Introduction to SIBER

Andrew L Jackson.   2017-04-28

SIBER introduction and guide

This user manual introduces the basic functionality of this standalone version of SIBER, which was previously part of the siar package. SIBER contains two types of analysis, although both are founded on the same principle of fitting ellipses to groups of data. The questions are very different though and it is important that you satisfy yourself with which one you want for the questions you have of your own data. There are additional learning resources available at SIAR-examples-and-queries. N.B. these examples currently still use the SIBER functions embedded within the SIAR package, which I will update shortly, but the concepts are exactly the same.

Setting up our R session for this demonstration

In this example, we are going to work with a bundled example dataset that I created previously using the function `generateSiberData()`. This dataset is loaded using `data("demo.siber.data")` but it is also provided as a raw *.csv file which is more usually the format you would work with when organising your own dataset for analysis using SIBER. We then use `createSiberObject()` to convert this raw data form into an object that contains information summary statistics that are useful for plotting, and a z-score transformed version of the data which is used in the model fitting process before being back-transformed using the summary statistic information.

```
# remove previously loaded items from the current environment and remove
previous graphics.
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)

library(SIBER)

# load in the included demonstration dataset
data("demo.siber.data")
# create the siber object
siber.example <- createSiberObject(demo.siber.data)


# Or if working with your own data read in from a *.csv file, you would use
# This *.csv file is included with this package. To find its location
# type
# fname <- system.file("extdata", "demo.siber.data.csv", package = "SIBER")
# in your command window. You could load it directly by using the
# returned path, or perhaps better, you could navigate to this folder
# and copy this file to a folder of your own choice, and create a
# script from this vingette to analyse it. This *.csv file provides
# a template for how your own files should be formatted.

# mydata <- read.csv(fname, header=T)
# 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.

Community 1 comprises 3 groups and drawn as black, red and green circles community 2 comprises 3 groups and drawn as black, red and green triangles

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 customised graphics.

`ax.pad` determines the padding applied around the extremes of the data.

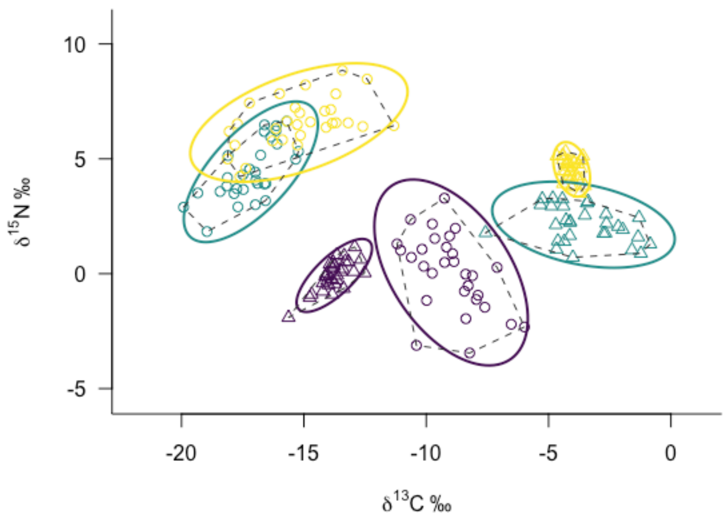
iso.order is a vector of length 2 specifying which isotope should be plotted on the x and y axes.N.B. there is currently a problem with the addition of the group ellipses using if you deviate from the default of `iso.order = c(1,2)`. This argument will be deprecated in a future release, and plotting order will be acheived at point of data-entry. I recommond 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 are drawn between the centres of each group within a community with `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 predicition 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.

Convex hulls are draw around each group independently with `group.hulls = T`.

```
# 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')
                  )
```



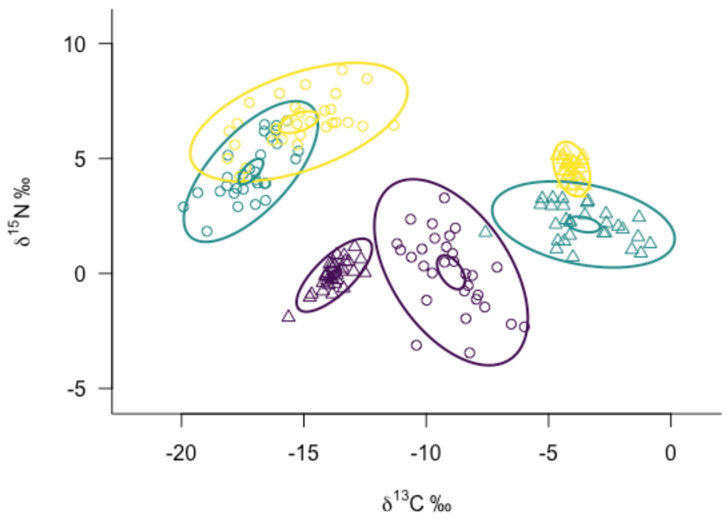
Summary statistics and custom graphic additions

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.

One feature of the Standard Ellipse is that it contains approximately 40% of the data. SIBER now includes code to scale this ellipse so that it contains approximately any % of the data you wish. Additionally, the ellipse can be scaled so that it represents a % confidence ellipse of the bivariate means (rather than of the data). We create the bi-plot again here and this time add the additional ellipses overlayed on the basic plot that this time omits group hulls and group standard ellipses.

```
par(mfrow=c(1,1))

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")
```

```
# this time we will make the points a bit smaller by
# cex = 0.5
plotSiberObject(siber.example,
                  ax.pad = 2,
                  hulls = F, community.hulls.args,
                  ellipses = F, group.ellipses.args,
                  group.hulls = F, group.hull.args,
                  bty = "L",
                  iso.order = c(1,2),
                  xlab=expression({delta}^13*C~'\u2030'),
                  ylab=expression({delta}^15*N~'\u2030'),
                  cex = 0.5
                  )

# Calculate summary statistics for each group: TA, SEA and SEAc
group.ML <- groupMetricsML(siber.example)
print(group.ML)
#>             1.1       1.2       1.3       2.1       2.2       2.3
#> TA   21.924922 10.917715 17.945127 3.0714363 11.476354 1.4818061
#> SEA   5.783417  3.254484  5.131601 0.8623300  3.458824 0.4430053
#> SEAc  5.989967  3.370715  5.314872 0.8931275  3.582354 0.4588269

# You can add more ellipses by directly calling plot.group.ellipses()
# Add an additional p.interval % prediction ellilpse
plotGroupEllipses(siber.example, n = 100, p.interval = 0.95,
                    lty = 1, lwd = 2)

# or you can add the XX% confidence interval around the bivariate means
# by specifying ci.mean = T along with whatever p.interval you want.
plotGroupEllipses(siber.example, n = 100, p.interval = 0.95, ci.mean = T,
                    lty = 1, lwd = 2)
```



Entire document at: https://cran.r-project.org/web/packages/SIBER/vignettes/Introduction-to-SIBER.html