Package 'MigConnectivity'

May 7, 2026

Type Package

Title Estimate Migratory Connectivity for Migratory Animals

Version 0.5.0

Date 2025-08-01

Description Allows the user to estimate transition probabilities for migratory animals between any two phases of the annual cycle, using a variety of different data types. Also quantifies the strength of migratory connectivity (MC), a standardized metric to quantify the extent to which populations co-occur between two phases of the annual cycle. Includes functions to estimate MC and the more traditional metric of migratory connectivity strength (Mantel correlation) incorporating uncertainty from multiple sources of sampling error. For cross-species comparisons, methods are provided to estimate differences in migratory connectivity strength, incorporating uncertainty. See Cohen et al. (2018) <doi:10.1111/2041-210X.12916>, Cohen et al. (2019) <doi:10.1111/ecog.03974>, Roberts et al. (2023) <doi:10.1002/eap.2788>, and Hostetler et al. (2025) <doi:10.1111/2041-210X.14467> for details on some of these methods.

License GPL (>= 3)

URL https://github.com/SMBC-NZP/MigConnectivity

Depends R (>= 3.5.0)

Imports coda, geodist, gplots, graphics, grDevices, MASS, methods, ncf, R2jags, RMark (>= 2.1.14), sf, shape, stats, terra, utils, VGAM

Suggests knitr, maps, rmarkdown, testthat

VignetteBuilder knitr

Encoding UTF-8

LazyData true

LazyDataCompression xz

RoxygenNote 7.3.2

SystemRequirements JAGS (https://mcmc-jags.sourceforge.net)

NeedsCompilation no

Author Jeffrey A. Hostetler [cre, aut],
Michael T. Hallworth [aut],
Clark S. Rushing [ctb],
Emily B. Cohen [ctb],
Valentine Herrmann [ctb],
Henry Stevens [ctb]

Maintainer Jeffrey A. Hostetler <jhostetler@usgs.gov>

Repository CRAN

Date/Publication 2025-08-27 08:00:13 UTC

Contents

abundExamples . . . 3
calcMantel . . . 3
calcMC . . . 4
calcNMC . . . 7
calcTransition . . . 8
diffMantel . . . 11
diffMC . . . 12
distFromPos . . . 14
estMantel . . . 15
estMC . . . 20
estNMC . . . 29
estStrength . . . 34
estTransition . . . 39
getCMRexample . . . 52
getIsoMap . . . 53
isoAssign . . . 54
MigConnectivity . . . 57
modelCountDataJAGS . . . 58
OVENdata . . . 60
plot.estMigConnectivity . . . 61
plot.intrinsicAssign . . . 63
projections . . . 64
reverseTransition . . . 65
sampleOriginN . . . 70
sampleOriginPos . . . 71
samplePsis . . . 72
simCMRData . . . 73
simCountData . . . 74
simGLData . . . 78
simMove . . . 80
simProbData . . . 83
simTelemetryData . . . 85
weightAssign . . . 86

Index **90**

`abundExamples` *Example relative abundance estimates from simulated data*

Description

A dataset containing mcmc relative abundance estimates from simulated BBS-type data from Cohen et al. (2018). Each estimate can be used in `estStrength` function to estimate MC with uncertainty.

Usage

```
abundExamples
```

Format

A list with 10 mcmc (coda) estimates in it.

`calcMantel` *Calculate Mantel correlation (rM) from points and/or distances.*

Description

Calculation of rM from POINTS geolocators and/or GPS data, not accounting for uncertainty. If you've already calculated distances between points, you can use those instead.

Usage

```
calcMantel(
  targetPoints = NULL,
  originPoints = NULL,
  targetDist = NULL,
  originDist = NULL
)
```

Arguments

`targetPoints`	A sf `POINTS` object, with length number of animals tracked. Each point indicates the point estimate location in the non-release season.
`originPoints`	A sf `POINTS` object, with length number of animals tracked. Each point indicates the release location of an animal.
`targetDist`	Distances between the target locations of the tracked animals. Symmetric matrix with number of animals rows and columns, although really you only need the lower triangle filled in.
`originDist`	Distances between the origin locations of the tracked animals. Symmetric matrix with number of animals rows and columns, although really you only need the lower triangle filled in.

Value

`calcMantel` returns a list with elements:

`pointCorr` Simple point estimate of Mantel correlation.

`originDist, targetDist` Distances between each pair of `originPoints` and each pair of `targetPoints`, respectively, in meters. If you used distances as inputs instead, then these are just what you fed in.

References

Ambrosini, R., A. P. Moller, and N. Saino. 2009. A quantitative measure of migratory connectivity. Journal of Theoretical Biology 257:203-211. doi:10.1016/j.jtbi.2008.11.019

See Also

`estMantel`, `calcMC`, `estMC`

Examples

```
data(OVENdata) # Ovenbird
rM0 <- calcMantel(originPoints = OVENdata$originPoints, # Capture Locations
                  targetPoints = OVENdata$targetPoints) # Target locations
str(rM0)
```

`calcMC` *Migratory connectivity strength function*

Description

Migratory connectivity strength function

Usage

```
calcMC(originDist, targetDist, originRelAbund, psi, sampleSize = NULL)

calcStrength(originDist, targetDist, originRelAbund, psi, sampleSize = NULL)
```

Arguments

`originDist`	Distances between the B origin sites. Symmetric B by B matrix.
`targetDist`	Distances between the W target sites. Symmetric W by W matrix.
`originRelAbund`	Relative abundances at B origin sites. Numeric vector of length B that sums to 1.
`psi`	Transition probabilities between B origin and W target sites. Matrix with B rows and W columns where rows sum to 1.
`sampleSize`	Total sample size of animals that psi was calculated from. Should be the number of animals released in one of the origin sites and observed in one of the target sites. Optional, but recommended.

Value

scalar real value, usually between 0 and 1 (can be negative), indicating the strength of migratory connectivity.

If `sampleSize` is provided, this function uses the standard (relative abundance and small-sample size corrected) formula for MC. If not, it uses the MC(R) formula, which only corrects for relative abundance.

References

Cohen, E. B., J. A. Hostetler, M. T. Hallworth, C. S. Rushing, T. S. Sillett, and P. P. Marra. 2018. Quantifying the strength of migratory connectivity. Methods in Ecology and Evolution 9: 513 - 524. doi:10.1111/2041210X.12916

Examples

```
# Example with three breeding and three nonbreeding sites
nBreeding <- 3
nNonBreeding <- 3
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)
breedDist <- matrix(c(0, 1, 2,
                      1, 0, 1,
                      2, 1, 0), nBreeding, nBreeding)
nonBreedDist <- matrix(c(0, 5, 10,
                      5, 0, 5,
                      10, 5, 0), nNonBreeding, nNonBreeding)
relN <- rep(1/nBreeding, nBreeding)
round(calcMC(breedDist, nonBreedDist, relN, psi), 3) # == 0.05

# Example with small sample size
sampleSize <- 20 * nBreeding
round(calcMC(breedDist, nonBreedDist, relN, psi, sampleSize = sampleSize), 3) # == 0.026

################################################################################
# Example data input values
################################################################################

#########################
#Input values 1 of 3
# Eight transition probability scenarios
#########################
nScenarios1 <- length(samplePsis)
MC1 <- rep(NA, nScenarios1)
for (i in 1:nScenarios1) {
  MC1[i] <- calcMC(sampleOriginDist[[1]], sampleTargetDist[[1]],
                   sampleOriginRelN[[1]], samplePsis[[i]])
}
names(MC1) <- names(samplePsis)
round(MC1, 6)
```

```
#########################
#Input values 2 of 3
# 12 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
#########################

# Get MC strengths
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(sampleOriginDist[[j]], sampleTargetDist[[j]],
                        sampleOriginRelN[[1]], samplePsis[[i]])
  }
}

t(round(MC2, 4))

# Different way of comparing results
MC.diff2 <- apply(MC2, 2, "-", MC2[ , 1])
t(round(MC.diff2, 4))

#########################
#Input values 3 of 3
# 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
#########################

nScenarios3 <- length(sampleOriginRelN)

# Get MC strengths for breeding linear/ winter linear arrangement
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(sampleOriginDist[[j]], sampleTargetDist[[j]],
                          sampleOriginRelN[[k]], samplePsis[[i]])
    }
  }
}
t(round(MC3, 4)) # linear arrangement

# Get MC strengths 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(sampleOriginDist[[j]], sampleTargetDist[[j]],
                          sampleOriginRelN[[k]], samplePsis[[i]])
    }
  }
}
t(round(MC4, 4)) # grid arrangement

# Get MC strengths 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(sampleOriginDist[[j]], sampleTargetDist[[j]],
                          sampleOriginRelN[[k]], samplePsis[[i]])
    }
  }
}
t(round(MC5, 4)) # breeding grid, winter linear arrangement
```

`calcNMC` *Calculate NMC_XY, another type of migratory connectivity strength*

Description

Provides simple calculation of NMC_XY (network migratory connectivity strength between seasons X and Y), NMCa_XY (abundance-weighted network migratory connectivity strength), and network migratory connectivity diversity (X node-specific version of NMC_XY) from point estimate of psi (transition probabilities) and `originRelAbund` (optional). Does not include measures of uncertainty.

Usage

```
calcNMC(psi, originRelAbund = NULL)
```

Arguments

psi	Matrix of transition probabilities
originRelAbund	(optional) Vector of relative (proportional) abundance at the origin sites. If entered, should sum to 1.

Value

calcNMC returns a list with elements:

NMC scalar real value between 0 and 1, indicating the strength of network migratory connectivity

NMCpop Vector of network migratory connectivity diversity values, also between 0 and 1, the X-node-specific version of NMC_XY

NMCa If originRelAbund was entered, the results will include this, an abundance weighted measure of the strength of network migratory connectivity (also between 0 and 1). If originRelAbund was not entered, this will be left out

See Also

estNMC, calcMC, estMC

Examples

```
nScenarios <- length(samplePsis)
NMC1 <- vector("list", nScenarios)
for (i in 1:nScenarios) {
  NMC1[[i]] <- calcNMC(samplePsis[[i]])
}
names(NMC1) <- names(samplePsis)
str(NMC1)

calcNMC(samplePsis[[7]], sampleOriginRelN[[2]])
```

---

calcTransition *Calculate psi (transition probabilities between sites in two phases of the annual cycle)*

Description

Provides simple maximum-likelihood point estimate of transition probabilities that does not include measures of uncertainty. Incorporates detection heterogeneity where appropriate (band/ring return data), but not location uncertainty. Shared primarily for testing; use of estTransition is recommended instead.

Usage

```
calcTransition(
  banded = NULL,
  reencountered = NULL,
  counts = NULL,
  originAssignment = NULL,
  targetAssignment = NULL,
  originNames = NULL,
  targetNames = NULL,
  method = "SANN"
)

calcPsi(
  banded = NULL,
  reencountered = NULL,
  counts = NULL,
  originAssignment = NULL,
  targetAssignment = NULL,
  originNames = NULL,
  targetNames = NULL,
  method = "SANN"
)
```

Arguments

banded	For band return data, a vector of the number of released animals from each origin site (including those never reencountered in a target region)
reencountered	For band return data, a matrix with B rows and W columns. Number of animals reencountered on each target site by origin site they came from
counts	Migration data without target-region detection heterogeneity (i.e., anything but band return data) can be entered one of two ways: either here or with `originAssignment` and `targetAssignment`. If here, a matrix with B rows and W columns with counts of animals observed in each combination of origin and target site
originAssignment	Assignment of animals (not including band return data) to origin season sites. A vector of integers (1-B) with length number of animals tracked. Note that these data can either be entered using this argument and `targetAssignment` or using `counts`, but not both
targetAssignment	Assignment of animals (not including band return data) to target season sites. A vector of integers (1-W) with length number of animals tracked. Note that these data can either be entered using this argument and `originAssignment` or using `counts`, but not both
originNames	Optional, but recommended to keep track. Vector of names for the origin sites. If not provided, the function will either try to get these from another input or provide default names (capital letters)

targetNames	Optional, but recommended to keep track. Vector of names for the target sites. If not provided, the function will either try to get these from another input or provide default names (numbers)
method	See optim. "SANN" is slow but reasonably accurate. "Nelder-Mead" is fast but not accurate here. "BFGS" is also fast but not very stable here

Value

calcTransition returns a list with the element(s):

psi Matrix with point estimate of transition probabilities

r Vector containing point estimate of reencounter probabilities at each target site. Not included unless data includes band reencounters

See Also

estTransition, optim

Examples

```
nOriginSites <- 4
nTargetSites <- 4
originNames <- LETTERS[1:nOriginSites]
targetNames <- 1:nTargetSites
psiTrue <- array(c(0.1, 0.2, 0.3, 0.4,
                   0.2, 0.3, 0.4, 0.1,
                   0.3, 0.4, 0.1, 0.2,
                   0.4, 0.1, 0.2, 0.3),
                 c(nOriginSites, nTargetSites),
                 dimnames = list(originNames, targetNames))
rowSums(psiTrue)
rTrue <- c(0.5, 0.05, 0.3, 0.6)
banded1 <- c(500, 1000, 2000, 3000)
reencountered1 <- simCMRData(psiTrue, banded1, rTrue)$reencountered
psi_r_calc_sloppy <- calcTransition(banded = banded1,
                            reencountered = reencountered1,
                            originNames = originNames,
                            targetNames = targetNames,
                            method = "BFGS")
psi_r_calc_sloppy

psi_r_calc <- calcTransition(banded = banded1,
                            reencountered = reencountered1,
                            originNames = originNames,
                            targetNames = targetNames,
                            method = "SANN")
psi_r_calc
psi_r_mcmc <- estTransition(banded = banded1, reencountered = reencountered1,
                           originNames = originNames,
                           targetNames = targetNames,
                           method = "MCMC",
                           nSamples = 45000, nBurnin = 5000, #reduced for example speed
```

```
                              nThin = 1, verbose = 0)
print(psi_r_mcmc)
psi_r_boot <- estTransition(banded = banded1, reencountered = reencountered1,
                              originNames = originNames,
                              targetNames = targetNames,
                              method = "bootstrap",
                              nSamples = 200) #reduced for example speed
print(psi_r_boot)
```

`diffMantel` *Pairwise differences between two or more independent Mantel correlation estimates*

Description

Estimates mean (and median) differences in Mantel correlations (rM), and includes measures of uncertainty (SE and CI). For those measures of uncertainty to be accurate, only apply this function to rM estimates where all data sources are independent (e.g., different species).

Usage

```
diffMantel(estimates, nSamples = 1e+05, alpha = 0.05, returnSamples = FALSE)

diffCorr(estimates, nSamples = 1e+05, alpha = 0.05, returnSamples = FALSE)
```

Arguments

`estimates`	List of at least two Mantel correlation estimates, provided by either the estMC or the estMantel functions. If this is a named list (recommended), the function will use these names in labeling the differences.
`nSamples`	A positive integer, number of samples (with replacement) to draw from each pair of MC estimates (default 100000). If set to NULL, compares all Mantel correlation samples from each pair.
`alpha`	Level for confidence/credible intervals provided.
`returnSamples`	Should the function return all the sampled differences? Defaults to FALSE to reduce storage requirements. Change to TRUE to compute your own summary statistics.

Value

`diffMantel` returns a list with elements:

`meanDiff`, `medianDiff` Vectors with mean and medians of sampled differences for each pairwise comparison. Estimates of difference between rM values incorporating parametric uncertainty.

`seDiff` Vector with standard errors of rM differences for each pairwise comparison, estimated from SD of sampled differences.

`simpleCI` Matrix of `1 - alpha` confidence intervals for rM differences, estimated as `alpha/2` and `1 - alpha/2` quantiles of `sampleCorr`.

`bcCI` Matrix of bias-corrected `1 - alpha` confidence intervals for rM differences for each pairwise comparison. Preferable to `simpleCI` when `meanDiff` is the best estimate of the rM difference. `simpleCI` is preferred when `medianDiff` is a better estimator. When `meanDiff==medianDiff`, these should be identical. Estimated as the `pnorm(2 * z0 + qnorm(alpha / 2))` and `pnorm(2 * z0 + qnorm(1 - alpha / 2))` quantiles of sampled differences, where z0 is the proportion of `sampleDiff < meanDiff`.

`sampleDiff` Only provided if `returnSamples` is TRUE. List of sampled values for each pairwise rM difference.

References

Cohen, E. B., C. S. Rushing, F. R. Moore, M. T. Hallworth, J. A. Hostetler, M. Gutierrez Ramirez, and P. P. Marra. 2019. The strength of migratory connectivity for birds en route to breeding through the Gulf of Mexico. Ecography 42: 658 - 669. doi:10.1111/ecog.03974

`diffMC` *Pairwise differences between two or more independent MC estimates*

Description

Estimates mean (and median) differences in MC, and includes measures of uncertainty (SE and CI). For those measures of uncertainty to be accurate, only apply this function to MC estimates where all data sources are independent (e.g., different species).

Usage

```
diffMC(estimates, nSamples = 1e+05, alpha = 0.05, returnSamples = FALSE)

diffStrength(estimates, nSamples = 1e+05, alpha = 0.05, returnSamples = FALSE)
```

Arguments

`estimates`	List of at least two MC estimates, provided by the estMC function. If this is a named list (recommended), the function will use these names in labeling the differences.
`nSamples`	A positive integer, number of samples (with replacement) to draw from each pair of MC estimates (default 100000). If set to NULL, compares all MC samples from each pair.
`alpha`	Level for confidence/credible intervals provided.
`returnSamples`	Should the function return all the sampled differences? Defaults to FALSE to reduce storage requirements. Change to TRUE to compute your own summary statistics.

Value

diffMC returns a list with elements:

meanDiff, medianDiff Vectors with mean and medians of sampled differences for each pairwise comparison. Estimates of difference between MC values incorporating parametric uncertainty.

seDiff Vector with standard errors of MC differences for each pairwise comparison, estimated from SD of sampled differences.

simpleCI Matrix of 1 - alpha confidence intervals for MC differences, estimated as alpha/2 and 1 - alpha/2 quantiles of sampleMC.

bcCI Matrix of bias-corrected 1 - alpha confidence intervals for MC differences for each pairwise comparison. Preferable to simpleCI when meanDiff is the best estimate of the MC difference. simpleCI is preferred when medianDiff is a better estimator. When meanDiff==medianDiff, these should be identical. Estimated as the pnorm(2 * z0 + qnorm(alpha / 2)) and pnorm(2 * z0 + qnorm(1 - alpha / 2)) quantiles of sampled differences, where z0 is the proportion of sampleDiff < meanDiff.

sampleDiff Only provided if returnSamples is TRUE. List of sampled values for each pairwise MC difference.

References

Cohen, E. B., C. S. Rushing, F. R. Moore, M. T. Hallworth, J. A. Hostetler, M. Gutierrez Ramirez, and P. P. Marra. 2019. The strength of migratory connectivity for birds en route to breeding through the Gulf of Mexico. Ecography 42: 658 - 669. doi:10.1111/ecog.03974

Examples

```
data('OVENdata')
ovenPsi <- estTransition(isGL = OVENdata$isGL, #Logical vector:light-level GL(T)
                 isTelemetry = !OVENdata$isGL,
                 geoBias = OVENdata$geo.bias, # Light-level GL location bias
                 geoVCov = OVENdata$geo.vcov, # Location covariance matrix
                 targetSites = OVENdata$targetSites, # Non-breeding target sites
                 originSites = OVENdata$originSites, # Breeding origin sites
                 originPoints = OVENdata$originPoints, # Capture Locations
                 targetPoints = OVENdata$targetPoints, # Device target locations
                 verbose = 0,   # output options
                 nSamples = 100, # This is set low for example
                 resampleProjection = sf::st_crs(OVENdata$targetSites))
ovenEst <- estStrength(targetDist = OVENdata$targetDist, # targetSites distance matrix
                 originDist = OVENdata$originDist, # originSites distance matrix
                 originRelAbund = OVENdata$originRelAbund,#Origin relative abund
                 psi = ovenPsi,
                 verbose = 1,   # output options
                 nSamples = 1000)
fm <- getCMRexample()
originPos13 <- matrix(c(rep(seq(-99, -81, 2), each = 10),
                        rep(seq(49, 31, -2), 10)), 100, 2)
targetPos13 <- matrix(c(rep(seq(-79, -61, 2), each = 10),
                        rep(seq(9, -9, -2), 10)), 100, 2)
originPosCMR <- rowsum(originPos13, c(rep(1:2, 5, each = 5),
```

```
                                      rep(3:4, 5, each = 5))) / 25
targetPosCMR <- rowsum(targetPos13, c(rep(1:2, 5, each = 5),
                                      rep(3:4, 5, each = 5))) / 25
originDist <- distFromPos(originPosCMR, 'ellipsoid')
targetDist <- distFromPos(targetPosCMR, 'ellipsoid')
originRelAbundTrue <- rep(0.25, 4)
theorEst <- estStrength(originRelAbund = originRelAbundTrue, psi = fm,
                  originDist = originDist, targetDist = targetDist,
                  originSites = 5:8, targetSites = c(3,2,1,4),
                  nSamples = 1000, verbose = 0,
                  sampleSize = length(grep("[2-5]", fm$data$data$ch)))
ovenEst
theorEst
diff1 <- diffMC(estimates = list(Ovenbird = ovenEst, Theorybird = theorEst),
                nSamples = 10000, returnSamples = TRUE)
```

`distFromPos` *Distance matrix from position matrix*

Description

Distance matrix from position matrix

Usage

```
distFromPos(
  pos,
  surface = "ellipsoid",
  units = c("km", "m", "miles", "nautical miles")
)
```

Arguments

`pos`	Number of sites by 2 matrix with positions of each site. If `surface` is 'ellipsoid' or 'sphere', then column 1 should be longitude and column 2 should be latitude. If `surface` is 'plane', column 1 can be x-position and column 2 y-position.
`surface`	Surface to calculate distances on. Either 'ellipsoid' (default), 'sphere', or 'plane'.
`units`	Units of return distance matrix. If `surface` is 'plane', then this argument is ignored and the return units will be the same as the `pos` units. Options are 'km' (kilometers, default), 'm' (meters), 'miles', and 'nautical miles'.

Value

Square matrix of distances between sites. If `surface` is 'ellipsoid' or 'sphere', then argument `units` will determine units; if `surface` is 'plane', the units will be the same as the `pos` units.

Note

In version 0.4.3 we switched package dependencies from `geosphere` to `geodist`. As a result, spherical distances (and possibly ellipsoid distances) may differ slightly from those calculated with earlier versions of our package.

Examples

```
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')
breedDist[1:12, 1:12]
breedDist[1:12, c(1,91,100)]
```

`estMantel` *Estimate Mantel correlation (rM) from geolocator, GPS, and/or raster data.*

Description

Resampling of uncertainty for migratory connectivity strength, as quantified by Mantel correlation (rM), from geolocators, GPS, and/or raster (e.g., genoscape or isotope) data.

Usage

```
estMantel(
  targetPoints = NULL,
  originPoints = NULL,
  isGL,
  geoBias = NULL,
  geoVCov = NULL,
  targetSites = NULL,
  nBoot = 1000,
 nSim = ifelse(any(isRaster & isGL), 5000, ifelse(any(isGL), 1000, ifelse(any(isRaster),
    10, 1))),
  verbose = 0,
  alpha = 0.05,
```

```
  resampleProjection = "ESRI:102010",
  maxTries = 300,
  maintainLegacyOutput = FALSE,
  originSites = NULL,
  isTelemetry = !isGL,
  isRaster = FALSE,
  captured = "origin",
  geoBiasOrigin = geoBias,
  geoVCovOrigin = geoVCov,
  targetRaster = NULL,
  originRaster = NULL,
  dataOverlapSetting = c("dummy", "none", "named"),
  originRelAbund = NULL,
  targetRelAbund = NULL
)

estCorr(
  targetPoints = NULL,
  originPoints = NULL,
  isGL,
  geoBias = NULL,
  geoVCov = NULL,
  targetSites = NULL,
  nBoot = 1000,
 nSim = ifelse(any(isRaster & isGL), 5000, ifelse(any(isGL), 1000, ifelse(any(isRaster),
    10, 1))),
  verbose = 0,
  alpha = 0.05,
  resampleProjection = "ESRI:102010",
  maxTries = 300,
  maintainLegacyOutput = FALSE,
  originSites = NULL,
  isTelemetry = !isGL,
  isRaster = FALSE,
  captured = "origin",
  geoBiasOrigin = geoBias,
  geoVCovOrigin = geoVCov,
  targetRaster = NULL,
  originRaster = NULL,
  dataOverlapSetting = c("dummy", "none", "named"),
  originRelAbund = NULL,
  targetRelAbund = NULL
)
```

Arguments

targetPoints	A POINTS from sf object, with length number of animals tracked. Each point indicates the point estimate location in the non-release season.

originPoints A POINTS from sf object, with length number of animals tracked. Each point indicates the release location of an animal.

isGL Indicates whether or which animals were tracked with geolocators Should be either single TRUE or FALSE value, or vector with length of number of animals tracked, with TRUE for animals in targetPoints with geolocators and FALSE for animals without.

geoBias For GL data, vector of length 2 indicating expected bias in longitude and latitude of targetPoints, in resampleProjection units (default meters).

geoVCov For GL data, 2x2 matrix with expected variance/covariance in longitude and latitude of targetPoints, in resampleProjection units (default meters).

targetSites A SpatialPolygons, SpatialPolygonsDataFrame, or POLYGONS sf object indicating valid target location(s). Not needed unless you want to mask out certain areas (e.g. water) and captured is "origin" or you want to use a weighted bootstrap based on targetRelAbund for animals captured on the target side.

nBoot Number of bootstrap runs. Animals are sampled with replacement for each, to estimate sampling uncertainty.

nSim Tuning parameter for GL or raster data. Affects only the speed; 1000 seems to work well with our GL data. Should be integer > 0.

verbose 0 (default) to 3. 0 prints no output during run. 1 prints a line every 100 bootstraps. 2 prints a line every bootstrap. 3 also prints the number of draws (for tuning nSim only).

alpha Level for confidence/credible intervals provided.

resampleProjection
Projection when sampling from geolocator bias/error. This projection needs units = m. Default is Equidistant Conic. The default setting preserves distances around latitude = 0 and longitude = 0. Other projections may work well, depending on the location of targetPoints.

maxTries Maximum number of times to run a single GL bootstrap before exiting with an error. Default is 300. Set to NULL to never stop. This parameter was added to prevent GL setups where some sample points never land on target sites from running indefinitely.

maintainLegacyOutput
version 0.4.0 of MigConnectivity updated the structure of the estimates. If you have legacy code that refers to elements within a estMigConnectivity object, you can set this to TRUE to also keep the old structure. Defaults to FALSE.

originSites A SpatialPolygons, SpatialPolygonsDataFrame, or POLYGONS sf object indicating valid origin location(s). Not needed unless you want to mask out certain areas (e.g. water) and captured is "target" or you want to use a weighted bootstrap based on originRelAbund for animals captured on the origin side.

isTelemetry Indicates whether or which animals were tracked with telemetry/GPS (no location uncertainty on either end). Should be either single TRUE or FALSE value, or vector with length of number of animals tracked, with TRUE or FALSE for each animal in data.

`isRaster` Indicates whether or which animals were tracked with intrinsic markers (e.g., genetics or isotopes), with location uncertainty expressed as a raster of probabilities by grid cells, either in `targetRaster` or `originRaster`. Should be either single TRUE or FALSE value, or vector with length of number of animals tracked, with TRUE or FALSE for each animal in data.

`captured` Indicates whether or which animals were captured in the origin sites, the target sites, or neither (another phase of the annual cycle). Location uncertainty will only be applied where the animal was not captured. So this doesn't matter for telemetry data. Should be either single "origin" (default), "target", or "neither" value, or a character vector with length of number of animals tracked, with "origin", "target", or "neither" for each animal.

`geoBiasOrigin` For GL data where `captured!="origin"`, vector of length 2 indicating expected bias in longitude and latitude of `originPoints`, in `resampleProjection` units (default meters).

`geoVCovOrigin` For GL data where `captured!="origin"`, 2x2 matrix with expected variance/covariance in longitude and latitude of `targetPoints`, in `resampleProjection` units (default meters).

`targetRaster` For intrinsic tracking data, the results of `isoAssign` or a similar function of class `intrinsicAssign` or class `RasterBrick/RasterStack`, for example from the package `assignR`. In any case, it expresses location uncertainty on target range, through a raster of probabilities by grid cells.

`originRaster` For intrinsic tracking data, the results of `isoAssign` or a similar function of class `intrinsicAssign` or class `RasterBrick/RasterStack`, for example from the package `assignR`. In any case, it expresses location uncertainty on origin range, through a raster of probabilities by grid cells.

`dataOverlapSetting`
When there is more than one type of data, this setting allows the user some flexibility for clarifying which type(s) of data apply to which animals. Setting "dummy" (the default) indicates that there are dummy values within each dataset for the animals that isGL, isTelemetry, etc. don't have that data type (FALSE values). If no animals have a data type, no dummy values are required. If no animals have more than one type of data, the user can simplify processing their data by choosing setting "none" here. In this case, there should be no dummy values, and only the animals with a type of data should be included in that dataset. The third setting ("named") is not yet implemented, but will eventually allow another way to allow animals with more than one type of data with named animals linking records. When there is only one type of data, it is fastest to leave this on the default.

`originRelAbund` the proportion of the total abundance in each of B `originSites`. Used to set up the bootstrap to be weighted by relative abundance (for animals captured on the origin side). Either a numeric vector of length B that sums to 1, or an mcmc object (such as is produced by `modelCountDataJAGS`) or matrix with at least B columns. If there are more than B columns, the relevant columns should be labeled "relN[1]" through "relN[B]". Optional, but if you don't set it and at least some animals are captured on the origin side, there's potential for rM to be biased (if sampling isn't proportional to abundance).

`targetRelAbund` the proportion of the total abundance in each of W `targetSites`. Used to set up the bootstrap to be weighted by relative abundance (for animals captured on the target side). Either a numeric vector of length W that sums to 1, or an mcmc object (such as is produced by `modelCountDataJAGS`) or matrix with at least W columns. If there are more than W columns, the relevant columns should be labeled "relN[1]" through "relN[W]". Optional, but if you don't set it and at least some animals are captured on the target side, there's potential for rM to be biased (if sampling isn't proportional to abundance).

Value

`estMantel` returns a list with elements:

`corr` List containing estimates of rM:

- `sample` `nBoot` sampled values for Mantel correlation. Provided to allow the user to compute own summary statistics.
- `mean`, `se`, `simpleCI`, `bcCI`, `median`, `point` Summary statistics for Mantel correlation bootstraps.

`input` List containing the inputs to `estMantel`

References

Cohen, E. B., J. A. Hostetler, M. T. Hallworth, C. S. Rushing, T. S. Sillett, and P. P. Marra. 2018. Quantifying the strength of migratory connectivity. Methods in Ecology and Evolution 9: 513 - 524. doi:10.1111/2041210X.12916

See Also

`estMC`

Examples

```
data('OVENdata')
rM1 <- estMantel(isGL=OVENdata$isGL,#Logical vector: light-level GL(T)/GPS(F)
                 geoBias = OVENdata$geo.bias, # Geolocator location bias
                 geoVCov = OVENdata$geo.vcov, # Location covariance matrix
                 targetSites = OVENdata$targetSites,#Nonbreeding/target sites
                 originPoints = OVENdata$originPoints, # Capture Locations
                 targetPoints = OVENdata$targetPoints, # Target locations
                 verbose = 1,   # output options
                 nBoot = 10, # This is set low for example
                 resampleProjection = sf::st_crs(OVENdata$targetSites))
rM1
str(rM1, max.level = 2)
```

estMC *Estimate migratory connectivity*

Description

Resampling of uncertainty for migratory connectivity strength (MC) and transition probabilities (psi) from RMark psi matrix estimates or samples of psi and/or JAGS relative abundance MCMC samples OR SpatialPoints geolocators and/or GPS data OR intrinsic markers such as isotopes. NOTE: active development of this function is ending. We suggest users estimate psi with `estTransition`, MC with `estStrength`, and Mantel correlations (rM) with `estMantel`.

Usage

```
estMC(
  originDist,
  targetDist = NULL,
  originRelAbund,
  psi = NULL,
  sampleSize = NULL,
  originSites = NULL,
  targetSites = NULL,
  originPoints = NULL,
  targetPoints = NULL,
  originAssignment = NULL,
  targetAssignment = NULL,
  originNames = NULL,
  targetNames = NULL,
  nSamples = 1000,
  nSim = ifelse(isTRUE(isIntrinsic), 10, 1000),
  isGL = FALSE,
  geoBias = NULL,
  geoVCov = NULL,
  row0 = 0,
  verbose = 0,
  calcCorr = FALSE,
  alpha = 0.05,
  approxSigTest = FALSE,
  sigConst = 0,
  resampleProjection = "ESRI:102010",
  maxTries = 300,
  targetIntrinsic = NULL,
  isIntrinsic = FALSE,
  maintainLegacyOutput = FALSE
)
```

Arguments

originDist	Distances between the B origin sites. Symmetric B by B matrix

`targetDist` Distances between the W target sites. Symmetric W by W matrix. Optional for intrinsic data

`originRelAbund` Relative abundance estimates at B origin sites. Either a numeric vector of length B that sums to 1 or an mcmc object with `nSamples` rows and columns including 'relN[1]' through 'relN[B]'. Currently, an mcmc object doesn't work with geolocator, GPS, or intrinsic data

`psi` Transition probabilities between B origin and W target sites. Either a matrix with B rows and W columns where rows sum to 1, an array with dimensions x, B, and W (with x samples of the transition probability matrix from another model), or a MARK object with estimates of transition probabilities. If you are estimating MC from GPS, geolocator, or intrinsic data, leave this as NULL

`sampleSize` Total sample size of animals that psi will be estimated from. Should be the number of animals released in one of the origin sites and observed in one of the target sites. Optional, but recommended, unless you are estimating MC from GPS, geolocator, intrinsic, or direct band return data (in which case the function can calculate it for you)

`originSites` If `psi` is a MARK object, this must be a numeric vector indicating which sites are origin. If using GPS, geolocator, or intrinsic data, this can be the geographic definition of sites in the release season

`targetSites` If `psi` is a MARK object, this must be a numeric vector indicating which sites are target. If using GPS, geolocator, or intrinsic data, this must be the geographic definition of sites in the non-release season. Optional for intrinsic data; if left out, the function will use the `targetSites` defined in `targetIntrinsic`

`originPoints` A `POINT` sf object, with length number of animals tracked. Each point indicates the release location of an animal

`targetPoints` For GL or GPS data, a `POINT` sf object, with length number of animals tracked. Each point indicates the point estimate location in the non-release season

`originAssignment`
Assignment of `originPoints` to release season sites. Integer vector with length number of animals tracked. Optional, but if using GL or GPS data, either `originAssignment` or `originSites` and `originPoints` should be defined

`targetAssignment`
Optional. Point estimate assignment of `targetPoints` to non-release season sites. Integer vector with length number of animals tracked

`originNames` Optional but recommended. Vector of names for the release season sites

`targetNames` Optional but recommended. Vector of names for the non-release season sites

`nSamples` Number of times to resample `psi` and/or `originRelAbund` OR number of post-burn-in MCMC samples to store (band data) OR number of times to resample `targetPoints` for intrinsic data OR number of bootstrap runs for GL or GPS data. In the last two cases, animals are sampled with replacement for each. For all, the purpose is to estimate sampling uncertainty

`nSim` Tuning parameter for GL or intrinsic data. Affects only the speed; 1000 seems to work well with our GL data and 10 for our intrinsic data, but your results may vary. Should be integer > 0

isGL — Indicates whether or which animals were tracked with geolocators. Should be either single TRUE or FALSE value, or vector with length of number of animals tracked, with TRUE for animals in targetPoints with geolocators and FALSE for animals with GPS

geoBias — For GL data, vector of length 2 indicating expected bias in longitude and latitude of targetPoints, in resampleProjection units (default meters)

geoVCov — For GL data, 2x2 matrix with expected variance/covariance in longitude and latitude of targetPoints, in resampleProjection units (default meters)

row0 — If originRelAbund is an mcmc object, this can be set to 0 (default) or any greater integer to specify where to stop ignoring samples ("burn-in")

verbose — 0 (default) to 3. 0 prints no output during run. 1 prints a line every 100 samples or bootstraps and a summary every 10. 2 prints a line and summary every sample or bootstrap. 3 also prints the number of draws (for tuning nSim for GL/intrinsic data only)

calcCorr — In addition to MC, should function also estimate Mantel correlation between release and non-release locations (GPS or GL data only)? Default is FALSE

alpha — Level for confidence/credible intervals provided

approxSigTest — Should function compute approximate one-sided significance tests (p-values) for MC from the bootstrap? Default is FALSE

sigConst — Value to compare MC to in significance test. Default is 0

resampleProjection — Projection when sampling from geolocator bias/error. This projection needs units = m. Default is Equidistant Conic. The default setting preserves distances around latitude = 0 and longitude = 0. Other projections may work well, depending on the location of targetSites. Ignored unless data are geolocator or GPS

maxTries — Maximum number of times to run a single GL/intrinsic bootstrap before exiting with an error. Default is 300. Set to NULL to never stop. This parameter was added to prevent GL setups where some sample points never land on target sites from running indefinitely

targetIntrinsic — For intrinsic tracking data, the results of isoAssign or a similar function, of class intrinsicAssign

isIntrinsic — Logical indicating whether the animals are tracked via intrinsic marker (e.g. isotopes) or not. Currently estMC will only estimate connectivity for all intrinsically marked animals or all extrinsic (e.g., bands, GL, or GPS), so isIntrinsic should be a single TRUE or FALSE

maintainLegacyOutput — version 0.4.0 of MigConnectivity updated the structure of the estimates. If you have legacy code that refers to elements within a estMigConnectivity object, you can set this to TRUE to also keep the old structure. Defaults to FALSE

Value

NOTE: Starting with version 0.4.0 of MigConnectivity, we've updated the structure of MigConnectivityEstimate objects. Below we describe the updated structure. If parameter maintainLegacyOutput is set to

TRUE, the list will start with the old structure: sampleMC, samplePsi, pointPsi, pointMC, meanMC, medianMC, seMC, simpleCI, bcCI, hpdCI, simpleP, bcP, sampleCorr, pointCorr, meanCorr, medianCorr, seCorr, simpleCICorr, bcCICorr, inputSampleSize, alpha, and sigConst.

estMC returns a list with the elements:

psi List containing estimates of transition probabilities:

- sample Array of sampled values for psi. nSamples x [number of origin sites] x [number of target sites]. Provided to allow the user to compute own summary statistics.
- mean Main estimate of psi matrix. [number of origin sites] x [number of target sites].
- se Standard error of psi, estimated from SD of psi$sample.
- simpleCI 1 - alpha confidence interval for psi, estimated as alpha/2 and 1 - alpha/2 quantiles of psi$sample.
- bcCI Bias-corrected 1 - alpha confidence interval for psi. Preferable to simpleCI when mean is the best estimate of psi. simpleCI is preferred when median is a better estimator. When meanMC==medianMC, these should be identical. Estimated as the pnorm(2 * z0 + qnorm(alpha / 2)) and pnorm(2 * z0 + qnorm(1 - alpha / 2)) quantiles of sample, where z0 is the proportion of sample < mean.
- median Median estimate of psi matrix.
- point Simple point estimate of psi matrix, not accounting for sampling error. NULL when isIntrinsic == TRUE.

MC List containing estimates of migratory connectivity strength:

- sample nSamples sampled values for MC. Provided to allow the user to compute own summary statistics.
- mean Mean of MC$sample. Main estimate of MC, incorporating parametric uncertainty.
- se Standard error of MC, estimated from SD of MC$sample.
- simpleCI Default1 - alpha confidence interval for MC, estimated as alpha/2 and 1 - alpha/2 quantiles of MC$sample.
- bcCI Bias-corrected 1 - alpha confidence interval for MC. Preferable to MC$simpleCI when MC$mean is the best estimate of MC. MC$simpleCI is preferred when MC$median is a better estimator. When MC$mean==MC$median, these should be identical. Estimated as the pnorm(2 * z0 + qnorm(alpha / 2)) and pnorm(2 * z0 + qnorm(1 - alpha / 2)) quantiles of MC$sample, where z0 is the proportion of MC$sample < MC$mean.
- hpdCI 1 - alpha credible interval for MC, estimated using the highest posterior density (HPD) method.
- median Median of MC, alternate estimator also including parametric uncertainty.
- point Simple point estimate of MC, using the point estimates of psi and originRelAbund, not accounting for sampling error. NULL when isIntrinsic == TRUE.
- simpleP Approximate p-value for MC, estimated as the proportion of bootstrap iterations where MC < sigConst (or MC > sigConst if pointMC < sigConst). Note that if the proportion is 0, a default value of 0.5 / nSamples is provided, but this is best interpreted as p < 1 / nSamples. NULL when approxSigTest==FALSE.
- bcP Approximate bias-corrected p-value for MC, estimated as pnorm(qnorm(simpleP) - 2 * z0), where z0 is the proportion of sampleMC < meanMC. May be a better approximation of the p-value than simpleP, but many of the same limitations apply. NULL when approxSigTest==FALSE.

`corr` List containing estimates of rM, an alternate measure of migratory connectivity strength. NULL when `calcCorr==FALSE` or `!is.null(psi)`:

- `sample nBoot` sampled values for continuous correlation. Provided to allow the user to compute own summary statistics.
- `mean, se, simpleCI, bcCI, median, point` Summary statistics for continuous correlation bootstraps.

`input` List containing the inputs to `estMC`, or at least the relevant ones, such as sampleSize.

References

Cohen, E. B., J. A. Hostetler, M. T. Hallworth, C. S. Rushing, T. S. Sillett, and P. P. Marra. 2018. Quantifying the strength of migratory connectivity. Methods in Ecology and Evolution 9: 513 - 524. doi:10.1111/2041210X.12916

Cohen, E. B., C. S. Rushing, F. R. Moore, M. T. Hallworth, J. A. Hostetler, M. Gutierrez Ramirez, and P. P. Marra. 2019. The strength of migratory connectivity for birds en route to breeding through the Gulf of Mexico. Ecography 42: 658 - 669. doi:10.1111/ecog.03974

See Also

`estStrength`, `estTransition`, `estMantel`, `calcMC`, `projections`, `isoAssign`, `plot.estMigConnectivity`

Examples

```
set.seed(101)
# Uncertainty in detection ('RMark' estimates) with equal abundances
# Number of resampling iterations for generating confidence intervals

nSamplesCMR <- 100
nSimulationsCMR <- 10
originPos13 <- matrix(c(rep(seq(-99, -81, 2), each = 10),
                        rep(seq(49, 31, -2), 10)), 100, 2)
targetPos13 <- matrix(c(rep(seq(-79, -61, 2), each = 10),
                        rep(seq(9, -9, -2), 10)), 100, 2)
originPosCMR <- rowsum(originPos13, c(rep(1:2, 5, each = 5),
                                      rep(3:4, 5, each = 5))) / 25
originPosCMR
targetPosCMR <- rowsum(targetPos13, c(rep(1:2, 5, each = 5),
                                      rep(3:4, 5, each = 5))) / 25
targetPosCMR

originDist <- distFromPos(originPosCMR, 'ellipsoid')
targetDist <- distFromPos(targetPosCMR, 'ellipsoid')
originRelAbundTrue <- rep(0.25, 4)
# the second intermediate psi scenario, the "low" level
psiTrue <- samplePsis[["Low"]]
trueMC <- calcMC(originDist, targetDist, originRelAbundTrue, psiTrue)
trueMC

# Storage matrix for samples
cmrMCSample <- matrix(NA, nSamplesCMR, nSimulationsCMR)
```

```
summaryCMR <- data.frame(Simulation = 1:nSimulationsCMR, True=trueMC,
                         mean=NA, se=NA, lcl=NA, ucl=NA)
# Get RMark psi estimates and estimate MC from each
for (r in 1:nSimulationsCMR) {
  cat("Simulation",r,"of",nSimulationsCMR,"\n")
  # Note: getCMRexample() requires a valid internet connection and that GitHub
  # is accessible
  fm <- getCMRexample(r)
  results <- estMC(originRelAbund = originRelAbundTrue, psi = fm,
                   originDist = originDist, targetDist = targetDist,
                   originSites = 5:8, targetSites = c(3,2,1,4),
                   nSamples = nSamplesCMR, verbose = 0,
                   sampleSize = length(grep('[2-5]', fm$data$data$ch)))
  #sampleSize argument not really needed (big sample sizes)
  cmrMCSample[ , r] <- results$MC$sample
  summaryCMR$mean[r] <- results$MC$mean
  summaryCMR$se[r] <- results$MC$se
  # Calculate confidence intervals using quantiles of sampled MC
  summaryCMR[r, c('lcl', 'ucl')] <- results$MC$simpleCI
}

summaryCMR <- transform(summaryCMR, coverage = (True>=lcl & True<=ucl))
summaryCMR
summary(summaryCMR)
biasCMR <- mean(summaryCMR$mean) - trueMC
biasCMR
mseCMR <- mean((summaryCMR$mean - trueMC)^2)
mseCMR
rmseCMR <- sqrt(mseCMR)
rmseCMR


# Simulation of BBS data to quantify uncertainty in relative abundance

nSamplesAbund <- 700 #1700 are stored
nSimulationsAbund <- 10
# Storage matrix for samples
abundMCSample <- matrix(NA, nSamplesAbund, nSimulationsAbund)
summaryAbund <- data.frame(Simulation = 1:nSimulationsAbund, True = trueMC,
                           mean = NA, se = NA, lcl = NA, ucl = NA)
for (r in 1:nSimulationsAbund) {
  cat("Simulation",r,"of",nSimulationsAbund,"\n")
  row0 <- nrow(abundExamples[[r]]) - nSamplesAbund
  results <- estMC(originRelAbund = abundExamples[[r]], psi = psiTrue,
                   originDist = originDist, targetDist = targetDist,
                   row0 = row0, nSamples = nSamplesAbund, verbose = 1)
  abundMCSample[ , r] <- results$MC$sample
  summaryAbund$mean[r] <- results$MC$mean
  summaryAbund$se[r] <- results$MC$se
  # Calculate confidence intervals using quantiles of sampled MC
  summaryAbund[r, c('lcl', 'ucl')] <- results$MC$simpleCI
}
```

```
summaryAbund <- transform(summaryAbund,
                          coverage = (True >= lcl & True <= ucl))
summaryAbund
summary(summaryAbund)
biasAbund <- mean(summaryAbund$mean) - trueMC
biasAbund
mseAbund <- mean((summaryAbund$mean - trueMC)^2)
mseAbund
rmseAbund <- sqrt(mseAbund)
rmseAbund

# Ovenbird example with GL and GPS data
data(OVENdata) # Ovenbird

nSamplesGLGPS <- 100 # Number of bootstrap iterations

# Estimate MC only, treat all data as geolocator
GL_mc<-estMC(isGL=TRUE, # Logical vector: light-level geolocator(T)/GPS(F)
             geoBias = OVENdata$geo.bias, #Geolocator location bias
             geoVCov = OVENdata$geo.vcov, # Location covariance matrix
             targetDist = OVENdata$targetDist, # targetSites distance matrix
             originDist = OVENdata$originDist, # originSites distance matrix
             targetSites = OVENdata$targetSites, # Non-breeding target sites
             originSites = OVENdata$originSites, # Breeding origin sites
             originPoints = OVENdata$originPoints, # Capture Locations
             targetPoints = OVENdata$targetPoints, # Device target locations
             originRelAbund = OVENdata$originRelAbund,#Origin relative abund.
             verbose = 1,   # output options
             nSamples = nSamplesGLGPS,# This is set low for example
             resampleProjection = terra::crs(OVENdata$targetSites))

# Estimate MC and rM, treat all data as is
Combined<-estMC(isGL=OVENdata$isGL, #Logical vector:light-level GL(T)/GPS(F)
                geoBias = OVENdata$geo.bias, # Light-level GL location bias
                geoVCov = OVENdata$geo.vcov, # Location covariance matrix
                targetDist = OVENdata$targetDist, # Winter distance matrix
                originDist = OVENdata$originDist, # Breeding distance matrix
                targetSites = OVENdata$targetSites, # Nonbreeding/target sites
                originSites = OVENdata$originSites, # Breeding origin sites
                originPoints = OVENdata$originPoints, # Capture Locations
                targetPoints = OVENdata$targetPoints, #Device target locations
                originRelAbund = OVENdata$originRelAbund,#Relative abundance
                verbose = 1,   # output options
                calcCorr = TRUE, # estimate rM as well
                nSamples = nSamplesGLGPS, # This is set low for example
                approxSigTest = TRUE,
                resampleProjection = terra::crs(OVENdata$targetSites),
                originNames = OVENdata$originNames,
                targetNames = OVENdata$targetNames)

print(Combined)

# For treating all data as GPS,
```

```
# Move the latitude of birds with locations that fall offshore - only change
# Latitude
int <- sf::st_intersects(OVENdata$targetPoints, OVENdata$targetSites)
any(lengths(int)<1)
plot(OVENdata$targetPoints)
plot(OVENdata$targetSites,add=TRUE)
tp<-sf::st_coordinates(OVENdata$targetPoints)
text(tp[,1], tp[,2], label=c(1:39))

tp[5,2]<- -1899469
tp[10,2]<- -1927848
tp[1,2]<- -1927930
tp[11,2]<- -2026511
tp[15,2]<- -2021268
tp[16,2]<- -1976063

oven_targetPoints<-sf::st_as_sf(as.data.frame(tp),
                                coords = c("X","Y"),
                                crs = sf::st_crs(OVENdata$targetPoints))
inter <- sf::st_intersects(oven_targetPoints, OVENdata$targetSites)
any(lengths(inter)<1)
plot(oven_targetPoints,add=TRUE, col = "green")

# Estimate MC only, treat all data as GPS
GPS_mc<-estMC(isGL=FALSE, # Logical vector: light-level geolocator(T)/GPS(F)
              targetDist = OVENdata$targetDist, # targetSites distance matrix
              originDist = OVENdata$originDist, # originSites distance matrix
              targetSites = OVENdata$targetSites, # Non-breeding target sites
              originSites = OVENdata$originSites, # Breeding origin sites
              originPoints = OVENdata$originPoints, # Capture Locations
              targetPoints = oven_targetPoints, # Device target locations
              originRelAbund = OVENdata$originRelAbund,#Origin relative abund.
              verbose = 1,   # output options
              nSamples = nSamplesGLGPS) # This is set low for example

str(GPS_mc, max.level = 2)
str(Combined, max.level = 2)
str(GL_mc, max.level = 2)
plot(Combined, legend = "top", main = "Ovenbird GL and GPS")
text(1.1, 0.98, cex = 1,
     labels = paste("MC = ", round(Combined$MC$mean, 2), "+/-",
                    round(Combined$MC$se, 2)))


# Generate probabilistic assignments using intrinsic markers (stable-hydrogen
# isotopes)
getCSV <- function(filename) {
  tmp <- tempdir()
  url1 <- paste0('https://github.com/SMBC-NZP/MigConnectivity/blob/master/data-raw/',
                 filename, '?raw=true')
  temp <- paste(tmp, filename, sep = '/')
  utils::download.file(url1, temp, mode = 'wb')
  csv <- read.csv(temp)
```

```
  unlink(temp)
  return(csv)
}

getRDS <- function(speciesDist) {
  tmp <- tempdir()
  extension <- '.rds'
  filename <- paste0(speciesDist, extension)
  url1 <- paste0(
   'https://github.com/SMBC-NZP/MigConnectivity/blob/master/data-raw/Spatial_Layers/',
                filename, '?raw=true')
  temp <- paste(tmp, filename, sep = '/')
  utils::download.file(url1, temp, mode = 'wb')
  shp <- readRDS(temp)
  unlink(temp)
  return(shp)
}
OVENdist <- getRDS("OVENdist")

OVENvals <- getCSV("deltaDvalues.csv")

OVENvals <- OVENvals[grep(x=OVENvals$Sample,"NH", invert = TRUE),]

originSites <- getRDS("originSites")
originSites <- sf::st_as_sf(originSites)

EVER <- length(grep(x=OVENvals$Sample,"EVER"))
JAM <- length(grep(x=OVENvals$Sample,"JAM"))

originRelAbund <- matrix(c(EVER,JAM),nrow = 1,byrow = TRUE)
originRelAbund <- prop.table(originRelAbund,1)

op <- sf::st_centroid(originSites)

originPoints <- array(NA,c(EVER+JAM,2), list(NULL, c("x","y")))
originPoints[grep(x = OVENvals$Sample,"JAM"),1] <- sf::st_coordinates(op)[1,1]
originPoints[grep(x = OVENvals$Sample,"JAM"),2] <- sf::st_coordinates(op)[1,2]
originPoints[grep(x = OVENvals$Sample,"EVER"),1] <-sf::st_coordinates(op)[2,1]
originPoints[grep(x = OVENvals$Sample,"EVER"),2] <-sf::st_coordinates(op)[2,2]

originPoints <- sf::st_as_sf(data.frame(originPoints),
                             coords = c("x", "y"),
                             crs = sf::st_crs(originSites))
originDist <- distFromPos(sf::st_coordinates(op))


iso <- isoAssign(isovalues = OVENvals[,2],
                 isoSTD = 12,       # this value is for demonstration only
                 intercept = -10,   # this value is for demonstration only
                 slope = 0.8,       # this value is for demonstration only
                 odds = NULL,
                 restrict2Likely = TRUE,
                 nSamples = 1000,
```

```
                    sppShapefile = OVENdist,
                    assignExtent = c(-179,-60,15,89),
                    element = "Hydrogen",
                    period = "GrowingSeason",#this setting for demonstration only
                    seed = 12345,
                    verbose=1)

  targetSites <- sf::st_as_sf(iso$targetSites)
  targetSites <- sf::st_make_valid(targetSites)
  targetSites <- sf::st_union(targetSites, by_feature = TRUE)

  ovenMC <- estMC(originRelAbund = originRelAbund,
                  targetIntrinsic = iso,
                  originPoints = originPoints,
                  originSites = originSites,
                  originDist = originDist,
                  nSamples = 50, # set very low for example speed
                  verbose = 1,
                  calcCorr = TRUE,
                  alpha = 0.05,
                  approxSigTest = FALSE,
                  isIntrinsic = TRUE,
                  targetSites = targetSites)

  ovenMC
```

`estNMC` *Estimate NMC_XY, another type of migratory connectivity strength*

Description

Resampling of uncertainty for NMC_XY (network migratory connectivity strength between seasons X and Y), network migratory connectivity diversity (X node-specific version of NMC_XY), and NMCa_XY (abundance-weighted network migratory connectivity strength) from estimates of psi (transition probabilities). Psi estimates can come from an estMigConnectivity object, an RMark psi matrix, MCMC samples, or other samples expressed in array form.

Usage

```
estNMC(
  psi,
  originRelAbund = NULL,
  originNames = NULL,
  targetNames = NULL,
  originSites = NULL,
  targetSites = NULL,
  nSamples = 1000,
  row0 = 0,
```

```
  verbose = 0,
  alpha = 0.05,
  returnAllInput = TRUE
)
```

Arguments

psi	Transition probabilities between X origin and Y target sites/nodes. Either an array with dimensions n, X, and Y (with n samples of the transition probability matrix from another model), an 'estPsi' object (result of calling estTransition), or a MARK object with estimates of transition probabilities
originRelAbund	Optional. Relative abundance estimates at X origin sites (nodes). Either a numeric vector of length X that sums to 1, or an mcmc object (such as is produced by modelCountDataJAGS) or matrix with at least nSamples rows. If there are more than X columns, the relevant columns should be labeled "relN[1]" through "relN[X]"
originNames	Optional. Vector of names for the origin sites. Mostly for internal use
targetNames	Optional. Vector of names for the target sites. Mostly for internal use
originSites	If psi is a MARK object, this must be a numeric vector indicating which sites are origin
targetSites	If psi is a MARK object, this must be a numeric vector indicating which sites are target
nSamples	Number of times to resample psi. The purpose is to estimate sampling uncertainty; higher values here will do so with more precision
row0	If originRelAbund is an mcmc object or array, this can be set to 0 (default) or any greater integer to specify where to stop ignoring samples ("burn-in")
verbose	0 (default) to 2. 0 prints no output during run. 1 prints a progress update and summary every 100 samples. 2 prints a progress update and summary every sample
alpha	Level for confidence/credible intervals provided. Default (0.05) gives 95 percent CI
returnAllInput	if TRUE (the default) the output includes all of the inputs. If FALSE, only the inputs currently used by another MigConnectivity function are included in the output to save memory.

Value

estNMC returns a list with the elements:

NMC List containing estimates of network migratory connectivity strength:

- sample nSamples sampled values for NMC_XY. Provided to allow the user to compute own summary statistics.
- mean Mean of NMC$sample. Main estimate of NMC_XY, incorporating parametric uncertainty.
- se Standard error of NMC, estimated from SD of NMC$sample.

- simpleCI Default1 - alpha confidence interval for NMC_XY, estimated as alpha/2 and 1 - alpha/2 quantiles of NMC$sample.
- bcCI Bias-corrected 1 - alpha confidence interval for NMC_XY. May be preferable to NMC$simpleCI when NMC$mean is the best estimate of NMC_XY. NMC$simpleCI is preferred when NMC$median is a better estimator. When NMC$mean==NMC$median, these should be identical. Estimated as the pnorm(2 * z0 + qnorm(alpha / 2)) and pnorm(2 * z0 + qnorm(1 - alpha / 2)) quantiles of NMC$sample, where z0 is the proportion of NMC$sample < NMC$mean.
- hpdCI 1 - alpha credible interval for NMC, estimated using the highest posterior density (HPD) method.
- median Median of NMC_XY, alternate point estimate also including parametric uncertainty.
- point Simple point estimate of NMC_XY, using the point estimate of psi (usually the mean values), not accounting for sampling error.

NMCpop List containing estimates of network migratory connectivity diversity (X node-specific NMC_XY):

- sample Matrix of sampled values for network migratory connectivity diversity. nSamples x [number of origin sites]. Provided to allow the user to compute own summary statistics.
- mean Column means of NMCpop$sample. Main estimate of network migratory connectivity diversity, incorporating parametric uncertainty.
- se Standard errors of network migratory connectivity diversity, estimated from column standard deviations of NMCpop$sample.
- simpleCI Default1 - alpha confidence interval for network migratory connectivity diversity, estimated as alpha/2 and 1 - alpha/2 quantiles of each column of NMCpop$sample.
- bcCI Bias-corrected 1 - alpha confidence intervals for network migratory connectivity diversity. May be preferable to NMCpop$simpleCI when NMCpop$mean are the best estimate of network migratory connectivity diversity. NMCpop$simpleCI is preferred when NMCpop$median is a better estimator. When NMCpop$mean==NMCpop$median, these should be identical. Estimated as the pnorm(2 * z0 + qnorm(alpha / 2)) and pnorm(2 * z0 + qnorm(1 - alpha / 2)) quantiles of NMCpop$sample, where z0 is the proportion of NMCpop$sample < NMCpop$mean.
- hpdCI 1 - alpha credible intervals for network migratory connectivity diversity, estimated using the highest posterior density (HPD) method.
- median Medians of network migratory connectivity diversity, alternate point estimate also including parametric uncertainty.
- point Simple point estimates of network migratory connectivity diversity, using the point estimate of psi (usually the mean values), not accounting for sampling error.

NMCa If parameter originRelAbund is entered, a list containing estimates of abundance-weighted network migratory connectivity strength. This list has the same items as NMC, but possibly different values.

input List containing the inputs to estNMC.

See Also

calcNMC, estTransition, estStrength, estMantel, plot.estMigConnectivity

Examples

```
set.seed(101)
# Uncertainty in detection (RMark estimates)
# Number of resampling iterations for generating confidence intervals
nSamplesCMR <- 100
nSimulationsCMR <- 10

# the second intermediate psi scenario, the "low" level
psiTrue <- samplePsis[["Low"]]
originRelAbundTrue <- rep(0.25, 4)
trueNMC <- calcNMC(psiTrue, originRelAbund = originRelAbundTrue)
trueNMC

# Storage matrix for samples
cmrNMCSample <- matrix(NA, nSamplesCMR, nSimulationsCMR)
summaryCMR <- data.frame(Simulation = 1:nSimulationsCMR, True=trueNMC$NMC,
                         mean=NA, se=NA, lcl=NA, ucl=NA)
# Get 'RMark' psi estimates and estimate MC from each
for (r in 1:nSimulationsCMR) {
  cat("Simulation",r,"of",nSimulationsCMR,"\n")
  # Note: getCMRexample() requires a valid internet connection and that GitHub
  # is accessible
  fm <- getCMRexample(r)
  results <- estNMC(psi = fm,
                    originSites = 5:8, targetSites = c(3,2,1,4),
                    nSamples = nSamplesCMR, verbose = 0)
  cmrNMCSample[ , r] <- results$NMC$sample
  summaryCMR$mean[r] <- results$NMC$mean
  summaryCMR$se[r] <- results$NMC$se
  # Calculate confidence intervals using quantiles of sampled MC
  summaryCMR[r, c('lcl', 'ucl')] <- results$NMC$simpleCI
}

summaryCMR <- transform(summaryCMR, coverage = (True>=lcl & True<=ucl))
summaryCMR
summary(summaryCMR)
biasCMR <- mean(summaryCMR$mean) - trueNMC$NMC
biasCMR
mseCMR <- mean((summaryCMR$mean - trueNMC$NMC)^2)
mseCMR
rmseCMR <- sqrt(mseCMR)
rmseCMR

# Simulation of BBS data to quantify uncertainty in relative abundance
nSamplesAbund <- 700 #1700 are stored
nSimulationsAbund <- 10
#\dontrun{
#  nSamplesAbund <- 1700
#}
# Storage matrix for samples
abundNMCaSample <- matrix(NA, nSamplesAbund, nSimulationsAbund)
summaryAbund <- data.frame(Simulation = 1:nSimulationsAbund,
```

```
                          True = trueNMC$NMCa,
                          mean = NA, se = NA, lcl = NA, ucl = NA)
for (r in 1:nSimulationsAbund) {
  cat("Simulation",r,"of",nSimulationsAbund,"\n")
  row0 <- nrow(abundExamples[[r]]) - nSamplesAbund
  results <- estNMC(originRelAbund = abundExamples[[r]], psi = psiTrue,
                        row0 = row0, nSamples = nSamplesAbund, verbose = 2)
  abundNMCaSample[ , r] <- results$NMCa$sample
  summaryAbund$mean[r] <- results$NMCa$mean
  summaryAbund$se[r] <- results$NMCa$se
  # Calculate confidence intervals using quantiles of sampled MC
  summaryAbund[r, c('lcl', 'ucl')] <- results$NMCa$simpleCI
}

summaryAbund <- transform(summaryAbund, coverage = (True >= lcl & True <= ucl))
summaryAbund
summary(summaryAbund)
biasAbund <- mean(summaryAbund$mean) - trueNMC$NMCa
biasAbund
mseAbund <- mean((summaryAbund$mean - trueNMC$NMCa)^2)
mseAbund
rmseAbund <- sqrt(mseAbund)
rmseAbund

# Ovenbird example with GL and GPS data
data(OVENdata) # Ovenbird

nSamplesGLGPS <- 100 # Number of bootstrap iterations, set low for example

# Estimate transition probabilities
Combined.psi<-estTransition(isGL=OVENdata$isGL, #Light-level geolocator (T/F)
                            isTelemetry = !OVENdata$isGL,
                geoBias = OVENdata$geo.bias, # Light-level GL location bias
                geoVCov = OVENdata$geo.vcov, # Location covariance matrix
                targetSites = OVENdata$targetSites, # Nonbreeding/target sites
                originSites = OVENdata$originSites, # Breeding/origin sites
                originPoints = OVENdata$originPoints, # Capture Locations
                targetPoints = OVENdata$targetPoints, #Device target locations
                verbose = 3,   # output options
                nSamples = nSamplesGLGPS, # This is set low for example
                resampleProjection = sf::st_crs(OVENdata$targetPoints),
                nSim = 1000)

# Can estimate NMC from previous psi estimate
Combo.NMC1 <- estNMC(psi = Combined.psi,
                     nSamples = nSamplesGLGPS)
Combo.NMC1

# Doesn't have to be an estPsi object - can simply be array of psi samples
Combo.MC2 <- estNMC(psi = Combined.psi$psi$sample, # Array of samples
                    originNames = Combined.psi$input$originNames,
                    nSamples = nSamplesGLGPS)
Combo.MC2
```

```
# Can estimate NMC from previous psi estimate and abundance estimate
Combo.NMC3 <- estNMC(psi = Combined.psi,
                     originRelAbund = OVENdata$originRelAbund,
                     nSamples = nSamplesGLGPS)
Combo.NMC3
```

`estStrength` *Estimate MC, migratory connectivity strength*

Description

Resampling of uncertainty for MC (migratory connectivity strength) from estimates of psi (transition probabilities) and/or relative abundance. Psi estimates can come from an estMigConnectivity object, an RMark psi matrix, MCMC samples, or other samples expressed in array form. Abundance estimates for each origin site can be either just point estimates (no uncertainty propagated) or MCMC samples. Other inputs include distances between origin sites, distances between target sites, and sample size used to estimate psi.

Usage

```
estStrength(
  originDist,
  targetDist,
  originRelAbund,
  psi,
  sampleSize = NULL,
  originSites = NULL,
  targetSites = NULL,
  originNames = NULL,
  targetNames = NULL,
  nSamples = 1000,
  row0 = 0,
  verbose = 0,
  alpha = 0.05,
  approxSigTest = FALSE,
  sigConst = 0,
  maintainLegacyOutput = FALSE,
  returnAllInput = TRUE
)
```

Arguments

`originDist`	Distances between the B origin sites. Symmetric B by B matrix
`targetDist`	Distances between the W target sites. Symmetric W by W matrix

`originRelAbund` Relative abundance estimates at B origin sites. Either a numeric vector of length B that sums to 1, or an mcmc object (such as is produced by `modelCountDataJAGS`) or matrix with at least `nSamples` rows. If there are more than B columns, the relevant columns should be labeled "relN[1]" through "relN[B]"

`psi` Transition probabilities between B origin and W target sites. Either a matrix with B rows and W columns where rows sum to 1, an array with dimensions x, B, and W (with x samples of the transition probability matrix from another model), an 'estPsi' object (result of calling estTransition), or a MARK object with estimates of transition probabilities

`sampleSize` Total sample size of animals that psi will be estimated from. Should be the number of animals released in one of the origin sites and observed in one of the target sites (or vice-versa). Optional, but recommended, unless psi is an estPsi object (in which case this function can pull it from there)

`originSites` If `psi` is a MARK object, this must be a numeric vector indicating which sites are origin

`targetSites` If `psi` is a MARK object, this must be a numeric vector indicating which sites are target

`originNames` Optional. Vector of names for the origin sites. Mostly for internal use

`targetNames` Optional. Vector of names for the target sites. Mostly for internal use

`nSamples` Number of times to resample `psi` and/or `originRelAbund`. The purpose is to estimate sampling uncertainty; higher values here will do so with more precision

`row0` If `originRelAbund` is an mcmc object or array, this can be set to 0 (default) or any greater integer to specify where to stop ignoring samples ("burn-in")

`verbose` 0 (default) to 2. 0 prints no output during run. 1 prints a progress update and summary every 100 samples. 2 prints a progress update and summary every sample

`alpha` Level for confidence/credible intervals provided. Default (0.05) gives 95 percent CI

`approxSigTest` Should function compute approximate one-sided significance tests (p-values) for MC from the resampling? Default is FALSE

`sigConst` Value to compare MC to in significance test. Default is 0

`maintainLegacyOutput` version 0.4.0 of `MigConnectivity` updated the structure of the estimates. If you have legacy code that refers to elements within an `estMigConnectivity` object (results of `estMC`), you can set this to TRUE to also keep the old structure. Defaults to FALSE

`returnAllInput` if TRUE (the default) the output includes all of the inputs. If FALSE, only the inputs currently used by another MigConnectivity function are included in the output.

Value

`estStrength` returns a list with the elements:

`MC` List containing estimates of migratory connectivity strength:

- sample nSamples sampled values for MC. Provided to allow the user to compute own summary statistics.
- mean Mean of MC$sample. Main estimate of MC, incorporating parametric uncertainty.
- se Standard error of MC, estimated from SD of MC$sample.
- simpleCI Default1 - alpha confidence interval for MC, estimated as alpha/2 and 1 - alpha/2 quantiles of MC$sample.
- bcCI Bias-corrected 1 - alpha confidence interval for MC. May be preferable to MC$simpleCI when MC$mean is the best estimate of MC. MC$simpleCI is preferred when MC$median is a better estimator. When MC$mean==MC$median, these should be identical. Estimated as the pnorm(2 * z0 + qnorm(alpha / 2)) and pnorm(2 * z0 + qnorm(1 - alpha / 2)) quantiles of MC$sample, where z0 is the proportion of MC$sample < MC$mean.
- hpdCI 1 - alpha credible interval for MC, estimated using the highest posterior density (HPD) method.
- median Median of MC, alternate point estimate also including parametric uncertainty.
- point Simple point estimate of MC, using the point estimates of psi and originRelAbund (usually the mean values), not accounting for sampling error.
- simpleP Approximate p-value for MC, estimated as the proportion of bootstrap iterations where MC < sigConst (or MC > sigConst if pointMC < sigConst). Note that if the proportion is 0, a default value of 0.5 / nSamples is provided, but this is best interpreted as p < 1 / nSamples. NULL when approxSigTest==FALSE.
- bcP Approximate bias-corrected p-value for MC, estimated as pnorm(qnorm(simpleP) - 2 * z0), where z0 is the proportion of sampleMC < meanMC. May be a better approximation of the p-value than simpleP, but many of the same limitations apply. NULL when approxSigTest==FALSE.

input List containing the inputs to estStrength.

See Also

calcMC, estTransition, estMC, estMantel, plot.estMigConnectivity

Examples

```
set.seed(101)
# Uncertainty in detection (RMark estimates) with equal abundances
# Number of resampling iterations for generating confidence intervals
nSamplesCMR <- 100
nSimulationsCMR <- 10
originPos13 <- matrix(c(rep(seq(-99, -81, 2), each = 10),
                        rep(seq(49, 31, -2), 10)), 100, 2)
targetPos13 <- matrix(c(rep(seq(-79, -61, 2), each = 10),
                        rep(seq(9, -9, -2), 10)), 100, 2)
originPosCMR <- rowsum(originPos13, c(rep(1:2, 5, each = 5),
                                      rep(3:4, 5, each = 5))) / 25
originPosCMR
targetPosCMR <- rowsum(targetPos13, c(rep(1:2, 5, each = 5),
                                      rep(3:4, 5, each = 5))) / 25
targetPosCMR

originDist <- distFromPos(originPosCMR, 'ellipsoid')
```

```
targetDist <- distFromPos(targetPosCMR, 'ellipsoid')
originRelAbundTrue <- rep(0.25, 4)
# the second intermediate psi scenario, the "low" level
psiTrue <- samplePsis[["Low"]]
trueMC <- calcMC(originDist, targetDist, originRelAbundTrue, psiTrue)
trueMC

# Storage matrix for samples
cmrMCSample <- matrix(NA, nSamplesCMR, nSimulationsCMR)
summaryCMR <- data.frame(Simulation = 1:nSimulationsCMR, True=trueMC,
                         mean=NA, se=NA, lcl=NA, ucl=NA)
# Get 'RMark' psi estimates and estimate MC from each
for (r in 1:nSimulationsCMR) {
  cat("Simulation",r,"of",nSimulationsCMR,"\n")
  # Note: getCMRexample() requires a valid internet connection and that GitHub
  # is accessible
  fm <- getCMRexample(r)
  results <- estStrength(originRelAbund = originRelAbundTrue, psi = fm,
                  originDist = originDist, targetDist = targetDist,
                  originSites = 5:8, targetSites = c(3,2,1,4),
                  nSamples = nSamplesCMR, verbose = 0,
                  sampleSize = length(grep('[2-5]', fm$data$data$ch)))
  cmrMCSample[ , r] <- results$MC$sample
  summaryCMR$mean[r] <- results$MC$mean
  summaryCMR$se[r] <- results$MC$se
  # Calculate confidence intervals using quantiles of sampled MC
  summaryCMR[r, c('lcl', 'ucl')] <- results$MC$simpleCI
}

summaryCMR <- transform(summaryCMR, coverage = (True>=lcl & True<=ucl))
summaryCMR
summary(summaryCMR)
biasCMR <- mean(summaryCMR$mean) - trueMC
biasCMR
mseCMR <- mean((summaryCMR$mean - trueMC)^2)
mseCMR
rmseCMR <- sqrt(mseCMR)
rmseCMR


# Simulation of BBS data to quantify uncertainty in relative abundance

nSamplesAbund <- 700 #1700 are stored
nSimulationsAbund <- 10
#\dontrun{
#  nSamplesAbund <- 1700
#}
# Storage matrix for samples
abundMCSample <- matrix(NA, nSamplesAbund, nSimulationsAbund)
summaryAbund <- data.frame(Simulation = 1:nSimulationsAbund, True = trueMC,
                           mean = NA, se = NA, lcl = NA, ucl = NA)
for (r in 1:nSimulationsAbund) {
  cat("Simulation",r,"of",nSimulationsAbund,"\n")
```

```
  row0 <- nrow(abundExamples[[r]]) - nSamplesAbund
  results <- estStrength(originRelAbund = abundExamples[[r]], psi = psiTrue,
                  originDist = originDist, targetDist = targetDist,
                  row0 = row0, nSamples = nSamplesAbund, verbose = 1)
  abundMCSample[ , r] <- results$MC$sample
  summaryAbund$mean[r] <- results$MC$mean
  summaryAbund$se[r] <- results$MC$se
  # Calculate confidence intervals using quantiles of sampled MC
  summaryAbund[r, c('lcl', 'ucl')] <- results$MC$simpleCI
}

summaryAbund <- transform(summaryAbund, coverage = (True >= lcl & True <= ucl))
summaryAbund
summary(summaryAbund)
biasAbund <- mean(summaryAbund$mean) - trueMC
biasAbund
mseAbund <- mean((summaryAbund$mean - trueMC)^2)
mseAbund
rmseAbund <- sqrt(mseAbund)
rmseAbund

# Ovenbird example with GL and GPS data
data(OVENdata) # Ovenbird

nSamplesGLGPS <- 100 # Number of bootstrap iterations, set low for example

# Estimate transition probabilities
Combined.psi<-estTransition(isGL=OVENdata$isGL, #Light-level geolocator (T/F)
                            isTelemetry = !OVENdata$isGL,
                geoBias = OVENdata$geo.bias, # Light-level GL location bias
                geoVCov = OVENdata$geo.vcov, # Location covariance matrix
                targetSites = OVENdata$targetSites, # Nonbreeding/target sites
                originSites = OVENdata$originSites, # Breeding/origin sites
                originPoints = OVENdata$originPoints, # Capture Locations
                targetPoints = OVENdata$targetPoints, #Device target locations
                verbose = 3,   # output options
                nSamples = nSamplesGLGPS, # This is set low for example
                resampleProjection = sf::st_crs(OVENdata$targetPoints),
                nSim = 1000)

# Can estimate MC from previous psi estimate
Combo.MC1 <- estStrength(targetDist = OVENdata$targetDist, # Distance matrix
                         originDist = OVENdata$originDist, # Distance matrix
                         targetSites = OVENdata$targetSites, # Target sites
                         originSites = OVENdata$originSites, # Breeding sites
                         psi = Combined.psi,
                         originRelAbund = OVENdata$originRelAbund,
                         nSamples = nSamplesGLGPS,
                         sampleSize = nrow(OVENdata$targetPoints))
Combo.MC1

# Doesn't have to be an estPsi object - can simply be array of psi samples
Combo.MC2 <- estStrength(targetDist = OVENdata$targetDist,
```

```
                        originDist = OVENdata$originDist,
                        targetSites = OVENdata$targetSites,
                        originSites = OVENdata$originSites,
                        psi = Combined.psi$psi$sample, # Array of samples
                        originRelAbund = OVENdata$originRelAbund,
                        nSamples = nSamplesGLGPS,
                        sampleSize = nrow(OVENdata$targetPoints))
  Combo.MC2
```

`estTransition` *Estimate psi (transition probabilities between locations in two phases of the annual cycle)*

Description

Estimation and resampling of uncertainty for psi (transition probabilities between origin sites in one phase of the annual cycle and target sites in another for migratory animals). Data can be from any combination of geolocators (GL), telemetry/GPS, intrinsic markers such as isotopes and genetics, and band/ring reencounter data.

Usage

```
estTransition(
  originSites = NULL,
  targetSites = NULL,
  originPoints = NULL,
  targetPoints = NULL,
  originAssignment = NULL,
  targetAssignment = NULL,
  originNames = NULL,
  targetNames = NULL,
  nSamples = 1000,
  isGL = FALSE,
  isTelemetry = FALSE,
  isRaster = FALSE,
  isProb = FALSE,
  captured = "origin",
  geoBias = NULL,
  geoVCov = NULL,
  geoBiasOrigin = geoBias,
  geoVCovOrigin = geoVCov,
  targetRaster = NULL,
  originRaster = NULL,
  banded = NULL,
  reencountered = NULL,
  verbose = 0,
  alpha = 0.05,
```

```
  resampleProjection = "ESRI:102010",
 nSim = ifelse(any(isRaster & isGL) || any(isRaster & isProb) || any(isGL & isProb),
   5000, ifelse(any(isGL), 1000, ifelse(any(isRaster), 10, 1))),
  maxTries = 300,
  nBurnin = 5000,
  nChains = 3,
  nThin = 1,
  dataOverlapSetting = c("dummy", "none", "named"),
  fixedZero = NULL,
  targetRelAbund = NULL,
  method = c("bootstrap", "MCMC", "m-out-of-n-bootstrap"),
  m = NULL,
  psiPrior = NULL,
  returnAllInput = TRUE
)

estPsi(
  originSites = NULL,
  targetSites = NULL,
  originPoints = NULL,
  targetPoints = NULL,
  originAssignment = NULL,
  targetAssignment = NULL,
  originNames = NULL,
  targetNames = NULL,
  nSamples = 1000,
  isGL = FALSE,
  isTelemetry = FALSE,
  isRaster = FALSE,
  isProb = FALSE,
  captured = "origin",
  geoBias = NULL,
  geoVCov = NULL,
  geoBiasOrigin = geoBias,
  geoVCovOrigin = geoVCov,
  targetRaster = NULL,
  originRaster = NULL,
  banded = NULL,
  reencountered = NULL,
  verbose = 0,
  alpha = 0.05,
  resampleProjection = "ESRI:102010",
 nSim = ifelse(any(isRaster & isGL) || any(isRaster & isProb) || any(isGL & isProb),
   5000, ifelse(any(isGL), 1000, ifelse(any(isRaster), 10, 1))),
  maxTries = 300,
  nBurnin = 5000,
  nChains = 3,
  nThin = 1,
```

```
  dataOverlapSetting = c("dummy", "none", "named"),
  fixedZero = NULL,
  targetRelAbund = NULL,
  method = c("bootstrap", "MCMC", "m-out-of-n-bootstrap"),
  m = NULL,
  psiPrior = NULL,
  returnAllInput = TRUE
)
```

Arguments

originSites	A polygon spatial layer (sf - MULTIPOLYGON) defining the geographic representation of sites in the origin season.
targetSites	A polygon spatial layer (sf - MULTIPOLYGON) defining the geographic representation of sites in the target season.
originPoints	A sf or SpatialPoints object, with number of rows or length being the number of animals tracked. Each point indicates the origin location of an animal (or point estimate of same, for GL animals released on target sites). Note that to simplify input of multiple data types both between and for the same animal, if origin points are provided for any animal, they must be provided for all except banding data (can be dummy values), unless dataOverlapSetting is set to "none".
targetPoints	For GL or telemetry data, a sf or SpatialPoints object, with length or number of rows number of animals tracked. Each point indicates the point estimate location of an animal in the target season. Note that to simplify input of multiple data types both between and for the same animal, if target points are provided for any animal, they must be provided for all except banding data (can be dummy values), unless dataOverlapSetting is set to "none".
originAssignment	Assignment of animals to origin season sites. Either an integer vector with length number of animals tracked or a matrix of probabilities with number of animals tracked rows and number of origin sites columns (and rows summing to 1). The latter only applies to animals released in the target sites where there is uncertainty about their origin site, for example from genetic population estimates from the rubias package. Optional, but some combination of these inputs should be defined. Note that if originAssignment is a probability table, animals with known origin sites can have 1 in that column and 0s in all others. Also note that if method is "MCMC", anything in originAssignment and targetAssignment will be assumed to represent animals tracked via telemetry, with known origin and target sites.
targetAssignment	Assignment of animals to target season sites. Either an integer vector with length number of animals tracked or a matrix of probabilities with number of animals tracked rows and number of target sites columns (and rows summing to 1). The latter only applies to animals released in the origin sites where there is uncertainty about their target site, for example from genetic population estimates from the rubias package. Optional, but some combination of these inputs needs to be

defined. Note that if targetAssignment is a probability table, animals with known target sites can have 1 in that column and 0s in all others.

originNames — Optional, but recommended to keep track. Vector of names for the origin sites. If not provided, the function will either try to get these from another input or provide default names (capital letters).

targetNames — Optional, but recommended to keep track. Vector of names for the target sites. If not provided, the function will either try to get these from another input or provide default names (numbers).

nSamples — Number of post-burn-in MCMC samples to store (method == "MCMC") OR number of bootstrap runs for method == "bootstrap". In the latter case, animals are sampled with replacement for each. For all, the purpose is to estimate sampling uncertainty.

isGL — Indicates whether or which animals were tracked with geolocators. Should be either single TRUE or FALSE value, or vector with length of number of animals tracked, with TRUE or FALSE for each animal in data (except those in banded, which are handled separately). For TRUE animals, the model applies geoBias and geoVCov to targetPoints where captured == "origin" or "neither" and geoBiasOrigin and geoVCovOrigin to originPoints where captured == "target" or "neither". Geolocator data should be entered as originPoints and targetPoints.

isTelemetry — Indicates whether or which animals were tracked with telemetry/GPS (no location uncertainty on either end). Should be either single TRUE or FALSE value, or vector with length of number of animals tracked, with TRUE or FALSE for each animal in data (except those in banded, which are handled separately). Telemetry data can be entered as points or using the targetAssignment and originAssignment arguments.

isRaster — Indicates whether or which animals were tracked with intrinsic markers (e.g., genetics or isotopes), with location uncertainty expressed as a raster of probabilities by grid cells, either in targetRaster or originRaster. Should be either single TRUE or FALSE value, or vector with length of number of animals tracked, with TRUE or FALSE for each animal in data (except those in banded, which are handled separately).

isProb — Indicates whether or which animals were tracked with intrinsic markers (e.g., genetics or isotopes), with location uncertainty expressed as a probability table, either in targetAssignment or originAssignment. Should be either single TRUE or FALSE value, or vector with length of number of animals tracked, with TRUE or FALSE for each animal in data (except those in banded, which are handled separately).

captured — Indicates whether or which animals were captured in the origin sites, the target sites, or neither (another phase of the annual cycle). Location uncertainty will only be applied where the animal was not captured. So this doesn't matter for telemetry data, and is assumed to be "origin" for band return data. Should be either single "origin" (default), "target", or "neither" value, or a character vector with length of number of animals tracked, with "origin", "target", or "neither" for each animal.

geoBias — For GL data, vector of length 2 indicating expected bias in longitude and latitude of targetPoints, in resampleProjection units (default meters).

`geoVCov` For GL data, 2x2 matrix with expected variance/covariance in longitude and latitude of `targetPoints`, in `resampleProjection` units (default meters).

`geoBiasOrigin` For GL data where `captured`!="origin", vector of length 2 indicating expected bias in longitude and latitude of `originPoints`, in `resampleProjection` units (default meters).

`geoVCovOrigin` For GL data where `captured`!="origin", 2x2 matrix with expected variance/covariance in longitude and latitude of `targetPoints`, in `resampleProjection` units (default meters).

`targetRaster` For intrinsic tracking data, the results of `isoAssign` or a similar function of class `intrinsicAssign` or class `RasterBrick`/`RasterStack`, for example from the package `assignR`. In any case, it expresses location uncertainty on target range, through a raster of probabilities by grid cells.

`originRaster` For intrinsic tracking data, the results of `isoAssign` or a similar function of class `intrinsicAssign` or class `RasterBrick`/`RasterStack`, for example from the package `assignR`. In any case, it expresses location uncertainty on origin range, through a raster of probabilities by grid cells.

`banded` For band return data, a vector or matrix of the number of released animals from each origin site (including those never reencountered in a target site). If a matrix, the second dimension is taken as the number of age classes of released animals; the model estimates reencounter probability by age class but assumes transition probabilities are the same. Note that this age model is currently implemented only for `method` set to "MCMC", and only when banding data is analyzed alone (no telemetry data).

`reencountered` For band return data, either a matrix with B rows and W columns or a B x [number of ages] x W array. Number of animals reencountered on each target site (by age class banded as) by origin site they came from.

`verbose` 0 (default) to 3. 0 prints no output during run (except on convergence for `method` set to "MCMC"). 1 prints an update every 100 samples or bootstraps (or a status bar for "MCMC"). 2 prints an update every sample or bootstrap. 3 also prints the number of draws (for tuning `nSim`).

`alpha` Level for confidence/credible intervals provided. Default (0.05) gives 95 percent CI.

`resampleProjection`
Projection when sampling from location uncertainty. Default is Equidistant Conic. The default setting preserves distances around latitude = 0 and longitude = 0. Other projections may work well, depending on the location of sites. Ignored unless data are entered using sites and points and/or rasters.

`nSim` Tuning parameter for GL or intrinsic data. Affects only the speed; 1000 seems to work well with our GL data and 10 for our intrinsic data, but your results may vary. For data combinations, we put the default higher (5000) to allow for more data conflicts. Should be integer > 0. Ignored when `method` is "MCMC".

`maxTries` Maximum number of times to run a single GL/intrinsic bootstrap before exiting with an error. Default is 300; you may want to make a little higher if your `nSim` is low and `nSamples` is high. Set to NULL to never exit. This parameter was added to prevent setups where some sample points never land on target sites from running indefinitely.

nBurnin For method set to "MCMC", estTransition runs a JAGS multinomial non-Markovian transitions model, for which it needs the number of burn-in samples before beginning to store results. Default 5000.

nChains For method set to "MCMC", estTransition runs a JAGS multinomial non-Markovian transitions model, for which it needs the number of MCMC chains (to test for convergence). Default 3.

nThin For method set to "MCMC", estTransition runs a JAGS multinomial non-Markovian transitions model, for which it needs the thinning rate. Default 1.

dataOverlapSetting

When there is more than one type of data, this setting allows the user some flexibility for clarifying which type(s) of data apply to which animals. Setting "dummy" (the default) indicates that there are dummy values within each dataset for the animals that isGL, isTelemetry, etc. don't have that data type (FALSE values). If no animals have a data type, no dummy values are required. If no animals have more than one type of data, the user can simplify processing their data by choosing setting "none" here. In this case, there should be no dummy values, and only the animals with a type of data should be included in that dataset. The third setting ("named") is not yet implemented, but will eventually allow another way to allow animals with more than one type of data with named animals linking records. When there is only one type of data, it is fastest to leave this on the default. Note that banding data entered through banded and reencountered are assumed to have no overlap with other data types, so none of this applies to those.

fixedZero When the user has a priori reasons to believe one or more transition probabilities are zero, they can indicate those here, and the model will keep them fixed at zero. This argument should be a matrix with two columns (for row and column of the transition probability matrix) and number of transitions being fixed to zero rows. For MCMC modeling, substantial evidence that a transition fixed to zero isn't zero may cause an error. For bootstrap modeling, a warning will come up if any bootstrap runs generate the transition fixed to zero, and the function will quit with an error if a very large number of runs do (> 10 * nSamples). Fixing transitions to zero may also slow down the bootstrap model somewhat.

targetRelAbund When some/all data have location error at origin sites (i.e., GL, raster, or probability table data with captured = "target" or "none"), unless the data were collected in proportion to abundance at target sites, simulation work indicates substantial bias in transition probability estimates can result. However, if these data are resampled in proportion to target site abundance, this bias is removed. This argument allows the user to provide an estimate of relative abundance at the target sites. Either a numeric vector of length [number target sites] that sums to 1, or an mcmc object (such as is produced by modelCountDataJAGS) or matrix with at least nSamples rows. If there are more than [number target sites] columns, the relevant columns should be labeled "relN[1]" through "relN[number target sites]".

method This important setting lets the user choose the estimation method used: bootstrap or MCMC (Markov chain Monte Carlo). Bootstrap (the default) now works with any and all types of data, whereas MCMC currently only works with banding and telemetry data (enter telemetry data for MCMC using originAssignment

and targetAssignment, not originPoints and targetPoints). However, MCMC is usually faster (and may be a bit more accurate). The third option, "m-out-of-n-bootstrap", is still under development and should be left alone.

m — We read that the m-out-of-n-bootstrap method may improve the coverage of confidence intervals for parameters on or near a boundary (0 or 1 in this case). So we're testing that out. This still under development and not for the end user. In the m-out-of-n-bootstrap, m is the number of samples taken each time (less than the true sample size, n). If the "m-out-of-n-bootstrap" is chosen under method but this is left blank, currently the default is n/4, rounded up (no idea if that is reasonable).

psiPrior — matrix with same dimensions as psi. Only relevant when method is "MCMC". Each row provides a Dirichlet (https://en.wikipedia.org/wiki/Dirichlet_distribution) prior on the transition probabilities from that origin site. The default (NULL) supplies Dirichlet parameters of all 1s, which is a standard uninformative Dirichlet prior. Setting these to other positive numbers is useful when you think a priori that certain transitions are unlikely, but don't want to rule them out altogether using fixedZero.

returnAllInput — if TRUE (the default) the output includes all of the inputs. If FALSE, only the inputs currently used by another MigConnectivity function are included in the output. Switch this if you're worried about computer memory (and the output will be much slimmer).

Value

estTransition returns a list with the elements:

psi List containing estimates of transition probabilities:

- sample Array of sampled values for psi. nSamples x [number of origin sites] x [number of target sites]. Provided to allow the user to compute own summary statistics.
- mean Main estimate of psi matrix. [number of origin sites] x [number of target sites].
- se Standard error of psi, estimated from SD of psi$sample.
- simpleCI 1 - alpha confidence interval for psi, estimated as alpha/2 and 1 - alpha/2 quantiles of psi$sample.
- bcCI Bias-corrected 1 - alpha confidence interval for psi. May be preferable to simpleCI when mean is the best estimate of psi. simpleCI is preferred when median is a better estimator. When the mean and median are equal, these should be identical. Estimated as the pnorm(2 * z0 + qnorm(alpha / 2)) and pnorm(2 * z0 + qnorm(1 - alpha / 2)) quantiles of sample, where z0 is the proportion of sample < mean.
- hpdCI 1 - alpha credible interval for psi, estimated using the highest posterior density (HPD) method.
- median Median estimate of psi matrix.
- point Simple point estimate of psi matrix, not accounting for sampling error.

r List containing estimates of reencounter probabilities at each target site. NULL except when using direct band/ring reencounter data.

input List containing the inputs to estTransition.

BUGSoutput List containing R2jags output. Only present when using method of "MCMC".

References

Hostetler, J. A., E. B. Cohen, C. M. Bossu, A. L. Scarpignato, K. Ruegg, A. Contina, C. S. Rushing, and M. T. Hallworth. 2025. Challenges and opportunities for data integration to improve estimation of migratory connectivity. Methods in Ecology and Evolution 16: 362-376. doi:10.1111/2041-210X.14467

See Also

estStrength, plot.estMigConnectivity, estMC, estMantel

Examples

```
##############################################################################
# Examples 1 (banding data: first example is based on common tern banding
#   data; the second is made up data to demonstrate data with two ages)
##############################################################################
COTE_banded <- c(10360, 1787, 2495, 336)
COTE_reencountered <- matrix(c(12, 0, 38, 15,
                               111, 7, 6, 2,
                               5, 0, 19, 4,
                               1123, 40, 41, 7),
                             4, 4,
                             dimnames = list(LETTERS[1:4], 1:4))
COTE_psi <- estTransition(originNames = LETTERS[1:4],
                          targetNames = 1:4,
                          banded = COTE_banded,
                          reencountered = COTE_reencountered,
                          verbose = 1,
                          nSamples = 60000, nBurnin = 20000,
                          method = "MCMC")
COTE_psi

COTE_banded2 <- matrix(rep(COTE_banded, 2), 4, 2)
COTE_reencountered2 <- array(c(12, 0, 38, 15, 6, 0, 17, 7,
                               111, 7, 6, 2, 55, 3, 3, 1,
                               5, 0, 19, 4, 2, 0, 10, 2,
                               1123, 40, 41, 7, 660, 20, 20, 3),
                             c(4, 2, 4),
                             dimnames = list(LETTERS[1:4], c("J", "A"), 1:4))
COTE_psi2 <- estTransition(originNames = LETTERS[1:4],
                          targetNames = 1:4,
                          banded = COTE_banded2,
                          reencountered = COTE_reencountered2,
                          verbose = 0,
                          nSamples = 60000, nBurnin = 20000,
                          method = "MCMC")
COTE_psi2

##############################################################################
# Example 2 (geolocator and telemetry ovenbirds captured on origin sites)
##############################################################################
data(OVENdata) # Ovenbird
```

```
nSamplesGLGPS <- 100 # Number of bootstrap iterations

# Estimate transition probabilities; treat all data as geolocator
GL_psi <- estTransition(isGL=TRUE,
                        geoBias = OVENdata$geo.bias,
                        geoVCov = OVENdata$geo.vcov,
                        targetSites = OVENdata$targetSites,
                        originSites = OVENdata$originSites,
                        originPoints = OVENdata$originPoints,
                        targetPoints = OVENdata$targetPoints,
                        verbose = 2,
                        nSamples = nSamplesGLGPS,
                        resampleProjection=sf::st_crs(OVENdata$targetPoints))

# Treat all data as is
Combined.psi <- estTransition(isGL=OVENdata$isGL,
                        isTelemetry = !OVENdata$isGL,
                geoBias = OVENdata$geo.bias, # Light-level GL location bias
                geoVCov = OVENdata$geo.vcov, # Location covariance matrix
                targetSites = OVENdata$targetSites, # Nonbreeding/target sites
                originSites = OVENdata$originSites, # Breeding/origin sites
                originPoints = OVENdata$originPoints, # Capture Locations
                targetPoints = OVENdata$targetPoints, #Device target locations
                verbose = 2,   # output options
                nSamples = nSamplesGLGPS, # This is set low for example
                resampleProjection = sf::st_crs(OVENdata$targetPoints))

print(Combined.psi)

# For treating all data as GPS,
# Move the latitude of birds with locations that fall offshore
int <- sf::st_intersects(OVENdata$targetPoints, OVENdata$targetSites)
any(lengths(int)<1)
plot(OVENdata$targetPoints)
plot(OVENdata$targetSites,add=TRUE)
tp<-sf::st_coordinates(OVENdata$targetPoints)
text(tp[,1], tp[,2], label=c(1:39))

tp[5,2] <- 2450000
tp[10,2]<- 2240496
tp[1,2]<- 2240496
tp[11,2]<- 2026511
tp[15,2]<- 2031268
tp[16,2]<- 2031268

oven_targetPoints<-sf::st_as_sf(as.data.frame(tp),
                                coords = c("X","Y"),
                                crs = sf::st_crs(OVENdata$targetPoints))
inter <- sf::st_intersects(oven_targetPoints, OVENdata$targetSites)
any(lengths(inter)<1)
plot(oven_targetPoints,add=TRUE, col = "green")
plot(oven_targetPoints[lengths(inter)<1,],add=TRUE, col = "darkblue")
```

```
# Treat all data as GPS
GPS_psi <- estTransition(isTelemetry = TRUE,
              targetSites = OVENdata$targetSites, # Non-breeding/target sites
              originSites = OVENdata$originSites, # Breeding/origin sites
              originPoints = OVENdata$originPoints, # Capture Locations
              targetPoints = oven_targetPoints, # Device target locations
              verbose = 2,   # output options
              nSamples = nSamplesGLGPS) # This is set low for example



##############################################################################
# Example 3 (all released origin; some telemetry, some GL, some probability
# tables, some both GL and probability tables; data modified from ovenbird
# example)
##############################################################################
library(VGAM)
nAnimals <- 40
isGL <- c(OVENdata$isGL, FALSE)
isTelemetry <- c(!OVENdata$isGL, FALSE)
isRaster <- rep(FALSE, nAnimals)
isProb <- rep(FALSE, nAnimals)
targetPoints <- rbind(OVENdata$targetPoints, OVENdata$targetPoints[1,])
targetSites <- OVENdata$targetSites
originSites <- OVENdata$originSites
resampleProjection <- sf::st_crs(OVENdata$targetPoints)
targetNames <- OVENdata$targetNames
originNames <- OVENdata$originNames
targetAssignment <- array(0, dim = c(nAnimals, 3),
                          dimnames = list(NULL, targetNames))
assignment0 <- unclass(sf::st_intersects(x = targetPoints, y = targetSites,
                                         sparse = TRUE))
assignment0[sapply(assignment0, function(x) length(x)==0)] <- 0
assignment0 <- array(unlist(assignment0), nAnimals)
for (ani in 1:nAnimals) {
  if (assignment0[ani]>0)
    targetAssignment[ani, assignment0[ani]] <- 1
  else{
    targetAssignment[ani, ] <- rdiric(1, c(15, 1, 1))
    isProb[ani] <- TRUE
  }
}
targetAssignment
isProb
nSamplesTry <- 100 # Number of bootstrap iterations
originPoints <- rbind(OVENdata$originPoints,
                      OVENdata$originPoints[39,])
system.time(psi3 <-
              estTransition(isGL = isGL, isRaster = isRaster,
                            isProb = isProb,
                            isTelemetry = isTelemetry,
                            geoBias = OVENdata$geo.bias,
```

```
                              geoVCov = OVENdata$geo.vcov,
                              targetPoints = targetPoints,
                              targetAssignment = targetAssignment,
                              targetSites = targetSites,
                              resampleProjection = resampleProjection,
                              nSim = 20000, maxTries = 300,
                              originSites = originSites,
                              originPoints = originPoints,
                              captured = "origin",
                              originNames = OVENdata$originNames,
                              targetNames = OVENdata$targetNames,
                              verbose = 3,
                              nSamples = nSamplesTry))
psi3

nNonBreeding <- nrow(OVENdata$targetSites)

plot(psi3, legend = "top",
     main = paste("OVENlike w/", sum(isGL & !isProb), "GL,",
                  sum(!isGL & isProb), "probs,",
                  sum(isGL & isProb), "both, and", sum(isTelemetry), "GPS"))

##############################################################################
# Example 4 (add probability animals released on other end)
##############################################################################
nAnimals <- 45
captured <- rep(c("origin", "target"), c(40, 5))
isGL <- c(OVENdata$isGL, rep(FALSE, 6))
isTelemetry <- c(!OVENdata$isGL, rep(FALSE, 6))
isRaster <- rep(FALSE, nAnimals)
isProb <- rep(FALSE, nAnimals)
targetPoints <- rbind(OVENdata$targetPoints,
                      OVENdata$targetPoints[c(1:3,19,23,31),])
targetAssignment <- array(0, dim = c(nAnimals, 3),
                          dimnames = list(NULL, targetNames))
assignment0 <- unclass(sf::st_intersects(x = targetPoints, y = targetSites,
                                         sparse = TRUE))
assignment0[sapply(assignment0, function(x) length(x)==0)] <- 0
assignment0 <- array(unlist(assignment0), nAnimals)
for (ani in 1:nAnimals) {
  if (assignment0[ani]>0)
    targetAssignment[ani, assignment0[ani]] <- 1
  else{
    targetAssignment[ani, ] <- rdiric(1, c(15, 1, 1))
    isProb[ani] <- TRUE
  }
}
targetAssignment
isProb
originPoints <- rbind(OVENdata$originPoints,
                      OVENdata$originPoints[34:39,])

originPoints <- sf::st_transform(originPoints, crs = resampleProjection)
```

```
originSites <- sf::st_transform(OVENdata$originSites,
                                crs = resampleProjection)

assignment1 <- unclass(sf::st_intersects(x = originPoints, y = originSites,
                                         sparse = TRUE))
assignment1[sapply(assignment1, function(x) length(x)==0)] <- 0
assignment1 <- array(unlist(assignment1), nAnimals)

nOriginSites <- nrow(originSites)

originAssignment <- array(0, dim = c(nAnimals, nOriginSites),
                          dimnames = list(NULL, originNames))
for (ani in 1:40) {
  originAssignment[ani, assignment1[ani]] <- 1
}
for (ani in 41:nAnimals) {
  originAssignment[ani, ] <- rdiric(1, c(1, 1))
  isProb[ani] <- TRUE
}
originAssignment
isProb
system.time(psi4 <-
              estTransition(isGL = isGL, isRaster = isRaster,
                            isProb = isProb,
                            isTelemetry = isTelemetry,
                            geoBias = OVENdata$geo.bias,
                            geoVCov = OVENdata$geo.vcov,
                            targetPoints = targetPoints,
                            targetAssignment = targetAssignment,
                            targetSites = targetSites,
                            resampleProjection = resampleProjection,
                            nSim = 15000, maxTries = 300,
                            originSites = originSites,
                            originAssignment = originAssignment,
                            captured = captured,
                            originNames = OVENdata$originNames,
                            targetNames = OVENdata$targetNames,
                            verbose = 2,
                            nSamples = nSamplesTry,
                            targetRelAbund = c(0.1432, 0.3577, 0.4991)))
psi4

plot(psi4, legend = "top",
     main = paste(sum(isGL & !isProb), "GL,",
                  sum(!isGL & isProb & captured == "origin"), "prob.,",
                  sum(isGL & isProb), "both,",
                  sum(isTelemetry), "GPS (all\ncaptured origin), and",
                  sum(isProb & captured == "target"),
                  "prob. (captured target)"))
MC4 <- estStrength(OVENdata$originDist, OVENdata$targetDist,
                                    OVENdata$originRelAbund, psi4,
                                    sampleSize = nAnimals)
MC4
```

```
##############################################################################
# Example 5 (all raster, from our OVEN example)
##############################################################################
getCSV <- function(filename) {
  tmp <- tempdir()
  url1 <- paste0(
    'https://github.com/SMBC-NZP/MigConnectivity/blob/master/data-raw/',
                filename, '?raw=true')
  temp <- paste(tmp, filename, sep = '/')
  utils::download.file(url1, temp, mode = 'wb')
  csv <- read.csv(temp)
  unlink(temp)
  return(csv)

}

getRDS <- function(speciesDist) {
  tmp <- tempdir()
  extension <- '.rds'
  filename <- paste0(speciesDist, extension)
  url1 <- paste0(
   'https://github.com/SMBC-NZP/MigConnectivity/blob/master/data-raw/Spatial_Layers/',
                filename, '?raw=true')
  temp <- paste(tmp, filename, sep = '/')
  utils::download.file(url1, temp, mode = 'wb')
  shp <- readRDS(temp)
  unlink(temp)
  return(shp)
}
OVENdist <- getRDS("OVENdist")

OVENdist <- sf::st_as_sf(OVENdist)

OVENdist <- sf::st_transform(OVENdist, 4326)

OVENvals <- getCSV("deltaDvalues.csv")

OVENvals <- OVENvals[grep(x=OVENvals$Sample,"NH", invert = TRUE),]

originSites <- getRDS("originSites")
originSites <- sf::st_as_sf(originSites)

EVER <- length(grep(x=OVENvals$Sample,"EVER"))
JAM <- length(grep(x=OVENvals$Sample,"JAM"))

originRelAbund <- matrix(c(EVER,JAM),nrow = 1,byrow = TRUE)
originRelAbund <- prop.table(originRelAbund,1)

op <- sf::st_centroid(originSites)

originPoints <- array(NA,c(EVER+JAM,2), list(NULL, c("x","y")))
originPoints[grep(x = OVENvals$Sample,"JAM"),1] <- sf::st_coordinates(op)[1,1]
```

```
originPoints[grep(x = OVENvals$Sample,"JAM"),2] <- sf::st_coordinates(op)[1,2]
originPoints[grep(x = OVENvals$Sample,"EVER"),1]<-sf::st_coordinates(op)[2,1]
originPoints[grep(x = OVENvals$Sample,"EVER"),2]<-sf::st_coordinates(op)[2,2]

originPoints <- sf::st_as_sf(data.frame(originPoints),
                             coords = c("x", "y"),
                             crs = sf::st_crs(originSites))

iso <- isoAssign(isovalues = OVENvals[,2],
                 isoSTD = 12,       # this value is for demonstration only
                 intercept = -10,   # this value is for demonstration only
                 slope = 0.8,       # this value is for demonstration only
                 odds = NULL,
                 restrict2Likely = FALSE,
                 nSamples = 1000,
                 sppShapefile = terra::vect(OVENdist),
                 assignExtent = c(-179,-60,15,89),
                 element = "Hydrogen",
                 period = "GrowingSeason",#this setting for demonstration only
                 seed = 12345,
                 verbose=1)


nAnimals <- dim(iso$probassign)[3]
isGL <-rep(FALSE, nAnimals); isRaster <- rep(TRUE, nAnimals)
isProb <- rep(FALSE, nAnimals); isTelemetry <- rep(FALSE, nAnimals)
targetSites <- sf::st_as_sf(iso$targetSites)
targetSites <- sf::st_make_valid(targetSites)
targetSites <- sf::st_union(targetSites, by_feature = TRUE)


system.time(psi5 <-
              estTransition(isGL = isGL,
                            isRaster = isRaster,
                            isProb = isProb,
                            isTelemetry = isTelemetry,
                            targetSites = targetSites,
                            resampleProjection = resampleProjection,
                            targetRaster = iso,
                            originSites = originSites,
                            originPoints = originPoints,
                            captured = rep("origin", nAnimals),
                            verbose = 2,
                            nSamples = nSamplesTry))
psi5
```

`getCMRexample` *Grab (from https://github.com/SMBC-NZP/MigConnectivity) example RMark transition probability estimates obtained from simulated data*

Description

Get a dataset containing RMark transition probability estimates from simulated mark-recapture-recovery data from Cohen et al. (2014). These all represent the intermediate scenario for all settings (moderate connectivity, low re-encounter, 100,000 banded in each breeding area). Each estimate can be used in `estMC` function to estimate MC with uncertainty. Requires internet connection.

Usage

```
getCMRexample(number = 1)
```

Arguments

`number`	Integer 1 - 100, which simulation and RMark estimate you want

Value

RMark object

See Also

`estMC`

`getIsoMap` *Get Isoscape map getIsoMap*

Description

The `getIsoMap` function downloads predicted isoscape maps from https://wateriso.utah.edu/waterisotopes/. The function first checks whether the isoscapes are located within the directory `mapDirectory`. If a local copy of the isoscape is found, it's read into the environment. If not, the isoscape is downloaded and imported as a raster.

Usage

```
getIsoMap(
  element = "Hydrogen",
  surface = FALSE,
  period = "Annual",
  mapDirectory = NULL
)
```

Arguments

element	The elemental isotope of interest. Currently the only elements that are implemented are 'Hydrogen' (default) and 'Oxygen'
surface	DEPRECATED function no longer returns surface water values. Default is 'FALSE' which returns the precipitation isotopes ratio.
period	The time period of interest. If 'Annual' (default) returns a raster of mean annual values in precipitation for the element. If 'GrowingSeason' returns growing season values in precipitation for element of interest.
mapDirectory	Directory to save/read isotope map from. Can use relative or absolute addressing. The default value (NULL) downloads to a temporary directory, so we strongly recommend changing this from the default unless you're sure you're not going to need these data more than once.

Value

returns a global RasterLayer (resolution = 0.333'x0.3333') object for the element and period of interest

Examples

```
map <- getIsoMap(element = "Hydrogen", period = "GrowingSeason")
```

---

isoAssign *Generate probabilistic isotope assignments*

Description

The isoAssign function generates origin assignments using stable-hydrogen isotopes in tissue. The function generates a probability surface of origin assignment from a vector of stable-isotope values for each animal/sample of interest. Probabilistic assignments are constructed by first converting observed stable-isotope ratios (isoscape) in either precipitation or surface waters into a 'tissuescape' using a user-provided intercept, slope and standard deviation. See Hobson et. al. (2012).

Usage

```
isoAssign(
  isovalues,
  isoSTD,
  intercept,
  slope,
  odds = 0.67,
  restrict2Likely = TRUE,
  nSamples = NULL,
  sppShapefile = NULL,
  relAbund = NULL,
```

```
  isoWeight = NULL,
  abundWeight = NULL,
  population = NULL,
  assignExtent = c(-179, -60, 15, 89),
  element = "Hydrogen",
  surface = FALSE,
  period = "Annual",
  seed = NULL,
  verbose = 1,
  generateSingleCell = FALSE,
  mapDirectory = NULL
)
```

Arguments

isovalues	vector of tissue isotope values
isoSTD	standard deviation from calibration
intercept	intercept value from calibration
slope	value from calibration
odds	odds ratio to use to set likely and unlikely locations defaults to 0.67
restrict2Likely	if TRUE restricts locations to fall within the 'likely' assignment locations
nSamples	integer specifying how many random samples to draw from a multinomial distribution.
sppShapefile	A polygon spatial layer (sf - MULTIPOLYGON) defining species range. Assignments are restricted to these areas.
relAbund	raster (SpatRast) with relative abundance (must match extent of isotope assignment)
isoWeight	weighting value to apply to isotope assignment
abundWeight	weighting value to apply to relative abundance prior
population	vector identifying location where animal was captured. Same order as isovalues
assignExtent	definition for the extent of the assignment. Can be used in place of sppShapefile to limit assignment. Input should follow c(xmin,xmax,ymin,ymax) in degrees longitude and latitude
element	The elemental isotope of interest. Currently the only elements that are implemented are 'Hydrogen' (default) and 'Oxygen'
surface	DEPRECATED function no longer returns surface water values. Default is 'FALSE' which returns the precipitation isotopes ratio.
period	The time period of interest. If 'Annual' returns a raster of mean annual values in precipitation for the element. If 'GrowingSeason' returns growing season values in precipitation for element of interest
seed	numeric value fed to set.seed for random number generation. Default = NULL
verbose	takes values 0, 1 (default) or 2. 0 prints no output during run. 1 prints a message detailing where in the process the function is. 2 prints the animal currently being sampled.

`generateSingleCell`
: if 'TRUE' generates a single origin location using the posterior assignment distribution - this takes a while to run. If 'FALSE' (default), no coordinates are generated.

`mapDirectory`
: Directory to save/read isotope map from. Can use relative or absolute addressing. The default value (NULL) downloads to a temporary directory, so we strongly recommend changing this from the default unless you're sure you're not going to need these data more than once.

Value

returns an `isoAssign` object containing the following:

`probassign` SpatRast stack of individual probabilistic assignments

`oddsassign` SpatRast stack that includes likely vs unlikely origin for each animal

`popassign` a SpatRast for population level assignment (sum of `oodsassign` if `population = NULL`). If `population` is a vector then returns a raster stack for each unique `population` provided

`probDF` data.frame of individual probability surfaces

`oddsDF` data.frame of likely vs unlikely surfaces

`popDF` data.frame of population level assignment

`SingeCell` array of coordinates (longitude,latitude) for single cell assignment

`targetSites` `sf` - `MULTIPOLYGON` layer representing isotope bands equivalent to `isoSTD`

`RandomSeed` the RNG seed used when generating locations from the multinomial distribution

References

Cohen, E. B., C. S. Rushing, F. R. Moore, M. T. Hallworth, J. A. Hostetler, M. Gutierrez Ramirez, and P. P. Marra. 2019. The strength of migratory connectivity for birds en route to breeding through the Gulf of Mexico. Ecography 42: 658 - 669.

Hobson, K. A., S. L. Van Wilgenburg, L. I. Wassenaar, and K. Larson. 2012. Linking hydrogen isotopes in feathers and precipitation: sources of variance and consequences for assignment to isoscapes. PLoS ONE 7: e35137.

See Also

`weightAssign`

Examples

```
extensions <- c("shp", "shx", "dbf", "sbn", "sbx")
tmp <- tempdir()
for (ext in extensions) {
download.file(paste0(
              "https://raw.githubusercontent.com/SMBC-NZP/MigConnectivity",
                     "/master/data-raw/Spatial_Layers/OVENdist.",
                     ext),
```

```
              destfile = paste0(tmp, "/OVENdist.", ext), mode = "wb")
}
OVENdist <- sf::st_read(paste0(tmp, "/OVENdist.shp"))
OVENdist <- OVENdist[OVENdist$ORIGIN==2,] # only breeding
sf::st_crs(OVENdist) <- sf::st_crs(4326)

download.file(paste0(
  "https://raw.githubusercontent.com/SMBC-NZP/MigConnectivity",
                    "/master/data-raw/deltaDvalues.csv"),
              destfile = paste0(tmp, "/deltaDvalues.csv"))
OVENvals <- read.csv(paste0(tmp, "/deltaDvalues.csv"))


a <- Sys.time()
b <- isoAssign(isovalues = OVENvals[,2],
               isoSTD = 12,
               intercept = -10,
               slope = 0.8,
               odds = NULL,
               restrict2Likely = TRUE,
               nSamples = 1000,
               sppShapefile = OVENdist,
               assignExtent = c(-179,-60,15,89),
               element = "Hydrogen",
               period = "GrowingSeason") # this setting for demonstration only
Sys.time()-a
```

`MigConnectivity` *MigConnectivity: A package for quantifying migratory connectivity pattern and strength for migratory animals*

Description

The MigConnectivity package allows the user to estimate or calculate transition probabilities for migratory animals between any two phases of the annual cycle, using a variety of different data types, with the function `estTransition`. The user can also estimate or calculate the strength of migratory connectivity (MC), a standardized metric to quantify the extent to which populations co-occur between two phases of the annual cycle. MC is independent of data type and accounts for the relative abundance of populations distributed across a seasonal range. The package includes functions to estimate MC (`estStrength`) and the more traditional metric of migratory connectivity strength (Mantel correlation; rM; `estMantel`) incorporating uncertainty from multiple sources of sampling error. Description of the MC metric can be found in Cohen et al. (2018).

Key MigConnectivity Functions

`estTransition`: Estimate psi (transition probabilities between locations in two phases of the annual cycle)

estStrength: Estimate MC, migratory connectivity strength

_PACKAGE

`modelCountDataJAGS` *Estimates population-level relative abundance from count data*

Description

Uses a Bayesian hierarchical model to estimate relative abundance of regional populations from count-based data (e.g., Breeding Bird Survey)

Usage

```
modelCountDataJAGS(count_data, ni = 20000, nt = 5, nb = 5000, nc = 3)
```

Arguments

`count_data`	List containing the following elements: '
	`C` nYears by nRoutes matrix containing the observed number of individuals counted at each route in each year.
	`strat` Vector of length nRoutes indicating the population/region in which each route is located.
	`routesPerStrata` Vector of length 1 or nStrata containing the number of routes (i.e. counts) per population. If length(routesPerStrata) == 1, number of routes is identical for each population.
`ni`	Number of MCMC iterations. Default = 20000.
`nt`	Thinning rate. Default = 5.
`nb`	Number of MCMC iterations to discard as burn-in. Default = 5000.
`nc`	Number of chains. Default = 3.

Value

`modelCountDataJAGS` returns an mcmc object containing posterior samples for each monitored parameter.

References

Cohen, E. B., J. A. Hostetler, M. T. Hallworth, C. S. Rushing, T. S. Sillett, and P. P. Marra. 2018. Quantifying the strength of migratory connectivity. Methods in Ecology and Evolution 9: 513-524. doi:10.1111/2041210X.12916

Link, W. A. and J. R. Sauer. 2002. A hierarchical analysis of population change with application to Cerulean Warblers. Ecology 83: 2832-2840. doi:10.1890/00129658(2002)083[2832:AHAOPC]2.0.CO;2

Examples

```
set.seed(150)

### Set parameters for simulation ----

# Number of populations
nStrata. <- 4
# Number of routes w/i each population (assumed to be balanced)
routePerStrat. <- 30 # reduced from 90 for example speed
# Number of years
nYears. <- 5 # reduced from 10 for example speed
# log(Expected number of birds counted at each route)
alphaStrat. <- 1.95
# standard deviation of normal distribution assumed for route/observer random
# effects
sdRoute. <- 0.6
# standard deviation of normal distribution assumed for year random effects
sdYear. <- 0.18


# Number of simulated datasets to create and model
nsims <- 50 # reduced from 100 for example speed
# Number of MCMC iterations
ni. <- 1000 # reduced from 15000 for example speed
# Number of iterations to thin from posterior (reduced from 5)
nt. <- 1
# Number of iterations to discard as burn-in
nb. <- 500 # reduced from 5000 for example speed
# Number of MCMC chains
nc. <- 1 # reduced from 3 for example speed

### Create empty matrix to store model output ---
sim_in <- vector("list", nsims)
sim_out <- vector("list", nsims)


# Simulation ---

system.time(for(s in 1:nsims){
  cat("Simulation",s,"of",nsims,"\n")

  # Simulate data
  sim_data <- simCountData(nStrata = nStrata., routesPerStrata = routePerStrat.,
                           nYears = nYears., alphaStrat = alphaStrat.,
                           sdRoute = sdRoute., sdYear = sdYear.)
  sim_in[[s]] <- sim_data


  # Estimate population-level abundance
  out_mcmc <- modelCountDataJAGS(count_data = sim_data, ni = ni., nt = nt.,
                                 nb = nb., nc = nc.)
```

```
  # Store model output
  sim_out[[s]] <- out_mcmc
  remove(out_mcmc)

})


### Check that relative abundance is, on average, equal for each population
prop.table(sapply(sim_in, function(x) return(rowsum(colSums(x$C), x$strat))), 2)

rel_names <- paste0('relN[', 1:nStrata., ']')
rel_abund1 <- data.frame(sim=1:nsims,
                         ra1.mean=NA, ra2.mean=NA, ra3.mean=NA, ra4.mean=NA,
                         ra1.low=NA, ra2.low=NA, ra3.low=NA, ra4.low=NA,
                         ra1.high=NA, ra2.high=NA, ra3.high=NA, ra4.high=NA,
                         ra1.cover=0, ra2.cover=0, ra3.cover=0, ra4.cover=0)
for (s in 1:nsims) {
  rel_abund1[s, 2:5] <- summary(sim_out[[s]])$statistics[rel_names, "Mean"]
  rel_abund1[s, 6:9] <- summary(sim_out[[s]])$quantiles[rel_names, 1]
  rel_abund1[s, 10:13] <- summary(sim_out[[s]])$quantiles[rel_names, 5]
}
rel_abund1 <- transform(rel_abund1,
                        ra1.cover = (ra1.low<=0.25 & ra1.high>=0.25),
                        ra2.cover = (ra2.low<=0.25 & ra2.high>=0.25),
                        ra3.cover = (ra3.low<=0.25 & ra3.high>=0.25),
                        ra4.cover = (ra4.low<=0.25 & ra4.high>=0.25))

summary(rel_abund1)
```

`OVENdata` *Ovenbird light-level geolocator and GPS necessary data*

Description

Ovenbird data from Cohen et al. (2018) and Hallworth and Marra (2015).

Usage

```
OVENdata
```

Format

A named list with the necessary data to replicate the analyses found in Cohen et al. (2018) with archival light-level geolocator and GPS data. The data contained in the list are:

- geo.bias: Archival light-level geolocator bias estimates. Location bias estimates in light-level geolocator estimates calculated using birds captured at known locations in Florida, Jamaica and Puerto Rico. Location bias is reported in meters and is a vector of length two with bias

estimates in geolocator locations. Format: A vector of length two with bias estimates in geolocator locations.

- geo.vcov: Covariance estimates in light-level geolocator estimates calculated using birds captured at known locations in Florida, Jamaica, and Puerto Rico. Covariance is reported in meters. Format: A 2x2 matrix of covariance estimates.
- isGL: Archival light-level geolocator or PinPoint-10 GPS tag `logical` vector indicating whether location estimates were obtained with a light-level geolocator (`TRUE`) or PinPoint-10 GPS tag (`FALSE`). Format: `logical` of length 39
- targetPoints: Non-breeding locations for 39 Ovenbirds caught during the breeding season who carried either a light-level geolocator or PinPoint-10 GPS tag. Ovenbirds were captured at Hubbard Brook Experimental Forest, NH and Jug Bay Wetland Sanctuary, MD. These data are used as `originPoints` in the `estMC` function. `coords.x1` and `coords.x2` represent the longitude and latitude of the capture sites, respectively. The data are projected in Lambert Conformal Conic. Format: `SpatialPoints` "+proj=aea +lat_1=20 +lat_2=60 +lat_0=40 +lon_0=-96 +x_0=0 +y_0=0 +ellps=GRS80 +datum=NAD83 +units=m +no_defs +towgs84=0,0,0"
- originPoints: Capture locations for 39 Ovenbirds caught during the breeding season who carried either a light-level geolocator or PinPoint-10 GPS tag. Ovenbirds were captured at Hubbard Brook Experimental Forest, NH and Jug Bay Wetland Sanctuary, MD. These data are used as `originPoints` in the `estMC` function. `coords.x1` and `coords.x2` represent the longitude and latitude of the capture sites, respectively. The data are projected in Lambert Conformal Conic. Format: `SpatialPoints`
- targetSites: Non-breeding distribution target sites used in Cohen et al. (in prep) to estimate MC of Ovenbirds tracked with light-level geolocators and PinPoint-10 GPS tags. There are three non-breeding target sites 1) Florida, United States, 2) Cuba, and 3) Hispaniola (Dominican Republic and Haiti). Format: `SpatialPolygons`
- originSites: Breeding distribution origin sites used in Cohen et al. (in prep) to estimate MC of Ovenbirds tracked with light-level geolocators and PinPoint-10 GPS tags. There are two breeding origin sites, one that encompasses NH and another that encompasses MD capture deployment locations. Format: `SpatialPolygons`
- originRelAbund: A dataset containing relative abundance estimates from BBS data reported in Cohen et al. (in prep). These estimates can be used in `estMC` function as `originRelAbund` in conjunction with archival light-level geolocator and GPS locations. Format: A vector of length two with relative abundance estimates.
- originDist: The pairwise Great Circle Distance between the center of the polygons contained within `originSites`. See "Ovenbird breeding distribution origin sites" or `originSites`. Format: square distance matrix
- targetDist: The pairwise Great Circle Distance between the center of the polygons contained within `targetSites`. See "Ovenbird non-breeding distribution target sites" or `targetSites`. Format: square distance matrix

---

`plot.estMigConnectivity` *Basic plot function for estMigConnectivity objects*

Description

Basic plot function for estMigConnectivity objects

Usage

```
## S3 method for class 'estMigConnectivity'
plot(
  x,
 plot.which = ifelse(inherits(x, "estPsi"), "psi", ifelse(inherits(x, "estMC"), "MC",
  ifelse(inherits(x, "estGamma"), "gamma", ifelse(inherits(x, "estMantel"), "rM",
    "NMCpop")))),
  point = c("mean", "median", "point"),
  range = c("simpleCI", "bcCI", "se"),
  xlab = NULL,
  ylab = plot.which,
  originNames = NULL,
  targetNames = NULL,
  ageNames = NULL,
  col = NULL,
  pch = NULL,
  las = 1,
  gap = 0,
  sfrac = ifelse(range == "se", 0.01, 0),
  legend = FALSE,
  map = FALSE,
  ...
)
```

Arguments

x	an estMigConnectivity object (output of estTransition, estStrength, estMC, or estMantel)
plot.which	which parameter (psi, MC, rM, or r) to graph. Defaults to psi for estMC objects, to rM (Mantel correlation) otherwise
point	points on graph can represent mean, median, or point estimates (not considering error). Defaults to mean, the standard estimate from resampling
range	lines / error bars drawn around points can represent simple quantile-based confidence intervals (simpleCI), bias-corrected quantile- based confidence intervals (bcCI), or +- standard error (se). Defaults to simpleCI
xlab	label for the x-axis. Defaults to "Origin" for psi, otherwise ""
ylab	label for the y-axis. Defaults to the parameter being plotted
originNames	names of the origin sites (for plotting psi). If left NULL, the function attempts to get these from the estimate
targetNames	names of the target sites (for plotting psi or r). If left NULL, the function attempts to get these from the estimate

ageNames	names of the age classes (for plotting r with more than one age). If left NULL, the function uses 1:[number of ages]
col	colors to use for labeling transition probabilities for different target sites. If left NULL, defaults to 1:[number of target sites]
pch	symbols to use for labeling transition probabilities for different target sites. If left NULL, defaults to 21:25, then 0:([number of target sites]-5)
las	style of axis labels (0-3). We set the default at 1 (always horizontal) here, but if you prefer your labels parallel to the axis, set at 0
gap	space left between the center of the error bar and the lines marking the error bar in units of the height (width) of the letter "O". Defaults to 0
sfrac	width of "crossbar" at the end of error bar as a fraction of the x plotting region. Defaults to 0, unless range is set to "se", in which case it defaults to 0.01
legend	leave as FALSE to not print a legend. Otherwise the position of the legend (for psi or r (multi-age) only; one of "bottomright", "bottom", "bottomleft", "left", "topleft", "top", "topright", "right", or "center")
map	placeholder for eventually allowing users to plot psi estimates on a map
...	Additional parameters passed to plotCI

Value

No return value, called to generate plot.

See Also

estTransition, estStrength, estMantel

---

plot.intrinsicAssign *Basic plot function for the different isoAssign outputs*

Description

Generates a basic plot of the isotope assignments. If map = 'population' generates a single map. If map = 'probability' or map = 'odds' generates a map for each individual is generated. User is asked for input before each individual is drawn.

Usage

```
## S3 method for class 'intrinsicAssign'
plot(x, map, ...)
```

Arguments

x	an isoAssign object
map	which isoAssign output to plot either 'probability', 'population' or 'odds'
...	additional arguments passed to plot function

Value

No return value, called to generate plot(s).

See Also

isoAssign

Examples

```
extensions <- c("shp", "shx", "dbf", "sbn", "sbx")
tmp <- tempdir()
for (ext in extensions) {
download.file(paste0(
            "https://raw.githubusercontent.com/SMBC-NZP/MigConnectivity",
                  "/master/data-raw/Spatial_Layers/OVENdist.",
                  ext),
            destfile = paste0(tmp, "/OVENdist.", ext), mode = "wb")
}
OVENdist <- sf::st_read(paste0(tmp, "/OVENdist.shp"))
OVENdist <- OVENdist[OVENdist$ORIGIN==2,] # only breeding
sf::st_crs(OVENdist) <- sf::st_crs(4326)

download.file(paste0(
  "https://raw.githubusercontent.com/SMBC-NZP/MigConnectivity",
                  "/master/data-raw/deltaDvalues.csv"),
            destfile = paste0(tmp, "/deltaDvalues.csv"))
OVENvals <- read.csv(paste0(tmp, "/deltaDvalues.csv"))

b <- isoAssign(isovalues = OVENvals[,2],
               isoSTD = 12,
               intercept = -10,
               slope = 0.8,
               odds = NULL,
               restrict2Likely = TRUE,
               nSamples = 1000,
               sppShapefile = OVENdist,
               assignExtent = c(-179,-60,15,89),
               element = "Hydrogen",
               period = "GrowingSeason") # setting for demonstration only

plot(b, map = "population")
```

---

projections *Map projections*

Description

Map projections used when sampling from geolocator bias/error, for example. The argument `resampleProjection` in `estMC` and `estMantel` need units = m, which is true of all of these except WGS84 (the second). First item is Equidistant Conic, which preserves distances around latitude = 0 and longitude = 0. This is a good general purpose projection, but the ideal projection may depend on the locations of your points. See names in list for suggestions. Other potential projections can be found at https://spatialreference.org/ref/

Usage

```
projections
```

Format

A named list of strings.

`reverseTransition` *Reverse transition probabilities and origin relative abundance*

Description

Reverse transition probabilities (psi; sum to 1 for each origin site) and origin relative abundance (originRelAbund; sum to 1 overall) estimates to calculate or estimate target site to origin site transition probabilities (gamma; sum to 1 for each target site), target site relative abundances (targetRelAbund; sum to 1 overall), and origin/target site combination probabilities (pi; sum to 1 overall). If either psi or originRelAbund is an estimate with sampling uncertainty expressed, this function will propagate that uncertainty to provide true estimates of gamma, targetRelAbund, and pi; otherwise (if both are simple point estimates), it will also provide point estimates.

Usage

```
reverseTransition(
  psi = NULL,
  originRelAbund = NULL,
  pi = NULL,
  originSites = NULL,
  targetSites = NULL,
  originNames = NULL,
  targetNames = NULL,
  nSamples = 1000,
  row0 = 0,
  alpha = 0.05
)

reversePsiRelAbund(
  psi = NULL,
  originRelAbund = NULL,
```

```
  pi = NULL,
  originSites = NULL,
  targetSites = NULL,
  originNames = NULL,
  targetNames = NULL,
  nSamples = 1000,
  row0 = 0,
  alpha = 0.05
)

reverseTransitionRelAbund(
  psi = NULL,
  originRelAbund = NULL,
  pi = NULL,
  originSites = NULL,
  targetSites = NULL,
  originNames = NULL,
  targetNames = NULL,
  nSamples = 1000,
  row0 = 0,
  alpha = 0.05
)

reversePi(
  psi = NULL,
  originRelAbund = NULL,
  pi = NULL,
  originSites = NULL,
  targetSites = NULL,
  originNames = NULL,
  targetNames = NULL,
  nSamples = 1000,
  row0 = 0,
  alpha = 0.05
)
```

Arguments

psi	Transition probabilities between B origin and W target sites. Either a matrix with B rows and W columns where rows sum to 1, an array with dimensions x, B, and W (with x samples of the transition probability matrix from another model), an 'estPsi' object (result of calling estTransition), or a MARK object with estimates of transition probabilities
originRelAbund	Relative abundance estimates at B origin sites. Either a numeric vector of length B that sums to 1 or an mcmc object with at least nSamples rows and columns including 'relN[1]' through 'relN[B]'
pi	Migratory combination (joint) probabilities. Either a matrix with B rows and W columns where all entries sum to 1, an array with dimensions x, B, and W, or an

	'estPi' object (currently only the results of calling this function) Either pi or psi and originRelAbund should be specified.
originSites	If psi is a MARK object, this must be a numeric vector indicating which sites are origin
targetSites	If psi is a MARK object, this must be a numeric vector indicating which sites are target
originNames	Vector of names for the origin sites. If not provided, the function will try to get them from psi
targetNames	Vector of names for the target sites. If not provided, the function will try to get them from psi
nSamples	Number of times to resample psi and/or originRelAbund. The purpose is to estimate sampling uncertainty; higher values here will do so with more precision
row0	If originRelAbund is an mcmc object or array, this can be set to 0 (default) or any greater integer to specify where to stop ignoring samples (additional "burn-in")
alpha	Level for confidence/credible intervals provided. Default (0.05) gives 95 percent CI

Details

Alternatively, can be used to reverse migratory combination (joint) probabilities (pi; sum to 1 overall) to psi, originRelAbund, gamma, and targetRelAbund.

Value

If both psi and originRelAbund are simple point estimates, reversePsiRelAbund returns a list with point estimates of gamma, targetRelAbund, and pi. Otherwise, it returns a list with the elements:

gamma List containing estimates of reverse transition probabilities:

- sample Array of sampled values for gamma. nSamples x [number of target sites] x [number of origin sites]. Provided to allow the user to compute own summary statistics.
- mean Main estimate of gamma matrix. [number of target sites] x [number of origin sites].
- se Standard error of gamma, estimated from SD of gamma$sample.
- simpleCI 1 - alpha confidence interval for gamma, estimated as alpha/2 and 1 - alpha/2 quantiles of gamma$sample.
- bcCI Bias-corrected 1 - alpha confidence interval for gamma. May be preferable to simpleCI when mean is the best estimate of gamma. simpleCI is preferred when median is a better estimator. When the mean and median are equal, these should be identical. Estimated as the pnorm(2 * z0 + qnorm(alpha / 2)) and pnorm(2 * z0 + qnorm(1 - alpha / 2)) quantiles of sample, where z0 is the proportion of sample < mean.
- median Median estimate of gamma matrix.
- point Simple point estimate of gamma matrix, not accounting for sampling error.

targetRelAbund List containing estimates of relative abundance at target sites. Items within are the same as within gamma, except for having one fewer dimension.

pi List containing estimates of origin/target site combination probabilities (sum to 1). Items within are the same as within gamma, except for reversing dimensions (same order as psi).

`input` List containing the inputs to `reversePsiRelAbund`.

If the input is pi instead of psi and originRelAbund, then pi is not an output, but psi and originRelAbund are. Otherwise the same.

Examples

```
## Example 1: sample psis and relative abundances from Cohen et al. (2018)
## (no uncertainty in psi or relative abundance)
for (i in 1:length(samplePsis)) {
 for (j in 1:length(sampleOriginRelN)){
  cat("For psi:\n")
  print(samplePsis[[i]])
  cat("and origin relative abundance:", sampleOriginRelN[[j]], "\n")
  print(reverseTransition(samplePsis[[i]], sampleOriginRelN[[j]]))
 }
}

## Example 2: Common tern banding example (uncertainty in psi, not relative
## abundance)
# Number of MCMC iterations
ni. <- 1000 # reduced from 70000 for example speed
# Number of iterations to thin from posterior
nt. <- 1
# Number of iterations to discard as burn-in
nb. <- 500 # reduced from 20000 for example speed
# Number of MCMC chains
nc. <- 1 # reduced from 3 for example speed
COTE_banded <- c(10360, 1787, 2495, 336)
COTE_reencountered <- matrix(c(12, 0, 38, 15,
                               111, 7, 6, 2,
                               5, 0, 19, 4,
                               1123, 40, 41, 7),
                             4, 4,
                             dimnames = list(LETTERS[1:4], 1:4))
COTE_psi <- estTransition(originNames = LETTERS[1:4],
                          targetNames = 1:4,
                          banded = COTE_banded,
                          reencountered = COTE_reencountered,
                          verbose = 1,
                          nSamples = (ni. - nb.) / nt. * nc., nBurnin = nb.,
                          nThin = nt., nChains = nc.,
                          method = "MCMC")
COTE_psi
COTE_rev <- reverseTransition(COTE_psi, sampleOriginRelN[[1]],
                              nSamples = 2000)
COTE_rev


## Example 3: Uncertainty in both psi and relative abundance
# Number of populations
nOriginSites <- 3; originNames <- LETTERS[1:nOriginSites]
nTargetSites <- 4; targetNames <- 1:nTargetSites
```

```
originRelAbund <- c(1/3, 1/3, 1/3)

psiTrue <- array(0, c(nOriginSites, nTargetSites),
                 list(originNames, targetNames))
psiTrue[1,] <- c(0.22, 0.52, 0.16, 0.10)
psiTrue[2,] <- c(0.41, 0.31, 0.17, 0.11)
psiTrue[3,] <- c(0.10, 0.15, 0.42, 0.33)
rowSums(psiTrue)

rev <- reverseTransition(psiTrue, originRelAbund)

# Simulate abundance data on origin sites
# Number of routes w/i each population (assumed to be balanced)
routePerPop. <- 30 # reduced for example speed
# Number of years
nYears. <- 5 # reduced for example speed
# log(Expected number of birds counted at each route)
alphaPop. <- 1.95
# standard deviation of normal distribution assumed for route/observer random
# effects
sdRoute. <- 0.6
# standard deviation of normal distribution assumed for year random effects
sdYear. <- 0.18
# Number of MCMC iterations
ni. <- 1000 # reduced from 70000 for example speed
# Number of iterations to thin from posterior
nt. <- 1
# Number of iterations to discard as burn-in
nb. <- 500 # reduced from 20000 for example speed
# Number of MCMC chains
nc. <- 1 # reduced from 3 for example speed

sim_data <- simCountData(nStrata = nOriginSites, routesPerStrata = routePerPop.,
                         nYears = nYears., alphaStrat = alphaPop.,
                         sdRoute = sdRoute., sdYear = sdYear.)
# Estimate population-level abundance
out_mcmc <- modelCountDataJAGS(count_data = sim_data, ni = ni., nt = nt.,
                               nb = nb., nc = nc.)

# Simulate movement data
sampleSize <- list(rep(20, nOriginSites), NULL)
captured <- rep("origin", sum(sampleSize[[1]]))
isTelemetry <- rep(TRUE:FALSE, c(sum(sampleSize[[1]]), sum(sampleSize[[2]])))
isProb <- rep(FALSE:TRUE, c(sum(sampleSize[[1]]), sum(sampleSize[[2]])))

# Telemetry data (released origin)
data1 <- simTelemetryData(psi = psiTrue,
                          sampleSize = sampleSize[[1]],
                          captured = "origin")
tt <- data1$targetAssignment
oa <- data1$originAssignment
```

```
# Estimate transition probabilities (psi)
est1 <- estTransition(targetAssignment = tt,
                      originAssignment = oa,
                      originNames = originNames,
                      targetNames = targetNames,
                      nSamples = 500, isGL = FALSE,
                      isTelemetry = isTelemetry,
                      isRaster = FALSE,
                      isProb = isProb,
                      captured = captured,
                      nSim = 10, verbose = 0)
# Reverse estimates
rev1 <- reverseTransition(psi = est1, originRelAbund = out_mcmc)
# Compare estimates of gamma, target relative abundance, and pi with calculation
# from true values
rev
rev1
```

`sampleOriginN` *Example origin site abundances and relative abundances*

Description

`sampleOriginN` is a dataset containing example origin site abundances from 5 scenarios used in Cohen et al. (2018). For the same 5 scenarios, `sampleOriginRelN` contains the relative abundances.

Usage

```
sampleOriginN

sampleOriginRelN
```

Format

Each dataset is a named list with 5 vectors in it. Each vector has 4 elements (for the 4 origin sites). The relative abundance vectors each sum to 1. The 5 scenarios are:

- Base: Equal abundance at each origin site
- B Doub: The second origin site has twice the abundance of the other three sites
- B Half: The second origin site has half the abundance of the other three sites
- D Doub: The last origin site has twice the abundance of the other three sites
- D Half: The last origin site has half the abundance of the other three sites

An object of class `list` of length 5.

`sampleOriginPos` *Example origin and target site positions and distances on a 2-D plane*

Description

`sampleOriginPos` is a dataset containing example origin site positions from 12 scenarios used in Cohen et al. (2018). For the same 12 scenarios, `sampleOriginDist` contains the origin site distances, `sampleTargetPos` contains the target site positions, and `sampleTargetDist` contains the target site distances.

Usage

```
sampleOriginPos

sampleOriginDist

sampleTargetPos

sampleTargetDist
```

Format

Each dataset is a named list with 12 matrices in it, representing 12 scenarios. The position matrices each have 2 columns (x and y position) and 4 rows (for each origin or target site). The distance matrices are symmetrical and 4 x 4. The 12 scenarios are:

- Linear: Both origin and target sites arranged in horizontal linear fashion, with equal distances between each adjacent site
- B Dist BC*2: Linear, but the central origin sites are twice as far from each other as the edge sites are from the adjacent origin sites
- B Dist BC/2: Linear, but the central origin sites are half as far from each other as the edge sites are from the adjacent origin sites
- B Dist CD*2: Linear, but the last two origin sites are twice as far from each other as the other adjacent origin sites
- B Dist CD/2: Linear, but the last two origin sites are half as far from each other as the other adjacent origin sites
- B Grid: Origin sites arranged on a grid, target sites arranged linearly, both with all adjacent sites (excluding diagonals) equidistant
- NB Dist 23*2: Linear, but the central target sites are twice as far from each other as the edge sites are from the adjacent target sites
- NB Dist 23/2: Linear, but the central target sites are half as far from each other as the edge sites are from the adjacent target sites
- NB Dist 34*2: Linear, but the last two target sites are twice as far from each other as the other adjacent target sites

- NB Dist 34/2: Linear, but the last two target sites are half as far from each other as the other adjacent target sites
- NB Grid: Target sites arranged on a grid, origin sites arranged linearly, both with all adjacent sites (excluding diagonals) equidistant
- B/NB Grid: Origin and target sites each arranged on a grid, both with all adjacent sites (excluding diagonals) equidistant

An object of class `list` of length 12.

An object of class `list` of length 12.

An object of class `list` of length 12.

`samplePsis` *Example transition probabilities (psis) between origin and target sites*

Description

A dataset containing example psi matrices used in Cohen et al. (2018).

Usage

```
samplePsis
```

Format

A named list with 8 transition probability matrices in it. The direction is from origin site (rows) to target sites (columns), so each row of each matrix sums to 1. The psi matrices are:

- Full Mix: Full mixing from all origin sites to all target sites
- Avoid One Site: All origin sites have the same transition probabilities, mostly avoiding target site 4
- Full Connectivity: Each origin site transitions to only one target site
- Half Mix: Origin sites A and B mix fully between target sites 1 and 2, but don't move to target sites 3 or 4, while origin sites C and D mix fully between target sites 3 and 4, but don't move to target sites 1 or 2
- Low: Simulation scenario labelled "Moderate Connectivity" in Cohen et al. (2014)
- Medium: Simulation scenario labelled "Strong Connectivity" in Cohen et al. (2014)
- One Site Preference: Three origin sites have full mixing, but origin site D only goes to target site 4
- Negative: Artificial transition probability scenario developed to produce a negative MC value under some circumstances

simCMRData *Simulate capture-mark-reencounter (CMR) migratory movement data*

Description

Simulate capture-mark-reencounter (CMR) migratory movement data

Usage

```
simCMRData(psi, banded, r)
```

Arguments

psi	Transition probabilities between B origin sites and W target sites. B by W matrix
banded	A vector of the number of released animals from each origin site (including those never reencountered in a target site). Length B
r	A vector (length W) of reencounter probabilities at each target site

Value

simCMRData returns a list with the elements:

reencountered B by W matrix with numbers reencountered at each target site, by origin site

migrated B by W matrix with numbers migrated to each target site, by origin site. Assumes survival to arrival is 1

input List containing the inputs to function

Examples

```
originNames <- c("A", "B", "C")
nOriginSites <- length(originNames)
targetNames <- as.character(1:4)
nTargetSites <- length(targetNames)

psiTrue <- matrix(c(0.5, 0.25, 0.15, 0.1,
                    0.15, 0.4, 0.25, 0.2,
                    0.1, 0.15, 0.2, 0.55), nOriginSites, nTargetSites,
                  TRUE, list(originNames, targetNames))
psiTrue
rowSums(psiTrue)
banded <- c(3000, 6000, 12000); names(banded) <- originNames
rTrue <- c(0.5, 0.6, 0.4, 0.7) ; names(rTrue) <- targetNames
nSims <- 10
reencountered <- psiCalc <- psiEstMCMC <- psiEstBoot <- vector("list", nSims)

set.seed(9001)
for (i in 1:nSims) {
  dataCMR <- simCMRData(psiTrue, banded, rTrue)
```

```
  reencountered[[i]] <- dataCMR$reencountered
  psiCalc[[i]] <- calcTransition(banded = banded, reencountered = reencountered[[i]],
                                 originNames = originNames, targetNames = targetNames)

  psiEstMCMC[[i]] <- estTransition(originNames = originNames, targetNames = targetNames,
                              nSamples = 24000, banded = banded,
                              reencountered = reencountered[[i]],
                              method = "MCMC", verbose = 1)

  psiEstBoot[[i]] <- estTransition(originNames = originNames, targetNames = targetNames,
                              nSamples = 400, # this is set low for demonstration
                              banded = banded,
                              reencountered = reencountered[[i]],
                              method = "bootstrap", verbose = 1)
}
psiErrorBoot <- sapply(psiEstBoot, function(x) x$psi$mean - psiTrue, simplify = "array")
psiErrorMCMC <- sapply(psiEstMCMC, function(x) x$psi$mean - psiTrue, simplify = "array")
psiRhat <- sapply(psiEstMCMC, function(x) max(x$BUGSoutput$summary[,"Rhat"]))
psiConvergence <- psiRhat < 1.1
psiErrorMCMC
(psiBiasMCMC <- apply(psiErrorMCMC, 1:2, mean))
(psiBiasBoot <- apply(psiErrorBoot, 1:2, mean))
(psiMAEMCMC <- apply(psiErrorMCMC, 1:2, function(x) mean(abs(x), na.rm = TRUE)))
(psiMAEBoot <- apply(psiErrorBoot, 1:2, function(x) mean(abs(x), na.rm = TRUE)))
library(coda)
psiListsMCMC <- lapply(psiEstMCMC, function(x) as.mcmc.list((x$BUGSoutput)))
for (i in 1:nSims) {
  if (!psiConvergence[i])
    plot(psiListsMCMC[[i]])
}
```

`simCountData` *Simulates Breeding Bird Survey-style count data*

Description

Recently updated (version 0.4.3) to more properly match current BBS models. `modelCountDataJAGS` has not been updated yet

Usage

```
simCountData(
  nStrata,
  routesPerStrata,
  nYears,
  alphaStrat,
  beta = 0,
  eta = 0,
```

```
  sdRoute = 0,
  sdYear = 0,
  sdObs = 0,
  sdCount = 0,
  model = c("S", "Sh", "D", "Dh"),
  obsSurvival = 1,
  fixedyear = round(nYears/2),
  nuCount = 1
)
```

Arguments

nStrata	Number of populations/regions/strata
routesPerStrata	Vector of length 1 or nStrata containing the number of routes (i.e. counts) per stratum. If length(routesPerStrata) == 1, number of routes is identical for each population
nYears	Number of years surveys were conducted
alphaStrat	Vector of length 1 or nStrata containing the log expected number of individuals counted at each route for each population. If length(alphaStrat) == 1, expected counts are identical for each population
beta	Coefficient of linear year effect (default 0)
eta	Coefficient of first time run effect (default 0)
sdRoute	Standard deviation of random route-level variation. Default is 0, and if you're setting sdObs, it's probably best to keep it that way
sdYear	Standard deviation of random year-level variation (default 0)
sdObs	Standard deviation of random observer variation (default 0)
sdCount	Standard deviation of random count-level variation (default 0)
model	One of "S" (default), "Sh", "D", and "Dh". See Link et al. (2020) for descriptions of these models
obsSurvival	Annual probability that the observer that ran a route one year will run it the next. Default 1 (each route has only one observer)
fixedyear	The year within nYears that alphaStrat applies directly to (default halfway through)
nuCount	For the "h" models, parameter for extra variation in counts (default 1)

Value

simCountData returns a list containing:

nStrata Number of populations/regions.

nRoutes Total number of routes.

nYears Number of years.

routesPerStrata Number of routes per population.

year Vector of length nYears with standardized year values.

`strat` Vector of length nRoutes indicating the population/region in which each route is located.

`alphaStrat` log expected count for each populations.

`epsRoute` realized deviation from alphaStrat for each route.

`epsYear` realized deviation from alphaStrat for each year.

`beta` linear year effect.

`sdRoute` standard deviation of random route-level variation.

`sdYear` standard deviation of random year-level variation.

`expectedCount` nRoutes by nYears matrix containing deterministic expected counts.

`C` nRoutes by nYears matrix containing observed counts.

References

Cohen, E. B., J. A. Hostetler, M. T. Hallworth, C. S. Rushing, T. S. Sillett, and P. P. Marra. 2018. Quantifying the strength of migratory connectivity. Methods in Ecology and Evolution 9: 513-524. doi:10.1111/2041210X.12916

Link, W. A., J. R. Sauer, and D. K. Niven. 2020. Model selection for the North American Breeding Bird Survey. Ecological Applications 30: e02137. doi:10.1002/eap.2137

Examples

```
set.seed(150)

### Set parameters for simulation ----

# Number of populations
nStrata. <- 4
# Number of routes w/i each population (assumed to be balanced)
routePerStrat. <- 30 # reduced from 90 for example speed
# Number of years
nYears. <- 5 # reduced from 10 for example speed
# log(Expected number of birds counted at each route)
alphaStrat. <- 1.95
# standard deviation of normal distribution assumed for route/observer random
# effects
sdRoute. <- 0.6
# standard deviation of normal distribution assumed for year random effects
sdYear. <- 0.18


# Number of simulated datasets to create and model
nsims <- 50 # reduced from 100 for example speed
# Number of MCMC iterations
ni. <- 1000 # reduced from 15000 for example speed
# Number of iterations to thin from posterior (reduced from 5)
nt. <- 1
# Number of iterations to discard as burn-in
nb. <- 500 # reduced from 5000 for example speed
# Number of MCMC chains
nc. <- 1 # reduced from 3 for example speed
```

```
### Create empty matrix to store model output ---
sim_in <- vector("list", nsims)
sim_out <- vector("list", nsims)


# Simulation ---

system.time(for(s in 1:nsims){
  cat("Simulation",s,"of",nsims,"\n")

  # Simulate data
  sim_data <- simCountData(nStrata = nStrata., routesPerStrata = routePerStrat.,
                           nYears = nYears., alphaStrat = alphaStrat.,
                           sdRoute = sdRoute., sdYear = sdYear.)
  sim_in[[s]] <- sim_data


  # Estimate population-level abundance
  out_mcmc <- modelCountDataJAGS(count_data = sim_data, ni = ni., nt = nt.,
                                 nb = nb., nc = nc.)

  # Store model output
  sim_out[[s]] <- out_mcmc
  remove(out_mcmc)

})


### Check that relative abundance is, on average, equal for each population
prop.table(sapply(sim_in, function(x) return(rowsum(colSums(x$C), x$strat))), 2)

rel_names <- paste0('relN[', 1:nStrata., ']')
rel_abund1 <- data.frame(sim=1:nsims,
                         ra1.mean=NA, ra2.mean=NA, ra3.mean=NA, ra4.mean=NA,
                         ra1.low=NA, ra2.low=NA, ra3.low=NA, ra4.low=NA,
                         ra1.high=NA, ra2.high=NA, ra3.high=NA, ra4.high=NA,
                         ra1.cover=0, ra2.cover=0, ra3.cover=0, ra4.cover=0)
for (s in 1:nsims) {
  rel_abund1[s, 2:5] <- summary(sim_out[[s]])$statistics[rel_names, "Mean"]
  rel_abund1[s, 6:9] <- summary(sim_out[[s]])$quantiles[rel_names, 1]
  rel_abund1[s, 10:13] <- summary(sim_out[[s]])$quantiles[rel_names, 5]
}
rel_abund1 <- transform(rel_abund1,
                        ra1.cover = (ra1.low<=0.25 & ra1.high>=0.25),
                        ra2.cover = (ra2.low<=0.25 & ra2.high>=0.25),
                        ra3.cover = (ra3.low<=0.25 & ra3.high>=0.25),
                        ra4.cover = (ra4.low<=0.25 & ra4.high>=0.25))

summary(rel_abund1)
```

`simGLData` *Simulate geolocator (GL) migratory movement data*

Description

Simulate geolocator (GL) migratory movement data

Usage

```
simGLData(
  psi,
  originRelAbund = NULL,
  sampleSize,
  originSites = NULL,
  targetSites = NULL,
  geoBias = NULL,
  geoVCov = NULL,
  geoBiasOrigin = geoBias,
  geoVCovOrigin = geoVCov,
  S = 1,
  p = list(1, 1),
  requireEveryOrigin = FALSE
)
```

Arguments

`psi`	Transition probabilities between B origin and W target sites. A matrix with B rows and W columns where rows sum to 1.
`originRelAbund`	Relative abundances at B origin sites. Numeric vector of length B that sums to 1.
`sampleSize`	List of length two. The first element is either a vector of length B with the number of simulated animals to release with geolocators at each of the B origin sites, a single integer with the total number of simulated animals to release with geolocators at origin sites (in which case, the origin sites will be sampled according to the relative abundance), or NULL if all animals are released at target sites. The second element is either a vector of length W with the number of simulated animals to release with geolocators at each of the W target sites, a single integer with the total number of simulated animals to release with geolocators at target sites (in which case, the target sites will be sampled according to their relative abundance), or NULL if all animals are released at origin sites.
`originSites`	A polygon spatial layer (sf - MULTIPOLYGON) defining the geographic representation of sites in the origin season.
`targetSites`	A polygon spatial layer (sf - MULTIPOLYGON) defining the geographic representation of sites in the target season.
`geoBias`	Vector of length 2 indicating expected bias in longitude and latitude of animals captured and released at origin sites, in `targetSites` units.

`geoVCov` 2x2 matrix with expected variance/covariance in longitude and latitude of animals captured and released at origin sites, in `targetSites` units.

`geoBiasOrigin` Vector of length 2 indicating expected bias in longitude and latitude of animals captured and released at target sites, in `originSites` units.

`geoVCovOrigin` 2x2 matrix with expected variance/covariance in longitude and latitude of animals captured and released at target sites, in `originSites` units.

`S` Survival probabilities of released geolocator animals. Either a matrix with B rows and W columns (if survival depends on both origin site and target site), a vector of length W (if survival depends only on target site), or a single number (if survival is the same for all animals). Default 1 (all animals with geolocators survive a year).

`p` Recapture probabilities of released geolocator animals; list of length two. The first element is either a vector of length B (if recapture depends on origin site), or a single number (if recapture is the same for all animals released on origin sites). The second element is either a vector of length W (if recapture depends on target site), or a single number (if recapture is the same for all animals released on target sites). Default list(1, 1) (all animals that survive are recaptured).

`requireEveryOrigin` If TRUE, the function will throw an error if it looks like at least one origin site has no animals released in or migrating to it, or if it can, keep simulating until representation is met. This helps estTransition or estMC not throw an error. Default FALSE.

Value

`simGLData` returns a list with the elements:

`originAssignment` Vector with true origin site of each animal

`targetAssignment` Vector with true target site of each animal

`originPointsTrue` True origin location of each animal, type sf, same projection as originSites

`targetPointsTrue` True target location of each animal, type sf, same projection as targetSites

`originPointsObs` Observed origin location of each animal that survived and was recaptured, type sf, same projection as originSites. Same as originPointsTrue for animals captured at origin sites when S and p==1

`targetPointsObs` Observed target location of each animal that survived and was recaptured, type sf, same projection as targetSites. Same as targetPointsTrue for animals captured at target sites when S and p==1

`lived` 0/1 vector for each animal, indicating which survived

`recaptured` 0/1 vector for each animal, indicating which were recaptured

`input` List containing the inputs to function

simMove *Simulates position of birds by individual, season, year, and month.*

Description

Incorporates migratory connectivity, movement within season, and dispersal between seasons. Does not incorporate births or deaths.

Usage

```
simMove(
  breedingAbund,
  breedingDist,
  winteringDist,
  psi,
  nYears = 10,
  nMonths = 3,
  winMoveRate = 0,
  sumMoveRate = 0,
  winDispRate = 0,
  sumDispRate = 0,
  natalDispRate = 0,
  breedDispRate = 0,
  verbose = 0
)
```

Arguments

breedingAbund	Vector with number of birds to simulate starting at each breeding site.
breedingDist	Distances between the breeding sites. Symmetric matrix.
winteringDist	Distances between the wintering sites. Symmetric matrix.
psi	Transition probabilities between B origin and W target sites. A matrix with B rows and W columns where rows sum to 1.
nYears	Number of years to simulate movement.
nMonths	Number of months per breeding and wintering season.
winMoveRate	Within winter movement rate. Defaults to 0 (no movement).
sumMoveRate	Within summer movement rate. Defaults to 0 (no movement).
winDispRate	Between winter dispersal rate. Defaults to 0 (no dispersal).
sumDispRate	Between summer dispersal rate. Defaults to 0 (no dispersal). Setting this to a value above 0 is equivalent to setting both natal and breeding dispersal to that same value.
natalDispRate	Natal dispersal rate. Controls the movement of animals from their birthplace on their first return to the breeding grounds. Defaults to 0 (return to the birthplace for all).

`breedDispRate` Breeding dispersal rate. Controls the movement of animals between breeding sites on spring migrations after the first. Defaults to 0 (return to the same breeding site each year).

`verbose` If set to a value > 0, informs the user on the passage of years and seasons during the simulation. Defaults to 0 (no output during simulation).

Value

`simMove` returns a list with elements:

`animalLoc` `sum(breedingAbund)` (number of animals) by 2 by `nYears` by `nMonths` array with the simulated locations of each animal in each month of each season (summer or winter) of each year. Values of cells are 1...B (first column) and 1...W (second column) where B is the number of breeding sites and W is the number of wintering sites.

`breedDispMat` B by B matrix of probabilities of breeding dispersal between each pair of 1...B breeding sites. Direction is from row to column, so each row sums to 1.

`natalDispMat` B by B matrix of probabilities of natal dispersal between each pair of 1...B breeding sites. Direction is from row to column, so each row sums to 1.

`sumMoveMat` B by B matrix of probabilities of within season movement between each pair of 1...B breeding sites. Direction is from row to column, so each row sums to 1.

`winDispMat` W by W matrix of probabilities of dispersal between each pair of 1...W nonbreeding sites. Direction is from row to column, so each row sums to 1.

`winMoveMat` W by W matrix of probabilities of within season movement between each pair of 1...W nonbreeding sites. Direction is from row to column, so each row sums to 1.

References

Cohen, E. B., J. A. Hostetler, M. T. Hallworth, C. S. Rushing, T. S. Sillett, and P. P. Marra. 2018. Quantifying the strength of migratory connectivity. Methods in Ecology and Evolution 9: 513-524. doi:10.1111/2041210X.12916

Examples

```
### Dispersal simulation ----
## Utility functions for use in simulations

# Simple approach to estimate psi matrix and MC from simulated (or real) data
# (doesn't include uncertainty).  Only uses one year for computation
calcPsiMC <- function(originDist, targetDist, originRelAbund, locations,
                      years = 1, months = 1, verbose=FALSE) {
  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))
}

## Simulation
originNames <- c("A", "B", "C")
nBreeding <- length(originNames) # Number of sites reduced for example speed
targetNames <- as.character(1:4)
nWintering <- length(targetNames)

psi <- matrix(c(0.5, 0.25, 0.15, 0.1,
                0.15, 0.4, 0.25, 0.2,
                0.1, 0.15, 0.2, 0.55), nBreeding, nWintering,
              TRUE, list(originNames, targetNames))
psi
breedingPos <- matrix(c(seq(-99, -93, 3),
                        rep(40, nBreeding)), nBreeding, 2)
winteringPos <- matrix(c(seq(-88, -82, 2),
                         rep(0, nWintering)), nWintering, 2)
breedingPos
winteringPos

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

# Breeding Abundance
breedingN <- rep(50, nBreeding) # Reduced from 5000 for example speed
breedingRelN <- breedingN/sum(breedingN)


# Baseline strength of migratory connectivity

  MC <- calcMC(breedDist, nonbreedDist, breedingRelN, psi, sum(breedingN))
  round(MC, 4)

# Other basic simulation parameters

## Dispersal simulations---
set.seed(1516)
nYears <- 4 # Reduced from 15 for example speed
nMonths <- 2 # Each season, reduced from 4 for example speed
Drates <- c(0.04, 0.16) # Rates of dispersal, fewer for example speed
```

```
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]]$animalLoc, years = j)
    Disp.df$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
with(subset(Disp.df, Rate == 0.04),
     plot(Year, MC, "l", col = "blue", ylim = c(0, 0.3), lwd = 2))
lines(Disp.df$Year[Disp.df$Rate==0.16], Disp.df$MC[Disp.df$Rate==0.16],
      col = "darkblue", lwd = 2)
legend("bottomleft", legend = Drates, col = c("blue", "darkblue"), lty = 1,
       lwd = 2)
```

---

simProbData *Simulate Dirichlet-based probability table data*

Description

Simulate Dirichlet-based probability table data

Usage

```
simProbData(
  psi,
  originRelAbund,
  sampleSize,
  shapes,
  captured = "target",
  requireEveryOrigin = FALSE
)
```

Arguments

psi	Transition probabilities between B origin sites and W target sites. B by W matrix
originRelAbund	Vector of relative abundances at B origin sites
sampleSize	Either the total number of data points to simulate or a vector with the number at each target or origin site. If only the total is provided, sampling will be done in proportion to abundance
shapes	If captured == "target", a B by B matrix, each row of which is the shape parameters for the Dirichlet distribution of an animal whose true origin assignment is that row's. If captured == "origin", a W by W matrix, each row of which is the shape parameters for the Dirichlet distribution of an animal whose true target assignment is that row's.
captured	Either "target" (the default) or "origin", indicating which side animal data were collected on
requireEveryOrigin	If TRUE, the function will throw an error if it looks like at least one origin site has no animals released in or migrating to it, or if it can, keep simulating until representation is met. This helps estTransition or estMC not throw an error. Default FALSE

Value

simProbData returns a list with the elements:

originAssignment Vector with true origin site of each animal

targetAssignment Vector with true target site of each animal

genProbs Table of assignment site probabilities for each animal

input List containing the inputs to function

`simTelemetryData` *Simulate telemetry/GPS data*

Description

Simulate telemetry/GPS data

Usage

```
simTelemetryData(
  psi,
  sampleSize,
  originRelAbund = NULL,
  originSites = NULL,
  targetSites = NULL,
  captured = "origin",
  S = 1,
  p = 1,
  requireEveryOrigin = FALSE
)
```

Arguments

`psi`	Transition probabilities between B origin and W target sites. A matrix with B rows and W columns where rows sum to 1.
`sampleSize`	If captured is "origin", either a vector of length B with the number of simulated animals to release with geolocators at each of the B origin sites or a single integer with the total number of simulated animals to release with GPS at origin sites (in which case, the origin sites will be sampled according to the relative abundance). If captured is "target", either a vector of length W with the number of simulated animals to release with GPS at each of the W target sites or a single integer with the total number of simulated animals to release at target sites (in which case, the target sites will be sampled according to their relative abundance).
`originRelAbund`	Relative abundances at B origin sites. Numeric vector of length B that sums to 1. Optional unless providing target data and/or sample size of length 1.
`originSites`	A polygon spatial layer (sf - MULTIPOLYGON) defining the geographic representation of sites in the origin season. If left NULL, the simulation won't provide origin points.
`targetSites`	A polygon spatial layer (sf - MULTIPOLYGON) defining the geographic representation of sites in the target season. If left NULL, the simulation won't provide target points.
`captured`	Either "origin" (the default) or "target".
`S`	Survival probabilities of released animals. Probably only relevant for simulating archival tags. Either a matrix with B rows and W columns (if survival depends on both origin site and target site), a vector of length W (if survival depends

only on target site), or a single number (if survival is the same for all animals). Default 1 (all tagged animals survive a year).

`p` Recapture probabilities of released animals. Only relevant for simulating archival tags. Either a vector of length B (if captured on origin and recapture depends on origin site), a vector of length W (if captured on target and recapture depends on target site), or a single number (if recapture is the same for all animals). Default 1 (all animals that survive are recaptured).

`requireEveryOrigin`
If TRUE, the function will throw an error if it looks like at least one origin site has no animals released in or migrating to it, or if it can, keep simulating until representation is met. This helps estTransition not throw an error. Default FALSE.

Value

`simTelemetryData` returns a list with the elements:

`originAssignment` Vector with true origin site of each animal

`targetAssignment` Vector with true target site of each animal

`originPointsTrue` True origin location of each animal, type sf, same projection as originSites

`targetPointsTrue` True target location of each animal, type sf, same projection as targetSites

`originPointsObs` Observed origin location of each animal that survived and was recaptured, type sf, same projection as originSites. Same as originPointsTrue when S and p==1

`targetPointsObs` Observed target location of each animal that survived and was recaptured, type sf, same projection as targetSites. Same as targetPointsTrue when S and p==1

`lived` 0/1 vector for each animal, indicating which survived

`recaptured` 0/1 vector for each animal, indicating which were recaptured

`input` List containing the inputs to function

`weightAssign` *Calculate Weights for Isotope Assignments weightAssign*

Description

The primary purpose of this function is to determine whether weighting likelihood based isotope assignments and prior information, such as relative abundance can improve the model performance compared to the isotope-only model. To do this, we raise the likelihood and prior values to powers from 0.1 to 10 and measure model performance using the assignment error rate and assignment area. Weights < 1 flatten the likelihood/prior distributions (giving relatively more weight to smaller values) and weights > 1 sharpen the distributions (giving relatively less weight to smaller values. The `weightAssign` function generates origin assignments using stable-hydrogen isotopes in tissue. If first generates a probability surface of origin assignment from a vector of stable-isotope values for each animal/sample captured at a known location. Probabilistic assignments are constructed by first converting observed stable-isotope ratios (isoscape) in either precipitation or surface waters into a 'tissuescape' using a user-provided intercept, slope and standard deviation. See Hobson et. al. (2012).

Usage

```
weightAssign(
  knownLocs,
  isovalues,
  isoSTD,
  intercept,
  slope,
  odds = 0.67,
  relAbund,
  weightRange = c(-1, 1),
  sppShapefile = NULL,
  assignExtent = c(-179, -60, 15, 89),
  element = "Hydrogen",
  surface = FALSE,
  period = "Annual",
  verbose = 1,
  mapDirectory = NULL
)
```

Arguments

knownLocs	matrix of capture locations of the same length as isovalues
isovalues	vector of tissue isotope values from known locations
isoSTD	standard deviation from calibration
intercept	intercept value from calibration
slope	value from calibration
odds	odds ratio to use to set likely and unlikely locations defaults to 0.67
relAbund	raster layer of relative abundance that sums to 1.
weightRange	vector of length 2 within minimum and maximum values to weight isotope and relative abundance. Default = c(-1,1)
sppShapefile	A polygon spatial layer (sf - MULTIPOLYGON) defining species range. Assignments are restricted to these areas.
assignExtent	definition for the extent of the assignment. Can be used in place of sppShapefile to limit assignment. Input should follow c(xmin,xmax,ymin,ymax) in degrees longitude and latitude.
element	The elemental isotope of interest. Currently the only elements that are implemented are 'Hydrogen' (default) and 'Oxygen'
surface	DEPRECATED function no longer returns surface water values. Default is 'FALSE' which returns the precipitation isotopes ratio.
period	The time period of interest. If 'Annual' returns a raster of mean annual values in precipitation for the element. If 'GrowingSeason' returns growing season values in precipitation for element of interest.
verbose	takes values 0 or 1 (default). 0 prints no output during run. 1 prints a message detailing where in the process the function is.

`mapDirectory` Directory to save/read isotope map from. Can use relative or absolute addressing. The default value (NULL) downloads to a temporary directory, so we strongly recommend changing this from the default unless you're sure you're not going to need these data more than once.

Value

returns an `weightAssign` object containing the following:

`top` data.frame with the optimal weightings

`frontier` data.frame with values that fall along the Pareto frontier

`performance` data.frame with error rate and assignment area for each weight combination

References

Cohen, E. B., C. S. Rushing, F. R. Moore, M. T. Hallworth, J. A. Hostetler, M. Gutierrez Ramirez, and P. P. Marra. 2019. The strength of migratory connectivity for birds en route to breeding through the Gulf of Mexico. Ecography 42: 658 - 669.

Rushing, C. S., P. P. Marra and C. E. Studds. 2017. Incorporating breeding abundance into spatial assignments on continuous surfaces. Ecology and Evolution 3: 3847-3855. doi:10.1002/ece3.2605

Cohen, E. B., C. S. Rushing, F. R. Moore, M. T. Hallworth, J. A. Hostetler, M. Gutierrez Ramirez, and P. P. Marra. 2019. The strength of migratory connectivity for birds en route to breeding through the Gulf of Mexico. Ecography 42: 658 - 669.

Hobson, K. A., S. L. Van Wilgenburg, L. I. Wassenaar, and K. Larson. 2012. Linking hydrogen isotopes in feathers and precipitation: sources of variance and consequences for assignment to isoscapes. PLoS ONE 7: e35137.

Rushing, C. S., P. P. Marra, and C. E. Studds. 2017. Incorporating breeding abundance into spatial assignments on continuous surfaces. Ecology and Evolution 7: 3847 - 3855.

Examples

```
extensions <- c("shp", "shx", "dbf", "sbn", "sbx")
tmp <- tempdir()
for (ext in extensions) {
download.file(paste0(
              "https://raw.githubusercontent.com/SMBC-NZP/MigConnectivity",
                     "/master/data-raw/Spatial_Layers/OVENdist.",
                     ext),
              destfile = paste0(tmp, "/OVENdist.", ext), mode = "wb")
}
OVENdist <- sf::st_read(paste0(tmp, "/OVENdist.shp"))
OVENdist <- OVENdist[OVENdist$ORIGIN==2,] # only breeding
sf::st_crs(OVENdist) <- sf::st_crs(4326)

download.file(paste0("https://raw.githubusercontent.com/SMBC-NZP/MigConnectivity",
                     "/master/data-raw/deltaDvalues.csv"),
              destfile = paste0(tmp, "/deltaDvalues.csv"))
OVENvals <- read.csv(paste0(tmp, "/deltaDvalues.csv"))
```

```
HBEFbirds <- OVENvals[grep("NH",OVENvals[,1]),]

# Create a spatial object of known capture sites
knownLocs <- sf::st_as_sf(data.frame(Long = rep(-73,nrow(HBEFbirds)),
                                     Lat = rep(43,nrow(HBEFbirds))),
                          coords = c("Long","Lat"),
                          crs = 4326)

#Get OVEN abundance from BBS estimates and read into R #
utils::download.file("https://www.mbr-pwrc.usgs.gov/bbs/ra15/ra06740.zip",
                     destfile = paste0(tmp, "/oven.zip"))
utils::unzip(paste0(tmp, "/oven.zip"), exdir = tmp)
oven_dist <- sf::st_read(paste0(tmp, "/ra06740.shp"))

# Empty raster with the same dimensions as isoscape and Ovenbird distribution

# We do this manually here but the weightedAssign function has been updated
# to ensure the isoscape and abundance rasts have the same extent using
# resampling to match  relAbund to the isoscape.
r <- terra::rast(nrow = 331, ncol = 870,
                 res = c(0.0833333, 0.0833333),
                 xmin = -125.1667, xmax = -52.66672,
                 ymin = 33.49995, ymax = 61.08327,
                 crs = sf::st_crs(4326)$wkt)

# rasterize the polygons from BBS - this is not needed if working with a
# rasterized surface
relativeAbun<-terra::rasterize(terra::vect(sf::st_transform(oven_dist,4326)),
                               r,
                               field = "RASTAT")

relativeAbund <- relativeAbun/terra::global(relativeAbun, sum,
                                            na.rm = TRUE)$sum


BE <- weightAssign(knownLocs = knownLocs,
                   isovalues = HBEFbirds[,2],
                   isoSTD = 12,
                   intercept = -10,
                   slope = 0.8,
                   odds = 0.67,
                   relAbund = relativeAbund,
                   weightRange = c(-1, 1),
                   sppShapefile = OVENdist,
                   assignExtent = c(-179,-60,15,89),
                   element = "Hydrogen",
                   period = "Annual")
```

Index

∗ **datasets**
abundExamples, 3
OVENdata, 60
projections, 64
sampleOriginN, 70
sampleOriginPos, 71
samplePsis, 72

abundExamples, 3

calcMantel, 3
calcMC, *4*, 4, *8*, *24*, *36*
calcNMC, 7, *31*
calcPsi *(*calcTransition*)*, 8
calcStrength *(*calcMC*)*, 4
calcTransition, 8

diffCorr *(*diffMantel*)*, 11
diffMantel, 11
diffMC, 12
diffStrength *(*diffMC*)*, 12
distFromPos, 14

estCorr *(*estMantel*)*, 15
estMantel, *4*, 15, *20*, *24*, *31*, *36*, *46*, *57*, *63*
estMC, *4*, *8*, *19*, 20, *36*, *46*, *53*
estNMC, *8*, 29
estPsi *(*estTransition*)*, 39
estStrength, *20*, *24*, *31*, 34, *46*, *57*, *58*, *63*
estTransition, *8*, *10*, *20*, *24*, *31*, *36*, 39, *57*, *63*

getCMRexample, 52
getIsoMap, 53

isoAssign, *24*, 54

MigConnectivity, 57
modelCountDataJAGS, *18*, *19*, *30*, *35*, *44*, 58

optim, *10*
OVENdata, 60

plot.estMigConnectivity, *24*, *31*, *36*, *46*, 61
plot.intrinsicAssign, 63
plotCI, *63*
projections, *24*, 64

reversePi *(*reverseTransition*)*, 65
reversePsiRelAbund *(*reverseTransition*)*, 65
reverseTransition, 65
reverseTransitionRelAbund *(*reverseTransition*)*, 65

sampleOriginDist *(*sampleOriginPos*)*, 71
sampleOriginN, 70
sampleOriginPos, 71
sampleOriginRelN *(*sampleOriginN*)*, 70
samplePsis, 72
sampleTargetDist *(*sampleOriginPos*)*, 71
sampleTargetPos *(*sampleOriginPos*)*, 71
simCMRData, 73
simCountData, 74
simGLData, 78
simMove, 80
simProbData, 83
simTelemetryData, 85

weightAssign, *56*, 86