title: "Formatting popler datasets" author: "Aldo Compagnoni" date: "2018-02-26" output: rmarkdown::html_vignette vignette: > %\VignetteIndexEntry{Introduction to popler} %\VignetteEngine{knitr::rmarkdown}

%\VignetteEncoding{UTF-8}

Start from the metadata

To pick the studies you want to use, start looking at the metadata. Metadata information is obtained from the `browse` function. Browse contains all the metadata from all of the studies currently in popler. Running `browse` simply returns the metadata of all the studies currently in popler:

```
#devtools::install_github("AldoCompagnoni/popler", build_vignettes = TRUE)
library(popler)
browse()
```

```
## # A tibble: 215 x 19
##                        title proj_metadata_key lterid   datatype studytype
##  *                     <chr>             <int>  <chr>      <chr>     <chr>
##  1 SBC LTER: Reef: Kelp Fore                 1    SBC individual       obs
##  2 SBC LTER: Reef: Kelp Fore                 2    SBC      count       obs
##  3 SBC LTER: Reef: Kelp Fore                 3    SBC      count       obs
##  4 SBC LTER: Reef: Kelp Fore                 4    SBC      cover       obs
##  5 SBC LTER: Reef: Long-term                 5    SBC individual       exp
##  6 SBC LTER: Reef: Long-term                 6    SBC      count       exp
##  7 SBC LTER: Reef: Long-term                 7    SBC      count       exp
##  8 SBC LTER: Reef: Long-term                 8    SBC      cover       exp
##  9 SBC LTER: Reef: Long-term                 9    SBC    biomass       exp
## 10 SBC LTER: Reef: Abundance                11    SBC      count       obs
## # ... with 205 more rows, and 14 more variables: duration_years <int>,
## #   community <chr>, studystartyr <dbl>, studyendyr <dbl>,
## #   structured_type_1 <chr>, structured_type_2 <chr>,
## #   structured_type_3 <chr>, structured_type_4 <chr>,
## #   treatment_type_1 <chr>, treatment_type_2 <chr>,
## #   treatment_type_3 <chr>, lat_lter <dbl>, lng_lter <dbl>, taxas <list>
```

To zoom in the type of studies you are most interested in, you can define the criteria in the browse function. Say that you'd like only observational studies, with community data (data on more than 1 species), more than 20 years long, and with more than 5 spatial replicates:

```
browse(duration_years>20 & studytype == 'obs' & community == 'yes' & lterid !=
'SBC' &         tot_spat_rep > 5 )
```

```
## # A tibble: 10 x 19
```

```
##                       title proj_metadata_key lterid   datatype studytype
##  *                    <chr>             <int>  <chr>      <chr>     <chr>
##  1 Rabbit Population Dynamic                45    SEV      count       obs
##  2 SGS-LTER Standard Product                65    SGS    biomass       obs
##  3 Vegetation and Ground Cov                92    VCR      cover       obs
##  4 Primary succession on Mou               145    AND      cover       obs
##  5 Vegetation Plots of the B               194    BNZ      cover       obs
##  6 Vegetation Plots of the B               195    BNZ      count       obs
##  7 Permanent Plots at Pisgah               423    HFR individual       obs
##  8 Jornada Experimental Rang               677    JRN      count       obs
##  9 Spatial and Temporal Patt               679    JRN      count       obs
## 10 Transect Plant Line Inter               681    JRN      cover       obs
## # ... with 14 more variables: duration_years <int>, community <chr>,
## #   studystartyr <dbl>, studyendyr <dbl>, structured_type_1 <chr>,
## #   structured_type_2 <chr>, structured_type_3 <chr>,
## #   structured_type_4 <chr>, treatment_type_1 <chr>,
## #   treatment_type_2 <chr>, treatment_type_3 <chr>, lat_lter <dbl>,
## #   lng_lter <dbl>, taxas <list>
```

Above, you can see the study `title`, the ID of each study (`proj_metadata_key`), the three letter of the LTER site where the study was conducted (`lterid`), the type of abundance data contained in the dataset (`datatype`), whether the study is observational or experimental (`studytype`, potential values are 'obs' or 'exp'), and duration in years (`duration_years`). This file is much larger, but this shall suffice.

How to see if a study is what you are looking for

Before you download a data frame to format it, you 1) find it and 2) verify that it is actually what you are looking for. The easiest way to do this is to run browse by adding an argument `report = T`. This argument will open up a legible html document. This document produces open html page in your browser which have hyperlinks, and a list of the studies you selected through browse. Use this document to look up the description of each study, and, most importantly, look at each study's its ORIGINAL METADATA. To look at the original metadata, click on the hyperlink named `metadata_link`.

```
browse(duration_years>20 & studytype == 'obs' & community == 'yes' & lterid !=
'SBC' & tot_spat_rep > 5 , report = T)
```

Now, say that I checked the studies title, description, and, `metadata_link`. I ended up determining that the study called "SGS-LTER Standard Production Data : 1983-2008..." is what I am looking for. In popler, this study has ID number 65. To download this study, simply

```
# download SGS biomass data from popler
sgs_biom_raw <- get_data(proj_metadata_key == 65)
```

Format "Taxon counts"

First, format the actual abundance information from the original data set. Below is a snipped of code that performs that. I paste the code below, and then I explain its features.

```
sgs_biom_ab  <- sgs_biom_raw %>%
                  mutate( OBSERVATION_TYPE = 'TAXON_COUNT',
                          # Site id is a combination of the three nested spatial
levels
                          SITE_ID          = paste(spatial_replication_level_1,
                                                   spatial_replication_level_2,
                                                   spatial_replication_level_3, sep
= "_"),
                          DATE             = year,
                          # in this case X2_value == 'Species name"
                          VARIABLE_NAME    = paste(genus, species, sep=" "),
                          VARIABLE_UNITS   = NA,
                          VALUE            = abundance_observation) %>%
                  select( OBSERVATION_TYPE, SITE_ID, DATE, VARIABLE_NAME,
VARIABLE_UNITS, VALUE)
```

OBSERVATION_TYPE states that here we refer to "TAXON_COUNT" (the abundance of each species).

SITE_ID shows each separate spatial replicate. Most LTER studies are *spacially nested*. Not so in the *metacomm* working group format. Hence, you have to consider every combination of spatial replicates as a separate spatial replicate. To do so, 1. identify every column starting with `spatial_replication_level_` 2. paste these columns using the `paste` function and argument `sep="_"` The dataset I have downloaded here has three nested spatial levels.

DATE simply refers to the year. Every popler dataset provides a column with year, so this should not be an issue. UNLESS, censuses in the study were more frequent than once a year.

VARIABLE_NAME in taxon count data is the species name. In this case, we are lucky enough to have both `genus` and `species` columns. Simply join them together using the `paste` function. Sometimes, however, you will only have the species code data, contained in a column called `sppcode`. In this case, simply assign `sppcode` to VARIABLE_NAME.

VARIABLE_UNITS, and VALUES: you should always run the last three lines of code above!

Format LATITUDE/LONGITUDE information using the cov_unpack function

IMPORTANT NOTICE: few popler datasets have a latitude and longitude information for all spatial replicates. We sometimes provide lat/lon information associated with the `spatial_replication_level_1` (the site). However, some popler datasets have latitude and longitude information in the `covariates` column. This is a column that "packs" all information that does not fit into the structure of the popler database. In this example, the covariates contain latitude and longitude information:

```
# download SGS biomass data from popler
sgs_biom_raw$covariates  %>% head
```

```
## [1] "{'comments': 'NA', 'ScientificName': 'Artemisia frigida', 'Year': '1983',
'Latitude': '41.81355', 'Longitude': '-104.78492'}"
## [2] "{'comments': 'NA', 'ScientificName': 'Artemisia frigida', 'Year': '1983',
'Latitude': '41.81355', 'Longitude': '-104.78492'}"
## [3] "{'comments': 'NA', 'ScientificName': 'Artemisia frigida', 'Year': '1983',
'Latitude': '41.81355', 'Longitude': '-104.78492'}"
## [4] "{'comments': 'NA', 'ScientificName': 'Artemisia frigida', 'Year': '1983',
'Latitude': '41.81355', 'Longitude': '-104.78492'}"
## [5] "{'comments': 'NA', 'ScientificName': 'Artemisia frigida', 'Year': '1983',
'Latitude': '41.81355', 'Longitude': '-104.78492'}"
## [6] "{'comments': 'NA', 'ScientificName': 'Artemisia frigida', 'Year': '1984',
'Latitude': '41.81355', 'Longitude': '-104.78492'}"
```

You can unpack this column of gibberish into a proper dataframe using function `cov_unpack`:

```
# download SGS biomass data from popler
sgs_biom_raw  %>% cov_unpack(.) %>% head
```

```
##   X1_label X1_value       X2_label          X2_value X3_label X3_value
## 1 comments       NA ScientificName Artemisia frigida     Year     1983
## 2 comments       NA ScientificName Artemisia frigida     Year     1983
## 3 comments       NA ScientificName Artemisia frigida     Year     1983
## 4 comments       NA ScientificName Artemisia frigida     Year     1983
## 5 comments       NA ScientificName Artemisia frigida     Year     1983
## 6 comments       NA ScientificName Artemisia frigida     Year     1984
##   X4_label X4_value  X5_label   X5_value
## 1 Latitude 41.81355 Longitude -104.78492
## 2 Latitude 41.81355 Longitude -104.78492
## 3 Latitude 41.81355 Longitude -104.78492
## 4 Latitude 41.81355 Longitude -104.78492
## 5 Latitude 41.81355 Longitude -104.78492
## 6 Latitude 41.81355 Longitude -104.78492
```

In this dataframe, X1_label shows what each value refers to. Hence, `X1_label` tells us that `X1_value` contains "comments". `X2_label` tells us that `X2_value` contains a "scientific name" and so on.

Given the above, now we can format two separate data farames: one containing latitude information, the other containing longitude information. NOTE the `fac_char_num` function - I wrote this to convert factor variables into a numeric value.

```
# Spatial coordinates

# function to convert lat/lon from factor, to character, to numeric
fac_char_num <- function(x) x %>% as.character %>% as.numeric

# Format LATITUDE
```

```
sgs_biom_lat <- sgs_biom_raw %>%
                cbind( cov_unpack(.) ) %>%
                mutate( OBSERVATION_TYPE = 'SPATIAL_COORDINATE',
                        SITE_ID          = paste(spatial_replication_level_1,
                                                 spatial_replication_level_2,
                                                 spatial_replication_level_3, sep
= "_"),
                        DATE             = NA,
                        # in this case X2_value == 'Species name"
                        VARIABLE_NAME    = 'LATITUDE',
                        VARIABLE_UNITS   = 'decimal.degrees',
                        VALUE            = fac_char_num(X4_value) ) %>%
                select( OBSERVATION_TYPE, SITE_ID, DATE,
                        VARIABLE_NAME, VARIABLE_UNITS, VALUE) %>%
                unique

# Format LONGITUDE
sgs_biom_lon <- sgs_biom_raw %>%
                cbind( cov_unpack(.) ) %>%
                mutate( OBSERVATION_TYPE = 'SPATIAL_COORDINATE',
                        SITE_ID          = paste(spatial_replication_level_1,
                                                 spatial_replication_level_2,
                                                 spatial_replication_level_3, sep
= "_"),
                        DATE             = NA,
                        # in this case X2_value == 'Species name"
                        VARIABLE_NAME    = 'LONGITUDE',
                        VARIABLE_UNITS   = 'decimal.degrees',
                        VALUE            = fac_char_num(X5_value) ) %>%
                select( OBSERVATION_TYPE, SITE_ID, DATE,
                        VARIABLE_NAME, VARIABLE_UNITS, VALUE) %>%
                unique
```

Put it all together

Finally, you can stack these three files file together with a slick one line of code:

```
SGS_BIOMASS <- Reduce(function(...) rbind(...),
list(sgs_biom_lat,sgs_biom_lon,sgs_biom_lat) )
```