'MigConnectivity' package: some estMC function examples

Jeffrey A. Hostetler, Michael T. Hallworth

2018-07-03

`estMC` estimates the strength of migratory connectivity (MC) with estimates of uncertainty. MC estimated either with transition probability estimates or individual assignments to regions from either tracking data or stable-isotopes. Uses re-sampling to measure MC uncertainty from RMark transition probability matrix estimates and/or JAGS relative abundance MCMC samples OR geolocator and/or GPS data OR stable-isotope assignments.

`estMC` - Estimate strength of migratory connectivity incorporating location and other sampling uncertainty

Estimates of MC may also be influenced by error in individual assignment to regions. Data types vary in location accuracy and precision, which are likely to influence the accuracy with which individuals are assigned to regions.

Light-level geolocator / GPS

To estimate MC and include location uncertainty the following data are needed:

1. A logical vector indicating the type of device the location estimates were derived from (GPS = FALSE, Light-level geolocator = TRUE)

2. Location Bias - a vector that has error estimates for both longitude and latitude

3. Location error - a variance, covariance matrix of longitude and latitude

4. Distance matrix between breeding and non-breeding regions

5. A shapefile of both breeding and non-breeding regions

6. The deployment locations and the 'unknown' locations derived from the tracking devices

7. The relative (or absolute) abundance within each region where the birds originate from (deployment regions)

Below is a repeatable example for how to calculate location bias and location error using coordinates derived from light-level geolocators.

```r
# Load in projections
data("projections")

# Define deployment locations (winter) #
captureLocations<-matrix(c(-77.93,18.04,   # Jamaica
                           -80.94,25.13,   # Florida
                           -66.86,17.97),  # Puerto Rico
                            nrow=3, ncol=2, byrow = TRUE)
```

```r
# Convert capture locations into SpatialPoints #

CapLocs<-sp::SpatialPoints(captureLocations,sp::CRS(projections$WGS84))

# Project Capture locations #

CapLocsM<-sp::spTransform(CapLocs, sp::CRS(projections$EquidistConic))

# Retrieve raw non-breeding locations from github
# First grab the identity of the bird so we can loop through the files
# For this example we are only interested in the error
# around non-breeding locations
# here we grab only the birds captured during the non-breeding season
# Using paste0 for vignette formatting purposes

winterBirds <- dget(paste0("https://raw.githubusercontent.com/",
                     "SMBC-NZP/MigConnectivity/master/",
                     "data-raw/GL_NonBreedingFiles/winterBirds.txt"))

# create empty list to store the location data #
Non_breeding_files <- vector('list',length(winterBirds))

# Get raw location data from Github #
for(i in 1:length(winterBirds)){
Non_breeding_files[[i]] <- dget(paste0("https://raw.githubusercontent.com/",
                                       "SMBC-NZP/MigConnectivity/master/data-raw/",
                                       "GL_NonBreedingFiles/NonBreeding_",
                                       winterBirds[i],".txt"))
}

# Remove locations around spring Equinox and potential migration points
# same NB time frame as Hallworth et al. 2015

# two steps because subset on shapefile doesn't like it in a single step

Non_breeding_files <- lapply(Non_breeding_files,
                      FUN = function(x){
                      month <- as.numeric(format(x$Date,format = "%m"))
                               x[which(month != 3 & month != 4),]})


Jam <- c(1:9)   # locations w/in list of winterBirds captured in Jamaica
Fla <- c(10:12) # locations w/in list of winterBirds in Florida
PR <- c(13:16)  # locations w/in list of winterBirds in Puerto Rico

# Turn the locations into shapefiles #

NB_GL <- lapply(Non_breeding_files,
                FUN = function(x){
                  sp::SpatialPoints(cbind(x$Longitude,x$Latitude),
                                    sp::CRS(projections$WGS84))})

# Project into UTM projection #
```

```
NB_GLmeters <- lapply(NB_GL,
                      FUN = function(x){sp::spTransform(x,
                                        sp::CRS(projections$EquidistConic))})

# Process to determine geolocator bias and variance-covariance in meters #

# generate empty vector to store data #
LongError<-rep(NA,length(winterBirds))
LatError<-rep(NA,length(winterBirds))

# Calculate the error in longitude derived
# from geolocators from the true capture location

LongError[Jam] <- unlist(lapply(NB_GLmeters[Jam],
                         FUN = function(x){mean(x@coords[,1]-
                                              CapLocsM@coords[1,1])}))

LongError[Fla] <- unlist(lapply(NB_GLmeters[Fla],
                         FUN = function(x){mean(x@coords[,1]-
                                              CapLocsM@coords[2,1])}))

LongError[PR] <- unlist(lapply(NB_GLmeters[PR],
                        FUN = function(x){mean(x@coords[,1]-
                                             CapLocsM@coords[3,1])}))

# Calculate the error in latitude derived from
# geolocators from the true capture location

LatError[Jam] <- unlist(lapply(NB_GLmeters[Jam],
                        FUN = function(x){mean(x@coords[,2]-
                                             CapLocsM@coords[1,2])}))

LatError[Fla] <- unlist(lapply(NB_GLmeters[Fla],
                        FUN = function(x){mean(x@coords[,2]-
                                             CapLocsM@coords[2,2])}))

LatError[PR] <- unlist(lapply(NB_GLmeters[PR],
                        FUN = function(x){mean(x@coords[,2]-
                                             CapLocsM@coords[3,2])}))

# Get co-variance matrix for error of
# known non-breeding deployment sites

# lm does multivariate normal models if you give it a matrix dependent variable!

geo.error.model <- lm(cbind(LongError,LatError) ~ 1)

geo.bias <- coef(geo.error.model)
geo.vcov <- vcov(geo.error.model)
```

We measured MC for bootstrapped data of birds tracked from breeding to non-breeding regions using light-level and GPS geolocation when 1) GPS location uncertainty was applied to all individuals, 2) GPS and light-level location uncertainty were applied to individuals with those devices, and 3) light-level location uncertainty was applied to all individuals.

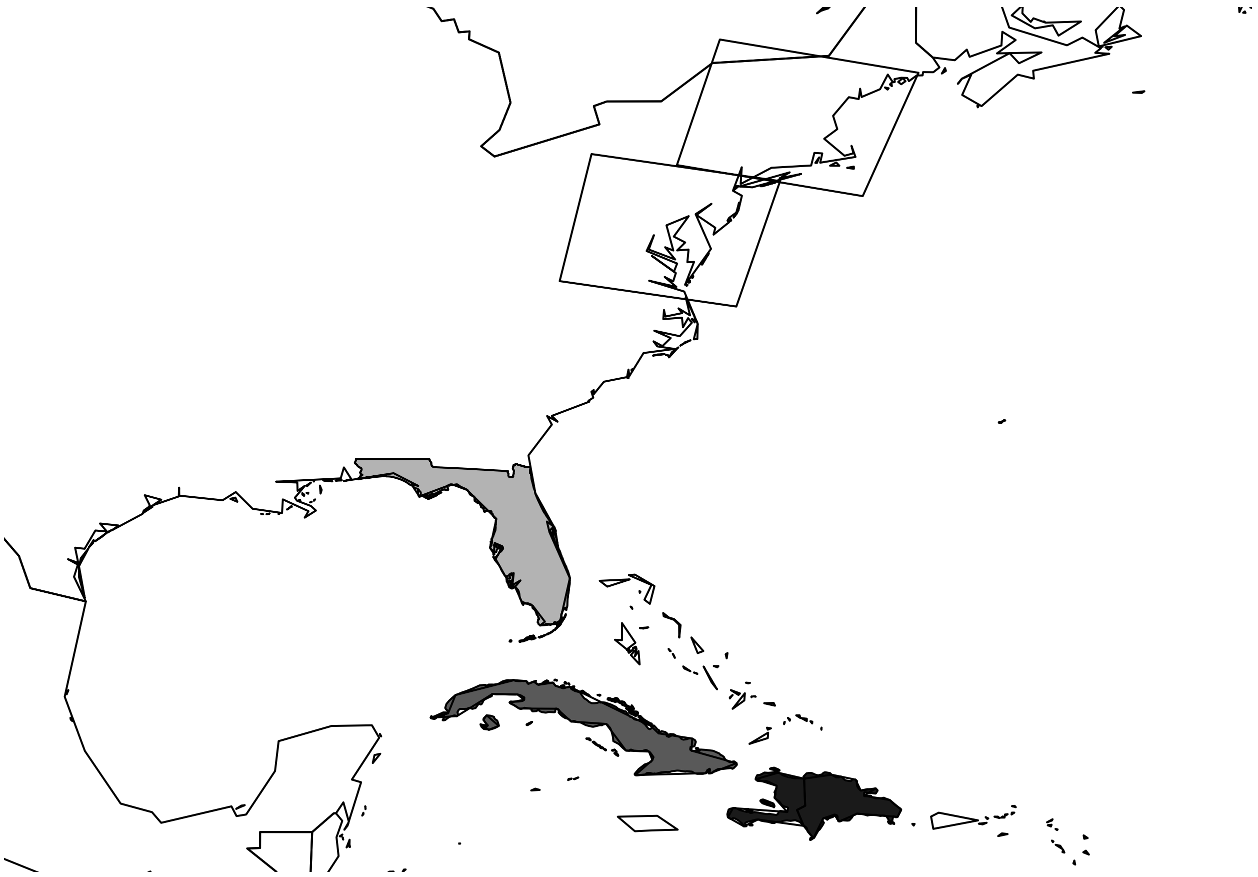
Load location data that accompanies the `MigConnectivity` package. The location data are data from breeding Ovenbirds that were fit with Light-level geolocators or PinPoint-10 GPS tags.

```
data(OVENdata) # Ovenbird

names(OVENdata)
#>  [1] "geo.bias"       "geo.vcov"       "isGL"           "targetPoints"
#>  [5] "originPoints"   "targetSites"    "originSites"    "originRelAbund"
#>  [9] "originDist"     "targetDist"
```

The figure below shows the two breeding regions (squares), and the three non-breeding regions (gray scale) used in Cohen et al. (*in press*) to estimate MC for Ovenbirds tracked with light-level geolocators and PinPoint-10 GPS tags.

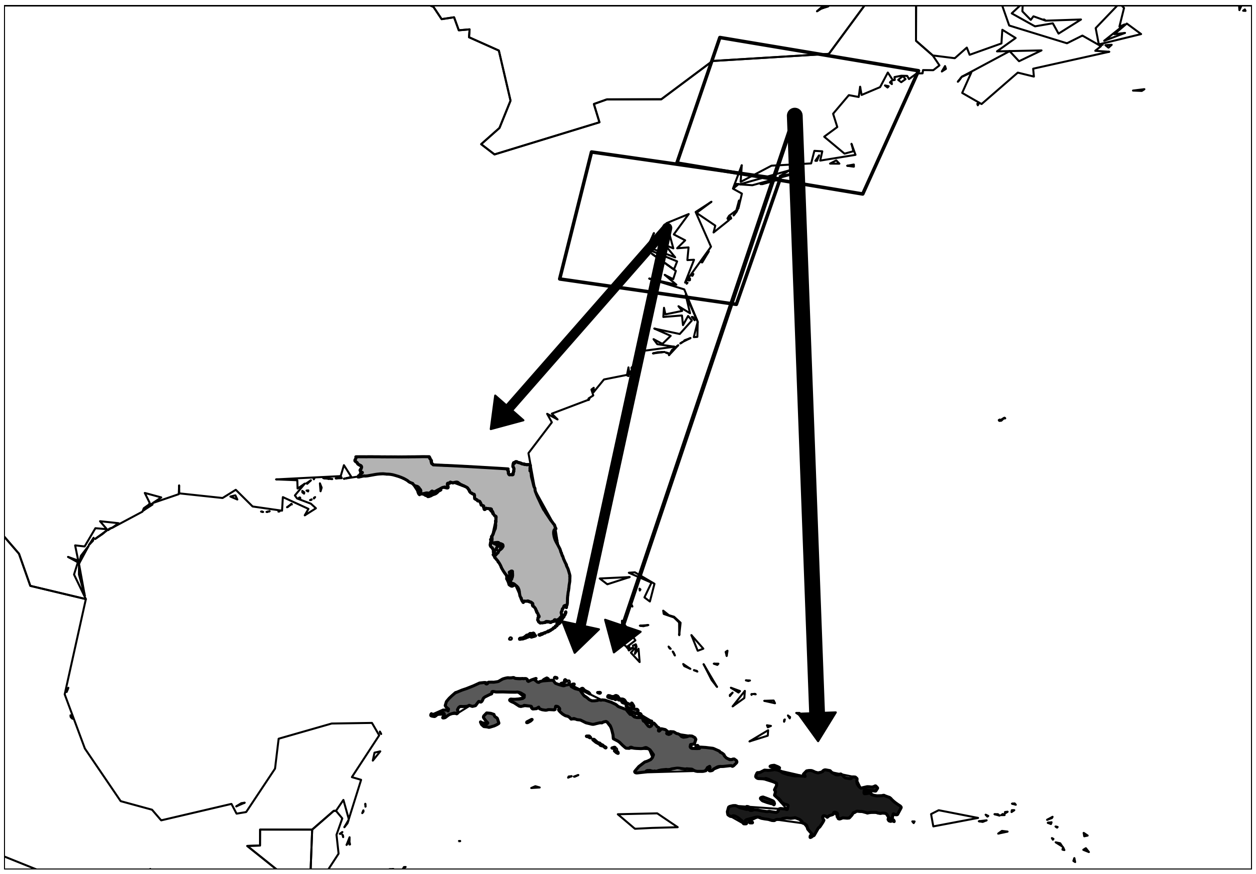
```
install.packages(c('rgeos', 'shape', 'raster', 'maptools', 'rgdal'))
```

```
library(rgeos)
library(shape)
library(raster)
library(maptools)
library(rgdal)

data(wrld_simpl)

raster::crs(wrld_simpl)
#> CRS arguments:
#>  +proj=longlat +ellps=WGS84 +datum=WGS84 +no_defs +towgs84=0,0,0

par(mar=c(0,0,0,0))
plot(OVENdata$originSites,
     xlim=c(raster::extent(OVENdata$targetSites)[1],
            raster::extent(OVENdata$targetSites)[2]),

     ylim=c(raster::extent(OVENdata$targetSites)[3],
            raster::extent(OVENdata$originSites)[4]))

plot(OVENdata$targetSites,
     add = TRUE,
     col=c("gray70","gray35","gray10"))

wrld_simple<-sp::spTransform(wrld_simpl,raster::crs(OVENdata$targetSites))

plot(wrld_simple,add=TRUE)
```



The following code demonstrates how to estimate MC using location data from light-level geolocators and PinPoint-10 GPS tags.

```r
M<-estMC(isGL=OVENdata$isGL, # Logical vector: light-level geolocator(T)/GPS(F)
         geoBias = OVENdata$geo.bias, # Light-level geolocator location bias
         geoVCov = OVENdata$geo.vcov, #Light-level geolocator covariance matrix
         targetDist = OVENdata$targetDist, # Target location distance matrix
         originDist = OVENdata$originDist, # Origin location distance matrix
         targetSites = OVENdata$targetSites, # Non-breeding / target sites
         originSites = OVENdata$originSites, # Breeding / origin sites
         originPoints = OVENdata$originPoints, # Capture Locations
         targetPoints = OVENdata$targetPoints, # Target locations from devices
         originRelAbund = OVENdata$originRelAbund, # Origin relative abundances
         resampleProjection = raster::projection(OVENdata$targetPoints),
         verbose = 0,   # output options - see help ??estMC
         nSamples = 10) # This is set low for example
```



```
str(M)
#> List of 22
#>  $ sampleMC       : num [1:10] 0.666 0.558 0.618 0.506 0.552 ...
#>  $ samplePsi      : num [1:10, 1:2, 1:3] 0 0 0 0 0 0 0 0 0 0 ...
#>   ..- attr(*, "dimnames")=List of 3
#>   .. ..$ : chr [1:10] "1" "2" "3" "4" ...
#>   .. ..$ : chr [1:2] "NH" "MD"
#>   .. ..$ : chr [1:3] "Florida" "Cuba" "Hisp"
#>  $ pointPsi       : num [1:2, 1:3] 0 0.4 0.174 0.6 0.826 ...
#>   ..- attr(*, "dimnames")=List of 2
#>   .. ..$ : chr [1:2] "NH" "MD"
#>   .. ..$ : chr [1:3] "Florida" "Cuba" "Hisp"
#>  $ pointMC        : num 0.621
#>  $ meanMC         : num 0.621
#>  $ medianMC       : num 0.603
#>  $ seMC           : num 0.0861
#>  $ simpleCI       : num [1:2] 0.506 0.811
#>  $ bcCI           : num [1:2] 0.51 0.811
#>  $ hpdCI          : num [1:2] 0.506 0.811
#>  $ simpleP        : NULL
#>  $ bcP            : NULL
#>  $ sampleCorr     : logi [1:10] NA NA NA NA NA NA ...
#>  $ pointCorr      : NULL
#>  $ meanCorr       : NULL
#>  $ medianCorr     : NULL
#>  $ seCorr         : NULL
#>  $ simpleCICorr   : NULL
#>  $ bcCICorr       : NULL
#>  $ inputSampleSize: int 39
#>  $ alpha          : num 0.05
#>  $ sigConst       : num 0
#>  - attr(*, "class")= chr [1:2] "estMC" "estMigConnectivity"
```

Stable-hydrogen isotopes

To estimate MC and include location uncertainty the following data are needed:

1. Distance matrix between breeding and non-breeding regions

2. A shapefile of both breeding and non-breeding regions

3. The capture locations and the 'unknown' locations derived from the isoAssign function.

4. The relative (or absolute) abundance within each region where the birds were captured(capture regions)

Below is a repeatable example for how to make isotope assignments, generate random point locations based on the assignments and to estimate the strength of migratory connectivity.

```
# Load in projections
data("projections")
# Read in distribution for the Ovenbird
OVENdist <- raster::shapefile("data-raw/Spatial_Layers/OVENdist.shp")
# subset out only the breeding distribution
OVENdist <- OVENdist[OVENdist$ORIGIN==2,] # only breeding
# assign coordinate reference system
crs(OVENdist) <- projections$WGS84

# Read in stable-hydrogen isotope values
OVENvals <- read.csv("data-raw/deltaDvalues.csv")
```

Read in raw isotope data collected from Ovenbirds from multiple locations.

Below is a brief walkthrough of how isotopes are typically used to assign breeding/molting origin. First, a 'tissue' or 'feather' isoscape is created either using 'known-origin' samples, or If those data are not available, values found in the literature. 'Tissues' or 'feathers' of 'known-origin' are used to generate the relationship between tissues values and the predicted isotope values in precipitation at sampling locations. The resulting relationship is then used to generate a 'tissue' or 'feather' isoscape that is then used to assign animals of unknown origin.

In the example below Ovenbirds were captured at known breeding locations in New Hampshire. The data are from Hallworth et al. 2013.

```
# THESE DATA NEED TO BE MOVED TO DATA-RAW FOLDER
OVENknown <- read.csv("F:/IsotopeGLmanuscript/Feathers_for_Fractionation_Equation.csv")
```

Get measured annual isotope values in precipitation

```
isomap <- getIsoMap(element = "Hydrogen",
                    period = "Annual")
```

Once you have predicted hydrogen isotope in precipitation, one can extract the predicted value for each known capture location.

```
predvals <- extract(isomap,cbind(OVENknown$lon,OVENknown$lat))
```

Use the extracted predicted values in a linear model to determine the intercept, slope and standard deviation of the residuals to turn the hydrogen isoscape into a feather isoscape used for assignment.

```
# linear model
iso.fit <- lm(OVENknown$dD~predvals)

# get the intercept and slope
coefficients(iso.fit)

# get standard deviation of residuals
sd(iso.fit$residuals)
```

```
a <- Sys.time()
b <- isoAssign(isovalues = OVENvals[,2],
               isoSTD = 13.73904, # from above
               intercept = -19.2703031,
               slope = 0.8079186,
               odds = NULL,
               restrict2Likely = TRUE,
               nSamples = 1000,
               sppShapefile = OVENdist,
               assignExtent = c(-179,-60,15,89),
               element = "Hydrogen",
               surface = FALSE,
               period = "Annual")
Sys.time()-a
```

We can now estimate the strength of MC from stable-hydrogen isotopes using the 1000 probable locations for each individual.

```
# Make origin sites - birds were captured in Florida and Jamaica
Countries <- shapefile("data-raw/Spatial_Layers/TM_WORLD_BORDERS-0.3.shp")

JAM <- Countries[Countries$NAME == "Jamaica",]
JAM <- as(JAM,"SpatialPolygons")

States <- shapefile("data-raw/Spatial_Layers/st99_d00.shp")

FL <- States[States$NAME == "Florida",]
FL <- rgeos::gUnaryUnion(FL,id = FL$STATE)

originSites <- rbind(JAM,FL)

# Distance between origin sites
originDist <- distFromPos(rgeos::gCentroid(originSites,byid = TRUE)@coords)

# Determine relative abundance based on number of birds captured
ever <- length(grep(OVENiso[,1],pattern = "EVER"))/
  length(grep(OVENiso[,1], pattern = "NH", invert = TRUE))
jam <- length(grep(OVENiso[,1],pattern = "JAM"))/
  length(grep(OVENiso[,1],pattern = "NH", invert = TRUE))
```

```r
originRelAbundance <- as.matrix(c(jam,ever))

# Generate random capture locations within each polygon as capture locations

set.seed(12)
originLongLat <- array(NA,c(length(grep(OVENiso[,1],pattern = "NH", invert = TRUE)),2))
sample1 <- sp::spsample(originSites[1, ],
                        length(grep(OVENiso[, 1], pattern = "JAM")),
                        type = "random")
sample2 <- sp::spsample(originSites[2, ],
                        length(grep(OVENiso[, 1],pattern = "EVER")),
                        type = "random")
originLongLat[grep(OVENiso[,1],pattern = "JAM"),]<-sample1@coords
originLongLat[grep(OVENiso[,1],pattern = "EVER"),]<-sample2@coords

originPoints <- sp::SpatialPoints(originLongLat,
                                  proj4string = sp::CRS(originSites@proj4string@projargs))

# Make sure all spatial layers have the same CRS (coordinate reference system)
originSites <- sp::spTransform(originSites,sp::CRS(projections$WGS84))
originPoints <- sp::spTransform(originPoints,sp::CRS(projections$WGS84))
```

```r
Sys.time()
a <- Sys.time()
ovenMC <- estMC(originRelAbund = originRelAbund,
              targetIntrinsic = b,
              originPoints = originPoints,
              originSites = originSites,
              originDist = originDist,
              verbose=0,
              calcCorr=TRUE,
              alpha = 0.05,
              approxSigTest = F,
              sigConst = 0,
              isIntrinsic = TRUE)
Sys.time()-a
Sys.time()
```