Residence Index

Updated: 2019-02-12

Contents

This residence index tool will take a detection event dataframe and caculate the residency index for each station/receiver in the detections. The information passed to the function is what is used to calculate the residence index, make sure you are only passing the data you want taken into consideration for the residence index (i.e. species, stations, tags, etc.).

0.1 Importing Libraries

We will import `glatos` and `plotly` to run then visualize the RI.

```
library(glatos)
library(plotly)
```

0.2 Importing and Compressing Data

We will import the sample data below using `glatos::read_glatos_detections()`. We will then compress the detections into detection events with `glatos::detection_events()`, using `station` as our location column.

```
det_file <- system.file("extdata", "walleye_detections.csv",
                        package = "glatos")

detections <- read_glatos_detections(det_file)
detection_events <- glatos::detection_events(detections, location_col = 'station')
```

0.3 Kessel Method

The Kessel method converts both the startdate and enddate columns into a date with no hours, minutes, or seconds. Next it creates a list of the unique days where a detection was seen. The size of the list is returned as the total number of days as an integer. This calculation is used to determine the total number of distinct days (T) and the total number of distinct days per station (S). Possible rounding error may occur as a detection on `2016-01-01 23:59:59` and a detection on `2016-01-02 00:00:01` would be counted as two days when it is really 2-3 seconds.

$RI = \frac{S}{T}$

$RI = ResidenceIndex$

$S =$ Distinct number of days detected at the station

$T =$ Distinct number of days detected anywhere on the array

```
ri <- glatos::residence_index(detection_events,calculation_method = 'kessel')
```

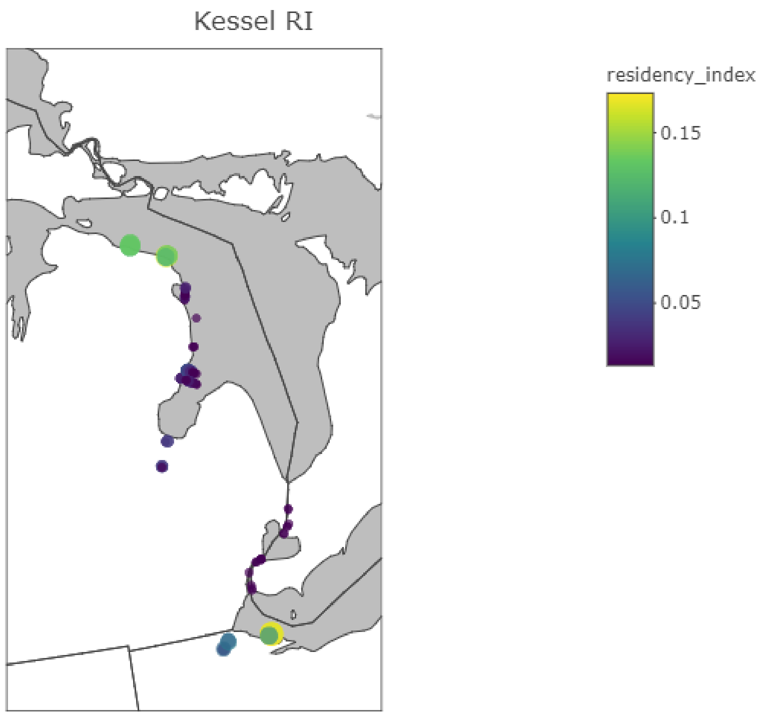
0.3.1 Plotting with Plotly

Below is the code for plotting the RI using `plotly`. `plotly` allows us to interact with the map rather than haveing a static image. More about `plotly` can be found here.

```r
geo <- list(
    scope = 'north america',
  showland = TRUE,
  landcolor = toRGB("white"),
  showocean = TRUE,
  oceancolor = toRGB("gray"),
  showcountries = TRUE,
  showlakes = TRUE,
  lakecolor = plotly::toRGB("gray"),

  resolution = 50,
  center = list(lat = median(ri$mean_latitude),
                lon = median(ri$mean_longitude)),
  lonaxis = list(range=c(min(ri$mean_longitude)-1, max(ri$mean_longitude)+1)),
  lataxis = list(range=c(min(ri$mean_latitude)-1, max(ri$mean_latitude)+1))
)


map <- ri %>%
  plot_geo(lat = ~mean_latitude, lon = ~mean_longitude, color = ~residency_index )%>%
  add_markers(
            text = ~paste(location, ': ', residency_index),
            hoverinfo = "text",
    size = ~c(residency_index * 5)
  )%>%
  layout(title = "Kessel RI",geo = geo)
```

To show the map you can just type out the variable name.

```r
map
```


Kessel RI
residency_index
0.15
0.1
0.05

0.4 Timedelta Method

The Timedelta calculation method determines the first startdate of all detections and the last enddate of all detections. The time difference is then taken as the values to be used in calculating the residence index. The timedelta for each station is divided by the timedelta of the array to determine the residence index.

$RI = \frac{\Delta S}{\Delta T}$

RI = Residence Index

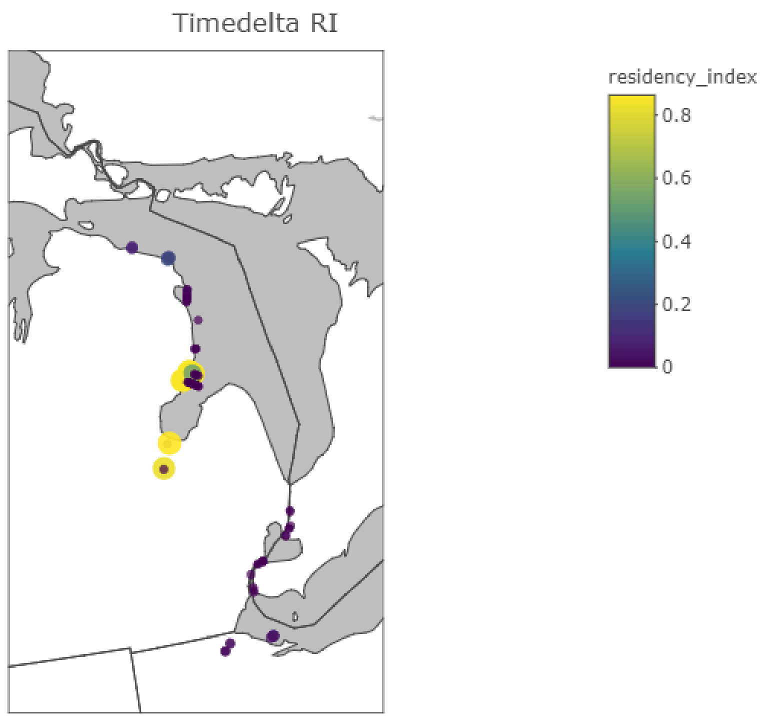
ΔS = Last detection time at a station - First detection time at the station

ΔT = Last detection time on an array - First detection time on the array

```
ri <- glatos::residence_index(detection_events,calculation_method = 'timedelta')

geo <- list(
    scope = 'north america',
  showland = TRUE,
  landcolor = toRGB("white"),
  showocean = TRUE,
  oceancolor = toRGB("gray"),
  showcountries = TRUE,
  showlakes = TRUE,
  lakecolor = plotly::toRGB("gray"),

  resolution = 50,
  center = list(lat = median(ri$mean_latitude),
                lon = median(ri$mean_longitude)),
  lonaxis = list(range=c(min(ri$mean_longitude)-1, max(ri$mean_longitude)+1)),
  lataxis = list(range=c(min(ri$mean_latitude)-1, max(ri$mean_latitude)+1))
)


map <- ri %>%
  plot_geo(lat = ~mean_latitude, lon = ~mean_longitude, color = ~residency_index )%>%
  add_markers(
            text = ~paste(location, ': ', residency_index),
            hoverinfo = "text",
    size = ~c(residency_index * 5)
  )%>%
  layout(title = "Timedelta RI",geo = geo)
```

To show the map you can just type out the variable name.

```
map
```


Timedelta RI
residency_index
0.8
0.6
0.4
0.2
0

0.5 Aggregate With Overlap Method

The Aggregate With Overlap calculation method takes the length of time of each detection and sums them together. A total is returned. The sum for each station is then divided by the sum of the array to determine the residence index.

$RI = \frac{AwOS}{AwOT}$

RI = Residence Index

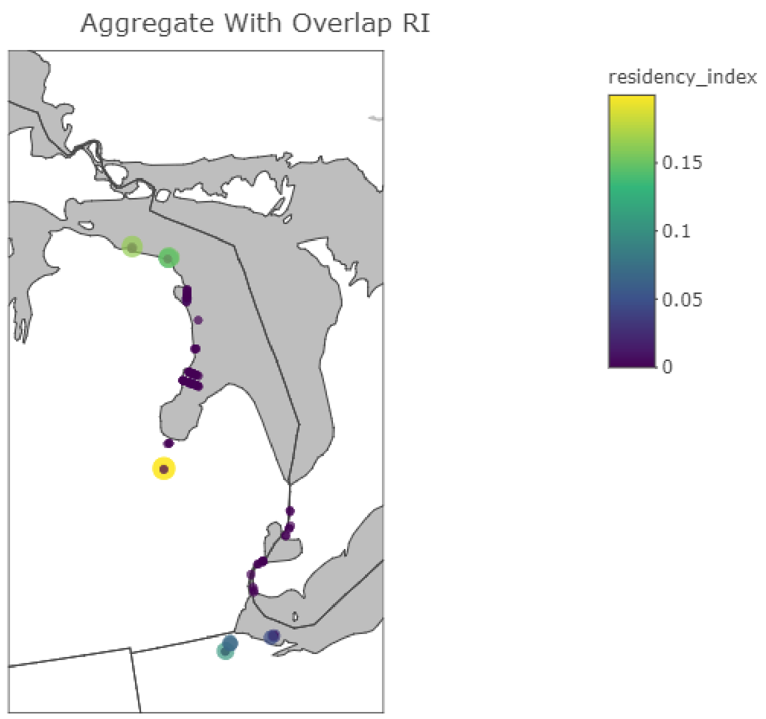
$AwOS$ = Sum of length of time of each detection at the station

$AwOT$ = Sum of length of time of each detection on the array

```
ri <- glatos::residence_index(detection_events,calculation_method = 'aggregate_with_overlap')

geo <- list(
    scope = 'north america',
  showland = TRUE,
  landcolor = toRGB("white"),
  showocean = TRUE,
  oceancolor = toRGB("gray"),
  showcountries = TRUE,
  showlakes = TRUE,
  lakecolor = plotly::toRGB("gray"),

  resolution = 50,
  center = list(lat = median(ri$mean_latitude),
                lon = median(ri$mean_longitude)),
  lonaxis = list(range=c(min(ri$mean_longitude)-1, max(ri$mean_longitude)+1)),
  lataxis = list(range=c(min(ri$mean_latitude)-1, max(ri$mean_latitude)+1))
)


map <- ri %>%
  plot_geo(lat = ~mean_latitude, lon = ~mean_longitude, color = ~residency_index )%>%
  add_markers(
            text = ~paste(location, ': ', residency_index),
            hoverinfo = "text",
    size = ~c(residency_index * 5)
  )%>%
  layout(title = "Aggregate With Overlap RI",geo = geo)
```

To show the map you can just type out the variable name.

```
map
```


Aggregate With Overlap RI
residency_index
0.15
0.1
0.05
0

0.6 Aggregate No Overlap Method

The Aggregate No Overlap calculation method takes the length of time of each detection and sums them together. However, any overlap in time between one or more detections is excluded from the sum. For example, if the first detection is from `2016-01-01 01:02:43` to `2016-01-01 01:10:12` and the second detection is from `2016-01-01 01:09:01` to `2016-01-01 01:12:43`, then the sume of those two detections would be 10 minutes.A total is returned once all detections of been added without overlap. The sum for each station is then divided by the sum of the array to determine the residence index.

$RI = \frac{AnOS}{AnOT}$

RI = Residence Index

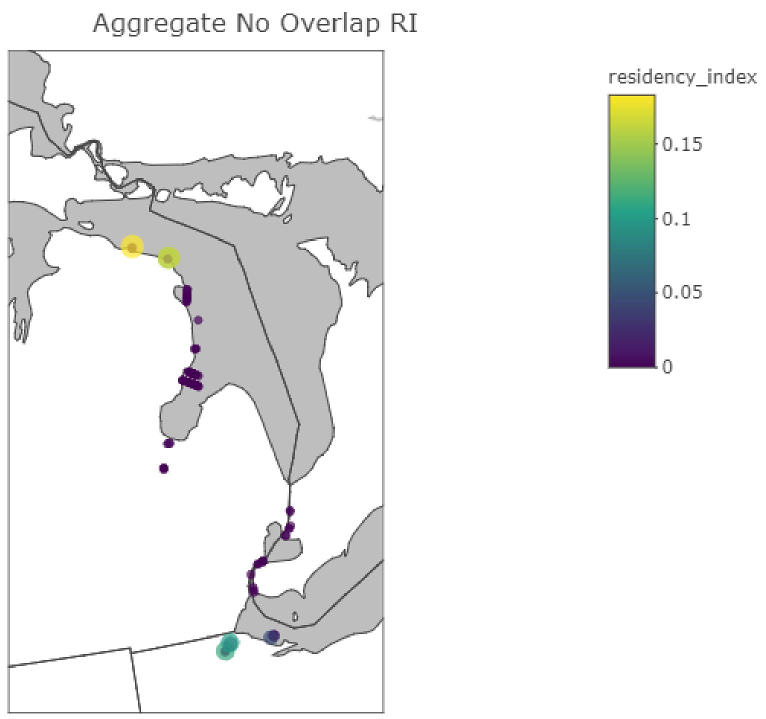
$AnOS$ = Sum of length of time of each detection at the station, excluding any overlap

$AnOT$ = Sum of length of time of each detection on the array, excluding any overlap

```r
ri <- glatos::residence_index(detection_events,calculation_method = 'aggregate_no_overlap')

geo <- list(
  scope = 'north america',
  showland = TRUE,
  landcolor = toRGB("white"),
  showocean = TRUE,
  oceancolor = toRGB("gray"),
  showcountries = TRUE,
  showlakes = TRUE,
  lakecolor = plotly::toRGB("gray"),

  resolution = 50,
  center = list(lat = median(ri$mean_latitude),
                lon = median(ri$mean_longitude)),
  lonaxis = list(range=c(min(ri$mean_longitude)-1, max(ri$mean_longitude)+1)),
  lataxis = list(range=c(min(ri$mean_latitude)-1, max(ri$mean_latitude)+1))
)


map <- ri %>%
  plot_geo(lat = ~mean_latitude, lon = ~mean_longitude, color = ~residency_index )%>%
  add_markers(
          text = ~paste(location, ': ', residency_index),
          hoverinfo = "text",
    size = ~c(residency_index * 5)
  )%>%
  layout(title = "Aggregate No Overlap RI",geo = geo)
```

To show the map you can just type out the variable name.

```r
map
```


Aggregate No Overlap RI
residency_index
0.15
0.1
0.05
0