineata	Neochromadora~	Neochromado~	National Center ~	0.988
41	Desmoscolex membranosus	Desmoscolex m~	Desmoscolex~	Encyclopedia of ~	0.988
42	Echeneidocoelium indicum	Echeneidocoel~	Echeneidoco~	Integrated Taxon~	0.988
43	Indodidymozoon suttiei	Indodidymozoo~	Indodidymoz~	National Center ~	0.988
44	Diploproctodaeum yosogi	Diploproctoda~	Diploprocto~	Encyclopedia of ~	0.988
45	Pseudopecoelus japonicus	Pseudopecoelu~	Pseudopecoe~	Integrated Taxon~	0.988
46	Pedibothrium lloydae	Pedibothrium ~	Pedibothriu~	Encyclopedia of ~	0.988
47	Amphitethya stipitata	Amphitethya s~	Amphitethya~	Wikispecies	0.988

```
48 Pseudosuberites mollis    Pseudosuberit~ Pseudosuber~ Encyclopedia of ~ 0.988
49 Psammochela psammodes     Psammochela p~ Psammochela~ Wikispecies       0.988
```

```
# Retrieve synonyms
tsn <- get_tsn(unique(afd$VALID_NAME)[1:5])
```

```
==  5 queries  ===============
   Not Found:  Prosphaerosyllis battiri
   Not Found:  Clavellopsis parasargi
   Found:  Platypontonia hyotis
   Not Found:  Palirhoeus eatoni
   Not Found:  Diastylis kapalae
==  Results  =================
```

- Total: 5
- Found: 1
- Not Found: 4

```
synonyms(tsn)
```

```
$<NA>
[1] NA

$<NA>
[1] NA

$`612530`
  sub_tsn acc_tsn   syn_author                   syn_name syn_tsn
1  612530  612530 Suzuki, 1971 Platypontonia pterostreae 1191962

$<NA>
[1] NA

$<NA>
[1] NA
```

6.3 Input from experts

Programmatic solutions for resolving synonymy can only go so far. Seeking validation from experts is sensible if your goal is to obtain a high quality species list. Museums or taxonomic societies are extensive sources of knowledge. Below we have provided a list of some of Australian taxonomic society groups.

6.3.1 Australian taxonomic society groups

VERTEBRATES

- Amphibians and reptiles - Australian Herpetological Society
- Birds - Birdlife Australia
- Fish - Australian Society for Fish Biology
- Mammals - The Australian Mammal Society

INVERTEBRATES

- Arachnology - Australasian Arachnological Society
- Entomology - Australian Entomological Society
- Malacology - The Malacological Society of Australasia
- Nematology - Australasian Association of Nematologists

6.3.2 Global taxonomy

- GBIF uses 100 different sources to assemble - their global taxonomic backbone
- Authoritative taxonomic information on plants, animals, fungi, and microbes - Integrated Taxonomic Information System, ITIS
- Global taxonomic catalogue

- Catalogue of Life

7 Taxonomy II

7.1 Joining datasets from different infrastructures

If you have downloaded data from different sources, you likely will need to collate your data into a singular database. It is important to make sure fields that have the same, are indeed the same variable, so prior to merging - take the time cross check the meta-data from different data providers (Ribeiro et al. 2022).

7.1.1 Variable names and case format

Data providers will have their own naming conventions for variables. For example, World Register of Marine Species uses a combination of lower case e.g scientific_name and camel case e.g isExtinct. While the naming authority - Australian Fauna Directory (AFD) uses upper, snake case e.g. SCIENTIFIC_NAME. What format you choose is a matter of personal preference, the key is to be consistent.

Here we will subset the variables we want, reformat them to the lower snake case names and then join by `VALID_NAME`

```
library(stringr)

habitat_data <- worms %>%
  select(valid_name, starts_with("is"))

names(habitat_data)[-1]
```

```
[1] "isMarine"      "isBrackish"    "isFreshwater"  "isTerrestrial"
[5] "isExtinct"
```

```
new_names <- str_split(names(habitat_data)[-1], pattern =  "(?<=[a-z])(?=[A-Z])", n = 2) %>%
  map(.x = .,
      .f = ~tolower(.x)) %>% # Convert to all lower case
    map(.x = .,
      .f = ~str_flatten(.x, "_")) %>%  # Bind the two seperated elements
  unlist()

new_names
```

```
[1] "is_marine"      "is_brackish"    "is_freshwater"  "is_terrestrial"
[5] "is_extinct"
```

```
# Replace old with new
names(habitat_data)[-1] <- new_names

# Join habitat data by VALID_NAME
afd %>% left_join(habitat_data, by = join_by(VALID_NAME == valid_name))
```

```
# A tibble: 49 x 37
   GROUP_NAME HIGHER_CLASSIFICATION    KINGDOM PHYLUM SUBPHYLUM SUPERCLASS CLASS
   <chr>      <chr>                    <chr>   <chr>  <chr>     <lgl>      <chr>
 1 ANNELIDA   Phylum ANNELIDA, Class ~ ANIMAL~ ANNEL~ <NA>      NA         POLY~
 2 ARTHROPODA Phylum ARTHROPODA, Subp~ ANIMAL~ ARTHR~ CRUSTACEA NA         MAXI~
 3 ARTHROPODA Phylum ARTHROPODA, Subp~ ANIMAL~ ARTHR~ CRUSTACEA NA         MALA~
 4 ARTHROPODA Phylum ARTHROPODA, Subp~ ANIMAL~ ARTHR~ HEXAPODA  NA         INSE~
 5 ARTHROPODA Phylum ARTHROPODA, Subp~ ANIMAL~ ARTHR~ CRUSTACEA NA         MALA~
 6 ARTHROPODA Phylum ARTHROPODA, Subp~ ANIMAL~ ARTHR~ HEXAPODA  NA         INSE~
 7 ARTHROPODA Phylum ARTHROPODA, Subp~ ANIMAL~ ARTHR~ CRUSTACEA NA         MALA~
 8 ARTHROPODA Phylum ARTHROPODA, Subp~ ANIMAL~ ARTHR~ CRUSTACEA NA         MALA~
 9 ARTHROPODA Phylum ARTHROPODA, Subp~ ANIMAL~ ARTHR~ CRUSTACEA NA         MALA~
10 ARTHROPODA Phylum ARTHROPODA, Subp~ ANIMAL~ ARTHR~ CRUSTACEA NA         OSTR~
# i 39 more rows
# i 30 more variables: SUBCLASS <chr>, SUPERORDER <chr>, ORDER <chr>,
#   SUBORDER <chr>, SUPERFAMILY <chr>, FAMILY <chr>, SUBFAMILY <chr>,
#   SUPERTRIBE <lgl>, TRIBE <chr>, SUBTRIBE <lgl>, GENUS <chr>,
```

```
#   SUB_GENUS <chr>, SPECIES <chr>, SUB_SPECIES <chr>, AUTHOR <chr>,
#   YEAR <dbl>, CHANGED_COMBINATION <chr>, VALID_NAME <chr>,
#   COMPLETE_NAME <chr>, SYNONYMS <chr>, CHANGED_COMB_NAMES <chr>, ...
```

One issue you might face is that higher taxonomy from different providers may not match. If this is the case, we suggest choosing the data provider with the higher taxonomy that is consistent with your naming authority and use it to back fill the higher taxonomy of the other data sources

```
higher_taxonomy <- inverts %>%
  select(scientificName) %>%
  distinct() %>%
  search_taxa()

higher_taxonomy
```

```
# A tibble: 36 x 15
   search_term     scientific_name scientific_name_auth~1 taxon_concept_id rank
   <chr>           <chr>           <chr>                  <chr>            <chr>
 1 Palirhoeus eat~ Palirhoeus eat~ (C.O. Waterhouse, 187~ https://biodive~ spec~
 2 Lasaea austral~ Lasaea austral~ (Lamarck, 1818)        https://biodive~ spec~
 3 Turbonilla tia~ Turbonilla tia~ May, 1911              https://biodive~ spec~
 4 Tucetona flabe~ Tucetona flabe~ (Tenison-Woods, 1878)  https://biodive~ spec~
 5 Achelia transf~ Achelia transf~ Stock, 1973            https://biodive~ spec~
 6 Coralliophila ~ Coralliophila ~ Kosuge, 1986           https://biodive~ spec~
 7 Amphitethya st~ Amphitethya st~ (Carter, 1886)         https://biodive~ spec~
 8 Australocyther~ Australocyther~ McKenzie, 1967         https://biodive~ spec~
 9 Plesiopenaeus ~ Plesiopenaeus ~ (Spence Bate, 1881)    https://biodive~ spec~
10 Chlorodiloma c~ Chlorodiloma c~ (Philippi, 1849)       https://biodive~ spec~
# i 26 more rows
# i abbreviated name: 1: scientific_name_authorship
# i 10 more variables: match_type <chr>, kingdom <chr>, phylum <chr>,
#   class <chr>, order <chr>, family <chr>, genus <chr>, species <chr>,
#   issues <chr>, vernacular_name <chr>
```

Remember to always check your changes after!

7.2 Extended taxonomic cleaning

Depending on your project's data scope, it may be necessary to remove certain groups of taxa. Below, we have provided a few examples. We will also briefly showcase `CoordinateCleaner` a useful R package for removing XX records.

7.2.1 Introduced or Invasive species

Remove non-native species: This step is a common requirement. A list can be obtained from the Global Register of Introduced and Invasive Species (GRIIS). The downloads are sorted my country. Once this list is read into R, you can proceed to exclude invasive species from your data as below:

```
library(tidyverse)
library(here)

griis_ls <- read_csv(here("data/lists/GRIIS_Australia_20230331-121730.csv"))

glimpse(griis_ls)
```

```
Rows: 2,979
Columns: 16
$ scientific_name                 <chr> "Oenothera longiflora L.", "Lampranth~
$ scientific_name_type            <chr> "species", "species", "species", "spe~
$ kingdom                         <chr> "Plantae", "Plantae", "Plantae", "Pla~
$ establishment_means             <chr> "alien", "alien", "alien", "alien", "~
$ is_invasive                     <chr> "null", "null", "null", "null", "null~
$ occurrence_status               <chr> "present", "present", "present", "pre~
$ checklist.name                  <chr> "Australia", "Australia", "Australia"~
$ checklist.iso_countrycode_alpha3 <chr> "AUS", "AUS", "AUS", "AUS", "AUS", "A~
$ accepted_name.species           <chr> "Oenothera longiflora", "Lampranthus ~
$ accepted_name.kingdom           <chr> "Plantae", "Plantae", "Plantae", "Pla~
$ accepted_name.phylum            <chr> "Tracheophyta", "Tracheophyta", "Trac~
$ accepted_name.class             <chr> "Magnoliopsida", "Magnoliopsida", "Ma~
$ accepted_name.order             <chr> "Myrtales", "Caryophyllales", "Erical~
$ accepted_name.family            <chr> "Onagraceae", "Aizoaceae", "Ericaceae~
$ accepted_name.habitat           <chr> "[\"terrestrial\"]", "[\"terrestrial\~
```

```
$ accepted_name                   <lgl> NA, NA, NA, NA, NA, NA, NA, NA, NA, N~
```

```
# Note which species matched with GRIIS list
matches <- plants |> filter(scientific_name %in% griis_ls$accepted_name.species)
matches
```

```
# A tibble: 2 x 13
  record_id    scientific_name vernacular_name kingdom phylum class order family
  <chr>        <chr>           <chr>           <chr>   <chr>  <chr> <chr> <chr>
1 94df1223-4c~ Lysimachia jap~ Creeping Loose~ Plantae Trach~ Magn~ Eric~ Primu~
2 8055e15e-36~ Lysimachia jap~ Creeping Loose~ Plantae Trach~ Magn~ Eric~ Primu~
# i 5 more variables: genus <chr>, species <chr>, subspecies <chr>,
#   latitude <dbl>, longitude <dbl>
```

```
# Exclude GRIIS matches
plants_no_griis <- plants |> filter(! scientific_name %in% matches)
```

7.2.2 Extinct species

In most cases, a year filter during in your download query should remove *most* extinct species.

You want to cross check for extinct species using the Interim Register of Marine and Nonmarine Genera (IRMNG). The list is comprehensive and actively maintained, the only caveat is that a lot of its data doesn't go down to species level. As such, we recommend using the following approach to find *potentially extinct* taxa and further investigate the records that we are flagged.

The required files are organised by year and can be downloaded from here. Once you have unzipped the file in your projected directory, we need to process the list a littlerbefore we use it to exclude extinct species.

```
library(data.table)
library(janitor)

# Taxonomic info
```

```
irmng_taxa <- fread("data/lists/IRMNG_genera_DwCA_2023-05-19/taxon.txt", na.strings = c(""),

# Species profile
irmng_sp <- fread("data/lists/IRMNG_genera_DwCA_2023-05-19/speciesprofile.txt", na.strings =
```

```
# Precleaning
awc_pattern <- "(awaiting allocation)"
insed_pattern <- "incertae sedis"

cleaned_irmng_taxa <- irmng_taxa |>
  mutate(class = ifelse(str_detect(class, pattern = paste0(awc_pattern,"|", insed_pattern)),
                        word(class), class) # Remove pesky values
         ) |>
  filter(taxonomicStatus == "accepted") |> # Filter to accepted names
  filter(kingdom %in% c("Animalia", "Plantae"),
         ! kingdom == "Questionable / non-biota (fossil)") # Filter to Animal and plants - c

# Join with species profile, remove pesky values and filter to extinct taxa
extinct_irmng <- irmng_taxa |>
  left_join(irmng_sp, by = "taxonID") |>
  filter(! scientificName == "Questionable / non-biota (fossil)") |>
  filter(isExtinct == TRUE)

# Summary of extinct species by taxonRank
extinct_irmng$taxonRank |> tabyl()
```

```
extinct_irmng$taxonRank     n      percent
                  Class    14 5.926177e-04
                 Family  1675 7.090247e-02
                  Genus 21718 9.193193e-01
             Infraclass     1 4.232983e-05
                  Order   210 8.889265e-03
                 Phylum     4 1.693193e-04
              Subclass     2 8.465967e-05
```

```
# Create genus
inverts <- inverts |>
  mutate(genus = word(scientificName, 1))
```

```
# Extract unique extinct genus and remove genus that have punctuation in them
extinct_genus <- extinct_irmng |>
  drop_na(genus) |>
  filter(!str_detect(genus, pattern = regex("[:punct:]"))) |>
  pull(genus) |>
  unique()

# Check if there are any matches at genus level
check <- inverts |>
filter(str_detect(genus, pattern = regex(paste0(extinct_genus, collapse="|")))) |>
  pull(scientificName) |>
  unique()

check
```

```
[1] "Halobates (Halobates) acherontis"
```

Alternatively, you can use the IUCN to retrieve a list of extinct species that are in their database

https://apiv3.iucnredlist.org/api/v3/docs#species-category
https://docs.ropensci.org/crul/articles/crul.html

```
library(rredlist)
library(skimr)

# Create IUCN token
rredlist::rl_use_iucn() # Application can take a day or two!
usethis::edit_r_environ() # Place the approved token in your R environment

extinct_iucn <- rl_sp_category('EX')
skim(extinct_iucn)

extinct_sp <- extinct_iucn$result |> tibble() # Note these are extinct species across the gl

# Find matches
inverts |> filter(scientificName %in% extinct_sp$scientific_name) # No matches
```

7.2.3 Certain lifestges

```
library(galah)
library(skimr)

galah_config(email = Sys.getenv("ALA_EMAIL"),
             atlas = "Australia")

bilby <- galah_call() |>
  galah_identify("Macrotis lagotis") |>
  galah_filter(year == 2022) |>
  galah_select(group = "basic", sex, lifeStage, reproductiveCondition) |>
  atlas_occurrences()

bilby |> filter(!sex == "MALE")
```

```
# A tibble: 35 x 11
   decimalLatitude decimalLongitude eventDate           scientificName
             <dbl>            <dbl> <dttm>              <chr>
 1           -34.2             143. 2022-06-16 14:00:00 Macrotis lagotis
 2           -34.2             143. 2022-06-14 14:00:00 Macrotis lagotis
 3           -34.2             143. 2022-03-22 13:00:00 Macrotis lagotis
 4           -34.2             143. 2022-03-22 13:00:00 Macrotis lagotis
 5           -34.2             143. 2022-03-22 13:00:00 Macrotis lagotis
 6           -34.2             143. 2022-03-23 13:00:00 Macrotis lagotis
 7           -34.2             143. 2022-03-21 13:00:00 Macrotis lagotis
 8           -34.2             143. 2022-06-12 14:00:00 Macrotis lagotis
 9           -34.2             143. 2022-03-23 13:00:00 Macrotis lagotis
10           -34.2             143. 2022-06-12 14:00:00 Macrotis lagotis
# i 25 more rows
# i 7 more variables: taxonConceptID <chr>, recordID <chr>,
#   dataResourceName <chr>, occurrenceStatus <chr>, sex <chr>, lifeStage <lgl>,
#   reproductiveCondition <chr>
```

7.2.4 Marine species

Remove specific taxa/ depending on the study: For example, if working with terrestrial data, it is necessary to remove marine taxa.

```r
library(worrms)

# Obtain species list
my_species <- inverts |>
  pull(scientificName) |>
  unique()

# Query WoRMs
marine_check <- map_dfr(my_species,
       possibly(~worrms::wm_records_name(name = .x) |> mutate(search_term = .x))
)

# Filter species that are TRUE for isMarine
marine_inverts <- marine_check |>
  filter(isMarine == TRUE) |>
  select(search_term)

# Exclude marine invertebrates
inverts |> filter(!scientificName %in% marine_inverts)
```

```
# A tibble: 2,637 x 9
   decimalLatitude decimalLongitude eventDate           scientificName
             <dbl>            <dbl> <dttm>              <chr>
 1           -46.9             37.8 1983-04-01 00:00:00 Palirhoeus eatoni
 2           -46.9             37.8 1984-09-01 00:00:00 Palirhoeus eatoni
 3           -46.9             37.9 1986-04-01 00:00:00 Palirhoeus eatoni
 4           -46.6             38.0 1985-04-01 00:00:00 Palirhoeus eatoni
 5           -46.6             38.0 1983-05-01 00:00:00 Palirhoeus eatoni
 6           -46.6             38.0 1984-09-01 00:00:00 Palirhoeus eatoni
 7           -46.6             38.0 1984-04-01 00:00:00 Palirhoeus eatoni
 8           -43.6            148.  NA                  Lasaea australis
 9           -43.6            147.  2008-12-28 00:00:00 Lasaea australis
10           -43.6            147.  2008-12-28 00:00:00 Lasaea australis
# i 2,627 more rows
# i 5 more variables: taxonConceptID <chr>, recordID <chr>,
#   dataResourceName <chr>, occurrenceStatus <chr>, genus <chr>
```

8 Spatial data

```
library(galah)

banksia_serrata <- galah_call() |>
  galah_identify("banksia_serrata") |>
  galah_filter(year > 2022) |>
  atlas_occurrences()
```

You've been through the taxonomic cleaning steps so now it's time to clean up the spatial elements. You may have flagged records as being taxonomically incorrect, it's important to keep those in mind as you go through the spatial cleaning steps as you might learn more about those records. We will discuss some different ways to check for spatial outliers as well as the removal of records in certain geographic areas known to be problematic.

8.1 Coordinate precision

Coordinate precision describes the consistency of values if one were to record the coordinates of the same location, multiple times. Coordinate precision can vary between data sources and recording equipment. For example, coordinates recorded with a GPS unit or a phone generally has higher precision compared to those manually determined from locality descriptions.

Coordinate precision below 100km represents the grain size of many macroecological analyses (Zizka et al. 2020b). Some studies have used a cut-off of spatial resolution >25,000m or precision with less than three decimal places (Godfree et al. 2021a). Rasterised collections often have a significant proportion of records that might have low coordinate precision.

Depending on the scope of your research question, you may need to limit your occurrence data to a certain level of coordinate.

We recommend first including `coordinatePrecision` in your download query and excluding its completeness and range before you exclude any data.

```
library(galah)
library(skimr)

banksia_serrata <- galah_call() |>
  galah_identify("banksia_serrata") |>
  galah_filter(year > 2022) |>
  galah_select(group = "basic", coordinatePrecision) |>
  atlas_occurrences()

# banksia_serrata |>
#   select(coordinatePrecision) |>
#   skim()

# Filter by number of decimal places
# banksia_serrata |>
#   filter(coordinatePrecision < XXX)
```

8.2 Coordinate Uncertainty

Alternatively, you can refine your data using coordinate uncertainty which describes the possible circular area in meters where the true location is in.

```
banksia_serrata <- galah_call() |>
  galah_identify("banksia_serrata") |>
  galah_filter(year > 2022) |>
  galah_select(group = "basic", coordinatePrecision, coordinateUncertaintyInMeters) |>
  atlas_occurrences()

# Filter by number of decimal places
# banksia_serrata |>
```

```
#   filter(coordinateUncertaintyInMeters < XXX)
```

8.2.1 Missing coordinate data

If your research question requires spatial information, then it may be useful to exclude records that are missing coordinates data. Many spatial analytical tools are not compatible with missing coordinate data. We recommend tallying and identifying the rows that have missing data before excluding.

You can use `drop_na()` to remove missing values from your dataset.

```
library(dplyr)

# Identify missing data in coordinates
banksia_serrata |>
  filter(is.na(decimalLatitude) | is.na (decimalLongitude))

# Excluding them
banksia_serrata |>
  drop_na(decimalLatitude, decimalLongitude)
```

8.3 Coordinate correction

Some of these steps may have been completed in a pre-cleaning step, however it's now time to be more rigorous. As always we'll start with fixing data before discarding, many coordinates issues can be solved with data manipulation instead of discarding:

Flipped coordinates: Flipped coordinates typically appear as a clustering of points, whereby swapping the latitude and longitude will place the coordinates where they are expected. (Jin and Yang 2020)

```
#example map of some flipped coordinates (what to look for)
# https://www.gbif.org/occurrence/3013406216 this has flipped coordinates, which GBIF has co
# https://www.gbif.org/occurrence/search?q=mammalia&continent=SOUTH_AMERICA&has_coordinate=t
```

Numerical sign confusion: As with flipped coordinates, if there is a clustering of points mirrored to another hemisphere, consider swapping the sign and correct rather than discarding the points.

```
#example map, like coordinates off the coast of japan

# https://biocache.ala.org.au/occurrences/search?q=lsid%3Ahttps%3A%2F%2Fid.biodiversity.org.

# eucs <- galah_call() %>%
#  galah_identify("Eucalyptus") %>%
#  galah_filter( year == 2005,
#             dataResourceName == "The University of Melbourne Herbarium (MELU) AVH data") %
#  atlas_occurrences()
```

Country field doesn't match coordinates: The coordinates could be wrong or just the country listed.

```
## this doesnt seem to be very common- atleast not in ALA data- because there is no neighbor
# https://biocache.ala.org.au/occurrences/a34fca43-9e7c-4b37-8fe4-07cc18369465 Australian co
# https://www.gbif.org/occurrence/search?advanced=true&continent=SOUTH_AMERICA&geometry=POLY
```

8.3.1 Quick visualisation

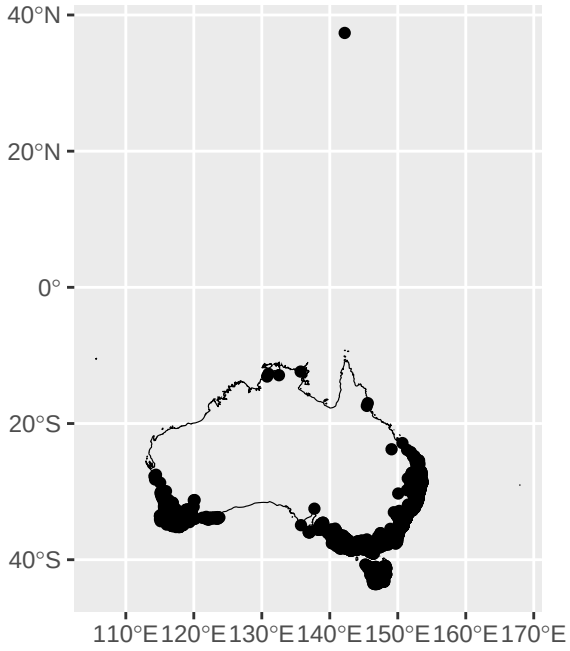
One of the most straightforward ways to check for spatial errors is to plot your data onto a map. More obvious spatial errors are much easier to spot visually.

```
library(ggplot2)
library(ozmaps)
library(sf)

# Retrieve map of Australia
aus <- st_transform(ozmap_country, 4326)
```

```r
# Remove missing coordinates in Banksia data
# Then transform into 'sf' object
banksia_sf <- banksia_serrata |>
  drop_na(starts_with("decimal")) |>
  st_as_sf(coords = c("decimalLongitude", "decimalLatitude"),
           crs = 4326)

# A quick plot
ggplot() +
  geom_sf(data = aus, colour = "black", fill = NA) +
  geom_sf(data = banksia_sf)
```



8.4 Coordinate cleaning

Once you have fixed everything you can, it's time to remove records that still have errors. This doesn't mean removing all outliers, you must have more than "it's far away from the others" to justify a records removal.

Remove records where longitude and latitude are

equal: High likelihood that this is not where the record was recorded and, check first, however likely will need to remove

Remove records with zero coordinates: When plotting it on a map, zero coordinates will be found around the point at zero latitudes and longitudes. These records will not accurately represent their valid location and must be removed.

```
#zero coordinates acacia

#https://biocache.ala.org.au/occurrences/search?q=lsid%3Ahttps%3A%2F%2Fid.biodiversity.org.a
```

8.5 Remove records plotted away from the known area of distribution of the species.

It is essential to check the metadata to ensure that it is a data entry error and not a real outlier. In some cases, it's worth checking the literature before discarding records like these. These can also be mis-identified species, if you're working with data from many species, and you find a species point in amongst the environmental bounds of a similar looking species it might be worth going back to the original record and taking a closer look. However, if no images exist it might be difficult to determine if it is a taxonomic or spatial issue.

8.5.1 Remove records with coordinates assigned to country and province centroids

Centroids are common when records are being assigned from georeferencing based on vague locality descriptions or from incorrect georeferencing. Sometimes, records are erroneously entered with the physical location of the specimen or because they represent individuals from captivity or grown in horticulture, which were not clearly labelled as such.

8.5.2 Remove records from biological institutions

such as botanic gardens, zoos, country capitals, biodiversity institutions, urban areas, and GBIF headquarters. In some cases these records will haven actually been recorded at a zoo for example, in other cases this is often incorrectly georeferenced records. They can be tricky to spot but there are a few packages that deal with centroid data. Exploratory visuals can also help support findings, making it easier to spot clusterings of points.

In a few cases, zoos and botanic gardens might be where the record was sighted. However, in this case, it is not naturally occurring and should be removed. Records in urban areas may not want to be removed by everyone, but it is essential to note that it could be old data or have vague locality descriptions.

Remove records outside of the country of interest: In some cases, records outside the country of origin may be outliers. In other cases, they may be perfectly valid. It is important to analyze case-by-case and remove the record if necessary.

8.5.3 CoordinateCleaner

9 Outliers

#note from meeting with Martin this section might include some alpha hull examples for how to detect outliers, or how they can skew your data, in addition to maybe an SDM for outlier detection (Simões and Peterson 2018) even if we won't be teaching people how to run an SDM

Outliers can be true outliers or data errors, true outliers are not necessarily to be removed. This could represent mis-identified specimens, etc

(#note to double check what i wrote for support article around inconsistent sampling and open source data)

- Alpha hull (some example species - check cailtin chat history)
- SDM Shandiya?
- Remove records with miss-identified taxonomy: Incorrectly identified species, or unrecognized species names compared to a naming authority, should be removed.

Incorrectly identified specimens can be difficult to identify with open source biodiversity data. Often these will be picked up by 1: an image of the species in question which does not match 2: If you notice a species outside of its geographic range, this could be a true outlier, it could be a spatial error, or it could be a different species. (**see — for more info**)

10 References

Führding-Potschkat, Petra, Holger Kreft, and Stefanie M. Ickert-Bond. 2022. "Influence of Different Data Cleaning Solutions of Point-Occurrence Records on Downstream Macroecological Diversity Models." *Ecology and Evolution* 12 (8): e9168. https://doi.org/10.1002/ece3.9168.

Garraffoni, André RS, Thiago Q Araújo, Anete P Lourenço, Loretta Guidi, and Maria Balsamo. 2019. "Integrative Taxonomy of a New Redudasys Species (Gastrotricha: Macrodasyida) Sheds Light on the Invasion of Fresh Water Habitats by Macrodasyids." *Scientific Reports* 9 (1): 2067.

Godfree, Robert C., Nunzio Knerr, Francisco Encinas-Viso, David Albrecht, David Bush, D. Christine Cargill, Mark Clements, et al. 2021a. "Implications of the 2019–2020 Megafires for the Biogeography and Conservation of Australian Vegetation." *Nature Communications* 12 (1): 1023. https://doi.org/10.1038/s41467-021-21266-5.

———, et al. 2021b. "Implications of the 2019–2020 Megafires for the Biogeography and Conservation of Australian Vegetation." *Nature Communications* 12 (1): 1023. https://doi.org/10.1038/s41467-021-21266-5.

Gueta, Tomer, and Yohay Carmel. 2016. "Quantifying the Value of User-Level Data Cleaning for Big Data: A Case Study Using Mammal Distribution Models." *Ecological Informatics* 34 (July): 139–45. https://doi.org/10.1016/j.ecoinf.2016.06.001.

Jin, Jing, and Jun Yang. 2020. "BDcleaner: A Workflow for Cleaning Taxonomic and Geographic Errors in Occurrence Data Archived in Biodiversity Databases." *Global Ecology and Conservation* 21 (March): e00852. https://doi.org/10.1016/j.gecco.2019.e00852.

Marsh, Jess, Payal Bal, Hannah Fraser, Kate Umbers, Aaron Greenville, Libby Rumpff, and John Woinarski. 2021. "Assessment of the Impacts of the 2019-20 Wildfires of South-

ern and Eastern Australia on Invertebrate Species Final Report."

Marsh, Jessica R., Payal Bal, Hannah Fraser, Kate Umbers, Tanya Latty, Aaron Greenville, Libby Rumpff, and John C. Z. Woinarski. 2022. "Accounting for the Neglected: Invertebrate Species and the 2019–2020 Australian Megafires." *Global Ecology and Biogeography* n/a (n/a). https://doi.org/10.1111/geb.13550.

Ribeiro, Bruno R., Santiago José Elías Velazco, Karlo Guidoni-Martins, Geiziane Tessarolo, Lucas Jardim, Steven P. Bachman, and Rafael Loyola. 2022. "Bdc: A Toolkit for Standardizing, Integrating and Cleaning Biodiversity Data." *Methods in Ecology and Evolution* 13 (7): 1421–28. https://doi.org/10.1111/2041-210X.13868.

Rodrigues, Arthur Vinicius, Gabriel Nakamura, Vanessa Graziele Staggemeier, and Leandro Duarte. 2022. "Species Misidentification Affects Biodiversity Metrics: Dealing with This Issue Using the New R Package naturaList." *Ecological Informatics* 69 (July): 101625. https://doi.org/10.1016/j.ecoinf.2022.101625.

Rowley, Jodi JL, and Corey T Callaghan. 2020. "The FrogID Dataset: Expert-Validated Occurrence Records of Australia's Frogs Collected by Citizen Scientists." *ZooKeys* 912: 139.

Simões, Marianna VP, and A Townsend Peterson. 2018. "Utility and Limitations of Climate-Matching Approaches in Detecting Different Types of Spatial Errors in Biodiversity Data." *Insect Conservation and Diversity* 11 (5): 407–14.

streamdna. 2020. "Sharing Is Caring: Working with Other People's Data." https://methodsblog.com/2020/09/04/sharing-is-caring-working-with-other-peoples-data/.

Zizka, Alexander, Fernanda Antunes Carvalho, Alice Calvente, Mabel Rocio Baez-Lizarazo, Andressa Cabral, Jéssica Fernanda Ramos Coelho, Matheus Colli-Silva, Mariana Ramos Fantinati, Moabe F Fernandes, and Thais Ferreira-Araújo. 2020a. "No One-Size-Fits-All Solution to Clean GBIF." *PeerJ* 8: e9916.

———. 2020b. "No One-Size-Fits-All Solution to Clean GBIF." *PeerJ* 8: e9916.

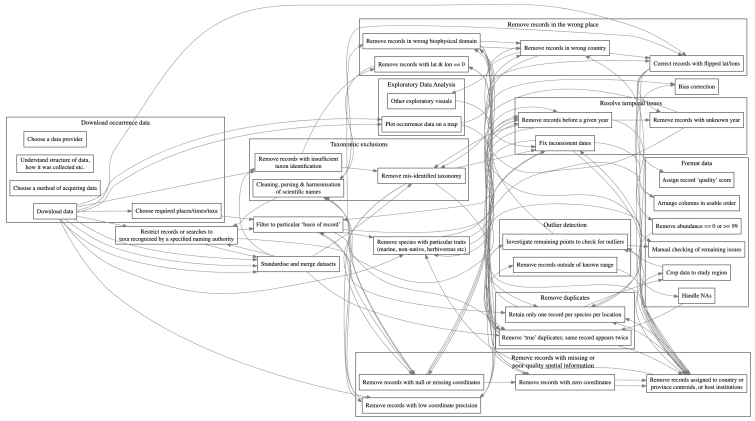
11 Appendix

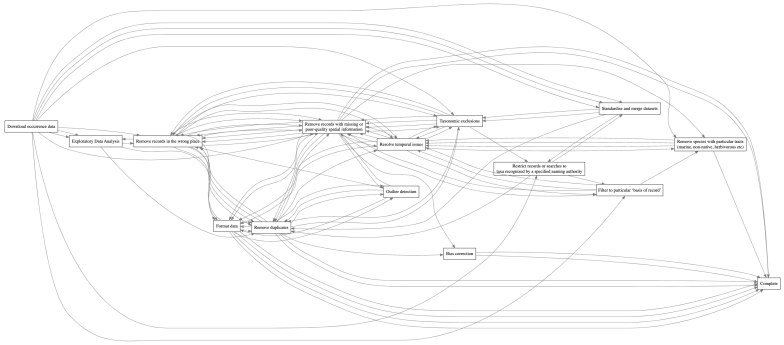
We don't claim to be experts in data cleaning, therefore in order to ensure content for this book was current and relevant we undertook an informal literature review of both peer reviewed and grey literature. Key themes searched were:

1. Cleaning data for species distribution models
2. Cleaning open biodiversity data
3. Australian and global naming authorities
4. R packages for biodiversity data cleaning

As this was not a comprehensive literature review recent papers were selected first as the R environment is a rapidly evolving space. Methods sections outlining data cleaning protocols were read and collated into a database. Papers which were frequently referenced were also chosen for review in order to not miss older seminal papers. Additionally our project partners outputs (J. Marsh et al. 2021; Godfree et al. 2021b) have been investigated in detail to understand and streamline their data cleaning processes. This has included detailed review of their code base as well as meetings with the authors of the papers to understand their processes, issues and needs.

All steps for acquiring and cleaning data were then looked at together in order to understand what were essential steps, versus what was done in certain use cases. We also investigated the order in which steps were undertaken with the idea of developing a streamlined workflow. However the diagrams below show the complexity of this, with data cleaning being extremely iterative.


Remove records in the wrong place
Remove records in wrong biophysical domain
Remove records with lat & lon == 0
Remove records in wrong country
Correct records with flipped lat/lons
Exploratory Data Analysis
Other exploratory visuals
Plot occurrence data on a map
Bias correction
Resolve temporal issues
Remove records before a given year
Remove records with unknown year
Fix inconsistent dates
Download occurrence data
Choose a data provider
Understand structure of data, how it was collected etc.
Choose a method of acquiring data
Download data
Choose required places/times/taxa
Taxonomic exclusions
Remove records with insufficient taxon identification
Cleaning, parsing & harmonisation of scientific names
Remove mis-identified taxonomy
Format data
Assign record 'quality' score
Arrange columns in usable order
Remove abundance == 0 or >= 99
Restrict records or searches to taxa recognised by a specified naming authority
Filter to particular 'basis of record'
Standardise and merge datasets
Remove species with particular traits (marine, non-native, herbivorous etc)
Outlier detection
Investigate remaining points to check for outliers
Remove records outside of known range
Manual checking of remaining issues
Crop data to study region
Handle NAs
Remove duplicates
Retain only one record per species per location
Remove 'true' duplicates; same record appears twice
Remove records with missing or poor-quality spatial information
Remove records with null or missing coordinates
Remove records with zero coordinates
Remove records assigned to country or province centroids, or host institutions
Remove records with low coordinate precision


Download occurrence data
Exploratory Data Analysis
Remove records in the wrong place
Remove records with missing or poor-quality spatial information
Resolve temporal issues
Taxonomic exclusions
Standardise and merge datasets
Remove species with particular traits (marine, non-native, herbivorous etc)
Restrict records or searches to taxa recognised by a specified naming authority
Filter to particular 'basis of record'
Outlier detection
Format data
Remove duplicates
Bias correction
Complete