#1: Isolating dispersal jumps and diffusive spread

Nadege Belouard*

2024-08-13

Contents

Aim and setup

The spread of invasive species is due both to diffusive spread and human-assisted jump dispersal, often caused by species hitchhiking on vehicles. The aim of this vignette is to differentiate diffusive spread from jump dispersal in invasive species based on occurrence data. This identification required the development of a directional analysis of species presence. We demonstrate this method using occurrence data of the spotted lanternfly, *Lycorma delicatula*, an invasive Hemiptera in the United States.

We load the necessary packages.

*iEco lab at Temple University, Ecobio lab at the University of Rennes, nadege.belouard@gmail.com

```
library(magrittr)
library(ggplot2)
library(dplyr)
```

```
##
## Attachement du package : 'dplyr'

## Les objets suivants sont masqués depuis 'package:stats':
##
##     filter, lag

## Les objets suivants sont masqués depuis 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
library(jumpID)
```

1. Data overlook

The SLF dataset is available from the `lydemapr` companion package. In the `download_data` folder, then in the `v2_2023` subfolder, download the `lyde_data_v2.zip` file and place it in the `data` folder of your local `jumpID` project. At the date at which this vignette is written, `lydemapr` contains data up to year 2022.

We load this dataset that contains all the occurrences of the spotted lanternfly, and take a look at it.

```
slf <- read.csv(file.path(here::here(), "data", "lyde_data_v2", "lyde.csv"))
slf %<>% select(bio_year, latitude, longitude, lyde_established)
tibble::tibble(slf)
```

```
## # A tibble: 831,039 x 4
##    bio_year latitude longitude lyde_established
##       <int>    <dbl>     <dbl> <lgl>
##  1     2015     40.4     -75.7 FALSE
##  2     2016     40.3     -75.6 FALSE
##  3     2016     40.4     -75.5 FALSE
##  4     2016     40.4     -75.6 FALSE
##  5     2016     40.4     -75.7 FALSE
##  6     2016     40.5     -75.6 FALSE
##  7     2016     40.6     -75.5 FALSE
##  8     2016     40.4     -75.7 FALSE
##  9     2016     40.4     -75.6 FALSE
## 10     2017     40.1     -75.5 FALSE
## # i 831,029 more rows
```

The dataset contains 831,039 rows, each corresponding to a SLF survey at a specific date and location. The columns that we will using are:
* bio_year: the biological year of the occurrence (see `lydemapr` for a description of the difference between year and bio_year)
* latitude and longitude (XY coordinates). The precise location of SLF surveys is here summarized in 1-km^2 cells. All coordinates are in WGS84 (EPSG 4326)
* lyde_established: whether an established SLF population was found in this survey (at least two live individuals or an egg mass, see `lydemapr`)

2. Data formatting

First, to use the `jumpID` package, we need to rename the columns with generic names expected by the package.

```
slf %<>% rename(year = bio_year,
                established = lyde_established)
```

Second, multiple surveys were sometimes conducted at the same location during the same year, resulting in a redundant dataset for the purpose of our analysis. We summarize the information available at each location every year, so that a row now represents the detection status at a location for a given year.

Note: when several surveys indicate that SLF are "present" the same year at the same location, it could be tempting to categorize SLF as "established". However, the category "present" often refers to dead individuals, although this information is not explicitly available. We use a conservative approach and consider SLF as established in a cell only if one of the surveys considered it as established.

```
grid_data <- slf %>%
  dplyr::group_by(year, latitude, longitude) %>%
  dplyr::summarise(established = any(established)) %>%
  ungroup()
```

```
## `summarise()` has grouped output by 'year', 'latitude'. You can override using
## the `.groups` argument.
```

```
tibble::tibble(grid_data)
```

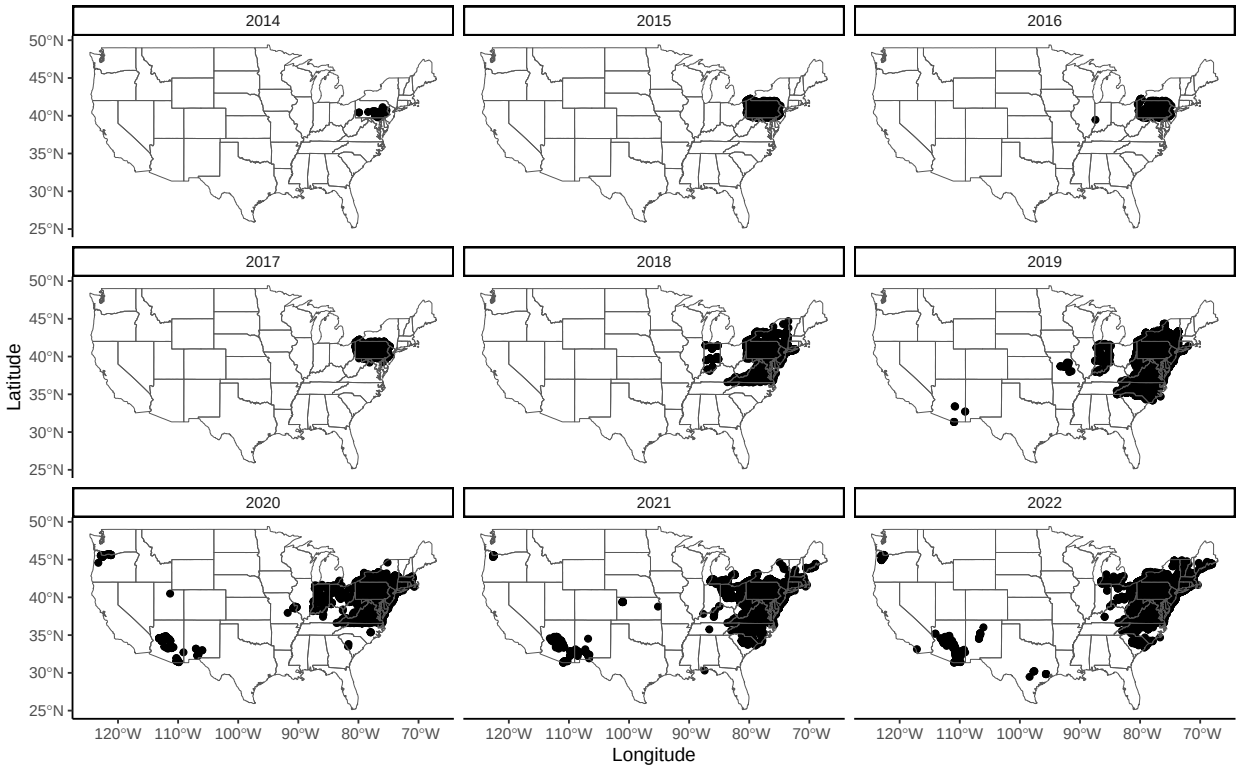
```
## # A tibble: 130,015 x 4
##     year latitude longitude established
##    <int>    <dbl>     <dbl> <lgl>
##  1  2014     39.8     -76.8 FALSE
##  2  2014     39.8     -76.3 FALSE
##  3  2014     39.8     -76.4 FALSE
##  4  2014     39.9     -76.6 FALSE
##  5  2014     39.9     -76.9 FALSE
##  6  2014     39.9     -76.5 FALSE
##  7  2014     39.9     -76.8 FALSE
##  8  2014     39.9     -76.8 FALSE
##  9  2014     39.9     -76.2 FALSE
## 10  2014     39.9     -76.8 FALSE
## # i 130,005 more rows
```

The dataset now has 130,015 rows, corresponding to 1280 x 1596 cells. Let's look at these data on a map.

```
# get a simple feature object for states
states <- sf::st_as_sf(maps::map("state", plot = FALSE, fill = TRUE)) %>%
  sf::st_transform(crs = 4326)

# Map all surveys
ggplot(data = states) +
  geom_point(data = grid_data, aes(x = longitude, y = latitude)) +
  xlab("Longitude") + ylab("Latitude") +
```

```
  geom_sf(data = states, alpha = 0) +
  facet_wrap(~year) +
  theme_classic()
```

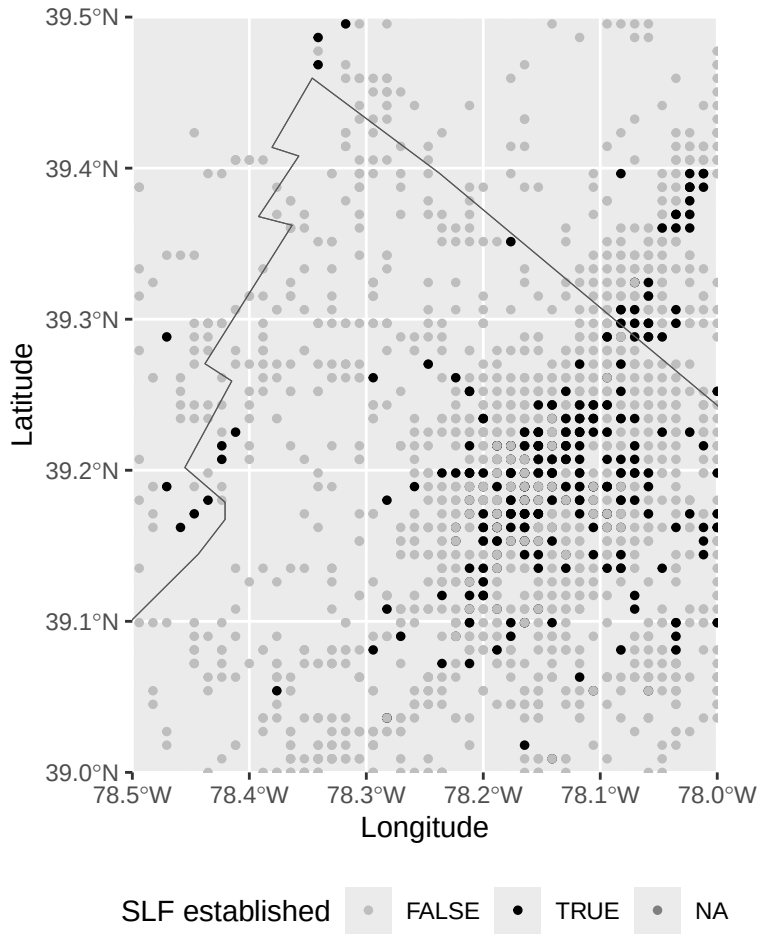


Cells were predominantly surveyed in the Eastern US, but there are also cells surveyed in other parts of the US, starting in 2019. Positive cells (established SLF) are only in the Eastern US, so maps will be zoomed in on the Eastern US from here.

Every year, there is a large number of cells surveyed with positive (SLF established) or negative results (SLF not established). A zoomed-in map allows to see the cell grid.

```
ggplot(data = states) +
  geom_point(data = grid_data, aes(x = longitude, y = latitude,
                                   col = established), size = 1) +
```

```
  scale_color_manual(values = c("gray", "black")) +
  labs(col = "SLF established") +
  xlab("Longitude") + ylab("Latitude") +
  geom_sf(data = states, alpha = 0) +
  coord_sf(xlim = c(-78.5, -78), ylim = c(39, 39.5), expand = FALSE) +
  theme(legend.position = "bottom")
```



The progression of the invasion will be measured starting from the introduction point that has been documented in Barringer et al. 2015. We store the coordinates of the introduction point in an object for the next analyses. If the precise introduction point is unknown, it may be replaced by the centroid of the invasion at the time the invasive species was discovered in the invasive range.

```
# Coordinates of the introduction site, extracted from Barringer et al. 2015
# As a table (for distance calculations):
centroid_df <- data.frame(longitude = -75.675340,
                          latitude = 40.415240)


# Replace by the centroid of the invasion in the first years of the invasion if the
# introduction point is unknown:
# centroid_df <- slf %>% filter(year == 2014) %>%
#   summarise(latitude = mean(.$latitude),
#             longitude = mean(.$longitude))
```

To finish preparing the dataset for analysis, we calculate the distance between each survey cell and the introduction point.

```
grid_data %<>%
  dplyr::mutate(DistToIntro = geosphere::distGeo(grid_data[,c('longitude','latitude')],
                                                 centroid_df[,c('longitude', 'latitude')])/1000)
# the distance is obtained in meters, and we divide it by 1000 to obtain kilometers
# given the scale of this invasion.

summary(grid_data)
```

```
##       year         latitude       longitude       established
##  Min.   :2014   Min.   :29.48   Min.   :-123.27   Mode :logical
##  1st Qu.:2019   1st Qu.:39.89   1st Qu.: -77.07   FALSE:72980
##  Median :2020   Median :40.33   Median : -75.81   TRUE :48601
##  Mean   :2020   Mean   :40.20   Mean   : -76.43   NA's :8434
##  3rd Qu.:2021   3rd Qu.:40.84   3rd Qu.: -74.98
##  Max.   :2022   Max.   :45.71   Max.   : -68.00
##   DistToIntro
##  Min.   :   0.414
##  1st Qu.:  71.635
##  Median : 111.422
##  Mean   : 177.846
##  3rd Qu.: 192.218
##  Max.   :3883.978
```

```
# save this dataset
write.csv(grid_data, file.path(here::here(), "exported-data", "grid_data.csv"), row.names = F)
```

The dataset is now ready for the jump analysis.

3. Differentiating diffusive spread and jump dispersal: principle

A set of four successive functions will be run to separate cells where SLF is established due to continuous, diffusive spread, from cells where SLF is established due to jump dispersal. We will first run the functions on a tentative set of parameters to understand how the analysis works. The optimization of the parameters will be described later in this vignette.

a- attribute_sectors()

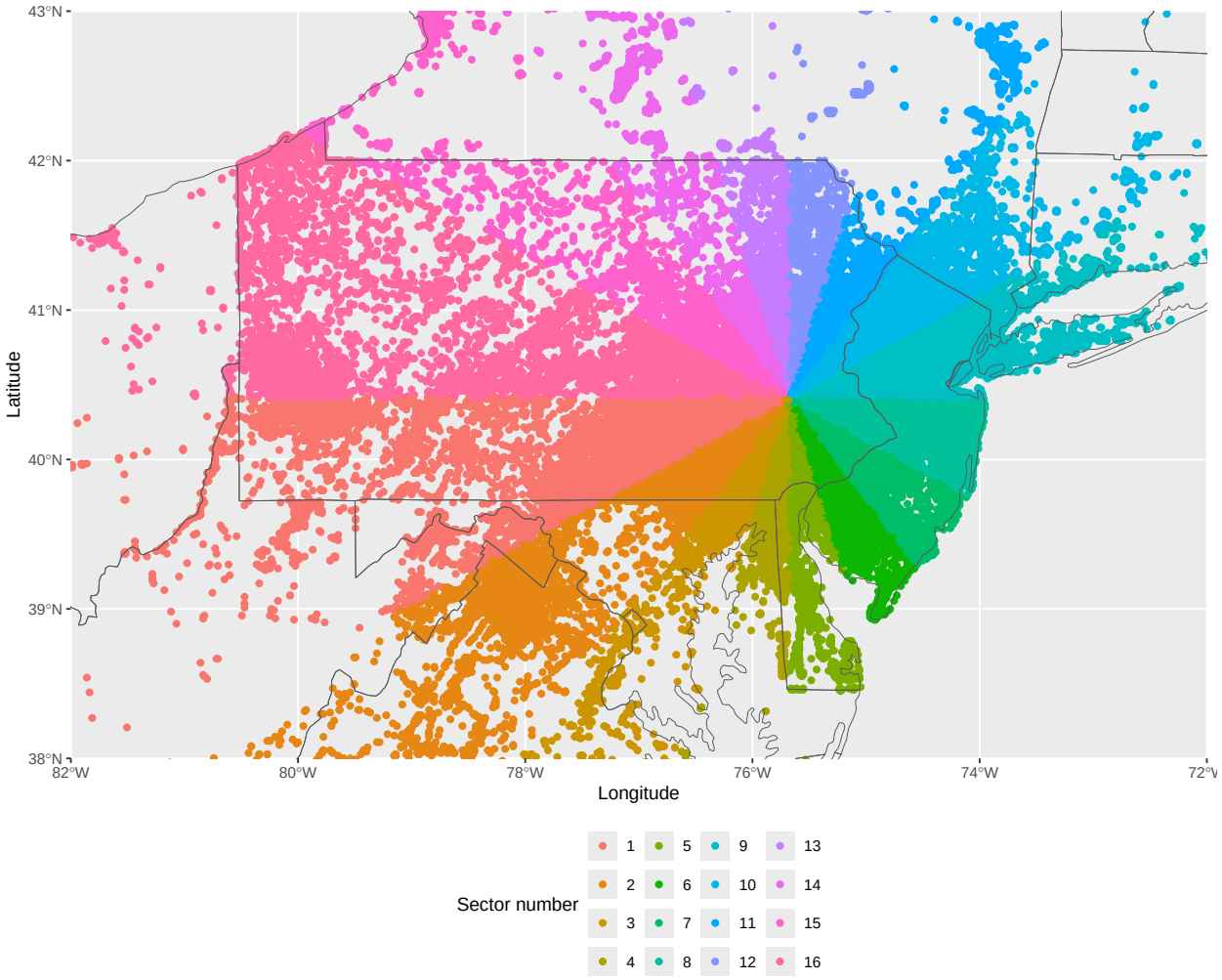
Considering that the expansion of the invasion is heterogeneous in space, the jump analyses requires the invasion range to be divided into sectors with the introduction site as the origin. The first function, `attribute_sectors()` attributes a sector number to each cell of the dataset, to be used in the jump analysis. Here, we divide the invasion range into 16 sectors. See vignette #2 "Decreasing calculation time" to learn how changing the number of sectors may make your analysis faster.

```
grid_data_sectors <- jumpID::attribute_sectors(dataset = grid_data,
                                               # dataset to be explored
                            nb_sectors = 16,
                            # number of sectors to divide the invasion range
```

```
                              centroid = c(-75.675340, 40.415240)
                              # vector containing the centroid coordinates as long/lat
                              )
```

```
## 2024-08-13 11:07:16.62766 Start sector attribution... Sector attribution completed.
```

```
# Map results
ggplot(data = states) +
  geom_point(data = grid_data_sectors,
             aes(x = longitude, y = latitude,
                 col = as.factor(sectors_nb))) +
  geom_sf(data = states, alpha = 0) +
  theme(legend.position = "bottom") +
  xlab("Longitude") + ylab("Latitude") +
  coord_sf(xlim = c(-82, -72), ylim = c(38, 43), expand = FALSE) +
  labs(col = "Sector number")
```



The jump analysis will go through each sector successively.

b- find_thresholds()

The function `find_thresholds()` will then search for the first discontinuity in the SLF distribution. Points are sorted by increasing distance to the introduction point, and the function goes through each consecutive pair of points, starting from the introduction point and going outwards. It stops once it identifies a distance larger than a defined `gap_size` between two consecutive points, marking a discontinuity in the SLF distribution. The last point before this discontinuity marks the putative limit of the diffusive spread. The function runs independently for each sector. It is considered that populations do not go extinct, and data are cumulated over time. The limit of diffusive spread cannot go back towards the introduction point over time.

The input dataset is the output of attribute_sectors(). The `gap_size` parameter defines the minimal distance

between two consecutive points that marks a discontinuity in the SLF distribution, in km. In this example, it is set to 15 km.
If the option "negatives" is set to TRUE (default), the function will check that there are negative surveys in the discontinuity, so that the absence of SLF established is not due to an absence of surveys in the area.

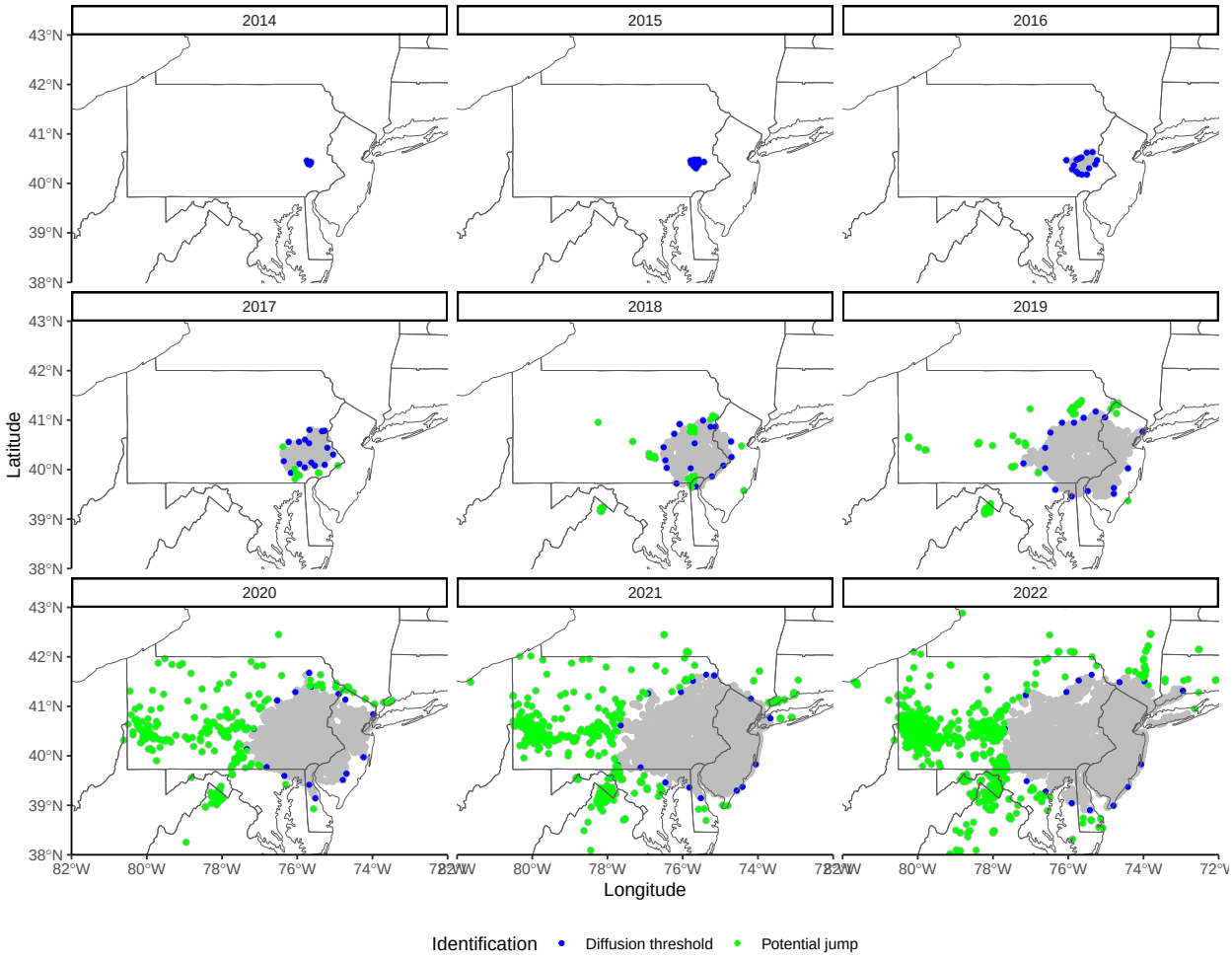
```
Results_thresholds <- jumpID::find_thresholds(dataset = grid_data_sectors,
                                              gap_size = 15,
                                              negatives = T)
```

```
## 2024-08-13 11:07:25.698518 Start finding thresholds... Sector 1/16...  2/16...  3/16...  4/16...  5/1
## Threshold analysis done. 4243 potential jumps were found.
```

```
# Resulting objects are
#- Results_thresholds$preDist, a data frame of threshold cells = extreme points of the
# diffusive spread. Will be completed in find_secDiff()
#- Results_thresholds$potJumps, a data frame of potential jumps. Will be
# pruned in find_jumps()

# Make a single object for the map
thresholds_map <- dplyr::bind_rows(Results_thresholds$preDist %>%
                                     dplyr::mutate(Type = "Diffusion threshold"),
                                   Results_thresholds$potJumps %>%
                                     dplyr::mutate(Type = "Potential jump"))

# Map results
ggplot(data = states) +
  geom_point(data = grid_data %>% filter(established == T),
             aes(x = longitude, y = latitude), col = "gray") +
  geom_point(data = thresholds_map, aes(x = longitude, y = latitude, col = Type), size = 1) +
  geom_sf(data = states, alpha = 0) +
  facet_wrap(~year) +
  theme_classic() +
  theme(legend.position = "bottom") +
  scale_color_manual(values = c("blue", "green")) +
  xlab("Longitude") + ylab("Latitude") + labs(col = "Identification") +
  coord_sf(xlim = c(-82, -72), ylim = c(38, 43), expand = FALSE)
```



On this map, blue points indicate the invasion front identified by `find_thresholds()`, i.e., the limits of the diffusive spread in every direction, for every year. At this point of the analysis, all points farther than these thresholds are stored as potential jumps (green points).

c- find_jumps()

The function `find_jumps()` prunes the list of potential jumps of cells that are less than from a positive cell from a previous year, as the species likely spread from this other cell. The final list of jumps is obtained in the object `Jumps`.

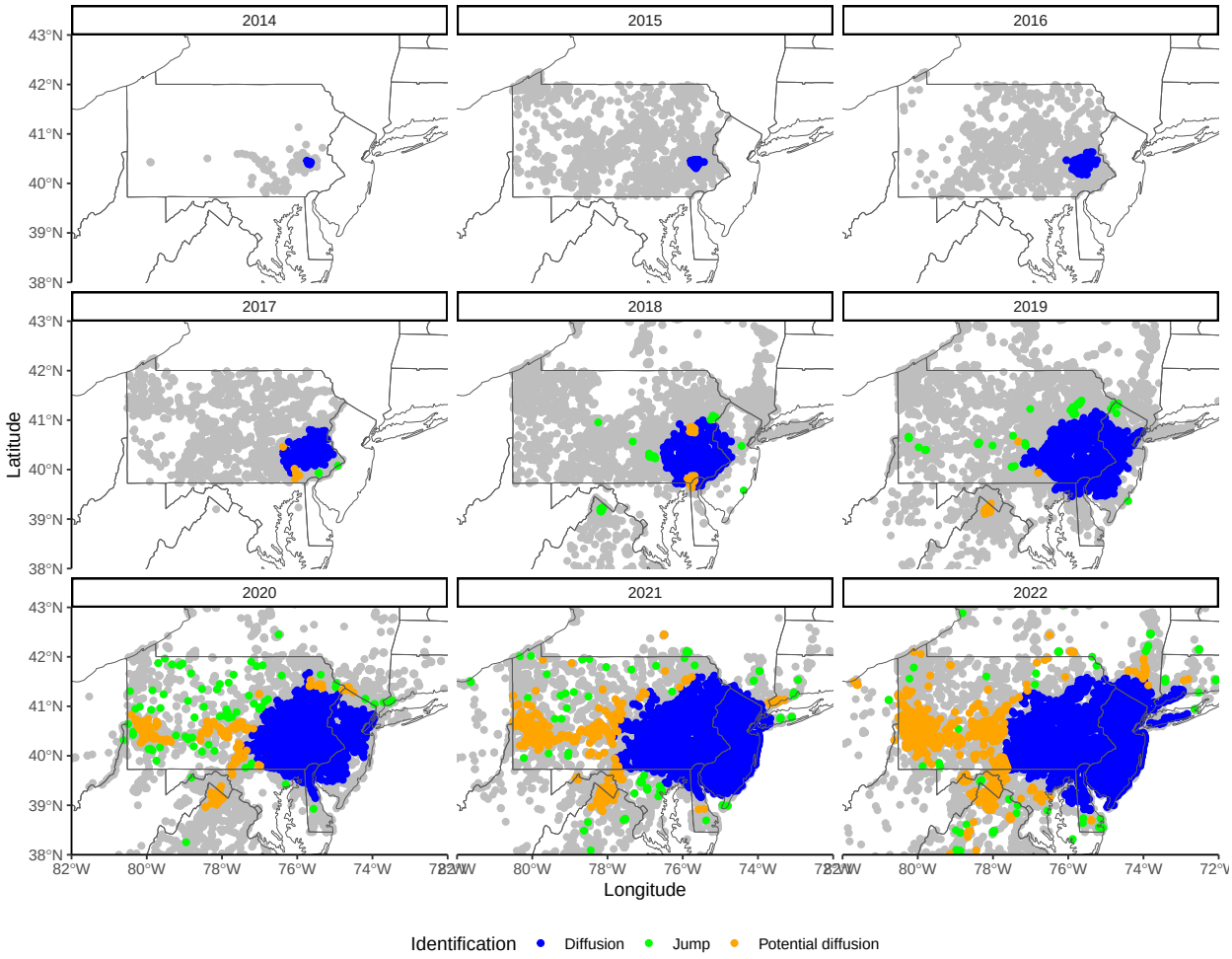
```
Results_jumps <- jumpID::find_jumps(grid_data = grid_data,
                        potJumps = Results_thresholds$potJumps,
                        gap_size = 15)
```

```
## 2024-08-13 11:14:26.523995 Start finding jumps... Year 2014 ... Year 2015 ... Year 2016 ... Year 201
```

```r
# Resulting objects are:
#- Results_jumps$Jumps, a data frame containing all jumps
#- Results_jumps$diffusers, a data frame of positive cells stemming from diffusive spread
#- Results_jumps$potDiffusion, a data frame of remaining cells, containing a mix of
# secondary diffusion and additional threshold points. Will be pruned in find_secDiff()

# Make a single object for the map
jumps_map <- dplyr::bind_rows(Results_jumps$diffusers %>%
                                mutate(Type = "Diffusion"),
                              Results_jumps$Jumps %>%
                                mutate(Type = "Jump"),
                              Results_jumps$potDiffusion %>%
                                mutate(Type = "Potential diffusion"))

# Map results
ggplot(data = states) +
  geom_point(data = grid_data,
             aes(x = longitude, y = latitude), col = "gray") +
  geom_point(data = jumps_map, aes(x = longitude, y = latitude, col = Type)) +
  geom_sf(data = states, alpha = 0) +
  facet_wrap(~year) +
  theme_classic() +
  theme(legend.position = "bottom") +
  scale_color_manual(values = c("blue", "green", "orange")) +
  xlab("Longitude") + ylab("Latitude") + labs(col = "Identification") +
  coord_sf(xlim = c(-82, -72), ylim = c(38, 43), expand = FALSE)
```



The analysis may be stopped here if only dispersal jumps are of biological interest. As a complement, secondary diffusion may be examined in the last function, `find_secDiff()`.

d- find_secDiff()

Cells that were discarded from the jump list are either additional threshold points or secondary diffusion from a previous jump. `find_secDiff()` attributes them to the correct category by checking whether they are close from a previous jump, or a diffusion point.

```
Results_secDiff <- jumpID::find_secDiff(potDiffusion = Results_jumps$potDiffusion,
                                        Jumps = Results_jumps$Jumps,
                                        diffusers = Results_jumps$diffusers,
                                        Dist = Results_thresholds$preDist,
```

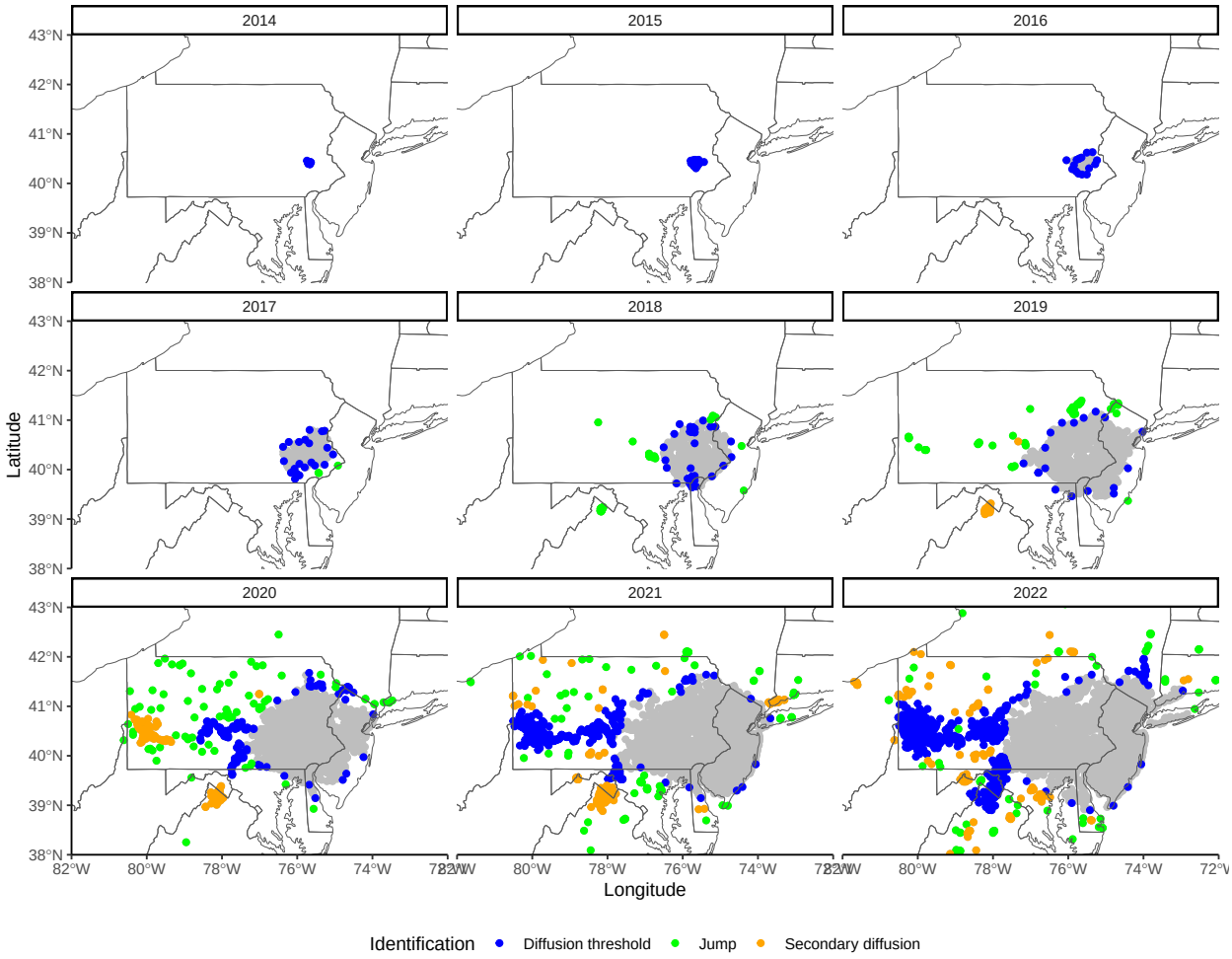
```
                                gap_size = 15)
```

```
## 2024-08-13 11:21:50.667125 Start finding secondary diffusion... Year 2017 ...Year 2018 ...Year 2019
```

```
# Resulting objects are:
#- Results_secDiff$secDiff, a data frame of all cells that are secondary diffusion
# from a previous jump
#- Results_secDiff$Dist, a data frame of all extreme points on the invasion front (thresholds)

# Make a single object for the map
secDiff_map <- dplyr::bind_rows(Results_secDiff$Dist %>%
                                  mutate(Type = "Diffusion threshold"),
                                Results_jumps$Jumps %>%
                                  mutate(Type = "Jump"),
                                Results_secDiff$secDiff %>%
                                  mutate(Type = "Secondary diffusion"))

# Map results
ggplot(data = states) +
  geom_point(data = grid_data %>% filter(established == T),
             aes(x = longitude, y = latitude), col = "gray") +
  geom_point(data = secDiff_map, aes(x = longitude, y = latitude, col = Type)) +
  geom_sf(data = states, alpha = 0) +
  facet_wrap(~year) +
  theme_classic() +
  theme(legend.position = "bottom") +
  scale_color_manual(values = c("blue", "green", "orange")) +
  xlab("Longitude") + ylab("Latitude") + labs(col = "Identification") +
  coord_sf(xlim = c(-82, -72), ylim = c(38, 43), expand = FALSE)
```



4. Wrapper for jump analysis

All these analyses can be done in one instance using the wrapper function.

```
jumps_wrapper <- jumpID::find_jumps_wrapper(dataset = grid_data,
                                            nb_sectors = 16,
                                            centroid = c(-75.675340, 40.415240),
                                            gap_size = 15,
                                            negatives = T)
```

```
## 2024-08-13 11:23:09.114976 Start sector attribution... Sector attribution completed.
## 2024-08-13 11:23:09.140752 Start finding thresholds... Sector 1/16...  2/16...  3/16...  4/16...  5/
```

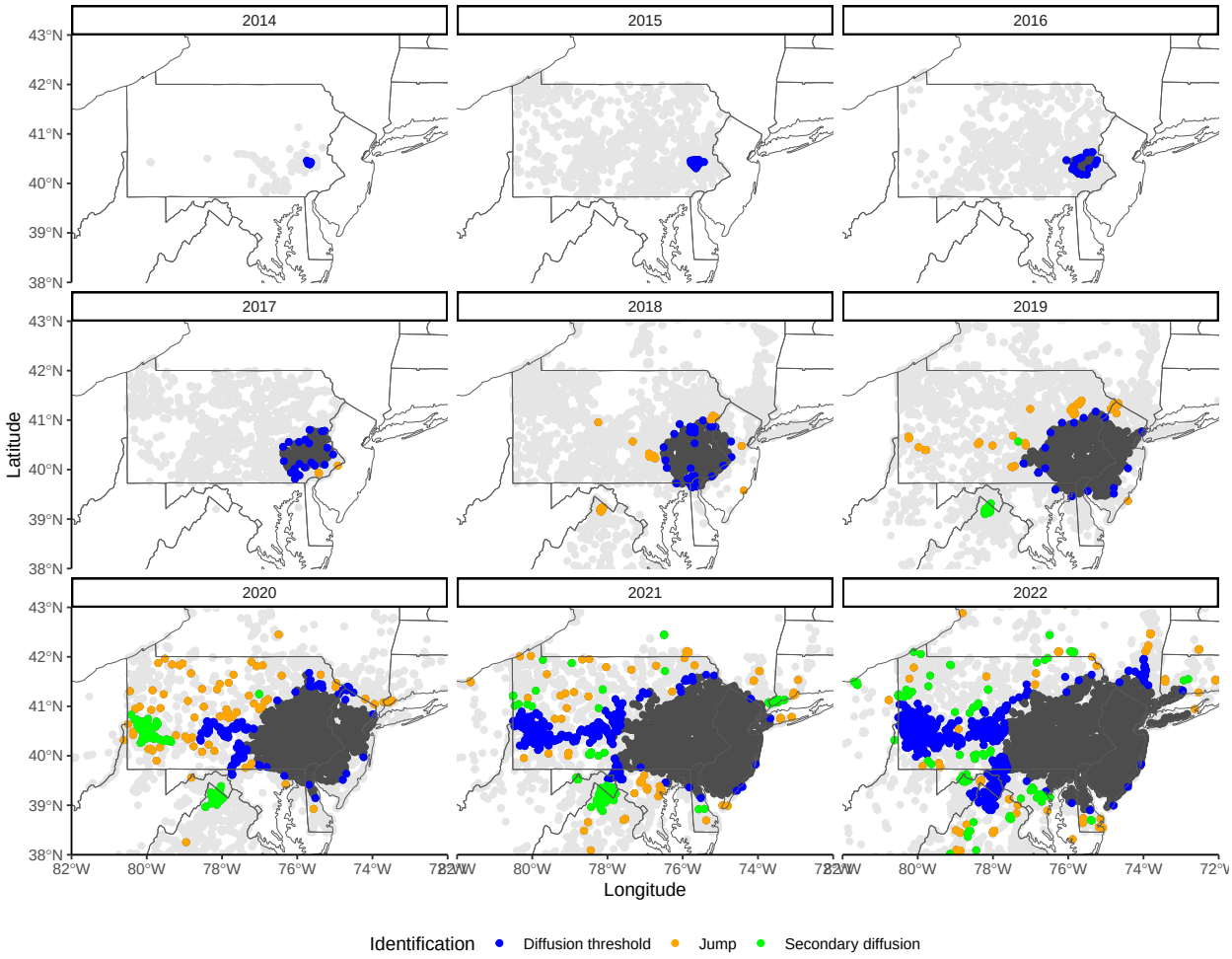
```
## Threshold analysis done. 4243 potential jumps were found.
## 2024-08-13 11:29:54.800121 Start finding jumps... Year 2014 ... Year 2015 ... Year 2016 ... Year 201
## 2024-08-13 11:36:39.67167 Start finding secondary diffusion... Year 2017 ...Year 2018 ...Year 2019 .
```

```r
# save resulting objects
write.csv(jumps_wrapper$Dist,
          file.path(here::here(), "exported-data", "thresholds.csv"), row.names = F)
write.csv(jumps_wrapper$Jumps,
          file.path(here::here(), "exported-data", "jumps.csv"), row.names = F)
write.csv(jumps_wrapper$secDiff,
          file.path(here::here(), "exported-data", "secDiffusion.csv"), row.names = F)
```

We plot the results on a map.

```r
# Make a single object for the map
jumps_wrapper_map <- dplyr::bind_rows(jumps_wrapper$Dist %>%
                                        dplyr::mutate(Type = "Diffusion threshold"),
                                      jumps_wrapper$Jumps %>%
                                        dplyr::mutate(Type = "Jump"),
                                      jumps_wrapper$secDiff %>%
                                        dplyr::mutate(Type = "Secondary diffusion"))

# Map results
ggplot(data = states) +
  geom_point(data = grid_data %>% filter(established == F),
             aes(x = longitude, y = latitude), col = "gray90") +
  geom_point(data = grid_data %>% filter(established == T),
             aes(x = longitude, y = latitude), col = "gray30") +
  geom_point(data = jumps_wrapper_map, aes(x = longitude, y = latitude, col = Type)) +
  geom_sf(data = states, alpha = 0) +
  facet_wrap(~year) +
  theme_classic() +
  theme(legend.position = "bottom") +
  scale_color_manual(values = c("blue", "orange", "green")) +
  xlab("Longitude") + ylab("Latitude") + labs(col = "Identification") +
  coord_sf(xlim = c(-82, -72), ylim = c(38, 43), expand = FALSE)
```



5. Rarefy jump clusters

```
Jumps <- read.csv(file.path(here::here(), "exported-data", "jumps.csv"))
```

Some jumps were clustered in the same area. Jump clusters may stem from independent dispersal jumps, i.e. SLF hitchhiked multiple times to these locations the same year. Alternatively, jump clusters may result from SLF quickly spreading around a single dispersal jump. Finally, they can be a mix between these two hypotheses. We identify these jump clusters to better understand jump locations.

First, we delineate these jump clusters as jumps located less than the gap size from each other. Check how many points there are per jump cluster.

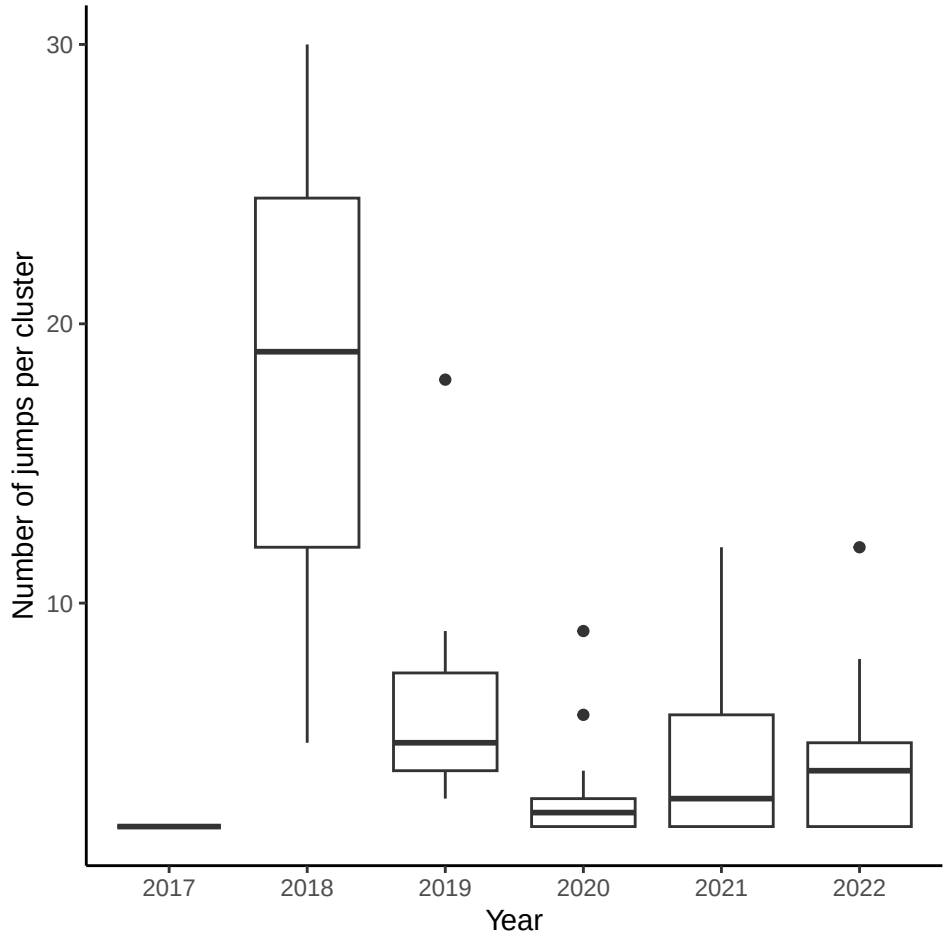
a- group_jumps()

```
Jump_groups <- jumpID::group_jumps(Jumps, gap_size = 15)

Groups <- Jump_groups %>%
  dplyr::group_by(year, Group) %>%
  dplyr::summarise(Nb = n()) %>%
  dplyr::arrange(-Nb) %>%
  dplyr::filter(Nb > 1)
```

```
## `summarise()` has grouped output by 'year'. You can override using the
## `.groups` argument.
```

```
ggplot(Groups, aes(x = as.factor(year), y = Nb)) +
  geom_boxplot() + theme_classic() +
  ylab("Number of jumps per cluster") + xlab("Year")
```



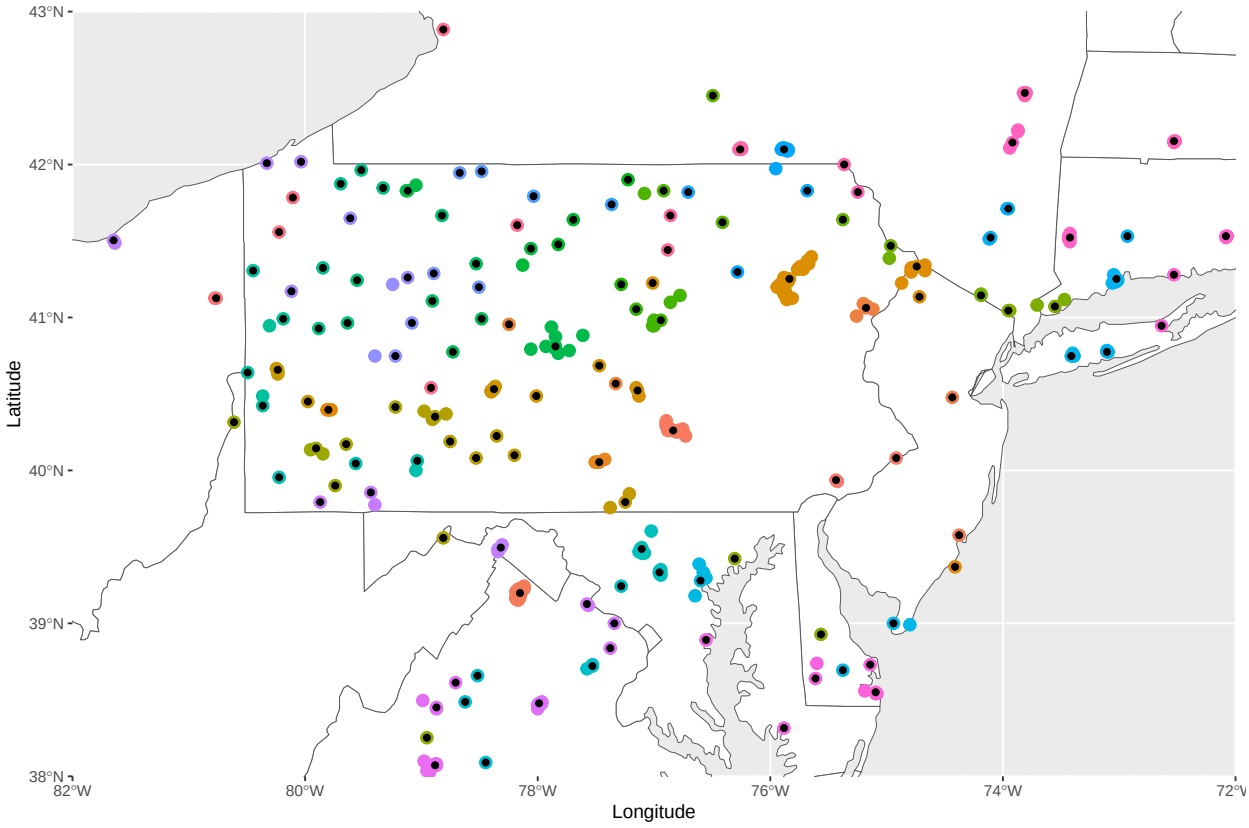
b- rarefy_groups()

We summarize each jump cluster by summarizing each of them by their most central point.

```r
Jump_clusters <- jumpID::rarefy_groups(Jump_groups) %>%
  dplyr::mutate(Rarefied = TRUE)

write.csv(Jump_clusters,
          file.path(here::here(), "exported-data", "jump_clusters.csv"),
          row.names = F)


# Map these groups and the rarefied points
ggplot(data = states) +
  geom_sf(data = states, fill = "white") +
  coord_sf(xlim = c(-82, -72), ylim = c(38, 43), expand = FALSE) +
  geom_point(data = Jump_groups,
             aes(x = longitude, y = latitude,
                 col = as.factor(Group)), shape = 19, size = 3,
             show.legend = F) +
  geom_point(data = Jump_clusters,
             aes(x = longitude, y = latitude)) +
  labs(x = "Longitude", y = "Latitude")
```



These 387 jumps were rarefied into 152 jump clusters.

6. Data is ready for biological analysis and interpretation!

– end of vignette –