WIAS Fundaments of Animal Tracking

Day 2: Automated radio telemetry practical

2023-02-22

Getting started with R

Open R studio. We will start by downloading the R code and data files to process the detections from GitHub. To do so, go to File > New Project > Version Control > Git and in the Repository URL box, paste: https://github.com/cwtyson/WIAS_tracking_course.git

Choose a subdirectory (folder) where you would like to save these files. Then select 'Open in new session' and lastly click 'Create Project'

You should now have 'cloned' this repository from GitHub to your computer. Navigate to the folder where you chose to save the files. You should see a new folder 'WIAS_tracking_course' and within this folder are several files. Browse through the folders to orient yourself. We will fill in some of these files during the first part of the practical and use the R scripts in the second part.

Data entry

Before we start processing the data, we need to enter our field notes. We also need to copy some files to our computers. These steps can be done in any order.

- Copy your group's test data into the 'group_tests_log.xlsx' file. Be sure to write the 'point' name exactly as you entered it in your GPS. For 'time_start' and 'time_end' record the time in a 24-hour format without any space or punctuation between the hours and minutes, e.g. 0921. For the description, the formatting is up to you, but don't use too many words (fewer than 6).
- Copy the test point coordinates from your GPS to your computer. Save these coordinates in the folder 'WIAS_tracking_course/field_data/group_tests' with the name 'gps_points.GPX'. You will also get a copy of the coordinates of the nodes. Save this file ('grid_points.GPX') in the 'field_data/grid_points' folder.
- Get a copy of all the detections from the sensor station. Place this folder in 'field_data/detections' and be sure the folder is called 'sensor_station_detections'.

When you have finished these steps, open the R script file 'aggregate_detections.R' and run the code chunk shown below to install the packages that we will need for today. Once this is done, wait for the other groups to finish.

```
## Houskeeping ########
packages_needed <- c("tidyverse","vroom","lubridate","sf", "data.table", "readxl",
                     "geosphere")
new_packages <- packages_needed[!(packages_needed %in%
                                    installed.packages()[,"Package"])]
```

```
if(length(new_packages)){
  install.packages(new_packages, repos = "https://cloud.r-project.org")
}

# Now load all packages
library(tidyverse); library(lubridate); library(vroom); library(data.table)
library(readxl); library(geosphere)
```

Processing detections

Our next step is to get the detections from the sensor station into a more suitable format. Start by running the following chunk.

```
## Detections from Sensor Station
dets_ss_raw <- vroom::vroom(list.files(here::here(),
                                       recursive = TRUE,
                                       full.names = T,
                                       pattern = "raw"))
```

Then proceed through the rest of the code, checking out the structure of the data, taking note of the number of detections that are retained at each step. Ultimately, save the filtered detections and then open up 'model_rss_dst.R' and proceed.

Modeling the RSS~distance relationship

We will now use the detections we collected at set distances from the receivers to make a model of how received signal strength (RSS) changes with distance, which we will then use to predict the distance of new RSS values from the different receivers. Start by checking that the 'rssi_distance_calibration_log.xlsx' file is filled in and then go through the code, checking the objects that get created. Try to understand what the various chunks of code are accomplishing. In particular, when you come to the chunk below, interpret the plot that you see. Explore whether there are differences in the relationship between RSS and distance for different nodes and tags.

```
## Plot predicted values
ggplot(model_data,
       aes(x = fit,
           y = rssi)) +

  ## Modeled data
  geom_point() +
  geom_ribbon(aes(xmin = lwr,
                  xmax = upr,
                  color = NULL),
              alpha = .15) +

  ## Actual data
  geom_point(aes(x = distance,
                 y = rssi),
             color = "red") +

  theme_minimal() +
  labs(x = "Distance", y = "RSS")
```

Finish running through the end of the script to save the model. When you are done, open the R script 'filter_detections.R' and proceed to the next section.

Filter detections

This next script is one of the longer data processing scripts as it does the majority of the work to get the detections from your group's tag(s) that occurred during your group's test intervals. Step through the code in chunks and try to understand what each accomplishes. The following chunk will summarize the filtered detections for each test point (and tag if you used multiple) and give the number of detections, the mean RSS value, and the standard deviation of the RSS value.

```
## Summarize detections: Number of detections, mean RSS value,
## and SD RSSI value for each tag and each point
det_sum <- field_cal_dets %>%
  group_by(point,description,tag, grid_point) %>%
  summarise(num_detections = n(),
            mean_rssi = mean(rssi),
            sd_rssi = sd(rssi))

View(det_sum)
```

Look at the summary table. Do you see any patterns? Make some plots to further explore any differences between the points.

Finally, finish the script by saving the output, then open the file 'process_detections.R' and proceed to the next section.

Process detections

This next short script finishes processing the filtered detections so that we can proceed with localizing the tag signals using multilateration. For this, we need to get the average signal strength registered by a tag by each receiver during each test interval. This is done with the following chunk. The last line of this chunk counts the number of unique nodes that detected the tag during the test interval. Ultimately, we use this information to remove any test intervals that did not have detections by at least 3 nodes since it would not be possible to use multilateration for these cases.

```
## Prepare filtered detections
prepared_dets <- filtered_ddets %>%
  dplyr::group_by(point,
                  description,
                  tag,
                  grid_point) %>%
  dplyr::summarise(mean_rssi = mean(rssi)) %>%
  dplyr::group_by(point,
                  description,
                  tag) %>%
  dplyr::mutate(number_grid_point = n())
```

How many rows are in the final data frame named 'multilateration_dets'? If at least 3 nodes detected your group's tag(s) at each test point than you should have as many rows as test points multiplied by the number of tags your group used.

After this, we use the model we made for distance as a function of RSS to predict the distance of each of the observed mean RSS values from each node at each test point. You can check out the different distances with a histogram.

Finish running the code in this script to save the processed detections. When you are done, open the script 'localize_detections.R' and proceed with the next section.

Localize detections

Finally, we are ready to localize the detections from our test points. In this script, we will use multilateration to estimate the locations of the tag at each test point based on the mean RSS registered from each receiver. This script takes the processed detections we just finished creating and then uses a for-loop to iteratively subset the data by each test point (and tag if multiple tags were used) and use nonlinear least squares test to the estimate the location of the tag given the locations of the receivers that detected the node and the estimated distance between the tag and each receiver.

The results from this model gives the estimated location of the tag, which can then be compared to the known location of the tag to estimate the error. For this script, simply highlight everything and hit run. This will go through each test point and estimate the location of the tag(s) detections at that point.

Well done! You now have location estimates for the test points. When you are finished with this section, open the R script 'analyze_locations.R' and proceed to the next section.

Analylzing and plotting localization

Now it is time to start exploring the data. Follow the provided code to make some plots of the data. Think about the questions and come up with some other plots to explore the data.