Simulated example: Estimating diet proportions form fatty acids and stable isotopes

Philipp Neubauer

April 17, 2014

Preamble

DISCLAIMER: This is an evolving package and vignette, please read instructions here for installation and to see what dependencies are required. If you find a bug or want so suggest improvements, please submit an issue, collaborations and contributions are very welcome!

DISCLAIMER 2: This tutorial is by no means a real analysis, which should proceed more carefully, and with longer MCMC runs. The example is designed to run through in a few minutes, and is therefore only an illustration of how an analysis of real data could proceed.

```
library(fastinR, warn.conflicts = F)

## Loading required package:  rjags
## Loading required package:  coda
## Loading required package:  lattice
## Linked to JAGS 3.4.0
## Loaded modules:  basemod,bugs
```

Introduction

We start with a very simple simulated example. It is possible to simulate relevant data using the built in simulation GUI, which is called from the command line once fastinR has been loaded. Plese type ?simulation to obtain help on simulating your own 'dataset'. Note, however, that the recommended way to interact with the package is the command line - the GUI is somewhat unstable and has a mind of its own (see warnings in gui help files).

The package comes with a simulated dataset which can be called with , which loads an object called *datas* into the current environment.

Loading data from files

Raw data should be stored in .csv files in a prescribed format, see *add_FA*, *add_SI* and *add_Covs* for details on formatting. You can also inspect saved output from *simulation* to get a better idea of the correct file format.

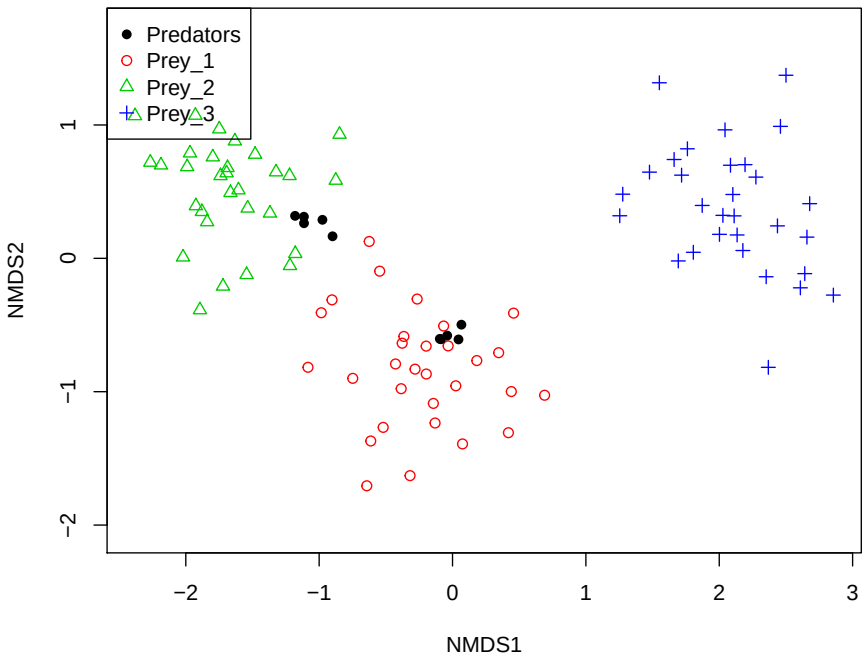
Files are imported with the help of three constructor functions: *add_FA*to import fatty acid data, *add_SI* to import Stable Isotope data and *add_Covs* to import data on covariates or groupings that may be influencing predator diets.

The package comes with simulated data in .csv files, which can be found using the *system.file* function. The *add_SI* constructor takes separate files for predator and prey Stable Isotope data, as well as files for additive fractionation coefficient means and variances - these are optional and can be specified manually in the function call, see the function help for details.

```
# these commands just read data supplied with the package. For a real analysis these would
# be substituted with paths to data files, or with R data objects.
SI.predators <- system.file("extdata", "Simdata_SI_preds.csv", package = "fastinR")
SI.preys <- system.file("extdata", "Simdata_SI_preys.csv", package = "fastinR")
Frac.Coeffs.mean <- system.file("extdata", "Simdata_SI_fc_means.csv", package = "fastinR")
Frac.Coeffs.var <- system.file("extdata", "Simdata_SI_fc_var.csv", package = "fastinR")

dats <- add_SI(SI.predators, SI.preys, Frac.Coeffs.mean, Frac.Coeffs.var)
```

We can now visualize the data on an Non-Metric Dimensional Scaling plot like so:

```
dataplot(dats)

## Run 0 stress 0
## Run 1 stress 8.85e-05
## ... procrustes: rmse 8.802e-05  max resid 0.0002831
## *** Solution reached

## Warning:  Stress is (nearly) zero - you may have insufficient data
## Warning:  Species scores not available
```



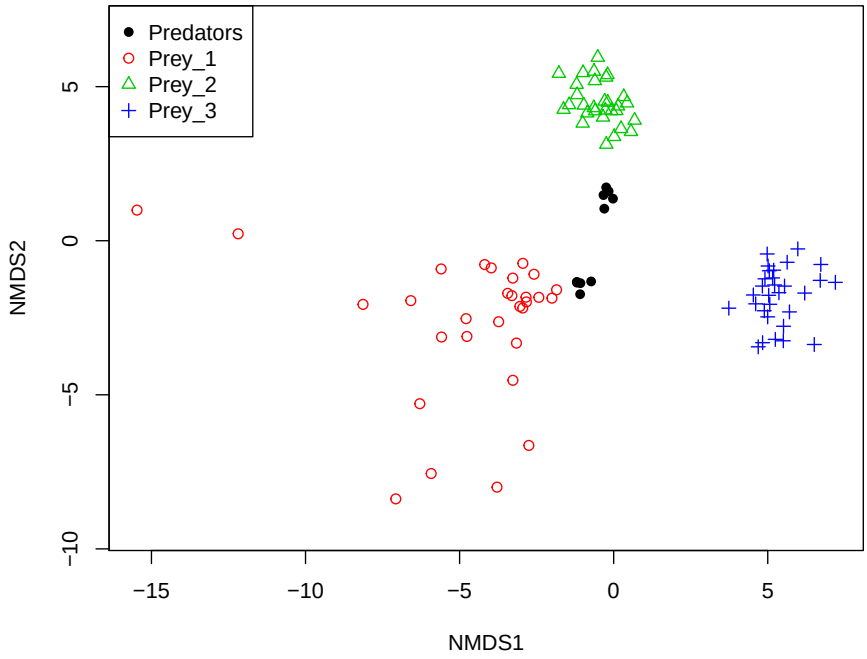
The dats object now has a set of data in the right format for further analysis. We'll add fatty acids before we proceed. As with *add_SI*, the *add_FA* constructor takes separate files for predator and prey fatty acid data, conversion coefficient means and variances as well as fat content - these are again optional and can be specified manually in the function call, see the function help for details.

```
FAP.predators <- system.file("extdata", "Simdata_FA_preds.csv", package = "fastinR")
FAP.preys <- system.file("extdata", "Simdata_FA_preys.csv", package = "fastinR")
Conv.Coeffs.mean <- system.file("extdata", "Simdata_FA_cc_means.csv", package = "fastinR")
Conv.Coeffs.var <- system.file("extdata", "Simdata_FA_cc_var.csv", package = "fastinR")
fat.conts <- system.file("extdata", "Simdata_fat_cont.csv", package = "fastinR")

dats <- add_FA(FAP.predators, FAP.preys, fat.conts, Conv.Coeffs.mean, Conv.Coeffs.var, datas
```

Note the last argument in *add_FA* now contains a reference to the data object that was constructed from the Stable Isotopes beforehand. *add_FA* just adds fatty acid data to the mix - the same would apply if data were added the other way around (only *add_Covs* works separately, it needs it's own object).

Plotting the joint dataset:

```
dataplot(dats)

## Run 0 stress 0.06012
## Run 1 stress 0.06012
## ... New best solution
## ... procrustes: rmse 6.826e-05  max resid 0.0006405
## *** Solution reached

## Warning:  Species scores not available
```



1 Data Grooming

The grooming step in this case corresponds to selecting relevant variables for the analysis. In practice, one could measure lots of Fatty Acids, and researchers often choose arbitrary cutoff points in proportions to reduce the dataset. However, even Fatty Acids occurring in low proportions may be useful for source discrimination and diet estimation if they introduce systematic differences between sources (prey items). The *select_vars* function provides a graphical way to choose a subset of Fatty Acids according to their contribution to prey separation and reduction in collinearity in the prey matrix. Using *select_vars* in the

simulated example shows that only really 2 Fatty Acids contribute to source separation - the cumulative separation displayed in the console converges to one after adding the two most important fatty acid proportions: all remaining FAs only add to collinearity in the source matrix. We choose Fatty Acid no 3,2 and 6 for analysis (the second call does this directly without prompting the user, by giving the index of the fatty acids to choose).

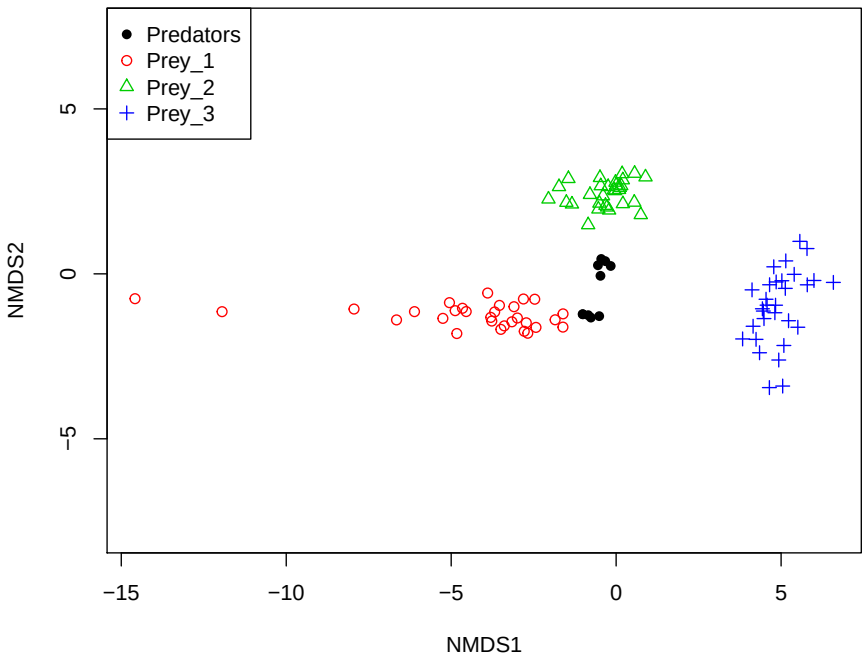
```
dats.subset <- select_vars(dats, plot = F)
```

or

```
dats.subset <- select_vars(dats, c(3, 2, 6))

# inspecting
dats.subset$datas.FA$n.fats

## [1] 3
```

The data object *dats* is now ready for analysis, we can plot the dataset with it's new subset of fatty acids as before:

```
dataplot(dats.subset)

## Run 0 stress 0.02072
## Run 1 stress 0.02072
## ... New best solution
## ... procrustes: rmse 0.001177  max resid 0.011
## Run 2 stress 0.02072
## ... procrustes: rmse 0.001205  max resid 0.01128
## Run 3 stress 0.02072
## ... procrustes: rmse 0.001092  max resid 0.01013
## Run 4 stress 0.02072
## ... procrustes: rmse 0.00122  max resid 0.01142
## Run 5 stress 0.02072
## ... procrustes: rmse 0.001256  max resid 0.01174
## Run 6 stress 0.02072
## ... procrustes: rmse 4.725e-05  max resid 0.0004399
## *** Solution reached

## Warning:  Species scores not available
```



Apart from a rotation, the overall configuration should be similar.

Bayesian Analysis

The actual analysis usues Markov Chain Monte Carlo to estimate posterior distributions for diet parameters. Estiamtion can be performed locally (in the active R session) or in a distributed way, using as many R sessions as Markov Chains. A good strategy is usually to run a few chains locally for short runs, and then run 3 or more chains in parallel in a distributed way once a satisfactory set of priors, thinning interval etc has been found (setting the number of chains to a maximum of n-1 cores on your CPU is a good idea to not hog ALL resources). The *run_MCMC* functions sets up the MCMC runs and takes *spawn* $= T$ or *spawn* $= F$ as parameter to run chains in R slave processes or locally, respectively.

The Rnot parameter sets the prior scale for the predator logistic normal covariance matrix, and will often determine how well the chain mixes, that is, how well it explores the parameter space, or if it gets stuck in local modes (that can be of low probability). In the latter case, one would see strong autocorrelation in the Markov Chains for individual paramters. Increasing *Rnot* and/or *Rnot_SI* can help, but will make it harder to get precise estimates. There is

thus a tradeoff between accuracy and mixing here, and often one will just have to run an analysis for many iterations and with a large thinning interval (e.g., ¿100k iterations with a thinning interval of ¿100). This should be done after finding reasonable parameters for Rnot on a shorter run, and then letting the final analysis run with long chains.

For a combined analysis of stable isotopes and fatty acids, it is often useful to run the two datatypes separately to assure that good priors can be found for both datasets independently, and then combining them for a final analysis. The analysis type is chosen in the appropriate option in the function call.

Estimating population proportions

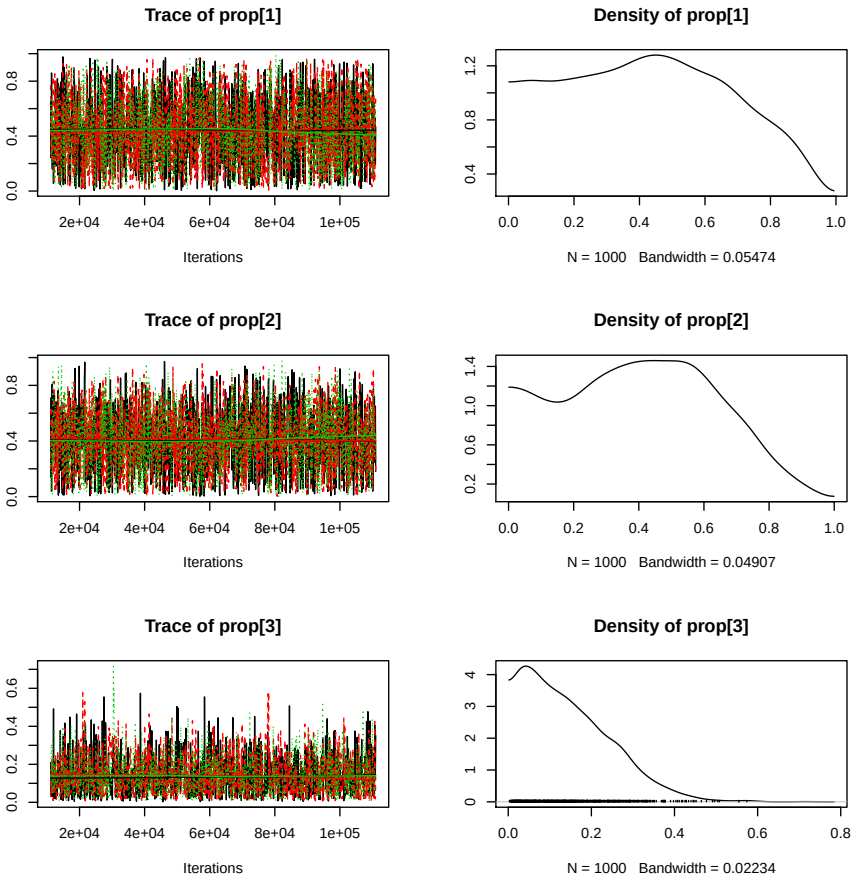
Stable Isotopes alone

Lets start with an analysis of the stable isotopes, estimating only global (population) level diets. We will use the default prior on the predator covariance matrix, and will adjust this prior subsequently. *WARNING* This might take a while depending on your resources, the size of the dataset and the parameters used for the MCMC. If you are not familiar with Markov Chain Monte Carlo, please take a minute to read up on it, there are lots of great explanations and summaries on the web.

```
Pop.SI <- run_MCMC(datas = dats.subset, nIter = 1e+05, nBurnin = 10000, nChains = 3, nThin =
    Data.Type = "Stable.Isotopes", Analysis.Type = "Population.proportions", Rnot_SI = 0.1,
    plott = F, spawn = F)

## Compiling model graph
##    Resolving undeclared variables
##    Allocating nodes
##    Graph Size: 280
##
## Initializing model
##
##
##  proceeding to burn-in phase
##
##  sampling from parameter distributions
```

Plotting the MCMC is the easiest way to ensure that the sampler is mixing - meaning that the chain explores the posterior distribution of each parameter efficiently.

```
MCMCplot(Pop.SI)
```



The mixing isn't great, meaning that some autocorrelation is evident in the Markov Chain time series in left-hand plots. The *diags* function gives more information, displaying two types of diagnostics in the console.:

```
diags(Pop.SI)

##
##
## ####################################
## Raftery-Lewis diagnostics
## ####################################
##
## [[1]]
##
## Quantile (q) = 0.025
## Accuracy (r) = +/- 0.01
## Probability (s) = 0.95
```

```
##
##          Burn-in  Total  Lower bound  Dependence
##          (M)      (N)    (Nmin)       factor (I)
##  prop[1] 200      96900  937          103.0
##  prop[2] 200      89300  937           95.3
##  prop[3] 300      105300 937          112.0
##
##
## [[2]]
##
## Quantile (q) = 0.025
## Accuracy (r) = +/- 0.01
## Probability (s) = 0.95
##
##          Burn-in  Total  Lower bound  Dependence
##          (M)      (N)    (Nmin)       factor (I)
##  prop[1] 200      89300  937           95.3
##  prop[2] 300      114300 937          122.0
##  prop[3] 300      105300 937          112.0
##
##
## [[3]]
##
## Quantile (q) = 0.025
## Accuracy (r) = +/- 0.01
## Probability (s) = 0.95
##
##          Burn-in  Total  Lower bound  Dependence
##          (M)      (N)    (Nmin)       factor (I)
##  prop[1] 300      105300 937          112
##  prop[2] 400      124300 937          133
##  prop[3] 200      96900  937          103
##
##
##
##  Based on these diagnostics you should repeat the MCMC with  124300  iterations
##  and a thinning interval of  133  ,if these values are higher than the values
##  used to produce these diagnostics
##
##
##
## ################################
## Gelman-Rubin diagnostics
## ################################
##
```

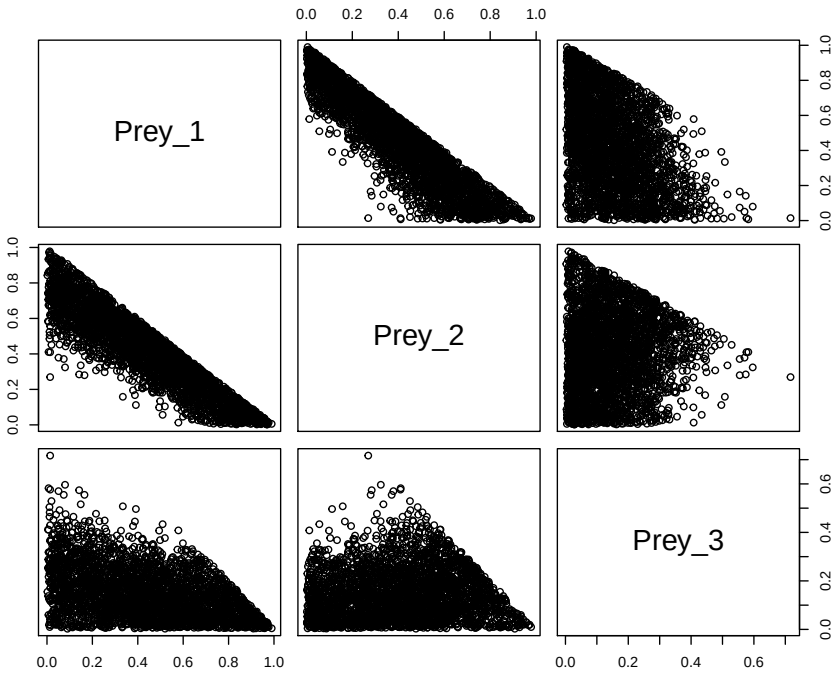
```
## Potential scale reduction factors:
##
##         Point est. Upper C.I.
## prop[1]      0.999          1
## prop[2]      1.000          1
## prop[3]      1.001          1
##
## Multivariate psrf
##
## 1
##
##  Both univariate upper C.I. and multivariate psrf
##  should be close to 1 if the chains converged
##
##
```

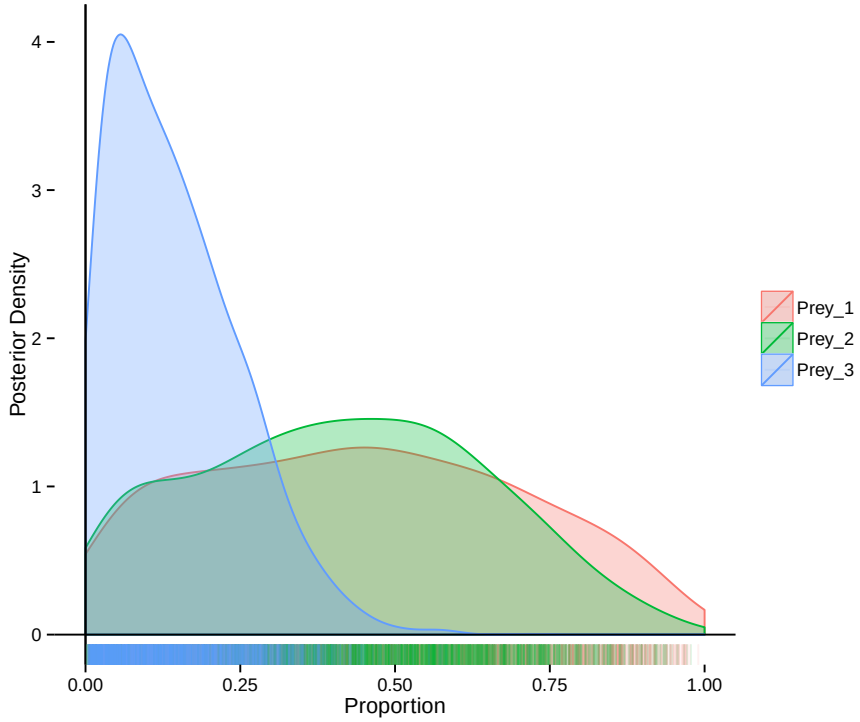
Based on the diagnostics, it seems that we're doing OK, but that the Stable Isotopes don't give much certainty about diet proportions:

```
summary(Pop.SI)

##
##
##  xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
## Printing results for MCMC chain 1
##  xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
##
##
##
##  population diet proportions
##
##
##        Prey_1  Prey_2  Prey_3
## 2.5%  0.03003 0.02365 0.01183
## 50%   0.44405 0.40098 0.12405
## 97.5% 0.90979 0.84995 0.40353
##
##
##  xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
## Printing results for MCMC chain 2
##  xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
##
##
##
##  population diet proportions
##
```

```
##
##        Prey_1  Prey_2  Prey_3
## 2.5%  0.02517 0.02743 0.01039
## 50%   0.43505 0.41745 0.12644
## 97.5% 0.91095 0.82528 0.37885
##
##
##  xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
## Printing results for MCMC chain 3
##  xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
##
##
##
##  population diet proportions
##
##
##        Prey_1  Prey_2  Prey_3
## 2.5%  0.02739 0.01718 0.01043
## 50%   0.43457 0.41102 0.13130
## 97.5% 0.91191 0.84763 0.37726
```

```
plot(Pop.SI, save = F)

## Using  as id variables
```


Correlation of proportion estimates
Prey_1
Prey_2
Prey_3
0.0
0.2
0.4
0.6
0.8
1.0



The credible intervals are very large for Prey items 1 and 2, and the correlation plot suggests that there is a reasonably strong posterior correlation between estimates of prey items 1 & 2.

Increasing the default prior $Rnot_S I$ on the predator covariance matrix sometimes leads to better mixing. We'll also run more iterations and set $spawn = T$ for this.

```
Pop.SI2 <- run_MCMC(datas = dats.subset, nIter = 1e+05, nBurnin = 10000, nChains = 3, nThin
    Data.Type = "Stable.Isotopes", Analysis.Type = "Population.proportions", Rnot_SI = 1, pl
    spawn = T)

## +++++++++++++++++++
```

¡¡MCMC output plot 2,fig.width=7¿= MCMCplot(Pop.SI2) @

```
diags(Pop.SI2)

##
##
## ###################################
## Raftery-Lewis diagnostics
```

```
## ###################################
##
## [[1]]
##
## Quantile (q) = 0.025
## Accuracy (r) = +/- 0.01
## Probability (s) = 0.95
##
##          Burn-in  Total Lower bound  Dependence
##          (M)      (N)   (Nmin)       factor (I)
##  prop[1] 200      89300 937           95.3
##  prop[2] 200      89300 937           95.3
##  prop[3] 200      96900 937          103.0
##
##
## [[2]]
##
## Quantile (q) = 0.025
## Accuracy (r) = +/- 0.01
## Probability (s) = 0.95
##
##          Burn-in  Total Lower bound  Dependence
##          (M)      (N)   (Nmin)       factor (I)
##  prop[1] 200      93300 937           99.6
##  prop[2] 200      89300 937           95.3
##  prop[3] 200      96900 937          103.0
##
##
## [[3]]
##
## Quantile (q) = 0.025
## Accuracy (r) = +/- 0.01
## Probability (s) = 0.95
##
##          Burn-in  Total Lower bound  Dependence
##          (M)      (N)   (Nmin)       factor (I)
##  prop[1] 200      96900 937          103.0
##  prop[2] 200      89300 937           95.3
##  prop[3] 200      89300 937           95.3
##
##
##
##  Based on these diagnostics you should repeat the MCMC with  96900  iterations
##  and a thinning interval of  103  ,if these values are higher than the values
##  used to produce these diagnostics
```

```
##
##
##
## ################################
## Gelman-Rubin diagnostics
## ################################
##
## Potential scale reduction factors:
##
##         Point est. Upper C.I.
## prop[1]          1       1.00
## prop[2]          1       1.02
## prop[3]          1       1.00
##
## Multivariate psrf
##
## 1.01
##
##  Both univariate upper C.I. and multivariate psrf
##  should be close to 1 if the chains converged
##
##
```

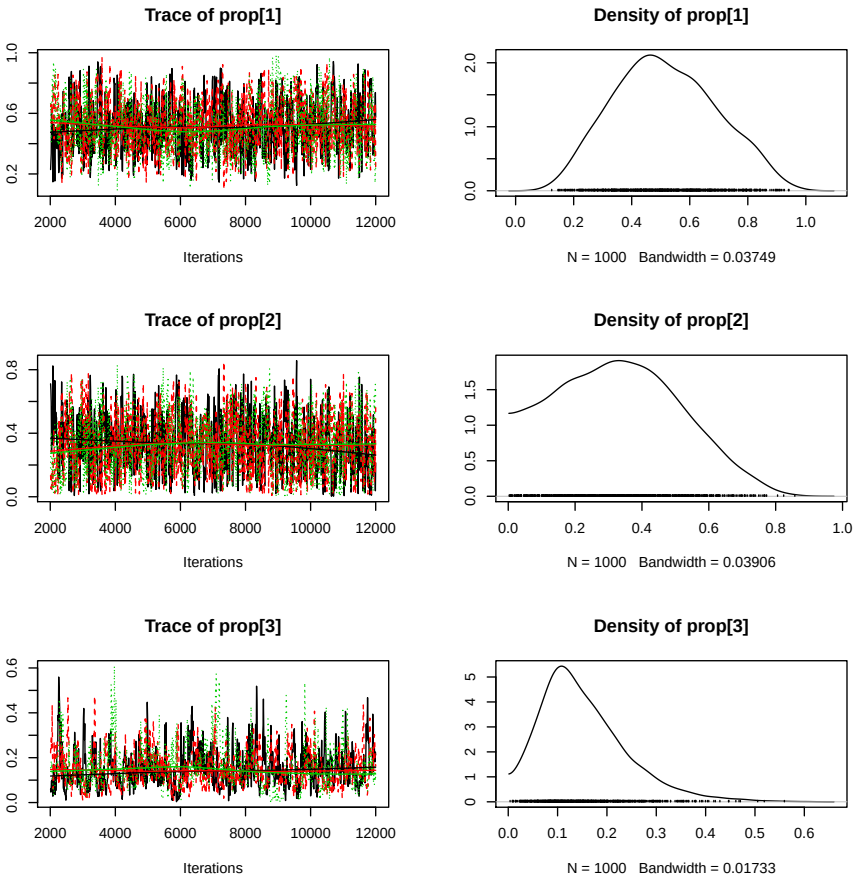
Mixing doesn't seem to get much better, indicating that this is probably as good as it gets with stable isotopes alone.

1.0.1 Fatty Acids alone

We repeat this with Fatty Acids, again starting with the default prior. Note that $spawn = T$ this time to save some time.

```
Pop.FA <- run_MCMC(datas = dats.subset, nIter = 10000, nBurnin = 1000, nChains = 3, nThin =
    Data.Type = "Fatty.Acid.Profiles", Analysis.Type = "Population.proportions", Rnot = 0.2,
    plott = F, spawn = T)

## +++++++++
```

```
MCMCplot(Pop.FA)
```



Once again the mixing isn't great:

```
diags(Pop.FA)

##
##
## ####################################
## Raftery-Lewis diagnostics
## ####################################
##
## [[1]]
##
## Quantile (q) = 0.025
## Accuracy (r) = +/- 0.01
## Probability (s) = 0.95
##
##          Burn-in  Total Lower bound  Dependence
```

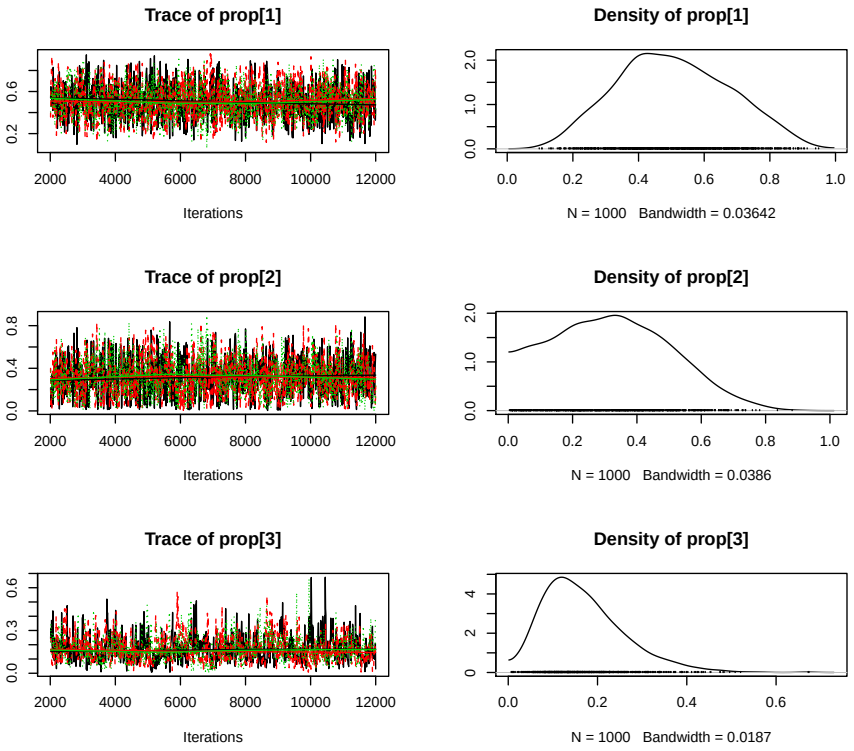
```
##          (M)      (N)   (Nmin)       factor (I)
##  prop[1] 30       11430 937          12.2
##  prop[2] 30       11430 937          12.2
##  prop[3] 80       21170 937          22.6
##
##
## [[2]]
##
## Quantile (q) = 0.025
## Accuracy (r) = +/- 0.01
## Probability (s) = 0.95
##
##          Burn-in  Total Lower bound  Dependence
##          (M)      (N)   (Nmin)       factor (I)
##  prop[1] 30       11430 937          12.2
##  prop[2] 50       13530 937          14.4
##  prop[3] 90       23360 937          24.9
##
##
## [[3]]
##
## Quantile (q) = 0.025
## Accuracy (r) = +/- 0.01
## Probability (s) = 0.95
##
##          Burn-in  Total Lower bound  Dependence
##          (M)      (N)   (Nmin)       factor (I)
##  prop[1] 30       10530 937          11.2
##  prop[2] 30       11430 937          12.2
##  prop[3] 240      74320 937          79.3
##
##
##
##  Based on these diagnostics you should repeat the MCMC with  74320  iterations
##  and a thinning interval of  79  ,if these values are higher than the values
##  used to produce these diagnostics
##
##
##
## ###################################
## Gelman-Rubin diagnostics
## ###################################
##
## Potential scale reduction factors:
##
```

```
##          Point est. Upper C.I.
## prop[1]       1.01       1.01
## prop[2]       1.00       1.00
## prop[3]       1.04       1.06
##
## Multivariate psrf
##
## 1.02
##
##  Both univariate upper C.I. and multivariate psrf
##  should be close to 1 if the chains converged
##
##
```

Again not too bad according to the diagnostics, although the chains should be run for longer (see Raftery-Lewis diagnostics) - but lets try again with

```
Pop.FA2 <- run_MCMC(datas = dats.subset, nIter = 10000, nBurnin = 1000, nChains = 3, nThin =
    Data.Type = "Fatty.Acid.Profiles", Analysis.Type = "Population.proportions", Rnot = 1,
    plott = F, spawn = T)

## ++++++++
```

```
MCMCplot(Pop.FA2)
```




```
diags(Pop.FA2)

##
##
## ###################################
## Raftery-Lewis diagnostics
## ###################################
##
## [[1]]
##
## Quantile (q) = 0.025
## Accuracy (r) = +/- 0.01
## Probability (s) = 0.95
##
##          Burn-in  Total Lower bound  Dependence
##          (M)      (N)   (Nmin)       factor (I)
##  prop[1] 20       8930  937           9.53
##  prop[2] 30       10530 937          11.20
##  prop[3] 50       13890 937          14.80
##
```

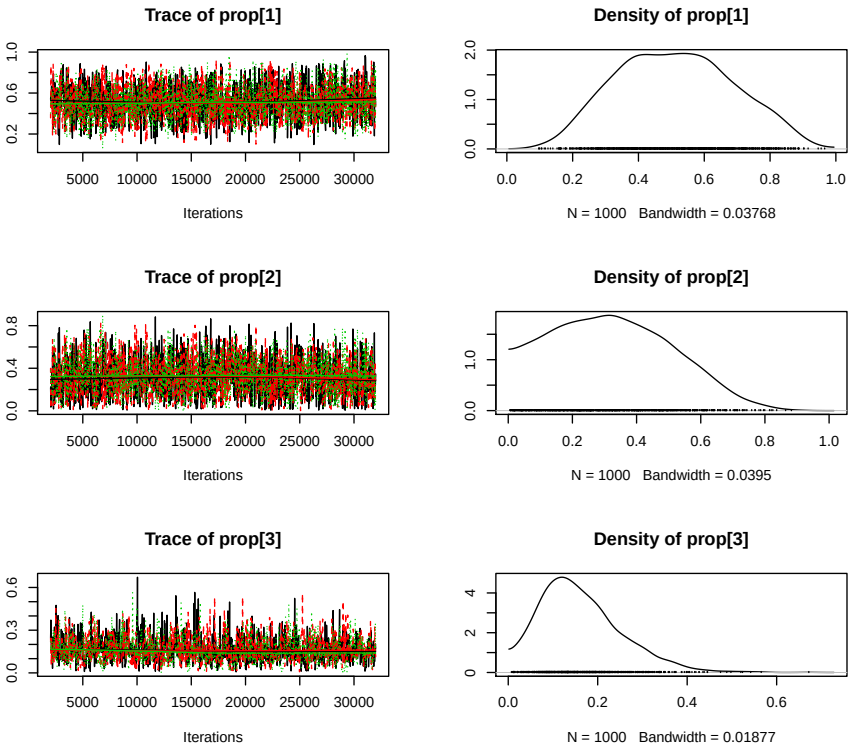
```
##
## [[2]]
##
## Quantile (q) = 0.025
## Accuracy (r) = +/- 0.01
## Probability (s) = 0.95
##
##          Burn-in  Total Lower bound  Dependence
##          (M)      (N)   (Nmin)       factor (I)
##  prop[1] 30       11430 937          12.2
##  prop[2] 30       10530 937          11.2
##  prop[3] 50       13530 937          14.4
##
##
## [[3]]
##
## Quantile (q) = 0.025
## Accuracy (r) = +/- 0.01
## Probability (s) = 0.95
##
##          Burn-in  Total Lower bound  Dependence
##          (M)      (N)   (Nmin)       factor (I)
##  prop[1] 30       10530 937          11.2
##  prop[2] 40       12430 937          13.3
##  prop[3] 50       14730 937          15.7
##
##
##
##  Based on these diagnostics you should repeat the MCMC with  14730  iterations
##  and a thinning interval of  16  ,if these values are higher than the values
##  used to produce these diagnostics
##
##
##
## ################################
## Gelman-Rubin diagnostics
## ################################
##
## Potential scale reduction factors:
##
##         Point est. Upper C.I.
## prop[1]       1.01       1.02
## prop[2]       1.00       1.00
## prop[3]       1.01       1.03
##
```

```
## Multivariate psrf
##
## 1.01
##
##  Both univariate upper C.I. and multivariate psrf
##  should be close to 1 if the chains converged
##
##
```

Looks very similar, so lets do a final run with 30k iterations as suggested by the *diags* output and a thinning interval of 30 should be better. (Note that values suggested by the diagnostics may vary from one run to the next. Also note that we already ran 30k as we ran 3 parallel chains here...).

```
Pop.FA3 <- run_MCMC(datas = dats.subset, nIter = 30000, nBurnin = 1000, nChains = 3, nThin =
    Data.Type = "Fatty.Acid.Profiles", Analysis.Type = "Population.proportions", Rnot = 1,
    plott = F, spawn = T)

## ++++++++++++++++++++
```

```
MCMCplot(Pop.FA3)
```




```
diags(Pop.FA3)

##
##
## ###################################
## Raftery-Lewis diagnostics
## ###################################
##
## [[1]]
##
## Quantile (q) = 0.025
## Accuracy (r) = +/- 0.01
## Probability (s) = 0.95
##
##          Burn-in  Total Lower bound  Dependence
##          (M)      (N)   (Nmin)       factor (I)
##  prop[1] 60       29070 937          31.0
##  prop[2] 60       26790 937          28.6
##  prop[3] 90       31590 937          33.7
##
```

```
##
## [[2]]
##
## Quantile (q) = 0.025
## Accuracy (r) = +/- 0.01
## Probability (s) = 0.95
##
##          Burn-in  Total Lower bound  Dependence
##          (M)      (N)   (Nmin)       factor (I)
##  prop[1] 90       31590 937          33.7
##  prop[2] 90       34290 937          36.6
##  prop[3] 90       34290 937          36.6
##
##
## [[3]]
##
## Quantile (q) = 0.025
## Accuracy (r) = +/- 0.01
## Probability (s) = 0.95
##
##          Burn-in  Total Lower bound  Dependence
##          (M)      (N)   (Nmin)       factor (I)
##  prop[1] 60       29070 937          31.0
##  prop[2] 60       29070 937          31.0
##  prop[3] 300      87000 937          92.8
##
##
##
##  Based on these diagnostics you should repeat the MCMC with  87000  iterations
##  and a thinning interval of  93  ,if these values are higher than the values
##  used to produce these diagnostics
##
##
##
## ###################################
## Gelman-Rubin diagnostics
## ###################################
##
## Potential scale reduction factors:
##
##         Point est. Upper C.I.
## prop[1]       1.00       1.00
## prop[2]       1.00       1.01
## prop[3]       1.03       1.07
##
```

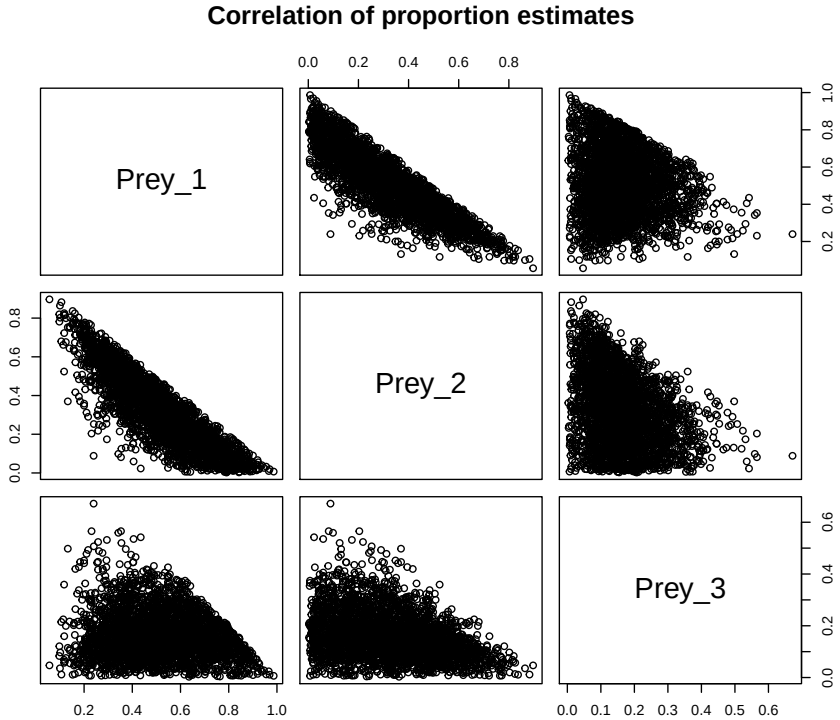
```
## Multivariate psrf
##
## 1.02
##
##  Both univariate upper C.I. and multivariate psrf
##  should be close to 1 if the chains converged
##
##
```

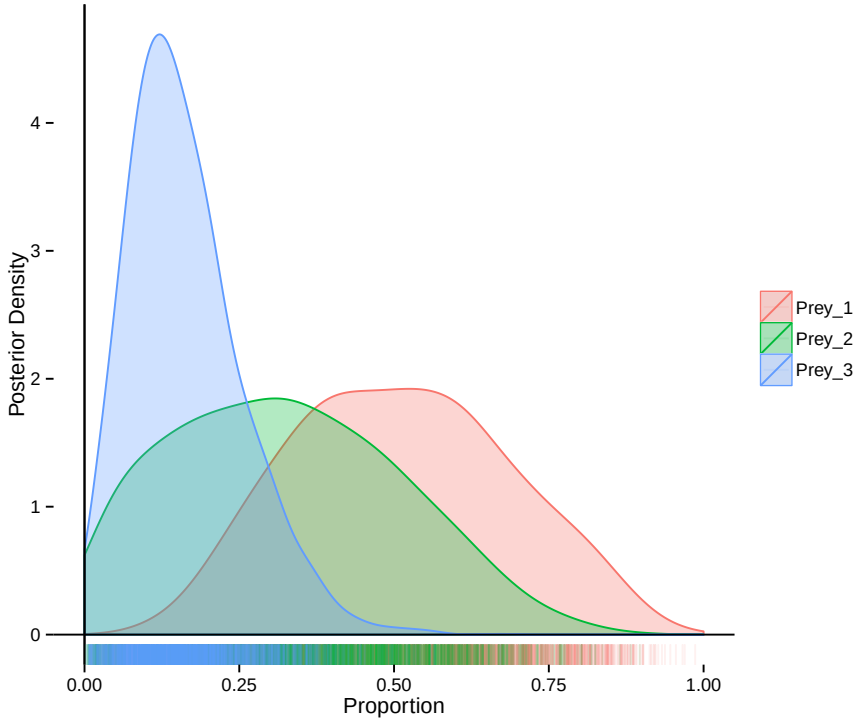
```
summary(Pop.FA3)
```

```
##
##
##  xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
## Printing results for MCMC chain 1
##  xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
##
##
##
##  population diet proportions
##
##
##       Prey_1  Prey_2 Prey_3
## 2.5%  0.1817 0.02067 0.0290
## 50%   0.5209 0.29861 0.1515
## 97.5% 0.8459 0.70741 0.3870
##
##
##  xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
## Printing results for MCMC chain 2
##  xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
##
##
##
##  population diet proportions
##
##
##       Prey_1  Prey_2  Prey_3
## 2.5%  0.2053 0.02463 0.02769
## 50%   0.5084 0.32046 0.15383
## 97.5% 0.8453 0.67634 0.37002
##
##
##  xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
## Printing results for MCMC chain 3
##  xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
```

```
##
##
##
##  population diet proportions
##
##
##         Prey_1  Prey_2  Prey_3
## 2.5%   0.2058 0.02624 0.01905
## 50%    0.4977 0.32201 0.14011
## 97.5% 0.8529 0.68709 0.36649
```

```
plot(Pop.FA3, save = F)

## Using  as id variables
```



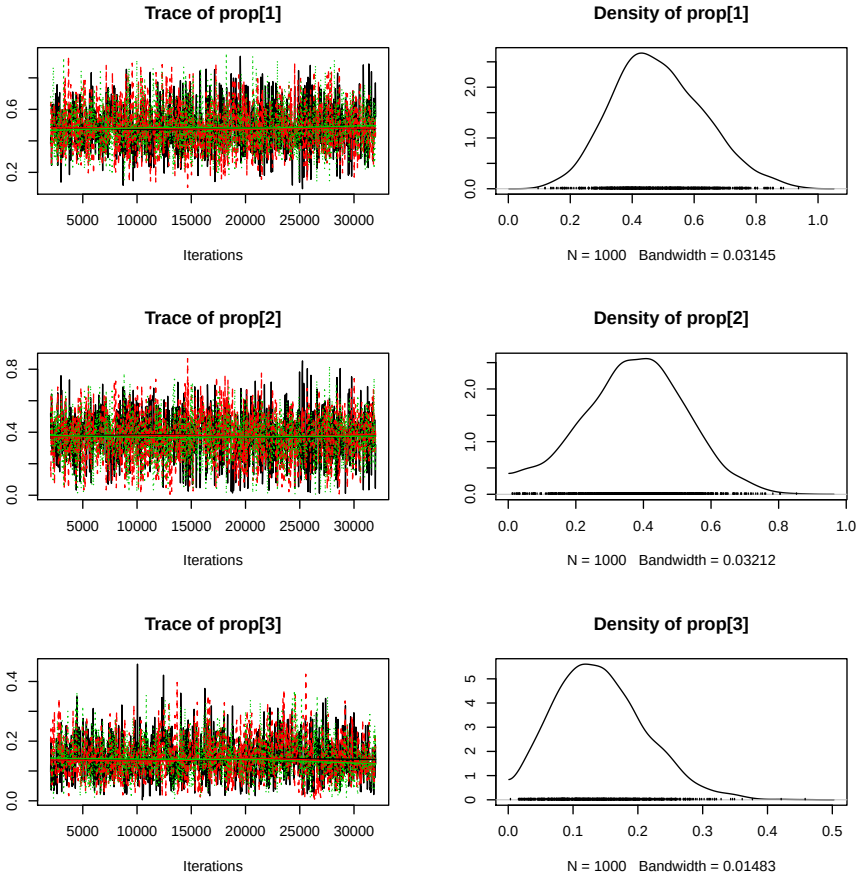


There is still substantial uncertainty about the diet proportions: it looks as though Prey 1 is a dominant source, but the credible intervals are large. Let's try combining stable isotopes and fatty acids:

1.0.2 Combining fatty acids and stable isotopes

```
Pop.Combined <- run_MCMC(datas = dats.subset, nIter = 30000, nBurnin = 1000, nChains = 3, nT
    Data.Type = "Combined.Analysis", Analysis.Type = "Population.proportions", Rnot = 1, Rno
    plott = F, spawn = T)

## ++++++++++++++++++++++++++
```

```
MCMCplot(Pop.Combined)
```




```
diags(Pop.Combined)

##
##
## ###################################
## Raftery-Lewis diagnostics
## ###################################
##
## [[1]]
##
## Quantile (q) = 0.025
## Accuracy (r) = +/- 0.01
## Probability (s) = 0.95
##
##          Burn-in  Total Lower bound  Dependence
##          (M)      (N)   (Nmin)       factor (I)
```

```
##  prop[1] 60       29070 937          31.0
##  prop[2] 90       31590 937          33.7
##  prop[3] 150      40590 937          43.3
##
##
## [[2]]
##
## Quantile (q) = 0.025
## Accuracy (r) = +/- 0.01
## Probability (s) = 0.95
##
##          Burn-in  Total Lower bound  Dependence
##          (M)      (N)   (Nmin)       factor (I)
##  prop[1] 60       29070 937          31.0
##  prop[2] 60       29070 937          31.0
##  prop[3] 210      57780 937          61.7
##
##
## [[3]]
##
## Quantile (q) = 0.025
## Accuracy (r) = +/- 0.01
## Probability (s) = 0.95
##
##          Burn-in  Total Lower bound  Dependence
##          (M)      (N)   (Nmin)       factor (I)
##  prop[1] 60       26790 937          28.6
##  prop[2] 150      40590 937          43.3
##  prop[3] 120      37290 937          39.8
##
##
##
##  Based on these diagnostics you should repeat the MCMC with  57780  iterations
##  and a thinning interval of  62  ,if these values are higher than the values
##  used to produce these diagnostics
##
##
##
## ###################################
## Gelman-Rubin diagnostics
## ###################################
##
## Potential scale reduction factors:
##
##         Point est. Upper C.I.
```

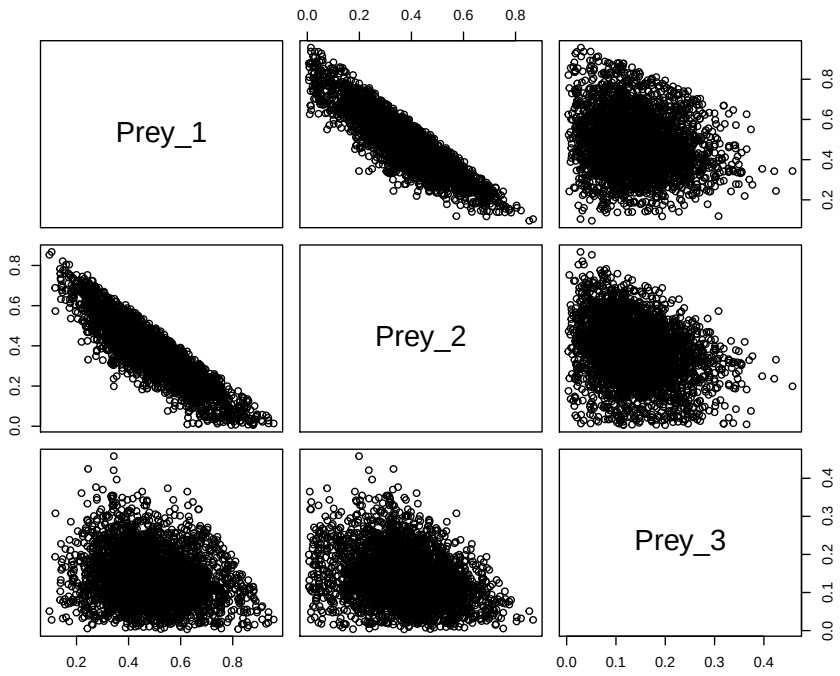
```
## prop[1]          1          1
## prop[2]          1          1
## prop[3]          1          1
##
## Multivariate psrf
##
## 1
##
##  Both univariate upper C.I. and multivariate psrf
##  should be close to 1 if the chains converged
##
##
```

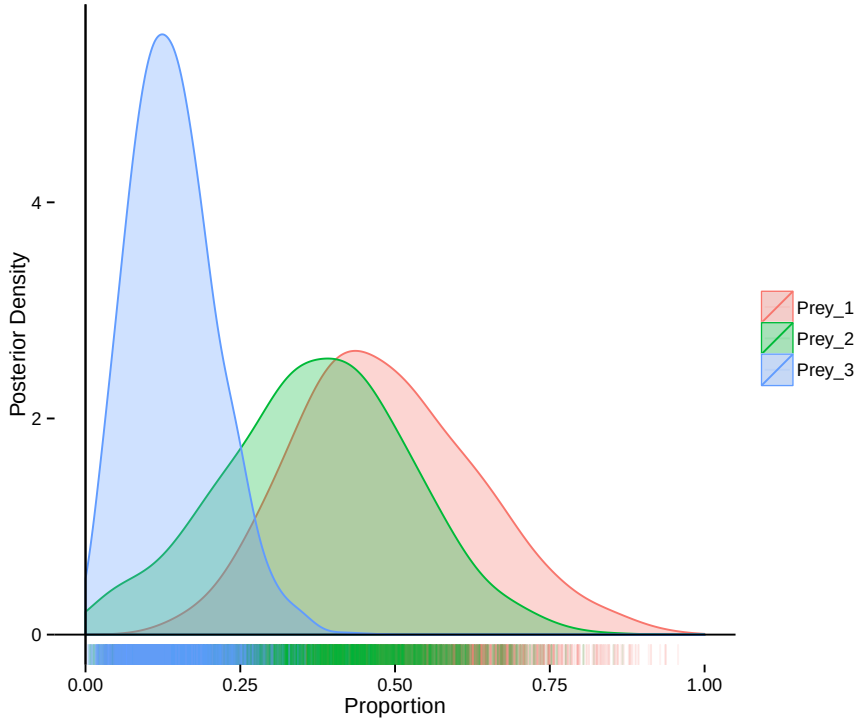
```
summary(Pop.Combined)
```

```
##
##
##  xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
## Printing results for MCMC chain 1
##  xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
##
##
##
##  population diet proportions
##
##
##       Prey_1  Prey_2  Prey_3
## 2.5%  0.2262 0.06287 0.02806
## 50%   0.4662 0.37423 0.13712
## 97.5% 0.7739 0.66963 0.28401
##
##
##  xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
## Printing results for MCMC chain 2
##  xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
##
##
##
##  population diet proportions
##
##
##       Prey_1  Prey_2  Prey_3
## 2.5%  0.2181 0.06218 0.02471
## 50%   0.4751 0.38103 0.13641
## 97.5% 0.7958 0.67408 0.29769
##
```

```
##
##  xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
## Printing results for MCMC chain 3
##  xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
##
##
##
##  population diet proportions
##
##
##       Prey_1  Prey_2  Prey_3
## 2.5%  0.2317 0.04416 0.02712
## 50%   0.4793 0.37625 0.13584
## 97.5% 0.8207 0.65240 0.30052
```

```
plot(Pop.Combined, save = F)

## Using  as id variables
```

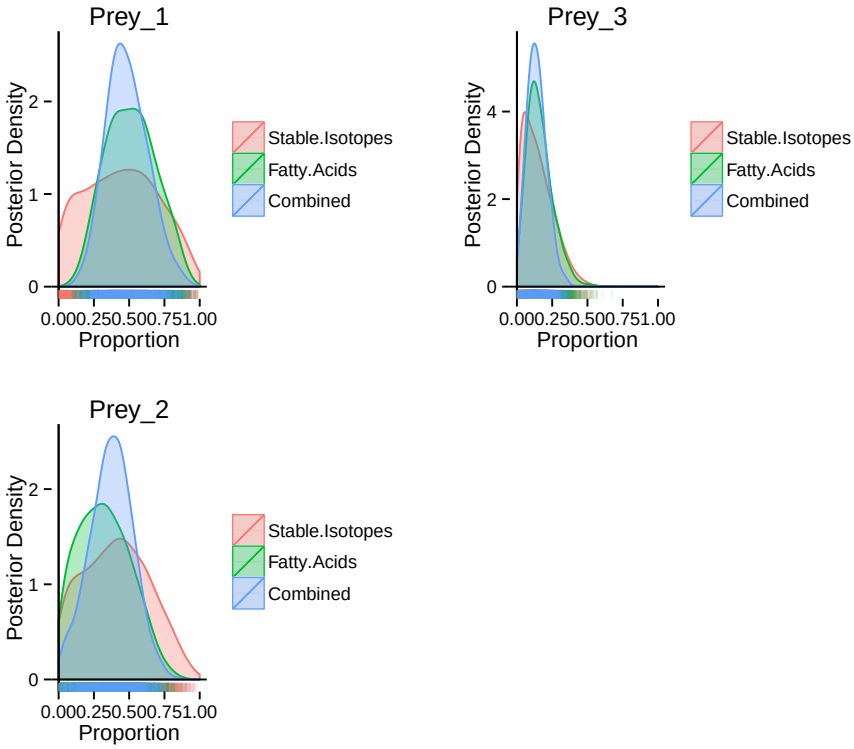


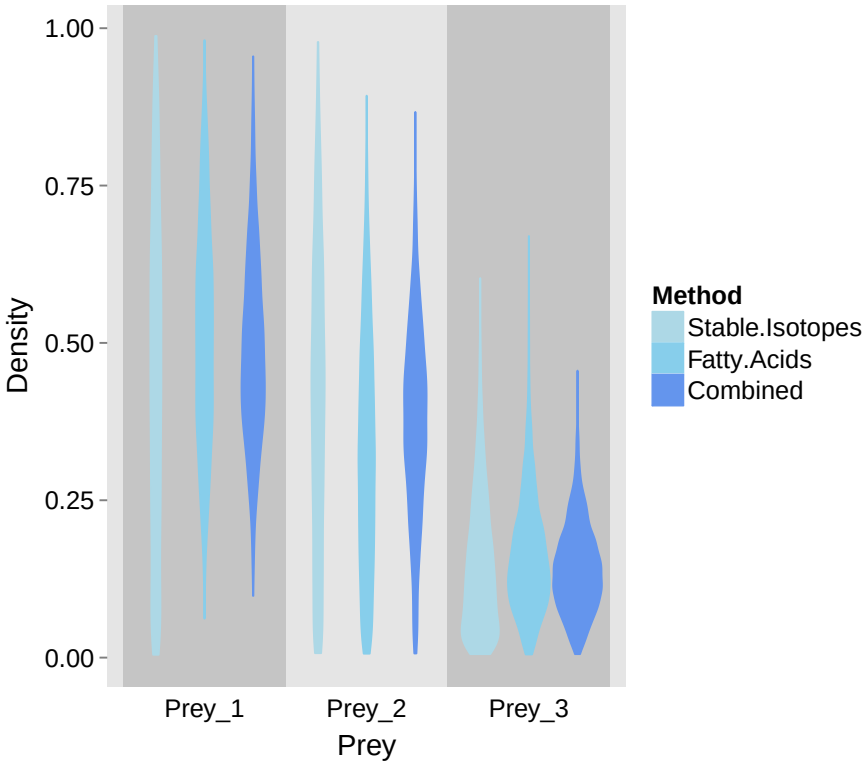


The combined analysis reduces uncertainty slightly, especially for source 3, but dietary sources remain uncertain with somewhat large credible intervals.

To compare the three approaches explicitly, we can use *multiplot*. For *multiplot* the three results need to be combined in a list:

```
Pop.list <- list(Stable.Isotopes = Pop.SI2, Fatty.Acids = Pop.FA3, Combined = Pop.Combined)

multiplot(Pop.list, save = F)

## Using V1 as id variables
## Using V1 as id variables
## Loading required package:  ggplot2
```


Prey_1
Posterior Density
0
1
2
0.000.250.500.751.00
Proportion
Stable.Isotopes
Fatty.Acids
Combined
Prey_3
Posterior Density
0
2
4
0.000.250.500.751.00
Proportion
Stable.Isotopes
Fatty.Acids
Combined
Prey_2
Posterior Density
0
1
2
0.000.250.500.751.00
Proportion
Stable.Isotopes
Fatty.Acids
Combined



The original data was simulated to include groups of predators with rather different proportions (visible in the dataplots above). Lets have a look at the proportions that were used to simulate the dataset:

```
proportions.path <- system.file("extdata", "Simdata_props.csv", package = "fastinR")
proportions <- read.csv(proportions.path, header = F, row.names = 1)

colnames(proportions) <- unique(dats$prey.ix)

proportions
```

```
##    Prey_1  Prey_2  Prey_3
## 1  0.1315 0.78796 0.08052
## 2  0.8168 0.03774 0.14550
## 3  0.1349 0.76823 0.09684
## 4  0.2429 0.64427 0.11285
## 5  0.1494 0.72239 0.12826
## 6  0.7965 0.07278 0.13076
## 7  0.7307 0.08529 0.18398
## 8  0.8189 0.07136 0.10973
## 9  0.8169 0.07488 0.10824
```

```
## 10 0.1727 0.74464 0.08268

# overall mean proportions
colMeans(proportions)

## Prey_1 Prey_2 Prey_3
## 0.4811 0.4010 0.1179
```

Looking at the proportions, it is clear that there are two groups of predators (e.g., juveniles and adults), one group preys preferentially on Prey 1 while the other preys mostly on Prey 2. Let's see if we can pick this up by estimating individual predator proportions:

1.1 Estimating individual proportions

Lets try this with fatty acids first - given that there was limited information about diets in the stable isotopes alone, they can't be expected SI alone to provide the extra information we're after.

We now need to deal with one extra prior to be set: the prior for the variance of diet proportions. After some exploratory runs, *even=2* seems like a reasonable compromise between our ability to detect differences (facilitated for smaller values of *even*) and the ability of the Markov Chains to mix properly (easier for larger *even*- see the documentation of *run_MCMC*).

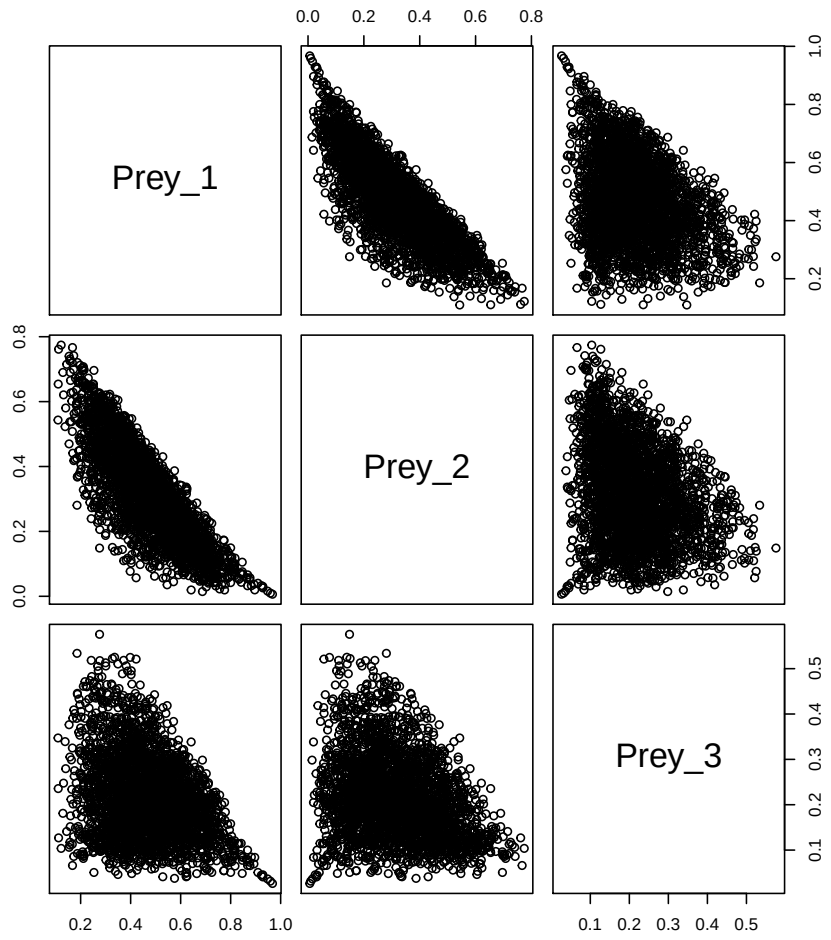
1.1.1 From Fatty Acids

```
Ind.FA <- run_MCMC(datas = dats.subset, nIter = 10000, nBurnin = 1000, nChains = 3, nThin =
    Data.Type = "Fatty.Acid.Profiles", Analysis.Type = "Individual.proportions", Rnot = 1,
    even = 1, plott = F, spawn = T)

## +++++++++
```

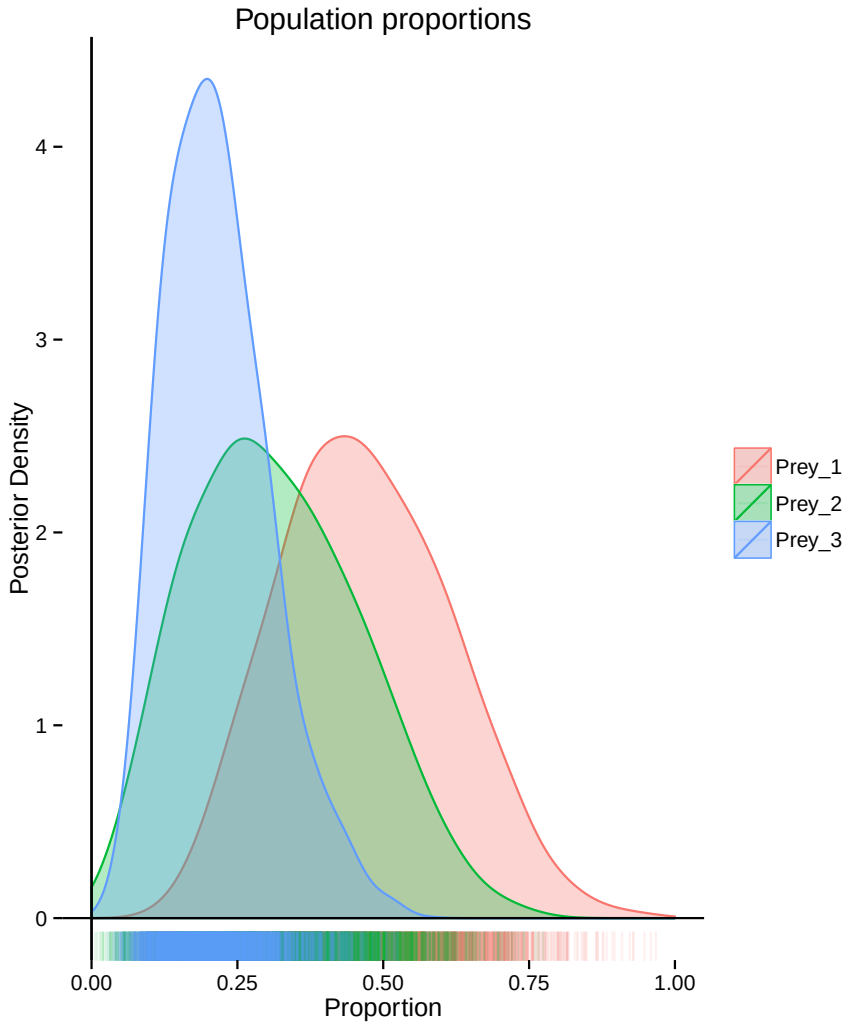
We won't show the output of the next commands anymore since it is far too long. The commands are the same as for a population proportion analysis:

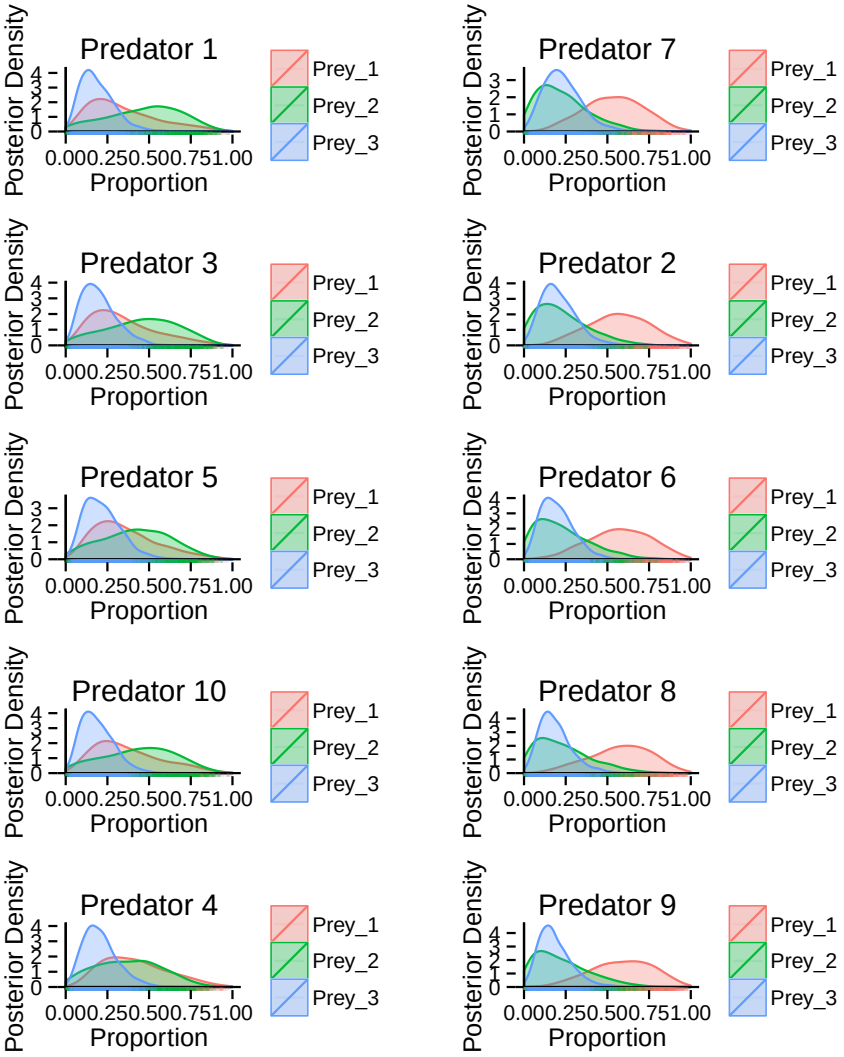
```
MCMCplot(Ind.FA)

diags(Ind.FA)

summary(Ind.FA)
```

```
plot(Ind.FA, save = F)

## Using  as id variables
```




```
## Using rep(i, nrow(outs)) as id variables
```


Population proportions
Posterior Density
4
3
2
1
0
0.00
0.25
0.50
0.75
1.00
Proportion
Prey_1
Prey_2
Prey_3



Despite this MCMC run being an order of magnitude too short, we can see that the grouped pattern emerges quite clearly in the last plot. Nevertheless, posterior correlations remain for prey items 1&2 remain strong (we'll soon see why!). First, let's see if the grouping becomes more pronounced with the inclusion of stable isotopes:

1.1.2 Combined analysis

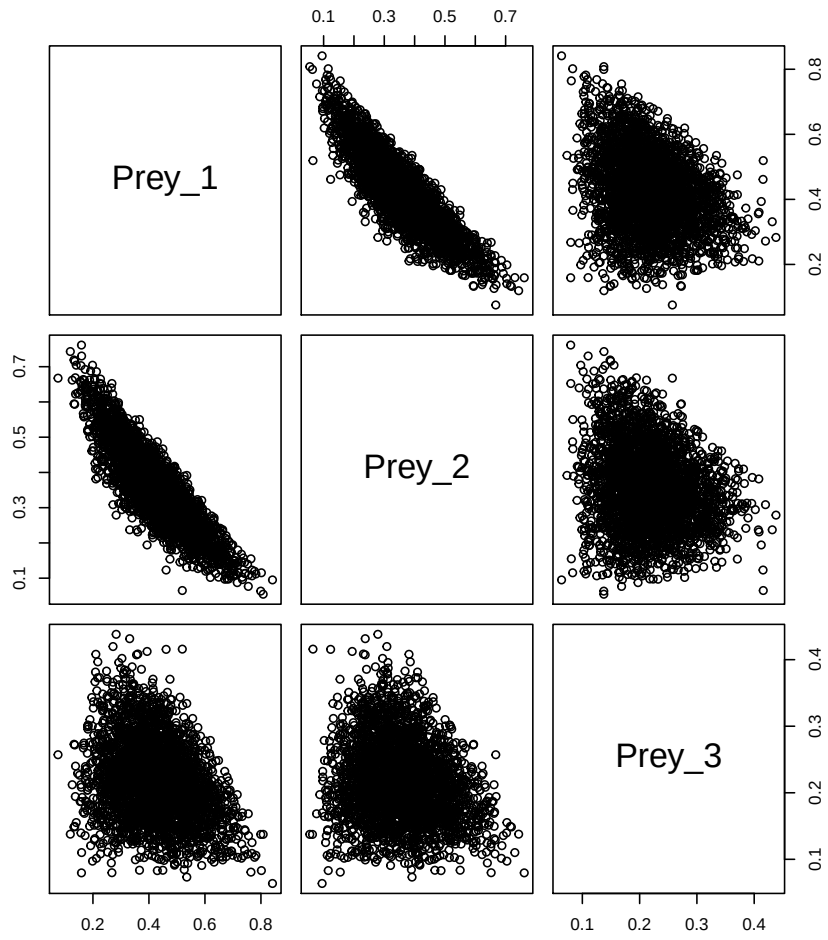
```
Ind.Combined <- run_MCMC(datas = dats.subset, nIter = 10000, nBurnin = 1000, nChains = 3, nT
    Data.Type = "Combined.Analysis", Analysis.Type = "Individual.proportions", Rnot = 1, Rno
    even = 2, plott = F, spawn = T)
```

```
## ++++++++++++
```

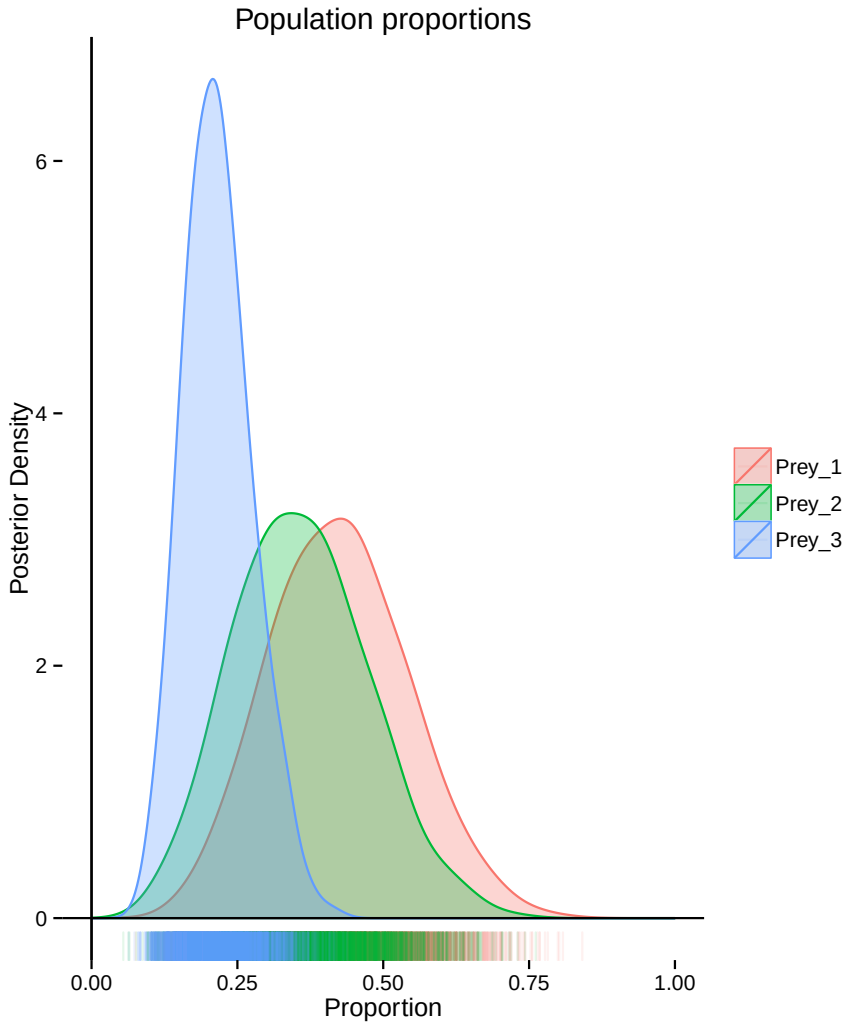
Again skipping diagnostics output:

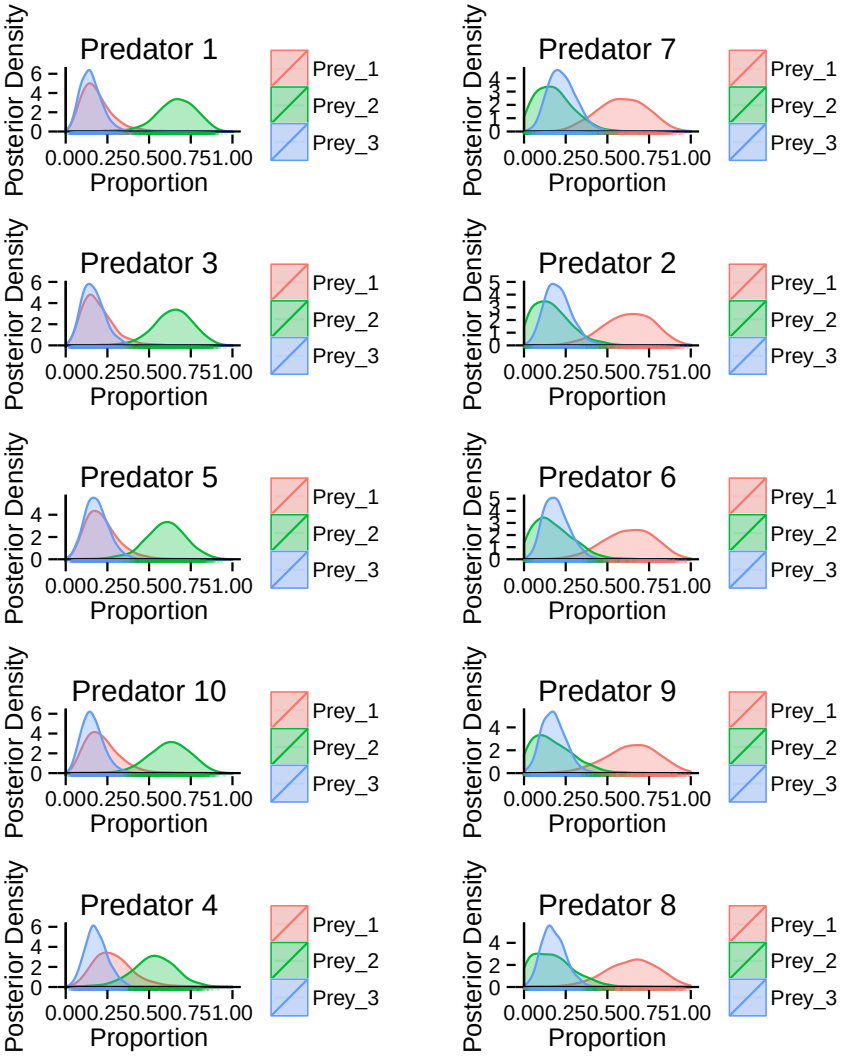
```
MCMCplot(Ind.Combined)

diags(Ind.Combined)

summary(Ind.Combined)
```

```
plot(Ind.Combined, save = F)

## Using  as id variables
```




```
## Using rep(i, nrow(outs)) as id variables
```


Population proportions
Posterior Density
6
4
2
0
0.00
0.25
0.50
0.75
1.00
Proportion
Prey_1
Prey_2
Prey_3



The uncertainty for predators preying predominantly on Prey 2 is significantly reduced (but once again the analysis should be run for an order of magnitude longer with a larger thinning interval to ensure that the estimates are reliable).

While the patterns here offer great insights into individual diet proportions, they do not provide a means to estimate the population distribution of diet proportions for these two groups. This can be achieved in an anova type linear model:

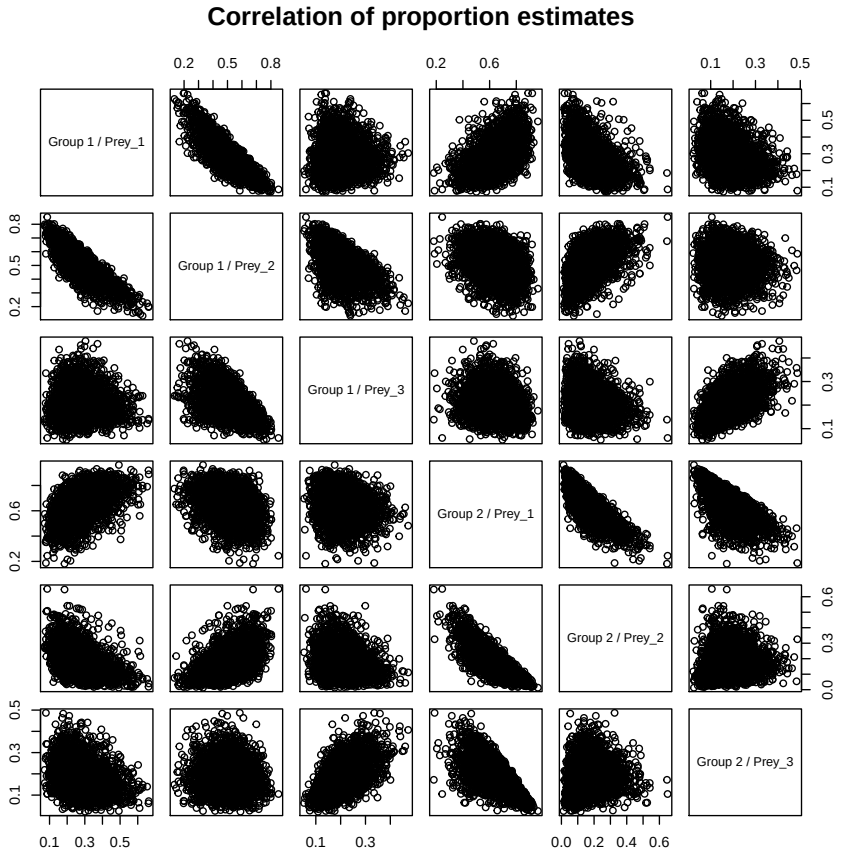
1.2 Estimating group proportions

We now need to add covariates in the form of group membership. The reasoning and procedure is the same for continuous covariates (e.g., size). To add the covariates, we use the *add_Covs*constructor:

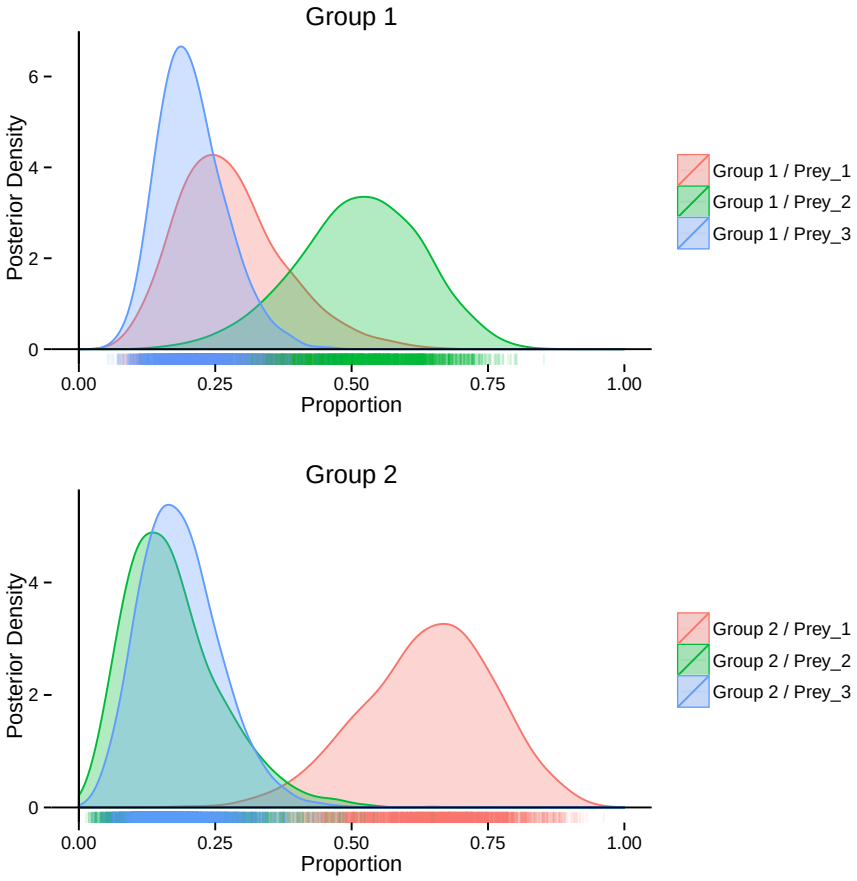
```
group.path <- system.file("extdata", "Simdata_groups.csv", package = "fastinR")
Covs <- add_Covs(Groups = group.path)

Cov.Combined <- run_MCMC(datas = dats.subset, Covs = Covs, nIter = 1e+05, nBurnin = 10000,
    nChains = 3, nThin = 100, Data.Type = "Combined.Analysis", Analysis.Type = "Analysis.wit
    Rnot = 1, Rnot_SI = 1, even = 2, plott = F, spawn = T)

## ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
```

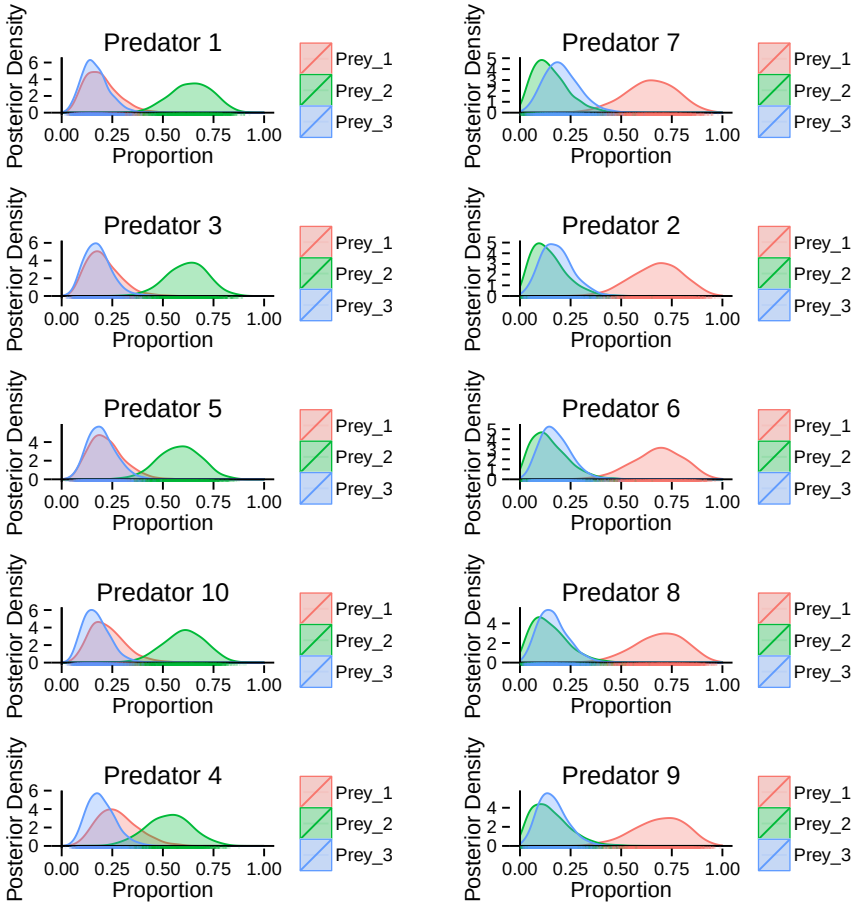
```
MCMCplot(Cov.Combined)

diags(Cov.Combined)

summary(Cov.Combined)
```

```
plot(Cov.Combined, save = F)

## Using  as id variables
## Using  as id variables
```




```
## Using rep(i, nrow(outs)) as id variables
```


Group 1
Posterior Density
6
4
2
0
0.00
0.25
0.50
0.75
1.00
Proportion
Group 1 / Prey_1
Group 1 / Prey_2
Group 1 / Prey_3
Group 2
Posterior Density
4
2
0
0.00
0.25
0.50
0.75
1.00
Proportion
Group 2 / Prey_1
Group 2 / Prey_2
Group 2 / Prey_3



We can now clearly see the difference between the two group's diet proportions!