## Now let's do the above with the other females

## NOHA 628 Salty

library(move)

Salty\_move <- move(x = "~/Desktop/R\_Forever/Dissertation/noha-move-hab/Data/NOHA 628.csv")

show(Salty\_move)

n.locs(Salty\_move) # number of locations

head(timeLag(Salty\_move, units="mins")) # time difference between locations - this is misleading because her first few days were set to two hour locations, but were then changd to 1 hour locations; make sure to look at ALL time differences

head(timestamps(Salty\_move))

summary(timeLag(Salty\_move, units="mins"))

Salty\_lag <- timeLag(Salty\_move, units="mins")

write.csv(Salty\_lag, file = "Salty\_lag.csv")

# burst the movestack object to exclude any loactions that are greater than 32.3 minutes apart (because a several locations are 32.3 mins and not 30 mis) - this is to prevent calculations of bridges and motion variance overnight between the last location of the previous day and the first location of the next morning, which are typically 400+ mins

Salty\_bursted <- move::burst(Salty\_move, c('normal','long')[1+(timeLag(Salty\_move, units='mins')>32.3)])

#plot Salty's locations

#par(mfcol=1:2)

plot(Salty\_bursted, type="o", col=3, lwd=2, pch=20, xlab="location\_long",ylab="location\_lat")

#plot Salty's locations with ggmap over map layer just to see if it is geographically correct! - it is

install.packages("ggmap")

library(ggmap)

require(ggmap) #these packages are necessary to work with google maps

#require(mapproj)

Salty\_df <- as(Salty\_bursted, "data.frame")

m <- get\_map(bbox(extent(Salty\_bursted)\*1.1), source="stamen", zoom=12)

ggmap(m)+geom\_path(data=Salty\_df, aes(x=location.long, y=location.lat))

# transform coordinates from lat lon, center = T is requiSalty for the dbbmm to operate properly according to Bart on the movebank help chat

Salty\_bursted\_trans <- spTransform(x = Salty\_bursted, CRSobj = '+proj=utm +zone=10 +datum=NAD83 +units=m +ellps=GRS80 +towgs84=0,0,0 +lon\_0=-122.0374075 +lat\_0=38.2021575', center = T)

proj4string(Salty\_bursted\_trans)

str(Salty\_bursted)

# bring in raster UTM with NAD 83 projection from ArcMAP because above code isn't working

nlcd\_utm\_Salty <- raster("~/Desktop/R\_Forever/RRF/Data/Raster\_UTM/NLCD\_UTM.tif")

#plot(nlcd\_utm)

#str(nlcd\_utm)

#as.data.frame(nlcd\_utm, xy = TRUE)

#current projection - this is not working - something happens in this proejction transformation and the landcover values are lost for some reason.

str(nlcd\_utm\_Salty)

str(Salty\_bursted\_trans)

Suisun\_nlcd\_trans\_Salty <- raster(nlcd\_utm\_Salty)

crs(Suisun\_nlcd\_trans\_Salty) <- "+proj=utm +zone=10 +datum=NAD83 +units=m +ellps=GRS80 +towgs84=0,0,0 +lon\_0=-122.0374075 +lat\_0=38.2021575 "

#now they match

str(Salty\_bursted\_trans)

str(Suisun\_nlcd\_trans\_Salty)

Salty\_dbbmm <- brownian.bridge.dyn(Salty\_bursted\_trans, burstType = 'normal', raster = Suisun\_nlcd\_trans\_Salty, location.error = 10, ext = .3, time.step = 30, margin = 3, window.size = 7) #location error is 10 m as per the transmitter specifications, extent is 30% of raster extent, time step is 30 mins becasue locations were approximately every hour, margin is 3 which is the minimum number of locations needed to calculate Saltyakpoints a a leave-one-out approach, and window size is 7 because this is equivalent to 7 locations, which equals 7 hours and may be able to detect behavioral changes within this relatively short window.

## below are the UDs calculated from the dbbmm

Salty\_dbbmm\_UD<-new(".UD",calc(Salty\_dbbmm, sum)) ## it works!!!

head(Salty\_dbbmm\_UD)

str(Salty\_dbbmm\_UD)

summary(Salty\_dbbmm\_UD)

## get the UD raster layer??

#Salty\_ud <- UDStack(Salty\_dbbmm)

#summary(Salty\_ud)

#now plot the UD on the left and the actual movement path on the right

#I can't figure out how to change the map area such that the map area is zoomed in, but whatever

par(mfrow=c(1,2))

plot(Salty\_dbbmm\_UD, xlab="longitude", ylab="latitude")

#zoom(Salty\_dbbmm\_UD)

plot(Salty\_dbbmm\_UD, xlab="longitude", ylab="latitude")

lines(Salty\_bursted\_trans, center=TRUE, col=3, lwd=2)

#plot(mama\_dbbmm, xlab="location\_long", ylab="location\_lat")

#points(spTransform(mama\_bursted, center=TRUE), col=8)

# this plots the 50% and 95% UD contour lines around the UDs

plot(Salty\_dbbmm\_UD, xlab="location\_long", ylab="location\_lat")

contour(Salty\_dbbmm\_UD, levels=c(.5, .95), col=c(6,2), add=TRUE, lwd=2)

#get the area of the 95% UD - i think these areas are in meters

Salty\_cont95 <- getVolumeUD(Salty\_dbbmm\_UD)

Salty\_cont95 <- Salty\_cont95<=.95

area95 <- sum(values(Salty\_cont95))

area95

#get the area of the 50% UD - i think these areas are in meters

Salty\_cont5 <- getVolumeUD(Salty\_dbbmm\_UD)

Salty\_cont5 <- Salty\_cont5<=.5

area5 <- sum(values(Salty\_cont5))

area5

## Ok, now let's follow Brian's steps and convert the DBBMM object to a SpatialLineDataFrame

#dbbmm dataframe- keep this!

Salty.dbbmm.df <- as.data.frame(Salty\_dbbmm\_UD, xy = TRUE)

Salty\_ud\_raster <- rasterFromXYZ(Salty.dbbmm.df, crs = "+proj=utm +zone=10 +datum=NAD83 +units=m +no\_defs +ellps=GRS80 +towgs84=0,0,0", digits = 5 )

## write raster - this is returning an empty raster and we don't know why

#writeRaster(Salty\_ud\_raster, "C:/Users/sskalos/Documents/noha-move-hab/Output/mama\_ud\_raster.tif", overwrite = TRUE)

#writeRaster(Salty\_dbbmm, "C:/Users/sskalos/Documents/noha-move-hab/Output/mama\_ud\_raster2.tif", overwrite = TRUE)

#writeRaster(Salty\_ud, "C:/Users/sskalos/Documents/noha-move-hab/Output/lauren\_ud\_raster3.tif", overwrite = TRUE)

writeRaster(Salty\_dbbmm\_UD, "~/Desktop/R\_Forever/Dissertation/noha-move-hab/Output/Salty\_ud\_raster.tif", overwrite = TRUE)

#but it does create the raster appropriately and plots below (weird)

plot(Salty\_ud\_raster)

str(Salty\_ud\_raster) # and there are values within this new raster (extra weird)

#save contours to shapefiles - this works

#example from Bart on Movebank - it works

#require(move)

#example(brownian.bridge.dyn)

#cont<-raster2contour(dbbmm, level=c(.5,.95))

#writeOGR(cont, dsn = '.', layer = 'mycontLines', driver = "ESRI Shapefile")

## try with mama data - it works

require(move)

#cont2 <-raster2contour(Salty\_ud, level=c(.5,.95))

#writeOGR(cont2, dsn = '.', layer = 'Salty\_contour2', driver = "ESRI Shapefile")

#works with either the mama\_ud raster (above) or the mama\_dbbmm\_UD raster below, but below seems to be correct as it contains one layer, not every dbbmm step

require(move)

cont\_Salty <-raster2contour(Salty\_dbbmm\_UD, level=c(.5,.95))

writeOGR(cont\_Salty, dsn = '.', layer = 'Salty\_contour', driver = "ESRI Shapefile")

#getwd()

#ploygon <- readOGR(dsn = "/Users/Shannon/Desktop/R\_Forever/Dissertation/noha-move-hab/Output" ,layer = "mama\_contour3")

#mama\_WGS <- spTransform(cont3, CRS("+proj=longlat +ellps=WGS84 +datum=WGS84"))

#writeOGR(cont3, dsn = "mama\_contours.kml", layer = "mama\_WGS", driver = "KML", overwrite\_layer = TRUE)

#using the nlcd\_utm raster lines up with the correct number of rows and columns from our dbbmm dataframe, and we can extract the landcover values (finally!)

nlcd\_sp\_Salty <- SpatialPoints(Salty.dbbmm.df[,1:2], proj4string = crs(Suisun\_nlcd\_trans\_Salty))

nlcd\_extract\_Salty <- extract(nlcd\_utm\_Salty, nlcd\_sp\_Salty)

head(nlcd\_extract\_Salty)

nlcd\_extract\_Salty[which(!is.na(nlcd\_extract\_Salty))]

#check to make sure the have the same # of columns and rows

str(Suisun\_nlcd\_trans\_Salty)

str(Salty\_dbbmm\_UD)

# test to make sure it works - it does (Salty square represents the nlcd raster layer)

plot(Salty\_dbbmm\_UD)

library(scales)

plot(nlcd\_utm\_Salty, col = alpha("Salty", .5), add = TRUE)

# combine the raster cell probabilities with their coord pairs with landcover grid cells

final\_Salty <- cbind.data.frame(Salty.dbbmm.df, nlcd\_extract\_Salty)

head(final\_Salty)

#above works, but returns all columns, including empty grid cells with NA and 0 values

# below code removes NA in the 4th column (the landcover column) and returns only columns with landcover values

final\_Salty <- final\_Salty[which(!is.na(final\_Salty[,4])),]

head(final\_Salty)

# for loop to calculate probabilities of use within each landcover types using the UDs

prob.vec <- rep(NA, length(unique(final\_Salty[,4])))

unique.vec <- unique(final\_Salty[,4])

tot.prob <- sum(final\_Salty[,3])

for (i in 1:length(prob.vec)){

prob.vec[i] <- sum(final\_Salty[which(final\_Salty[,4] == unique.vec[i]),3])/tot.prob

}

#check that the for loop worked and the probabilities sum to 1 - they do

sum(prob.vec)

#save the probability table for each landcover class - it works!

probs.cover.tables <- cbind(prob.vec, unique.vec)

#view the entire table

probs.cover.tables

write.csv(probs.cover.tables, file = "Salty\_landcover\_probs\_final.csv")