'MigConnectivity' package: isotope simulation

Jeffrey A. Hostetler, Michael T. Hallworth

2018-07-03

Load required packages

```
library(raster)
library(MigConnectivity)
library(rgeos)
```

The following is a simulation that tests the how the spatial arrangement of target sites influences MC from stable-hydrogen isotopes. The following simulation is run using data generated within the code but we use the Ovenbird as an example species.

Ovenbird distribution

Read in the Ovenbird distribution and create a species distribution map from the abundance data.

```
# read in raster layer
download.file(paste0("https://raw.githubusercontent.com/SMBC-NZP/MigConnectivity",
                     "/master/data-raw/Spatial_Layers/bbsoven.txt"),
              destfile = "bbsoven.txt")

OVENabund <- raster::raster("bbsoven.txt")

OVENdist <- OVENabund
OVENdist[OVENdist>0]<-1
OVENdist[OVENdist==0]<-NA

OVEN_single_poly <- raster::rasterToPolygons(OVENdist, dissolve = TRUE)
raster::crs(OVEN_single_poly) <- MigConnectivity::projections$WGS84
raster::crs(OVENabund) <- MigConnectivity::projections$WGS84
```

To complete the simulation we need a template to ensure the raster resolution is the same as the assignment raster. To do this, we use the isotope data as our template. We grab the isotope base-map using the getIsoMap function.

```
### get isotope raster for template
rasterTemplate <- MigConnectivity::getIsoMap()
```

Crop template to distribution and mask with distribution

```
rasterTemplate <- raster::mask(raster::crop(rasterTemplate, OVEN_single_poly),
                               OVEN_single_poly)

# rasterize the distribution for relative abundance so that raster
# dimensions and resolution match the isotope layer
relativeAbund <- projectRaster(OVENabund,rasterTemplate)
relativeAbund  <- relativeAbund/cellStats(relativeAbund,sum)
```

Generate target sites

The simulation is focused on the effect of target sites on MC when using stable-hydrogen isotopes. The following code generates various target site layers used in the simulation.

```
# generate target sites
targetRanges <- vector('list',5)
# 3' latitude
targetRanges[[1]] <- rasterToPolygons(raster(xmn = -180,
                                             xmx = -40,
                                             ymn = 25,
                                             ymx = 85,
                                             res = c(140,3)))

# 5' latitude
targetRanges[[2]] <- rasterToPolygons(raster(xmn = -180,
                                                 xmx = -40,
                                                 ymn = 25,
                                                 ymx = 85,
                                                 res = c(140,5)))

# 10' latitude
targetRanges[[3]] <- rasterToPolygons(raster(xmn = -180,
                                             xmx = -40,
                                             ymn = 25,
                                             ymx = 85,
                                             res = c(140,10)))

# 12 isotope units
featherIso <- (0.95*getIsoMap(period = "GrowingSeason")+(-17.57))
iso <- crop(featherIso, extent(c(-180,-40,25,85)))
isocut <- cut(iso, breaks= seq(cellStats(iso,min),cellStats(iso,max),12))
targetRanges[[4]] <- rasterToPolygons(isocut, dissolve = T)


# 12*2 isotope units
isocut <- cut(iso, breaks= seq(cellStats(iso,min),cellStats(iso,max),24))
targetRanges[[5]] <- rasterToPolygons(isocut, dissolve = TRUE)



# Keep only the targetSites that intersect with the OVEN polygon
targetRanges <- lapply(targetRanges,raster::intersect,OVEN_single_poly)
targetRanges <- lapply(targetRanges, sp::spTransform,CRS(projections$WGS84))

for(i in 1:5){
targetRanges[[i]]$Target <- 1:nrow(targetRanges[[i]])
}
```

Generate simulated data

```
#Generate random breeding locations using the 10' target sites
Site1 <- spsample(targetRanges[[3]][1:2,],n=100,type = "random")
Site2 <- spsample(targetRanges[[3]][2:3,],n=100, type = "random")
```

```r
Site3 <- spsample(targetRanges[[3]][2:4,], n= 100, type = "random")

# Capture coordinates
capCoords <- array(NA,c(3,2))
capCoords[1,] <- cbind(-98.17,28.76)
capCoords[2,] <- cbind(-93.70,29.77)
capCoords[3,] <- cbind(-85.000,29.836)

featherIso <- (0.95*getIsoMap(period = "GrowingSeason")+(-17.57))

# Extract simulated data
iso_dat <- data.frame(Site = rep(1:3,each = 100),
                      xcoords = c(Site1@coords[,1],Site2@coords[,1],Site3@coords[,1]),
                      ycoords = c(Site1@coords[,2],Site2@coords[,2],Site3@coords[,2]),
                      targetSite = over(rbind(Site1,Site2,Site3),
                                        targetRanges[[3]])$Target,
                      featherIso = extract(featherIso,rbind(Site1,Site2,Site3)))

iso_dat <- iso_dat[complete.cases(iso_dat),]

# generate transition data from simulation
sim_psi <- table(iso_dat$Site,iso_dat$targetSite)

for(i in 1:nrow(sim_psi)){
  sim_psi[i,]<-sim_psi[i,]/sum(sim_psi[i,])
}

states <- raster::getData("GADM", country = "United States", level = 1)
originSites <- states[(states$NAME_1 %in% c("Texas","Louisiana","Florida")),]
originSites <- sp::spTransform(originSites,
                               sp::CRS(MigConnectivity::projections$WGS84))

originDist <- MigConnectivity::distFromPos(rgeos::gCentroid(originSites,
                                                            byid = TRUE)@coords)
targetDistMC <- MigConnectivity::distFromPos(rgeos::gCentroid(targetRanges[[3]],
                                                              byid = TRUE)@coords)
```

Start of simulations

warning this takes a long time to run.

```r
originRelAbund <- rep(1/3, 3)
nTargetSetups <- 5
nSims <- 1             #SET LOW FOR EXAMPLE
# nSims <- 200
nOriginSites = 3

targetPoints0 <- matrix(NA, 300, 2)

sim.output <- data.frame(targetSetup = rep(NA,nSims),
                         sim = rep(NA,nSims),
                         MC.generated = rep(NA,nSims),
                         MC.realized = rep(NA,nSims),
```

```
                          MC.est = rep(NA,nSims),
                          MC.low = rep(NA,nSims),
                          MC.high = rep(NA,nSims),
                          rM.realized = rep(NA,nSims),
                          rM.est = rep(NA,nSims),
                          rM.low = rep(NA,nSims),
                          rM.high = rep(NA,nSims))


library(doSNOW);library(foreach)
cl <- parallel::makeCluster(5)
doSNOW::registerDoSNOW(cl)


Sys.time()
a <- Sys.time()
output.sims <- foreach(target=1:nTargetSetups,
                       .combine = rbind,
                       .export = ls(.GlobalEnv),
                       .packages = c("raster","sp","MigConnectivity")) %dopar% {
#for(target in 1:nTargetSetups){
  set.seed(9001)
  targetSites <- targetRanges[[target]]
  targetDist <- distFromPos(rgeos::gCentroid(targetSites, byid = TRUE)@coords)
  # Extract simulated data
  iso_dat <- data.frame(Site = rep(1:3,each = 100),
                        xcoords = c(Site1@coords[,1],Site2@coords[,1],Site3@coords[,1]),
                        ycoords = c(Site1@coords[,2],Site2@coords[,2],Site3@coords[,2]),
                        targetSite = over(rbind(Site1,Site2,Site3),targetSites)$Target,
                        featherIso = extract(featherIso,rbind(Site1,Site2,Site3)))

  iso_dat <- iso_dat[complete.cases(iso_dat),]

  # generate transition data from simulation
  sim_psi <- prop.table(table(iso_dat$Site,factor(iso_dat$targetSite,
                                                  1:nrow(targetSites))), 1)

  MC.generated <- calcMC(originDist = originDist,
                         targetDist = targetDist,
                         originRelAbund = originRelAbund,
                         psi = sim_psi)

  for (sim in 1:nSims) {
    cat("Simulation Run", sim, "of", nSims, "for target",target,"at", date(), "\n")
    sim_move <- simMove(rep(100, nOriginSites), originDist, targetDist, sim_psi, 1, 1)
    originAssignment <- sim_move$animalLoc[,1,1,1]
    targetAssignment <- sim_move$animalLoc[,2,1,1]
    for (i in 1:300) {
      targetPoint1 <- spsample(targetSites[targetAssignment[i],],
                               n= 1, type = "random", iter = 25)
      targetPoints0[i,] <- targetPoint1@coords
    }
    targetPoints <- SpatialPoints(targetPoints0, CRS(projections$WGS84))
```

```r
# Extract simulated data
iso_dat <- data.frame(Site = originAssignment,
                      xcoords = targetPoints0[,1],
                      ycoords = targetPoints0[,2],
                      targetSite = over(targetPoints,targetSites)$Target,
                      featherIso = extract(featherIso,targetPoints))

iso_dat <- iso_dat[complete.cases(iso_dat),]

# generate transition data from simulation
sim_psi0 <- table(iso_dat$Site,
                  factor(iso_dat$targetSite,
                         min(targetSites$Target):max(targetSites$Target)))
sim_psi_realized <- prop.table(sim_psi0, 1)

# get points ready for analysis
nSite1 <- table(iso_dat$Site)[1]
nSite2 <- table(iso_dat$Site)[2]
nSite3 <- table(iso_dat$Site)[3]

nTotal <- nSite1+nSite2+nSite3

originCap <- array(NA, c(nTotal,2))

wherecaught <- rep(paste0("Site", 1:3), c(nSite1, nSite2, nSite3))

originCap[which(pmatch(wherecaught,"Site1",dup = TRUE)==1),1] <- capCoords[1,1]
originCap[which(pmatch(wherecaught,"Site1",dup = TRUE)==1),2] <- capCoords[1,2]

originCap[which(pmatch(wherecaught,"Site2",dup = TRUE)==1),1] <- capCoords[2,1]
originCap[which(pmatch(wherecaught,"Site2",dup = TRUE)==1),2] <- capCoords[2,2]

originCap[which(pmatch(wherecaught,"Site3",dup = TRUE)==1),1] <- capCoords[3,1]
originCap[which(pmatch(wherecaught,"Site3",dup = TRUE)==1),2] <- capCoords[3,2]

originPoints <- SpatialPoints(originCap)
crs(originPoints) <- sp::CRS(MigConnectivity::projections$WGS84)

MC.realized <- calcMC(originDist = originDist,
                      targetDist = targetDist,
                      originRelAbund = originRelAbund,
                      psi = sim_psi_realized,
                      sampleSize=nTotal)



originPointDists <- distFromPos(originCap)
targetPointDists <- distFromPos(cbind(iso_dat$xcoords, iso_dat$ycoords))


simAssign <- isoAssign(isovalues = iso_dat$featherIso,
                       isoSTD = 12,
                       intercept = -17.57,
```

```
                            slope = 0.95,
                            odds = 0.67,
                            restrict2Likely = FALSE,
                            nSamples = 500,
                            sppShapefile = OVEN_single_poly,
                            relAbund = relativeAbund,
                            verbose = 0,
                            isoWeight = -0.7,
                            abundWeight = 0,
                            assignExtent = c(-179,-60,15,89),
                            element = "Hydrogen",
                            surface = FALSE,
                            period = "GrowingSeason")

    simEst <- estMC(originRelAbund = originRelAbund,
                    targetDist = targetDist,
                    targetIntrinsic = simAssign,
                    targetSites = targetSites,
                    originPoints = originPoints,
                    originSites = originSites,
                    originDist = originDist,
                    nSamples = 100,
                    verbose = 0,
                    calcCorr = TRUE,
                    alpha = 0.05,
                    approxSigTest = F,
                    sigConst = 0,
                    isIntrinsic = TRUE,
                    nSim = 5)

    sim.output$targetSetup[sim] <- target
    sim.output$sim[sim]<-sim
    sim.output$MC.generated[sim] <- MC.generated
    sim.output$MC.realized[sim] <- MC.realized
    sim.output$MC.est[sim] <- simEst$meanMC
    sim.output$MC.low[sim] <- simEst$bcCI[1]
    sim.output$MC.high[sim] <- simEst$bcCI[2]
    sim.output$rM.realized[sim] <- ncf::mantel.test(originPointDists,
                                                    targetPointDists,
                                                    resamp=0,
                                                    quiet = TRUE)$correlation
    sim.output$rM.est[sim] <- simEst$meanCorr
    sim.output$rM.low[sim] <- simEst$bcCICorr[1]
    sim.output$rM.high[sim] <- simEst$bcCICorr[2]

  }
  return(sim.output)
}
Sys.time()-a

stopCluster(cl)
```

Summarize the output

```r
sim.output <- transform(output.sims,
                        MC.generated.cover = as.integer((MC.low <= MC.generated &
                                                           MC.high >= MC.generated)),
                        MC.realized.cover = as.integer((MC.low <= MC.realized &
                                                          MC.high >= MC.realized)),
                        MC.generated.error = MC.est - MC.generated,
                        MC.realized.error = MC.est - MC.realized,
                        rM.cover = as.integer((rM.low <= rM.realized &
                                                 rM.high >= rM.realized)),
                        rM.error = rM.est - rM.realized)

summary(sim.output)
# Examine results
aggregate(MC.generated.error ~ targetSetup, sim.output, mean)
aggregate(MC.generated.error ~ targetSetup, sim.output, function (x) mean(abs(x)))
aggregate(MC.generated.cover ~ targetSetup, sim.output, mean)
aggregate(MC.realized.error ~ targetSetup, sim.output, mean)
aggregate(MC.realized.error ~ targetSetup, sim.output, function (x) mean(abs(x)))
aggregate(MC.realized.cover ~ targetSetup, sim.output, mean)
aggregate(I(MC.realized - MC.generated) ~ targetSetup, sim.output, mean)
aggregate(rM.error ~ targetSetup, sim.output, mean)
aggregate(rM.error ~ targetSetup, sim.output, function (x) mean(abs(x)))
aggregate(rM.cover ~ targetSetup, sim.output, mean)
aggregate(MC.est ~ targetSetup, sim.output, mean)
```