Package 'tRophicPosition'

April 6, 2019

Type Package

Title Bayesian Trophic Position Calculation with Stable Isotopes

Version 0.7.7

Date 2019-04-05

Depends R (>= 3.4.0)

Imports coda, data.table, ggplot2, gridExtra, hdrcde, MCMCglmm, plyr, rjags, stats

Author Claudio Quezada-Romegialli, Andrew L Jackson, Chris Harrod

URL https://github.com/clquezada/tRophicPosition

BugReports https://groups.google.com/d/forum/trophicposition-support

Maintainer Claudio Quezada-Romegialli <clquezada@harrodlab.net>

Description Estimates the trophic position of a consumer relative to a baseline species. It implements a Bayesian approach which combines an interface to the 'JAGS' MCMC library of 'rjags' and stable isotopes. Users are encouraged to test the package and send bugs and/or errors to trophicposition-support@googlegroups.com.

License GPL (>= 2)

LazyData TRUE

RoxygenNote 6.1.1

Suggests dplyr, knitr, rmarkdown, testthat, covr

VignetteBuilder knitr

NeedsCompilation no

Repository CRAN

Date/Publication 2019-04-06 06:00:02 UTC

R topics documented:

`Bilagay` *Data frame containing stable isotope values of Bilagay.*

Description

A dataset containing stable isotope values (d13C and d15N) for the Bilagay *Cheilodactylus variegatus* (http://www.fishbase.se/summary/Cheilodactylus-variegatus.html), a fish common to the coastal kelp forests of N Chile.

Usage

```
data("Bilagay")
```

Format

A data frame with 841 rows and 7 variables:

Study factor, character describing which study funded data collection

Location factor, character describing where samples were taken

Spp factor, character describing which scientific name of species

FG factor, character describing functional group of species

d13C numeric, stable isotope d13C values

d15N numeric, stable isotope d15N values

NS numeric, integer describing north to south ordering (1-10)

---

compareTwoDistributions

Function to compare two distributions and test a hypothesis, in a Bayesian context

---

Description

Function to compare two distributions and test a hypothesis, in a Bayesian context

Usage

```
compareTwoDistributions(dist1 = NULL, dist2 = NULL, test = "<=",
  sample = NULL, round = 3, ...)
```

Arguments

dist1	A collection of numerical values (posterior distribution).
dist2	A collection of numerical values (posterior distribution).
test	A logical operator which states what to test for. Might be "<", "<=", ">" or ">=".
sample	If sample is numeric, it will take 'sample' elements of each of the distributions.
round	integer to indicate number of decimals kept.
...	extra arguments are passed to compareTwoDistributions().

Value

probability given sum(dist1 "test" dist2) / length(dist1)

Examples

```
a <- rnorm(100, 2, 0.1)
b <- rnorm(100, 1.8, 0.1)
compareTwoDistributions(a, b, test = ">=")
compareTwoDistributions(a, b, test = "bhatt")
```

`credibilityIntervals` *Plot credibility intervals and central tendency descriptor from posterior distributions of trophic position and/or alpha parameter*

Description

This function plots a data frame in ggplot2 format (variables in columns, observations in rows), likely returned by the functions `multiModelTP` and `multiSpeciesTP`. This is especially useful when there are several species or communities to compare, and a combined plot is preferred.

Usage

```
credibilityIntervals(df, x = "consumer", plotAlpha = TRUE,
  legend = NULL, legendAlpha = NULL, y1 = "mode", y1min = "lower",
  y1max = "upper", y1lim = NULL, y2 = "alpha.mode",
  y2min = "alpha.lower", y2max = "alpha.upper",
  xlab = "Bayesian models", ylab1 = "Posterior trophic position",
  ylab2 = "Posterior alpha", group_by = NULL,
  scale_colour_manual = NULL, labels = NULL, plot = TRUE, ...)
```

Arguments

`df`	data frame with at least 4 columns, a grouping variable, maximum, minimum and a central tendency descriptor (median, mode, etc.).
`x`	string defining the grouping variable.
`plotAlpha`	logical. If TRUE it expects that the data frame has at least 7 columns, another descriptor of central tendency, its maximum and minimum.
`legend`	list, position of the legend if not NULL, e.g. c(0.8, 0.8).
`legendAlpha`	list, position of the legend for the alpha plot, if not NULL, e.g. c(0.8, 0.8).
`y1`	string of the column with the central tendency descriptor of trophic position. By default, is the mode.
`y1min`	lower value plotted for trophic position. For the 95 credibility interval, this value would be 0.025 percentile.
`y1max`	higher value plotted for trophic position. For the 95 credibility interval, this value would be 0.975 percentile.
`y1lim`	vector of length 2, with limits of the y axis of trophic position.
`y2`	string of the column with the central tendency descriptor of alpha.
`y2min`	lower value plotted for alpha. For the 95 this value would be percentile 0.025.
`y2max`	higher value plotted for alpha. For the 95 interval, this value would be percentile 0.0975.
`xlab`	string of the label of the X axis.
`ylab1`	string of the label of Y1 axis (trophic position).
`ylab2`	string of the label of Y2 axis (alpha).

group_by	grouping variable (factor) in case of using colours.
scale_colour_manual	a list of colours (ggplot2 syntaxis) to use with group_by.
labels	string, manual labels for the x axis.
plot	logical, by default TRUE. In case of saving the output as a variable, the user can decide not to plot the output.
...	additional parameters passed to credibilityIntervals().

Value

a gtable (if alpha is plotted) with two ggplot2 objects or a ggplot2 object (if alpha is not plotted)

Examples

```
isotopeData <- generateTPData()
models <- multiModelTP(isotopeData, n.adapt = 200, n.iter = 200,
burnin = 200)
credibilityIntervals(models$gg, x = "model")
```

---

extractIsotopeData *Extract stable isotope data from a data frame*

Description

This function generates a list of isotopeData class objects parsing a data frame of stable isotope values analysed for one or more consumers and one or two baselines. The data frame can be organized in one or more communities (or sampling sites, samples in time, multiple studies, etc.).

Usage

```
extractIsotopeData(df = NULL, b1 = "Baseline 1", b2 = NULL,
  baselineColumn = "FG", consumersColumn = "Spp",
  groupsColumn = NULL, deltaC = NULL, deltaN = NULL, d13C = "d13C",
  d15N = "d15N", seed = 3, ...)
```

Arguments

df	data frame containing raw isotope data, with one or more grouping variables.
b1	string or vector with the text for baseline 1.
b2	string or vector with the text for baseline 2.
baselineColumn	string of the column where baselines are grouped.
consumersColumn	string of the column where consumers/species are grouped.
groupsColumn	string of the column where groups/communities are grouped.

deltaC	vector of values with trophic discrimination factor for carbon. If NULL it will use Post's assumptions (56 values with 3.4 mean +- 0.98 sd).
deltaN	vector of values with trophic discrimination factor for nitrogen. If NULL it will use Post's assumptions (107 values with 0.39 mean +- 1.3 sd).
d13C	string of the column that has d13C isotope values.
d15N	string of the column that has d15N isotope values.
seed	integer to get reproducible results. By default, seed = 3.
...	Additional arguments passed to this funcion.

Value

a list with isotopeData class objects

Examples

```
data("Bilagay")
head(Bilagay)
isotopeList <- extractIsotopeData(Bilagay, b1 = "Benthic_BL",
b2 = "Pelagic_BL", baselineColumn = "FG", consumersColumn = "Spp",
groupsColumn = "Location", d13C = "d13C", d15N = "d15N")
```

---

extractPredictiveData *Function to extract raw data from posterior predictive model-checking distributions*

Description

This function parses a data frame where it was stored the predicted data generated by the function TPmodel, and returns a list with all the predictive posterior data monitored.

Usage

```
extractPredictiveData(df = NULL, get = NULL, all = FALSE)
```

Arguments

df	data frame, variable where it was saved the output from a posterior predictive model-checking TPmodel run.
get	string, could be either "dNcPred" and "dCcPred" (for consumer data), "dNb1Pred" and "dCb1Pred" (for baseline 1 data), or "dNb2Pred" and "dNC2Pred" (for baseline 2 data).
all	logical, if TRUE it will return a combined list of all monitored data.

Value

a list of the data generated from a posterior predictive model-checking procedure, if all is TRUE, a vector is returned instead.

Examples

```
## Not run:
# Generate isotope data
isotopeData <- generateTPData(n.obsB = 45,
                              sd.dNb1 = 1, sd.dCb1 = 1,
                              dCb1 = -34, dCb2 = -24,
                              n.obsC = 60, dCc = -29)

# Define a one baseline model
model.string <- jagsOneBaseline()

# Initialize the model
model_d <- TPmodel(data = isotopeData,
                   model.string = model.string,
                   n.chains = 2,
                   n.adapt = 5000)

# Generate posterior samples of TP, alpha and dNcPred
# dNcPred stands for the predicted data dNc (nitrogen values of consumer)
samples_d <- posteriorTP(model_d,
                         variable.names = c("TP", "alpha", "dNcPred"),
                         burnin = 5000,
                         n.iter = 5000,
                         thin = 10)

# Extract posterior predictive data
dNcPred <- extractPredictiveData(samples_predicted, get = "dNcPred")

# Calculate residuals
dNcPred_res <- sweep(dNcPred, 2, isotopeData$dNc, "-")

# Combine all residuals
dNcPred_resall <- as.numeric(do.call(rbind, dNcPred_res))

# Plot a sample of them
plot(sample(dNcPred_resall, 10000), xlab = "Index",
     ylab = "Residuals")

# Extract posterior predictive data combined
dNcPred_all <- extractPredictiveData(samples_predicted, get = "dNcPred",
                                     all = TRUE)

# Plot a histogram of observed data and a density function of predicted data
# We need ggplot2 installed previously
ggplot2::ggplot() +
  ggplot2::geom_histogram(ggplot2::aes(x = a$dNc, y = ..density..),
                          binwidth = 0.3, fill = "grey", color = "black") +
  ggplot2::geom_density(ggplot2::aes(x = dNcPred_all), color = "blue") +
  ggplot2::xlab(expression(paste(delta^{15}, "N (\u2030)"))) +
  ggplot2::theme_bw()

## End(Not run)
```

Finnish_Lakes *Data frame of food webs in Inari and Kilpis Lakes (Finland)*

Description

Dataset of stable isotope values from two relatively large and deep oligotrophic Finnish lakes Inari and Kilpis.

Usage

```
data("Finnish_Lakes")
```

Format

A data frame with 7 variables:

Lake factor, with two levels, each representing one Lake

Species.group factor, with 20 levels, each representing one species

d13C numeric, representing delta 13 C isotope values

d15N numeric, representing delta 15 N isotope values

C numeric, amount of carbon

N numeric, amount of nitrogen

C.N numeric, C/N ratio

fromParallelTP *Function to extract raw data from parallel calculations of trophic position*

Description

This function parses a data frame where it was stored the data from parallel calculations using multiModelTP and a list of isotopeData class objects.

Usage

```
fromParallelTP(df = NULL, get = NULL)
```

Arguments

df	data frame, variable where it was saved the output from parallel calculations of TP
get	string, could be either "TP", "alpha" or "summary." In case of TP this function will extract the trophic position data, in case of alpha, this function will extract the alpha parameter data, and in case of summary, it will return a data frame ready to plot with the function credibilityIntervals.

Value

when selecting "TP" or "alpha", this function returns a posteriorTP or a posteriorAlpha object with the data. When selecting "summary", this function returns a data frame ready to be used with credibilityIntervals().

Examples

```
## Not run:
data("Bilagay")
BilagayList <- extractIsotopeData(Bilagay,
communityColumn = "Location",speciesColumn = "FG",
b1 = "Pelagic_BL", b2 = "Benthic_BL",
baselineColumn = "FG",
deltaC = TDF(author = "McCutchan", element = "C", type = "muscle"),
deltaN = TDF(author = "McCutchan", element = "N", type = "muscle"))
Bilagay_TPmodels <- parallel::parLapply(cluster,
BilagayList, multiModelTP, adapt = 20000,
n.iter = 20000, burnin = 20000, n.chains = 5,
model = "twoBaselinesFull")
ggplot_df <- fromParallelTP(Bilagay_TPmodels, get = "summary")
credibilityIntervals(ggplot_df, x = "community",
xlab = "Location along N-S gradient")

## End(Not run)
```

`generateTPData` *A function to generate random stable isotope data for trophic position calculation*

Description

This function generates random stable isotope (d13C and d15N) data to use basic functions and calculations coded within the package.

Usage

```
generateTPData(n.baselines = 2, n.obsB = 25, dNb1 = NULL,
  sd.dNb1 = 1, dCb1 = NULL, sd.dCb1 = 1, dNb2 = NULL,
  sd.dNb2 = 1, dCb2 = NULL, sd.dCb2 = 1, n.obsC = 25,
  consumer = NULL, dNc = NULL, sd.dNc = 1, dCc = NULL,
  sd.dCc = 1, DeltaN = 3.4, sd.DeltaN = 0.98, n.obsDeltaN = 56,
  DeltaC = 0.39, sd.DeltaC = 1.3, n.obsDeltaC = 107, seed = 3)
```

Arguments

`n.baselines`	number of baselines (could be 1 or 2), default is 2.
`n.obsB`	number of observations for baselines. Default is 25.

dNb1	mean value for d15N of baseline 1. Default is a random number between -5 and 5.
sd.dNb1	standard deviation for d15N of baseline 1.
dCb1	mean value for d13C of baseline 1.
sd.dCb1	standard deviation for d13C of baseline 1.
dNb2	mean value for d15N of baseline 2.
sd.dNb2	standard deviation for d15N of baseline 2.
dCb2	mean value for d13C of baseline 2.
sd.dCb2	standard deviation for d13C of baseline 2.
n.obsC	number of observations for consumer. Default is 25.
consumer	string for consumer.
dNc	mean value for d15N of consumer. Default value is dNb1 multiplied 2 times the trophic discrimination factor.
sd.dNc	standard deviation for d15N of consumer.
dCc	mean value for d13C of consumer.
sd.dCc	standard deviation for d13C of consumer.
DeltaN	mean value for trophic discrimination factor of nitrogen. Default value is 3.4.
sd.DeltaN	standard deviation for trophic discrimination factor of nitrogen. Default value is 0.98.
n.obsDeltaN	number of observations of deltaN (trophic discrimination factor). Default value is 56.
DeltaC	mean value for trophic discrimination factor of carbon. Default value is 0.39.
sd.DeltaC	standard deviation for trophic discrimination factor for carbon. Default value is 1.3.
n.obsDeltaC	number of observations of DeltaC (trophic discrimination factor). Default is 107.
seed	numerical value to get reproducible results.

Value

An isotopeData class object (named list) with dNb1, dNc and deltaN randomly generated observations. If n.baselines = 2, then dCb1, dNb2, dCb2, dCc and deltaC are also returned.

Examples

```
## Good data
a <-generateTPData(dCb1 = -10, dNb1 = -10,
dCc = -4, dNc = 4,
dCb2 = 2, dNb2 = 0)
plot(a)

## Consumer more enriched in carbon
b <-generateTPData(dCb1 = -10, dCc = 0, dCb2 = -2)
```

```
plot(b)

## Consumer much more enriched
c <-generateTPData(dCb1 = -10, dCc = 3, dCb2 = -2)
plot(c)
```

`getPosteriorMode` *Function to get mode from a posterior distribution*

Description

This function is a wrapper of hdr, it returns one mode (if receives a vector), otherwise it returns a list of modes (if receives a list of vectors). If receives an mcmc object it returns the marginal parameter mode using Kernel density estimation (posterior.mode).

Usage

```
getPosteriorMode(df = NULL, round = 3)
```

Arguments

df	data frame, list or vector with posterior distribution(s).
round	numeric, number of decimals rounded.

Value

a vector or a list of modes

Examples

```
# List example
a <- list("First" = rnorm(100,1), "Second" = rnorm(100,2))
getPosteriorMode(a)

# vector example
getPosteriorMode(rnorm(100,5), round = 2)
```

jagsBayesianModel *Returns a JAGS-based Bayesian model to use within tRophicPosition.*

Description

This function returns a string with a Bayesian model to be used with trophic position calculations

Usage

```
jagsBayesianModel(model = NULL, ...)
```

Arguments

model	string. Can be "oneBaseline", "twoBaselines" or "twoBaselinesFull" at the moment.
...	additional arguments passed to jagsOneBaseline, jagsTwoBaselines or jagsTwoBaselinesFull.

Value

a jags model as a character string

Examples

```
# Example with priors for TP.
# One baseline Bayesian model with prior for trophic position of consumer
# defined as a normal distribution with mean 3 and sd 1
model.string <- jagsBayesianModel(model = "oneBaseline", TP = "dnorm(3,1)")

# Two baselines model with trophic level of baseline = 1
model.string <- jagsBayesianModel(model = "twoBaselines", lambda = 1)

# Two baselines full model with priors for alpha
model.string <- jagsBayesianModel(model = "twoBaselinesFull",
alpha = "dbeta(10,1)")
```

jagsOneBaseline *Defines a jags Bayesian model to fit a single baseline trophic position model*

Description

This function takes some parameters and returns a jags model object as a character string for passing to jags.model.

Usage

```
jagsOneBaseline(muB = NULL, sigmaB = NULL, muDeltaN = NULL,
  sigmaDeltaN = NULL, sigma = NULL, TP = NULL, lambda = NULL, ...)
```

Arguments

muB	a distribution defining prior for mean (mu) of baseline. By default is dnorm(0, 0.0001).
sigmaB	a distribution defining sigma (standard deviation) of baseline. By default is dunif(0, 100).
muDeltaN	a distribution defining prior for the mean (mu) of deltaN. deltaN stands for trophic discrimination factor of Nitrogen. By default is dnorm(0, 0.0001).
sigmaDeltaN	a distribution defining sigma (standard deviation) of deltaN. By default is dunif(0, 100).
sigma	a value defining sigma (standard deviation) of baseline. By default is dunif(0, 100).
TP	a distribution defining prior of trophic position. By default is dunif(lambda, 10), with lambda = 2 if no defined before.
lambda	an integer indicating the trophic level of the baseline. Default is 2.
...	additional arguments passed to jagsOneBaseline.

Details

The single baseline trophic position model is defined as:

$$dNc = dNb + deltaN * (TP - lambda)$$

where dNc are d15N values of consumer, dNb1 are d15N values of baseline, deltaN is the trophic discrimination factor for N, TP is trophic position of the consumer and lambda is the trophic level of baseline. Furthermore, as a Bayesian approach, dNb, deltaN and dNc are defined as random parameters with a normal distribution with mean mu_i and precision tau_i, TP is a random parameter with a uniform distribution and lambda is a constant. All these distributions can be changed modifying them as priors, while defining lambda within the call to the function.

Although it is possible to use a number of predefined or customized distributions (see distribution aliases in JAGS documentation), it is likely that most of the time you will be using a normal distribution as prior for most parameters. This is the default option (i.e. when the function is called without arguments). To change it, you need to indicate a mean and standard deviation for the parameter of interest, for example "dnorm(0, 0.0001)". Here, a prior of normally distributed mu is defined, with a mean 0, and a standard deviation of 0.0001. This constitutes a normally distributed prior, although uninformative. You might want to change the mean and/or the standard deviation according to your prior knowledge of the system/consumer you are working on. As well as the priors for mu, JAGS uses "tau", which is the precision for defining the standard deviation of mu. Precision is a deterministic function (instead of the distributional "~"), and it is calculated as "tau <- power(sigma, -2)", thus you could define as well sigma_i, which stands for the standard deviation of the parameter of interest.

Value

A jags model (BUGS-language) as a character string

`jagsTwoBaselines` *Defines a jags Bayesian model to fit a two baselines trophic position model (without fractionation for C)*

Description

Takes some parameters and returns a jags model object as a character string for passing to `jags.model`.

Usage

```
jagsTwoBaselines(sigmaNc = NULL, sigmaCc = NULL, muCb1 = NULL,
  sigmaCb1 = NULL, muNb1 = NULL, sigmaNb1 = NULL, muCb2 = NULL,
  sigmaCb2 = NULL, muNb2 = NULL, sigmaNb2 = NULL, lambda = NULL,
  TP = NULL, alpha = NULL, muDeltaN = NULL, sigmaDeltaN = NULL,
  ...)
```

Arguments

`sigmaNc`	a distribution defining sigma (standard deviation) for N of consumer. Default is dunif(0, 100).
`sigmaCc`	a distribution defining sigma (standard deviation) for C of consumer. Default is dunif(0, 100).
`muCb1`	a distribution defining prior for mean (mu) for C of baseline 1. Default is dnorm(0, 0.0001).
`sigmaCb1`	a distribution defining sigma (standard deviation) for C of baseline 1. Default is dunif(0, 100).
`muNb1`	a distribution defining prior for mean (mu) for N of baseline 1. dnorm(0, 0.0001)
`sigmaNb1`	a distribution defining sigma (standard deviation) for N of baseline 1. Default is dunif(0, 100).
`muCb2`	a distribution defining prior for mean (mu) for C of baseline 2. dnorm(0, 0.0001)
`sigmaCb2`	a distribution defining sigma (standard deviation) for C of baseline 2. Default is dunif(0, 100).
`muNb2`	a distribution defining prior for mean (mu) for N of baseline 2. dnorm(0, 0.0001)
`sigmaNb2`	a distribution defining sigma (standard deviation) for N of baseline 2. Default is dunif(0, 100).
`lambda`	an integer indicating the trophic position of the baseline. Default is 2.
`TP`	a distribution defining prior of trophic position. Default is dunif(lambda, 10), with lambda defined above.
`alpha`	a distribution defining alpha (mixing model between 2 sources). Default is dbeta(1,1).

muDeltaN	a distribution defining prior for the mean (mu) of deltaN, which stands for trophic discrimination factor for Nitrogen. Default is dnorm(0, 0.0001).
sigmaDeltaN	a value defining sigma (standard deviation) for the mean (mu) of deltaN. Default is dunif(0, 100).
...	additional arguments passed to this function.

Details

The two baselines trophic position model is defined as:

$$dNc\ dnorm(deltaN * (TP - lambda) + dNb1 * alpha + dNb2 * (1 - alpha), tauNc)$$

and

$$dCc\ dnorm(alpha * (dCb1 - dCb2) + dCb2, tauCc)$$

where dNc and dCc are d15N and d13C values of consumer, dNb1 and dCb1 are d15N and d13C values of baseline 1, dNb2 and dCb2 are d15N and d13C values of baseline 2, alpha is the relative proportion of N derived from baseline 1, deltaN is the trophic discrimination factor for N, TP is trophic position of the consumer and lambda is the trophic level of baselines.

In this Bayesian model, both dNc and dCc are modelled as having a normal distribution with means calculated with above equations and precision (tau) calculated as standard deviation ^-2. Furthermore, dNb1, dCb1, dNb2, dCb2 and deltaN are defined as random parameters with a normal distribution with mean mu_i and precision tau_i, TP is a random parameter with a uniform distribution, alpha is a random parameter with a beta distribution and lambda is a constant. All these distributions can be changed modifying them as priors, while defining lambda within the call to the function.

You might want to change the mean, standard deviation or other parameters of the distributions according to your prior knowledge of the system/consumer you are working on. Although it is possible to use a number of predefined or customized distributions (see distribution aliases in JAGS documentation), it is likely that most of the time you will be using a normal distribution as prior for most parameters. This is the default option (i.e. when the function is called without arguments). To change it, you need to indicate a mean and standard deviation for the i-est parameter of interest, for example "dnorm(0, 0.0001)". Here, a prior of normally distributed mu_i is defined, with a mean 0, and a standard deviation of 0.0001. This constitutes an uninformative and normally distributed prior, for the mean of the i-est parameter. As well as the priors for mu_i, JAGS uses "tau", which is the precision for defining the standard deviation of mu_i. Precision is a deterministic function (instead of the distributional "~"), and it is calculated as "tau_i <- power(sigma_i, -2)", thus you could define as well sigma_i, which stands for the standard deviation of the i-est parameter of interest. In the case of alpha, the default is a beta distribution with parameters a = 1 and b = 1.

Value

A jags model as a character string

`jagsTwoBaselinesFull` *Defines a jags Bayesian model to fit a two baselines trophic position full model (with fractionation for C)*

Description

Takes some parameters and returns a jags model object as a character string for passing to `jags.model`.

Usage

```
jagsTwoBaselinesFull(sigmaNc = NULL, sigmaCc = NULL, muCb1 = NULL,
  sigmaCb1 = NULL, muNb1 = NULL, sigmaNb1 = NULL, muCb2 = NULL,
  sigmaCb2 = NULL, muNb2 = NULL, sigmaNb2 = NULL, lambda = NULL,
  TP = NULL, alpha = NULL, muDeltaN = NULL, sigmaDeltaN = NULL,
  muDeltaC = NULL, sigmaDeltaC = NULL, ...)
```

Arguments

`sigmaNc`	a distribution defining sigma (standard deviation) for N of consumer. Default is dunif(0, 100).
`sigmaCc`	a distribution defining sigma (standard deviation) for C of consumer. Default is dunif(0, 100).
`muCb1`	a distribution defining prior for mean (mu) for C of baseline 1. Default is dnorm(0, 0.0001).
`sigmaCb1`	a distribution defining sigma (standard deviation) for C of baseline 1. Default is dunif(0, 100).
`muNb1`	a distribution defining prior for mean (mu) for N of baseline 1. dnorm(0, 0.0001)
`sigmaNb1`	a distribution defining sigma (standard deviation) for N of baseline 1. Default is dunif(0, 100).
`muCb2`	a distribution defining prior for mean (mu) for C of baseline 2. dnorm(0, 0.0001)
`sigmaCb2`	a distribution defining sigma (standard deviation) for C of baseline 2. Default is dunif(0, 100).
`muNb2`	a distribution defining prior for mean (mu) for N of baseline 2. dnorm(0, 0.0001)
`sigmaNb2`	a distribution defining sigma (standard deviation) for N of baseline 2. Default is dunif(0, 100).
`lambda`	an integer indicating the trophic position of the baseline. Default is 2.
`TP`	a distribution defining prior of trophic position. Default is dunif(lambda, 10), with lambda defined above.
`alpha`	a distribution defining alpha (mixing model between 2 sources). Default is dbeta(1,1).
`muDeltaN`	a distribution defining prior for the mean (mu) of deltaN, which stands for trophic discrimination factor of Nitrogen. Default is dnorm(0, 0.0001).

`sigmaDeltaN`	a value defining sigma (standard deviation) for the mean (mu) of deltaN. Default is dunif(0, 100).
`muDeltaC`	a distribution defining prior for the mean (mu) of deltaC, which stands for trophic discrimination factor of Carbon
`sigmaDeltaC`	a value defining sigma (standard deviation) for the mean (mu) of deltaC.
`...`	additional arguments passed to this function.

Details

The two baselines trophic position full model is defined as:

$$dNc\ dnorm(deltaN * (TP - lambda) + dNb1 * alpha + dNb2 * (1 - alpha), tauNc)$$

and

$$dCc\ dnorm(dCb2 + (deltaC * (TP - lambda)) + (alpha * (dCb1 - dCb2)), tauCc))$$

where dNc and dCc are d15N and d13C values of consumer, dNb1 and dCb1 are d15N and d13C values of baseline 1, dNb2 and dCb2 are d15N and d13C values of baseline 2, alpha is the relative proportion of N derived from baseline 1, deltaN is the trophic discrimination factor for N, deltaC is the trophic discrimination factor for C, TP is trophic position of the consumer and lambda is the trophic level of baselines.

In this Bayesian model, both dNc and dCc are modelled as having a normal distribution with means calculated with above equations and precision (tauNc and tauCc) calculated as standard deviation ^-2. Furthermore, dNb1, dCb1, dNb2, dCb2, deltaN and deltaC are defined as random parameters with a normal distribution with mean mu_i and precision tau_i, TP is a random parameter with a uniform distribution, alpha is a random parameter with a beta distribution and lambda is a constant. All these distributions can be changed modifying them as priors, while defining lambda within the call to the function.

You might want to change the mean, standard deviation or other parameters of the distributions according to your prior knowledge of the system/consumer you are working on. Although it is possible to use a number of predefined or customized distributions (see distribution aliases in JAGS documentation), it is likely that most of the time you will be using a normal distribution as prior for most parameters. This is the default option (i.e. when the function is called without arguments). To change it, you need to indicate a mean and standard deviation for the i-est parameter of interest, for example "dnorm(0, 0.0001)". Here, a prior of normally distributed mu_i is defined, with a mean 0, and a standard deviation of 0.0001. This constitutes an uninformative and normally distributed prior, for the mean of the i-est parameter. As well as the priors for mu_i, JAGS uses "tau", which is the precision for defining the standard deviation of mu_i. Precision is a deterministic function (instead of the distributional "~"), and it is calculated as "tau_i <- power(sigma_i, -2)", thus you could define as well sigma_i, which stands for the standard deviation of the i-est parameter of interest. In the case of alpha, the default is a beta distribution with parameters a = 1 and b = 1.

Value

A jags model as a character string

`loadIsotopeData` *Extract and load stable isotope data for selected consumers from a data frame*

Description

This function extracts only selected consumers/species with their respective baseline(s) and returns an isotopeData class object (or list). It is useful when there are a lot of information in a data frame and you want to calculate trophic position only for selected consumers in one or more communities.

Usage

```
loadIsotopeData(df = NULL, consumer = NULL, group = NULL,
  b1 = "Baseline 1", b2 = NULL, baselineColumn = "FG",
  consumersColumn = "FG", groupsColumn = NULL, d13C = "d13C",
  d15N = "d15N", deltaC = NULL, deltaN = NULL, seed = 666, ...)
```

Arguments

`df`	data frame containing raw isotope data with at least one grouping column.
`consumer`	string or character vector indicating which consumer/species will be extracted.
`group`	string or character vector indicating which group(s) will be extracted.
`b1`	string or character vector indicating which baseline(s) will be extracted as baseline 1.
`b2`	string or character vector indicating which baseline(s) will be extracted as baseline 2.
`baselineColumn`	string of the column where baselines are grouped.
`consumersColumn`	string of the column where species/consumer(s) are grouped.
`groupsColumn`	string of the column where groups/communities are grouped.
`d13C`	string indicating from which column extract d13C isotope values.
`d15N`	string indicating from which column extract d15N isotope values.
`deltaC`	vector of values with trophic discrimination factor for carbon. If NULL it will use Post's assumptions (56 values with 3.4 mean +- 0.98 sd).
`deltaN`	vector of values with trophic discrimination factor for nitrogen. If NULL it will use Post's assumptions (107 values with 0.39 mean +- 1.3 sd).
`seed`	numerical value to get reproducible results with trophic discrimination factors (because they are simulated each time this function is called). By default, is 3.
`...`	Additional arguments passed to this function.

Value

an isotopeData class object if one consumer and one group are selected. A list of isotopeData class objects if more than one consumer or more than one group are selected.

Examples

```
data("Bilagay")
head(Bilagay)
loadIsotopeData(df = Bilagay, consumer = "Bilagay", consumersColumn = "FG",
group = c("CHI", "COL"), groupsColumn = "Location",
b1 = "Benthic_BL", b2 = "Pelagic_BL", baselineColumn = "FG")
```

`multiModelTP` *Multiple model calculation of trophic position*

Description

This function takes an isotopeData class object and calculates by default three Bayesian models: one and two baselines without carbon fractionation and two baselines with carbon fractionation.

Usage

```
multiModelTP(siData = siData, lambda = 2, n.chains = 2,
  n.adapt = 20000, n.iter = 20000, burnin = 20000, thin = 10,
  models = c("oneBaseline", "twoBaselines", "twoBaselinesFull"),
  print = FALSE, quiet = FALSE, ...)
```

Arguments

`siData`	an isotopeData class object.
`lambda`	numerical value, represents the trophic level of baseline(s).
`n.chains`	number of parallel chains for the model. If convergence diagnostics (such as Gelman-Rubin) are printed, n.chains needs to be >= 2.
`n.adapt`	number of adaptive iterations, before the actual sampling.
`n.iter`	number of iterations for Bayesian modelling (posterior sampling).
`burnin`	number of iterations discarded as burn in.
`thin`	thinning. Number of samples discarded while performing posterior sampling.
`models`	string or list representing Bayesian models. At the moment they can be "oneBaseline", "twoBaselines" and/or "twoBaselinesFull".
`print`	logical value to indicate whether Gelman and Rubin's convergence diagnostic and summary of samples are printed.
`quiet`	logical value to indicate whether messages generated during compilation will be suppressed, as well as the progress bar during adaptation.
`...`	additional arguments passed to this function.

Value

For each model calculated, returns a data frame of 4 elements with raw posterior samples, a list with posterior TP samples, a list with posterior muDeltaN (if one baseline model was chosen) or alpha (if a two baselines model was chosen) and a data frame with a summary of posterior samples named gg.

Examples

```
## Not run:
isotopeData <- generateTPData()
models <- multiModelTP(isotopeData, n.adapt = 500, n.iter = 500,
burnin = 500)
credibilityIntervals(models$gg, x = "model")

## End(Not run)
```

`multiSpeciesTP` *Multiple species calculation of trophic position*

Description

This function takes a named list of isotopeData class objects and calculates one or more Bayesian models of trophic position for each element of the list.

Usage

```
multiSpeciesTP(siDataList = siDataList, lambda = 2, n.chains = 2,
  n.adapt = 20000, n.iter = 20000, burnin = 20000, thin = 10,
  model = "oneBaseline", print = FALSE, quiet = FALSE, ...)
```

Arguments

`siDataList`	a named list of isotopeData class objects.
`lambda`	numerical value, represents the trophic level for baseline(s).
`n.chains`	number of parallel chains for the model. If convergence diagnostics (such as Gelman-Rubin) are printed, n.chains needs to be >= 2.
`n.adapt`	number of adaptive iterations, before the actual sampling.
`n.iter`	number of iterations for Bayesian modelling (posterior sampling).
`burnin`	number of iterations discarded as burn in.
`thin`	thinning. Number of samples discarded while performing posterior sampling.
`model`	string or list representing Bayesian models. At the moment they can be "oneBaseline", "twoBaselines" and/or "twoBaselinesFull".
`print`	logical value to indicate whether Gelman and Rubin's convergence diagnostic and summary of samples are printed.
`quiet`	logical value to indicate whether messages generated during compilation will be suppressed, as well as the progress bar during adaptation.
`...`	additional arguments passed to this function.

Value

A list of 4 elements. The output is organised as lists nested. The first element (multiSpeciesTP) has the gg data frame returned by multiModelTP, the second element (df) is a data frame with summary information for all consumers and models, the third element (TPs) has the raw posterior trophic position for all consumers and models, and the last element (Alphas) has raw posterior of muDeltaN (if one baseline model was chosen) or alpha (if a two baselines model was chosen) for all consumers and models.

Examples

```
siDataList <- list("consumer1" = generateTPData(consumer = "consumer1"),
"consumer2" = generateTPData(consumer = "consumer2"))
models <- multiSpeciesTP(siDataList, model = "twoBaselines", n.adapt = 500,
n.iter = 500, burnin = 500)
credibilityIntervals(models$df, x = "consumer")
```

Orestias *Named list containing stable isotope values of Orestias chungarensis*

Description

A dataset containing stable isotope values (d13C and d15N) for the killifish *Orestias chungarensis* (http://www.fishbase.se/summary/Orestias-chungarensis.html), an endemic fish found in the Chungara Lake at 4,500 meters above sea level, Chilean Altiplano. Harrod, C. & I. Vila unpublished results.

Usage

```
data("Orestias")
```

Format

A named list with 8 elements:

dCc numeric, stable isotope values of d13C for Orestias chungarensis

dNc numeric, stable isotope values of d15N for Orestias chungarensis

dCb1 numeric, stable isotope values of d13C for baseline 1

dNb1 numeric, stable isotope values of d15N for baseline 1

dCb2 numeric, stable isotope values of d13C for baseline 2

dNb2 numeric, stable isotope values of d15N for baseline 2

deltaN numeric, trophic discrimination factor for nitrogen

deltaC numeric, trophic discrimination factor for carbon

`pairwiseComparisons` *Function to perform pairwise comparisons between two or more posterior distributions*

Description

Function to compare two or more posterior distributions and test a hypothesis, in a Bayesian context

Usage

```
pairwiseComparisons(df, test = "<=", print = FALSE)
```

Arguments

`df`	data frame with a collection of numerical values (posterior samples) to be compared.
`test`	string with the logical test to be used in comparisons. Can be <, <=, > or >=.
`print`	logical value to indicate whether the output should be printed or not.

Value

a symmetrical matrix with probabilities given sum(dist1 >= dist2) / length(dist1) for each comparison.

Examples

```
a <- rnorm(100, 2, 0.1)
b <- rnorm(100, 1.8, 0.1)
c <- rnorm(100, 2.2, 0.1)
pairwiseComparisons(list("a" = a, "b" = b, "c" = c))
```

`parametricTP` *Parametric trophic position*

Description

Calculation of parametric trophic position (with means) partially based on Post (2002: Using Stable Isotopes to Estimate Trophic Position: Models, Methods, and Assumptions. Ecology 83, 703).

Usage

```
parametricTP(siData, lambda = 2, print = TRUE)
```

Arguments

siData	an isotopeData class object.
lambda	numerical value representing trophic level of baseline(s).
print	a logical value to indicate whether the output is printed or not.

Details

In case of the one baseline model, trophic position is calculated as

$$TP = lambda + ((dNc - dNb1)/deltaN)$$

where lambda is trophic level of baseline 1, dNc are d15N values of consumer, dNb1 are d15N values of baseline 1 and deltaN are trophic discrimination factor values of N.

In case of the two baselines model, trophic position is calculated as

$$TP = lambda + ((dNc - ((dNb1 * alpha) + (dNb2 * (1 - alpha))))/deltaN)$$

and

$$alpha = (dCc - dCb2)/(dCb1 - dCb2)$$

Additional variables are dCc (d13C values of consumer), dNb2 (d15N values of baseline 2), alpha (relative contribution of N from baseline 1), and dCb1 and dCb2 (d13C values of baselines 1 and 2 respectively).

In case of the two baselines full model, trophic position is calculated with the same equation as the two baselines model, but alpha is calculated as

$$alpha = ((dCc - (deltaC * TP/lambda)) - dCb2)/(dCb1 - dCb2)$$

and includes deltaC (trophic discrimination factor for C).

In all cases trophic position is calculated based on means of isotope values and trophic discrimination factors. For the two baselines full model, an iteration is needed to get convergence of trophic position, starting with alpha calculated with the two baselines simple model. If no convergence is gotten after 50 iterations a message is plotted and both alpha and trophic position are printed.

Value

a list with parametric trophic position calculated with a one baseline model, a two baselines model and its alpha value, and a two baselines full model and its alpha value.

Examples

```
consumer <- generateTPData()
parametricTP(consumer)
```

`plot.isotopeData` *Plot stable isotope data (2 elements) with one or two baselines*

Description

Plot stable isotope data (2 elements) with one or two baselines

Usage

```
## S3 method for class 'isotopeData'
plot(x, consumer = NULL, b1 = NULL, b2 = NULL,
  legend = c(1.15, 1.15), density = "both", ...)
```

Arguments

`x`	an isotopeData class object.
`consumer`	string representing the consumer.
`b1`	string representing baseline 1.
`b2`	string representing baseline 2.
`legend`	coordinates representing where to locate the legend.
`density`	string representing whether the density function is plotted. Accepted characters are "both" in which case this function will plot the density function above and to the right, "right", "above" or "none".
`...`	additional arguments passed to this function.

Value

a ggplot2 object with the biplot of isotopes.

Examples

```
a <- generateTPData()
plot(a)
```

plotMCMC *Internal function that plot a mcmc.list object.*

Description

Not intended to be used by the user.

Usage

```
plotMCMC(x, trace = TRUE, density = TRUE, smooth = TRUE, bwf,
  auto.layout = TRUE, ask = graphics::par("ask"), ...)
```

Arguments

x	null
trace	null
density	null
smooth	null
bwf	null
auto.layout	null
ask	null
...	null

plotTP *Function to plot a trophic position distribution*

Description

Wrapper of siberDensityPlot.

Usage

```
plotTP(TPdist = NULL, ...)
```

Arguments

TPdist	vector. One posterior distribution (or a collection) of trophic position. In case of wanting to plot two or more posterior distributions, needs to be passed as a data.frame object.
...	additional arguments passed to this function.

Value

A new figure window

Examples

```
species1 <- stats::rnorm(1000, 4, 0.1)
species2 <- stats::rnorm(1000, 3, 0.8)
plotTP(data.frame(species1, species2))
```

`posteriorTP` *Function to generate posterior samples of a trophic position JAGS model*

Description

This is a wrapper of `coda.samples` which in turn, is a wrapper of `jags.samples`. It extracts random samples from the posterior distribution of the parameters of a jags model.

Usage

```
posteriorTP(model, variable.names = c("TP", "muDeltaN"),
  n.iter = 10000, burnin = NULL, thin = 10, quiet = FALSE, ...)
```

Arguments

`model`	a JAGS model object returned by any of functions `jagsOneBaseline`, `jagsTwoBaselines`, `jagsTwoBaselinesFull` or `jagsBayesianModel`
`variable.names`	vector of characters giving the names of variables to be monitored.
`n.iter`	integer defining the number of iterations. By default is 10000
`burnin`	number of iterations discarded as burn in.
`thin`	thinning interval to get posterior samples.
`quiet`	logical value to indicate whether messages generated during posterior sampling will be suppressed, as well as the progress bar.
`...`	additional arguments passed to `coda.samples`.

Value

mcmc.list object containing posterior samples of the Bayesian model.

Examples

```
## Not run:
isotopeData <- generateTPData()
model.string <- jagsBayesianModel()
model <- TPmodel(data = isotopeData, model.string = model.string,
n.adapt = 500)
posterior.samples <- posteriorTP(model, n.iter = 500)

## End(Not run)
```

Roach — *Data frame of Roach in Lough Neagh*

Description

The roach is a cyprinid freshwater-brackish benthopelagic fish, common to most of Europe and western Asia [http://www.fishbase.org/summary/Rutilus-rutilus.html] (http://www.fishbase.org/summary/Rutilus-rutilus.html). Larvae and juveniles are typically pelagic, consuming zooplankton, with a switch to more benthic diets as they grow, including plant material and detritus. The dataset included here examines if a consumer shows an ontogenetic shift in their trophic position, studying how TP varies across different size classes.

Usage

```
data("Roach")
```

Format

A data frame with 6 variables:

Taxon factor, with 5 levels, the common name of each baseline species and Roach

FG factor, with 3 levels, each representing three functional groups: Benthic_BL (bith, theodoxus and valvata), Pelagic_BL (zebra mussel) and Roach (consumer)

Fork.length numeric, fork length of roach in mm

Size.class numeric, each representing deciles of fork length of roach

d13C numeric, stable isotope values of d13C

d15N numeric, stable isotope values of d15N

screenFoodWeb — *Function that creates a biplot of a food web with stable isotope values (d13C and d15N)*

Description

Function that creates a biplot of a food web with stable isotope values (d13C and d15N)

Usage

```
screenFoodWeb(df = NULL, grouping = c("Species", "FG"),
  printSummary = FALSE, ...)
```

Arguments

df	a data frame that contains the isotope values. By defaults, the data frame needs to have the following columns: d13C, d15N, Species and FG. Species stands for the scientific name (or common name), and FG stands for the functional group for each species.
grouping	a vector with the name of the columns (variables) that will be used to summarize, and plot the data frame.
printSummary	a logical value to indicate whether the summary is printed
...	optional arguments that are passed to the function for later use.

Value

a ggplot2 object with the biplot of the data frame. Also prints the summary of the data frame as needed.

Examples

```
data("Bilagay")
subset_CHI <- Bilagay[Bilagay[,"Location"] %in% "CHI",]
screenFoodWeb(subset_CHI, grouping = c("Spp", "FG"))
```

---

screenIsotopeData *Function to plot and screen stable isotope data with one or more baselines.*

Description

This function receives a named list of vectors (isotopeData class object), and plots a biplot with 2 sources and a consumer. The user can states whether he/she wants a density function plotted above, to the right, at both sides or does not want it to be plotted.

Usage

```
screenIsotopeData(isotopeData = NULL, density = "both",
  consumer = "Consumer", b1 = "Pelagic baseline",
  b2 = "Benthic baseline", legend = c(1.15, 1.15), title = NULL, ...)
```

Arguments

isotopeData	an isotopeData class object.
density	string representing whether the density function is plotted. Accepted characters are "both" in which case will plot the density function above and to the right, "right", "above" or "none".
consumer	string representing the consumer.
b1	string representing baseline 1.

b2	string representing baseline 2.
legend	coordinates representing where to locate the legend.
title	string representing title.
...	additional arguments passed to this function.

Value

none

Examples

```
a <- generateTPData()
screenIsotopeData(a)
```

`simulateTDF` *Simulate trophic discrimination factors*

Description

This function returns trophic discrimination factors (TDF), given a number of observations, a mean and/or a standard deviation for deltaN and/or deltaC.

Usage

```
simulateTDF(nN = 56, meanN = NULL, sdN = 0.98, nC = 107,
  meanC = NULL, sdC = 1.3, seed = 3)
```

Arguments

nN	number of observations for deltaN.
meanN	mean for deltaN.
sdN	standard deviation for deltaN.
nC	number of observations for deltaC.
meanC	mean for deltaC.
sdC	standard deviation for deltaC.
seed	numerical value to indicate reproducible results.

Value

a named list with TDF values for nitrogen and/or carbon

Examples

```
# 25 values of TDF for nitrogen, mean 3, sd, 1
simulateTDF(nN = 25, meanN = 3, sdN = 1)

# 18 values of TDF for carbon, mean 0.6, sd, 0.7
simulateTDF(nC = 18, meanC = 0.6, sdC = 0.7)
```

`simulateTEF` *Simulate trophic enrichment factors*

Description

This function returns trophic enrichment factors (TEF), given a number of observations, a mean and a standard deviation for deltaN and/or deltaC. This function has been replaced by simulateTDF, following the convention of naming trophic discrimination factors (TDF) instead of trophic enrichment factors (TEF).

Usage

```
simulateTEF(nN = 56, meanN = NULL, sdN = 0.98, nC = 107,
  meanC = NULL, sdC = 1.3)
```

Arguments

`nN`	number of observations for deltaN.
`meanN`	mean for deltaN.
`sdN`	standard deviation for deltaN.
`nC`	number of observations for deltaC.
`meanC`	mean for deltaC.
`sdC`	standard deviation for deltaC.

Value

a named list with TEF values for nitrogen and/or carbon

Examples

```
#simulateTEF() is deprecated, use simulateTDF() instead:

# 25 values of TEF for nitrogen, mean 3, sd, 1
simulateTDF(nN = 25, meanN = 3, sdN = 1)

# 18 values of TEF for carbon, mean 0.6, sd, 0.7
simulateTDF(nC = 18, meanC = 0.6, sdC = 0.7)
```

summariseIsotopeData *Function that summarises a data frame containing stable isotope values (d13C and d15N) grouping by Species and FG columns*

Description

A wrapper of plyr:ddply to summarise a data frame.

Usage

```
summariseIsotopeData(df = NULL, grouping = c("Species", "FG"),
  printSummary = FALSE, ...)
```

Arguments

df	a data frame that contains the isotope values. It needs to have the following columns: d13C, d15N, Species and FG. Species stands for the scientific name (or common name), and FG stands for the functional group for each species. If the data frame does not have Species and FG columns, it will raise an error. If the columns change their names, they need to be stated as well in the grouping variable.
grouping	a vector with the name of the columns (variables) that will be used to summarize, and plot the data frame.
printSummary	logical value indicating whether the summary is printed.
...	optional arguments that are passed to the function for later use.

Value

a data frame with the summary of the data frame.

Examples

```
data("Bilagay")
subset_CHI <- Bilagay[Bilagay[,"Location"] %in% "CHI",]
summariseIsotopeData(subset_CHI, grouping = c("Spp", "FG"))
```

summary.isotopeData *Summary for stable isotope data*

Description

Summary for stable isotope data

Usage

```
## S3 method for class 'isotopeData'
summary(object, print = TRUE, round_dec = 1, ...)
```

Arguments

object	an isotopeData class object.
print	a logical value to indicate whether the summary is printed.
round_dec	number of decimals kept.
...	additional arguments passed to this function.

Value

a list with number of observations, mean, standard deviation, standard error, minimum, maximum and median for each element of an isotopeData class object.

Examples

```
a <- generateTPData()
summary(a)
```

TDF *Trophic discrimination factors from bibliography*

Description

This function returns trophic discrimination factors (TDF), given an author, element and a type. For convenience 'type' includes a number of categories depending on the author. At the moment it includes TDF data from Post (2002) and from McCutchan et al (2003).

Usage

```
TDF(author = "Post", element = "both", type = NULL, seed = 3)
```

Arguments

author	could be either "Post" or "McCutchan" at the moment.
element	can be "both", "N" or "C"
type	this argument only works for "McCutchan" author (their Table 3). "all" returns all TDF data; "whole" and "muscle" returns TDF separated per type analysis; "acidified" and "unacidified" returns TDF separated per acidification; and "Rainbow Trout" and "Brook Trout" returns TDF separated per fish species (according to their Table 1).
seed	integer to have replicated results

Value

a list (if element = "both") or a vector (if element ="N" or element = "C") containing TDF values

Examples

```
TDF(author = "McCutchan", element = "N")
```

TEF *Trophic enrichment factors from bibliography*

Description

This function returns trophic enrichment factors (TEF), given an author, element and a type. For convenience 'type' includes a number of categories depending on the author. At the moment it includes TEF data from Post (2002) and from McCutchan et al (2003). This function is maintained for compatibility backwards of version 0.6.8.

Usage

```
TEF(author = "Post", element = "both", type = "all", seed = 3)
```

Arguments

author	could be either "Post" or "McCutchan" at the moment.
element	can be "both", "N" or "C"
type	this argument only works for "McCutchan" author (their Table 3). "all" returns all TEF data; "whole" and "muscle" returns TEF separated per type analysis; "acidified" and "unacidified" returns TEF separated per acidification; and "Rainbow Trout" and "Brook Trout" returns TEF separated per fish species (according to their Table 1).
seed	integer to have replicated results

Value

a list (if element = "both") or a vector (if element ="N" or element = "C") containing TDF values

Examples

```
# TEF() is deprecated, use TDF() instead:
TDF(author = "McCutchan", element = "N")
```

TPmodel *Function to create a JAGS-based Bayesian model to calculate trophic position*

Description

This function is a wrapper of jags.model. It receives an isotopeData class object containing the data, a model string returned by either jagsOneBaseline, jagsTwoBaselines, jagsTwoBaselinesFull or jagsBayesianModel, and creates a JAGS model object.

Usage

```
TPmodel(data = NULL, model.string = NULL, n.chains = 2,
  n.adapt = 10000, quiet = FALSE, ...)
```

Arguments

data	a list containing the data.
model.string	model string containing a description of the model.
n.chains	number of parallel chains for the model.
n.adapt	number of iterations for adaptation (initial sampling phase)
quiet	logical value to indicate whether messages generated during compilation will be suppressed, as well as the progress bar during adaptation.
...	additional arguments passed to jags.model.

Value

TPmodel returns an object inheriting from class jags which can be used to generate dependent samples from the posterior distribution of the parameters

Examples

```
## Not run:
isotopeData <- generateTPData()
model.string <- jagsBayesianModel()
model <- TPmodel(data = isotopeData, model.string = model.string,
n.adapt = 500)

## End(Not run)
```

trophicDensityPlot *Function to plot posterior samples of trophic position estimates*

Description

This function receives a data frame with sampled posterior trophic position estimates. The user may set if he/she wants quantiles to be plotted, in which case the 95 can states whether he/she wants the density plots grouped or not. For visualization purposes, density functions look better if they are not grouped, when quantiles are added.

Usage

```
trophicDensityPlot(df = NULL, quantiles = FALSE, grouped = TRUE)
```

Arguments

df	data frame with 2 variables: "TP" and "Species". TP can be posterior samples of trophic position and Species must be a factor.
quantiles	logical variable. If TRUE 95 distribution, plus mean and median are added to the plot.
grouped	logical variable. If TRUE trophic position density plots are grouped.

Value

a ggplot2::ggplot object

Examples

```
species1 <- stats::rnorm(1000, 4, 0.1)
species2 <- stats::rnorm(1000, 3, 0.8)
TP <- c(species1, species2)
Species <- c(rep("Species 1", length(species1)),
rep("Species 2", length(species2)))
df <- data.frame(TP, Species)
trophicDensityPlot(df)
```

Trout *Named list containing stable isotope values of Oncorhynchus mykiss*

Description

A dataset containing stable isotope values (d13C and d15N) for the invasive trout *Oncorhynchus mykiss* (www.fishbase.se/summary/oncorhynchus-mykiss.html), found in the Chungara Lake at 4,500 meters above sea level, Chilean Altiplano.

Usage

```
data("Trout")
```

Format

A named list with 8 elements:

dCc numeric, stable isotope values of d13C for Oncorhynchus mykiss

dNc numeric, stable isotope values of d15N for Oncorhynchus mykiss

dCb1 numeric, stable isotope values of d13C for baseline 1

dNb1 numeric, stable isotope values of d15N for baseline 1

dCb2 numeric, stable isotope values of d13C for baseline 2

dNb2 numeric, stable isotope values of d15N for baseline 2

deltaN numeric, trophic discrimination factor for nitrogen

deltaC numeric, trophic discrimination factor for carbon

Index