SpatIS tutorial: Spatial Individual Specialization Index

Appendix 1. Kerches-Rogeri, P.; Niebuhr, B. B. S.; Muylaert, R. L., Mello, M. A. R. Individual specialization in the space use of frugivorous bats. In prep.

Introduction

Individuals are naturally different and heterogeneous within populations. These differences reflect in their physiology, morphology, as well as in their behavior and preferences. To investigate the causes and consequences of interindividual variation, Bolnick et al. (2003) have related that with niche theory and coined the term "individual specialization". According to this concept, a specialist individual would be one whose niche is much narrower than the niche of the population it is part of. In turn, to be considered generalist a population can follow different scenarios: to be composed by generalist individuals, by specialists individuals with distinct niches, or even by a gradient between this two extremes.

In this context, the niche is generally represented by the type and amount of food items or other resources that are consumed by individuals. As resources are heterogeneously distributed in space in different habitats, it is expected that individual specialization leaves its shadow over space, or even that different environments or space itself may be viewed as resources, so that individuals may be more or less specialists regarding their movement patterns, habitat selection, and use of space.

Here we describe how to calculate the Spatial Individual Specialization index (SpatIS)

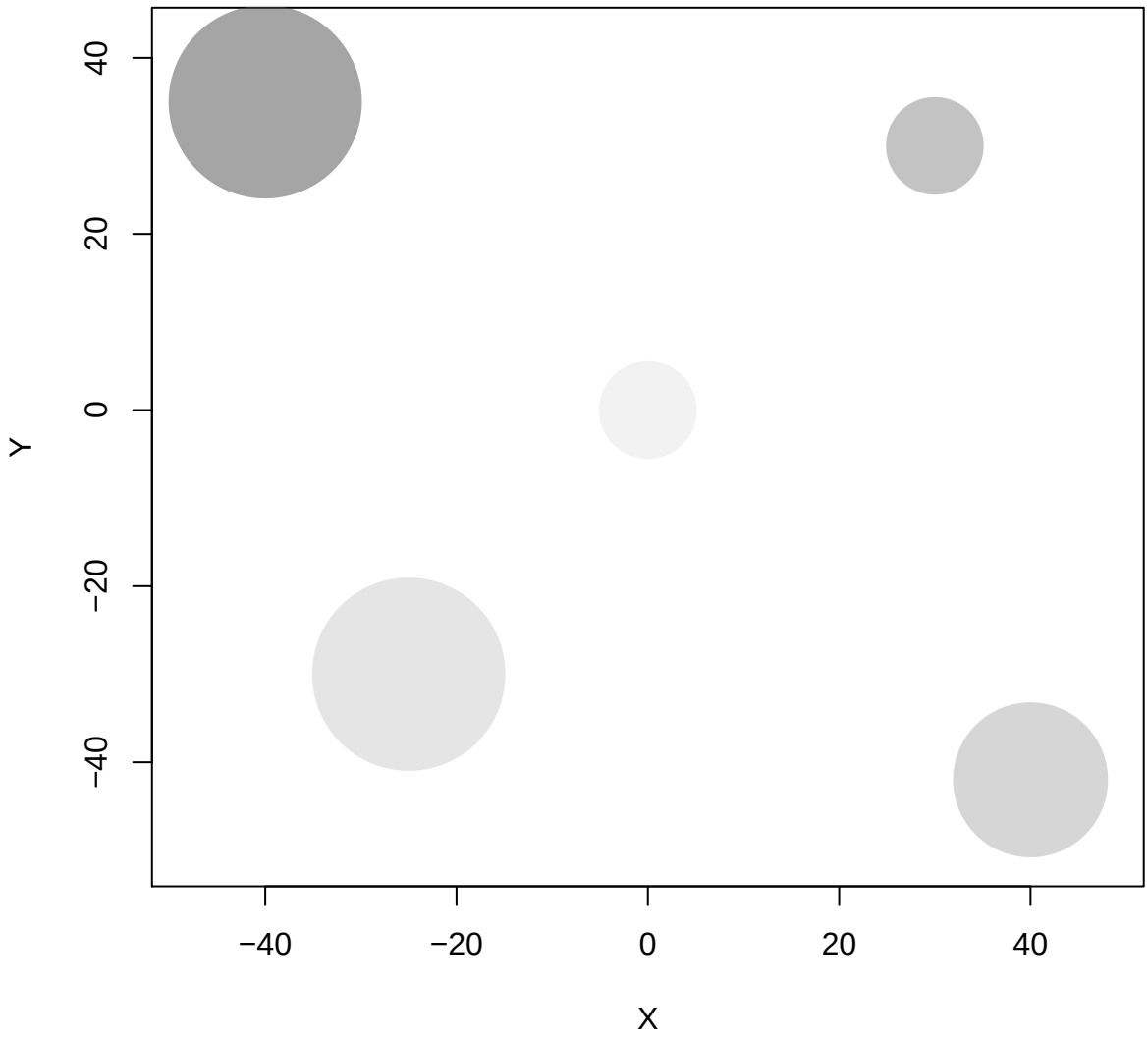
Here we describe the procedures to calculate the Spatial Individual Specialization index (*SpatIS*) using the R function `SpatIS`, based on movement data of individuals of the same population. We also use the function `SpatIS_randomize` to check if the SpatIS for a population is significant, i.e., different from what it would be expected at random, if (at least some of) individuals were not specialists in their use of space. To that end, first we simulate a virtual landscape with some different resources located heterogeously in space. Then we simulate five individuals that present preferences for different types of resources, which is reflected in their movement patterns. Finally we calculate *SpatIS* and run `SpatIS_randomize` for this simulated population.

Simulating space and individuals

To represent the resources spread in space, we are going to create five resource items (e.g. trees) located in a bidimensional landscape at locations (-40,35), (30,30), (40,-42), (-25,-30), and (0,0), with different sizes (or radii, in meters: 10, 5, 8, 10, 5). To draw that, we use the function `draw_circle` from plotrix package.

```
# Load library
library(plotrix)

# Location of resources
x.resources <- c(-40,30,40,-25,0)
y.resources <- c(35,30,-42,-30,0)
radius <- c(10, 5, 8, 10, 5)

# Draw landscape with prefered resources
cols <- grey.colors(length(x.resources), alpha = 0.5) # colors for each resource
# draw a landscape that encompasses the position of all resources
matplot(x.resources, y.resources, type="n",
        xlim=c(1.2*min(x.resources),1.2*max(x.resources)),
        ylim=c(1.2*min(y.resources),1.2*max(y.resources)),
        xlab = "X", ylab = "Y")
```

```
for(i in 1:length(x.resources)) {
  draw.circle(x.resources[i], y.resources[i], radius = radius[i], border = cols[i],
              col = cols[i]) # draw resources
}
```

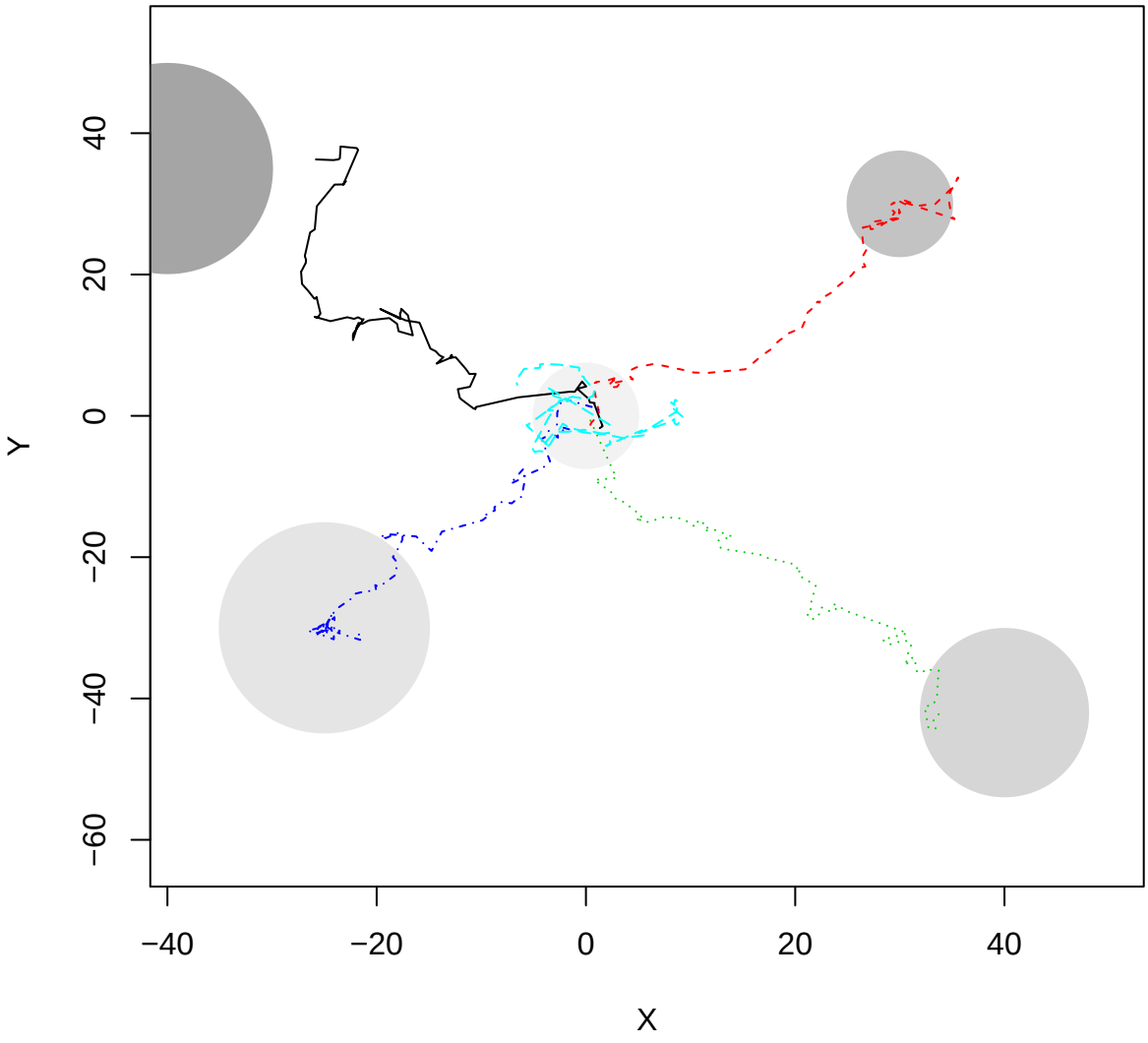


To generate individual trajectories we are going to simulate 100-step paths of 5 individuals that follow a biased random walk, i.e., a random walk with attraction from a certain point in space (here defined by the location of resource items). Each individual starts near the origin (0,0) and will move biased to a different resource item, with greater or lower intensity, which represents their preference for distinct resources at different degrees. This is only one possible mechanism that may generate individuals to present different movement and space use patterns, but it was used here to generate location points and exemplify the application of *SpatIS*.

Biased random walk code was adapted from Prof. Juan M. Morales. The step length of walks is difined by a Weibull distribution (with shape and scale parameters) and turning angles are drawn from a wrapped Cauchy distribution (whit μ and ρ parameters). In turn, the prefered direction of travel μ is a function of β, the coeffient of attraction or bias, which controls how strongly individuals' movements are biased towards a certain resource. Below we simulate these 5 tracks.

```r
# Load library circular
library(circular)

# Seed for random number generation
set.seed(122)

# Random walk parameters
# Coefficient of attraction or bias - positive values correspond to attraction,
#   negative values correspond to avoidance
beta <- c(1.5, 2, 1.8, 2.2, 0.1)
rho <- 0.6 # Concentration parameter around the bias absolute angle
scale <- 1 # Scale of the Weibull distribution for step length
shape <- 1 # Shape of the Weibull distribution for step length

# Number of individuals
ntracks <- 5
# Number of steps per trajectory/individual
nsteps <- 100

# Matrices of x and y locations - initialized with NA
X <- matrix(NA, nsteps, ntracks)
Y <- matrix(NA, nsteps, ntracks)

# Coordinates of the point of attraction/repulsion for each individual
# These coordinated correspond to the 5 different resources created in the landscape
xh <- x.resources
yh <- y.resources

# Simulating tracks
for(i in 1:ntracks){
  x <- numeric(nsteps)
  y <- numeric(nsteps)
  h <- numeric(nsteps)
  steps <- numeric(nsteps)

  # Initial positions of all individuals around the point (0,0)
  h[1] <- runif(1,1,2*pi)
  x[1] <- rnorm(1,0,1)
  y[1] <- rnorm(1,0,1)

  # Simulating following positions
  for(t in 2:nsteps){
    adj <- xh[i] - x[t-1]
    op  <- yh[i] - y[t-1]
    r   <- sqrt(adj^2 + op^2)
    ya <- sin(h[t-1]) + beta[i]*(op/r)
    xa <- cos(h[t-1]) + beta[i]*(adj/r)
    m_t <- atan2(ya,xa)
    h[t] <- rwrappedcauchy(1,mu=circular(m_t),rho=rho)
    steps[t-1] <- rweibull(1,scale=scale, shape=shape)
    x[t] <- x[t-1] + cos(h[t])*steps[t-1]
    y[t] <- y[t-1] + sin(h[t])*steps[t-1]
  }
```

```
  X[,i] <- x
  Y[,i] <- y
}

# Draw landscape with prefered resources
matplot(X,Y, type="n", xlim=c(1.4*min(X),1.4*max(X)), ylim=c(1.4*min(Y),1.4*max(Y)))
for(i in 1:length(x.resources)) {
  draw.circle(x.resources[i], y.resources[i], radius = radius[i], border = cols[i],
              col = cols[i])
}

# Plot individual trajectories
matplot(X, Y, type="l", pch=16, col=1:ntracks, asp=1,
        xlim=c(min(X),max(X)), ylim=c(min(Y),max(Y)), add = T)
```



Calculating SpatIS

To calculate *SpatIS*, first we have to load the `SpatIS` function. To do that we may use the `source` function, loading it directly from Github - which may not run depending on your Operational System - or downloading SpatIS source code into your local computer, unpacking it and loading it from your local environment. Below we show both options.

```
# Loading from the web
# source("https://github.com/LEEClab/SpatIS/blob/master/code/spatis_source_code.R")

# Loading from the local environment, after downloading and unpacking
# Path to code folder in your computer
codedir <- "/home/leecb/Github/SpatIS/code/"
source(paste(codedir, "spatis_source_code.R", sep = ""))
```

Then it is necessary to transform individuals' locations into a SpatialPointsDataFrame (basically a data frame with spatial information embedded; take a look here for more information), so that one of the columns indicates the individual ID. Below we show an example of how the space use data may be organized and how it looks like after the transformation into a SpatialPointsDataFrame.

```
# Organizing individual locations as an example of tabular data
ids <- sapply(1:ncol(X), rep, times = nrow(X)) # Generating array of IDs
ids.vector <- as.vector(ids)
X.vector <- as.vector(X) # Array of x positions
Y.vector <- as.vector(Y) # Array of y positions

# Creating a data frame with an ID column that represents individuals
data <- data.frame(ID = ids.vector, X = X.vector, Y = Y.vector)

# This is how the original data may look like
head(data)
```

```
##   ID         X         Y
## 1  1 1.3216090 -1.740196
## 2  1 1.5943292 -1.466210
## 3  1 0.7755649  1.762281
## 4  1 0.8398471  1.803869
## 5  1 0.3697159  1.910707
## 6  1 0.2945413  2.348632
```

```
# Transforming the original data into a SpatialPointsDataFrame
spdata <- SpatialPointsDataFrame(coords = cbind(data$X, data$Y),
                                 data = subset(data, select = -c(X,Y)))

# This is how the SpatialPointsDataFrame will look like
head(spdata)
```

```
##           coordinates ID
## 1  (1.32161, -1.7402)  1
## 2 (1.59433, -1.46621)  1
## 3 (0.775565, 1.76228)  1
## 4 (0.839847, 1.80387)  1
```

```
## 5 (0.369716, 1.91071)  1
## 6 (0.294541, 2.34863)  1
## Coordinate Reference System (CRS) arguments: NA
```

After transforming the original data, it easy to run `SpatIS` with this SpatialPointsDataFrame as input by specifying which column corresponds to individual ID. `SpatIS` is based on `kerneloverlap` function from adehabitatHR package and calculates the overlap in the utilization distribution (using any of the methods of `kerneloverlap` function, look here for more details) between individuals and the whole population (assumed to be the combination of all individuals sampled).

There are two options for calling `SpatIS`. One is to create in the input SpatialPointsDataFrame a "new individual" that represents the population, with locations of all individuals gathered, and pass the "population ID" as one of the arguments when calling `SpatIS`. The other one is to let the function do it for you, if you have not done it yet (in this case, the option `population.ID` should be set to `NULL`). As we still do not have the locations and an ID that represents all the population, we are going to run the second (and easiest) option. We put the results into an object called `observed.SpatIS`.

```
# Reploting data
# cor <- rainbow(length(unique(spdata$ID)))
# plot(coordinates(spdata)[,1], coordinates(spdata)[,2], type = "n")
# points(spdata, pch = 20, col = cor[as.factor(spdata$ID)])

# Calculating SpatIS and throwing the results into the object observed.SpatIS
observed.SpatIS <- SpatIS(spdata, individuals.col = "ID")
```

The result of `SpatIS` is a list of four elements:

1) `data`: The input locations points, with the points that represent the whole population appended in the end (in our case, they receive an ID "all"):

```
# First lines
head(observed.SpatIS$data)
```

```
##           coordinates ID
## 1  (1.32161, -1.7402)  1
## 2 (1.59433, -1.46621)  1
## 3 (0.775565, 1.76228)  1
## 4 (0.839847, 1.80387)  1
## 5 (0.369716, 1.91071)  1
## 6 (0.294541, 2.34863)  1
## Coordinate Reference System (CRS) arguments: NA
```

```
# Last lines
tail(observed.SpatIS$data)
```

```
##              coordinates  ID
## 995   (-4.3938, 6.79819) all
## 996  (-5.91664, 6.63695) all
## 997  (-5.99262, 6.48406) all
## 998   (-6.6824, 4.99822) all
## 999  (-6.65972, 4.89188) all
## 1000 (-6.56448, 4.39379) all
## Coordinate Reference System (CRS) arguments: NA
```

2) `parms`: The parameters used as input to call `SpatIS`:

```
# Parameters
observed.SpatIS$parms
```

```
## $individuals.col
## [1] "ID"
##
## $population.ID
## [1] "all"
##
## $method
## [1] "VI"
```

3) `SpatIS.individual`: the value of the Spatial Individual Specialization index for each individual (i.e., the level of overlap between their utilization distribution and the population's utilization distribution):

```
# Individual SpatIS
observed.SpatIS$SpatIS.individual
```

```
##         1         2         3         4         5
## 0.6661083 0.6567337 0.7330505 0.6594467 0.7847597
```

4) `SpatIS.population`: the value of the Spatial Individual Specialization index for the whole population, defined as the SpatIS averaged over individuals:

```
# Population SpatIS
observed.SpatIS$SpatIS.population
```

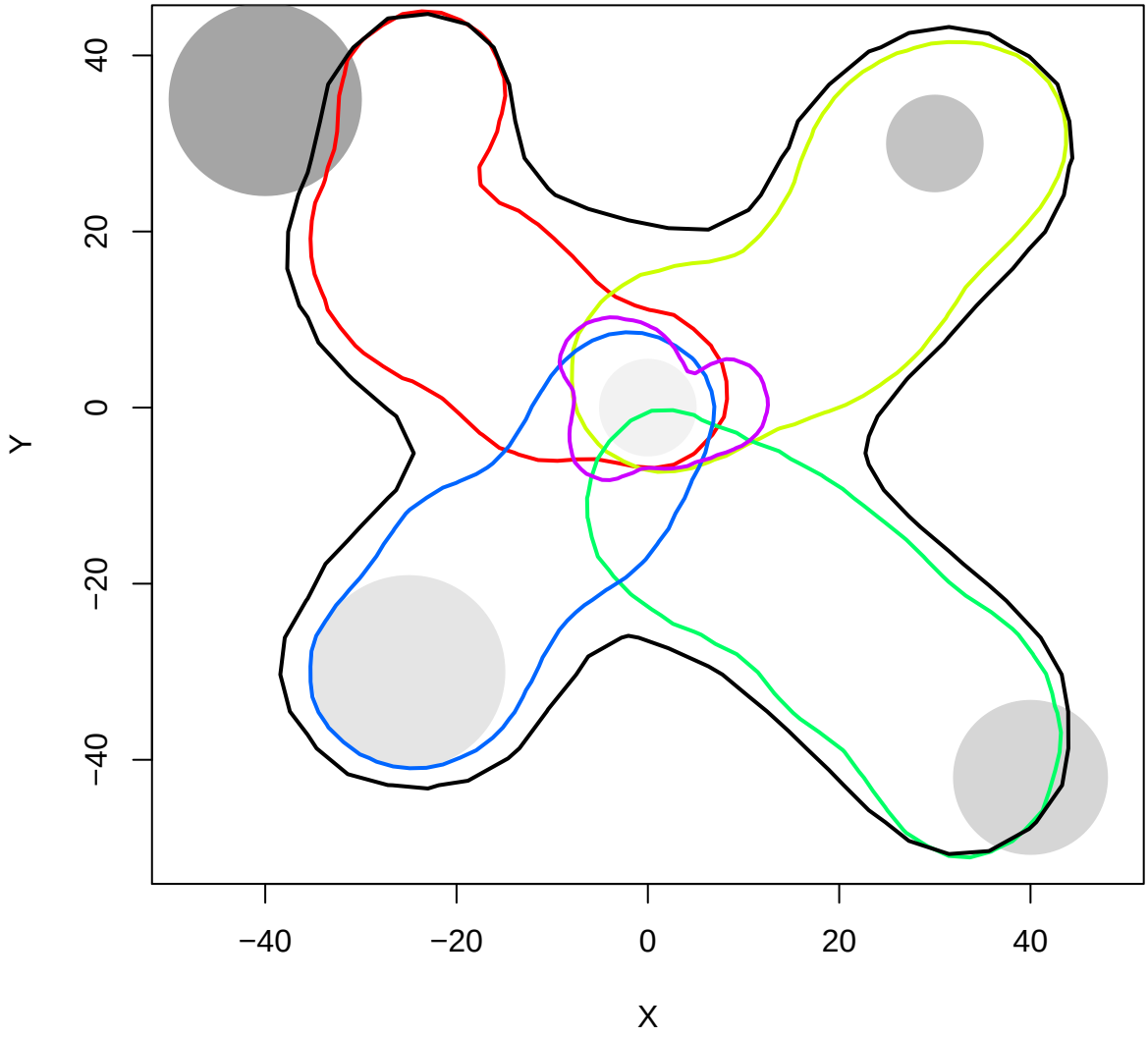
```
## [1] 0.7000198
```

Plotting 95% kernels to represent the overlap in space use between individuals and the population

Below we plot the 95% kernel isopleths for each individuald and the whole population to illustrate the overlap in their space use. The line in black show the population isopleth, and the colors represent different individuals.

```
# Calculate utilization distributions
spdata <- observed.SpatIS$data
ids <- unique(spdata$ID)
UDs <- list()
for(i in 1:length(ids)) {
  kud <- adehabitatHR::kernelUD(spdata[,1][spdata$ID == ids[i],], h = "href")
  UDs[[i]] <- kud
}

# Drawing the landscape and the 95% kernel isoplepths
colors <- c(rainbow(length(ids)-1), "black") # the last one is for the whole population
for(i in (1:length(ids))) {
```

```
  kernel95 <- adehabitatHR::getverticeshr(UDs[[i]], percent = 95)
  if(i == 1){
    # Draw the virtual landscape

    # draw a landscape that emcompasses the position of all resources
    matplot(x.resources, y.resources, type="n",
            xlim=c(1.2*min(x.resources), 1.2*max(x.resources)),
            ylim=c(1.2*min(y.resources),1.2*max(y.resources)),
            xlab = "X", ylab = "Y")
    for(j in 1:length(x.resources)) {
      draw.circle(x.resources[j], y.resources[j], radius = radius[j], border = cols[j],
                  col = cols[j]) # draw resources
    }
  }
  plot(kernel95, border=colors[i], lwd = 2, add = T)
}
```

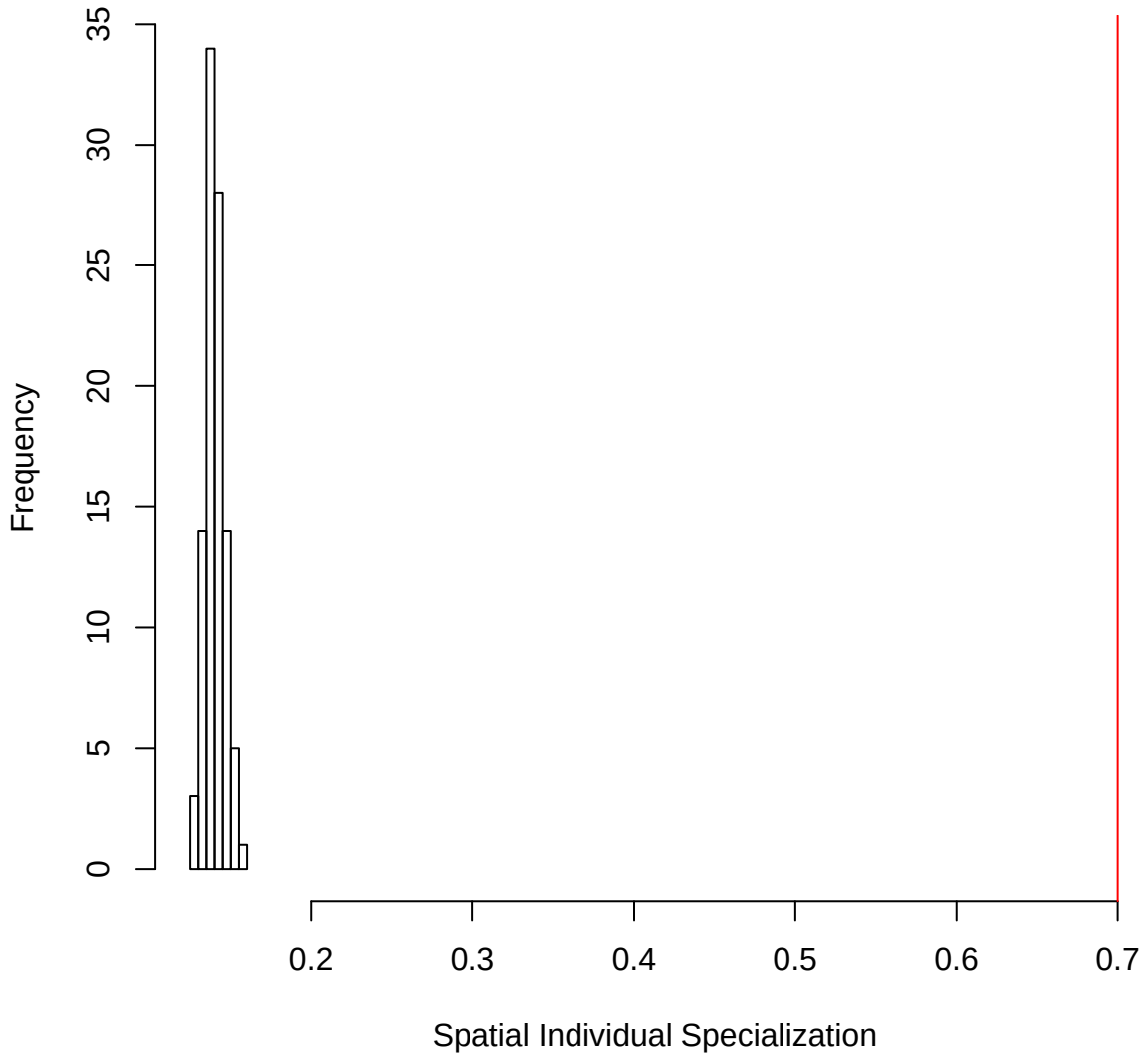


Calculating population SpatIS significance

To check whether the population SpatIS observed is different from random, we use the function `SpatIS_randomize`. This function recalculates the population SpatIS value after randomizing the population locations between individuals. This randomization procedure is repeated a number of times (the default is `iterations = 99`) and used to calculated significance (a p value) as the number of times a random calculated population SpatIS is equal or greater than the observed one. The randomization process may be a bootstrap (randomize locations with replacement) or a permutation procedure (without replacement; this option is controled by the argument `bootstrap`). The input used for running `SpatIS_randomize` is the list that result from `SpatIS` function. The function also plots (if `plot = TRUE`) how far is the observed value (in red) from permutated ones.

The output is a list with the random SpatIS values, the observed SpatIS calculated through `SpatIS` function, and the significance (p) value.

```
# Run permutation
permutations <- SpatIS_randomize(observed.SpatIS, iterations = 99)
```



```r
# Show permutation results
permutations
```

```
## $SpatIS.population.random
##   [1] 0.7000198 0.1336518 0.1348940 0.1480620 0.1422582 0.1363780 0.1320915
##   [8] 0.1471521 0.1473762 0.1389606 0.1544504 0.1411958 0.1393117 0.1455376
##  [15] 0.1508199 0.1451626 0.1398002 0.1319166 0.1459914 0.1440498 0.1471100
##  [22] 0.1381652 0.1418889 0.1416793 0.1390012 0.1350448 0.1415970 0.1445516
##  [29] 0.1371288 0.1448986 0.1375117 0.1354158 0.1349310 0.1393413 0.1337542
##  [36] 0.1384510 0.1431784 0.1401243 0.1423438 0.1425301 0.1450452 0.1325925
##  [43] 0.1436468 0.1388173 0.1441463 0.1404393 0.1452261 0.1379620 0.1439171
##  [50] 0.1441479 0.1463938 0.1371079 0.1364046 0.1584654 0.1403543 0.1391056
##  [57] 0.1519512 0.1395093 0.1422541 0.1501404 0.1325249 0.1352031 0.1369724
##  [64] 0.1435425 0.1413679 0.1372578 0.1420869 0.1377906 0.1387062 0.1329323
##  [71] 0.1316689 0.1430391 0.1285786 0.1453681 0.1463787 0.1369124 0.1415030
##  [78] 0.1373894 0.1371114 0.1404849 0.1359595 0.1323132 0.1397002 0.1377653
##  [85] 0.1346623 0.1377949 0.1500077 0.1409137 0.1365668 0.1398264 0.1347349
##  [92] 0.1453037 0.1323520 0.1258652 0.1421465 0.1425942 0.1459117 0.1388885
##  [99] 0.1299219 0.1361797
##
## $SpatIS.population.observed
## [1] 0.7000198
##
## $p
## [1] 0.01
```

Citation

If you need more information or use *SpatIS*, please refer to

Kerches-Rogeri, P.; Niebuhr, B. B. S.; Muylaert, R. L., Mello, M. A. R. Individual specialization in the space use of frugivorous bats. *In prep.*