# Tracking Dragons: Stable isotopes reveal the annual cycle of a migratory dragonfly

Michael T. Hallworth1,2,\*, Peter P. Marra1, Kent McFarland3, Sara Zahendra3 and Colin E. Studds1,2

1 Migratory Bird Center, Smithsonian Conservation Biology Institute, Washington, D.C. 20008.

2 University of Maryland, Baltimore County, Baltimore, MD. 21250.

3 Vermont Center for Ecosystem Studies, White River Junction, VT. 05001

\*Corresponding author

## ABSTRACT

Insect migration is a globally ubiquitous behavior that can involve continental-scale movements and complex life-histories1. Apart from select species of migratory moths and butterflies2-5, little is known about the structure of the annual cycle for migratory insects. Using stable-hydrogen isotope analysis of 852 wing samples from eight countries spanning 140 years, combined with 21 years of citizen science data, we describe the migratory phenology and annual cycle of a large charismatic dragonfly, the common green darner (*Anax junius*). We demonstrate that darners undertake complex long-distance annual migrations governed largely by temperature that involve at least three generations. In early spring, the first generation makes a one-way, long distance northward migration (659.46 ± 49.11 km) from the southern to northern range limits, dying after migration. A second generation makes a one-way return journey to the southern range edge (680.97 ± 178.84 km), where they lay eggs and die. Finally, a third non-migratory generation emerges, reproducing locally giving rise to the cohort that migrates north the following spring. Our findings provide evidence that common green darners exhibit a novel migratory strategy, highly dependent on temperature, and intermediate between directed migrations similar to birds and multi-generational migration exhibited by some Lepidoptera3,4.

### Load packages

#devtools::install\_github("SMBC-NZP/MigConnectivity")  
library(MigConnectivity)  
library(raster)  
library(dismo)

### Gather occurrence data

# read in the full dataset  
AJgbif <- read.csv("AnaxJunius\_occupancy\_GBIF/occurrence.txt",sep="\t")  
  
# keep only these columns of data  
keepers <- c("occurrenceID","catalogNumber","recordedBy","individualCount",  
 "sex","lifeStage","year","month","day","decimalLatitude","decimalLongitude")  
  
# subset to include only those columns of data  
AJgbif <- AJgbif[,keepers]  
  
# read in captured dragonflies within dataset  
AJsampled <- read.csv("Data/MDPsamples\_PredictedDvalues.csv")  
  
# build data.frame to match AJgbif   
AJcaps <- data.frame(occurrenceID = AJsampled$VCE\_ID,  
 catalogNumber = AJsampled$MuseumSpecimenKey,  
 recordedBy = AJsampled$Collector,  
 individualCount = rep(1, nrow(AJsampled)),  
 sex = AJsampled$Sex,  
 lifeStage = toupper(AJsampled$Stage),  
 year = as.numeric(format(as.POSIXlt(AJsampled$CollectionDate,  
 format = "%m/%d/%Y"),"%Y")),  
 month = format(as.POSIXlt(AJsampled$CollectionDate,  
 format = "%m/%d/%Y"),"%m"),  
 day = format(as.POSIXlt(AJsampled$CollectionDate,  
 format = "%m/%d/%Y"),"%d"),  
 decimalLatitude = AJsampled$Lat\_Final,  
 decimalLongitude = AJsampled$Long\_Final)  
  
AJcaps$year <- ifelse(AJcaps$year > 2018,as.numeric(paste0("19",substr(AJcaps$year,3,4))),AJcaps$year)  
  
  
# Make master data   
AJlocs <- rbind(AJgbif,AJcaps)  
nrow(AJcaps)  
# remove captures without locations  
AJlocs <- AJlocs[!is.na(AJlocs$decimalLatitude)&!is.na(AJlocs$decimalLongitude),]  
  
# make into spatial points data frame  
AJlocs <- sp::SpatialPointsDataFrame(  
 sp::SpatialPoints(cbind(AJlocs$decimalLongitude,AJlocs$decimalLatitude),  
 proj4string = sp::CRS(projections$WGS84)),  
 data = AJlocs)  
  
# remove duplicate locations   
AJlocs <- remove.duplicates(AJlocs, zero = 0.0)  
  
# let's keep only the plausible Mainland locations (remove UK and Hawaii)  
AJlocs <- AJlocs[AJlocs$decimalLongitude < -40 & AJlocs$decimalLongitude > -145 , ]  
  
# witholding a 20% sample for testing  
fold <- kfold(AJlocs, k=5)  
testData <- AJlocs[fold == 1, ]  
trainingData <- AJlocs[fold != 1, ]

head(AJlocs)

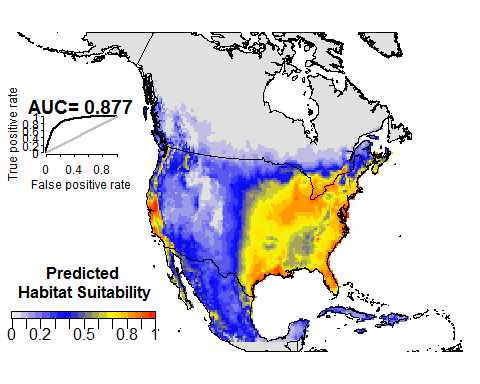
## occurrenceID catalogNumber recordedBy individualCount sex  
## 1 6459.001 QMOR6459 Robert, Adrien 1 MALE  
## 2 6462.001 QMOR6462 Robert, Adrien 1 FEMALE  
## 3 6461.001 QMOR6461 Robert, Adrien 1 FEMALE  
## 5 6464.001 QMOR6464 Gravel, G. 1 MALE  
## 6 6458.001 QMOR6458 Couillard, P. 1 MALE  
## 9 23649.001 QMOR23649 Robert, A. 1   
## lifeStage year month day decimalLatitude decimalLongitude  
## 1 ADULT 1937 6 10 45.4667 -74.3000  
## 2 ADULT 1942 9 4 45.5000 -73.5833  
## 3 ADULT 1958 6 9 46.7500 -74.5000  
## 5 ADULT 1965 8 24 54.0000 -72.0000  
## 6 ADULT 1978 9 3 46.2167 -73.0167  
## 9 1940 7 18 46.0833 -73.1667

### Gather environmental data

EnvData <- getData("worldclim", var="bio", res=2.5)  
ElevDataUS <- getData("alt", country = "United States")  
ElevDataCA <- getData("alt", country = "Canada")  
ElevDataMX <- getData("alt", country = "Mexico")  
ElevDataCU <- getData("alt", country = "Cuba")  
ElevDataJM <- getData("alt", country = "Jamaica")  
ElevDataPR <- getData("alt", country = "Puerto Rico")  
ElevDataDR <- getData("alt", country = "Dominican Republic")  
ElevDataHA <- getData("alt", country = "Haiti")  
  
elevrasters <- list(ElevDataUS[[1]],  
 ElevDataCA,  
 ElevDataMX,  
 ElevDataCU,  
 ElevDataJM,  
 ElevDataDR,  
 ElevDataHA)  
elevrasters$fun <- max  
Elev <- do.call(mosaic,elevrasters)  
  
#saveRDS(Elev,"Elevation.rds")

### Habitat suitability - Maxent

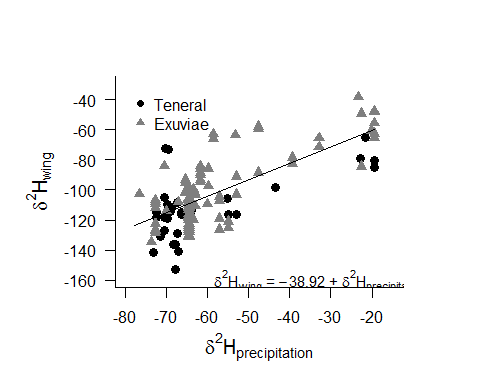
EnvData <- crop(EnvData,AJlocs)  
  
rasterTemplate <- MigConnectivity::getIsoMap()  
rasterTemplate <- crop(rasterTemplate,AJlocs)  
  
# project Elevation to match rasterTemplate  
Elevproj <- projectRaster(Elev,rasterTemplate)  
EnvData <- projectRaster(EnvData,rasterTemplate)  
  
EnvData$Elev <- Elevproj  
  
#fit the maxent model  
  
# Keep only   
#BIO01 = Mean Annual Temp  
#BIO06 = Min Temp Coldest Month  
#BIO12 = Annual Precipitation  
#BIO13 = Precipitation of Wettest Month  
#BIO14 = Precipitation of Driest Month  
  
EnvData <- dropLayer(EnvData,paste0("bio",c(2:5,7:11,15:19)))  
  
# Run maxent with full dataset  
AJmaxent <- maxent(EnvData, trainingData)  
  
# make predictions  
AJocc <- predict(AJmaxent, EnvData)  
  
# background data  
bg <- randomPoints(EnvData, 1000) #background "pseudoabsences"  
  
#simplest way to use 'evaluate'  
AUCeval <- evaluate(AJmaxent, p=testData, a=bg, x=EnvData)



### Generating origin assignments

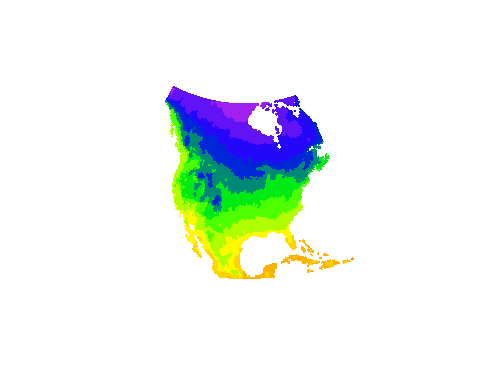
#### Creating the base map from known origin individuals

Dvalues <-read.csv("Data/MDPsamples\_PredictedDvalues.csv",header=TRUE)   
  
Teneral<-subset(Dvalues,subset=(Stage=="Teneral"|Stage=="exuviae"))  
  
# Determine the relationship between MAD and locally grown wing tissue   
# using a linear regression. Save standard deviation of residuals for   
# probablisitic likelihood assignments  
y <- Teneral[,"D1"] # measured isotope values of feathers  
x <- Teneral[,40] # mean annual deuterium in precip  
Iso<-lm(y~x)  
STD<-sd(Iso$residuals)  
Teneral <- Teneral[!is.na(Teneral$D1) & !is.na(Teneral$Long\_Final),]



### Classify ‘populations’ by USDA 2012 hardiness zones

#BIO06 = Minimum temperature of coldest month   
# Already in EnvData   
  
# convert from C\*10 to F #  
minTempF<-calc(EnvData[["bio6"]],fun=function(x){((x/10)\*(9/5)) + 32})  
  
# Set up Hardiness zone temperature regions - based on USDA 2012 reclassification #  
  
# Reclassify the minTempF to hardiness zones #  
  
reclass<-array(NA,c(28,3))  
reclass[,1]<-seq(from=-60,to=75,by=5)  
reclass[,2]<-seq(from=-55,to=80,by=5)  
reclass[,3]<-rep(1:14,each = 2)  
  
HardinessZones<-reclassify(minTempF,rcl=reclass,right=NA)  
  
# Extract the hardiness zone of each sample location and attach to datafile   
Dvalues$HardinessZone <- extract(HardinessZones,cbind(Dvalues$Long\_Final,Dvalues$Lat\_Final))  
  
# Some hardiness values are NA - create a buffer around those and return the mode  
getMode <- function(x) {  
 b <- unique(x)  
 b[which.max(tabulate(match(x, b)))]}  
  
HardBuffer <- extract(HardinessZones,  
 cbind(Dvalues$Long\_Final,Dvalues$Lat\_Final),  
 buffer =150000,  
 fun = getMode)  
  
# fill in missing hardiness values with mode  
Dvalues$HardinessZone[is.na(Dvalues$HardinessZone)] <- HardBuffer[is.na(Dvalues$HardinessZone)]



### Make origin assignments for unknown individuals

Here we use the predicted habitat suitability as a prior

# Subset out young and missing D values  
Adults <- subset(Dvalues,subset=(D1!="NA" & Stage == "adult"))  
  
h <- isoAssign(isovalues = Adults$D1,  
 isoSTD = STD,  
 intercept = coefficients(Iso)[1],  
 slope = coefficients(Iso)[2],  
 relAbund = AJocc,  
 assignExtent = extent(AJocc))  
  
#saveRDS(h,"dragonMCassign.rds")

### Limit assignments based on first flight date

WorldClimData0.3<-lapply(list.files("Spatial\_Layers/",pattern = ".grd",full.names = TRUE),raster)  
  
WorldClimData0.3<-brick(WorldClimData0.3)  
  
# WorldClim temperatures are degC\*10 - so need to divide by 10 to get degC #  
  
WorldClimData<- WorldClimData0.3/10  
  
# Crop to match assignments  
WorldClimData <- crop(WorldClimData,h$probassign[[1]])  
  
# Create first flight raster for each month #  
# generate empty list to store rasters #  
  
# Determine flight limit based on temperature threshold of 9.17 #  
  
threshold <- reclassify(WorldClimData,  
 rcl= matrix(c(-100,9.17,0,  
 9.17,100,1),  
 byrow = TRUE,   
 nrow = 2))  
  
################################################################################  
# Create a northern limit to assignments based on flight dates #  
Month<-Adults$Month  
Month[is.na(Month)]<-8  
  
# Truncate only movement going north and not movement going south #  
Month[Month>8]<-8  
  
LimitFlight<-vector('list',length(Month))  
  
for(i in 1:nrow(Adults)){  
LimitFlight[[i]]<-overlay(h$oddsassign[[i]],threshold[[Month[i]]],  
 fun=function(x,y){return(x\*y)})  
}  
  
LimitFlight <- stack(LimitFlight)

### Make assignments based on month captured and hardiness zone

# spatial points data frame  
Adults\_spdf <- SpatialPointsDataFrame(SpatialPoints(cbind(Adults$Long\_Final,  
 Adults$Lat\_Final)),  
 Adults)  
  
# Collapse hardiness zones below 7  
Adults\_spdf$HardinessZone[Adults\_spdf$HardinessZone<7]<-7  
  
# Collapse hardiness zones above 11  
Adults\_spdf$HardinessZone[Adults\_spdf$HardinessZone>11]<-11  
  
# Generate a table showing the sample size in each month in each hardiness zone #  
MonthZoneTable<-array(NA,c(11,12))  
rownames(MonthZoneTable)<-seq(1,11,1)  
colnames(MonthZoneTable)<-c("Jan","Feb","Mar","Apr","May","Jun","Jul","Aug","Sept","Oct","Nov","Dec")  
  
for(i in 1:11){  
for(m in 1:12){  
MonthZoneTable[i,m]<-length(which(Adults\_spdf$HardinessZone==i & Adults\_spdf$Month==m))  
}  
}  
  
MonthZoneTable<-MonthZoneTable[rownames(MonthZoneTable)%in%c(7:13),]  
  
  
HardZones\_spdf<-vector("list",10)  
zone<-c(7:11)  
  
for(i in 1:length(zone)){  
HardZones\_spdf[[i]]<-subset(Adults\_spdf,subset=(HardinessZone==zone[i]))  
}  
  
zMonth<-vector('list',12)  
  
Zone<-list(zMonth,zMonth,zMonth,zMonth,zMonth,zMonth,zMonth,zMonth,zMonth)  
  
Nocaps<-LimitFlight[[7]]  
Nocaps[Nocaps==0]<-0  
Nocaps[Nocaps>0]<-0  
  
names(Zone)<-paste0("zone",7:11)  
for(i in 1:length(zone)){  
 Z=zone[i]  
 N<-MonthZoneTable[row.names(MonthZoneTable)==Z,]  
   
 M<-names(N)  
 for(y in 1:12){ # Months  
 # if no samples create a raster of value 0   
 if(N[y]==0){  
 Zone[[i]][[y]] <- Nocaps }  
   
 if(N[y]==1){  
 Zone[[i]][[y]] <-LimitFlight[[which(Adults\_spdf$HardinessZone==Z & Adults\_spdf$Month==y)]]}  
   
 if(N[y]>1){  
 Zone[[i]][[y]] <-sum(LimitFlight[[which(Adults\_spdf$HardinessZone==Z & Adults\_spdf$Month==y)]])}  
 }  
}

### Migratory distance

# Calculate shortest migratory distance from origin to capture location   
crs(Adults\_spdf) <- projections$WGS84  
A.J\_proj <- sp::spTransform(Adults\_spdf,CRS(projections$EquidistConic))  
  
MigDistances <- rep(NA,nrow(A.J\_proj))  
  
s <- projectRaster(LimitFlight,crs = projections$EquidistConic)  
  
for(i in 1:nrow(A.J\_proj)){  
r <- s[[i]]  
r[r==0]<-NA  
r[r>0]<-1  
rpoly <- rgeos::gUnaryUnion(rasterToPolygons(r))  
MigDistances[i]<-rgeos::gDistance(A.J\_proj[i,],rpoly)/1000  
}  
  
MigDist <- data.frame(MinDistance = MigDistances,  
 Zone = A.J\_proj$HardinessZone,   
 Month = A.J\_proj$Month)  
  
MigDist$Zone[MigDist$Zone == 999]<-NA  
  
write.csv(MigDist, "MinMigDistanceOcc.csv")

### Determine proportion of local vs migratory

zones <- 7:11  
  
propLocal <- sampLocal <- array(NA,c(length(zones),12))  
rownames(propLocal)<-zones  
colnames(propLocal)<-1:12  
  
for(i in 1:length(zones)){  
 Z=zones[i]  
 N<-MonthZoneTable[row.names(MonthZoneTable)==Z,]  
   
 M<-names(N)  
 for(y in 1:12){ # Months  
 y = 5  
 i = 1  
 # if no samples create a raster of value 0   
 if(N[y]==0){propLocal[i,y]<-NA}  
 if(N[y]>0){  
 temp <- s[[which(Adults\_spdf$HardinessZone==Z & Adults\_spdf$Month==y)]]  
 rcl <- matrix(c(-10,0,0,  
 1,10,1),byrow = TRUE, nrow = 2, ncol = 3)  
 temp <- reclassify(temp,rcl)  
 temp.pt <- Adults\_spdf[which(Adults\_spdf$HardinessZone==Z & Adults\_spdf$Month==y),]  
 vals <- rep(NA,nrow(temp.pt))  
 for(n in 1:nrow(temp.pt)){w  
 vals[n]<-extract(temp[[n]],temp.pt[n,])  
 }  
 propLocal[i,y] <- sum(vals,na.rm = TRUE)/nrow(temp.pt)  
 sampLocal[i,y] <- sum(vals,na.rm = TRUE)  
 }  
 }  
}  
  
propMig <- 1-propLocal

