Package 'simmr'

June 15, 2019

Type Package

Title A Stable Isotope Mixing Model

Version 0.4.0

Date 2019-06-14

Author Andrew Parnell

Language en-US

Maintainer Andrew Parnell <andrew.parnell@mu.ie>

Description Fits a stable isotope mixing model via 'JAGS' in R. The package allows for any number of isotopes or sources, as well as concentration dependencies. The methods on the papers Parnell et al 2010 <doi:10.1371/journal.pone.0009672>, and Parnell et al 2013 <doi:10.1002/env.2221>.

Depends R (>= 3.2.2), R2jags, ggplot2

Imports MASS, compositions, boot, reshape2, coda, graphics, stats, viridis, bayesplot

Encoding UTF-8

License GPL (>= 2)

LazyData TRUE

Suggests knitr, rmarkdown, readxl

VignetteBuilder knitr

NeedsCompilation no

RoxygenNote 6.1.1

Repository CRAN

Date/Publication 2019-06-14 22:10:10 UTC

R topics documented:

combine_sources 2
compare_groups 4
compare_sources 7
plot.simmr_input 9

plot.simmr_output 14
posterior_predictive 16
print.simmr_input 17
print.simmr_output 17
prior_viz 18
simmr 19
simmr_elicit 21
simmr_load 23
simmr_mcmc 25
simmr_mcmc_tdf 32
summary.simmr_output 35
summary.simmr_output_tdf 37

Index **40**

`combine_sources` *Combine the dietary proportions from two food sources after running simmr*

Description

This function takes in an object of class `simmr_output` and combines two of the food sources. It works for single and multiple group data.

Usage

```
combine_sources(simmr_out,
  to_combine = simmr_out$input$source_names[1:2],
  new_source_name = "combined_source")
```

Arguments

`simmr_out`	An object of class `simmr_output` created from `simmr_mcmc`.
`to_combine`	The names of exactly two sources. These should match the names given to `simmr_load`.
`new_source_name`	A name to give to the new combined source

Details

Often two sources either (1) lie in similar location on the iso-space plot, or (2) are very similar in phylogenetic terms. In case (1) it is common to experience high (negative) posterior correlations between the sources. Combining them can reduce this correlation and improve precision of the estimates. In case (2) we might wish to determine the joint amount eaten of the two sources when combined. This function thus combines two sources after a run of `simmr_mcmc` (known as a posteriori combination). The new object can then be called with `plot.simmr_input` or `plot.simmr_output` to produce iso-space plots of summaries of the output after combination.

Value

A new `simmr_output` object

Author(s)

Andrew Parnell <andrew.parnell@mu.ie>

See Also

See `simmr_mcmc` and the associated vignette for examples.

Examples

```
## Not run:
# Data set 1: 10 obs on 2 isos, 4 sources, with tefs and concdep

# The data
mix = matrix(c(-10.13, -10.72, -11.39, -11.18, -10.81, -10.7, -10.54,
-10.48, -9.93, -9.37, 11.59, 11.01, 10.59, 10.97, 11.52, 11.89,
11.73, 10.89, 11.05, 12.3), ncol=2, nrow=10)
colnames(mix) = c('d13C','d15N')
s_names=c('Source A','Source B','Source C','Source D')
s_means = matrix(c(-14, -15.1, -11.03, -14.44, 3.06, 7.05, 13.72, 5.96), ncol=2, nrow=4)
s_sds = matrix(c(0.48, 0.38, 0.48, 0.43, 0.46, 0.39, 0.42, 0.48), ncol=2, nrow=4)
c_means = matrix(c(2.63, 1.59, 3.41, 3.04, 3.28, 2.34, 2.14, 2.36), ncol=2, nrow=4)
c_sds = matrix(c(0.41, 0.44, 0.34, 0.46, 0.46, 0.48, 0.46, 0.66), ncol=2, nrow=4)
conc = matrix(c(0.02, 0.1, 0.12, 0.04, 0.02, 0.1, 0.09, 0.05), ncol=2, nrow=4)

# Load into simmr
simmr_1 = simmr_load(mixtures=mix,
                     source_names=s_names,
                     source_means=s_means,
                     source_sds=s_sds,
                     correction_means=c_means,
                     correction_sds=c_sds,
                     concentration_means = conc)

# Plot
plot(simmr_1)

# Print
simmr_1

# MCMC run
simmr_1_out = simmr_mcmc(simmr_1)

# Print it
print(simmr_1_out)

# Summary
summary(simmr_1_out)
summary(simmr_1_out,type='diagnostics')
```

```
summary(simmr_1_out,type='correlations')
summary(simmr_1_out,type='statistics')
ans = summary(simmr_1_out,type=c('quantiles','statistics'))

# Plot
plot(simmr_1_out)
plot(simmr_1_out,type='boxplot')
plot(simmr_1_out,type='histogram')
plot(simmr_1_out,type='density')
plot(simmr_1_out,type='matrix')

simmr_out_combine = combine_sources(simmr_1_out,
to_combine=c('Source A','Source D'),
new_source_name='Source A+D')
plot(simmr_out_combine$input)
plot(simmr_out_combine,type='boxplot',title='simmr output: combined sources')


## End(Not run)
```

`compare_groups` *Compare dietary proportions for a single source across different groups*

Description

This function takes in an object of class `simmr_output` and creates probabilistic comparisons for a given source and a set of at least two groups.

Usage

```
compare_groups(simmr_out, source_name = simmr_out$input$source_names[1],
  groups = 1:2, plot = TRUE)
```

Arguments

`simmr_out`	An object of class `simmr_output` created from `simmr_mcmc`.
`source_name`	The name of a source. This should match the names exactly given to `simmr_load`.
`groups`	The integer values of the group numbers to be compared. At least two groups must be specified.
`plot`	A logical value specifying whether plots should be produced or not.

Details

When two groups are specified, the function produces a direct calculation of the probability that one group is bigger than the other. When more than two groups are given, the function produces a set of most likely probabilistic orderings for each combination of groups. The function produces boxplots by default and also allows for the storage of the output for further analysis if required.

Value

If there are two groups, a vector containing the differences between the two groups proportions for that source. If there are multiple groups, a list containing the following fields:

`Ordering`	The different possible orderings of the dietary proportions across groups
`out_all`	The dietary proportions for this source across the groups specified as columns in a matrix

Author(s)

Andrew Parnell <andrew.parnell@mu.ie>

See Also

See `simmr_mcmc` for complete examples.

Examples

```
## Not run:
mix = matrix(c(10.22, 10.37, 10.44, 10.52, 10.19, 10.45, 9.91, 11.27,
               9.34, 11.68, 12.29, 11.04, 11.46, 11.73, 12.29, 11.79, 11.49,
               11.73, 11.1, 11.36, 12.19, 11.03, 11.21, 10.58, 11.61, 12.16,
               10.7, 11.47, 12.07, 11.75, 11.86, 12.33, 12.36, 11.13, 10.92,
               12.42, 10.95, 12.28, 11.04, 10.76, 10.99, 10.78, 11.07, 10.2,
               11.67, 7.53, 10.65, 10.58, 11.13, 7.73, 10.79, 10.47, 10.82,
               10.41, 11.1, 10.95, 10.76, 10.83, 10.25, 10.52, 9.94, 9.94, 11.61,
               10.65, 10.76, 11.11, 10.2, 11.27, 10.21, 10.88, 11.21, 11.36,
               10.75, 12.38, 11.16, 11.57, 10.79, 11.13, 10.72, 10.99, 10.38,
               10.95, 10.75, 10.75, 11.05, 10.66, 10.61, 10.9, 11.14, 10.33,
               10.83, 10.75, 9.18, 9.03, 9.05, 8.6, 8.29, 10.32, 10.28, 6.47,
               11.36, 10.75, 11.13, 11.37, 10.86, 10.54, 10.39, 10.66, 9.99,
               11.65, 11.02, 10.67, 8.15, 11.12, 10.95, 11.2, 10.76, 11.32,
               10.85, 11.74, 10.46, 10.93, 12.3, 10.67, 11.51, 10.56, 12.51,
               13.51, 11.98, 12.2, 10.48, 12.4, 13, 11.36, 12.08, 12.39, 12.28,
               12.6, 11.3, 11.1, 11.42, 11.49, 12, 13.35, 11.97, 13.35, 12.75,
               12.55, 12.3, 12.51, 12.61, 10.98, 11.82, 12.27, 12.11, 12.11,
               12.89, 12.99, 12.29, 11.89, 12.74, 12.29, 11.89, 10.56, 9.27,
               10.54, 10.97, 10.46, 10.56, 10.86, 10.9, 11.06, 10.76, 10.64,
               10.94, 10.85, 10.45, 11.15, 11.23, 11.16, 10.94, 11.2, 10.71,
               9.55, 8.6, 9.67, 8.17, 9.81, 10.94, 9.49, 9.46, 7.94, 9.77, 8.07,
               8.39, 8.95, 9.83, 8.51, 8.86, 7.93, 8, 8.33, 8, 9.39, 8.01, 7.59,
               8.26, 9.49, 8.23, 9.1, 8.21, 9.59, 9.37, 9.47, 8.6, 8.23, 8.39,
               8.24, 8.34, 8.36, 7.22, 7.13, 10.64, 8.06, 8.22, 8.92, 9.35,
               7.32, 7.66, 8.09, 7.3, 7.33, 7.33, 7.36, 7.49, 8.07, 8.84, 7.93,
               7.94, 8.74, 8.26, 9.63, 8.85, 7.55, 10.05, 8.23, 7.74, 9.12,
               7.33, 7.54, 8.8, -11.36, -11.88, -10.6, -11.25, -11.66, -10.41,
               -10.88, -14.73, -11.52, -15.89, -14.79, -17.64, -16.97, -17.25,
               -14.77, -15.67, -15.34, -15.53, -17.27, -15.63, -15.94, -14.88,
               -15.9, -17.11, -14.93, -16.26, -17.5, -16.37, -15.21, -15.43,
               -16.54, -15, -16.41, -15.09, -18.06, -16.27, -15.08, -14.39,
               -21.45, -22.52, -21.25, -21.84, -22.51, -21.97, -20.23, -21.64,
```

```
                -22.49, -21.91, -21.65, -21.37, -22.9, -21.13, -19.33, -20.29,
                -20.56, -20.87, -21.07, -21.69, -21.17, -21.74, -22.69, -21.06,
                -20.42, -21.5, -20.15, -21.99, -22.3, -21.71, -22.48, -21.86,
                -21.68, -20.97, -21.91, -19.05, -22.78, -22.36, -22.46, -21.52,
                -21.84, -21.3, -21.39, -22.1, -21.59, -20.14, -20.67, -20.31,
                -20.07, -21.2, -20.44, -22.06, -22.05, -21.44, -21.93, -22.47,
                -22.27, -22.19, -22.81, -20.48, -22.47, -18.06, -20.72, -20.97,
                -19.11, -18.4, -20.45, -21.2, -19.74, -20.48, -21.48, -17.81,
                -19.77, -22.56, -14.72, -12.21, -12.35, -13.88, -14.43, -14.65,
                -13.9, -14.12, -10.88, -10.44, -15.33, -13.78, -13.98, -15.22,
                -15.25, -15.76, -15.78, -15.49, -13.02, -15.3, -15.55, -14.35,
                -14.99, -14.83, -16.18, -15.01, -12.87, -14.67, -13.84, -14.89,
                -13.33, -15.04, -14.29, -15.62, -13.99, -15.06, -15.06, -15,
                -14.55, -13.32, -14.34, -14.47, -14.31, -14.18, -16.18, -16.25,
                -15.92, -15.35, -14.29, -15.92, -15.35, -20.22, -21.4, -19.97,
                -20.78, -20.61, -20.58, -20.19, -20.71, -20.59, -20.09, -19.37,
                -20.41, -20.84, -20.75, -20.29, -20.89, -19.69, -20.41, -21.24,
                -19.33, -25.87, -25.4, -27.23, -27.52, -24.55, -17.36, -24.7,
                -27.76, -28.92, -25.98, -26.77, -28.76, -27.7, -24.75, -25.47,
                -26.58, -28.94, -29.13, -26.65, -28.04, -27.5, -29.28, -27.85,
                -27.41, -27.57, -29.06, -25.98, -28.21, -25.27, -14.43, -27.4,
                -27.76, -28.45, -27.35, -28.83, -29.39, -28.86, -28.61, -29.27,
                -20.32, -28.21, -26.3, -28.27, -27.75, -28.55, -27.38, -29.13,
                -28.66, -29.02, -26.04, -26.06, -28.52, -28.51, -27.93, -29.07,
                -28.41, -26.42, -27.71, -27.75, -24.28, -28.43, -25.94, -28,
                -28.59, -22.61, -27.34, -27.35, -29.14), ncol=2, nrow=251)
colnames(mix) = c('d13C','d15N')
s_names = c("Zostera", "Grass", "U.lactuca", "Enteromorpha")
s_means = matrix(c(6.49, 4.43, 11.19, 9.82, -11.17, -30.88, -11.17,
                   -14.06), ncol=2, nrow=4)
s_sds = matrix(c(1.46, 2.27, 1.11, 0.83, 1.21, 0.64, 1.96, 1.17), ncol=2, nrow=4)
c_means = matrix(c(3.54, 3.54, 3.54, 3.54, 1.63, 1.63, 1.63, 1.63), ncol=2, nrow=4)
c_sds = matrix(c(0.74, 0.74, 0.74, 0.74, 0.63, 0.63, 0.63, 0.63), ncol=2, nrow=4)
conc = matrix(c(0.03, 0.04, 0.02, 0.01, 0.36, 0.4, 0.21, 0.18), ncol=2, nrow=4)
grp = as.integer(c(1, 1, 1, 1, 1, 1, 1, 1, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
        2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 3, 3, 3,
        3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3,
        3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3,
        3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3,
        3, 3, 3, 3, 3, 3, 3, 3, 4, 4, 4, 4, 4, 4, 4, 4, 4, 5, 5, 5,
        5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5,
        5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 6, 6, 6, 6,
        6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 7, 7, 7, 7, 7,
        7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7,
        7, 7, 7, 7, 7, 7, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8,
        8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8))

# Load this in:
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,
                    concentration_means = conc,
                    group=grp)

# Print
simmr_in

# Plot
plot(simmr_in,group=1:8,xlab=expression(paste(delta^13, "C (\u2030)",sep="")),
     ylab=expression(paste(delta^15, "N (\u2030)",sep="")),
     title='Isospace plot of Inger et al Geese data')

# Run MCMC for each group
simmr_out = simmr_mcmc(simmr_in)

# Print output
simmr_out

# Summarise output
summary(simmr_out,type='quantiles',group=1)
summary(simmr_out,type='quantiles',group=c(1,3))
summary(simmr_out,type=c('quantiles','statistics'),group=c(1,3))

# Plot - only a single group allowed
plot(simmr_out,type='boxplot',group=2,title='simmr output group 2')
plot(simmr_out,type=c('density','matrix'),grp=6,title='simmr output group 6')

# Compare groups
compare_groups(simmr_out,source='Zostera',groups=1:2)
compare_groups(simmr_out,source='Zostera',groups=1:3)
compare_groups(simmr_out,source='U.lactuca',groups=c(4:5,7,2))

## End(Not run)
```

`compare_sources` *Compare dietary proportions between multiple sources*

Description

This function takes in an object of class `simmr_output` and creates probabilistic comparisons between the supplied sources. The group number can also be specified.

Usage

```
compare_sources(simmr_out, source_names = simmr_out$input$source_names,
  group = 1, plot = TRUE)
```

Arguments

simmr_out	An object of class simmr_output created from simmr_mcmc.
source_names	The names of at least two sources. These should match the names exactly given to simmr_load.
group	The integer values of the group numbers to be compared. If not specified assumes the first or only group
plot	A logical value specifying whether plots should be produced or not.

Details

When two sources are specified, the function produces a direct calculation of the probability that the dietary proportion for one source is bigger than the other. When more than two sources are given, the function produces a set of most likely probabilistic orderings for each combination of sources. The function produces boxplots by default and also allows for the storage of the output for further analysis if required.

Value

If there are two sources, a vector containing the differences between the two dietary proportion proportions for these two sources. If there are multiple sources, a list containing the following fields:

Ordering	The different possible orderings of the dietary proportions across sources
out_all	The dietary proportions for these sources specified as columns in a matrix

Author(s)

Andrew Parnell <andrew.parnell@mu.ie>

See Also

See simmr_mcmc for complete examples.

Examples

```
## Not run:
# Data set 1: 10 obs on 2 isos, 4 sources, with tefs and concdep

# The data
mix = matrix(c(-10.13, -10.72, -11.39, -11.18, -10.81, -10.7, -10.54,
-10.48, -9.93, -9.37, 11.59, 11.01, 10.59, 10.97, 11.52, 11.89,
11.73, 10.89, 11.05, 12.3), ncol=2, nrow=10)
colnames(mix) = c('d13C','d15N')
s_names=c('Source A','Source B','Source C','Source D')
s_means = matrix(c(-14, -15.1, -11.03, -14.44, 3.06, 7.05, 13.72, 5.96), ncol=2, nrow=4)
s_sds = matrix(c(0.48, 0.38, 0.48, 0.43, 0.46, 0.39, 0.42, 0.48), ncol=2, nrow=4)
c_means = matrix(c(2.63, 1.59, 3.41, 3.04, 3.28, 2.34, 2.14, 2.36), ncol=2, nrow=4)
c_sds = matrix(c(0.41, 0.44, 0.34, 0.46, 0.46, 0.48, 0.46, 0.66), ncol=2, nrow=4)
conc = matrix(c(0.02, 0.1, 0.12, 0.04, 0.02, 0.1, 0.09, 0.05), ncol=2, nrow=4)
```

```
# Load into simmr
simmr_1 = simmr_load(mixtures=mix,
                     source_names=s_names,
                     source_means=s_means,
                     source_sds=s_sds,
                     correction_means=c_means,
                     correction_sds=c_sds,
                     concentration_means = conc)

# Plot
plot(simmr_1)

# Print
simmr_1

# MCMC run
simmr_1_out = simmr_mcmc(simmr_1)

# Print it
print(simmr_1_out)

# Summary
summary(simmr_1_out)
summary(simmr_1_out,type='diagnostics')
summary(simmr_1_out,type='correlations')
summary(simmr_1_out,type='statistics')
ans = summary(simmr_1_out,type=c('quantiles','statistics'))

# Plot
plot(simmr_1_out,type='boxplot')
plot(simmr_1_out,type='histogram')
plot(simmr_1_out,type='density')
plot(simmr_1_out,type='matrix')

# Compare two sources
compare_sources(simmr_1_out,source_names=c('Source B','Source D'))

# Compare multiple sources
compare_sources(simmr_1_out)

## End(Not run)
```

`plot.simmr_input` *Plot the* `simmr_input` *data created from* `simmr_load`

Description

This function creates iso-space (AKA tracer-space or delta-space) plots. They are vital in determining whether the data are suitable for running in a SIMM.

Usage

```
## S3 method for class 'simmr_input'
plot(x, tracers = c(1, 2),
  title = "Tracers plot", xlab = colnames(x$mixtures)[tracers[1]],
  ylab = colnames(x$mixtures)[tracers[2]], sigmas = 1, group = 1,
  mix_name = "Mixtures", ggargs = NULL, colour = TRUE, ...)
```

Arguments

x	An object created via the function simmr_load
tracers	The choice of tracers to plot. If there are more than two tracers, it is recommended to plot every pair of tracers to determine whether the mixtures lie in the mixing polygon defined by the sources
title	A title for the graph
xlab	The x-axis label. By default this is assumed to be delta-13C but can be made richer if required. See examples below.
ylab	The y-axis label. By default this is assumed to be delta-15N in per mil but can be changed as with the x-axis label
sigmas	The number of standard deviations to plot on the source values. Defaults to 1.
group	Which groups to plot. Can be a single group or multiple groups
mix_name	A optional string containing the name of the mixture objects, e.g. Geese.
ggargs	Extra arguments to be included in the ggplot (e.g. axis limits)
colour	If TRUE (default) creates a plot. If not, puts the plot in black and white
...	Not used

Details

It is desirable to have the vast majority of the mixture observations to be inside the convex hull defined by the food sources. When there are more than two tracers (as in one of the examples below) it is recommended to plot all the different pairs of the food sources. See the vignette for further details of richer plots.

Author(s)

Andrew Parnell <andrew.parnell@mu.ie>

See Also

See plot.simmr_output for plotting the output of a simmr run. See simmr_mcmc for running a simmr object once the iso-space is deemed acceptable.

Examples

```
# A simple example with 10 observations, 4 food sources and 2 tracers
mix = matrix(c(-10.13, -10.72, -11.39, -11.18, -10.81, -10.7, -10.54,
-10.48, -9.93, -9.37, 11.59, 11.01, 10.59, 10.97, 11.52, 11.89,
11.73, 10.89, 11.05, 12.3), ncol=2, nrow=10)
colnames(mix) = c('d13C','d15N')
s_names=c('Source A','Source B','Source C','Source D')
s_means = matrix(c(-14, -15.1, -11.03, -14.44, 3.06, 7.05, 13.72, 5.96), ncol=2, nrow=4)
s_sds = matrix(c(0.48, 0.38, 0.48, 0.43, 0.46, 0.39, 0.42, 0.48), ncol=2, nrow=4)
c_means = matrix(c(2.63, 1.59, 3.41, 3.04, 3.28, 2.34, 2.14, 2.36), ncol=2, nrow=4)
c_sds = matrix(c(0.41, 0.44, 0.34, 0.46, 0.46, 0.48, 0.46, 0.66), ncol=2, nrow=4)
conc = matrix(c(0.02, 0.1, 0.12, 0.04, 0.02, 0.1, 0.09, 0.05), ncol=2, nrow=4)

simmr_1 = simmr_load(mixtures=mix,
                     source_names=s_names,
                     source_means=s_means,
                     source_sds=s_sds,
                     correction_means=c_means,
                     correction_sds=c_sds,
                     concentration_means = conc)

# Plot
plot(simmr_1)

### A more complicated example with 30 obs, 3 tracers and 4 sources
mix = matrix(c(-11.67, -12.55, -13.18, -12.6, -11.77, -11.21, -11.45,
               -12.73, -12.49, -10.6, -12.26, -12.48, -13.07, -12.67, -12.26,
               -13.12, -10.83, -13.2, -12.24, -12.85, -11.65, -11.84, -13.26,
               -12.56, -12.97, -12.18, -12.76, -11.53, -12.87, -12.49, 7.79,
               7.85, 8.25, 9.06, 9.13, 8.56, 8.03, 7.74, 8.16, 8.43, 7.9, 8.32,
               7.85, 8.14, 8.74, 9.17, 7.33, 8.06, 8.06, 8.03, 8.16, 7.24, 7.24,
               8, 8.57, 7.98, 7.2, 8.13, 7.78, 8.21, 11.31, 10.92, 11.3, 11,
               12.21, 11.52, 11.05, 11.05, 11.56, 11.78, 12.3, 10.87, 10.35,
               11.66, 11.46, 11.55, 11.41, 12.01, 11.97, 11.5, 11.18, 11.49,
               11.8, 11.63, 10.99, 12, 10.63, 11.27, 11.81, 12.25), ncol=3, nrow=30)
colnames(mix) = c('d13C','d15N','d34S')
s_names = c('Source A', 'Source B', 'Source C', 'Source D')
s_means = matrix(c(-14, -15.1, -11.03, -14.44, 3.06, 7.05, 13.72, 5.96,
                   10.35, 7.51, 10.31, 9), ncol=3, nrow=4)
s_sds = matrix(c(0.46, 0.39, 0.42, 0.48, 0.44, 0.37, 0.49, 0.47, 0.49,
                 0.42, 0.41, 0.42), ncol=3, nrow=4)
c_means = matrix(c(1.3, 1.58, 0.81, 1.7, 1.73, 1.83, 1.69, 3.2, 0.67,
                   2.99, 3.38, 1.31), ncol=3, nrow=4)
c_sds = matrix(c(0.32, 0.64, 0.58, 0.46, 0.61, 0.55, 0.47, 0.45, 0.34,
                 0.45, 0.37, 0.49), ncol=3, nrow=4)
conc = matrix(c(0.05, 0.1, 0.06, 0.07, 0.07, 0.03, 0.07, 0.05, 0.1,
                0.05, 0.12, 0.11), ncol=3, nrow=4)

# Load this in:
simmr_3 = simmr_load(mixtures=mix,
                     source_names=s_names,
```

```
                    source_means=s_means,
                    source_sds=s_sds,
                    correction_means=c_means,
                    correction_sds=c_sds,
                    concentration_means = conc)


# Plot 3 times - first default d13C vs d15N
plot(simmr_3)
# Now plot d15N vs d34S
plot(simmr_3,tracers=c(2,3))
# and finally d13C vs d34S
plot(simmr_3,tracers=c(1,3))
# See vignette('simmr') for fancier x-axis labels

# An example with multiple groups - the Geese data from Inger et al 2006
# Do this in raw data format - Note that there's quite a few mixtures!
mix = matrix(c(-11.36, -11.88, -10.6, -11.25, -11.66, -10.41,
               -10.88, -14.73, -11.52, -15.89, -14.79, -17.64, -16.97, -17.25,
               -14.77, -15.67, -15.34, -15.53, -17.27, -15.63, -15.94, -14.88,
               -15.9, -17.11, -14.93, -16.26, -17.5, -16.37, -15.21, -15.43,
               -16.54, -15, -16.41, -15.09, -18.06, -16.27, -15.08, -14.39,
               -21.45, -22.52, -21.25, -21.84, -22.51, -21.97, -20.23, -21.64,
               -22.49, -21.91, -21.65, -21.37, -22.9, -21.13, -19.33, -20.29,
               -20.56, -20.87, -21.07, -21.69, -21.17, -21.74, -22.69, -21.06,
               -20.42, -21.5, -20.15, -21.99, -22.3, -21.71, -22.48, -21.86,
               -21.68, -20.97, -21.91, -19.05, -22.78, -22.36, -22.46, -21.52,
               -21.84, -21.3, -21.39, -22.1, -21.59, -20.14, -20.67, -20.31,
               -20.07, -21.2, -20.44, -22.06, -22.05, -21.44, -21.93, -22.47,
               -22.27, -22.19, -22.81, -20.48, -22.47, -18.06, -20.72, -20.97,
               -19.11, -18.4, -20.45, -21.2, -19.74, -20.48, -21.48, -17.81,
               -19.77, -22.56, -14.72, -12.21, -12.35, -13.88, -14.43, -14.65,
               -13.9, -14.12, -10.88, -10.44, -15.33, -13.78, -13.98, -15.22,
               -15.25, -15.76, -15.78, -15.49, -13.02, -15.3, -15.55, -14.35,
               -14.99, -14.83, -16.18, -15.01, -12.87, -14.67, -13.84, -14.89,
               -13.33, -15.04, -14.29, -15.62, -13.99, -15.06, -15.06, -15,
               -14.55, -13.32, -14.34, -14.47, -14.31, -14.18, -16.18, -16.25,
               -15.92, -15.35, -14.29, -15.92, -15.35, -20.22, -21.4, -19.97,
               -20.78, -20.61, -20.58, -20.19, -20.71, -20.59, -20.09, -19.37,
               -20.41, -20.84, -20.75, -20.29, -20.89, -19.69, -20.41, -21.24,
               -19.33, -25.87, -25.4, -27.23, -27.52, -24.55, -17.36, -24.7,
               -27.76, -28.92, -25.98, -26.77, -28.76, -27.7, -24.75, -25.47,
               -26.58, -28.94, -29.13, -26.65, -28.04, -27.5, -29.28, -27.85,
               -27.41, -27.57, -29.06, -25.98, -28.21, -25.27, -14.43, -27.4,
               -27.76, -28.45, -27.35, -28.83, -29.39, -28.86, -28.61, -29.27,
               -20.32, -28.21, -26.3, -28.27, -27.75, -28.55, -27.38, -29.13,
               -28.66, -29.02, -26.04, -26.06, -28.52, -28.51, -27.93, -29.07,
               -28.41, -26.42, -27.71, -27.75, -24.28, -28.43, -25.94, -28,
               -28.59, -22.61, -27.34, -27.35, -29.14, 10.22, 10.37, 10.44,
               10.52, 10.19, 10.45, 9.91, 11.27,
               9.34, 11.68, 12.29, 11.04, 11.46, 11.73, 12.29, 11.79, 11.49,
               11.73, 11.1, 11.36, 12.19, 11.03, 11.21, 10.58, 11.61, 12.16,
               10.7, 11.47, 12.07, 11.75, 11.86, 12.33, 12.36, 11.13, 10.92,
```

```
                12.42, 10.95, 12.28, 11.04, 10.76, 10.99, 10.78, 11.07, 10.2,
                11.67, 7.53, 10.65, 10.58, 11.13, 7.73, 10.79, 10.47, 10.82,
                10.41, 11.1, 10.95, 10.76, 10.83, 10.25, 10.52, 9.94, 9.94,
                11.61,
                10.65, 10.76, 11.11, 10.2, 11.27, 10.21, 10.88, 11.21, 11.36,
                10.75, 12.38, 11.16, 11.57, 10.79, 11.13, 10.72, 10.99, 10.38,
                10.95, 10.75, 10.75, 11.05, 10.66, 10.61, 10.9, 11.14, 10.33,
                10.83, 10.75, 9.18, 9.03, 9.05, 8.6, 8.29, 10.32, 10.28, 6.47,
                11.36, 10.75, 11.13, 11.37, 10.86, 10.54, 10.39, 10.66, 9.99,
                11.65, 11.02, 10.67, 8.15, 11.12, 10.95, 11.2, 10.76, 11.32,
                10.85, 11.74, 10.46, 10.93, 12.3, 10.67, 11.51, 10.56, 12.51,
                13.51, 11.98, 12.2, 10.48, 12.4, 13, 11.36, 12.08, 12.39, 12.28,
                12.6, 11.3, 11.1, 11.42, 11.49, 12, 13.35, 11.97, 13.35, 12.75,
                12.55, 12.3, 12.51, 12.61, 10.98, 11.82, 12.27, 12.11, 12.11,
                12.89, 12.99, 12.29, 11.89, 12.74, 12.29, 11.89, 10.56, 9.27,
                10.54, 10.97, 10.46, 10.56, 10.86, 10.9, 11.06, 10.76, 10.64,
                10.94, 10.85, 10.45, 11.15, 11.23, 11.16, 10.94, 11.2, 10.71,
                9.55, 8.6, 9.67, 8.17, 9.81, 10.94, 9.49, 9.46, 7.94, 9.77,
                8.07,
                8.39, 8.95, 9.83, 8.51, 8.86, 7.93, 8, 8.33, 8, 9.39, 8.01,
                7.59,
                8.26, 9.49, 8.23, 9.1, 8.21, 9.59, 9.37, 9.47, 8.6, 8.23, 8.39,
                8.24, 8.34, 8.36, 7.22, 7.13, 10.64, 8.06, 8.22, 8.92, 9.35,
                7.32, 7.66, 8.09, 7.3, 7.33, 7.33, 7.36, 7.49, 8.07, 8.84,
                7.93,
                7.94, 8.74, 8.26, 9.63, 8.85, 7.55, 10.05, 8.23, 7.74, 9.12,
                7.33, 7.54, 8.8), ncol=2, nrow=251)
colnames(mix) = c('d13C','d15N')
s_names = c("Zostera", "Grass", "U.lactuca", "Enteromorpha")
s_means = matrix(c(-11.17, -30.88, -11.17,
                   -14.06, 6.49, 4.43, 11.19, 9.82), ncol=2, nrow=4)
s_sds = matrix(c(1.21, 0.64, 1.96, 1.17, 1.46, 2.27, 1.11, 0.83), ncol=2, nrow=4)
c_means = matrix(c(1.63, 1.63, 1.63, 1.63, 3.54, 3.54, 3.54, 3.54), ncol=2, nrow=4)
c_sds = matrix(c(0.63, 0.63, 0.63, 0.63, 0.74, 0.74, 0.74, 0.74), ncol=2, nrow=4)
conc = matrix(c(0.36, 0.4, 0.21, 0.18, 0.03, 0.04, 0.02, 0.01), ncol=2, nrow=4)
grp = as.integer(c(1, 1, 1, 1, 1, 1, 1, 1, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
        2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 3, 3, 3,
        3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3,
        3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3,
        3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3,
        3, 3, 3, 3, 3, 3, 3, 3, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 5, 5, 5,
        5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5,
        5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 6, 6, 6, 6,
        6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 7, 7, 7, 7, 7,
        7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7,
        7, 7, 7, 7, 7, 7, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8,
        8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8))

# Load this in:
simmr_4 = simmr_load(mixtures=mix,
                     source_names=s_names,
                     source_means=s_means,
                     source_sds=s_sds,
```

```
                    correction_means=c_means,
                    correction_sds=c_sds,
                    concentration_means = conc,
                    group=grp)

# Print
simmr_4

# Plot
plot(simmr_4,group=1:8,title='Isospace plot of Inger et al Geese data')
```

`plot.simmr_output` *Plot different features of an object created from* `simmr_mcmc`.

Description

This function allows for 4 different types of plots of the simmr output created from `simmr_mcmc`. The types are: histogram, kernel density plot, matrix plot (most useful) and boxplot. There are some minor customisation options.

Usage

```
## S3 method for class 'simmr_output'
plot(x, type = c("isospace", "histogram",
  "density", "matrix", "boxplot"), group = 1, binwidth = 0.05,
  alpha = 0.5, title = if (length(group) == 1) {
  "simmr output plot" } else {     paste("simmr output plot: group", group)
  }, ggargs = NULL, ...)
```

Arguments

`x`	An object of class `simmr_output` created via `simmr_mcmc`
`type`	The type of plot required. Can be one or more of 'histogram', 'density', 'matrix', or 'boxplot'
`group`	Which group to plot. Currently only one group allowed at a time
`binwidth`	The width of the bins for the histogram. Defaults to 0.05
`alpha`	The degree of transparency of the plots. Not relevant for matrix plots
`title`	The title of the plot.
`ggargs`	Extra arguments to be included in the ggplot (e.g. axis limits)
`...`	Currently not used

Details

The matrix plot should form a necessary part of any SIMM analysis since it allows the user to judge which sources are identifiable by the model. Further detail about these plots is provided in the vignette.

Author(s)

Andrew Parnell <andrew.parnell@mu.ie>

See Also

See simmr_mcmc for creating objects suitable for this function, and many more examples. See also simmr_load for creating simmr objects, plot.simmr_input for creating isospace plots, summary.simmr_output for summarising output.

Examples

```
## Not run:
# A simple example with 10 observations, 2 tracers and 4 sources

# The data
mix = matrix(c(-10.13, -10.72, -11.39, -11.18, -10.81, -10.7, -10.54,
-10.48, -9.93, -9.37, 11.59, 11.01, 10.59, 10.97, 11.52, 11.89,
11.73, 10.89, 11.05, 12.3), ncol=2, nrow=10)
colnames(mix) = c('d13C','d15N')
s_names=c('Source A','Source B','Source C','Source D')
s_means = matrix(c(-14, -15.1, -11.03, -14.44, 3.06, 7.05, 13.72, 5.96), ncol=2, nrow=4)
s_sds = matrix(c(0.48, 0.38, 0.48, 0.43, 0.46, 0.39, 0.42, 0.48), ncol=2, nrow=4)
c_means = matrix(c(2.63, 1.59, 3.41, 3.04, 3.28, 2.34, 2.14, 2.36), ncol=2, nrow=4)
c_sds = matrix(c(0.41, 0.44, 0.34, 0.46, 0.46, 0.48, 0.46, 0.66), ncol=2, nrow=4)
conc = matrix(c(0.02, 0.1, 0.12, 0.04, 0.02, 0.1, 0.09, 0.05), ncol=2, nrow=4)

# Load into simmr
simmr_1 = simmr_load(mixtures=mix,
                     source_names=s_names,
                     source_means=s_means,
                     source_sds=s_sds,
                     correction_means=c_means,
                     correction_sds=c_sds,
                     concentration_means = conc)

# Plot
plot(simmr_1)


# MCMC run
simmr_1_out = simmr_mcmc(simmr_1)

# Plot
plot(simmr_1_out) # Creates all 4 plots
plot(simmr_1_out,type='boxplot')
plot(simmr_1_out,type='histogram')
plot(simmr_1_out,type='density')
plot(simmr_1_out,type='matrix')

## End(Not run)
```

`posterior_predictive` *Plot the posterior predictive distribution for a simmr run*

Description

This function takes the output from `simmr_mcmc` and plots the posterior predictive distribution to enable visualisation of model fit. The simulated posterior predicted values are returned as part of the object and can be saved for external use

Usage

```
posterior_predictive(simmr_out, group = 1, prob = 0.5,
  plot_ppc = TRUE)
```

Arguments

`simmr_out`	A run of the simmr model from `simmr_mcmc`
`group`	Which group to run it for (currently only numeric rather than group names)
`prob`	The probability interval for the posterior predictives. The default is 0.5 (i.e. 50pc intervals)
`plot_ppc`	Whether to create a bayesplot of the posterior predictive or not.

Examples

```
## Not run:
mix = matrix(c(-10.13, -10.72, -11.39, -11.18, -10.81, -10.7, -10.54,
-10.48, -9.93, -9.37, 11.59, 11.01, 10.59, 10.97, 11.52, 11.89,
11.73, 10.89, 11.05, 12.3), ncol=2, nrow=10)
colnames(mix) = c('d13C','d15N')
s_names=c('Source A','Source B','Source C','Source D')
s_means = matrix(c(-14, -15.1, -11.03, -14.44, 3.06, 7.05, 13.72, 5.96), ncol=2, nrow=4)
s_sds = matrix(c(0.48, 0.38, 0.48, 0.43, 0.46, 0.39, 0.42, 0.48), ncol=2, nrow=4)
c_means = matrix(c(2.63, 1.59, 3.41, 3.04, 3.28, 2.34, 2.14, 2.36), ncol=2, nrow=4)
c_sds = matrix(c(0.41, 0.44, 0.34, 0.46, 0.46, 0.48, 0.46, 0.66), ncol=2, nrow=4)
conc = matrix(c(0.02, 0.1, 0.12, 0.04, 0.02, 0.1, 0.09, 0.05), ncol=2, nrow=4)

# Load into simmr
simmr_1 = simmr_load(mixtures=mix,
                     source_names=s_names,
                     source_means=s_means,
                     source_sds=s_sds,
                     correction_means=c_means,
                     correction_sds=c_sds,
                     concentration_means = conc)

# Plot
plot(simmr_1)
```

```
# Print
simmr_1

# MCMC run
simmr_1_out = simmr_mcmc(simmr_1)

# Prior predictive
post_pred = posterior_predictive(simmr_1_out)

## End(Not run)
```

`print.simmr_input` *Print simmr input object*

Description

Print simmr input object

Usage

```
## S3 method for class 'simmr_input'
print(x, ...)
```

Arguments

`x`	An object of class `simmr_input`
`...`	Other arguments (not supported)

Value

A neat presentation of your simmr object.

`print.simmr_output` *Print a simmr output object*

Description

Print a simmr output object

Usage

```
## S3 method for class 'simmr_output'
print(x, ...)
```

Arguments

x	An object of class simmr_output
...	Other arguments (not supported)

Value

Returns a neat summary of the object

See Also

simmr_mcmc for creating simmr_output objects

---

prior_viz *Plot the prior distribution for a simmr run*

Description

This function takes the output from simmr_mcmc and plots the prior distribution to enable visual inspection. This can be used by itself or as part of posterior_predictive to visually evaluate the influence of the prior on the posterior distribution.

Usage

```
prior_viz(simmr_out, group = 1, plot = TRUE,
  include_posterior = TRUE, n_sims = 10000)
```

Arguments

simmr_out	A run of the simmr model from simmr_mcmc
group	Which group to run it for (currently only numeric rather than group names)
plot	Whether to create a density plot of the prior or not. The simulated prior values are returned as part of the object
include_posterior	Whether to include the posterior distribution on top of the priors. Defaults to TRUE
n_sims	The number of simulations from the prior distribution

Examples

```
## Not run:
mix = matrix(c(-10.13, -10.72, -11.39, -11.18, -10.81, -10.7, -10.54,
-10.48, -9.93, -9.37, 11.59, 11.01, 10.59, 10.97, 11.52, 11.89,
11.73, 10.89, 11.05, 12.3), ncol=2, nrow=10)
colnames(mix) = c('d13C','d15N')
s_names=c('Source A','Source B','Source C','Source D')
s_means = matrix(c(-14, -15.1, -11.03, -14.44, 3.06, 7.05, 13.72, 5.96), ncol=2, nrow=4)
```

```
s_sds = matrix(c(0.48, 0.38, 0.48, 0.43, 0.46, 0.39, 0.42, 0.48), ncol=2, nrow=4)
c_means = matrix(c(2.63, 1.59, 3.41, 3.04, 3.28, 2.34, 2.14, 2.36), ncol=2, nrow=4)
c_sds = matrix(c(0.41, 0.44, 0.34, 0.46, 0.46, 0.48, 0.46, 0.66), ncol=2, nrow=4)
conc = matrix(c(0.02, 0.1, 0.12, 0.04, 0.02, 0.1, 0.09, 0.05), ncol=2, nrow=4)

# Load into simmr
simmr_1 = simmr_load(mixtures=mix,
                     source_names=s_names,
                     source_means=s_means,
                     source_sds=s_sds,
                     correction_means=c_means,
                     correction_sds=c_sds,
                     concentration_means = conc)

# Plot
plot(simmr_1)

# Print
simmr_1

# MCMC run
simmr_1_out = simmr_mcmc(simmr_1)

# Prior predictive
prior = prior_viz(simmr_1_out)
head(prior)
summary(prior)

## End(Not run)
```

`simmr` *simmr: A package for fitting stable isotope mixing models via JAGS in R*

Description

This package runs a simple Stable Isotope Mixing Model (SIMM) and is meant as a longer term replacement to the previous function SIAR.. These are used to infer dietary proportions of organisms consuming various food sources from observations on the stable isotope values taken from the organisms' tissue samples. However SIMMs can also be used in other scenarios, such as in sediment mixing or the composition of fatty acids. The main functions are `simmr_load` and `simmr_mcmc`. The help files contain examples of the use of this package. See also the vignette for a longer walkthrough.

Details

An even longer term replacement for properly running SIMMs is MixSIAR, which allows for more detailed random effects and the inclusion of covariates.

Author(s)

Andrew Parnell <andrew.parnell@mu.ie>

References

Andrew C. Parnell, Donald L. Phillips, Stuart Bearhop, Brice X. Semmens, Eric J. Ward, Jonathan W. Moore, Andrew L. Jackson, Jonathan Grey, David J. Kelly, and Richard Inger. Bayesian stable isotope mixing models. Environmetrics, 24(6):387–399, 2013.

Andrew C Parnell, Richard Inger, Stuart Bearhop, and Andrew L Jackson. Source partitioning using stable isotopes: coping with too much variation. PLoS ONE, 5(3):5, 2010.

Examples

```
## Not run:
# A first example with 2 tracers (isotopes), 10 observations, and 4 food sources

# Add in the data
mix = matrix(c(-10.13, -10.72, -11.39, -11.18, -10.81, -10.7, -10.54,
-10.48, -9.93, -9.37, 11.59, 11.01, 10.59, 10.97, 11.52, 11.89,
11.73, 10.89, 11.05, 12.3), ncol=2, nrow=10)
colnames(mix) = c('d13C','d15N')
s_names=c('Source A','Source B','Source C','Source D')
s_means = matrix(c(-14, -15.1, -11.03, -14.44, 3.06, 7.05, 13.72, 5.96), ncol=2, nrow=4)
s_sds = matrix(c(0.48, 0.38, 0.48, 0.43, 0.46, 0.39, 0.42, 0.48), ncol=2, nrow=4)
c_means = matrix(c(2.63, 1.59, 3.41, 3.04, 3.28, 2.34, 2.14, 2.36), ncol=2, nrow=4)
c_sds = matrix(c(0.41, 0.44, 0.34, 0.46, 0.46, 0.48, 0.46, 0.66), ncol=2, nrow=4)
conc = matrix(c(0.02, 0.1, 0.12, 0.04, 0.02, 0.1, 0.09, 0.05), ncol=2, nrow=4)

# Load into simmr
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,
                      concentration_means = conc)

# Plot
plot(simmr_in)

# MCMC run
simmr_out = simmr_mcmc(simmr_in)

# Summary
summary(simmr_out)

# Plot
plot(simmr_out)

## End(Not run)
```

`simmr_elicit` *Function to allow informative prior distribution to be included in simmr*

Description

The main `simmr_mcmc` function allows for a prior distribution to be set for the dietary proportions. The prior distribution is specified by transforming the dietary proportions using the centralised log ratio (CLR). The `simmr_elicit` and `simmr_elicit` functions allows the user to specify prior means and standard deviations for each of the dietary proportions, and then finds CLR-transformed values suitable for input into `simmr_mcmc`.

Usage

```
simmr_elicit(n_sources, proportion_means = rep(1/n_sources, n_sources),
  proportion_sds = rep(0.1, n_sources), n_sims = 1000)
```

Arguments

`n_sources`	The number of sources required
`proportion_means`	The desired prior proportion means. These should sum to 1. Should be a vector of length `n_sources`
`proportion_sds`	The desired prior proportions standard deviations. These have no restricted sum but should be reasonable estimates for a proportion.
`n_sims`	The number of simulations for which to run the optimisation routine.

Details

The function takes the desired proportion means and standard deviations, and fits an optimised least squares to the means and standard deviations in turn to produced CLR-transformed estimates for use in `simmr_mcmc`. Using prior information in SIMMs is highly desirable given the restricted nature of the inference. The prior information might come from previous studies, other experiments, or other observations of e.g. animal behaviour.

Due to the nature of the restricted space over which the dietary proportions can span, and the fact that this function uses numerical optimisation, the procedure will not match the target dietary proportion means and standard deviations exactly. If this problem is severe, try increasing the `n_sims` value.

Value

A list object with two components

`mean`	The best estimates of the mean to use in `control.prior` in `simmr_mcmc`
`sd`	The best estimates of the standard deviations to use in `control.prior` in `simmr_mcmc`

Author(s)

Andrew Parnell <andrew.parnell@mu.ie>

Examples

```
## Not run:
mix = matrix(c(-10.13, -10.72, -11.39, -11.18, -10.81, -10.7, -10.54,
               -10.48, -9.93, -9.37, 11.59, 11.01, 10.59, 10.97, 11.52, 11.89,
               11.73, 10.89, 11.05, 12.3), ncol=2, nrow=10)
colnames(mix) = c('d13C','d15N')
s_names=c('Source A','Source B','Source C','Source D')
s_means = matrix(c(-14, -15.1, -11.03, -14.44, 3.06, 7.05, 13.72, 5.96), ncol=2, #' nrow=4)
s_sds = matrix(c(0.48, 0.38, 0.48, 0.43, 0.46, 0.39, 0.42, 0.48), ncol=2, nrow=4)
c_means = matrix(c(2.63, 1.59, 3.41, 3.04, 3.28, 2.34, 2.14, 2.36), ncol=2, nrow=4)
c_sds = matrix(c(0.41, 0.44, 0.34, 0.46, 0.46, 0.48, 0.46, 0.66), ncol=2, nrow=4)
conc = matrix(c(0.02, 0.1, 0.12, 0.04, 0.02, 0.1, 0.09, 0.05), ncol=2, nrow=4)


# Load into simmr
simmr_1 = simmr_load(mixtures=mix,
                     source_names=s_names,
                     source_means=s_means,
                     source_sds=s_sds,
                     correction_means=c_means,
                     correction_sds=c_sds,
                     concentration_means = conc)

# MCMC run
simmr_1_out = simmr_mcmc(simmr_1)

# Summary
summary(simmr_1_out,'quantiles')
# A bit vague:
#           2.5%   25%   50%   75% 97.5%
# Source A 0.029 0.115 0.203 0.312 0.498
# Source B 0.146 0.232 0.284 0.338 0.453
# Source C 0.216 0.255 0.275 0.296 0.342
# Source D 0.032 0.123 0.205 0.299 0.465

# Now suppose I had prior information that:
# proportion means = 0.5,0.2,0.2,0.1
# proportion sds = 0.08,0.02,0.01,0.02
prior=simmr_elicit(4, c(0.5,0.2,0.2,0.1),c(0.08,0.02,0.01,0.02))

simmr_1a_out = simmr_mcmc(simmr_1,prior_control=list(means=prior$mean,sd=prior$sd#' ))

summary(simmr_1a_out,'quantiles')
# Much more precise:
#           2.5%   25%   50%   75% 97.5%
# Source A 0.441 0.494 0.523 0.553 0.610
# Source B 0.144 0.173 0.188 0.204 0.236
```

```
# Source C 0.160 0.183 0.196 0.207 0.228
# Source D 0.060 0.079 0.091 0.105 0.135

## End(Not run)
```

`simmr_load` *Function to load in simmr data and check for errors*

Description

This function takes in the mixture data, food source means and standard deviations, and (optionally) correction factor means and standard deviations, and concentration proportions. It performs some (non-exhaustive) checking of the data to make sure it will run through simmr. It outputs an object of class `simmr_input`.

Usage

```
simmr_load(mixtures, source_names, source_means, source_sds,
  correction_means = NULL, correction_sds = NULL,
  concentration_means = NULL, group = NULL)
```

Arguments

`mixtures`	The mixture data given as a matrix where the number of rows is the number of observations and the number of columns is the number of tracers (usually isotopes)
`source_names`	The names of the sources given as a character string
`source_means`	The means of the source values, given as a matrix where the number of rows is the number of sources and the number of columns is the number of tracers
`source_sds`	The standard deviations of the source values, given as a matrix where the number of rows is the number of sources and the number of columns is the number of tracers
`correction_means`	The means of the correction values, given as a matrix where the number of rows is the number of sources and the number of columns is the number of tracers. If not provided these are set to 0.
`correction_sds`	The standard deviations of the correction values, given as a matrix where the number of rows is the number of sources and the number of columns is the number of tracers. If not provided these are set to 0.
`concentration_means`	The means of the concentration values, given as a matrix where the number of rows is the number of sources and the number of columns is the number of tracers. These should be between 0 and 1. If not provided these are all set to 1.
`group`	A grouping variable. These can be a character or factor variable

Details

For standard stable isotope mixture modelling, the mixture matrix will contain a row for each individual and a column for each isotopic value. `simmr` will allow for any number of isotopes and any number of observations, within computational limits. The source means/sds should be provided for each food source on each isotope. The correction means (usually trophic enrichment factors) can be set as zero if required, and should be of the same shape as the source values. The concentration dependence means should be estimated values of the proportion of each element in the food source in question and should be given in proportion format between 0 and 1. At present there is no means to include concentration standard deviations.

Value

An object of class `simmr_input` with the following elements:

`mixtures`	The mixture data
`source_neams`	Source means
`sources_sds`	Source standard deviations
`correction_means`	Correction means
`correction_sds`	Correction standard deviations
`concentration_means`	Concentration dependence means
`n_obs`	The number of observations
`n_tracers`	The number of tracers/isotopes
`n_sources`	The number of sources
`n_groups`	The number of groups

Author(s)

Andrew Parnell <andrew.parnell@mu.ie>

See Also

See `simmr_mcmc` for complete examples.

Examples

```
# A simple example with 10 observations, 2 tracers and 4 sources
mix = matrix(c(-10.13, -10.72, -11.39, -11.18, -10.81, -10.7, -10.54,
-10.48, -9.93, -9.37, 11.59, 11.01, 10.59, 10.97, 11.52, 11.89,
11.73, 10.89, 11.05, 12.3), ncol=2, nrow=10)
colnames(mix) = c('d13C','d15N')
s_names=c('Source A','Source B','Source C','Source D')
s_means = matrix(c(-14, -15.1, -11.03, -14.44, 3.06, 7.05, 13.72, 5.96), ncol=2, nrow=4)
s_sds = matrix(c(0.48, 0.38, 0.48, 0.43, 0.46, 0.39, 0.42, 0.48), ncol=2, nrow=4)
c_means = matrix(c(2.63, 1.59, 3.41, 3.04, 3.28, 2.34, 2.14, 2.36), ncol=2, nrow=4)
```

```
c_sds = matrix(c(0.41, 0.44, 0.34, 0.46, 0.46, 0.48, 0.46, 0.66), ncol=2, nrow=4)
conc = matrix(c(0.02, 0.1, 0.12, 0.04, 0.02, 0.1, 0.09, 0.05), ncol=2, nrow=4)

# Load in with simmr_load
simmr_1 = simmr_load(mixtures=mix,
                     source_names=s_names,
                     source_means=s_means,
                     source_sds=s_sds,
                     correction_means=c_means,
                     correction_sds=c_sds,
                     concentration_means = conc)

print(simmr_1)
```

`simmr_mcmc` *Run a* `simmr_input` *object through the main simmr Markov chain Monte Carlo (MCMC) function*

Description

This is the main function of simmr. It takes a `simmr_input` object created via `simmr_load`, runs an MCMC to determine the dietary proportions, and then outputs a `simmr_output` object for further analysis and plotting via `summary.simmr_output` and `plot.simmr_output`.

Usage

```
simmr_mcmc(simmr_in, prior_control = list(means = rep(0,
  simmr_in$n_sources), sd = rep(1, simmr_in$n_sources)),
  mcmc_control = list(iter = 10000, burn = 1000, thin = 10, n.chain = 4))
```

Arguments

`simmr_in`	An object created via the function `simmr_load`
`prior_control`	A list of values including arguments named `means` and `sd` which represent the prior means and standard deviations of the dietary proportions in centralised log-ratio space. These can usually be left at their default values unless you wish to include to include prior information, in which case you should use the function `simmr_elicit`.
`mcmc_control`	A list of values including arguments named `iter` (number of iterations), `burn` (size of burn-in), `thin` (amount of thinning), and `n.chain` (number of MCMC chains).

Details

If, after running `simmr_mcmc` the convergence diagnostics in `summary.simmr_output` are not satisfactory, the values of `iter`, `burn` and `thin` in `mcmc_control` should be increased by a factor of 10.

Value

An object of class `simmr_output` with two named top-level components:

`input`	The `simmr_input` object given to the `simmr_mcmc` function
`output`	A set of MCMC chains of class `mcmc.list` from the coda package. These can be analysed using the `summary.simmr_output` and `plot.simmr_output` functions.

Author(s)

Andrew Parnell <andrew.parnell@mu.ie>

References

Andrew C. Parnell, Donald L. Phillips, Stuart Bearhop, Brice X. Semmens, Eric J. Ward, Jonathan W. Moore, Andrew L. Jackson, Jonathan Grey, David J. Kelly, and Richard Inger. Bayesian stable isotope mixing models. Environmetrics, 24(6):387–399, 2013.

Andrew C Parnell, Richard Inger, Stuart Bearhop, and Andrew L Jackson. Source partitioning using stable isotopes: coping with too much variation. PLoS ONE, 5(3):5, 2010.

See Also

`simmr_load` for creating objects suitable for this function, `plot.simmr_input` for creating isospace plots, `summary.simmr_output` for summarising output, and `plot.simmr_output` for plotting output.

Examples

```
## Not run:
## See the package vignette for a detailed run through of these 4 examples

# Data set 1: 10 obs on 2 isos, 4 sources, with tefs and concdep

# The data
mix = matrix(c(-10.13, -10.72, -11.39, -11.18, -10.81, -10.7, -10.54,
-10.48, -9.93, -9.37, 11.59, 11.01, 10.59, 10.97, 11.52, 11.89,
11.73, 10.89, 11.05, 12.3), ncol=2, nrow=10)
colnames(mix) = c('d13C','d15N')
s_names=c('Source A','Source B','Source C','Source D')
s_means = matrix(c(-14, -15.1, -11.03, -14.44, 3.06, 7.05, 13.72, 5.96), ncol=2, nrow=4)
s_sds = matrix(c(0.48, 0.38, 0.48, 0.43, 0.46, 0.39, 0.42, 0.48), ncol=2, nrow=4)
c_means = matrix(c(2.63, 1.59, 3.41, 3.04, 3.28, 2.34, 2.14, 2.36), ncol=2, nrow=4)
c_sds = matrix(c(0.41, 0.44, 0.34, 0.46, 0.46, 0.48, 0.46, 0.66), ncol=2, nrow=4)
conc = matrix(c(0.02, 0.1, 0.12, 0.04, 0.02, 0.1, 0.09, 0.05), ncol=2, nrow=4)

# Load into simmr
simmr_1 = simmr_load(mixtures=mix,
                     source_names=s_names,
                     source_means=s_means,
                     source_sds=s_sds,
```

```
                      correction_means=c_means,
                      correction_sds=c_sds,
                      concentration_means = conc)

# Plot
plot(simmr_1)

# Print
simmr_1

# MCMC run
simmr_1_out = simmr_mcmc(simmr_1)

# Print it
print(simmr_1_out)

# Summary
summary(simmr_1_out,type='diagnostics')
summary(simmr_1_out,type='correlations')
summary(simmr_1_out,type='statistics')
ans = summary(simmr_1_out,type=c('quantiles','statistics'))

# Plot
plot(simmr_1_out,type='boxplot')
plot(simmr_1_out,type='histogram')
plot(simmr_1_out,type='density')
plot(simmr_1_out,type='matrix')

# Compare two sources
compare_sources(simmr_1_out,source_names=c('Source A','Source D'))

# Compare multiple sources
compare_sources(simmr_1_out)

##############################################################################

# A version with just one observation
simmr_2 = simmr_load(mixtures=mix[1,,drop=FALSE], # drop required to keep the mixtures as a matrix
                      source_names=s_names,
                      source_means=s_means,
                      source_sds=s_sds,
                      correction_means=c_means,
                      correction_sds=c_sds,
                      concentration_means = conc)

# Plot
plot(simmr_2)

# MCMC run - automatically detects the single observation
simmr_2_out = simmr_mcmc(simmr_2)

# Print it
print(simmr_2_out)
```

```
# Summary
summary(simmr_2_out)
summary(simmr_2_out,type='diagnostics')
ans = summary(simmr_2_out,type=c('quantiles'))

# Plot
plot(simmr_2_out)
plot(simmr_2_out,type='boxplot')
plot(simmr_2_out,type='histogram')
plot(simmr_2_out,type='density')
plot(simmr_2_out,type='matrix')

###############################################################################

# Data set 2: 3 isotopes (d13C, d15N and d34S), 30 observations, 4 sources

# The data
mix = matrix(c(-11.67, -12.55, -13.18, -12.6, -11.77, -11.21, -11.45,
               -12.73, -12.49, -10.6, -12.26, -12.48, -13.07, -12.67, -12.26,
               -13.12, -10.83, -13.2, -12.24, -12.85, -11.65, -11.84, -13.26,
               -12.56, -12.97, -12.18, -12.76, -11.53, -12.87, -12.49, 7.79,
               7.85, 8.25, 9.06, 9.13, 8.56, 8.03, 7.74, 8.16, 8.43, 7.9, 8.32,
               7.85, 8.14, 8.74, 9.17, 7.33, 8.06, 8.06, 8.03, 8.16, 7.24, 7.24,
               8, 8.57, 7.98, 7.2, 8.13, 7.78, 8.21, 11.31, 10.92, 11.3, 11,
               12.21, 11.52, 11.05, 11.05, 11.56, 11.78, 12.3, 10.87, 10.35,
               11.66, 11.46, 11.55, 11.41, 12.01, 11.97, 11.5, 11.18, 11.49,
               11.8, 11.63, 10.99, 12, 10.63, 11.27, 11.81, 12.25), ncol=3, nrow=30)
colnames(mix) = c('d13C','d15N','d34S')
s_names = c('Source A', 'Source B', 'Source C', 'Source D')
s_means = matrix(c(-14, -15.1, -11.03, -14.44, 3.06, 7.05, 13.72, 5.96,
                   10.35, 7.51, 10.31, 9), ncol=3, nrow=4)
s_sds = matrix(c(0.46, 0.39, 0.42, 0.48, 0.44, 0.37, 0.49, 0.47, 0.49,
                 0.42, 0.41, 0.42), ncol=3, nrow=4)
c_means = matrix(c(1.3, 1.58, 0.81, 1.7, 1.73, 1.83, 1.69, 3.2, 0.67,
                   2.99, 3.38, 1.31), ncol=3, nrow=4)
c_sds = matrix(c(0.32, 0.64, 0.58, 0.46, 0.61, 0.55, 0.47, 0.45, 0.34,
                 0.45, 0.37, 0.49), ncol=3, nrow=4)
conc = matrix(c(0.05, 0.1, 0.06, 0.07, 0.07, 0.03, 0.07, 0.05, 0.1,
                0.05, 0.12, 0.11), ncol=3, nrow=4)

# Load into simmr
simmr_3 = simmr_load(mixtures=mix,
                     source_names=s_names,
                     source_means=s_means,
                     source_sds=s_sds,
                     correction_means=c_means,
                     correction_sds=c_sds,
                     concentration_means = conc)

# Get summary
print(simmr_3)
```

```
# Plot 3 times
plot(simmr_3)
plot(simmr_3,tracers=c(2,3))
plot(simmr_3,tracers=c(1,3))
# See vignette('simmr') for fancier axis labels

# MCMC run
simmr_3_out = simmr_mcmc(simmr_3)

# Print it
print(simmr_3_out)

# Summary
summary(simmr_3_out)
summary(simmr_3_out,type='diagnostics')
summary(simmr_3_out,type='quantiles')
summary(simmr_3_out,type='correlations')

# Plot
plot(simmr_3_out)
plot(simmr_3_out,type='boxplot')
plot(simmr_3_out,type='histogram')
plot(simmr_3_out,type='density')
plot(simmr_3_out,type='matrix')

###############################################################################

# Data set 4 - identified by Fry (2014) as a failing of SIMMs
# See the vignette for more interpreation of these data and the output

# The data
mix = matrix(c(-14.65, -16.39, -14.5, -15.33, -15.76, -15.15, -15.73,
               -15.52, -15.44, -16.19, 8.45, 8.08, 7.39, 8.68, 8.23, 7.84, 8.48,
               8.47, 8.44, 8.37), ncol=2, nrow=10)
s_names = c('Source A', 'Source B', 'Source C', 'Source D')
s_means = matrix(c(-25, -25, -5, -5, 4, 12, 12, 4), ncol=2, nrow=4)
s_sds = matrix(c(1, 1, 1, 1, 1, 1, 1, 1), ncol=2, nrow=4)

# Load into simmr - note no corrections or concentrations
simmr_4 = simmr_load(mixtures=mix,
                     source_names=s_names,
                     source_means=s_means,
                     source_sds=s_sds)

# Get summary
print(simmr_4)

# Plot
plot(simmr_4)

# MCMC run - needs slightly longer
simmr_4_out = simmr_mcmc(simmr_4,
mcmc_control=list(iter=100000,burn=10000,thin=100,n.chain=4))
```

```
# Print it
print(simmr_4_out)

# Summary
summary(simmr_4_out)
summary(simmr_4_out,type='diagnostics')
ans = summary(simmr_4_out,type=c('quantiles','statistics'))

# Plot
plot(simmr_4_out)
plot(simmr_4_out,type='boxplot')
plot(simmr_4_out,type='histogram')
plot(simmr_4_out,type='density')
plot(simmr_4_out,type='matrix') # Look at the massive correlations here

###############################################################################

# Data set 5 - Multiple groups Geese data from Inger et al 2006

# Do this in raw data format - Note that there's quite a few mixtures!
mix = matrix(c(10.22, 10.37, 10.44, 10.52, 10.19, 10.45, 9.91, 11.27,
               9.34, 11.68, 12.29, 11.04, 11.46, 11.73, 12.29, 11.79, 11.49,
               11.73, 11.1, 11.36, 12.19, 11.03, 11.21, 10.58, 11.61, 12.16,
               10.7, 11.47, 12.07, 11.75, 11.86, 12.33, 12.36, 11.13, 10.92,
               12.42, 10.95, 12.28, 11.04, 10.76, 10.99, 10.78, 11.07, 10.2,
               11.67, 7.53, 10.65, 10.58, 11.13, 7.73, 10.79, 10.47, 10.82,
               10.41, 11.1, 10.95, 10.76, 10.83, 10.25, 10.52, 9.94, 9.94, 11.61,
               10.65, 10.76, 11.11, 10.2, 11.27, 10.21, 10.88, 11.21, 11.36,
               10.75, 12.38, 11.16, 11.57, 10.79, 11.13, 10.72, 10.99, 10.38,
               10.95, 10.75, 10.75, 11.05, 10.66, 10.61, 10.9, 11.14, 10.33,
               10.83, 10.75, 9.18, 9.03, 9.05, 8.6, 8.29, 10.32, 10.28, 6.47,
               11.36, 10.75, 11.13, 11.37, 10.86, 10.54, 10.39, 10.66, 9.99,
               11.65, 11.02, 10.67, 8.15, 11.12, 10.95, 11.2, 10.76, 11.32,
               10.85, 11.74, 10.46, 10.93, 12.3, 10.67, 11.51, 10.56, 12.51,
               13.51, 11.98, 12.2, 10.48, 12.4, 13, 11.36, 12.08, 12.39, 12.28,
               12.6, 11.3, 11.1, 11.42, 11.49, 12, 13.35, 11.97, 13.35, 12.75,
               12.55, 12.3, 12.51, 12.61, 10.98, 11.82, 12.27, 12.11, 12.11,
               12.89, 12.99, 12.29, 11.89, 12.74, 12.29, 11.89, 10.56, 9.27,
               10.54, 10.97, 10.46, 10.56, 10.86, 10.9, 11.06, 10.76, 10.64,
               10.94, 10.85, 10.45, 11.15, 11.23, 11.16, 10.94, 11.2, 10.71,
               9.55, 8.6, 9.67, 8.17, 9.81, 10.94, 9.49, 9.46, 7.94, 9.77, 8.07,
               8.39, 8.95, 9.83, 8.51, 8.86, 7.93, 8, 8.33, 8, 9.39, 8.01, 7.59,
               8.26, 9.49, 8.23, 9.1, 8.21, 9.59, 9.37, 9.47, 8.6, 8.23, 8.39,
               8.24, 8.34, 8.36, 7.22, 7.13, 10.64, 8.06, 8.22, 8.92, 9.35,
               7.32, 7.66, 8.09, 7.3, 7.33, 7.33, 7.36, 7.49, 8.07, 8.84, 7.93,
               7.94, 8.74, 8.26, 9.63, 8.85, 7.55, 10.05, 8.23, 7.74, 9.12,
               7.33, 7.54, 8.8, -11.36, -11.88, -10.6, -11.25, -11.66, -10.41,
               -10.88, -14.73, -11.52, -15.89, -14.79, -17.64, -16.97, -17.25,
               -14.77, -15.67, -15.34, -15.53, -17.27, -15.63, -15.94, -14.88,
               -15.9, -17.11, -14.93, -16.26, -17.5, -16.37, -15.21, -15.43,
               -16.54, -15, -16.41, -15.09, -18.06, -16.27, -15.08, -14.39,
               -21.45, -22.52, -21.25, -21.84, -22.51, -21.97, -20.23, -21.64,
```

```
                -22.49, -21.91, -21.65, -21.37, -22.9, -21.13, -19.33, -20.29,
                -20.56, -20.87, -21.07, -21.69, -21.17, -21.74, -22.69, -21.06,
                -20.42, -21.5, -20.15, -21.99, -22.3, -21.71, -22.48, -21.86,
                -21.68, -20.97, -21.91, -19.05, -22.78, -22.36, -22.46, -21.52,
                -21.84, -21.3, -21.39, -22.1, -21.59, -20.14, -20.67, -20.31,
                -20.07, -21.2, -20.44, -22.06, -22.05, -21.44, -21.93, -22.47,
                -22.27, -22.19, -22.81, -20.48, -22.47, -18.06, -20.72, -20.97,
                -19.11, -18.4, -20.45, -21.2, -19.74, -20.48, -21.48, -17.81,
                -19.77, -22.56, -14.72, -12.21, -12.35, -13.88, -14.43, -14.65,
                -13.9, -14.12, -10.88, -10.44, -15.33, -13.78, -13.98, -15.22,
                -15.25, -15.76, -15.78, -15.49, -13.02, -15.3, -15.55, -14.35,
                -14.99, -14.83, -16.18, -15.01, -12.87, -14.67, -13.84, -14.89,
                -13.33, -15.04, -14.29, -15.62, -13.99, -15.06, -15.06, -15,
                -14.55, -13.32, -14.34, -14.47, -14.31, -14.18, -16.18, -16.25,
                -15.92, -15.35, -14.29, -15.92, -15.35, -20.22, -21.4, -19.97,
                -20.78, -20.61, -20.58, -20.19, -20.71, -20.59, -20.09, -19.37,
                -20.41, -20.84, -20.75, -20.29, -20.89, -19.69, -20.41, -21.24,
                -19.33, -25.87, -25.4, -27.23, -27.52, -24.55, -17.36, -24.7,
                -27.76, -28.92, -25.98, -26.77, -28.76, -27.7, -24.75, -25.47,
                -26.58, -28.94, -29.13, -26.65, -28.04, -27.5, -29.28, -27.85,
                -27.41, -27.57, -29.06, -25.98, -28.21, -25.27, -14.43, -27.4,
                -27.76, -28.45, -27.35, -28.83, -29.39, -28.86, -28.61, -29.27,
                -20.32, -28.21, -26.3, -28.27, -27.75, -28.55, -27.38, -29.13,
                -28.66, -29.02, -26.04, -26.06, -28.52, -28.51, -27.93, -29.07,
                -28.41, -26.42, -27.71, -27.75, -24.28, -28.43, -25.94, -28,
                -28.59, -22.61, -27.34, -27.35, -29.14), ncol=2, nrow=251)
    colnames(mix) = c('d13C','d15N')
    s_names = c("Zostera", "Grass", "U.lactuca", "Enteromorpha")
    s_means = matrix(c(6.49, 4.43, 11.19, 9.82, -11.17, -30.88, -11.17,
                       -14.06), ncol=2, nrow=4)
    s_sds = matrix(c(1.46, 2.27, 1.11, 0.83, 1.21, 0.64, 1.96, 1.17), ncol=2, nrow=4)
    c_means = matrix(c(3.54, 3.54, 3.54, 3.54, 1.63, 1.63, 1.63, 1.63), ncol=2, nrow=4)
    c_sds = matrix(c(0.74, 0.74, 0.74, 0.74, 0.63, 0.63, 0.63, 0.63), ncol=2, nrow=4)
    conc = matrix(c(0.03, 0.04, 0.02, 0.01, 0.36, 0.4, 0.21, 0.18), ncol=2, nrow=4)
    grp = as.integer(c(1, 1, 1, 1, 1, 1, 1, 1, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
            2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 3, 3, 3,
            3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3,
            3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3,
            3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3,
            3, 3, 3, 3, 3, 3, 3, 3, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 5, 5, 5,
            5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5,
            5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 6, 6, 6, 6,
            6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 7, 7, 7, 7, 7,
            7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7,
            7, 7, 7, 7, 7, 7, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8,
            8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8))

    # Load this in:
    simmr_5 = simmr_load(mixtures=mix,
                         source_names=s_names,
                         source_means=s_means,
                         source_sds=s_sds,
                         correction_means=c_means,
```

```
                    correction_sds=c_sds,
                    concentration_means = conc,
                    group=grp)

# Plot
plot(simmr_5,group=1:8,xlab=expression(paste(delta^13, "C (\u2030)",sep="")),
     ylab=expression(paste(delta^15, "N (\u2030)",sep="")),
     title='Isospace plot of Inger et al Geese data')

# Run MCMC for each group
simmr_5_out = simmr_mcmc(simmr_5)

# Summarise output
summary(simmr_5_out,type='quantiles',group=1)
summary(simmr_5_out,type='quantiles',group=c(1,3))
summary(simmr_5_out,type=c('quantiles','statistics'),group=c(1,3))

# Plot - only a single group allowed
plot(simmr_5_out,type='boxplot',group=2,title='simmr output group 2')
plot(simmr_5_out,type=c('density','matrix'),grp=6,title='simmr output group 6')

# Compare sources within a group
compare_sources(simmr_5_out,source_names=c('Zostera','U.lactuca'),group=2)
compare_sources(simmr_5_out,group=2)

# Compare between groups
compare_groups(simmr_5_out,source='Zostera',groups=1:2)
compare_groups(simmr_5_out,source='Zostera',groups=1:3)
compare_groups(simmr_5_out,source='U.lactuca',groups=c(4:5,7,2))



## End(Not run)
```

`simmr_mcmc_tdf` *Estimate correction factors from stable isotope data with known dietary proportions*

Description

This function runs a slightly different version of the main `simmr_mcmc` function with the key difference that it estimates the correction factors (sometimes called trophic enrichment or trophic discrimination factors; TEFs/TDFs) for a given set of dietary proportions.

Usage

```
simmr_mcmc_tdf(simmr_in, p = matrix(rep(1/simmr_in$n_sources,
  simmr_in$n_sources), ncol = simmr_in$n_sources, nrow = simmr_in$n_obs,
  byrow = TRUE), prior_control = list(c_mean_est = rep(2,
```

```
  simmr_in$n_tracers), c_sd_est = rep(2, simmr_in$n_tracers)),
  mcmc_control = list(iter = 10000, burn = 1000, thin = 10, n.chain = 4))
```

Arguments

simmr_in	An object created via the function simmr_load
p	The known dietary proportions for the feeding study. Dietary proportions should be given per individual (even if they are all identical)
prior_control	A list of values including arguments named means and sd which represent the prior means and standard deviations of the correction factors. These can usually be left at their default values unless you wish to include to include prior information on them.
mcmc_control	A list of values including arguments named iter (number of iterations), burn (size of burn-in), thin (amount of thinning), and n.chain (number of MCMC chains).

Details

The idea is that this code can be used for feeding studies where an organism is fed a known proportional diet with a view to estimating the correction factors to be used in a later stable isotope mixing model when the organisms are observed in the field.

The main argument of the function is an object created from simmr_load which contains mixture data on a number of tracers and food source means and standard deviations. Any correction factors included in this object will be ignored. The known dietary proportions should be provided for each individual (i.e. should be a matrix with the same number of rows as mix). It is advisable to have multiple different dietary proportion values as part of the feeding experimental design

The output of the function is a posterior distribution on the correction factors for each food source. Just like the output from simmr_mcmc, this should be checked for convergence. Examples are included below to help assist with this check and further plots

If, after running simmr_mcmc_tdf the convergence diagnostics in summary.simmr_output_tdf are not satisfactory, the values of iter, burn and thin in mcmc_control should be increased by e.g. a factor of 10.

Value

An object of class simmr_tdf with two named top-level components:

input	The simmr_input object given to the simmr_mcmc function
output	A set of MCMC chains of class mcmc.list from the coda package. These can be analysed using summary.simmr_output_tdf

.

Author(s)

Andrew Parnell <andrew.parnell@mu.ie>

References

Andrew C. Parnell, Donald L. Phillips, Stuart Bearhop, Brice X. Semmens, Eric J. Ward, Jonathan W. Moore, Andrew L. Jackson, Jonathan Grey, David J. Kelly, and Richard Inger. Bayesian stable isotope mixing models. Environmetrics, 24(6):387–399, 2013.

Andrew C Parnell, Richard Inger, Stuart Bearhop, and Andrew L Jackson. Source partitioning using stable isotopes: coping with too much variation. PLoS ONE, 5(3):5, 2010.

See Also

simmr_load for creating objects suitable for this function, simmr_mcmc for estimating dietary proportions, plot.simmr_input for creating isospace plots, summary.simmr_output_tdf for summarising output

Examples

```
## Not run:
## Example of estimating TDFs for a simple system with known dietary
proportions

# Data set 1: 10 obs on 2 isos, 4 sources, with tefs and concdep
# Assume p = c(0.25, 0.25, 0.25, 0.25)

# The data
mix = matrix(c(-10.13, -10.72, -11.39, -11.18, -10.81, -10.7, -10.54,
-10.48, -9.93, -9.37, 11.59, 11.01, 10.59, 10.97, 11.52, 11.89,
11.73, 10.89, 11.05, 12.3), ncol=2, nrow=10)
colnames(mix) = c('d13C','d15N')
s_names=c('Source A','Source B','Source C','Source D')
s_means = matrix(c(-14, -15.1, -11.03, -14.44, 3.06, 7.05, 13.72, 5.96), ncol=2, nrow=4)
s_sds = matrix(c(0.48, 0.38, 0.48, 0.43, 0.46, 0.39, 0.42, 0.48), ncol=2, nrow=4)
conc = matrix(c(0.02, 0.1, 0.12, 0.04, 0.02, 0.1, 0.09, 0.05), ncol=2, nrow=4)

# Load into simmr
simmr_tdf = simmr_load(mixtures=mix,
                    source_names=s_names,
                    source_means=s_means,
                    source_sds=s_sds,
                    concentration_means = conc)

# Plot
plot(simmr_tdf)

# MCMC run
simmr_tdf_out = simmr_mcmc_tdf(simmr_tdf,
p = matrix(rep(1/simmr_tdf$n_sources,
simmr_tdf$n_sources),
ncol = simmr_tdf$n_sources,
nrow = simmr_tdf$n_obs, byrow = TRUE))

# Summary
summary(simmr_tdf_out,type='diagnostics')
```

```
summary(simmr_tdf_out,type='quantiles')

# Now put these corrections back into the model and check the
# iso-space plots and dietary output
simmr_tdf_2 = simmr_load(mixtures=mix,
                     source_names=s_names,
                     source_means=s_means,
                     source_sds=s_sds,
                     correction_means = simmr_tdf_out$c_mean_est,
                     correction_sds = simmr_tdf_out$c_sd_est,
                     concentration_means = conc)

# Plot with corrections now
plot(simmr_tdf_2)

simmr_tdf_2_out = simmr_mcmc(simmr_tdf_2)
summary(simmr_tdf_2_out, type = 'diagnostics')
plot(simmr_tdf_2_out, type = 'boxplot')

## End(Not run)
```

`summary.simmr_output` *Summarises the output created with* `simmr_mcmc`

Description

Produces textual summaries and convergence diagnostics for an object created with `simmr_mcmc`. The different options are: 'diagnostics' which produces Brooks-Gelman-Rubin diagnostics to assess MCMC convergence, 'quantiles' which produces credible intervals for the parameters, 'statistics' which produces means and standard deviations, and 'correlations' which produces correlations between the parameters.

Usage

```
## S3 method for class 'simmr_output'
summary(object, type = c("diagnostics",
  "quantiles", "statistics", "correlations"), group = 1, ...)
```

Arguments

`object`	An object of class `simmr_output` produced by the function `simmr_mcmc`
`type`	The type of output required. At least none of 'diagnostics', 'quantiles', 'statistics', or 'correlations'.
`group`	Which group or groups the output is required for.
`...`	Not used

Details

The quantile output allows easy calculation of 95 per cent credible intervals of the posterior dietary proportions. The correlations, along with the matrix plot in `plot.simmr_output` allow the user to judge which sources are non-identifiable. The Gelman diagnostic values should be close to 1 to ensure satisfactory convergence.

When multiple groups are included, the output automatically includes the results for all groups.

Value

An list containing the following components:

`gelman`	The convergence diagnostics
`quantiles`	The quantiles of each parameter from the posterior distribution
`statistics`	The means and standard deviations of each parameter
`correlations`	The posterior correlations between the parameters

Note that this object is reported silently so will be discarded unless the function is called with an object as in the example below.

Author(s)

Andrew Parnell <andrew.parnell@mu.ie>

See Also

See `simmr_mcmc` for creating objects suitable for this function, and many more examples. See also `simmr_load` for creating simmr objects, `plot.simmr_input` for creating isospace plots, `plot.simmr_output` for plotting output.

Examples

```
## Not run:
# A simple example with 10 observations, 2 tracers and 4 sources

# The data
mix = matrix(c(-10.13, -10.72, -11.39, -11.18, -10.81, -10.7, -10.54,
-10.48, -9.93, -9.37, 11.59, 11.01, 10.59, 10.97, 11.52, 11.89,
11.73, 10.89, 11.05, 12.3), ncol=2, nrow=10)
colnames(mix) = c('d13C','d15N')
s_names=c('Source A','Source B','Source C','Source D')
s_means = matrix(c(-14, -15.1, -11.03, -14.44, 3.06, 7.05, 13.72, 5.96), ncol=2, nrow=4)
s_sds = matrix(c(0.48, 0.38, 0.48, 0.43, 0.46, 0.39, 0.42, 0.48), ncol=2, nrow=4)
c_means = matrix(c(2.63, 1.59, 3.41, 3.04, 3.28, 2.34, 2.14, 2.36), ncol=2, nrow=4)
c_sds = matrix(c(0.41, 0.44, 0.34, 0.46, 0.46, 0.48, 0.46, 0.66), ncol=2, nrow=4)
conc = matrix(c(0.02, 0.1, 0.12, 0.04, 0.02, 0.1, 0.09, 0.05), ncol=2, nrow=4)

# Load into simmr
simmr_1 = simmr_load(mixtures=mix,
                     source_names=s_names,
                     source_means=s_means,
```

```
                      source_sds=s_sds,
                      correction_means=c_means,
                      correction_sds=c_sds,
                      concentration_means = conc)

# Plot
plot(simmr_1)


# MCMC run
simmr_1_out = simmr_mcmc(simmr_1)

# Summarise
summary(simmr_1_out) # This outputs all the summaries
summary(simmr_1_out,type='diagnostics') # Just the diagnostics
ans = summary(simmr_1_out,type=c('quantiles','statistics')) # Store the output in an object

## End(Not run)
```

`summary.simmr_output_tdf`

Summarises the output created with `simmr_mcmc_tdf`

Description

Produces textual summaries and convergence diagnostics for an object created with `simmr_mcmc_tdf`. The different options are: 'diagnostics' which produces Brooks-Gelman-Rubin diagnostics to assess MCMC convergence, 'quantiles' which produces credible intervals for the parameters, 'statistics' which produces means and standard deviations, and 'correlations' which produces correlations between the parameters.

Usage

```
## S3 method for class 'simmr_output_tdf'
summary(object, type = c("diagnostics",
  "quantiles", "statistics", "correlations"), ...)
```

Arguments

`object`	An object of class `simmr_output_df` produced by the function `simmr_mcmc_tdf`
`type`	The type of output required. At least none of 'diagnostics', 'quantiles', 'statistics', 'correlations'
`...`	Not used

Details

The quantile output allows easy calculation of 95 per cent credible intervals of the posterior dietary proportions. The Gelman diagnostic values should be close to 1 to ensure satisfactory convergence.

Multiple groups are not currently supported for estimating tdfs

Value

An list containing the following components:

`gelman`	The convergence diagnostics
`quantiles`	The quantiles of each parameter from the posterior distribution
`statistics`	The means and standard deviations of each parameter
`correlations`	The posterior correlations between the parameters

Note that this object is reported silently so will be discarded unless the function is called with an object as in the example below.

Author(s)

Andrew Parnell <andrew.parnell@mu.ie>

See Also

See `simmr_mcmc` for creating objects suitable for this function, and many more examples. See also `simmr_load` for creating simmr objects, `plot.simmr_input` for creating isospace plots, `plot.simmr_output` for plotting output.

Examples

```
## Not run:
# A simple example with 10 observations, 2 tracers and 4 sources

# The data
mix = matrix(c(-10.13, -10.72, -11.39, -11.18, -10.81, -10.7, -10.54,
-10.48, -9.93, -9.37, 11.59, 11.01, 10.59, 10.97, 11.52, 11.89,
11.73, 10.89, 11.05, 12.3), ncol=2, nrow=10)
colnames(mix) = c('d13C','d15N')
s_names=c('Source A','Source B','Source C','Source D')
s_means = matrix(c(-14, -15.1, -11.03, -14.44, 3.06, 7.05, 13.72, 5.96), ncol=2, nrow=4)
s_sds = matrix(c(0.48, 0.38, 0.48, 0.43, 0.46, 0.39, 0.42, 0.48), ncol=2, nrow=4)
c_means = matrix(c(2.63, 1.59, 3.41, 3.04, 3.28, 2.34, 2.14, 2.36), ncol=2, nrow=4)
c_sds = matrix(c(0.41, 0.44, 0.34, 0.46, 0.46, 0.48, 0.46, 0.66), ncol=2, nrow=4)
conc = matrix(c(0.02, 0.1, 0.12, 0.04, 0.02, 0.1, 0.09, 0.05), ncol=2, nrow=4)

# Load into simmr
simmr_1 = simmr_load(mixtures=mix,
                     source_names=s_names,
                     source_means=s_means,
                     source_sds=s_sds,
                     correction_means=c_means,
                     correction_sds=c_sds,
                     concentration_means = conc)

# Plot
plot(simmr_1)
```

```
# MCMC run
simmr_1_out = simmr_mcmc(simmr_1)

# Summarise
summary(simmr_1_out) # This outputs all the summaries
summary(simmr_1_out,type='diagnostics') # Just the diagnostics
ans = summary(simmr_1_out,type=c('quantiles','statistics')) # Store the output in an object

## End(Not run)
```

Index

∗Topic **multivariate**
simmr, 19

combine_sources, 2
compare_groups, 4
compare_sources, 7

plot.simmr_input, *2*, 9, *15*, *26*, *34*, *36*, *38*
plot.simmr_output, *2*, *10*, 14, *25, 26*, *36*, *38*
posterior_predictive, 16, *18*
print.simmr_input, 17
print.simmr_output, 17
prior_viz, 18

simmr, 19
simmr-package *(*simmr*)*, 19
simmr_elicit, *21*, 21, *25*
simmr_load, *2*, *4*, *8*, *10*, *15*, *19*, 23, *25, 26*, *33, 34*, *36*, *38*
simmr_mcmc, *2–5*, *8*, *10*, *14–16*, *18, 19*, *21*, *24, 25*, 25, *32–36*, *38*
simmr_mcmc_tdf, 32, *33*, *37*
summary.simmr_output, *15*, *25, 26*, 35
summary.simmr_output_tdf, *33, 34*, 37