Package ‘SimSurvey’

July 21, 2025

Type Package

Title Test Surveys by Simulating Spatially-Correlated Populations

Version 0.1.6

Maintainer Paul Regular <`Paul.Regular@dfo-mpo.gc.ca`>

Description Simulate age-structured populations that vary in space and time and explore the efficacy of a range of built-in or user-defined sampling protocols to reproduce the population parameters of the known population. (See Regular et al. (2020) <doi:10.1371/journal.pone.0232822> for more details).

Depends R (>= 3.5.0)

License GPL-3

Additional_repositories `https://inla.r-inla-download.org/R/stable/`

LazyData true

ByteCompile true

URL `https://paulregular.github.io/SimSurvey/`

BugReports `https://github.com/PaulRegular/SimSurvey/issues`

Imports sf, stars, data.table, magrittr, progress, doParallel, parallel, foreach, plotly, rlang, lifecycle

Suggests fields, rmarkdown, flexdashboard, shiny, crosstalk, htmltools, viridis, lme4, ggplot2, INLA, INLAspacetime, knitr, bezier

RoxygenNote 7.2.3

VignetteBuilder knitr

NeedsCompilation no

Author Paul Regular [aut, cre] (ORCID:
<`https://orcid.org/0000-0003-0318-2615`>),
Jonathan Babyn [ctb],
Greg Robertson [ctb]

Repository CRAN

Date/Publication 2023-09-19 10:50:10 UTC

Contents

bathy 2
convert_N 3
error_stats 3
expand_surveys 4
fibonacci 4
group_lengths 5
icc 5
land 6
make_grid 6
make_mesh 7
object_size 8
plot_trend 9
round_sim 11
run_strat 11
sim_abundance 12
sim_ays_covar 14
sim_ays_covar_spde 15
sim_distribution 16
sim_logistic 17
sim_nlf 18
sim_parabola 19
sim_R 20
sim_sets 22
sim_survey 23
sim_survey_parallel 25
sim_vonB 27
strat_data 28
strat_error 28
strat_means 29
survey_grid 30
survey_lite_mesh 30
survey_mesh 31
test_surveys 31
vis_sim 34

Index 36

`bathy` *Southern Newfoundland bathymetry*

Description

Southern Newfoundland bathymetry

Usage

```
bathy
```

Format

A stars object

Derived from data downloaded from http://www.gebco.net/. Details provided in the data-raw folder for this package.

`convert_N` *Convert abundance-at-age matrix to abundance-at-length*

Description

Function for converting abundance-at-age matrix to abundance-at-length given a length-age-key. Expects matrices to be named.

Usage

```
convert_N(N_at_age = NULL, lak = NULL)
```

Arguments

`N_at_age`	Abundance-at-age matrix
`lak`	Length-age-key (i.e. probability of being in a specific length group given age)

Value

Returns abundance-at-length matrix.

`error_stats` *Calculate common error statistics*

Description

Calculate common error statistics

Usage

```
error_stats(error)
```

Arguments

`error`	Vector of errors

Value

Returns a named vector of error statistics including mean error ("ME"), mean absolute error ("MAE"), mean squared error ("MSE") and root mean squared error ("RMSE")

expand_surveys *Set-up a series of surveys from all combinations of settings supplied*

Description

Function is simply a wrapper for expand.grid that adds a survey number to the returned object

Usage

```
expand_surveys(
  set_den = c(0.5, 1, 2, 5, 10)/1000,
  lengths_cap = c(5, 10, 20, 50, 100, 500, 1000),
  ages_cap = c(2, 5, 10, 20, 50)
)
```

Arguments

set_den	Vector of set densities (number of sets per grid unit squared)
lengths_cap	Vector of maximum number of lengths measured per set
ages_cap	Vector of maximum number of otoliths to collect per length group per division per year

Value

Returns a data.frame including all combinations of the supplied vectors.

fibonacci *Generate Fibonacci sequence*

Description

Generate Fibonacci sequence

Usage

```
fibonacci(from, to)
```

Arguments

from, to	Approximate start and end values of the sequence

Value

Returns a Fibonacci sequence as a vector.

Examples

```
fibonacci(2, 200)
```

`group_lengths` *Convert length to length group*

Description

Helper function for converting lengths to length groups (Note: this isn't a general function; the output midpoints defining the groups aligns with DFO specific method/labeling)

Usage

```
group_lengths(length, group)
```

Arguments

`length`	Interval from `findInterval`
`group`	Length group used to cut the length data

Value

Returns a vector indicating the mid-point of the length group.

`icc` *Calculate intraclass correlation*

Description

This is a simple function for calculating intraclass correlation. It uses `lmer` to run the formula described here: https://en.wikipedia.org/wiki/Intraclass_correlation

Usage

```
icc(x, group)
```

Arguments

`x`	Response variable
`group`	Group

Value

Returns estimate of intraclass correlation.

land *Southern Newfoundland coastline*

Description

Southern Newfoundland coastline

Usage

```
land
```

Format

A sf object (MULTIPOLYGON)

Derived from global administrative boundaries data (http://gadm.org/) downloaded using the `getData` function. Details provided in the data-raw folder for this package.

make_grid *Make a depth stratified survey grid*

Description

This function sets up a depth stratified survey grid. A simple gradient in depth is simulated using `stats::spline` (default) with a shallow portion, shelf and deep portion. Adding covariance to the depth simulation is an option.

Usage

```
make_grid(
  x_range = c(-140, 140),
  y_range = c(-140, 140),
  res = c(3.5, 3.5),
  shelf_depth = 200,
  shelf_width = 100,
  depth_range = c(0, 1000),
  n_div = 1,
  strat_breaks = seq(0, 1000, by = 40),
  strat_splits = 2,
  method = "spline"
)
```

Arguments

x_range	Range (min x, max x) in x dimension in km
y_range	Range (min y, max y) in y dimension in km
res	Resolution, in km, of the grid cells
shelf_depth	Approximate depth of the shelf in m
shelf_width	Approximate width of the shelf in km
depth_range	Range (min depth, max depth) in depth in m
n_div	Number of divisions to include
strat_breaks	Define strata given these depth breaks
strat_splits	Number of times to horizontally split strat (i.e. easy way to increase the number of strata)
method	Use a "spline", "loess" or "bezier" to generate a smooth gradient or simply use "linear" interpolation?

Value

Returns a stars object with 2 dimensions (x and y) and 4 attributes (depth, cell, division, strat).

See Also

survey_grid

Examples

```
r <- make_grid(res = c(10, 10))
plot(r)

p <- sf::st_as_sf(r["strat"], as_points = FALSE, merge = TRUE)
plot(p)
```

make_mesh *Make an R-INLA mesh based off a grid*

Description

This will make a mesh based off a given grid. Ideally the mesh construction and validation should be done by hand, but this exists for convenience. Meshes are used for sim_ays_covar_spde. The defaults are designed for the default grid. Just a basic interface between the grid and inla.mesh.2d.

Usage

```
make_mesh(
  grid = make_grid(),
  max.edge = 50,
  bound.outer = 150,
  cutoff = 10,
  offset = c(max.edge, bound.outer),
  ...
)
```

Arguments

`grid`	grid object to make a mesh of
`max.edge`	The largest allowed triangle edge length. One or two values. This is passed to inla.mesh.2d
`bound.outer`	The optional outer extension value given to offset.
`cutoff`	Minimum distance allowed between points
`offset`	The automatic extension distance given to inla.mesh.2d
`...`	Other options to pass to inla.mesh.2d

Value

Returns an object of class `inla.mesh`.

Examples

```
if (requireNamespace("INLA")) {
  basic_mesh <- make_mesh()
  plot(basic_mesh)
}
```

`object_size` *Print object size*

Description

A wrapper for `object.size` that prints in Mb by default

Usage

```
object_size(x, units = "Mb")
```

Arguments

`x`	an R object
`units`	the units to be used in printing the size

Value

Returns a character with the object size followed by the unit.

`plot_trend` *Simple plotting functions*

Description

These functions are simple plotting helpers to get some quick visuals of values produced by `sim_abundance`, `sim_distribution`, etc.

Usage

```
plot_trend(sim, sum_ages = sim$ages, col = viridis::viridis(1), ...)

plot_surface(sim, mat = "N", xlab = "Age", ylab = "Year", zlab = mat, ...)

plot_grid(grid, ...)

plot_distribution(
  sim,
  ages = sim$ages,
  years = sim$years,
  type = "contour",
  scale = "natural",
  ...
)

plot_survey(sim, which_year = 1, which_sim = 1)

plot_total_strat_fan(sim, surveys = 1:5, quants = seq(90, 10, by = -10), ...)

plot_length_strat_fan(
  sim,
  surveys = 1:5,
  years = 1:10,
  lengths = 1:50,
  select_by = "year",
  quants = seq(90, 10, by = -10),
  ...
)
```

```
plot_age_strat_fan(
  sim,
  surveys = 1:5,
  years = 1:10,
  ages = 1:10,
  select_by = "year",
  quants = seq(90, 10, by = -10),
  ...
)

plot_error_surface(sim, plot_by = "rule")

plot_survey_rank(sim, which_strat = "age")
```

Arguments

`sim`	Object returned by `sim_abundance`, `sim_distribution`, etc.
`sum_ages`	Sum across these ages
`col`	Plot color
`...`	Additional arguments to pass to `plot_ly`.
`mat`	Name of matrix in `sim` list to plot.
`xlab, ylab, zlab`	Axes labels.
`grid`	Grid produced by `make_grid`.
`ages`	Subset data to one or more ages.
`years`	Subset data to one or more years.
`type`	Plot type: "contour" or "heatmap".
`scale`	Plot response on "natural" or "log" scale?
`which_year`	Subset to specific year
`which_sim`	Subset to specific sim
`surveys`	Subset data to one or more surveys.
`quants`	Quantile intervals to display on fan plot
`lengths`	Subset data to one or more length groups.
`select_by`	Select plot by "age", "length" or "year"?
`plot_by`	Plot error surface by "rule" or "samples"?
`which_strat`	Which strat values to focus on? (total, length, or age)

Value

Returns a plot of class `plotly`.

round_sim *Round simulated population*

Description

Round simulated population

Usage

```
round_sim(sim)
```

Arguments

sim	Simulation from sim_distribution

Value

Returns a rounded simulation object. Largely used as a helper in sim_survey.

run_strat *Run stratified analysis on simulated data*

Description

Run stratified analysis on simulated data

Usage

```
run_strat(
  sim,
  length_group = "inherit",
  alk_scale = "division",
  strat_data_fun = strat_data,
  strat_means_fun = strat_means
)
```

Arguments

sim	Simulation from sim_survey
length_group	Size of the length frequency bins for both abundance at length calculations and age-length-key construction. By default this value is inherited from the value defined in sim_abundance from the closure supplied to sim_length ("inherit"). A numeric value can also be supplied, however, a mismatch in length groupings will cause issues with strat_error as true vs. estimated length groupings will be mismatched.

alk_scale	Spatial scale at which to construct and apply age-length-keys: "division" or "strat".
strat_data_fun	Function for preparing data for stratified analysis (e.g. strat_data)
strat_means_fun	Function for calculating stratified means (e.g. strat_means)

Details

The "strat_data_fun" and "strat_means_fun" allow the use of custom strat_data and strat_means functions.

Value

Adds stratified analysis results for the total population ("total_strat") and the population aggregated by length group and age ("length_strat" and "age_strat", respectively) to the sim list.

Examples

```
sim <- sim_abundance(ages = 1:5, years = 1:5,
                     R = sim_R(log_mean = log(1e+7)),
                     growth = sim_vonB(length_group = 1)) %>%
           sim_distribution(grid = make_grid(res = c(20, 20)),
                            ays_covar = sim_ays_covar(sd = 1)) %>%
           sim_survey(n_sims = 1, q = sim_logistic(k = 2, x0 = 3)) %>%
           run_strat()
```

sim_abundance *Simulate basic population dynamics model*

Description

Simulate basic population dynamics model

Usage

```
sim_abundance(
  ages = 1:20,
  years = 1:20,
  Z = sim_Z(),
  R = sim_R(),
  N0 = sim_N0(),
  growth = sim_vonB()
)
```

Arguments

ages — Ages to include in the simulation.

years — Years to include in the simulation.

Z — Total mortality function, like sim_Z, for generating mortality matrix.

R — Recruitment (i.e. abundance at min(ages)) function, like sim_R, for generating recruitment vector.

N0 — Starting abundance (i.e. abundance at min(years)) function, like sim_N0, for generating starting abundance vector.

growth — Closure, such as sim_vonB, for simulating length given age. The function is used here to generate a abundance-at-age matrix and it is carried forward for later use in sim_survey to simulate lengths from survey catch at age.

Details

Abundance from is calculated using a standard population dynamics model. An abundance-at-length matrix is generated using a growth function coded as a closure like sim_vonB. The function is retained for later use in sim_survey to simulate lengths given simulated catch at age in a simulated survey. The ability to simulate distributions by length is yet to be implemented.

Value

A list of length 9:

- ages - Vector of ages in the simulation
- lengths - Vector of length groups (depends on growth function)
- years - Vector of years in the simulation
- R - Vector of recruitment values
- N0 - Vector of starting abundance values
- Z - Matrix of total mortality values
- N - Matrix of abundance values
- N_at_length - Abundance at length matrix
- sim_length - Function for simulating lengths given ages

Examples

```
R_fun <- sim_R(log_mean = log(100000), log_sd = 0.1, random_walk = TRUE, plot = TRUE)
R_fun(years = 1:100)
sim_abundance(R = sim_R(log_mean = log(100000), log_sd = 0.5))
sim_abundance(years = 1:20,
              R = sim_R(log_mean = log(c(rep(100000, 10), rep(10000, 10))), plot = TRUE))

Z_fun <- sim_Z(log_mean = log(0.5), log_sd = 0.1, phi_age = 0.9, phi_year = 0.9, plot = TRUE)
Z_fun(years = 1:100, ages = 1:20)
sim_abundance(Z = sim_Z(log_mean = log(0.5), log_sd = 0.1, plot = TRUE))
Za_dev <- c(-0.2, -0.1, 0, 0.1, 0.2, 0.3, 0.3, 0.2, 0.1, 0)
Zy_dev <- c(-0.2, -0.2, -0.2, -0.2, -0.2, 2, 2, 2, 2, 0.2, 0.2, 0.2, 0.2, 0.2, 0, 0, 0, 0, 0, 0)
```

```
Z_mat <- outer(Za_dev, Zy_dev, "+") + 0.5
sim_abundance(ages = 1:10, years = 1:20,
              Z = sim_Z(log_mean = log(Z_mat), plot = TRUE))
sim_abundance(ages = 1:10, years = 1:20,
         Z = sim_Z(log_mean = log(Z_mat), log_sd = 0, phi_age = 0, phi_year = 0, plot = TRUE))

N0_fun <- sim_N0(N0 = "exp", plot = TRUE)
N0_fun(R0 = 1000, Z0 = rep(0.5, 20), ages = 1:20)
sim_abundance(N0 = sim_N0(N0 = "exp", plot = TRUE))

growth_fun <- sim_vonB(Linf = 100, L0 = 5, K = 0.2, log_sd = 0.05, length_group = 1, plot = TRUE)
growth_fun(age = rep(1:15, each = 100))
growth_fun(age = 1:15, length_age_key = TRUE)
sim_abundance(growth = sim_vonB(plot = TRUE))

sim <- sim_abundance()
plot_trend(sim)
plot_surface(sim, mat = "N")
plot_surface(sim, mat = "Z")
plot_surface(sim, mat = "N_at_length", xlab = "Length", zlab = "N")
```

sim_ays_covar *Simulate age-year-space covariance*

Description

These functions return a function to use inside sim_distribution.

Usage

```
sim_ays_covar(
  sd = 2.8,
  range = 300,
  lambda = 1,
  model = "matern",
  phi_age = 0.5,
  phi_year = 0.9,
  group_ages = 5:20,
  group_years = NULL
)
```

Arguments

sd	Variance (can be age specific).
range	Decorrelation range
lambda	Controls the degree of smoothness of Matern covariance process
model	String indicating either "exponential" or "matern" as the correlation function

`phi_age` Defines autocorrelation through ages. Can be one value or a vector of the same length as ages

`phi_year` Defines autocorrelation through years. Can be one value or a vector of the same length as years

`group_ages` Make space-age-year noise equal across these ages

`group_years` Make space-age-year noise equal across these years

Value

Returns a function for use inside `sim_distribution`.

`sim_ays_covar_spde` *Simulate age-year-space covariance using SPDE approach*

Description

[Experimental]

Returns a function to use inside `sim_distribution` to generate the error term.

Usage

```
sim_ays_covar_spde(
  sd = 2.8,
  range = 300,
  model = "spde",
  phi_age = 0.5,
  phi_year = 0.9,
  group_ages = 5:20,
  group_years = NULL,
  mesh,
  barrier.triangles
)
```

Arguments

`sd` Variance (can be age specific)

`range` Decorrelation range

`model` String indicating "barrier" or "spde" to generate Q with

`phi_age` Defines autocorrelation through ages. Can be one value or a vector of the same length as ages.

`phi_year` Defines autocorrelation through years. Can be one value or a vector of the same length as years.

`group_ages` Make space-age-year variance equal across these ages

`group_years` Make space-age-year variance equal across these years

`mesh` The mesh used to generate the precision matrix

`barrier.triangles`
the set of triangles in the barrier of the mesh for the barrier model

Value

Returns a function for use in sim_distribution.

Examples

```
if (requireNamespace("INLA")) {

  ## Make a grid
  my_grid <- make_grid(res = c(10,10))

  ## Make a mesh based off it

  my_mesh <- make_mesh(my_grid)
  sim <- sim_abundance(ages = 1:10, years = 1:10) %>%
          sim_distribution(grid = my_grid,
                           ays_covar = sim_ays_covar_spde(phi_age = 0.8,
                                                          phi_year = 0.1,
                                                          model = "spde",
                                                          mesh = my_mesh),
                           depth_par = sim_parabola(mu = 200,
                                                    sigma = 50))
  plot_distribution(sim, ages = 1:5, years = 1:5, type = "heatmap")

}
```

sim_distribution *Simulate spatial and temporal distribution*

Description

Provided an abundance at age matrix and a survey grid to populate, this function applies correlated space, age and year error to simulate the distribution of the population. The ability to simulate distributions by length is yet to be implemented.

Usage

```
sim_distribution(
  sim,
  grid = make_grid(),
  ays_covar = sim_ays_covar(),
  depth_par = sim_parabola()
)
```

Arguments

`sim` A list with ages, years and an abundance at age matrix like produced by `sim_abundance`.

`grid` A stars object defining the survey grid, like `survey_grid` or one produced by `make_grid`

`ays_covar` Closure for simulating age-year-space covariance, like `sim_ays_covar`

`depth_par` Closure for defining relationship between abundance and depth, like `sim_parabola`

Details

This function simulates the probability of simulated fish inhabiting a cell as a function of a parabolic relationship with depth and space, age, and year autocorrelated errors. WARNING: it make take a long time to simulate abundance in a large grid across many ages and years - start small first.

Value

Appends three objects to the `sim` list:

- `grid` - A stars object with the grid details
- `grid_xy` - Grid details as a data.table in xyz format
- `sp_N` - A data.table with abundance split by age, year and cell

Examples

```
sim <- sim_abundance(ages = 1:5, years = 1:5) %>%
           sim_distribution(grid = make_grid(res = c(20, 20)),
                            ays_covar = sim_ays_covar(phi_age = 0.8,
                                                      phi_year = 0.1),
                            depth_par = sim_parabola(mu = 200,
                                                     sigma = 50))
head(sim$sp_N)
head(sim$grid_xy)
```

`sim_logistic` *Closure for simulating logistic curve*

Description

This closure is useful for simulating q inside the `sim_survey` function

Usage

```
sim_logistic(k = 2, x0 = 3, plot = FALSE)
```

Arguments

`k` The steepness of the curve

`x0` The x-value of the sigmoid's midpoint

`plot` Plot relationship

Value

Returns a function for use in `sim_survey`.

Examples

```
logistic_fun <- sim_logistic(k = 2, x0 = 3, plot = TRUE)
logistic_fun(x = 1:10)
```

`sim_nlf` *Define a non-linear relationship*

Description

[Experimental]

Closure to be used in `sim_distribution`.

Usage

```
sim_nlf(
  formula = ~alpha - ((depth - mu)^2)/(2 * sigma^2),
  coeff = list(alpha = 0, mu = 200, sigma = 70)
)
```

Arguments

`formula` Formula describing parametric relationships between data and coefficients. The data used in `sim_distribution` are grid coordinates expanded across ages and years (i.e., includes columns `"x"`, `"y"`, `"depth"`, `"cell"`, `"division"`, `"strat"`, `"age"`, `"year"`). Values of the coefficients must be included in argument `coeff` as a named list.

`coeff` Named list of coefficient values used in `formula`.

Value

Returns a function for use inside `sim_distribution`.

Examples

```
## Make a grid and replicate data for 5 ages and 5 years
## (This is similar to what happens inside sim_distribution)
grid <- make_grid(shelf_width = 10)
grid_xy <- data.frame(grid)
i <- rep(seq(nrow(grid_xy)), times = 5)
a <- rep(1:5, each = nrow(grid_xy))
grid_xy <- grid_xy[i, ]
grid_xy$age <- a
i <- rep(seq(nrow(grid_xy)), times = 5)
y <- rep(1:5, each = nrow(grid_xy))
grid_xy <- grid_xy[i, ]
grid_xy$year <- y

## Now using sim_nlf, produce a function to apply to the expanded grid_xy data
## For this firs example, the depth effect is parabolic and the vertex is deeper by age
## (i.e., to impose ontogenetic deepening)
nlf <- sim_nlf(formula = ~ alpha - ((depth - mu + beta * age) ^ 2) / (2 * sigma ^ 2),
               coeff = list(alpha = 0, mu = 200, sigma = 70, beta = -70))
grid_xy$depth_effect <- nlf(grid_xy)

library(plotly)
grid_xy %>%
  filter(year == 1) %>%
  plot_ly(x = ~depth, y = ~depth_effect, split = ~age) %>%
  add_lines()
```

sim_parabola *Define a parabolic relationship*

Description

Closure to be used in sim_distribution. Form is based on the bi-gaussian function described here: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC2993707/.

Usage

```
sim_parabola(
  alpha = 0,
  mu = 200,
  sigma = 70,
  sigma_right = NULL,
  log_space = FALSE,
  plot = FALSE
)
```

Arguments

alpha, mu, sigma	Parameters that control the shape of the parabola. Can be one value or a vector of equal length to the number of ages in the simulation (e.g. age-specific depth associations can be specified).
sigma_right	Optional parameter to impose asymmetry by supplying a sigma parameter for the right side. If used, sigma will be used to define the width of the left side. Ignored if NULL.
log_space	Should shape of the parabola be defined in log space? If TRUE, logged parameters are assumed to be supplied and x values used in the parabola equation are log transformed. This allows a more lognormal curve to be defined and, hence, allows a heavier tail and it forces very low values near zero.
plot	Produce a simple plot of the simulated values?

Value

Returns a function for use inside sim_distribution.

Examples

```
parabola_fun <- sim_parabola(mu = 50, sigma = 5, plot = TRUE)
parabola_fun(data.frame(depth = 0:100))

parabola_fun <- sim_parabola(mu = log(40), sigma = 0.5, log_space = FALSE, plot = TRUE)
parabola_fun(data.frame(depth = 0:100))

parabola_fun <- sim_parabola(mu = c(50, 120), sigma = c(5, 3), plot = TRUE)
parabola_fun(expand.grid(depth = 1:200, age = 1:2))
```

sim_R *Simulate starting abundance, random recruitment and total mortality*

Description

These functions return a function to use inside sim_abundance. Given parameters, it generates N0, R and Z values.

Usage

```
sim_R(log_mean = log(3e+07), log_sd = 0.5, random_walk = TRUE, plot = FALSE)

sim_Z(
  log_mean = log(0.5),
  log_sd = 0.2,
  phi_age = 0.9,
  phi_year = 0.5,
  plot = FALSE
)

sim_N0(N0 = ”exp”, plot = FALSE)
```

Arguments

`log_mean` One mean value or a vector of means, in log scale, of length equal to years for `sim_R` or a matrix of means with rows equaling the number of ages and columns equaling the number of years for `sim_Z`.

`log_sd` Standard deviation of the variable in the log scale.

`random_walk` Simulate recruitment as a random walk?

`plot` produce a simple plot of the simulated values?

`phi_age` Autoregressive parameter for the age dimension.

`phi_year` Autoregressive parameter for the year dimension.

`N0` Either specify "exp" or numeric vector of starting abundance excluding the first age. If "exp" is specified using sim_N0, then abundance at age are calculated using exponential decay.

Details

sim_R generates uncorrelated recruitment values or random walk values from a log normal distribution. sim_Z does the same as sim_R when phi_age and phi_year are both 0, otherwise values are correlated in the age and/or year dimension. The covariance structure follows that described in Cadigan (2015).

Value

Returns a function for use inside `sim_abundance`.

References

Cadigan, Noel G. 2015. A State-Space Stock Assessment Model for Northern Cod, Including Under-Reported Catches and Variable Natural Mortality Rates. Canadian Journal of Fisheries and Aquatic Sciences 73 (2): 296-308.

Examples

```
R_fun <- sim_R(log_mean = log(100000), log_sd = 0.1, random_walk = TRUE, plot = TRUE)
R_fun(years = 1:100)
sim_abundance(R = sim_R(log_mean = log(100000), log_sd = 0.5))
sim_abundance(years = 1:20,
              R = sim_R(log_mean = log(c(rep(100000, 10), rep(10000, 10))), plot = TRUE))

Z_fun <- sim_Z(log_mean = log(0.5), log_sd = 0.1, phi_age = 0.9, phi_year = 0.9, plot = TRUE)
Z_fun(years = 1:100, ages = 1:20)
sim_abundance(Z = sim_Z(log_mean = log(0.5), log_sd = 0.1, plot = TRUE))
Za_dev <- c(-0.2, -0.1, 0, 0.1, 0.2, 0.3, 0.3, 0.2, 0.1, 0)
Zy_dev <- c(-0.2, -0.2, -0.2, -0.2, -0.2, 2, 2, 2, 2, 0.2, 0.2, 0.2, 0.2, 0.2, 0, 0, 0, 0, 0, 0)
Z_mat <- outer(Za_dev, Zy_dev, ”+”) + 0.5
```

```
sim_abundance(ages = 1:10, years = 1:20,
              Z = sim_Z(log_mean = log(Z_mat), plot = TRUE))
sim_abundance(ages = 1:10, years = 1:20,
         Z = sim_Z(log_mean = log(Z_mat), log_sd = 0, phi_age = 0, phi_year = 0, plot = TRUE))

N0_fun <- sim_N0(N0 = "exp", plot = TRUE)
N0_fun(R0 = 1000, Z0 = rep(0.5, 20), ages = 1:20)
sim_abundance(N0 = sim_N0(N0 = "exp", plot = TRUE))
```

sim_sets *Simulate survey sets*

Description

Simulate survey sets

Usage

```
sim_sets(
  sim,
  subset_cells,
  n_sims = 1,
  trawl_dim = c(1.5, 0.02),
  min_sets = 2,
  set_den = 2/1000,
  resample_cells = FALSE
)
```

Arguments

`sim` Simulation object from `sim_distribution`

`subset_cells` Logical expression indicating the elements (`x`, `y`, `depth`, `cell`,`division`, `strat`, `year`) of the survey grid to keep (e.g., `cell` < `100`)

`n_sims` Number of simulations to produce

`trawl_dim` Trawl width and distance (same units as grid)

`min_sets` Minimum number of sets per strat

`set_den` Set density (number of sets per grid unit squared)

`resample_cells` Allow resampling of sampling units (grid cells)? (Note: allowing resampling may create bias because depletion is imposed at the cell level)

Value

Returns a data.table including details of each set location.

Examples

```
sim <- sim_abundance(ages = 1:5, years = 1:5) %>%
         sim_distribution(grid = make_grid(res = c(20, 20)))

## Multiple calls can be useful for defining a custom series of sets
standard_sets <- sim_sets(sim, year <= 2, set_den = 2 / 1000)
reduced_sets <- sim_sets(sim, year > 2 & !cell %in% 1:100, set_den = 1 / 1000)
sets <- rbind(standard_sets, reduced_sets)
sets$set <- seq(nrow(sets)) # Important - make sure set has a unique ID.

survey <- sim_survey(sim, custom_sets = sets)

plot_survey(survey, which_year = 3, which_sim = 1)
```

`sim_survey` *Simulate stratified-random survey*

Description

Simulate stratified-random survey

Usage

```
sim_survey(
  sim,
  n_sims = 1,
  q = sim_logistic(),
  trawl_dim = c(1.5, 0.02),
  resample_cells = FALSE,
  binom_error = TRUE,
  min_sets = 2,
  set_den = 2/1000,
  lengths_cap = 500,
  ages_cap = 10,
  age_sampling = "stratified",
  age_length_group = 1,
  age_space_group = "division",
  custom_sets = NULL,
  light = TRUE
)
```

Arguments

`sim` Simulation from `sim_distribution`

n_sims	Number of surveys to simulate over the simulated population. Note: requesting a large number of simulations may max out your RAM. Use sim_survey_parallel if many simulations are required.
q	Closure, such as sim_logistic, for simulating catchability at age (returned values must be between 0 and 1)
trawl_dim	Trawl width and distance (same units as grid)
resample_cells	Allow resampling of sampling units (grid cells)? Setting to TRUE may introduce bias because depletion is imposed at the cell level.
binom_error	Impose binomial error? Setting to FALSE may introduce bias in stratified estimates at older ages because of more frequent rounding to zero.
min_sets	Minimum number of sets per strat
set_den	Set density (number of sets per grid unit squared). WARNING: may return an error if set_den is high and resample_cells = FALSE because the number of sets allocated may exceed the number of cells in a strata.
lengths_cap	Maximum number of lengths measured per set
ages_cap	If age_sampling = "stratified", this cap represents the maximum number of ages to sample per length group (defined using the age_length_group argument) per division or strat (defined using the age_space_group argument) per year. If age_sampling = "random", it is the maximum number of ages to sample from measured fish per set.
age_sampling	Should age sampling be "stratified" (default) or "random"?
age_length_group	Numeric value indicating the size of the length bins for stratified age sampling. Ignored if age_sampling = "random".
age_space_group	Should age sampling occur at the "division" (default), "strat" or "set" spatial scale? That is, age sampling can be spread across each "division", "strat" or "set" in each year to a maximum number within each length bin (cap is defined using the age_cap argument). Ignored if age_sampling = "random".
custom_sets	Supply an object of the same structure as returned by sim_sets which specifies a custom series of set locations to be sampled. Set locations are automated if custom_sets = NULL.
light	Drop some objects from the output to keep object size low?

Value

A list including rounded population simulation, set locations and details and sampling details. Note that that N = "true" population, I = population available to the survey, n = number caught by survey.

Examples

```
sim <- sim_abundance(ages = 1:5, years = 1:5) %>%
           sim_distribution(grid = make_grid(res = c(20, 20))) %>%
           sim_survey(n_sims = 5, q = sim_logistic(k = 2, x0 = 3))
plot_survey(sim, which_year = 3, which_sim = 1)
```

sim_survey_parallel *Simulate stratified random surveys using parallel computation*

Description

This function is a wrapper for sim_survey except it allows for many more total iterations to be run than sim_survey before running into RAM limitations. Unlike test_surveys, this function retains the full details of the survey and it may therefore be more useful for testing alternate approaches to a stratified analysis for obtaining survey indices.

Usage

```
sim_survey_parallel(
  sim,
  n_sims = 1,
  n_loops = 100,
  cores = 1,
  quiet = FALSE,
  ...
)
```

Arguments

sim	Simulation from sim_distribution
n_sims	Number of times to simulate a survey over the simulated population. Requesting a large number of simulations here may max out your RAM.
n_loops	Number of times to run the sim_survey function. Total simulations run will be the product of n_sims and n_loops arguments. Low numbers of n_sims and high numbers of n_loops will be easier on RAM, but may be slower.
cores	Number of cores to use in parallel. More cores should speed up the process.
quiet	Print message on what to expect for duration?
...	Arguments passed on to sim_survey

- q Closure, such as sim_logistic, for simulating catchability at age (returned values must be between 0 and 1)
- trawl_dim Trawl width and distance (same units as grid)
- resample_cells Allow resampling of sampling units (grid cells)? Setting to TRUE may introduce bias because depletion is imposed at the cell level.
- binom_error Impose binomial error? Setting to FALSE may introduce bias in stratified estimates at older ages because of more frequent rounding to zero.
- min_sets Minimum number of sets per strat

set_den Set density (number of sets per grid unit squared). WARNING: may return an error if set_den is high and resample_cells = FALSE because the number of sets allocated may exceed the number of cells in a strata.

lengths_cap Maximum number of lengths measured per set

ages_cap If age_sampling = "stratified", this cap represents the maximum number of ages to sample per length group (defined using the age_length_group argument) per division or strat (defined using the age_space_group argument) per year. If age_sampling = "random", it is the maximum number of ages to sample from measured fish per set.

age_sampling Should age sampling be "stratified" (default) or "random"?

age_length_group Numeric value indicating the size of the length bins for stratified age sampling. Ignored if age_sampling = "random".

age_space_group Should age sampling occur at the "division" (default), "strat" or "set" spatial scale? That is, age sampling can be spread across each "division", "strat" or "set" in each year to a maximum number within each length bin (cap is defined using the age_cap argument). Ignored if age_sampling = "random".

custom_sets Supply an object of the same structure as returned by sim_sets which specifies a custom series of set locations to be sampled. Set locations are automated if custom_sets = NULL.

light Drop some objects from the output to keep object size low?

Details

sim_survey is hard-wired here to be "light" to minimize object size.

Value

Returns an object of the same structure as sim_survey.

Examples

```
## This call runs a total of 25 simulations of the same survey over
## the same population (Note: total number of simulations are low to
## decrease computation time for the example)
sim <- sim_abundance(ages = 1:20, years = 1:5) %>%
           sim_distribution(grid = make_grid(res = c(10, 10))) %>%
           sim_survey_parallel(n_sims = 5, n_loops = 5, cores = 1,
                               q = sim_logistic(k = 2, x0 = 3),
                               quiet = FALSE)
```

sim_vonB *Closure for simulating length given age using von Bertalanffy notation*

Description

This function outputs a function which holds the parameter values supplied and the function either simulates lengths given ages or generates a length age key give a sequence of ages.

Usage

```
sim_vonB(
  Linf = 120,
  L0 = 5,
  K = 0.1,
  log_sd = 0.1,
  length_group = 3,
  digits = 0,
  plot = FALSE
)
```

Arguments

`Linf` Mean asymptotic length

`L0` Length at birth

`K` Growth rate parameter

`log_sd` Standard deviation of the relationship in log scale

`length_group` Length group for length age key. Note that labels on the matrix produced are midpoints using the DFO conventions; see `group_lengths`. Also note that this length group will dictate the length group used in the stratified analysis run by `run_strat`.

`digits` Integer indicating the number of decimal places to round the values to

`plot` Produce a simple plot of the simulated values?

Value

Returns a function for use inside `sim_abundance`.

Examples

```
growth_fun <- sim_vonB(Linf = 100, L0 = 5, K = 0.2, log_sd = 0.05, length_group = 1, plot = TRUE)
growth_fun(age = rep(1:15, each = 100))
growth_fun(age = 1:15, length_age_key = TRUE)
sim_abundance(growth = sim_vonB(plot = TRUE))
```

strat_data *Prepare simulated data for stratified analysis*

Description

Generate set details (setdet), length-frequency (lf) and age-frequency (af) data for stratified analysis

Usage

```
strat_data(sim, length_group = 3, alk_scale = "division")
```

Arguments

sim	Simulation from sim_survey
length_group	Size of the length frequency bins
alk_scale	Spatial scale at which to construct and apply age-length-keys: "division", "strat" or "set".

Value

Returns a list including set details (setdet), length-frequencies (lf), and age-frequencies (af).

strat_error *Calculate error of stratified estimates*

Description

Calculate error of stratified estimates

Usage

```
strat_error(sim)
```

Arguments

sim	Object from run_strat (includes simulated population and survey along with stratified analysis results)

Value

Adds details and summary stats of stratified estimate error to the sim list, ending with "_strat_error" or "_strat_error_stats". Error statistics includes mean absolute error ("MAE"), mean squared error ("MSE"), and root mean squared error ("RMSE")

Examples

```
sim <- sim_abundance(ages = 1:5, years = 1:5,
                     R = sim_R(log_mean = log(1e+7)),
                     growth = sim_vonB(length_group = 1)) %>%
           sim_distribution(grid = make_grid(res = c(20, 20)),
                            ays_covar = sim_ays_covar(sd = 1)) %>%
           sim_survey(n_sims = 1, q = sim_logistic(k = 2, x0 = 3)) %>%
           run_strat() %>%
           strat_error()
```

strat_means *Calculate stratified means, variances and confidence intervals across groups*

Description

Calculate stratified means, variances and confidence intervals across groups

Usage

```
strat_means(
  data = NULL,
  metric = NULL,
  strat_groups = NULL,
  survey_groups = NULL,
  confidence = 95
)
```

Arguments

data	Expects data.table with all grouping variables in stacked format (must include strat_area and tow_area for scaling values)
metric	Variable in specified data.table. e.g. "number", "mass"
strat_groups	Grouping variables for calculations of the fine-scale strat-level means (strat and strat_area are required). e.g. c("year", "species", "shiptrip", "NAFOdiv", "strat", "strat_area","age")
survey_groups	Grouping variables for large-scale summary calculations. e.g. ("year","species")
confidence	Percent for confidence limits

Details

Function was mainly created for use in the run_strat function. It first calculates strat-level statistics and then the larger-scale statistics like total abundance

Value

Returns a data.table including stratified estimates of abundance.

`survey_grid` *Sample survey simulation grid.*

Description

A exemplar for the structure of a survey grid object to supply to the functions in this package.

Usage

```
survey_grid
```

Format

A stars object with 4 attributes:

cell Survey cell identifier

division NAFO division

strat Survey strata number

depth Mean depth of the waters under each cell, units = m

For further details on how this file was created, see the data-raw folder for this package.

`survey_lite_mesh` *Lite sample survey mesh and related items*

Description

Lite sample survey mesh and related items

Usage

```
survey_lite_mesh
```

Format

A list containing the same items as survey_mesh, but with fewer nodes to save on computational time

`survey_mesh` *Sample survey meshes and related items*

Description

@format A list containing the R-INLA survey mesh, the set of triangles in the barrier and the barrier polygons for plotting

Usage

```
survey_mesh
```

Format

An object of class `list` of length 3.

Details

An example of a mesh containing barrier information for use with sim_ays_covar_spde. Also derived from global administrative boundaries data (http://gadm.org). Details on creation provided in the data-raw folder of this package in the survey_mesh.R file. Includes the set of barrier triangles needed to use the barrier approach, barrier polygons for plotting and the set of triangles in the barrier.

`test_surveys` *Test sampling design of multiple surveys using a stratified analysis*

Description

This function allows a series of sampling design settings to be set and tested on the simulated population. True population values are compared to stratified estimates of abundance.

Usage

```
test_surveys(
  sim,
  surveys = expand_surveys(),
  keep_details = 1,
  n_sims = 1,
  n_loops = 100,
  cores = 2,
  export_dir = NULL,
  length_group = "inherit",
  alk_scale = "division",
  progress = TRUE,
  ...
)

resume_test(export_dir = NULL, ...)
```

Arguments

`sim` Simulation from `sim_distribution`.

`surveys` A data.frame or data.table with a sequence of surveys and their settings with a format like the data.table returned by `expand_surveys`.

`keep_details` Survey and stratified analysis details are dropped here to minimize object size. This argument allows the user to keep the details of one survey by specifying the survey number in the data.frame supplied to `surveys`.

`n_sims` Number of times to simulate a survey over the simulated population. Requesting a large number of simulations here may max out your RAM.

`n_loops` Number of times to run the `sim_survey` function. Total simulations run will be the product of `n_sims` and `n_loops` arguments. Low numbers of `n_sims` and high numbers of `n_loops` will be easier on RAM, but may be slower.

`cores` Number of cores to use in parallel. More cores should speed up the process.

`export_dir` Directory for exporting results as they are generated. Main use of the export is to allow this process to pick up where `test_survey` left off by calling `resume_test`. If NULL, nothing is exported.

`length_group` Size of the length frequency bins for both abundance at length calculations and age-length-key construction. By default this value is inherited from the value defined in `sim_abundance` from the closure supplied to `sim_length` ("inherit"). A numeric value can also be supplied, however, a mismatch in length groupings will cause issues with `strat_error` as true vs. estimated length groupings will be mismatched.

`alk_scale` Spatial scale at which to construct and apply age-length-keys: "division" or "strat".

`progress` Display progress bar and messages?

`...` Arguments passed on to `sim_survey`

- `q` Closure, such as `sim_logistic`, for simulating catchability at age (returned values must be between 0 and 1)
- `trawl_dim` Trawl width and distance (same units as grid)
- `resample_cells` Allow resampling of sampling units (grid cells)? Setting to TRUE may introduce bias because depletion is imposed at the cell level.
- `binom_error` Impose binomial error? Setting to FALSE may introduce bias in stratified estimates at older ages because of more frequent rounding to zero.
- `min_sets` Minimum number of sets per strat
- `age_sampling` Should age sampling be "stratified" (default) or "random"?
- `age_length_group` Numeric value indicating the size of the length bins for stratified age sampling. Ignored if `age_sampling = "random"`.

`age_space_group` Should age sampling occur at the "division" (default), "strat" or "set" spatial scale? That is, age sampling can be spread across each "division", "strat" or "set" in each year to a maximum number within each length bin (cap is defined using the `age_cap` argument). Ignored if `age_sampling = "random"`.

`custom_sets` Supply an object of the same structure as returned by `sim_sets` which specifies a custom series of set locations to be sampled. Set locations are automated if `custom_sets = NULL`.

Details

Depending on the settings, `test_surveys` may take a long time to run. The `resume_test` function is for resuming partial runs of `test_surveys`. Note that progress bar time estimates will be biased here by previous completions. `test_loop` is a helper function used in both `test_surveys` and `resume_test`. CAUTION: while the dots construct is available in the `resume_test` function, be careful adding arguments as it will change the simulation settings if the arguments added were not specified in the initial `test_surveys` run.

Value

Adds a table of survey designs tested. Also adds details and summary stats of stratified estimate error to the `sim` list, ending with `"_strat_error"` or `"_strat_error_stats"`. Error statistics includes mean error (`"ME"`), mean absolute error (`"MAE"`), mean squared error (`"MSE"`), and root mean squared error (`"RMSE"`). Also adds a sample size summary table (`"samp_totals"`) to the list. Survey and stratified analysis details are not kept to minimize object size.

Examples

```
pop <- sim_abundance(ages = 1:20, years = 1:5) %>%
           sim_distribution(grid = make_grid(res = c(10, 10)))

surveys <- expand_surveys(set_den = c(1, 2) / 1000,
                          lengths_cap = c(100, 500),
                          ages_cap = c(5, 20))

## This call runs 25 simulations of 8 different surveys over the same
## population, and then runs a stratified analysis and compares true vs
## estimated values. (Note: total number of simulations are low to decrease
## computation time for the example)
tests <- test_surveys(pop, surveys = surveys, keep_details = 1,
                      n_sims = 5, n_loops = 5, cores = 1)

library(plotly)
tests$total_strat_error %>%
    filter(survey == 8, sim %in% 1:50) %>%
    group_by(sim) %>%
    plot_ly(x = ~year) %>%
    add_lines(y = ~I_hat, alpha = 0.5, name = "estimated") %>%
    add_lines(y = ~I, color = I("black"), name = "true") %>%
    layout(xaxis = list(title = "Year"),
           yaxis = list(title = "Abundance index"))

plot_total_strat_fan(tests, surveys = 1:8)
plot_length_strat_fan(tests, surveys = 1:8)
plot_age_strat_fan(tests, surveys = 1:8)
plot_age_strat_fan(tests, surveys = 1:8, select_by = "age")

plot_error_surface(tests, plot_by = "rule")
plot_error_surface(tests, plot_by = "samples")

plot_survey_rank(tests, which_strat = "length")
plot_survey_rank(tests, which_strat = "age")
```

vis_sim — *Make a flexdashboard for visualizing the simulation*

Description

Assumes the working directory is the project directory

Usage

```
vis_sim(sim, ...)
```

Arguments

`sim` Object produced by `sim_abundance`, `sim_distribution`, `sim_survey` or `test_surveys`.

`...` Additional arguments to send to run

Value

No value returned; function produces an interactive dashboard.

Examples

```
if (interactive()) {

  pop <- sim_abundance(ages = 1:20, years = 1:20)
  vis_sim(pop)

  dist <- sim_distribution(pop, grid = make_grid(res = c(10, 10)))
  vis_sim(dist)

  ## Run one survey design
  survey <- sim_survey(dist, n_sims = 5)
  vis_sim(survey)

  ## Run several survey designs and assess stratified estimates
 ## (Note: total number of simulations are low to decrease computation time for the example)
  surveys <- expand_surveys(set_den = c(1, 2) / 1000,
                            lengths_cap = c(100, 500),
                            ages_cap = c(5, 20))
  tests <- test_surveys(dist, surveys = surveys, keep_details = 1,
                        n_sims = 5, n_loops = 5, cores = 1)
  vis_sim(tests)

}
```

Index

∗ **datasets**
  bathy, 2
  land, 6
  survey_grid, 30
  survey_lite_mesh, 30
  survey_mesh, 31

bathy, 2

convert_N, 3

error_stats, 3
expand.grid, *4*
expand_surveys, 4, *32*

fibonacci, 4
findInterval, *5*

getData, *6*
group_lengths, 5, *27*

icc, 5

land, 6
lmer, *5*

make_grid, 6, *10*, *17*
make_mesh, 7

object.size, *8*
object_size, 8

plot_age_strat_fan *(*plot_trend*)*, 9
plot_distribution *(*plot_trend*)*, 9
plot_error_surface *(*plot_trend*)*, 9
plot_grid *(*plot_trend*)*, 9
plot_length_strat_fan *(*plot_trend*)*, 9
plot_ly, *10*
plot_surface *(*plot_trend*)*, 9
plot_survey *(*plot_trend*)*, 9
plot_survey_rank *(*plot_trend*)*, 9
plot_total_strat_fan *(*plot_trend*)*, 9
plot_trend, 9

resume_test *(*test_surveys*)*, 31
round_sim, 11
run, *34*
run_strat, 11, *27–29*

sim_abundance, *9–11*, 12, *17*, *20, 21*, *27*, *32*, *34*
sim_ays_covar, 14, *17*
sim_ays_covar_spde, 15
sim_distribution, *9–11*, *14–16*, 16, *18–20*, *22, 23*, *25*, *32*, *34*
sim_logistic, 17, *24, 25*, *32*
sim_N0, *13*
sim_N0 *(*sim_R*)*, 20
sim_nlf, 18
sim_parabola, *17*, 19
sim_R, *13*, 20
sim_sets, 22, *24*, *26*, *33*
sim_survey, *11*, *13*, *17, 18*, 23, *25, 26*, *28*, *32*, *34*
sim_survey_parallel, *24*, 25
sim_vonB, *13*, 27
sim_Z, *13*
sim_Z *(*sim_R*)*, 20
stats::spline, *6*
strat_data, *12*, 28
strat_error, *11*, 28, *32*
strat_means, *12*, 29
survey_grid, *7*, *17*, 30
survey_lite_mesh, 30
survey_mesh, 31

test_surveys, *25*, 31, *34*

vis_sim, 34