Package ‘halk’

February 13, 2023

Type Package

Title Methods to Create Hierarchical Age Length Keys for Age Assignment

Version 0.0.1

Maintainer Paul Frater <paul.frater@wisconsin.gov>

Description Provides methods for implementing hierarchical age length keys to estimate fish ages from lengths using data borrowing. Users can create hierarchical age length keys and use them to assign ages given length.

Depends R (>= 2.10)

Imports rlang (>= 1.0.4), tibble (>= 3.1.7), dplyr (>= 1.0.9), tidyr (>= 1.2.0), purrr, tidyselect, magrittr

Suggests testthat (>= 3.1.4), knitr, rmarkdown

Config/testthat/edition 3

License MIT + file LICENSE

Encoding UTF-8

LazyData true

RoxygenNote 7.1.2

VignetteBuilder knitr

NeedsCompilation no

Author Paul Frater [aut, cre] (<https://orcid.org/0000-0002-7237-6563>)

Repository CRAN

Date/Publication 2023-02-13 08:40:02 UTC

R topics documented:

adjust_ages ......................................................... 2
ages_as_ordered .................................................. 3
assign_ages ......................................................... 3
assign_alk_attributes ........................................... 4
bin_lengths ......................................................... 5
adjust_ages

Adjusts data to account for plus group or minimum age

Description

These functions perform two tasks. It lumps all ages greater than the plus group into that age, and it filters data only to those greater than or equal to the minimum age. `adjust_plus_min_ages` works on a vector whereas `adjust_plus_min_ages_df` works on a data.frame.

Usage

```r
adjust_plus_min_ages_df(data, minage = NULL, pls_grp = NULL)

adjust_plus_min_ages(age_vec, minage = NULL, pls_grp = NULL)
```

Arguments

- **data**: Data with age as a column, or a numeric vector of ages
- **minage**: Numeric. The minimum age; everything else is excluded
- **pls_grp**: Numeric. The plus group; all ages older will be lumped into this group
- **age_vec**: A vector of ages

Value

A data.frame similar to `data`, but with ages less than `minage` excluded and ages `>=` `plus_group` aggregated into that age
ages_as_ordered

Convert ages from/to ordered factor

Description
In order for the machine learning models to properly predict ages, the known ages should be converted to an ordered factor during model fitting. This will ensure that the predict.* functions return age values that actually make sense.

Usage

ages_as_ordered_factor(data, age_col = "age")
ages_as_integer(data, age_col = "est.age")

Arguments

data: A data.frame with a column corresponding to age_col or a vector of values
age_col: Character. The name of the column that contains ages

Value
A data.frame with the values in age_col converted to an ordered factor

assign_ages

Assign ages to non-aged data based on a fitted age model

Description
Assign ages to non-aged data based on a fitted age model

Usage

assign_ages(newdata, object, ...)

Arguments

newdata: A vector or data.frame with size/length measurements
object: An object of class "alk", "alk_fit", "rf_fit", or "gbm_fit" as produced by fit_age_model
...: Additional parameters to pass to the S3 object methods

Value
A data.frame the same as newdata, but with ages assigned based on the model provided in object
Examples

```r
spp_alk <- fit_age_model(spp_data, levels = "spp")
spp_est_ages <- assign_ages(spp_data, spp_alk)
```

---

**assign_alk_attributes**  
Assign associated age-length key attributes to a data.frame

**Description**

This is just a helper function to assign the needed attributes and classes to a data.frame that is produced by either `make_alk` or `fit_age_model`.

**Usage**

```r
assign_alk_attributes(
  data,  
  size_col = "length",  
  age_col = "age",  
  autobin = TRUE,  
  size_bin = 1,  
  min_age = NULL,  
  plus_group = NULL,  
  alk_n = NULL,  
  classes = "alk",  
  dnorm_params = NULL,  
  levels = NULL
)
```

**Arguments**

- **data**: A data.frame
- **size_col**: Character. Name of the column representing sizes
- **age_col**: Character. Name of the column representing ages
- **autobin**: Logical to set the attribute of autobin
- **size_bin**: Numeric. What is the width of size bins
- **min_age**: Numeric. The minimum age that was included in the alk
- **plus_group**: Numeric. The age that represents the plus group
- **alk_n**: Numeric. The number of samples that went into creating the alk
- **classes**: Character. The class that should get prepended to the data.frame class(es)
- **dnorm_params**: The value of parameters that went into creating the normal distributions on the age groups
- **levels**: Character vector of the levels used. This creates the "levels" attribute if present

**Value**

A data.frame with associated attributes assigned
**bin_lengths**

Convert a vector of lengths into binned values

**Description**

This will take a vector of numeric values and bin them according to the value specified in `binwidth`

**Usage**

`bin_lengths(x, binwidth, include_upper = FALSE, ...)`

**Arguments**

- `x` Numeric vector of values
- `binwidth` Numeric vector specifying how wide the length bins should be
- `include_upper` Logical. Append the upper value of the bin and return the length range as a character string (TRUE), or return the lower value as numeric (FALSE, default)
- `...` Additional arguments passed onto `cut`

**Value**

A vector of values the same length as `x`, but binned to the values according to `binwidth`

**Examples**

`bin_lengths(length_data$length, binwidth = 2)`

---

**calc_mse**

Calculate mean-squared-error (MSE) and root mean-squared-error (RMSE) of estimated ages

**Description**

These functions will calculate MSE and RMSE for estimated ages produced by `assign_ages`. Output is specific to each level used by the age-length key to assign ages

**Usage**

`calc_mse(data, age_col = "age")`

`calc_rmse(data, age_col = "age")`

**Arguments**

- `data` A data.frame as created by `assign_ages`
- `age_col` Character. Name of the age column in `data`
Value

Numeric value for estimated ages with no levels or a data.frame with a MSE or RMSE value for each level used to fit ages

Examples

```r
wa_e_data <- spp_data[spp_data$spp == 'walleye', ]
alk <- make_alk(wae_data)
wa_e_est_age <- assign_ages(wae_data, alk)
calc_mse(wae_est_age)
calc_rmse(wae_est_age)
```

**calc_mse_**  
*Backend helper function to compute MSE or RMSE*

Description

This function is the engine for `calc_mse` and `calc_rmse`. It was only created to remove the `root` argument from the user in the main `calc_mse` function

Usage

```r
calc_mse_(data, age_col = 'age', root = FALSE)
```

Arguments

- **data**  
  A data.frame as created by `assign_ages`
- **age_col**  
  Character. Name of the age column in `data`
- **root**  
  Logical. compute MSE (FALSE, default) or RMSE (TRUE)

**calc_stat_scores**  
*Compute test statistics for comparing actual and estimated ages*

Description

Using these functions you can compute either a Kolmogorov-Smirnov (KS) statistic or a Chi-squared test statistic to compare estimated ages to actual ages. See details for how each test works and what is reported.
Usage

calc_ks_score(
  data,
  summary_fun = mean,
  age_col = "age",
  suppress_warnings = TRUE,
  return_val = "statistic",
  ...  
)

calc_chi_score(
  data,
  age_col = "age",
  suppress_warnings = TRUE,
  return_val = "statistic",
  ...  
)

Arguments

data A data.frame containing estimated ages as returned by assign_ages
summary_fun Function used to compute summary statistics for calc_ks_score for each age group (default is mean)
age_col Character string specifying the name of the age column
suppress_warnings Logical. Should any warnings from the function call to ks.test or chisq.test be suppressed (TRUE, the default)
return_val Character. The name of the object to return from the given test
...

Details

The KS test compares length distributions for each age class from known ages against that of estimated ages computed by the assign_ages function. The output is a summary value of the test statistics as specified by summary_fun.

The calc_chi_score function performs a Chi-square test (using the chisq.test function) on the number of estimated and actual ages for each age group.

Value

A numeric value for each level that was used in the model to assign ages

Examples

halk <- make_halk(spp_data, levels = c("spp"))
newdat <- laa_data
check_model_type <- "bluegill"
pred_ages <- assign_ages(newdat, halk)
calc_ks_score(pred_ages)
calc_chi_score(pred_ages)

check_agelen_data
Check for age/length data in the data being estimated or predicted

Description

These are just simple helper functions used within other functions that check to make sure that ages
and lengths are present in the data and stop the function call if they are missing

Usage

check_age_data(data, age_col)
check_length_data(data, size_col)

Arguments

data A data.frame
age_col Character. The column name for the age column in data
size_col Character. The column name for the size column in data

Value

NULL. An error will be called if age/length data is missing

check_model_type
Check the model type and return standardized version

Description

This is a non-exported function to check whether the model type specified is available and return a
standardized version of the model name. This standardized version will then feed into a S3 method
for the given model.

Usage

check_model_type(model)

Arguments

model A character string naming the model

Value

A standardized version of the model name, or an error if model doesn't exist yet
fit_age_model

Fit a model (or age-length key) that will be used to estimate ages

Description

This function will create the appropriate model (or age-length key) that can then be used to predict ages based on length (or sizes, more generally). The model types that are currently available are 'age-length key', a 'smart age-length key' (see Details), random forest, and gradient boosting machine.

Usage

fit_age_model(
  data,
  model = "halk",
  levels = NULL,
  age_col = "age",
  size_col = "length",
  ...
)

Arguments

data An object of class data.frame
model Character. The model type to fit. Options are 'alk', 'halk', 'rf' (for random forest), 'gbm' (for gradient boosting machine)
levels Character vector. The levels that the age estimating model will fit to. Each level specified must correspond to a column in data. For models 'alk' and 'halk' an age-length key gets created at each level. For 'rf' and 'gbm' the levels get converted to a formula (see Details)
age_col Character. The name of the column in data that contains ages
size_col Character. The name of the column in data that contains sizes
... Additional arguments passed onto the various methods

Details

fit_age_model will take the provided length-at-age data and create a model for predicting ages based on length (or whatever is specific as the size_col argument). The different methods for doing this are:

- alk – a basic age-length key
- smart_alk – a more advanced method that creates an age-length key from data when enough is available, but borrows data from elsewhere when it’s not. This option requires that you specify levels at which to fit each alk (i.e. species, county, waterbody, year, etc.). As an example, if species, county, county, and waterbody are each specified as levels, then fit_age_model will create age-length keys for each waterbody, for each county, and ultimately for each species. The highest level most general ALK is created at the first level specified.
• rf – a random forest model that includes any levels specified in `levels`
• gbm – a gradient boosting machine model that includes any levels specified in `levels`

For the rf and gbm models, the strings specified in `levels` get converted to and added to the formula passed those models. For example, if `levels = c("spp", "county", "waterbody")`, then the resulting formula in the call to the model would be `age ~ length + spp + county + waterbody`.

**Value**
An object of the appropriate model type according to what was provided by the `model` argument

**Examples**
```r
spp_halk <- fit_age_model(spp_data, levels = "spp")
```

---

**Description**
Simple age-structured population data with age and length records for each individual. `laa_data` represents a well-sampled age-length dataset, whereas `laa_data_low_n` is one with few total samples, `laa_data_low_age_n` is one with few samples in some ages, and `laa_data_few_ages` is a dataset with few age groups sampled. Species specific datasets are similar, but with the prefix `laa_` replaced by `spp_`. These datasets contain species specific length-at-age data

**Usage**
```r
laa_data
laa_data_low_n
laa_data_low_age_n
laa_data_few_ages
spp_data
spp_data_low_n
spp_data_low_age_n
spp_data_few_ages
```
length_data

Format

## 'laa_data' A data.frame with 244 rows and 2 columns:

spp  Species, only applicable for spp_data_* data.frames
age  Age of individual
length  Length of individual (arbitrary units)

## 'laa_data_low_n' A data.frame with 27 rows and 2 columns:
## 'laa_data_low_age_n' A data.frame with 74 rows and 2 columns:
## 'laa_data_few_ages' A data.frame with 49 rows and 2 columns:
## 'spp_data' A data.frame with 1022 rows and 3 columns:
## 'spp_data_low_n' A data.frame with 87 rows and 3 columns:
## 'spp_data_low_age_n' A data.frame with 160 rows and 3 columns:
## 'spp_data_few_ages' A data.frame with 261 rows and 3 columns:

---

**length_data**

**Example length data**

---

Description

Simple vector and data.frame containing length measurements. These are used in examples for functions that assign ages.

Usage

length_data
spp_length_data

Format

## length data A data.frame with one column and 244 rows

spp  Species, only in spp_length_data
length  Length of individual (arbitrary units)

## 'spp_length_data' A data.frame with 1022 rows and 2 columns:
**make_alk**

*Make an age-length key out of length-at-age data*

**Description**

Make an age-length key out of length-at-age data

**Usage**

```r
make_alk(
  laa_data,
  sizecol = "length",
  autobin = TRUE,
  binwidth = 1,
  agecol = "age",
  min_age = NULL,
  plus_group = NULL,
  numcol = NULL,
  min_age_sample_size = 5,
  min_total_sample_size = min_age_sample_size * min_age_groups,
  min_age_groups = 5,
  warnings = TRUE
)
```

**Arguments**

- **laa_data**: A data.frame with length-at-age data
- **sizecol**: Character string naming the column that holds size data
- **autobin**: Logical. Should the function automatically assign length bins (default is TRUE)
- **binwidth**: Numeric. If `autobin = TRUE` this is the width for the size bins
- **agecol**: Character string naming the column that holds age data
- **min_age**: Numeric. All ages less than this value will not be used in ALK
- **plus_group**: Numeric value of the oldest age to include in the ALK. All older individuals will be included in this plus group
- **numcol**: Character string naming the column that holds numbers data
- **min_age_sample_size**: Only applicable to alk models. The minimum number of samples that must be in each age group in order to create an alk
- **min_total_sample_size**: Only applicable to alk models. The minimum number of samples that must be in data (within level) in order to create an alk
- **min_age_groups**: Only applicable to alk models. The minimum number of age groups that must be in data (within level) in order to create an alk
- **warnings**: Logical. Display warnings (TRUE, default)
Description

This function creates a hierarchically nested age-length key that can be used to estimate age of an organism based on proportion of sampled organisms in each age group.

Usage

make_halk(data, levels, age_col = "age", size_col = "length", ...)

Arguments

data     A data.frame with age and size samples
levels   Character vector specifying the levels for HALK creation
age_col  Optional. String of the column name in data housing age data
size_col Optional. String of the column name in data housing size data
...      Additional arguments passed to make_alk

Value

A tibble with columns for each level and a column called alk that houses the age-length key for that particular level

Examples

make_halk(spp_data, levels = "spp")
min_samples

Count number of length-at-age samples or age groups at each level and return those with greater than equal to the minimum desired number

Description

These are helper shortcut functions to determine if data meet the minimum desired number of age groups and/or sample sizes.

Usage

min_count_laa_data(
  data,
  sub_levels = NULL,
  min_age_sample_size = NULL,
  min_total_sample_size = NULL,
  min_age_groups = NULL
)

min_age_groups(data, sub_levels = NULL, min_age_grps)

Arguments

data Data.frame with length-at-age data
sub_levels The levels at which to check
min_age_sample_size Only applicable to alk models. The minimum number of samples that must be in each age group in order to create an alk
min_total_sample_size Only applicable to alk models. The minimum number of samples that must be in data (within level) in order to create an alk
min_age_groups Only applicable to alk models. The minimum number of age groups that must be in data (within level) in order to create an alk
min_age_grps The minimum number of age groups that must be present in data to create an ALK

Value

A data.frame just like data, but with samples excluded that don’t meet the required number of samples in min_sample_size
rename_laa_cols

**Description**

Simple helper function to rename size and age column names to age and length

**Usage**

```r
class(rename_laa_cols(  
  data,  
  size_col = "length",  
  age_col = "age",  
  num_col = NULL,  
  goback = FALSE  
))
```

**Arguments**

- `data` Any data.frame with some columns representing age and size
- `size_col` Character. The name of the column containing sizes
- `age_col` Character. The name of the column containing ages
- `num_col` Character. The name of the column containing number of individuals
- `goback` Logical. Reverse names once they’ve already been renamed

**Value**

A data.frame the same as `data`, but with names changed

---

spp_levels

**Description**

Check for species in columns and/or levels and add to levels if present

**Description**

These helper functions just check to see if a species column exists in the data (designated as 'spp' or 'species'). If one of those columns exists, but the column name is not in the levels argument it will get added to levels.
Usage

is_spp_in_levels(levels)

is_spp_in_data(data)

spp_level(levels)

rm_spp_level(levels)

add_spp_level(data, levels)

Arguments

levels The levels argument passed from fit_age_model
data A data.frame with length-at-age data

Value

A character vector of levels possibly with ‘spp’ or ‘species’ added

wb_spp_data

Separate species, county, waterbody example length-at-age and length data

Description

Simple age-structured population with age and/or length records, but expanded across multiple counties and waterbodies for tests and examples in fit_age_model used with levels.

Usage

wb_spp_laa_data

wb_spp_length_data

Format

## ‘wb_spp_laa_data’ A data.frame with 36,849 records and 5 columns

spp Species
county Arbitrary example county name
waterbody Arbitrary example waterbody name nested within county
age Age of individual, only in wb_spp_laa_data
length Length of individual (arbitrary units)

An object of class tbl_df (inherits from tbl, data.frame) with 9182 rows and 4 columns.
Index

* **datasets**
  - laa_data, 10
  - length_data, 11
  - wb_spp_data, 16

  - add_spp_level (spp_levels), 15
  - adjust_ages, 2
  - adjust_plus_min_ages (adjust_ages), 2
  - adjust_plus_min_ages_df (adjust_ages), 2
  - ages_as_integer (ages_as_ordered), 3
  - ages_as_ordered, 3
  - ages_as_ordered_factor
    - (ages_as_ordered), 3
  - assign_ages, 3, 5–7
  - assign_alk_attributes, 4

  - bin_lengths, 5

  - calc_chi_score (calc_stat_scores), 6
  - calc_ks_score (calc_stat_scores), 6
  - calc_mse, 5, 6
  - calc_mse_, 6
  - calc_rmse, 6
  - calc_rmse (calc_mse), 5
  - calc_stat_scores, 6
  - check_age_data (check_agelen_data), 8
  - check_agelen_data, 8
  - check_length_data (check_agelen_data), 8
  - check_model_type, 8
  - chisq.test, 7
  - cut, 5

  - fit_age_model, 3, 4, 9, 16

  - is_spp_in_data (spp_levels), 15
  - is_spp_in_levels (spp_levels), 15

  - laa_data, 10
  - laa_data_few_ages (laa_data), 10
  - laa_data_low_age_n (laa_data), 10
  - laa_data_low_n (laa_data), 10

  - length_data, 11
  - make_alk, 4, 12, 13
  - make_halk, 13
  - min_age_groups (min_samples), 14
  - min_count_laa_data (min_samples), 14
  - min_samples, 14

  - rename_laa_cols, 15
  - rm_spp_level (spp_levels), 15

  - spp_data (laa_data), 10
  - spp_data_few_ages (laa_data), 10
  - spp_data_low_age_n (laa_data), 10
  - spp_data_low_n (laa_data), 10
  - spp_length_data (length_data), 11
  - spp_level (spp_levels), 15
  - spp_levels, 15

  - tibble, 13

  - wb_spp_data, 16
  - wb_spp_laa_data (wb_spp_data), 16
  - wb_spp_length_data (wb_spp_data), 16