# This is an introduction to calculating isotopic Standard Ellipse Areas in Program R. The examples below come from Andrew Jackson’s GitHub page. This website has lots of additional details and helpful examples.

# <https://github.com/AndrewLJackson/SIBER>

# BASIC BAYESIAN ELLIPSES WITH SIBER – GETTING STARTED

# Don’t forget to set your working directory!! Here, it is my desktop.

setwd("~/Desktop/")

# Cleaning out the work space rm(list=ls())

graphics.off()

# Here, I set the seed each time so that the results are comparable.

# This is useful as it means that anyone that runs your code, \*should\*

# get the same results as you, although random number generators change # from time to time.

set.seed(1)

# load SIBER package

library(SIBER)

# load in the included demonstration dataset

# Here we are using data from Emma on Ancient and Modern Sea Otters

# If working with your own data read in from a \*.csv file. You can load this # manually (i.e., “File/Import Dataset” in RStudio), or using the

# “read.csv()” function within base R.

mydata <- read.csv('ottSOUTH.csv', header=T)

mydata$X <- NULL # removing a column that gets added during import

str(mydata) # check the structure of your data

# create the siber object

siber.example <- createSiberObject(mydata)

**PLOTTING THE RAW DATA**

With the siber object created, we can now use various functions to create isotope biplots of the data, and also calculate some summary statistics on each group and/or community in the dataset.

Various plotting options are collated into lists and then passed to the high-level SIBER plotting function plotSiberObject which is a wrapper function for easy plotting. We will access the more specific plotting functions directly a little later on to create more customized graphics. ax.pad determines the padding applied around the extremes of the data.

iso.order is a vector of length 2 specifying which isotope should be plotted on the x and y axes. There is currently a problem with the addition of the group ellipses using if you deviate from the default of iso.order = c(1,2). This argument will be deprecated in a future release, and plotting order will be achieved at point of data-entry. I recommend you set up your original data with the chemical element you want plotted on the x-axis being the first column, and the y axis in the second.

Convex hulls can. be drawn between the centers of each group within a community with hulls = T. Convex hulls are draw around each group independently with group.hulls = T. Ellipses are drawn for each group independently with ellipses = T. These ellipses can be made to be maximum likelihood standard ellipses by setting p = NULL, or can be made to be prediction ellipses that contain approximately p proportion of data. For example, p = 0.95 will draw an ellipse that encompasses approximately 95% of the data. The parameter n determines how many points are used to make each ellipse and hence how smooth the curves are.

# Create lists of plotting arguments to be passed onwards to each # of the three plotting functions.

community.hulls.args <- list(col = 1, lty = 1, lwd = 1)

group.ellipses.args <- list(n = 100, p.interval = 0.95, lty = 1, lwd = 2) group.hull.args <- list(lty = 2, col = "grey20")

par(mfrow=c(1,1))

plotSiberObject(siber.example,

ax.pad = 2,

hulls = F,

community.hulls.args,

ellipses = T, group.ellipses.args,

group.hulls = T,

group.hull.args, bty = "L", iso.order = c(1,2),

xlab = expression({delta}^13\*C~'\u2030'), ylab = expression({delta}^15\*N~'\u2030')

)

# *Note that many of the parameters in this plot can be adjusted. Refer to Andrew Jackson’s GitHub page (*[*https://github.com/AndrewLJackson/SIBER*](https://github.com/AndrewLJackson/SIBER)*) for lots of helpful details and examples!*

# SUMMARY STATISTICS

Although the intention of SIBER is to use Bayesian methods to allow us to make statistical comparisons of what are otherwise typically point estimates of dispersion within and among communities and groups, the basic summary statistics are informative and useful for checking that our Bayesian analysis is working as intended.

# Calculate summary statistics for each group: TA, SEA and SEAc

group.ML <- groupMetricsML(siber.example) print(group.ML)

# Calculate the various Layman metrics on each of the communities.

community.ML <- communityMetricsML(siber.example) print(community.ML)

# FITTING THE BAYESIAN MODELS TO THE DATA

Whether your intended analysis is to compare isotopic niche width among groups, or among communities, the initial step is to fit Bayesian multivariate normal distributions to each group in the dataset. The decision as to whether you then want to compare the area of the ellipses among groups, or any / all of the 6 Layman metrics comes later.

These multivariate normal distributions are fitted using the jags software run via the package rjags. This method relies on an interated Gibbs Sampling technique and some information on the length, number and iterations of sampling chains is required. Additionally, the prior distributions for the parameters need to be specified. In SIBER, these are bundled into two list objects: parms which holds the parameters defining how the sampling algorithm is to run; and priors which holds information on the prior distributions of the parameters to be estimated.

**Typically, the priors are left vague and you should use these same values in your own analysis.** Since the data are z-scored internally before the models are fitted to the data, the expected means are inherently close to zero, and the mariginal variances close to one. This greatly aids the jags fitting process. After calling siberMVN() you will see output in the command window indicating that the jags models are being fitted, one block of output for each group in your dataset.

# options for running jags

parms <- list()

parms$n.iter <- 2 \* 10^4 # number of iterations to run the model parms$n.burnin <- 1 \* 10^3 # discard the first set of values

parms$n.thin <- 10 # thin the posterior by this many

parms$n.chains <- 2 # run this many chains

# define the priors

priors <- list()

priors$R <- 1 \* diag(2) priors$k <- 2 priors$tau.mu <- 1.0E-3

# Fit the model (note, this DOES NOT run it. See below)

# fit the ellipses which uses an Inverse Wishart prior

# on the covariance matrix Sigma, and a vague normal prior on the

# means. Fitting is via the JAGS method.

ellipses.posterior <- siberMVN(siber.example, parms, priors)

# COMPARING AMONG GROUPS USING STANDARD ELLIPSE AREA

When comparing individual groups with each other, be it within a single community, or groups among communities, the Standard Ellipse Area (SEA) is the recommended method. Since the multivariate normal distributions have already been fitted to each group, it only remains to calculate the SEA on the posterior distribution of covariance matrices for each group, thereby yielding the Bayesian SEA or SEA-B. We can also use the summary statistics we calculated earlier to add the maximum likelihood estimates of SEA-c to the Bayesian estimates.

Plotting is via the function siberDensityPlot(). Credible intervals can be extracted by calling the function hdr from the hdrcde package.

# The posterior estimates of the ellipses for each group can be used to # calculate the SEA.B for each group.

# RUNNING THE BAYESIAN MODEL:

SEA.B <- siberEllipses(ellipses.posterior)

# plot the results.

siberDensityPlot(SEA.B, xticklabels = colnames(group.ML),

xlab = c("Community | Group"),

ylab = expression("Standard Ellipse Area " ('\u2030' ^2) ), bty = "L", las = 1,

main = "SIBER ellipses on each group"

)

# Calculate some credible intervals

cr.p <- c(0.95, 0.99) # vector of quantiles

# call to hdrcde:hdr using lapply()

SEA.B.credibles <- lapply(as.data.frame(SEA.B),

function(x,...){tmp<- hdrcde::hdr(x)$hdr},

prob = cr.p)

# do similar to get the modes, taking care to pick up multimodal posterior

# distributions if present

SEA.B.modes <- lapply(as.data.frame(SEA.B),

function(x,...){tmp<- hdrcde::hdr(x)$mode},

prob = cr.p, all.modes=T)

# and using print command you can summarize credible intervals and modes

print(SEA.B.credibles)

print(SEA.B.modes)

# COMPARISON OF ENTIRE COMMUNITIES USING LAYMAN METRICS

Entire communities can be compared by recognizing that convex hulls (or any other metric) based on the centroids of each group are subject to uncertainty in the location of the centroids. We can exploit this and calculate the distribution of convex hull areas (and the other 5 metrics) based on the posterior distribution of the means of each group’s x and y data.

The first thing to do is to extract the posterior means from the mcmc.list object that jags creates to make plotting and analysis easier. Thereafter, we can plot their distribution using highest density region boxplots to visualize the credible intervals that describe it.

# extract the posterior means

mu.post <- extractPosteriorMeans(siber.example, ellipses.posterior)

# calculate the corresponding distribution of layman metrics layman.B <- bayesianLayman(mu.post)

# Visualize the first community

siberDensityPlot(layman.B[[1]],

xticklabels = colnames(layman.B[[1]]),

bty="L", ylim = c(0,20))

# Visualize the second community

siberDensityPlot(layman.B[[2]],

xticklabels = colnames(layman.B[[2]]),

bty="L", ylim = c(0,20))

# Alternatively, pull out TA from both and aggregate them into a

# single matrix using cbind() and plot them together on one graph.

# go back to a 1x1 panel plot par(mfrow=c(1,1))

siberDensityPlot(cbind(layman.B[[1]][,"TA"], layman.B[[2]][,"TA"]),

xticklabels = c("Community 1", "Community 2"),

bty="L",

ylim = c(0,20),

las = 1,

ylab = "TA - Convex Hull Area",

xlab = "")