BASIC MIXING MODEL WITH SIMMR (AN EASY VERSION OF MIXSIAR)

In this example, we will analyze "goose" data.

Load SIMMR.

```
library(simmr)
```

Read in data from either .csv or .txt files.

```
mix.data <- read.csv("geese.csv", header = TRUE, stringsAsFactors = FALSE)

sources.data <- read.csv("sources.csv", header = TRUE, stringsAsFactors =
FALSE)

corrections.data <- read.csv("corrections.csv", header = TRUE,
stringsAsFactors = FALSE)
```

OR

```
mix.data <- read.table("geese.txt", sep = "\t",
                   header = TRUE, stringsAsFactors = FALSE)

sources.data <- read.table("sources.txt", sep = "\t",
                       header = TRUE, stringsAsFactors = FALSE)

corrections.data <- read.table("corrections.txt", sep = "",
                           header = TRUE, stringsAsFactors = FALSE)
```

While `siar` takes these files in this format, `simmr` requires us to strip out some of the data and put them in their own vector objects. It then bundles them into a special object of class `simmr` which makes it tidier in the long run, as special functions within the package can recognize the object and calculate and plot all the various analyses we seek just by being given this single object. Note that in all that follows I am specifying the d13C data first and the d15N data second.

The mixtures (consumers) need to be a two-column matrix:

```
mix <- cbind(mix.data$d13CPl, mix.data$d15NPl)
```

The groups need to be integer codes that identify each group of consumers and correspond to the mix data above:

```
grp <- mix.data$Group
```

The sources are split into a vector of names, and matrices of means and standard deviations:

```
s_names <- sources.data$Means
s_means <- cbind(sources.data$meand13C, sources.data$meand15N)
s_sds   <- cbind(sources.data$sdd13C,   sources.data$sdd15N)
```

The corrections are specified similarly but do not need a name vector:

```
c_means <- cbind(corrections.data$Mean13C, corrections.data$Mean15N)
c_sds   <- cbind(corrections.data$Sd13C, corrections.data$Sd15N)
```

Final formatting to package the data for the model:

```
simmr_in = simmr_load(mixtures=mix,
                      source_names=s_names,
                      source_means=s_means,
                      source_sds=s_sds,
                      correction_means=c_means,
                      correction_sds=c_sds,
                      group=grp)
```

The raw data can then be plotted as a biplot and the `plot.simmr_in` method allows some customization: see `?plot.simmr_in` for more options. But, bear in mind that if you want to do anything specific with your plot, you will need to write your own `ggplot` script.

Plot the data:

```
plot(simmr_in, xlab=expression(paste(delta^13, "C (\u2030)",sep="")),
     ylab=expression(paste(delta^15, "N (\u2030)",sep="")),
     title='Isospace plot of goose data', group = 1:8)
```

The next thing we might do is to fit the model. Again, the help file for `simmr_mcmc` provides details on the number of iterations to be sampled from the posterior distributions of all the parameters, and also allows you to set the priors. It defaults to a basic run, which will give you a good feel for most datasets at least as a starting point. As the model is fit, `JAGS`, which does the actual mcmc model fitting, prints its progress to the console window.

Fit (run) the model:

```
simmr_out <- simmr_mcmc(simmr_in)
```

With the model run, we want to get some information on the model using `summary()`. This is a special summary function that provides information on both the model fit, and also the parameters which is really what we are interested in. Model fit is assessed by the Gelman diagnostic where we want all parameters to be close to 1 - in this case they are exactly 1. This diagnostic determines how well the chains (4 in this case) have converged. The summary of the parameter estimates includes some common quantiles (2.5, 25, 50, 75 and 97.5% quantiles), along with their mean and standard deviation. We also get a matrix of the correlations among the posterior samples for these parameters - more on this later. The plots of this convergence always start off away from 1, but should move towards, and stay close to 1 for the remainder of the trace.

Summary of model diagnostics:

```
summary(simmr_out, type = "diagnostics")
```

Summary of source proportions:

```
summary(simmr_out, type = "statistics", group = c(1:8))
```

The basic way to visualize the posterior estimates is to use either density plots of the posterior distribution, or boxplot summaries, or we might try to use the siar/siber style density plots which use the package `hdrcde`. We need to pick which group we want to visualize with these basic plotting functions, and in this goose example we have 8 groups to choose from.

Plot the model output (density plot):

```
plot(simmr_out, type = "density", group = 1)
```

Plot the model output (boxplot):

```
plot(simmr_out, type = "boxplot", group = 1)
```

Those figures provide information on the estimates of the key parameters on their own (their marginal estimates) but many parameters in a model are correlated in some way, and this is especially true of the proportions since they must sum to one. Understanding the correlations among the estimated proportions provides some really useful information that is key to the interpretation of the output.

Plot the model output (matrix):

```
plot(simmr_out, type='matrix', group = 1)
```

Often, we are interested in specific sources and how their proportion in the diet differs among groups, rather than just simply presenting what each group is eating. In the case of these geese, we are particularly interested in how the geese differ in their consumption of *Zostera* and terrestrial grass over the course of time where our 8 groups represent 8 time points over two seasons. Some simple functions allow us to do this.

To compare contributions from two specific sources:

```
compare_sources(simmr_out,source_names=c('Zostera','Grass'), group = c(4))
```

Comparing groups is done in a similar manner, and this time we choose which source we want to visualize across groups. As well as plotting, these functions also return the probabilities of all the possible orders of proportions from largest to smallest.

```
compare_zostera <- compare_groups(simmr_out,source = 'Zostera', groups = 1:8)

compare_grass <- compare_groups(simmr_out,source = 'Grass'  , groups = 1:8)
```

Andrew has strong feelings about combining sources, and far prefers to see **_a posteriori_** aggregation of the proportions. Since each iteration of the mcmc fitting process produces proportions that sum to one, we can simply add up two or more of these proportions, for every iteration. What is nice about this approach is that the resultant distribution is usually more precise, since there is generally negative correlation among these proportions (see the matrix plots above), and when we add two variances, the equation is var(a+b) = var(a) + var(b) + 2*cov(a,b), so when the covariance is negative, the summed variance can shrink. In this dataset, we don't really care much that the geese are eating *U. lactuca* or *Enteromorpha* (and indeed, it turns out these are actually two phenotypes of the same species), so we might just lump these into a category of "Green Algae". All the plotting and summary options we presented above will work on this new combined data structure.

Combine *U. lactuca* and *Enteromorpha*:

```
combined_sources <- combine_sources(simmr_out,
            to_combine = c('U.lactuca','Enteromorpha'),
            new_source_name = 'Green Algae')

plot(combined_sources, group = 1, type = "boxplot")

plot(combined_sources, group = 8, type = "boxplot")
```