02- Data Cleaning

Shelly Gaynor
University of Florida

iDigBio
Integrated Digitized Biocollections

02_Data_Cleaning.R

Data cleaning

0. Exploring the dataset
1. Taxonomic Harmonization
2. Clean localities
    1. Precision
    2. Remove impossible points
        - Most common coordinate: 0.00, 0.00
        - Cultivated zones, botanical gardens, etc
3. Flag and Filter Cultivated Records
4. Duplicate Removal
5. Spatial deduplication
6. Spatial thinning
7. Visualize
8. Save cleaned.csv

- REPEAT -

02_Data_Cleaning.R

Data cleaning

0. Exploring the dataset

B. Taxonomic Harmonization

C. Clean localities

   1. Precision

   2. Remove impossible points

      - Most common coordinate: 0.00, 0.00
      - Cultivated zones, botanical gardens, etc

D. Flag and Filter Cultivated Records

E. Duplicate Removal

F. Spatial deduplication

G. Spatial thinning

H – I. Visualize

J. Save cleaned.csv

- REPEAT -

Load Packages

```
library(gatoRs)
library(fields)
library(sf)
library(ggplot2)
library(ggspatial)
library(leaflet)
```

- **gatoRs** custom package for processing biodiversity data
- Geospatial packages include **sf** and **fields**
- **ggplot2, ggspatial,** and **leaflet** are for visualization

A. Load Data File

```
rawdf <- read.csv("data/01_download/raw/Shortia_galacifolia_raw_2025_06_27.csv")
nrow(rawdf)  # Starting number of records
```

```
## [1] 1534
```

- **read.csv** is a base function of R, it reads csv files
- **nrow** print the **n**umber of **row**s

B. Taxonomic Harmonization

```
unique(rawdf$scientificName)
```

```
## [1] "Shortia galacifolia Torr. & A.Gray"
## [2] "Sherwoodia galacifolia (Torr. & A.Gray) House"
## [3] "Shortia galacifolia"
## [4] "Shortia galacifolia Torr. & A. Gray"
## [5] "Shortia galacifolia Torrey & A. Gray"
## [6] "Shortia galacifolia var. galacifolia"
## [7] "Shortia galacifolia var. brevistyla"
## [8] "Shortia galacifolia var. brevistyla P. A. Davies"
## [9] "Sherwoodia galacifolia"
```

Create a list of accepted names based on the name column in your data frame

```
search <-  c("Shortia galacifolia", "Sherwoodia galacifolia")
```

- **unique** prints unique values in the column ***name*** from the dataframe ***rawdf***.
- **c** creates a list stored in the object ***search*** that contains ***"Shortia galacifolia"*** and ***"Sherwoodia galacifolia"***

B. Taxonomic Harmonization

Filter to only include accepted name:

```
df <-  taxa_clean(df = rawdf,
                  synonyms.list = search,
                  taxa.filter = "fuzzy",
                  accepted.name = "Shortia galacifolia")
```

- Here we are going to harmonize taxonomy using **taxa_clean().**
- This function has three filter options: exact, fuzzy, or interactive.

C. Locality Cleaning

Here we remove any records with missing coordinates, impossible coordinates, coordinates at (0,0), and any that are flagged as skewed. We also round the provided latitude and longitude values to a specified number of decimal places.

```
# Remove invalid/skewed records and round coordinates
df <- basic_locality_clean(df = df,
                           remove.zero = TRUE,
                           precision = TRUE,
                           digits = 2,
                           remove.skewed = TRUE)
nrow(df)
```

```
## [1] 77
```

- **basic_locality_clean()** is from the package **gatoRs**

C. Locality Cleaning

```
df <- basic_locality_clean(df = df,
                           remove.zero = TRUE, # Records at (0,0) are removed
```

- Only retains records in the column long/lat that are not (!=) equal to 0.00
- ”There’s no place like 0,0”

C. Locality Cleaning

```
df <- basic_locality_clean(df = df,
                           remove.zero = TRUE, # Records at (0,0) are removed
                           precision = TRUE, # latitude and longitude are rounded
                           digits = 2, # round to 2 decimal places
```

Precision	0 degrees Latitude	30 degrees Latitude	60 degrees Latitude	85 degrees Latitude
1.0 degrees	156904 m	146962 m	124605 m	112109 m
0.1 degrees	15691 m	14697 m	12461 m	11211 m
0.01 degrees	1570 m	1470 m	1247 m	1122m
0.001 degrees	157 m	147 m	125 m	113 m
0.0001 degrees	16 m	15 m	13 m	12 m
0.00001 degrees	2 m	2 m	2 m	2 m

- Our climate layers are 30 arc-sec resolution which is about 1000 meters

C. Locality Cleaning

```
df <- basic_locality_clean(df = df,
                           remove.zero = TRUE, # Records at (0,0) are removed
                           precision = TRUE, # latitude and longitude are rounded
                           digits = 2, # round to 2 decimal places
                           remove.skewed = TRUE)
```

- The skewed records can be identified with based on the ‘InformationWitheld’ column.

```
remove_skewed <- function(df, info.withheld = "informationWithheld"){
  df <- df[grepl("Coordinate uncertainty increased", df[[info.withheld]]) == FALSE, ]
  return(df)
}
```

D. Flag and Filter Cultivated Records

Removes points in a radius
around biodiversity institutions

Remove unlikely points

Remove coordinates in cultivated zones, botanical gardens, and outside our desired range. Here we set `interactive = FALSE`, however you can visualize the outliers by setting `interactive = TRUE`.

```
df <- process_flagged(df, interactive = FALSE)
```

Zizka et al. 2019. CoordinateCleaner: Standardized cleaning of occurrence records from biological collection databases. Methods in Ecology and Evolution.

E. Duplicate Removal

Here we identify and remove both (1) specimen duplicates and (2) aggregator duplicates based on each specimens coordinates, occurrenceID, and eventDate. To leverage all date information available, set `remove.unparseable = FALSE` to manually populate the year, month, and day columns. Here, we also confirm all ID (UUID and key) are unique to remove any within-aggregator duplicates that may accumulate due to processing errors.

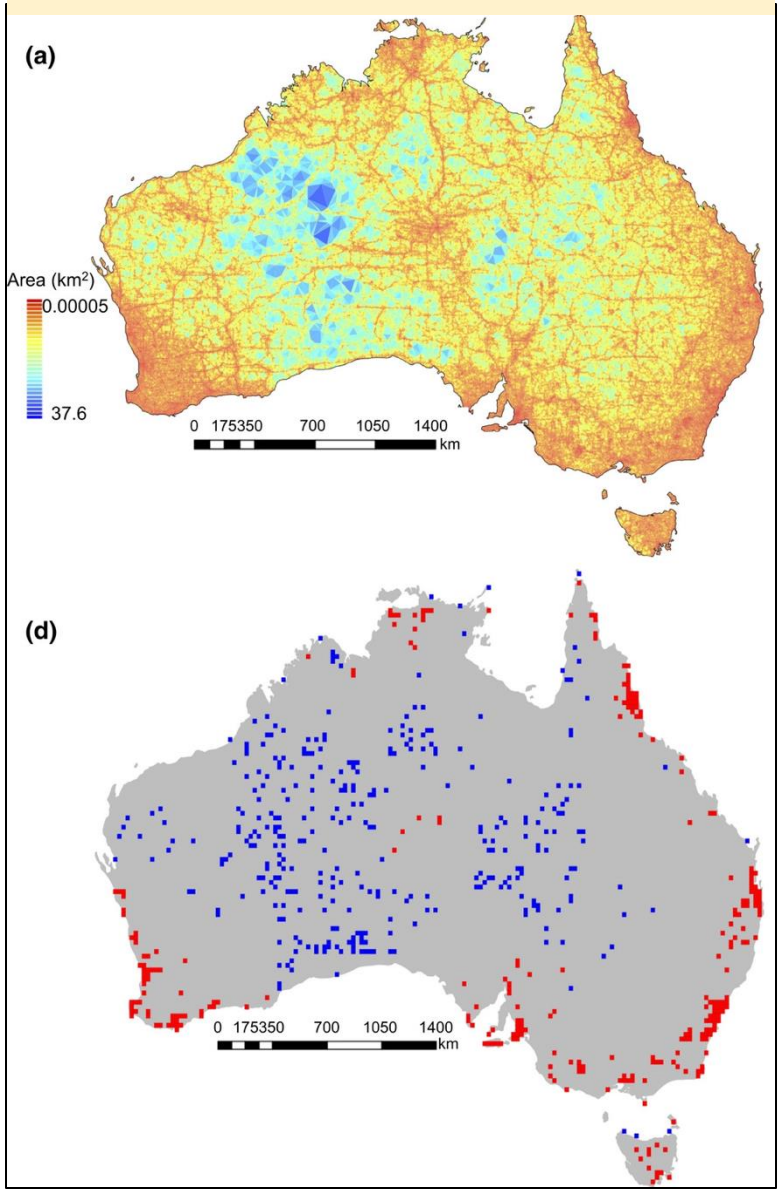
```
df <- remove_duplicates(df, remove.unparseable = TRUE)
```

- **parsedate** is a beautiful package that parses dates – However, it only parses dates after 1970!
  - Try **remove.unparable = FALSE**

F/G. Spatial Deduplication and Thinning

- Collection efforts can lead to clustering of points
  - Infrastructure (roads, herbaria, etc.)
  - Taxon bias
  - Temporal bias
- Filtering is a procedure to reduce the clustering of species records
  - Collection efforts can lead to clustering of points

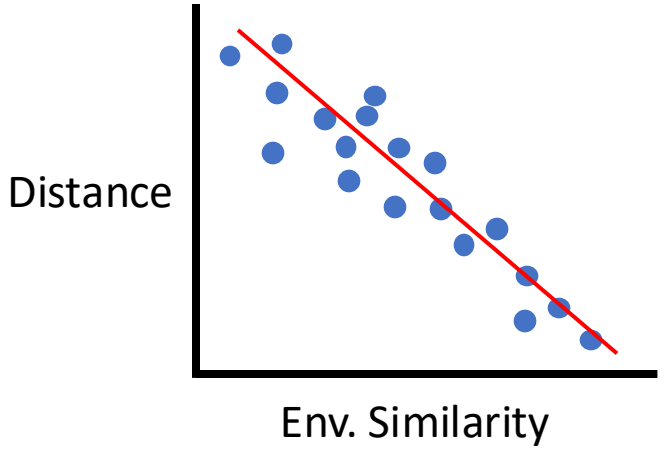
Sampling bias for 857,245 records



Daru et al. 2017. Widespread sampling biases in herbaria revealed from large-scale digitization. New Phytologist.

F/G. Spatial Deduplication and Thinning

- After filtering, there may still be spatial autocorrelation
  - This can be accounted for by data partitioning



Sillero N. and A. M. Barbosa. 2020. Common mistakes in ecological niche models. International Journal of Geographical Information Science.

F. Spatial Deduplication

Maxent will only retain one point per pixel. To make the ecological niche analysis comparable, we will retain only one point per pixel. Note: Default is a raster with 30 arc sec resolution.

One point per pixel

Maxent will only retain one point per pixel. To make the ecological niche analysis comparable, we will retain only one point per pixel. Note: Default is a raster with 30 arc sec resolution.

```
df <- one_point_per_pixel(df)
```

How many observations do we have now?

```
nrow(df)
```

```
## [1] 21
```

G. Spatial Thinning

Step 1: What should your minimum distance be?

First, lets assess the current minimum distance among your occurrence records. Here, we calculate minimum nearest neighbor distance in km.

```
nnDm <- rdist.earth(as.matrix(data.frame(lon = df$long, lat = df$lat)), m
iles = FALSE, R = NULL)
nnDmin <- do.call(rbind, lapply(1:5, function(i) sort(nnDm[,i])[2]))
min(nnDmin)
```

```
## [1] 2.226477
```

Here the current minimum distance is 2.22 km. Based on literature, we find a 2 meters (or 0.002 km) distance was enough to collect unique genets, so we do not need to thin our points.

- **rdist.earth** can be used to look at nearest neighbor distance in kilometers
- **do.call** applies to a list, **rbind** binds rows, and the **sort** is set up to select the lowest value.
- We want to look at the **min** distance between points remaining

G. Spatial Thinning

- **spThin** function ***thin***
- Set thin.par = minimum nearest neighbor distance
- Next, the function calculates pairwise distance between all records
- Then:
  - 1) For each record, IDs the # of occurrence records within distance thin.par
  - 2) One record of those which share the greatest # from (1) is removed at random.
  - 3) Repeat 1 – 2 till no record has a nearest neighbor closer than thin.par

Aiello-Lammens et al. 2015. spThin: an R package for spatial thinning of species occurrence records for use in ecological niche models. Ecography.

G. Spatial Thinning

Step 2: Thin occurrence records using spThin through gatoRs.

When you do need to thin your records, here is a great function to do so!

```
df <- thin_points(df,
                  distance = 0.002, # in km
                  reps = 100)
```

H. Static Map of Cleaned Points

Make points spatial

```
df_fixed <- st_as_sf(df,
                     coords = c("longitude", "latitude"),
                     crs = 4326)
```

Set up base maps

```
USA <- borders(database = "usa",
               colour = "gray80",
               fill = "gray80")
state <- borders(database = "state",
                 colour = "black",
                 fill = NA)
```

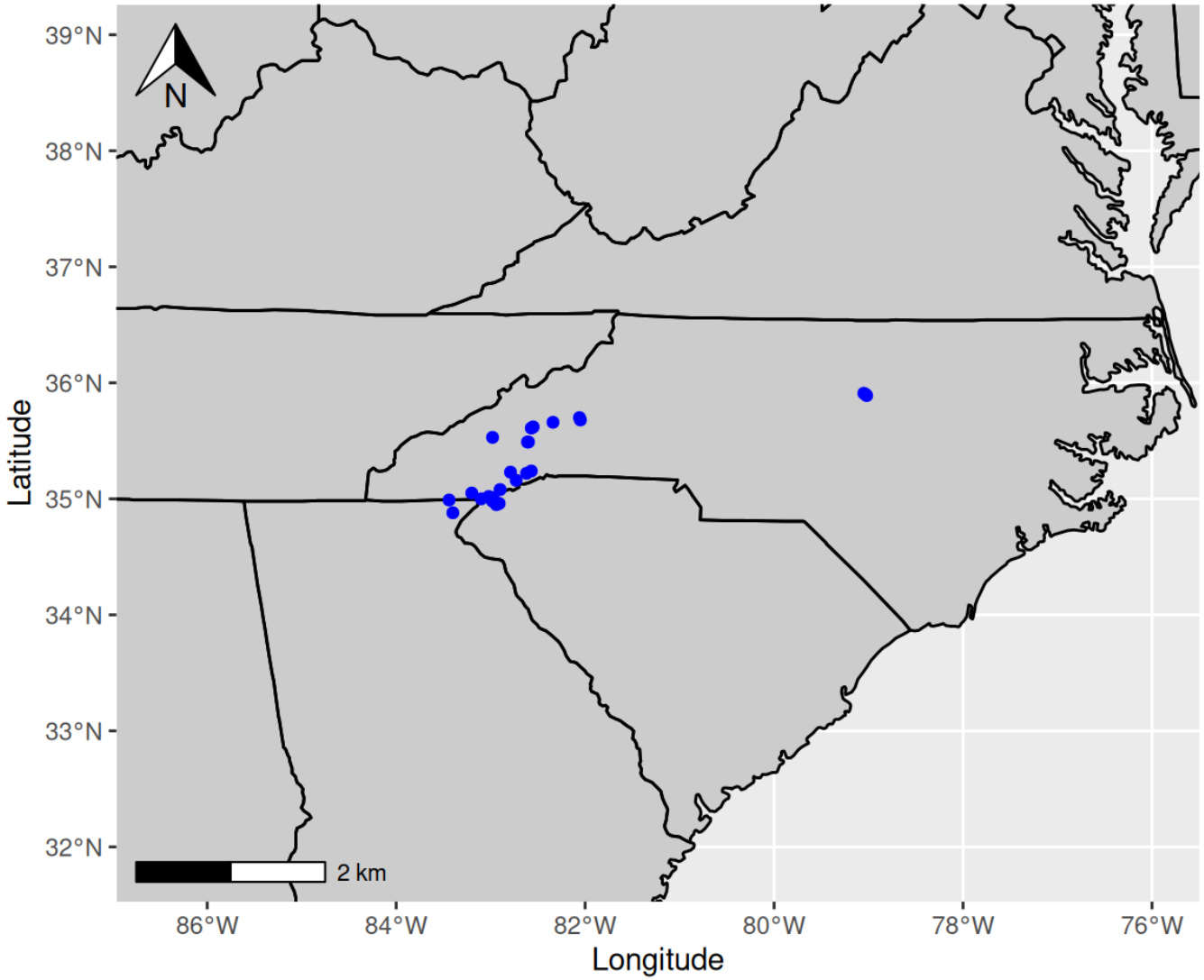
- crs = 4326 is WGS84
- Using the function **borders** from the package **ggplot2**, we download basemaps to plot our points on.

H. Static Map of Cleaned Points

```
simple_map <-ggplot() +
              USA +
              state +
              geom_sf(df_fixed,
                      mapping = aes(col = name),
                      col = "blue") +
              coord_sf(xlim = c(min(df$longitude) - 3,
                                max(df$longitude) + 3),
                      ylim = c(min(df$latitude) - 3,
                                max(df$latitude) + 3)) +
              xlab("Longitude") +
              ylab("Latitude") +
              annotation_scale(plot_unit = "km") +
              annotation_north_arrow(height = unit(1, "cm"),
                                     width = unit(1, "cm"),
                                     location = "tl")
simple_map
```

1. Add basemaps
2. Add points
3. Zoom in
4. Fix axis labels
5. Add scale and North arrow

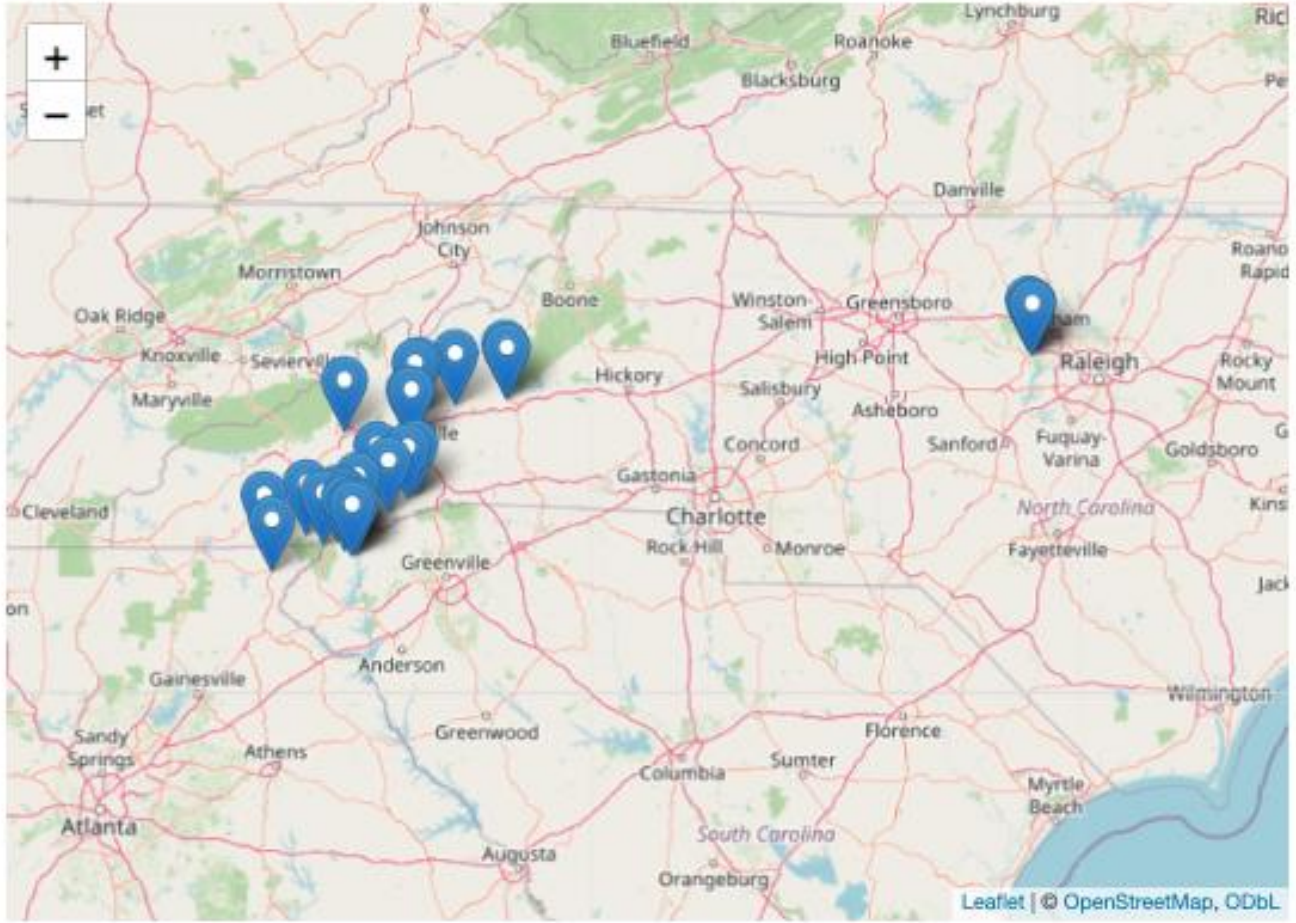
H. Static Map of Cleaned Points



I. Interactive Map with Leaflet

Another fun way to visualize occurrence data is using leaflet's interactive maps.

```
leaflet(df_fixed) %>%
  addMarkers(label = paste0(df$longitude, ", ", df$latitude)) %>%
  addTiles()
```



J. Saved Cleaned.csv

```
write.csv(df, "data/cleaning_demo/Shortia_galacifolia_20230605-cleaned.csv", row.names
= FALSE)
```

K. Repeat for all taxa

```
## Set up for loop
files <- list.files("data/download/raw", full.names = TRUE)[1:3]
synonymns <- list(Galax_urceolata = c("Galax urceolata",
                                      "Galax urceolata (Poir.) Brummitt",
                                      "Galax urceolata (Poiret) Brummitt",
                                      "Galax urceolaa",
                                      "Galax aphylla L.",
                                      "Galax aphylla"),
                  Pyxidanthera_barbulata =c("Pyxidanthera barbulata",
                                            "Pyxidanthera barbulata Michx.",
                                            "Pyxidanthera barbulata var. barbulata" ,
                                            "Pyxidenthera barbulata"),
                  Pyxidanthera_brevifolia = c("Pyxidanthera brevifolia",
                                              "Pyxidanthera brevifolia Wells",
                                              "Pyxidanthera barbulata var. brevifolia (We
lls) H.E.Ahles",
                                              "Pyxidanthera barbulata var. brevifolia"
))
```

1. Create list of files with paths

2. Create list of synonyms

K. Repeat for all taxa

```
## Repeat cleaning steps for the remaining taxa
for(i in 1:3){
    df <- read.csv(files[i])
    # Taxa clean
    search <-  synonymns[[i]]
    df <-  taxa_clean(df = df, synonyms.list = search,
                      taxa.filter = "exact",
                      accepted.name = search[1])
    # Locality clean
    df <- basic_locality_clean(df = df,  remove.zero = TRUE,
                               precision = TRUE, digits = 2,
                               remove.skewed = TRUE)
    df <- process_flagged(df, interactive = FALSE)
    # Remove duplicates
    df <- remove_duplicates(df, remove.unparseable = TRUE)
    # Spatial correct
    df <- one_point_per_pixel(df)
    df <- thin_points(df,  distance = 0.002, reps = 100)
    # Save file
    outfile <- paste0("data/cleaning_demo/",
                      gsub(" ", "_", search[1]),
                      "_20230605-cleaned.csv")
    write.csv(df, outfile,
              row.names = FALSE)
    rm(df, search, outfile)
}
```

0. Read in downloaded data
1. Resolve taxon names
2. Clean localities
    1. Precision
    2. Remove impossible points
        - Most common coordinate: 0.00, 0.00
        - Cultivated zones, botanical gardens, ect.
3. Remove duplicates
4. Spatial correction
5. Visualize
6. Save cleaned.csv

L. Make Maxent Ready

Read in all cleaned files

```
alldf <- list.files("data/cleaning_demo/", full.names = TRUE,
                    recursive = FALSE, include.dirs = FALSE, pattern = "*.csv")
alldf <- lapply(alldf, read.csv)
alldf <- do.call(rbind, alldf)
```

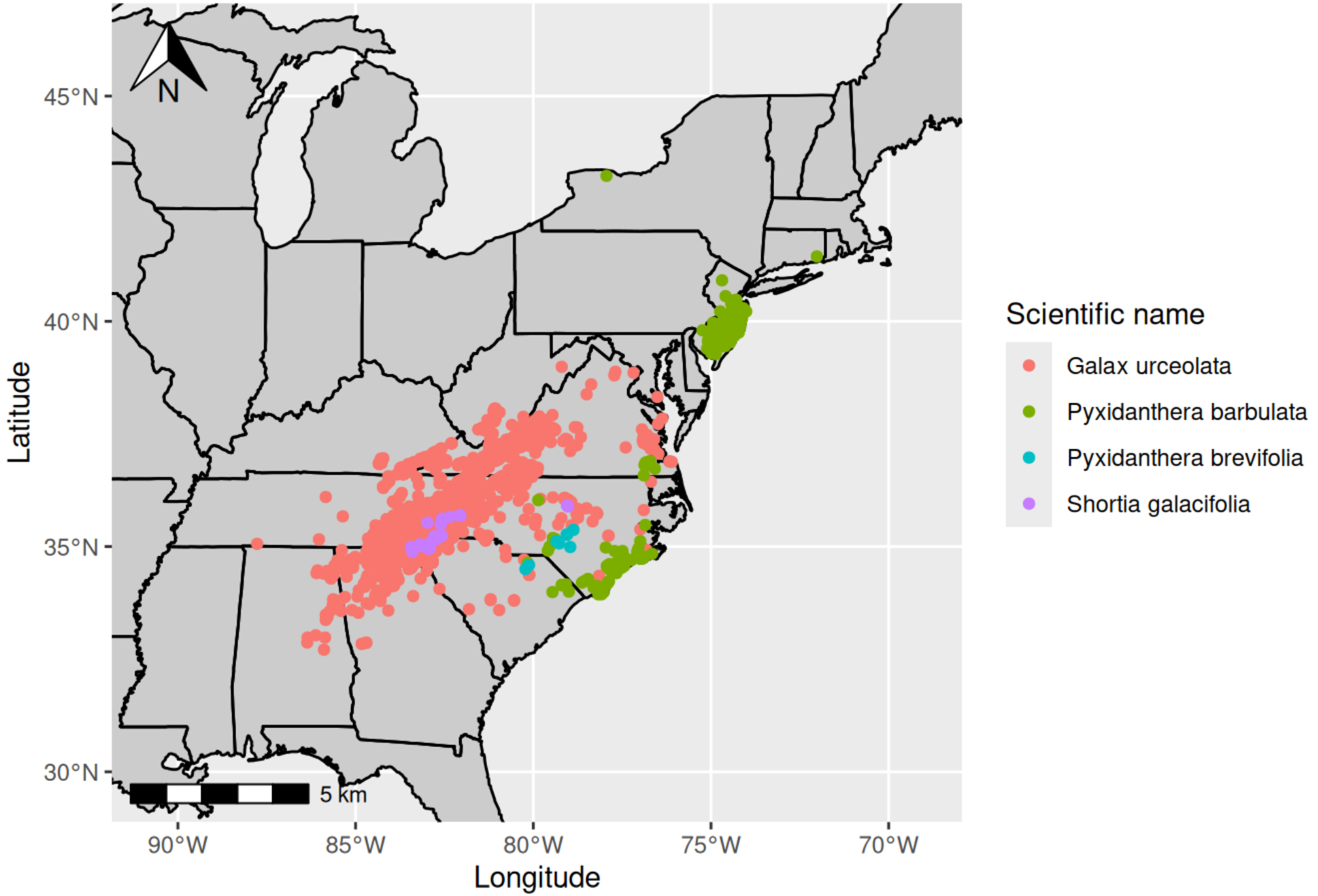
M. Map All Records

Visually inspect records to verify no additional records should be removed.

```
### Make points spatial
alldf_fixed <- st_as_sf(alldf, coords = c("longitude", "latitude"),
                        crs = 4326)

### Set basemap
USA <- borders(database = "usa", colour = "gray50", fill = "gray50")
state <- borders(database = "state", colour = "black", fill = NA)

### Plot
all_map <- ggplot() +
            USA +
            state +
            geom_sf(alldf_fixed,
                    mapping = aes(col = factor(accepted_name))) +
            coord_sf(xlim = c(min(alldf$longitude) - 3, max(alldf$longitude) + 3),
                     ylim = c(min(alldf$latitude) - 3, max(alldf$latitude) + 3)) +
            xlab("Longitude") +
            ylab("Latitude") +
            labs(color = "Scientific name") +
            annotation_scale(plot_unit = "km") +
            annotation_north_arrow(height = unit(1, "cm"),
                                   width = unit(1, "cm"),
                                   location = "tl")
all_map
```


N
45°N
40°N
35°N
30°N
Latitude
5 km
90°W
85°W
80°W
75°W
70°W
Longitude
Scientific name
Galax urceolata
Pyxidanthera barbulata
Pyxidanthera brevifolia
Shortia galacifolia

Minimum occurrence records

- Your points should represent the env. variability true to your species.
  - Too few points may not classify your species suitable conditions properly.
- It is not clear the minimum number of points needed.
  - Might be dependent on the extent of the study area
- 3, 4, or 5 have been suggested
  - 5 per env. variable has also been suggested
  - More env. variables may not significantly change the minimum sample size needed (see study below).

Proosdij et al. 2016. Minimum required number of specimen records to develop accurate species distribution models. Ecography.

N. Georeferencing

Prepare GeoLocate Batch File

Some records do not contain lat/lon info, but do have useful information in the locality field that can give us a reasonable estimate of location + some error. Here we'll use the gatoRs fxn need_to_georeference() to isolate these records for batch processing on the georeferencing platform GeoLocate.

Read in *raw* data and check what needs to be georeferenced

```
rawdf <- read.csv("data/01_download/raw/Shortia_galacifolia_raw_2025_06_27.csv")
rawdf_GeoRef <- need_to_georeference(rawdf)
```

Format the columns/fields for GeoLocate's standards, then write out for batch processing.

```
# Format columns for GeoLocate submission
rawdf_GeoRef <- rawdf_GeoRef %>%
  dplyr::select("locality string" = locality,
                country,
                state = stateProvince,
                county,
                latitude,
                longitude,
                ID,
                name = scientificName,
                basis = basisOfRecord)

# Add required empty columns
rawdf_GeoRef$'correction status' <- ""
rawdf_GeoRef$precision <- ""
rawdf_GeoRef$'error polygon' <- ""
rawdf_GeoRef$'multiple results' <- ""

# Reorder columns
rawdf_GeoRef2 <- rawdf_GeoRef[, c("locality string", "country",
                                  "state", "county", "latitude",
                                  "longitude","correction status",
                                  "precision", "error polygon",
                                  "multiple results","ID",
                                  "name", "basis")]

# write out csv
write.csv(rawdf_GeoRef2,
          "data/03_georeferencing/Shortia_galacifolia_Needing_GeoRef_2025_06_27.csv",
          row.names = FALSE)
```