Package: fbi (via r-universe)

August 30, 2024

Title Finnish Biodiversity Indicators

Version 0.11.22.9000

Description Finnish biodiversity indicators is a service providing time series of abundance indices and related metrics for Finland. The input data for the indices are provided by the Finnish Biodiversity Information Facility.

Depends R (>= 3.5.0)

Imports arm, blob, config, dbplyr, dplyr, finbif, ggplot2, grDevices, lme4, lubridate, pool, rbms, rtrim, stats, svglite, yaml

Remotes RetoSchmucki/rbms

Suggests knitr, rmarkdown, tinytest, DBI, RPostgres

License MIT + file LICENSE

URL `https://github.com/luomus/fin-biodiv-indicators`, `https://indicators.laji.fi`

VignetteBuilder knitr

BugReports `https://github.com/luomus/fin-biodiv-indicators/issues`

Encoding UTF-8

Roxygen list(markdown = TRUE)

RoxygenNote 7.3.2

Repository https://luomus.r-universe.dev

RemoteUrl https://github.com/luomus/fin-biodiv-indicators

RemoteRef dev

RemoteSha 1db275f77c321e88a0a66bc9056d1a310731b146

Contents

`check_input` *Check input*

Description

Check indicator inputs

Usage

```
check_input(index, model, taxon)
```

Arguments

`index`	Character. Which index?
`model`	Character. Which model?
`taxon`	Character. Which taxon?

`clean_cache` *Clean cache*

Description

Remove unneeded tables and rows from database cache.

Usage

```
clean_cache(db)
```

Arguments

`db`	Connection. Database cache.

`combine_with_surveys` *Combine with surveys*

Description

Combine count data with survey data

Usage

```
combine_with_surveys(counts, surveys, ...)
```

Arguments

`counts`	Count data.
`surveys`	Survey data.
`...`	Additional arguments.

Details

This function combines `counts` and `surveys` data. It performs an inner join of `counts` on `surveys` by `document_id`. The function assumes that both `counts` and `surveys` data include `document_id`.

`format_date` *Format date*

Description

Combine year, month, day of survey into a single date string

Usage

```
format_date(surveys, ...)
```

Arguments

`surveys`	Survey data.
`...`	Additional arguments.

Details

This function combines survey `year`, `month` and `day` into a character string with `-` as a separator. The function assumes that survey data includes `year`, `month` and `day`.

`get_indices` *Get indices*

Description

Get indices from a configuration file.

Usage

```
get_indices(file = Sys.getenv("R_CONFIG_FILE"))
```

Arguments

`file`	Configuration file.

`get_output` *Get output*

Description

Get serialized indicator outputs

Usage

```
get_output(output, index, model, taxon, region, db)
```

Arguments

`output`	Character. Which type of output?
`index`	Character. Update which index?
`model`	Character. Which model to use?
`taxon`	Character. Which taxon?
`region`	Character. Which region?
`db`	Connection. Database from which to get output.

`pick_first_survey_in_fortnight` *Pick first survey in fortnight*

Description

Pick first survey in each fortnight discarding subsequent surveys

Usage

```
pick_first_survey_in_fortnight(surveys, ...)
```

Arguments

`surveys`	Survey data.
`...`	Additional arguments.

Details

This function groups surveys by `location_id`, `year` and `fortnight`then orders them by date. All but the first survey in each group is removed. If two or more surveys share the same date and `location_id` then one is picked at random and the rest are removed. The function assumes that the `surveys` data includes `day`, and `year` (as integers) and `location_id`, and has been processed by the function `require_seven_fortnights`.

`pick_first_survey_in_winter` *Pick first survey in winter*

Description

Pick first winter survey in each year discarding subsequent surveys

Usage

```
pick_first_survey_in_winter(surveys, ...)
```

Arguments

`surveys`	Survey data.
`...`	Additional arguments.

Details

This function moves surveys occurring in December ahead one `year`. This enables all December surveys to be grouped with subsequent surveys occurring in the January of the same winter. Surveys are then grouped by `location_id` and `year` and then ordered by date. Then all but the first survey in each group is removed. If two or more surveys share the same date and `location_id` then one is picked at random and the rest are removed. This function works on the assumption that surveys are in winter from December to January and that the `surveys` data includes `day`, `month` and `year` (as integers) and `location_id`.

`pick_first_survey_in_year` *Pick first survey in year*

Description

Pick first survey in each year discarding subsequent surveys

Usage

```
pick_first_survey_in_year(surveys, ...)
```

Arguments

`surveys`	Survey data.
`...`	Additional arguments.

Details

This function groups surveys by `location_id` and `year` then orders them by date. All but the first survey in each group is removed. If two or more surveys share the same date and `location_id` then one is picked at random and the rest are removed. The function assumes that the `surveys` data includes `day`, `month` and `year` (as integers) and `location_id`.

`process_funs` *Process functions*

Description

Functions to process indicator input data.

Usage

```
process_funs()
```

`remove_all_zero_locations` *Remove all-zero locations*

Description

Discard locations where taxa always had zero abundance

Usage

```
remove_all_zero_locations(counts, ...)
```

Arguments

`counts`	Count data.
`...`	Additional arguments.

Details

This function groups `counts` by `location_id` and then removes all `counts` includes `location_id` and `abundance`.

`require_minimum_gaps` *Require minimum gaps*

Description

Remove survey site-years that too many or to large sampling gaps.

Usage

```
require_minimum_gaps(surveys, ...)
```

Arguments

`surveys`	Survey data.
`...`	Additional arguments.

Details

This function groups surveys data by `location_id` and `year`. It then removes groups where the survey period has too many or too large sampling gaps. Where too many is defined as a total gap length over the `year` of 21 days and too large is any single sampling gap of more than 7 days. The function expects the `surveys` data to have at least `location_id`, `year`, `ordinal_day_start` and `ordinal_day_end`.

`require_minimum_weeks` *Require minimum weeks*

Description

Remove survey site-years from a region covering less than a minimum number of weeks.

Usage

```
require_minimum_weeks(surveys, ...)
```

Arguments

`surveys`	Survey data.
`...`	Additional arguments.

Details

This function groups surveys data by `location_id` and `year`. It then removes groups where the survey period is less than a minimum number of weeks for a given `region`. It expects the `surveys` data to have at least `location_id`, `year`, `region`, `ordinal_day_start` and `ordinal_day_end`.

`require_seven_fortnights` *Require seven fortnights*

Description

Divide year into approximate 2 week blocks, selecting blocks 10-16 and discarding locations without a survey in each remaining block

Usage

```
require_seven_fortnights(surveys, ...)
```

Arguments

`surveys`	Survey data.
`...`	Additional arguments.

Details

This function assigns each survey to an approximate fortnight. A fortnight is defined as all the days before the 16th `day` of each `month` and all the days after the 15th `day` of each `month`. Then all the surveys falling outside of the date range of the seven fortnights from the second fortnight of May to the second fortnight of August are removed. Surveys are then grouped by `location_id` and `year` and all surveys belonging to groups that do not have at least one survey occurring in each of the seven remaining fortnights are discarded. The function assumes that the `surveys` data has `day`, `month` and `year` (as integers) and `location_id`.

`require_two_years` *Require at least two years*

Description

Discard locations with less than two survey years

Usage

```
require_two_years(surveys, ...)
```

Arguments

`surveys`	Survey data.
`...`	Additional arguments.

Details

This function groups `surveys` by `location_id` and then removes all surveys for locations that do not have data in more than one `year`. The function assumes that `surveys` has data for `location_id` and `year`.

`set_start_year` *Set start year*

Description

Discard counts from years before the start year

Usage

```
set_start_year(counts, taxon, ...)
```

Arguments

`counts`	Count data.
`taxon`	Taxon configuration.
`...`	Additional arguments.

Details

This function sets a start year for a taxon `counts`. If a variable `start_year` has been configured for the given taxon all count data prior to the `start_year` is removed.

`sum_by_event` *Sum by event*

Description

Sum the counts over the surveys or taxa in each year

Usage

```
sum_by_event(counts, ...)
```

Arguments

`counts`	Count data.
`...`	Additional arguments.

Details

This functions groups count data by `location_id` and `year`. If multiple taxa `counts` are input then data is also grouped by taxa. Counts are then summed across the survey events at the locations and years.

`sum_over_sections` *Sum over sections*

Description

Sum counts over the sections of surveys

Usage

```
sum_over_sections(counts, ...)
```

Arguments

`counts`	Count data.
`...`	Additional arguments.

Details

This functions groups count data by `document_id` (the IDs of the individual surveys). If multiple taxa `counts` are input then data is also grouped by taxa. Counts are then summed across survey sections when count data has been provided as surveys split into parts.

`update_data` *Update data*

Description

Update input data from FinBIF.

Usage

```
update_data(type, index, taxon, db, do_update = FALSE)
```

Arguments

`type`	Character. Which type of input data (e.g., surveys or counts)
`index`	Character. Update the data of which index?
`taxon`	Character. Update the data for which taxon? Ignored if `type = "surveys"`
`db`	Connection. Database in which to update the data from FinBIF.
`do_update`	Logical. Update data regardless of need.

update_index *Update index*

Description

Update index output data.

Usage

```
update_index(index, model, region, db)
```

Arguments

`index`	Character. Update which index?
`model`	Character. Which model to use?
`region`	Character. Which region?
`db`	Connection. Database in which to update index.

update_taxon_index *Update taxon index*

Description

Update the relative abundance index for a taxon.

Usage

```
update_taxon_index(index, model, taxon, db)
```

Arguments

`index`	Character. Update which index?
`model`	Character. Which model to use?
`taxon`	Character. Update the data for which taxa?
`db`	Connection. Database in which to update index.

zero_fill *Zero fill*

Description

Combine count data with survey data filling missing surveys in count data with zero counts.

Usage

```
zero_fill(counts, surveys, ...)
```

Arguments

`counts`	Count data.
`surveys`	Survey data.
`...`	Additional arguments.

Details

This function combines `counts` and `surveys` data. It performs a right outer join of `counts` on `surveys` by `document_id`. Then all surveys with no corresponding data for abundance are filled with zero. The function assumes that both `counts` and `surveys` data include `document_id` and that `counts` data includes `abundance`.

Index