data_compile

Webb T. & Johnson T.F.

Accessing and compiling data to support analysis of biodiversity trends.

Required packages

This workflow is developed in R (R Core Team 2022) using the `tidyverse` suite of packages (Wickham et al. 2019), in addition using `here` (Müller 2020) for robust file referencing and `janitor` (Firke 2023) to automate some data cleaning operations. For R version, platform, package versions, etc. see Reproducibility.

Load required packages:

```r
library(tidyverse)
library(here)
library(janitor)
```

Process Individual Datasets

RivFishTIME data

RivFishTime is a global database of freshwater fish time-series to study global change ecology in riverine systems, fully described in Comte et al. 2020, with the data hosted by iDiv here. Please see the full description and meta data of this database for more information.

The full zipped database can be downloaded directly and unzipped (before removing the zipped version):

```r
download.file("https://idata.idiv.de/ddm/Data/DownloadZip/1873?version=5514",
              destfile = here::here("data", "raw_data/1873_2_RivFishTIME.zip"))
unzip(here::here("data", "raw_data/1873_2_RivFishTIME.zip"),
      exdir = here::here("data", "raw_data/1873_2_RivFishTIME"))
unlink(here::here("data", "raw_data/1873_2_RivFishTIME.zip"))
```

The two files required are the main survey dataset '1873_2_RivFishTIME_SurveyTable.csv' (this contains the record-level data, i.e. abundance of a given species at a given site in a given year), and the dataset describing each individual time series '1873_2_RivFishTIME_TimeseriesTable.csv' (this contains information on the specific location of each time series). To read these both in:

```r
RivFish_survey <- read_csv(
  here::here("data/raw_data",
             "1873_2_RivFishTIME/1873_2_RivFishTIME_SurveyTable.csv"))
```

```
## Rows: 646270 Columns: 7
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr (5): TimeSeriesID, SurveyID, Quarter, Species, UnitAbundance
## dbl (2): Year, Abundance
##
## i Use ‘spec()‘ to retrieve the full column specification for this data.
## i Specify the column types or set ‘show_col_types = FALSE‘ to quiet this message.
```

```
RivFish_timeseries <- read_csv(
  here::here("data/raw_data",
             "1873_2_RivFishTIME/1873_2_RivFishTIME_TimeseriesTable.csv"))
```

```
## New names:
## Rows: 11386 Columns: 13
## -- Column specification
## -------------------------------------------------------- Delimiter: "," chr
## (9): SiteID, TimeSeriesID, Protocol, BioRealm, Country, Region, Province... dbl
## (4): SourceID, Latitude, Longitude, HydroBasin
## i Use ‘spec()‘ to retrieve the full column specification for this data. i
## Specify the column types or set ‘show_col_types = FALSE‘ to quiet this message.
## * ‘‘ -> ‘...13‘
```

Now join the timeseries information to the main survey data, select required variables and rename as needed, and add a dataset ID variable:

```
RivFish_survey <- RivFish_survey %>%
  left_join(select(RivFish_timeseries,
                   SiteID, TimeSeriesID, Latitude, Longitude, Country),
            join_by(TimeSeriesID)) %>%
  select(TimeSeriesID, Country, Latitude, Longitude,
         Year, Species, Abundance, UnitAbundance) %>%
  janitor::clean_names() %>%
  rename(site = time_series_id, date = year, unit = unit_abundance) %>%
  mutate(dataset_id = "RivFish") %>%
  select(dataset_id, everything())
```

This results in a single table with 646270 observations of 9 variables: site ID (the timeseries ID from RivFishTIME), country, latitude and longitude, as well as date (year), species, abundance, and units of abundance.

Write this to file as a first derived dataset:

```
write_csv(RivFish_survey, here::here("data", "derived_data/RivFish_processed.csv"))
```

North American Breeding Bird Survey

The 2022 release of the North American Breeding Bird Survey dataset (1966-2021) is available from Ziolkowski Jr. et al. (2022). From the dataset description there: This dataset contains avian point count data for more than 700 North American bird taxa (species, races, and unidentified species groupings), collected annually during the breeding season along thousands of randomly established roadside survey routes in the United States and Canada. Routes are roughly 24.5 miles (39.2 km) long with counting locations placed at approximately half-mile (800-m) intervals, for a total of 50 stops. At each stop, a citizen scientist highly skilled in avian identification conducts a 3-minute point count, recording all birds seen within a quarter-mile (400-m) radius and all birds heard. Surveys begin 30 minutes before local sunrise and take approximately 5 hours to complete. Routes are sampled once per year, with the total number of routes sampled per year growing over time; just over 500 routes were sampled in 1966, while in recent decades approximately 3000 routes have been sampled annually. No data are provided for 2020. BBS field activities were cancelled in 2020 because of the coronavirus disease (COVID-19) global pandemic and observers were directed to not sample routes. Route location information includes country, state, and BCR, as well as geographic coordinates of route start point, and an indicator of run data quality.

We require the 'States' and 'Routes' zipped datasets, and the species list (provided as a text file):

```
download.file("https://www.sciencebase.gov/catalog/file/get/625f151ed34e85fa62b7f926?f=__disk__42%2Fed%
              destfile = here::here("data", "raw_data/BBS_states.zip"))
unzip(here::here("data", "raw_data/BBS_states.zip"),
      exdir = here::here("data", "raw_data/NAm_BBS"))

download.file("https://www.sciencebase.gov/catalog/file/get/625f151ed34e85fa62b7f926?f=__disk__95%2F4e%
              destfile = here::here("data", "raw_data/BBS_routes.zip"))
unzip(here::here("data", "raw_data/BBS_routes.zip"),
      exdir = here::here("data", "raw_data/NAm_BBS"))

download.file("https://www.sciencebase.gov/catalog/file/get/625f151ed34e85fa62b7f926?f=__disk__fb%2F82%
              destfile = here::here("data", "raw_data/NAm_BBS/BBS_sp_list.txt"))

unlink(here::here("data", "raw_data/BBS_states.zip"))
unlink(here::here("data", "raw_data/BBS_routes.zip"))
```

First read in the routes data:

```
bbs_routes <- read_csv(here::here("data", "raw_data/NAm_BBS/routes.csv"))
```

```
## Rows: 5784 Columns: 11
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr (3): StateNum, Route, RouteName
## dbl (8): CountryNum, Active, Latitude, Longitude, Stratum, BCR, RouteTypeID,...
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

The species list is provided with several leading rows of description, then with data in fixed width columns. We read in skipping the leading rows:

```
bbs_species <- read_fwf(
  file = here::here("data", "raw_data/NAm_BBS/BBS_sp_list.txt"),
  skip = 12)
```

```
## Rows: 763 Columns: 9
## -- Column specification --------------------------------------------------------
##
## chr (9): X1, X2, X3, X4, X5, X6, X7, X8, X9
##
```

```
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

The second row is a row of dashes used for presentation in the text file; we remove these, elevate row 1 to be the variable names, and create a new sciname (scientific name) variable as Genus + species:

```
bbs_species <- bbs_species %>%
  slice(-2) %>%
  janitor::row_to_names(row_number = 1) %>%
  unite("sciname", Genus:Species, sep = " ") %>%
  select(Seq, AOU, sciname, everything())

bbs_species
```

```
## # A tibble: 761 x 8
##    Seq   AOU   sciname                    English_Common_Name French_Common_Name
##    <chr> <chr> <chr>                      <chr>               <chr>
##  1 6     01770 Dendrocygna autumnalis     Black-bellied Whis~ "Dendrocygne \xe0~
##  2 7     01780 Dendrocygna bicolor        Fulvous Whistling-~ "Dendrocygne fauv~
##  3 8     01760 Anser canagicus            Emperor Goose       "Oie empereur"
##  4 9     01690 Anser caerulescens         Snow Goose          "Oie des neiges"
##  5 10    01691 Anser caerulescens (blue ~ (Blue Goose) Snow ~ "Oie des neiges (~
##  6 11    01700 Anser rossii               Ross's Goose        "Oie de Ross"
##  7 13    01710 Anser albifrons            Greater White-fron~ "Oie rieuse"
##  8 18    01730 Branta bernicla            Brant               "Bernache cravant"
##  9 19    01740 Branta bernicla nigricans  (Black Brant) Brant "Bernache cravant~
## 10 21    01725 Branta hutchinsii          Cackling Goose      "Bernache de Hutc~
## # i 751 more rows
## # i 3 more variables: Spanish_Common_Name <chr>, ORDER <chr>, Family <chr>
```

The states data are provided as a zipped file for each state, but each of these is just a single csv file so can be read directly with `read_csv` To read them all into one big dataframe (specifying which columns to read, and stating their required data types):

```
bbs_states <-
  list.files(path = here::here("data", "raw_data/NAm_BBS/states"),
             full.names = TRUE) %>%
  map_df(~read_csv(.,
                   col_types = cols_only(
                     RouteDataID = col_double(),
                     CountryNum = col_double(),
                     StateNum = col_character(),
                     Route = col_character(),
                     RPID = col_double(),
                     Year = col_double(),
                     AOU = col_character(),
                     SpeciesTotal = col_double()
                   )))

glimpse(bbs_states)
```

```
## Rows: 6,946,871
## Columns: 8
## $ RouteDataID  <dbl> 6234747, 6234747, 6234747, 6234747, 6234747, 6234747, 623~
## $ CountryNum   <dbl> 840, 840, 840, 840, 840, 840, 840, 840, 840, 840, 840, 84~
## $ StateNum     <chr> "02", "02", "02", "02", "02", "02", "02", "02", "02", "02~
## $ Route        <chr> "001", "001", "001", "001", "001", "001", "001", "001", "~
## $ RPID         <dbl> 101, 101, 101, 101, 101, 101, 101, 101, 101, 101, 101, 10~
## $ Year         <dbl> 1967, 1967, 1967, 1967, 1967, 1967, 1967, 1967, 1967, 196~
## $ AOU          <chr> "02010", "02020", "02030", "02730", "02890", "03131", "03~
## $ SpeciesTotal <dbl> 2, 1, 1, 8, 52, 11, 43, 19, 14, 6, 11, 4, 8, 1, 35, 6, 10~
```

Processing now involves adding location details and species names to this large dataset, creating a unique 'site' ID (from country, state, and route information), and mutating, renaming, or adding the other required variables:

```
bbs_states <- bbs_states %>%
  left_join(select(bbs_routes, CountryNum:Longitude),
            join_by(CountryNum, StateNum, Route)) %>%
  left_join(select(bbs_species, AOU, sciname),
            join_by(AOU)) %>%
  janitor::clean_names() %>%
  unite("route", c(country_num, state_num, route, route_name), remove = FALSE) %>%
  mutate(country = case_match(
    country_num,
    840 ~ "USA", 124 ~ "CAN"
  )) %>%
  select(route, country, latitude, longitude, year, sciname, species_total) %>%
  rename(site = route, date = year, species = sciname, abundance = species_total) %>%
  mutate(unit = "n_individuals", dataset_id = "NAm_BBS") %>%
  select(dataset_id, everything())
```

This is a large dataset, with 6946871 observations of 9 variables, occupying 5.0067842×10^8 of disc space. Write this to our derived data folder:

```
write_csv(bbs_states,
          here::here("data", "derived_data/NAm_BBS_processed.csv"))
```

BioTIME

BioTIME (Dornelas et al. 2018) is a comprehensive collection of assemblage time-series in which the abundances of the species that comprise ecological communities have been monitored over a number of years. BioTIME data span the globe and encompass land and seas; they also include freshwater systems. The current version of BioTIME contains over 12 million records, features almost 50 thousand species, covers over 600 thousand distinct geographic locations and is representative of over 20 biomes, occurring over 6 different climatic zones. This dataset requires registration prior to download. We downloaded the June 2021 version (the latest available) of the raw data in CSV format, together with the meta data and citations files, from the link provided following registration at https://biotime.st-andrews.ac.uk/download.php.

First, unzip the data and remove the zipped archive:

```
unzip(here::here("data", "raw_data/BioTIMEQuery_24_06_2021.zip"),
      exdir = here::here("data", "raw_data/BioTIME"))
unlink(here::here("data", "raw_data/BioTIMEQuery_24_06_2021.zip"))
```

Read in the raw data and metadata:

```
biotime <- read_csv(here::here(
  "data", "raw_data/BioTIME/BioTIMEQuery_24_06_2021.csv")) %>%
  janitor::clean_names()
```

```
## New names:
## Rows: 8552249 Columns: 15
## -- Column specification
## -------------------------------------------------------- Delimiter: "," chr
## (5): SAMPLE_DESC, PLOT, GENUS, SPECIES, GENUS_SPECIES dbl (10): ...1, STUDY_ID,
## DAY, MONTH, YEAR, ID_SPECIES, LATITUDE, LONGITUDE,...
## i Use ‘spec()‘ to retrieve the full column specification for this data. i
## Specify the column types or set ‘show_col_types = FALSE‘ to quiet this message.
## * ‘‘ -> ‘...1‘
```

```
biotime_meta <- read_csv(here::here(
  "data", "raw_data/BioTIME/BioTIMEMetadata_24_06_2021.csv")) %>%
  janitor::clean_names()
```

```
## Rows: 381 Columns: 42
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr (28): REALM, CLIMATE, GENERAL_TREAT, TREATMENT, TREAT_COMMENTS, TREAT_DA...
## dbl (13): STUDY_ID, DATA_POINTS, START_YEAR, END_YEAR, CENT_LAT, CENT_LONG, ...
## lgl  (1): PROTECTED_AREA
##
## i Use ‘spec()‘ to retrieve the full column specification for this data.
## i Specify the column types or set ‘show_col_types = FALSE‘ to quiet this message.
```

Processing the BioTIME data requires selecting relevant columns, and joining to the metadata in order to filter out studies that recorded only presence/absence, or only biomass (no abundance data). A ‘site’ variable is created by pasting the study ID, latitude, and longitude. No country information is provided.

```
biotime <- biotime %>% select(
  study_id, latitude, longitude, year, genus_species, sum_allrawdata_abundance) %>%
  left_join(select(biotime_meta, study_id, abundance_type),
            join_by(study_id)) %>%
  filter(abundance_type != "Presence/Absence" & !is.na(abundance_type)) %>%
  mutate(dataset_id = "BioTIME",
         site = paste(study_id, round(latitude, 2), round(longitude, 2), sep = "_"),
         country = NA,
         unit = case_match(
           abundance_type,
           "Count" ~ "n_individuals",
           "MeanCount" ~ "mean_n_individuals",
           "Density" ~ "density"
         )) %>%
  rename(date = year, species = genus_species, abundance = sum_allrawdata_abundance) %>%
  select(dataset_id, site, country, latitude, longitude, date, species, abundance, unit)
```

This results in a dataframe with 5892450 observations of the 9 variables, which is written to file:

```r
write_csv(biotime,
          here::here("data", "derived_data/biotime_processed.csv"))
```

Living Planet Index

The Living Planet Index (LPI; LPI 2022) is a measure of the state of the world's biological diversity based on population trends of vertebrate species from terrestrial, freshwater and marine habitats. The LPI is based on trends of thousands of population time series collected from monitored sites around the world. Accessing the dataset requires registration prior to download from https://www.livingplanetindex.org/data_portal. We downloaded the latest available (2022) zipped version of the database into our raw data folder.

First, unzip the data and remove the zipped archive:

```r
unzip(here::here("data", "raw_data/LivingPlanetIndexDatabase_2023-04-05_12.50.51.zip"),
      exdir = here::here("data", "raw_data/LPI"))
unlink(here::here("data", "raw_data/LivingPlanetIndexDatabase_2023-04-05_12.50.51.zip"))
```

The subfolder 'LivingPlanetIndex_2022_PublicData' includes the LPI data agreement (data_agreement_2022.pdf) and metadata (LPD_metadata.pdf) as well as the public data as a csv file (LPD2022_public.csv). Read in this csv - NB 'NULL' is used to indicate missing abundance values so that is specified here:

```r
lpi <- read_csv(here::here(
  "data", "raw_data/LPI/LivingPlanetIndex_2022_PublicData/LPD2022_public.csv"),
  na = c("", "NA", "NULL"))
```

```
## Rows: 32680 Columns: 103
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr (26): Binomial, Citation, Class, Order, Family, Genus, Species, Subspeci...
## dbl (77): ID, Replicate, Latitude, Longitude, Specific_location, Migratory_f...
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

Country names are supplied, to convert these to 3 character ISO codes we use the `countrycode` package (Arel-Bundock et al. 2018). There are two countries in the LPI data that do not match due to character encoding issues, plus 'International Waters', so we first set up custom matches for thes:

```r
library(countrycode)
custom_match <- c(`Cura??ao` = "CUW", `?Ōland Islands` = "ALA", `International Waters` = "ABNJ")
```

Then process the LPI data. First remove all populations identified as 'replicates'. Then create a site ID as the first word of the citation, citation year (if available, NA if not - this will throw a warning), and location pasted together. Then select required columns, and pivot to long format (single columns for year and for abundance), filtering out NA values for abundance. Mutate country to ISO code, and do some renaming and reordering of columns.

```r
lpi <- subset(lpi, Replicate == 0)
lpi <- lpi %>%
  mutate(
    citation_year = str_replace_all(Citation, "\\\"|\\.|-|\\\'", ""),
    citation_year = parse_number(citation_year),
    site = paste(word(Citation, 1, sep = " |,"),
                 citation_year,
                 Location,
                 sep = "_")
  ) %>%
  select(site, Country, Latitude, Longitude, Binomial, Units, `1950`:`2020`) %>%
  pivot_longer(cols = `1950`:`2020`,
               names_to = "date",
               values_to = "abundance") %>%
  filter(!is.na(abundance)) %>%
  janitor::clean_names() %>%
  mutate(dataset_id = "LPI",
         country = countrycode(country,
                               origin = "country.name", destination = "iso3c",
                               custom_match = custom_match)) %>%
  rename(species = binomial, unit = units) %>%
  select(dataset_id, site:longitude, date, species, abundance, unit)
lpi$species = gsub("_", " ", lpi$species)
```

Note that Abundance units are highly varied - there are over 2400 distinct measures:

```r
(lpi_abundance_units <- lpi %>% count(unit) %>% arrange(desc(n)))
```

```
## # A tibble: 2,052 x 2
##    unit                                      n
##    <chr>                                 <int>
##  1 Sample: abundance (counts)            76061
##  2 Average number of individuals         20599
##  3 Number of individuals                 15491
##  4 Catch per tow                         10273
##  5 SSB-MT                                 8303
##  6 Individuals                            8267
##  7 Index                                  6947
##  8 Number of breeding pairs               3658
##  9 Biomass (Kg)                           2539
## 10 CPUE (Weight/ (minute towed*tow length))  2109
## # i 2,042 more rows
```

This results in a huge range of abundances - e.g. very high abundances, which are all from fisheries surveys:

```r
lpi %>% filter(abundance > 10e+10) %>% select(unit) %>% distinct()
```

```
## # A tibble: 2 x 1
##   unit
##   <chr>
## 1 Spawning stock biomass (Thousand billion eggs)
## 2 SSB-MT
```

There are also many observations (n = 56328) with abundance = 0; these involve a wide range of different abundance units:

```
lpi %>% filter(abundance == 0) %>% select(unit) %>% distinct()
```

```
## # A tibble: 353 x 1
##    unit
##    <chr>
##  1 Number of individuals
##  2 Mean number of individuals
##  3 Individuals
##  4 Number of breeding pairs
##  5 Number of individuals seen
##  6 Log number of nestlings
##  7 Number of calling males
##  8 Number of individuals per 1000 trap nights
##  9 Lemming abundance rank
## 10 Number of individual frogs observed
## # i 343 more rows
```

Write the processed LPI data to file:

```
write_csv(lpi, here::here("data/derived_data", "lpi_processed.csv"))
```

CaPTrends

CaPTrends (Johnson et al. 2022) is a database of 1,122 population trends from around the world, describing changes in abundance over time in large mammal species (n =.50) from four families (Canidae, Felidae, Hyaenidae and Ursidae) in the order Carnivora. Trends represent 621 unique locations across the globe (latitude: -51.0 to 80.0; longitude: -166.0 to 166.0), from 1726 to 2017. The dataset itself is hosted on Zenodo here: https://zenodo.org/record/6949487. First, download the zipped dataset and unzip:

```
download.file("https://zenodo.org/records/6949487/files/GitTFJ/CaPTrends-V1.0.0.zip?download=1",
              destfile = here::here("data", "raw_data/CaPTrends-V1.0.0.zip"))

unzip(here::here("data", "raw_data/CaPTrends-V1.0.0.zip"),
      exdir = here::here("data", "raw_data/CaPTrends"))
unlink(here::here("data", "raw_data/CaPTrends-V1.0.0.zip"))
```

This results in a folder containing various files:

```
list.files(here::here("data", "raw_data/CaPTrends/GitTFJ-CaPTrends-999c7a2"))
```

```
##  [1] "abundance.csv"                "amended_upham_phylogeny.nex"
##  [3] "captrends_live"               "captrends.csv"
##  [5] "direction.csv"                "license.txt"
##  [7] "metadata_abundance.pdf"       "metadata_captrends.pdf"
##  [9] "metadata_direction.pdf"       "metadata_source.pdf"
## [11] "readme.txt"                   "reference_tables"
## [13] "source.csv"
```

Examining the meta data and descriptions, the two files we require are ‘captrends.csv’ (details of each individual study) and ‘abundance.csv’ (the actual abundance values). Read these in:

```r
captrends <- read_csv(here::here("data/raw_data/CaPTrends",
                                 "GitTFJ-CaPTrends-999c7a2/captrends.csv")) %>%
  janitor::clean_names()
```

```
## New names:
## Rows: 1122 Columns: 63
## -- Column specification
## -------------------------------------------------------- Delimiter: "," chr
## (25): Species_corrected, Species_reported, Sub_species, Citation_key, Lo... dbl
## (37): ...1, DataTableID, IUCN_id, Spatial_locality, Temporal_locality, L... lgl
## (1): Significant_trend
## i Use ‘spec()‘ to retrieve the full column specification for this data. i
## Specify the column types or set ‘show_col_types = FALSE‘ to quiet this message.
## * ‘‘ -> ‘...1‘
```

```r
captrends_ab <- read_csv(here::here("data/raw_data/CaPTrends",
                                    "GitTFJ-CaPTrends-999c7a2/abundance.csv")) %>%
  janitor::clean_names()
```

```
## New names:
## Rows: 4632 Columns: 4
## -- Column specification
## -------------------------------------------------------- Delimiter: "," dbl
## (4): ...1, DataTableID, Value, Year
## i Use ‘spec()‘ to retrieve the full column specification for this data. i
## Specify the column types or set ‘show_col_types = FALSE‘ to quiet this message.
## * ‘‘ -> ‘...1‘
```

Do some processing of the study-level data - this involves using the `countrycodes` package again to convert country names to ISO codes - again with some custom matches set up in advance:

```r
custom_match <- c(`C\xf4te D'Ivoire` = "CIV", `Kosovo` = "XXK", `GLOBAL` = "GLOBAL")
```

Some studies cover multiple countries - we assign the code to the first country mentioned. A ‘site’ variable is created by pasting `data_table_id`, `citation_key`, and `locality_name`. Latitude and longitude values are not available so these are set to NA. The abundance unit is set to the value in `population_metric` if available, otherwise to the `field_method` value.

```r
captrends <- captrends %>% mutate(
  dataset_id = "CaPTrends",
  site = paste("dt_id", data_table_id, citation_key, locality_name, sep = "_"),
  country = ifelse(!is.na(singular_country), singular_country, word(multiple_countries, 1)),
  country = str_replace(country, "\xf4", "o"),
  country = countrycode::countrycode(country,
                                     origin = "country.name", destination = "iso3c",
                                     custom_match = custom_match),
  latitude = as.numeric(latitude), longitude = as.numeric(longitude),
  species = ifelse(!is.na(sub_species), paste(species_corrected, sub_species), species_corrected),
  unit = ifelse(!is.na(population_metric), population_metric, field_method)) %>%
  select(dataset_id, data_table_id, site, country, latitude, longitude, species, unit)
```

This processed version of the CaPTrends study information is linked to the abundance data by `data_table_id`, and variables are renamed as needed to create the final pocessed version of the CaPTrends data:

```r
captrends_ab <- captrends_ab %>%
  select(-x1) %>%
  left_join(captrends, join_by(data_table_id)) %>%
  rename(abundance = value, date = year) %>%
  mutate(date = as.integer(floor(date))) %>%
  select(dataset_id, site, country, latitude, longitude, date, species, abundance, unit)
```

This results in a dataset of 4632 observations, which is written to the derived data folder:

```r
write_csv(captrends_ab, here::here("data/derived_data", "captrends_processed.csv"))
```

ReSurvey Germany: vegetation-plot resurvey data from Germany

ReSurvey Germany: vegetation-plot resurvey data from Germany (Jandt et al. 2022) is a compilation of harmonised vegetation-plot resurvey data from Germany covering almost 100 years. The data allow calculating temporal biodiversity change at the community scale. They also enable tracking changes in the incidence and distribution of individual species across Germany. Cover records are available for 1,794 vascular plant species in 7,738 (semi-)permanent vegetation plots from Germany, resurveyed from 2 to 54 times, in total resulting in 23,641 vegetation records and 458,311 species cover records, comprising the years from 1927 to 2020 and 97 EUNIS habitat types. The data is available to download from: https://doi.org/10.25829/idiv.3514-0qsq70

First, download the zipped dataset, unzip, and remove the zipped archive:

```r
download.file("https://idata.idiv.de/ddm/Data/DownloadZip/3514?version=5513",
              destfile = here::here("data", "raw_data/ReSurvey.zip"))

unzip(here::here("data", "raw_data/ReSurvey.zip"),
      exdir = here::here("data", "raw_data/ReSurvey"))

unlink(here::here("data", "raw_data/ReSurvey.zip"))
```

The unzipped folder contains a further zipped file containing all the data, to unzip:

```r
unzip(here::here("data", "raw_data/ReSurvey/ReSurveyGermany.zip"),
      exdir = here::here("data", "raw_data/ReSurvey"))
```

Explore the folder contents:

```r
list.files(here::here("data", "raw_data/ReSurvey"))
```

```
##  [1] "3514_9_manifest.xml"
##  [2] "3514_9_metadata.html"
##  [3] "Header_ReSurveyGermany_Description.csv"
##  [4] "Header_ReSurveyGermany.csv"
##  [5] "ReSurveyGermany_Description.csv"
##  [6] "ReSurveyGermany_Project_List.csv"
##  [7] "ReSurveyGermany_References.csv"
##  [8] "ReSurveyGermany.csv"
##  [9] "ReSurveyGermany.zip"
## [10] "Schema"
```

The main files required are the 'header' data (project / site level data) in 'Header_ReSurveyGermany.csv', and the main dataset of species-level abundances in 'ReSurveyGermany.csv'.

```
resurvey_header <- read_csv(here::here("data/raw_data/ReSurvey",
                                       "Header_ReSurveyGermany.csv")) %>%
  janitor::clean_names()
```

```
## Rows: 23641 Columns: 47
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr (20): RS_PROJECT, RS_PLOT, RS_SITE, LOCALITY, RS_OBSERV, PROJECT_ID_RELE...
## dbl (27): PROJECT_ID, RELEVE_NR, DATE, YEAR, SURF_AREA, LOC_METHOD, LONGITUD...
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
resurvey_dat <- read_csv(here::here("data/raw_data/ReSurvey",
                                    "ReSurveyGermany.csv")) %>%
  janitor::clean_names()
```

```
## Rows: 610313 Columns: 6
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr (2): PROJECT_ID_RELEVE_NR, TaxonName
## dbl (4): PROJECT_ID, RELEVE_NR, LAYER, Cover_Perc
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

Both of these have associated meta-data tables too which we also read in:

```
resurvey_header_meta <- read_csv(here::here("data/raw_data/ReSurvey",
                                            "Header_ReSurveyGermany_description.csv"))
```

```
## Rows: 47 Columns: 4
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr (3): Field name, Type, Description
## dbl (1): Number of NAs
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
resurvey_dat_meta <- read_csv(here::here("data/raw_data/ReSurvey",
                                         "ReSurveyGermany_description.csv"))
```

```
## Rows: 6 Columns: 3
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr (3): Field name, Type, Description
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

From `resurvey_header_meta` we can see that `project_id_releve_nr` gives a unique plot-level year-level ID. Also needed from `resurvey_header` are year, longitude, and latitude, and to consstruct the site variable: project_id, rs_plot,. Extract these and join to taxon name and percent cover from `resurvey_dat`, then add and rename variables as needed for consistency with the other processed datasets:

```r
resurvey_dat <- resurvey_dat %>%
  select(project_id_releve_nr, taxon_name, cover_perc) %>%
  left_join(select(resurvey_header, project_id_releve_nr, project_id, rs_plot, rs_site, year, longitude
            join_by(project_id_releve_nr)) %>%
  mutate(site = paste(project_id, rs_plot, rs_site, sep = "_")) %>%
  rename(species = taxon_name,
         abundance = cover_perc,
         date = year) %>%
  mutate(dataset_id = "ReSurvey_Germany", country = "DEU", unit = "percent_cover") %>%
  select(dataset_id, site, country, latitude, longitude, date, species, abundance, unit)
```

This results in a dataset with 610313 observations. Write this to the derived data folder:

```r
write_csv(resurvey_dat, here::here("data/derived_data", "ReSurvey_processed.csv"))
```

Environment Agency NFPD Fish Counts

The Environment Agency undertakes fisheries monitoring work on rivers, lakes and transitional and coastal waters (TraC) throughout England. The freshwater fish survey dataset (or *National Fish Populations Database*, NFPD, Environment Agency 2020) contains site and survey information, as well as the numbers and species of fish caught, for all the freshwater fish surveys carried out across England (with a small number from Wales and Scotland) from 1975 onwards. We accessed the Freshwater Fish Count dataset from https://environment.data.gov.uk/ecology/explorer/downloads/. Meta-data is available in a pdf document here.

First, we download and unzip the NFPD Fish Counts database:

```r
download.file(
  "https://environment.data.gov.uk/ecology/explorer/downloads/FW_Fish_Counts.zip",
  destfile = here::here("data", "raw_data/EA_NFPD.zip"))

unzip(here::here("data", "raw_data/EA_NFPD.zip"),
      exdir = here::here("data", "raw_data/EA_NFPD"))

unlink(here::here("data", "raw_data/EA_NFPD.zip"))
```

Read in the NFPD database:

```r
ea_nfpd <- read_csv("data/raw_data/EA_NFPD/FW_Fish_Counts.csv") %>%
  janitor::clean_names()
```

```
## Rows: 337433 Columns: 48
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr (23): DATA_OWNER, COUNTRY, LOCATION_NAME, REGION, AREA, ABC_SITE, TOP_TI...
## dbl (25): SITE_PARENT_ID, SITE_ID, SURVEY_ID, EVENT_DATE_YEAR, SURVEY_RANKED...
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

The geographic coordinates in this database are provided as eastings and northings in the UK national projection (EPSG 27700). In addition, there are some location errors placing surveys offshore or with incomplete coordinates (e.g. easting and northing both = 1). To address these, we use the `sf` library (Pebesma 2018) to convert the database into spatial format, reproject to WGS84 latitude and longitude coordinates, and we combine it with a UK coastline map to exclude surveys that occur at sea. First, load `sf` and make the NFPD dataset spatial:

```
library(sf)
```

```
## Linking to GEOS 3.11.0, GDAL 3.5.3, PROJ 9.1.0; sf_use_s2() is TRUE
```

```
ea_nfpd_s <- ea_nfpd %>% st_as_sf(coords = c("survey_ranked_easting",
                                              "survey_ranked_northing"),
                                  crs = 27700,
                                  remove = FALSE)
```

We obtained the digital vector boundaries for Countries in the United Kingdom as at December 2022 at full resolution, clipped to the coastline (Mean High Water mark), from the Office for National Statistics, available https://geoportal.statistics.gov.uk/datasets/ons::countries-december-2022-uk-bfc/. We downloaded the shapefile, 'Countries_December_2022_UK_BFC_3731595901038458592.zip', directly downloaded and unzipped here:

```
download.file("https://stg-arcgisazurecdataprod1.az.arcgis.com/exportfiles-1559-18703/Countries_Decembe
              destfile = here::here("data", "raw_data/GB_shapefile.zip"))

unzip(here::here("data", "raw_data/GB_shapefile.zip"),
      exdir = here::here("data", "raw_data/GB_shapefile"))

unlink(here::here("data", "raw_data/GB_shapefile.zip"))
```

Next, read in the countries shapefile and reproject to the UK grid projection, EPSG 27700:

```
gbr <- st_read(here::here("data/raw_data/GB_shapefile",
                          "CTRY_DEC_2022_UK_BFC.shp")) %>%
  st_transform(crs = 27700)
```

```
## Reading layer `CTRY_DEC_2022_UK_BFC' from data source
##   `/Users/bo1tfj/Library/CloudStorage/GoogleDrive-t.f.johnson@sheffield.ac.uk/My Drive/research/stab
##   using driver `ESRI Shapefile'
## Simple feature collection with 4 features and 7 fields
## Geometry type: MULTIPOLYGON
## Dimension:     XY
## Bounding box:  xmin: -116.1928 ymin: 5336.966 xmax: 655653.8 ymax: 1220302
## Projected CRS: OSGB36 / British National Grid
```

Then use a spatial join to remove survey points from the NFPD data that are do not fall within the country boundaries:

```
ea_nfpd_s <- ea_nfpd_s %>% st_join(select(gbr, geometry), join = st_intersects, left = FALSE)
```

Now we can project this dataset to WGS84 (EPSG 4326), extract the longitude and latitude for each survey point, and convert back to a regular dataset:

```r
ea_nfpd <- ea_nfpd_s %>%
  st_transform(crs = 4326) %>%
  st_coordinates() %>%
  as_tibble() %>%
  rename(longitude = X, latitude = Y) %>%
  bind_cols(ea_nfpd_s) %>%
  st_drop_geometry()
```

Now we can process this dataset. For the ‘site’ variable, there are two levels of site identification - the local survey site (identified by `site_id`, typically on the order of 100m river length) and a larger scale ‘parent’ site (`site_parent_id`), which may constitute several surveys within the same local area (typically all within a few km of each other). We retain both of these in a composite ‘site’ variable, meaning that the data are at the level of the site but that aggregation to parent site remains possible if required. Abundance is measured as the total number of fish sampled. This is done over one or more survey runs at each site in each time period. For around half the surveys, estimates of total population density are available by using the Carle & Strub equation (Carle & Strub 1978) over a three run catch depletion survey. However, this method can only be used on multiple run surveys, which would result in discarding over half available surveys, and so we take as our abundance measure the counts from the first run of the multiple run surveys, or the only run from single run surveys (as is done for Water Framework Directive classification; Philip Rudd, Fisheries Technical Specialist, Environment Agency, pers. comm. April 2023). Counts are then divided by the survey area (ength of river fished multiplied by the average width) to give abundance as individuals per 100 m^2. For some surveys, exact counts are not given - these are excluded. Zero catches are recorded in a separate variable; these are converted to 0s in our abundance variable.

```r
ea_nfpd <- ea_nfpd %>%
  mutate(dataset_id = "EA_NFPD",
         site = paste(site_parent_id, site_id, sep = "_"),
         country = "GBR",
         abundance = ifelse(zero_catch == "Yes", 0, run1),
         abundance = abundance / (survey_area / 100),
         unit = "individuals_100m2") %>%
  filter(!is.na(abundance)) %>%
  rename(date = event_date_year, species = latin_name) %>%
  select(dataset_id, site, country, latitude, longitude, date,
         species, abundance, unit)
```

This results in 279099 observations. Write this processed dataset to the derived data folder:

```r
write_csv(ea_nfpd, here::here("data/derived_data", "EA_NFPD_processed.csv"))
```

FishGlob Global Bottom Trawl Survey Database

FishGlob_data is a global database of bottom trawl survey data for marine fish, described in Maureaud et al. (2023). The database contains a cleaned collation of 26 publicly available bottom-trawl surveys conducted in national waters of 18 countries that are standardised and pre-processed, covering a total of 2,162 sampled fish taxa and 232,800 hauls collected from 1963 to 2020. The database is available from Zenodo at https://zenodo.org/record/7527447#.ZDhrFuzMIqt.

First, download the full zipped database from Zenodo, and unzip. Because this is a large dataset even when compressed, it may case `download.file` to time out, so we first set a 10 minute timeout, before downloading, unzipping, and removing the zipped archive:

```
options(timeout = max(600, getOption("timeout")))

download.file(
  "https://zenodo.org/record/7527447/files/AquaAuma/FishGlob_data-v1.9.zip?download=1",
              destfile = here::here("data", "raw_data/Global_fish_data.zip"))

unzip(here::here("data", "raw_data/Global_fish_data.zip"),
      exdir = here::here("data/raw_data", "Global_fish_data"))

unlink(here::here("data", "raw_data/Global_fish_data.zip"))
```

The full clean standardised dataset is found in the 'ouputs' subfolder as a .RData binary file, 'FishGlob_std_public_clean.RData'. This contains site level estimates of abundance per km2 from multiple bottom trawl surveys across multiple species. Load the data:

```
load(here::here("data/raw_data/Global_fish_data/AquaAuma-FishGlob_data-ea56c56/outputs/Compiled_data",
                "FishGlob_public_std_clean.RData"))
```

This includes two objects, the main data as `data`, and meta-data in `readme`. Country is included in the dataset, but as country name. To convert to country code, create a dataframe of distinct countries, and use the `countrycode` (Arel-Bundock et al. 2018) package to obtain relevant ISO3 codes (all surveys listed as 'multi-countries' are in Europe, so we use EUROPE as a code for these). Then join these codes back to the main dataset:

```
countries <- data %>%
  select(country) %>%
  distinct() %>%
  mutate(country_code =
           countrycode::countrycode(country,
                                    origin = "country.name",
                                    destination = "iso3c",
                                    custom_match = c(`multi-countries` = "EUROPE")
                                    )
         )

data <- data %>% left_join(countries, join_by(country))

data %>% count(country_code)
```

```
## # A tibble: 8 x 2
##   country_code       n
##   <chr>          <int>
## 1 CAN           397854
## 2 EUROPE        451253
## 3 FRA            92965
## 4 GBR           121320
## 5 IRL            69350
## 6 NOR           250562
## 7 PRT            13821
## 8 USA          1556532
```

Now process the data. As these abundances are derived through trawling, there is no discrete spatial repetition in sampling i.e. the exact same site is not sampled every year. Instead, the trawlers collecting data can deviate slightly from the exact site, but do often stay within the same general region/area. To handle these discrepancies in sampling location, we upscale the sampling locations (latitude and longitude) to a 1-degree resolution and take the average abundance estimates from the same sampling scheme,for each species in each year within this 1-degree gird cell. Simply put, we upscale the spatial resolution to allow temporal comparisons in trawling data. Note: A 1-degree resolution was decided on as it allowed us to develop these temporal comparisons whilst maintaining spatial structure

```
fish_survey <- data %>%
  filter(station != "<NA>" | stat_rec != "<NA>") %>%
  mutate(dataset_id = "FishGlob",
         lat_round = round(latitude,0),
         lon_round = round(longitude,0),
         site = paste(survey_unit, lat_round, lon_round),
         unit = "individuals_km2") %>%
    filter(!is.na(num_cpua) & num_cpua != Inf & num_cpua >= 0) %>%
    select(-country) %>%
    rename(country = country_code, date = year,
         species = accepted_name, abundance = num_cpua) %>%
    group_by(dataset_id, site, country, lat_round, lon_round, date,
         species) %>%
    summarise(abundance = mean(abundance, na.tm = T)) %>%
    rename(latitude = lat_round, longitude = lon_round) %>%
    mutate(unit = "avg_individuals_km2") %>%
    ungroup()
```

```
## `summarise()` has grouped output by 'dataset_id', 'site', 'country',
## 'lat_round', 'lon_round', 'date'. You can override using the `.groups`
## argument.
```

This processed dataset contains 727128 observations, and can be written to file as:

```
write_csv(fish_survey,
          here::here("data/derived_data", "fishglob_processed.csv"))
```

TimeFISH

The TimeFISH database (Quimbayo et al. 2022) provides the first public time-series dataset on reef fish assemblages in the southwestern Atlantic (SWA), comprising 15 years of data (2007–2022) based on standardized Underwater Visual Censuses (UVCs) in nine locations along the southern Brazilian coast (25–29°S). All fish individuals in the water column (up to 2m above the substratum) and at the bottom were targeted. In total, 202,965 individuals belonging to 163 reef fish species and 53 families were recorded across 1857 UVCs. Data are available to use with no restrictions, and can be downloaded from Zenodo: https://zenodo.org/record/7317084#.ZFJRK-zMIqs

First, download the zipped database, extract, and remove the zipped version:

```
download.file("https://zenodo.org/record/7317084/files/quimbayo-jp/TimeFISH-Summary.zip?download=1",
              destfile = here::here("data", "raw_data/TimeFISH-Summary.zip"))

unzip(here::here("data", "raw_data/TimeFISH-Summary.zip"),
      exdir = here::here("data", "raw_data/TimeFISH"))

unlink(here::here("data", "raw_data/TimeFISH-Summary.zip"))
```

Check the contents of the downloaded folder:

```
list.files(here::here("data", "raw_data/TimeFISH"), recursive = TRUE)
```

```
## [1] "quimbayo-jp-TimeFISH-5bab403/ECY22-0768_TimeFISH_census_data.csv"
## [2] "quimbayo-jp-TimeFISH-5bab403/ECY22-0768_TimeFISH_location_information.csv"
## [3] "quimbayo-jp-TimeFISH-5bab403/ECY22-0768_TimeFISH_taxonomic_information.csv"
## [4] "quimbayo-jp-TimeFISH-5bab403/README.md"
```

Read in the main census data:

```
timefish_dat <- read_csv(here::here("data/raw_data",
                                    "TimeFISH/quimbayo-jp-TimeFISH-5bab403/ECY22-0768_TimeFISH_census_da
```

```
## Rows: 27377 Columns: 5
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr (3): transect_id, species_name, species_code
## dbl (2): total_length, abundance
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

Read in the location data:

```
timefish_loc <- read_csv(here::here("data/raw_data",
                                    "TimeFISH/quimbayo-jp-TimeFISH-5bab403/ECY22-0768_TimeFISH_location_
```

```
## Rows: 1957 Columns: 24
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr  (12): transect_id, province, state, location, site, thermal_category, n...
## dbl  (10): longitude, latitude, ntransect, sampling_area_m2, day, year, dept...
## time  (2): start_time, end_time
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

Process the data. First, aggregate census data to species per transect (as we do not need individual size class abundances). Then fomat species names, and join to location data. Note - two transect IDs appear twice in the location data:

```
timefish_loc %>% count(transect_id) %>% arrange(desc(n))
```

```
## # A tibble: 1,955 x 2
##    transect_id     n
##    <chr>       <int>
##  1 2019_86         2
##  2 2019_91         2
##  3 2007_1          1
##  4 2007_10         1
##  5 2007_11         1
##  6 2007_12         1
##  7 2007_13         1
##  8 2007_14         1
##  9 2007_15         1
## 10 2007_16         1
## # i 1,945 more rows
```

It looks like this is because they were sampled on successive days - they are otherwise identical:

```
timefish_loc %>% filter(transect_id %in% c("2019_86", "2019_91"))
```

```
## # A tibble: 4 x 24
##   transect_id province state        location site  thermal_category no_take_zone
##   <chr>       <chr>    <chr>        <chr>    <chr> <chr>            <chr>
## 1 2019_86     swa      santa_catar~ arvored~ ranc~ warmer           yes
## 2 2019_86     swa      santa_catar~ arvored~ ranc~ warmer           yes
## 3 2019_91     swa      santa_catar~ arvored~ ranc~ warmer           yes
## 4 2019_91     swa      santa_catar~ arvored~ ranc~ warmer           yes
## # i 17 more variables: longitude <dbl>, latitude <dbl>, ntransect <dbl>,
## #   observer <chr>, sampling_method <chr>, sampling_area_m2 <dbl>, day <dbl>,
## #   month <chr>, year <dbl>, sampling_season <chr>, depth <dbl>,
## #   temperature <dbl>, complexity <dbl>, visibility <dbl>, start_time <time>,
## #   end_time <time>, biotop <chr>
```

As we are not interested in dates beyond years, we can disregard this in joining below by setting `multiple = "first"`. Then add dataset_id, create new site variable, add country code and unit, rename variables as needed and select the final set:

```
timefish_dat <- timefish_dat %>%
  group_by(transect_id, species_name) %>%
  summarise(abundance = sum(abundance), .groups = "drop") %>%
  mutate(species_name = str_replace_all(species_name, pattern = "_", replacement = " "),
         species_name = str_to_sentence(species_name)) %>%
  left_join(select(timefish_loc, transect_id, state, location, site, longitude, latitude, year),
            join_by(transect_id), multiple = "first") %>%
  mutate(dataset_id = "TimeFISH",
         site = paste(state, location, site, sep = "_"),
         country = countrycode::countrycode("Brazil",
                                           origin = "country.name",
                                           destination = "iso3c"),
         unit = "n_individuals"
  ) %>%
  rename(date = year, species = species_name) %>%
  select(dataset_id, site, country, latitude, longitude, date, species, abundance, unit)
```

This now has 19787 observations. Write this to the derived data folder:

```
write_csv(timefish_dat,
          here::here("data/derived_data", "timefish_processed.csv"))
```

Pilotto

In 2020, Pilotto published a study on biodiversity trends in Europe, largely relaying on Europe's network of 'long-term ecological research sites'. Unlike the other datasets compiled in this Rmarkdown, Pilotto does not provide the full dataset, instead only sharing links to the raw data. To make use of the Pilotto data, we compiled this raw data into a series of .csv files. Specifically, this involved working through the links available in Pilotto's data, downloading the raw data from these links into the directory 'data/raw_data/Pilotto/unique_id'. With each downloaded dataset, we then extracted information into three .csv files: compile_ts.csv - this file contains the abundance time-series, reported taxa, year, and name of the site; compile_sp.csv - this file contains the coordinates linked to sites contained in compile_ts.csv; and compile_tx.csv - this file contains species names, and is neccasary as reported taxa in 'compile_ts.csv' sometimes are described with codes instead of their actual name e.g. 'species_xyz123'. 'compile_tx.csv' can be used to convert these codes into actual species names

In some cases, we decided the information used by Pilotto was not suitable for our study (e.g. sometimes data contained presence/absence instead of abundance values). We have carefully annotated the data we extracted from Pilotto within the file 'Pilotto/master.csv'.

```
pilotto_ab <- read_csv(here::here("data/raw_data/Pilotto",
                                  "compile/compile_ts.csv")) %>%
  janitor::clean_names()
```

```
## Rows: 277715 Columns: 10
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr (5): sub_site, site, region, reported_taxa, unit
## dbl (5): unique_id, year, month, day, abundance
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
pilotto_sp <- read_csv(here::here("data/raw_data/Pilotto",
                                  "compile/compile_sp.csv")) %>%
  janitor::clean_names()
```

```
## Rows: 1464 Columns: 5
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr (2): site, country
## dbl (3): unique_id, latitude, longitude
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
pilotto_tx <- read_csv(here::here("data/raw_data/Pilotto",
                                  "compile/compile_tx.csv")) %>%
  janitor::clean_names()
```

```
## Rows: 14443 Columns: 3
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr (2): reported_taxa, corrected_taxa
## dbl (1): unique_id
##
## i Use ‘spec()‘ to retrieve the full column specification for this data.
## i Specify the column types or set ‘show_col_types = FALSE‘ to quiet this message.
```

```
pilotto_ab = left_join(pilotto_ab, pilotto_sp, by = c("unique_id", "site"))
```

```
## Warning in left_join(pilotto_ab, pilotto_sp, by = c("unique_id", "site")): Detected an unexpected man
## i Row 34802 of ‘x‘ matches multiple rows in ‘y‘.
## i Row 1 of ‘y‘ matches multiple rows in ‘x‘.
## i If a many-to-many relationship is expected, set ‘relationship =
##   "many-to-many"‘ to silence this warning.
```

```
pilotto_ab = subset(pilotto_ab, !is.na(latitude))
pilotto_ab = left_join(pilotto_ab, pilotto_tx[,c(2,3)], by = c( "reported_taxa"))
```

```
## Warning in left_join(pilotto_ab, pilotto_tx[, c(2, 3)], by = c("reported_taxa")): Detected an unexpe
## i Row 10 of ‘x‘ matches multiple rows in ‘y‘.
## i Row 207 of ‘y‘ matches multiple rows in ‘x‘.
## i If a many-to-many relationship is expected, set ‘relationship =
##   "many-to-many"‘ to silence this warning.
```

```
pilotto_ab$corrected_taxa = ifelse(is.na(pilotto_ab$corrected_taxa), pilotto_ab$reported_taxa, pilotto_a
```

Next we tidy the Pilotto data, removing columns that arent needed and renaming all columns to meet the standards set in the other datasets

```
pilotto_ab <- pilotto_ab %>%
  mutate(site = paste(unique_id, site), dataset_id = "Pilotto") %>%
  select(-unique_id, - sub_site, -region, -month, -day, -reported_taxa) %>%
  rename(date = year, species = corrected_taxa) %>%
  select(dataset_id, site, country, latitude, longitude, date, species, abundance, unit)
```

This results in a dataset of 4632 observations, which is written to the derived data folder:

```
write_csv(pilotto_ab, here::here("data/derived_data", "pilotto_processed.csv"))
```

Data Compilation

To compile the full dataset, first get the filepaths of all the processed datasets:

```
processed_files <- list.files(path = here::here("data/derived_data"),
                              pattern = "*processed.csv",
                              full.names = TRUE)
```

Set the variable types for all expected variables:

```r
variable_type <- cols(
  "dataset_id" = col_character(),
  "site" = col_character(),
  "country" = col_character(),
  "latitude" = col_double(),
  "longitude" = col_double(),
  "date" = col_integer(),
  "species" = col_character(),
  "abundance" = col_double(),
  "unit" = col_character()
)
```

And read in all datasets into a single dataframe:

```r
compiled_data <- processed_files %>%
  map_df(~read_csv(., col_types = variable_type))
```

Because this is a large dateset (approximately 1.2GB), we export it in parquet format using the `arrow` package (Richardson et al. 2023):

```r
library(arrow)
```

Then use `write_parquet` to export the data:

```r
write_parquet(compiled_data,
              sink = here::here("data/derived_data", "compiled_data.parquet"))
```

The resulting file is considerably smaller (<100MB) and can be read in rapidly as a standard data frame using `arrow::read_parquet`, then converted to a tibble, using:

This is a dataset with 15,805,051 observations across the 9 variables. There are 36,373 species, from 219 countries spanning -89 to 89.7 degrees of latitude and -180 to 180 degrees of longitude, from 1820 to 2023. # References

Arel-Bundock et al. (2018) *countrycode: An R package to convert country names and country codes.* Journal of Open Source Software https://doi.org/10.21105/joss.00848

Carle F.L. & Strub M.R. (1978) *A new method for estimating population size from removal data.* Biometrics https://doi.org/10.2307/2530381

Comte L. et al. (2020) *RivFishTIME: A global database of fish time-series to study global change ecology in riverine systems.* Global Ecology and Biogeography https://doi.org/10.1111/geb.13210

Dornelas M. et al. (2018) *BioTIME: A database of biodiversity time series for the Anthropocene.* Global Ecology and Biogeography https://doi.org/10.1111/geb.12729

Environment Agency (2020) *Freshwater fish surveys (NFPD)* Open Government Licence, available from https://environment.data.gov.uk/ecology/explorer/downloads/

Firke S. (2023) *janitor: Simple Tools for Examining and Cleaning Dirty Data.* R package version 2.2.0, https://CRAN.R-project.org/package=janitor

Jandt, U. et al. (2022) *ReSurvey Germany: vegetation-plot resurvey data from Germany [Dataset].* iDiv Data Repository. https://doi.org/10.25829/idiv.3514-0qsq70

Johnson T.F. et al. (2022) *CaPTrends: A database of large carnivoran population trends from around the world.* Global Ecology and Biogeography https://doi.org/10.1111/geb.13587

LPI 2022. *Living Planet Index database.* www.livingplanetindex.org/. Downloaded on 5th April 2023

Maureaud, A.A. et al, (2023) *FishGlob_data: an integrated database of fish biodiversity sampled with scientific bottom-trawl surveys.* https://doi.org/10.31219/osf.io/2bcjw

Müller K. (2020) *here: A Simpler Way to Find Your Files.* R package version 1.0.1, https://CRAN.R-project.org/package=here

Pebesma, E. (2018) *Simple Features for R: Standardized Support for Spatial Vector Data.* The R Journal https://doi.org/10.32614/RJ-2018-009

Quimbayo, J.P. et al. (2022) *TimeFISH: Long-term assessment of reef fish assemblages in a transition zone in the Southwestern Atlantic.* Ecology https://doi.org/10.1002/ecy.3966

R Core Team (2022) *R: A language and environment for statistical computing.* R Foundation for Statistical Computing, Vienna, Austria. https://www.R-project.org/

Richardson, N. et al. (2023) *arrow: Integration to 'Apache' 'Arrow'.* R package version 11.0.0.3, https://CRAN.R-project.org/package=arrow

Wickham H. et al. (2019) *Welcome to the tidyverse.* Journal of Open Source Software https://doi.org/10.21105/joss.01686

Ziolkowski Jr. et al. (2022) *North American Breeding Bird Survey Dataset 1966 - 2021* U.S. Geological Survey data release, https://doi.org/10.5066/P97WAZE5

Reproducibility

Date rendered

```
## [1] "2023-10-17 17:14:40 BST"
```

Session info

```
## R version 4.2.3 (2023-03-15)
## Platform: aarch64-apple-darwin20 (64-bit)
## Running under: macOS Monterey 12.1
##
## Matrix products: default
## BLAS:   /Library/Frameworks/R.framework/Versions/4.2-arm64/Resources/lib/libRblas.0.dylib
## LAPACK: /Library/Frameworks/R.framework/Versions/4.2-arm64/Resources/lib/libRlapack.dylib
##
## locale:
## [1] en_US.UTF-8/en_US.UTF-8/en_US.UTF-8/C/en_US.UTF-8/en_US.UTF-8
##
## attached base packages:
## [1] stats     graphics  grDevices utils     datasets  methods   base
##
## other attached packages:
##  [1] arrow_12.0.1      sf_1.0-13         countrycode_1.5.0 janitor_2.2.0
##  [5] here_1.0.1        lubridate_1.9.2   forcats_1.0.0     stringr_1.5.0
##  [9] dplyr_1.1.2       purrr_1.0.1       readr_2.1.4       tidyr_1.3.0
## [13] tibble_3.2.1      ggplot2_3.4.2     tidyverse_2.0.0
##
## loaded via a namespace (and not attached):
##  [1] tidyselect_1.2.0   xfun_0.39          snakecase_0.11.1   colorspace_2.1-0
##  [5] vctrs_0.6.3        generics_0.1.3     htmltools_0.5.5    yaml_2.3.7
##  [9] utf8_1.2.3         rlang_1.1.1        e1071_1.7-13       pillar_1.9.0
## [13] glue_1.6.2         withr_2.5.0        DBI_1.1.3          bit64_4.0.5
## [17] pryr_0.1.6         lifecycle_1.0.3    munsell_0.5.0      gtable_0.3.3
## [21] codetools_0.2-19   evaluate_0.21      knitr_1.43         tzdb_0.4.0
## [25] fastmap_1.1.1      class_7.3-21       parallel_4.2.3     fansi_1.0.4
## [29] Rcpp_1.0.11        KernSmooth_2.23-20 classInt_0.4-9     scales_1.2.1
## [33] vroom_1.6.3        lobstr_1.1.2       bit_4.0.5          hms_1.1.3
## [37] digest_0.6.33      stringi_1.7.12     grid_4.2.3         rprojroot_2.0.3
## [41] cli_3.6.1          tools_4.2.3        magrittr_2.0.3     proxy_0.4-27
## [45] crayon_1.5.2       pkgconfig_2.0.3    timechange_0.2.0   assertthat_0.2.1
## [49] rmarkdown_2.23     rstudioapi_0.15.0  R6_2.5.1           units_0.8-2
## [53] compiler_4.2.3
```