'MigConnectivity' package

Jeffrey A. Hostetler, Michael T. Hallworth

2022-09-19

Introduction

NOTE: this vignette is a bit out of date. Everything in it should still work (let us know if it doesn't), but it doesn't highlight the new functions from versions 0.3 and 0.4, such as `estStrength`, `estTransition`, and `diffMC`. We are working on an updated (or new) vignette that will highlight the new features.

Technological advancements have spurred rapid growth in the study of migratory connectivity, the spatial and temporal linkage of migratory populations throughout the annual cycle. However, the lack of a quantitative definition for the strength of migratory connectivity (MC) has limited our ability to draw inferences across species, studies, and data types about the seasonal co-occurrence of populations. MC is a standardized metric to quantify migratory connectivity between two phases of the annual cycle. It is independent of data type and accounts for the relative abundance of populations distributed across a seasonal range. Negative values of MC indicate that individuals close to each other in one season are further apart in the other season. If MC = 0, no relationship exists between distances in one season and another; if MC = 1, the relative distances between individuals in one season are the same in the other, although the scale can differ.

Three data inputs are needed to calculate MC, and one is optional:

1. The probabilities of movement between regions (i.e., transition probabilities);
2. Distances between regions within the two distinct seasonal ranges (e.g., a matrix of distances between all regions within the breeding range and a matrix of distances between all regions within the non-breeding range);
3. Relative abundance among regions within the seasonal range from which the transition probabilities originate (e.g., relative abundance among breeding regions);
4. (Optional) Total sample size of animals used to estimate the transition probabilities.

MC takes the distances between the regions within the seasonal range from which the transition probabilities originate (e.g. breeding) and the distances between the regions within the other seasonal range (e.g. non-breeding) and approximates the correlation between individuals at those distances. Earlier methods (Mantel correlation; rM) used the correlation between two distance matrices of individual animals captured during the breeding phase and recaptured during the non-breeding phase as a measure of the strength of migratory connectivity. Our method builds on this method with a distance-based correlation coefficient approximation, but is not specific to a data type and uses transition probabilities from discreet regions, as opposed to distance matrices between individuals. Further, we include relative abundance within one seasonal range to account for uneven sampling among regions.

Cohen, E. B., J.A. Hostetler, M.T. Hallworth, C.S. Rushing, T.S. Sillett, P.P. Marra. 2018. Quantifying the strength of migratory connectivity. Methods in Ecology and Evolution. 9:3. 513-524.

Overview of Functions

1. `calcMC` calculates the strength of migratory connectivity (MC) with the transitional probabilities between regions, distances between regions within the two seasonal ranges, relative abundance among

regions within the seasonal range from which the transition probabilities originate, and (optionally) the total sample size of the transitional probability data.

2. `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 tracking data. 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.

3. `simMove` Simulates annual movement of animals between regions across seasons for years and months with individual variability and strength of migratory connectivity (MC). Incorporates movement among regions within a season and movement among regions between seasons. Between seasons, animals either return to the same region each year or return to a different region in the subsequent year (dispersal). Between year dispersal rates occur during the first year (natal) or any subsequent year (breeding or non-breeding). Simulation does not incorporate births or deaths.

Getting Started

Installing 'MigConnectivity' from GitHub

In order to install the vignette along with the package use the following code with `build_vignettes = TRUE`. **Note** it takes quite a bit longer to download the package when `build_vignettes = TRUE`.

```
install.packages('devtools')

devtools::install_github("SMBC-NZP/MigConnectivity", build_vignettes = TRUE)

library(MigConnectivity)
```

Functions

`calcMC` Calculate strength of migratory connectivity (MC)
`estMC` Estimate the strength of migratory connectivity (MC) while incorporating sampling uncertainty
`simMove` Simulates annual and seasonal movement of animals between regions

Examples

Utility functions for use in vignette

`mlogitMat`: Calculates probability matrix based on exponential decline with distance

```
mlogitMat <- function(slope, dist) {
   preMat <- exp(-slope/mean(dist)*dist)
   diag(preMat) <- 0
   outMat <- t(apply(preMat, 1, function(x) x/(1 + sum(x))))
   diag(outMat) <- 1 - rowSums(outMat)
   return(outMat)
}
```

`mlogitMC`: Crude optimization function for developing migratory connectivity pattern based on migratory connectivity strength. Uses same argument (origin.abund) for relative or absolute abundance.

```
mlogitMC <- function(slope,
                     MC.in,
                     origin.dist,
                     target.dist,
                     origin.abund,
                     sample.size) {
```

```
    nBreeding <- nrow(origin.dist)
    nWintering <- nrow(target.dist)
    psi <- mlogitMat(slope, origin.dist)

  if (any(psi<0))

    return(5*slope^2)

  MC <- calcMC(origin.dist,
               target.dist,
               originRelAbund = origin.abund,
               psi,
               sampleSize = sample.size)

  return((MC.in - MC)^2)
}
```

`calcStrengthInd`: rM function for individuals

```
calcStrengthInd <- function(originDist,
                            targetDist,
                            locations,
                            resamp=1000,
                            verbose = 0) {
nInd <- dim(locations)[1]
originDist2 <- targetDist2 <- matrix(0, nInd, nInd)
for (i in 1:(nInd-1)) {
 for (j in (i+1):nInd) {
      originDist2[i,j] <- originDist[locations[i,1,1,1], locations[j,1,1,1]]
      targetDist2[i,j] <- targetDist[locations[i,2,1,1], locations[j,2,1,1]]
      originDist2[j,i] <- originDist[locations[i,1,1,1], locations[j,1,1,1]]
      targetDist2[j,i] <- targetDist[locations[i,2,1,1], locations[j,2,1,1]]
    }
  }
return(ncf::mantel.test(originDist2, targetDist2, resamp=resamp, quiet = !verbose))
}
```

`calcPsiMC`: Simple approach to estimate psi matrix (transition probabilities) and MC from simulated or real data (does not incorporate sampling uncertainty)

```
calcPsiMC <- function(originDist,
                      targetDist,
                      originRelAbund,
                      locations,
                      years = 1,
                      months = 1,
                      verbose=F) {

  nOrigin <- nrow(originDist)
  nTarget <- nrow(targetDist)
  psiMat <- matrix(0, nOrigin, nTarget)
  nInd <- dim(locations)[1]
  nYears <- dim(locations)[3]
  nMonths <- dim(locations)[4]
for (i in 1:nInd) {
```

```
  if (i %% 1000 == 0 && verbose) #
      cat("Individual", i, "of", nInd, "\n")
    originMat <- locations[i, 1, years, months]
    targetMat <- locations[i, 2, years, months]
    bIndices <- which(!is.na(originMat))
    wIndices <- which(!is.na(targetMat))
  if (length(bIndices) && length(wIndices))
    for (bi in bIndices)
      for (wi in wIndices)
       psiMat[originMat[bi], targetMat[wi]] <- psiMat[originMat[bi],
                                                      targetMat[wi]] + 1
  }
  psiMat <- apply(psiMat, 2, "/", rowSums(psiMat))
  MC <- calcMC(originDist, targetDist, psi = psiMat,
               originRelAbund = originRelAbund, sampleSize = nInd)
  return(list(psi=psiMat, MC=MC))
}
```

changeLocations: transfers the simulated bird locations from the true populations to the researcher defined regions

```
changeLocations <- function(animalLoc,
                            breedingSiteTrans,
                            winteringSiteTrans) {
  animalLoc[,1,,] <- breedingSiteTrans[animalLoc[,1,,]]
  animalLoc[,2,,] <- winteringSiteTrans[animalLoc[,2,,]]
  return(animalLoc)
}
```

simLocationError: Simulates location error with defined bias and variance / covariance matrix

```
simLocationError <- function(targetPoints,
                             targetSites,
                                        geoBias,
                                        geoVCov,
                             projection,
                                        verbose = 0,
                                        nSim = 1000) {

if(inherits(targetPoints,"SpatialPoints")){
  targetPoints <- sf::st_as_sf(targetPoints)}
if(inherits(targetSites,"SpatialPolygons") |
   inherits(targetSites,"SpatialPolygonsDataFrame")){
  targetSites <- sf::st_as_sf(targetPoints)}

nAnimals <- nrow(targetPoints)

geoBias2 <- matrix(rep(geoBias, nSim), nrow=nSim, byrow=T)
target.sample <- rep(NA, nAnimals)
target.point.sample <- matrix(NA, nAnimals, 2)

for(i in 1:nAnimals){
  if (verbose > 0)
    cat('Animal', i, 'of', nAnimals)
  draws <- 0
```

```
  while (is.na(target.sample[i])) {
    draws <- draws + 1
    # Sample random point for each bird
    # from parametric distribution of NB error
    ps <- MASS::mvrnorm(n=nSim,
                  mu=cbind(sf::st_coordinates(targetPoints)[i,1],
                           sf::st_coordinates(targetPoints)[i,2]),
                  Sigma=geoVCov)+
                  geoBias2

    colnames(ps) <- c("Longitude","Latitude")

    point.sample <- sf::st_as_sf(
                  data.frame(ps),
                  coords = c("Longitude","Latitude"),
                  crs = sf::st_crs(targetPoints))

    # filtered to stay in NB areas (land)
    #target.sample0 <- sp::over(point.sample, targetSites)

    target.sample0 <- unlist(unclass(sf::st_intersects(point.sample,
                                                       targetSites)))
    target.sample[i]<-target.sample0[!is.na(target.sample0)][1]
  }

  target.point.sample[i, ]<-sf::st_coordinates(point.sample[!is.na(target.sample0),])[1,]
  if (verbose > 0)
    cat(' ', draws, 'draws\n')
}
  return(list(targetSample = target.sample,
              targetPointSample = target.point.sample))
}
```

toPoly: Helper function to convert a string of XY coordinates of centroids to polygons

```
toPoly <- function(siteCentroids,
                   projection.in = 4326,
                   projection.out = NA,
                   resolution = NA,
                   order.by.input = TRUE)
{
# This automatically sets the resolution so that all polygons touch and
# cover the entire surface. Alternatively the user can supply the
# resolution of the raster cells in the units of the input projection
#(projection.in)
  colnames(siteCentroids) <- c("Longitude","Latitude")

  if(is.na(resolution)){
     long <- unique(siteCentroids[,1])
     lat <- unique(siteCentroids[,2])
     long.res <- long[2]-long[1]
     lat.res <- lat[2]-lat[1]
     resolution <- c(long.res,long.res)
  }
```

```
  rast <- suppressWarnings(
          raster::rasterFromXYZ(cbind(cbind(siteCentroids),
                                      (1:nrow(siteCentroids))),
                                       res = resolution,
                                        crs = sf::st_crs(projection.in)$proj4string))

  polys <- raster::rasterToPolygons(rast)

  # Give unique names to polygons #
  polys <- sf::st_as_sf(polys)
  sf::st_crs(polys) <- sf::st_crs(projection.in)
 # polys <- sp::SpatialPolygons(polys@polygons)
 # raster::crs(polys)<-projection.in

  # Reorders the polygons to match that of the input #
  centers <- sf::st_as_sf(data.frame(siteCentroids),
                          coords = c("Longitude","Latitude"),
                          crs = projection.in)
  #centers <- sp::SpatialPoints(siteCentroids, sp::CRS(projection.in))

  if(order.by.input){
 # orders <- sp::over(centers,polys)
  orders <- suppressMessages(unlist(unclass(sf::st_intersects(centers,
                                                              polys,
                                                              sparse = TRUE))))
  polys <- polys[orders,]
  }
  #sp::spChFIDs(polys,as.character(1:length(polys)))

  if(!is.na(projection.out)){
    polys <- sf::st_transform(polys, projection.out)
  }

  return(polys)
}
```

EXAMPLE 1

`calcMC` - Calculate strength of migratory connectivity

This function calculates the strength of migratory connectivity between populations during two different time periods within the annual cycle. Below, we illustrate how to calculate the strength of MC between three breeding and non-breeding regions.

To calculate the strength of MC, you will need:

1. to define the number of breeding and non-breeding regions
2. the distance between the breeding regions and the distance between the non-breeding regions
3. the transition probabilities between each breeding and non-breeding region (`psi`)
4. the relative abundance within each of the regions where individuals originate from (here the breeding regions)
5. (optional) the total sample size of animals used to estimate the transition probabilities.

Calculate MC between breeding and non-breeding regions

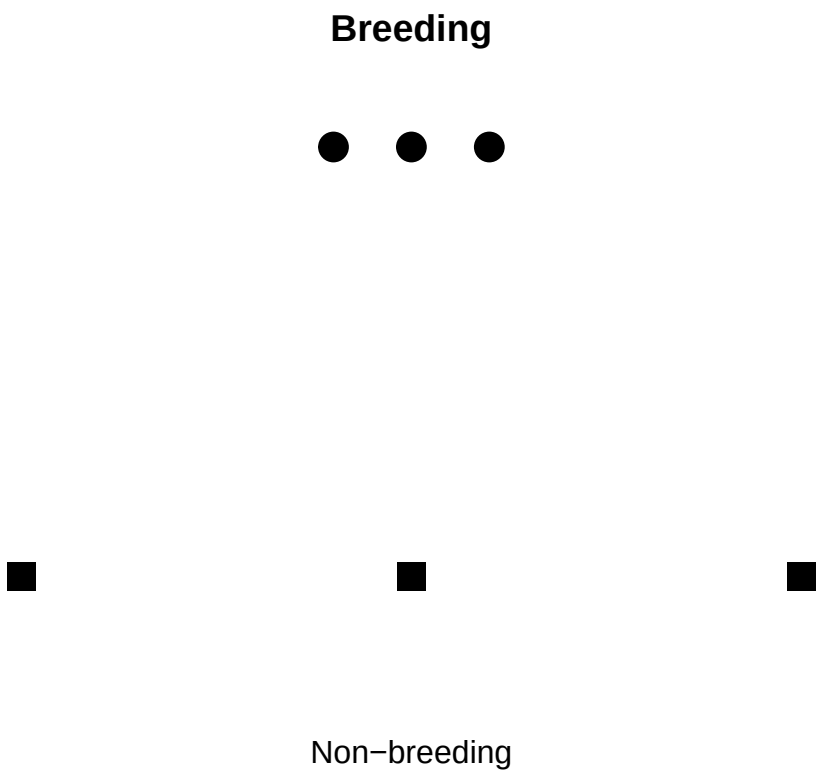
Simple example with three breeding and three non-breeding regions

Define the number of breeding and non-breeding populations

```
nBreeding <- 3 #number of breeding regions
nNonBreeding <- 3 #number of non-breeding regions
```

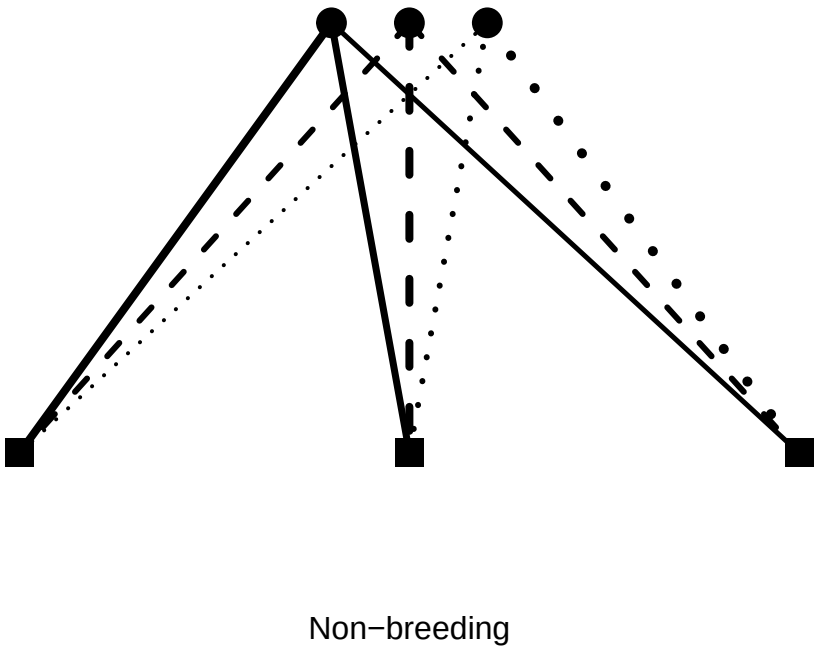
Generate a distance matrix between the different regions

```
#distances between centroids of three breeding regions
breedDist <- matrix(c(0, 1, 2,
                      1, 0, 1,
                      2, 1, 0), nBreeding, nBreeding)

#distances between centroids of three non-breeding regions
nonBreedDist <- matrix(c(0, 5, 10,
                         5, 0, 5,
                        10, 5, 0), nNonBreeding, nNonBreeding)
```



Define the transition probabilities between the breeding and non-breeding regions

```
#transition probabilities form each breeding to each non-breeding region
psi <- matrix(c(0.4, 0.35, 0.25,
                0.3, 0.4, 0.3,
                0.2, 0.3, 0.5), nBreeding, nNonBreeding, byrow = TRUE)
```



Define the relative abundance within the three breeding regions

```
#equal relative abundance among the three breeding regions, must sum to 1
relN <- rep(1/nBreeding, nBreeding)

relN
#> [1] 0.3333333 0.3333333 0.3333333

# Define total sample size for psi data
# for small sample corrected version of MC

n <- 250
```

Calculate the strength of migratory connectivity using the inputs above

```
# Calculate the strength of migratory connectivity
MC <- calcMC(originDist = breedDist,
             targetDist = nonBreedDist,
             originRelAbund = relN,
             psi = psi)

round(MC, 3)
#> [1] 0.05

MC <- calcMC(originDist = breedDist,
             targetDist = nonBreedDist,
             originRelAbund = relN,
             psi = psi,
```

```
             sampleSize = n)

round(MC, 3)
#> [1] 0.044
```

EXAMPLE 2

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.

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.

```
# 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)

colnames(captureLocations) <- c("Longitude","Latitude")

# Convert capture locations into SpatialPoints #

CapLocs <- sf::st_as_sf(data.frame(captureLocations),
                        coords = c("Longitude","Latitude"),
                        crs = 4326)

# Project Capture locations #

CapLocsM<-sf::st_transform(CapLocs, 'ESRI:54027')
```

```r
# 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){
                  sf::st_as_sf(data.frame(x),
                               coords = c("Longitude",
                                          "Latitude"),
                               crs = 4326)})

# Project into UTM projection #

NB_GLmeters <- lapply(NB_GL,
                      FUN = function(x){sf::st_transform(x,'ESRI:54027')})

# 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(sf::st_coordinates(x)[,1]-
                                               sf::st_coordinates(CapLocsM)[1,1])}))

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

LongError[PR] <- unlist(lapply(NB_GLmeters[PR],
                        FUN = function(x){mean(sf::st_coordinates(x)[,1]-
                                              sf::st_coordinates(CapLocsM)[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(sf::st_coordinates(x)[,2]-
                                               sf::st_coordinates(CapLocsM)[1,2])}))

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

LatError[PR] <- unlist(lapply(NB_GLmeters[PR],
                        FUN = function(x){mean(sf::st_coordinates(x)[,2]-
                                               sf::st_coordinates(CapLocsM)[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.

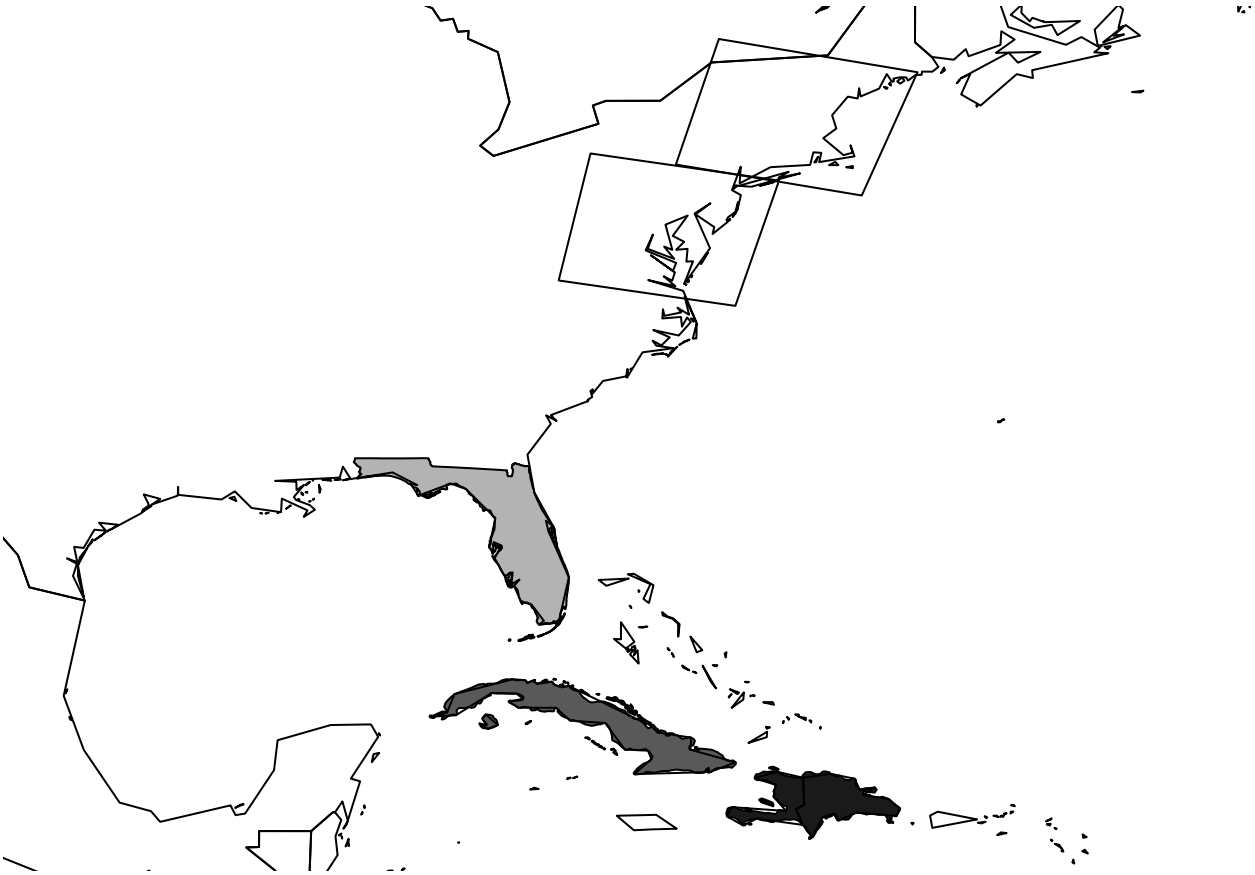
```
data(OVENdata) # Ovenbird

names(OVENdata)
```

```
#>  [1] "geo.bias"       "geo.vcov"       "isGL"           "targetPoints"
#>  [5] "originPoints"   "targetSites"    "originSites"    "originRelAbund"
#>  [9] "originDist"     "targetDist"     "originNames"    "targetNames"
```

The figure below shows the two breeding regions (squares), and the three non-breeding regions (gray scale) used in Cohen et al. (2018) 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)

wrld_simpl <- sf::st_as_sf(wrld_simpl, crs = 4326)

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<-sf::st_transform(wrld_simpl, sf::st_crs(OVENdata$originSites))

plot(sf::st_geometry(wrld_simple),add=TRUE)
```



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

```
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$originSites, # Breeding / origin sites
         targetNames = OVENdata$targetNames, # Names of nonbreeding/target sites
         originNames = OVENdata$originNames, # Names of breeding/origin sites
         originPoints = OVENdata$originPoints, # Capture Locations
         targetPoints = OVENdata$targetPoints, # Target locations from devices
         originRelAbund = OVENdata$originRelAbund, # Origin relative abundances
         resampleProjection = sf::st_crs(OVENdata$targetPoints),
         verbose = 0,   # output options - see help ??estMC
         nSamples = 10) # This is set low for example

M
#> Migratory Connectivity Estimates
#>
#> Transition probability (psi) estimates (mean):
#>        FL   Cuba   Hisp
#> NH 0.0000 0.1893 0.8107
#> MD 0.4167 0.5833 0.0000
#> +/- SE:
```

```
#>         FL    Cuba    Hisp
#> NH 0.0000 0.04808 0.04808
#> MD 0.1378 0.13777 0.00000
#> 95% confidence interval (simple quantile):
#>    FL              Cuba            Hisp
#> NH 0.0000 - 0.0000 0.1111 - 0.2500 0.7500 - 0.8889
#> MD 0.2500 - 0.6667 0.3333 - 0.7500 0.0000 - 0.0000
#>
#> MC estimate (mean): 0.6159 +/- (SE) 0.06817
#> 95% confidence interval (simple quantile): 0.5461 - 0.7623
#>
#> This is a subset of what's available inside this estMigConnectivity output.
#> For more info, try ?estTransition or ?estMC or str(obj_name, max.levels = 2).
```

Take a closer look at what is included in the output

```
str(M, max.level = 2)
#> List of 4
#>  $ psi  :List of 7
#>   ..$ sample  : num [1:10, 1:2, 1:3] 0 0 0 0 0 0 0 0 0 0 ...
#>   .. ..- attr(*, "dimnames")=List of 3
#>   ..$ mean    : num [1:2, 1:3] 0 0.417 0.189 0.583 0.811 ...
#>   .. ..- attr(*, "dimnames")=List of 2
#>   ..$ se      : num [1:2, 1:3] 0 0.1378 0.0481 0.1378 0.0481 ...
#>   .. ..- attr(*, "dimnames")=List of 2
#>   ..$ simpleCI: num [1:2, 1:2, 1:3] 0 0 0.25 0.667 0.111 ...
#>   .. ..- attr(*, "dimnames")=List of 3
#>   ..$ bcCI    : num [1:2, 1:2, 1:3] 0 0 0.25 0.667 0.111 ...
#>   .. ..- attr(*, "dimnames")=List of 3
#>   ..$ median  : num [1:2, 1:3] 0 0.401 0.201 0.599 0.799 ...
#>   .. ..- attr(*, "dimnames")=List of 2
#>   ..$ point   : num [1:2, 1:3] 0 0.4 0.174 0.6 0.826 ...
#>   .. ..- attr(*, "dimnames")=List of 2
#>  $ MC   :List of 10
#>   ..$ sample  : num [1:10] 0.546 0.625 0.609 0.549 0.705 ...
#>   ..$ mean    : num 0.616
#>   ..$ se      : num 0.0682
#>   ..$ simpleCI: num [1:2] 0.546 0.762
#>   ..$ bcCI    : num [1:2] 0.554 0.762
#>   ..$ hpdCI   : num [1:2] 0.546 0.762
#>   ..$ median  : num 0.599
#>   ..$ point   : num 0.621
#>   ..$ simpleP : NULL
#>   ..$ bcP     : NULL
#>  $ corr :List of 7
#>   ..$ sample  : logi [1:10] NA NA NA NA NA NA ...
#>   ..$ mean    : NULL
#>   ..$ se      : NULL
#>   ..$ simpleCI: NULL
#>   ..$ bcCI    : NULL
#>   ..$ median  : NULL
#>   ..$ point   : NULL
#>  $ input:List of 26
#>   ..$ originDist          : num [1:2, 1:2] 0 581231 581231 0
```

```
#>   .. ..- attr(*, "dimnames")=List of 2
#>   ..$ targetDist          : num [1:3, 1:3] 0 852312 1570037 852312 0 ...
#>   .. ..- attr(*, "dimnames")=List of 2
#>   ..$ originRelAbund      : num [1:2, 1] 0.623 0.377
#>   ..$ sampleSize          : int 39
#>   ..$ originSites         :Classes 'sf' and 'data.frame':    2 obs. of  1 variable:
#>   .. ..- attr(*, "sf_column")= chr "geometry"
#>   .. ..- attr(*, "agr")= Factor w/ 3 levels "constant","aggregate",..:
#>   .. .. ..- attr(*, "names")= chr(0)
#>   ..$ targetSites         :Classes 'sf' and 'data.frame':    3 obs. of  1 variable:
#>   .. ..- attr(*, "sf_column")= chr "geometry"
#>   .. ..- attr(*, "agr")= Factor w/ 3 levels "constant","aggregate",..:
#>   .. .. ..- attr(*, "names")= chr(0)
#>   ..$ originPoints        :Classes 'sf' and 'data.frame':    39 obs. of  1 variable:
#>   .. ..- attr(*, "sf_column")= chr "geometry"
#>   .. ..- attr(*, "agr")= Factor w/ 3 levels "constant","aggregate",..:
#>   .. .. ..- attr(*, "names")= chr(0)
#>   ..$ targetPoints        :Classes 'sf' and 'data.frame':    39 obs. of  1 variable:
#>   .. ..- attr(*, "sf_column")= chr "geometry"
#>   .. ..- attr(*, "agr")= Factor w/ 3 levels "constant","aggregate",..:
#>   .. .. ..- attr(*, "names")= chr(0)
#>   ..$ originAssignment    : int [1:39] 1 1 1 1 1 1 1 1 1 1 ...
#>   ..$ targetAssignment    : int [1:39] NA 3 3 3 NA 3 3 2 3 NA ...
#>   ..$ originNames         : chr [1:2] "NH" "MD"
#>   ..$ targetNames         : chr [1:3] "FL" "Cuba" "Hisp"
#>   ..$ nSamples            : num 10
#>   ..$ nSim                : num 1000
#>   ..$ isGL                : logi [1:39] TRUE TRUE TRUE TRUE TRUE TRUE ...
#>   ..$ geoBias             : num [1, 1:2] 2423 265432
#>   .. ..- attr(*, "dimnames")=List of 2
#>   ..$ geoVCov             : num [1:2, 1:2] 5.08e+08 -1.64e+09 -1.64e+09 1.28e+10
#>   .. ..- attr(*, "dimnames")=List of 2
#>   ..$ row0                : num 0
#>   ..$ verbose             : num 0
#>   ..$ calcCorr            : logi FALSE
#>   ..$ alpha               : num 0.05
#>   ..$ approxSigTest       : logi FALSE
#>   ..$ sigConst            : num 0
#>   ..$ resampleProjection  :List of 2
#>   .. ..- attr(*, "class")= chr "crs"
#>   ..$ maxTries            : num 300
#>   ..$ maintainLegacyOutput: logi FALSE
#>  - attr(*, "class")= chr [1:3] "estMC" "estPsi" "estMigConnectivity"
```

EXAMPLE 3

`simMove` Simulates position of animals by individual, season, year, and month

Calculate the strength of migratory connectivity based on simulation data for 10 years of movement between breeding and non-breeding seasonal ranges. Breeding and non-breeding ranges are equally divided into 100 regions on an ellipsoid globe. See above for details regarding the utility functions.

Simulate data to demonstrate the use of `calcMC`

```r
set.seed(75)

# Parameters for simulations

nSeasons <- 2 # population with two discrete seasons - breeding / non-breeding
nYears <- 10 # Ten years of data
nMonths <- 4 # Four months within each season

nBreeding <- 100 # Number of populations
nWintering <- 100 # Number of populations
```

Generate the spatial arrangement of breeding and non-breeding populations

```r
breedingPos <- matrix(c(rep(seq(-99,-81,2), each=sqrt(nBreeding)),
                        rep(seq(49,31,-2), sqrt(nBreeding))), nBreeding, 2)
winteringPos <- matrix(c(rep(seq(-79,-61,2), each=sqrt(nWintering)),
                         rep(seq(9,-9,-2), sqrt(nWintering))), nWintering, 2)

# calculate distance between populations
breedDist <- distFromPos(breedingPos, 'ellipsoid')

nonbreedDist <- distFromPos(winteringPos, 'ellipsoid')
```

The relative abundance of the study species is needed in each population. Here we generate those data below.

```r
# Breeding Abundance
breedingN <- rep(500, nBreeding)
breedingRelN <- breedingN/sum(breedingN)
```

The transition probability (psi) between the breeding and non-breeding populations is calculated below.

```r
# Set up psi matrix
o <- optimize(mlogitMC, MC.in = 0.25, origin.dist = breedDist,
              target.dist = nonbreedDist, origin.abund = breedingRelN,
              sample.size = sum(breedingN),
              interval = c(1, 3), tol = .Machine$double.eps^0.5)#

slope <- o$minimum

psi <- mlogitMat(slope, breedDist)
```

Now that we have all the data needed to calculate the strength of migratory connectivity we can use the `calcMC` function in the `MigConnecitivty` package to generate a standardized MC metric.

```r
# Baseline strength of migratory connectivity
MC <- calcMC(originDist = breedDist,
             targetDist = nonbreedDist,
             psi = psi,
             originRelAbund = breedingRelN,
             sampleSize = sum(breedingN))

# Show Results
MC
#> [1] 0.25
```

`simMove` Simulation of movement between breeding and non-breeding

Recycle data generated for the calcMC function - see above

```
sims <- simMove(breedingAbund = breedingN, # Breeding relative abundance
                breedingDist = breedDist,  # Breeding distance
                winteringDist = nonbreedDist, # Non-breeding distance
                psi = psi, # Transition probabilities
                nYears = nYears, # Number of years
                nMonths = nMonths, # Number of months
                winMoveRate = 0, # winter movement rate
                sumMoveRate = 0, # breeding movement rate
                winDispRate = 0, # winter dispersal rate
                sumDispRate = 0, # summer disperal rate
                natalDispRate = 0, # natal dispersal rate
                breedDispRate = 0, # breeding dispersal rate
                verbose = 0) # verbose output


str(sims)
#> List of 6
#>  $ animalLoc   : int [1:50000, 1:2, 1:10, 1:4] 1 1 1 1 1 1 1 1 1 1 ...
#>  $ natalDispMat: NULL
#>  $ breedDispMat: NULL
#>  $ sumMoveMat  : NULL
#>  $ winDispMat  : NULL
#>  $ winMoveMat  : NULL
```

Additional Examples / Scenarios

EXAMPLE 4

Calculate migratory connectivity for a range of input values

Cohen et al. (*2018*) used simulation to assess the influence of the three data inputs needed to calculate MC, including:

1. matrix of transition probabilities from breeding regions to non-breeding regions
2. matrices of distances between breeding regions and between non-breeding regions
3. relative abundance among breeding regions.

Input value simulations included transition probabilities from four breeding to four non-breeding regions and relative abundance measured among breeding regions.

There were eight transition probability scenarios:

1. Full Mix
2. Avoid One Site
3. Full Connectivity
4. Half Mix
5. Low
6. Medium
7. One Site Preference
8. Negative

There were twelve spatial arrangement / distance scenarios:

1. Base distances, linear/ linear
2. Distance between breeding sites 2 and 3 doubled
3. Distance between breeding sites 2 and 3 halved
4. Distance between breeding sites 3 and 4 doubled
5. Distance between breeding sites 3 and 4 halved
6. Breeding sites on square grid/ winter linear
7. Distance between wintering sites 2 and 3 doubled
8. Distance between wintering sites 2 and 3 halved
9. Distance between wintering sites 3 and 4 doubled
10. Distance between wintering sites 3 and 4 halved
11. Breeding linear, Wintering sites on square grid
12. Wintering and breeding on square grid

There were five relative breeding abundance scenarios:

1. Base, all equal
2. Abundance at site B doubled
3. Abundance at site B halved
4. Abundance at site D doubled
5. Abundance at site D halved

First we calculate MC for the eight transition probability scenarios (with base spatial arrangement and relative abundance):

```
nScenarios1 <- length(samplePsis) # samplePsis - comes with MigConnectivity package

# Create vector of length nScenarios1
MC1 <- rep(NA, nScenarios1)

# Loop through the different senarios outlined above #
for (i in 1:nScenarios1) {
  MC1[i] <- calcMC(originDist = sampleOriginDist[[1]],
                   targetDist = sampleTargetDist[[1]],
                   psi = samplePsis[[i]],
                   originRelAbund = sampleOriginRelN[[1]])
}

# Give meaningful names to the MC1 vector
names(MC1) <- names(samplePsis)

# Print results
round(MC1, 6)
#>            Full Mix      Avoid One Site   Full Connectivity            Half Mix
```

```
#>            0.000000            0.000000            1.000000            0.600000
#>                 Low              Medium One Site Preference            Negative
#>            0.196000            0.504000            0.164145           -0.066560
```

Add the scenarios for the spatial arrangements that result in different distances between regions:

```
nScenarios2 <- length(sampleOriginPos)

MC2 <- matrix(NA, nScenarios1, nScenarios2)

rownames(MC2) <- names(samplePsis)

colnames(MC2) <- names(sampleOriginPos)

for (i in 1:nScenarios1) {
  for (j in 1:nScenarios2) {
    MC2[i, j] <- calcMC(originDist = sampleOriginDist[[j]],
                        targetDist = sampleTargetDist[[j]],
                        psi = samplePsis[[i]],
                        originRelAbund = sampleOriginRelN[[1]])
  }
}

# Print results #
t(round(MC2, 4))
#>              Full Mix Avoid One Site Full Connectivity Half Mix    Low Medium
#> Linear              0              0            1.0000   0.6000 0.1960 0.5040
#> B Dist BC*2         0              0            0.9739   0.6956 0.1818 0.4879
#> B Dist BC/2         0              0            0.9798   0.4899 0.2000 0.4964
#> B Dist CD*2         0              0            0.9558   0.5147 0.1912 0.4784
#> B Dist CD/2         0              0            0.9734   0.6299 0.1878 0.4932
#> B Grid              0              0            0.7734   0.5256 0.1277 0.3601
#> NB Dist 23*2        0              0            0.9739   0.6956 0.1818 0.4879
#> NB Dist 23/2        0              0            0.9798   0.4899 0.2000 0.4964
#> NB Dist 34*2        0              0            0.9558   0.5147 0.1931 0.4882
#> NB Dist 34/2        0              0            0.9734   0.6299 0.1863 0.4856
#> NB Grid             0              0            0.7734   0.5256 0.1277 0.3601
#> B/NB Grid           0              0            1.0000   0.4605 0.1638 0.4340
#>              One Site Preference Negative
#> Linear                    0.1641  -0.0666
#> B Dist BC*2               0.1514  -0.0463
#> B Dist BC/2               0.1683  -0.0815
#> B Dist CD*2               0.1838  -0.0734
#> B Dist CD/2               0.1388  -0.0572
#> B Grid                    0.1695   0.0669
#> NB Dist 23*2              0.1428  -0.0463
#> NB Dist 23/2              0.1781  -0.0815
#> NB Dist 34*2              0.2163  -0.0734
#> NB Dist 34/2              0.1149  -0.0572
#> NB Grid                   0.1263   0.0669
#> B/NB Grid                 0.1603   0.0728
```

Another way of comparing results:

```
MC.diff2 <- apply(MC2, 2, "-", MC2[ , 1])

t(round(MC.diff2, 4))
#>              Full Mix Avoid One Site Full Connectivity Half Mix     Low  Medium
#> Linear              0              0            0.0000   0.0000  0.0000  0.0000
#> B Dist BC*2         0              0           -0.0261   0.0956 -0.0142 -0.0161
#> B Dist BC/2         0              0           -0.0202  -0.1101  0.0040 -0.0076
#> B Dist CD*2         0              0           -0.0442  -0.0853 -0.0048 -0.0256
#> B Dist CD/2         0              0           -0.0266   0.0299 -0.0082 -0.0108
#> B Grid              0              0           -0.2266  -0.0744 -0.0683 -0.1439
#> NB Dist 23*2        0              0           -0.0261   0.0956 -0.0142 -0.0161
#> NB Dist 23/2        0              0           -0.0202  -0.1101  0.0040 -0.0076
#> NB Dist 34*2        0              0           -0.0442  -0.0853 -0.0029 -0.0158
#> NB Dist 34/2        0              0           -0.0266   0.0299 -0.0097 -0.0184
#> NB Grid             0              0           -0.2266  -0.0744 -0.0683 -0.1439
#> B/NB Grid           0              0            0.0000  -0.1395 -0.0322 -0.0700
#>              One Site Preference Negative
#> Linear                    0.0000   0.0000
#> B Dist BC*2              -0.0128   0.0203
#> B Dist BC/2               0.0042  -0.0150
#> B Dist CD*2               0.0197  -0.0068
#> B Dist CD/2              -0.0253   0.0094
#> B Grid                    0.0054   0.1334
#> NB Dist 23*2             -0.0213   0.0203
#> NB Dist 23/2              0.0139  -0.0150
#> NB Dist 34*2              0.0521  -0.0068
#> NB Dist 34/2             -0.0493   0.0094
#> NB Grid                  -0.0378   0.1334
#> B/NB Grid                -0.0038   0.1394
```

Add the relative abundance breeding scenarios

Calculate MC across abundance and transition probability scenarios for breeding linear / winter linear arrangment:

```
nScenarios3 <- length(sampleOriginRelN)

MC3 <- matrix(NA, nScenarios1, nScenarios3)

rownames(MC3) <- names(samplePsis)
colnames(MC3) <- names(sampleOriginRelN)

for (i in 1:nScenarios1) {
for (j in 1) {
for (k in 1:nScenarios3) {
MC3[i, k] <- calcMC(originDist = sampleOriginDist[[j]],
                    targetDist = sampleTargetDist[[j]],
                    psi = samplePsis[[i]],
                    originRelAbund = sampleOriginRelN[[k]])
}
}
}
```

```
t(round(MC3, 4)) # linear arrangement
#>        Full Mix Avoid One Site Full Connectivity Half Mix    Low Medium
#> Base          0              0                 1   0.6000 0.1960 0.5040
#> B Doub        0              0                 1   0.5856 0.2060 0.5173
#> B Half        0              0                 1   0.6323 0.1942 0.5027
#> D Doub        0              0                 1   0.6360 0.1663 0.4608
#> D Half        0              0                 1   0.5807 0.2062 0.5235
#>        One Site Preference Negative
#> Base                0.1641  -0.0666
#> B Doub              0.1338  -0.0650
#> B Half              0.1847  -0.0660
#> D Doub              0.2992  -0.0219
#> D Half              0.0886  -0.0399
```

Calculate MC across abundance and transition probability scenarios for breeding grid/ winter grid arrangement:

```
MC4 <- matrix(NA, nScenarios1, nScenarios3)

rownames(MC4) <- names(samplePsis)
colnames(MC4) <- names(sampleOriginRelN)

for (i in 1:nScenarios1) {
for (j in nScenarios2) {
for (k in 1:nScenarios3) {
MC4[i, k] <- calcMC(originDist = sampleOriginDist[[j]],
                    targetDist = sampleTargetDist[[j]],
                    psi = samplePsis[[i]],
                    originRelAbund = sampleOriginRelN[[k]])
}
}
}

t(round(MC4, 4)) # grid arrangement
#>        Full Mix Avoid One Site Full Connectivity Half Mix    Low Medium
#> Base          0              0                 1   0.4605 0.1638 0.4340
#> B Doub        0              0                 1   0.4668 0.1614 0.4293
#> B Half        0              0                 1   0.4639 0.1699 0.4445
#> D Doub        0              0                 1   0.4668 0.1614 0.4293
#> D Half        0              0                 1   0.4639 0.1699 0.4445
#>        One Site Preference Negative
#> Base                0.1603   0.0728
#> B Doub              0.0970   0.0822
#> B Half              0.1996   0.0778
#> D Doub              0.3649   0.0822
#> D Half              0.0519   0.0778
```

Calculate MC across abundance and transition probability scenarios for for breeding grid, winter linear arrangement

```
MC5 <- matrix(NA, nScenarios1, nScenarios3)

rownames(MC5) <- names(samplePsis)
colnames(MC5) <- names(sampleOriginRelN)

for (i in 1:nScenarios1) {
```

```
for (j in 6) {
for (k in 1:nScenarios3) {
MC5[i, k] <- calcMC(originDist = sampleOriginDist[[j]],
                    targetDist = sampleTargetDist[[j]],
                    psi = samplePsis[[i]],
                    originRelAbund = sampleOriginRelN[[k]])
}
}
}

t(round(MC5, 4)) # breeding grid, winter linear arrangement
#>          Full Mix Avoid One Site Full Connectivity Half Mix    Low Medium
#> Base            0              0            0.7734   0.5256 0.1277 0.3601
#> B Doub          0              0            0.7538   0.5384 0.1380 0.3677
#> B Half          0              0            0.8229   0.5323 0.1385 0.3861
#> D Doub          0              0            0.8290   0.5384 0.1096 0.3456
#> D Half          0              0            0.7447   0.5323 0.1313 0.3629
#>          One Site Preference Negative
#> Base                   0.1695   0.0669
#> B Doub                 0.1140   0.0277
#> B Half                 0.2102   0.0801
#> D Doub                 0.3419   0.1334
#> D Half                 0.0737   0.0647
```

###EXAMPLES 5-12 (from sampling regime simulations in Cohen et al. 2018)

Simulation was used to assess the influence of biased or incomplete sampling on the measurement of MC. Below, we measure the influence of several potential sources of sampling error:

1. Incorrect grouping of seasonal ranges into regions
2. Sampling not proportional to abundance
3. Migratory connectivity strength varies across a range
4. Low sample size
5. Uncertainty in assignments to regions from tracking data
6. Uncertainty in estimates of abundance
7. Heterogeneity in detection among regions from mark re-encounter data

We also demonstrate how to propagate sampling uncertainty into measurement of MC using a combination of parametric and non-parametric bootstrapping. For most simulations, we compare MC to the results of a distance-based correlation coefficient (Mantel correlation; rM) that does not incorporate relative abundance but has been used as a measure of migratory connectivity.

###EXAMPLE 5

Sampling regime 1 of 3

Researchers divide populations differently than reality Delineation of seasonal ranges into regions

I) Breeding range divided along equal longitudinal breaks into ten regions
II) Non-breeding range divided along equal longitudinal breaks into ten regions
III) Breeding and non-breeding ranges divided along equal longitudinal breaks into ten regions
IV) Breeding range divided along the longitudinal and latitudinal midpoint into four regions
V) Non-breeding range divided along the longitudinal and latitudinal midpoint into four regions
VI) Breeding range divided along the longitudinal and latitudinal midpoint into four regions and non-breeding range divided along equal longitudinal breaks into ten regions
VII) Breeding range divided along equal longitudinal breaks into ten regions and non-breeding range divided along the longitudinal and latitudinal midpoint into four regions

```r
#Run functions and parameters above first
set.seed(75)

scenarios14 <- c("Base",
                 "Breeding10",
                 "Wintering10",
                 "Breeding10Wintering10",
                 "Breeding4",
                 "Wintering4",
                 "Breeding4Wintering10",
                 "Breeding10Wintering4",
                 "Breeding4Wintering4",
                 "CentroidSampleBreeding10",
                 "CentroidSampleBreeding10Wintering10",
                 "CentroidSampleBreeding10Wintering4",
                 "CentroidSampleBreeding4",
                 "CentroidSampleBreeding4Wintering10",
                 "CentroidSampleBreeding4Wintering4")

# Each element is for a scenario (see above 1-8), transferring from natural
# breeding populations to defined ones
breedingSiteTrans14 <- list(1:nBreeding,
                            rep(1:10, each=10),
                            1:nBreeding,
                            rep(1:10, each=10),
                            c(rep(1:2, 5, each=5), rep(3:4, 5, each=5)),
                            1:nBreeding,
                            c(rep(1:2, 5, each=5), rep(3:4, 5, each=5)),
                            rep(1:10, each=10),
                            c(rep(1:2, 5, each=5), rep(3:4, 5, each=5)),
                            rep(1:10, each=10),
                            rep(1:10, each=10),
                            rep(1:10, each=10),
                            c(rep(1:2, 5, each=5), rep(3:4, 5, each=5)),
                            c(rep(1:2, 5, each=5), rep(3:4, 5, each=5)),
                            c(rep(1:2, 5, each=5), rep(3:4, 5, each=5)))

# Same for non-breeding populations
winteringSiteTrans14 <- list(1:nWintering,
                             1:nWintering,
                             rep(1:10, each=10),
                             rep(1:10, each=10),
                             1:nWintering,
                             c(rep(1:2, 5, each=5), rep(3:4, 5, each=5)),
                             rep(1:10, each=10),
                             c(rep(1:2, 5, each=5), rep(3:4, 5, each=5)),
                             c(rep(1:2, 5, each=5), rep(3:4, 5, each=5)),
                             1:nWintering,
                             rep(1:10, each=10),
                             c(rep(1:2, 5, each=5), rep(3:4, 5, each=5)),
                             1:nWintering,
                             rep(1:10, each=10),
                             c(rep(1:2, 5, each=5), rep(3:4, 5, each=5)))
```

```r
# Examine the transfers in matrix form
#lapply(breedingSiteTrans14, matrix, nrow=10, ncol=10)
#lapply(winteringSiteTrans14, matrix, nrow=10, ncol=10)

#positions of the human defined populations
breedingPos14 <- list(breedingPos,
                      rowsum(breedingPos, rep(1:10, each=10))/10,
                      breedingPos,
                      rowsum(breedingPos, rep(1:10, each=10))/10,
                      rowsum(breedingPos, c(rep(1:2, 5, each=5),
                                            rep(3:4, 5, each=5)))/25,
                      breedingPos,
                      rowsum(breedingPos, c(rep(1:2, 5, each=5),
                                            rep(3:4, 5, each=5)))/25,
                      rowsum(breedingPos, rep(1:10, each=10))/10,
                      rowsum(breedingPos, c(rep(1:2, 5, each=5),
                                            rep(3:4, 5, each=5)))/25,
                      rowsum(breedingPos, rep(1:10, each=10))/10,
                      rowsum(breedingPos, rep(1:10, each=10))/10,
                      rowsum(breedingPos, rep(1:10, each=10))/10,
                      rowsum(breedingPos, c(rep(1:2, 5, each=5),
                                            rep(3:4, 5, each=5)))/25,
                      rowsum(breedingPos, c(rep(1:2, 5, each=5),
                                            rep(3:4, 5, each=5)))/25,
                      rowsum(breedingPos, c(rep(1:2, 5, each=5),
                                            rep(3:4, 5, each=5)))/25)

winteringPos14 <- list(winteringPos,
                       winteringPos,
                       rowsum(winteringPos, rep(1:10, each=10))/10,
                       rowsum(winteringPos, rep(1:10, each=10))/10,
                       winteringPos,
                       rowsum(winteringPos, c(rep(1:2, 5, each=5),
                                              rep(3:4, 5, each=5)))/25,
                       rowsum(winteringPos,rep(1:10, each=10))/10,
                       rowsum(winteringPos, c(rep(1:2, 5, each=5),
                                              rep(3:4, 5, each=5)))/25,
                       rowsum(winteringPos, c(rep(1:2, 5, each=5),
                                              rep(3:4, 5, each=5)))/25,
                       winteringPos,
                       rowsum(winteringPos, rep(1:10, each=10))/10,
                       rowsum(winteringPos, c(rep(1:2, 5, each=5),
                                              rep(3:4, 5, each=5)))/25,
                       winteringPos,
                       rowsum(winteringPos, rep(1:10, each=10))/10,
                       rowsum(winteringPos, c(rep(1:2, 5, each=5),
                                              rep(3:4, 5, each=5)))/25)


# Calculate distances between defined breeding populations
breedDist14 <- lapply(breedingPos14, distFromPos, surface = 'ellipsoid')

# Calculate distances between defined non-breeding populations
```

```r
nonbreedDist14 <- lapply(winteringPos14, distFromPos, surface = 'ellipsoid')

# Numbers of defined populations
nBreeding14 <- c(100, 10, 100, 10, 4, 100, 4, 10, 4, 10, 10, 10, 4, 4, 4)

nWintering14 <- c(100, 100, 10, 10, 100, 4, 10, 4, 4, 100, 10, 4, 100, 10, 4)

# Relative abundance by scenario and breeding population
breedingRelN14 <- lapply(nBreeding14, function(x) rep(1/x, x))

MC14 <- 0.25 # Same for all scenarios in this set

nSample14 <- 1000 # Total number sampled per simulation

# How many sampled from each natural population (sampling scenarios separate
# from definition scenarios)

sampleBreeding14 <- list(round(breedingRelN14[[1]]*nSample14),
                         c(rep(0, 22), round(breedingRelN14[[5]][1]*nSample14),
                           rep(0, 4), round(breedingRelN14[[5]][2]*nSample14),
                           rep(0, 44), round(breedingRelN14[[5]][3]*nSample14),
                           rep(0, 4), round(breedingRelN14[[5]][4]*nSample14),
                           rep(0, 22)),
                         rep(c(rep(0, 4),
                             rep(round(breedingRelN14[[2]][1]*nSample14/2), 2),
                             rep(0, 4)), 10))

# for the baseline use the simulation from above, sims

animalLoc14base <- sims$animalLoc


# Number of scenarios and number of simulations to run
nScenarios14 <- length(breedingSiteTrans14)
nSims14 <- 100

# Connections between scenarios and sampling regimes
scenarioToSampleMap14 <- c(rep(1, 9), rep(3, 3), rep(2, 3))

# Set up data structures for storing results
animalLoc14 <- vector("list", nScenarios14) #making an empty list to fill

compare14 <- data.frame(Scenario = c("True", scenarios14),
                        MC = c(MC14, rep(NA, nScenarios14)),
                        Mantel = c(MC14, rep(NA, nScenarios14)))

compare14.array <- array(NA, c(nSims14, nScenarios14, 2), dimnames =
                           list(1:nSims14,
                           scenarios14,
                           c("MC", "Mantel")))

results14 <- vector("list", nScenarios14)
```

```
# Run simulations
for (sim in 1:nSims14) {
  cat("Simulation", sim, "of", nSims14, '\n')
  sim14 <- lapply(sampleBreeding14,
                  simMove,
                  breedingDist = breedDist14[[1]],
                  winteringDist=nonbreedDist14[[1]],
                  psi=psi,
                  nYears=nYears,
                  nMonths=nMonths)

  for (i in 1:nScenarios14) {
   cat("\tScenario", i, "\n")
   animalLoc14[[i]]<-changeLocations(
     sim14[[scenarioToSampleMap14[i]]]$animalLoc,
                                    breedingSiteTrans14[[i]],
                                    winteringSiteTrans14[[i]])

   results14[[i]] <- calcPsiMC(breedDist14[[i]], nonbreedDist14[[i]],
                               animalLoc14[[i]],
                               originRelAbund = breedingRelN14[[i]],
                               verbose = F)

   compare14.array[sim, i, 'MC'] <- results14[[i]]$MC

   compare14.array[sim,i,'Mantel']<-calcStrengthInd(breedDist14[[1]],
                                                    nonbreedDist14[[1]],
                                   sim14[[scenarioToSampleMap14[i]]]$animalLoc,
                                                    resamp=0)$correlation
  }
}

# Compute means for each scenario
compare14$MC[1:nScenarios14 + 1] <- apply(compare14.array[,,'MC'], 2, mean)
compare14$Mantel[1:nScenarios14 + 1] <- apply(compare14.array[,,'Mantel'], 2,
                                              mean)

compare14 <- transform(compare14, MC.diff=MC - MC[1],
                       Mantel.diff=Mantel - Mantel[1])
compare14
```

###EXAMPLE 6

Sampling regime 2 of 3

Researchers divide populations differently than reality PLUS
Different distributions of sampling animals across breeding range PLUS
Sample sizes don't always match relative abundances PLUS
Compare our approach and simple Mantel approach

1. Base (10 years, uneven abundances but matches sampling)
2. Breeding pops divided into 4 squares, sample across breeding range
3. Breeding pops divided into 4 squares, sample at centroid of each square
4. Sampling high in low abundance populations plus base
5. Scenarios 2 plus 4

6. Scenarios 3 plus 4

```
set.seed(75)

# Transfer between true populations and researcher defined ones (only for
# breeding, as not messing with winter populations here)

breedingSiteTrans15 <- list(1:100,
                            c(rep(1:2, 5, each=5), rep(3:4, 5, each=5)),
                            c(rep(1:2, 5, each=5), rep(3:4, 5, each=5)),
                            1:100,
                            c(rep(1:2, 5, each=5), rep(3:4, 5, each=5)),
                            c(rep(1:2, 5, each=5), rep(3:4, 5, each=5)))

nScenarios15 <- length(breedingSiteTrans15)

nSims15 <- 100

# Basing positions of researcher defined breeding populations on above
breedingPos15 <- list(breedingPos,
                      breedingPos14[[5]],
                      breedingPos14[[5]],
                      breedingPos,
                      breedingPos14[[5]],
                      breedingPos14[[5]])

winteringPos15 <- rep(list(winteringPos), nScenarios15)

breedDist15 <- lapply(breedingPos15, distFromPos)

nonbreedDist15 <- lapply(winteringPos15, distFromPos)

nBreeding15 <- rep(c(100, 4, 4), 2)

nWintering15 <- rep(100, nScenarios15)

# Highest abundance in lower right corner, lowest in top left
# Making symmetrical

breedingN15base <- rep(NA, 100)
for (i in 1:10) #row
  for (j in 1:10)  #column
    breedingN15base[i+10*(j-1)] <- 500 + 850*i*j

sum(breedingN15base)

# For researcher defined populations
breedingN15 <- lapply(breedingSiteTrans15, rowsum, x=breedingN15base)

breedingRelN15 <- lapply(breedingN15, "/", sum(breedingN15base))

nSample15 <- 1000 # Total number sampled per simulation

# Number sampled per natural population
```

```
sampleBreeding15 <- list(round(breedingRelN15[[1]]*nSample15),
                         c(rep(0, 22), round(breedingRelN15[[3]][1]*nSample15),
                           rep(0, 4), round(breedingRelN15[[3]][2]*nSample15),
                           rep(0, 44), round(breedingRelN15[[3]][3]*nSample15),
                           rep(0, 4), round(breedingRelN15[[3]][4]*nSample15),
                           rep(0, 22)),
                         round(breedingRelN15[[1]]*nSample15)[100:1],
                         c(rep(0, 22), round(breedingRelN15[[3]][1]*nSample15),
                           rep(0, 4), round(breedingRelN15[[3]][2]*nSample15),
                           rep(0, 44), round(breedingRelN15[[3]][3]*nSample15),
                           rep(0, 4), round(breedingRelN15[[3]][4]*nSample15),
                           rep(0, 22))[100:1])

# Set up psi matrix
o15 <- optimize(mlogitMC, MC.in = 0.25, origin.dist = breedDist15[[1]],
                target.dist = nonbreedDist15[[1]],
                origin.abund = breedingN15[[1]]/sum(breedingN15[[1]]),
                sample.size = sum(breedingN15[[1]]),
                interval = c(0,10),
                tol = .Machine$double.eps^0.5)

slope15 <- o15$minimum

psi15 <- mlogitMat(slope15, breedDist15[[1]])

# Baseline strength of migratory connectivity
MC15 <- calcMC(originDist = breedDist15[[1]],
               targetDist = nonbreedDist15[[1]],
               psi = psi15,
               originRelAbund = breedingN15[[1]]/sum(breedingN15[[1]]),
               sampleSize = sum(breedingN15[[1]]))

# Run sampling regimes
scenarioToSampleMap15 <- c(1, 1, 2, 3, 3, 4)

animalLoc15 <- vector("list", nScenarios15)

results15 <- vector("list", nScenarios15)

compare15 <- data.frame(Scenario = c("True",
                                     "Base",
                                     "Breeding4",
                                     "CentroidSampleBreeding4",
                                     "BiasedSample",
                                     "BiasedSampleBreeding4",
                                     "BiasedCentroidSampleBreeding4"),
                        MC = c(MC15, rep(NA, nScenarios15)),
                        Mantel = c(MC15, rep(NA, nScenarios15)))

compare15.array <- array(NA, c(nSims15, nScenarios15, 2),
                         dimnames = list(1:nSims15,
                                         c("Base", "Breeding4",
                                           "CentroidSampleBreeding4",
```

```
                                    "BiasedSample", "BiasedSampleBreeding4",
                                    "BiasedCentroidSampleBreeding4"),
                                  c("MC", "Mantel")))


for (sim in 1:nSims15) {
  cat("Simulation", sim, "of", nSims15, '\n')

  sim15 <- lapply(sampleBreeding15,
                  simMove,
                  breedingDist = breedDist15[[1]],
                  winteringDist=nonbreedDist15[[1]],
                  psi=psi15,
                  nYears=nYears,
                  nMonths=nMonths)

for (i in 1:nScenarios15) {
  cat("\tScenario", i, "\n")
    animalLoc15[[i]] <- changeLocations(
                        animalLoc=sim15[[scenarioToSampleMap15[i]]]$animalLoc,
                        breedingSiteTrans = breedingSiteTrans15[[i]],
                        winteringSiteTrans = 1:nWintering15[i])

    results15[[i]] <- calcPsiMC(originDist = breedDist15[[i]],
                                targetDist = nonbreedDist15[[i]],
                                originRelAbund = breedingRelN15[[i]],
                                locations = animalLoc15[[i]],
                                verbose = F)

compare15.array[sim, i, 'MC'] <- results15[[i]]$MC

compare15.array[sim, i, 'Mantel'] <- calcStrengthInd(breedDist15[[1]],
                                                     nonbreedDist15[[1]],
                                   sim15[[scenarioToSampleMap15[i]]]$animalLoc,
                                                     resamp=0)$correlation
  }
}

compare15$MC[1:nScenarios15+1] <- apply(compare15.array[,,'MC'], 2, mean,
                                        na.rm = TRUE)

compare15$Mantel[1:nScenarios15+1] <- apply(compare15.array[,,'Mantel'], 2,
                                            mean, na.rm = TRUE)
compare15 <- transform(compare15, MC.diff=MC - MC[1],
                       Mantel.diff=Mantel - Mantel[1],
                       MC.prop=MC/MC[1],
                       Mantel.prop=Mantel/Mantel[1])


compare15a <- as.matrix(compare15[1 + 1:nScenarios15,
                                  c("MC", "MC.diff", "Mantel", "Mantel.diff")])

rownames(compare15a) <- compare15$Scenario[1 + 1:nScenarios15]
```

###EXAMPLE 7 Sampling regime 3 of 3 Researchers divide populations differently than reality (simulations) PLUS Different distributions of sampled animals across breeding range PLUS Sample sizes don't always match relative abundances PLUS Compare our approach and simple Mantel approach PLUS MC not same across subsections of range

1. Base (uneven MC (0.15 for NW breeding, 0.3 for SW, 0.45 for NE, and 0.6 for SE), uneven abundances (lowest in NW, highest in SE), sampling proportional to abundance
2. Breeding pops divided into 4 squares, sample across breeding range
3. Breeding pops divided into 4 squares, sample at centroid of each square
4. Sampling high in low abundance populations
5. Scenarios 2 plus 4
6. Scenarios 3 plus 4

```
set.seed(75)

# Transfer between true populations and researcher defined ones
# (only for breeding, as not messing with winter populations here)
breedingSiteTrans16 <- list(1:100, c(rep(1:2, 5, each=5), rep(3:4, 5, each=5)),
                            c(rep(1:2, 5, each=5), rep(3:4, 5, each=5)), 1:100,
                            c(rep(1:2, 5, each=5), rep(3:4, 5, each=5)),
                            c(rep(1:2, 5, each=5), rep(3:4, 5, each=5)))

#lapply(breedingSiteTrans16, matrix, nrow=10, ncol=10)

nScenarios16 <- length(breedingSiteTrans16)

nSims16 <- 100

# Basing positions of researcher defined breeding populations on above
breedingPos16 <- breedingPos15
winteringPos16 <- winteringPos15

breedDist16 <- breedDist15
nonbreedDist16 <- nonbreedDist15
nBreeding16 <- nBreeding15
nWintering16 <- rep(100, nScenarios16)

# Highest abundance in lower right corner, lowest in top left
# In fact basing on distance from top left population

breedingN16base <- breedingN15base
breedingN16 <- breedingN15
breedingRelN16 <- lapply(breedingN16, "/", sum(breedingN16base))

# Set up psi matrix
# Each quadrant of breeding range has different MC

MC.levels16 <- seq(0.15, 0.6, 0.15)

nLevels16 <- 4

psi16 <- matrix(NA, nBreeding16[1], nWintering16[1])


for (i in 1:nLevels16) {
```

```
  cat("MC", MC.levels16[i])
  # Find a psi matrix that produces the given MC (for whole species)
  o16a <- optimize(mlogitMC, MC.in = MC.levels16[i],
                   origin.dist = breedDist16[[1]],
                   target.dist = nonbreedDist16[[1]],
                   origin.abund = breedingN16[[1]]/sum(breedingN16[[1]]),
                   sample.size = sum(breedingN16[[1]]),
                   interval=c(0,10), tol=.Machine$double.eps^0.5)

  slope16a <- o16a$minimum

  cat(" slope", slope16a, "\n")

  psi16a <- mlogitMat(slope16a, breedDist16[[1]])

  # Then use the rows of that psi matrix only for the one breeding quadrant
  rows <- 50*(i %/% 3) + rep(1:5, 5) + rep(seq(0, 40, 10), each=5) +
    ((i-1) %% 2) * 5
  psi16[rows, ] <- psi16a[rows, ]
}


# Baseline strength of migratory connectivity
MC16 <- calcMC(originDist = breedDist16[[1]],
               targetDist = nonbreedDist16[[1]],
               psi = psi16,
               originRelAbund = breedingN16[[1]]/sum(breedingN16[[1]]),
               sampleSize = sum(breedingN16[[1]]))


# Set up sampling regimes (different number than number of scenarios)
nSample16 <- 1000
sampleBreeding16 <- list(round(breedingRelN16[[1]]*nSample16),
                         c(rep(0, 22), round(breedingRelN16[[3]][1]*nSample16),
                           rep(0, 4), round(breedingRelN16[[3]][2]*nSample16),
                           rep(0, 44), round(breedingRelN16[[3]][3]*nSample16),
                           rep(0, 4), round(breedingRelN16[[3]][4]*nSample16),
                           rep(0, 22)),
                         round(breedingRelN16[[1]]*nSample16)[100:1],
                         c(rep(0, 22), round(breedingRelN16[[3]][1]*nSample16),
                           rep(0, 4), round(breedingRelN16[[3]][2]*nSample16),
                           rep(0, 44), round(breedingRelN16[[3]][3]*nSample16),
                           rep(0, 4), round(breedingRelN16[[3]][4]*nSample16),
                           rep(0, 22))[100:1])

sapply(sampleBreeding16, sum)

# Run sampling regimes
scenarioToSampleMap16 <- c(1, 1, 2, 3, 3, 4)
animalLoc16 <- vector("list", nScenarios16)
results16 <- vector("list", nScenarios16)
compare16 <- data.frame(Scenario = c("True",
                                     "Base",
```

```
                                    "Breeding4",
                                    "CentroidSampleBreeding4",
                                    "BiasedSample",
                                    "BiasedSampleBreeding4",
                                    "BiasedCentroidSampleBreeding4"),
                        MC = c(MC16, rep(NA, nScenarios16)),
                        Mantel = c(MC16, rep(NA, nScenarios16)))

compare16.array <- array(NA, c(nSims16, nScenarios16, 2),
                         dimnames = list(1:nSims16,
                                         c("Base", "Breeding4",
                                           "CentroidSampleBreeding4",
                                           "BiasedSample",
                                           "BiasedSampleBreeding4",
                                           "BiasedCentroidSampleBreeding4"),
                                         c("MC", "Mantel")))

set.seed(80)
for (sim in 1:nSims16) {
  cat("Simulation", sim, "of", nSims16, '\n')

  sim16 <- lapply(sampleBreeding16, simMove, breedingDist = breedDist16[[1]],
                  winteringDist=nonbreedDist16[[1]], psi=psi16, nYears=nYears,
                  nMonths=nMonths)

for (i in 1:nScenarios16) {
  cat("\tScenario", i, "\n")
  animalLoc16[[i]]<-changeLocations(sim16[[scenarioToSampleMap16[i]]]$animalLoc,
                                    breedingSiteTrans16[[i]],
                                    1:nWintering16[i])

  results16[[i]] <- calcPsiMC(originDist = breedDist16[[i]],
                              targetDist = nonbreedDist16[[i]],
                              locations = animalLoc16[[i]],
                              originRelAbund = breedingRelN16[[i]],
                              verbose = FALSE)

compare16.array[sim, i, 'MC'] <- results16[[i]]$MC

compare16.array[sim, i, 'Mantel'] <- calcStrengthInd(breedDist16[[1]],
                                                     nonbreedDist16[[1]],
                                  sim16[[scenarioToSampleMap16[i]]]$animalLoc,
                                                     resamp=0)$correlation
  }
}

compare16$MC[1:nScenarios16+1] <- apply(compare16.array[,,'MC'], 2, mean, na.rm = TRUE)

compare16$Mantel[1:nScenarios16+1] <- apply(compare16.array[,,'Mantel'], 2,
                                            mean, na.rm = TRUE)

compare16 <- transform(compare16, MC.diff=MC - MC[1],
                       Mantel.diff=Mantel - Mantel[1])
```

```
compare16a <- as.matrix(compare16[1 + 1:nScenarios16, c('MC','MC.diff',
                                                        "Mantel",
                                                        "Mantel.diff")])

rownames(compare16a) <- compare16$Scenario[1 + 1:nScenarios16]
```

###EXAMPLE 8 `estMC` Estimate strength of migratory connectivity incorporating uncertainty from limited sample size (no other sampling error). Here we simulated data using the 13th scenario from example 5 above ("CentroidSampleBreeding4"; make sure you run example 5 code before running this section). We changed the total sample size from 1000 to 100 animals and the total abundance from 50000 to 2500 animals.

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

set.seed(75)
capLocs14 <- lapply(breedingPos14,
                    FUN = function(x){
                      colnames(x)<-c("Longitude","Latitude")
                      sf::st_as_sf(data.frame(x),
                                   coords = c("Longitude","Latitude"),
                                   crs = 4326)})

targLocs14 <- lapply(winteringPos14,
                     FUN = function(x){
                      colnames(x)<-c("Longitude","Latitude")
                      sf::st_as_sf(data.frame(x),
                                   coords = c("Longitude","Latitude"),
                                   crs = 4326)})


# Relative abundance by scenario and breeding population
breedingN14base <- rep(25, nBreeding14[1])
breedingN14 <- lapply(breedingSiteTrans14, rowsum, x=breedingN14base)
breedingRelN14 <- lapply(breedingN14, "/", sum(breedingN14base))


MC14 <- 0.25

nSample14.1 <- 100 # Total number sampled per simulation

# How many sampled from each natural population (sampling scenarios separate
# from definition scenarios)
sampleBreeding14.1 <- list(round(breedingRelN14[[1]]*nSample14.1),
                           c(rep(0,22),round(breedingRelN14[[5]][1]*nSample14.1),
                             rep(0,4), round(breedingRelN14[[5]][2]*nSample14.1),
                             rep(0,44),round(breedingRelN14[[5]][3]*nSample14.1),
                             rep(0,4), round(breedingRelN14[[5]][4]*nSample14.1),
                             rep(0, 22)),
                             rep(c(rep(0, 4),
                             rep(round(breedingRelN14[[2]][1]*nSample14.1/2), 2),
                             rep(0, 4)), 10))

lapply(sampleBreeding14.1, matrix, nrow=10, ncol=10)
```

```
# Number of simulations to run
nSims14 <- 100
nSimsLarge14 <- 250 #Originally 2500
nYears <- 1
nMonths <- 1

# Set up data structures for storing results
animalLoc14 <- vector("list", nScenarios14) #making an empty list to fill
sim14 <- vector("list", nSimsLarge14) #making an empty list to fill

compare14.1.array <- array(NA, c(nSimsLarge14, 2),
                           dimnames = list(1:nSimsLarge14,
                                           c("MC", "Mantel")))

results14 <- vector("list", nScenarios14)

# Run simulations
set.seed(7)

system.time(for (sim in 1:nSimsLarge14) {

  cat("Simulation", sim, "of", nSimsLarge14, '\n')

  sim14[[sim]] <- lapply(sampleBreeding14.1,
                         simMove,
                         breedingDist = breedDist14[[1]],
                         winteringDist=nonbreedDist14[[1]],
                         psi=psi,
                         nYears=nYears,
                         nMonths=nMonths)

for (i in 13) {
 animalLoc14[[i]] <- changeLocations(sim14[[sim]][[scenarioToSampleMap14[i]]]$animalLoc,
                                     breedingSiteTrans14[[i]],
                                     winteringSiteTrans14[[i]])

 results14[[i]] <- calcPsiMC(breedDist14[[i]],
                             nonbreedDist14[[i]],
                             animalLoc14[[i]],
                             originRelAbund = breedingRelN14[[i]],
                             verbose = F)

  compare14.1.array[sim,'MC'] <- results14[[i]]$MC

  compare14.1.array[sim,'Mantel'] <- calcStrengthInd(breedDist14[[1]],
                                                     nonbreedDist14[[1]],
                                sim14[[sim]][[scenarioToSampleMap14[i]]]$animalLoc,
                                                     resamp=0)$correlation
  }
})

means14 <- apply(compare14.1.array, 2, mean)
```

```
vars14 <- apply(compare14.1.array, 2, var)

rmse14 <- apply(compare14.1.array, 2, function(x) sqrt(mean((x - MC14)^2)))

# Set up data structures for storing estimation results
est14.array <- array(NA, c(nSims14, 2),
                     dimnames = list(1:nSims14,c("MC", "Mantel")))

var14.array <- array(NA, c(nSims14, 2),
                     dimnames =list(1:nSims14,c("MC", "Mantel")))

ci14.array <- array(NA, c(nSims14, 2, 2),
                    dimnames = list(1:nSims14,
                                    c("MC", "Mantel"),
                                    c('lower', 'upper')))

#making an empty list to fill

animalLoc14 <- vector("list", nSims14)

results14 <- vector("list", nSims14)

# Run estimations
set.seed(567)

sim14.sub <- sim14[sample.int(nSimsLarge14, nSims14, T)]

for (sim in 1:nSims14) {
  cat("Estimation", sim, "of", nSims14, '\n')

 for (i in 13) {#:nScenarios14) {
  animalLoc14[[sim]] <-
    changeLocations(sim14.sub[[sim]][[scenarioToSampleMap14[i]]]$animalLoc,
                                       breedingSiteTrans14[[i]],
                                       winteringSiteTrans14[[i]])

  results14[[sim]] <- estMC(originDist = breedDist14[[i]],
                            targetDist = nonbreedDist14[[i]],
                            originRelAbund = breedingRelN14[[i]],
              originPoints = capLocs14[[i]][animalLoc14[[sim]][, 1, 1, 1], ],
              targetPoints = targLocs14[[i]][animalLoc14[[sim]][, 2, 1, 1], ],
                            originAssignment = animalLoc14[[sim]][, 1, 1, 1],
                            targetAssignment = animalLoc14[[sim]][, 2, 1, 1],
                            nSamples = 1000,
                            verbose = 1,
                            calcCorr = T,
                            geoBias = c(0, 0),
                            geoVCov = matrix(0, 2, 2))

    est14.array[sim, 'MC'] <- results14[[sim]]$MC$mean
    est14.array[sim, 'Mantel'] <- results14[[sim]]$corr$mean
    var14.array[sim, 'MC'] <- results14[[sim]]$MC$se ^ 2
    var14.array[sim, 'Mantel'] <- results14[[sim]]$corr$se ^ 2
```

```
    ci14.array[sim, 'MC', ] <- results14[[sim]]$MC$bcCI
    ci14.array[sim, 'Mantel', ] <- results14[[sim]]$corr$bcCI
  }
}

# Crude point estimates (bootstrap means are better)
pointEsts14.1 <- t(sapply(results14, function(x) c(x$MC$point, x$corr$point)))

# Summarize results
summary(var14.array)
vars14
summary(est14.array)
summary(pointEsts14.1)
means14
colMeans(pointEsts14.1) - MC14
sqrt(colMeans((pointEsts14.1 - MC14)^2))
summary(ci14.array[, "MC", 'lower'] <= MC14 &
          ci14.array[, "MC", 'upper'] >= MC14)
summary(ci14.array[, "Mantel", 'lower'] <= MC14 &
          ci14.array[, "Mantel", 'upper'] >= MC14)

# Plot histograms of bootstrap estimation results
est.df <- data.frame(Parameter = rep(c("MC", 'rM'), 2, each = nSims14),
                     Quantity = rep(c('Mean', 'Variance'), each = 2 * nSims14),
                     sim = rep(1:nSims14, 4),
                     Estimate = c(est14.array, var14.array))

trues.df <- data.frame(Parameter = rep(c("MC", 'rM'), 2),
                       Quantity = rep(c('Mean', 'Variance'), each = 2),
                       Value = c(MC14, MC14, vars14))

library(ggplot2)
g.est <- ggplot(est.df, aes(Estimate)) + geom_histogram(bins = 15) +
  facet_grid(Parameter ~ Quantity, scales = 'free_x') +
  geom_vline(aes(xintercept = Value), data = trues.df) +
  theme_bw() + scale_x_continuous(breaks = c(0.002, 0.003, 0.1, 0.2, 0.3))

g.est


# Summary table of bootstrap estimation results
qualities14 <- data.frame(Parameter = rep(c("MC", 'rM'), 5),
                          Quantity = rep(rep(c('Mean', 'Variance'), c(2, 2)),
                                         3, 10),
                          Measure = rep(c('Bias', 'RMSE', "Coverage"),
                                        c(4, 4, 2)),
                          Value = c(colMeans(est14.array - MC14),
                                    colMeans(var14.array) - vars14,
                                    sqrt(colMeans((est14.array - MC14)^2)),
                                    sqrt(mean((var14.array[,1]-vars14[1])^2)),
                                    sqrt(mean((var14.array[,2]-vars14[2])^2)),
                                    mean(ci14.array[, "MC", 'lower'] <= MC14 &
                                         ci14.array[, "MC", 'upper'] >= MC14),
```

```
                                    mean(ci14.array[,"Mantel",'lower']<=MC14 &
                                         ci14.array[,"Mantel",'upper']>=MC14)))

format(qualities14, digits = 2, scientific = F)
```

###EXAMPLE 9 `estMC` Estimate strength of migratory connectivity incorporating uncertainty from location error associated with light-level geolocators. Again, we simulated data using the 13th scenario from example 5 above ("CentroidSampleBreeding4"; make sure you run examples 5 and 8 code before running this section).

```
# Simulation using error associated with light-level geolocators
# Assign geolocator bias / variance co-variance matrix

geoBias <- c(-10000, 50000)
geoVCov <- matrix(c(1.2e+8, 0, 0, 1.2e+9), 2, 2)
sqrt(diag(geoVCov))

targLocs14.2 <- lapply(targLocs14,
                       FUN = function(x){sf::st_transform(x, 6933)})

# convert wintering locations to polygons using the helper function
WinteringPolys <- lapply(winteringPos14,
                         toPoly,
                         projection.in = 4326,
                         projection.out = 6933)

winteringPolys <- lapply(winteringPos14,
                         toPoly,
                         projection.in = 4326,
                         projection.out = 4326)

winteringPolys2 <- lapply(winteringPos14,
                          toPoly,
                          projection.in = 4326,
                          projection.out = 'ESRI:54027')

# Double check that the input center points and output polygons are ordered
# correctly
#cbind(breedingPos14[[1]][,1:2],
#      t(sapply(slot(breedingPolys[[1]], "polygons"),
#               function(x){ slot(x,"labpt")})))

#cbind(winteringPos14[[1]][,1:2],
#      t(sapply(slot(winteringPolys[[1]], "polygons"),
#               function(x){ slot(x,"labpt")})))

# Simulate capture and non-breeding locations of 100 individuals #

nSample14.2 <- 100 # Total number sampled per simulation

# How many sampled from each natural population (sampling scenarios separate
# from definition scenarios)
sampleBreeding14.2 <- list(round(breedingRelN14[[1]]*nSample14.2),
                         c(rep(0,22),round(breedingRelN14[[5]][1]*nSample14.2),
                           rep(0, 4),round(breedingRelN14[[5]][2]*nSample14.2),
```

```
                          rep(0,44),round(breedingRelN14[[5]][3]*nSample14.2),
                          rep(0, 4),round(breedingRelN14[[5]][4]*nSample14.2),
                          rep(0, 22)),
                        rep(c(rep(0, 4),
                              rep(round(breedingRelN14[[2]][1]*nSample14.2/2),
                                  2),
                              rep(0, 4)), 10))

lapply(sampleBreeding14.2, matrix, nrow=10, ncol=10)

# Number of simulations to run
nSims14.2 <- 10
nSimsLarge14.2 <- 25
nYears <- 1
nMonths <- 1

# Set up data structures for storing results

#making an empty list to fill
animalLoc14 <- vector("list", nScenarios14)
sim14.2 <- vector("list", nSimsLarge14.2)

compare14.2.array <- array(NA, c(nSimsLarge14.2, 2),
                           dimnames = list(1:nSimsLarge14.2,
                                           c("MC", "Mantel")))

results14 <- vector("list", nScenarios14)
results14.2 <- vector("list", nSimsLarge14.2)
originDist1 <- targetDist1 <- matrix(NA, nSample14.2, nSample14.2)
targetDist1[lower.tri(targetDist1)] <- 1
distIndices <- which(!is.na(targetDist1), arr.ind = T)

# Run simulations
set.seed(7)

system.time(for (sim in 1:nSimsLarge14.2) {
  cat("Simulation", sim, "of", nSimsLarge14.2, '\n')
  sim14.2[[sim]] <- lapply(sampleBreeding14.2,
                           simMove,
                           breedingDist = breedDist14[[1]],
                           winteringDist=nonbreedDist14[[1]],
                           psi=psi,
                           nYears=nYears,
                           nMonths=nMonths)

for (i in 13) { # only run one scenario
  animalLoc14.0 <- sim14.2[[sim]][[scenarioToSampleMap14[i]]]$animalLoc
  animalLoc14[[i]] <- changeLocations(animalLoc14.0,
                                      breedingSiteTrans14[[i]],
                                      winteringSiteTrans14[[i]])

    breedingTruePoints14 <- capLocs14[[i]][animalLoc14[[i]][,1,1,1], ]
```

```
    winteringTruePoints14 <- targLocs14.2[[i]][animalLoc14.0[,2,1,1], ]

    results14.2[[sim]] <- simLocationError(winteringTruePoints14,
                                           WinteringPolys[[i]],
                                           geoBias,
                                           geoVCov,
                                           projection = sf::st_crs(winteringTruePoints14))

    animalLoc14.2 <- animalLoc14[[i]]

    animalLoc14.2[ , 2, 1, 1] <- results14.2[[sim]]$targetSample

    results14[[i]] <- calcPsiMC(breedDist14[[i]],
                                nonbreedDist14[[i]],
                                animalLoc14.2,
                                originRelAbund = breedingRelN14[[i]],
                                verbose = F)

    compare14.2.array[sim, 'MC'] <- results14[[i]]$MC

    originDist0 <- geosphere::distVincentyEllipsoid(
                                    as(breedingTruePoints14[distIndices[,'row'],],"Spatial"),
                                    as(breedingTruePoints14[distIndices[,'col'],],"Spatial"))

    originDist1[lower.tri(originDist1)] <- originDist0

    target.point.sample <- sf::st_as_sf(
      data.frame(results14.2[[sim]]$targetPointSample),
      coords = c(1,2),
      crs = 6933)

    target.point.sample2 <- sf::st_transform(target.point.sample, 4326)

    targetDist0 <-geosphere::distGeo(as(target.point.sample2[distIndices[,'row'],],"Spatial"),
                                    as(target.point.sample2[distIndices[,'col'],],"Spatial"))

    targetDist1[lower.tri(targetDist1)] <- targetDist0

    compare14.2.array[sim, 'Mantel'] <- calcMantel(originDist = originDist1,
                                                   targetDist = targetDist1)$pointCorr
  }
})

means14.2 <- apply(compare14.2.array, 2, mean)
vars14.2 <- apply(compare14.2.array, 2, var)
rmse14.2 <- apply(compare14.2.array, 2, function(x) sqrt(mean((x - MC14)^2)))

# Set up data structures for storing estimation results
est14.2.array <- array(NA, c(nSims14.2, 2),
                       dimnames = list(1:nSims14.2,
                                       c("MC", "Mantel")))

var14.2.array <- array(NA, c(nSims14.2, 2),
```

```
                      dimnames = list(1:nSims14.2,
                                      c("MC", "Mantel")))

ci14.2.array <- array(NA, c(nSims14.2, 2, 2),
                      dimnames = list(1:nSims14.2,
                                      c("MC", "Mantel"),
                                      c('lower', 'upper')))

animalLoc14 <- vector("list", nSims14.2) #making an empty list to fill
results14 <- vector("list", nSims14.2)

# Run estimations

set.seed(567)
sim14.2.sub <- sim14.2[sample.int(nSimsLarge14.2, nSims14.2, T)]

for (sim in 1:nSims14.2) {
  cat("Estimation", sim, "of", nSims14.2, '\n')
 for (i in 13) {
  points0 <- sf::st_as_sf(data.frame(results14.2[[sim]]$targetPointSample),
                          coords=c("X1","X2"),
                          crs = 6933)

  points1 <- sf::st_transform(points0, crs = "ESRI:54027")

  animalLoc14[[sim]] <- changeLocations(
    sim14.2.sub[[sim]][[scenarioToSampleMap14[i]]]$animalLoc,
                                        breedingSiteTrans14[[i]],
                                        winteringSiteTrans14[[i]])

  results14[[sim]] <- estMC(originDist = breedDist14[[i]],
                            targetDist = nonbreedDist14[[i]],
                            originRelAbund = breedingRelN14[[i]],
                originPoints = capLocs14[[i]][animalLoc14[[sim]][, 1, 1, 1], ],
                            targetPoints = points1,
                            originAssignment = animalLoc14[[sim]][, 1, 1, 1],
                            targetAssignment = results14.2[[sim]]$targetSample,
                            nSamples = 1000,
                            verbose = 0,
                            calcCorr = T,
                            geoBias = geoBias,
                            geoVCov = geoVCov,
                            isGL = T,
                            targetSites = winteringPolys2[[i]],
                            maintainLegacyOutput = TRUE)

    est14.2.array[sim, 'MC'] <- results14[[sim]]$meanMC

    est14.2.array[sim, 'Mantel'] <- results14[[sim]]$meanCorr

    var14.2.array[sim, 'MC'] <- results14[[sim]]$seMC ^ 2

    var14.2.array[sim, 'Mantel'] <- results14[[sim]]$seCorr ^ 2
```

```
    ci14.2.array[sim, 'MC', ] <- results14[[sim]]$bcCI

    ci14.2.array[sim, 'Mantel', ] <- results14[[sim]]$bcCICorr

  }
}

pointEsts14.2 <- t(sapply(results14, function(x) c(x$pointMC, x$pointCorr)))

# Summarize estimation results
summary(var14.2.array)
vars14.2
summary(est14.2.array)
summary(pointEsts14.2)
colMeans(pointEsts14.2) - MC14
sqrt(colMeans((pointEsts14.2 - MC14)^2))
means14.2
mean(ci14.2.array[, "MC", 'lower'] <= MC14 &
       ci14.2.array[, "MC", 'upper'] >= MC14)
mean(ci14.2.array[, "Mantel", 'lower'] <= MC14 &
       ci14.2.array[, "Mantel", 'upper'] >= MC14)
sqrt(vars14.2) / MC14
summary(sqrt(var14.2.array)/est14.2.array)

# Plot histograms of bootstrap estimation results
est14.2.df <- data.frame(Parameter = rep(c("MC", 'rM'), 2, each = nSims14.2),
                     Quantity = rep(c('Mean', 'Variance'),
                                    each = 2 * nSims14.2),
                     sim = rep(1:nSims14.2, 4),
                     Estimate = c(est14.2.array, var14.2.array))

trues14.2.df <- data.frame(Parameter = rep(c("MC", 'rM'), 2),
                       Quantity = rep(c('Mean', 'Variance'), each = 2),
                       Value = c(MC14, MC14, vars14.2))

g.est <- ggplot(est14.2.df, aes(Estimate)) + geom_histogram(bins = 15) +
  facet_grid(Parameter ~ Quantity, scales = 'free_x') +
  geom_vline(aes(xintercept = Value), data = trues14.2.df) +
  theme_bw() + scale_x_continuous(breaks = c(0.002, 0.003, 0.1, 0.2, 0.3))

g.est


# Summary table of bootstrap estimation results
qualities14.2 <- data.frame(Parameter = rep(c("MC", 'rM'), 5),
                            Quantity = rep(rep(c('Mean', 'Variance'),
                                               c(2, 2)), 3, 10),
                            Measure = rep(c('Bias', 'RMSE', "Coverage"),
                                          c(4, 4, 2)),
                            Value = c(colMeans(est14.2.array - MC14),
                                      colMeans(var14.2.array) - vars14.2,
                                      sqrt(colMeans((est14.2.array - MC14)^2)),
                                      sqrt(mean((var14.2.array[,1] -
```

```
                                                  vars14[1])^2)),
                                        sqrt(mean((var14.2.array[,2] -
                                                  vars14[2])^2)),
                                        mean(ci14.2.array[,"MC",'lower']<=MC14 &
                                             ci14.2.array[,"MC",'upper']>=MC14),
                                  mean(ci14.2.array[, "Mantel", 'lower'] <= MC14 &
                                       ci14.2.array[, "Mantel", 'upper'] >= MC14)))

format(qualities14.2, digits = 2, scientific = F)



# Make summary table and figures of results with (Example 9) and without
# (Example 8) location error

qualities.all <- rbind(data.frame(DataType = 'GPS', qualities14),
                       data.frame(DataType = 'GL', qualities14.2))


est14.all.df <- rbind(data.frame(DataType = 'GPS', est.df),
                       data.frame(DataType = 'GL', est14.2.df))

trues14.all.df <- data.frame(DataType = rep(c("GPS", "GL"), 2, each = 2),
                             Parameter = rep(c("MC", 'rM'), 4),
                       Quantity = rep(c('Mean', 'Variance'), each = 4),
                       Value = c(rep(MC14, 4), vars14, vars14.2))

g.est <- ggplot(est14.all.df, aes(Estimate)) + geom_histogram(bins = 12) +
  facet_grid(DataType + Parameter ~ Quantity, scales = 'free_x') +
  geom_vline(aes(xintercept = Value), data = trues14.all.df) +
  theme_bw() + scale_x_continuous(breaks = c(0.002, 0.003, 0.1, 0.2, 0.3))

g.est
```

###EXAMPLE 10 `estMC` Estimate strength of migratory connectivity incorporating uncertainty in estimates of breeding abundance (see estMC help file)

###EXAMPLE 11 `estMC` Estimate strength of migratory connectivity incorporating detection heterogeneity (see estMC help file)

###EXAMPLE 12 `estMC` Estimate strength of migratory connectivity incorporating location 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. 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. See Example 2 above and estMC help file.

###EXAMPLE 13 (from process error simulation in Cohen et al. 2018)

`simMove` for measuring the influence of dispersal rates on MC

Long-distance dispersal occurs when individuals that originate or breed in one population do not return to the same population to breed the next year. To quantify the sensitivity of MC to dispersal, we used simulations in which dispersal probability between breeding regions varied from low to high to measure the sensitivity of MC to dispersal.

```r
## Simulation ----
nBreeding <- 100
nWintering <- 100
breedingPos <- matrix(c(rep(seq(-99, -81, 2), each=sqrt(nBreeding)),
                        rep(seq(49, 31, -2), sqrt(nBreeding))), nBreeding, 2)
winteringPos <- matrix(c(rep(seq(-79, -61, 2), each=sqrt(nWintering)),
                         rep(seq(9, -9, -2), sqrt(nWintering))), nWintering, 2)
head(breedingPos)
tail(breedingPos)
head(winteringPos)
tail(winteringPos)

breedDist <- distFromPos(breedingPos, 'ellipsoid')
nonbreedDist <- distFromPos(winteringPos, 'ellipsoid')

# Breeding Abundance
breedingN <- rep(5000, nBreeding)
breedingRelN <- breedingN/sum(breedingN)

# Set up psi matrix
o <- optimize(mlogitMC, MC.in = 0.25, origin.dist = breedDist,
              target.dist = nonbreedDist, origin.abund = breedingRelN,
              sample.size = sum(breedingN),
              interval = c(0, 10), tol = .Machine$double.eps^0.5)

slope <- o$minimum
psi <- mlogitMat(slope, breedDist)

# Baseline strength of migratory connectivity
MC <- calcMC(breedDist, nonbreedDist, breedingRelN, psi, sum(breedingN))
MC

# Other basic simulation parameters

## Dispersal simulations---
set.seed(1516)

nYears <- 15

nMonths <- 4 # Each season

Drates <- c(0.02, 0.04, 0.08, 0.16, 0.32, 0.64)    #rates of dispersal

birdLocDisp <- vector('list', length(Drates))

Disp.df  <- data.frame(Year=rep(1:nYears, length(Drates)),
                       Rate=rep(Drates, each = nYears), MC = NA)

for(i in 1:length(Drates)){
  cat('Dispersal Rate', Drates[i], '\n')
  birdLocDisp[[i]] <- simMove(breedingN,
                              breedDist,
                              nonbreedDist,
```

```
                              psi,
                              nYears,
                              nMonths,
                              sumDispRate = Drates[i])
  for(j in 1:nYears){
    cat('\tYear', j, '\n')
    temp.results <- calcPsiMC(breedDist,
                              nonbreedDist,
                              breedingRelN,
                              birdLocDisp[[i]]$animalLoc,
                              years=j,
                              verbose = F)

    Disp.df$MC[j + (i - 1) * nYears] <- temp.results$MC
  }
} # end i loop

Disp.df$Year <- Disp.df$Year - 1 #just run once!

data.frame(Disp.df, roundMC = round(Disp.df$MC, 2),
           nearZero = Disp.df$MC < 0.01)


# Convert dispersal rates to probabilities of dispersing at least certain distance
threshold <- 1000
probFarDisp <- matrix(NA, nBreeding, length(Drates), dimnames = list(NULL, Drates))

for (i in 1:length(Drates)) {
  for (k in 1:nBreeding) {
    probFarDisp[k, i] <-
      sum(birdLocDisp[[i]]$natalDispMat[k, which(breedDist[k, ]>= threshold)])
  }
}

summary(probFarDisp)

#plot results
\dontrun{
require(ggplot2)
require(ggthemes)

line_set=c(0.5, 0.75, 1, 1.25, 1.75, 2.25)
ggplot(Disp.df, aes(x=Year, y=MC, size=as.factor(Rate)))+
 geom_line()+
 scale_size_manual(values=line_set)+
 labs(size="Dispersal Rate")+
 scale_y_continuous("MC", limits=c(-.005, 0.26),
                    breaks=c(0, 0.05, 0.1, 0.15, 0.2, 0.25))+
 scale_x_continuous(breaks=c(0, 3, 6, 9, 12, 15))+
 theme_bw()+
 theme(axis.title=element_text(size=16, face ="bold"),
       axis.text=element_text(size=14),
       panel.grid.major = element_line(color="grey90"),
```

```
        panel.grid.major.x = element_blank(),
        legend.title=element_text(size=14),
        legend.text=element_text(size=14))
}
```