Practical: Using SIAR

Andrew Parnell

Introduction

Welcome to the practical! We will learn about:

- Loading data into SIAR
- Running SIAR
- Getting output from SIAR
- Using SIAR for single observations
- Setting prior distributions on the dietary proportions
- Customising your own SIAR output

It's assumed that you already have SIAR installed via:

```r
library(devtools)
install_github('andrewljackson/siar')
```

You should then be able to run

```r
library(siar)
```

without error.

This document is a slightly friendlier and more up to date version of the SIAR manual.

You should follow and run the commands shown in the grey boxes below. At various points you will see a horizontal line in the text which indicates a question you should try to answer, like this:

What words does the following command print to the console?

```r
print("Hello World")
```

If you get stuck, please get our attention and we will try to help! There are no prepared answers to these questions so keep you own record as you go. At the end of the practical are harder questions which you can attempt if you get through all of the material. If you find any mistakes in the document please let us know.

You can run the code from these practicals by loading up the `.Rmd` file in the same directory in Rstudio. This is an R markdown document containing all the text. Feel free to add in your own answers, or edit the text to give yourself extra notes. You can also run the code directly by highlighting the relevant code and clicking `Run`.

Loading data into SIAR

There are two main ways of working with SIAR. One is through the menu system, which you can access through `siarmenu()` upon loading the package. This version is for beginning users and only allows access to all but the most basic of features. The second way is to use the command line. This is a far more powerful way of using SIAR and gives full access to all quantities created as part of the model. We will focus on using the command line to run SIAR.

Included with SIAR are some example data sets which we will work with throughout this document. When running your own models you should try to keep your data in the same format as these examples. At minimum you need two files to get SIAR working; a consumers file and a sources file. The simplest geese data set is obtained via:

```r
data(geese1demo)
print(geese1demo)
```

```
##       d15NPl d13CPl
##  [1,]  10.22 -11.36
##  [2,]  10.37 -11.88
##  [3,]  10.44 -10.60
##  [4,]  10.52 -11.25
##  [5,]  10.19 -11.66
##  [6,]  10.45 -10.41
##  [7,]   9.91 -10.88
##  [8,]  11.27 -14.73
##  [9,]   9.34 -11.52
```

This data set has two columns, one for each isotope, and 9 individuals. A useful command for learning about the structure of an R data set is `str`, especially for large data objects:

```r
str(geese1demo)
```

```
##  num [1:9, 1:2] 10.2 10.4 10.4 10.5 10.2 ...
##  - attr(*, "dimnames")=List of 2
##   ..$ : NULL
##   ..$ : chr [1:2] "d15NPl" "d13CPl"
```

We can see that it has 9 rows, 2 columns, and is of numeric (`num`) mode. The two column labels refer to the δ^{15}N and δ^{13}C isotope values.

The source data can be obtained from:

```r
data(sourcesdemo)
print(sourcesdemo)
```

```
##        Sources  Meand15N    SDd15N  Meand13C    SDd13C
## 1      Zostera  6.488984 1.4594632 -11.17023 1.2149562
## 2        Grass  4.432160 2.2680709 -30.87984 0.6413182
## 3    U.lactuca 11.192613 1.1124385 -11.17090 1.9593306
## 4 Enteromorpha  9.816280 0.8271039 -14.05701 1.1724677
```

We can see that there are 4 sources, with their names in the first column. The remaining columns refer to the means and standard deviations for each source on each isotope. The isotopes need to be in the same order as the consumer data in `geese1demo`. Note the structure of this object

```r
str(sourcesdemo)
```

```
## 'data.frame':    4 obs. of  5 variables:
##  $ Sources : Factor w/ 4 levels "Enteromorpha",..: 4 2 3 1
##  $ Meand15N: num  6.49 4.43 11.19 9.82
##  $ SDd15N  : num  1.459 2.268 1.112 0.827
##  $ Meand13C: num  -11.2 -30.9 -11.2 -14.1
##  $ SDd13C  : num  1.215 0.641 1.959 1.172
```

It's a data frame. This is an R data type which can store both text and numbers, useful for storing the source names as well as their isotope values.

We could run SIAR with just these two data files. However, this would produce a pretty poor model as we don't have any corrections for the TEFs. The TEFs file looks just like the source file:

```
data(correctionsdemo)
print(correctionsdemo)
```

```
##          Source Mean15N Sd15N Mean13C Sd13C
## 1       Zostera    3.54  0.74    1.63  0.63
## 2         Grass    3.54  0.74    1.63  0.63
## 3     U.lactuca    3.54  0.74    1.63  0.63
## 4 Enteromorpha    3.54  0.74    1.63  0.63
```

If you were loading these data sets in yourself, it's best to store them in the same directory and then load them in from there, e.g.:

```
# Set the working directory (where R looks first for files)
setwd('path/to/files')
# Read in consumers
consumers = read.table('my_consumer_file.txt',header=TRUE)
# Read in sources
sources = read.table('my_sources_file.txt',header=TRUE)
# Read in TEFs
TEFs = read.table('my_TEF_file.txt',header=TRUE)
```

The extra `header=TRUE` argument tells R that there are column names at the top of the file.

---

1. What is the structure of the TEFs object? How many rows and columns does it have?
2. There's another data object that comes with SIAR called `geese2demo`. How many rows and columns does this have?
3. Create some simple scatter plots of the `geese1demo` data using `plot`. See if you can add in the source means corrected for the TEF means (hint: add the means together and then plot using `points`)

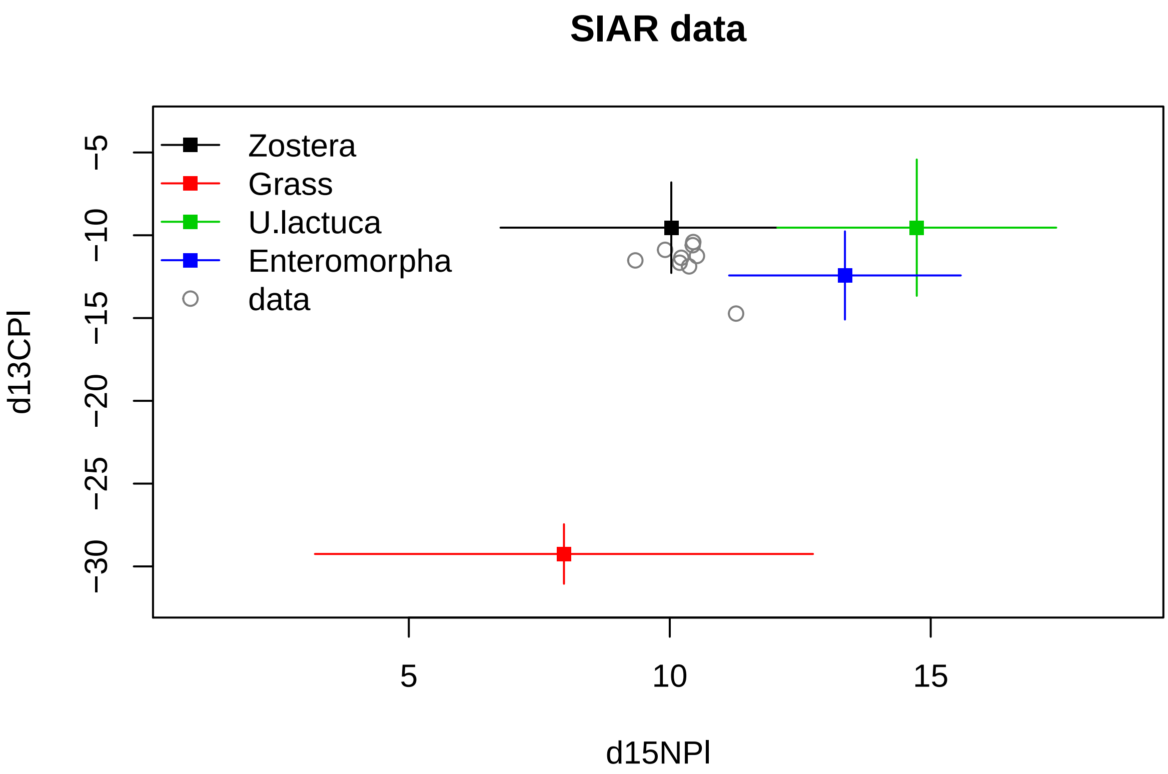
Running SIAR

We are now in a position to run SIAR. The function to use is `siarmcmcdirichletv4`. You can find help on this function by typing the name with a `?` in front. If you are using Rstudio you can use the `<TAB>` key to complete your command once you have typed in the first few characters. To run SIAR, type:

```
out = siarmcmcdirichletv4(geese1demo,sourcesdemo,correctionsdemo)
```

SIAR now runs the MCMC algorithm (just like JAGS) and, whilst running, reports the number of iterations that it has achieved. When finished, the first thing to do ALWAYS is to create an isospace plot:

```
siarplotdata(out)
```



You should see that the consumers are inside the mixing polygon (or *convex hull*) of the sources. The consumers are close to the Zostera source, so we would expect this to come out as the main dietary proportion.

We can plot the posterior distributions of the dietary proportions with:

```
siarhistograms(out)
```

This will ask you whether you want them all on the same panel or separately.

If you want more textual output you can get it with:

```
siarhdrs(out)
```

```
## Summary information for the output file ...
##                Low 95% hdr High 95% hdr       mode       mean
## Zostera          0.5087821    0.8240344 0.66798276 0.66379770
## Grass            0.0366220    0.1449540 0.09143140 0.09184657
## U.lactuca        0.0000000    0.2400611 0.03606249 0.10451769
## Enteromorpha     0.0000000    0.3088168 0.10882299 0.13983804
## SD1              0.0000000    1.3168299 0.12015812 0.49926411
## SD2              0.0000000    2.0838041 0.84906122 0.98209225
##
## Running convergence diagnostics on output.
## Output parameters need to have been loaded in or created.
##
## Worst parameters are ...
##        Grass    U.lactuca      Zostera          SD1          SD2
##    0.1751009    0.2922422    0.3475910    0.3829422    0.4451434
## Enteromorpha
##    0.4725043
## If lots of the p-values are very small, try a longer run of the MCMC.
```

This will produce a 95% HDR interval for the posterior proportions, and the estimated mode and mean. It will also give the same estimates for the residual standard deviations for each isotope. These will tend to be large when the consumers lie outside the source mixing polygon. At the end, the command will produce Geweke p-value estimates to help you check convergence. The rule of thumb if that many of these are small (e.g. < 0.01) you should probably try a longer run.

The last data set to include is that of concentration dependence. These are given as proportions and again is in the same format as the sources and TEFs:

```
data(concdepdemo)
print(concdepdemo)
```

```
##        Sources Meand15N SDd15N Meand13C SDd13C
## 1      Zostera   0.0297 0.0097   0.3593 0.0561
## 2        Grass   0.0355 0.0063   0.4026 0.0380
## 3    U.lactuca   0.0192 0.0053   0.2098 0.0327
## 4 Enteromorpha   0.0139 0.0057   0.1844 0.1131
```

```
str(concdepdemo)
```

```
## 'data.frame':    4 obs. of  5 variables:
##  $ Sources : Factor w/ 4 levels "Enteromorpha",..: 4 2 3 1
##  $ Meand15N: num  0.0297 0.0355 0.0192 0.0139
##  $ SDd15N  : num  0.0097 0.0063 0.0053 0.0057
##  $ Meand13C: num  0.359 0.403 0.21 0.184
##  $ SDd13C  : num  0.0561 0.038 0.0327 0.1131
```

Note that although this data set includes standard deviations on the sources, they are currently not used by SIAR to run the model.

To run the model with concentration dependence, include this new data set as an extra argument to the `siarmcmcdirichletv4` function:

```
out = siarmcmcdirichletv4(geese1demo,sourcesdemo,correctionsdemo,concdepdemo)
```

---

1. What's the structure of the out object? Can you see anything you recognise in it? Try accessing different parts of it using the `$` notation, e.g. `out$TITLE`
2. Try the command `siarproportionbygroupplot(out)`. What does this produce?
3. Try running the model again without including the `correctionsdemo` argument. What happens to the isospace plot?

Longer SIAR runs

If you want to be really certain of convergence you can run SIAR for more iterations with some extra arguments. The extra arguments are:

1. `iterations` which sets the total number of iterations. The default is 200,000
2. `burnin` which sets the number of initial iterations to remove. The default is 50,000
3. `thinby` which sets the amount of thinning (removal) of iterations to avoid autocorrelation in the output values. The default is 15, which means SIAR will keep only every 15th iteration

Usually the default values will be fine, but you could double them if you wanted a longer run. If you're annoyed by how often SIAR reports its progress you can change this with the `howmany` argument. The resulting number of iterations kept by SIAR for the posterior distribution is `(iterations-burnin)/thinby`. It's usually not a good idea to store more than 10,000 iterations unless you have lots of RAM.

A longer run for SIAR might thus be:

```
out_2 = siarmcmcdirichletv4(geese1demo,sourcesdemo,correctionsdemo,iterations=400000, burnin=200000,thi
```

1. Without checking, how many iterations will the command above save?
2. Did the results change much between the shorter and longer run?
3. Were the convergence results better for the longer run (i.e. were the *p*-values for the Geweke test bigger?)

Working with multiple groups

Sometimes you might be interested in running SIAR for multiple different groups of consumers. These different groups might be different sexes, different sampling periods, different locations, etc. SIAR will run these simultaneously and store the output for easier plots and comparison.

The data which are included in SIAR for multiple groups analysis can be found with:

```
data(geese2demo)
head(geese2demo,15)
```

```
##       Group d15NPl d13CPl
##  [1,]     1  10.22 -11.36
##  [2,]     1  10.37 -11.88
##  [3,]     1  10.44 -10.60
##  [4,]     1  10.52 -11.25
##  [5,]     1  10.19 -11.66
##  [6,]     1  10.45 -10.41
##  [7,]     1   9.91 -10.88
##  [8,]     1  11.27 -14.73
##  [9,]     1   9.34 -11.52
## [10,]     2  11.68 -15.89
## [11,]     2  12.29 -14.79
## [12,]     2  11.04 -17.64
## [13,]     2  11.46 -16.97
## [14,]     2  11.73 -17.25
## [15,]     2  12.29 -14.77
```

```
str(geese2demo)
```

```
##  num [1:251, 1:3] 1 1 1 1 1 1 1 1 1 2 ...
##  - attr(*, "dimnames")=List of 2
##   ..$ : NULL
##   ..$ : chr [1:3] "Group" "d15NPl" "d13CPl"
```

This is a much bigger data set. The first column contains the group number. When SIAR sees data with an integer in the first column it automatically knows to run the group version of its analysis steps. We can see how many and how large the groups are with:

```
table(geese2demo[,'Group'])
```

```
## 
##  1  2  3  4  5  6  7  8 
##  9 29 74 10 41 20 32 36
```

so 8 groups ranging from 9 to 74 observations. SIAR will work with up to 30 groups. There needs to be at least 2 observations per group for SIAR to run, but really 5 or more is desirable if you want to properly estimate the residual error.

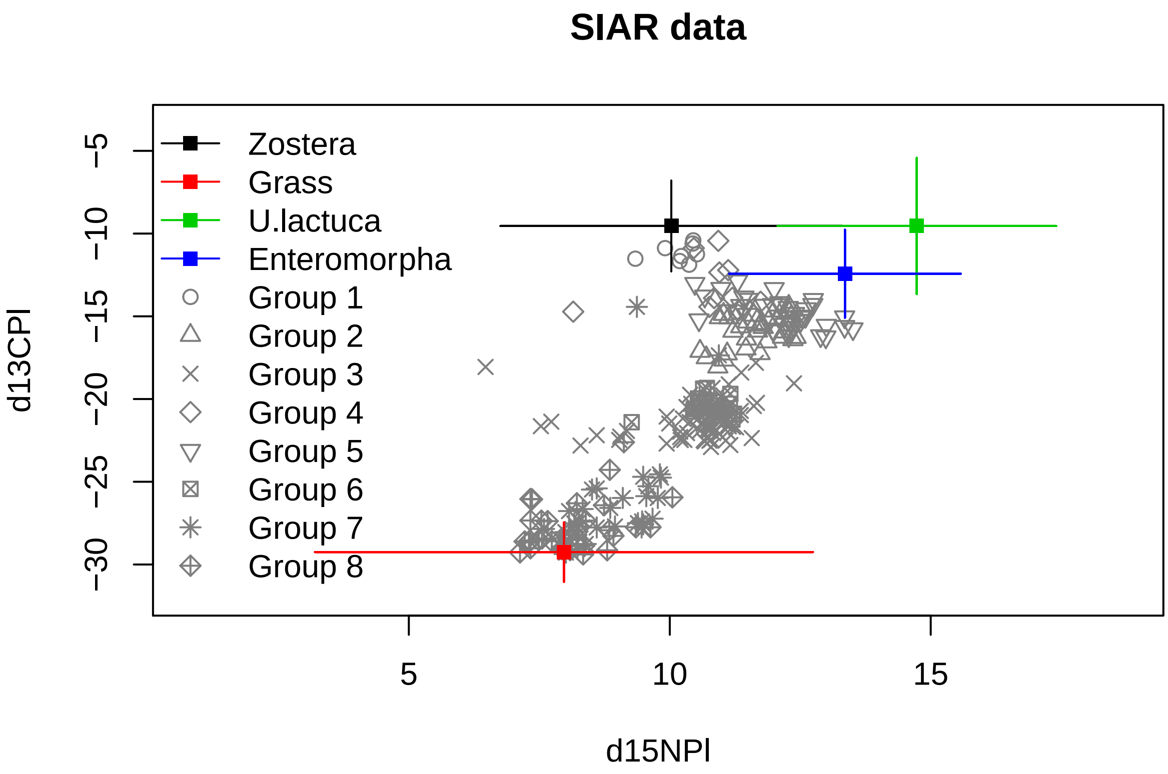
To run SIAR with this data set it's the same as before:

```
out_3 = siarmcmcdirichletv4(geese2demo,sourcesdemo,correctionsdemo)
```

You'll see lots of output this time as SIAR is running on each group in turn. It shouldn't take very long though.

You can now get further analysis using many of the same commands as before. A isospace plot is obtained with:

```
siarplotdata(out_3)
```



The HDRs and convergence diagnostics are created with:

```
siarhdrs(out_3)
```

We can get some within-group boxplots with:

```
siarproportionbygroupplot(out_3)
```

and proportions by source with

```
siarproportionbysourceplot(out_3)
```

Finally, the matrix plot (discussed in the module) can be created with:

```
siarmatrixplot(out_3)
```

This is a really useful plot as it provides the histograms and the relationships between the sources, potentially identifying which sources are impossible to discern between in the model. It takes a little bit of practice to interpret a matrix plot.

Running the model for individual observations

When you have just a single observation it is impossible to estimate the residual standard deviation. However you can still estimate the dietary proportions and SIAR has a special function for this, called `siarsolomcmcv4`. We can create a single sample by just taking a row from the geese data:

```
geese2demo_1row = as.matrix(geese2demo[50,2:3])
out_4 = siarsolomcmcv4(geese2demo_1row,sourcesdemo,correctionsdemo,concdepdemo)
```

```
siarhdrs(out_4)
```

```
## Summary information for the output file ...
##              Low 95% hdr High 95% hdr         mode        mean
## Zostera        0.9806504   0.99885714 0.9940059976 0.990588150
## Grass          0.0000000   0.00440299 0.0003327770 0.001453835
## U.lactuca      0.0000000   0.00938947 0.0006319974 0.003508294
## Enteromorpha   0.0000000   0.01151093 0.0008628171 0.004449721
## SD1            0.0000000   0.00000000 0.0000000000 0.000000000
## SD2            0.0000000   0.00000000 0.0000000000 0.000000000
## Ignore SD columns for siarsolo runs.
##
## Running convergence diagnostics on output.
## Output parameters need to have been loaded in or created.
##
## Worst parameters are ...
##    U.lactuca      Zostera Enteromorpha        Grass         <NA>
##  0.001194949  0.143567090  0.147937996  0.308380824           NA
##         <NA>
##           NA
## Ignore NAs for siar solo runs.
##
## If lots of the p-values are very small, try a longer run of the MCMC.
```

Adding in your own prior information

Occasionally it is the case that previous studies have given insight into the likely values of the dietary proportions for your study. You can use this external information to guide the model by changing the prior distributions used for the Dirichlet distribution (see lecture: 'The statistical model behind SIAR'). If prior information is available, it is usually a good idea to use it, as it means the model will often converge quicker, and yield more realistic results.

SIAR has a function for the inclusion of new Dirichlet parameters (the default is to set all the αs to 1) called `siarelicit`. To use the function you have to follow these steps:

1. Run your analysis as normal with the default SIAR settings
2. Run, e.g. `siarelicit(out)` where `out` is the name of the model run
3. Put in your best guess as to the mean proportions for each group, separated by a space
4. Choose a particular source to set the standard deviation
5. Provide that standard deviation
6. Re-run SIAR with the new α values in the `prior` argument of e.g. `siarmcmcdirichletv4`

The reason for the weird set up (i.e. having to run the model first and then giving a standard deviation for only one source) is because (a) it is easier for the model to know what the data look like before it runs the elicitation step, and (b) because the Dirichlet distribution has a restricted variance (see here for details) which means that, given one of the source standard deviations, all the others are defined.

As an example, let's suppose a previous study for the Geese data had estimated that the dietary proportions for Zostera, Grass, Ulva Lactuca and Enteromorpha respectively are 0.7, 0.1, 0.15 and 0.05, and that the standard deviation of Zostera was 0.05. When we run this through `siarelicit` we get the new α values as 58.1, 8.3, 12.45, and 4.15. We now re-run SIAR with:

```
out_5 = siarmcmcdirichletv4(geese1demo,sourcesdemo,correctionsdemo,concdepdemo,
                            prior=c(58.1,8.3,12.45,4.15))
```

---

1. Try several different prior structures and assess how they change the posterior dietary proportions with `siarhdrs`.
2. An alternative default prior for SIAR would be when all of the α values are set to 1 divided by the number of sources. This is known as the *Jeffreys prior* or *reference prior* and is used because it's often very stable (see information here). What effect does using the Jeffreys prior have on the different Geese data sets?

Creating your own plots and tables

Often what you want to create isn't exactly part of the SIAR toolkit. Maybe the plots don't look right, or maybe you want to compare two different groups in a particular way. To do this, you can get at the SIAR output yourself, and then play with it as you want.

Whenever SIAR creates the dietary proportions using e.g. `siarmcmcdirichletv4`, it stores the output as an R *list*. You can see everything in the list with:

```
str(out)
```

```
## List of 15
##  $ targets    : num [1:9, 1:2] 10.2 10.4 10.4 10.5 10.2 ...
##   ..- attr(*, "dimnames")=List of 2
##   .. ..$ : NULL
##   .. ..$ : chr [1:2] "d15NPl" "d13CPl"
##  $ sources    :'data.frame': 4 obs. of  5 variables:
##   ..$ Sources : Factor w/ 4 levels "Enteromorpha",..: 4 2 3 1
##   ..$ Meand15N: num [1:4] 6.49 4.43 11.19 9.82
##   ..$ SDd15N  : num [1:4] 1.459 2.268 1.112 0.827
##   ..$ Meand13C: num [1:4] -11.2 -30.9 -11.2 -14.1
##   ..$ SDd13C  : num [1:4] 1.215 0.641 1.959 1.172
##  $ corrections:'data.frame': 4 obs. of  5 variables:
##   ..$ Source : Factor w/ 4 levels "Enteromorpha",..: 4 2 3 1
##   ..$ Mean15N: num [1:4] 3.54 3.54 3.54 3.54
##   ..$ Sd15N  : num [1:4] 0.74 0.74 0.74 0.74
##   ..$ Mean13C: num [1:4] 1.63 1.63 1.63 1.63
##   ..$ Sd13C  : num [1:4] 0.63 0.63 0.63 0.63
##  $ concdep    :'data.frame': 4 obs. of  5 variables:
##   ..$ Sources : Factor w/ 4 levels "Enteromorpha",..: 4 2 3 1
##   ..$ Meand15N: num [1:4] 0.0297 0.0355 0.0192 0.0139
##   ..$ SDd15N  : num [1:4] 0.0097 0.0063 0.0053 0.0057
##   ..$ Meand13C: num [1:4] 0.359 0.403 0.21 0.184
##   ..$ SDd13C  : num [1:4] 0.0561 0.038 0.0327 0.1131
##  $ PATH       : NULL
##  $ TITLE      : chr "SIAR data"
##  $ numgroups  : int 1
##  $ numdata    : int 9
##  $ numsources : int 4
##  $ numiso     : int 2
##  $ SHOULDRUN  : logi TRUE
##  $ GRAPHSONLY : logi FALSE
##  $ EXIT       : logi FALSE
##  $ SIARSOLO   : logi FALSE
##  $ output     : num [1:10000, 1:6] 0.504 0.5 0.488 0.414 0.444 ...
##   ..- attr(*, "dimnames")=List of 2
##   .. ..$ : NULL
##   .. ..$ : chr [1:6] "Zostera" "Grass" "U.lactuca" "Enteromorpha" ...
```

This will provide quite a lot of output, but the most important part is the last element, named `output` which contains all of the posterior samples. You can see the first few with:

```
head(out$output)
```

```
##        Zostera      Grass  U.lactuca Enteromorpha       SD1       SD2
## [1,] 0.5044867 0.06331611 0.02264816    0.4095491 0.3354450 1.4351786
## [2,] 0.4998669 0.04403443 0.04546659    0.4106321 0.5550515 1.1318135
## [3,] 0.4877621 0.03323612 0.05653415    0.4224676 0.6164409 0.4281673
## [4,] 0.4143691 0.04320018 0.04257583    0.4998549 0.5145369 0.4373142
## [5,] 0.4441253 0.06176124 0.02779245    0.4663210 0.4575578 1.3982462
## [6,] 0.4162686 0.06269498 0.02729998    0.4937364 0.1187694 0.6313341
```

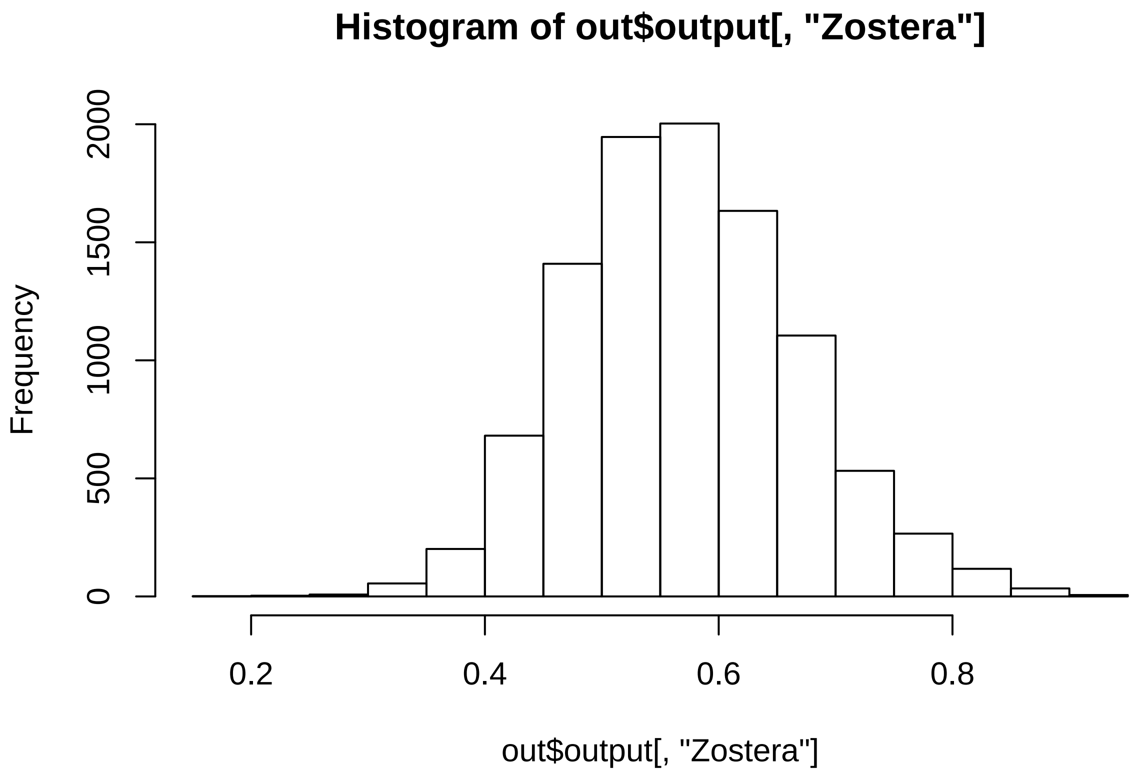
You will see that, for each row, each of the four sources sum to 1:

```
sum(out$output[1,1:4])
```

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

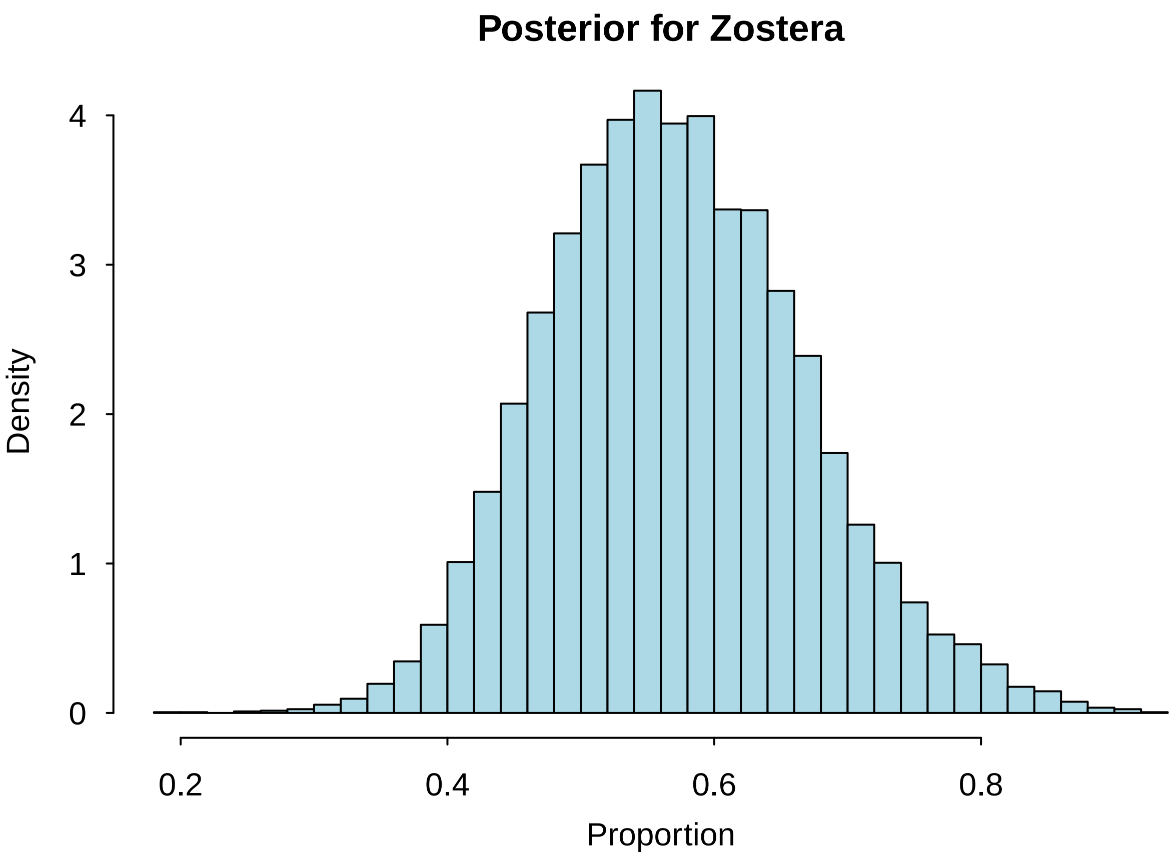
You can now create any further that you wish, for example a simple histogram of the posterior proportion of Zostera:

```
hist(out$output[,'Zostera'])
```



This is a bit crude, but with some extra options, you can make this look quite neat:

```
# Set some better options for graphs
par(mar=c(3,3,2,1), mgp=c(2,.7,0), tck=-.01,las=1)
hist(out$output[,'Zostera'],freq=FALSE,main='Posterior for Zostera',
     xlab='Proportion',col='lightblue',breaks=30)
```



You can also create your own output analysis. For example, what is the 90% credible interval for Grass?

```
quantile(out$output[,'Grass'],probs=c(0.05,0.95))
```

```
##         5%        95% 
## 0.03464589 0.10653984
```

What is the probability that the consumers ate more Ulva Lactuca than Enteromorpha?

```
sum(out$output[,'U.lactuca']>out$output[,'Enteromorpha'])/nrow(out$output)
```

```
## [1] 0.3379
```

The above counts the number of rows (i.e. iterations) in the output where Ulva Lactuca is higher than Enteromorpha and divides this by the total number of rows.

Finally, if you want to see what SIAR is doing behind the scenes, simply type the name of the function without brackets, for example

```
siarplotdata
```

If the SIAR plot or table doesn't exactly match what you want you can create your own function based on the original one which includes everything you need.

---

1. Try accessing the output from the second Geese data set (stored above in `out_3`). Try to re-create the above histograms for some of the groups.
2. Continuing the above, try and calculate the probability that one group ate more of a certain source than another.

Some extra tasks

If you finish all the above and want some further tasks to complete try these.

---

1. See if you can re-create the iso-space plot from the raw data from scratch. Refer back to the code in `siarplotdata` if you need to.
2. Try and write your own function to process the output from a SIAR model run. What would you like to include? Below is a function which just lists the first 15 iterations. You could create something far richer, including means (via `mean`), credible intervals (via `quantile`), correlations (via `cor`) or plots. The `apply` function is often useful here as it will run a function over the rows or columns of a matrix.

```r
my_summary = function(x) {
  head(x$output,15)
}
my_summary(out)
```

```
##         Zostera      Grass   U.lactuca Enteromorpha       SD1       SD2
##  [1,] 0.5044867 0.06331611 0.022648161    0.4095491 0.3354450 1.4351786
##  [2,] 0.4998669 0.04403443 0.045466591    0.4106321 0.5550515 1.1318135
##  [3,] 0.4877621 0.03323612 0.056534145    0.4224676 0.6164409 0.4281673
##  [4,] 0.4143691 0.04320018 0.042575825    0.4998549 0.5145369 0.4373142
##  [5,] 0.4441253 0.06176124 0.027792450    0.4663210 0.4575578 1.3982462
##  [6,] 0.4162686 0.06269498 0.027299976    0.4937364 0.1187694 0.6313341
##  [7,] 0.4525692 0.02717471 0.010415134    0.5098410 0.1692382 0.3017963
##  [8,] 0.4417511 0.04190042 0.007648876    0.5086996 1.1306021 1.3531995
##  [9,] 0.4509466 0.04022205 0.090200640    0.4186307 0.4096391 0.9634585
## [10,] 0.4434527 0.03633608 0.110432677    0.4097786 0.3406712 1.0025829
## [11,] 0.4024402 0.07108709 0.093528569    0.4329441 0.4934100 2.9863012
## [12,] 0.3899746 0.03085801 0.079430923    0.4997365 1.0169610 1.0154053
## [13,] 0.4176567 0.06754881 0.084369324    0.4304251 0.5880931 1.5251634
## [14,] 0.4155890 0.08740525 0.043286990    0.4537188 0.2536194 1.4840052
## [15,] 0.3747017 0.03539885 0.045535874    0.5443636 0.1246442 1.1273576
```
