Package ‘fishboot’

April 19, 2025

Type Package

Title Bootstrap-Based Methods for the Study of Fish Stocks and Aquatic Populations

Version 1.0.0

Date 2025-04-17

Description A suite of bootstrap-based models and tools for analyzing fish stocks and aquatic populations. Designed for ecologists and fisheries scientists, it supports data from length-frequency distributions, tag-and-recapture studies, and hard structure readings (e.g., otoliths). See Schwamborn et al., 2019 <doi:10.1016/j.ecolmodel.2018.12.001> for background. The package includes functions for bootstrapped fitting of growth curves and plotting.

Depends R (>= 4.1.0)

Imports doParallel, parallel, foreach, grDevices, graphics, stats, utils, ks, TropFishR, fishmethods

License GPL-3

BugReports `https://github.com/rschwamborn/fishboot/issues`

URL `https://github.com/rschwamborn/fishboot`

Encoding UTF-8

LazyData true

RoxygenNote 7.3.2

NeedsCompilation no

Author Ralf Schwamborn [aut, cre] (<`https://orcid.org/0000-0001-9150-8720`>),
Tobias K. Mildenberger [aut],
Marc H. Taylor [aut],
Margit Wilhelm [aut] (<`https://orcid.org/0000-0001-9271-2109`>),
Wencheng Lau-Medrano [ctb] (<`https://orcid.org/0000-0002-7979-9710`>)

Maintainer Ralf Schwamborn <`ralf.schwamborn@ufpe.br`>

Repository CRAN

Date/Publication 2025-04-19 12:52:09 UTC

Contents

`alba_boot` *Bootstrapped VBGF estimates for the alba data set*

Description

Bootstrapped VBGF estimates for the alba length frequency data set as estimated by `ELEFAN_GA`.

Usage

```
alba_boot
```

Format

`alba_boot`:

A `lfqBoot` object with two levels:

`$bootRaw` A `data.frame` of fitted VBGF parameters (columns) by resampling (rows).

`$seed` A `numeric` vector of seed values set prior to each resampling call to lfqResample.

`bonito_boot` *Bootstrapped growth estimates for the bonito data set*

Description

Bootstrapped growth estimates for the bonito age-growth data calculated using the grotag_boot as this:

```
data("bonito", package = "fishmethods")

grotag_boot(input.data = bonito,
            alpha = 35, beta = 55,
            design  = list(nu = 1, m = 1,p = 1, sea = 1),
```

```
            stvalue = list(sigma = 0.9, nu = 0.4, m = -1,
                           p = 0.2, u = 0.4, w = 0.4),
            upper   = list(sigma = 5, nu = 1, m = 2,
                           p = 0.5, u = 1, w = 1),
            lower   = list(sigma = 0, nu = 0, m = -2,
                           p = 0.0, u = 0, w = 0),
            control = list(maxit = 1e4),
            seed = 1234, nresamp = 100, cc.only = TRUE)
```

Usage

```
bonito_boot
```

Format

`bonito_boot`:

A `grotagBoot` object with four columns: `Linf`, `K`, `PhiL`, `u`, `w` and `seed`.

`ELEFAN_GA_boot` *Bootstraped ELEFAN_GA*

Description

This function performs a bootstrapped fitting of a von Bertalanffy growth function (VBGF) via the `ELEFAN_GA` function. Most of the arguments are simply passed to the function within many permutations (resampling) of the original `lfq` data. As the original function, `ELEFAN_GA` also conducts Electronic LEngth Frequency ANalysis using a genetic algorithm (GA) to estimate growth parameters. Partial (repeated fitting on original data) and full bootstrap (with resampling) routines are possible, depending on `resample`.

Usage

```
ELEFAN_GA_boot(
  lfq,
  seasonalised = FALSE,
  low_par = NULL,
  up_par = NULL,
  popSize = 50,
  maxiter = 100,
  run = maxiter,
  parallel = FALSE,
  pmutation = 0.1,
  pcrossover = 0.8,
  elitism = base::max(1, round(popSize * 0.05)),
  MA = 5,
  addl.sqrt = FALSE,
  agemax = NULL,
```

```
  ...,
  seed = NULL,
  nresamp = 10,
  resample = TRUE,
  outfile = NA
)
```

Arguments

`lfq`	a length frequency object of the class `lfq` (see lfqCreate).
`seasonalised`	logical; indicating if the seasonalised von Bertalanffy growth function should be applied (default: FALSE).
`low_par`	a list providing the minimum of the search space in case of real-valued or permutation encoded optimizations. When set to NULL the following default values are used: • **Linf** length infinity in cm (default is calculated from maximum length class in the data), • **K** curving coefficient (default: 0.01), • **t_anchor** time point anchoring growth curves in year-length coordinate system, corresponds to peak spawning month (range: 0 to 1, default: 0), • **C** amplitude of growth oscillation (range: 0 to 1, default: 0), • **ts** summer point (ts = WP - 0.5) (range: 0 to 1, default: 0);
`up_par`	a list providing the maximum of the search space in case of real-valued or permutation encoded optimizations. When set to NULL the following default values are used: • **Linf** length infinity in cm (default is calculated from maximum length class in the data), • **K** curving coefficient (default: 1), • **t_anchor** time point anchoring growth curves in year-length coordinate system, corresponds to peak spawning month (range: 0 to 1, default: 1), • **C** amplitude of growth oscillation (range: 0 to 1, default: 1), • **ts** summer point (ts = WP - 0.5) (range: 0 to 1, default: 1);
`popSize`	the population size. Default: 50
`maxiter`	the maximum number of iterations to run before the GA search is halted. default:100
`run`	the number of consecutive generations without any improvement in the best fitness value before the GA is stopped. Default: equals `maxiter`.
`parallel`	Whether a `logical` or `integer` argument specifying the configuration of parallel computing. See `ga` for details.
`pmutation`	the probability of mutation in a parent chromosome. Usually mutation occurs with a small probability, and by default is set to 0.1.
`pcrossover`	the probability of crossover between pairs of chromosomes. Typically this is a large value and by default is set to 0.8.

`elitism`	the number of best fitness individuals to survive at each generation. By default the top 5% individuals will survive at each iteration.
`MA`	number indicating over how many length classes the moving average should be performed (default: 5, for more information see lfqRestructure)
`addl.sqrt`	additional square root transformation of positive values according to Brey et al. (1988) (default: FALSE, for more information see lfqRestructure)
`agemax`	maximum age of species; default `NULL`, then estimated from L_{inf}.
`...`	additional parameters to pass to `ga`.
`seed`	seed value for random number reproducibility.
`nresamp`	`numeric`, the number of permutations to run (by default `nresamp = 10`).
`resample`	`logical`. Do you want that `lfq` object be resampled (`TRUE` by default).
`outfile`	`character`; path of the file which will register the progress of the permutation completions. If it is set as `false`, `NA` or `NULL`, no file will be created.

Details

If `resample = TRUE`, a **full non-parametric bootstrap** is performed with resampling from the original length-frequencies by using the function lfqResample. Otherwise, if `resample = FALSE`, a partial bootstrap is performed, reflecting solution variation due only to the search algorithm, with repeated fitting to the original data (no resampling is performed).

Value

An object of class `lfqBoot` containing 2 levels:

`$bootRaw` A `data.frame` of fitted VBGF parameters (columns) by resampling (rows).

`$seed` A `numeric` vector of seed values set prior to each resampling call to lfqResample.

References

- Brey, T., Soriano, M., and Pauly, D. 1988. Electronic length frequency analysis: a revised and expanded user's guide to ELEFAN 0, 1 and 2.
- Efron, B., & Tibshirani, R. (1986). Bootstrap methods for standard errors, confidence intervals, and other measures of statistical accuracy. Statistical Science, 54-75.
- Mildenberger, T., Taylor, M. H., & Wolff, A. M., 2017. TropFishR: an R package for fisheries analysis with length-frequency data. Methods in Ecology and Evolution, 8(11), 1520-1527.
- Pauly, D. 1981. The relationship between gill surface area and growth performance in fish: a generalization of von Bertalanffy's theory of growth. Meeresforsch. 28:205-211.
- Pauly, D. and N. David, 1981. ELEFAN I, a BASIC program for the objective extraction of growth parameters from length-frequency data. Meeresforschung, 28(4):205-211.
- Schwamborn, R., Mildenberger, T. K., & Taylor, M. H., 2019. Assessing sources of uncertainty in length-based estimates of body growth in populations of fishes and macroinvertebrates with bootstrapped ELEFAN. Ecological Modelling, 393, 37-51.

- Schwamborn, R., Freitas, M. O., Moura, R. L., & Aschenbrenner, A. 2023. Comparing the accuracy and precision of novel bootstrapped length-frequency and length-at-age (otolith) analyses, with a case study of lane snapper (*Lutjanus synagris*) in the SW Atlantic. Fisheries Research, 264, 106735.
- Scrucca, L., 2013. GA: a package for genetic algorithms in R. Journal of Statistical Software, 53(4), 1-37.
- von Bertalanffy, L., 1938. A quantitative theory of organic growth. Human Biology 10, 181-213.

Examples

```
# load data
data("alba", package = "TropFishR")

# Define settings (for demo only, fast settings)
MA        <- 7
low_par   <- list(Linf = 9, K = 0.4, t_anchor = 0.5, C = 0, ts = 0)
up_par    <- list(Linf = 11, K = 0.6, t_anchor = 0.8, C = 1, ts = 1)
popSize   <- 12
maxiter   <- 5
run       <- 4
pmutation <- 0.1
nresamp   <-  2

# Non-parallel version
res <- ELEFAN_GA_boot(lfq = alba, MA = MA, seasonalised = FALSE,
                      up_par = up_par, low_par = low_par,
                      parallel = FALSE,
                      popSize = popSize, maxiter = maxiter,
                      run = run, pmutation = pmutation,
                      nresamp = nresamp)

res



# Define settings (for demo only)
MA        <- 7
low_par   <- list(Linf = 8, K = 0.2, t_anchor = 0, C = 0, ts = 0)
up_par    <- list(Linf = 12, K = 0.9, t_anchor = 1, C = 1, ts = 1)
popSize   <- 40
maxiter   <- 30
run       <- 10
pmutation <- 0.2
nresamp   <-  3

# Parallel version
res <- ELEFAN_GA_boot(lfq = alba, MA = MA, seasonalised = FALSE,
                      up_par = up_par, low_par = low_par,
                      parallel = TRUE,
                      popSize = popSize, maxiter = maxiter,
                      run = run, pmutation = pmutation,
```

```
                        nresamp = nresamp)

res
```

`ELEFAN_SA_boot` *Bootstraped ELEFAN_SA*

Description

This function performs a bootstrapped fitting of a von Bertalanffy growth function (VBGF) via the `ELEFAN_SA` function. Most of the arguments are simply passed to the function within many permutations (resampling) of the original `lfq` data. Partial (repeated fitting on original data) and full bootstrap (with resampling) routines are possible, depending on `resample`.

Usage

```
ELEFAN_SA_boot(
  lfq,
  seasonalised = FALSE,
  init_par = list(Linf = 50, K = 0.5, t_anchor = 0.5, C = 0, ts = 0),
  low_par = NULL,
  up_par = NULL,
  SA_time = 60 * 1,
  maxit = NULL,
  nb.stop.improvement = NULL,
  SA_temp = 1e+05,
  MA = 5,
  addl.sqrt = FALSE,
  agemax = NULL,
  seed = NULL,
  nresamp = 10,
  resample = TRUE,
  parallel = FALSE,
  outfile = NA
)
```

Arguments

`lfq`	a length frequency object of the class `lfq` (see lfqCreate).
`seasonalised`	logical; indicating if the seasonalised von Bertalanffy growth function should be applied (default: FALSE).
`init_par`	a list providing the Initial values for the components to be optimized. When set to NULL the following default values are used: • **Linf** length infinity in cm (default is the maximum length class in the data), • **K** curving coefficient (default: 0.5),

	• **t_anchor** time point anchoring growth curves in year-length coordinate system, corresponds to peak spawning month (range: 0 to 1, default: 0.5), • **C** amplitude of growth oscillation (range: 0 to 1, default: 0), • **ts** summer point (ts = WP - 0.5) (range: 0 to 1, default: 0);
`low_par`	a list providing the lower bounds for components. When set to NULL the following default values are used: • **Linf** length infinity in cm (default is calculated from maximum length class in the data), • **K** curving coefficient (default: 0.01), • **t_anchor** time point anchoring growth curves in year-length coordinate system, corresponds to peak spawning month (range: 0 to 1, default: 0), • **C** amplitude of growth oscillation (range: 0 to 1, default: 0), • **ts** summer point (ts = WP - 0.5) (range: 0 to 1, default: 0);
`up_par`	a list providing the upper bounds for components. When set to NULL the following default values are used: • **Linf** length infinity in cm (default is calculated from maximum length class in the data), • **K** curving coefficient (default: 0.01), • **t_anchor** time point anchoring growth curves in year-length coordinate system, corresponds to peak spawning month (range: 0 to 1, default: 0), • **C** amplitude of growth oscillation (range: 0 to 1, default: 0), • **ts** summer point (ts = WP - 0.5) (range: 0 to 1, default: 0);
`SA_time`	numeric; Maximum running time in seconds (default : 60 * 1).
`maxit`	Integer. Maximum number of iterations of the algorithm. Default is NULL.
`nb.stop.improvement`	Integer. The program will stop when there is no any improvement in ’nb.stop.improvement’ steps. Default is NULL
`SA_temp`	numeric; Initial value for temperature (default : 1e5).
`MA`	number indicating over how many length classes the moving average should be performed (defalut: 5, for more information see lfqRestructure).
`addl.sqrt`	Passed to lfqRestructure. Applied an additional square-root transformation of positive values according to Brey et al. (1988). (default: FALSE, for more information see lfqRestructure).
`agemax`	maximum age of species; default NULL, then estimated from L_{inf}.
`seed`	seed value for random number reproducibility.
`nresamp`	`numeric`, the number of permutations to run (by default `nresamp = 10`).
`resample`	`logical`. Do you want that `lfq` object be resampled (`TRUE` by default).
`parallel`	Whether a `logical` or `integer` argument specifying the configuration of parallel computing. If `parallel = TRUE`, the number of threads will be defined as `parallel::detectCores() - 2`.
`outfile`	`character`; path of the file which will register the progress of the permutation completions. If it is set as `false`, `NA` or `NULL`, no file will be created.

Details

If `resample = TRUE`, a **full non-parametric bootstrap** is performed with resampling from the original length-frequencies by using the function lfqResample. Otherwise, if `resample = FALSE`, a partial bootstrap is performed, reflecting solution variation due only to the search algorithm, with repeated fitting to the original data (no resampling is performed).

Value

An object of class `lfqBoot` containing 2 levels:

`$bootRaw` A `data.frame` of fitted VBGF parameters (columns) by resampling (rows).

`$seed` A `numeric` vector of seed values set prior to each resampling call to lfqResample.

References

- Brey, T., Soriano, M., and Pauly, D. 1988. Electronic length frequency analysis: a revised and expanded user's guide to ELEFAN 0, 1 and 2.
- Efron, B., & Tibshirani, R. (1986). Bootstrap methods for standard errors, confidence intervals, and other measures of statistical accuracy. Statistical Science, 54-75.
- Mildenberger, T., Taylor, M. H., & Wolff, A. M., 2017. TropFishR: an R package for fisheries analysis with length-frequency data. Methods in Ecology and Evolution, 8(11), 1520-1527.
- Pauly, D. 1981. The relationship between gill surface area and growth performance in fish: a generalization of von Bertalanffy's theory of growth. Meeresforsch. 28:205-211.
- Pauly, D. and N. David, 1981. ELEFAN I, a BASIC program for the objective extraction of growth parameters from length-frequency data. Meeresforschung, 28(4):205-211.
- Schwamborn, R., Mildenberger, T. K., & Taylor, M. H., 2019. Assessing sources of uncertainty in length-based estimates of body growth in populations of fishes and macroinvertebrates with bootstrapped ELEFAN. Ecological Modelling, 393, 37-51.
- Schwamborn, R., Freitas, M. O., Moura, R. L., & Aschenbrenner, A. 2023. Comparing the accuracy and precision of novel bootstrapped length-frequency and length-at-age (otolith) analyses, with a case study of lane snapper (*Lutjanus synagris*) in the SW Atlantic. Fisheries Research, 264, 106735.
- Scrucca, L., 2013. GA: a package for genetic algorithms in R. Journal of Statistical Software, 53(4), 1-37.
- von Bertalanffy, L., 1938. A quantitative theory of organic growth. Human Biology 10, 181-213.

Examples

```
# load data
data("alba", package = "TropFishR")

# Define settings (for demo only, fast settings)
MA       <- 7
low_par   <- list(Linf = 9, K = 0.4, t_anchor = 0.5, C = 0, ts = 0)
up_par    <- list(Linf = 11, K = 0.6, t_anchor = 0.8, C = 1, ts = 1)
init_par <- list(Linf = 10, K = 0.5, t_anchor = 0.6, C = 0.5, ts = 0.5)
```

```
SA_time  <- 1.5
SA_temp  <- 1e5
nresamp  <- 2

# Non-parallel bootstrapped curve fiting
res <- ELEFAN_SA_boot(lfq = alba, MA = MA, seasonalised = FALSE,
                      init_par = init_par, up_par = up_par, low_par = low_par,
                      SA_time = SA_time, SA_temp = SA_temp,
                      nresamp = nresamp)

res


# Define settings (for demo only)
MA       <- 7
low_par  <- list(Linf = 8, K = 0.1, t_anchor = 0, C = 0, ts = 0)
init_par <- list(Linf = 12, K = 0.5, t_anchor = 0.5, C = 0.5, ts = 0.5)
up_par   <- list(Linf = 15, K = 5, t_anchor = 1, C = 1, ts = 1)
SA_time  <- 10
SA_temp  <- 1e5
nresamp  <- 12

# parallel version
res <- ELEFAN_SA_boot(lfq = alba, MA = MA, seasonalised = FALSE,
                      init_par = init_par, up_par = up_par, low_par = low_par,
                      SA_time = SA_time, SA_temp = SA_temp,
                      nresamp = nresamp,
                      parallel = TRUE)

res
```

`grolenage_boot` *Bootstrapped length-at-age analysis*

Description

This function obtains growth parameter estimates from length-at-age data. Since it internally uses the function growth_length_age, this function allows to perform different methods: Gulland and Holt, Ford Walford, Chapman, Bertalanffy, or non linear least squares method (LSM).

This function performs bootstrapped fitting of the von Bertalanffy growth function with estimated growth parameters (L_{inf}, K and t_0) from length-at-age data, based on the function growth_length_age. The output is an object containing the parameters L_{inf}, K and t_0 (named here `t_anchor`) as well as the growth performance index Phi' (named `PhiL`).

Usage

```
grolenage_boot(
  param,
```

```
  method = "LSM",
  Linf_est = NA,
  Linf_init = 100,
  K_init = 0.1,
  t0_init = 0,
  seed = NULL,
  nresamp = 200,
  nan_action = c("nothing", "nanrm", "narm", "force"),
  time_lim = 5 * 60
)
```

Arguments

`param`	a `list` (or `data.frame`) consisting of following parameters (levels/columns): • **age**: age measurements (e.g. from otoliths), • **length**: corresponding length measurements.
`method`	indicating which of following methods should be applied: `"GullandHolt"`, `"FordWalford"`, `"Chapman"`, `"BertalanffyPlot"`, or `"LSM"` (it corresponds to the non-linear least squares fitting method, and is the default, which is recommended for bootstrapping growth).
`Linf_est`	BertalanffyPlot requires an estimate for L_{inf} to derive K and t_0 (for more information see Details).
`Linf_init`	initial parameter of L_{inf} for non-linear squares fitting (default 100).
`K_init`	initial parameter of K for non-linear squares fitting (default 0.1).
`t0_init`	initial parameter of L_0 for non-linear squares fitting (default 0).
`seed`	seed value for random number reproducibility (if it `NULL` by default, it will set internally as `seed = as.numeric(Sys.time())`).
`nresamp`	`numeric`; the number of permutations to run (Default: `nresamp = 200`).
`nan_action`	`character` that defines the action that the function will execute if there is a row with `NaN` (growth rate parameters inestimable for that resample): • `nothing`: the function will return the results including the `NaN`s (default). • `nanrm` or `narm`: after having the results, it will only returns the rows without `NaN`s. For this case `narm` and `nanrm` are equivalent, but it should be noted that the function will look for and omit the `NaN`s (and not the `NA`s). See Details. • `force`: The function will start an iterative process changing the internal seed values until it fulfills the `nresamp`. It only works together with the `time_lim` argument. See Details.
`time_lim`	If `nan_action = "force"`, it defines the maximum time (in seconds) that the function will last resampling until it achieves a result output with no-`NaN` rows.

Details

It is important to take into account the particular considerations of each method regarding the required parameters, so it is recommended to read the Details of the documentation of growth_length_age.

CI and plotting arguments (of growth_length_age) are set as `FALSE` for each bootstrap call here. By default, growth_length_age generates a plot when it is called, so internally `grolenage_boot` executes a `dev.off` call in order to prevent it.

`nan_action = "force"` should be used carefully, as estimated `NaN` VBGF parameter values are not always a result of bootstrap data selection factors. Few resamples should first be tested with different `Linf_init`, `K_init` and `t0_init` values. No selection of the realistic initial parameters may also result in `NaN` values being obtained. The search time may depend on the size of the input set. For example, if you have many thousands of individuals or if (in addition) the value of `nresamp` is too high, it is possible that the function will take a long time before obtaining complete results. Even though `time_lim` avoids falling into an infinite loop by limiting the time used by this process to 5 minutes, this value is referential and might be insufficient due to the factors mentioned above.

`t_anchor` is the true t_0 estimate in the case of true length-at-age data, but it will only be available from "BertalanffyPlot" or "LSM" methods. For the other methods, a vector of `NA`s will be returned instead.

Value

An object of class `lfqBoot` containing 2 levels:

`$bootRaw` A `data.frame` of fitted VBGF parameters (columns) by resampling (rows).

`$seed` A `numeric` vector of seed values set prior to each resampling call.

References

- Efron, B., & Tibshirani, R., 1986. Bootstrap methods for standard errors, confidence intervals, and other measures of statistical accuracy. Statistical Science, 54-75.
- Pauly, D. 1981. The relationship between gill surface area and growth performance in fish: a generalization of von Bertalanffy's theory of growth. Meeresforsch. 28:205-211.
- Schwamborn, R., Mildenberger, T. K., & Taylor, M. H., 2019. Assessing sources of uncertainty in length-based estimates of body growth in populations of fishes and macroinvertebrates with bootstrapped ELEFAN. Ecological Modelling, 393, 37-51.
- Schwamborn, R., Freitas, M. O., Moura, R. L., & Aschenbrenner, A. 2023. Comparing the accuracy and precision of novel bootstrapped length-frequency and length-at-age (otolith) analyses, with a case study of lane snapper (*Lutjanus synagris*) in the SW Atlantic. Fisheries Research, 264, 106735.
- von Bertalanffy, L., 1938. A quantitative theory of organic growth. Human Biology 10, 181-213.

Examples

```
# Synthetical length at age data
dat <- list(age = rep(x = 1:7,each = 15),
            length = c(rnorm(n = 15, mean = 4.6, sd = 4),
                       rnorm(n = 15, mean = 22.8, sd = 7),
                       rnorm(n = 15, mean = 35, sd = 7),
                       rnorm(n = 15, mean = 43, sd = 7),
                       rnorm(n = 15, mean = 49, sd = 5),
                       rnorm(n = 15, mean = 53, sd = 5),
```

```
                        rnorm(n = 15, mean = 57, sd = 3)))

# Perform bootstrapped curve fitting with grolenage_boot
res <- grolenage_boot(param = dat, nresamp = 70)

# Plot scatter histograms of Linf and K
LinfK_scatterhist(res = res)

# Plot univariate density plots of all parameters
univariate_density(res = res)

# Plot swarm plots of all n bootstraps
vbgfCI_time(res = res)

# Extract data.frame with all parameter estimates
# for comparisons of posterior distributions
print(res$bootRaw)
```

`grotag_boot` *Bootstrapped tag-and-recapture growth analysis*

Description

This function performs bootstrapped fitting of the von Bertalanffy growth function (VBGF) with estimated growth parameters (L_{inf}, K and t_0) from tag-and-recapture data, based on the function grotag, that estimates VBGF parameters according to Francis (1988). The output is an object containing the parameters L_{inf} and K, as well as the growth performance index Phi' (named `PhiL`).

This function resamples the `input.data` data by rows (i.e., by recapture date) several times (`nresamp` times, default: `nresamp = 200`). Then, a VBGF curve is fitted to each resampled data set. The output (a `list` of class `lfqBoot`) will store results (e.g., VGBGF function parameters K and Linf) in a `data.frame` accessible through `$bootRaw`. The `$bootRaw` table also includes the growth performance index **Phi'**, seasonal parameters **u** and **w** (sensu Francis, 1988), which are equal to **C** (sensu Pauly and Gaschütz, 1979) and and **ts** (sensu Mildenberger et al., 2017). The `$bootRaw` table also includes seed values and system time.

Usage

```
grotag_boot(
  L1 = NULL,
  L2 = NULL,
  T1 = NULL,
  T2 = NULL,
  alpha = NULL,
  beta = NULL,
  design = list(nu = 0, m = 0, p = 0, sea = 0),
  stvalue = list(sigma = 0.9, nu = 0.4, m = -1, p = 0.1, u = 0.4, w = 0.4),
  upper = list(sigma = 5, nu = 1, m = 2, p = 1, u = 1, w = 1),
  lower = list(sigma = 0, nu = 0, m = -2, p = 0, u = 0, w = 0),
```

```
  gestimate = TRUE,
  st.ga = NULL,
  st.gb = NULL,
  st.galow = NULL,
  st.gaup = NULL,
  st.gblow = NULL,
  st.gbup = NULL,
  control = list(maxit = 10000),
  input.data = NULL,
  seed = NULL,
  nresamp = 200,
  na_action = c("nothing", "narm", "force"),
  time_lim = 5 * 60
)
```

Arguments

`L1, L2, T1, T2`	Name of the columns to be extracted from `input.data` and used by the grotag function for the arguments `L1`, `L2`, `T1` and `T2`, respectively. See Details.
`alpha`	`numeric` value giving an arbitrary length alpha.
`beta`	`numeric` value giving an arbitrary length beta (`beta` > `alpha`).
`design`	`list` specifying the design of the model to estimate. Use 1 to designate whether a parameter(s) should be estimated. Type of parameters are: • `nu`: growth variability (1 parameter). • `m`: bias parameter of measurement error (1 parameter). • `p`: outlier probability (1 parameter). • `sea`: seasonal variation (2 parameters: u and w). Model 1 of Francis is the default settings of 0 for `nu`, `m`, `p` and `sea`.
`stvalue`	Starting values of sigma(s) and depending on the `design` argument, `nu`, `m`, `p`, `u`, and `w` used as input in the nonlinear estimation (function optim) routine.
`upper, lower`	Upper and lower limits of the model parameters' (`nu`, `m`, `p`, `u`, and `w`) region to be investigated.
`gestimate`	`logical` specifying whether starting values of **ga** and **gb** (growth increments of `alpha` and `beta`) should be estimated automatically. `TRUE` by default.
`st.ga, st.gb`	If `gestimate=FALSE`, user-specified starting value for ga and gb respectively.
`st.galow, st.gaup`	If `gestimate=FALSE`, user-specified lower and upper limits for `st.ga` used in optimization.
`st.gblow, st.gbup`	If `gestimate=FALSE`, user-specified lower and upper limits for `st.gb` used in optimization.
`control`	Additional controls passed to the optimization function optim.
`input.data`	A growth increment object of the class `data.frame`.
`seed`	seed value for random number reproducibility (if it `NULL` by default, it will set internally as `seed` = `as.numeric(Sys.time())`).

`nresamp`	`numeric`; the number of permutations to run (Default: `nresamp = 200`).
`na_action`	`character` that defines the action that the function will execute if there is a row with NA: • `nothing`: the function will return the results including the `NA`s (default). • `narm`: after having the results, it will only returns the rows without `NA`s. See Details. • `force`: The function will start an iterative process changing the internal `seed` values until it fulfills the `nresamp`. It works just together `time_lim` argument. See Details.
`time_lim`	If `na_action = "force"`, it defines the maximum time (in seconds) that the function will last resampling until it achieves a result output with no-`NaN` rows.

Details

There are 2 ways to specify the main input arguments (related to the size and timing of the mark-recapture): (1) in the classical way, i.e. by defining `L1`, `L2`, `T1` and `T2` as `numeric` vectors as indicated in the grotag documentation or (2) through a `data.frame` indicated in the `input.data` argument. In the latter case, the arguments `L1`, `L2`, `T1` and `T2` must be 1-length `character` vectors and they will serve to indicate the column names of the corresponding variables. If only one value is specified for `input.data` and any of the other arguments is NULL, a default name equal to the variable name will be assigned (e.g. `L1 <- “L1”`).

`na_action = "force"` should be used carefully, as it is not always due to bootstrap data selection factors, but also to an inadequate selection of the estimation parameters that the `NA` values are obtained. Also, the search time may depend on the size of the input set, if you have many thousands of individuals or if (in addition) the value of `nresamp` is high, it is possible that the function will take a long time before obtaining complete results. `time_lim` avoids falling into an infinite loop by limiting the time used by this process to 5 minutes, but this value is referential and may be insufficient due to the factors mentioned above.

Value

A `data.frame` of fitted VBGF parameters (columns) by resampling (rows). It includes a column (`seed`) with seed values set prior to each resampling call.

References

- Efron, B., & Tibshirani, R., 1986. Bootstrap methods for standard errors, confidence intervals, and other measures of statistical accuracy. Statistical Science, 54-75.
- Francis, R.I.C.C., 1988. Maximum likelihood estimation of growth and growth variability from tagging data. New Zealand Journal of Marine and Freshwater Research, 22, p.42-51.
- Pauly, D., 1981. The relationship between gill surface area and growth performance in fish: a generalization of von Bertalanffy’s theory of growth. Meeresforsch. 28:205-211.
- Schwamborn, R., Mildenberger, T. K., & Taylor, M. H., 2019. Assessing sources of uncertainty in length-based estimates of body growth in populations of fishes and macroinvertebrates with bootstrapped ELEFAN. Ecological Modelling, 393, 37-51.

- Schwamborn, R. & Schwamborn, D. F. M. C. Growth and mortality of the endangered land crab *Cardisoma guanhumi* assessed through tagging with PITs and novel bootstrapped methods. Pan-American Journal of Aquatic Sciences, 16(1): 57-78.
- von Bertalanffy, L., 1938. A quantitative theory of organic growth. Human Biology 10, 181-213.

Examples

```
# Load example DB from fishmethods package
data(bonito, package = "fishmethods")

# Run the example cited on ?grotag
fishmethods::grotag(L1 = bonito$L1,
                    L2 = bonito$L2,
                    T1 = bonito$T1,
                    T2 = bonito$T2,
                    alpha   = 35, beta = 55,
                    design  = list(nu = 1, m = 1,p = 1, sea = 1),
                 stvalue = list(sigma = 0.9, nu = 0.4, m = -1, p = 0.2, u = 0.4, w = 0.4),
                    upper   = list(sigma = 5, nu = 1, m = 2, p = 0.5, u = 1, w = 1),
                    lower   = list(sigma = 0, nu = 0, m = -2, p = 0.0, u = 0, w = 0),
                    control = list(maxit = 1e4))

# Run the example using grotag_boot
res <- grotag_boot(L1 = bonito$L1,
                   L2 = bonito$L2,
                   T1 = bonito$T1,
                   T2 = bonito$T2,
                   alpha   = 35, beta = 55,
                   design  = list(nu = 1, m = 1,p = 1, sea = 1),
                 stvalue = list(sigma = 0.9, nu = 0.4, m = -1, p = 0.2, u = 0.4, w = 0.4),
                   upper   = list(sigma = 5, nu = 1, m = 2, p = 0.5, u = 1, w = 1),
                   lower   = list(sigma = 0, nu = 0, m = -2, p = 0.0, u = 0, w = 0),
                   control = list(maxit = 1e4),
                   nresamp = 3, na_action = "narm")

res
```

`lfqResample` *Resampling of length-frequency data*

Description

This function resamples the `lfq` data by sampling dates. Sampling is done in a non-parametric way following the relative frequencies of the original data, allowing for individual counts to be selected more than once (i.e. `replace = TRUE` in sample), and resulting in total counts (by sample) equal to the original data.

Usage

```
lfqResample(lfq)
```

Arguments

`lfq`	A length frequency object of the class `lfq`.

Value

A resampled version of the `lfq` class dataset.

Examples

```
# Load data
data("alba", package = "TropFishR")

# Resample lfq data
alba_p <- lfqResample(lfq = alba)

# Side-by-side plot
op <- par(no.readonly = TRUE)
par(mfcol = c(2, 1), mar = c(4, 4, 2, 1))

# Original
plot(x = TropFishR::lfqRestructure(alba), Fname = "rcounts")
mtext("original", side=3, line=0.25)

# Resampled
plot(TropFishR::lfqRestructure(alba_p), Fname = "rcounts")
mtext("resampled", side=3, line=0.25)

par(op)
```

`LinfK_scatterhist` *Linf/K scatterplot of bootstrapping results*

Description

This function plots a scatterplot of von Bertalanffy growth function parameters K vs L_{inf} (“Kimura plot”) with histograms.

Usage

```
LinfK_scatterhist(
  res,
  Linf.breaks = "Sturges",
  K.breaks = "Sturges",
  gridsize = rep(151, 2),
  H = NULL,
```

```
  shading = TRUE,
  shading.cols = NULL,
  dens.contour = TRUE,
  probs = c(25, 50, 75, 95),
  phi.contour = TRUE,
  phi.levels = NULL,
  phi.contour.col = 8,
  phi.contour.lty = 2,
  phi.contour.lwd = 1,
  phi.contour.labcex = 0.75,
  pt.pch = 16,
  pt.col = adjustcolor(1, 0.25),
  pt.cex = 0.5,
  pt.bg = 4,
  xlab = expression(italic("L")[infinity]),
  ylab = expression(italic("K")),
  ...
)
```

Arguments

`res`	Object of class `data.frame`, `tbl_df`, `lfqBoot` or `grotagBoot`.
`Linf.breaks`, `K.breaks`	Arguments passed to hist function to compute the breakpoints (argument `breaks`) for Linf and K histograms respectively.
`gridsize`	`numeric` 2-length vector specifying the resolution of the grid.
`H`	Plug-in bandwidth object from `Hpi` (Default: `H = ks::Hpi(res[,c("Linf", "K")]))`
`shading`	`logical`. Do you want to colour 2D field of density estimates? (Default `TRUE`)
`shading.cols`	Colors or color palette used for background shading of 2D field of density estimates. No considered if `shading = FALSE`.
`dens.contour`	`logical`. Do you want to add contour lines? (`TRUE` by default).
`probs`	`numeric` Density probability cutoffs (in by contours. By default `probs = c(25, 50, 75, 95)` and not considered if `dens.contour = FALSE`.
`phi.contour`	`logical`. Do you want to display phi prime isolines? (`FALSE` by default)
`phi.levels`	`numeric` vector that controls Phi prime values. Omitted if `phi.contour = FALSE` and if `NULL` values will be chosen automatically by the contour function.
`phi.contour.col`, `phi.contour.lty`, `phi.contour.lwd`, `phi.contour.labcex`	Extra arguments used to control the color, line type, line width and labels size of the phi prime contour isolines.
`pt.pch`, `pt.col`, `pt.cex`, `pt.bg`	Extra arguments to control type, color, size and background color of resampling points.
`xlab`, `ylab`	Labels for X and Y axis respectively.
`...`	Extra arguments passed to main plot function.

Details

Isolines of growth performance (Phi') can be plotted, as well as bivariate 95 used for plotting is usually the result of a bootstrapped growth analysis (i.e. a `lfqBoot` object generated by **fishboot** functions such as ELEFAN_SA_boot, ELEFAN_GA_boot, grotag_boot, or grolenage_boot).

If NULL, it will be defined as `colorRampPalette(c("white", blues9))(1e3)`.

Value

This function returns just the described plot.

Examples

```
data(alba_boot) # lfqBoot object
LinfK_scatterhist(res = alba_boot)

data(bonito_boot) # grotagBoot object
LinfK_scatterhist(res = bonito_boot)
```

`univariate_density` *Univariate kernel density estimate plot of VBGF parameter from bootstrapping results*

Description

This function plots a set of vertical plots with kernel density distributions for univariate posterior distributions of the VBGF growth parameters L_{inf}, K, and Phi'. The 95 interval and the most likely optimum fit estimate are shown for each parameter.

Usage

```
univariate_density(
  res,
  CI = 95,
  use_hist = FALSE,
  nbreaks = 10,
  mar = c(1.5, 2, 2, 0),
  oma = c(1.5, 0, 0, 0.5),
  mgp = c(2, 0.5, 0),
  tcl = -0.25,
  cex = 1,
  ...
)
```

Arguments

`res`	Object of class `lfqBoot`.
`CI`	`numeric`. Confidence interval in % (default: 95).
`use_hist`	`logical` Plot histogram in addition to smoothed kernel density.
`nbreaks`	`numeric` vector specifying the number of breaks in the histogram.
`mar`, `oma`, `mgp`, `tcl`, `cex`, `...`	Additional arguments passed to par.

Details

This function used the function kde to obtain kernel density estimates for the VBGF growth parameters L_{inf}, K, and Phi'. The 95 posterior distribution) and the most likely optimum fit estimate (i.e., the mode of each posterior distribution) are then plotted inside each vertical plot. The input used for plotting is usually the result of a bootstrapped growth analysis (i.e. a `lfqBoot` object generated by **fishboot** functions such as ELEFAN_SA_boot, ELEFAN_GA_boot, grotag_boot, or grolenage_boot).

Value

This function returns just the described plot.

Examples

```
data(alba_boot)
univariate_density(alba_boot)
```

`vbgfCI_time` *VBGF plot and CI*

Description

This function plots a swarm of von Bertalanffy growth functions (VBGF), as length vs time (age) curves, based on the results of bootstrap runs, with confidence intervals (CI).

Usage

```
vbgfCI_time(
  res,
  CI = 95,
  agemax = NULL,
  plot = TRUE,
  add_legend = TRUE,
  add_max_dens_legend = TRUE,
  xlab = "Relative time",
  ylab = "Length",
  perm.col = adjustcolor("grey50", 0.1),
```

```
  perm.lwd = 1,
  ci.col = "black",
  ci.lty = 2,
  ci.lwd = 1,
  maxd.col = "black",
  maxd.lty = 1,
  maxd.lwd = 2,
  ...
)
```

Arguments

`res`	Object with L_{inf}, K and t_0, it could be a `data.frame`, a `tbl_df`, a `list`, `grotagBoot` or a `lfqBoot` object. See Details.
`CI`	`numeric`. Confidence interval in % (default: 95).
`agemax`	`numeric` values indicating the maximum number of years to project.
`plot`	`logical`. If `TRUE` (default), a plot is returned, otherwise just a list with levels `limCI`, `inCI`, `density` and `max_dens`. See Value for a detailed description of each one.
`add_legend`	`logical`. Should CI and max. density legend be added (Default: ‘add_legend = TRUE‘).
`add_max_dens_legend`	logical. Should maximum density line be added (Default: ‘add_max_dens_legend = TRUE‘).
`xlab`	Label for x-axis
`ylab`	Label for y-axis
`perm.col`, `perm.lwd`	Color and width for each resample estimate line.
`ci.col`, `ci.lty`, `ci.lwd`	Color, type and width for CI line.
`maxd.col`, `maxd.lty`, `maxd.lwd`	Color, type and width for maximum density line.
`...`	Extra arguments passed to the main plot function.

Details

Each thin grey line represents the output of a single bootstrap run. The most likely optimum (i.e., the mode of the posterior distribution) is shown as a thick black line. The dashed lines show the upper and lower limits of the 95 confidence envelope. The input used for plotting is usually the result of a bootstrapped growth analysis (i.e. a `lfqBoot` object generated by **fishboot** functions such as ELEFAN_SA_boot, ELEFAN_GA_boot, grotag_boot, or grolenage_boot).

Value

A `list` containing:

`$limCI` A `data.frame` with CI limits by time.

- `$inCI` A `data.frame` with logical values defining whether bootstrapping samples are within each of the defined CIs.
- `$density` The multivariate kernel density estimates for each sample.
- `$max_dens` A `list` with the VBGF parameter combination having the maximum density estimate.

Examples

```
data(alba_boot) # lfqBoot object
vbgfCI_time(res = alba_boot)

vbgfCI_time(res = alba_boot, CI = c(50, 95),
            ci.col = c("red", "orange"))

data(bonito_boot) # grotagBoot object
LinfK_scatterhist(res = bonito_boot)
```

Index