Check Occurrences

When you download occurrence data from GBIF, it often contains various errors and issues. Common problems include occurrences located at (0, 0), coordinates with inverted signs (e.g., -73.00 recorded as 73.00 or 120.00 recorded as -120.00), and points that fall in the ocean. Some coordinates may also be misformatted, such as -73.2000 incorrectly transformed into -732000. In this example, you'll work with occurrence data for the order Chiroptera, all recorded in the United States.

First, let's load the required packages and import the occurrence points. The data downloaded directly from GBIF has its columns separated by tabs (), which is why we use the *read.delim()* function instead of *read.csv()*. You can use the *dim()* function to check the size of your dataset, and *str()* to inspect its internal structure and content.

```
library(sf)
library(raster)
library(dplyr)
library(ggplot2)

occ_bats <- read.delim("../GBIF/Chiroptera_USA.csv", header = T)
dim(occ_bats)
```

```
## [1] 82396    50
```

```
str(occ_bats, max.level=1)
```

```
## 'data.frame':    82396 obs. of  50 variables:
##  $ gbifID                         : num  5.15e+09 5.15e+09 5.15e+09 5.15e+09 5.15e+09 ...
##  $ datasetKey                     : chr  "0daed095-478a-4af6-abf5-18acb790fbb2" "0daed095-478a-4af6-
##  $ occurrenceID                   : chr  "https://arctos.database.museum/guid/MVZ:Mamm:37343" "https
##  $ kingdom                        : chr  "Animalia" "Animalia" "Animalia" "Animalia" ...
##  $ phylum                         : chr  "Chordata" "Chordata" "Chordata" "Chordata" ...
##  $ class                          : chr  "Mammalia" "Mammalia" "Mammalia" "Mammalia" ...
##  $ order                          : chr  "Chiroptera" "Chiroptera" "Chiroptera" "Chiroptera" ...
##  $ family                         : chr  "Vespertilionidae" "Vespertilionidae" "Phyllostomidae" "Ves
##  $ genus                          : chr  "Eptesicus" "Myotis" "Macrotus" "Myotis" ...
##  $ species                        : chr  "Eptesicus fuscus" "Myotis yumanensis" "Macrotus californi
##  $ infraspecificEpithet           : chr  "fuscus" "yumanensis" "" "melanorhinus" ...
##  $ taxonRank                      : chr  "SUBSPECIES" "SUBSPECIES" "SPECIES" "SUBSPECIES" ...
##  $ scientificName                 : chr  "Eptesicus fuscus fuscus" "Myotis yumanensis yumanensis" "
##  $ verbatimScientificName         : chr  "Eptesicus fuscus fuscus" "Myotis yumanensis yumanensis" "
##  $ verbatimScientificNameAuthorship: chr  "" "" "" "" ...
##  $ countryCode                    : chr  "US" "US" "US" "US" ...
##  $ locality                       : chr  "John Brown's Cave near Harper's Ferry" "Pyramid Lake" "To
##  $ stateProvince                  : chr  "West Virginia" "Nevada" "California" "California" ...
##  $ occurrenceStatus               : chr  "PRESENT" "PRESENT" "PRESENT" "PRESENT" ...
##  $ individualCount                : int  NA NA NA NA NA NA NA NA NA NA ...
##  $ publishingOrgKey               : chr  "8edbbde0-055e-11d8-b850-b8a03c50a862" "8edbbde0-055e-11d8
```

```
##  $ decimalLatitude                : num  39.3 40.1 33.6 33.5 32.9 ...
##  $ decimalLongitude               : num  -77.7 -119.6 -116.2 -116.5 -116.6 ...
##  $ coordinateUncertaintyInMeters  : num  3219 25777 1609 1000 4828 ...
##  $ coordinatePrecision            : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ elevation                      : num  NA NA NA 1829 NA ...
##  $ elevationAccuracy              : num  NA NA NA 0 NA 0 0 0 0 NA ...
##  $ depth                          : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ depthAccuracy                  : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ eventDate                      : chr  "1925-03-15" "1926-07-09" "1908-04-15" "1908-06-25" ...
##  $ day                            : int  15 9 15 25 16 7 9 3 3 1 ...
##  $ month                          : int  3 7 4 6 8 6 6 7 7 10 ...
##  $ year                           : int  1925 1926 1908 1908 1908 1928 1928 1928 1928 1908 ...
##  $ taxonKey                       : int  7261810 7261885 2433296 12176064 4265985 8397398 8397398 6
##  $ speciesKey                     : int  2432352 2432461 2433296 2432456 2432352 8397398 8397398 243
##  $ basisOfRecord                  : chr  "PRESERVED_SPECIMEN" "PRESERVED_SPECIMEN" "PRESERVED_SPECI
##  $ institutionCode                : chr  "MVZ" "MVZ" "MVZ" "MVZ" ...
##  $ collectionCode                 : chr  "Mamm" "Mamm" "Mamm" "Mamm" ...
##  $ catalogNumber                  : chr  "MVZ:Mamm:37343" "MVZ:Mamm:37376" "MVZ:Mamm:1216" "MVZ:Mamm
##  $ recordNumber                   : chr  "" "" "Walter P. Taylor 72" "Joseph Grinnell 2283" ...
##  $ identifiedBy                   : chr  "Museum of Vertebrate Zoology" "Museum of Vertebrate Zoolog
##  $ dateIdentified                 : chr  "1999-01-27T00:00:00" "1999-01-27T00:00:00" "1999-01-27T00
##  $ license                        : chr  "CC0_1_0" "CC0_1_0" "CC0_1_0" "CC0_1_0" ...
##  $ rightsHolder                   : chr  "" "" "" "" ...
##  $ recordedBy                     : chr  "Collector(s): Vernon Orlando Bailey" "Collector(s): E. Ray
##  $ typeStatus                     : chr  "" "" "" "" ...
##  $ establishmentMeans             : chr  "" "" "" "" ...
##  $ lastInterpreted                : chr  "2025-05-28T00:48:04.177Z" "2025-05-28T00:48:04.178Z" "2025
##  $ mediaType                      : chr  "" "" "" "" ...
##  $ issue                          : chr  "CONTINENT_DERIVED_FROM_COORDINATES;INSTITUTION_MATCH_FUZZY
```

Now, lets load the polygon. In this example lets read the SHP file for all the US territories.

```
US_border <- read_sf("../shp/us-state-boundaries/us-state-boundaries.shp")
```

Currently, the points are stored as a *data.frame*, but we need to convert them into a spatial object to perform further analyses. It is quite common to find occurrence records without longitude and latitude values, even when the "include coordinates" option is selected during the download.

First, we use the *filter()* function to remove records that lack coordinate information. Then, we convert the data.frame into a spatial object using the *st_as_sf()* function, specifying the column names that contain the longitude and latitude values.

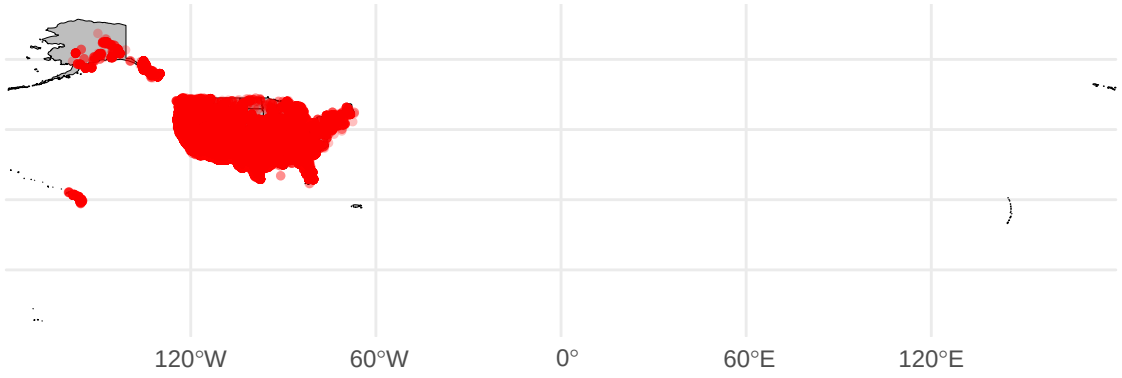
Remember: when working with coordinates, X corresponds to longitude and Y to latitude.

```
occ_bats <- occ_bats %>% filter(!is.na(decimalLongitude) & !is.na(decimalLatitude))
occ_bats <- occ_bats %>% st_as_sf(coords = c("decimalLongitude","decimalLatitude"), crs = 4326)
occ_bats <- st_transform(occ_bats, st_crs(US_border))

occ_bats
```

Now, let's plot the map along with the occurrence points. As you can see, the map looks a bit strange. This is because the U.S. map includes all its overseas territories (e.g., the Virgin Islands, Marshall Islands).

However, you can also observe that all the points remain in the Western Hemisphere, which indicates that we don't have any occurrences with inverted coordinates (e.g., longitude mistakenly recorded as positive instead of negative).

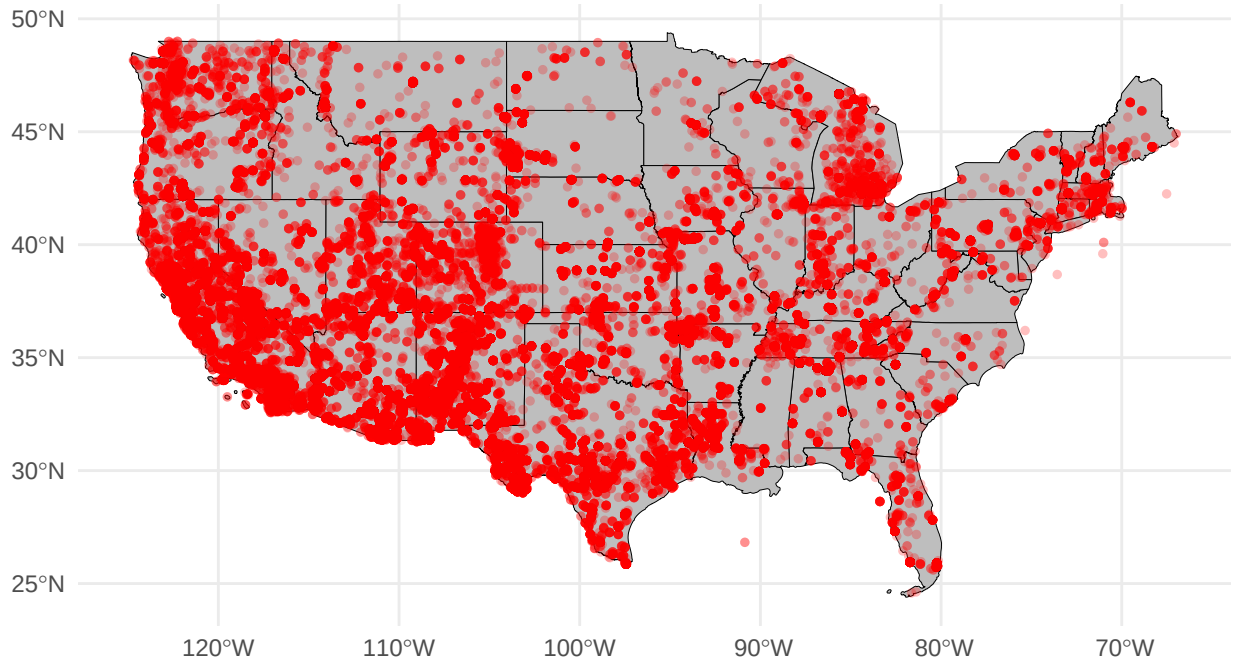
```
ggplot()+
  geom_sf(data = US_border, fill="gray", color="black", lwd = 0.1)+
  geom_sf(data = occ_bats, color="red", size =1, alpha =0.25)+
  theme_minimal()
```



Now, you can see that some points on the map are from Hawaii, Alaska, and Puerto Rico. In this case, that information is unnecessary, as we are only interested in occurrences from the mainland United States. So, let's crop the data to focus on the area of interest.

There are several ways to do this, including using external software like QGIS. Here, we'll create a bounding box (bbox) that covers only the mainland U.S., between Canada and Mexico. Using this bbox, we'll crop both the occurrence points and the map accordingly.

```
US_Crop <- st_bbox(c(xmin = -125, ymin = 22,
                     xmax = -66, ymax = 49),
                     crs = 4326)

US_border <- st_crop(US_border, US_Crop)
occ_bats <- st_crop(occ_bats, US_Crop)


ggplot()+
  geom_sf(data = US_border, fill="gray", color="black", lwd = 0.1)+
  geom_sf(data = occ_bats, color="red", size =1, alpha =0.25)+
  theme_minimal()
```



Now you have a cleaned set of points—with no inverted coordinates and no (0, 0) values. Next, let's check how many of these points fall in the ocean. This is a common issue for occurrences recorded near the coastline, where GPS devices or georeferencing methods may have high positional error.

First, you'll calculate the intersection between the points and the land polygon using the *st_intersects()* function. Be sure to set the parameter sparse = FALSE so that the result is returned as a matrix. Then, extract only the first column ([, 1]), which will be a logical vector: TRUE for points inside the polygon, and FALSE for points outside.

Note that the polygon is wrapped inside the *st_union()* function. This is because the original map consists of a MULTIPOLYGON object representing individual states. Using *st_union()* merges them into a single unified polygon, which improves the performance and accuracy of the intersection operation.

Finally, you'll split the points into two separate datasets: one for occurrences inside the polygon and another for those outside.

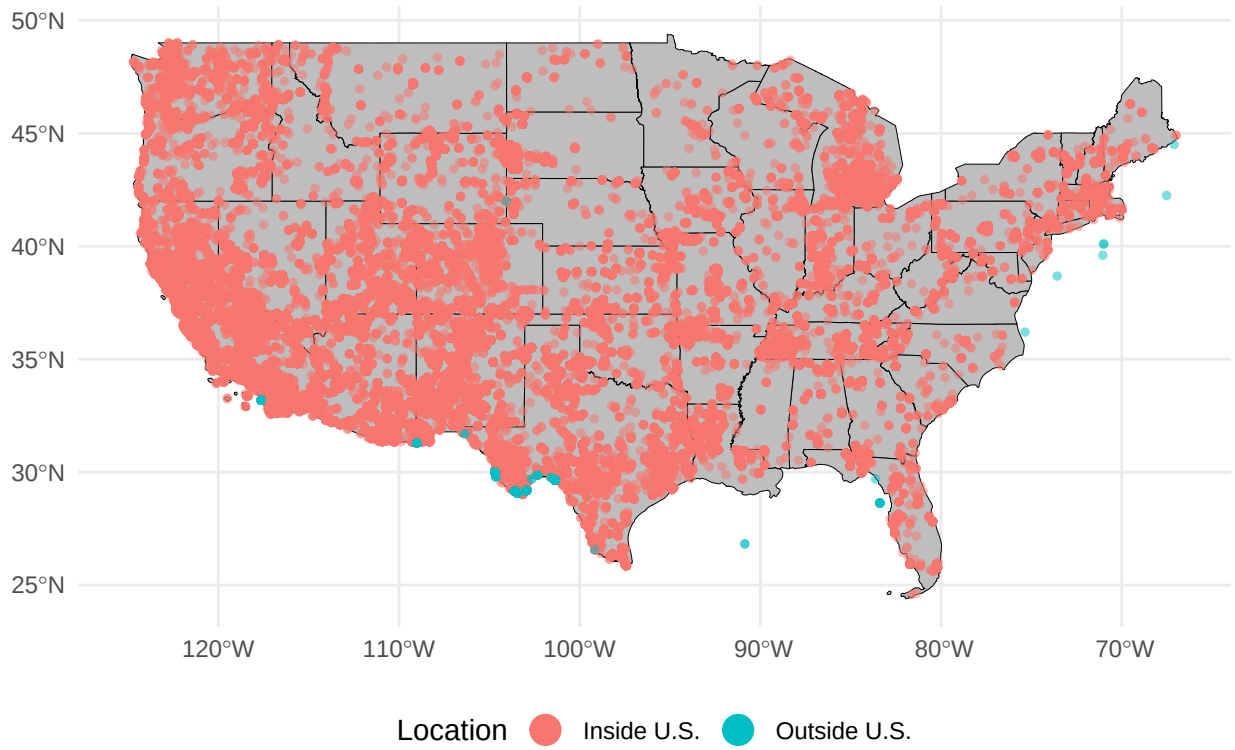
```
occ_intersect <- st_intersects(occ_bats, st_union(US_border), sparse = F)[,1]

occ_inside <- occ_bats[occ_intersect,]
occ_outside <- occ_bats[!occ_intersect,]

dim(occ_inside)
```

```
## [1] 80318    49
```

```
dim(occ_outside)
```

```
## [1] 508  49
```

```
ggplot()+
  geom_sf(data = US_border, fill="gray", color="black", lwd = 0.1)+
  geom_sf(data = occ_inside, aes(color="Inside U.S."), size =1, alpha =0.5)+
  geom_sf(data = occ_outside, aes(color="Outside U.S."), size =1, alpha =0.5)+
  scale_color_discrete(name="Location",
                       guide = guide_legend( override.aes = list(size = 5, alpha = 1)))+
  theme_minimal()+
  theme(legend.position = "bottom")
```



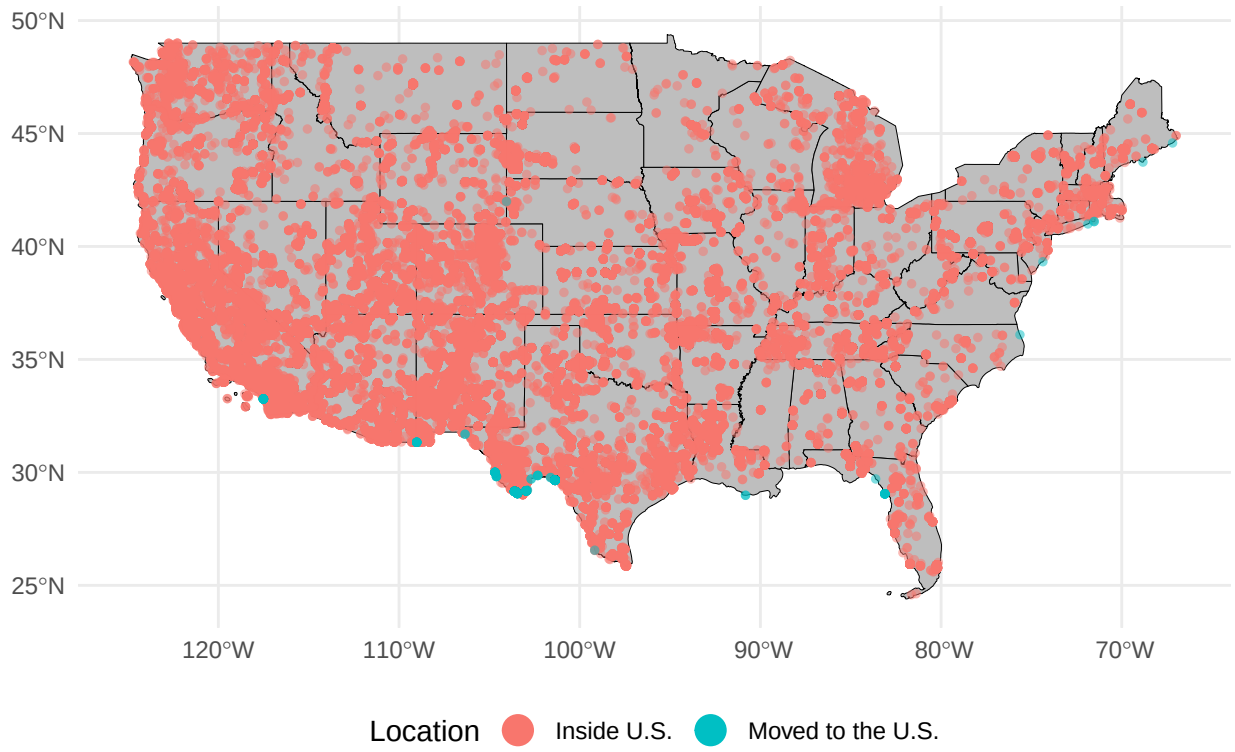
Now, you have two different subsets, so you can explore and try to fix the outlier records manually using QGIS or ArcGIS, and then merge all of your data together. Be sure to save your datasets as CSV files.

```
write.table(as.data.frame(occ_inside), file = "../GBIF/Occ_Inside.csv", sep = "\t", quote = F)
write.table(as.data.frame(occ_outside), file = "../GBIF/Occ_Outside.csv", sep = "\t", quote = F)
```

There are some quick options to fix those outlier points. These are usually records located near the sea or national borders, where device accuracy may have caused the coordinates to shift. One fast strategy is to

find the nearest point on the polygon (e.g., a country boundary) and move the outlier there. To do this, identify the nearest line on the polygon border, convert that line to points, and keep the closest point. These new points will serve as the corrected occurrences for the original outliers.

```
# 1. Find the nearest line
nearest_lines <- st_nearest_points(occ_outside, st_union(US_border))

# 2. Convert lines to points (each line becomes 2 points)
all_points <- st_cast(nearest_lines, "POINT")

# 3. Create pairing index (2 points per original record)
point_groups <- rep(1:nrow(occ_outside), each = 2)

# 4. Keep only the second point from each pair (the border point)
border_points <- all_points[seq(2, length(all_points), by = 2)] %>%
  st_as_sf()

ggplot()+
  geom_sf(data = US_border, fill="gray", color="black", lwd = 0.1)+
  geom_sf(data = occ_inside, aes(color="Inside U.S."), size =1, alpha =0.5)+
  geom_sf(data = border_points, aes(color="Moved to the U.S."), size =1, alpha =0.5)+
  scale_color_discrete(name="Location",
                       guide = guide_legend( override.aes = list(size = 5, alpha = 1)))+
  theme_minimal()+
  theme(legend.position = "bottom")
```



Now, merge all the corrected data together and save the final dataset.

```
occ_inside <- cbind(st_drop_geometry(occ_inside),st_coordinates(occ_inside))
occ_inside

border_points <- cbind(st_drop_geometry(occ_outside), st_coordinates(border_points))
border_points

occ_final <- rbind(occ_inside, border_points)

write.table(as.data.frame(occ_final), file = "../GBIF/Occ_Final.csv", sep = "\t", quote = F)
```