Package ‘QFASA’

June 14, 2019

Title  Quantitative Fatty Acid Signature Analysis
Version 1.0.3
Date 2019-06-12

Description Accurate estimates of the diets of predators are required in many areas of ecology, but for many species current methods are imprecise, limited to the last meal, and often biased. The diversity of fatty acids and their patterns in organisms, coupled with the narrow limitations on their biosynthesis, properties of digestion in monogastric animals, and the prevalence of large storage reservoirs of lipid in many predators, led us to propose the use of quantitative fatty acid signature analysis (QFASA) to study predator diets.

Depends R (>= 3.2)
License MIT + file LICENSE
Encoding UTF-8
LazyData true
Imports stats, Rsolnp, boot, futile.logger, gamlss, gamlss.dist
RoxygenNote 6.1.0
VignetteBuilder rmarkdown, knitr
Suggests knitr, rmarkdown, testthat, plyr, gtools

BugReports https://github.com/justinkamerman/QFASA/issues
URL https://CRAN.R-project.org/package=QFASA

NeedsCompilation no

Author Sara Iverson [aut],
Chris Field [aut],
Don Bowen [aut],
Wade Blanchard [aut],
Connie Stewart [aut, cph],
Shelley Lang [aut],
Justin Kamerman [cre, cph],
Hongchang Bao [ctb]
Maintainer  Justin Kamerman <justin@kaleco.net>

Repository  CRAN

Date/Publication  2019-06-14 17:20:05 UTC

R topics documented:

AIT.dist .................................................. 3
AIT.more .................................................. 3
AIT.obj ................................................... 4
beta.meths.CI ............................................ 4
bias.all .................................................. 6
CC .......................................................... 8
chisq.CA .................................................. 8
chisq.dist ................................................ 9
create.d.mat .............................................. 9
CS.more .................................................. 10
CS.obj .................................................... 10
FAset ...................................................... 11
KL.dist .................................................... 11
KL.more .................................................. 12
KL.obj ..................................................... 12
mean.geometric ........................................... 13
MEANmeth ............................................... 13
p.QFASA .................................................... 14
predatorFAs .............................................. 15
prey.cluster .............................................. 18
prey.on.prey ............................................. 19
preyFAs .................................................... 19
pseudo.pred .............................................. 22
pseudo.seal .............................................. 23
QFASA ...................................................... 24
QFASA.const.eqn ......................................... 24
split.prey .................................................. 24
testfordiff.ind.boot ................................... 25
testfordiff.ind.boot.fun ............................... 25
testfordiff.ind.pval .................................... 26

Index  27
AIT.dist

Returns the distance between two compositional vectors using Aitchison’s distance measure.

Description

Returns the distance between two compositional vectors using Aitchison’s distance measure.

Usage

\[
\text{AIT.dist}(x.1, x.2)
\]

Arguments

- \( x.1 \): compositional vector
- \( x.2 \): compositional vector

References


AIT.more

Used to provide additional information on various model components evaluated at the optimal solution, i.e., using the QFASA diet estimates and Aitchison distance measure.

Description

Used to provide additional information on various model components evaluated at the optimal solution, i.e., using the QFASA diet estimates and Aitchison distance measure.

Usage

\[
\text{AIT.more}(\alpha, \text{predator}, \text{prey.quantiles})
\]

Arguments

- \( \alpha \): compositional QFASA diet estimate.
- \( \text{predator} \): fatty acid signature of predator.
- \( \text{prey.quantiles} \): matrix of fatty acid signatures of prey. Each row contains an individual prey signature from a different species.
AIT.obj

 Used in solnp() as the objective function to be minimized when Aitchison distance measure is chosen.

Description

Used in solnp() as the objective function to be minimized when Aitchison distance measure is chosen.

Usage

AIT.obj(alpha, predator, prey.quantiles)

Arguments

alpha vector over which minimization takes place.
predator fatty acid signature of predator.
prey.quantiles matrix of fatty acid signatures of prey. Each row contains an individual prey signature from a different species.

beta.meths.CI

Returns individual confidence intervals and simultaneous confidence intervals based on the zero-inflated beta distribution (not bias corrected - see note below).

Description


Usage

beta.meths.CI(predator.mat, prey.mat, cal.mat = rep(1, length(ext.fa)),
dist.meas, noise = 0, nprey, R.p, R.ps, R, p.mat, alpha, FC = rep(1,
nrow(prey.mat)), ext.fa)

Arguments

predator.mat matrix containing the fatty acid signatures of the predators.
prey.mat prey database. A dataframe with first column a Species label and other columns fatty acid proportions. Fatty acid proportions are compositional.
cal.mat matrix of calibration coefficients of predators. Each column corresponds to a different predator. At least one calibration coefficient vector must be supplied.
dist.meas distance measure to use for estimation: 1=KL, 2=AIT or 3=CS
noise proportion of noise to include in the simulation.
nprey number of prey to sample from the prey database when generating pseudo-predators for the nuisance parameter estimation.
R.p number of beta diet distributions to generate for the nuisance parameters.
R.ps number of pseudo predators to generate when estimating nuisance parameters.
R number of bootstrap replicates to use when generating p-values for confidence interval estimation.
p.mat matrix of predator diet estimates for which we are trying to find confidence intervals.
alpha confidence interval confidence level.
FC vector of prey fat content. Note that this vector is passed to the gen.pseudo.seals which expects fat content values for individual prey samples while pseudo.seal and p.QFASA expect a species average.
ext.fa subset of fatty acids to be used to obtain QFASA diet estimates.

Details
Note:
- These intervals are biased and should be corrected using the output from bias.all.
- CI.L.1 and CI.U.1 contain the simultaneous confidence intervals.
- Slow because of bisection and lots of repetition.

Value
Individual confidence intervals and simultaneous confidence intervals based on the zero-inflated beta distribution. These intervals are biased and should be corrected using the output from bias.all. ci.l.1 and ci.u.1 contain the simultaneous confidence intervals.

References

Examples
```r
## Fatty Acids
data(FAset)
fa.set = as.vector(unlist(FAset))

## Predators
data(predatorFAs)
tombstone.info = predatorFAs[,1:4]
predator.matrix = predatorFAs[, fa.set]
npredators = nrow(predator.matrix)

## Prey
prey.sub = preyFAs[, fa.set]
```
bias.all

Calculate bias correction for confidence intervals from beta.meths.CI.

Description

Calculate bias correction for confidence intervals from beta.meths.CI.

Usage

bias.all(p.mat, prey.mat, cal.mat = rep(1, length(ext.fa)),
         fat.cont = rep(1, nrow(prey.mat)), R.bias, noise, nprey, specify.noise,
         dist.meas, ext.fa)

Arguments

p.mat matrix containing the fatty acid signatures of the predators.
prey.mat matrix containing a representative fatty acid signature

bias.all = prey.sub / apply(prey.sub, 1, sum)
group = as.vector(preyFAs$Species)
prey.matrix.full = cbind(group, prey.sub)
prey.matrix = MEANmeth(prey.matrix.full)

## Calibration Coefficients
data(CC)
cal.vec = CC[CC$FA %in% fa.set, 2]
cal.mat = replicate(npredators, cal.vec)

# Note: uncomment examples to run. CRAN tests fail because execution time > 5 seconds
# set.seed(1234)
# diet.est <- p.QFASA(predator.mat = predator.matrix,
#                      prey.mat = prey.matrix,
#                      cal.mat = cal.mat,
#                      dist.meas = 2,
#                      start.val = rep(1,nrow(prey.matrix)),
#                      ext.fa = fa.set)[['Diet Estimates']]
#
# ci = beta.meths.CI(predator.mat = predator.matrix,
#                     prey.mat = prey.matrix.full,
#                     cal.mat = cal.mat,
#                     dist.meas = 2,
#                     nprey = 10,
#                     R.p = 1,
#                     R.ps = 10, #
#                     R = 1,
#                     p.mat = diet.est,
#                     alpha = 0.05,
#                     ext.fa = fa.set)
bias.all

```r
cal.mat  # matrix of calibration factors where the i th column is to be used with the i th predator. If modelling is to be done without calibration coefficients, simply pass a vector or matrix of ones.

fat.cont  # prey fat content
R.bias    # bootstrap replicates
noise     # noise
nprey     # number of prey
specify.noise  # noise
dist.meas # distance measure
ext.fa    # subset of FA's to use.

Value
Row 1 is Lambda CI, row 2 is Lambda skew, and row 3 is Beta CI

Examples
```r
data(FAset)
fa.set = as.vector(unlist(FAset))

# Predators
data(predatorFAs)
tombstone.info = predatorFAs[,1:4]
predator.matrix = predatorFAs[, fa.set]
npredators = nrow(predator.matrix)

# Prey
prey.sub = preyFAs[, fa.set]
prey.sub = prey.sub / apply(prey.sub, 1, sum)
group = as.vector(preyFAs$Species)
prey.matrix.full = cbind(group, prey.sub)
prey.matrix = MEANmeth(prey.matrix.full)

# Calibration Coefficients
data(CC)
cal.vec = CC[CC$FA %in% fa.set, 2]
cal.mat = replicate(npredators, cal.vec)

# Note: uncomment examples to run. CRAN tests fail because execution time > 5 seconds
# diet.est <- p.QFASA(predator.mat = predator.matrix,
#        prey.mat = prey.matrix,
#        cal.mat = cal.mat,
#        dist.meas = 2,
#        start.val = rep(1,nrow(prey.matrix)),
#        ext.fa = fa.set)[['Diet Estimates']]
# bias <- bias.all(p.mat = diet.est,
#        prey.mat = prey.matrix.full,
#        cal.mat = cal.mat,
#        ext.fa = fa.set)[['Diet Estimates']]
chisq.CA

# R.bias = 10,
# noise = 0,
# nprey = 50,
# dist.meas = 2,
# ext.fa = fa.set

CC  

Fatty acid calibration coefficients.

Description

Fatty acid calibration coefficients.

Usage

CC

Format

A data frame with 66 observations and 2 variables:

FA  fatty acid names
CC  calibration coefficient for corresponding fatty acid

chisq.CA

Called by create.d.mat() to compute the chi-square distance.

Description

Called by create.d.mat() to compute the chi-square distance.

Usage

chisq.CA(x1, x2)

Arguments

x1  vector
x2  vector
chisq.dist

Returns the distance between two compositional vectors using the chi-square distance.

Usage

chisq.dist(x.1, x.2, gamma)

Arguments

x.1 compositional vector
x.2 compositional vector
gamma power transform taken to be 1.

References


create.d.mat

Called by testfordiff.ind.boot.fun() to create a matrix of distances.

Description

Called by testfordiff.ind.boot.fun() to create a matrix of distances.

Usage

create.d.mat(Y.1, Y.2)

Arguments

Y.1 vector
Y.2 vector
### CS.more

Used to provide additional information on various model components evaluated at the optimal solution, i.e., using the QFASA diet estimates and chi-square distance measure.

**Description**

Used to provide additional information on various model components evaluated at the optimal solution, i.e., using the QFASA diet estimates and chi-square distance measure.

**Usage**

```r
CS.more(alpha, predator, prey.quantiles, gamma)
```

**Arguments**

- `alpha`: compositional QFASA diet estimate.
- `predator`: fatty acid signature of predator.
- `prey.quantiles`: matrix of fatty acid signatures of prey. Each row contains an individual prey signature from a different species.
- `gamma`: power transform exponent (see `chisq.dist()`).

### CS.obj

Used in `solnp()` as the objective function to be minimized when chi-square distance measure is chosen. Unlike `AIT.obj()` and `KL.obj()`, does not require modifying zeros.

**Description**

Used in `solnp()` as the objective function to be minimized when chi-square distance measure is chosen. Unlike `AIT.obj()` and `KL.obj()`, does not require modifying zeros.

**Usage**

```r
CS.obj(alpha, predator, prey.quantiles, gamma)
```

**Arguments**

- `alpha`: vector over which minimization takes place.
- `predator`: fatty acid signature of predator.
- `prey.quantiles`: matrix of fatty acid signatures of prey. Each row contains an individual prey signature from a different species.
- `gamma`: power transform exponent (see `chisq.dist()`).
**FAset**

List of fatty acids used in sample prey and predator data sets, `preyfas` and `predatorfas` respectively.

---

**Description**

List of fatty acids used in sample prey and predator data sets, `preyfas` and `predatorfas` respectively.

**Usage**

`FAset`

**Format**

A data frame with 39 observations and 1 variable:

- **FA** Fatty acid name

---

**KL.dist**

Returns the distance between two compositional vectors using Kullback–Leibler distance measure.

---

**Description**

Returns the distance between two compositional vectors using Kullback–Leibler distance measure.

**Usage**

`KL.dist(x.1, x.2)`

**Arguments**

- **x.1** compositional vector
- **x.2** compositional vector

**References**

**KL.more**  
*Used to provide additional information on various model components evaluated at the optimal solution, i.e., using the QFASA diet estimates and Kullback-Leibler distance measure.*

**Description**

Used to provide additional information on various model components evaluated at the optimal solution, i.e., using the QFASA diet estimates and Kullback-Leibler distance measure.

**Usage**

```R
KL.more(alpha, predator, prey.quantiles)
```

**Arguments**

- `alpha`: compositional QFASA diet estimate.
- `predator`: fatty acid signature of predator.
- `prey.quantiles`: matrix of fatty acid signatures of prey. Each row contains an individual prey signature from a different species.

---

**KL.obj**  
*Used in solnp() as the objective function to be minimized when Kullback–Leibler distance measure is chosen.*

**Description**

Used in `solnp()` as the objective function to be minimized when Kullback–Leibler distance measure is chosen.

**Usage**

```R
KL.obj(alpha, predator, prey.quantiles)
```

**Arguments**

- `alpha`: vector over which minimization takes place.
- `predator`: fatty acid signature of predator.
- `prey.quantiles`: matrix of fatty acid signatures of prey. Each row contains an individual prey signature from a different species.
**mean.geometric**

*Returns the geometric mean of a compositional vector*

**Description**

Returns the geometric mean of a compositional vector

**Usage**

```r
## S3 method for class 'geometric'
mean(x)
```

**Arguments**

- `x`: compositional vector

---

**MEANmeth**

*Returns the multivariate mean FA signature of each prey group entered into the QFASA model. Result can be passed to prey.mat in p.QFASA().*

**Description**

Returns the multivariate mean FA signature of each prey group entered into the QFASA model. Result can be passed to prey.mat in p.QFASA().

**Usage**

```r
MEANmeth(prey.mat)
```

**Arguments**

- `prey.mat`: matrix containing the FA signatures of the prey. The first column indexes the prey group.
p.QFASA Computes the diet estimate for each predator in seal.mat using either the Kullback-Leibler Distance (KL), the Aitchison Distance (AIT) or the Chi-Square Distance (CS).

Description

Computes the diet estimate for each predator in seal.mat using either the Kullback-Leibler Distance (KL), the Aitchison Distance (AIT) or the Chi-Square Distance (CS).

Usage

p.QFASA(predator.mat, prey.mat, cal.mat, dist.meas, gamma = 1,
FC = rep(1, nrow(prey.mat)), start.val = rep(0.99999,
 nrow(prey.mat)), ext.fa)

Arguments

- predator.mat: matrix containing the FA signatures of the predators.
- prey.mat: matrix containing a representative FA signature from each prey group (usually the mean). The first column must index the prey group.
- cal.mat: matrix of calibration factors where the $i$th column is to be used with the $i$th predator. If modelling is to be done without calibration coefficients, simply pass a vector or matrix of ones.
- dist.meas: distance measure to use for estimation: 1=KL, 2=AIT or 3=CS
- gamma: parameter required for calculations using CS distance (passed to CS.obj). Currently being set to 1.
- FC: vector of fat content
- start.val: initial vector of parameters to be optimized
- ext.fa: subset of fatty acids to be used to obtain QFASA diet estimates.

Value

a list with components:

- **Diet Estimates** A matrix of the diet estimates for each predator where each row corresponds to a predator and the columns to prey species. The estimates are expressed as proportions summing to one.

- **Additional Measures**
  - **ModFAS** the value of the modelled fatty acid (i.e., after CCs have been applied and the fatty acids subsetted and renormalised over the designated fatty acid set). These are expressed as proportions summing to one.
  - **DistCont** The contribution of each fatty acid to the final minimized distance.
`predatorFAs`

**PropDistCont**  
The contribution of each fatty acid to the final minimized distance as a proportion of the total.

**MinDist**  
The final minimized distance.

**Examples**

```r
## Fatty Acids
data(FAset)
fa.set = as.vector(unlist(FAset))

## Predators
data(predatorFAs)
tombstone.info = predatorFAs[,1:4]
predator.matrix = predatorFAs[,5:(ncol(predatorFAs))] npredators = nrow(predator.matrix)

## Prey
data(preyFAs)
prey.sub=(preyFAs[,4:(ncol(preyFAs))][fa.set]
prey.sub=prey.sub/apply(prey.sub,1,sum)
group=as.vector(preyFAs$Species)
prey.matrix=cbind(group,prey.sub)
prey.matrix=MEANmeth(prey.matrix)

FC = preyFAs[,c(2,3)]
FC = as.vector(tapply(FC$lipid,FC$Species,mean,na.rm=TRUE))

## Calibration Coefficients
data(CC)
cal.vec = CC[,2]
cal.mat = replicate(npredators, cal.vec)

# Run QFASA
Q = p.QFASA(predator.matrix,
prey.matrix,
cal.mat,
dist.meas = 1,
gamma=1,
FC,
start.val = rep(1,nrow(prey.matrix)),
fa.set)
```

**predatorFAs**  
*Predator fatty acid signatures. Each predator signature is a row with fatty acid proportions in columns.*
Description

Fatty acid signatures are subsetted for the chosen fatty acid set and renormalized during the modelling so there is no need to subset and/or renormalize prior to running p.QFASA. However, make sure that the the same fatty acids appear in the predator and prey files (if a FA appears in one but not the other the code will give you an error).

Usage

predatorFAs

Format

A data frame with 10 observations and 70 variables:

SampleCode  TODO
AnimalCode  TODO
SampleGroup  TODO
Biopsy  TODO
c12.0
c13.0
Iso14
c14.0
c14.1w9
c14.1w7
c14.1w5
Iso15
Anti15
c15.0
c15.1w8
c15.1w6
Iso16
c16.0
c16.1w11
c16.1w9
c16.1w7
c7Mec16.0
c16.1w5
c16.2w6
Iso17
c16.2w4
c16.3w6
c17.0
16.3w4
c17.1
16.4w3
16.4w1
c18.0
c18.1w13
18.1w11
18.1w9
18.1w7
18.1w5
c18.2d5.11
18.2w7
18.2w6
18.2w4
18.3w6
18.3w4
18.3w3
18.3w1
c18.4w3
c18.4w1
c20.0
c20.1w11
c20.1w9
c20.1w7
c20.2w9
c20.2w6
c20.3w6
c20.4w6
c20.3w3
c20.4w3
c20.5w3
c22.1w11
c22.1w9
c22.1w7
c22.2w6
c21.5w3
prey.cluster

Details

Unlike the original QFASApack code the predator data can contain as much tombstone data in columns as you wish but the predator FA signatures must be extracted as a separate input in order to run in p.QFASA.

prey.cluster

This function performs a hierarchical cluster analysis of prey fatty acid signatures using a matrix of dissimilarities for the n objects being clustered. Initially, each object is assigned as its own cluster and then the algorithm proceeds iteratively, at each stage joining the two most similar clusters, until there is just a single cluster.

Description

This function performs a hierarchical cluster analysis of prey fatty acid signatures using a matrix of dissimilarities for the n objects being clustered. Initially, each object is assigned as its own cluster and then the algorithm proceeds iteratively, at each stage joining the two most similar clusters, until there is just a single cluster.

Usage

prey.cluster(prey.fa, method, FUN)

Arguments

- **prey.fa**: data frame of prey fatty acid signature samples. Species column is used to group samples. Other columns are assumed to be fatty acid proportions.
- **method**: the agglomeration method to be used. This should be one of 'single', 'complete', 'average', 'median', 'centroid'.
- **FUN**: distance function

Value

an object of class hclust which describes the tree produced by the clustering process.
prey.on.prey

Each prey fatty acid signature is systematically removed from the supplied prey database and its QFASA diet estimate is obtained by treating the individual as a predator.

Description

Each prey fatty acid signature is systematically removed from the supplied prey database and its QFASA diet estimate is obtained by treating the individual as a predator.

Usage

prey.on.prey(preysbase, dist.meas, gamma = 1)

Arguments

preysbase first column is name of species and remaining columns are fatty acids.
dist.meas see help file for p.QFASA.
gamma see help file for p.QFASA.

Value

diet estimate

Examples

data(preyFAs)
my.preysbase <- preyFAs[, -c(1,3)]

# Note: uncomment examples to run. CRAN tests fail because execution time > 5 seconds
# diets.out <- prey.on.prey(my.preysbase, 2)
# round(MEANmeth(diets.out), 3)

preyFAs

Prey fatty acid signatures. Each prey signature is a row with fatty acid proportions in columns.

Description

The prey file should contain all of the individual fatty acid signatures of the prey and their lipid contents (where appropriate) - a matrix of the mean values for the FAs (prey.matrix) by the designated prey modelling group is then calculated using the MEANmeth function.
Usage

preyFAs

Format

A data frame with 302 observations and 70 variables:

Lab.Code  TODO
Species  TODO
lipid  TODO
c12.0
c13.0
Iso14
c14.0
c14.1w9
c14.1w7
c14.1w5
Iso15
Anti15
c15.0
c15.1w8
c15.1w6
Iso16
c16.0
c16.1w11
c16.1w9
c16.1w7
c7Me16.0
c16.1w5
c16.2w6
Iso17
c16.2w4
c16.3w6
c17.0
c16.3w4
c17.1
c16.3w1
c16.4w3
c16.4w1
c18.0
c18.1w13
c18.1w11
c18.1w9
c18.1w7
c18.1w5
c18.2d5.11
c18.2w7
c18.2w6
c18.2w4
c18.3w6
c18.3w4
c18.3w3
c18.3w1
c18.4w3
c18.4w1
c20.0
c20.1w11
c20.1w9
c20.1w7
c20.2w9
c20.2w6
c20.3w6
c20.4w6
c20.3w3
c20.4w3
c20.5w3
c22.1w11
c22.1w9
c22.1w7
c22.2w6
c21.5w3
c22.4w6
c22.5w6
c22.4w3
c22.5w3
c22.6w3
c24.1w9
Details

Like the predator .csv file you can have as many tombstone data columns as required but there must be at least one column that identifies the modelling group, in this case, Species.

Unlike the predator data, the prey data is not subsetted and renormalized during the modelling so the prey file needs to be subsetted for the desired fatty acid set and renormalized to sum to 1 prior to calculating the mean values.

The full FA set is extracted from the data frame (columns 4 onward), subsetted for the FA set in use and then renormalized over 1. The modelling group names (the "Species" column in this case) is then added back to the subsetted and renormalized data (as the first column) and the average values calculated using the MEANmeth function. Note that for the MEANmeth function to work the modelling group name must be in the first column.

pseudo.pred

Generate a pseudo predator by sampling with replacement from prey database.

Description

Note: if preysize=1, then one prey is selecting from each species. otherwise, a sample of size n_k (number of species k) is sampled with replacement.

Usage

pseudo.pred(diet, preybase, cal.vec, fat.vec, preysize = 2)

Arguments

diet compositional vector of proportions that sums to one. Length is equal to the number of prey species.

preybase prey database with first column providing the species name.

cal.vec vector of calibration coefficients.

fat.vec vector of fat content whose length is the same as the number of species.

preysize number of prey to sample from prey database.

Value

a simulated predator FA signature

Examples

data(preyFAs)

# Note: uncomment examples to run. CRAN tests fail because execution time > 5 seconds
# p.mat <- matrix(rep(NA,100*11),nrow=100)
# for (i in 1:100) {
#  my.seal <- pseudo.pred(rep(1/11,11),
pseudo.seal

#
# preyFAs[-c(1,3)],
# rep(1, ncol(preyFAs[-c(1,3)]) - 1),
# rep(1, 11))
# p.mat[1,] <- p.QFASA(my.seal,
# MEANmeth(preyFAs[-c(1,3)]),
# rep(1, length(my.seal)),
# 2,
# ext.fa=colnames(preyFAs[-c(1:3)])$Diet Estimates'
# }
#
# Average diet estimate
# round(apply(p.mat, 2, mean), 3)

---

**pseudo.seal**

*Generate a single pseudo predator FA signature*

**Description**

THIS IS THE NEW pseudo.seal FUNCTION THAT ALLOWS 1) FAT CONTENT TO BE INCLUDED IN THE GENERATED SEALS AND 2) SOME SPECIES TO BE TRULY ZERO (THAT IS, "ZERO SPECIES" DO NOT HAVE TO BE INCLUDED IN THE "NOISE") NOTE: IT IS ASSUMED THAT SUM(DIET) IS 1-NOISE

**Usage**

```r
pseudo.seal(prey.sim, diet, noise, nprey, cal, fat.cont, specify.noise)
```

**Arguments**

- **prey.sim** - OUTPUT OF split.prey
- **diet** - DIET COMPOSITION VECTOR (NOTE: THIS VECTOR SHOULD SUM TO 1-NOISE. THE NOISE WILL BE ADDED TO THE diet VECTOR.)
- **noise** - AMOUNT OF NOISE
- **nprey** - nprey TOTAL NUMBER OF PREY TO BE SAMPLED
- **cal** - CALIBRATION FACTORS
- **fat.cont** - VECTOR OF FAT CONTENT OF LENGTH=I (# OF SPECIES)
- **specify.noise** - A BOOLEAN VECTOR WITH TRUES DENOTING SPECIES TO USE IN NOISE.

**Value**

`seal.star` SIMULATED SEAL FA SIGNATURE.
**QFASA**

QFASA: A package for Quantitative Fatty Acid Signature Analysis

**Description**

Accurate estimates of the diets of predators are required in many areas of ecology, but for many species current methods are imprecise, limited to the last meal, and often biased. The diversity of fatty acids and their patterns in organisms, coupled with the narrow limitations on their biosynthesis, properties of digestion in monogastric animals, and the prevalence of large storage reservoirs of lipid in many predators, led us to propose the use of quantitative fatty acid signature analysis (QFASA) to study predator diets.

**QFASA.const.eqn**

*Returns sum(alpha) and used in solnp().*

**Description**

Returns sum(alpha) and used in solnp().

**Usage**

QFASA.const.eqn(alpha, predator, prey.quantiles, gamma)

**Arguments**

- alpha: vector over which minimization takes place.
- predator: fatty acid signature of predator.
- prey.quantiles: matrix of fatty acid signatures of prey. Each row contains an individual prey signature from a different species.
- gamma: power transform exponent (see chisq.dist).

**split.prey**

*Splits prey database into a simulation set (1/3) and a modelling set (2/3). Returns a list:*

**Description**

1. simulation prey database 2. modelling prey database

**Usage**

```r
# S3 method for class 'prey'
split(prey.mat)
```
Arguments

prey.mat  matrix of individual prey fatty acid signatures where the first column denotes the prey type

Details

IF number of samples of a prey type \(\leq 5\), then prey.mod AND prey.sim are duplicated instead of split.

testfordiff.ind.boot  Called by testfordiff.ind.pval().

description

Called by testfordiff.ind.pval().

Usage

testfordiff.ind.boot(data, nsQ, R)

Arguments

data  sample of compositional data
nsQ    sample size of compdata.1
R       number of bootstrap samples. default is 500.

testfordiff.ind.boot.fun

Called by testfordiff.ind.boot().

Description

Called by testfordiff.ind.boot().

Usage

testfordiff.ind.boot.fun(data, i, nsQ, change.zero = 1e-05)

Arguments

data  sample of compositional data
i      row index
nsQ    sample size of compdata.1
change.zero  tolerance
Test for a difference between two independent samples of compositional data. Zeros of any type are allowed.

Description

Test for a difference between two independent samples of compositional data. Zeros of any type are allowed.

Usage

testfordiff.ind.pval(compdata.1, compdata.2, ns1, R = 500)

Arguments

- compdata.1: sample of compositional data.
- compdata.2: sample of compositional data.
- ns1: sample size of compdata.1.
- R: number of bootstrap samples, default is 500.

Value

p-value obtained through a multivariate permutation test with test statistic based on chi-square distances.

References


Examples

```r
## Prey
data(preyFAs)

## Capelin FA sig
capelin.sig=preyFAs[preyFAs$Species=="capelin",4:(ncol(preyFAs))]
capelin.sig=capelin.sig/apply(capelin.sig,1,sum)

## Sandlance FA sig
sandlance.sig=preyFAs[preyFAs$Species=="sandlance",4:(ncol(preyFAs))]
sandlance.sig=sandlance.sig/apply(sandlance.sig,1,sum)

# Note: uncomment examples to run. CRAN tests fail because execution time > 5 seconds
# testfordiff.ind.pval(as.matrix(capelin.sig),
# as.matrix(sandlance.sig),
# nrow(capelin.sig))
```
Index

*Topic **datasets**
  CC, 8
  FSet, 11
  predatorFAs, 15
  preyFAs, 19

AIT.dist, 3
AIT.more, 3
AIT.obj, 4

beta.meths.CI, 4, 6
bias.all, 5, 6

CC, 8
chisq.CA, 8
chisq.dist, 9
create.d.mat, 9
CS.more, 10
CS.obj, 10

FSet, 11

gen.pseudo.seals, 5

KL.dist, 11
KL.more, 12
KL.obj, 12

mean.geometric, 13
MEANmeth, 13

p.QFASA, 5, 14, 19
predatorFAs, 15
prey.cluster, 18
prey.on.prey, 19
preyFAs, 19
pseudo.pred, 22
pseudo.seal, 5, 23

QFASA, 24
QFASA-package (QFASA), 24

QFASA.const.eqn, 24
split.prey, 24
testfordiff.ind.boot, 25
testfordiff.ind.boot.fun, 25
testfordiff.ind.pval, 26