Matching survey data with catalogued specimens

**Description:** This tutorial describes the process for matching historical environmental and survey data with associated catalogued museum specimens. Here we match historical survey data from the Institute of Fisheries Research with records from the University of Michigan Museum of Zoology which can be accessed through the Global Biodiversity Information Facility (GBIF). In this example we focus on matching bluegill specimens to historical data from the surveys in which they were captured.

All data sets and code for this tutorial can be downloaded from github, <https://github.com/CHANGES-UM/data_curation_examples>.

# Part 1: Load libraries and data

## Survey data

This tutorial uses lake survey data from summary cards. You can find an example of the original historical data format by looking at this summary card for a survey of Bankers Lake in 1931. Front of card <https://changes-ifr.s3.us-east-2.amazonaws.com/ss1/SUMM/double/a_first/Hillsdale_Bankers_SUMM_005.jpg> Back of card <https://changes-ifr.s3.us-east-2.amazonaws.com/ss1/SUMM/double/a_first/Hillsdale_Bankers_SUMM_006.jpg>

We first read in the cleaned data file, “summary\_card\_data.csv”, for 2120 summary cards, which were transcribed through Zooniverse workflows and aggregated and cleaned.

The file contains unique identifiers for a lake (new\_key), the lake name, county, and dates when the sampling was done. This information can be used to match lake survey data with specimens that were collected and deposited from these surveys. The summary dataset also includes environmental variables for the lake, such as lake area and depth, temperature, and shoreline development.

#### load libraries ####   
library(dplyr) #library for data wrangling   
library(stringr) #library for extracting data   
library(tidyr) #library for data munging   
library(ggplot2) #library for graphs  
  
#### load summary card data ####   
summ\_dat<-read.csv("summary\_card\_data.csv")

## GBIF data

GBIF is an infrastructure which aggregates biodiversity data including those from natural history collections. Data from the catalogue of the University of Michigan Museum of Zoology can be accessed through GBIF here: <https://www.gbif.org/occurrence/search> by searching the collection code ‘UMMZ’. We performed and downloaded data from a search for bluegill (Lepomis macrochirus Rafinesque, 1819) records in the United States.

Download Information: DOI: <https://doi.org/10.15468/dl.cknu3x> Creation Date: 18:38:35 23 August 2022 Records included: 1900 records from 1 published datasets Compressed data size: 166.1 kB Download format: simple tab-separated values (TSV) Filter used: { “and” : [ “Country is United States of America”, “InstitutionCode is UMMZ”, “TaxonKey is Lepomis macrochirus Rafinesque, 1819” ] } The citation for this search is: GBIF.org (23 August 2022) GBIF Occurrence Download <https://doi.org/10.15468/dl.cknu3x>

Data from our search are in the file “blg\_UMMZ\_GBIF.csv”

You can familiarize yourself with the fields in this dataset using this reference: Darwin Core Maintenance Group. 2021. Darwin Core Quick Reference Guide. Biodiversity Information Standards (TDWG). <https://dwc.tdwg.org/terms/>

We begin by loading those data and select fields to match with the summary data including the collector (recordedBy field), dates, state, county, latitude/longitude, and locality, which includes the lake name.

In the UMMZ fish collection, like many fish collections, the vast majority of specimens are stored in ‘lots’, which are jars of specimens in ethanol that were all the individuals of one species, collected at the same time and place. Each lot has a catalogue number (‘catalogNumber’). The ‘preparations’ field contains the number of individuals in the lot, so we separate this number into its own column, called “num\_individuals”. Then we filter the museum records to select only observations in Michigan because this is where our historical survey data was collected.

There are also a number of records which have empty ‘year’ fields for which we know year is embedded within the ‘fieldNumber’ usually as initials of the collector or the county, followed by the last two digits of year, a dash, and a unique collection number. For example, from the survey of Bankers Lake in Hillsdale county in 1931 that we examined earlier, the field number in GBIF is H31-42, but the year is missing from GBIF. We separate the year from the field number and add this to the year field where this was missing. We then filter the years to 1915-1995 to match the historical survey data timeline.

The output table (museum\_dat) has 485 observations from GBIF from Michigan between the years 1915-1995.

#read in bluegill records from GBIF and select only Michigan records recorded by IFR   
  
museum\_dat<-read.csv("blg\_UMMZ\_GBIF.csv") %>%   
 select(gbifID, identifier, basisOfRecord, occurrenceID, catalogNumber, preparations, fieldNumber, eventDate, year, month, day, stateProvince, county, decimalLatitude, decimalLongitude, locality, recordedBy ) %>%   
 mutate(num\_individuals = as.numeric(str\_extract(preparations, "(?<=EtOH - )\\d+")))%>% #extract number of individuals from the preparations column, where ?<= is a positive look-behind from the pattern "EtOH - " and \\d+ pulls out the digit  
 filter(stateProvince == "Michigan") %>%   
 mutate(year2 = str\_extract(fieldNumber, "[^-]+"),   
 year2 = as.numeric(str\_extract(year2, "[[:digit:]]+")) ) %>% #pull the year from the field number   
 mutate(year3=ifelse(year2 >= 19 & year2 <= 96, paste0("19",year2), NA)) %>% #add 19 in front of the year   
 mutate(year = as.integer(ifelse(is.na(year), year3, year))) %>% #if the year is missing then add the new year from fieldNumber  
 filter(year >= 1915 & year <= 1995) # filter to match the years for the summary cards

# Part 2: Match lake summary cards to museum records

In order to match these two datasets, we need to create a new column that includes only the lake name from the locality column and do some other standardizing. We make both the locality column and the county column uppercase to reduce differences in the way a lake name/county name is written as well as remove apostrophes and parentheses. Then we extract the word right before or after lake or pond into a new column, which is the lake name. We can check this table and see that anything with “NA” in our new column is a river or creek site. We also find that this method does not work for some lake names, for example any lake name that is two words (e.g. LAKE ST CLAIR) only selects the first word or lakes like “DEVILS” is sometimes recorded “DEVIL”. So, we go through and modify these names accordingly.

museum\_dat<-museum\_dat%>%   
mutate(locality = toupper(locality), #make locality uppercase   
 county = toupper(county), #make county uppercase   
 locality = gsub("'",'',locality), #strip apostrophes  
 locality = gsub(")" ,'',locality), #strip parentheses  
 locality = gsub("L\\.",'LAKE',locality)) %>% #replace L. with Lake   
 mutate(lakename = str\_extract(locality, regex("\\b\\w+\\b\\s(?=LAKE)|\\b\\w+\\b\\s(?=POND)"))) %>% #extract word before lake or pond  
 mutate(lakename = ifelse(is.na(lakename),   
 str\_extract(locality, regex("(?<=\\bLAKE\\s)(\\w+)|(?<=\\bPOND\\s)(\\w+)" ) ),  
 lakename )   
 ) %>% #extract word after lake/pond   
mutate(lakename = trimws(lakename, which = c("both"))) %>% #remove white space around the lake name   
 mutate(lakename = ifelse(gbifID == 1889129462, "STCLAIR", lakename), #correct names  
 lakename = ifelse(gbifID == 1888996867 | gbifID == 1888996445 |gbifID == 1888996849 | gbifID == 1888996419, "FIVELAKE", lakename),  
 lakename = ifelse(lakename == "BEESE", "BAWBEESE", lakename),  
 lakename = ifelse(lakename == "DEVIL", "DEVILS", lakename),  
 lakename = ifelse(lakename == "FIRST", "STHELENS", lakename), #St Helens lake  
 lakename = ifelse(gbifID == 1889017451, "CHAINOFLAKES", lakename),  
 lakename = ifelse(gbifID ==1889003005, "DEVILSWASHBASIN", lakename),   
 lakename = ifelse(lakename == "PAW", "PAWPAW", lakename),   
 lakename = ifelse(lakename == "SELKIRK", "SELKIRKNORTH", lakename),   
 lakename = ifelse(gbifID ==1889079575, "FIRSTSISTER", lakename),  
 lakename = ifelse(gbifID ==1889007643, "THIRDSISTER", lakename),  
 lakename = ifelse(lakename == "STAR", "STARBIG", lakename),   
 lakename = ifelse(gbifID ==1888977161, "TWINLITTLE", lakename),   
 lakename = ifelse(gbifID ==1888977053, "TWINLITTLE", lakename)  
   
 )

We then match the museum and lake summary information by the county, lake name, and the year that the sample was taken. This ensures that the samples were taken from the correct lake in the same year. If the year is left blank in the GBIF data or survey data or if the lake names do not match up exactly then we will not get a match.

#### matching ####   
#\* match by lake name and county and year ####  
matches<-inner\_join(museum\_dat, summ\_dat, by=c("county", "lakename", "year"= "begin\_date\_year")) %>%   
 filter(subject\_id != 59070950) #remove duplicate card   
  
n\_distinct(matches$lakename, matches$county) # how many distinct lakes?

## [1] 45

Our resulting “matches” output has 98 museum records that match historical cards with data for 45 unique lakes.

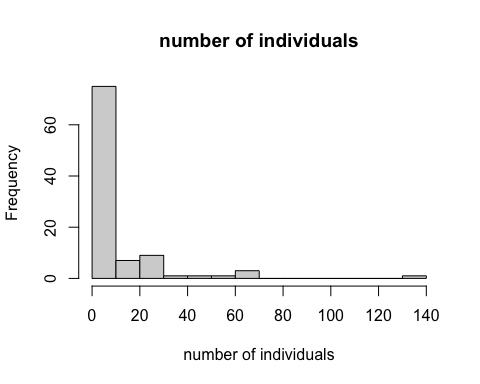
# Part 3: Summary of lakes and museum records

We can summarize these data and make some simple figures to investigate the information that we have for these lakes.  
First, we summarize the number of fish individuals from the museum records.

#summary  
matches %>%  
 summarise(  
 mean=mean(num\_individuals),  
 sd=sd(num\_individuals)  
 )

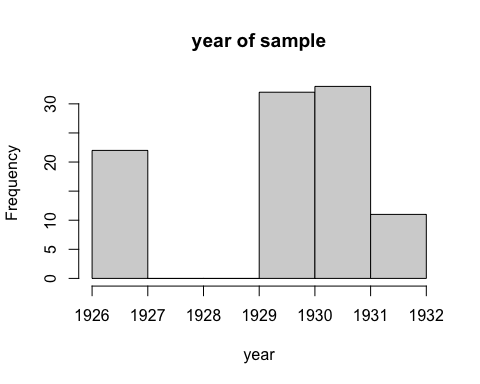
## mean sd  
## 1 10.10204 18.78884

#histogram   
hist(matches$num\_individuals, breaks = 10,   
 main = paste("number of individuals"),  
 xlab = 'number of individuals')



Second, we summarize the years of sampling.

#histogram  
hist(matches$year,   
 main = paste("year of sample"),  
 xlab = 'year')

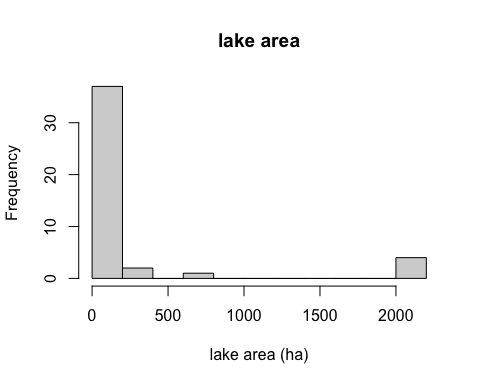


We can look at the attributes of the lakes for which there are catalogued specimens, including lake surface area (ha), maximum depth (m), and surface temperature (degC) during the survey from the summary card data.

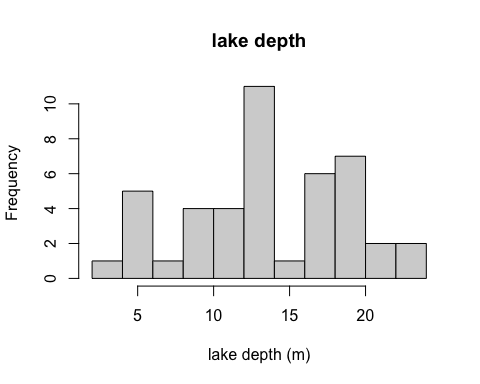
#summary table  
matches %>%  
 summarise(  
 area\_mean=mean(lake\_area\_ha, na.rm = TRUE),  
 area\_sd=sd(lake\_area\_ha, na.rm = TRUE),  
 depth\_mean=mean(max\_depth\_min\_m, na.rm = TRUE),  
 depth\_sd=sd(max\_depth\_min\_m, na.rm = TRUE),  
 temp\_mean=mean(temp\_surface\_min\_c, na.rm = TRUE),  
 temp\_sd=sd(temp\_surface\_min\_c, na.rm = TRUE),  
   
 )

## area\_mean area\_sd depth\_mean depth\_sd temp\_mean temp\_sd  
## 1 245.082 577.0608 13.6563 5.407293 25.09014 1.715002

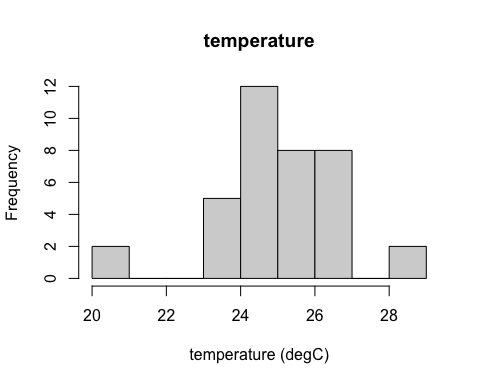
#histograms   
hist(matches$lake\_area\_ha, breaks = 10,   
 main = paste("lake area"),  
 xlab = 'lake area (ha)')



hist(matches$max\_depth\_min\_m, breaks = 10,   
 main = paste("lake depth"),  
 xlab = 'lake depth (m)')

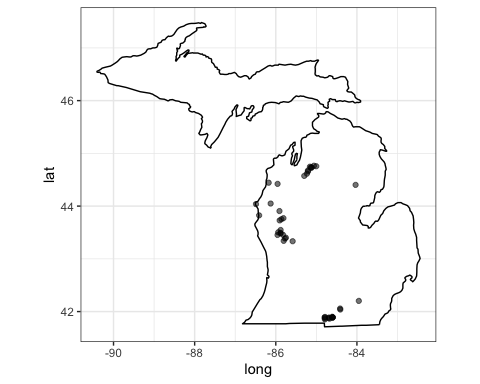


hist(matches$temp\_surface\_min\_c, breaks = 10,   
 main = paste("temperature"),  
 xlab = 'temperature (degC)')



Finally, we can map the lakes with survey data and associated catalogued specimens.

#select one point per lake   
lake\_matches<-matches%>%  
 distinct(lakename, county, .keep\_all = TRUE)  
#get the Michigan basemap  
MI\_basemap<-map\_data("state") %>%  
 subset(region %in% c("michigan")) # select michigan   
p<-ggplot(data = MI\_basemap) +   
 geom\_polygon(aes(x = long, y = lat, group = group), fill = "white", color = "black") + #this fill MI white and outline is black  
 coord\_fixed(1.3)   
  
# lake locations in MI  
p+ geom\_point(data=lake\_matches, aes(x = decimalLongitude, y = decimalLatitude, alpha = 0.1 ) ) +   
 theme\_bw() + # black and white theme for the background   
 theme(legend.position = "none") #remove legend



# we can color the points by an attribute, like the number of individuals in the museum collection   
p+ geom\_point(data=matches, aes(x = decimalLongitude, y = decimalLatitude, colour = c(num\_individuals))) +   
 scale\_color\_gradient(low = "blue", high = "yellow") +  
 labs(color="number individuals")+ #changes the labels on the legend  
 theme\_bw()

