Data Curation Workshop

BioTIME Core Team

Updated June 29 2023

Introduction

BioTIME (http://biotime.st-andrews.ac.uk) is an open-access database for long-term assemblage data contributed from research groups globally. The power of our database is quite literally in the numbers. We currently have:

443 studies

12,964,172 records

681,830 locations

49,464 species

But in order for data to be useful and comparable in our database, we need to check and adapt datasets to a common standard and format as they come in from various researchers and institutions. This primarily falls to the task of data curation. This workshop will walk you through adapting a dataset into the common BioTIME dataset format.

Objectives

Hopefully, by the end of this workshop, you'll be able to:

- Learn how to validate and check large datasets using R
- Use data wrangling to rectify any issues that come up in data checking
- Aggregate and omit data records, join and manipulate data tables
- Integrate version control using the RStudio Git interface

Definitions

Before we start, here are some terms we use to describe dataset/sampling design elements:

- **Record**: numerical abundance (or biomass, % cover) of species per observation event (i.e. per sample); might be necessary to pool abundance of different life stages, sizes or sex (same for biomass).
- **Sampling event / Sample ID/ Sample**: An event at a specific place and time where assemblages were observed and can't be broken further down with existing information. It gives information about sampling effort and how a study's sampling design is structured and "nested". We show this information by combining the different levels into a string called **sample description** (e.g. `studyID100_Quadrat1_Transect2_BigIsland_summer_2009`). This string distinguishes different samples from each other. **Sample name** gives information about what fields construct a **sample description**, like `quadrat_transect_island_season_year` for the string before.
- **Site**: smallest possible area that has been repeatedly sampled through time in the dataset, usually a sum of all survey locations within a dataset.
- **Plot**: A **permanent** location with a defined area that has been revisited for surveying through time. Anything that is not a permanent plot like a general study location or information about transects, traps, etc. goes into Sample
- **Grain size**: The area covered by each sampling event. If a dataset incorporates plots, the area of the plot would be the grain size. At the site level, we

often need this provided through methods.
If a site contains plots that have all been sampled consistently through time (e.g. some plots were not left out/added in some years), we still call the aggregate of plots as a site.

Set up working environment

let's load our necessary packages and import the dataset we'll be working on. Depending on your coding style and preferences, we'll be showing examples with both base R and `tidyverse` functions, but some tasks like taxonomy fixes will rely more heavily on packages. If you don't have these packages, run `install.packages(c('tidyverse', 'readxl', 'maps'))` in your console.

Download this dataset is on arachnids/spiders from the Hoge Kempen National Park located in Belgium: click here (https://universityofstandrews907-my.sharepoint.com/:x:/g/personal/fycc1_st-andrews_ac_uk/ETyEHl_jDy5OuNMwIczd60MBF445_UP5_p5NBG2ERNrxMA?e=ojuIPt)

```
# load the required packages
# you can tailor your own package preferences and functions used once you get used to curating!
require(dplyr)
require(ggplot2)
require(stringr)
require(readxl)
require(maps)

# import our data from the Excel spreadsheet
dt <- read_excel('./example_single/S007.xlsx', sheet=1, skip=3, col_names=T, na='')
# skip = 3 is to avoid unnecessary metadata at the top which we don't want to read

#View(dt) # always do a visual check :)
```

Dataset criteria

When we get a contributed dataset, we first check whether they fit our criteria. Often, this can be done by scanning the relevant methods provided in the dataset metadata. Previously, good datasets have come from data integration sources such as OBIS, GBIF, Ecological Archives and data papers from reputable journals.

1. Dataset consists of at least 2 years of sampling (they do not have to be consecutive).
2. Dataset consists of entire assemblages, not just populations (*sensu lato*, i.e. does not exclude some taxa intentionally).
3. Data should record abundance, biomass or both.
4. Sampling methods *and effort* are consistent through time.
5. Individuals are identified to species level where possible.

Here are the detailed methods provided to use by the dataset author. Do you think they fit our criteria?

> Pitfall traps were installed in different UTM plots in the NPHK (50 km^2), which consists different habitats (see description habitats). Traps were left on location for approximately one month, then emptied and refilled. Per location spring - summer, but not every location yearly (see dataset). In different UTM grids, different habitats were sampled several times over the period of 30 years (1986 - 2017) This dataset includes only heathland from the UTM FS8546 grid for temporal regularity (see Pilotto et al. https://doi.org/10.1038/s41467-020-17171 (https://doi.org/10.1038/s41467-020-17171))

Let's use R to help us check the first criteria:

```
# are there 2 or more unique years in the dataset?
n_distinct(dt$year) >= 2
```

```
## [1] TRUE
```

```
# n_distinct is identical to length(unique(x)) in base R, which counts the unique values
```

Target format

Also, before we move forward, have a look at the data template/format we want this dataset to reflect. Familiarising yourself with this will be helpful to know what fields need to be transformed, added, or omitted to fit this. Even if the dataset does not have all the data to fill these fields, they need to be put in as blank columns. * *StudyID is left as a blank column to be automatically filled in when we add this to the database server. Don't worry about it.*

Table 1: BioTIME raw data template format.

	A	B	C	D	E	F	G	H	I	J	K	L	M	N
1	**Abundance**	**Biomass**	**Family**	**Genus**	**Species**	**SampleDesc**	**Plot**	**Latitude**	**Longitude**	**DepthElevation**	**Day**	**Month**	**Year**	**StudyID**
2						(leave blank)								(leave blank)
3														
4														

Overall checks

And like most work dealing with datasets, it's good to look at the basic structure before we do anything. I'll show you how we check this, but try and think about how you might fix these issues before clicking onto the fix tab.

Check Fixes

```
dim(dt) # check dimensions
```

```
## [1] 1746    7
```

```
# 1746 records
str(dt) # check structure
```

```
## tibble [1,746 × 7] (S3: tbl_df/tbl/data.frame)
##  $ Site     : chr [1:1746] "FS8546" "FS8546" "FS8546" "FS8546" ...
##  $ Habitat  : chr [1:1746] "HEI" "HEI" "HEI" "HEI" ...
##  $ year     : num [1:1746] 1986 1987 1988 1988 1988 ...
##  $ Group    : chr [1:1746] "ARACHNIDA" "ARACHNIDA" "ARACHNIDA" "ARACHNIDA" ...
##  $ Family   : chr [1:1746] "Theridiidae" "Theridiidae" "Theridiidae" "Linyphiidae" ...
##  $ Taxon    : chr [1:1746] "Pholcomma gibbum" "Pholcomma gibbum" "Euryopsis flavomaculata" "Allomengea scopigera"
...
##  $ Abundance: num [1:1746] 1 1 1 1 1 1 1 1 1 ...
```

```
summary(dt)
```

```
##      Site             Habitat              year         Group
##  Length:1746        Length:1746        Min.   :1986   Length:1746
##  Class :character   Class :character   1st Qu.:1990   Class :character
##  Mode  :character   Mode  :character   Median :2001   Mode  :character
##                                        Mean   :1999
##                                        3rd Qu.:2001
##                                        Max.   :2013
##     Family             Taxon             Abundance
##  Length:1746        Length:1746        Min.   : 1.000
##  Class :character   Class :character   1st Qu.: 1.000
##  Mode  :character   Mode  :character   Median : 1.000
##                                        Mean   : 3.651
##                                        3rd Qu.: 3.000
##                                        Max.   :71.000
```

To make sure fields behave properly if we manipulate them, they have to be of the right data type:

- **Abundance or biomass**: numeric
- **Coordinates**: numeric
- **Dates**: POSIXct or
- **Year, month, day* columns**,: integers or factors
- **Plot**: factors or integers
- **DepthElevation**: numeric or factors (if they're a treatment category)
- **Taxonomic**: characters or factors

```
# Abundance and/or biomass, latitude and longitude numeric?
# Deal with coordinates later
summary(dt)
```

```
##      Site             Habitat              year         Group
##  Length:1746        Length:1746        Min.   :1986   Length:1746
##  Class :character   Class :character   1st Qu.:1990   Class :character
##  Mode  :character   Mode  :character   Median :2001   Mode  :character
##                                        Mean   :1999
##                                        3rd Qu.:2001
##                                        Max.   :2013
##     Family             Taxon             Abundance
##  Length:1746        Length:1746        Min.   : 1.000
##  Class :character   Class :character   1st Qu.: 1.000
##  Mode  :character   Mode  :character   Median : 1.000
##                                        Mean   : 3.651
##                                        3rd Qu.: 3.000
##                                        Max.   :71.000
```

```
# Year, month and day must be integers or factors

# Secondary fields such as trawl, plot, transect etc must be factors or integers? NA
# Secondary fields such as elevation or depth will normally be numeric unless in a treatment format. NA here
# Date should be POSIXct (not applicable in this case) NA, just year

# Taxonomic fields must be characters or factors?
```

Primary fields

To check primary fields (abundance, coordinates, and dates), we just need to make sure that:

- Fields don't contain blanks, zeroes, negative values, NAs, or NaNs
- Fields are logical, real values, i.e. within possible limits
- We need to pool abundances for different sexes or life-forms (adults & juveniles). An exception to this are amphibia (anurans in particular) because the methodology to sample adults & tadpoles can be very different. If the authors submit the data separately, we keep it this way.

```
# Abundance
min(dt$Abundance) > 0 # no zeroes?
```

```
## [1] TRUE
```

```
sum(dt$Abundance=="") > 0 # no blanks
```

```
## [1] FALSE
```

```
# Year < 2021, month < 12, day < 31
summary(dt[,1])
```

```
##       Year
##  2001   :626
##  2013   :330
##  1989   :255
##  1988   :155
##  1990   :127
##  1992   :120
##  (Other):133
```

```
# if there are rows that need to be removed
# dt <- dt[!is.na(dt$Abundance),]
# dt <- dt[!which(dt$Abundance == 0),]
# or with dplyr's filter() function, which can handle multiple conditions
# dt <- dt %>% filter(!Abundance == '' | Abundance == 0 | is.na(Abundance))
```

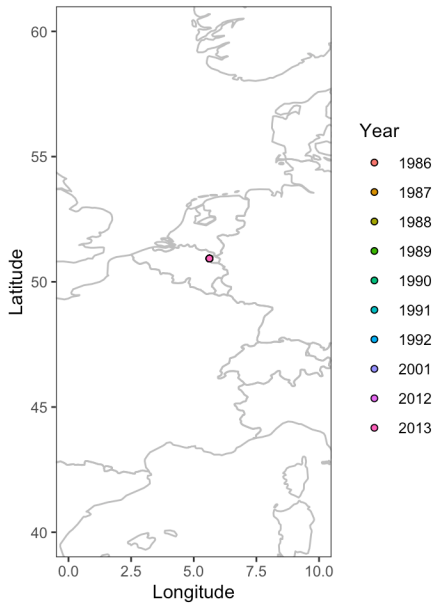
This dataset does not separate abundances by sex or ontogeny, but if you come across a dataset like that, just remove the columns that specify sex/ontogeny and the records will get merged into one per species later on.

For coordinates in latitude and longitude, we work primarily in WGS84, so latitude has to be constrained between -90 to 90, and longitude from 0 to 180. Our dataset contributor only provided coordinates `50.931, 5.626` for the entire site, so we'll just manually put them in.

```
dt$Latitude <- 50.931
dt$Longitude <- 5.626
```

We'll also plot this to visually confirm that the coordinates are as expected. Often times people might switch longitude with latitude and still be logical, real coordinates to our code. We're also coloring them by year so that if any color shows up in isolation from the group, that means that location does not have a complete coverage through time and will need more checks.

Plot Code



Secondary fields

Sometimes data will have trawl, plot, depth, bottom depth or similar secondary fields. They must be inspected but we don't always want to remove records that have errors in these fields. It will depend on the type of error.**The premise when tidying these fields is that values make sense and are in accordance with the source methods description.** It'll often vary on a case by case basis and require executive judgement by the curator.

Also make sure to keep in columns that are necessary for constructing your sampling event description

Criteria:

- If the error (e.g. a NULL) makes it impossible to assign the value to a group, then remove it.
- If the error is a misspelling or a missing value that can be clearly re-assigned by reading the methods, then do not remove the record and fix it in the appropriate way.

```
# Plot and treatment must be inspected for NA, NULL, blank or values unspecified in source methods
# We must check for misspellings and revalue levels if needed

# For this, we can use:
sort(unique(dt$plot))
# and rename levels using:
levels(dt$plot) <- plyr::revalue(levels(dt$plot),
                                 c("old_name"="new_name",
                                   "old_name1"="new_name1"))

# Repeat processes used for primary fields.
# DepthElevation can be 0, but not NaN, NA, NULL or blank only in some records.
```

Taxonomic fields

Now, we check the taxonomic fields. From experience, this is where the bulk of data curation efforts go toward because taxonomy/species IDs are messy and data aren't perfect. `stringr` is a great package to use for automating detecting patterns in text strings. It comes in handy for replacements too. I refer to this cheatsheet (http://edrub.in/CheatSheets/cheatSheetStringr.pdf) a lot. But for visual checking of typos, this is largely by eye.

Cleaning and checking objectives

- Eliminate blanks, NAs (but uncertain IDs up to family level are allowed), and zeroes.
- Fit uncertain IDs to our syntax or eliminate abbreviations that dataset contributors used.
- Fit proper taxa level with the correct fields (`Family, Genus, Species`)
- Remove non-organismal records (e.g. rock, sand) and incidental taxa records.
- Check that genus column doesn't have family names (end in *-eae* or *-idae*) or species column has genus names.
- Re-allocate taxonomic classifications into the appropriate column.
- Check for duplication and misspellings in the species names (e.g. *Puffnius sp*, *Puffinus sp*; unidentified/unknown; *Buenia jeffreysi / Buenia jeffreysii*; double spacing; several spaces, authors name & date, etc.). We're not validating or updating species, but we do quick searches in Google Scholar to figure out which spelling is correct.
- Subspecies and annotated species names should be disregarded (e.g. *Amphiura (acrocnida) brachiate* should be considered as *Acrocnida brachiate*)
- If a record only has data until the genus level, it must have "sp" in species column.

Nomenclature

Uncertain or unidentified IDs: "Genus sp" (without the full stop/period). If the dataset distinguishes between different uncertain IDs, we reformat them while making sure they're recognised as separate species: like "Genus sp1" and "Genus sp2"
No subspecies or varieties: Lowest level = "Genus species"
Species groups: Some taxa are less resolved and are placed into species complexes, e.g. *Dascyllus trimaculatus agg.* or *Rubus fruticosus agg.*. We denote this with

“agg.” to make sure it’s not confused with other specific epithets. Uncertain IDs with several potential options do not count.

Reminder: For `Species` , we only leave in species epithets! Don’t put it into the usual *Genus species* convention.

```
dt$Species <- dt$Taxon # always work from a copy
# check that genera are genera, not family names (-idae/eae)
# this returns the record index number if there are any
str_which(dt$Species, 'idae$|eae$')
```

```
## integer(0)
```

```
# check the species list for misspellings or non-BioTIME taxonomic convention names
# Do visual checks before splitting taxa up into the three columns.
sort(unique(dt$Species)) %>% word(., start=2, end=-1)
```

```
##   [1] "labyrinthica"       "brunnea"            "dentigera"
##   [4] "proxima"            "decora"             "rurestris"
##   [7] "saxatilis"          "subtilis"           "scopigera"
##  [10] "barbipes"           "cuneata"            "pulverulenta"
##  [13] "elegans"            "misera"             "leopardus"
##  [16] "phalerata"          "affinis"            "albimana"
##  [19] "approximatus"       "gracilis"           "nigrinus"
##  [22] "parvulus"           "bicolor"            "concinna"
##  [25] "dilutus"            "pabulator"          "prudens"
##  [28] "sylvaticus"         "cicur"              "comta"
##  [31] "diversa"            "frutetorum"         "germanica"
##  [34] "subtilis"           "terrestris"         "obscurus"
##  [37] "guttata"            "nigrum"             "nigrum brevisetorum"
##  [40] "cristatus"          "picinus"            "concolor"
##  [43] "fimbriatus"         "socialis"           "cupreus"
##  [46] "lapidosus"          "pubescens"          "pusillus"
##  [49] "lutetianus"         "praeficus"          "thoracica"
##  [52] "atra"               "dentipalpis"        "furcata"
##  [55] "frontalis"          "flavomaculata"      "arcuata"
##  [58] "falcata"            "leporina"           "nigerrima"
##  [61] "rubens"             "latebricola"        "vivum"
##  [64] "rufipes"            "helveola"           "montana"
##  [67] "nava"               "dalmatensis"        "signifer"
##  [70] "silvestris"         "umbratilis"         "rubrofasciata"
##  [73] "albovittata"        "sanguinea"          "pullata"
##  [76] "tristis"            "triangularis"       "punctatum"
##  [79] "palpinalis"         "rufus"              "picta"
##  [82] "acalypha"           "nebulosus"          "affinis"
##  [85] "trilobata"          "pulicaria"          "herbigradus"
##  [88] "pusilla"            "viaria"             "pusillus"
##  [91] "lugubre"            "reticulatus"        "adianta"
##  [94] "clathrata"          "gibbosus"           "retusus"
##  [97] "ramosus"            "clercki"            "degeeri"
## [100] "listeri"            "pallidus"           "ericaeus"
## [103] "hortensis"          "lugubris"           "monticola"
## [106] "nigriceps"          "palustris"          "prativaga"
## [109] "pullata"            "tripunctatus"       "ludricum"
## [112] "braccatus"          "histrio"            "gibbum"
## [115] "festivus"           "hygrophilus"        "latitans"
## [118] "piraticus"          "piscatorius"        "tenuitarsis"
## [121] "uliginosus"         "mirabilis"          "juncea"
## [124] "pumila"             "campbelli"          "oblitum"
## [127] "lividus"            "abnormis"           "firma"
## [130] "cingulatus"         "senoculata"         "aurocinctus"
## [133] "lineatus"           "experta"            "setosus"
## [136] "flavipes"           "mengei"             "tenebricola"
## [139] "tenuis"             "zimmermanni"        "cristatus"
## [142] "obtusa"             "striatus"           "minutissima"
## [145] "uhlighi"            "biovatus"           "oblongus"
## [148] "vagans"             "pedestris"          "robusta"
## [151] "ruricola"           "spinipalpis"        "terricola"
## [154] "acuminata"          "alticeps"           "antica"
## [157] "atrotibialis"       "cucullata"          "cuspidata"
## [160] "dysderoides"        "nodosa"             "nudipalpis"
## [163] "obtusa"             "nemoralis"          "audax"
## [166] "cristatus"          "erraticus"          "ferrugineus"
## [169] "latreillei"         "longipes"           "petrensis"
## [172] "subterraneus"       "spinimana"
```

```
# this keeps IDs adjacent to their same-genus misspellings, but only looking at the last few words to check

sort(unique(word(dt$Species, 1))) # check genera
```

```
##   [1] "Agelena"           "Agroeca"           "Agyneta"
##   [4] "Allomengea"        "Alopecosa"         "Antistea"
##   [7] "Aphileta"          "Arctosa"           "Asagena"
##  [10] "Atypus"            "Aulonia"           "Bathyphantes"
##  [13] "Centromerita"      "Centromerus"       "Cicurina"
##  [16] "Clubiona"          "Cnephalocotes"     "Crustulina"
##  [19] "Dicymbium"         "Diplocephalus"     "Diplostyla"
##  [22] "Dolomedes"         "Drapetisca"        "Drassodes"
##  [25] "Drassyllus"        "Drassylus"         "Enoplognatha"
##  [28] "Erigone"           "Ero"               "Euophrys"
##  [31] "Euryopsis"         "Evarcha"           "Gnaphosa"
##  [34] "Gonatium"          "Gongylidiellum"    "Gongylidium"
##  [37] "Hahnia"            "Haplodrassus"      "Hygrolycosa"
##  [40] "Hyposinga"         "Hypsosinga"        "Kaestneria"
##  [43] "Laseola"           "Linyphia"          "Lophomma"
##  [46] "Lophopilio"        "Macrargus"         "Malthonica"
##  [49] "Mangora"           "Megalepthyphantes" "Meioneta"
##  [52] "Mermessus"         "Micaria"           "Micrargus"
##  [55] "Microlinyphia"     "Microneta"         "Minyriolus"
##  [58] "Nemastoma"         "Neon"              "Neoscona"
##  [61] "Neriene"           "Oedothorax"        "Oxyopes"
##  [64] "Pachygnatha"       "Palliduphantes"    "Palludiphantes"
##  [67] "Pardosa"           "Pellenes"          "Peponocranium"
##  [70] "Phaeocedus"        "Philodromus"       "Pholcomma"
##  [73] "Phrurolithus"      "Pirata"            "Pisaura"
##  [76] "Pocadicnemis"      "Porhomma"          "Porrhomma"
##  [79] "Robertus"          "Saaristoa"         "Salticus"
##  [82] "Segestria"         "Sibianor"          "Stemonyphantes"
##  [85] "Tallusia"          "Taranucnus"        "Tenuiphantes"
##  [88] "Tenuiphanthes"     "Tetragnatha"       "Thanatus"
##  [91] "Thenoë"            "Theridion"         "Thyreosthenius"
##  [94] "Tibellus"          "Tiso"              "Trachyzelotes"
##  [97] "Trochosa"          "Walckenaeria"      "Xerolycosa"
## [100] "Xysticus"          "Zelotes"           "Zora"
```

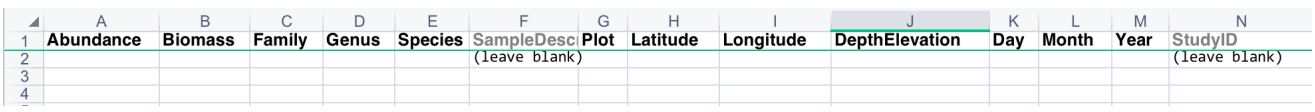
```
sort(unique(dt$Family)) # check family
```

```
##  [1] "Agelenidae"     "Araneidae"      "Atypidae"       "Clubionidae"
##  [5] "Dictynidae"     "Gnaphosidae"    "Hahniidae"      "Linyphiidae"
##  [9] "Liocranidae"    "Lycosidae"      "Mimetidae"      "Opliones"
## [13] "Oplionidae"     "Oxyopidae"      "Philodromidae"  "Pisauridae"
## [17] "Salticidae"     "Segestriidae"   "Tetragnathidae" "Theidiidae"
## [21] "Theridiidae"    "Thomisidae"     "Zoridae"
```

```
replace_g <- c( # make a replacement vector, pattern = replacement
  'Drassylus' = 'Drassyllus',
  'Hyposinga' = 'Hypsosinga',
  'Palludiphantes' = 'Palliduphantes',
  'Porhomma' = 'Porrhomma',
  'Tenuiphanthes' = 'Tenuiphantes'
  )

dt$Genus <- word(dt$Taxon, 1) # separate genus to its own column
dt$Species <- word(dt$Species, start=2) # species to its own column. Only the second word to ignore subspecies

# dt$Species <- str_replace_all(dt$Species, 'sp.$', 'sp$')
dt$Genus <- str_replace_all(dt$Genus, replace_g) # use the replacement vector to replace typos
# dt %>% distinct(Taxon, Genus, Species) %>% View()
# check visually, but do this in your console.

dt$Taxon <- NULL # get rid of it now that we've split and fixed
```

Prepare for export

Now that everything has been checked and fixed, we can prepare the dataframe to look more like our target format and export! As a reminder, this is what we want:



	A	B	C	D	E	F	G	H	I	J	K	L	M	N
1	**Abundance**	**Biomass**	**Family**	**Genus**	**Species**	SampleDesc	**Plot**	**Latitude**	**Longitude**	**DepthElevation**	**Day**	**Month**	**Year**	StudyID
2						(leave blank)								(leave blank)
3														
4														

We have to create a lot of blank columns, aggregate abundances according to site, plot, latitude, longitude, and date. This way we have one data record per species per sampling event. `SampleDescription` will also reflect this unique sampling event in a string.

```
# aggregate abundance records that are same species, plot, and survey day.
dt_merged <- dt %>% group_by_at(vars(-Abundance)) %>%
  summarise(Abundance=sum(Abundance)) %>% ungroup() %>% arrange(Year, Family, Genus, Species)
```

```
## `summarise()` has grouped output by 'Year', 'Family', 'Latitude', 'Longitude',
## 'Species'. You can override using the `.groups` argument.
```

```
dim(dt)[1]-dim(dt_merged)[1] # any change in aggregating?
```

```
## [1] 1229
```

```
dt_merged$Biomass <- ''
dt_merged$Plot <- ''
dt_merged$DepthElevation <- ''
dt_merged$Day <- ''
dt_merged$Month <- ''
dt_merged$StudyID <- ''


# put in as many non-blank fields unique to the sampling event
# this study is a single location study with annual sampling, so only year is needed
# I would fill in SampleType = year
dt_merged$SampleDescription <- dt_merged$Year
n_distinct(dt_merged$SampleDescription) # how many sampling events?
```

```
## [1] 10
```

Sometimes you might have needed to keep some columns needed to construct the SampleDescription. Now that you've made that column, you can get rid of those, maybe using a function like `dt_merged <- dt_merged %>% select(!c(Quadrat, VegetationLayer))`

Now we'll reorder the columns to the proper format:

```
dt_merged <- dt_merged[c('Abundance',
                         'Biomass',
                         'Family',
                         'Genus',
                         'Species',
                         'SampleDescription',
                         'Plot',
                         'Latitude',
                         'Longitude',
                         'DepthElevation',
                         'Day',
                         'Month',
                         'Year',
                         'StudyID')] %>% arrange(Year, Family, Genus, Species)
head(dt_merged) # final check!
```

Abundance	Biomass	Family	Genus	Species	SampleDescription	Plot	Latitude	Longitude	DepthElevation	Day	Month	Year	StudyID
1		Theridiidae	Pholcomma	gibbum	1986		50.931	5.626				1986	
1		Theridiidae	Pholcomma	gibbum	1987		50.931	5.626				1987	
1		Clubionidae	Clubiona	frutetorum	1988		50.931	5.626				1988	
1		Clubionidae	Clubiona	germanica	1988		50.931	5.626				1988	
3		Gnaphosidae	Drassodes	cupreus	1988		50.931	5.626				1988	
3		Gnaphosidae	Drassyllus	lutetianus	1988		50.931	5.626				1988	

Export and spreadsheet prep

The raw data is ready to be exported! Hopefully you were working in your own RProject so working directory is set up.

```
write.csv(dt_merged, 'LTER_HogeKempenNationalPark_Arachnids_rawdata_CC.csv', row.names=F, na='') # replace your initia
        ls here
```

And now we switch over to Excel to fill out the curation spreadsheet. Our database has several data tables for each dataset and they're all linked and cross-referenced by the StudyID. The raw data you just prepared is one data table, but we now have to fill in other data tables on metaData. Most of the metadata will either be found in the source paper or provided by the dataset contributor.

One that we may have to calculate ourselves are the central coordinates (`CentralLatitude, CentralLongitude`) and sampling area (also called Study area in the template). For datasets grained to one location, these fields should already be provided. For datasets where authors provide us with multiple locations, we use R spatial analysis to calculate the central point and area. This can be easily done using a few lines of code to create a convex hull with the multiple locations.

```
# load libraries
library(sf)
library(clipr)

# 1. Convert data points into point spatial object
# assumes your rawdata dataframe is called rawdata
dt_coord <- dt_merged %>% select(Longitude, Latitude) %>% distinct() %>%
   st_as_sf(., coords = c('Longitude', 'Latitude'), crs = 4326) %>% st_union()

# 2. Calculate convex hull, area and centroid
centroid <- dt_coord %>% st_convex_hull() %>% st_centroid() %>% unlist # get centroid
write_clip(centroid[c(2,1)]) # copy as lat-long

# transform to mercator to get area in sq km
area <- st_transform(dt_coord, st_crs("+proj=merc +lon_0=0 +k=1 +x_0=0 +y_0=0 +ellps=WGS84 +datum=WGS84 +units=km +no_
        defs")) %>%
  st_convex_hull() %>% st_area()  ##get area in sq km
write_clip(area)

## also useful snippet if coordinates are ever in degree minutes seconds

angle2dec <- function(angle) {
  angle <- as.character(angle)
  x <- do.call(rbind, strsplit(angle, split=' '))
  x <- apply(x, 1L, function(y) {
    y <- as.numeric(y)
    y[1] + y[2]/60 + y[3]/3600
  })
  return(x)
}

# Plot the geometries ----------------------------------------------------

require(ggplot2)
world_map <- map_data('world') # check whether the GPS coordinates match expectations
ggplot(world_map) +
   geom_polygon(aes(x=long, y=lat, group=group), color='grey', fill='transparent') +
   geom_point(data = dt_merged, aes(x = Longitude, y = Latitude), size = 2, shape = 21, colour = 'blue') +
   geom_point(aes(x = centroid[1], y = centroid[2]), size = 3, colour = 'green') +
   geom_sf(data = dt_coord %>% st_convex_hull(), colour = 'blue', fill = 'transparent') +
   coord_sf(xlim = c(-115, -110), ylim = c(30,35)) + # make sure these fit your coordinates!
   labs(x = NULL, y = NULL) +
   theme_minimal()
```

Final notes

1. Name of the script = name of final data.
2. The script MUST BE STORED in the server.
3. The name of the script goes to “Comments” in the “DataCuration” table.
4. Instructions on how to fill all the rest of the templates are in “BioTIMETemplate.csv”
5. The script to obtain the cent lat and cent long for the MetaDataSite table is in “convexhull.R”.