Package ‘SimSurvey’

August 22, 2025

Type Package

Title Test Surveys by Simulating Spatially-Correlated Populations

Version 0.1.7

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 (>= 4.1.0)

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, progress, doParallel, parallel, foreach, plotly, rlang, lifecycle

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

RoxygenNote 7.3.2

Encoding UTF-8

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 2025-08-22 09:00:02 UTC

Contents

bathy . . . 2
convert_N . . . 3
error_stats . . . 3
expand_surveys . . . 4
fibonacci . . . 5
group_lengths . . . 5
icc . . . 6
land . . . 6
make_grid . . . 7
make_mesh . . . 8
object_size . . . 9
plot_trend . . . 9
round_sim . . . 11
run_strat . . . 12
sim_abundance . . . 13
sim_ays_covar . . . 15
sim_ays_covar_spde . . . 16
sim_distribution . . . 17
sim_logistic . . . 19
sim_nlf . . . 19
sim_parabola . . . 21
sim_R . . . 22
sim_sets . . . 23
sim_survey . . . 25
sim_survey_parallel . . . 27
sim_vonB . . . 28
strat_data . . . 29
strat_error . . . 30
strat_means . . . 31
survey_grid . . . 32
survey_lite_mesh . . . 32
survey_mesh . . . 33
test_surveys . . . 33
vis_sim . . . 36

Index **38**

`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

Converts an abundance-at-age matrix to an abundance-at-length matrix using a length-age key. Both input matrices must be named appropriately.

Usage

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

Arguments

`N_at_age`	A matrix of abundance-at-age values.
`lak`	A length-age key matrix — i.e., the probability of being in a specific length group given age.

Value

A matrix of abundance-at-length values.

`error_stats` *Calculate common error statistics*

Description

Calculate common error statistics

Usage

```
error_stats(error)
```

Arguments

`error`	A numeric vector of errors.

Value

A named vector of error statistics including:

- **ME**: Mean error
- **MAE**: Mean absolute error
- **MSE**: Mean squared error
- **RMSE**: Root mean squared error

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

Description

A convenience function that wraps `base::expand.grid()` to generate all combinations of survey design parameters and adds a unique survey number to each row.

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`	A vector of set densities (sets per grid unit squared).
`lengths_cap`	A vector of maximum numbers of lengths measured per set.
`ages_cap`	A vector of maximum numbers of otoliths to collect per length group per division per year.

Value

A `data.frame` containing all combinations of the supplied vectors, with an added `survey` column identifying each combination.

`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 is not a general-purpose function — the output midpoints defining the groups are aligned with DFO-specific methods and labeling conventions.

Usage

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

Arguments

`length`	Numeric vector of lengths to be grouped. Used with `base::findInterval()`.
`group`	Numeric value specifying the width of the length group (i.e., bin size).

Value

A numeric vector indicating the midpoint of the assigned length group.

`icc` *Calculate intraclass correlation*

Description

A simple function for calculating intraclass correlation using `lme4::lmer()`. The formula follows the description provided on Wikipedia.

Usage

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

Arguments

`x`	Response variable.
`group`	Grouping variable.

Value

An 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/), which was downloaded using the `raster::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. Optionally, covariance can be added to the depth simulation.

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 the x dimension (km).
y_range	Range (min y, max y) in the y dimension (km).
res	Resolution of the grid cells (km).
shelf_depth	Approximate depth of the shelf (m).
shelf_width	Approximate width of the shelf (km).
depth_range	Range (min depth, max depth) of the depth values (m).
n_div	Number of divisions to include.
strat_breaks	Depth breaks used to define strata.
strat_splits	Number of times to horizontally split strata (i.e., a way to increase the number of strata).
method	Choose "spline", "loess", or "bezier" to generate a smooth gradient, or use "linear" for linear interpolation.

Value

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 on a grid*

Description

This function creates a mesh based on a given grid. While mesh construction and validation should ideally be done manually, this function provides a convenient default interface between a grid and `inla.mesh.2d`. It is designed to support usage with `sim_ays_covar_spde()`, and the default parameters are tuned for use with the default grid setup.

Usage

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

Arguments

`grid` A grid object to generate a mesh from.

`max.edge` The maximum allowed triangle edge length. One or two numeric values. Passed to `inla.mesh.2d`.

`bound.outer` Optional outer extension value passed to `offset`.

`cutoff` Minimum distance allowed between mesh points.

`offset` Automatic extension distance used by `inla.mesh.2d`.

`...` Additional options passed to `inla.mesh.2d`.

Value

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 `utils::object.size()` that prints in megabytes (Mb) by default.

Usage

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

Arguments

x	An R object.
units	The units to be used when printing the size.

Value

A character string with the object size followed by the unit.

plot_trend *Simple plotting functions*

Description

These are simple plotting helpers to quickly visualize outputs from `sim_abundance()`, `sim_distribution()`, and related simulation functions.

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`	Vector of ages to sum across.
`col`	Plot color.
`...`	Additional arguments passed to `plotly::plot_ly()`.
`mat`	Name of the matrix in the `sim` list to plot.
`xlab, ylab, zlab`	Axis labels.
`grid`	Grid produced by `make_grid()`.
`ages`	Subset the data to one or more ages.

years	Subset the data to one or more years.
type	Plot type: "contour" or "heatmap".
scale	Plot response on "natural" or "log" scale.
which_year	Subset to a specific year.
which_sim	Subset to a specific simulation replicate.
surveys	Subset the data to one or more surveys.
quants	Quantile intervals to display on the fan plot.
lengths	Subset the 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	Stratification focus: "total", "length", or "age".

Value

A plot of class plotly.

round_sim *Round simulated population*

Description

Rounds values in a simulation object, typically used as a helper function within sim_survey().

Usage

```
round_sim(sim)
```

Arguments

sim	A simulation object returned by sim_distribution().

Value

A rounded version of the simulation object.

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 object from `sim_survey()`.

`length_group` Size of the length frequency bins used for both abundance-at-length calculations and age-length-key construction. By default, this is inherited from the value defined in `sim_abundance()` via the closure supplied to `sim_length` (`"inherit"`). You may also supply a numeric value; however, mismatches in length groupings may cause issues with `strat_error()` if true vs. estimated groupings are not aligned.

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

`strat_data_fun` Function used to prepare data for stratified analysis (e.g., `strat_data()`).

`strat_means_fun` Function used to calculate stratified means (e.g., `strat_means()`).

Details

The `strat_data_fun` and `strat_means_fun` arguments allow you to use custom `strat_data()` and `strat_means()` functions.

Value

Adds stratified analysis results to the `sim` list:

- `"total_strat"`: Results for the total population
- `"length_strat"`: Results aggregated by length group
- `"age_strat"`: Results aggregated by age

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

Simulates a basic age-structured population using recruitment (`R`), total mortality (`Z`), and initial abundance (`N0`) functions. Optionally, a growth function may be provided to simulate lengths given age and generate an abundance-at-length matrix.

Usage

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

Arguments

`ages`	A numeric vector of ages to include in the simulation.
`years`	A numeric vector of years to include in the simulation.
`Z`	A function for generating a total mortality matrix, such as `sim_Z()`.
`R`	A function for generating a recruitment vector (i.e., abundance at `min(ages)`), such as `sim_R()`.
`N0`	A function for generating a starting abundance vector (i.e., abundance at `min(years)`), such as `sim_N0()`.
`growth`	A closure, such as `sim_vonB()`, for simulating length given age. This is used both to generate an abundance-at-length matrix and later for length simulation in `sim_survey()`.

Details

Abundance is simulated using a standard population dynamics model. If a growth function such as sim_vonB() is provided, it is used to create a corresponding abundance-at-length matrix. The same growth function is retained for use in sim_survey() to simulate lengths from catch-at-age survey data.

Note: The ability to simulate distributions by length is not yet implemented.

Value

A list with the following elements:

- ages: Vector of ages used in the simulation
- lengths: Vector of length groups (depends on growth function)
- years: Vector of years used in the simulation
- R: Vector of recruitment values
- N0: Vector of starting abundance values
- Z: Matrix of total mortality values
- N: Matrix of abundance-at-age
- N_at_length: Matrix of abundance-at-length
- 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 be used inside sim_distribution() to simulate spatially and temporally autocorrelated error.

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 in the Matérn covariance process.

`model` String indicating the correlation function to use: either `"exponential"` or `"matern"`.

`phi_age` Autocorrelation through ages. Can be a single value or a vector the same length as `ages`.

`phi_year` Autocorrelation through years. Can be a single value or a vector the same length as `years`.

`group_ages` A vector of ages to group together with shared space-age-year noise.

`group_years` A vector of years to group together with shared space-age-year noise.

Value

A function to be passed to `sim_distribution()` as the `ays_covar` argument.

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

Description

[Experimental]

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 of the process (can be age-specific).

`range` Decorrelation range.

`model` Either `"barrier"` or `"spde"`; determines how the precision matrix `Q` is generated.

`phi_age` Autocorrelation through ages. Can be a single value or a vector the same length as `ages`.

`phi_year` Autocorrelation through years. Can be a single value or a vector the same length as `years`.

`group_ages` Ages to group together for shared space-age-year variance.

`group_years` Years to group together for shared space-age-year variance.

`mesh` The mesh used to generate the precision matrix.

`barrier.triangles` The set of mesh triangles that define the barrier (used only in the barrier model).

Details

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

Value

A function that can be passed to `sim_distribution()` as the `ays_covar` argument.

Examples

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

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

  # Make a mesh based on the grid
  my_mesh <- make_mesh(my_grid)

  # Simulate and plot
  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 with an abundance-at-age matrix and a survey grid, this function applies correlated space–age–year error to simulate the spatial distribution of a population. Simulation by length is not yet 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 one produced by `sim_abundance()`.
`grid`	A `stars` object defining the survey grid, such as `survey_grid` or one created with `make_grid()`.
`ays_covar`	A closure that simulates age-year-space covariance, such as `sim_ays_covar()`.
`depth_par`	A closure that defines the relationship between abundance and depth, such as `sim_parabola()`.

Details

This function simulates the probability of fish inhabiting a cell based on a parabolic relationship with depth and spatially/temporally autocorrelated noise across age and year.

Warning: Simulating a large grid across many ages and years may be computationally intensive. Start with smaller simulations to test your setup.

Value

Appends the following objects to the `sim` list:

- `grid`: A `stars` object with grid details
- `grid_xy`: A `data.table` representation of the grid in XYZ format
- `sp_N`: A `data.table` of 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 catchability (q) inside the sim_survey() function.

Usage

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

Arguments

k	Steepness of the curve.
x0	The x-value at the midpoint of the sigmoid.
plot	Logical. Should the relationship be plotted?

Value

A function that can be passed to 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]

Usage

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

Arguments

formula A formula describing parametric relationships between the data and coefficients. The data used in sim_distribution() consist of grid coordinates expanded across ages and years, and include columns such as "x", "y", "depth", "cell", "division", "strat", "age", and "year". Coefficient values referenced in the formula must be provided in the coeff argument as a named list.

coeff A named list of coefficient values used in formula.

Details

Closure to be used in sim_distribution().

Value

A function that can be passed to sim_distribution().

Examples

```
## Make a grid and replicate data for 5 ages and 5 years
## (This mimics 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

## Define a non-linear function to apply to the expanded grid
## This example imposes ontogenetic deepening via a parabolic depth effect
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()`. The form is based on the bi-Gaussian function described in doi:10.1186/1471210511559.

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 a single value or a vector of the same length as the number of ages in the simulation (e.g., allowing age-specific depth associations).
`sigma_right`	Optional parameter to impose asymmetry by specifying a different `sigma` for the right side. If used, `sigma` defines the width of the left side. Ignored if `NULL`.
`log_space`	Logical. Should the shape of the parabola be defined in log space? If `TRUE`, parameters are assumed to be logged, and input `x` values are log-transformed. This produces a lognormal-like curve with a heavier right tail and forces low values near zero.
`plot`	Logical. Should a simple plot of the simulated values be produced?

Value

A function that can be passed to `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 closures for use inside sim_abundance(). Given user-defined parameters, they simulate recruitment (R), total mortality (Z), or initial abundance (N0) as a function of age and year.

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	For sim_R, a single mean or a vector of means (log scale) with length equal to the number of years. For sim_Z, a matrix of log-scale means with rows equal to the number of ages and columns equal to the number of years.
log_sd	Standard deviation on the log scale.
random_walk	Logical. Should recruitment be simulated as a random walk?
plot	Logical. Should a simple plot of the simulated values be displayed?
phi_age	Autoregressive parameter across the age dimension.
phi_year	Autoregressive parameter across the year dimension.
N0	For sim_N0, either "exp" (for exponential decay) or a numeric vector of starting abundances (excluding the first age).

Details

- sim_R() generates uncorrelated or random-walk recruitment values from a log-normal distribution.
- sim_Z() behaves like sim_R() when both phi_age and phi_year are zero. When either is non-zero, it introduces correlation in the age and/or year dimension, based on the covariance structure described in Cadigan (2015).
- sim_N0() provides starting abundance either via exponential decay or a user-defined vector.

Value

A function to be passed to `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

Simulates survey set locations from a population generated by `sim_distribution()`. Often used prior to running `sim_survey()`.

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` A simulation object returned by `sim_distribution()`.

`subset_cells` A logical expression to filter elements of the survey grid. Can reference columns like `x`, `y`, `depth`, `cell`, `division`, `strat`, or `year` (e.g., `cell < 100` or `year == 3`).

`n_sims` Number of survey simulations to generate.

`trawl_dim` Numeric vector specifying trawl width and distance (same units as the grid).

`min_sets` Minimum number of sets per stratum.

`set_den` Set density (number of sets per unit area).

`resample_cells` Logical. If `TRUE`, allows resampling of grid cells. Note: allowing resampling may introduce bias, since depletion is applied at the cell level.

Value

A `data.table` containing details of each survey set location.

Examples

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

# Define different sets for early and later years
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))  # Ensure each 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

Simulates a stratified-random survey on a population produced by `sim_distribution()`. Supports optional catchability functions, sampling caps, and custom set locations.

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` A simulation object returned by `sim_distribution()`.

`n_sims` Number of surveys to simulate. Be cautious: large values may consume significant memory. Use `sim_survey_parallel()` if many simulations are needed.

`q` A closure (e.g., `sim_logistic()`) for simulating catchability at age. Returned values must range between 0 and 1.

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

`resample_cells` Logical. If `TRUE`, allows grid cells to be resampled. May introduce bias, as depletion is applied at the cell level.

`binom_error` Logical. Should binomial error be imposed? If `FALSE`, stratified estimates at older ages may be biased due to rounding zeros.

`min_sets` Minimum number of sets per stratum.

`set_den` Set density (sets per grid unit squared). **Warning:** May error if `set_den` is high and `resample_cells = FALSE`, because allocated sets may exceed available cells.

`lengths_cap` Maximum number of lengths measured per set.

ages_cap
: Cap on the number of ages to sample, depending on age_sampling type:

- If "stratified": maximum per length bin (via age_length_group) and per age_space_group (e.g., "division", "strat").
- If "random": maximum number of fish aged per set.

age_sampling
: Type of age sampling strategy: "stratified" (default) or "random".

age_length_group
: Width of length bins for stratified age sampling. Ignored if age_sampling = "random".

age_space_group
: Spatial scale for stratified age sampling. Options: "division" (default), "strat", or "set". Ignored if age_sampling = "random".

custom_sets
: A data.table of set locations (same structure as returned by sim_sets()). If NULL, set locations are generated automatically.

light
: Logical. If TRUE, drops some objects from output to reduce memory footprint.

Value

A list including:

- Rounded simulation results
- Set location details
- Sampling results

Includes:

- N: true population
- I: individuals available to the survey
- n: individuals caught by the 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

A wrapper around sim_survey() that enables a much larger number of survey simulations to be performed using parallel processing. Unlike test_surveys(), this function retains full survey details and is suitable for evaluating alternate stratified analysis approaches for generating survey indices.

Usage

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

Arguments

sim — A simulation object returned by sim_distribution().

n_sims — Number of surveys to simulate per loop. Large values may increase memory usage significantly.

n_loops — Number of times to call sim_survey(). Total number of simulations = n_sims × n_loops. Using a smaller n_sims and larger n_loops reduces memory demand but may increase runtime.

cores — Number of processor cores to use in parallel. More cores typically reduce total time.

quiet — Logical. If FALSE, prints messages estimating run time.

... — Arguments passed on to sim_survey

- q A closure (e.g., sim_logistic()) for simulating catchability at age. Returned values must range between 0 and 1.
- trawl_dim Trawl width and distance (same units as the grid).
- resample_cells Logical. If TRUE, allows grid cells to be resampled. May introduce bias, as depletion is applied at the cell level.
- binom_error Logical. Should binomial error be imposed? If FALSE, stratified estimates at older ages may be biased due to rounding zeros.
- min_sets Minimum number of sets per stratum.
- set_den Set density (sets per grid unit squared). **Warning:** May error if set_den is high and resample_cells = FALSE, because allocated sets may exceed available cells.

lengths_cap Maximum number of lengths measured per set.

ages_cap Cap on the number of ages to sample, depending on age_sampling type:

- If "stratified": maximum per length bin (via age_length_group) and per age_space_group (e.g., "division", "strat").
- If "random": maximum number of fish aged per set.

age_sampling Type of age sampling strategy: "stratified" (default) or "random".

age_length_group Width of length bins for stratified age sampling. Ignored if age_sampling = "random".

age_space_group Spatial scale for stratified age sampling. Options: "division" (default), "strat", or "set". Ignored if age_sampling = "random".

custom_sets A data.table of set locations (same structure as returned by sim_sets()). If NULL, set locations are generated automatically.

light Logical. If TRUE, drops some objects from output to reduce memory footprint.

Details

This function runs sim_survey() with light = TRUE to reduce object size and minimize RAM usage.

Value

A list of the same structure as returned by sim_survey(), containing the results of all simulations.

Examples

```
# Run 25 total simulations (5 per loop × 5 loops) over the same population
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 returns a closure that holds the supplied parameter values and can be used to either simulate lengths given ages or generate a length-at-age key from 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 length-at-age relationship on the log scale.

`length_group` Length group width for constructing the length-at-age key. Labels on the resulting matrix use midpoints according to DFO conventions; see `group_lengths()`. This value will also determine the length groupings used in the stratified analysis via `run_strat()`.

`digits` Number of decimal places to round simulated lengths to.

`plot` Logical. Should a simple plot of the simulated values be produced?

Value

A function that can be passed to `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 object returned by `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

A list containing:

- `setdet`: Set details
- `lf`: Length-frequency data
- `af`: Age-frequency data

`strat_error` *Calculate error of stratified estimates*

Description

Calculate error of stratified estimates

Usage

```
strat_error(sim)
```

Arguments

`sim`	Object returned by `run_strat()`, which includes the simulated population, survey results, and stratified analysis outputs.

Value

Adds error details and summary statistics to the `sim` list, ending with `"*_strat_error"` and `"*_strat_error_stats"`. Error statistics include:

- **MAE**: Mean absolute error
- **MSE**: Mean squared error
- **RMSE**: Root mean squared error

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

This function is primarily designed for use within `run_strat()`. It first calculates statistics at the stratum level and then computes broader summaries like total abundance.

Usage

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

Arguments

`data`	A `data.table` with all grouping variables in stacked format. Must include `strat_area` and `tow_area` for scaling values.
`metric`	Name of the variable in `data.table` to summarize (e.g., `"number"`, `"mass"`).
`strat_groups`	Grouping variables for fine-scale stratum-level means. Must include `"strat"` and `"strat_area"`. Example: `c("year", "species", "shiptrip", "NAFOdiv", "strat", "strat_area", "age")`
`survey_groups`	Grouping variables for large-scale summary calculations Example: `c("year", "species")`
`confidence`	Confidence limit percentage (e.g., 95 for 95% CI).

Value

A `data.table` containing stratified estimates of abundance.

`survey_grid` *Sample survey simulation grid*

Description

An exemplar of the structure of a survey grid object used by 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 (in meters)

For more details on how this file was created, see the `data-raw` folder in 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 tested on a simulated population. True population values are compared to stratified estimates of abundance using a user-specified number of simulated surveys.

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 A simulation object returned by sim_distribution().

surveys A data.frame or data.table of survey configurations, formatted like the object returned by expand_surveys().

keep_details Integer. Retain full details for one survey (specified by survey number), and drop the rest to reduce object size.

n_sims Number of surveys to simulate per design. Large values may consume significant RAM.

n_loops Number of times to loop sim_survey(). Total number of simulations = n_sims × n_loops. A lower n_sims and higher n_loops combination is more memory efficient but may take longer.

cores Number of processor cores to use in parallel.

export_dir Optional directory path to export intermediate results. Useful for resuming later with resume_test(). If NULL, nothing is exported.

length_group Size of the length frequency bins used for both abundance-at-length calculations and age-length-key construction. By default, this is inherited from the value defined in sim_abundance() via the closure supplied to sim_length ("inherit"). You may also supply a numeric value; however, mismatches in length groupings may cause issues with strat_error() if true vs. estimated groupings are not aligned.

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

progress Logical. Should progress bar and messages be displayed?

... Arguments passed on to sim_survey

q A closure (e.g., sim_logistic()) for simulating catchability at age. Returned values must range between 0 and 1.

trawl_dim Trawl width and distance (same units as the grid).

resample_cells Logical. If TRUE, allows grid cells to be resampled. May introduce bias, as depletion is applied at the cell level.

binom_error Logical. Should binomial error be imposed? If FALSE, stratified estimates at older ages may be biased due to rounding zeros.

min_sets Minimum number of sets per stratum.

age_sampling Type of age sampling strategy: "stratified" (default) or "random".

age_length_group Width of length bins for stratified age sampling. Ignored if age_sampling = "random".

age_space_group Spatial scale for stratified age sampling. Options: "division" (default), "strat", or "set". Ignored if age_sampling = "random".

custom_sets A data.table of set locations (same structure as returned by sim_sets()). If NULL, set locations are generated automatically.

Details

Depending on the number of surveys and simulations, `test_surveys()` can take a long time to run.

The `resume_test()` function can be used to resume partial runs. Note: progress bar time estimates may be biased if resuming previously completed iterations.

Internally, this function calls a helper called `test_loop()` to process each survey simulation.

Caution: When using `...` inside `resume_test()`, be careful not to pass arguments that were not part of the original `test_surveys()` call, as this could change simulation settings.

Value

The returned object includes:

- A table of survey designs tested
- Stratified error results (`*_strat_error` and `*_strat_error_stats`)
- Error statistics:
  - `ME`: Mean error
  - `MAE`: Mean absolute error
  - `MSE`: Mean squared error
  - `RMSE`: Root mean squared error
- A summary table of total sample sizes (`samp_totals`)

Survey and stratified analysis details are dropped for all but one retained survey (via `keep_details`).

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)
)

# Simulate 25 surveys for each of 8 survey designs (low for example speed)
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

Launches an interactive flexdashboard to visualize simulation outputs. Assumes the working directory is the root project directory.

Usage

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

Arguments

sim	An object produced by sim_abundance(), sim_distribution(), sim_survey(), or test_surveys().
...	Additional arguments passed to rmarkdown::run().

Value

No return value. This function launches an interactive dashboard in the Viewer pane or browser.

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)

  # Single survey
  survey <- sim_survey(dist, n_sims = 5)
  vis_sim(survey)
  # Multiple survey designs
  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, 32
survey_lite_mesh, 32
survey_mesh, 33

base::expand.grid(), *4*
base::findInterval(), *5*
bathy, 2

convert_N, 3

error_stats, 3
expand_surveys, 4
expand_surveys(), *34*

fibonacci, 5

group_lengths, 5
group_lengths(), *29*

icc, 6

land, 6
lme4::lmer(), *6*

make_grid, 7
make_grid(), *10*, *18*
make_mesh, 8

object_size, 9

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_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
plotly::plot_ly(), *10*

raster::getData(), *6*
resume_test *(*test_surveys*)*, 33
resume_test(), *34, 35*
rmarkdown::run(), *36*
round_sim, 11
run_strat, 12
run_strat(), *29–31*

sim_abundance, 13
sim_abundance(), *9, 10*, *12*, *18*, *22, 23*, *29*, *34*, *36*
sim_ays_covar, 15
sim_ays_covar(), *18*
sim_ays_covar_spde, 16
sim_ays_covar_spde(), *8*
sim_distribution, 17
sim_distribution(), *9–11*, *15–17*, *20, 21*, *23–25*, *27*, *34*, *36*
sim_logistic, 19
sim_logistic(), *25*, *27*, *34*
sim_N0 *(*sim_R*)*, 22
sim_N0(), *13*
sim_nlf, 19
sim_parabola, 21
sim_parabola(), *18*
sim_R, 22
sim_R(), *13*
sim_sets, 23
sim_sets(), *26*, *28*, *34*
sim_survey, 25, *27*, *34*
sim_survey(), *11–14*, *19*, *23*, *27, 28*, *30*, *34*, *36*
sim_survey_parallel, 27
sim_survey_parallel(), *25*
sim_vonB, 28
sim_vonB(), *13, 14*

sim_Z *(*sim_R*)*, 22
sim_Z(), *13*
stats::spline(), *7*
strat_data, 29
strat_data(), *12*
strat_error, 30
strat_error(), *12*, *34*
strat_means, 31
strat_means(), *12*
survey_grid, *8*, *18*, 32
survey_lite_mesh, 32
survey_mesh, 33

test_surveys, 33
test_surveys(), *27*, *36*

utils::object.size(), *9*

vis_sim, 36