Visualizing fish movements with animated videos in R: The ggplot2 and gganimate packages

Todd Hayden, Christopher Holbrook, Tom Binder, Mike Lowe

2019-02-08

Introduction

The `glatos` package includes several functions that facilitate the creation of animated videos of animal movements from acoustic telemetry data. The primary functions for building animated videos are the `interpolate_path` and `make_frames` functions. `interpolate_path` calculates interpolated positions of animals between detections and `make_frames` creates a series of snapshots of interpolated and detected fish locations over time from the output of `interpolate_path`. In addition to creating and outputting frames, `make_frames` can also stitch frames together to create an animated video of movements using the open source FFmpeg software. Although `make_frames` was developed to allow customization of the output content and layout of frames, some may find the level of customization to be insufficient for their needs. In this vignette, we present a short example of an alternative approach for creating animations using the `ggplot2` and `gganimate` R packages. [1] `ggplot2` is a popular and powerful package for plotting figures based on The Grammer of Graphics approach to visualizing data and developed for R by Hadley Wickham. [2] `ggplot2` differs from other plotting functions in R by its layered, interactive approach to building graphical output. Variables in data sets are mapped to aesthetic categories used to describe and build the plot. This approach differs from other packages in R that require the structure and layout of the plot to generally be specified simultaneously. The `gganimate` package is an extension to `ggplot2` that provides support for animations [3]. In the following code, we use the `make_transition` function on an example dataset included in `glatos` to create a custom transition layer necessary for interpolating fish movements. We then use the `interpolate_path` function in `glatos` to interpolate fish positions in space and time [4].Finally, we end the vignette with a demonstration of code to create an animation of fish movements using `ggplot2` and `gganimate`.

In the first chunk we load necessary packages, obtain example data from `glatos`, clean up the data, and extract and create a figure of lakes Huron and Erie (Figure 1). The `data.table` R package is used for data manipulations in this vignette. [5]

```
library(glatos)
library(sp)
library(raster)
library(gganimate)
```

[1] For more information about obtaining `gganmiate` and `ggplot2`, see https://gganimate.com/ and https://ggplot2.tidyverse.org/

[2] Leland Wilkinson,https://www.amazon.com/Grammar-Graphics-Statistics-Computing/dp/0387245448

[3] see https://gganimate.com/articles/gganimate.html

[4] See `glatos` package help for more information about functional arguments and outputs of `make_frames` and `interpolate_path`. Additional details about creating animated videos of fish movements using the `glatos` package is found in the workshop manual "A guide to R for analyzing Acoustic Telemetry Data V2.0". A pdf of manual can be downloaded here https://gitlab.oceantrack.org/GreatLakes/glatos/wikis/Past%20r%20workshops%20and%20manuals

[5] Data.table information: https://github.com/Rdatatable/data.table/wiki

```
library(ggplot2)
library(data.table)

# get polygon of the Great Lakes
data(greatLakesPoly)

# crop polygon to extract Lake Huron and Lake
# Erie
lakes <- crop(greatLakesPoly, extent(-84.9, -78.7,
    41.3, 46.6))
plot(lakes, col = "grey")

# get example walleye detection data
det_file <- system.file("extdata", "walleye_detections.csv",
    package = "glatos")
det <- read_glatos_detections(det_file)

# get example receiver file
rec_file <- system.file("extdata", "sample_receivers.csv",
    package = "glatos")
recs <- read_glatos_receivers(rec_file)
```

Figure 1: Lakes Huron and Erie extracted from Great Lakes polygon included in the 'glatos' package

In the next code chunk, we extract the geographical coordinates for the GLATOS acoustic receiver array and convert them to a spatial object using the `sp` packages.

```
# convert to data.table
setDT(recs)
rec_coords <- recs[, c("deploy_lat", "deploy_long")]

# convert to spatial object
coordinates(rec_coords) <- ~deploy_long + deploy_lat
proj <- "+proj=longlat +datum=WGS84 +no_defs +ellps=WGS84 +towgs84=0,0,0"
proj4string(rec_coords) <- CRS(proj)
```

In the next line of code, we create a transition layer from `lakes` object created previously. The transition layer is needed to calculate non-linear interpolated movement paths that avoid impossible overland movements of fish. The `rec_coords` object provided to `receiver_points` argument in `make_transition` allows small errors in receiver locations that result in receivers falling outside of the polygon object to be included in the transition layer. See `glatos` package help for more information.

```
# make transition layer.
tran <- make_transition(lakes, res = c(0.01, 0.01),
    receiver_points = rec_coords)
```

```
# create quick plot of fish and receivers to
# make sure they are 'on the map'
unique_fish <- unique(det, by = c("deploy_lat",
    "deploy_long"))

plot(tran$rast)
lines(lakes)
points(rec_coords, pch = 20, col = "orange")
points(unique_fish$deploy_long, unique_fish$deploy_lat,
    pch = 20, col = "red")

# interpolate detections.
pos2 <- interpolate_path(det, trans = tran$transition,
    lnl_thresh = 0.99)

## Calculating least-cost (non-linear) distances... (step 1 of 3)

## Starting linear interpolation... (step 2 of 3)

##
## Starting non-linear interpolation... (step 3 of 3)

##
## Finalizing results.

# convert pos2 to data.table
setDT(pos2)

# extract unique bin_timestamp
int <- unique(pos2, by = "bin_timestamp")
```

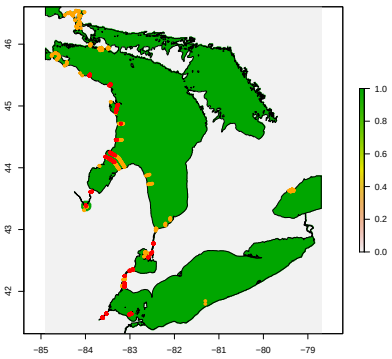

Figure 2: Plot of transition layer, receivers (orange circles), fish detection location (red circles) in lakes Huron and Erie.

In the next code chunk, we use `data.table` to add the `bin_timestamp` output in the interpolated positions (`pos2` object) to the receiver object based on receiver deployment and recovery timestamps. This is so we can display receiver deployment and recovery in the animation. We essentially need to know what receivers were deployed during each `bin_timestamp` interval. We do this with a `data.table` join.

```
# Add bin_timestamp to receivers, classifying
# based on deploy/recover timestamps and
# removes unnecessary columns in output
recs <- recs[int, .(deploy_lat = x.deploy_lat,
    deploy_long = x.deploy_long, station = x.station,
    deploy_date_time = x.deploy_date_time, recover_date_time = x.recover_date_time,
    bin_timestamp = i.bin_timestamp), on = .(deploy_date_time <=
    bin_timestamp, recover_date_time >= bin_timestamp),
    allow.cartesian = TRUE]
```

Next, we are ready to create the animation. The first step is to create a static image of all receivers, detections, and interpolated detections with `ggplot2`. We changed the color of fish detections to red circles, interpolated positions to blue circles and made detections larger than interpolated detections.

```
ggplot(data = lakes, aes(x = long, y = lat, group = group)) +
    geom_polygon(color = "black", fill = "paleturquoise") +
    geom_point(data = recs, aes(x = deploy_long,
        y = deploy_lat), size = 2, color = "orange",
        inherit.aes = FALSE) + geom_point(data = pos2,
    aes(x = longitude, y = latitude, group = animal_id,
        color = record_type, size = record_type),
    inherit.aes = FALSE) + coord_map("polyconic") +
    xlab("Longitude") + ylab("Latitude") + scale_color_manual(values = c("red",
    "blue")) + scale_size_manual(values = c(2,
    1))
```

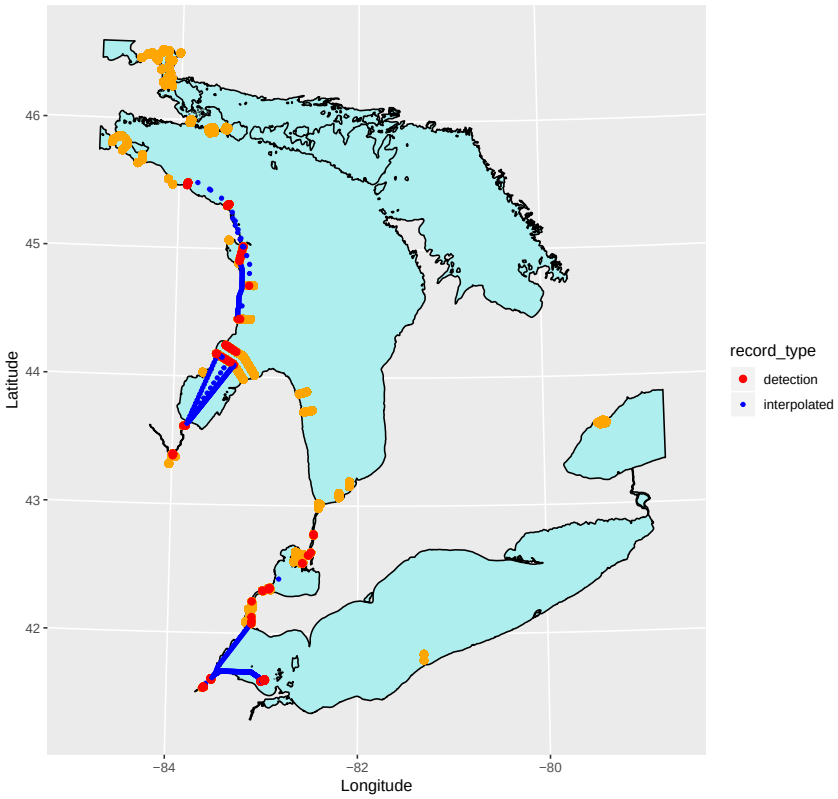

Figure 3: Static figure of fish detections (red), interpolated positions (blue), and receivers (orange) plotted using 'ggplot2'

Now we will animate the same plot using `gganimate`

```
ggplot(data = lakes, aes(x = long, y = lat, group = group)) +
    geom_polygon(color = "black", fill = "paleturquoise") +
```

```
geom_point(data = recs, aes(x = deploy_long,
    y = deploy_lat), size = 2, color = "orange",
    inherit.aes = FALSE) + geom_point(data = pos2,
aes(x = longitude, y = latitude, group = animal_id,
    color = record_type, size = record_type),
inherit.aes = FALSE) + coord_map("polyconic") +
xlab("Longitude") + ylab("Latitude") + scale_color_manual(values = c("red",
"blue")) + scale_size_manual(values = c(2,
1)) + transition_time(bin_timestamp) + ggtitle("{frame_time}")
```

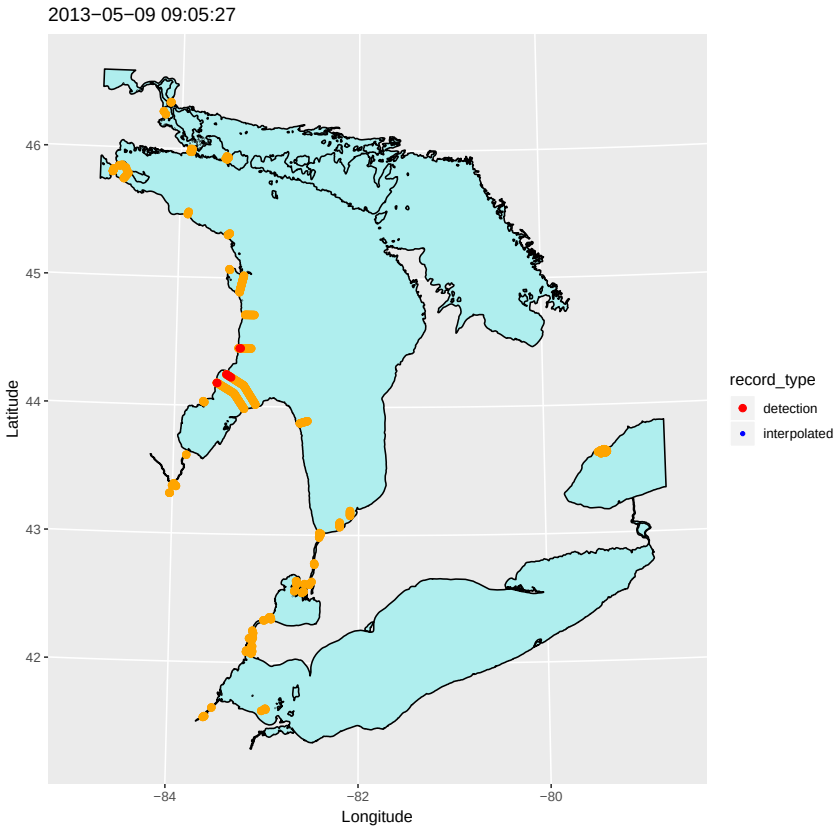

Figure 4: Animated figure of fish movements in lakes Huron and Erie. Only first image of animation is included in this vignette

This vignette provides a basic, high-level summary of creating an animated video with `ggplot2` and `gganimate`. These packages provides much more functionality than included in this vignette. Please refer to documentation for `gganimate` and `ggplot2` for additional examples and customization that may be useful for visualizing fish movements.