#1B Creating the figures associated with the jump analysis

Nadege Belouard* Isabella G. Smith†

2024-08-20

Contents

Aim and setup

This vignette computes all the figures included in the manuscript that is associated with the `jumpID` package. This vignette requires loading data frames generated in the first vignette of this package.

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

```
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(magrittr)
library(ggplot2)
library(cowplot)
library(jumpID)
```

Load files generated in the previous vignette

```
slf <- read.csv(file.path(here::here(), "data", "lyde_data_v2", "lyde.csv"), h=T)
grid_data <- read.csv(file.path(here::here(), "exported-data", "grid_data.csv"), h=T)
centroid <- data.frame(longitude_rounded = -75.675340, latitude_rounded = 40.415240)

Jumps <- read.csv(file.path(here::here(), "exported-data", "jumps.csv"))
Jump_clusters <- read.csv(file.path(here::here(), "exported-data", "jump_clusters.csv"))
Thresholds <- read.csv(file.path(here::here(), "exported-data", "thresholds.csv"))
diffusion <- read.csv(file.path(here::here(), "exported-data", "diffusion.csv"))
secDiffusion <- read.csv(here::here("exported-data", "secdiffusion.csv"))
```

Prepare the states background map

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

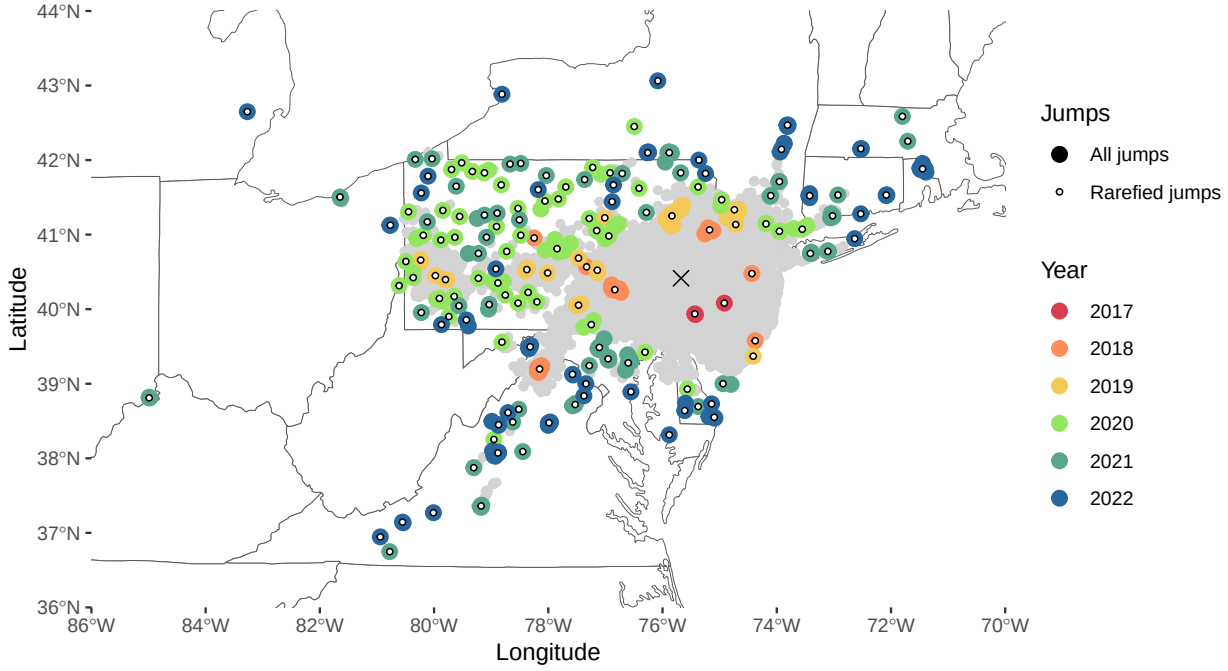
Figure 2: Faceted jump map, barplot & distance

2A: jump map

Map the position of jumps and identify jump clusters per year

```
map_rarefied <- ggplot(data = states) +
  geom_sf(data = states, fill = "white") +
  # positive points
  geom_point(data = grid_data %>% filter(established == TRUE),
             aes(x = longitude, y = latitude), col = "lightgrey") +
  # introduction point
  annotate("point", x = -75.675340, y = 40.415240,
```

```
            col = "black", shape = 4, size = 3) +
  # all jumps
  geom_point(data = Jumps,
             aes(x = longitude, y = latitude, col = as.factor(year),
             shape = "All jumps"), size = 3) +
  # jump clusters
  geom_point(data = Jump_clusters,
             aes(x = longitude, y = latitude, shape = "Rarefied jumps"),
             col = "black", fill = "white", size = 1) +

  scale_color_manual(name = "Year",
                     values = c("#d53e4f", "#fc8d59", "#f4c957", "#94e65f", "#59a68e", "#27679e"),
                     labels = c("2017", "2018", "2019", "2020", "2021", "2022")) +
  scale_shape_manual(name = "Jumps",
                     values = c("All jumps" = 19, "Rarefied jumps" = 21)) +
  coord_sf(xlim = c(-86, -70), ylim = c(36, 44), expand = FALSE) +
  labs(x = "Longitude", y = "Latitude") +
  theme(panel.background = element_rect(fill = "white"))

map_rarefied
```



2B: number of jumps per year bar plot

Count how many jumps there are per year

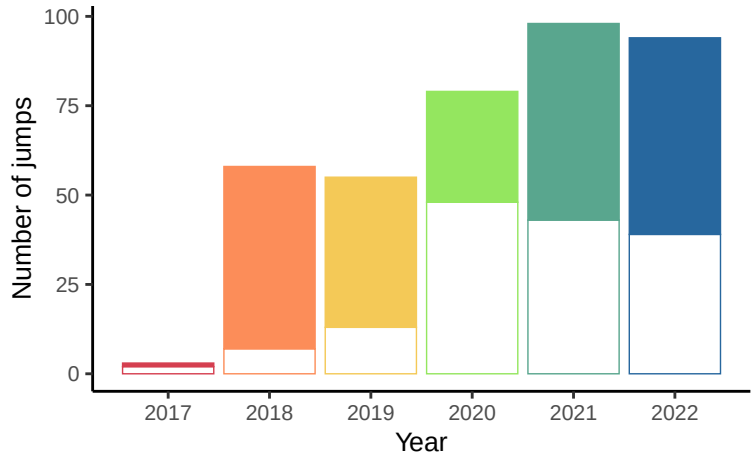
```
# add the type of data to each dataset
Jump_clusters %<>% mutate(Type = "Rarefied jumps")
Jumps %<>% mutate(Type = "All jumps")

# count the number of clusters per year
Clusters_year <- Jump_clusters %>%
  group_by(year, Type) %>%
  summarise(n = n())
```

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

```
# count the total number of jumps per year
Jumps_year <- Jumps %>%
  group_by(year, Type) %>%
  summarise(n = n()) %>%
  left_join(Clusters_year, by = "year") %>%
  mutate(n = n.x - n.y) %>%
  rename(Type = Type.x)
```

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

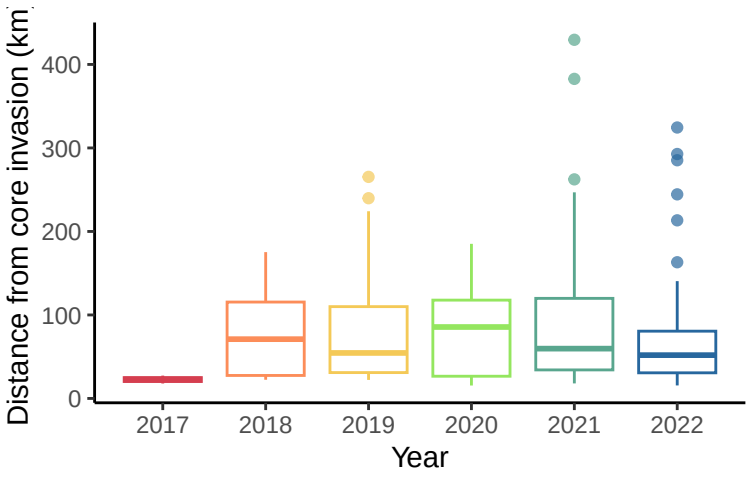
```
# bind the two tables
Jumps_total <- bind_rows(Clusters_year, Jumps_year %>% select(year, Type, n))


jumps_plot <- ggplot() +
  geom_bar(data = Jumps_total,
           aes(x = year, y = n, fill = as.factor(year), col = as.factor(year),
               group = Type, alpha = Type),
           stat = "identity", lwd = .25, show.legend = F) +
  scale_fill_manual(values = c("#d53e4f", "#fc8d59", "#f4c957", "#94e65f", "#59a68e", "#27679e")) +
  scale_color_manual(values = c("#d53e4f", "#fc8d59", "#f4c957", "#94e65f", "#59a68e", "#27679e")) +
  scale_alpha_manual(values = c(1, 0)) +
  xlab("Year") + ylab("Number of jumps") +
  theme_classic() +
  scale_x_continuous(breaks = seq(2017, 2022, by = 1)) +
  theme(text = element_text(size = 10))

jumps_plot
```



2C: distance of jumps box plot

calculate the distance between the invasion front and jumps every year

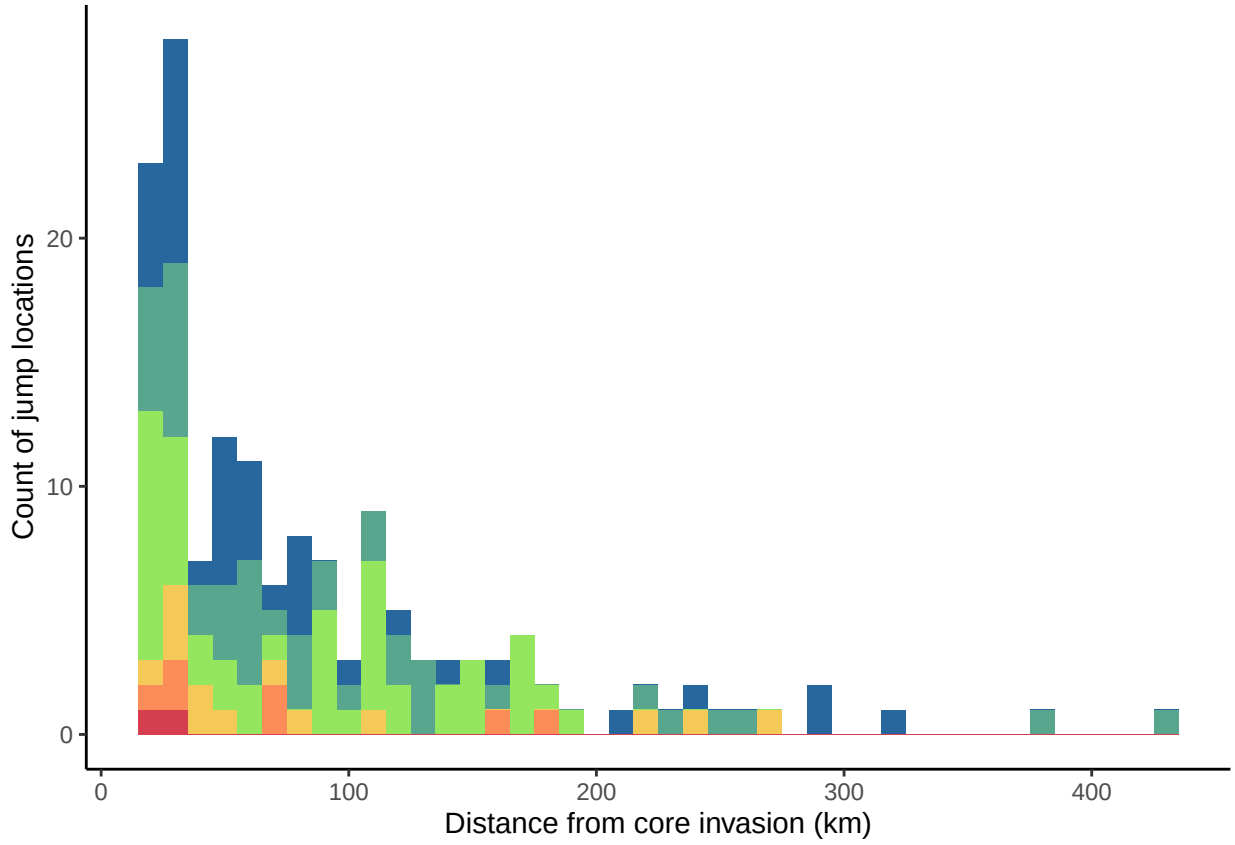
```r
# 1. select all diffusion points for the rarefied jump dataset
diffusion_rarefied <- setdiff(grid_data %>% filter(established == T),
                      Jump_clusters %>% select(year, latitude, longitude, established, DistToIntro)) %
  setdiff(secDiffusion) # remove secondary diffusion too

#2. calculate jump distance for each jump
for (jump in 1:length(Jump_clusters$DistToIntro)){ #for each jump
  y = Jump_clusters[jump,] %>% pull(year)
  diffusion_y <- diffusion %>% filter(year == y)
  pairwise_dist <- geosphere::distGeo(diffusion_y[,c('longitude','latitude')], Jump_clusters[jump,c('lo
  Jump_clusters$DistToFront[jump] = min(pairwise_dist)
}


MeanDist_jump <- ggplot() +
  geom_boxplot(data = Jump_clusters, show.legend = F,
               aes(x = year,
                 y = DistToFront,
                   col = as.factor(year)
                 ),
               alpha = 0, outlier.alpha = 0.7) +

  scale_color_manual(name = "Year", values = c("#d53e4f", "#fc8d59", "#f4c957", "#94e65f", "#59a68e", "#
  scale_x_continuous(breaks = seq(2017, 2022, by = 1)) +
  labs(x = "Year",
    y = "Distance from core invasion (km)") +
  theme_classic()

MeanDist_jump
```



2D Evolution of jump distances

```r
Distrib_jumpDist <- ggplot(Jump_clusters, aes(x = DistToFront)) +
  geom_histogram(aes(fill = as.factor(-year)), binwidth = 10, show.legend = F) +
  scale_fill_manual(name = "Year", values = c("#27679e", "#59a68e", "#94e65f", "#f4c957", "#fc8d59", "#
```

```
  theme_classic() +
  labs(x = "Distance from core invasion (km)",
       y = "Count of jump locations")

Distrib_jumpDist
```



Linear model for statistical test

```
# generate model
model <- lm(log(DistToFront) ~ year, data = Jump_clusters)
# look at residuals
shapiro.test(model$residuals)
```

```
##
##  Shapiro-Wilk normality test
##
## data:  model$residuals
## W = 0.96383, p-value = 0.0005085
```

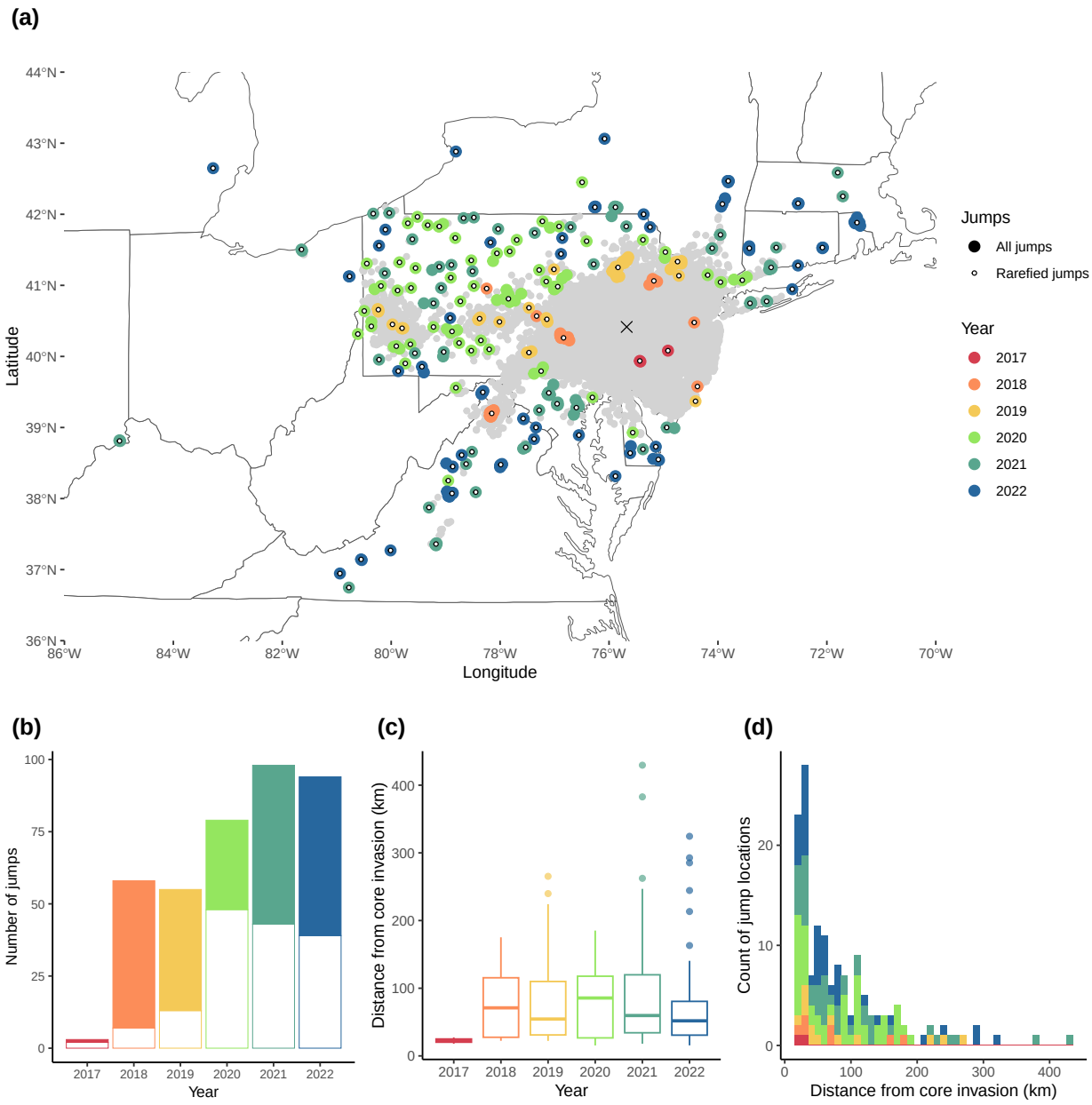
```
# look at results
summary(model)
```

```
##
## Call:
```

```
## lm(formula = log(DistToFront) ~ year, data = Jump_clusters)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1.41134 -0.74153 -0.09242  0.61409  1.92588
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) -38.04036  115.38662  -0.330    0.742
## year          0.02087    0.05711   0.365    0.715
##
## Residual standard error: 0.8217 on 150 degrees of freedom
## Multiple R-squared:  0.0008896,  Adjusted R-squared:  -0.005771
## F-statistic: 0.1336 on 1 and 150 DF,  p-value: 0.7153
```

Assemble figure 2ABCD

```
fig2 <- ggdraw() +
  draw_plot(map_rarefied, x = 0, y = .33, width = 1, height = .66) +
  draw_plot(jumps_plot, 0, 0, .33, .33) +
  draw_plot(MeanDist_jump, .33, 0, .33, .33) +
  draw_plot(Distrib_jumpDist, .66, 0, .33, .33) +
  draw_plot_label(c("(a)", "(b)", "(c)", "(d)"), c(0, 0, 0.33, 0.66), c(1, 0.36, 0.36, 0.36), size = 15)
  theme(plot.background = element_rect(fill="#FFFFFF", color = NA))

fig2
```




```
ggsave(file.path(here::here(), "figures", "2. jump_description.jpg"),
       fig2, height = 10, width = 10)
```

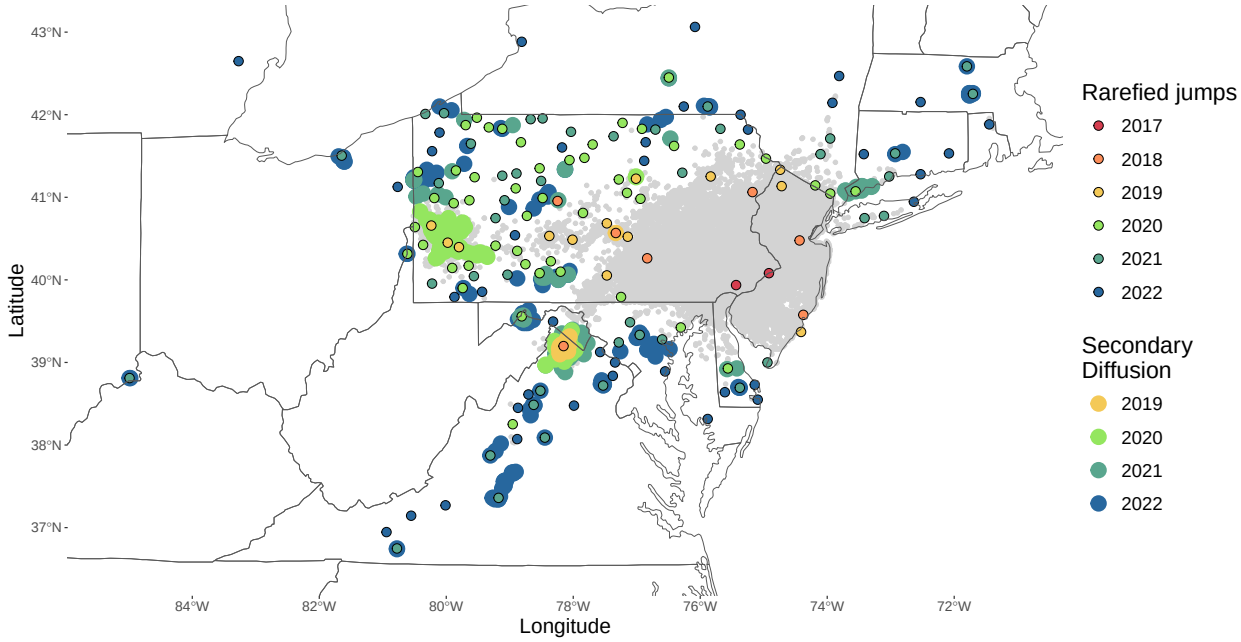
Figure 3: secondary diffusion

3A: Map

Map the points identified as secondary diffusion around dispersal jumps.

```
secDiff_map <- ggplot() +
  geom_point(data = grid_data %>% filter(established == TRUE),
```

```
            aes(x = longitude, y = latitude),
            col = "lightgrey") +
  geom_point(data = secDiffusion %>% arrange(desc(year)),
            aes(x = longitude, y = latitude,
                col = as.factor(year)),
            stroke = 2, size = 4) +
  geom_point(data = Jump_clusters %>% arrange(desc(year)),
            aes(x = longitude, y = latitude, col = as.factor(year), fill = as.factor(year)),
            color = "black",
             shape = 21,
            stroke = 0.5, size = 3.5) +
  scale_color_manual(values = c( "#f4c957", "#94e65f", "#59a68e", "#27679e")) +
  scale_fill_manual(values = c( "#d53e4f", "#fc8d59", "#f4c957", "#94e65f", "#59a68e", "#27679e")) +
  labs(x = "Longitude", y = "Latitude") +
  theme(legend.position = "right", text = element_text(size = 10),
        panel.background = element_rect(fill = "white"),
        legend.key = element_rect(fill = "white"),
        legend.title = element_text(size = 20),
        legend.text = element_text(size = 16),
        legend.key.size = unit(2, "lines"),
        axis.title = element_text(size = 18),
        axis.text = element_text(size = 12)) +
    geom_sf(data = states, alpha = 0) +
    coord_sf(xlim = c(-85.25, -71), ylim = c(36.5, 43)) +
  guides(colour = guide_legend(paste0("Secondary", "\n", "Diffusion")),
         fill = guide_legend("Rarefied jumps"))

secDiff_map
```



3B: Secondary diffusion Bar Plot

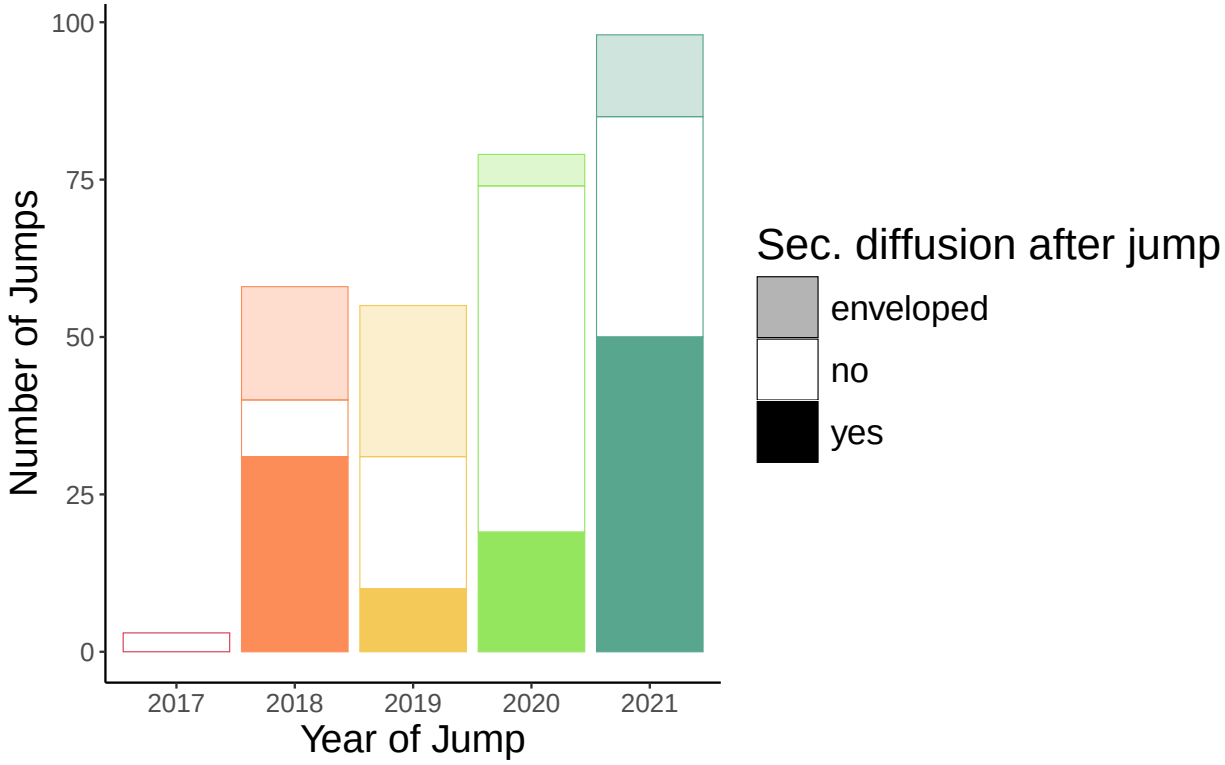
Count how many jumps were followed by secondary diffusion

```
# select jumps for which we have data for year n+1
Jumps_2021 <- Jumps %>% filter(year %in% c(2014:2021)) %>%
  mutate(secDiff = NA)

# determine if they are followed by secondary diffusion, or enveloped by diffusion
for (jump in 1:length(Jumps_2021$DistToIntro)){ # for each jump
  # is this jump within 1 km of any diffusion point the year after? ie. enveloped by diffusion
  testDiff <- diffusion %>% filter(year == Jumps_2021$year[jump]+1)
  pairwise_dist <- geosphere::distGeo(testDiff[,c('longitude','latitude')], Jumps_2021[jump,c('longitud

  if (min(pairwise_dist) < 1){ Jumps_2021$secDiff[jump] = "enveloped"
  } else {
  # is there secondary diffusion within 15 km of this jump the year after?
  # calculate pairwise distance with secDiff the year after
    testSecDiff <- secDiffusion %>% filter(year == Jumps_2021$year[jump]+1)
    if (dim(testSecDiff)[1] > 0){
      pairwise_dist <- geosphere::distGeo(testSecDiff[,c('longitude','latitude')], Jumps_2021[jump,c('l
      Jumps_2021$secDiff[jump] = ifelse(min(pairwise_dist) < 15, "yes", "no")
    } else { Jumps_2021$secDiff[jump] = "no" } # if there is no secDiff
  }
}

# count the number of jumps in each category (yes, no, enveloped)
Jumps_secDiff <- Jumps_2021 %>%
  group_by(year, secDiff) %>%
  summarise(count = n()) %>%
  ungroup()
```

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

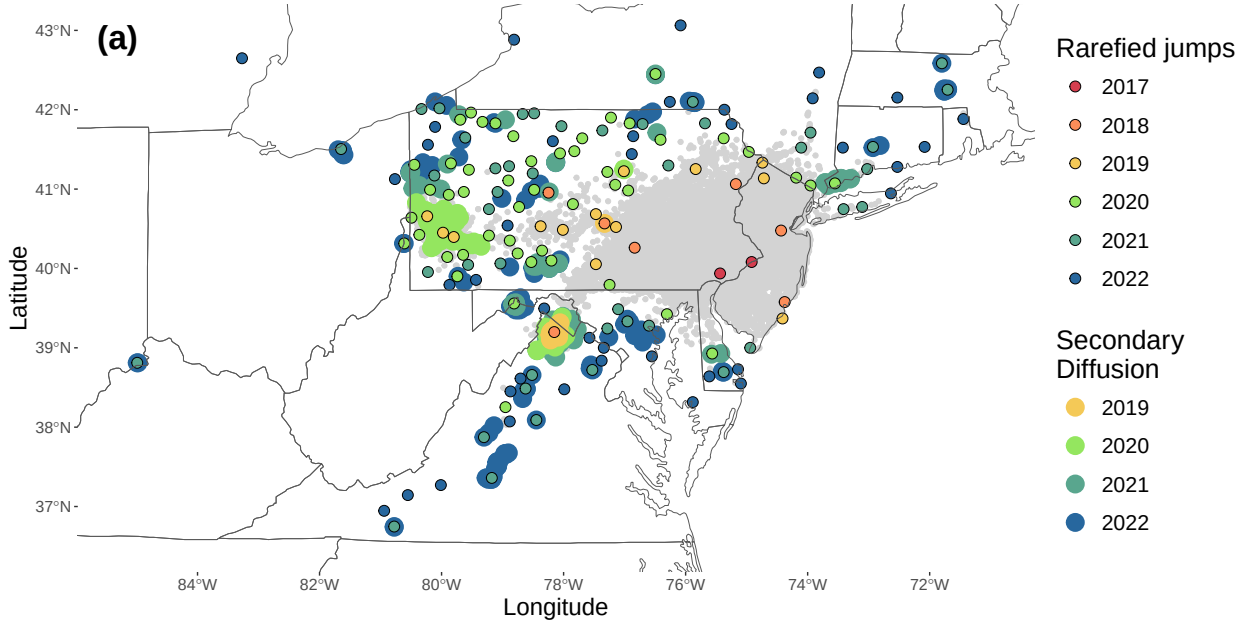
```
# order this factor
Jumps_secDiff$secDiff <- factor(Jumps_secDiff$secDiff,
                                levels = c("enveloped", "no", "yes"))

#Bar Plot
Jumps_secDiff_plot <- ggplot() +
  geom_bar(data = Jumps_secDiff,
           aes(x = as.factor(year),
               y = count,
               group = secDiff,
               fill = as.factor(year),
               col = as.factor(year),
               alpha = secDiff),
           lwd = .25,
           stat = "identity") +
  scale_fill_manual(values = c("#d53e4f", "#fc8d59", "#f4c957", "#94e65f",
                               "#59a68e", "#27679e"), guide = "none") +
  scale_color_manual(values = c("#d53e4f", "#fc8d59", "#f4c957", "#94e65f",
```

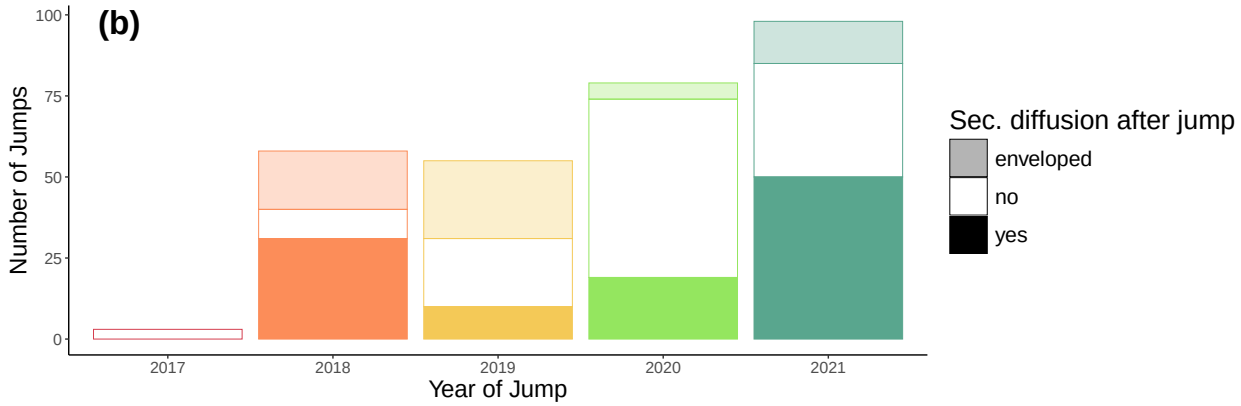
```
                                  "#59a68e", "#27679e"), guide = "none") +
  scale_alpha_manual(values = c(0.3, 0, 1)) +
  theme_classic() +
  xlab("Year of Jump") +
  ylab("Number of Jumps") +
  guides(alpha = guide_legend(
    paste0("Sec. diffusion after jump"),
    override.aes = list(fill = "black", color = "black", linetype = 1, shape = 21))) +
  theme(legend.title = element_text(size = 20),
        legend.text = element_text(size = 16),
        legend.key.size = unit(2, "lines"),
        axis.title = element_text(size = 18),
        axis.text = element_text(size = 12))

Jumps_secDiff_plot
```



Assemble figure 3AB

```
secDiff_graphs_combined <- cowplot::ggdraw() +
  cowplot::draw_plot(secDiff_map, 0, .33, 1, 0.66) +
  cowplot::draw_plot(Jumps_secDiff_plot, 0, 0, 1, 0.33) +
  cowplot::draw_plot_label(c("(a)", "(b)"), c(0.06, 0.06), c(0.90, 0.33), size = 25)

secDiff_graphs_combined
```






```
ggsave(file.path(here::here(), "figures", "3. secDiff_combined.jpeg"),
       secDiff_graphs_combined, height = 13, width = 13)
```

Figure 4: Invasion radius

To estimate the spread of the SLF, we extract for each year the radius of the invasion in each sector. We can look at how the radius of the invasion increases over time, when differentiating diffusive spread and jump dispersal.

```
sectors_used = 16

# calculate max radius for diffusion only
```

```r
thresholdMaxSector <- data.frame(NULL)

# take the threshold data (most extreme diffusion points)
for (y in unique(Thresholds$year)){ # for each year,
  # distribute data in sectors
  thresholdSectors <- jumpID::attribute_sectors(Thresholds %>%
                                                  select(year, latitude, longitude, DistToIntro) %>%
                                                  filter(year %in% c(2014:y)),
                                                nb_sectors = sectors_used,
                                                centroid = c(-75.67534, 40.41524))

  # for each sector, take the maximum radius
  thresholdMaxSectorYear <- thresholdSectors %>%
    group_by(sectors_nb) %>%
    summarise(maxDistToIntro = max(DistToIntro)) %>%
    mutate(year = y)

  # add it to the table
  thresholdMaxSector <- rbind(thresholdMaxSector, thresholdMaxSectorYear)
}
```

```
## 2024-08-20 22:49:16.752606 Start sector attribution... Sector attribution completed.
## 2024-08-20 22:49:16.772497 Start sector attribution... Sector attribution completed.
## 2024-08-20 22:49:16.790311 Start sector attribution... Sector attribution completed.
## 2024-08-20 22:49:16.805764 Start sector attribution... Sector attribution completed.
## 2024-08-20 22:49:16.821054 Start sector attribution... Sector attribution completed.
## 2024-08-20 22:49:16.837059 Start sector attribution... Sector attribution completed.
## 2024-08-20 22:49:16.891427 Start sector attribution... Sector attribution completed.
## 2024-08-20 22:49:16.908583 Start sector attribution... Sector attribution completed.
## 2024-08-20 22:49:16.924006 Start sector attribution... Sector attribution completed.
```

```r
# do the same thing for jumps
# calculate max radius for jumps
jumpMaxSector = data.frame(NULL)

for (y in unique(Jumps$year)){
  jumpSectors <- jumpID::attribute_sectors(Jumps %>%
                                             select(year, latitude, longitude, DistToIntro) %>%
                                             filter(year %in% c(2014:y)),
                                           nb_sectors = sectors_used,
                                           centroid = c(-75.67534, 40.41524))

  jumpsMaxSectorYear <- jumpSectors %>%
    group_by(sectors_nb) %>%
    summarise(maxDistToIntro = max(DistToIntro)) %>%
    mutate(year = y)

  jumpMaxSector <- rbind(jumpMaxSector, jumpsMaxSectorYear)
}
```

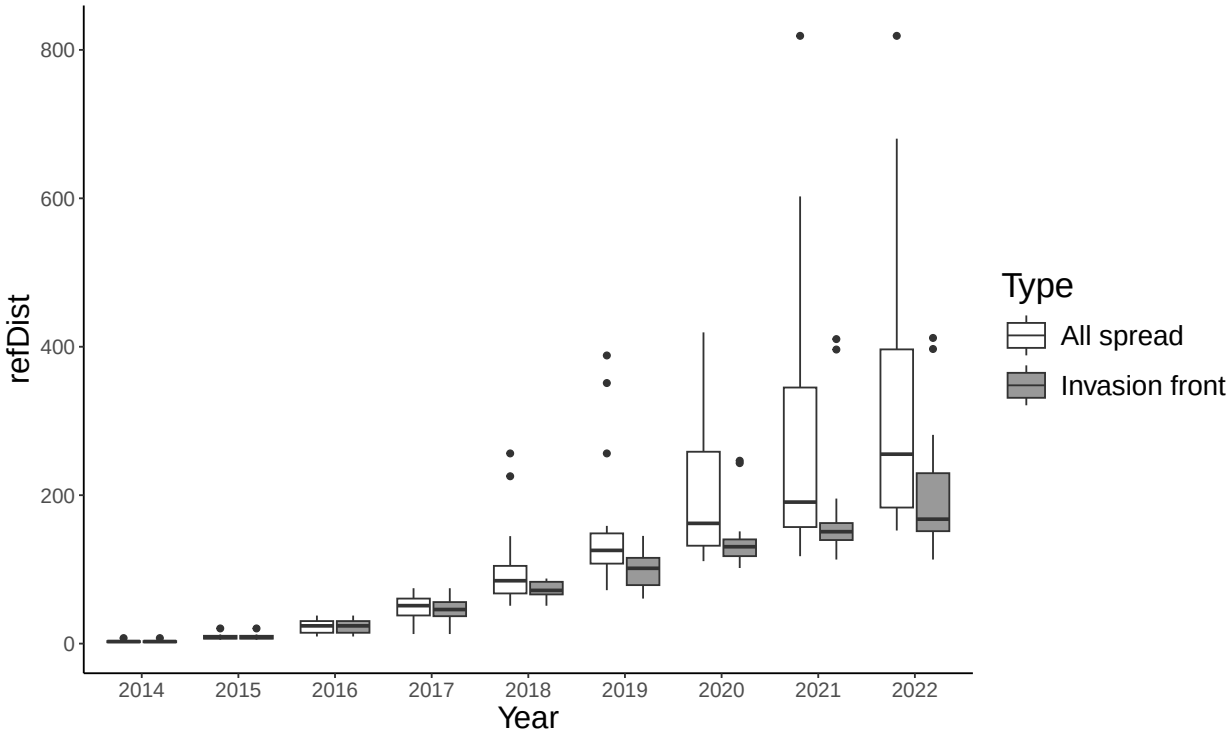
```
## 2024-08-20 22:49:16.954632 Start sector attribution... Sector attribution completed.
## 2024-08-20 22:49:16.969252 Start sector attribution... Sector attribution completed.
## 2024-08-20 22:49:16.985418 Start sector attribution... Sector attribution completed.
```

```
## 2024-08-20 22:49:17.000429 Start sector attribution... Sector attribution completed.
## 2024-08-20 22:49:17.016153 Start sector attribution... Sector attribution completed.
## 2024-08-20 22:49:17.031255 Start sector attribution... Sector attribution completed.
```

```r
# we combine thresholds and jumps to get the "all spread" dataset
allMaxSector <- bind_rows(thresholdMaxSector, jumpMaxSector) %>%
  group_by(year, sectors_nb) %>%
  summarise(maxDistToIntro = max(maxDistToIntro))
```

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

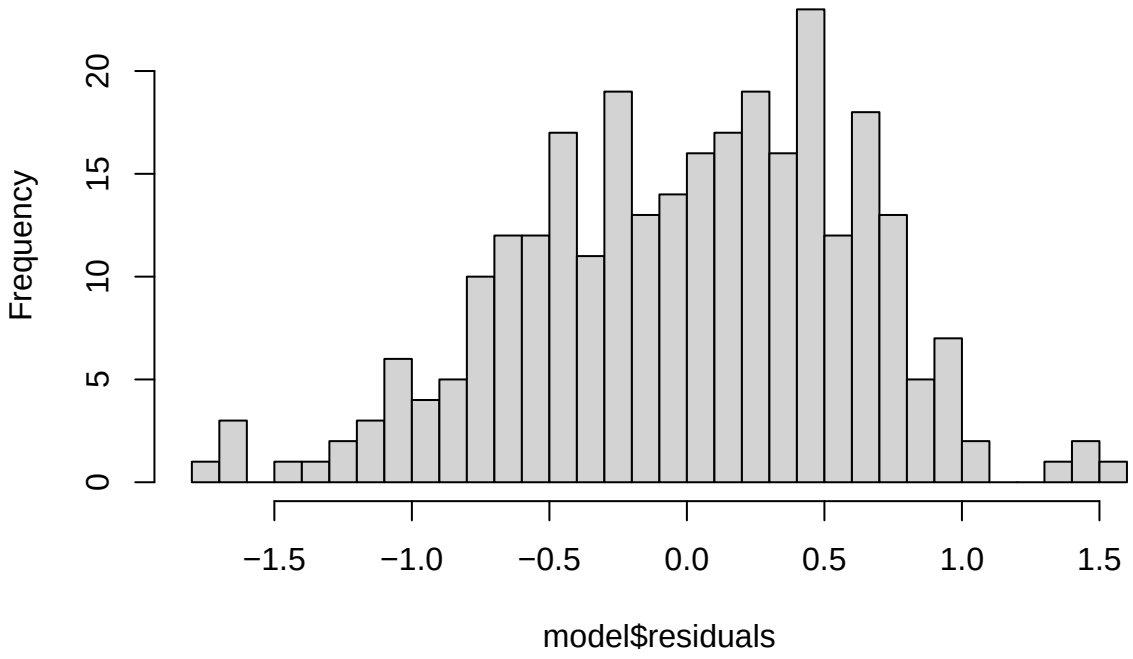
```r
# combine all spread and invasion front data
radiusData <- bind_rows(allMaxSector %>%
                          mutate(Type = "All spread"),
                        thresholdMaxSector %>% mutate(Type = "Invasion front"))

# plot results
invasionRadius <- ggplot(data = radiusData,
                         aes(x = as.factor(year), y = maxDistToIntro,
                             fill = Type)) +
  geom_boxplot() +
  scale_fill_manual(name = "Type", values = c("All spread" = "white",
                                              "Invasion front" = "gray60")) +
  theme_classic() +
  labs(x = "Year", y = "refDist") +
  theme(legend.title = element_text(size = 20),
      legend.text = element_text(size = 16),
      legend.key.size = unit(2, "lines"),
      axis.title = element_text(size = 18),
      axis.text = element_text(size = 12))

invasionRadius
```




```
ggsave(file.path(here::here(), "figures", "4. invasionRadius.jpeg"),
       invasionRadius, height = 6, width = 10)
```

Test the difference in invasion radius between spread types

```
# generate model
model <- lm(log(maxDistToIntro) ~ year*Type, data = radiusData)
# look at residuals
hist(model$residuals, breaks = 30)
```




```
# look at results
anova(model)
```

```
## Analysis of Variance Table
##
## Response: log(maxDistToIntro)
##            Df Sum Sq Mean Sq   F value    Pr(>F)
## year        1 552.44  552.44 1510.4347 < 2.2e-16 ***
## Type        1   2.92    2.92    7.9834  0.005058 **
## year:Type   1   1.89    1.89    5.1753  0.023661 *
## Residuals 282 103.14    0.37
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Calculate the yearly increase in invasion radius

```
# calculate the mean radius per year
meanRadius <- radiusData %>% group_by(Type, year) %>%
  summarise(mean = mean(maxDistToIntro)) %>%
  ungroup()
```

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

```r
# All spread:
meanRadius %>% filter(Type == "All spread") %>%
  mutate(radiusIncrease = c(NA, diff(mean))) %>%
  summarise(mean = mean(radiusIncrease, na.rm = T),
            sd = sd(radiusIncrease, na.rm = T))
```

```
## # A tibble: 1 x 2
##    mean    sd
##   <dbl> <dbl>
## 1  41.3  23.6
```

```r
# Invasion front:
meanRadius %>% filter(Type == "Invasion front") %>%
  mutate(radiusIncrease = c(NA, diff(mean))) %>%
  summarise(mean = mean(radiusIncrease, na.rm=T),
            sd = sd(radiusIncrease, na.rm = T))
```

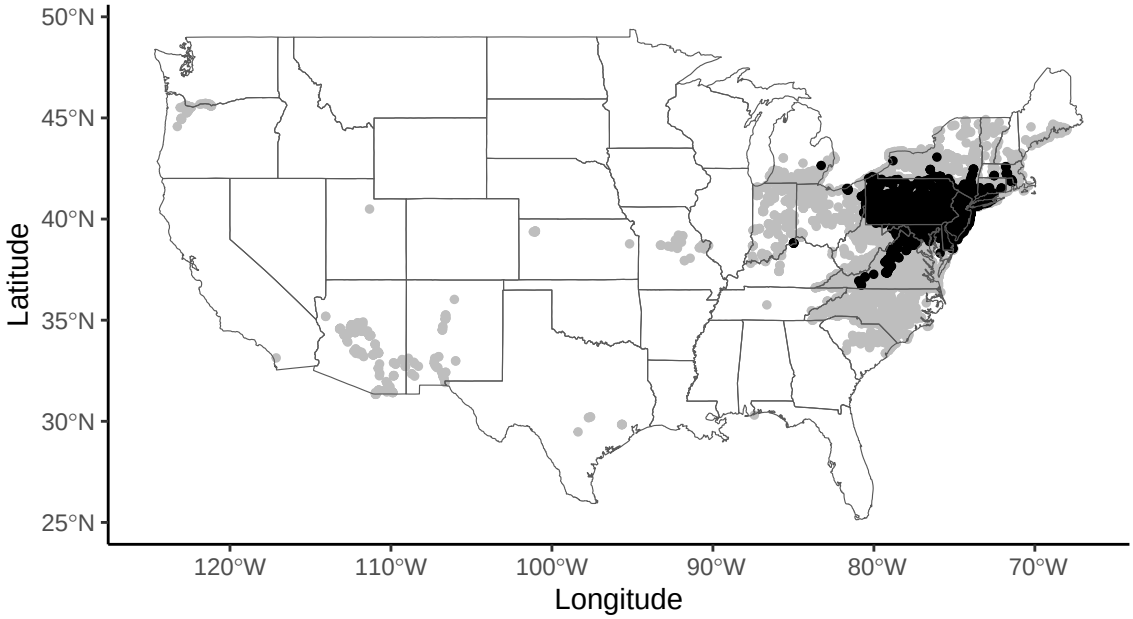
```
## # A tibble: 1 x 2
##    mean    sd
##   <dbl> <dbl>
## 1  25.1  11.4
```

Supplementary figures

Figure S1: Map all points

Facet A: Overview of all SLF surveys

```r
# plot US map
mapUS <- ggplot(data = states) +
  geom_point(data = grid_data %>% filter(established == FALSE),
             aes(x = longitude, y = latitude), col = "gray", size = 1) +
  geom_point(data = grid_data %>% filter(established == TRUE),
             aes(x = longitude, y = latitude), col = "black", size = 1) +
  geom_sf(alpha = 0) +
  labs(x = "Longitude", y = "Latitude") +
  theme_classic() +
  theme(legend.position = "bottom", legend.key = element_rect(fill = "white", colour = NA))

mapUS
```

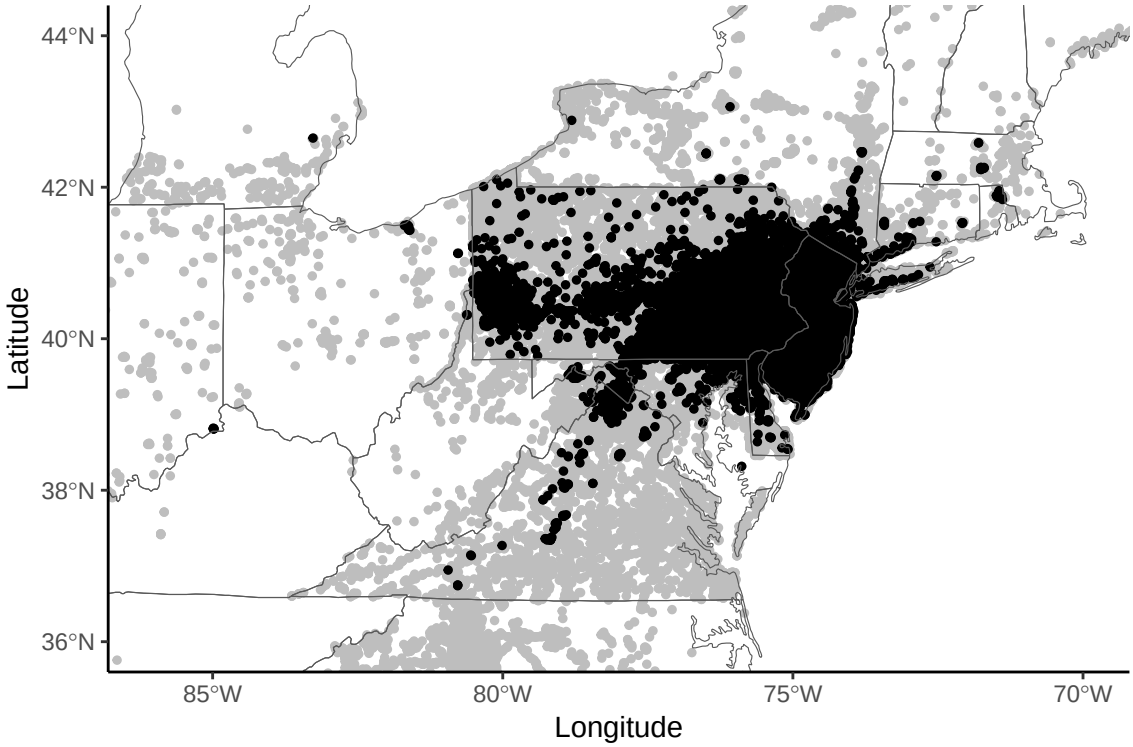



```
# save it
ggsave(file.path(here::here(), "figures", "S1A. points_all.jpg"),
       mapUS, width = 6, height = 6)
```

Facet B: Zoomed map on established SLF

```
# create a variable for a meaningful legend
grid_data %<>% mutate(SLF = ifelse(established == T, "Present", "Absent"))

# plot zoomed map
zoomedMap <- ggplot(data = states) +
  geom_point(data = grid_data %>% filter(established == FALSE),
             aes(x = longitude, y = latitude), col = "gray", size = 1) +
  geom_point(data = grid_data %>% filter(established == TRUE),
             aes(x = longitude, y = latitude), col = "black", size = 1) +
  geom_sf(alpha = 0) +
```

```
  labs(x = "Longitude", y = "Latitude") +
  coord_sf(xlim = c(-86, -70), ylim = c(36, 44)) +
  theme_classic() +
  theme(legend.position="bottom", legend.key = element_rect(fill = "white", colour = NA))

zoomedMap
```

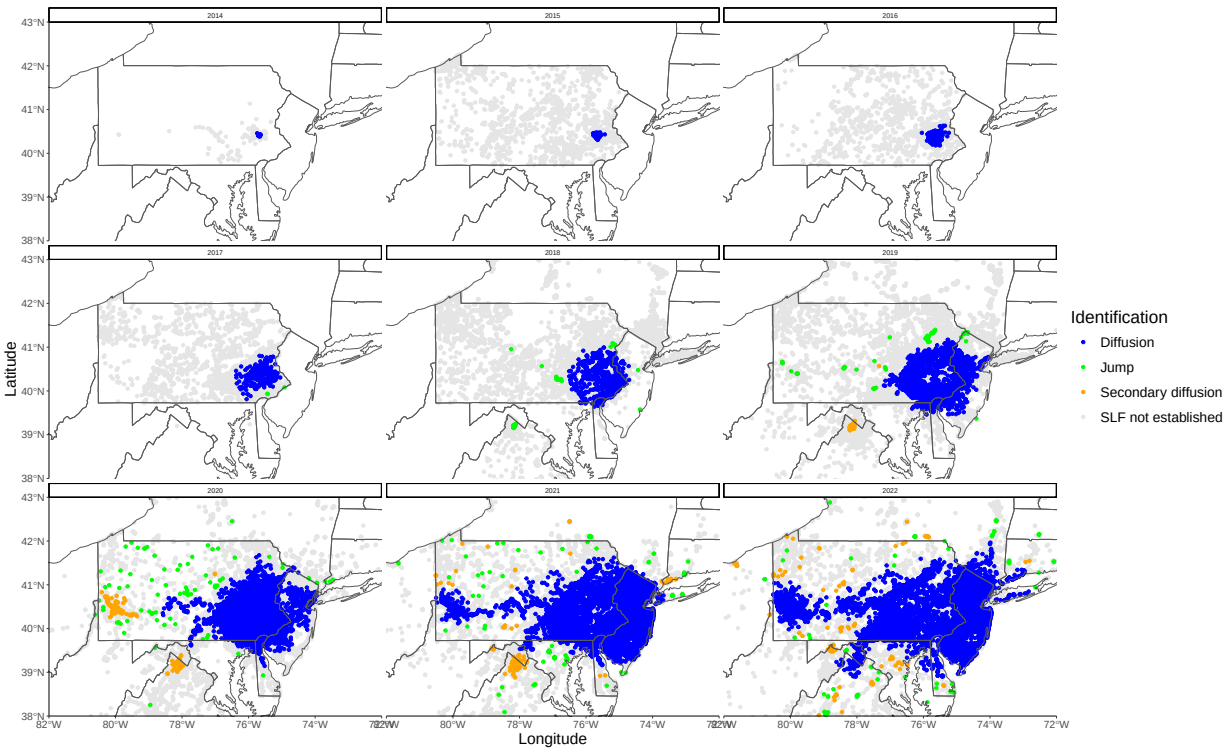



```
# save it
ggsave(file.path(here::here(), "figures", "S1B. zoomed_points.jpg"),
       zoomedMap, width = 6, height = 6)
```

Figure S2: Visualizing Jumps, Thresholds, and SecDiff

Visualize all results, faceted by year

```r
# Make a single object for the map
jumps_wrapper_map <- dplyr::bind_rows(grid_data %>% filter(established == F) %>%
                                        dplyr::mutate(Type = "SLF not established"),
                                      diffusion %>%
                                        dplyr::mutate(Type = "Diffusion"),
                        Jumps %>% dplyr::mutate(Type = "Jump"),
                        secDiffusion %>% dplyr::mutate(Type = "Secondary diffusion"))

facetedResults <- ggplot(data = states) +
  geom_point(data = jumps_wrapper_map, aes(x = longitude, y = latitude, col = Type)) +
  geom_sf(data = states, alpha = 0) +
  facet_wrap(~year) +
  theme(legend.position = "bottom") +
  theme_classic() +
  scale_color_manual(values = c("blue", "green", "orange", "gray90")) +
  xlab("Longitude") + ylab("Latitude") + labs(col = "Identification") +
  coord_sf(xlim = c(-82, -72), ylim = c(38, 43), expand = FALSE) +
  theme(legend.position = "right", text = element_text(size = 10),
      panel.background = element_rect(fill = "white"),
      legend.key = element_rect(fill = "white"),
      legend.title = element_text(size = 20),
      legend.text = element_text(size = 16),
      legend.key.size = unit(2, "lines"),
      axis.title = element_text(size = 18),
      axis.text = element_text(size = 12))

facetedResults
```

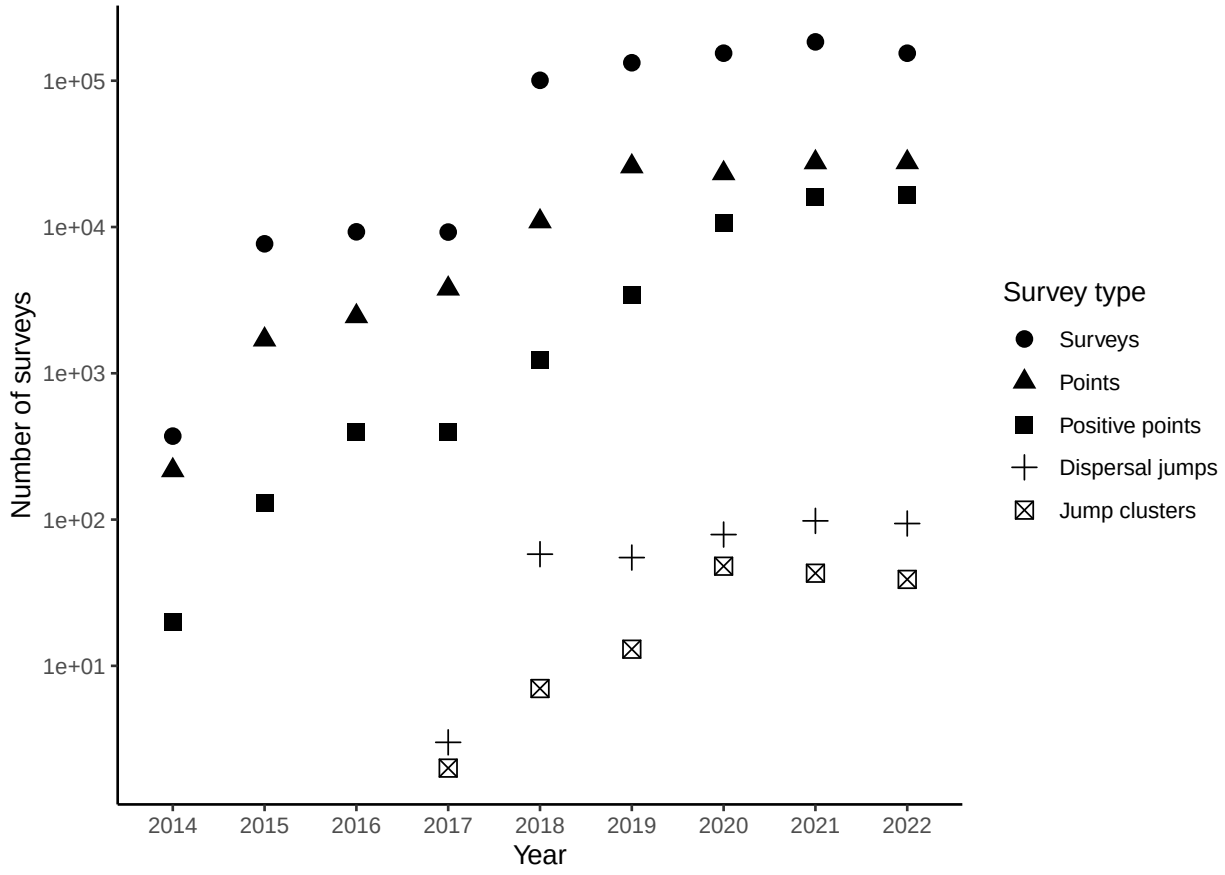


```
ggsave(file.path(here::here(), "figures", "S2. faceted_results.jpeg"),
       facetedResults, height = 12, width = 20)
```

Figure S3: Sampling effort

Show the evolution of the sampling effort and jump occurrences over time

```
slf %<>% filter(!is.na(lyde_established))

surveys <- as.data.frame(table(slf$bio_year)) %>%
  mutate(Type = "Surveys") %>%
  rename(year = Var1, n = Freq)

points <- as.data.frame(table(grid_data$year)) %>%
  mutate(Type = "Points") %>%
  rename(year = Var1, n = Freq)

positives <- grid_data %>% filter(established == T)
positive_points <- as.data.frame(table(positives$year)) %>%
  mutate(Type = "Positive points") %>%
  rename(year = Var1, n = Freq)

jumps <- Jumps %>%
  count(year) %>%
  mutate(Type = "Dispersal jumps")

clusters <- Jump_clusters %>%
  count(year) %>%
  mutate(Type = "Jump clusters")

effort <- rbind(surveys, points, positive_points, jumps, clusters)
effort$Type <- factor(effort$Type,
                      levels = c("Surveys", "Points", "Positive points",
                                 "Dispersal jumps", "Jump clusters"))

effort_plot <- ggplot() +
  geom_point(data = effort, aes(x = year, y = n, shape = Type),
             size = 3) +
  theme_classic() +
  scale_y_log10() +
  xlab("Year") + ylab("Number of surveys") +
  guides(shape = guide_legend("Survey type"))

effort_plot
```




```
ggsave(file.path(here::here(), "figures", "S3. number of surveys.jpg"),
       effort_plot, width = 7, height = 5)
```

– end of vignette –