Package 'qfasar'

January 10, 2017

Type Package

Title Quantitative Fatty Acid Signature Analysis in R

Version 1.2.0

Description An implementation of Quantitative Fatty Acid Signature Analysis (QFASA) in R. QFASA is a method of estimating the diet composition of predators. The fundamental unit of information in QFASA is a fatty acid signature (signature), which is a vector of proportions describing the composition of fatty acids within lipids. Signature data from at least one predator and from samples of all potential prey types are required. Calibration coefficients, which adjust for the differential metabolism of individual fatty acids by predators, are also required. Given those data inputs, a predator signature is modeled as a mixture of prey signatures and its diet estimate is obtained as the mixture that minimizes a measure of distance between the observed and modeled signatures. A variety of estimation options and simulation capabilities are implemented. Please refer to the vignette for additional details and references.

License Unlimited

LazyData TRUE

Imports Rsolnp (>= 1.16)

RoxygenNote 5.0.1

Suggests knitr, rmarkdown, testthat

VignetteBuilder knitr

Date 2017-01-09

NeedsCompilation no

Author Jeffrey F. Bromaghin [aut, cre]

Maintainer Jeffrey F. Bromaghin <jbromaghin@usgs.gov>

Repository CRAN

Date/Publication 2017-01-10 00:04:44

R topics documented:

`add_cc_err` *Add error to the calibration coefficients*

Description

Bromaghin et al (2016) studied the performance of QFASA estimators when predator diets were estimated using calibration coefficients that incorporated a degree of error. `add_cc_err` implements their method of adding error to a set of calibration coefficients.

Usage

```
add_cc_err(cc_true, err_bound)
```

Arguments

cc_true	A vector of calibration coefficients, intended to be the object cc returned by the function prep_fa.
err_bound	A proportion strictly greater than 0 and less than 1 used to control the lower and upper bounds of calibration coefficient error.

Value

A list containing the following elements:

cc A numeric vector of calibration coefficients with error incorporated.

err The mean relative absolute error in the calibration coefficients.

err_code An integer error code (0 if no error is detected).

err_message A string contains a brief summary of the execution.

Details

One of the major assumptions of QFASA is that the calibration coefficients are known perfectly. Bromaghin et al. (2016) investigated the robustness of diet estimators to violations of this assumption. The function add_cc_err uses the methods of Bromaghin et al. (2016) to add error to a set of calibration coefficients.

The argument err_bound is used to compute box constraints for the calibration coefficients: lower bound equals (1 - err_bound)*cc_true and upper bound equals (1 + err_bound)*cc_true. A uniformly distributed random number is generated between the bounds for each calibration coefficient and the vector of coefficients is scaled so that their sum equals the sum of the true calibration coefficients. Because only the relative magnitudes of the calibration coefficients are important in diet estimation, scaling the coefficients to have a common sum ensures comparability between multiple sets of coefficients.

The mean relative absolute difference between the true and error-added calibration coefficients is computed as a measure of error for the entire vector.

References

Bromaghin, J.F., S.M. Budge, G.W. Thiemann, and K.D. Rode. 2016. Assessing the robustness of quantitative fatty acid signature analysis to assumption violations. *Methods in Ecology and Evolution* 7:51-59.

Examples

```
add_cc_err(cc_true = c(0.75, 1.00, 1.50, 1.15),
           err_bound = 0.25)
```

adj_diet_fat *Adjust diet composition estimates for prey fat content*

Description

The function est_diet estimates diet composition in terms of the mass of fatty acids consumed. The function adj_diet_fat uses estimates of fatty acid mass per prey type to transform estimates of diet composition in terms of fatty acid mass to another scale.

Usage

```
adj_diet_fat(prey_fat, diet_est, diet_var = NA)
```

Arguments

prey_fat	A numeric vector of the mean fatty acid mass for each prey type.
diet_est	A numeric vector or matrix of the estimated diet composition(s) of individual predator(s) or predator type(s). Intended to be the object est_ind or est_meanreturned by the function est_diet.
diet_var	A numeric matrix or array containing the estimated variance matrix for the estimated diet(s) of individual predator(s) or predator type(s). Intended to be the object var_ind or var_mean) returned by the function est_diet. Optional.

Value

A list containing the following elements:

diet_est A numeric vector or matrix of the estimated diet composition of individual predator(s) or predator type(s) in terms of the scale represented by prey_fat.

diet_var A numeric matrix or array containing the estimated variance matrix for the estimated diet of individual predator(s) or predator type(s), in terms of the scale represented by prey_fat.

err_code An integer error code (0 if no error is detected).

err_message A string contains a brief summary of the execution.

Details

The function est_diet estimates diet composition in terms of the mass of fatty acids consumed. Such estimates may be of direct ecological interest, especially for ecosystems in which lipids are particularly important. In other cases, one may wish to transform estimates to a different scale. For example, if the units of prey_fat are (fatty acid mass)/ (animal mass), the function adj_diet_fat returns diet composition estimates in terms of the relative prey mass consumed. Alternatively, if the units of prey_fat are (fatty acid mass)/(animal), the function adj_diet_fat returns diet composition estimates in terms of the relative numbers of prey animals consumed.

adj_diet_fat uses the fat transformation of Iverson et al. (2004). Variance matrices are estimated using the delta method (Seber 1982).

NOTE: values of mass per prey type are treated as known constants without variance.

References

Iverson, S.J., C. Field, W.D. Bowen, and W. Blanchard. 2004. Quantitative fatty acid signature analysis: A new method of estimating predator diets. *Ecological Monographs* 74:211-235.

Seber, G.A.F. 1982. The Estimation of Animal Abundance and Related Parameters, second edition. Macmillan Publishing Co., New York.

Examples

```
adj_diet_fat(prey_fat = c(0.5, 1, 2),
             diet_est = c(0.3, 0.2, 0.5),
             diet_var = matrix(c( 0.030,  0.004, -0.003,
                                  0.004,  0.025, -0.007,
                                 -0.003, -0.007,  0.045),
                               nrow = 3, ncol = 3))

adj_diet_fat(prey_fat = c(0.5, 1, 2),
             diet_est = c(0.3, 0.2, 0.5))
```

---

cc_aug *Calibration coefficient for an augmented signature proportion*

Description

cc_aug computes the optimal calibration coefficient for an augmented signature proportion (Bromaghin et al. 2016). If signature augmentation is to be used, the user must call cc_aug after first calling prep_sig with the prey signature data, in order to derive a calibration coefficient for the augmented signature.

Usage

```
cc_aug(sig_rep, sig_scale, cc_all, use_fa, dist_meas = 1, gamma = 1)
```

Arguments

sig_rep	A numeric matrix containing fatty acid signatures with proportions from all fatty acids. See Details.
sig_scale	A numeric matrix containing fatty acid signatures with proportions from a subset of all fatty acids and an augmented proportion. See Details.
cc_all	A numeric vector of calibration coefficients for the fatty acids in sig_rep.
use_fa	A logical vector denoting the fatty acids in sig_rep that are also in sig_scale.
dist_meas	An integer indicator of the distance measure to compute. Default value 1.
gamma	The power parameter of the chi-square distance measure. Default value 1.

Value

A list containing the following elements:

cc A numeric vector of calibration coefficients for the augmented signatures.

err_code An integer error code (0 if no error is detected).

err_message A string contains a brief summary of the execution.

Details

Calibration coefficients provide a one-to-one mapping between the prey and predator spaces (Bromaghin et al. 2015). However, when using signature augmentation (Bromaghin et al. 2016), no calibration coefficient is available for the augmented proportion and the function `cc_aug` was developed to remedy that lack.

`cc_aug` transforms complete prey signatures in `sig_rep` to the predator space, censors them using `fa`, and then augments them. The subset of calibration coefficients in `cc_all` corresponding to `fa` are combined with a calibration coefficient for the augmented proportion, the censored signatures in `sig_scale` are also transformed to the predator space, and the distance between the two sets of censored signatures is computed. The calibration coefficient for the augmented proportion is taken as the value that minimizes the distance. The function Rsolnp::solnp() is used to minimize the distance.

The entity passed as the argument `fa` is intended to be the corresponding entity returned by a call to `prep_fa`. Similarly, the entities passed as the arguments `sig_rep` and `sig_scale` are intended to be the corresponding entities returned by a call to `prep_sig` with data in a prey library. Consequently, no error checks are made on these objects. Please refer to the documentation for `prep_fa` and `prep_sig` for additional details.

Use of Rsolnp::solnp limits the ability to return any errors from the function `dist_between_2_sigs`. A crash may be caused by passing invalid values for the arguments `dist_meas` or `gamma`. Please refer to documentation for the function `dist_between_2_sigs` for additional information about valid values for these arguments.

Utility and external functions called by `cc_aug`:

- `dist_between_2_sigs`
- Rsolnp::solnp

References

Bromaghin, J.F., S.M. Budge, and G.W. Thiemann. 2016. Should fatty acid signature proportions sum to 1 for diet estimation? *Ecological Research* 31:597-606.

Bromaghin, J.F., K.D. Rode, S.M. Budge, and G.W. Thiemann. 2015. Distance measures and optimization spaces in quantitative fatty acid signature analysis. *Ecology and Evolution* 5:1249-1262.

Examples

```
cc_aug(sig_rep = matrix(c(0.05, 0.10, 0.30, 0.55,
                          0.04, 0.11, 0.29, 0.56,
                          0.10, 0.05, 0.35, 0.50), ncol = 3),
```

```
        sig_scale = matrix(c(0.40, 0.50, 0.10,
                             0.45, 0.49, 0.06,
                             0.35, 0.45, 0.20), ncol = 3),
        cc_all = c(0.75, 1.05, 1.86, 0.80),
        use_fa = c(FALSE, FALSE, TRUE, TRUE))

cc_aug(sig_rep = matrix(c(0.05, 0.10, 0.30, 0.55,
                          0.04, 0.11, 0.29, 0.56,
                          0.10, 0.05, 0.35, 0.50), ncol = 3),
        sig_scale = matrix(c(0.40, 0.50, 0.10,
                             0.45, 0.49, 0.06,
                             0.35, 0.45, 0.20), ncol = 3),
        cc_all = c(0.75, 1.05, 1.86, 0.80),
        use_fa = c(FALSE, FALSE, TRUE, TRUE),
        dist_meas = 1)

cc_aug(sig_rep = matrix(c(0.05, 0.10, 0.30, 0.55,
                          0.04, 0.11, 0.29, 0.56,
                          0.10, 0.05, 0.35, 0.50), ncol = 3),
        sig_scale = matrix(c(0.40, 0.50, 0.10,
                             0.45, 0.49, 0.06,
                             0.35, 0.45, 0.20), ncol = 3),
        cc_all = c(0.75, 1.05, 1.86, 0.80),
        use_fa = c(FALSE, FALSE, TRUE, TRUE),
        dist_meas = 2)

cc_aug(sig_rep = matrix(c(0.05, 0.10, 0.30, 0.55,
                          0.04, 0.11, 0.29, 0.56,
                          0.10, 0.05, 0.35, 0.50), ncol = 3),
        sig_scale = matrix(c(0.40, 0.50, 0.10,
                             0.45, 0.49, 0.06,
                             0.35, 0.45, 0.20), ncol = 3),
        cc_all = c(0.75, 1.05, 1.86, 0.80),
        use_fa = c(FALSE, FALSE, TRUE, TRUE),
        dist_meas = 3,
        gamma = 0.25)

cc_aug(sig_rep = matrix(c(0.05, 0.10, 0.30, 0.55,
                          0.04, 0.11, 0.29, 0.56,
                          0.10, 0.05, 0.35, 0.50), ncol = 3),
        sig_scale = matrix(c(0.40, 0.50, 0.10,
                             0.45, 0.49, 0.06,
                             0.35, 0.45, 0.20), ncol = 3),
        cc_all = c(0.75, 1.05, 1.86, 0.80),
        use_fa = c(FALSE, FALSE, TRUE, TRUE),
        dist_meas = 3)
```

---

comp_chi_gamma *Compute gamma parameter of chi-square distance measure*

Description

The function comp_chi_gamma computes the gamma parameter of the chi-square distance measure using the algorithm of Stewart et al. (2014).

Usage

```
comp_chi_gamma(sigs, cc, near_zero = 1e-05, min_gamma = 0.05, space = 1)
```

Arguments

sigs	A matrix of fatty acid signatures ready for analysis. Intended to be the object sig_rep returned by a call to the function prep_sig.
cc	A vector of calibration coefficients, intended to be the object cc returned by the function prep_fa or cc_aug.
near_zero	A small constant used to terminate the algorithm. Default value 0.00001.
min_gamma	Smallest desired value of gamma, potentially used to terminate the algorithm. Default value 0.05.
space	An integer indicator of the estimation space to be used. Default value 1.

Value

A list containing the following elements:

gamma The estimated value of gamma.

gamma_vec A numeric vector containing the value of gamma at each step of the iteration.

prop_vec A numeric vector containing the proportion of all possible two-element signatures with distance exceeding that of the full signatures at each step of the iteration. This value is compared to the argument near_zero.

err_code An integer error code (0 if no error is detected).

err_message A string contains a brief summary of the execution.

Details

The chi-square distance involves a power transformation of signature proportions, with the power parameter being denoted gamma. comp_chi_gamma implements the algorithm of Stewart et al. (2014) to find a suitable value of gamma.

The algorithm is initialized with inv_gamma equal to 1 and gamma is computed as 1/inv_gamma. The distances between all possible pairs of full signatures are computed (distances). For each pair of full signatures, the distances between all possible sub-signatures comprised of only two fatty acid proportions are computed (sub-distances). The proportion of sub-distances that exceed the corresponding distance is computed across all possible pairs of signatures. If that proportion is less than the argument near_zero, the function returns with gamma equal to 1. Otherwise, the function enters an iterative phase. At each iteration, inv_gamma is incremented by 1, gamma is computed as 1/inv_gamma, distances and sub-distances are recomputed, and the proportion of the sub-distances that exceed their corresponding distance is recomputed. The algorithm terminates when the proportion is less than the argument near_zero or the value of gamma is less than min_gamma.

The argument space must equal 1 or 2 (see est_diet). If its value is 1, the calibration coefficients are used to map the signatures to the predator space prior to initializing the algorithm.

As the number of signatures in the library and/or the number of fatty acids in a signature increases, the number of possible pairs of signatures and the number of all possible two-proportion sub-signatures increases rapidly. Consequently, this algorithm may require long run times. However, it only needs to be run once for any particular library of signatures.

References

Stewart, C., S. Iverson, and C. Field. 2014. Testing for a change in diet using fatty acid signatures. *Environmental and Ecological Statistics* 21:775-792.

Examples

```
comp_chi_gamma(sigs = matrix(c(0.05, 0.10, 0.30, 0.55,
                               0.04, 0.11, 0.29, 0.56,
                               0.10, 0.05, 0.35, 0.50,
                               0.12, 0.03, 0.37, 0.48,
                               0.10, 0.06, 0.35, 0.49,
                               0.05, 0.15, 0.35, 0.45), ncol=6),
               cc = c(0.75, 1.00, 1.50, 0.90),
               near_zero = 0.05,
               min_gamma = 0.01,
               space = 1)
```

---

diet_obj_func *Diet estimation objective function*

Description

The utility function diet_obj_func computes the distance between an observed fatty acid signature and a modeled signature computed as a mixture of mean prey signatures.

Usage

```
diet_obj_func(diet, obs_sig, mean_sigs, dist_meas = 1, gamma = 1)
```

Arguments

diet	A numeric vector of diet composition.
obs_sig	A numeric vector containing an observed fatty acid signature.
mean_sigs	A numeric matrix of the mean fatty acid signature for each prey type in the prey library, in column-major format.
dist_meas	An integer indicator of the distance measure to compute. Default value 1.
gamma	The power parameter of the chi-square distance measure. Default value 1.

Value

The distance between the observed and modeled signatures.

Details

This is an internal utility function. Consequently, to increase execution speed, no numeric error checking is performed within `diet_obj_func`. Rather, error checking is presumed to have occurred at a higher level in the calling sequence.

The argument `obs_sig` is presumed to be a fatty acid signature that has been prepared for analysis, which is best accomplished by a call to the function `prep_sig` with the predator data frame. Similarly, the contents of `mean_sig` should be mean signatures computed from signatures that were prepared for analysis by call to the function `prep_sig`.

The argument `diet` is presumed to contain non-negative proportions that sum to 1.0.

The arguments `dist_meas` and `gamma` must be compatible with the function `dist_between_2_sigs`.

Please refer to the vignette and documentation for the functions `prep_sig`, `sig_scale`, and `dist_between_2_sigs` for additional details.

`diet_obj_func` models a predator signature as a mixture of the mean prey-type signatures, with the diet proportions as the mixture proportions, returning the distance between the observed and modeled signatures. The diet composition of a predator is estimated by minimizing this function with respect to the `diet` using the function Rsolnp::solnp.

---

diet_pool *Pool diet estimates to combined prey types*

Description

`diet_pool` pools estimated diets and variance matrices to a smaller number of combined prey types. If `est_diet` is used to estimate predator diet composition using a partitioned prey library (`make_prey_part`), `diet_pool` pools the partitioned results back to the original, unpartitioned prey types.

Usage

```
diet_pool(rep_grp, est_ind, var_ind = NA, est_mean = NA, var_mean = NA)
```

Arguments

`rep_grp`	The post-multiplication matrix returned by a call to `make_prey_part` as the object `pool_post`, or a user-defined matrix for custom pooling. Each column defines a prey type to which estimates should be pooled.
`est_ind`	A numeric matrix of the estimated diet compositions of individual predators using a partitioned prey library, intended to be the object est_ind returned by a call to `est_diet`.

var_ind — A numeric array containing the estimated variance matrix for the estimated diet of each predator, intended to be the object var_ind returned by a call to est_diet. Optional.

est_mean — A numeric matrix containing the estimated mean diet of each predator type, intended to be the object est_mean returned by a call to est_diet. Optional.

var_mean — A numeric array containing the estimated variance matrix for the estimated mean diet of each predator type, intended to be the object var_mean returned by a call to est_diet. Optional.

Value

A list containing the following elements, all of which are organized on the basis of the original, unpartitioned prey types:

est_ind A numeric matrix of the estimated diet compositions of individual predators.

var_ind A numeric array containing the estimated variance matrix for the estimated mean diet of each predator.

est_mean A numeric matrix containing the estimated mean diet of each predator type.

var_mean A numeric array containing the estimated variance matrix for the estimated mean diet of each predator type.

err_code An integer error code (0 if no error is detected).

err_message A string containing a brief summary of the results.

Details

The function dimac explores the prey library for additional structure with identified prey types. If significant structure is found within a library, estimating diet composition on the basis of a partitioned prey library may lead to estimates with less bias and possibly less variation through reduced prey confounding (Bromaghin et al. 2016). The function make_prey_part takes the clustering results returned by dimac and user specification of the number of clusters in which to partition each prey type and returns a partitioned prey library that is ready for use in diet estimation.

However, when estimating diet composition using a partitioned prey library one may still wish to pool partitioned estimates back to the original, unpartitioned prey types for reporting purposes. That is the purpose of the function diet_pool.

NOTE: diet_pool can also be used to pool estimates into a smaller number of combined prey types for reporting purposes. For example, imagine a prey library with a large number of prey types. If subsets of the prey types have similar ecological function, their signatures may share some similarities (prey confounding, Bromaghin et al. 2016). In such a case, one may wish to estimate diet on the basis of the full prey library, but subsequently pool the resulting estimates to a smaller number of combined prey types for reporting purposes (reporting groups, Bromaghin 2008) to reduce the effect of prey confounding. diet_pool can also be used for this purpose, though the user would need to manually construct the reporting group matrix rep_grp.

References

Bromaghin, J.F. 2008. BELS: Backward elimination locus selection for studies of mixture composition or individual assignment. *Molecular Ecology Resources* 8:568-571.

Bromaghin, J.F., S.M. Budge, and G.W. Thiemann. 2016. Should fatty acid signature proportions sum to 1 for diet estimation? *Ecological Research* 31:597-606.

Examples

```
diet_pool(rep_grp = matrix(c(1, 0, 0, 0, 0, 0, 0,
                             0, 1, 0, 0, 0, 0, 0,
                             0, 1, 0, 0, 0, 0, 0,
                             0, 0, 1, 0, 0, 0, 0,
                             0, 0, 0, 1, 0, 0, 0,
                             0, 0, 0, 1, 0, 0, 0,
                             0, 0, 0, 0, 1, 0, 0,
                             0, 0, 0, 0, 0, 1, 0,
                             0, 0, 0, 0, 0, 1, 0,
                             0, 0, 0, 0, 0, 0, 1),
                           nrow = 10, byrow = TRUE),
          est_ind = matrix(c(0.116, 0.315,
                             0.028, 0.073,
                             0.000, 0.000,
                             0.131, 0.120,
                             0.000, 0.000,
                             0.000, 0.000,
                             0.723, 0.452,
                             0.000, 0.000,
                             0.000, 0.000,
                             0.002, 0.040),
                           nrow = 10, byrow = TRUE))
```

`dimac` *Diversive magnetic clustering*

Description

The function `dimac` implements the **di**visive **ma**gnetic **c**lustering algorithm to partition fatty acid signatures into clusters. The DiMaC algorithm was modified from the diana algorithm of the package cluster (Maechler et al. 2016). `dimac` is intended to be called by the user, but only after the fatty acid signatures have been prepared for analysis by calls to the functions `prep_fa` and `prep_sig`. Consequently, error checking of the arguments associated with the signatures (`sigs`, `id`, `type`, and `loc`) is necessarily limited, and calling `dimac` without preceding calls to `prep_fa` and `prep_sig` could return meaningless results. Please see Details or the vignette for additional information.

Usage

```
dimac(sigs, id, type, loc, dist_meas = 1, gamma = 1)
```

Arguments

sigs	A numeric matrix of fatty acid signatures in column-major format.
id	A character vector with a unique sample ID for each signature.
type	A character vector of prey or predator type names.
loc	A numeric matrix specifying the location of signatures within sig for each type.
dist_meas	A integer indicator of the distance measure to use. Default value 1.
gamma	The power parameter of the chi-square distance measure. Default value 1.

Value

A list containing the following elements:

clust A data frame denoting cluster assignments at each iteration of the algorithm.

clust_dist A numeric matrix of the summed distance within clusters at each iteration.

err_code An integer error code(0 if no error is detected).

err_message A string containing a brief summary of the results.

Details

The signatures in sigs are presumed to be ready for analysis, which is best accomplished by a call to the function prep_sig. Please refer to the documentation for prep_sig and/or the vignette for additional details.

The matrix loc provides a mapping of the location of data for each type within sig. It must contain a row for each type and two columns, which contain integers designating the first and last signature of each type within sigs. Such a matrix is returned by the function prep_sig.

Please refer to the documentation for the function dist_between_2_sigs for information regarding permissable values for the arguments dist_meas and gamma.

The DiMaC algorithm is initialized with all signatures in one cluster. The first two magnets are chosen as the two signatures having the greatest distance between them and each non-magnet signature is placed in the cluster associated with the closest magnet. The algorithm then enters an iterative phase. At each iteration, the cluster with the greatest average distance between its signatures and the mean signature is identified as the "active" cluster. The two signatures within the active cluster having the greatest distance between them are selected as new magnets. One of the two new magnets replaces the original magnet for the active cluster and the second starts the formation of an additional cluster. Each non-magnet signature is placed in the cluster associated with the closest magnet, without regard for its cluster designation in the preceding iteration. Consequently, the algorithm is not simply bifurcating, but rather is much more dynamic and flexible. The iterations continue until each signature is in its own cluster.

Unfortunately, there is no objective method to determine the most appropriate number of clusters for each prey or predator type. Our suggestion is to examine the distance results and identify any substantial reductions in distance, which are likely caused by the discovery of structure within that type, that are followed by a more gradual decrease in distance as the number of clusters increases. For diet estimation applications, partitioning a prey library into more clusters than the number of fatty acids used to estimate diet may result in estimates that are not unique. In such a case, estimates

of diet composition need to be pooled into a smaller number of "reporting groups" (e.g., Bromaghin 2008; Meynier et al. 2010).

Utility functions called by `dimac`:

- `dist_pairs_map`
- `dist_sigs_2_mean`

References

Bromaghin, J.F. 2008. BELS: Backward elimination locus selection for studies of mixture composition or individual assignment. *Molecular Ecology Resources* 8:568-571.

Maechler, M., P. Rousseeuw, A. Struyf, M. Hubert, and K. Hornik. 2016. cluster: cluster analysis basics and extensions. R package version 2.0.4.

Meynier, L., P.C.H. morel, B.L. Chilvers, D.D.S. Mackenzie, and P. Duignan. 2010. Quantitative fatty acid signature analysis on New Zealand sea lions: model sensitivity and diet estimates. *Journal of Mammalogy* 91:1484-1495.

Examples

```
dimac(sigs = matrix(c(0.05, 0.10, 0.30, 0.55,
                      0.04, 0.11, 0.29, 0.56,
                      0.10, 0.05, 0.35, 0.50,
                      0.12, 0.03, 0.37, 0.48,
                      0.10, 0.06, 0.35, 0.49,
                      0.05, 0.15, 0.35, 0.45), ncol=6),
      id = c("ID_1", "ID_2", "ID_3", "ID_4", "ID_5", "ID_6"),
      type = c("Type_1", "Type_2", "Type_3"),
      loc = matrix(c(1, 3, 5, 2, 4, 6), ncol=2),
      dist_meas = 1,
      gamma = NA)
dimac(sigs = matrix(c(0.05, 0.10, 0.30, 0.55,
                      0.04, 0.11, 0.29, 0.56,
                      0.10, 0.05, 0.35, 0.50,
                      0.12, 0.03, 0.37, 0.48,
                      0.10, 0.06, 0.35, 0.49,
                      0.05, 0.15, 0.35, 0.45), ncol=6),
      id = c("ID_1", "ID_2", "ID_3", "ID_4", "ID_5", "ID_6"),
      type = c("Type_1", "Type_2", "Type_3"),
      loc = matrix(c(1, 3, 5, 2, 4, 6), ncol=2),
      dist_meas = 2,
      gamma = NA)
dimac(sigs = matrix(c(0.05, 0.10, 0.30, 0.55,
                      0.04, 0.11, 0.29, 0.56,
                      0.10, 0.05, 0.35, 0.50,
                      0.12, 0.03, 0.37, 0.48,
                      0.10, 0.06, 0.35, 0.49,
                      0.05, 0.15, 0.35, 0.45), ncol=6),
      id = c("ID_1", "ID_2", "ID_3", "ID_4", "ID_5", "ID_6"),
      type = c("Type_1", "Type_2", "Type_3"),
      loc = matrix(c(1, 3, 5, 2, 4, 6), ncol=2),
      dist_meas = 3,
```

```
      gamma = 0.5)
dimac(sigs = matrix(c(0.05, 0.10, 0.30, 0.55,
                      0.04, 0.11, 0.29, 0.56,
                      0.10, 0.05, 0.35, 0.50,
                      0.12, 0.03, 0.37, 0.48,
                      0.10, 0.06, 0.35, 0.49,
                      0.05, 0.15, 0.35, 0.45), ncol=6),
      id = c("ID_1", "ID_2", "ID_3", "ID_4", "ID_5", "ID_6"),
      type = c("Type_1", "Type_2", "Type_3"),
      loc = matrix(c(1, 3, 5, 2, 4, 6), ncol=2))
```

`dist_between_2_sigs` *Compute the distance between two fatty acid signatures*

Description

The utility function `dist_between_2_sigs` computes the distance between two fatty acid signatures.

Usage

```
dist_between_2_sigs(sig_1, sig_2, dist_meas = 1, gamma = 1)
```

Arguments

`sig_1, sig_2`	Equal-length numeric vectors of fatty acid signature proportions.
`dist_meas`	An integer indicator of the distance measure to compute. Default value 1.
`gamma`	The power parameter of the chi-square distance measure. See Details. Default value 1.

Value

The selected distance between the two signatures.

Details

This is an internal utility function. The signatures in `sig_1` and `sig_2` are presumed to be ready for analysis, which is best accomplished by a call to the function `prep_sig`. Consequently, to increase execution speed during simulations, no numeric error checking is performed. Please refer to documentation for the function `prep_sig` for additional details.

If the argument `dist_meas` is not one of the following integers, a value of NA is returned:

- dist_meas == 1 yields the Aitchison distance measure (Stewart et al. 2014). This is the default value.
- dist_meas == 2 yields the Kullback-Leibler distance measure of Iverson et al. (2004).
- dist_meas == 3 yields the chi-square distance measure (Stewart et al. 2014).

The argument gamma is only used if dist_meas == 3 and need not be passed otherwise. If dist_meas == 3, gamma must be greater than 0 and cannot exceed 1. If dist_meas == 3 and a value for gamma is not passed, a default value of 1 is used.

References

Iverson, S.J., C. Field, W.D. Bowen, and W. Blanchard. 2004. Quantitative fatty acid signature analysis: A new method of estimating predator diets. *Ecological Monographs* 74:211-235.

Stewart, C., S. Iverson, and C. Field. 2014. Testing for a change in diet using fatty acid signatures. *Environmental and Ecological Statistics* 21:775-792.

---

dist_pairs_map *Creats a map of the distance between pairs of fatty acid signatures*

Description

The utility function dist_pairs_map computes the distance between all possible pairs of fatty acid signatures within each type of prey or predator.

Usage

```
dist_pairs_map(sig_data, dist_meas = 1, gamma = 1)
```

Arguments

sig_data	A numeric matrix of fatty acid signatures in column-major format.
dist_meas	An integer indicator of the distance measure to compute. Default value 1.
gamma	The power parameter of the chi-square distance measure. Default value 1.

Value

A list containing the following elements:

n_sig The number of signatures (columns) in sig_data.

sig_1 The column of sig_data containing one signature.

sig_2 The column of sig_data containing the other signature.

dist The distance between signatures sig_1 and sig_2.

Details

This is an internal utility function. The signature data in sig_data are presumed to be ready for analysis, which is best accomplished by a call to the function prep_sig. Consequently, to increase execution speed during simulations, no numeric error checking is performed. Please refer to the documentation for prep_sig for additional information.

Please refer to the documentation for dist_between_2_sigs for additional information regarding distance measures.

Storing the distances between all possible pairs of fatty acid signatures along with the locations of each pair requires less memory than a square matrix of all possible pairs, while allowing the location of the signatures to be easily determined.

Utility functions called by `dist_sigs_2_mean`:

- `dist_between_2_sigs`

`dist_sigs_2_mean` *Distance between fatty acid signatures and their mean*

Description

The utility function `dist_sigs_2_mean` computes the summed and mean distance between a collection of fatty acid signatures and their mean signature.

Usage

```
dist_sigs_2_mean(sig_data, dist_meas = 1, gamma = 1)
```

Arguments

`sig_data`	A numeric matrix of fatty acid signatures in column-major format.
`dist_meas`	An integer indicator of the distance measure to compute. Default value 1.
`gamma`	The power parameter of the chi-square distance measure. Default value 1.

Value

A list containing the following elements:

dist_sum The summed distance between each signature and the mean signature.

dist_mean The mean distance between each signature and the mean signature.

Details

This is an internal utility function. The signature data in `sig_data` are presumed to be ready for analysis, which is best accomplished by a call to the function `prep_sig`. Consequently, to increase execution speed during simulations, no numeric error checking is performed. Please refer to the documentation for `prep_sig` for additional information.

Please refer to the documentation for `dist_between_2_sigs` for additional information regarding distance measures.

Utility functions called by `dist_sigs_2_mean`:

- `dist_between_2_sigs`

dist_sum_pairwise *Distance between pairs of fatty acid signatures*

Description

The utility function dist_sum_pairwise computes the total distance between all possible pairs of fatty acid signatures.

Usage

```
dist_sum_pairwise(sig_data, dist_meas = 1, gamma = 1)
```

Arguments

sig_data	A numeric matrix of fatty acid signatures in column-major format.
dist_meas	An integer indicator of the distance measure to compute. Default value 1.
gamma	The power parameter of the chi-square distance measure. Default value 1.

Value

The summed distance between all possible pairs of signatures.

Details

dist_sum_pairwise is an internal utility function. The signature data in sig_data are presumed to be ready for analysis, which is best accomplished by a call to the function prep_sig. Consequently, to increase execution speed during simulations, no numeric error checking of the signatures is performed. Please refer to documentation for the function prep_sig for information regarding signature preparation.

Please refer to documentation for the function dist_between_2_sigs for additional information regarding the arguments dist_meas and gamma.

Utility functions called by dist_sigs_2_mean:

- dist_between_2_sigs

est_diet *Estimate predator diet composition*

Description

est_diet estimates the diet of one or more predators.

Usage

```
est_diet(pred_sigs, pred_uniq_types, pred_loc, prey_sigs, prey_uniq_types,
  prey_loc, cc, space = 1, dist_meas = 1, gamma = 1, ind_boot = 100,
  mean_meth = 1, var_meth = 1, mean_boot = 100)
```

Arguments

pred_sigs	A vector or matrix of predator signatures ready for analysis, intended to be the object sig_scale returned by a call to the function prep_sig with the predator data frame.
pred_uniq_types	A character vector of the unique predator types, intended to be the object uniq_types returned by a call to the function prep_sig with the predator data frame.
pred_loc	A vector or matrix giving the first and last locations of the signatures of each predator type within pred_sigs, intended to be the object loc returned by a call to the function prep_sig with the predator data frame.
prey_sigs	A matrix of prey signatures ready for analysis, intended to be the object sig_scale returned by a call to the function prep_sig with the prey data frame or the object sig_part returned by make_prey_part.
prey_uniq_types	A character vector of the unique prey types, intended to be the object uniq_types returned by a call to the function prep_sig with the prey data frame.
prey_loc	A matrix giving the first and last locations of the signatures of each prey type within prey_sigs, intended to be the object loc returned by a call to the function prep_sig with the prey data frame.
cc	A vector of calibration coefficients, intended to be the object cc returned by the function prep_fa.
space	An integer indicator of the estimation space to be used. Default value 1.
dist_meas	An integer indicator of the distance measure to be used. Default value 1.
gamma	The power parameter of the chi-square distance measure. Default value 1.
ind_boot	The number of bootstrap replications to use in the estimation of the variance of an individual predator's diet. Default value 100.
mean_meth	An integer indicator of the estimation method for mean diet. Default value 1.
var_meth	An integer indicator of the estimation method for the variance of mean diet. Default value 1 (bootstrap estimator).
mean_boot	The number of bootstrap replications to use, needed only if the bootstrap method of estimating the variance of meat diet is selected (var_meth == 1). Default value 100.

Value

A list containing the following elements:

pred_sigs A numeric matrix of predator signatures, potentially transformed to the prey space.

prey_sigs A numeric matrix of prey signatures, potentially transformed to the predator space.

mean_sigs A numeric matrix of mean prey-type signatures, potentially transformed to the predator space.

est_ind A numeric matrix of the estimated diet compositions of individual predators.

conv A logical vector indicating whether the optimization function successfully converged.

obj_func A numeric vector of the values of the objective function at each predator's estimated diet.

mod_sigs A numeric matrix of the modeled signature of each predator at its estimated diet.

var_ind A numeric array containing the estimated variance matrix for the estimated mean diet of each predator.

est_mean A numeric matrix containing the estimated mean diet of each predator type.

conv_mean A logical vector indicating whether the estimated mean diet of each predator type is based on at least one diet estimate that converged.

var_mean A numeric array containing the estimated variance matrix for the estimated mean diet of each predator type.

err_code An integer error code (0 if no error is detected).

err_message A string contains a brief summary of the execution.

Details

`est_diet` estimates the diet of one or more predators. It implements a variety of estimation options, and is therefore one of the more complicated functions in `qfasar`. Please read the following information and the Diet Estimation section of the vignette for a description of the options.

The objects passed via the arguments `pred_sig`, `pred_uniq_types`, and `pred_loc` are presumed to be the objects `sig_scale`, `uniq_types`, and `loc`, respectively, returned by a call to the function `prep_sig` with the predator data frame.

The objects passed via the arguments `prey_sig`, `prey_uniq_types`, and `prey_loc` are presumed to be the objects `sig_scale`, `uniq_types`, and `loc`, respectively, returned by a call to the function `prep_sig` with the prey data frame.

The object passed via the argument `cc` is presumed to be the object `cc` returned by a call to the function `prep_fa` with the fatty acid suite data frame.

Bromaghin et al. (2015) introduced the terms **prey space** and **predator space**. These terms refer to the simplexes in which the prey and predator signatures reside. The spaces differ due to predator metabolism of ingested prey tissue and the resulting modification of signature proportions. The calibration coefficients `cc` provide a one-to-one mapping or transformation between the prey and predator spaces. Diet estimation can be performed in either space. Iverson et al. (2004) used calibration coefficients to map predator signatures to the prey space, while Bromaghin et al. (2013) took the opposite approach and mapped prey signatures to the predator space. Simulation work has not revealed any strong reason to prefer one space over the other (Bromaghin et al. 2015). However, be aware that some distance measures will produce different diet estimates in the two spaces. Please see the vignette for more information.

Estimation space options:

- space == 1 Estimation in the predator space, the default value.
- space == 2 Estimation in the prey space.

qfasar implements three distance measures that have been used by QFASA practitioners and researchers: Aitchison, Kullback-Leibler, and chi-square. The argument gamma is a parameter of the chi-square distance measure and its value must be strictly greater than 0 and less than or equal to 1. The distance measure options are:

- dist_meas == 1 yields the Aitchison distance measure (Stewart et al. 2014). This is the default value.
- dist_meas == 2 yields the Kullback-Leibler distance measure of Iverson et al. (2004).
- dist_meas == 3 yields the chi-square distance measure (Stewart et al. 2014).

Please refer to the vignette for additional information about distance measures.

The covariance matrix of each estimated diet can be estimated by bootstrap sampling the prey library. The signatures of each prey type are independently sampled with replacement and the predator diet is estimated with the bootstrapped library. This is replicated ind_boot times and the covariance matrix is estimated from the replicated estimates (Beck et al. 2007, Bromaghin et al. 2015). If you do not wish to estimate variances for the individual diet estimates, pass a bootstrap sample size of 0 via the argument ind_boot.

qfasar implements two methods of estimating the mean diet of each class of predator. The first is the empirical mean of the estimated diets. In the second method, called the **parameterized mean** method, the model is parameterized with a single vector of diet proportions common to all predators and mean diet is estimated by minimizing the distance between the signature modeled from the mean diet proportions and each predator's observed signature, summed over all predators. The parameterized mean method has not yet been thoroughly tested and its inclusion is intended to facilitate future research. Our limited and unpublished work with the parameterized mean estimator suggests it may perform well when predator signatures are homogeneous, but may be more sensitive to the presence of predators with quite different signatures than the empirical estimator. The options for mean_meth are:

- mean_meth == 0 skips estimation of mean diet.
- mean_meth == 1 yields the empirical estimate of mean diet. This is the default value.
- mean_meth == 2 yields the parameterized mean estimate of mean diet.

qfasar implements two methods of estimating the variance of mean diet estimates, the variance estimator of Beck et al. (2007) and a bootstrap estimator, controlled by the argument var_meth. The bootstrap estimator draws independent samples of each prey type to form a bootstrap prey library and a random sample of each predator type, with sample sizes equal to the observed sample sizes. Mean diet is estimated using the method indicated by mean_meth. The argument mean_boot controls the number of times this is repeated, and the replications are used to estimate the covariance matrix for each predator type. Unpublished work suggests that the bootstrap estimator is more reliable. Note that if using the parameterized-mean estimator for mean diet composition, the Beck estimator is not appropriate. The options for var_meth are:

- var_meth == 0 skips variance estimation for mean diets.
- var_meth == 1 yields the bootstrap estimator. This is the default value.
- var_meth == 2 yields the Beck et al. (2007) estimator.

NOTE: The numerical optimization and bootstrap sampling performed by est_diet are numerically intensive and can cause long runs times. Patience is advised! The primary factors causing slow execution are the number of predator signatures, the number of predator and prey types, and bootstrap sample sizes.

References

Beck, C.A., S.J. Iverson, W.D. Bowen, and W. Blanchard. 2007. Sex differences in grey seal diet reflect seasonal variation in foraging behaviour and reproductive espenditure: evidence from quantitative fatty acid signature analysis. *Journal of Animal Ecology* 76:490-502.

Bromaghin, J.F., M.M. Lance, E.W. Elliott, S.J. Jeffries, A. Acevedo-Gutierrez, and J.M. Kennish. 2013. New insights into the diets of harbor seals (*Phoca vitulina*) in the Salish Sea revealed by analysis of fatty acid signatures. *Fishery Bulletin* 111:13-26.

Bromaghin, J.F., K.D. Rode, S.M. Budge, and G.W. Thiemann. 2015. Distance measures and optimization spaces in quantitative fatty acid signature analysis. *Ecology and Evolution* 5:1249-1262.

Iverson, S.J., C. Field, W.D. Bowen, and W. Blanchard. 2004. Quantitative fatty acid signature analysis: A new method of estimating predator diets. *Ecological Monographs* 74:211-235.

Stewart, C., S. Iverson, and C. Field. 2014. Testing for a change in diet using fatty acid signatures. *Environmental and Ecological Statistics* 21:775-792.

Examples

```
est_diet(pred_sigs = matrix(c(0.05, 0.10, 0.30, 0.55,
                              0.04, 0.11, 0.29, 0.56,
                              0.10, 0.06, 0.35, 0.49,
                              0.05, 0.15, 0.35, 0.45), ncol=4),
         pred_uniq_types = c("Pred_1", "Pred_2"),
         pred_loc = matrix(c(1, 3, 2, 4), ncol=2),
         prey_sigs = matrix(c(0.06, 0.09, 0.31, 0.54,
                              0.05, 0.09, 0.30, 0.56,
                              0.03, 0.10, 0.30, 0.57,
                              0.08, 0.07, 0.30, 0.55,
                              0.09, 0.05, 0.33, 0.53,
                              0.09, 0.06, 0.34, 0.51,
                              0.09, 0.07, 0.34, 0.50,
                              0.08, 0.11, 0.35, 0.46,
                              0.06, 0.14, 0.36, 0.44), ncol=9),
         prey_uniq_types = c("Prey_1", "Prey_2", "Prey_3"),
         prey_loc = matrix(c(1, 4, 7, 3, 6, 9), ncol=2),
         cc = c(0.75, 1.00, 1.50, 1.15),
         space = 1, dist_meas = 1, ind_boot = 2,
         mean_meth = 0)
```

---

find_boot_ss *Find realistic bootstrap sample sizes*

Description

In QFASA simulation studies, predator signatures are often simulated by bootstrap resampling prey signature data. find_boot_ss finds bootstrap sample sizes for each prey type that produce simulated predator signatures with realistic levels of variation.

Usage

```
find_boot_ss(pred_sigs, pred_diets, prey_sigs, prey_loc, n_pred_boot = 1000)
```

Arguments

pred_sigs	A vector or matrix of predator signatures, intended to be the object pred_sigs returned by a call to the function est_diet.
pred_diets	A numeric matrix of the estimated diet compositions of individual predators, intended to be the object est_ind returned by a call to est_diet.
prey_sigs	A matrix of prey signatures, intended to be the object prey_sigs returned by a call to the function est_diet.
prey_loc	A matrix giving the first and last locations of the signatures of each prey type within prey_sigs, intended to be the object loc returned by a call to the function prep_sig with the prey data frame, or by a call to the function make_prey_part if a partitioned prey library was used for diet estimation.
n_pred_boot	An integer designating the number of predator signatures to bootstrap. See Details. Default value 1000.

Value

A list containing the following elements:

var_diet A numeric vector of the variance between the estimated diets of all possible pairs of predators, sorted in increasing order.

var_sig A numeric vector of the variance between the signatures of all possible pairs of predators, sorted consistently with var_diet.

var_sig_smooth A loess-smoothed version of var_sig.

mod_sig_var A numeric vector of the modeled variance between bootstrapped predator signatures at each iteration of the algorithm.

var_target The target level of variance between bootstrapped predator signatures.

boot_ss An integer vector of bootstrap sample sizes for each prey type.

err_code An integer error code (0 if no error is detected).

err_message A string containing a brief summary of the results.

Details

QFASA simulation studies may require the generation of predator signatures given a specified diet, against which estimates of diet composition can then be compared (e.g., Bromaghin et al. 2015). Given a specified diet, a bootstrap sample of each prey type is drawn and mean prey-type signatures are computed. A predator signature is then generated by multiplying the mean bootstrapped prey signatures by the diet proportions.

Although authors often fail to report the bootstrap sample sizes used for each prey type when describing simulations (e.g., Haynes et al. 2015; Thiemann et al. 2008; Wang et al. 2010), they are subjectively selected (e.g., Iverson et al. 2004; Bromaghin et al. 2015). However, Bromaghin et al. (2016) found that bootstrap sample sizes strongly influence both the bias and the variance of diet

composition estimates in simulation studies. Consequently, the selection of bootstrap sample sizes is an important aspect of simulation design.

Bromaghin (2015) presented an objective method of establishing a bootstrap sample size for each prey type that produces simulated predator signatures having a realistic level of between-signature variation. The function `find_boot_ss` implements the algorithm of Bromaghin (2015). A brief summary of the algorithm, sufficient to understand the objects returned by `find_boot_ss`, follows below. Please refer to Bromaghin (2015) for additional details.

The concept underlying the algorithm is that the variation in predator signatures can be partitioned into variation due to differences in diet and variation due to prey animals consumed given diet. Consequently, a realistic level of variation between signatures for predators sharing the same diet is approximated from the empirical relationship between a measure of variation between pairs of predator signatures `var_sig` and a measure of variation between the estimated diets `var_diets` of the same predator pairs. As the variance in diets approaches zero, the predators are effectively eating the same diet and variation in their signatures therefore approaches the level of variation due to prey selection only. `find_boot_ss` models the relationship between `var_diet` and `var_sig` using a loess smooth, and the modeled signature variance `var_sig_smooth` for the pair of predators whose value of `var_diet` is smallest is taken as the target level of variation for predator signatures.

The algorithm is initialized with a sample size of 1 from each prey type. A sample of `n_pred_boot` predator signatures is generated using those sample sizes and the measure of variance among the signatures is computed. If the variance measure exceeds the target level, the prey type contributing most to the variance measure is identified and its sample size is increased by 1. The algorithm then iterates, increasing the sample size by one at each iteration, until the measure of variation is less than the target level. The level of variation at each iteration and the target level of variation are returned in the objects `mod_sig_var` and `var_terget`, respectively.

The argument `n_boot_pred` should be large enough to return an estimate of the variance measure that itself has low variance, so that the algorithm returns numerically stable results. We suspect that the default value of 1000 errs on the side of caution.

NOTE: Because `find_boot_ss` is intended to operate on the predator and prey signatures returned by a call to the function `est_diet`, `find_boot_ss` can be based on diet estimates obtained in either the predator or prey space, using an original or partitioned prey library. However, it is imperative that the arguments are compatible.

References

Bromaghin, J.F. 2015. Simulating realistic predator signatures in quantitative fatty acid signature analysis. *Ecological Informatics* 30:68-71.

Bromaghin, J.F., S.M. Budge, and G.W. Thiemann. 2016. Should fatty acid signature proportions sum to 1 for diet estimation? *Ecological Research* 31:597-606.

Bromaghin, J.F., K.D. Rode, S.M. Budge, and G.W. Thiemann. 2015. Distance measures and optimization spaces in quantitative fatty acid signature analysis. *Ecology and Evolution* 5:1249-1262.

Haynes, T.B., J.A. Schmutz, J.F. Bromaghin, S.J. Iverson, V.M. Padula, and A.E. Rosenberger. 2015. Diet of yellow-billed loons (*Gavia adamsii*) in Arctic lakes during the nesting season inferred from fatty acid analysis. *Polar Biology* 38:1239-1247.

Iverson, S.J., C. Field, W.D. Bowen, and W. Blanchard. 2004. Quantitative fatty acid signature analysis: A new method of estimating predator diets. *Ecological Monographs* 74:211-235.

Thiemann, G.W., S.J. Iverson, and I. Stirling. 2008. Polar bear diets and Arctic marine food webs: insights from fatty acid analysis. *Ecological Monographs* 78:591-613.

Wang, S.W., T.E. Hollmen, and S.J. Iverson. 2010. Validating quantitative fatty acid signature analysis to estimate diets of spectacled and Stellers eiders (*Somateria fischeri* and *Polysticta stelleri*). *Journal of Comparative Physiology B* 180:125-139.

Examples

```
find_boot_ss(pred_sigs = matrix(c(0.05, 0.10, 0.30, 0.55,
                                  0.04, 0.11, 0.29, 0.56,
                                  0.10, 0.05, 0.35, 0.50,
                                  0.12, 0.03, 0.37, 0.48,
                                  0.10, 0.06, 0.35, 0.49,
                                  0.05, 0.15, 0.35, 0.45), ncol = 6),
             pred_diets = matrix(c(0.33, 0.34, 0.33,
                                   0.10, 0.80, 0.10,
                                   0.35, 0.50, 0.15,
                                   0.20, 0.35, 0.45,
                                   0.20, 0.45, 0.35,
                                   0.15, 0.65, 0.20), ncol = 6),
             prey_sigs = matrix(c(0.06, 0.09, 0.31, 0.54,
                                  0.05, 0.09, 0.30, 0.56,
                                  0.03, 0.10, 0.30, 0.57,
                                  0.08, 0.07, 0.30, 0.55,
                                  0.09, 0.05, 0.33, 0.53,
                                  0.09, 0.06, 0.34, 0.51,
                                  0.09, 0.07, 0.34, 0.50,
                                  0.08, 0.11, 0.35, 0.46,
                                  0.06, 0.14, 0.36, 0.44), ncol = 9),
             prey_loc = matrix(c(1, 4, 7, 3, 6, 9), ncol=2),
             n_pred_boot = 500)

find_boot_ss(pred_sigs = matrix(c(0.05, 0.10, 0.30, 0.55,
                                  0.04, 0.11, 0.29, 0.56,
                                  0.10, 0.05, 0.35, 0.50,
                                  0.12, 0.03, 0.37, 0.48,
                                  0.10, 0.06, 0.35, 0.49,
                                  0.05, 0.15, 0.35, 0.45), ncol = 6),
             pred_diets = matrix(c(0.33, 0.34, 0.33,
                                   0.10, 0.80, 0.10,
                                   0.35, 0.50, 0.15,
                                   0.20, 0.35, 0.45,
                                   0.20, 0.45, 0.35,
                                   0.15, 0.65, 0.20), ncol = 6),
             prey_sigs = matrix(c(0.06, 0.09, 0.31, 0.54,
                                  0.05, 0.09, 0.30, 0.56,
                                  0.03, 0.10, 0.30, 0.57,
                                  0.08, 0.07, 0.30, 0.55,
                                  0.09, 0.05, 0.33, 0.53,
                                  0.09, 0.06, 0.34, 0.51,
                                  0.09, 0.07, 0.34, 0.50,
                                  0.08, 0.11, 0.35, 0.46,
```

```
                          0.06, 0.14, 0.36, 0.44), ncol = 9),
prey_loc = matrix(c(1, 4, 7, 3, 6, 9), ncol=2))
```

---

gof *Goodness-of-fit for modeled predator signatures*

Description

The function gof uses estimated diet compositions and bootstrap resampling of the prey library to construct a statistic that may conservatively indicate predator fatty acid signatures that were not accurately modeled during diet estimation.

Usage

```
gof(prey_sigs, prey_loc, mean_sigs, diet_est, conv, obj_func, dist_meas = 1,
  gamma = 1, boot_gof = 500)
```

Arguments

prey_sigs	A matrix of prey signatures in the optimization space used for diet estimation. Intended to be the object prey_sigs returned by the function est_diet.
prey_loc	A matrix giving the first and last locations of the signatures of each prey type within prey_sigs. Intended to be the object loc returned by the function prep_sig if diets were estimated using an unpartitioned prey library or make_prey_part if diets were estimated using a partitioned library.
mean_sigs	A numeric matrix of mean prey-type signatures in the optimization space used for diet estimation. Intended to be the object mean_sigs returned by the function est_diet.
diet_est	A numeric matrix of estimated diet compositions. Intended to be the object est_ind returned by the function est_diet.
conv	A logical vector indicating whether the optimization function successfully converged during diet estimation. Intended to be the object conv returned by the function est_diet.
obj_func	A numeric vector of the value of the minimized objective function for each predator. Intended to be the object obj_func returned by the function est_diet.
dist_meas	An integer indicator of the distance measure used for diet estimation. Default value 1.
gamma	The power parameter of the chi-square distance measure. Default value 1.
boot_gof	The number of bootstrap replications to use. Default value 500.

Value

A list containing the following elements:

gof_ss The number of diet estimates that converged for each predator, therefore producing a simulated value of the objective function.

p_val The proportion of the simulated objective function values that exceeded the value produced during diet estimation.

err_code An integer error code (0 if no error is detected).

err_message A string contains a brief summary of the execution.

Details

Diet estimation involves modeling an observed predator fatty acid signature as a mixture of prey signatures. However, methods to assess how well predator signatures are modeled have received little attention in the literature (but see Bromaghin et al. 2015).

One byproduct of diet estimation is the value of the distance measure that is minimized during diet estimation (`est_diet`), called the objective function. If a predator signature is accurately modeled, the value of the objective function will be relatively small. Conversely, the more poorly the signature is approximated, the larger the objective function will tend to be. However, what value of the objective function to use as a warning flag for a potentially poor fit is not clear.

The function `gof` represents one attempt to answer this question. The algorithm is based on the following logic. First, we assume that a predator consumes the mixture of prey specified by its estimated diet composition. Given that assumption, the expected value of the objective function is, in a sense, fixed (Bromaghin 2015). Large values of the objective function are then most likely to occur when variation in a predator signature, which results from the selection of individual prey within prey types, is maximized. Within the framework of simulating predator signatures, variation in the signatures is maximized when the bootstrap sample sizes of the prey signatures used to construct a predator signature are minimized (Bromaghin et al. 2016).

Implementing the above logic, `gof` randomly samples a single prey signature from each prey type and weights the resulting signatures with a predator's estimated diet composition to construct a modeled signature. The modeled signature is then used to estimate diet. If the optimization function converges, the value of the objective function obtained with the modeled signature is compared to the value of the objective function obtained while estimating diet with the observed signature (argument `obj_func`. This is repeated `boot_gof` times and the proportion of the simulated objective function values that exceed the observed objective function value is computed. `gof` therefore constructs a statistic similar to a p-value, with small values being suggestive of a predator signature that was not closely approximated during diet estimation.

NOTE: the method implemented in `gof` is at this point only an idea whose performance has not been explored. It has been included in `qfasar` to support future research on this topic.

References

Bromaghin, J.F. 2015. Simulating realistic predator signatures in quantitative fatty acid signature analysis. Ecological Informatics 30:68-71.

Bromaghin, J.F., S.M. Budge, and G.W. Thiemann. 2016. Should fatty acid signature proportions sum to 1 for diet estimation? Ecological Research 31:597-606.

Bromaghin, J.F., K.D. Rode, S.M. Budge, and G.W. Thiemann. 2015. Distance measures and optimization spaces in quantitative fatty acid signature analysis. *Ecology and Evolution* 5:1249-1262.

Examples

```
gof(prey_sigs = matrix(c(0.06, 0.09, 0.31, 0.54,
                         0.05, 0.09, 0.30, 0.56,
                         0.03, 0.10, 0.30, 0.57,
                         0.08, 0.07, 0.30, 0.55,
                         0.09, 0.05, 0.33, 0.53,
                         0.09, 0.06, 0.34, 0.51,
                         0.09, 0.07, 0.34, 0.50,
                         0.08, 0.11, 0.35, 0.46,
                         0.06, 0.14, 0.36, 0.44), ncol = 9),
    prey_loc = matrix(c(1, 4, 7, 3, 6, 9), ncol=2),
    mean_sigs = matrix(c(0.047, 0.093, 0.303, 0.557,
                         0.087, 0.050, 0.323, 0.530,
                         0.077, 0.106, 0.350, 0.467), ncol = 3),
    diet_est = matrix(c(0.394, 0.356, 0.250,
                        0.336, 0.365, 0.299), ncol = 2),
    conv = c(TRUE, TRUE),
    obj_func = c(1.13, 2.24),
    dist_meas = 1,
    boot_gof = 10)
```

---

lopo *Leave-one-prey-out analysis*

Description

The function lopo evaluates the distinctiveness of a prey library by performing a leave-one-prey-out analysis.

Usage

```
lopo(sigs, type, uniq_types, type_ss, loc, dist_meas = 1, gamma = 1)
```

Arguments

sigs	A numeric matrix of fatty acid signatures in column-major format.
type	A character vector of prey or predator type names.
uniq_types	A character vector of the unique types, sorted alphanumerically.
type_ss	The number of signatures (sample size) for each unique type.
loc	A numeric matrix specifying the location of signatures within sigs for each uniq_types.

dist_meas	A integer indicator of the distance measure to use. Default value 1.
gamma	The power parameter of the chi-square distance measure. Default value 1.

Value

A list containing the following elements:

est A square matrix containing the mean distribution of estimates among all prey types, by prey-type.

mean_correct The mean proportion correctly estimated across prey types, unweighted by prey-type sample sizes.

total_correct The proportion of all signatures correctly estimated.

n_conv An integer vector containing the number of estimates that converged.

err_code An integer error code (0 if no error is detected).

err_message A string containing a brief summary of the results.

Details

The object passed as the argument sigs is intended to be the signature object returned by sig_scale or, if the prey library has been partitioned, by make_prey_part.

The objects passed as the arguments type, uniq_types, type_ss, and loc are intended to be the corresponding objects returned by prep_sig or, if the prey library has been partitioned, by make_prey_part.

The arguments dist_meas and gamma must be compatible with the function dist_between_2_sigs.

lopo performs a leave-one-prey-out analysis with a prey library and defined prey types (Bromaghin et al. 2016b). Each signature is temporarily removed from the library, the mean prey-type signature is recomputed, and the "diet" of the removed signature is estimated, after which the removed signature is returned to the library. This is done for each signature in turn. The mean estimate for each prey type is returned as a row of est. Perfect estimation would result in the square matrix est having 1.0 along its diagonal and 0.0 in all off-diagonal positions. Large off-diagonal elements are indicative of confounding, or similarity, between the corresponding prey types. The returned object mean_correct is the mean of the diagonal elements of est, while the returned object total_correct is the mean computed over all signatures in the prey library.

Note: the statistics are computed based on the estimates that successfully converge (n_conv) and prey types that only have a sample size of 1 are skipped.

Because of the numerical optimization involved in a leave-one-prey-out analysis, lopo can take a few minutes to run with a large prey library.

The statistics computed by lopo are one measure of the distinctiveness of prey types within a prey library. However, it is important to be aware that such statistics are not necessarily informative of the ability of QFASA to accurately estimate predator diets, as Bromaghin et al. (2015, 2016a, 2016b) found that QFASA performance depends strongly on the interaction between characteristics of a prey library, the specific diet of a predator, and the accuracy of the calibration coefficients. Consequently, the user is warned not to misinterpret or misrepresent these statistics.

References

Bromaghin, J.F., S.M. Budge, and G.W. Thiemann. 2016b. Should fatty acid signature proportions sum to 1 for diet estimation? *Ecological Research* 31:597-606.

Bromaghin, J.F., S.M. Budge, G.W. Thiemann, and K.D. Rode. 2016b. Assessing the robustness of quantitative fatty acid signature analysis to assumption violations. *Methods in Ecology and Evolution* 7:51-59.

Bromaghin, J.F., K.D. Rode, S.M. Budge, and G.W. Thiemann. 2015. Distance measures and optimization spaces in quantitative fatty acid signature analysis. *Ecology and Evolution* 5:1249-1262.

Examples

```
lopo(sigs = matrix(c(0.05, 0.10, 0.30, 0.55,
                     0.04, 0.11, 0.29, 0.56,
                     0.10, 0.05, 0.35, 0.50,
                     0.12, 0.03, 0.37, 0.48,
                     0.10, 0.06, 0.35, 0.49,
                     0.05, 0.15, 0.35, 0.45), ncol=6),
     type = c("Type_1", "Type_1", "Type_2", "Type_2", "Type_3", "Type_3"),
     uniq_types = c("Type_1", "Type_2", "Type_3"),
     type_ss <- c(2, 2, 2),
     loc = matrix(c(1, 3, 5, 2, 4, 6), ncol=2),
     dist_meas = 1)

lopo(sigs = matrix(c(0.05, 0.10, 0.30, 0.55,
                     0.04, 0.11, 0.29, 0.56,
                     0.10, 0.05, 0.35, 0.50,
                     0.12, 0.03, 0.37, 0.48,
                     0.10, 0.06, 0.35, 0.49,
                     0.05, 0.15, 0.35, 0.45), ncol=6),
     type = c("Type_1", "Type_1", "Type_2", "Type_2", "Type_3", "Type_3"),
     uniq_types = c("Type_1", "Type_2", "Type_3"),
     type_ss <- c(2, 2, 2),
     loc = matrix(c(1, 3, 5, 2, 4, 6), ncol=2),
     dist_meas = 2)

lopo(sigs = matrix(c(0.05, 0.10, 0.30, 0.55,
                     0.04, 0.11, 0.29, 0.56,
                     0.10, 0.05, 0.35, 0.50,
                     0.12, 0.03, 0.37, 0.48,
                     0.10, 0.06, 0.35, 0.49,
                     0.05, 0.15, 0.35, 0.45), ncol=6),
     type = c("Type_1", "Type_1", "Type_2", "Type_2", "Type_3", "Type_3"),
     uniq_types = c("Type_1", "Type_2", "Type_3"),
     type_ss <- c(2, 2, 2),
     loc = matrix(c(1, 3, 5, 2, 4, 6), ncol=2),
     dist_meas = 3,
     gamma = 0.25)

lopo(sigs = matrix(c(0.05, 0.10, 0.30, 0.55,
                     0.04, 0.11, 0.29, 0.56,
```

```
                    0.10, 0.05, 0.35, 0.50,
                    0.12, 0.03, 0.37, 0.48,
                    0.10, 0.06, 0.35, 0.49,
                    0.05, 0.15, 0.35, 0.45), ncol=6),
     type = c("Type_1", "Type_1", "Type_2", "Type_2", "Type_3", "Type_3"),
     uniq_types = c("Type_1", "Type_2", "Type_3"),
     type_ss <- c(2, 2, 2),
     loc = matrix(c(1, 3, 5, 2, 4, 6), ncol=2),
     dist_meas = 3)

lopo(sigs = matrix(c(0.05, 0.10, 0.30, 0.55,
                    0.04, 0.11, 0.29, 0.56,
                    0.10, 0.05, 0.35, 0.50,
                    0.12, 0.03, 0.37, 0.48,
                    0.10, 0.06, 0.35, 0.49,
                    0.05, 0.15, 0.35, 0.45), ncol=6),
     type = c("Type_1", "Type_1", "Type_2", "Type_2", "Type_3", "Type_3"),
     uniq_types = c("Type_1", "Type_2", "Type_3"),
     type_ss <- c(2, 2, 2),
     loc = matrix(c(1, 3, 5, 2, 4, 6), ncol=2))
```

---

lopo_pool *Pool lopo results to original prey types*

Description

If lopo is used to perform a leave-one-prey-out analysis with a partitioned prey library (make_prey_part), lopo_pool pools the partitioned results back to the original unpartitioned prey types.

Usage

```
lopo_pool(est, n_conv, type_ss, pre, post)
```

Arguments

est	The estimation matrix of a leave-one-prey-out analysis performed by the function lopo, returned as the est object.
n_conv	An integer vector denoting the number of signature estimates in the partitioned prey types that converged, returned by a call to lopo as the n_conv object.
type_ss	An integer vector with the number of signatures (sample size) in each of the partitioned prey types, returned by a call to make_prey_part as the type_ss object.
pre	The pre-multiplication matrix returned by a call to make_prey_part as the pool_pre object.
post	The post-multiplication matrix returned by a call to make_prey_part as the pool_post object.

Value

A list containing the following elements, all of which are organized on the basis of the original unpartitioned prey types:

est A square matrix containing the mean distribution of leave-one-prey-out estimates among all prey types.

mean_correct The mean proportion correctly estimated across prey types, unweighted by prey-type sample sizes.

total_correct The proportion of all signatures correctly estimated.

n_conv An integer vector containing the number of estimates that converged.

err_code An integer error code (0 if no error is detected).

err_message A string containing a brief summary of the results.

Details

The statistics computed by `lopo` and `lopo_pool` are one measure of the distinctiveness of prey types within a prey library. However, it is important to be aware that such statistics are not necessarily informative of the ability of QFASA to accurately estimate predator diets, as Bromaghin et al. (2015, 2016a, 2016b) found that QFASA performance depends strongly on the interaction between characteristics of a prey library, the specific diet of a predator, and the accuracy of the calibration coefficients. Consequently, the user is warned not to misinterpret or misrepresent these statistics.

References

Bromaghin, J.F., S.M. Budge, and G.W. Thiemann. 2016b. Should fatty acid signature proportions sum to 1 for diet estimation? *Ecological Research* 31:597-606.

Bromaghin, J.F., S.M. Budge, G.W. Thiemann, and K.D. Rode. 2016a. Assessing the robustness of quantitative fatty acid signature analysis to assumption violations. *Methods in Ecology and Evolution* 7:51-59.

Bromaghin, J.F., K.D. Rode, S.M. Budge, and G.W. Thiemann. 2015. Distance measures and optimization spaces in quantitative fatty acid signature analysis. *Ecology and Evolution* 5:1249-1262.

Examples

```
lopo_pool(est = matrix(c(0.90, 0.05, 0.30, 0.02,
                         0.04, 0.84, 0.09, 0.03,
                         0.02, 0.06, 0.35, 0.57,
                         0.05, 0.10, 0.15, 0.70), nrow = 4, byrow = TRUE),
          n_conv = c(2, 8, 8, 11),
          type_ss = c(2, 8, 8, 12),
          pre = matrix(c(0.2, 0.8, 0.0, 0.0,
                         0.0, 0.0, 0.4, 0.6), nrow = 2, byrow = TRUE),
          post = matrix(c(1, 1, 0, 0,
                          0, 0, 1, 1), ncol = 2)
          )
```

make_diet_grid *Generate a regular grid of diet compositions*

Description

The function make_diet_grid generates a systematic grid of regularly-spaced diet compositions with a user-specified resolution.

Usage

```
make_diet_grid(uniq_types, inv_inc)
```

Arguments

uniq_types	A factor of unique prey-type names.
inv_inc	The integer inverse of the resolution between consecutive diet compositions.

Value

A list containing the following elements:

diet_grid A numeric matrix of grid diet compositions, in column-major format.

err_code An integer error code (0 if no error is detected).

err_message A string contains a brief summary of the execution.

Details

The function make_diet_grid generates a systematic grid of regularly-spaced diet compositions throughout the space of all possible diets for a given number of prey types. Such a diet composition grid may be useful in some simulation studies of estimator performance (e.g., Bromaghin et al. 2016). Given a diet composition, predator fatty acid signatures can be generated using the function make_pred_sigs. The diets of such simulated predators can then be estimated, and the resulting estimates can be compared to the known diet composition to evaluate bias, variance, and perhaps other properties.

The algorithm starts with a diet proportion of 1.0 assigned to the first prey type, and therefore 0.0 for the other prey types. The algorithm then begins an iterative loop in which an increment of diet proportion is repeatedly shifted to the other prey types, stopping when the last prey type has a diet proportion of 1.0. The user controls the resolution of the grid by specifying the integer inverse of the desired diet increment. For example, an inverse increment of 10 would produce diet compositions with proportions shifted by an increment of 0.1. See Bromaghin et al. (2016) for a small example with three prey types and a diet increment of 0.25. However, note that unlike Bromaghin et al. (2016), make_diet_grid retains diet compositions comprised of a single prey type.

It is critical that the prey-type names match those in the prey library. The easiest way to ensure this happens is to pass the object uniq_types returned a call to the function prep_sig as the uniq_types argument. Alternatively, and more risky, a vector of unique prey names can be created using the

concatenate function and cast as a factor, i.e., uniq_types <- as.factor(c("Prey_1", "Prey_2", ..., "Prey_P)).

NOTE: The number of possible diets grows quickly as the number of prey types increases and the diet increment decreases, and may exceed memory limits.

References

Bromaghin, J.F., S.M. Budge, and G.W. Thiemann. 2016. Should fatty acid signature proportions sum to 1 for diet estimation? Ecological Research 31:597-606.

Examples

```
make_diet_grid(uniq_types = as.factor(c("Bearded",
                                        "Beluga",
                                        "Bowhead",
                                        "Ribbon",
                                        "Ringed",
                                        "Spotted",
                                        "Walrus")),
               inv_inc = 10)
```

---

make_diet_rand *Generate random diet compositions*

Description

The function make_diet_rand generates a user-specified number of random diet compositions.

Usage

```
make_diet_rand(uniq_types, n_diet)
```

Arguments

uniq_types	A factor of unique prey-type names.
n_diet	The integer number of diet compositions to generate.

Value

A list containing the following elements:

diet_rand A numeric matrix of random diet compositions, in column-major format.

err_code An integer error code (0 if no error is detected).

err_message A string contains a brief summary of the execution.

Details

The function make_diet_rand generates a specified number of random diet compositions to support simulation-based research of the performance of QFASA diet estimation procedures. Given a diet composition, predator fatty acid signatures can be generated using the function make_pred_sigs. The diets of such simulated predators can then be estimated, and the diet estimates can be compared to the known diet composition to evaluate bias, variance, and perhaps other properties.

The algorithm starts by generating a uniformly distributed random number between 0 and 1 as the diet proportion for the first prey type. The algorithm then considers each additional prey type in turn, generating a uniform random number between zero and 1 minus the sum of the proportions assigned to the preceding prey types. The diet proportion for the last prey type is 1 minus the sum of the other diet proportions. As a hedge against limitations in the random number generator, the proportions are then randomly ordered among prey types.

It is critical that the prey-type names match those in the prey library. The easiest way to ensure this happens is to pass the object uniq_types returned a call to the function prep_sig as the uniq_types argument. Alternatively, and more risky, a vector of unique prey names can be created using the concatenate function and cast as a factor, i.e., uniq_types <- as.factor(c("Prey_1", "Prey_2", ..., "Prey_P)).

Examples

```
make_diet_rand(uniq_types = as.factor(c("Bearded",
                                        "Beluga",
                                        "Bowhead",
                                        "Ribbon",
                                        "Ringed",
                                        "Spotted",
                                        "Walrus")),
               n_diet = 100)
```

---

make_ghost *Make a ghost prey signature*

Description

Bromaghin et al (2016) studied the performance of QFASA estimators when predators consumed a prey type that was not represented in the prey library, termed a ghost prey. make_ghost constructs a signature for a ghost prey type.

Usage

```
make_ghost(prey_sigs, loc, ghost_err = 0.25, dist_meas = 1, gamma = 1)
```

Arguments

prey_sigs	A matrix of prey signatures ready for analysis, intended to be the object sig_scale returned by a call to the function prep_sig with the prey data frame or the object sig_part returned by make_prey_part.
loc	A matrix giving the first and last locations of the signatures of each prey type within prey_sigs, intended to be the object loc returned by a call to the function prep_sig with the prey data frame or the object loc returned by make_prey_part.
ghost_err	A proportion strictly greater than 0 and less than 1 used to control the lower and upper bounds of ghost prey signature proportions. Default value 0.25.
dist_meas	An integer indicator of the distance measure to be used. Default value 1.
gamma	The power parameter of the chi-square distance measure. Default value 1.

Value

A list containing the following elements:

sig A numeric vector containing the ghost prey signature.

dist Summed distance between the ghost signature and the mean prey signatures.

err_code An integer error code (0 if no error is detected).

err_message A string contains a brief summary of the execution.

Details

One of the major assumptions of QFASA is that the prey library contains representatives of all prey types consumed by a predator. Bromaghin et al. (2016) investigated the robustness of diet estimators to violations of this assumption. The function make_ghost constructs a ghost prey signature using the methods of Bromaghin et al. (2016).

The ghost prey signature is constructed by maximizing the summed distance between the ghost prey signature and the mean prey signatures, while constraining the ghost signature proportions within reasonable bounds to ensure that the signature is somewhat realistic for the prey library. The definition of reasonable bounds is embodied in the argument ghost_err. ghost_err is a proportion greater than or equal to zero and less than 1 that is used to construct lower and upper bounds of the signature proportions. The lower bound is obtained by multiplying 1 - ghost_err by the minimum mean prey proportion for each fatty acid. Similarly, the upper bound is obtained by multiplying 1 + ghost_err by the maximum mean prey proportion for each fatty acid. The ghost prey signature is then obtained by maximizing the summed distance between the signature and the mean prey signatures, constraining the signature to lie within the bounds and sum to 1. See est_diet for information regarding distance measures.

This method ensures that the ghost prey signature is somewhat distinct from the other prey types, but not so wildly different that it represents a completely different pattern from the other prey types. Although research into suitable values for ghost_err has not been conducted, it is probably advisable to use small to moderate values. Bromaghin et al. (2016) used a value of 0.25. As the value of ghost_err is increased, the resulting signature will tend to become increasing different from any prey type in the library.

References

Bromaghin, J.F., S.M. Budge, G.W. Thiemann, and K.D. Rode. 2016. Assessing the robustness of quantitative fatty acid signature analysis to assumption violations. *Methods in Ecology and Evolution* 7:51-59.

Examples

```
make_ghost(prey_sigs = matrix(c(0.05, 0.10, 0.30, 0.55,
                                0.04, 0.11, 0.29, 0.56,
                                0.10, 0.05, 0.35, 0.50,
                                0.12, 0.03, 0.37, 0.48,
                                0.10, 0.06, 0.35, 0.49,
                                0.05, 0.15, 0.35, 0.45), ncol=6),
           loc = matrix(c(1, 3, 5, 2, 4, 6), ncol=2),
           ghost_err = 0.15,
           dist_meas = 1,
           gamma = NA)

make_ghost(prey_sigs = matrix(c(0.05, 0.10, 0.30, 0.55,
                                0.04, 0.11, 0.29, 0.56,
                                0.10, 0.05, 0.35, 0.50,
                                0.12, 0.03, 0.37, 0.48,
                                0.10, 0.06, 0.35, 0.49,
                                0.05, 0.15, 0.35, 0.45), ncol=6),
           loc = matrix(c(1, 3, 5, 2, 4, 6), ncol=2))
```

---

make_pred_sigs *Simulate predator signatures*

Description

`make_pred_sigs` generates predator signatures based on a specified predator diet composition and bootstrap sampling signatures from a prey library.

Usage

```
make_pred_sigs(prey_sigs, prey_loc, cc, diet, prey_ss, n_pred)
```

Arguments

`prey_sigs`	A matrix of prey signatures in the prey space, intended to be the object `sig_scale` returned by a call to the function `prep_sig`, or the object `sig_part` returned by a call to the function `make_prey_part`.
`prey_loc`	A matrix giving the first and last locations of the signatures of each prey type within `prey_sigs`, intended to be the object `loc` returned by a call to one of the functions `prep_sig` or `make_prey_part`.
`cc`	A numeric vector containing the calibration coefficients.

`diet`	A numeric vector specifying the predator diet composition as proportions.
`prey_ss`	An integer vector specifying the bootstrap sample size to use for each prey type.
`n_pred`	An integer specifying the number of predator signatures to generate.

Value

A list containing the following elements:

sim_pred_sigs A numeric matrix containing simulated predator signatures in the predator space.

err_code An integer error code (0 if no error is detected).

err_message A string containing a brief summary of the results.

Details

QFASA simulation studies often require the generation of predator signatures given a specified diet, against which subsequent estimates of diet composition can then be compared (e.g., Bromaghin et al. 2016). Given a specified diet, a bootstrap sample of each prey type is drawn and mean prey-type signatures are computed. A predator signature is then generated by multiplying the mean bootstrapped prey signatures by the diet proportions. Finally, the calibration coefficients are then used to transform the predator signatures to the predator space (Bromaghin et al. 2015).

References

Bromaghin, J.F., S.M. Budge, and G.W. Thiemann. 2016. Should fatty acid signature proportions sum to 1 for diet estimation? *Ecological Research* 31:597-606.

Bromaghin, J.F., K.D. Rode, S.M. Budge, and G.W. Thiemann. 2015. Distance measures and optimization spaces in quantitative fatty acid signature analysis. *Ecology and Evolution* 5:1249-1262.

Examples

```
make_pred_sigs(prey_sigs = matrix(c(0.06, 0.09, 0.31, 0.54,
                                    0.05, 0.09, 0.30, 0.56,
                                    0.03, 0.10, 0.30, 0.57,
                                    0.08, 0.07, 0.30, 0.55,
                                    0.09, 0.05, 0.33, 0.53,
                                    0.09, 0.06, 0.34, 0.51,
                                    0.09, 0.07, 0.34, 0.50,
                                    0.08, 0.11, 0.35, 0.46,
                                    0.06, 0.14, 0.36, 0.44), ncol = 9),
               prey_loc = matrix(c(1, 4, 7, 3, 6, 9), ncol=2),
               cc = c(0.75, 1.05, 0.55, 1.75),
               diet = c(0.25, 0.25, 0.50),
               prey_ss = c(5, 3, 7),
               n_pred = 50)
```

make_prey_part *Make prey partition*

Description

The function make_prey_part partitions a prey library into clusters based on user specifications informed by the results of a call to the function dimac.

Usage

```
make_prey_part(sig, clust, n_clust)
```

Arguments

sig	A matrix of scaled signatures ready for analysis, intended to be the object sig_scale returned by the function prep_sig.
clust	A data frame containing cluster definitions, intended to be the object clust returned by the function dimac.
n_clust	An integer vector constructed by the user to specify the number of clusters into which each prey type should be partitioned.

Value

A list containing the following elements:

type A character vector of the partitioned type of each signature.

id A character vector of the unique sample ID of each signature.

n_types The number of unique types in the partitioned library.

uniq_types A character vector of the unique types, sorted alphanumerically.

type_ss The number of signatures for each unique type.

loc A vector or matrix giving the first and last locations of the signatures of each type, after being sorted by type and id.

sig_part A matrix of partitioned signatures ready for analysis, sorted by type and id, in column-major format.

pool_pre A matrix to pre-multiply diet estimates associated with a partitioned library to pool estimates back to the original prey types.

pool_post A matrix to post-multiply diet estimates associated with a partitioned library to pool estimates back to the original prey types.

err_code An integer error code (0 if no error is detected).

err_message A string contains a brief summary of the execution.

Details

The function make_prey_part partitions a matrix of prey signatures and into a larger number of prey types based on user input (in the vector n_clust) informed by the results of a preceding call to the clustering function dimac. The signatures in sig are presumed to be ready for analysis, which is best accomplished by a call to the function prep_sig.

For each prey type, the column in clust designated by the corresponding integer in n_clust is accessed and used to partition the prey type. For example, if the element of n_clust is 1, the first column of clust is accessed and the prey type is not partitioned. If the element of n_clust is 3, the third column of clust is accessed and the prey type is partitioned into three clusters.

The length of the integer vector n_clust must equal the number of unique types in type. The integers themselves should be between 1 and the number of signatures of each type.

After all prey types have been partitioned, the prey signatures are sorted by type and id. The matrix rep_grp is created to map the new prey types to the original prey types. Multiplying diet estimates corresponding to a partitioned prey library sig_part by rep_grp pools the diet estimates into the original prey types.

Please refer to the vignette and documentation for the functions dimac and prep_sig for additional information.

Examples

```
make_prey_part(sig = matrix(c(0.01, 0.20, 0.30, 0.49,
                              0.05, 0.14, 0.39, 0.42,
                              0.07, 0.21, 0.28, 0.44,
                              0.04, 0.19, 0.34, 0.43,
                              0.12, 0.29, 0.39, 0.20,
                              0.15, 0.28, 0.34, 0.23,
                              0.17, 0.21, 0.31, 0.31,
                              0.18, 0.22, 0.28, 0.32), ncol = 8),
               data.frame(type = c("prey_1", "prey_1", "prey_1", "prey_2",
                                   "prey_2", "prey_2", "prey_2", "prey_2"),
                          id = c("1-1", "1-2", "1-3", "2-1",
                                 "2-2", "2-3", "2-4", "2-5"),
                          clust_1 = c(1, 1, 1, 1, 1, 1, 1, 1),
                          clust_2 = c(1, 2, 1, 2, 1, 1, 2, 2),
                          clust_3 = c(1, 2, 3, 3, 1, 2, 3, 3),
                          clust_4 = c(0, 0, 0, 4, 1, 2, 3, 4)),
               n_clust = c(1, 2))
```

---

pm_obj_func *Parameterized mean objective function*

Description

The utility function pm_obj_func computes the total distance between observed predator signatures and vector of mean diet proportions common to all predators.

Usage

```
pm_obj_func(diet, obs_sig, mean_sigs, dist_meas = 1, gamma = 1)
```

Arguments

diet	A numeric vector of mean diet composition.
obs_sig	A numeric matrix containing observed predator signatures, in column-major format.
mean_sigs	A numeric matrix of the mean fatty acid signature for each prey type in the prey library, in column-major format.
dist_meas	An integer indicator of the distance measure to compute. Default value 1.
gamma	The power parameter of the chi-square distance measure. Default value 1.

Value

The total distance between observed and modeled signatures.

Details

This is an internal utility function. Consequently, to increase execution speed, no numeric error checking is performed within pm_obj_func. Rather, error checking is presumed to have occurred at a higher level in the calling sequence.

The argument obs_sig is presumed to be a matrix of predator signatures that has been prepared for analysis, which is best accomplished by a call to the function prep_sig with the predator data. Similarly, the contents of mean_sigs should be mean signatures computed from signatures that were prepared for analysis by a call to the function prep_sig.

The argument diet is presumed to contain non-negative proportions that sum to 1.0.

The arguments dist_meas and gamma must be compatible with the function dist_between_2_sigs.

Please refer to the vignette and documentation for the functions prep_sig, sig_scale, and dist_between_2_sigs for additional details.

diet_obj_func models a predator signature as a mixture of the mean prey-type signatures, with the diet proportions as the mixture proportions, returning the distance between the observed and modeled signatures. The diet composition of a predator is estimated by minimizing this function with respect to the diet using the function Rsolnp::solnp.

---

pred_beyond_prey *Identify predator signature proportions beyond range of prey*

Description

The function pred_beyond_prey identifies predator signaturee proportions that are outside the range of proportions observed in the individual and mean prey signatures.

Usage

```
pred_beyond_prey(pred_sigs, prey_sigs, mean_sigs)
```

Arguments

pred_sigs	A numeric matrix of predator signature(s) in column-major format. Intended to be the object pred_sigs returned by the function est_diet.
prey_sigs	A numeric matrix of prey signatures in column-major format. Intended to be the object prey_sigs returned by the function est_diet.
mean_sigs	A numeric matrix of mean prey-type signatures. Intended to be the object prey_sigs returned by the function est_diet.

Value

A list containing the following elements:

beyond_ind A logical matrix with TRUE indicating that the corresponding predator proportion is outside the range of individual prey proportions.

beyond_mean A logical matrix with TRUE indicating that the corresponding predator proportion is outside the range of mean prey proportions.

err_code An integer error code (0 if no error is detected).

err_message A string contains a brief summary of the execution.

Details

In quantitative fatty acid signature analysis, predator signatues are assumed to be a linear mixture of mean prey signatures (Iverson et al. 2004). Predator signature proportions should therefore be within the range of the prey signature proportions. Signature proportions outside the range of prey proportions are indicative of a violation of one or both of the primary model assumptions, i.e., the prey library is incomplete or the calibration coefficients are inaccurate (Bromaghin et al. 2015, 2016a). Consequently, checking for predator proportions that are outside the range of mean prey proportions is an important diagnostic aid to evaluate the reliability of diet estimates.

The function pred_beyond_prey identifies predator signature proportions that outside the range of proportions observed among the individual and mean prey signatures. For purposes of diet estimation, proportions outside the range of the mean signatures are most important. However, pred_beyond_prey also identifies predator proportions that are outside the range of the individual prey proportions for exploratory purposes.

pred_beyond_prey is designed to be called with inputs returned by the function est_diet. Although it is not conceptually necessary to estimate diets before performing this diagnostic check, doing so ensures that the predator and prey signatures have been transformed to the optimization space (Bromaghin et al. 2015) in which diets have been estimated.

References

Iverson, S.J., C. Field, W.D. Bowen, and W. Blanchard. 2004. Quantitative fatty acid signature analysis: A new method of estimating predator diets. *Ecological Monographs* 74:211-235.

Bromaghin, J.F., S.M. Budge, G.W. Thiemann, and K.D. Rode. 2016. Assessing the robustness of quantitative fatty acid signature analysis to assumption violations. *Methods in Ecology and Evolution* 7:51-59.

Bromaghin, J.F., K.D. Rode, S.M. Budge, and G.W. Thiemann. 2015. Distance measures and optimization spaces in quantitative fatty acid signature analysis. *Ecology and Evolution* 5:1249-1262.

Examples

```
pred_beyond_prey(pred_sigs = matrix(c(0.05, 0.10, 0.30, 0.55,
                                      0.04, 0.11, 0.29, 0.56,
                                      0.10, 0.05, 0.35, 0.50,
                                      0.12, 0.03, 0.37, 0.48,
                                      0.10, 0.06, 0.35, 0.49,
                                      0.05, 0.15, 0.35, 0.45), ncol = 6),
                 prey_sigs = matrix(c(0.06, 0.09, 0.31, 0.54,
                                      0.05, 0.09, 0.30, 0.56,
                                      0.03, 0.10, 0.30, 0.57,
                                      0.08, 0.07, 0.30, 0.55,
                                      0.09, 0.05, 0.33, 0.53,
                                      0.09, 0.06, 0.34, 0.51,
                                      0.09, 0.07, 0.34, 0.50,
                                      0.08, 0.11, 0.35, 0.46,
                                      0.06, 0.14, 0.36, 0.44), ncol = 9),
                 mean_sigs = matrix(c(0.047, 0.093, 0.303, 0.557,
                                      0.087, 0.050, 0.323, 0.530,
                                      0.077, 0.106, 0.350, 0.467), ncol = 3))
```

---

prep_fa *Prepare fatty acid information analysis*

Description

The function `prep_fa` processes the information in a fatty acid suites data frame and prepares that information for application to fatty acid signatures.

Usage

```
prep_fa(df_fa)
```

Arguments

`df_fa`	A data frame containing fatty acid names, calibration coefficients, and 0/1 definitions of fatty acid suites. `qfasar` has strict formatting requirements for `df_fa`; please see Details and/or the vignette.

Value

A list containing the following elements:

cc A numeric vector of calibration coefficients.

use A logical vector defining a fatty acid suite.

fa_names A character vector of fatty acid names.

err_code An integer error code (0 if no error is detected).

err_message A string contains a brief summary of the execution.

Details

This function is designed to be called by the user after the fatty acid data frame has been read. The data frame should contain a complete list of all fatty acids in the prey and predator signature data, one or more sets of calibration coefficients with an indicator of which set to use, one or more fatty acid suite definitions with an indicator of which suite to use, and optional comments. Please refer to the vignette for additional information.

The fatty acid data frame must strictly meet the following formatting requirements.

- The first row must contain a header for each column.
- The second row must list "use_me" in the first column, a 1 in the column for the set of calibration coefficients to be used, a 1 in the column for the fatty acid suite to be used, and a 0 in all other columns.
- Starting with row three, the first column must contain fatty acid names, which must exactly match the corresponding components of the headers in any prey and predator signature data frames.
- Starting with row three, Columns 2 to k must contain calibration coefficients for each fatty acid. Multiple sets of calibration coefficients can be in the data frame. The set to be used must contain a 1 in Row 1 and the others must contain a 0 in Row 1.
- Columns k+1 to m must contain one or more definitions of fatty acid suites. Membership in a suite is defined by 0/1 indicators, with a 1 indicating membership. Definitions for multiple suites can be in the data frame. For example, two columns could contain indicators defining membership in the dietary and extended-dietary suites of fatty acids (Iverson et al. 2004). The suite to be used must contain a 1 in Row 1 and the others must contain a 0 in Row 1.
- An optional last column can contain comments.
- Please see the vignette for examples of how to format this data frame.

References

Iverson, S.J., C. Field, W.D. Bowen, and W. Blanchard. 2004. Quantitative fatty acid signature analysis: A new method of estimating predator diets. *Ecological Monographs* 74:211-235.

Examples

```
prep_fa(data.frame(fa = c("use_me", "fa_1", "fa_2", "fa_3"),
                   cc = c(1, 0.75, 1.25, 1.0),
                   use = c(1, 1, 1, 1)))

prep_fa(data.frame(fa = c("use_me", "fa_1", "fa_2", "fa_3"),
                   cc1 = c(0, 0.75, 1.25, 1.00),
                   cc2 = c(1, 1.2, 0.8, 0.9),
                   use_1 = c(0, 1, 1, 0),
                   use_2 = c(1, 1, 1, 0)))
```

---

prep_sig *Prepare fatty acid signature data for analysis*

Description

The function prep_sig prepares raw fatty acid signatures for analysis. Signature proportions that are missing, negative, or equal to zero are replaced with a small user-specified constant and the signatures are scaled to sum to 1.0. The fatty acids that are not to be used in the analysis are censored and the signatures are scaled using one of three options (Bromaghin et al. In press).

Usage

```
prep_sig(df_sig, fa_names, use_fa, zero_rep = 75, scale = 3)
```

Arguments

df_sig	A data frame containing prey fatty acid signature data. qfasar has strict formatting requirements for df_sig; please see Details and/or the vignette.
fa_names	A character vector of all fatty acid names.
use_fa	A logical vector defining a fatty acid suite.
zero_rep	A constant associated with the method and value to replace signature proportions that are missing or less than or equal to 0. Default value 75.
scale	An integer indicator of the desired scaling option. Default value 3.

Value

A list containing the following elements:

type A character vector of the type of each signature.

id A character vector of the unique sample ID of each signature.

n_types The number of unique types.

uniq_types A character vector of the unique types, sorted alphanumerically.

n_sig The total number of signatures.

type_ss The number of signatures for each unique `type`.

loc A vector or matrix giving the first and last locations of the signatures of each `type`, after being sorted by `type` and `id`.

sig_rep A vector or matrix of the original signatures, with any values missing or less than or equal to 0 replaced, in column-major format.

n_fa_rep The number of fatty acids in `sig_rep`.

sig_scale A vector or matrix of scaled signatures ready for analysis, sorted by `type` and `id`, in column-major format.

n_fa_suite The number of fatty acids in `sig_scale`.

fa_suite A character vector of the names of fatty acids in the suite to be used in the analysis.

zero_rep_val A constant associated with the method and value to be used to replace proportions that are missing or less than or equal to 0. See Details.

err_code An integer error code (0 if no error is detected).

err_message A string contains a brief summary of the execution.

Details

This function is designed to be called by the user to prepare fatty acid signatures for analysis. For most analyses, `prep_sig` should be called immediately after the fatty acid suites and fatty acid signatures have been read into data frames, and after the fatty acid suites data frame has been processed by the function `prep_fa`. Please refer to the vignette for additional information.

The data frame with fatty acid signatures must meet the following formatting requirements:

- The file must be in row-major format, i.e., each row contains the information for an individual animal.
- The first column must contain a designation of animal `type`. For prey data, `type` often denotes species. For predator data, `type` denotes classes of predators for which separate estimates of mean diet composition are desired.
- The second column must contain an identifier unique to each signature, i.e. a sample ID.
- The remaining columns must contain fatty acid signature proportions or percentages.
- The data frame must contain a header record, with a name for each column, such as "type", "id", name of fatty acid 1, name of fatty acid 2, ...
- The file should contain data from all available fatty acids, rather than a subset. The fatty acid suite to be used in the analysis is defined by the argument `fa`.

Please refer to the documentation for the utility function `sig_rep_zero` for information regarding the argument `zero_rep`.

Please refer to the documentation for the utility function `sig_scale` for information regarding the argument `scale`.

References

Bromaghin, J.F., S.M. Budge, and G.W. Thiemann. In press. Should fatty acid signature proportions sum to 1 for diet estimation? *Ecological Research*.

Iverson, S.J., C. Field, W.D. Bowen, and W. Blanchard. 2004. Quantitative fatty acid signature analysis: A new method of estimating predator diets. *Ecological Monographs* 74:211-235.

Examples

```
prep_sig(df_sig = data.frame(type = c("Type_1", "Type_1", "Type_2",
                                      "Type_2"),
                             id = c("ID_1", "ID_2", "ID_3", "ID_4"),
                             fa_1 = c(0.0, 0.2, 0.3, 0.6),
                             fa_2 = c(0.1, 0.3, 0.3, 0.4),
                             fa_3 = c(0.9, 0.5, 0.4, NA),
                             row.names = c("Prey_1", "Prey_2", "Prey_3",
                                           "Prey_4")),
         fa_names = c("fa_1", "fa_2", "fa_3"),
         use_fa = c(TRUE, FALSE, TRUE),
         zero_rep = 0.0001,
         scale=2)

prep_sig(df_sig = data.frame(type = c("Type_1", "Type_1", "Type_2",
                                      "Type_2"),
                             id = c("ID_1", "ID_2", "ID_3", "ID_4"),
                             fa_1 = c(0.0, 0.2, 0.3, 0.6),
                             fa_2 = c(0.1, 0.3, 0.3, 0.4),
                             fa_3 = c(0.9, 0.5, 0.4, NA),
                             row.names = c("Prey_1", "Prey_2", "Prey_3",
                                           "Prey_4")),
         fa_names = c("fa_1", "fa_2", "fa_3"),
         use_fa = c(TRUE, FALSE, TRUE),
         zero_rep = 90,
         scale=1)

prep_sig(df_sig = data.frame(type = c("Type_1", "Type_1", "Type_2",
                                      "Type_2"),
                             id = c("ID_1", "ID_2", "ID_3", "ID_4"),
                             fa_1 = c(0.0, 0.2, 0.3, 0.6),
                             fa_2 = c(0.1, 0.3, 0.3, 0.4),
                             fa_3 = c(0.9, 0.5, 0.4, NA),
                             row.names = c("Prey_1", "Prey_2", "Prey_3",
                                           "Prey_4")),
         fa_names = c("fa_1", "fa_2", "fa_3"),
         use_fa = c(TRUE, FALSE, TRUE),
         scale=3)

prep_sig(df_sig = data.frame(type = c("Type_1", "Type_1", "Type_2",
                                      "Type_2"),
                             id = c("ID_1", "ID_2", "ID_3", "ID_4"),
                             fa_1 = c(0.0, 0.2, 0.3, 0.6),
                             fa_2 = c(0.1, 0.3, 0.3, 0.4),
                             fa_3 = c(0.9, 0.5, 0.4, NA),
                             row.names = c("Prey_1", "Prey_2", "Prey_3",
                                           "Prey_4")),
         fa_names = c("fa_1", "fa_2", "fa_3"),
         use_fa = c(TRUE, FALSE, TRUE))
```

`sig_rep_zero` *Replace invalid fatty acid signature proportions*

Description

The utility function `sig_rep_zero` replaces fatty acid signature proportions that are less than or equal to zero or missing with a small constant and uses the multiplicative method (Martin-Fernandez et al. 2011) to scale the proportions to sum to 1.

Usage

```
sig_rep_zero(sig_data, zero_rep = 75)
```

Arguments

`sig_data`	A numeric matrix containing signature data as either proportions or percentages in column-major format.
`zero_rep`	A constant associated with the method and value to be used to replace invalid values. See Details. Default value 75.

Value

A list containing the following elements:

sig_adj The signature data with non-positive or missing proportions replaced and scaled to sum to 1.0.

rep_val The value used to replace invalid proporitons.

err_code An integer error code (0 if no error is detected).

err_message A string contains a brief summary of the execution.

Details

The function `sig_rep_zero` is an internal utility function.

The Kullback-Leibler (Iverson et al. 2004) and Aitchison (Stewart et al. 2014) distance measures are not defined for proportions of zero. Consequently, if either of these distance measures will be used in an analysis, the argument `zero_rep` should be strictly greater than 0. The chi-square distance measure (Stewart et al. 2014) is defined for proportions of zero, so if that distance measure will be used in the analysis, the argument `zero_rep` may equal zero. For simulation or other comparative work involving multiple distance measures, it may be advisable to use a common value to replace zeros.

The argument `zero_rep` must be either:

- Greater than or equal to 0 and no greater than 0.01, in which case the specified value is used to replace invalid proportions.

- Between 10 and 100, with an uninformed default of 75. In this case, `zero_rep` is interpreted as a percentage. The smallest non-zero proportion in `sig_data` is multiplied by the percentage and divided by 100. The result is used to replace invalid proportions.

Although Bromaghin et al. (2016) found that scaling signatures by varying constants introduces a bias in diet estimation, the slight distortion of the signatures caused by replacing invalid proportions with a small constant that varies between signatures is unlikely to introduce meaningful bias.

References

Bromaghin, J.F., S.M. Budge, and G.W. Thiemann. 2016. Should fatty acid signature proportions sum to 1 for diet estimation? *Ecological Research* 31:597-606.

Iverson, S.J., C. Field, W.D. Bowen, and W. Blanchard. 2004. Quantitative fatty acid signature analysis: A new method of estimating predator diets. *Ecological Monographs* 74:211-235.

Martin-Fernandez, J.A., J. Palarea-Albaladejo, and R.A. Olea. 2011. Dealing with zeros. P. 43-58 in V. Pawlowsky-Glahn and A. Buccianto, eds. Compositional data analysis: theory and application. John Wiley, Chichester.

Stewart, C., and C. Field. 2011. Managing the essential zeros in quantitative fatty acid signature analysis. *Journal of Agricultural, Biological, and Environmental Statistics* 16:45?69.

`sig_scale` *Scale fatty acid signature proportions*

Description

The utility function `sig_scale` implements the three options for scaling fatty acid signature data summarized by Bromaghin et al. (2016). A logical vector denotes the subset of all fatty acids to be used in the analysis. The fatty acids that are not to be used are censored and one of three scaling options is implemented. See Details.

Usage

```
sig_scale(sig_data, fa_use, scale = 3)
```

Arguments

`sig_data`	A numeric matrix containing prey signature data as proportions in column-major. These data should have previously been processed by `sig_rep_zero`.
`fa_use`	A logical vector denoting the fatty acids to be used, of length equal to the total number of fatty acids. This vector originates from a data file required by `qfasar`. See the vignette for details.
`scale`	An integer indicator of the desired scaling option. See Details. Default value 3.

Value

A list containing the following elements:

n_fa The number of fatty acids in the processed signatures.

sig A numeric matrix of processed signatures in column-major format.

err_code An integer error code (0 if no error is detected).

err_message A string contains a brief summary of the execution.

Details

This is an internal utility function.

The argument `scale` must be one of three integer values and its value denotes the scaling option that will be implemented:

- `scale` == 1. The proportions within each censored signature are scaled to sum to 1.0. This option is not recommended for routine use in QFASA applications, as Bromaghin et al. (2016) found that it can meaningfully bias diet estimates under some conditions. It is implemented here to provide compatibility with original methods and to facilitate potential future research.
- `scale` == 2. The proportions within each censored signature are not scaled, so each signature will have a different partial sum.
- `scale` == 3. Each censored signature is augmented with an additional proportion whose value equals the sum of the censored proportions, so that the proportions in each signature sum to 1. This is the default option.

References

Bromaghin, J.F., S.M. Budge, and G.W. Thiemann. 2016. Should fatty acid signature proportions sum to 1 for diet estimation? *Ecological Research* 31:597-606.

Index