Sampling Frequency Impact on Interpreation of Osprey Movement Strategies

Lewis Hakam

3/14/2024

Load packages and functions.

```
knitr::opts_chunk$set(echo = TRUE)
library(formatR) #Rmarkdown format
library(move) #Download, manipulate, anlayze data from Movebank
library(sp) #Spatial objects in Movebank
library(moveVis) #Converting a data frame to a move object
library(plyr) #Batch calculating stats for subsets of data
library(dplyr) #Deleted and re installed rlang
library(tidyr) #Extracting character string values
library(readr) #Read in CSV files
library(stats) #Basic stats functions and tools
library(devtools) #Download non-cran packages
library(lubridate) #Ensuring date formats are correct
library(solartime) #Calculate daylight hours
find_mode <- function(x) {
  u <- unique(x)
  tab <- tabulate(match(x, u))
  u[tab == max(tab)]
} #No built in function for mode in R so I got this one from the web.
library(data.table) #Manipulating data tables
library(adehabitatLT) #Creating trajectories and calculating characteristics.
library(leaflet) #Interactive maps
library(htmltools) #Popups for interactive maps
library(ggplot2) #Graphs and maps
library(sf) #Spatial objects for leaflet and ggplot
library(tmap) #Package containing world country polygons
wgs84<-CRS("+proj=longlat +datum=WGS84") #GCS in WGS 1984
mercator = CRS("+init=epsg:4326") #this projection covers the world so no need to custom
↪  define projection attributes
azim_Bierr = CRS("+proj=aeqd +lat_0=19 +lon_0=-72 +x_0=0 +y_0=0 +ellps=WGS84 +datum=WGS84
↪  +units=m +no_defs") #PCS in custom azimuthal equidistant projection.
data(World) #World data with country polygons needed for mapping
world_sf<-st_transform(World, azim_Bierr) #Projected world data
world_filter<-world_sf[,"name"] #Reduce info for spatial join
world_filter$name<-as.character(world_filter$name) #Ensure world polygon data is an
↪  appropriate data type
plotshit<-function(y) {
  ggplot(y) +
    geom_sf(data = world_sf, fill = "#464646") +
```

```
    geom_sf(size = 1, aes(color = year)) +
    coord_sf(datum = st_crs(azim_Bierr), xlim = c(-4000000, 3000000), ylim = c(3500000,
    ↪  -2500000)) +
    theme(legend.position = "bottom", legend.text = element_text(size = 7)) +
    labs(color = "year") +
    ggtitle(paste0(y$trackId))
} #Function for batch plotting in a for loop
plotshit2<-function(y) {
  ggplot(y) +
  geom_sf(data = world_sf, fill = "#464646") +
  geom_sf(size = 1, aes(color = DaylightUTC)) +
  coord_sf(datum = st_crs(azim_Bierr), xlim = c(-4000000, 3000000), ylim = c(3500000,
  ↪  -2500000))+
  ggtitle(paste0(y$trackId))
} #Daylight assessment
library(car) # Q-Q plots for EDA
library(rstatix) #Shapiro-Wilk Normality test
library(vegan) #PERMANOVA
library(pairwiseAdonis) # PERMANOVA post-hoc multilevel comparison test
library(MicEco) # Omega-squared values
library(stringr) #Extracting string info from dataframes. Useful for manipulating text
↪  column values.
library(factoextra) #Customizing PCA biplots
library(FactoMineR) #PCA
library(corrplot) #contribution heatmaps for PCA
library(RColorBrewer) #Creating custom color palette
colors_Events<-colorRampPalette(c("#E0AA20", "#ECC713", "#F1E306", "#17E48B","#1BA554",
↪  "#268E74", "#0B68FF", "#182DD4", "#073C73", "#39056F"))(16) #custom color pallete for
↪  migration events
colors_Ind<-colorRampPalette(c("#E0AA20", "#ECC713", "#F1E306", "#17E48B","#1BA554",
↪  "#268E74", "#0B68FF", "#182DD4", "#073C73", "#39056F"))(8) #custom color pallete for
↪  individuals
library(ComplexHeatmap) #Correlation plots in R
library(circlize) #color bar for ComplexHeatmap
library(stats) #TukeyHSD - package comes with base R
library(gridExtra) #arranging grids together
library(ggpubr) #ggarrange, arranging ggplot grids together
library(rootdetectR) #Making results of TukeyHSD into a matrix
```

Introduction

Hey there! If you are viewing this, you are interested in how I analyzed osprey tracking data for my final thesis, completed in Spring 2024. I will guide through my step by step process to answer three questions:

1. **Do strategies vary between and within individuals?**
2. **Do ospreys use the same migration strategies between regions?**
3. **How does changing the temporal sampling frequency affect the results of my analysis?**

This document was created using R Markdown and will not cover the nuances I used to code the Markdown file (if you want to access the full Markdown script, you can view the file on my Github at https://github.com/Hakalar2000/OspreyMigrationStrategies). For privacy, not all

code lines are shown in the Rmarkdown output. I have done my best to annotate my code for easy interpretation, but this is not a guide on how to use R. Some aspects of coding will *not* be explained.

A Note on Installing Packages

You will likely have to use the function install.packages ("Package") before you can access packages using the library (Package) function if you are new to R. I had a few instances where I had supporting packages downloaded, but they were not up-to-date nor could I force them to update. For these packages I had to locally delete the packages that were not updating and then re-install them. Many of these packages have dependencies (they require other packages) so do not be alarmed if download takes a while. Lastly, some packages are not from CRAN, rather they are from personal repositories (repos) on github or from Bioconductor. These packages require different download code. To download packages from repos, you need to use the install_github("username/repo/package file name",) from the 'devtools' package. For Bioconductor packages you need to use the install("package name") function from the 'BioManager' package.

- CRAN: install.packages() from base R
- BioConductor: install() from BioManager
- Personal Repositories: install_github() from devtools

Downloading and Filtering Tracking Data

Setup

First, If you do not have a Movebank account, you must make a free account before you can access data. Even after you make an account, not all data are available for download. There are three levels of privacy on Movebank: downloadable (orange), visible fixes with permission required to download (blue), and no visible fixes with permission required to download (grey). More information about navigating Movebank can be found on the Movebank website: https://www.movebank.org/cms/movebank-main

After reviewing accessible data sets, I determined that I could only use is "Osprey Bierregaard North and South America". I chose this data set since it provided me with the most individuals with 1-hour or finer sampling frequencies and allowed me to control for possible regional differences among worldwide osprey populations. The dataset consists of ospreys from Newfoundland to South Carolina tracked from 1995-2019. Not all of these individuals are tracked using GPS or GPS/GSM devices with a 1-hour or finer sampling frequency.

To start, you will consult the reference data and see if you can reduce the dataset before downloading all fixes. This dataset is large, and the more data you pull, the longer it will take. Reducing the number of points before downloading will significantly increase processing speeds. To pull reference data or datasets from Movebank you will use the 'move' package. Before you can pull data with the move package you will need to create an object called loginStored which contains your login info. The code below is an example, since I do not want to give out my actual password.

Your code will look something like this:

```
loginStored <- movebankLogin(username = "UserName", password = "Password")
```

Downloading Movebank Summary Table

> Next you will download the Movebank study synopsis and summary tables. It is possible to search for studies within R, but you will save yourself some time using the Movebank website's interactive map. * The synopsis can provide details on how many individuals were tracked, the time frame tracking occurred, the type of device(s) used, demographic information (sex, age, etc.), and permission/privacy restrictions. * The summary table will provide a summary of each individual tracked per device. This means that some individuals may have multiple entries in a summary table if they were fitted with more than one transmitter. You can use the summary table info to partially fill out demographic information and determine how many individuals were tracked using GPS or GPS/GSM devices. I kept a running document of individuals with their demographic and geographic information that will come in handy later on.

```
#download Bierregaard data Synopsis
reference<-getMovebankStudy(study ="Osprey Bierregaard North and South America", login =
↪  loginStored)
reference$study_objective
```

```
## [1] "These data are available for downloading but MAY NOT BE USED in
publications without my permission. I am happy to collaborate on analyses and
publications, but I must be included early in the process. This study includes
47 adults but has focused on tagging juveniles (61 tagged) prior to their first
migration. It is the first dedicated study of juvenile dispersal, mortality,
and migration for this species. Beginning in 2012 GSM transmitters have been
deployed on adult males to study their foraging behavior around nests and fine
details of migration--thermal soaring over land and dynamic soaring over water,
for instance."
```

```
#download the summary table
Bierregaard_Study_summary<-getMovebankReferenceTable(study ="Osprey Bierregaard North and
↪  South America", login = loginStored, allAttributes = FALSE)
#Notice that each individual has up to 3 entries. Some individuals only have one entry
↪  with sensor_type_id 82789. These individuals were tracked with devices that only
↪  collected doppler shift locations. Some of these individuals may have two entries,
↪  but you can always look at tag comments which provide info on whether locations  were
↪  estimated via doppler shift or GPS.

#Explore sensor attributes and associated codes. We only want a summary of individuals
↪  with GPS or GPS/GSM tracks. We also do not need accessory or engineering data.
attributes<-getMovebankSensorsAttributes(study="Osprey Bierregaard North and South
↪  America",login=loginStored) #sensor_type_id for GPS locations that we will need is
↪  653. 7842954 is the GPS engineering data and 82798 is doppler shift data.

#Filter the summary table to keep only entries with the sensor_type_id of 653.
Bierregaard_Osprey_summary<-filter(Bierregaard_Study_summary,
                                   Bierregaard_Study_summary$sensor_type_id == 653)
#Fill in attributes of individuals on the Individuals that possibly meet the 1-hour or
↪  finer selection criteria on the Metadata Spreadsheet.
```

The summary tables give me an idea of the total number of individuals in each study and the sex and age of each individual. However, it does not tell me the number of fall migrations for each individual, whether those migrations were complete or not, nor if the individual even started migration. Age and sex data may also be missing.

Download Movebank Dataset

The dataset is really large, so I filtered the data to make it easier to work with.

Remove: * individuals with sampling frequencies courser than 1-hour (using mode and median) * GPS fixes outside of July-December (7-12) when osprey are "stationary" or on spring migrations * erroneous points that don't meet a defined biological threshold.

```
#Download data from Movebank for GPS tracked individuals.This takes some time. When
↪  downloading Movebank data, I can specify which individuals I want in my dataset using
↪  animalName = (list of animal local identifiers)
Bierregaard_Osprey<-getMovebankData(study ="Osprey Bierregaard North and South America",
                                    animalName =
                                    ↪  c(Bierregaard_Osprey_summary$animal_local_identifier),
                                    login = loginStored, removeDuplicatedTimestamps=TRUE)
                                    ↪
#over 2 million points...yikes
#Get the time between consecutive locations using the move package
#Start by calculating the time lag between points for each individual. The output is a
↪  list.
list_Bierr<-timeLag(Bierregaard_Osprey, units = "hours")
#lapply function applies a function to a list. Each individual within the data set should
↪  have a median or mode time lag of 1 hour or finer. I used median/mode instead of mean
↪  since most studies only have trackers turned on during daylight hours. This means
↪  that  mean lag values will be larger than 1 hour.
sapply(list_Bierr, mean) #You can see thant most study averages are larger than 1 point
↪  per hour .
```

```
##           Art         Artoo         Aster           Bea         Belle
##    1.70965139    1.65549283    1.77272727    1.53314237    1.77627923
##        Bergen       Blackie      Borealis       Bridger       Bridget
##    1.64552804    1.59292516    1.74058717    0.63193817    1.60944206
##         Buck         Caleb         Caley    Captain.Liz       Charlie
##    1.61932035    1.61279576    2.19307822    1.62867755    0.08922288
##       Chester          Chip         Claws         Clyde         Crabby
##    0.10919517    1.50915432    1.83764820    3.53786857    0.07481467
##         Cutch        Daphne            DJ        Donovan          Duke
##    1.48648649    1.63200216    0.09844502    1.66654736    1.97701149
##         Edwin         Felix          Flow         Gerry         Goody
##    0.13946885    1.73110721    1.70878536    0.05601594    2.16400581
##     Gundersen         Gunny       Hackett      Henrietta        Hix.Jr
##    1.70332877    1.65137711    0.09211484    1.54788732    1.63802083
##         Holly         Hudson       Icarius        Isabel          Jill
##    0.12367246    1.58909161    0.09965901    1.54060914    1.54254969
##       Jocelyn         Juliet       Katbird          Katy         Layla
##    1.51315789    1.63721232    1.50106891    1.64345193    1.70523599
##          Leif   Little.Ricky        Lizzie          Luke      Mackenzie
```

```
##     1.58746492     2.03010235     1.64179104     1.64451538     1.60889598
##         Meadow        Mittark         Moffet     Mr..Hannah          Neale
##     1.95954357     1.90243270     1.65746479     1.58027699     1.66665383
##           Nick North.Fork.Bob         Ozzie         Pearl         Peirce
##     0.06191351     1.62772391     1.66162869     1.62931203     1.66956522
##       Penelope          Quin         Rafael         Rammie          Rock
##     1.99508317     0.08383134     1.57774855     0.33973001     1.57434114
##         Rodney     Roger.Tory            Ron          Saco        Sanford
##     1.60264282     0.09009127     1.68980751     1.59868852     1.59284966
##   Shanawdithit          Snowy      Sr..Bones       Staddler          Tango
##     2.00959893     1.63027668     1.89713134     1.67962525     0.12228736
##       Thatcher         Tilton         Tommy      Trepassey         Tucker
##     1.61694141    32.71412698     1.67567568     1.65727700     1.56097561
##          Uncas       Virginia         Wausau         Weber          Whit
##     0.06700376     1.49859944     1.67839275    11.96673487     1.68906647
##          Woody
##     0.11430186
```

```
sapply(list_Bierr, find_mode) # birds with a finer than 1-hour sampling frequency were
↪  tracked using GPS/GSM transmitters.
```

```
##            Art          Artoo         Aster           Bea         Belle
##     1.00000000     1.00000000     1.00000000     1.00000000     1.00000000
##         Bergen        Blackie       Borealis        Bridger        Bridget
##     1.00000000     1.00000000     1.00000000     0.20166667     1.00000000
##           Buck          Caleb         Caley    Captain.Liz        Charlie
##     1.00000000     1.00000000     1.00000000     1.00000000     0.01666667
##        Chester           Chip         Claws         Clyde         Crabby
##     0.01666667     1.00000000     1.00000000     1.00000000     0.01666667
##          Cutch         Daphne             DJ        Donovan          Duke
##     1.00000000     1.00000000     0.01666667     1.00000000     1.00000000
##          Edwin          Felix          Flow          Gerry          Goody
##     0.01666667     1.00000000     1.00000000     0.01666667     1.00000000
##       Gundersen          Gunny        Hackett       Henrietta         Hix.Jr
##     1.00000000     1.00000000     0.01666667     1.00000000     1.00000000
##          Holly         Hudson        Icarius         Isabel           Jill
##     0.01638889     1.00000000     0.01638889     1.00000000     1.00000000
##         Jocelyn         Juliet        Katbird          Katy         Layla
##     1.00000000     1.00000000     1.00000000     1.00000000     1.00000000
##           Leif   Little.Ricky         Lizzie          Luke      Mackenzie
##     1.00000000     1.00000000     1.00000000     1.00000000     1.00000000
##         Meadow        Mittark         Moffet     Mr..Hannah          Neale
##     1.00000000     1.00000000     1.00000000     1.00000000     1.00000000
##           Nick North.Fork.Bob         Ozzie         Pearl         Peirce
##     0.01666667     1.00000000     1.00000000     1.00000000     1.00000000
##       Penelope          Quin         Rafael         Rammie          Rock
##     1.00000000     0.01666667     1.00000000     0.20166667     1.00000000
##         Rodney     Roger.Tory            Ron          Saco        Sanford
##     1.00000000     0.01666667     1.00000000     1.00000000     1.00000000
##   Shanawdithit          Snowy      Sr..Bones       Staddler          Tango
##     1.00000000     1.00000000     1.00000000     1.00000000     0.01638889
##       Thatcher         Tilton         Tommy      Trepassey         Tucker
##     1.00000000     1.00000000     1.00000000     1.00000000     1.00000000
```

```
##          Uncas       Virginia         Wausau          Weber           Whit
##     0.01666667     1.00000000     1.00000000     1.00000000     1.00000000
##          Woody
##     0.01666667
```

```
sapply(list_Bierr, median) #results are similar to mode.
```

```
##            Art          Artoo          Aster            Bea          Belle
##     1.00000000     1.00000000     1.00000000     1.00000000     1.00000000
##         Bergen        Blackie       Borealis        Bridger        Bridget
##     1.00000000     1.00000000     1.00000000     0.20388889     1.00000000
##           Buck          Caleb          Caley    Captain.Liz        Charlie
##     1.00000000     1.00000000     1.00000000     1.00000000     0.01750000
##        Chester           Chip          Claws          Clyde         Crabby
##     0.01750000     1.00000000     1.00000000     1.00000000     0.01722222
##          Cutch         Daphne             DJ        Donovan           Duke
##     1.00000000     1.00000000     0.01916667     1.00000000     1.00000000
##          Edwin          Felix           Flow          Gerry          Goody
##     0.02916667     1.00000000     1.00000000     0.01694444     1.00000000
##       Gundersen          Gunny        Hackett       Henrietta         Hix.Jr
##     1.00000000     1.00000000     0.01777778     1.00000000     1.00000000
##          Holly         Hudson        Icarius         Isabel           Jill
##     0.01861111     1.00000000     0.01888889     1.00000000     1.00000000
##         Jocelyn          Juliet        Katbird           Katy          Layla
##     1.00000000     1.00000000     1.00000000     1.00000000     1.00000000
##           Leif    Little.Ricky         Lizzie           Luke       Mackenzie
##     1.00000000     1.00000000     1.00000000     1.00000000     1.00000000
##         Meadow         Mittark         Moffet      Mr..Hannah          Neale
##     1.00000000     1.00000000     1.00000000     1.00000000     1.00000000
##           Nick North.Fork.Bob          Ozzie          Pearl         Peirce
##     0.01722222     1.00000000     1.00000000     1.00000000     1.00000000
##        Penelope           Quin         Rafael         Rammie           Rock
##     1.00000000     0.01722222     1.00000000     0.20388889     1.00000000
##         Rodney      Roger.Tory            Ron           Saco        Sanford
##     1.00000000     0.01722222     1.00000000     1.00000000     1.00000000
##    Shanawdithit          Snowy       Sr..Bones       Staddler          Tango
##     1.00000000     1.00000000     1.00000000     1.00000000     0.01972222
##        Thatcher         Tilton          Tommy       Trepassey         Tucker
##     1.00000000     1.00000000     1.00000000     1.00000000     1.00000000
##          Uncas       Virginia         Wausau          Weber           Whit
##     0.01694444     1.00000000     1.00000000     1.00000000     1.00000000
##          Woody
##     0.03083333
```

```
#caculate turning angle
Bierregaard_Osprey <- spTransform(Bierregaard_Osprey, CRSobj=mercator)
Bierregaard_Osprey$rel_angle<-unlist(lapply(turnAngleGc(Bierregaard_Osprey),function(x)
↪  c(NA, x, NA)))

#make into a data frame to easily filter, plot, and manipulate the data.
Bierregaard_Osprey_DF<-as.data.frame(Bierregaard_Osprey)
#Filter out non GPS location entries
Bierregaard_Osprey_DF_GPS<-Bierregaard_Osprey_DF %>%
```

```
  filter(sensor_type_id == 653) #reduces dataset by around 50,000 points.
#Extract month data for filtering
Bierregaard_Osprey_DF_GPS$month<-month(ymd_hms(Bierregaard_Osprey_DF_GPS$timestamp))
#Extract year data for plotting later
Bierregaard_Osprey_DF_GPS$year<-year(ymd_hms(Bierregaard_Osprey_DF_GPS$timestamp))
length(unique(Bierregaard_Osprey_DF_GPS$trackId)) #86 individuals
```

```
## [1] 86
```

Remove outliers: filter erroneous points using quantiative methods. Erroneous points are those with unrealistically high speed and turning angle (course). For course (often referred to and confused with heading) we will look at the 90th percentile of points as suggested by Gupte et al., 2021. Course is measured as the angle from magnetic north (clockwise) in which the tracker is moving. For speed I used Kerlinger 1989 maximum recorded flight speed for ospreys which is 33.4 meters/second.

```
quant<-quantile(Bierregaard_Osprey_DF_GPS$rel_angle, probs = 0.9, na.rm = T)
Bierregaard_Osprey_DF_Filter<-filter(Bierregaard_Osprey_DF_GPS, ground_speed <= 33.4 &
↪  heading < quant)
#still over 2 million points...
#Filter data further by removing timestamps from January through June (keep months 7-12)
Bierregaard_Osprey_Data <- Bierregaard_Osprey_DF_Filter %>%
  filter(month %in% c(7:12)) #Down to 1 million points!
#Create migration event IDs. Later I will create segment IDs.
Bierregaard_Osprey_Data$migrationEvent<-paste(Bierregaard_Osprey_Data$trackId,
↪  Bierregaard_Osprey_Data$year)
length(unique(Bierregaard_Osprey_Data$migrationEvent)) #145 migration events between 86
↪  individuals.
```

```
## [1] 145
```

```
#Format the migration event names so that they don't have any spaces or special
↪  characters
Bierregaard_Osprey_Data$migrationEvent<-gsub("\\.", "_",
↪  Bierregaard_Osprey_Data$migrationEvent) #remove periods
Bierregaard_Osprey_Data$migrationEvent<-gsub(" ", "_",
↪  Bierregaard_Osprey_Data$migrationEvent) #remove spaces
```

Mapping and Filtering Data

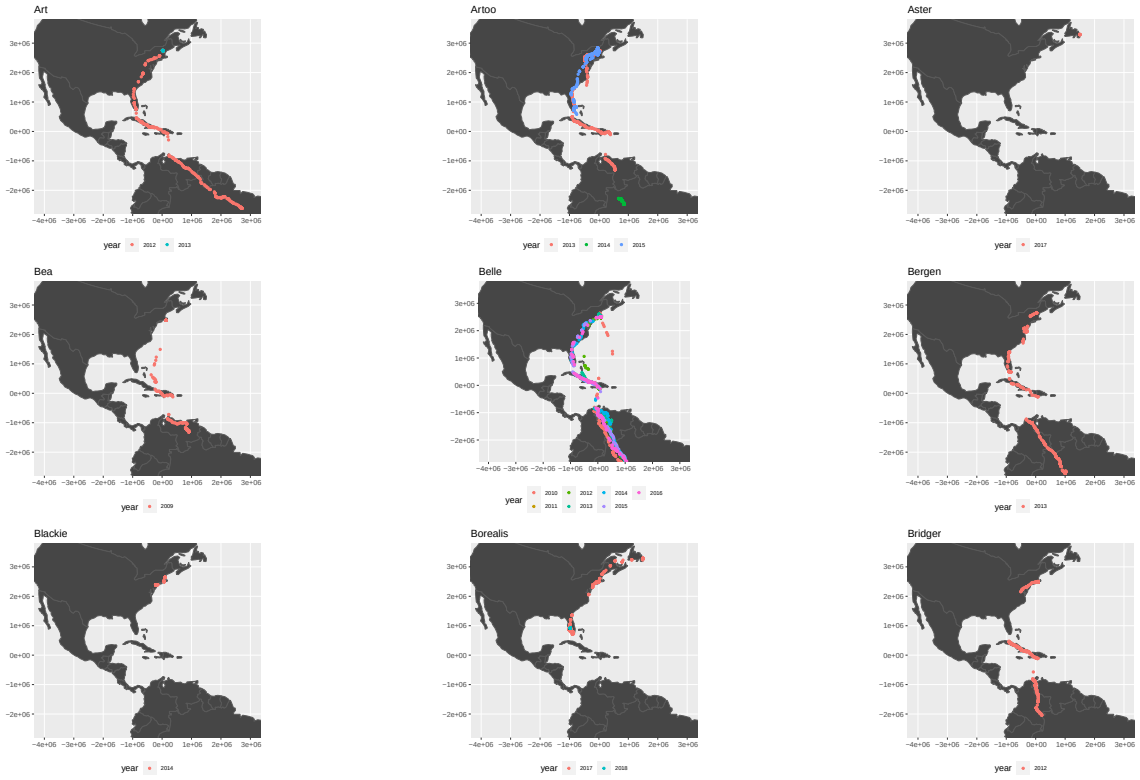
First, I will quickly map the data for each individual by year to determine if I should remove any individuals that obviously did not complete at least two migration segments between all tracking years. Ospreys typically have three migration segments defined as follows: * travel from and across the US mainland. Exceptions are for those ospreys that started by travelling over the Atlantic. The ending point of the first segment is the point closest to the border between the first and second segment. * Across the Caribbean starting from the ending point of the first second to the point closest tothe border between the second and third segemnt. * Across central/South America until the final migration point. There are a few exceptions to the three segments as some osprey wintered in Florida or Cuba.

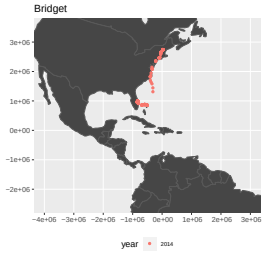
I will use 'ggplot' and 'leaflet', which both rely on 'sf' spatial objects for mapping purposes. I will map each individual bird, with tracks color-coded by year so I know if the bird has more than one fall migration that I can use. This requires the custom 'plotshit' function and a for loop. Data on the background world countries is from the 'tmap'. Data is projected using a custom aziumthal equidistant projection. The actual projection used to map data is not all that important at the moment since we are not measuring anything, however the projection will be important later on. The projection was defined by using google maps to estimate the center of all of my data.

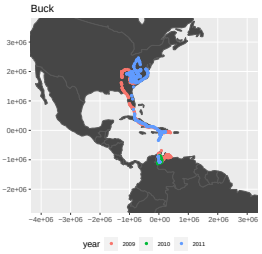
Characteristics of individuals I am looking for:

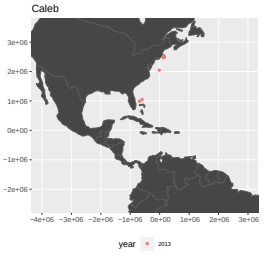
1. At least two complete adult migrations
2. Wintering in South/Central America with migration time longer than 3 days in each region
3. Fixes do not look erroneous

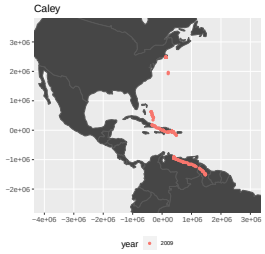
```
#Spatially project the Bierregaard data
Bierregaard_Osprey_Data_sf<-st_as_sf(x=Bierregaard_Osprey_Data,
                                     coords = c("location_long.1", "location_lat.1"), crs
                                     ↪  = wgs84)
Bierregaard_Osprey_Data_sf<-st_transform(Bierregaard_Osprey_Data_sf, azim_Bierr)
#to color code year, ggplot needs the year column to be a factor data type.
Bierregaard_Osprey_Data_sf$year<-as.factor(Bierregaard_Osprey_Data_sf$year)
#batch map the GPS points for each individual using ggplot
names<-unique(Bierregaard_Osprey_Data_sf$trackId)
for (i in names) {
  y = filter(Bierregaard_Osprey_Data_sf, trackId == i)
  pws<-plotshit(y)
  print(pws)
}
```

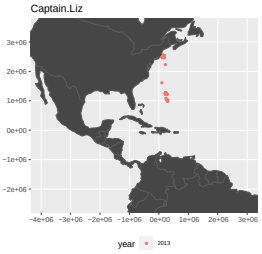


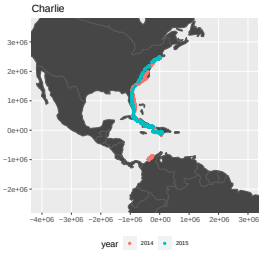

Bridget
year 2014

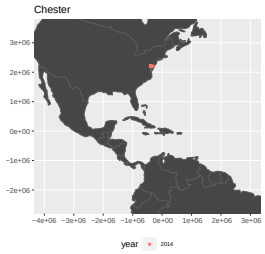

Buck
year 2009 2010 2011

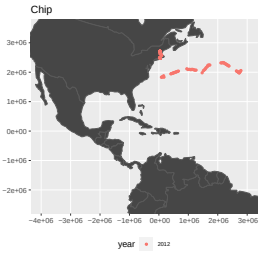

Caleb
year 2013

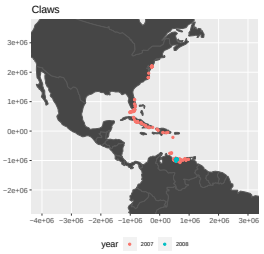

Caley
year 2009

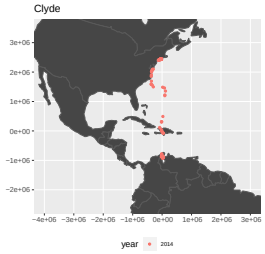

Captain.Liz
year 2013

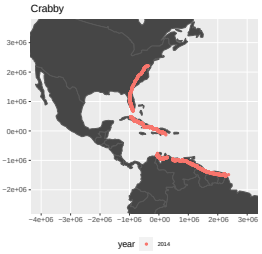

Charlie
year 2014 2015

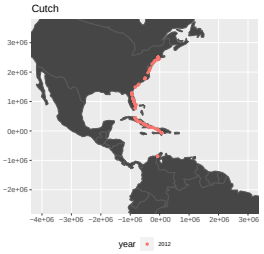

Chester
year 2014

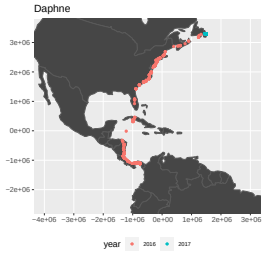

Chip
year 2012

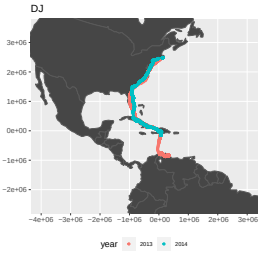

Claws
year 2007 2008

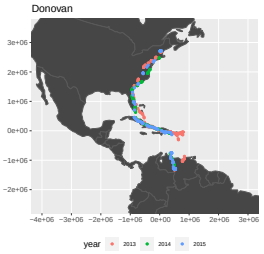

Clyde
year 2014

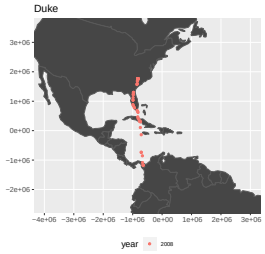

Crabby
year 2014

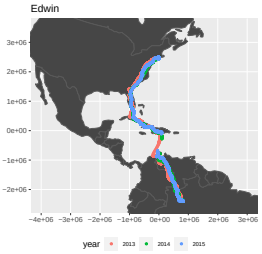

Cutch
year 2012

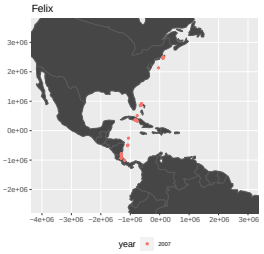

Daphne
year 2016 2017

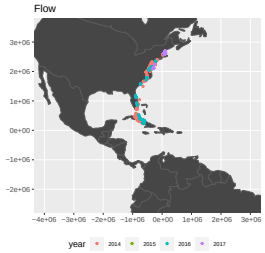

DJ
year 2013 2014

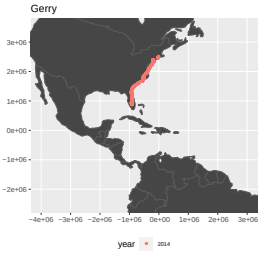

Donovan
year 2013 2014 2015

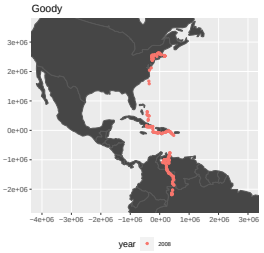

Duke
year 2008

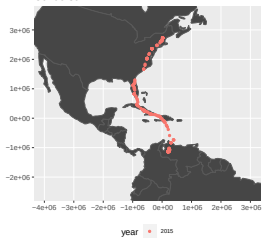

Edwin
year 2013 2014 2015


Felix
year 2007

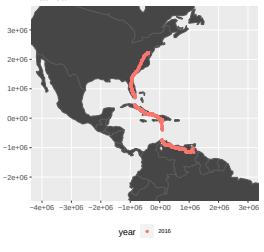

Flow
year 2014 2015 2016 2017

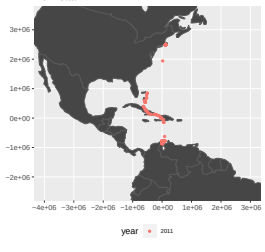

Gerry
year 2014

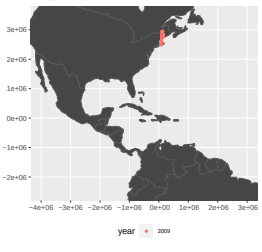

Goody
year 2008

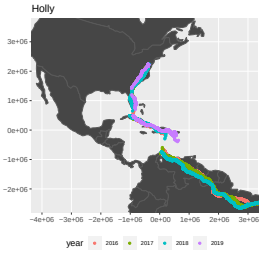

Gundersen
year 2015

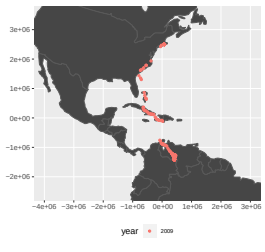

Gunny
year 2010

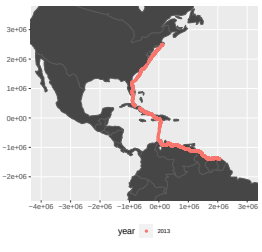

Hackett
year 2016

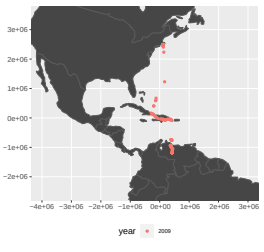

Henrietta
year 2011

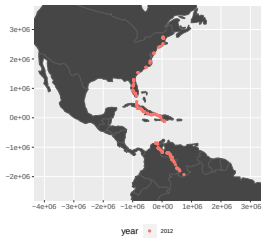

Hix.Jr
year 2009

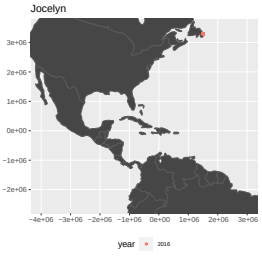

Holly
year 2016 2017 2018 2019

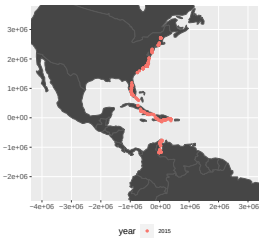

Hudson
year 2009

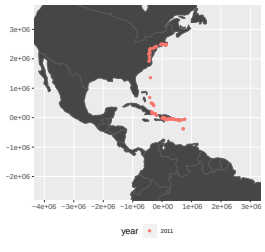

Icarius
year 2013

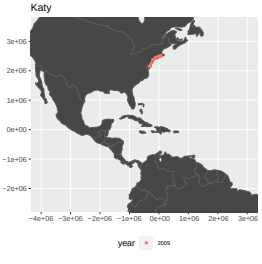

Isabel
year 2009

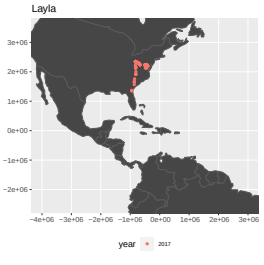

Jill
year 2012

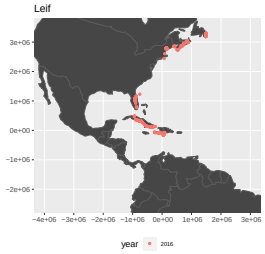

Jocelyn
year 2016


Juliet
year 2015

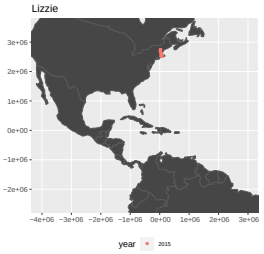

Katbird
year 2011

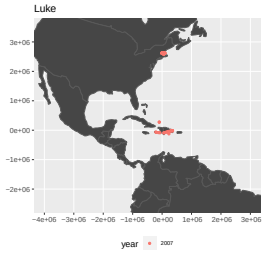

Katy
year 2009

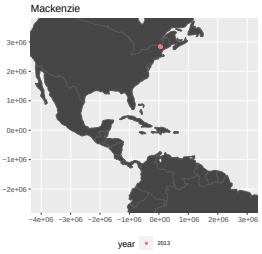

Layla
year 2017

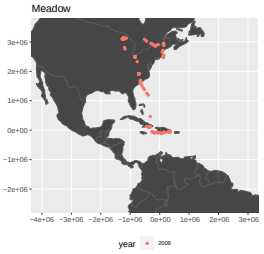

Leif
year 2016

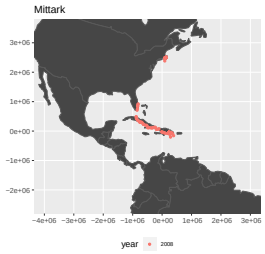

Little.Ricky
year 2008

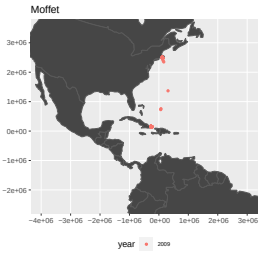

Lizzie
year 2015

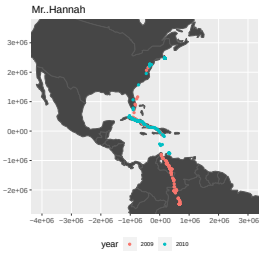

Luke
year 2007

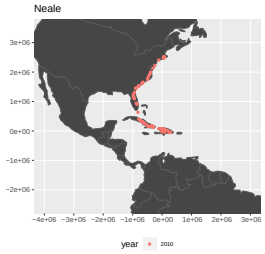

Mackenzie
year 2013

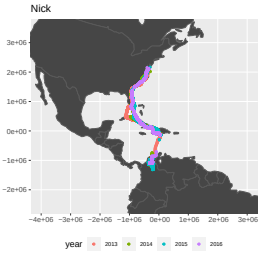

Meadow
year 2008

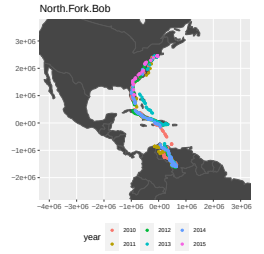

Mittark
year 2008

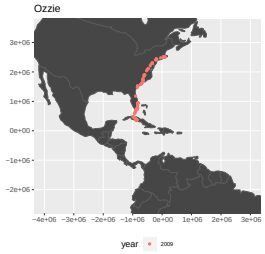

Moffet
year 2009

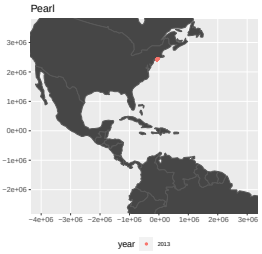

Mr..Hannah
year 2009 2010

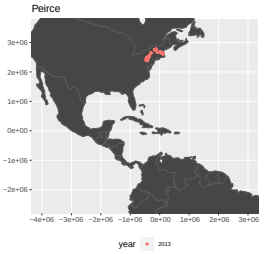

Neale
year 2010

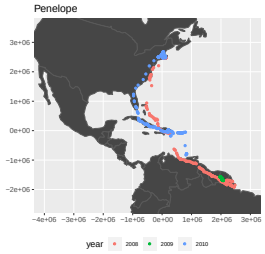

Nick
year 2013 2014 2015 2016

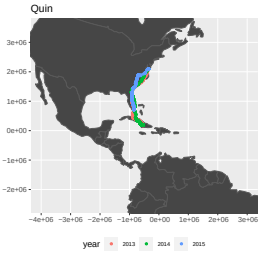

North.Fork.Bob
year 2010 2012 2014 2011 2013 2015

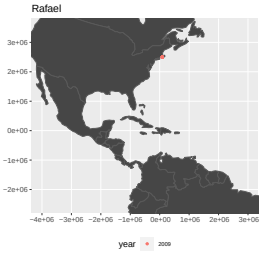

Ozzie
year 2009

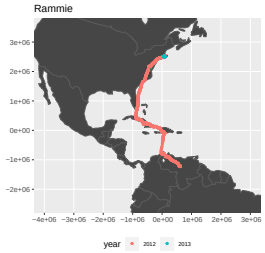

Pearl
year 2013

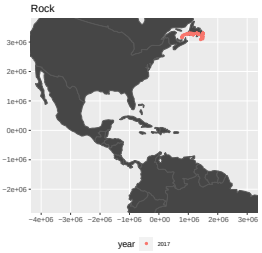

Peirce
year 2013

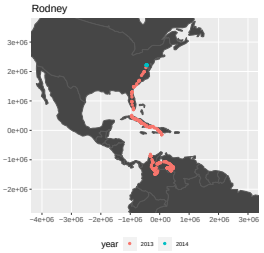

Penelope
year 2008 2009 2010

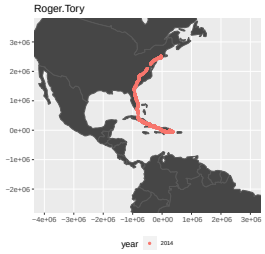

Quin
year 2013 2014 2015

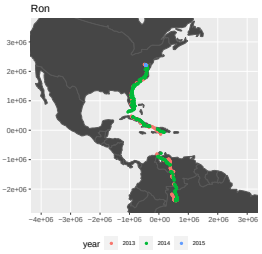

Rafael
year 2009

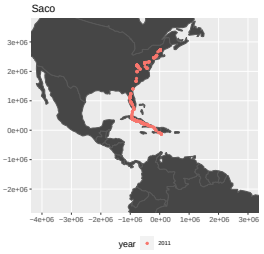

Rammie
year 2012 2013

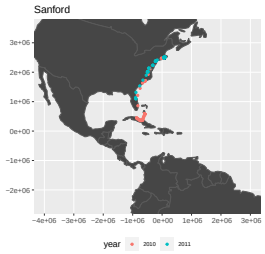

Rock
year 2017

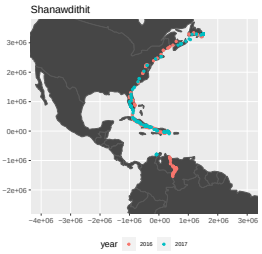

Rodney
year 2013 2014

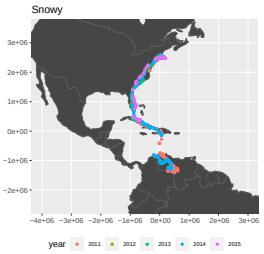

Roger.Tory
year 2014

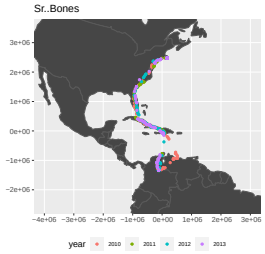

Ron
year 2013 2014 2015

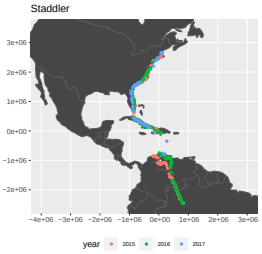

Saco
year 2011

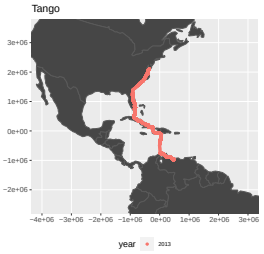

Sanford
year 2010 2011

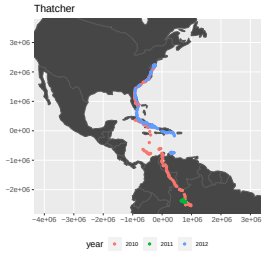

Shanawdithit
year 2016 2017

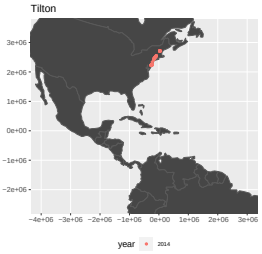

Snowy
year 2011 2012 2013 2014 2015

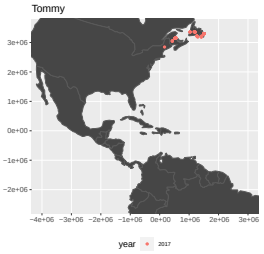

Sr..Bones
year 2010 2011 2012 2013

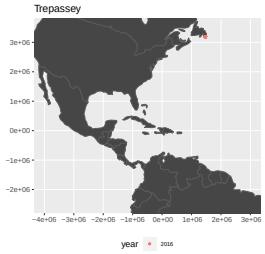

Staddler
year 2015 2016 2017

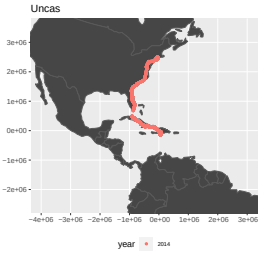

Tango
year 2013

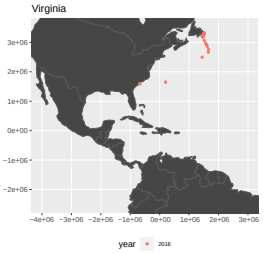

Thatcher
year 2010 2011 2012

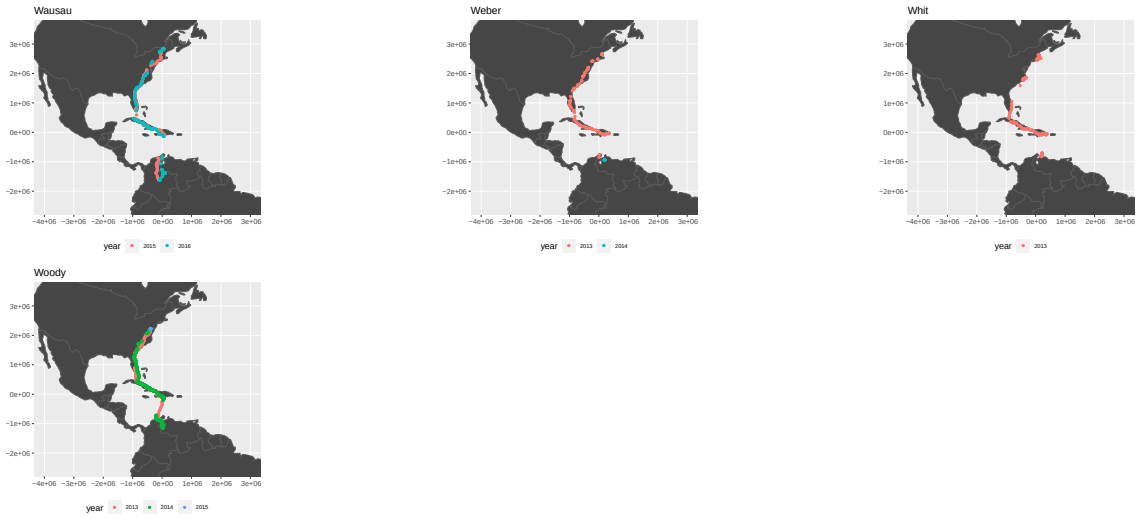

Tilton
year 2014


Tommy
year 2017


Trepassey
year 2016


Uncas
year 2014


Virginia
year 2016



From the plots, I can get an idea of how many fall migration events occurred for each bird. I can also tell if some individuals were juveniles during their first migration as juveniles spend 18 months wintering and return to breeding areas at three-years old. Juvenile wintering periods during the expected fall migration period were removed.

Next, I used an interactive map to determine the start/end dates of migration and the start/end latitude and longitude of each segment. I also determined if migration events were complete or not.

```
#project data using a GCS
Bierregaard_Osprey_Data_sf<-st_as_sf(x=Bierregaard_Osprey_Data, coords =
↪  c("location_long", "location_lat"), crs = wgs84)
#Create and interactive map where you can click on points for each migration event and
↪  visually determien start/end of migration. I did this for each event separately so I
↪  left an example of a plot here.
leaflet(filter(Bierregaard_Osprey_Data_sf, Bierregaard_Osprey_Data_sf$trackId ==
↪  "Belle_2015")) %>%
  addTiles() %>%
  addCircleMarkers(radius = 2, color = "red", opacity = 100) %>%
  addPopups(popup = ~htmlEscape(timestamp))
```

```
## PhantomJS not found. You can install it with webshot::install_phantomjs(). If it is installed, pleas
```

Keep only events that match the selection criteria: * Had a 1-hour or finer sampling frequency (tracked with GPS or GPS/GSM devices) * Completed at least two fall migrations * Completed at least two migrations as an adult * Male * Wintered in Central/South America. * Travelled for a minimum of 2 days through each migration region

```
#create a list of migration events to remove.
select<-c("Belle_2015", "Belle_2016", "Donovan_2014", "Donovan_2015", "Edwin_2013",
↪  "Edwin_2014", "North_Fork_Bob_2013", "North_Fork_Bob_2014", "Ron_2013", "Ron_2014",
↪  "Sr__Bones_2013", "Sr__Bones_2012", "Staddler_2015", "Staddler_2016", "Wausau_2015",
↪  "Wausau_2016")
```

```
Bierregaard_Osprey_Data$migrationEvent<-as.character(Bierregaard_Osprey_Data$migrationEvent)
Bierregaard_Osprey_Data<-Bierregaard_Osprey_Data[(Bierregaard_Osprey_Data$migrationEvent
↪  %in% select), ]
length(unique(Bierregaard_Osprey_Data$migrationEvent))
```

```
## [1] 16
```

```
length(unique(Bierregaard_Osprey_Data$trackId)) #16 migration events for 8 individuals
```

```
## [1] 8
```

#While using leaflet works, this method will take a long grueling time to complete. If you want to expedite this process you can export the data points to CSV files and find start/end dates in ArcGIS or QGIS.

```
Bierr_Event<-unique(Bierregaard_Osprey_Data$migrationEvent)
Bierregaard_Osprey_Data$migrationEvent<-gsub(" ", "_",
↪  Bierregaard_Osprey_Data$migrationEvent)
for (i in Bierr_Event) {
  ID2 <- subset(Bierregaard_Osprey_Data, migrationEvent == i)
  write.csv(ID2, file = paste0("Z://Personal_Folders/Lewis/Thesis/ROutput/",i, ".csv"))
} #make sure you change the file path so data is saved to your own folder.
```

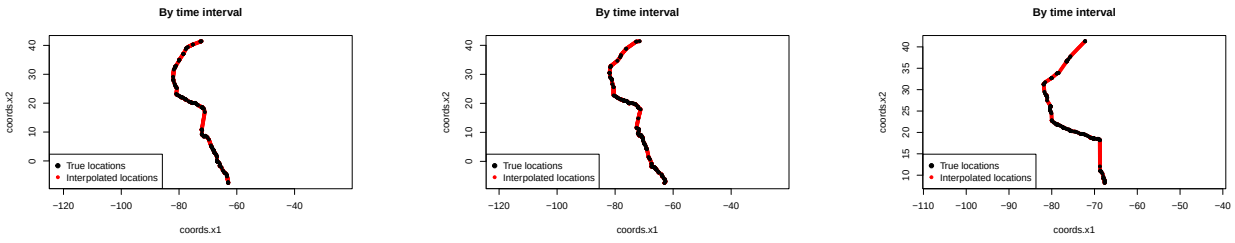
Next you will remove all fixes for each migration event that fall outside of the start and end date of migration. You will have to load in the metadata file with the migration event start/end dates.

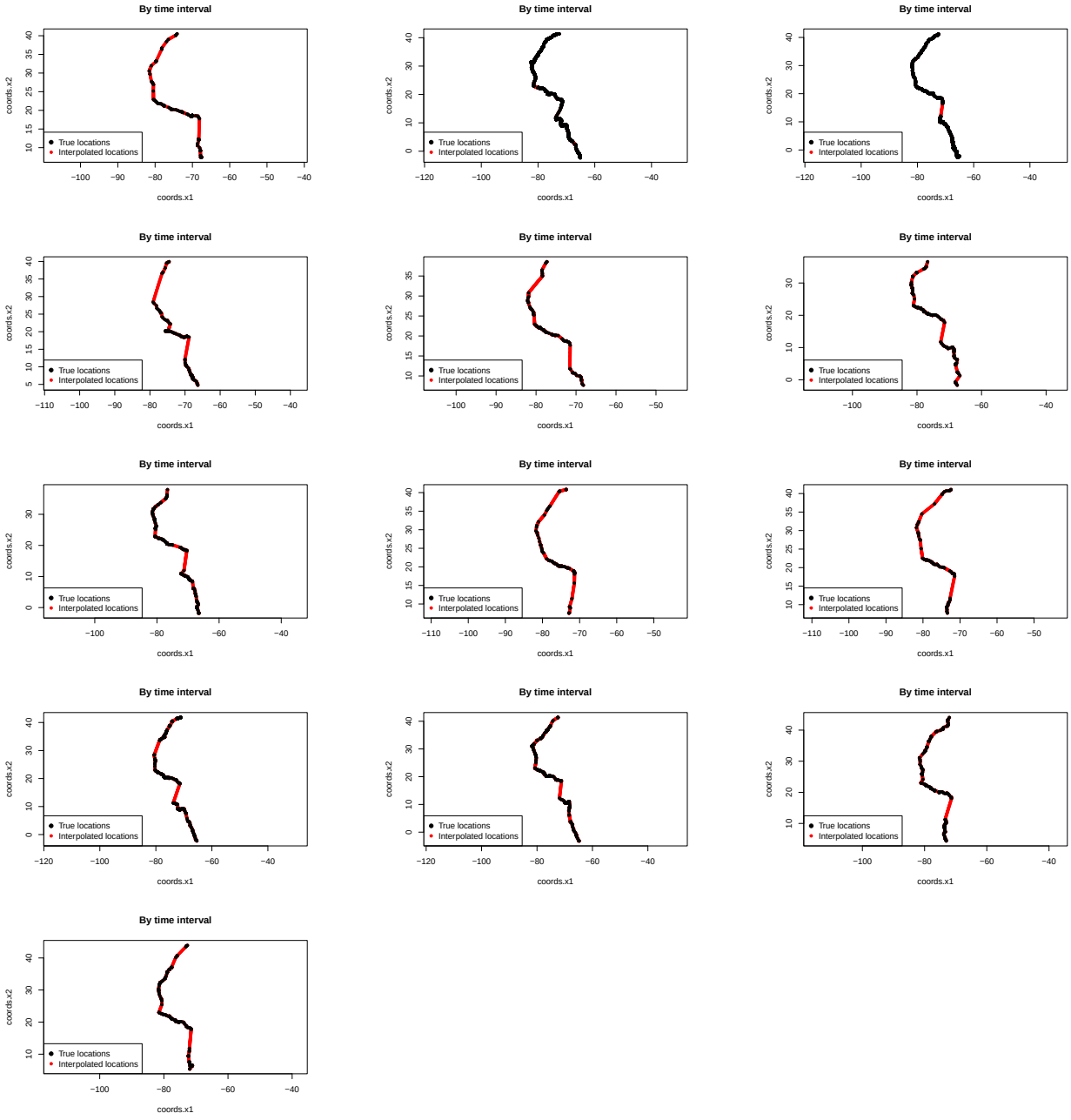
```
IndividualMetadata <- read_csv("IndividualMetadata.csv")
#make sure this is a dataframe
IndividualMetadata<-as.data.frame(IndividualMetadata)
#Format start/end timestamps so they are date/time format. I keep having to go back to
↪  the CSV file and adjust these values to yyyy-mm-dd hh:mm:ss
IndividualMetadata$Seg_End<-ymd_hms(IndividualMetadata$Seg_End)
IndividualMetadata$Seg_Start<-ymd_hms(IndividualMetadata$Seg_Start)
IndividualMetadata$Mig_Start<-ymd_hms(IndividualMetadata$Mig_Start)
IndividualMetadata$Mig_End<-ymd_hms(IndividualMetadata$Mig_End)
#I suggest checking the column class with the class() to ensure this worked properly.
#Now filter out non-migratory fixes for each migration event.
events<-unique(Bierregaard_Osprey_Data$migrationEvent)
Filtered<-data.frame()
for (i in events) {
  y = filter(Bierregaard_Osprey_Data, migrationEvent == i)
  x = filter(IndividualMetadata, migrationEvent == i)
  n = filter(y, timestamp <= x$Mig_End & timestamp >= x$Mig_Start)
  Filtered = rbind(Filtered, n)
}
#18,000 fixes left. You can check using the str() function.
```

Regularize the Trajectories

Regularized the data. Data was not collected at perfect 1-hour intervals for a 24 hour period. This means that when I calculate trajectory characteristics, points from the end of one day may connect to points at the start of the next day. If the osprey was sleeping during this time period, speed values would be very low and add inaccuracy to my dataset since I am only concerned with active migration. Regularizing will help me pinpoint times when osprey roosted so those points can be taken out later for 1-hour and 3-hour sampling frequencies after calculating trajectory characteristics. There are of course exceptions when osprey cross over open water. In these cases, speed should be high even after dark and will be delt with later.

```
#Convert to a move object. To do this I had to use the moveVis package since converting a
↪  DF to a move object with the move function was finicky.
OSPR<-df2move(Filtered,
        proj = wgs84,
        x = "location_long", y = "location_lat",
        time = "timestamp", track_id = "migrationEvent")
#Each event in the movestack is named with a unique trackID.
length(unique(OSPR@trackId)) #still have 16 migration events!

#regularize the time series through the move package. To do this we will have to apply
↪  the regularization to each move object within the Movestack (annoying, I know). The
↪  interpolate time function fills in positional points along a standard time series. We
↪  will use 1-hour to do this.
Stacked_1hour<-list() #empty list for output

for (i in 1:16) {
  x<-interpolateTime(OSPR[[i]], time=as.difftime(1, units="hours"),
↪  spaceMethod='rhumbline')
  plot(x, col="red",pch=20, main="By time interval")
  points(OSPR[[i]], col = "black", pch = 20)
  print(legend("bottomleft", c("True locations", "Interpolated locations"),
               col=c("black", "red"), pch=c(19,20)))
  Stacked_1hour[[i]] = x
}

#convert output list of move objects into a move stack.
OSPR_1hour<-moveStack(Stacked_1hour)
length(unique(OSPR_1hour@trackId)) #check that all migration events are accounted for
#check that the time lag is 1 hour via the mean time lag.
list_1hour<-timeLag(OSPR_1hour, units = "hours")
sapply(list_1hour, mean)
```






```
#Separate fixes into regions based on the start/end dates and combine into a singular
↪  data frame
#convert move object into a dataframe.
OSPR_1hour_DF<-as.data.frame(OSPR_1hour)
#annotate dataframe
events<-unique(OSPR_1hour_DF$trackId)
segs<-data.frame()
for (i in events) {
  y = filter(OSPR_1hour_DF, trackId == i)
```

```
  x = filter(IndividualMetadata, migrationEvent == i)
  for (j in 1:nrow(x)) {
    z = x[j,] #you have to specify the row the row
    n = filter(y, timestamps <= z$Seg_End & timestamps > z$Seg_Start)
    b = merge(n, z, by.x = "trackId", by.y ="migrationEvent")
    segs = rbind(segs, b)
  }
}
length(unique(segs$SegmentID)) #48 segments total (16 x 3)
```

Add in regional data

```
## [1] 48
```

Thin Trajectories

Next I will thin my trajectories to 3-hour and 1-day sampling frequencies. I will do this my migration event region rather than migration event. This just streamlines the code more since I will have to take random samples by migration event region later.

```
#First create a move object
Subset_1hour_move<-df2move(segs,
        proj = wgs84,
        x = "coords.x1", y = "coords.x2",
        time = "timestamps", track_id = "SegmentID")
#Regularizing create 1-hour equal intervals for the base dataset, so need to thin to 1
↪  hour again.

#3-hour sampling frequency
Subset_3hr<-list()
for (i in 1:48) {
  x<-thinTrackTime(Subset_1hour_move[[i]], interval = as.difftime(3, units="hours"),
                           tolerance = as.difftime(15, units="mins"), criteria =
                           ↪  "closest")
  split<-move::split(x)
  Subset_3hr[[i]] = split$selected
}
Subset_3hr_move<-moveStack(Subset_3hr)
str(Subset_3hr_move@trackId)
```

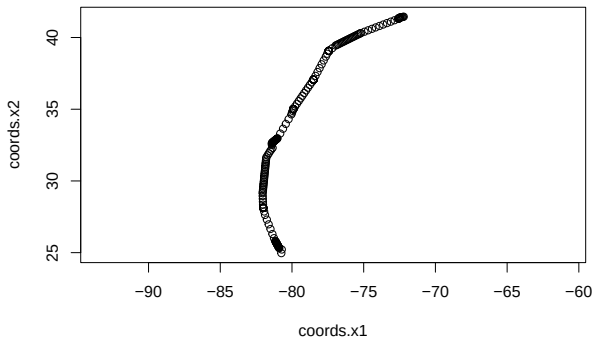
```
##  Factor w/ 48 levels "Belle_2015_US",..: 1 1 1 1 1 1 1 1 1 1 ...
```

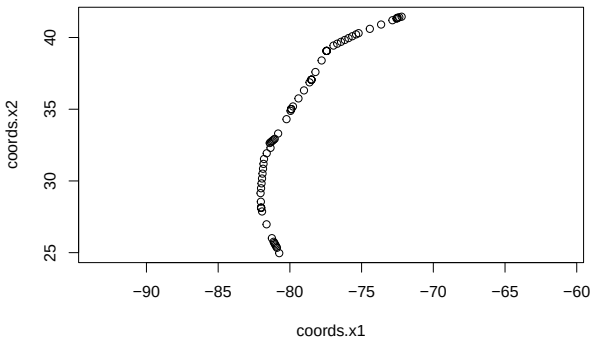
```
#1-day sampling frequency
Subset_1day<-list()
for (i in 1:48) {
  x<-thinTrackTime(Subset_1hour_move[[i]], interval = as.difftime(1, units="days"),
                           tolerance = as.difftime(2, units="hours"), criteria =
                           ↪  "closest")
  split<-move::split(x)
  Subset_1day[[i]] = split$selected
}
```

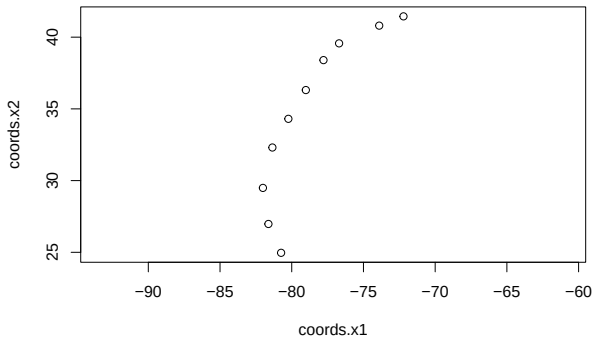
```
Subset_1day_move<-moveStack(Subset_1day)
str(Subset_1day_move@trackId)
```

```
##  Factor w/ 48 levels "Belle_2015_US",..: 1 1 1 1 1 1 1 1 1 1 ...
```

```
#Check to see if thinning looks right.
plot(Subset_1hour_move[[1]])
plot(Subset_3hr_move[[1]])
plot(Subset_1day_move[[1]])
```







Calculate Trajectory Signals and Summarize Data

Now we have all four data sets. The final step before analysis is to calculate trajectory characteristics.

Calculate signals with 'move'

When calculating distance measurements, Keep in mind that data must be projected using a projection that preserves distance (best projection to use is an azimuthal equidstant projection), while angle measurements need to be calculated using a mercator projection.

```
#Project data with custom azimuthal equidistant projection
Subset_1hour_move_azim <- spTransform(Subset_1hour_move, CRSobj=azim_Bierr)
Subset_3hr_move_azim <- spTransform(Subset_3hr_move, CRSobj=azim_Bierr)
Subset_1day_move_azim <- spTransform(Subset_1day_move, CRSobj=azim_Bierr)

#calculate distance between consecutive GPS points (meters)
Subset_1hour_move_azim$distance<-unlist(lapply(distance(Subset_1hour_move_azim), c, NA))
Subset_3hr_move_azim$distance<-unlist(lapply(distance(Subset_3hr_move_azim), c, NA))
Subset_1day_move_azim$distance<-unlist(lapply(distance(Subset_1day_move_azim), c, NA))

#calculate speed/velocity between consecutive GPS points (m/s)
Subset_1hour_move_azim$speed<-unlist(lapply(speed(Subset_1hour_move_azim), c, NA))
Subset_3hr_move_azim$speed<-unlist(lapply(speed(Subset_3hr_move_azim), c, NA))
Subset_1day_move_azim$speed<-unlist(lapply(speed(Subset_1day_move_azim), c, NA))

#calculating lag between fixes.
Subset_1hour_move_azim$lag<-unlist(lapply(timeLag(Subset_1hour_move_azim,units =
↪  "hours"), c, NA))
Subset_3hr_move_azim$lag<-unlist(lapply(timeLag(Subset_3hr_move_azim,units = "hours"), c,
↪  NA))
Subset_1day_move_azim$lag<-unlist(lapply(timeLag(Subset_1day_move_azim,units = "hours"),
↪  c, NA))

#Project data with conformal projection: the Mercator projection is a conformal
↪  cylindrical projection
Subset_1hour_move_mercator <- spTransform(Subset_1hour_move, CRSobj=mercator)
Subset_3hr_move_mercator <- spTransform(Subset_3hr_move, CRSobj=mercator)
Subset_1day_move_mercator <- spTransform(Subset_1day_move, CRSobj=mercator)

#calculating absolute angle
Subset_1hour_move_mercator$abs_angle<-unlist(lapply(angle(Subset_1hour_move_mercator), c,
↪  NA))
Subset_3hr_move_mercator$abs_angle<-unlist(lapply(angle(Subset_3hr_move_mercator), c,
↪  NA))
Subset_1day_move_mercator$abs_angle<-unlist(lapply(angle(Subset_1day_move_mercator), c,
↪  NA))

#calculate Relative Angle
Subset_1hour_move_mercator$rel_angle<-unlist(lapply(turnAngleGc(Subset_1hour_move_mercator),function(x)
↪  c(NA, x, NA)))
Subset_3hr_move_mercator$rel_angle<-unlist(lapply(turnAngleGc(Subset_3hr_move_mercator),function(x)
↪  c(NA, x, NA)))
Subset_1day_move_mercator$rel_angle<-unlist(lapply(turnAngleGc(Subset_1day_move_mercator),function(x)
↪  c(NA, x, NA)))
```

Calculate signals with Base R and 'adehabitatLT'

Next I will transform my move objects into data frames and extract the characteristics I calculated previously. I will also calculate turn speed in based R.

```
#1-hour
Subset_1hour_move_azim_DF<-as.data.frame(Subset_1hour_move_azim)
```

```
Subset_1hour_move_mercator_DF<-as.data.frame(Subset_1hour_move_mercator)
Subset_1hour_move_azim_DF$rel_angle<-Subset_1hour_move_mercator_DF$rel_angle
Subset_1hour_move_azim_DF$abs_angle<-Subset_1hour_move_mercator_DF$abs_angle
Subset_1hour_move_azim_DF$turn_speed<-Subset_1hour_move_azim_DF$speed *
↪  cos(Subset_1hour_move_azim_DF$rel_angle)

#3-hour
Subset_3hr_move_azim_DF<-as.data.frame(Subset_3hr_move_azim)
Subset_3hr_move_mercator_DF<-as.data.frame(Subset_3hr_move_mercator)
Subset_3hr_move_azim_DF$rel_angle<-Subset_3hr_move_mercator_DF$rel_angle
Subset_3hr_move_azim_DF$abs_angle<-Subset_3hr_move_mercator_DF$abs_angle
Subset_3hr_move_azim_DF$turn_speed<-Subset_3hr_move_azim_DF$speed *
↪  cos(Subset_3hr_move_azim_DF$rel_angle)

#1-day
Subset_1day_move_azim_DF<-as.data.frame(Subset_1day_move_azim)
Subset_1day_move_mercator_DF<-as.data.frame(Subset_1day_move_mercator)
Subset_1day_move_azim_DF$rel_angle<-Subset_1day_move_mercator_DF$rel_angle
Subset_1day_move_azim_DF$abs_angle<-Subset_1day_move_mercator_DF$abs_angle
Subset_1day_move_azim_DF$turn_speed<-Subset_1day_move_azim_DF$speed *
↪  cos(Subset_1day_move_azim_DF$rel_angle)
```

Finally, I will transform my dataframe into an ltraj object with the azimuthal equidistant projection so I can calculate First Passage Time (FPT).

```
#transform datasets into ltraj objects
Subset_1hour_traj<-Subset_1hour_move_azim_DF
coordinates(Subset_1hour_traj) <- c("coords.x1", "coords.x2")
proj4string(Subset_1hour_traj) <- azim_Bierr
Subset_1hour_traj <- as.ltraj(coordinates(Subset_1hour_traj),
                          date=Subset_1hour_traj$timestamps,
                          id=Subset_1hour_traj$trackId, typeII=TRUE)
Subset_3hr_traj<-Subset_3hr_move_azim_DF
coordinates(Subset_3hr_traj) <- c("coords.x1", "coords.x2")
proj4string(Subset_3hr_traj) <- azim_Bierr
Subset_3hr_traj <- as.ltraj(coordinates(Subset_3hr_traj),
                          date=Subset_3hr_traj$timestamps,
                          id=Subset_3hr_traj$trackId, typeII=TRUE)
Subset_1day_traj<-Subset_1day_move_azim_DF
coordinates(Subset_1day_traj) <- c("coords.x1", "coords.x2")
proj4string(Subset_1day_traj) <- azim_Bierr
Subset_1day_traj <- as.ltraj(coordinates(Subset_1day_traj),
                          date=Subset_1day_traj$timestamps,
                          id=Subset_1day_traj$trackId, typeII=TRUE)
```

```
#before I can calculate FPT I need to calculate the variogram maximum which will provide
↪  the spatial continuity at which interactions between individuals and their
↪  environment are most apparent. I want the spatial continuity to be the same for all
↪  each frequency so I can compare how FPT changes between frequency.

#1-hour fpt
fpt_max_subset_1hour<-data.frame()
```

```
for (i in 1:48) {
  x<-fpt(Subset_1hour_traj[i], radii=0:100, units="hours") #calculate fpt values
  y<- varlogfpt(x, graph=TRUE) #variogram of fpt
  z<-as.numeric(y[1,])
  fpt<-max(z, na.rm = TRUE) #maximum of the variogram
  fpt_max_subset_1hour<-rbind(fpt_max_subset_1hour, fpt)
}
mean(fpt_max_subset_1hour[,1]) #11 km (round up to nearest whole number)

fpt_max_subset_3hour<-data.frame()
for (i in 1:48) {
  x<-fpt(Subset_3hr_traj[i], radii=0:100, units="hours")
  y<- varlogfpt(x, graph=TRUE)
  z<-as.numeric(y[1,])
  fpt<-max(z, na.rm = TRUE)
  fpt_max_subset_3hour<-rbind(fpt_max_subset_3hour, fpt)
}
mean(fpt_max_subset_3hour[,1]) #10 km

fpt_max_subset_1day<-data.frame()
for (i in 1:48) {
  x<-fpt(Subset_1day_traj[i], radii=0:100, units="hours")
  y<- varlogfpt(x, graph=TRUE)
  z<-as.numeric(y[1,])
  fpt<-max(z, na.rm = TRUE)
  fpt_max_subset_1day<-rbind(fpt_max_subset_1day, fpt)
}
mean(fpt_max_subset_1day[,1]) #3 km with an error
```

Since all three frequencies show different max spatial continuity, I decided to use the 1-hour continuity of 11 km to stay consistant.

```
#calculate FPT
#1-hour fpt
fpt_subset_1hour<-data.frame()
for (i in 1:48) {
  x<-fpt(Subset_1hour_traj[i], radii=0:100, units="hours")
  y<-x[[1]]$r11 #use 11 km radii
  r<-as.data.frame(y)
  id<-id(Subset_1hour_traj[i])
  r$migrationEvent<-paste(id)
  fpt_subset_1hour<-rbind(fpt_subset_1hour, r)
}
colnames(fpt_subset_1hour)<- c("FPT", "migrationEvent")
Subset_1hour_fin<-cbind(Subset_1hour_move_azim_DF, fpt_subset_1hour)

#3-hour fpt
fpt_subset_3hr<-data.frame()
for (i in 1:48) {
  x<-fpt(Subset_3hr_traj[i], radii=0:100, units="hours")
  y<-x[[1]]$r11
  r<-as.data.frame(y)
  id<-id(Subset_3hr_traj[i])
```

```
  r$migrationEvent<-paste(id)
  fpt_subset_3hr<-rbind(fpt_subset_3hr, r)
}
colnames(fpt_subset_3hr)<- c("FPT", "migrationEvent")
Subset_3hr_fin<-cbind(Subset_3hr_move_azim_DF, fpt_subset_3hr)

#1-day
fpt_subset_1day<-data.frame()
for (i in 1:48) {
  x<-fpt(Subset_1day_traj[i], radii=0:100, units="hours")
  y<-x[[1]]$r11
  r<-as.data.frame(y)
  id<-id(Subset_1day_traj[i])
  r$migrationEvent<-paste(id)
  fpt_subset_1day<-rbind(fpt_subset_1day, r)
}
colnames(fpt_subset_1day)<- c("FPT", "migrationEvent")
Subset_1day_fin<-cbind(Subset_1day_move_azim_DF, fpt_subset_1day)
```

Remove Nightime Fixes and prepare data for analysis

Lastly, I will remove fixes calculated during hours of darkness where the bird was overland. While osprey may migrate at night on occasion, this typically only occurs over water or deserts. The osprey in this study do not encounter deserts, so I will remove any fixes during hours of darkness that intersect with land. I will only do this for 1-hour and 3-hours since the 1-day frequency is to course to determine roosting vs. active migration.

```
#calculate daylight. I used a 30 minute daylight buffer to account for possible movement
↪  at dusk.
Subset_1hour_fin$DaylightUTC<-computeIsDayByLocation(Subset_1hour_fin$timestamps,
                                                     latDeg = Subset_1hour_fin$y,
                                                     longDeg = Subset_1hour_fin$x,
                                                     timeZone =
                                                     ↪  getHoursAheadOfUTC(Subset_1hour_fin$timestamps
                                                     ↪  duskOffset = 0.5)

Subset_3hr_fin$DaylightUTC<-computeIsDayByLocation(Subset_3hr_fin$timestamps,
                                                     latDeg = Subset_3hr_fin$y,
                                                     longDeg = Subset_3hr_fin$x,
                                                     timeZone =
                                                     ↪  getHoursAheadOfUTC(Subset_3hr_fin$timestamps)
                                                     ↪  duskOffset = 0.5)

#create sf objects.
Subset_1hour_fin_sf<-st_as_sf(x=Subset_1hour_fin, coords = c("x", "y"), crs = wgs84)
Subset_1hour_fin_sf<-st_transform(Subset_1hour_fin_sf, azim_Bierr)
Subset_3hr_fin_sf<-st_as_sf(x=Subset_3hr_fin, coords = c("x", "y"), crs = wgs84)
Subset_3hr_fin_sf<-st_transform(Subset_3hr_fin_sf, azim_Bierr)

#filter out points during dark hours where birds where not over the ocean.
Subset_1hour_fin_sf_filter<-st_join(Subset_1hour_fin_sf, world_filter, join = st_within)
Subset_1hour_fin_sf_filter$name[is.na(Subset_1hour_fin_sf_filter$name)] <- "Ocean" #need
↪  to replace NA values with a character for filtering
```

```
Subset_1hour_fin_sf_filter$DaylightUTC<-as.character(Subset_1hour_fin_sf_filter$DaylightUTC)
Subset_1hour_fin_sf_filter<-Subset_1hour_fin_sf_filter[!(Subset_1hour_fin_sf_filter$name
↪  != "Ocean" &
                                Subset_1hour_fin_sf_filter$DaylightUTC == "FALSE"),]
Subset_1hour_fin_filter<-st_drop_geometry(Subset_1hour_fin_sf_filter)

Subset_3hr_fin_sf_filter<-st_join(Subset_3hr_fin_sf, world_filter, join = st_within)
Subset_3hr_fin_sf_filter$name[is.na(Subset_3hr_fin_sf_filter$name)] <- "Ocean"
Subset_3hr_fin_sf_filter$DaylightUTC<-as.character(Subset_3hr_fin_sf_filter$DaylightUTC)
Subset_3hr_fin_sf_filter<-Subset_3hr_fin_sf_filter[!(Subset_3hr_fin_sf_filter$name !=
↪  "Ocean" &
                                Subset_3hr_fin_sf_filter$DaylightUTC == "FALSE"),]
Subset_3hr_fin_filter<-st_drop_geometry(Subset_3hr_fin_sf_filter)
```

Remove NA values and clean up data columns

```
#Remove NA values
Subset_1hour_fin_filter<-na.omit(Subset_1hour_fin_filter[,c("time", "distance", "speed",
↪  "lag","trackId", "rel_angle", "abs_angle", "turn_speed", "FPT" )])
Subset_3hr_fin_filter<-na.omit(Subset_3hr_fin_filter[,c("time", "distance", "speed",
↪  "lag","trackId", "rel_angle", "abs_angle", "turn_speed", "FPT" )])
Subset_1day_fin_filter<-na.omit(Subset_1day_fin[,c("time", "distance", "speed",
↪  "lag","trackId", "rel_angle", "abs_angle", "turn_speed", "FPT" )])

#Migration Event ID
Subset_1hour_fin_filter<-merge(Subset_1hour_fin_filter, IndividualMetadata, by.x =
↪  "trackId", by.y = "SegmentID", all.x = T)
Subset1hour<-Subset_1hour_fin_filter[ ,c(1,4,7:12,14,16)] #"Massachusetts" "New
↪  Hampshire" "Connecticut"   "Maryland"
Subset_3hr_fin_filter<-merge(Subset_3hr_fin_filter, IndividualMetadata, by.x = "trackId",
↪  by.y = "SegmentID", all.x = T)
Subset3hour<-Subset_3hr_fin_filter[ ,c(1,4,7:12,14,16)]
Subset_1day_fin_filter<-merge(Subset_1day_fin_filter, IndividualMetadata, by.x =
↪  "trackId", by.y = "SegmentID", all.x = T)
Subset1day<-Subset_1day_fin_filter[ ,c(1,4,7:12,14,16)]

#Take the absolute value of absolute angle
Subset1hour$abs_angle<-abs(Subset1hour$abs_angle)
Subset3hour$abs_angle<-abs(Subset3hour$abs_angle)
Subset1day$abs_angle<-abs(Subset1day$abs_angle)
```

Take random samples

```
set.seed(1998)

#Random samples
min_Subset1hour<-Subset1hour %>%
  group_by(trackId) %>%
  summarise(N = n()) %>%
  as.data.frame()
min(min_Subset1hour$N) #22 random samples needed
```

```
## [1] 22
```

```
Samp_Subset1hour <- Subset1hour %>%
  group_by(trackId) %>%
  sample_n(22)

min_Subset3hour<-Subset3hour %>%
  group_by(trackId) %>%
  summarise(N = n()) %>%
  as.data.frame()
min(min_Subset3hour$N) #6 random samples
```

```
## [1] 6
```

```
Samp_Subset3hour <- Subset3hour %>%
  group_by(trackId) %>%
  sample_n(6)

min_Subset1day<-Subset1day %>%
  group_by(trackId) %>%
  summarise(N = n()) %>%
  as.data.frame()
min(min_Subset1day$N) #1 random samples
```

```
## [1] 1
```

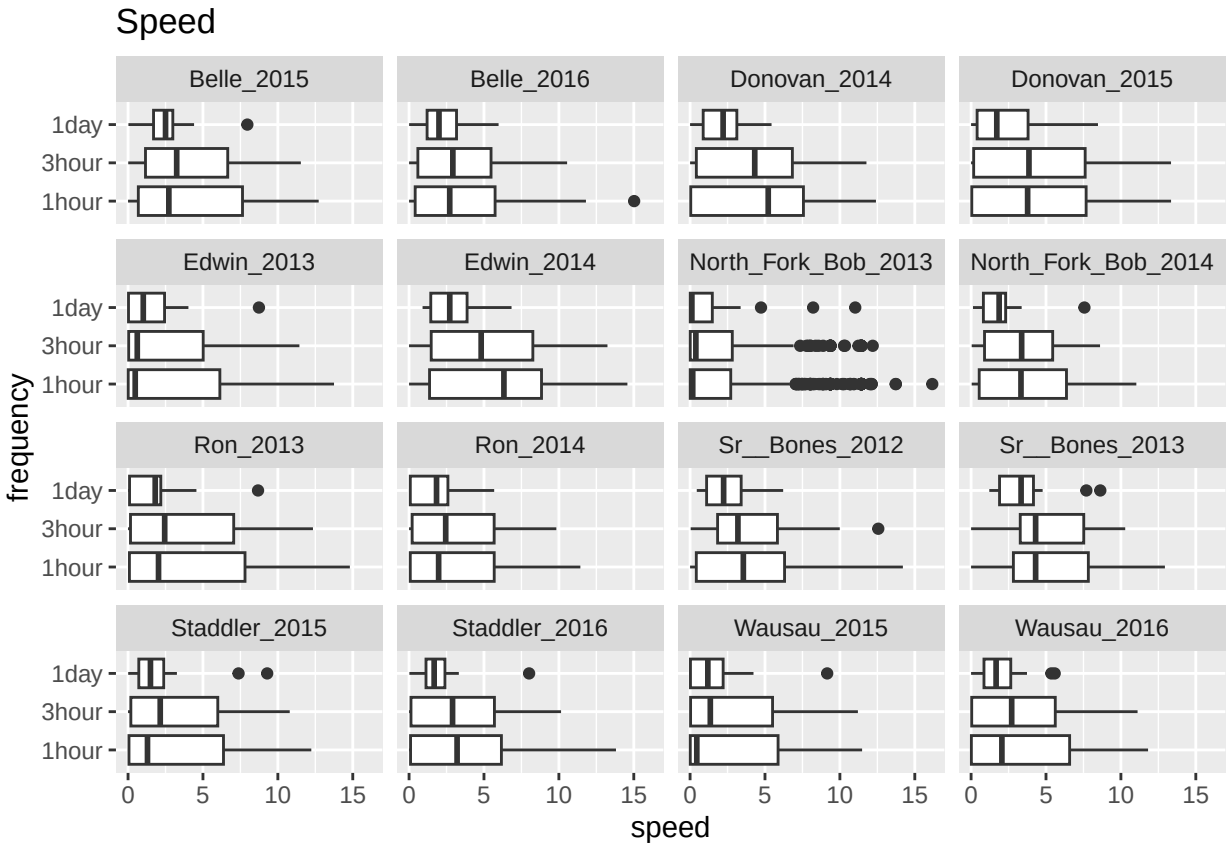
```
Samp_Subset1day <- Subset1day %>%
  group_by(trackId) %>%
  sample_n(1)
```

Data Analysis

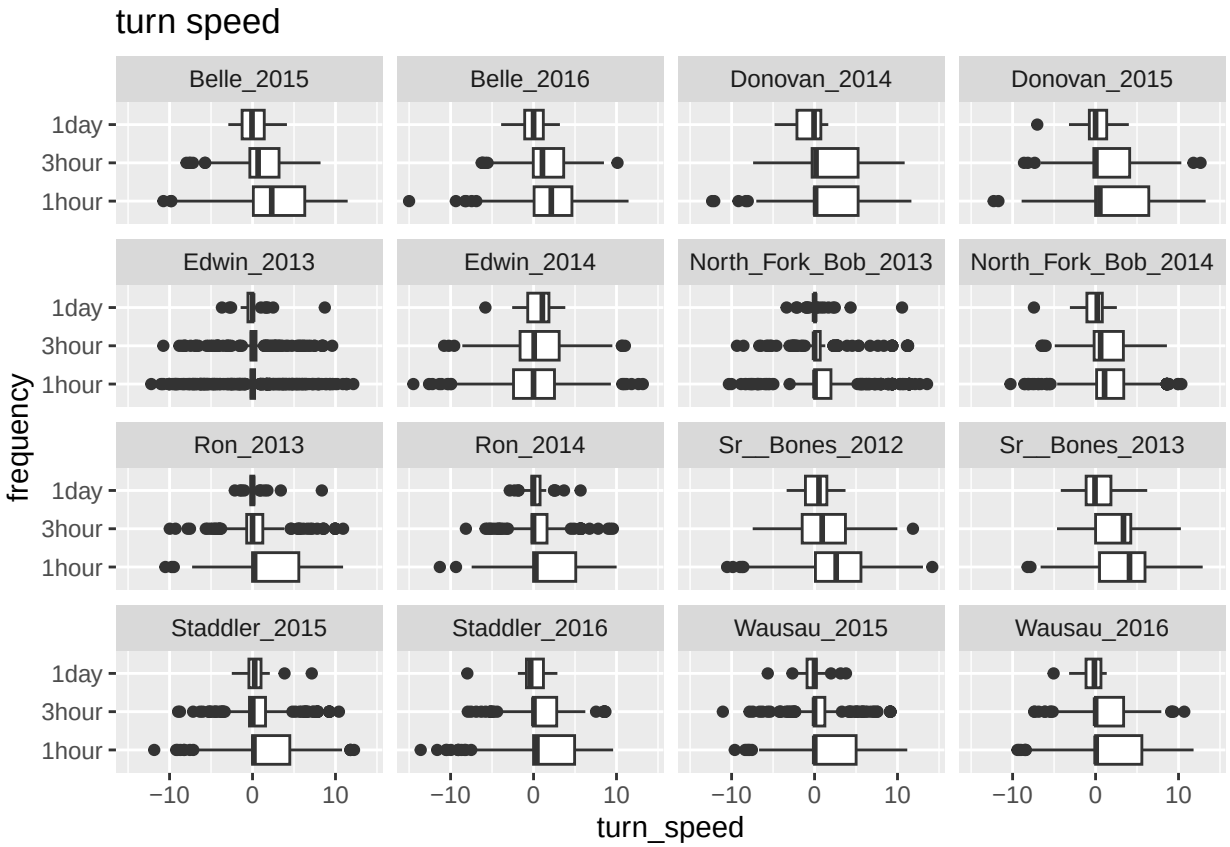
Exploratory data analysis

Start by summarizing the raw data

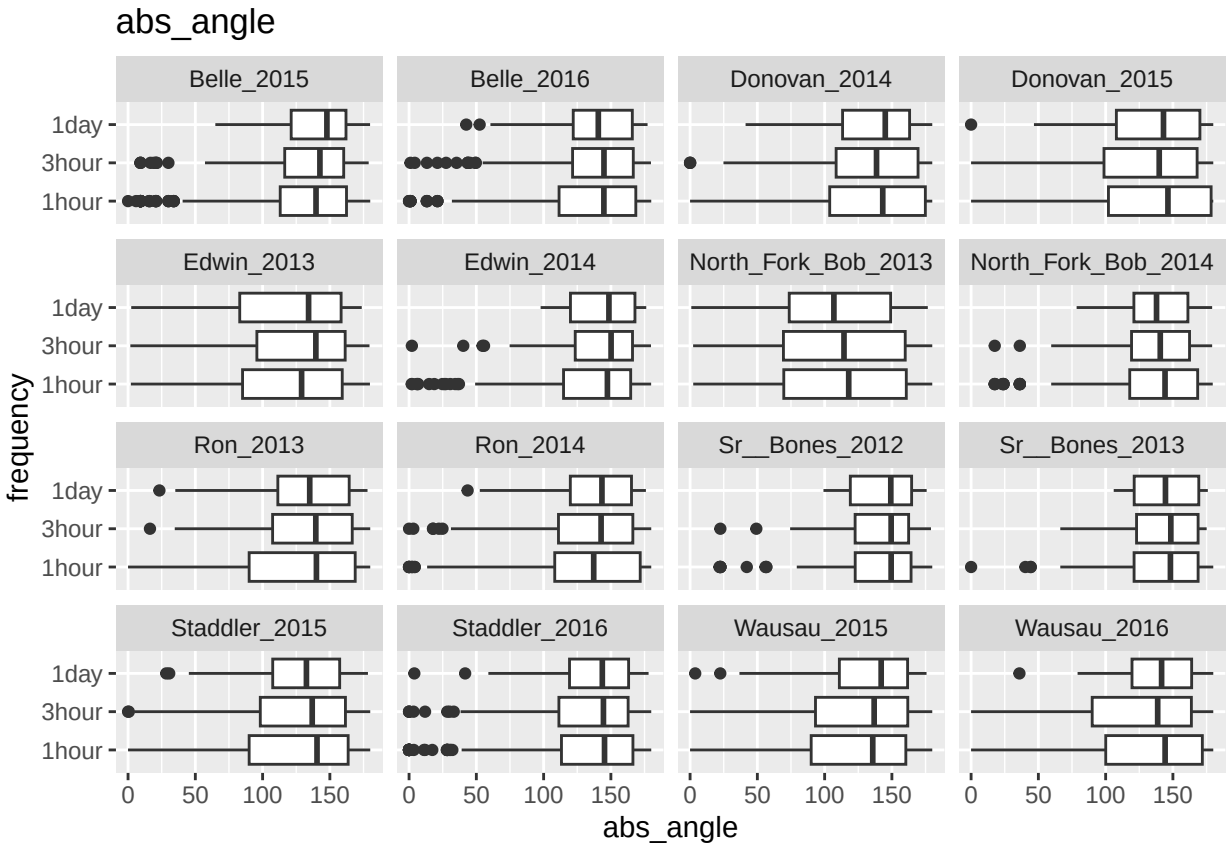
```
#explore signal means
Subset1hour$frequency<-paste("1hour")
Subset3hour$frequency<-paste("3hour")
Subset1day$frequency<-paste("1day")
comparison<-rbind(Subset1hour,Subset3hour,Subset1day)
comparison$frequency <- factor(comparison$frequency,
    levels = c('1hour','3hour', '1day'),ordered = TRUE)

#by migration event
ggplot(comparison, aes(x=speed, y = frequency)) +
  geom_boxplot() +
  ggtitle("Speed")+
  facet_wrap(~migrationEvent, nrow = 4)
```



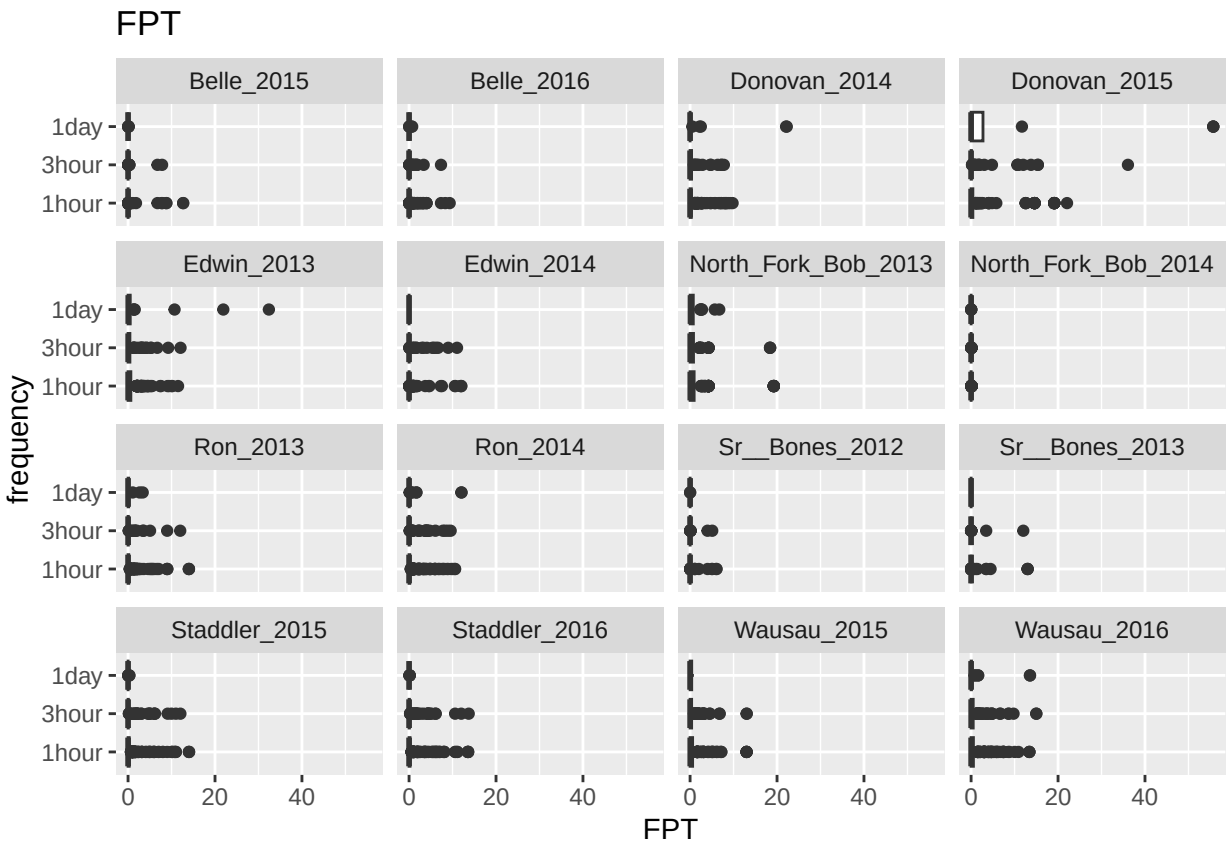

```
ggplot(comparison, aes(x=turn_speed, y = frequency)) +
  geom_boxplot() +
  ggtitle("turn speed")+
  facet_wrap(~migrationEvent, nrow = 4)
```



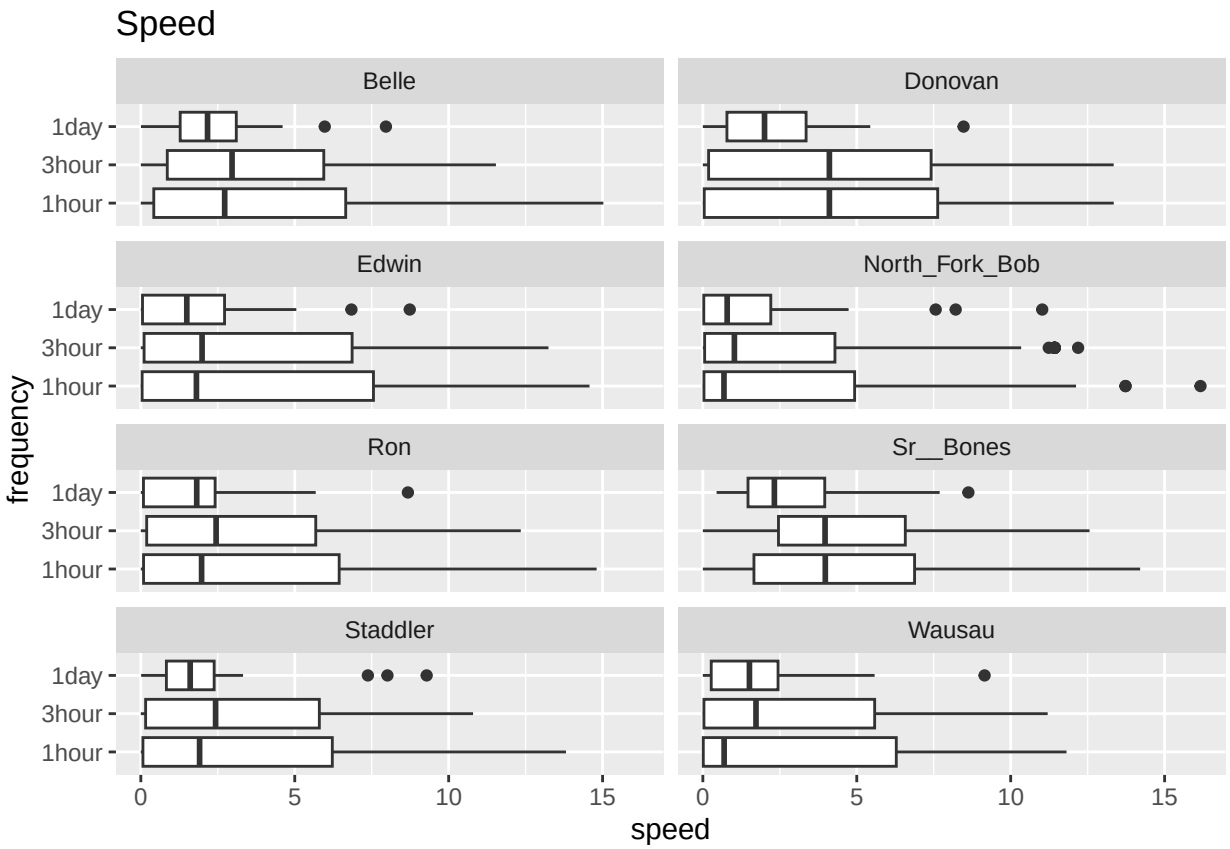

```
ggplot(comparison, aes(x=abs_angle, y = frequency)) +
  geom_boxplot() +
  ggtitle("abs_angle")+
  facet_wrap(~migrationEvent, nrow = 4)
```



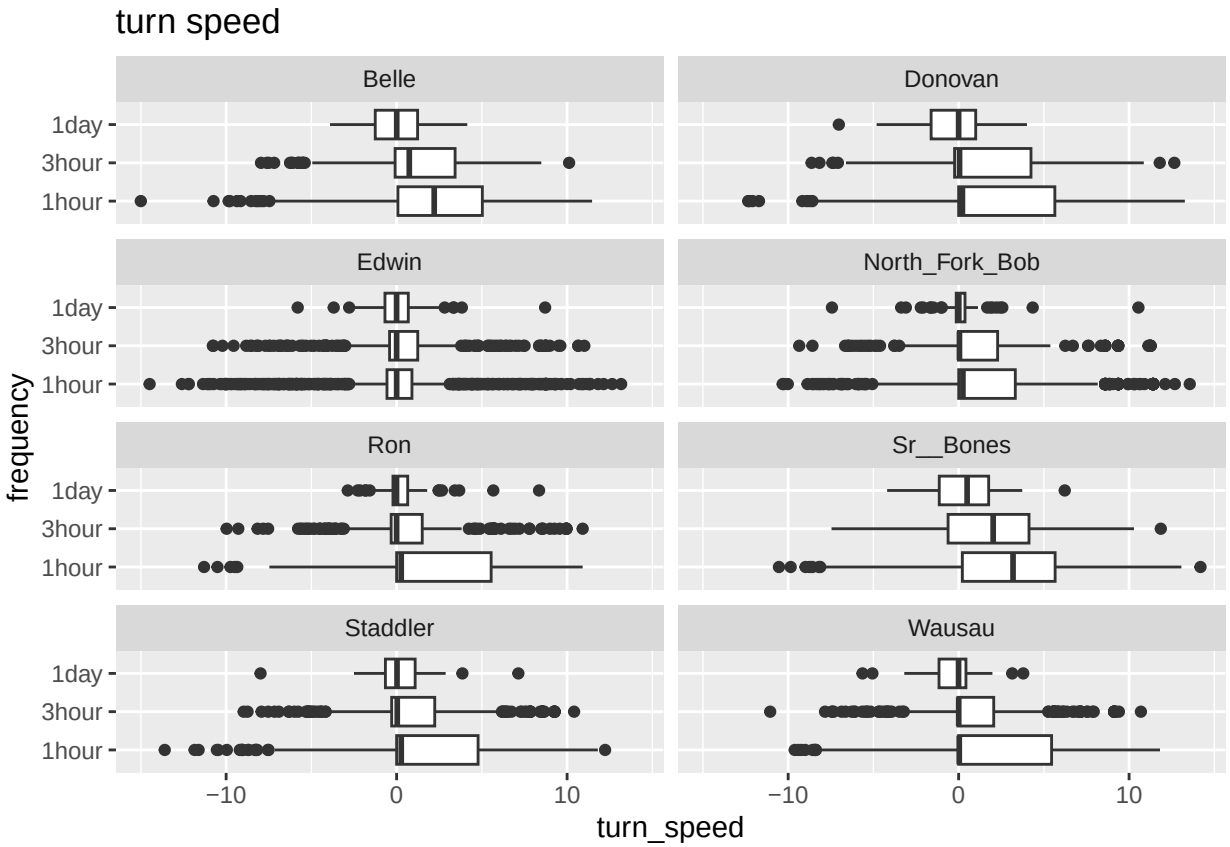

```
ggplot(comparison, aes(x=FPT, y = frequency)) +
  geom_boxplot() +
  ggtitle("FPT")+
  facet_wrap(~migrationEvent, nrow = 4)
```



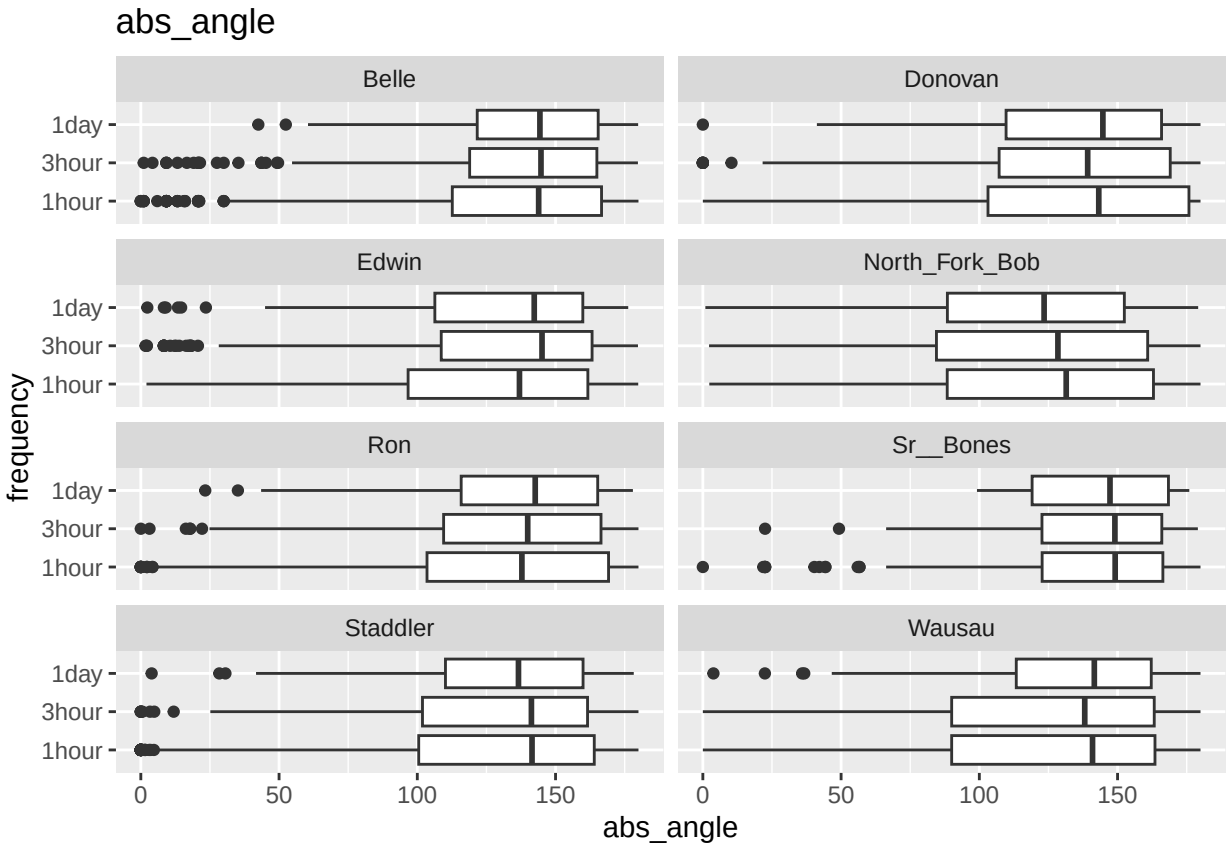

```
#by individual
ggplot(comparison, aes(x=speed, y = frequency)) +
  geom_boxplot() +
  ggtitle("Speed")+
  facet_wrap(~Indvidual, nrow = 4)
```



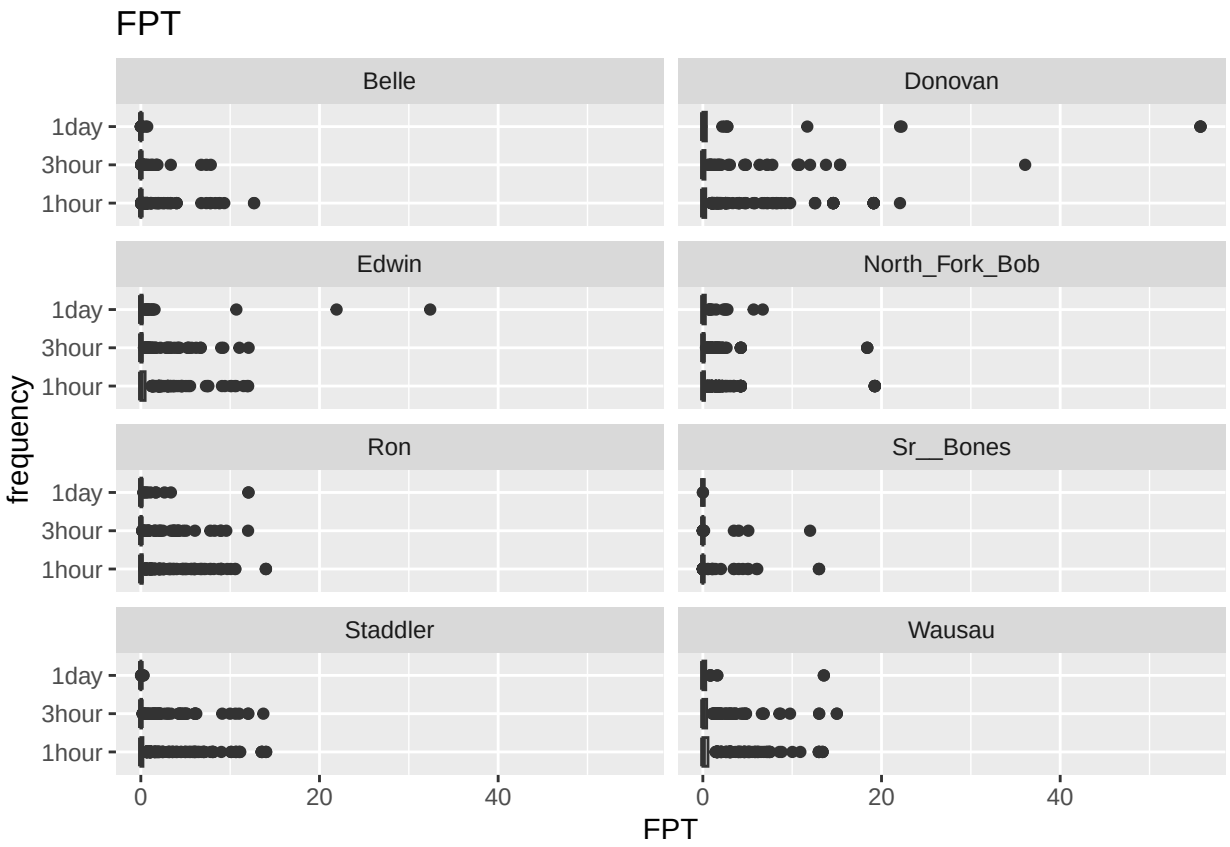

```
ggplot(comparison, aes(x=turn_speed, y = frequency)) +
  geom_boxplot() +
  ggtitle("turn speed")+
  facet_wrap(~Indvidual, nrow = 4)
```



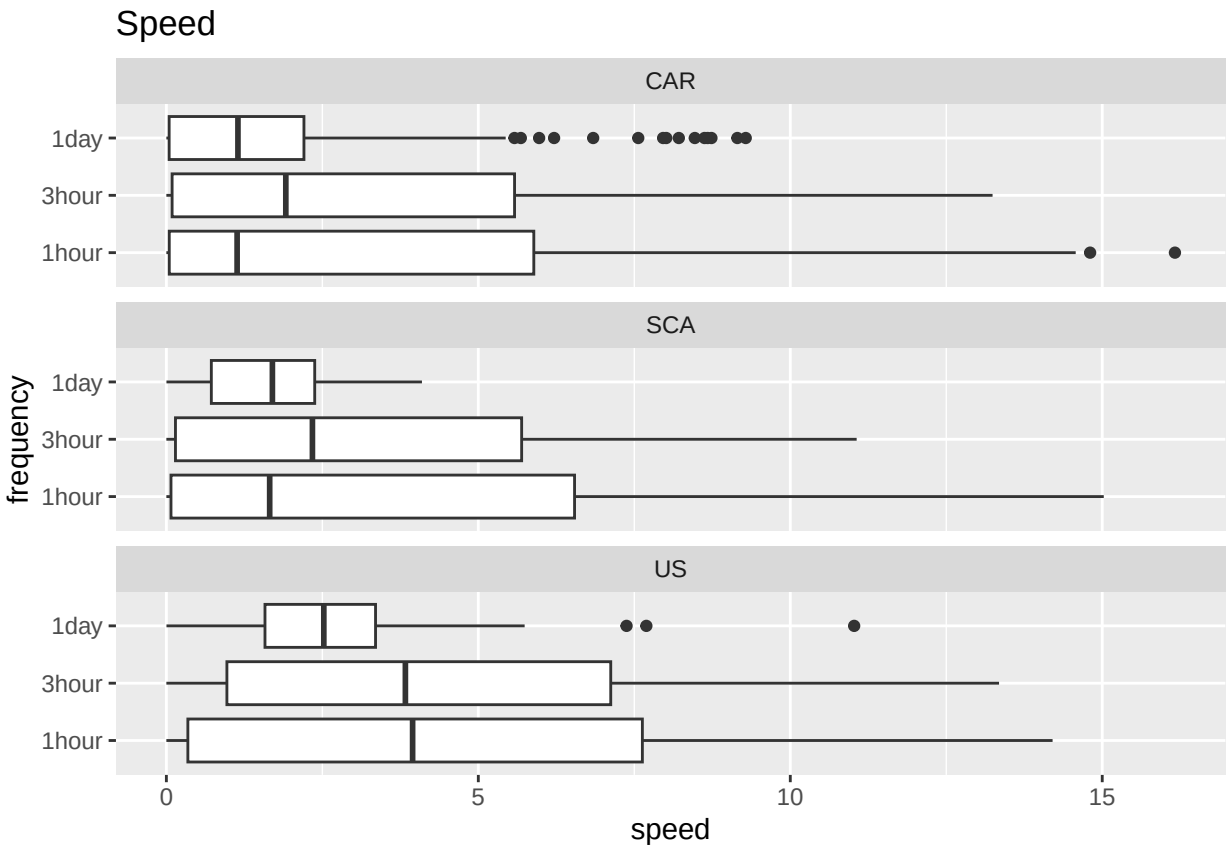

```
ggplot(comparison, aes(x=abs_angle, y = frequency)) +
  geom_boxplot() +
  ggtitle("abs_angle")+
  facet_wrap(~Indvidual, nrow = 4)
```



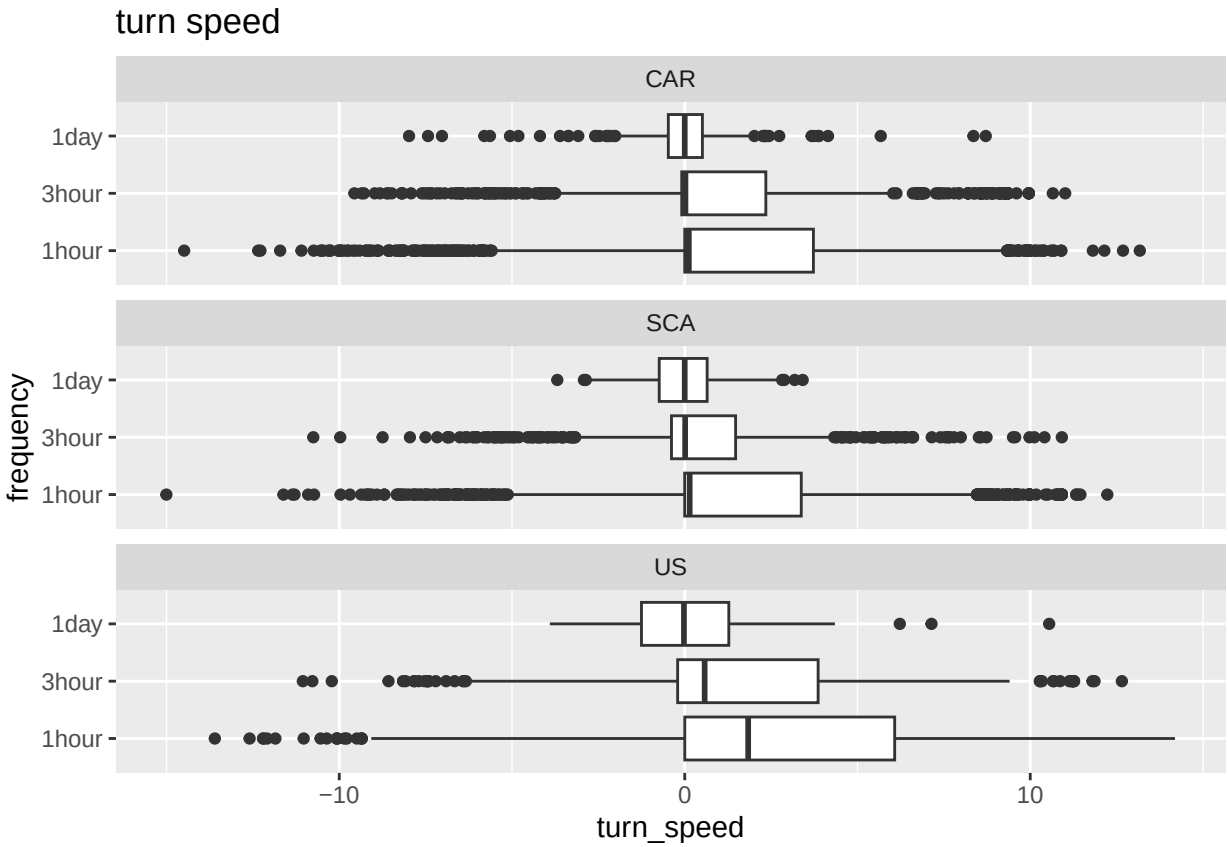

```
ggplot(comparison, aes(x=FPT, y = frequency)) +
  geom_boxplot() +
  ggtitle("FPT")+
  facet_wrap(~Indvidual, nrow = 4)
```



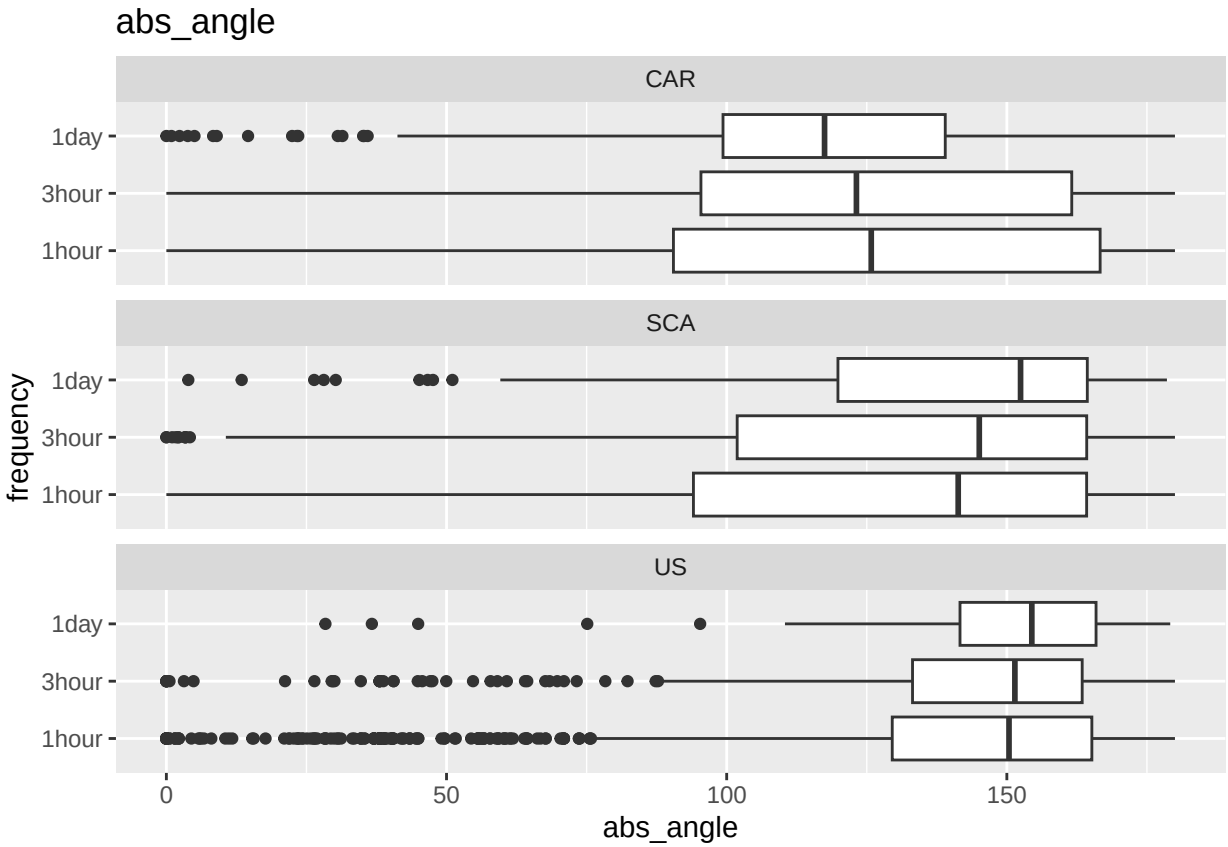

```
#by region
ggplot(comparison, aes(x=speed, y = frequency)) +
  geom_boxplot() +
  ggtitle("Speed")+
  facet_wrap(~Segment, nrow = 4)
```



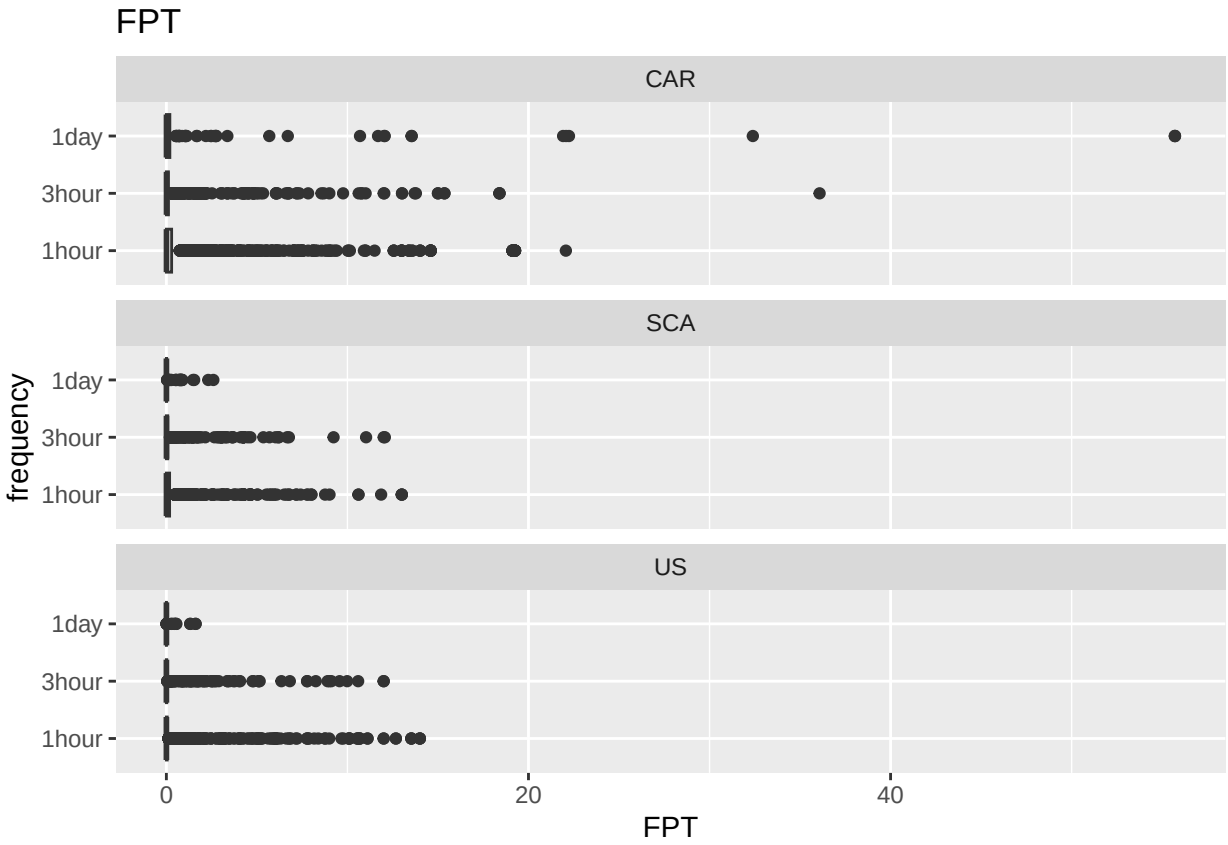

```
ggplot(comparison, aes(x=turn_speed, y = frequency)) +
  geom_boxplot() +
  ggtitle("turn speed")+
  facet_wrap(~Segment, nrow = 4)
```



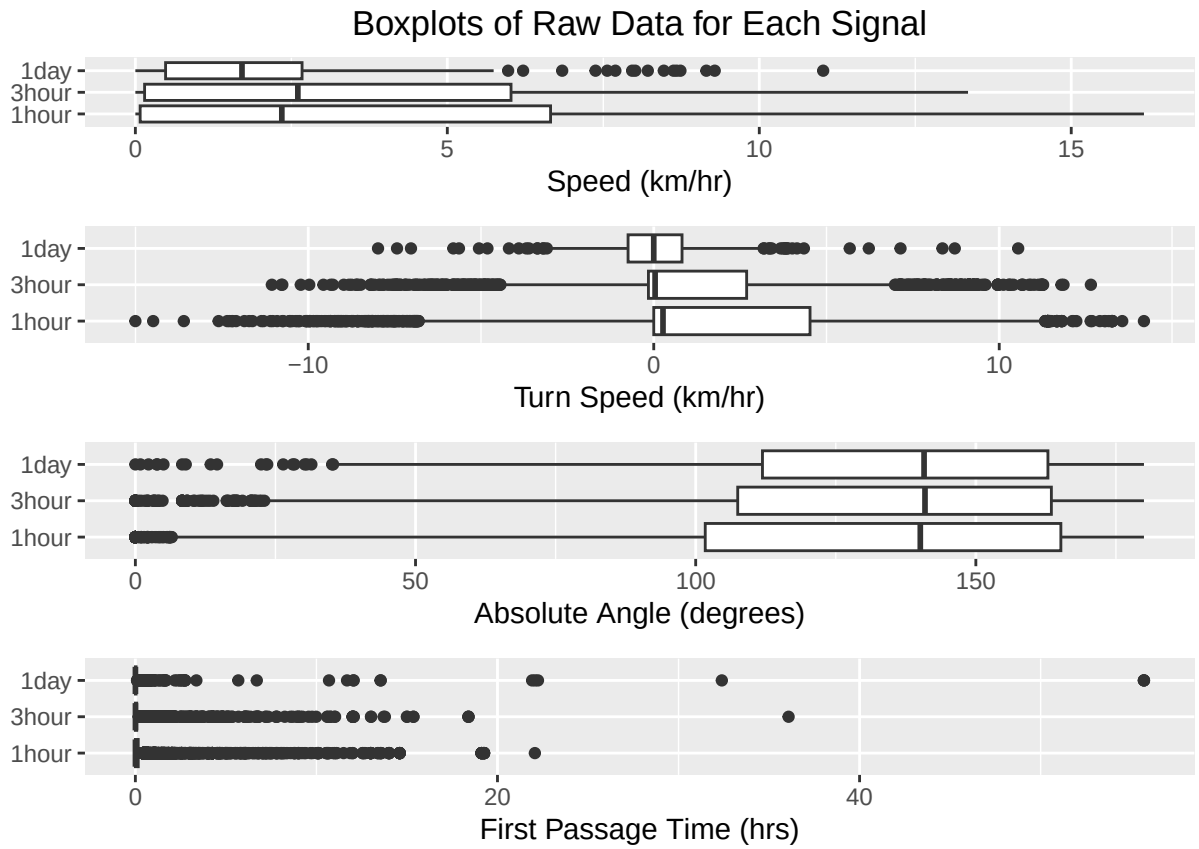

```
ggplot(comparison, aes(x=abs_angle, y = frequency)) +
  geom_boxplot() +
  ggtitle("abs_angle")+
  facet_wrap(~Segment, nrow = 4)
```




```
ggplot(comparison, aes(x=FPT, y = frequency)) +
  geom_boxplot() +
  ggtitle("FPT")+
  facet_wrap(~Segment, nrow = 4)
```




```
#overall
speed<-ggplot(comparison, aes(x=speed, y = frequency)) +
  geom_boxplot() +
  ggtitle("Boxplots of Raw Data for Each Signal") +
  ylab(" ") +
  xlab ("Speed (km/hr)") +
  theme(plot.title = element_text(hjust = 0.5))

TS<-ggplot(comparison, aes(x=turn_speed, y = frequency)) +
  geom_boxplot() +
  ylab(" ") +
  xlab("Turn Speed (km/hr)")

TA<-ggplot(comparison, aes(x=abs_angle, y = frequency)) +
  geom_boxplot() +
  ylab(" ") +
  xlab("Absolute Angle (degrees)")

FPT<-ggplot(comparison, aes(x=FPT, y = frequency)) +
  geom_boxplot() +
  ylab(" ") +
  xlab("First Passage Time (hrs)")

grid.arrange(speed, TS, TA, FPT, nrow = 4)
```




```
#absolute angle remains rather constant in mean and range, with some deviation in the
↪  quartiles. Most robust of the measurements. Speed range, mean, and quartiles all
↪  differ at each frequency. Differences in means is driven by differences in speed
↪  measurements with mean speed decreasing between 1-hour and 1-day. At 3-days mean
↪  speed is largest. The upper range of speed decreases consistently across time. Turn
↪  speed and FPT means remain similar across frequency. Turn speed upper and lower range
↪  decreases as frequency becomes coarser. First passage time upper range increases as
↪  frequency becomes courser. Turn speed influences the dispersion of data the most,
↪  while speed influences the mean.
```

Next find differences between group means so that I can relate real values to effect sizes and determine which signals are influencing differences.

```
# Dataframe of means for each migration event at each frequency
means_migrationEvent<-data.frame()
for (i in unique(comparison$frequency)) {
  x <- comparison %>%
  filter(frequency == i) %>%
  group_by(migrationEvent) %>%
  summarize(speed_mean = mean(speed),
            abs_angle_mean = mean(abs_angle),
            turn_speed_mean = mean(turn_speed),
            FPT_mean = mean(FPT))
  x$frequency <- paste(i)
  means_migrationEvent<-rbind(means_migrationEvent, x)
```

```
}

Diff_migrationEvent<- means_migrationEvent %>%
  group_by(frequency) %>%
  summarize(Speed_Max = max(speed_mean),
            Speed_Min = min(speed_mean),
            TS_Max = max(turn_speed_mean),
            TS_Min = min(turn_speed_mean),
            TA_Max = max(abs_angle_mean),
            TA_Min = min(abs_angle_mean),
            FPT_Max = max(FPT_mean),
            FPT_Min = min(FPT_mean),
            FPT_Dif = FPT_Max - FPT_Min,
            TA_Dif = TA_Max - TA_Min,
            TS_Dif = TS_Max - TS_Min,
            Speed_Dif = Speed_Max - Speed_Min
            )
Diff_migrationEvent[c("TS_Dif", "TA_Dif", "FPT_Dif", "Speed_Dif")]
```

```
## # A tibble: 3 x 4
##   TS_Dif TA_Dif FPT_Dif Speed_Dif
##    <dbl>  <dbl>   <dbl>     <dbl>
## 1   1.31   40.0    9.21      2.39
## 2   3.57   32.1    1.37      3.26
## 3   2.60   36.3    1.50      2.74
```

```
# Dataframe of means for each Individual
means_Individual<-data.frame()
for (i in unique(comparison$frequency)) {
  x <- comparison %>%
  filter(frequency == i) %>%
  group_by(Indvidual) %>%
  summarize(speed_mean = mean(speed),
            abs_angle_mean = mean(abs_angle),
            turn_speed_mean = mean(turn_speed),
            FPT_mean = mean(FPT))
  x$frequency <- paste(i)
  means_Individual<-rbind(means_Individual, x)
}

Diff_Individual<- means_Individual %>%
  group_by(frequency) %>%
  summarize(Speed_Max = max(speed_mean),
            Speed_Min = min(speed_mean),
            TS_Max = max(turn_speed_mean),
            TS_Min = min(turn_speed_mean),
            TA_Max = max(abs_angle_mean),
            TA_Min = min(abs_angle_mean),
            FPT_Max = max(FPT_mean),
            FPT_Min = min(FPT_mean),
            FPT_Dif = FPT_Max - FPT_Min,
            TA_Dif = TA_Max - TA_Min,
            TS_Dif = TS_Max - TS_Min,
```

```
            Speed_Dif = Speed_Max - Speed_Min
            )
Diff_Individual[c("TS_Dif", "TA_Dif", "FPT_Dif", "Speed_Dif")]
```

```
## # A tibble: 3 x 4
##   TS_Dif TA_Dif FPT_Dif Speed_Dif
##    <dbl>  <dbl>   <dbl>     <dbl>
## 1  0.667   26.4   5.56       1.54
## 2  2.99    21.7   0.829      1.68
## 3  1.74    23.9   0.886      1.75
```

```
# Dataframe of means for each region
means_Region<-data.frame()
for (i in unique(comparison$frequency)) {
  x <- comparison %>%
  filter(frequency == i) %>%
  group_by(Segment) %>%
  summarize(speed_mean = mean(speed),
            abs_angle_mean = mean(abs_angle),
            turn_speed_mean = mean(turn_speed),
            FPT_mean = mean(FPT))
  x$frequency <- paste(i)
  means_Region<-rbind(means_Region, x)
}

Diff_Region<- means_Region %>%
  group_by(frequency) %>%
  summarize(Speed_Max = max(speed_mean),
            Speed_Min = min(speed_mean),
            TS_Max = max(turn_speed_mean),
            TS_Min = min(turn_speed_mean),
            TA_Max = max(abs_angle_mean),
            TA_Min = min(abs_angle_mean),
            FPT_Max = max(FPT_mean),
            FPT_Min = min(FPT_mean),
            FPT_Dif = FPT_Max - FPT_Min,
            TA_Dif = TA_Max - TA_Min,
            TS_Dif = TS_Max - TS_Min,
            Speed_Dif = Speed_Max - Speed_Min
            )
Diff_Region[c("TS_Dif", "TA_Dif", "FPT_Dif", "Speed_Dif")]
```

```
## # A tibble: 3 x 4
##   TS_Dif TA_Dif FPT_Dif Speed_Dif
##    <dbl>  <dbl>   <dbl>     <dbl>
## 1  0.285   36.3   1.81      0.938
## 2  1.38    18.2   0.293     1.32
## 3  1.10    21.2   0.287     1.22
```

Find differences between ranges for some idea about the differences in dispersion.

```r
# Dataframe of means for each migration event at each frequency
range_migrationEvent<-data.frame()
for (i in unique(comparison$frequency)) {
  x <- comparison %>%
  filter(frequency == i) %>%
  group_by(migrationEvent) %>%
  summarize(speed_range = max(speed) - min(speed),
            abs_angle_range = max(abs_angle) - min(abs_angle),
            turn_speed_range = max(turn_speed) - min(turn_speed),
            FPT_range = max(FPT) - min(FPT))
  x$frequency <- paste(i)
  range_migrationEvent<-rbind(range_migrationEvent, x)
}

DiffR_migrationEvent<- range_migrationEvent %>%
  group_by(frequency) %>%
  summarize(Speed_Max = max(speed_range),
            Speed_Min = min(speed_range),
            TS_Max = max(turn_speed_range),
            TS_Min = min(turn_speed_range),
            TA_Max = max(abs_angle_range),
            TA_Min = min(abs_angle_range),
            FPT_Max = max(FPT_range),
            FPT_Min = min(FPT_range),
            FPT_DifR = FPT_Max - FPT_Min,
            TA_DifR = TA_Max - TA_Min,
            TS_DifR = TS_Max - TS_Min,
            Speed_DifR = Speed_Max - Speed_Min
            )
DiffR_migrationEvent[c("TS_DifR", "TA_DifR", "FPT_DifR", "Speed_DifR")]
```

```
## # A tibble: 3 x 4
##   TS_DifR TA_DifR FPT_DifR Speed_DifR
##     <dbl>   <dbl>    <dbl>      <dbl>
## 1    7.51   110.      55.7       5.59
## 2    7.06    21.9     21.9       5.18
## 3    6.86    71.2     35.9       4.78
```

```r
# Dataframe of means for each Individual
range_Ind<-data.frame()
for (i in unique(comparison$frequency)) {
  x <- comparison %>%
  filter(frequency == i) %>%
  group_by(Indvidual) %>%
  summarize(speed_range = max(speed) - min(speed),
            abs_angle_range = max(abs_angle) - min(abs_angle),
            turn_speed_range = max(turn_speed) - min(turn_speed),
            FPT_range = max(FPT) - min(FPT))
  x$frequency <- paste(i)
  range_Ind<-rbind(range_Ind, x)
}

DiffR_Ind<- range_Ind %>%
```

```
  group_by(frequency) %>%
  summarize(Speed_Max = max(speed_range),
            Speed_Min = min(speed_range),
            TS_Max = max(turn_speed_range),
            TS_Min = min(turn_speed_range),
            TA_Max = max(abs_angle_range),
            TA_Min = min(abs_angle_range),
            FPT_Max = max(FPT_range),
            FPT_Min = min(FPT_range),
            FPT_DifR = FPT_Max - FPT_Min,
            TA_DifR = TA_Max - TA_Min,
            TS_DifR = TS_Max - TS_Min,
            Speed_DifR = Speed_Max - Speed_Min
            )
DiffR_Ind[c("TS_DifR", "TA_DifR", "FPT_DifR", "Speed_DifR")]
```

```
## # A tibble: 3 x 4
##   TS_DifR TA_DifR FPT_DifR Speed_DifR
##     <dbl>   <dbl>    <dbl>      <dbl>
## 1    9.93  103.       55.7       3.07
## 2    6.23    2.33     10.1       4.35
## 3    3.72   23.5      28.2       2.55
```

```
# Dataframe of means for each region
range_region<-data.frame()
for (i in unique(comparison$frequency)) {
  x <- comparison %>%
  filter(frequency == i) %>%
  group_by(Segment) %>%
  summarize(speed_range = max(speed) - min(speed),
            abs_angle_range = max(abs_angle) - min(abs_angle),
            turn_speed_range = max(turn_speed) - min(turn_speed),
            FPT_range = max(FPT) - min(FPT))
  x$frequency <- paste(i)
  range_region<-rbind(range_region, x)
}

DiffR_region<- range_region %>%
  group_by(frequency) %>%
  summarize(Speed_Max = max(speed_range),
            Speed_Min = min(speed_range),
            TS_Max = max(turn_speed_range),
            TS_Min = min(turn_speed_range),
            TA_Max = max(abs_angle_range),
            TA_Min = min(abs_angle_range),
            FPT_Max = max(FPT_range),
            FPT_Min = min(FPT_range),
            FPT_DifR = FPT_Max - FPT_Min,
            TA_DifR = TA_Max - TA_Min,
            TS_DifR = TS_Max - TS_Min,
            Speed_DifR = Speed_Max - Speed_Min
            )
DiffR_region[c("TS_DifR", "TA_DifR", "FPT_DifR", "Speed_DifR")]
```

```
## # A tibble: 3 x 4
##   TS_DifR TA_DifR FPT_DifR Speed_DifR
##     <dbl>   <dbl>    <dbl>      <dbl>
## 1   9.59     29.2    54.1        6.93
## 2   0.554     0       9.06       1.96
## 3   3.13      0      24.1        2.28
```

Scale/normalize data for PERMANOVA. I scaled data since I am using multivariate statistics that require distance matrixs. Distance matrices are heavily influenced by variable measurement sizes. Measurements at a larger scale (such as between 50 to 200) will overshadow measurements at a lower scale (such as 0 - 1).

```
Norm_1hour<- as.data.frame(scale(Samp_Subset1hour[2:5]))
Norm_1hour<-cbind(Norm_1hour, Samp_Subset1hour$trackId, Samp_Subset1hour$Indvidual,
↪  Samp_Subset1hour$`Migration Year`, Samp_Subset1hour$Segment,
↪  Samp_Subset1hour$migrationEvent, Samp_Subset1hour$Starting_Location)
colnames(Norm_1hour)<-c("speed", "abs_angle", "turn_speed", "FPT", "SegmentID",
↪  "Individual", "Mig_Year", "Segment", "migrationEvent", "Starting_Location")

Norm_3hour<- as.data.frame(scale(Samp_Subset3hour[2:5]))
Norm_3hour<-cbind(Norm_3hour, Samp_Subset3hour$trackId, Samp_Subset3hour$Indvidual,
↪  Samp_Subset3hour$`Migration Year`, Samp_Subset3hour$Segment,
↪  Samp_Subset3hour$migrationEvent, Samp_Subset3hour$Starting_Location)
colnames(Norm_3hour)<-c("speed", "abs_angle", "turn_speed", "FPT", "SegmentID",
↪  "Individual", "Mig_Year", "Segment", "migrationEvent", "Starting_Location")

Norm_1day<- as.data.frame(scale(Samp_Subset1day[2:5]))
Norm_1day<-cbind(Norm_1day, Samp_Subset1day$trackId, Samp_Subset1day$Indvidual,
↪  Samp_Subset1day$`Migration Year`, Samp_Subset1day$Segment,
↪  Samp_Subset1day$migrationEvent, Samp_Subset1day$Starting_Location)
colnames(Norm_1day)<-c("speed", "abs_angle", "turn_speed", "FPT", "SegmentID",
↪  "Individual", "Mig_Year", "Segment", "migrationEvent", "Starting_Location")
```

Next, I will look at the distribution of normalized variables/characteristics using histograms and Q-Q plots. A non-normal distribution requires non-parametric approaches.

```
#1-hour
hist(Norm_1hour$speed)
hist(Norm_1hour$abs_angle)
hist(Norm_1hour$turn_speed)
hist(Norm_1hour$FPT)
qqPlot(Norm_1hour$speed)
```

```
## [1] 670 405
```

```
qqPlot(Norm_1hour$abs_angle)
```

```
## [1] 164 167
```

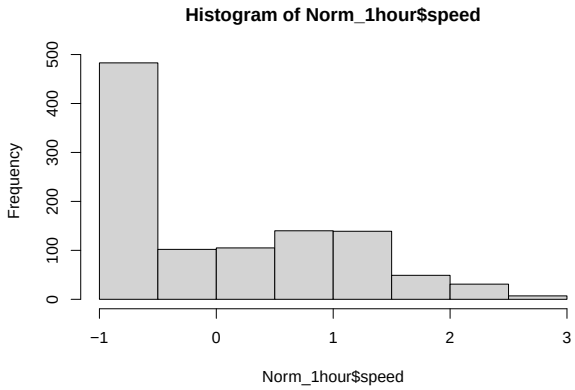
```
qqPlot(Norm_1hour$turn_speed)
```

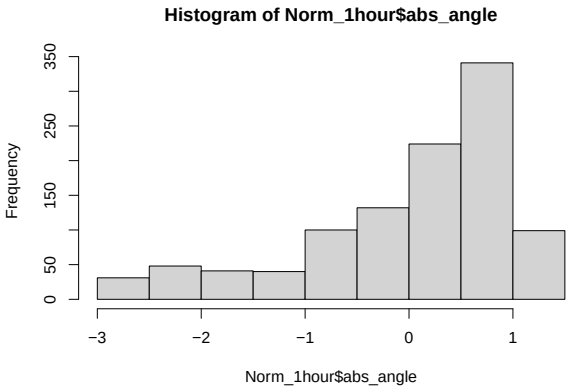
```
## [1] 279 267
```

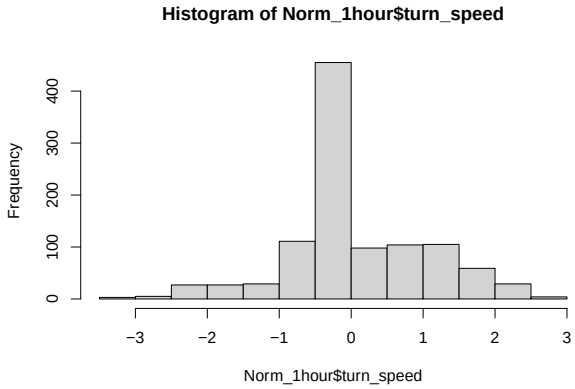
```
qqPlot(Norm_1hour$FPT)
```

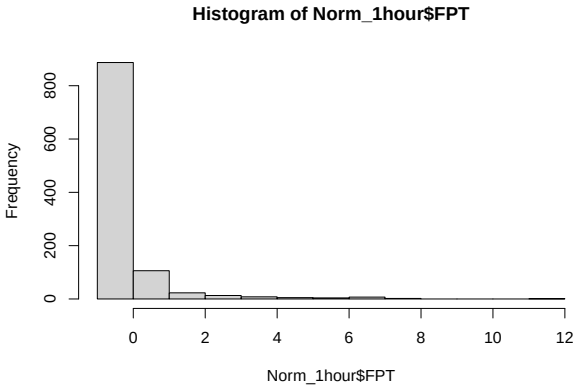
```
## [1] 240 239
```

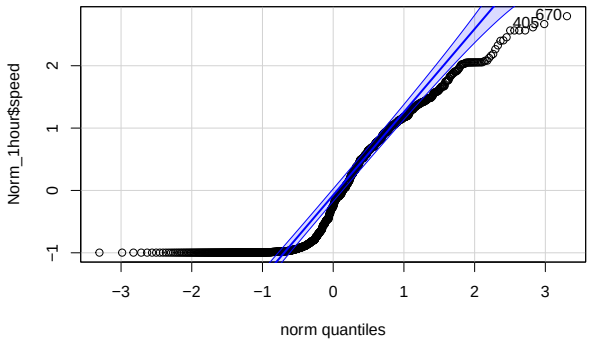
```
#non-normal for all variables
```

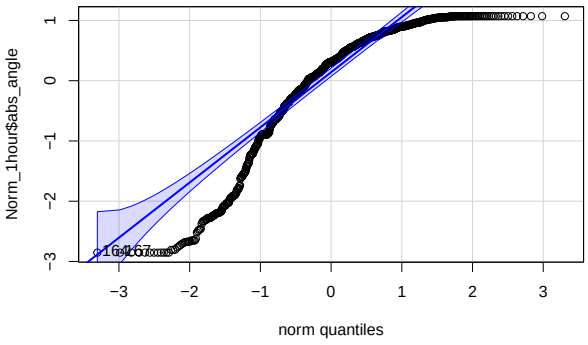


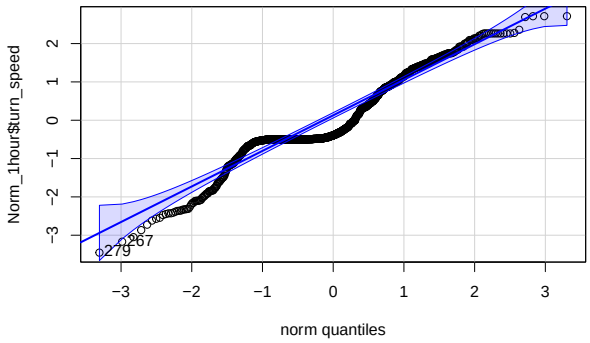


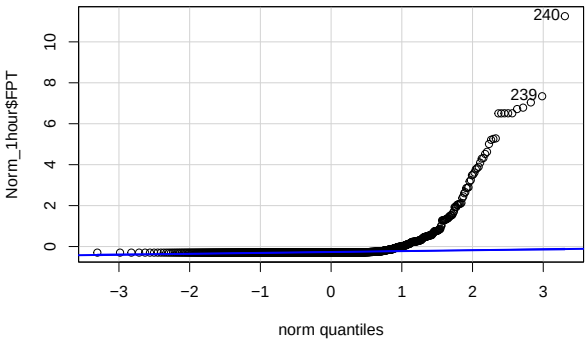













```
#3-hours
hist(Norm_3hour$speed)
hist(Norm_3hour$abs_angle)
hist(Norm_3hour$turn_speed)
hist(Norm_3hour$FPT)
qqPlot(Norm_3hour$speed)
```

```
## [1]  95 112
```

```
qqPlot(Norm_3hour$abs_angle)
```

```
## [1]  50 105
```

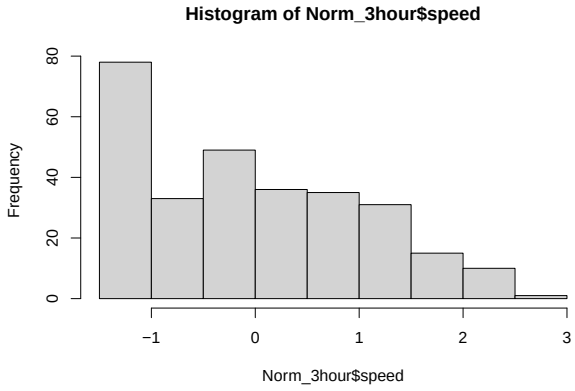
```
qqPlot(Norm_3hour$turn_speed)
```

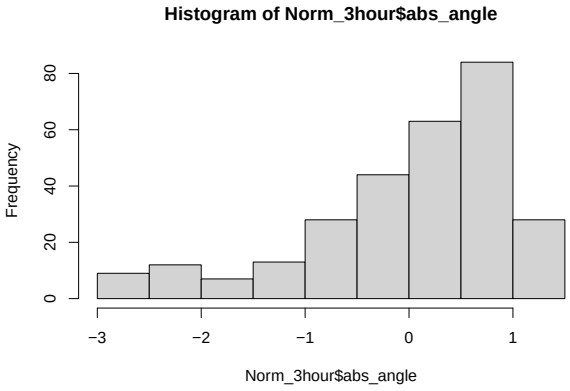
```
## [1] 257  93
```

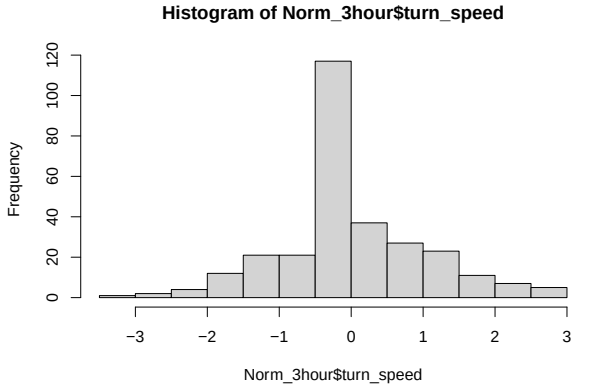
```
qqPlot(Norm_3hour$FPT)
```

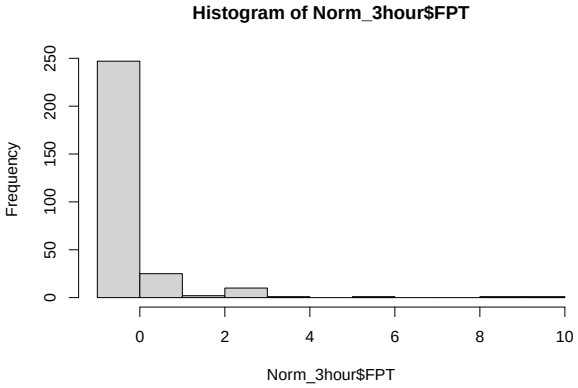
```
## [1] 279 244
```

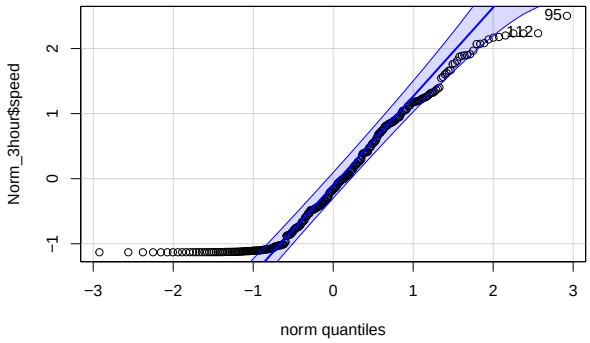
```
#non-normal for most variables. Speed looking a bit more normal than the rest.
```

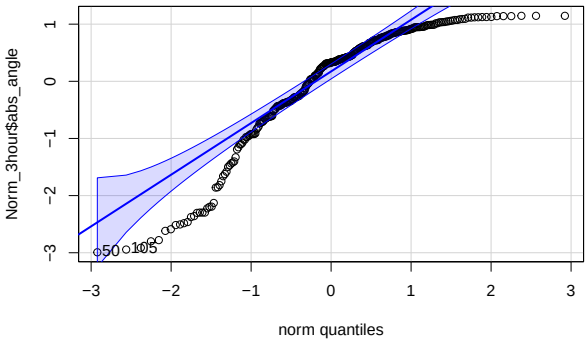


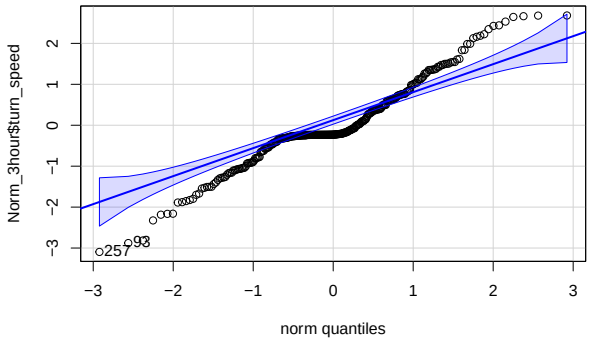


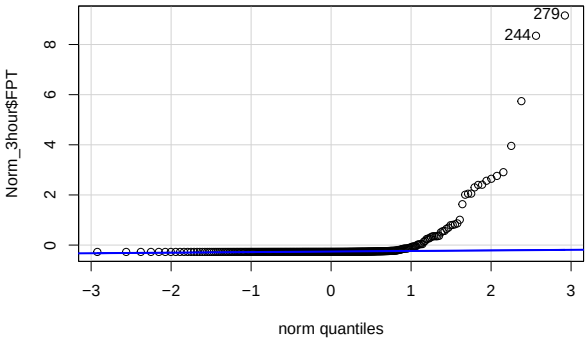













```
#1-day
hist(Norm_1day$speed)
hist(Norm_1day$abs_angle)
hist(Norm_1day$turn_speed)
hist(Norm_1day$FPT)
qqPlot(Norm_1day$speed)
```

```
## [1] 47 31
```

```
qqPlot(Norm_1day$abs_angle)
```

```
## [1] 20 15
```

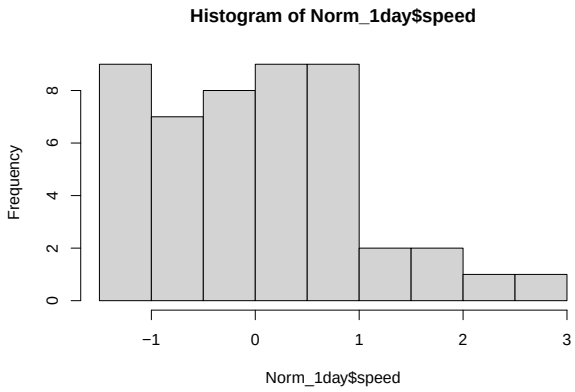
```
qqPlot(Norm_1day$turn_speed)
```

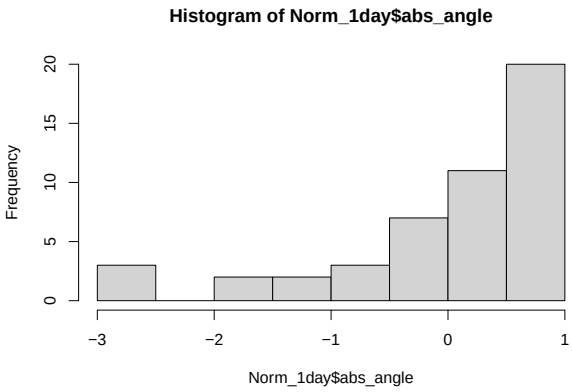
```
## [1] 47 19
```

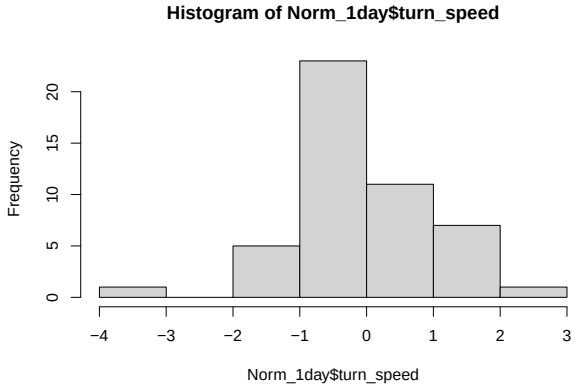
```
qqPlot(Norm_1day$FPT)
```

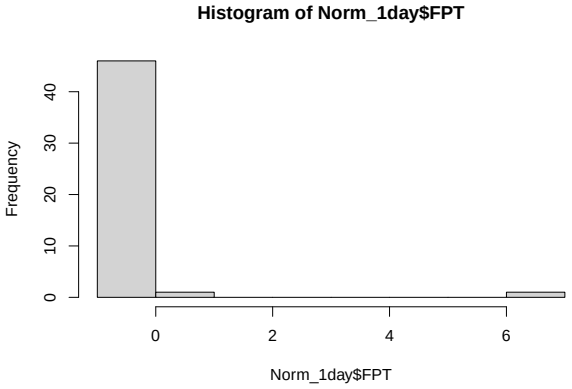
```
## [1] 14 31
```

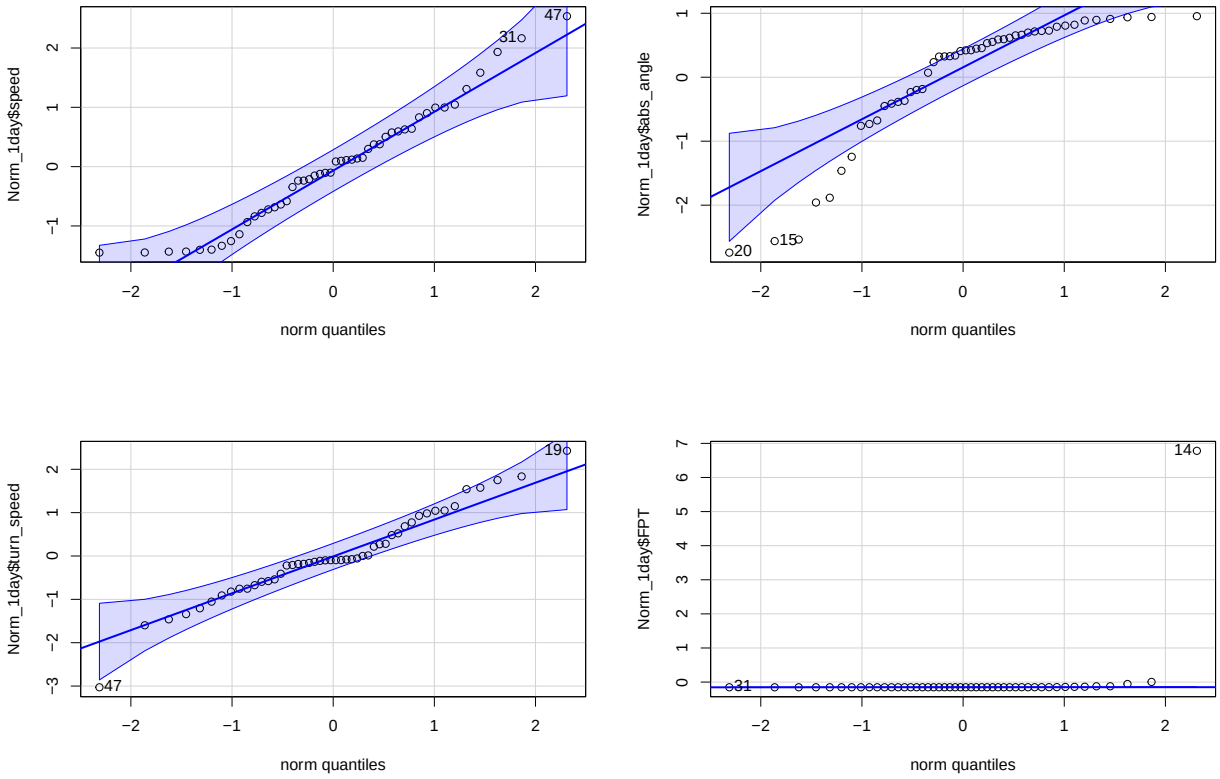
```
#absolute angle non-normal, although the data is becoming more normally distributed with
↪  coarser sampling frequency.
```












```
#Multivariate normality using modified Shapiro-Wilk's test
mshapiro_test(Norm_1hour[1:4])
```

```
## # A tibble: 1 x 2
##   statistic  p.value
##       <dbl>    <dbl>
## 1     0.498 2.08e-47
```

```
mshapiro_test(Norm_3hour[1:4])
```

```
## # A tibble: 1 x 2
##   statistic  p.value
##       <dbl>    <dbl>
## 1     0.375 2.84e-30
```

```
mshapiro_test(Norm_1day[1:4])
```

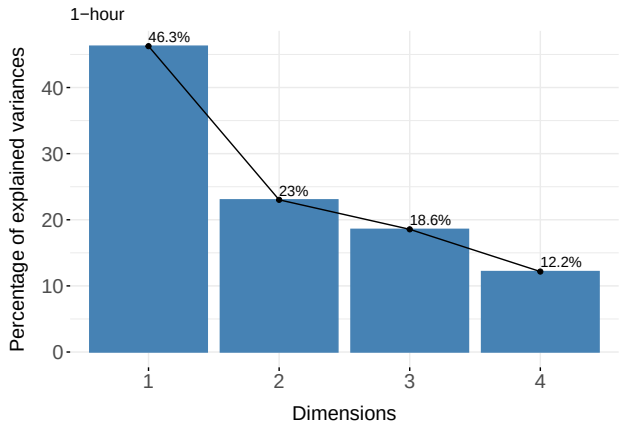
```
## # A tibble: 1 x 2
##   statistic  p.value
##       <dbl>    <dbl>
## 1     0.141 3.05e-15
```

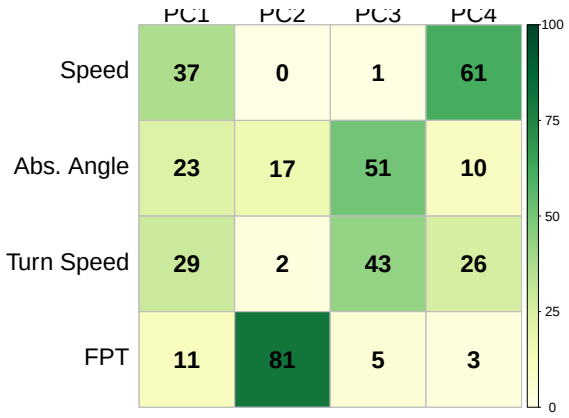
```
# reject null hypothesis of normal distribution because of low p-value for all
↪  frequencies
```

PCA - subset 1

```
#1-hour
EventsPCA_1hour<-PCA(Norm_1hour[,1:4], scale = F, graph = F)
fviz_screeplot(EventsPCA_1hour, addlabels = TRUE, title = "1-hour") +
  theme(legend.position = "none",
        axis.title.x = element_text(margin = margin(t = 10), size = 15),
        axis.title.y = element_text(margin = margin(r = 10), size = 15),
        axis.text = element_text(size = 15))
#Loadings plot, size = 938 x 697
LoadingPlot_1hour<-fviz_pca_var(EventsPCA_1hour,
             col.var =  "cos2", arrowsize = 1.5, repel = T, #change to repel = T to add
             ↪  labels to graph
             title = "1-hour") +
  theme(legend.position = "none", legend.key.height = unit(2, 'cm'),
        axis.title.x = element_text(margin = margin(t = 10), size = 15),
        axis.title.y = element_text(margin = margin(r = 10), size = 15),
        axis.text = element_text(size = 15),
        plot.title = element_text(hjust = 0.5)) +
  scale_color_gradient2(low = "red", mid = "blue", high = "green",
                        midpoint = 0.5, limits = c(0,1)) +
  xlim(c(-1,1)) +
  ylim(c(-1,1))
#Contribution of variables to each PC
contrib_1hour<-get_pca_var(EventsPCA_1hour)
colnames(contrib_1hour$contrib)<- c("PC1", "PC2", "PC3", "PC4")
rownames(contrib_1hour$contrib)<- c("Speed", "Abs. Angle", "Turn Speed", "FPT")
corrplot(contrib_1hour$contrib, method = 'color',
         is.corr = F,
         addgrid.col = "grey",
         col = COL1('YlGn'),
         addCoef.col = 'black',
         tl.col = "black",
         tl.srt = 0,
         tl.offset = 0.5,
         mar = c(0.5, 0.5, 0.5, 0.5),
         cl.length = 5,
         col.lim=c(0,100),
         tl.cex = 1.5,
         number.cex = 1.5,
         number.digits = 0,
         cl.pos="r"
         )
#Score Plot for migration events
ScorePlot_1hour_ME <-fviz_pca_ind(EventsPCA_1hour,
             geom.ind = "none",
             palette = colors_Events,
             col.ind = as.factor(Norm_1hour$migrationEvent),
             addEllipses = T,
             mean.point = T,
```

```
              ellipse.alpha = 0,
              mean.point.size = 2,
              scaleshape = "none",
              title = "1-hour"
              ) +
  theme(legend.position = "none",
        axis.title.x = element_text(margin = margin(t = 10), size = 15),
        axis.title.y = element_text(margin = margin(r = 10), size = 15),
        axis.text = element_text(size = 15)) +
  guides(fill=guide_legend(ncol = 2)) +
  scale_shape_identity() +
  scale_shape_manual(values = c(64,64,36,36,35,35,17,17,7,7,15,15,16,16,8,8))

#Score plot for individuals
ScorePlot_1hour_Ind<-fviz_pca_ind(EventsPCA_1hour,
              geom.ind = "none",
              palette = colors_Ind,
              col.ind = as.factor(Norm_1hour$Individual),
              addEllipses = T,
              mean.point = T,
              ellipse.alpha = 0,
              mean.point.size = 2,
              scaleshape = "none",
              title = "1-hour"
              ) +
  theme(legend.position = "none", #change to bottom or right to add legend
        axis.title.x = element_text(margin = margin(t = 10), size = 15),
        axis.title.y = element_text(margin = margin(r = 10), size = 15),
        axis.text = element_text(size = 15)) +
  guides(color = guide_legend(nrow = 4))+
  scale_shape_identity() +
  scale_shape_manual(values = c(64,36,35,17,7,15,16,8))

#Score plot for regions
ScorePlot_1hour_Region<-fviz_pca_ind(EventsPCA_1hour,
              geom.ind = "none",
              palette = c("#3816E6", "#1ABF51", "#2CD2B6"),
              col.ind = as.factor(Norm_1hour$Segment),
              addEllipses = T,
              mean.point = T,
              ellipse.alpha = 0,
              mean.point.size = 4,
              scaleshape = "none",
              title = "1-hour"
              ) +
  theme(legend.position = "none",
        axis.title.x = element_text(margin = margin(t = 10), size = 15),
        axis.title.y = element_text(margin = margin(r = 10), size = 15),
        axis.text = element_text(size = 15)) +
  guides(fill=guide_legend(ncol = 3)) +
  scale_shape_identity() +
  scale_shape_manual(values = c(15,16,17))
```

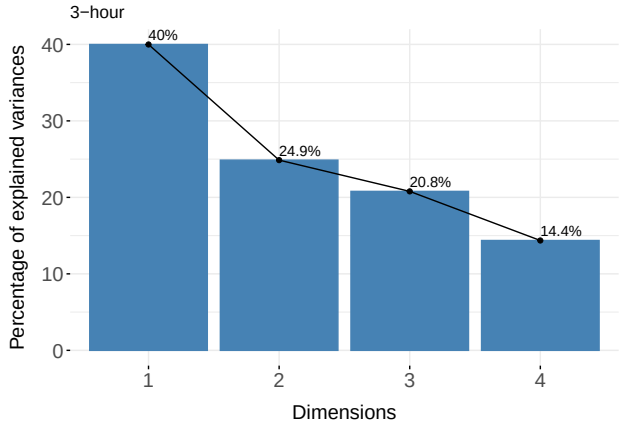


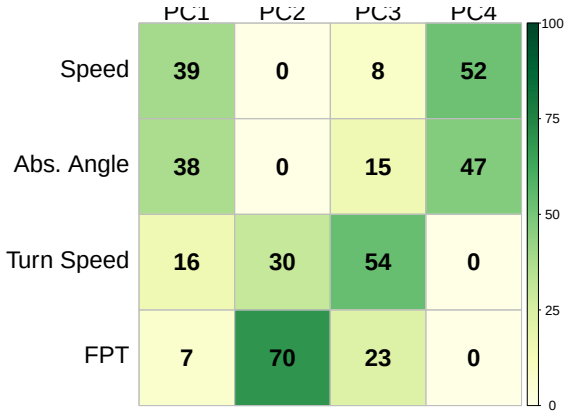



```
EventsPCA_3hour<-PCA(Norm_3hour[,1:4], scale = F, graph = F)
fviz_screeplot(EventsPCA_3hour, addlabels = TRUE, title = "3-hour") +
  theme(legend.position = "none",
        axis.title.x = element_text(margin = margin(t = 10), size = 15),
        axis.title.y = element_text(margin = margin(r = 10), size = 15),
        axis.text = element_text(size = 15))
#Loadings plot
LoadingPlot_3hour<-fviz_pca_var(EventsPCA_3hour,
             col.var =  "cos2", arrowsize = 1.5, repel = T,
             title = "3-hours") +
  theme(legend.position = "none",
        axis.title.x = element_text(margin = margin(t = 10), size = 15),
        axis.title.y = element_text(margin = margin(r = 10), size = 15),
        axis.text = element_text(size = 15),
        plot.title = element_text(hjust = 0.5)) +
  scale_color_gradient2(low = "red", mid = "blue", high = "green",
                        midpoint = 0.5, limits = c(0,1)) +
  xlim(c(-1,1)) +
  ylim(c(-1,1))
#Contribution of variables to each PC
contrib_3hour<-get_pca_var(EventsPCA_3hour)
colnames(contrib_3hour$contrib)<- c("PC1", "PC2", "PC3", "PC4")
rownames(contrib_3hour$contrib)<- c("Speed", "Abs. Angle", "Turn Speed", "FPT")
corrplot(contrib_3hour$contrib, method = 'color',
         is.corr = F,
         addgrid.col = "grey",
         col = COL1('YlGn'),
         addCoef.col = 'black',
         tl.col = "black",
         tl.srt = 0,
         tl.offset = 0.5,
         mar = c(0.5, 0.5, 0.5, 0.5),
         cl.length = 5,
         col.lim=c(0,100),
         tl.cex = 1.5,
         number.cex = 1.5,
         number.digits = 0,
         cl.pos="r"
         )
```

```
#Events score plot
ScorePlot_3hour_ME<-fviz_pca_ind(EventsPCA_3hour,
             geom.ind = "none",
             palette = colors_Events,
             col.ind = as.factor(Norm_3hour$migrationEvent),
             addEllipses = T,
             mean.point = T,
             ellipse.alpha = 0,
             mean.point.size = 2,
             scaleshape = "none",
             title = "3-hour"
             ) +
  theme(legend.position = "none",
        axis.title.x = element_text(margin = margin(t = 10), size = 15),
        axis.title.y = element_text(margin = margin(r = 10), size = 15),
        axis.text = element_text(size = 15)) +
  scale_shape_identity() +
  scale_shape_manual(values = c(64,64,36,36,35,35,17,17,7,7,15,15,16,16,8,8))

#Individual Score Plot
ScorePlot_3hour_Ind<-fviz_pca_ind(EventsPCA_3hour,
             geom.ind = "none",
             palette = colors_Ind,
             col.ind = as.factor(Norm_3hour$Individual),
             addEllipses = T,
             mean.point = T,
             ellipse.alpha = 0,
             mean.point.size = 2,
             scaleshape = "none",
             title = "3-hour"
             ) +
  theme(legend.position = "none",
        axis.title.x = element_text(margin = margin(t = 10), size = 15),
        axis.title.y = element_text(margin = margin(r = 10), size = 15),
        axis.text = element_text(size = 15)) +
  scale_shape_identity() +
  scale_shape_manual(values = c(64,36,35,17,7,15,16,8))

#Regions score plot
ScorePlot_3hour_Region<-fviz_pca_ind(EventsPCA_3hour,
             geom.ind = "none",
             palette = c("#3816E6", "#1ABF51", "#2CD2B6"),
             col.ind = as.factor(Norm_3hour$Segment),
             addEllipses = T,
             mean.point = T,
             ellipse.alpha = 0,
             mean.point.size = 4,
             scaleshape = "none",
             title = "3-hour"
             ) +
  theme(legend.position = "none",
        axis.title.x = element_text(margin = margin(t = 10), size = 15),
        axis.title.y = element_text(margin = margin(r = 10), size = 15),
```

```
        axis.text = element_text(size = 15)) +
  scale_shape_identity() +
  scale_shape_manual(values = c(15,16,17))
```

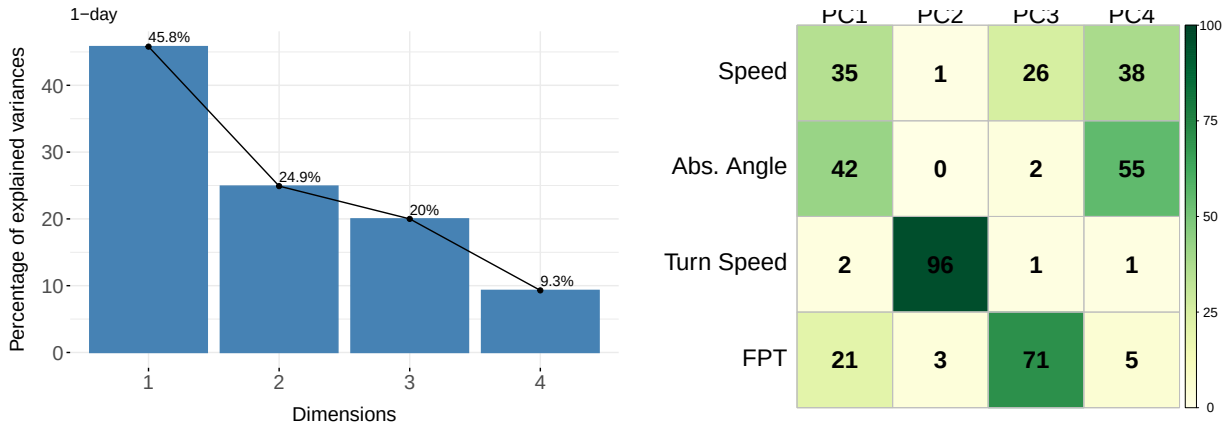





```
#1-day
EventsPCA_1day<-PCA(Norm_1day[,1:4], scale = F, graph = F)
fviz_screeplot(EventsPCA_1day, addlabels = TRUE, title = "1-day") +
  theme(legend.position = "none",
        axis.title.x = element_text(margin = margin(t = 10), size = 15),
        axis.title.y = element_text(margin = margin(r = 10), size = 15),
        axis.text = element_text(size = 15))
#Loadings plot
LoadingPlot_1day<-fviz_pca_var(EventsPCA_1day,
             col.var =  "cos2", arrowsize = 1.5, repel = T,
             title = "1-day") +
  theme(legend.position = "none",
        axis.title.x = element_text(margin = margin(t = 10), size = 15),
        axis.title.y = element_text(margin = margin(r = 10), size = 15),
        axis.text = element_text(size = 15),
        plot.title = element_text(hjust = 0.5)) +
  scale_color_gradient2(low = "red", mid = "blue", high = "green",
                        midpoint = 0.5, limits = c(0,1))+
  xlim(c(-1,1)) +
  ylim(c(-1,1))
#Contribution of variables to each PC
contrib_1day<-get_pca_var(EventsPCA_1day)
colnames(contrib_1day$contrib)<- c("PC1", "PC2", "PC3", "PC4")
rownames(contrib_1day$contrib)<- c("Speed", "Abs. Angle", "Turn Speed", "FPT")
corrplot(contrib_1day$contrib, method = 'color',
         is.corr = F,
         addgrid.col = "grey",
         col = COL1('YlGn'),
         addCoef.col = 'black',
         tl.col = "black",
         tl.srt = 0,
         tl.offset = 0.5,
         mar = c(0.5, 0.5, 0.5, 0.5),
         cl.length = 5,
```

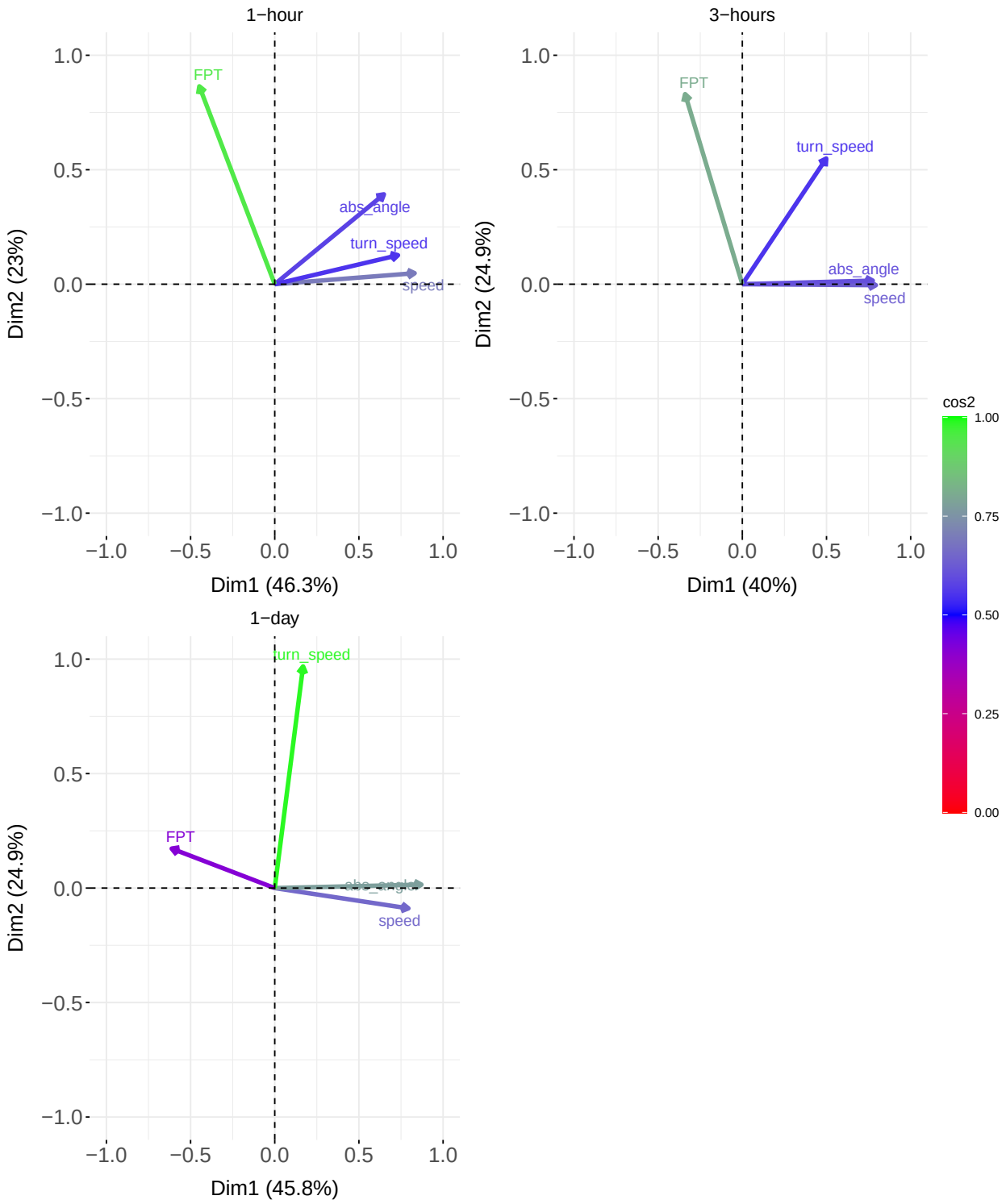
```
         col.lim=c(0,100),
         tl.cex = 1.5,
         number.cex = 1.5,
         number.digits = 0,
         cl.pos="r"
         )
#Event score plots
ScorePlot_1day_ME<-fviz_pca_ind(EventsPCA_1day,
             geom.ind = "point",
             palette = colors_Events,
             col.ind = as.factor(Norm_1day$migrationEvent),
             mean.point = T,
             mean.point.size = 3,
             scaleshape = "none",
             title = "1-day"
             ) +
  theme(legend.position = "none",
        axis.title.x = element_text(margin = margin(t = 10), size = 15),
        axis.title.y = element_text(margin = margin(r = 10), size = 15),
        axis.text = element_text(size = 15)) +
  guides(color=guide_legend(nrow=8,byrow=TRUE)) +
  scale_shape_identity() +
  scale_shape_manual(values = c(64,64,36,36,35,35,17,17,7,7,15,15,16,16,8,8))

#Individual Score Plot
ScorePlot_1day_Ind<-fviz_pca_ind(EventsPCA_1day,
             geom.ind = "none",
             palette = colors_Ind,
             col.ind = as.factor(Norm_1day$Individual),
             addEllipses = T,
             mean.point = T,
             ellipse.alpha = 0,
             mean.point.size = 2,
             scaleshape = "none",
             title = "1-day"
             ) +
  theme(legend.position = "none",
        axis.title.x = element_text(margin = margin(t = 10), size = 15),
        axis.title.y = element_text(margin = margin(r = 10), size = 15),
        axis.text = element_text(size = 15)) +
  scale_shape_identity() +
  scale_shape_manual(values = c(64,36,35,17,7,15,16,8))

#region score plots
ScorePlot_1day_Region<-fviz_pca_ind(EventsPCA_1day,
             geom.ind = "none",
             palette = c("#3816E6", "#1ABF51", "#2CD2B6"),
             col.ind = as.factor(Norm_1day$Segment),
             addEllipses = T,
             mean.point = T,
             ellipse.alpha = 0,
             mean.point.size = 4,
             scaleshape = "none",
```

```
            title = "1-day"
            ) +
  theme(legend.position = "none",
        axis.title.x = element_text(margin = margin(t = 10), size = 15),
        axis.title.y = element_text(margin = margin(r = 10), size = 15),
        axis.text = element_text(size = 15)) +
  scale_shape_identity() +
  scale_shape_manual(values = c(15,16,17))
```

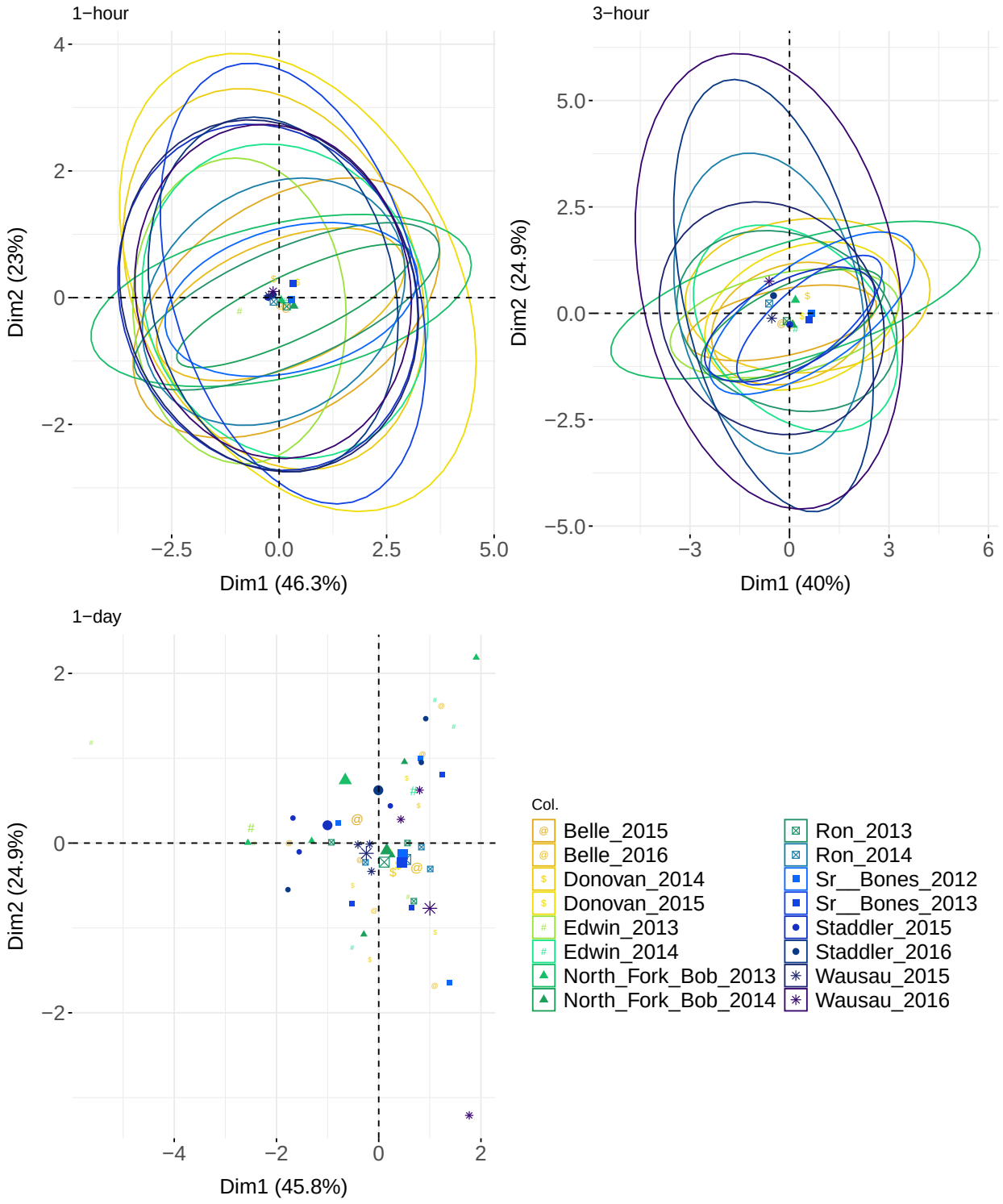


Display plots for thesis paper. Code for extracting legends not included.

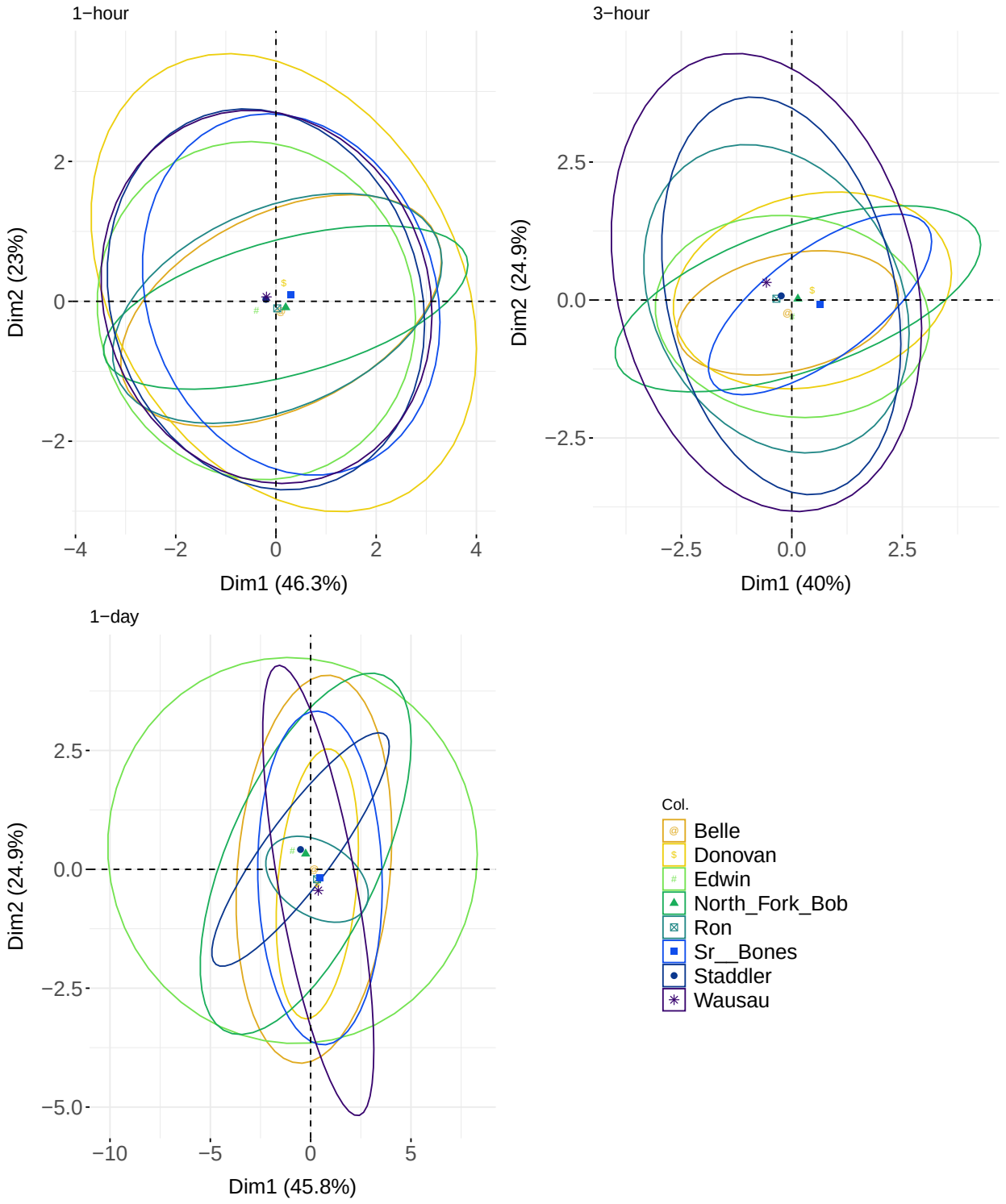
```
ggarrange(LoadingPlot_1hour, LoadingPlot_3hour, LoadingPlot_1day, ncol=2, nrow=2,
↪  common.legend = TRUE, legend="right")
```



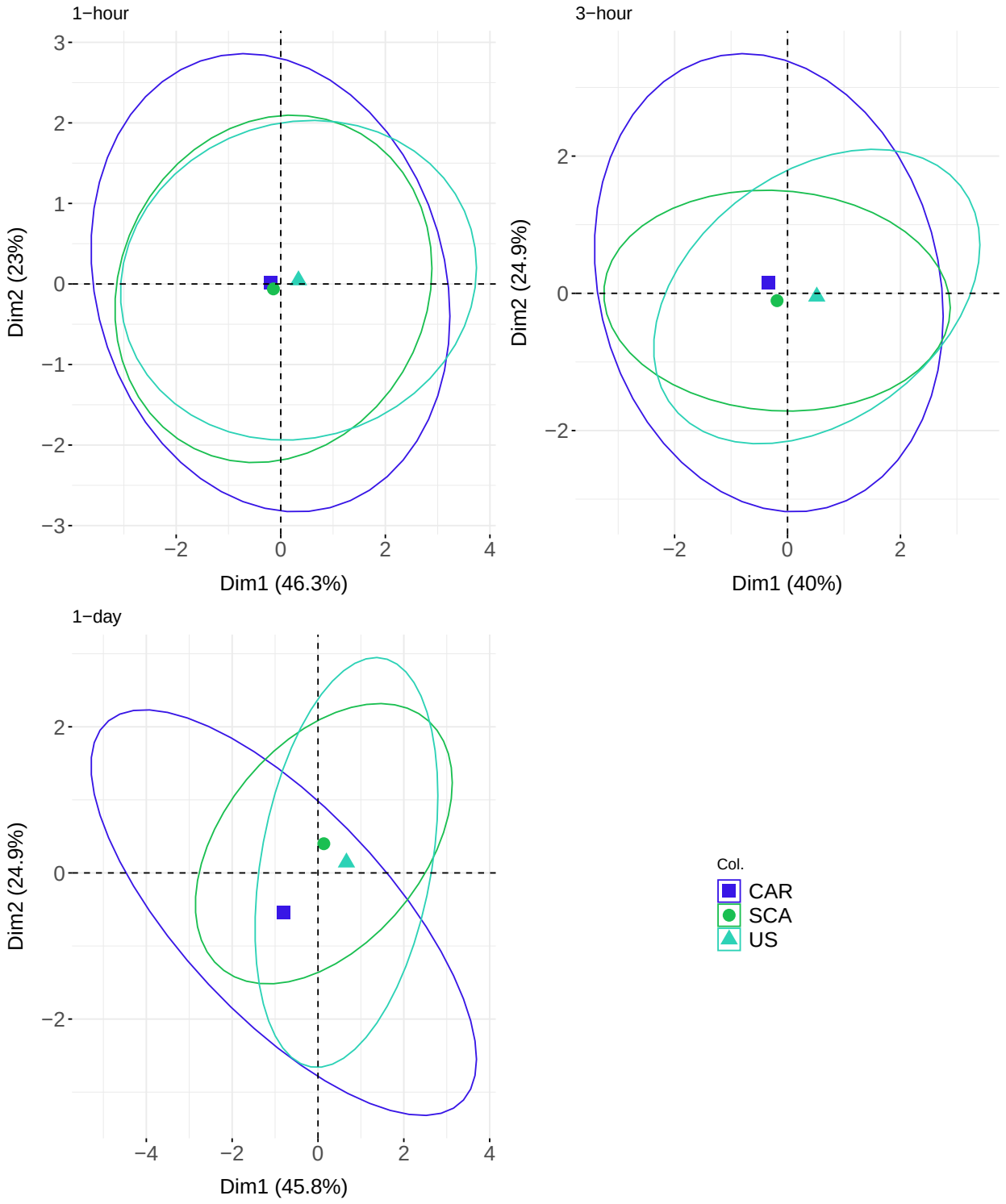

```
ggarrange(ScorePlot_1hour_ME, ScorePlot_3hour_ME, ScorePlot_1day_ME, leg_scoreME, ncol=2,
↪  nrow=2)
```




```
ggarrange(ScorePlot_1hour_Ind, ScorePlot_3hour_Ind, ScorePlot_1day_Ind, leg_scoreInd,
↪  ncol=2, nrow=2)
```




```
ggarrange(ScorePlot_1hour_Region, ScorePlot_3hour_Region, ScorePlot_1day_Region,
↪  leg_scoreRegion, ncol=2, nrow=2)
```



PERMANOVA and PERMDISP

Many variables do not seem normally distributed. To avoid issues with normal distribution assumptions, I will use a non-parametric test. Since I am comparing groups that each have multiple variables, I will use a PERMANOVA and PERMDISP. To start, you will first need

to see if you data meets the PERMANOVA dispersion assumptions. Interpretation of p-values will be heavily influenced based on whether the assumptions of PERMANOVA are met. You will have to manually calculate eta-squared by dividing the factor's SS/Total SS. Also note that these statistics will not be discussed in depth in this document. For a further dicussion on the limitations and methods used, see my thesis paper.

PERMDISP

```
#1-hour
set.seed(2016)
#determine if the test will be unbalanced. If there are hundreds of entries, an
↪  unbalanced design should not affect the results.
#Create a distance matrix
dis <- dist(Norm_1hour[,1:4])
#create a dispersion object
mod_event <- betadisper(dis, Norm_1hour$migrationEvent, type = "centroid")
mod_Segment <- betadisper(dis, Norm_1hour$Segment, type = "centroid")
mod_Ind <- betadisper(dis, Norm_1hour$Individual, type = "centroid")
#Test the homogeneity of dispersion between groups. If the p-value is > 0.05, then the
↪  assumptions of PERMANOVA are met: the null hypothesis is that overall dispersion
↪  between groups is not different.
permutest(mod_event) # small effect size, sig p-value
```

```
##
## Permutation test for homogeneity of multivariate dispersions
## Permutation: free
## Number of permutations: 999
##
## Response: Distances
##            Df Sum Sq Mean Sq      F N.Perm Pr(>F)
## Groups      15  46.92 3.12813 3.4203    999  0.001 ***
## Residuals 1040 951.16 0.91458
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
permutest(mod_Segment) # insignificant p-value
```

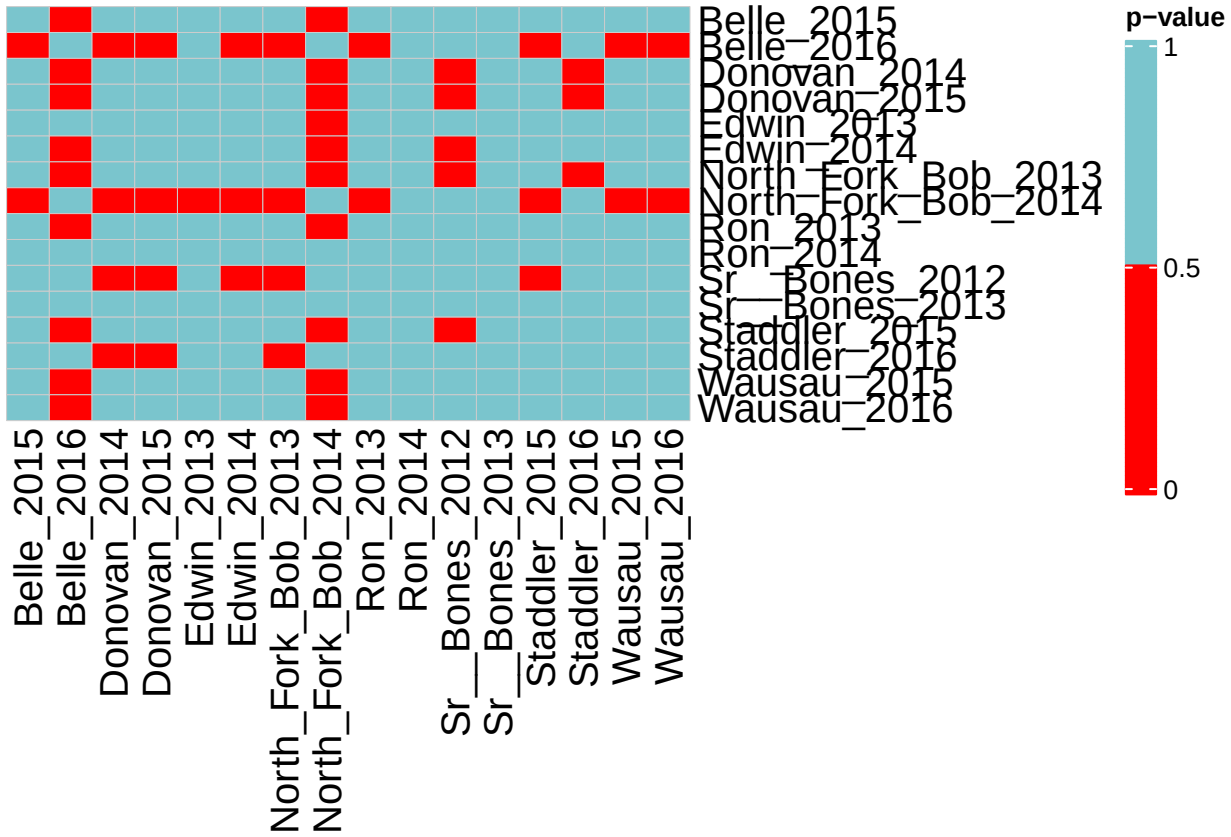
```
##
## Permutation test for homogeneity of multivariate dispersions
## Permutation: free
## Number of permutations: 999
##
## Response: Distances
##            Df  Sum Sq Mean Sq      F N.Perm Pr(>F)
## Groups      2    3.95 1.97522 2.0545    999  0.126
## Residuals 1053 1012.38 0.96143
```

```
permutest(mod_Ind) # small effect size, sig p-value
```

```
##
## Permutation test for homogeneity of multivariate dispersions
```

```
## Permutation: free
## Number of permutations: 999
##
## Response: Distances
##             Df Sum Sq Mean Sq      F N.Perm Pr(>F)
## Groups       7  23.51  3.3587 3.6251    999  0.002 **
## Residuals 1048 970.97  0.9265
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
#post-hoc for dispersions
dispersions_event<-TukeyHSD(mod_event)
dispersions_m_event<-tukey_to_matrix(dispersions_event$group) #make into a matrix for
↪  corrplot
dispersions_m_event[is.na(dispersions_m_event)]<- 1
dispersions_m_event_t<-t(dispersions_m_event)
dispersions_m_event_t[is.na(dispersions_m_event_t)]<- 1
dispersions_event_1hr<-dispersions_m_event*dispersions_m_event_t

#Make heatmaps
Heatmap(dispersions_event_1hr, name = "p-value",
        row_names_gp = gpar(fontsize = 15),
        column_names_gp= gpar(fontsize = 15),
        cluster_rows = F,
        cluster_columns = F,
        col = colorRamp2(breaks = c(0,0.5,0.5000001, 1),
                    colors = c("red", "red","cadetblue3", "cadetblue3")),
        color_space = c(0,1),
        rect_gp = gpar(col = "grey80", lwd = 0.5),
        heatmap_legend_param = list(legend_height = unit(6, "cm"))) #export size is 1500
        ↪  x 1160
```




```
#Belle_2016 and North Fork Bob 2014 have the most differences. Belle_2016 had a very wide
↪  range of values, while North Fork bob 2014 has a narrow range of values. They
↪  represent opposite ends of a spectrum. Donovan and Edwin have a moderate number of
↪  differences, largely with the same individuals.
```

```
set.seed(2016)
#determine if the test will be unbalanced. If there are hundreds of entries, an
↪  unbalanced design should not affect the results.
#Create a distance matrix
dis3 <- dist(Norm_3hour[,1:4])
#create a dispersion object
mod_event_3 <- betadisper(dis3, Norm_3hour$migrationEvent, type = "centroid")
mod_Segment_3 <- betadisper(dis3, Norm_3hour$Segment, type = "centroid")
mod_Ind_3 <- betadisper(dis3, Norm_3hour$Individual, type = "centroid")
#Test the homogeneity of dispersion between groups. If the p-value is > 0.05, then the
↪  assumptions of PERMANOVA are met: the null hypothesis is that overall dispersion
↪  between groups is not different.
permutest(mod_event_3) # medium effect size with sig difference
```

```
##
## Permutation test for homogeneity of multivariate dispersions
## Permutation: free
## Number of permutations: 999
##
## Response: Distances
```

```
##            Df Sum Sq Mean Sq      F N.Perm Pr(>F)
## Groups     15  31.77 2.11797 2.5598    999  0.002 **
## Residuals 272 225.05 0.82738
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

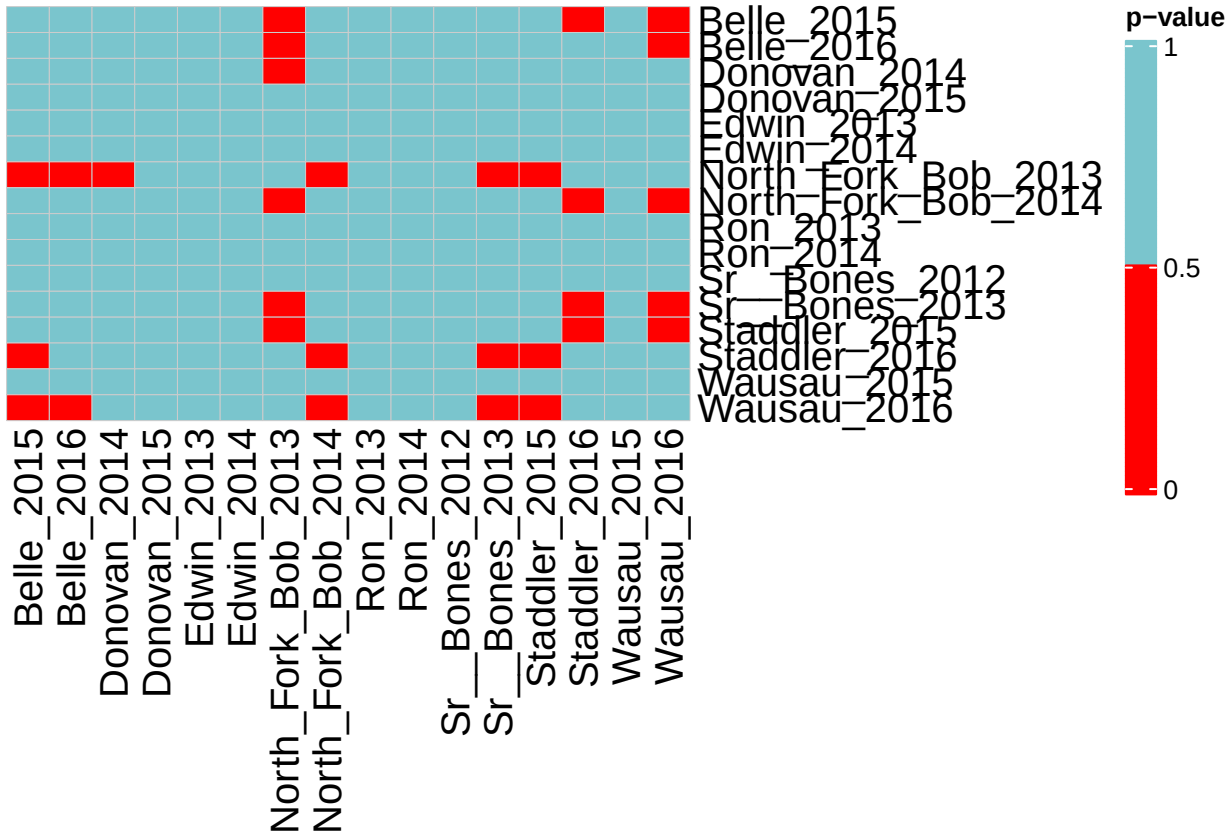
```
permutest(mod_Segment_3) # no sig difference
```

```
##
## Permutation test for homogeneity of multivariate dispersions
## Permutation: free
## Number of permutations: 999
##
## Response: Distances
##            Df  Sum Sq Mean Sq      F N.Perm Pr(>F)
## Groups      2   3.469 1.73463 1.8014    999  0.172
## Residuals 285 274.438 0.96294
```

```
permutest(mod_Ind_3) # Small effect size with sig difference.
```

```
##
## Permutation test for homogeneity of multivariate dispersions
## Permutation: free
## Number of permutations: 999
##
## Response: Distances
##            Df  Sum Sq Mean Sq      F N.Perm Pr(>F)
## Groups      7  14.405 2.05785 2.2345    999  0.033 *
## Residuals 280 257.866 0.92095
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
#post-hoc for dispersions
dispersions_event3<-TukeyHSD(mod_event_3)
dispersions_m_event3<-tukey_to_matrix(dispersions_event3$group) #make into a matrix for
↪  corrplot
dispersions_m_event3[is.na(dispersions_m_event3)]<- 1
dispersions_m_event_t3<-t(dispersions_m_event3)
dispersions_m_event_t3[is.na(dispersions_m_event_t3)]<- 1
dispersions_event_3hr<-dispersions_m_event3*dispersions_m_event_t3

#Make heatmaps
Heatmap(dispersions_event_3hr, name = "p-value",
        row_names_gp = gpar(fontsize = 15),
        column_names_gp= gpar(fontsize = 15),
        cluster_rows = F,
        cluster_columns = F,
        col = colorRamp2(breaks = c(0,0.5,0.5000001, 1),
                    colors = c("red", "red","cadetblue3", "cadetblue3")),
        color_space = c(0,1),
        rect_gp = gpar(col = "grey80", lwd = 0.5),
        heatmap_legend_param = list(legend_height = unit(6, "cm"))) #NFB and Belle have
        ↪  the most differences, followed by Sr. Bones, then Donovan 2014.
```




```
set.seed(2016)
#determine if the test will be unbalanced. If there are hundreds of entries, an
↪  unbalanced design should not affect the results.
#Create a distance matrix
dis24 <- dist(Norm_1day[,1:4])
#create a dispersion object
mod_event_24 <- betadisper(dis24, Norm_1day$migrationEvent, type = "centroid")
mod_Segment_24 <- betadisper(dis24, Norm_1day$Segment, type = "centroid")
mod_Ind_24 <- betadisper(dis24, Norm_1day$Individual, type = "centroid")
#Test the homogeneity of dispersion between groups. If the p-value is > 0.05, then the
↪  assumptions of PERMANOVA are met: the null hypothesis is that overall dispersion
↪  between groups is not different.
permutest(mod_event_24) # sig p-value, large effect size
```

```
##
## Permutation test for homogeneity of multivariate dispersions
## Permutation: free
## Number of permutations: 999
##
## Response: Distances
##           Df  Sum Sq Mean Sq     F N.Perm Pr(>F)
## Groups    15 26.1293 1.74195 5.655    999  0.001 ***
## Residuals 32  9.8571 0.30804
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

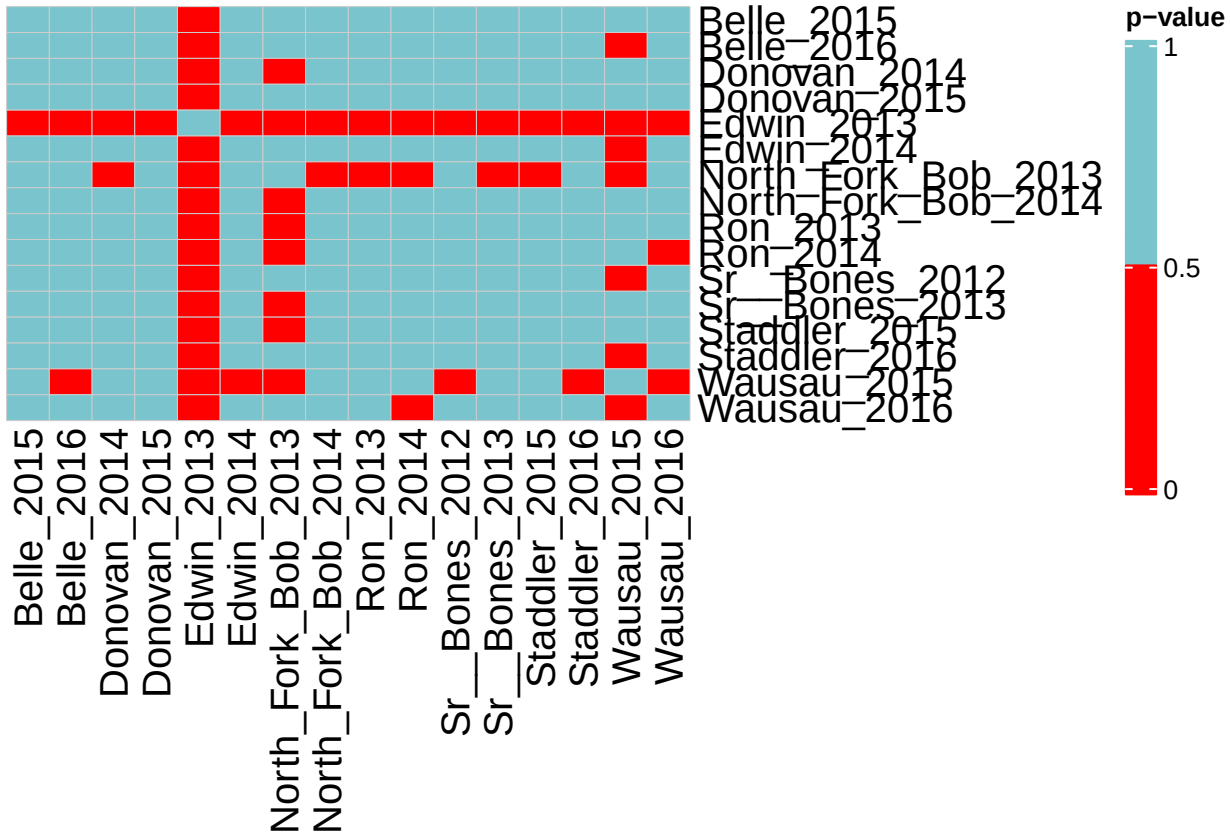
```
permutest(mod_Segment_24) #insignificant p-value, small effect size
```

```
##
## Permutation test for homogeneity of multivariate dispersions
## Permutation: free
## Number of permutations: 999
##
## Response: Distances
##           Df Sum Sq Mean Sq      F N.Perm Pr(>F)
## Groups     2  2.909  1.4544 1.2178    999    0.3
## Residuals 45 53.741  1.1943
```

```
permutest(mod_Ind_24) #sig p-value, large effect size
```

```
##
## Permutation test for homogeneity of multivariate dispersions
## Permutation: free
## Number of permutations: 999
##
## Response: Distances
##           Df Sum Sq Mean Sq      F N.Perm Pr(>F)
## Groups     7 15.942 2.27744 3.2106    999  0.009 **
## Residuals 40 28.374 0.70935
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
#post-hoc for dispersions
dispersions_event_24<-TukeyHSD(mod_event_24)
dispersions_m_event_24<-tukey_to_matrix(dispersions_event_24$group) #make into a matrix
↪  for corrplot
dispersions_m_event_24[is.na(dispersions_m_event_24)]<- 1
dispersions_m_event_24_t<-t(dispersions_m_event_24)
dispersions_m_event_24_t[is.na(dispersions_m_event_24_t)]<- 1
dispersions_event_24hr<-dispersions_m_event_24*dispersions_m_event_24_t

#Make heatmaps
Heatmap(dispersions_event_24hr, name = "p-value",
        row_names_gp = gpar(fontsize = 15),
        column_names_gp= gpar(fontsize = 15),
        cluster_rows = F,
        cluster_columns = F,
        col = colorRamp2(breaks = c(0,0.5,0.5000001, 1),
                    colors = c("red", "red","cadetblue3", "cadetblue3")),
        color_space = c(0,1),
        rect_gp = gpar(col = "grey80", lwd = 0.5),
        heatmap_legend_param = list(legend_height = unit(6, "cm"))) #export size is 1500
        ↪  x 1160
```




```
#Mostly Edwin_2013 differs followed by NFB_2013 and Wausau_2015
```

PERMANOVA

```
#1-hour
perm_event<-adonis2(dis ~ migrationEvent, data = Norm_1hour, method = "euclidean")
perm_event #sig p-value, small effect size
```

```
## Permutation test for adonis under reduced model
## Terms added sequentially (first to last)
## Permutation: free
## Number of permutations: 999
##
## adonis2(formula = dis ~ migrationEvent, data = Norm_1hour, method = "euclidean")
##                  Df SumOfSqs      R2     F Pr(>F)
## migrationEvent   15    199.4 0.04726 3.439  0.001 ***
## Residual       1040   4020.6 0.95274
## Total          1055   4220.0 1.00000
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
perm_segment<-adonis2(dis ~ Segment, data = Norm_1hour, method = "euclidean")
perm_segment #Sig p-value, small effect size
```

```
## Permutation test for adonis under reduced model
## Terms added sequentially (first to last)
## Permutation: free
## Number of permutations: 999
##
## adonis2(formula = dis ~ Segment, data = Norm_1hour, method = "euclidean")
##            Df SumOfSqs      R2      F Pr(>F)
## Segment     2     72.1 0.01709 9.1531  0.001 ***
## Residual 1053   4147.9 0.98291
## Total    1055   4220.0 1.00000
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
perm_ind<-adonis2(dis ~ Individual, data = Norm_1hour, method = "euclidean")
perm_ind #Sig p-value, small effect size
```

```
## Permutation test for adonis under reduced model
## Terms added sequentially (first to last)
## Permutation: free
## Number of permutations: 999
##
## adonis2(formula = dis ~ Individual, data = Norm_1hour, method = "euclidean")
##              Df SumOfSqs      R2      F Pr(>F)
## Individual    7    111.3 0.02636 4.0541  0.001 ***
## Residual   1048   4108.7 0.97364
## Total      1055   4220.0 1.00000
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
adonis_OmegaSq(perm_event, partial = T) #Add omega R-squared value (effect size)
```

```
## Permutation test for adonis under reduced model
## Terms added sequentially (first to last)
## Permutation: free
## Number of permutations: 999
##
## adonis2(formula = dis ~ migrationEvent, data = Norm_1hour, method = "euclidean")
##                  Df SumOfSqs     F parOmegaSq Pr(>F)
## migrationEvent   15    199.4 3.439   0.033485  0.001 ***
## Residual       1040   4020.6
## Total          1055   4220.0
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
adonis_OmegaSq(perm_segment, partial = T)
```

```
## Permutation test for adonis under reduced model
## Terms added sequentially (first to last)
## Permutation: free
## Number of permutations: 999
##
```

```
## adonis2(formula = dis ~ Segment, data = Norm_1hour, method = "euclidean")
##            Df SumOfSqs      F parOmegaSq Pr(>F)
## Segment     2     72.1 9.1531   0.015207  0.001 ***
## Residual 1053   4147.9
## Total    1055   4220.0
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
adonis_OmegaSq(perm_ind, partial = T)
```

```
## Permutation test for adonis under reduced model
## Terms added sequentially (first to last)
## Permutation: free
## Number of permutations: 999
##
## adonis2(formula = dis ~ Individual, data = Norm_1hour, method = "euclidean")
##              Df SumOfSqs      F parOmegaSq Pr(>F)
## Individual    7    111.3 4.0541   0.019843  0.001 ***
## Residual   1048   4108.7
## Total      1055   4220.0
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
#post-hoc test - region
pairperm_segment<-pairwise.adonis2(dis ~ Segment, data = Norm_1hour)
pairperm_segment #report as table
```

```
## $parent_call
## [1] "dis ~ Segment , strata = Null , permutations 999"
##
## $US_vs_CAR
##           Df SumOfSqs      R2      F Pr(>F)
## Segment    1    53.59 0.01793 12.819  0.001 ***
## Residual 702  2934.67 0.98207
## Total    703  2988.26 1.00000
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## $US_vs_SCA
##           Df SumOfSqs      R2      F Pr(>F)
## Segment    1    44.64 0.01699 12.133  0.001 ***
## Residual 702  2582.72 0.98301
## Total    703  2627.36 1.00000
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## $CAR_vs_SCA
##           Df SumOfSqs      R2      F Pr(>F)
## Segment    1     9.94 0.00356 2.5114  0.055 .
## Residual 702  2778.39 0.99644
## Total    703  2788.33 1.00000
## ---
```

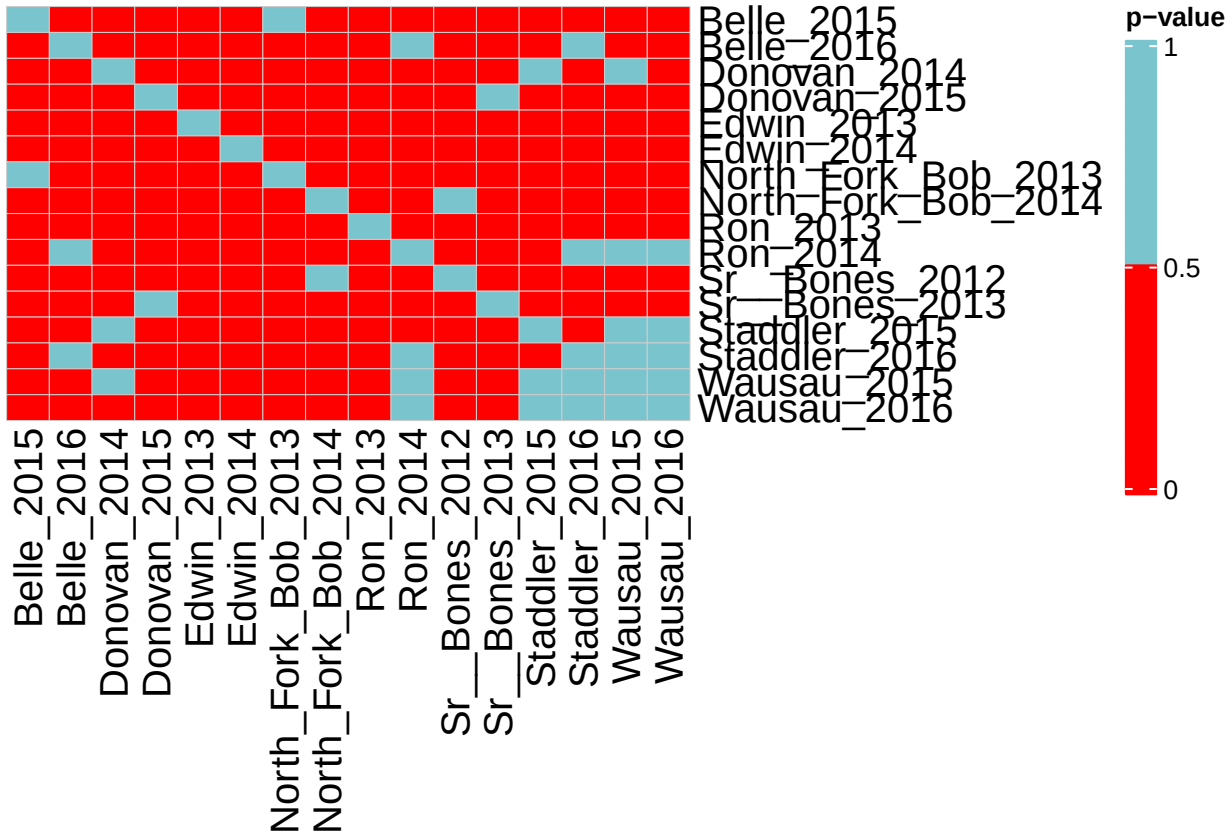
```
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## attr(,"class")
## [1] "pwadstrata" "list"
```

```
#post-hoc test - migration event
pairperm_event<-pairwise.adonis2(dis ~ migrationEvent, data = Norm_1hour)
pairperm_e<-pairperm_event[-1] #remove the first entry
pairperm_DF_e<-data.frame()
for (i in names(pairperm_e)) {
  x<-as.data.frame(pairperm_e[[i]])
  name<-i
  R2<-x$R2[1]
  P.Val<-x$`Pr(>F)`[1]
  val<-cbind(name, R2, P.Val)
  pairperm_DF_e<-rbind(val, pairperm_DF_e)
}
pairperm_DF_e$R2<-as.numeric(pairperm_DF_e$R2)
summary(pairperm_DF_e)
```

```
##      name                R2              P.Val
##  Length:120         Min.   :0.0006302   Length:120
##  Class :character   1st Qu.:0.0110251   Class :character
##  Mode  :character   Median :0.0168612   Mode  :character
##                     Mean   :0.0251450
##                     3rd Qu.:0.0305404
##                     Max.   :0.1291404
```

```
colnames(pairperm_DF_e)<-c("Comparison","R2", "p.adj")
pairperm_DF_e<-separate(data = pairperm_DF_e, col = Comparison, into = c("left",
↪  "right"), sep = "\\_vs_")
pairperm_DF_e<-select(pairperm_DF_e, "right", "left", "p.adj")
pairperm_DF_e<-pairperm_DF_e[, c("right", "left", "p.adj")]
Belle_2015<-filter(pairperm_DF_e, left == "Belle_2015")
Belle_2015<-select(Belle_2015, "left", "right", "p.adj")
Wausau_2016<-filter(pairperm_DF_e, right == "Wausau_2016")
Wausau_2016<-select(Wausau_2016, "left", "right", "p.adj")
Missing<-rbind(Wausau_2016, Belle_2015)
colnames(Missing)<-c("right", "left", "p.adj")
pairperm_DF_e_c<-rbind(Missing, pairperm_DF_e)
pairperm_DF_e_c$p.adj<-as.numeric(pairperm_DF_e_c$p.adj)
pairperm_DF_e_m<-as.matrix(xtabs(pairperm_DF_e_c[, 3] ~ pairperm_DF_e_c[,2] +
↪  pairperm_DF_e_c[, 1]))
pairperm_DF_e_m[pairperm_DF_e_m ==0]<-1
pairperm_DF_e_m_t<-t(pairperm_DF_e_m)
pairperm_DF_event<-pairperm_DF_e_m*pairperm_DF_e_m_t
#heat map time
Heatmap(pairperm_DF_event, name = "p-value",
        row_names_gp = gpar(fontsize = 15),
        column_names_gp= gpar(fontsize = 15),
        cluster_rows = F,
        cluster_columns = F,
        col = colorRamp2(breaks = c(0,0.5,0.51, 1),
```

```
                    colors = c("red", "red","cadetblue3", "cadetblue3")),
        color_space = c(0,1),
        rect_gp = gpar(col = "grey80", lwd = 0.5),
        heatmap_legend_param = list(legend_height = unit(6, "cm")))
```

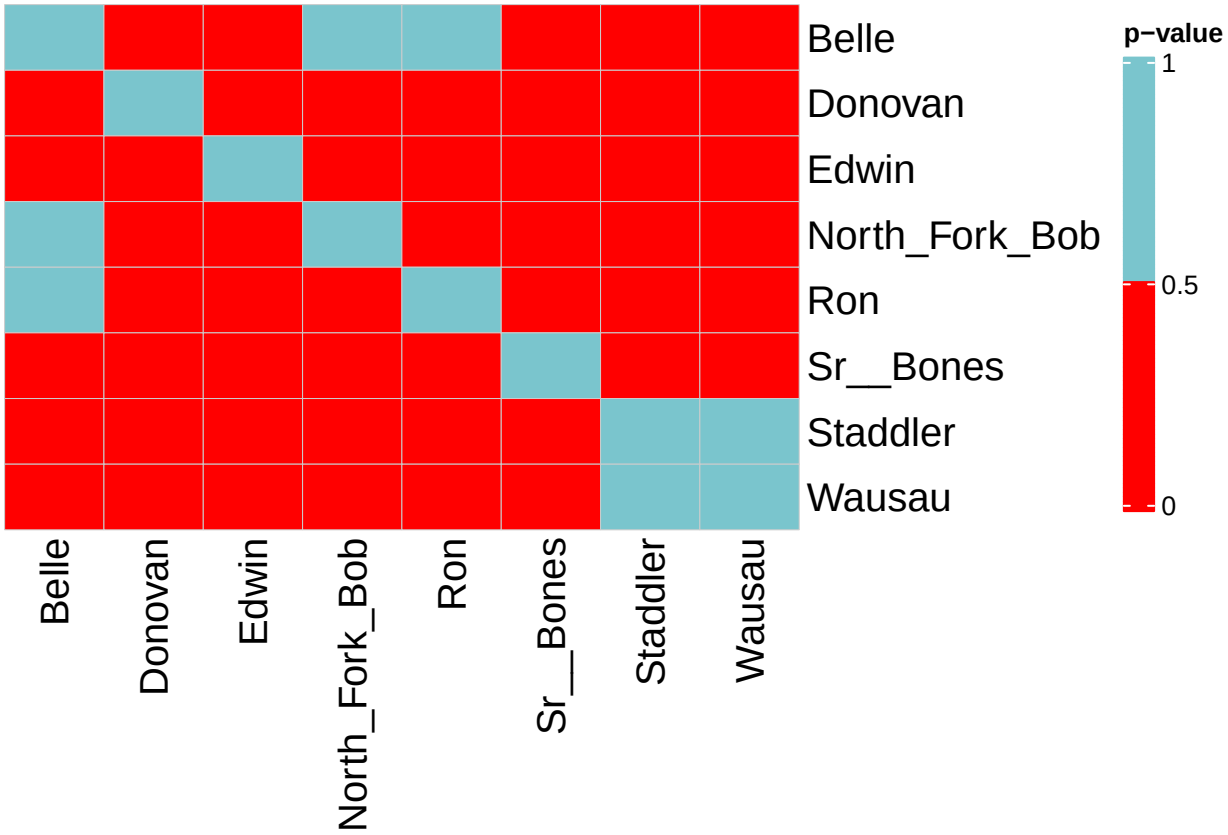



```
#post-hoc test - individual
pairperm_Ind<-pairwise.adonis2(dis ~ Individual, data = Norm_1hour)
pairperm_I<-pairperm_Ind[-1] #remove the first entry
pairperm_DF_I<-data.frame()
for (i in names(pairperm_I)) {
  x<-as.data.frame(pairperm_I[[i]])
  name<-i
  R2<-x$R2[1]
  P.Val<-x$`Pr(>F)`[1]
  val<-cbind(name, R2, P.Val)
  pairperm_DF_I<-rbind(val, pairperm_DF_I)
}
pairperm_DF_I$R2<-as.numeric(pairperm_DF_I$R2)
summary(pairperm_DF_I)
```

```
##      name                R2               P.Val
##  Length:28          Min.   :0.0001117   Length:28
##  Class :character   1st Qu.:0.0073596   Class :character
##  Mode  :character   Median :0.0100138   Mode  :character
```

```
##                     Mean   :0.0146204
##                     3rd Qu.:0.0188939
##                     Max.   :0.0484511
```

```
colnames(pairperm_DF_I)<-c("Comparison","R2", "p.adj")
pairperm_DF_I<-separate(data = pairperm_DF_I, col = Comparison, into = c("left",
↪  "right"), sep = "\\_vs_")
pairperm_DF_I<-select(pairperm_DF_I, "right", "left", "p.adj")
pairperm_DF_I<-pairperm_DF_I[, c("right", "left", "p.adj")]
Belle<-filter(pairperm_DF_I, left == "Belle")
Belle<-select(Belle, "left", "right", "p.adj")
Wausau<-filter(pairperm_DF_I, right == "Wausau")
Wausau<-select(Wausau, "left", "right", "p.adj")
Missing<-rbind(Wausau, Belle)
colnames(Missing)<-c("right", "left", "p.adj")
pairperm_DF_I_c<-rbind(Missing, pairperm_DF_I)
pairperm_DF_I_c$p.adj<-as.numeric(pairperm_DF_I_c$p.adj)
pairperm_DF_I_m<-as.matrix(xtabs(pairperm_DF_I_c[, 3] ~ pairperm_DF_I_c[,2] +
↪  pairperm_DF_I_c[, 1]))
pairperm_DF_I_m[pairperm_DF_I_m ==0]<-1
pairperm_DF_I_m_t<-t(pairperm_DF_I_m)
pairperm_DF_Individual<-pairperm_DF_I_m*pairperm_DF_I_m_t
#heat map time
Heatmap(pairperm_DF_Individual, name = "p-value",
        row_names_gp = gpar(fontsize = 15),
        column_names_gp= gpar(fontsize = 15),
        cluster_rows = F,
        cluster_columns = F,
        col = colorRamp2(breaks = c(0,0.5,0.51, 1),
                    colors = c("red", "red","cadetblue3", "cadetblue3")),
        color_space = c(0,1),
        rect_gp = gpar(col = "grey80", lwd = 0.5),
        heatmap_legend_param = list(legend_height = unit(6, "cm")))
```




```
#3hours
set.seed(2016)
perm_event3<-adonis2(dis3 ~ migrationEvent, data = Norm_3hour, method = "euclidean")
perm_event3 #Sig. p-value, medium effect size until you calculate omega-squared
```

```
## Permutation test for adonis under reduced model
## Terms added sequentially (first to last)
## Permutation: free
## Number of permutations: 999
##
## adonis2(formula = dis3 ~ migrationEvent, data = Norm_3hour, method = "euclidean")
##                 Df SumOfSqs      R2      F Pr(>F)
## migrationEvent  15   103.76 0.09038 1.8017  0.002 **
## Residual       272  1044.24 0.90962
## Total          287  1148.00 1.00000
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
perm_segment3<-adonis2(dis3 ~ Segment, data = Norm_3hour, method = "euclidean")
perm_segment3 #Sig p-value, small effect size
```

```
## Permutation test for adonis under reduced model
## Terms added sequentially (first to last)
## Permutation: free
```

```
## Number of permutations: 999
##
## adonis2(formula = dis3 ~ Segment, data = Norm_3hour, method = "euclidean")
##           Df SumOfSqs      R2      F Pr(>F)
## Segment    2    45.94 0.04002 5.9407  0.001 ***
## Residual 285  1102.06 0.95998
## Total    287  1148.00 1.00000
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
perm_ind3<-adonis2(dis3 ~ Individual, data = Norm_3hour, method = "euclidean")
perm_ind3 #sig p-value, small effect size
```

```
## Permutation test for adonis under reduced model
## Terms added sequentially (first to last)
## Permutation: free
## Number of permutations: 999
##
## adonis2(formula = dis3 ~ Individual, data = Norm_3hour, method = "euclidean")
##             Df SumOfSqs      R2      F Pr(>F)
## Individual   7    62.27 0.05424 2.2941  0.003 **
## Residual   280  1085.73 0.94576
## Total      287  1148.00 1.00000
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
adonis_OmegaSq(perm_event3, partial = T) #Add omega R-squared value (effect size)
```

```
## Permutation test for adonis under reduced model
## Terms added sequentially (first to last)
## Permutation: free
## Number of permutations: 999
##
## adonis2(formula = dis3 ~ migrationEvent, data = Norm_3hour, method = "euclidean")
##                 Df SumOfSqs      F parOmegaSq Pr(>F)
## migrationEvent  15   103.76 1.8017   0.040083  0.002 **
## Residual       272  1044.24
## Total          287  1148.00
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
adonis_OmegaSq(perm_segment3, partial = T)
```

```
## Permutation test for adonis under reduced model
## Terms added sequentially (first to last)
## Permutation: free
## Number of permutations: 999
##
## adonis2(formula = dis3 ~ Segment, data = Norm_3hour, method = "euclidean")
##           Df SumOfSqs      F parOmegaSq Pr(>F)
## Segment    2    45.94 5.9407   0.033172  0.001 ***
```

```
## Residual 285  1102.06
## Total    287  1148.00
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
adonis_OmegaSq(perm_ind3, partial = T)
```

```
## Permutation test for adonis under reduced model
## Terms added sequentially (first to last)
## Permutation: free
## Number of permutations: 999
##
## adonis2(formula = dis3 ~ Individual, data = Norm_3hour, method = "euclidean")
##             Df SumOfSqs      F parOmegaSq Pr(>F)
## Individual   7    62.27 2.2941   0.030495  0.003 **
## Residual   280  1085.73
## Total      287  1148.00
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
pairperm_segment3<-pairwise.adonis2(dis3 ~ Segment, data = Norm_3hour)
pairperm_segment3
```

```
## $parent_call
## [1] "dis3 ~ Segment , strata = Null , permutations 999"
##
## $US_vs_CAR
##           Df SumOfSqs      R2      F Pr(>F)
## Segment    1    36.84 0.04347 8.6355  0.001 ***
## Residual 190   810.56 0.95653
## Total    191   847.40 1.00000
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## $US_vs_SCA
##           Df SumOfSqs      R2      F Pr(>F)
## Segment    1    25.40 0.03852 7.6121  0.001 ***
## Residual 190   633.97 0.96148
## Total    191   659.37 1.00000
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## $CAR_vs_SCA
##           Df SumOfSqs      R2      F Pr(>F)
## Segment    1     6.68 0.00871 1.6699  0.147
## Residual 190   759.58 0.99129
## Total    191   766.25 1.00000
##
## attr(,"class")
## [1] "pwadstrata" "list"
```

```r
#No other post-hoc tests needed.
```

```r
#1day
set.seed(2016)
perm_event24<-adonis2(dis24 ~ migrationEvent, data = Norm_1day, method = "euclidean")
perm_event24 #Insig p-value with large effect size until you use omega-squared
```

```
## Permutation test for adonis under reduced model
## Terms added sequentially (first to last)
## Permutation: free
## Number of permutations: 999
##
## adonis2(formula = dis24 ~ migrationEvent, data = Norm_1day, method = "euclidean")
##                Df SumOfSqs      R2      F Pr(>F)
## migrationEvent 15   58.209 0.30962 0.9568  0.569
## Residual       32  129.791 0.69038
## Total          47  188.000 1.00000
```

```r
perm_segment24<-adonis2(dis24 ~ Segment, data = Norm_1day, method = "euclidean")
perm_segment24 #sig p-value with large effect size (medium effect size with omeg-squared)
```

```
## Permutation test for adonis under reduced model
## Terms added sequentially (first to last)
## Permutation: free
## Number of permutations: 999
##
## adonis2(formula = dis24 ~ Segment, data = Norm_1day, method = "euclidean")
##          Df SumOfSqs      R2      F Pr(>F)
## Segment   2   27.052 0.14389 3.7818  0.001 ***
## Residual 45  160.948 0.85611
## Total    47  188.000 1.00000
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```r
perm_ind24<-adonis2(dis24 ~ Individual, data = Norm_1day, method = "euclidean")
perm_ind24 #insig p-value, with medium effect size (small effect size with omega-squared)
```

```
## Permutation test for adonis under reduced model
## Terms added sequentially (first to last)
## Permutation: free
## Number of permutations: 999
##
## adonis2(formula = dis24 ~ Individual, data = Norm_1day, method = "euclidean")
##            Df SumOfSqs      R2      F Pr(>F)
## Individual  7   24.652 0.13113 0.8624  0.669
## Residual   40  163.348 0.86887
## Total      47  188.000 1.00000
```

```
adonis_OmegaSq(perm_event24, partial = T) #Add omega R-squared value (effect size)
```

```
## Permutation test for adonis under reduced model
## Terms added sequentially (first to last)
## Permutation: free
## Number of permutations: 999
##
## adonis2(formula = dis24 ~ migrationEvent, data = Norm_1day, method = "euclidean")
##                Df SumOfSqs      F parOmegaSq Pr(>F)
## migrationEvent 15   58.209 0.9568  -0.013693  0.569
## Residual       32  129.791
## Total          47  188.000
```

```
adonis_OmegaSq(perm_segment24, partial = T)
```

```
## Permutation test for adonis under reduced model
## Terms added sequentially (first to last)
## Permutation: free
## Number of permutations: 999
##
## adonis2(formula = dis24 ~ Segment, data = Norm_1day, method = "euclidean")
##          Df SumOfSqs      F parOmegaSq Pr(>F)
## Segment   2   27.052 3.7818    0.10387  0.001 ***
## Residual 45  160.948
## Total    47  188.000
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
adonis_OmegaSq(perm_ind24, partial = T)
```

```
## Permutation test for adonis under reduced model
## Terms added sequentially (first to last)
## Permutation: free
## Number of permutations: 999
##
## adonis2(formula = dis24 ~ Individual, data = Norm_1day, method = "euclidean")
##            Df SumOfSqs      F parOmegaSq Pr(>F)
## Individual  7   24.652 0.8624  -0.020483  0.669
## Residual   40  163.348
## Total      47  188.000
```

```
pairperm_segment24<-pairwise.adonis2(dis24 ~ Segment, data = Norm_1day)
pairperm_segment24
```

```
## $parent_call
## [1] "dis24 ~ Segment , strata = Null , permutations 999"
##
## $US_vs_CAR
##          Df SumOfSqs      R2      F Pr(>F)
## Segment   1   21.004 0.14332 5.0189  0.001 ***
```

```
## Residual 30  125.549 0.85668
## Total    31  146.553 1.00000
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## $US_vs_SCA
##          Df SumOfSqs      R2      F Pr(>F)
## Segment   1    4.470 0.06104 1.9501  0.117
## Residual 30   68.766 0.93896
## Total    31   73.236 1.00000
##
## $CAR_vs_SCA
##          Df SumOfSqs      R2      F Pr(>F)
## Segment   1   15.104 0.10586 3.5517  0.004 **
## Residual 30  127.581 0.89414
## Total    31  142.685 1.00000
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## attr(,"class")
## [1] "pwadstrata" "list"
```

```
#No other post-hoc tests needed.
```