'MigConnectivity' package

Jeff Hostetler, Mike Hallworth, Clark Rushing, Emily Cohen

2017-07-19

Introduction

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:

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).

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 determines the correlation between those distances. Earlier methods used the correlation between two distance matrices of individual birds 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, 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. In revision. A quantitative definition for the strength of migratory connectivity.

Overview of Functions

1. `calcMC` calculates the strength of migratory connectivity (MC) with the probabilities of movement between regions, distances between regions within the two seasonal ranges, and relative abundance among regions within the seasonal range from which the transition probabilities originate.

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

```r
install.packages('devtools')

devtools::install_github("SMBC-NZP/MigConnectivity")

library(MigConnectivity)
```

Functions

`calcMC` Calculate strength of migratory connectivity (MC)
`estMC` Estimate the strength of migratory connectivity (MC) while incorporating location uncertainty

Examples

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 or the absolute abundance within each of the regions where individuals originate from (here the breeding regions).

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

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

Generate a distance matrix between the different regions

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

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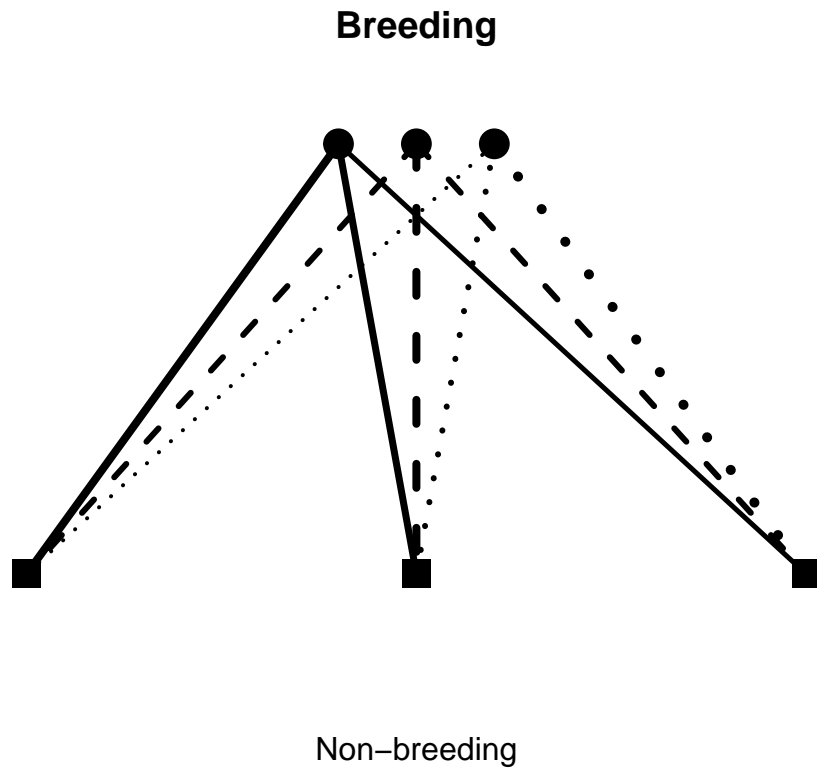
Breeding

● ● ●

■   ■   ■

Non−breeding

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)
```

Breeding

Non–breeding

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
#equal absolute abundance among the three breeding regions, each must be >= 1
N <- rep(3000, nBreeding)
```

Calculate the strength of migratory connectivity using the simulated data above

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

round(MC,3)
#> [1] 0.05
MC <- calcMC(originDist = breedDist,
             targetDist = nonBreedDist,
             psi = psi,
             originAbund = N)

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

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.

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)

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

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

```
data(OVENdata) # Ovenbird

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

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

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

library(rgeos)
library(shape)
library(raster)
library(maptools)

data(wrld_simpl)

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

par(mar=c(0,0,0,0))
```
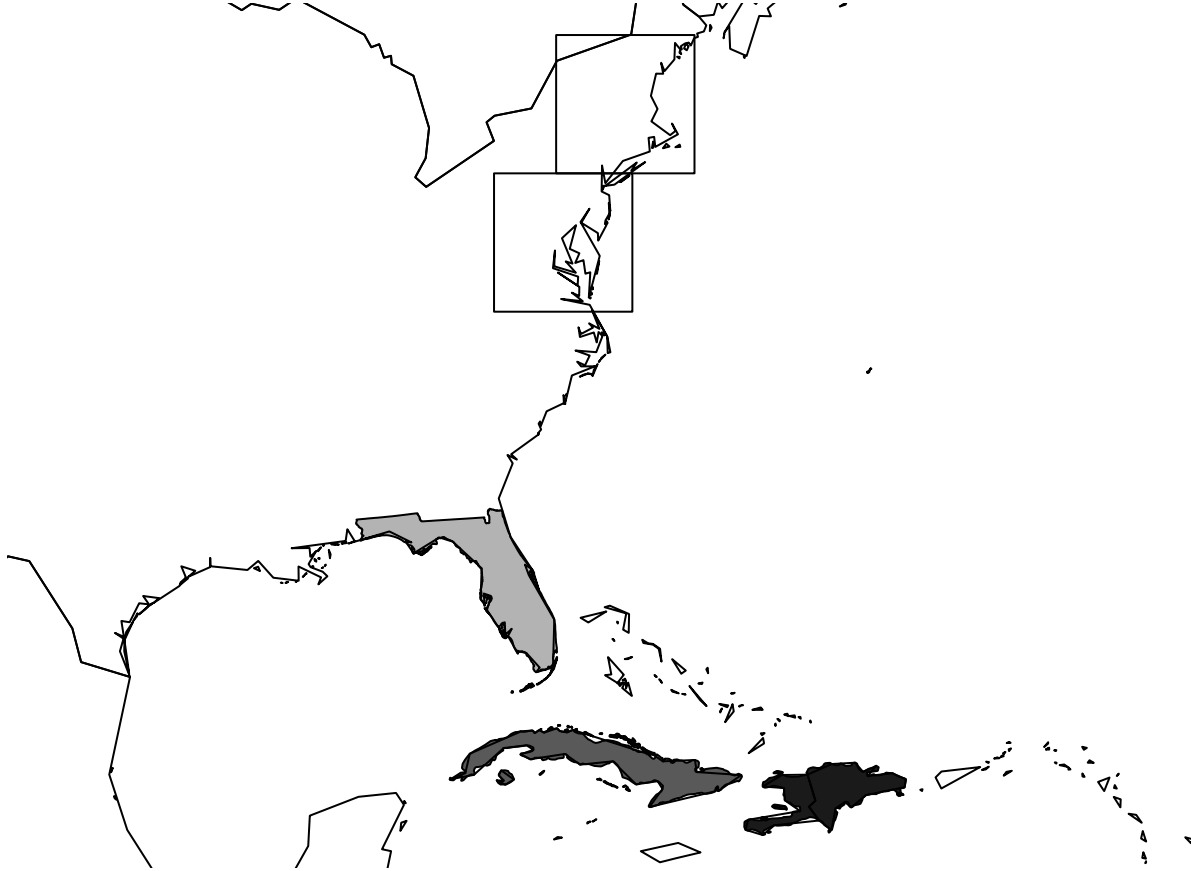
```
plot(OVENdata$originSites,xlim=c(raster::extent(OVENdata$targetSites)[1],raster::extent(OVENdata$target!
                          ylim=c(raster::extent(OVENdata$targetSites)[3],raster::extent(OVENdata$origin!
plot(OVENdata$targetSites, add = TRUE, col=c("gray70","gray35","gray10"))
wrld_simple<-sp::spTransform(wrld_simpl,raster::crs(OVENdata$targetSites))
plot(wrld_simple,add=TRUE)
```


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

```
M<-estMC(isGL=OVENdata$isGL, # Logical vector indicating light-level geolocator (T) GPS (F)
      geoBias = OVENdata$geo.bias, # Light-level geolocator location bias
      geoVCov = OVENdata$geo.vcov, # Light-level geolocator co-variance matrix
      targetDist = OVENdata$targetDist, # Non-breeding target distribution distance matrix
      originDist = OVENdata$originDist, # Breeding / origin distribution distance matrix
      targetSites = OVENdata$targetSites, # Non-breeding target sites
      originSites = OVENdata$originSites, # Breeding origin sites
      originPoints = OVENdata$originPoints, # Capture Locations
      targetPoints = OVENdata$targetPoints, # Non-breeding Locations derived from devices
      originRelAbund = OVENdata$originRelAbund, # Relative abundance within OriginSites
      verbose = 0,   # output options - see help ??estMC
      nSim = 10,     # This is set low for example
      nSamples = 10) # This is set low for example

par(mar=c(0,0,0,0))
plot(wrld_simple,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$originSites[1],add=TRUE,lwd=1.75)
plot(OVENdata$originSites[2],add=TRUE,lwd=1.75)
plot(OVENdata$targetSites,add=TRUE,lwd=1.5,col=c("gray70","gray35","gray10"))

legend(400000,-2105956,legend=paste("MC =",round(M$meanMC,2), "\u00b1", round(M$seMC,2)),bty="n",cex=1.8

shape::Arrows(gCentroid(OVENdata$originSites[1])@coords[,1],
              gCentroid(OVENdata$originSites[1])@coords[,2],
              gCentroid(OVENdata$targetSites[2])@coords[,1]+80000,
              extent(OVENdata$targetSites[2])[4]+150000,
              arr.length = 0.3,
              arr.adj = 0.5,
              arr.lwd = 1,
              arr.width = 0.4,
              arr.type = "triangle",
              lwd=(apply(M$samplePsi[,1,],2,mean)[2]*10),
              lty=1)

shape::Arrows(gCentroid(OVENdata$originSites[1])@coords[,1],
              gCentroid(OVENdata$originSites[1])@coords[,2],
              gCentroid(OVENdata$targetSites[3])@coords[,1],
              extent(OVENdata$targetSites[3])[4]+150000,
              arr.length = 0.3,
              arr.adj = 0.5,
              arr.lwd = 1,
              arr.width = 0.4,
              arr.type = "triangle",
              lwd=(apply(M$samplePsi[,1,],2,mean)[3]*10),
              lty=1)

shape::Arrows(gCentroid(OVENdata$originSites[2])@coords[,1],
              gCentroid(OVENdata$originSites[2])@coords[,2],
              gCentroid(OVENdata$targetSites[1])@coords[,1],
              extent(OVENdata$targetSites[1])[4]+150000,
              arr.length = 0.3,
              arr.adj = 0.5,
              arr.lwd = 1,
              arr.width = 0.4,
              arr.type = "triangle",
              lwd=(apply(M$samplePsi[,2,],2,mean)[1]*10),
              lty=1)

shape::Arrows(gCentroid(OVENdata$originSites[2])@coords[,1],
              gCentroid(OVENdata$originSites[2])@coords[,2],
              (gCentroid(OVENdata$targetSites[2])@coords[,1]-80000),
              extent(OVENdata$targetSites[2])[4]+150000,
              arr.length = 0.3,
              arr.adj = 0.5,
              arr.lwd = 1,
              arr.width = 0.4,
              arr.type = "triangle",
```
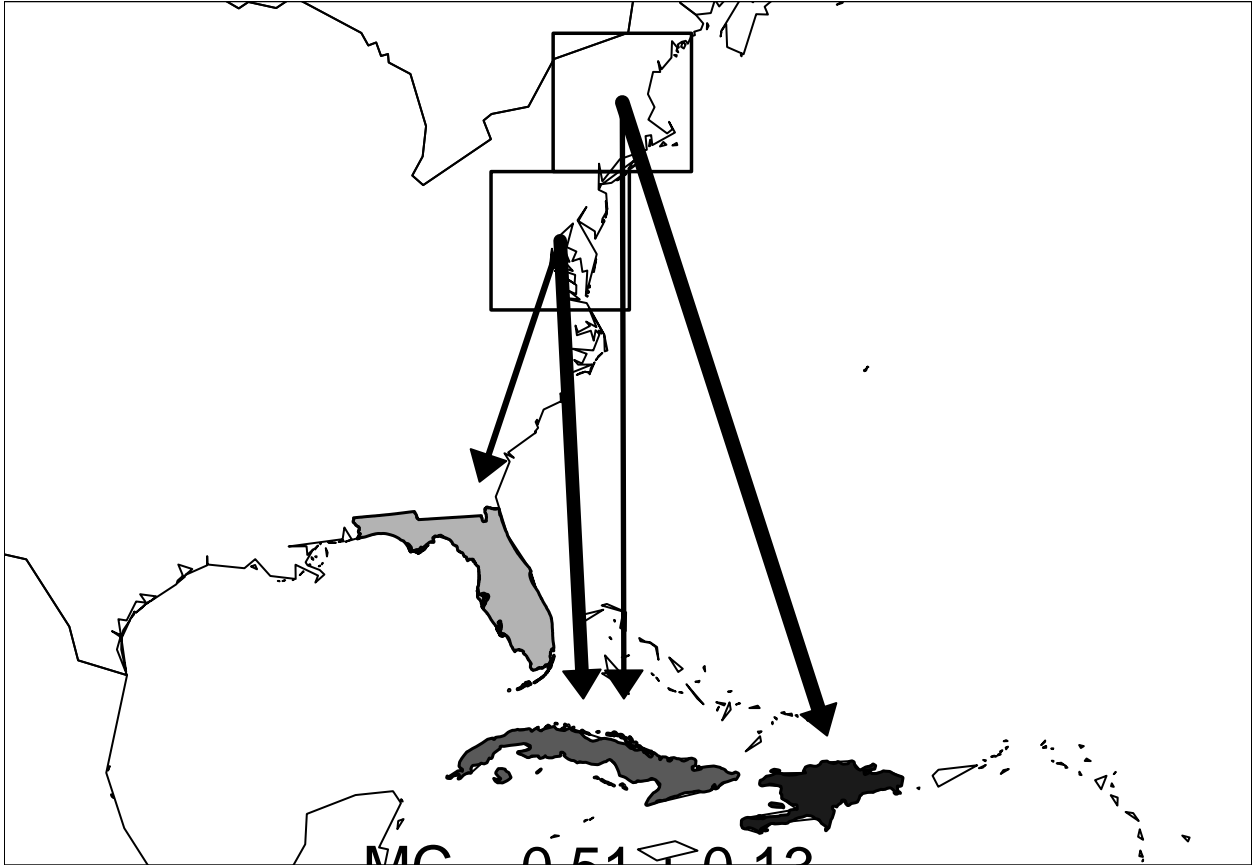
```
            lwd=(apply(M$samplePsi[,2,],2,mean)[2]*10))

box(which="plot")
```



```
str(M)
#> List of 19
#>  $ sampleMC   : num [1:10] 0.567 0.591 0.542 0.608 0.537 ...
#>  $ samplePsi  : num [1:10, 1:2, 1:3] 0 0 0 0 0 0 0 0 0 0 ...
#>   ..- attr(*, "dimnames")=List of 3
#>   .. ..$ : chr [1:10] "1" "2" "3" "4" ...
#>   .. ..$ : NULL
#>   .. ..$ : NULL
#>  $ pointPsi   : num [1:2, 1:3] 0 0.4 0.174 0.6 0.826 ...
#>   ..- attr(*, "dimnames")=List of 2
#>   .. ..$ : NULL
#>   .. ..$ : NULL
#>  $ pointMC    : num 0.628
#>  $ meanMC     : num 0.514
#>  $ medianMC   : num 0.539
#>  $ seMC       : num 0.126
#>  $ simpleCI   : Named num [1:2] 0.346 0.749
#>   ..- attr(*, "names")= chr [1:2] "2.5%" "97.5%"
#>  $ bcCI       : Named num [1:2] 0.346 0.736
#>   ..- attr(*, "names")= chr [1:2] "0.6819024%" "92.69256%"
#>  $ hpdCI      : num [1, 1:2] 0.346 0.749
#>   ..- attr(*, "dimnames")=List of 2
```

```
#>   .. ..$ : chr "var1"
#>   .. ..$ : chr [1:2] "lower" "upper"
#>   ..- attr(*, "Probability")= num 0.9
#> $ simpleP     : NULL
#> $ bcP         : NULL
#> $ sampleCorr  : logi [1:10] NA NA NA NA NA NA ...
#> $ pointCorr   : NULL
#> $ meanCorr    : NULL
#> $ medianCorr  : NULL
#> $ seCorr      : NULL
#> $ simpleCICorr: NULL
#> $ bcCICorr    : NULL
```

Function

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

Advanced example -

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. These simulations use utility functions included in the package. See below for details regarding the utility functions to help simulate data.

Simulate data to demonstrate the use of `calcMC`

Utility Functions for use in simulations

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

```
mlogitMat <- function(slope, dist) {
      preMat <- exp(-slope/mean(dist)*dist)
      diag(preMat) <- 0
      nr <- nrow(dist)
      nc <- ncol(dist)
      outMat <- matrix(0, nr, nc)
  for (b in 1:nr) {
     outMat[b,] <- preMat[b,]/(1+sum(preMat[b, ]))
     outMat[b,b] <- 1 - sum(outMat[b, ])
  }
  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) {
   nBreeding <- nrow(origin.dist)
   nWintering <- nrow(target.dist)
   psi <- mlogitMat(slope, origin.dist)
  if (any(psi<0))
    return(5*slope^2)
  if (sum(origin.abund) > 1)
```

```r
    MC <- calcMC(origin.dist, target.dist, psi, originAbund = origin.abund)
  else
    MC <- calcMC(origin.dist, target.dist, psi, originRelAbund = origin.abund)
  return((MC.in - MC)^2)
}
```

calcStrengthInd: rho function for individuals

```r
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] <- originDist2[j,i] <- originDist[locations[i,1,1,1], locations[j,1,1,1]]
      targetDist2[i,j] <- 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

```r
calcPsiMC <- function(originDist, targetDist, locations, originRelAbund = NULL,
                      originAbund = NULL, 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,,]
    targetMat <- locations[i,2,,]
    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 <- NULL #calcMC(originDist, targetDist, psiMat, originRelAbund, originAbund)
  return(list(psi=psiMat, MC=MC))
}
```

```r
set.seed(75)

# Parameters for simulations

nSeasons <- 2 # population with two discrete seasons - breeding / non-breeding
nYears <- 10 # Ten years of data
```

```r
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)


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

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,
              interval = c(0, 10), 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)

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

simMove Simulation of movement between breeding and non-breeding

Recycle data generated within the calcMC function - see above

```r
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

Calculate migratory connectivity for a range of input values

Cohen et al. (*in revision*) 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.

Below are eight transition probability scenarios. #Eight transition probability scenarios

Twelve spatial arrangements that result in different distances between regions

Distance scenarios

A) Base distances, linear/ linear 1

B) Distance between breeding sites 2 and 3 doubled

C) Distance between breeding sites 2 and 3 halved

D) Distance between breeding sites 3 and 4 doubled

E) Distance between breeding sites 3 and 4 halved

F) Breeding sites on square grid/ winter linear 6

G) Distance between wintering sites 2 and 3 doubled

H) Distance between wintering sites 2 and 3 halved

I) Distance between wintering sites 3 and 4 doubled

J) Distance between wintering sites 3 and 4 halved

K) Breeding linear, Wintering sites on square grid

L) Wintering and breeding on square grid 12

Five relative breeding abundance scenarios

Changes to relative breeding abundance:

1. Base

2. Abundance at site B doubled

3. Abundance at site B halved

4. Abundance at site D doubled

5. Abundance at site D halved

For all eight transition probability matrices and three distance scenarios

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

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

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

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

Another way of comparing results

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

t(round(MC.diff2, 4))
#>              Full Mix Avoid One Site Full Connectivity Half Mix    Low
#> Linear              0              0            0.0000   0.0000 0.0000
#> B Dist BC*2         0              0           -0.0261   0.0956 -0.0142
```

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

Add the relative abundance breeding scenarios

Calculate MC 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
```

15

```
#>        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 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 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 4-9 (from sampling regime simulations in Cohen et al. in revision)

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. uncertainty in estimates of abundance

5. heterogeneity in detection among regions from mark re-encounter data

6. uncertainty in assignments to regions from tracking 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 that does not incorporate relative abundance but has been used as a measure of migratory connectivity.

EXAMPLE 4

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)))

# 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,
```

```r
                         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
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))


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))

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

# for the baseline use the simulation from above, sims
animalLoc14base <- sims$animalLoc

#transferring the simulated bird locations from the true populations to the researcher defined populati
changeLocations <- function(animalLoc, breedingSiteTrans, winteringSiteTrans) {
  animalLoc[,1,,] <- breedingSiteTrans[animalLoc[,1,,]]
  animalLoc[,2,,] <- winteringSiteTrans[animalLoc[,2,,]]
  return(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(MC, rep(NA, nScenarios14)),
                        Mantel = c(MC, 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 5

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))
```

```r
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]], 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,
               originAbund = 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)),
                        MCA = c(MC15, rep(NA, nScenarios15)),
                        Mantel = c(MC15, rep(NA, nScenarios15)))

compare15.array <- array(NA, c(nSims15, nScenarios15, 3), dimnames = list(1:nSims15,
                                         c("Base", "Breeding4", "CentroidSampleBreeding4",
                                           "BiasedSample", "BiasedSampleBreeding4",
```

```
                                          "BiasedCentroidSampleBreeding4"),
                                c("MC", "MCA", "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(sim15[[scenarioToSampleMap15[i]]]$animalLoc,
                                        breedingSiteTrans15[[i]], 1:nWintering15[i])
    results15[[i]] <- calcPsiMC(breedDist15[[i]], nonbreedDist15[[i]],
                                animalLoc15[[i]], breedingRelN15[[i]], NULL, F)
    compare15.array[sim, i, 'MC'] <- results15[[i]]$MC
    compare15.array[sim, i, 'MCA'] <- calcMC(breedDist15[[i]], nonbreedDist15[[i]],
                                             results15[[i]]$psi,
                                             originAbund = breedingN15[[i]])
    compare15.array[sim, i, 'Mantel'] <- calcStrengthInd(breedDist15[[1]],
                                                         nonbreedDist15[[1]],
                                                         sim15[[scenarioToSampleMap15[i]]]$animalLoc,
                                                         resamp=0)$correlation
  }
}

compare15$MC[1:nScenarios15 + 1] <- apply(compare15.array[,,'MC'], 2, mean)
compare15$MCA[1:nScenarios15 + 1] <- apply(compare15.array[,,'MCA'], 2, mean)
compare15$Mantel[1:nScenarios15 + 1] <- apply(compare15.array[,,'Mantel'], 2,mean)

compare15 <- transform(compare15, MC.diff=MC - MC[1], MCA.diff=MCA - MCA[1],
                       Mantel.diff=Mantel - Mantel[1],
                       MC.prop=MC/MC[1], MCA.prop=MCA/MCA[1],
                       Mantel.prop=Mantel/Mantel[1])


compare15a <- as.matrix(compare15[1 + 1:nScenarios15, c(2,5,4,7)])

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

EXAMPLE 6

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

```r
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]],
                   interval=c(0,10), tol=.Machine$double.eps^0.5)

  slope16a <- o16a$minimum

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

```r
  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,
               originAbund = breedingN16[[1]])


# Set up sampling regimes (different number than number of scenarios)
nSample16 <- 150
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)),
                        MCA = c(MC16, rep(NA, nScenarios16)),
                        MCss1 = c(MC16, rep(NA, nScenarios16)),
                        MCss2 = c(MC16, rep(NA, nScenarios16)),
                        Mantel = c(MC16, rep(NA, nScenarios16)))

compare16.array <- array(NA, c(nSims16, nScenarios16, 5),
                         dimnames = list(1:nSims16,
                                         c("Base", "Breeding4",
                                           "CentroidSampleBreeding4",
```

```r
                                  "BiasedSample",
                                  "BiasedSampleBreeding4",
                                  "BiasedCentroidSampleBreeding4"),
                              c("MC", "MCA", "MCss1", "MCss2", "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]]]$animalLoc,
                                  breedingSiteTrans16[[i]], 1:nWintering16[i])
    results16[[i]] <- calcPsiMC(breedDist16[[i]], nonbreedDist16[[i]],
                            animalLoc16[[i]], breedingRelN16[[i]])
    compare16.array[sim, i, 'MC'] <- results16[[i]]$MC
    compare16.array[sim, i, 'MCA'] <- calcMC(breedDist16[[i]], nonbreedDist16[[i]],
                                        results16[[i]]$psi,
                                        originAbund = breedingN16[[i]])
    compare16.array[sim, i, 'MCss1'] <- calcMC(breedDist16[[i]], nonbreedDist16[[i]],
                                        results16[[i]]$psi,
                                        originAbund = table(animalLoc16[[i]][ , 1, 1, 1]))
    compare16.array[sim, i, 'MCss2'] <- calcMC(breedDist16[[i]], nonbreedDist16[[i]],
                                        results16[[i]]$psi,
                                        originAbund = breedingRelN16[[i]] * nSample16) # pmax(bree
    compare16.array[sim, i, 'Mantel'] <- calcStrengthInd(breedDist16[[1]],
                                                nonbreedDist16[[1]],
                                                sim16[[scenarioToSampleMap16[i]]]$animalLoc,
                                                resamp=0)$correlation
  }
}

compare16$MC[1:nScenarios16 + 1] <- apply(compare16.array[,,'MC'], 2, mean)
compare16$MCA[1:nScenarios16 + 1] <- apply(compare16.array[,,'MCA'], 2, mean)
compare16$MCss1[1:nScenarios16 + 1] <- apply(compare16.array[,,'MCss1'], 2, mean)
compare16$MCss2[1:nScenarios16 + 1] <- apply(compare16.array[,,'MCss2'], 2, mean)
compare16$Mantel[1:nScenarios16 + 1] <- apply(compare16.array[,,'Mantel'], 2, mean)
compare16 <- transform(compare16, MC.diff=MC - MC[1], MCA.diff=MCA - MCA[1],
                    MCss1.diff = MCss1 - MCss1[1],
                    MCss2.diff = MCss2 - MCss2[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 7

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

EXAMPLE 8

estMC Estimate strength of migratory connectivity incorporating detection heterogeneity (see estMC help file)

EXAMPLE 9

estMC Estimate strength of migratory connectivity incorporating location uncertainty (***Examples*** see above) 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.

EXAMPLE 10 (from process error simulation in Cohen et al. in revision)

estMC 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
# Calculates probability matrix based on exponential decline with distance
mlogitMat <- function(slope, dist) {
  preMat <- exp(-slope/mean(dist)*dist)
  diag(preMat) <- 0
  nr <- nrow(dist)
  nc <- ncol(dist)
  outMat <- matrix(0, nr, nc)
  for (b in 1:nr) {
    outMat[b,] <- preMat[b,]/(1+sum(preMat[b, ]))
    outMat[b,b] <- 1 - sum(outMat[b, ])
  }
  return(outMat)
}

# Crude optimizable function for developing migratory connectivity pattern based on MC
mlogitMC <- function(slope, MC.in, origin.dist, target.dist, origin.rel.abund) {
  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, psi, origin.rel.abund)
  return((MC.in - MC)^2)
}

# Simple approach to estimate psi matrix and MC from simulated (or real) data
# (doesn't include uncertainty).  Only uses one year for computation
calcPsiMC3 <- function(originDist, targetDist, originRelAbund, locations, year,
                       verbose=F) {
  nOrigin <- nrow(originDist)
  nTarget <- nrow(targetDist)
```

```r
  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, year, ]
    targetMat <- locations[i, 2, year, ]
    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, psiMat, originRelAbund)
  return(list(psi=psiMat, MC=MC))
}


## 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.rel.abund = breedingRelN,
              interval = c(0, 10), tol = .Machine$double.eps^0.5)
o
slope <- o$minimum
psi <- mlogitMat(slope, breedDist)

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

```r
# 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 <- calcPsiMC3(breedDist, nonbreedDist, breedingRelN,
                               birdLocDisp[[i]]$animalLoc, year=j, F)
    Disp.df$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)

write.csv(Disp.df, "../Disp.df.csv")

# 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)
png('../dispersal.plot.png', width = 6, height = 6, res = 600, units = 'in')
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()+
```

```r
  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))
dev.off()
}
```