Package 'serocalculator'

March 25, 2026

Type Package

Title Estimating Infection Rates from Serological Data

Version 1.4.1

Description Translates antibody levels measured in cross-sectional population samples into estimates of the frequency with which seroconversions (infections) occur in the sampled populations. Replaces the previous `seroincidence` package.

License GPL-3

URL https://ucd-serg.github.io/serocalculator/, https://github.com/UCD-SERG/serocalculator

BugReports https://github.com/UCD-SERG/serocalculator/issues

Depends R (>= 4.1.0)

Imports cli, doParallel, dplyr, foreach, ggplot2, patchwork, lifecycle, magrittr, Rcpp, rlang, rngtools, scales, stats, tibble, tidyr, tidyselect, utils, purrr, and, glue, stringr, parallel, labelled

Suggests bookdown, DT, fs, ggbeeswarm, knitr, mixtools, pak, quarto, rmarkdown, spelling, ssdtools (>= 1.0.6.9016), testthat (>= 3.0.0), tidyverse, qrcode, vdiffr, withr, forcats, rex, readr

LinkingTo Rcpp

Config/testthat/edition 3

Config/Needs/roxygen2 roxygen2, moodymudskipper/devtag

Config/Needs/lint r-lib/lintr

Config/Needs/website quarto, lewinfox/foodwebr

Config/Needs/check readr

Encoding UTF-8

Language en-US

LazyData true

NeedsCompilation yes

RoxygenNote 7.3.3

Author Kristina Lai [aut, cre],
Chris Orwa [aut],
Kwan Ho Lee [ctb],
Peter Teunis [aut, cph] (Author of the method and original code.),
Kristen Aiemjoy [aut],
Douglas Ezra Morrison [aut]

Maintainer Kristina Lai <kwlai@ucdavis.edu>

Repository CRAN

Date/Publication 2026-03-25 22:30:07 UTC

Contents

`analyze_sims` *Analyze simulation results*

Description

Analyze simulation results

Usage

```
analyze_sims(data)
```

Arguments

`data`	a tibble::tbl_df with columns: • `lambda.sim`, • `incidence.rate`, • `SE`, • `CI.lwr`, • `CI.upr` for example, as produced by `summary.seroincidence.by()` with `lambda.sim` as a stratifying variable

Value

a `sim_results` object (extends tibble::tbl_df)

Examples

```
dmcmc <- typhoid_curves_nostrat_100

n_cores <- 2

nclus <- 20
# cross-sectional sample size
nrep <- c(50, 200)

# incidence rate in e
lambdas <- c(.05, .8)

antibodies <- c("HlyE_IgA", "HlyE_IgG")
lifespan <- c(0, 10)
dlims = rbind(
"HlyE_IgA" = c(min = 0, max = 0.5),
"HlyE_IgG" = c(min = 0, max = 0.5)
```

```
)
sim_df <- sim_pop_data_multi(
num_cores = n_cores,
lambdas = lambdas,
nclus = nclus,
sample_sizes = nrep,
age_range = lifespan,
antigen_isos = antibodies,
renew_params = FALSE,
add_noise = TRUE,
curve_params = dmcmc,
noise_limits = dlims,
format = "long"
)
cond <- tibble::tibble(
antigen_iso = c("HlyE_IgG", "HlyE_IgA"),
nu = c(0.5, 0.5), # Biologic noise (nu)
eps = c(0, 0), # M noise (eps)
y.low = c(1, 1), # low cutoff (llod)
y.high = c(5e6, 5e6)
)
ests <-
est_seroincidence_by(
pop_data = sim_df,
sr_params = dmcmc,
noise_params = cond,
num_cores = n_cores,
strata = c("lambda.sim", "sample_size", "cluster"),
curve_strata_varnames = NULL,
noise_strata_varnames = NULL,
verbose = FALSE,
build_graph = FALSE, # slows down the function substantially
antigen_isos = c("HlyE_IgG", "HlyE_IgA")
)

ests |>
summary() |>
analyze_sims()
```

`as_noise_params` *Load noise parameters*

Description

Load noise parameters

Usage

```
as_noise_params(data, antigen_isos = NULL)
```

Arguments

data	a data.frame() or tibble::tbl_df
antigen_isos	character() vector of antigen isotypes to be used in analyses

Value

a noise_params object (a tibble::tbl_df with extra attribute antigen_isos)

Examples

```
library(magrittr)
noise_data <-
  serocalculator_example("example_noise_params.csv") %>%
  read.csv() %>%
  as_noise_params()

print(noise_data)
```

---

as_pop_data *Load a cross-sectional antibody survey data set*

Description

Load a cross-sectional antibody survey data set

Usage

```
as_pop_data(
  data,
  antigen_isos = NULL,
  age = "Age",
  value = "result",
  id = "index_id",
  standardize = TRUE
)
```

Arguments

data	a data.frame() or tibble::tbl_df
antigen_isos	character() vector of antigen isotypes to be used in analyses
age	a character() identifying the age column
value	a character() identifying the value column
id	a character() identifying the id column
standardize	a logical() to determine standardization of columns

Value

a pop_data object (a tibble::tbl_df with extra attribute antigen_isos)

Examples

```
library(magrittr)
xs_data <-
  serocalculator_example("example_pop_data.csv") |>
  read.csv() |>
  as_pop_data()

print(xs_data)
```

---

as_sr_params *Load longitudinal seroresponse parameters*

Description

Load longitudinal seroresponse parameters

Usage

```
as_sr_params(data, antigen_isos = NULL)
```

Arguments

data	a data.frame() or tibble::tbl_df
antigen_isos	a character() vector of antigen isotypes to be used in analyses

Value

a curve_data object (a tibble::tbl_df with extra attribute antigen_isos)

Examples

```
library(magrittr)
curve_data <-
  serocalculator_example("example_curve_params.csv") %>%
  read.csv() %>%
  as_curve_params()

print(curve_data)
```

`autoplot.curve_params` *Graph antibody decay curves by antigen isotype*

Description

Graph antibody decay curves by antigen isotype

Usage

```
## S3 method for class 'curve_params'
autoplot(
  object,
  method = c("graph.curve.params", "graph_seroresponse_model_1"),
  ...
)
```

Arguments

`object`	a `curve_params` object (constructed using `as_sr_params()`), which is a `data.frame()` containing MCMC samples of antibody decay curve parameters
`method`	a character string indicating whether to use • `graph.curve.params()` (default) or • `graph_seroresponse_model_1()` (previous default) as the graphing method.
`...`	additional arguments passed to the sub-function indicated by the `method` argument.

Details

Currently, the backend for this method is `graph.curve.params()`. Previously, the backend for this method was `graph_seroresponse_model_1()`. That function is still available if preferred.

Value

a `ggplot2::ggplot()` object

Examples

```
library(dplyr)
library(ggplot2)
library(magrittr)

curve <-
  serocalculator_example("example_curve_params.csv") |>
  read.csv() |>
  as_sr_params() |>
  filter(antigen_iso %in% c("HlyE_IgA", "HlyE_IgG")) |>
  autoplot()
```

curve

`autoplot.pop_data` *Plot distribution of antibodies*

Description

`autoplot()` method for `pop_data` objects

Usage

```
## S3 method for class 'pop_data'
autoplot(object, log = FALSE, type = "density", strata = NULL, ...)
```

Arguments

`object`	A `pop_data` object (from `load_pop_data()`)
`log`	whether to show antibody responses on logarithmic scale
`type`	an option to choose type of chart: the current options are `"density"` or `"age-scatter"`
`strata`	the name of a variable in `pop_data` to stratify by (or `NULL` for no stratification)
`...`	unused

Value

a ggplot2::ggplot object

Examples

```
library(dplyr)
library(ggplot2)
library(magrittr)

xs_data <-
  serocalculator_example("example_pop_data.csv") |>
  read.csv() |>
  as_pop_data()

xs_data |> autoplot(strata = "catchment", type = "density")
xs_data |> autoplot(strata = "catchment", type = "age-scatter")
```

`autoplot.seroincidence` *Plot the log-likelihood curve for the incidence rate estimate*

Description

Plot the log-likelihood curve for the incidence rate estimate

Usage

```
## S3 method for class 'seroincidence'
autoplot(object, log_x = FALSE, ...)
```

Arguments

`object`	a `seroincidence` object (from `est_seroincidence()`)
`log_x`	should the x-axis be on a logarithmic scale (`TRUE`) or linear scale (`FALSE`, default)?
`...`	unused

Value

a `ggplot2::ggplot()`

Examples

```
library(dplyr)
library(ggplot2)

xs_data <-
  sees_pop_data_pk_100

curve <-
  typhoid_curves_nostrat_100 %>%
  filter(antigen_iso %in% c("HlyE_IgA", "HlyE_IgG"))

noise <-
  example_noise_params_pk

est1 <- est_seroincidence(
  pop_data = xs_data,
  sr_param = curve,
  noise_param = noise,
  antigen_isos = c("HlyE_IgG", "HlyE_IgA"),
  build_graph = TRUE
)

# Plot the log-likelihood curve
```

```
autoplot(est1)
```

`autoplot.seroincidence.by` *Plot* `seroincidence.by` *log-likelihoods*

Description

Plots log-likelihood curves by stratum, for `seroincidence.by` objects

Usage

```
## S3 method for class 'seroincidence.by'
autoplot(object, ncol = min(3, length(object)), ...)
```

Arguments

`object`	a '"seroincidence.by"' object (from `est_seroincidence_by()`)
`ncol`	number of columns to use for panel of plots
`...`	Arguments passed on to `autoplot.seroincidence`
	`log_x` should the x-axis be on a logarithmic scale (`TRUE`) or linear scale (`FALSE`, default)?

Value

a `"patchwork"` object: a single or `list()` of `ggplot2::ggplot()`s

Examples

```
library(dplyr)
library(ggplot2)

xs_data <-
  sees_pop_data_pk_100

curve <-
  typhoid_curves_nostrat_100 %>%
  filter(antigen_iso %in% c("HlyE_IgA", "HlyE_IgG"))

noise <-
  example_noise_params_pk

est2 <- est_seroincidence_by(
  strata = c("catchment"),
  pop_data = xs_data,
  sr_params = curve,
  curve_strata_varnames= NULL,
```

```
  noise_strata_varnames = NULL,
  noise_params = noise,
  antigen_isos = c("HlyE_IgG", "HlyE_IgA"),
  #num_cores = 8, #Allow for parallel processing to decrease run time
  build_graph = TRUE
)

# Plot the log-likelihood curve
autoplot(est2)
```

`autoplot.sim_results` *Plot simulation results* `autoplot()` *method for* `sim_results` *objects*

Description

Plot simulation results `autoplot()` method for `sim_results` objects

Usage

```
## S3 method for class 'sim_results'
autoplot(object, statistic = "Empirical_SE", ...)
```

Arguments

`object`	a `sim_results` object (from `analyze_sims()`)
`statistic`	which column of `object` should be the y-axis?
`...`	unused

Value

a ggplot2::ggplot

Examples

```
dmcmc <- typhoid_curves_nostrat_100

n_cores <- 2

nclus <- 20
# cross-sectional sample size
nrep <- c(50, 200)

# incidence rate in e
lambdas <- c(.05, .8)
lifespan <- c(0, 10)
antibodies <- c("HlyE_IgA", "HlyE_IgG")
dlims <- rbind(
"HlyE_IgA" = c(min = 0, max = 0.5),
```

```
"HlyE_IgG" = c(min = 0, max = 0.5)
)
sim_df <-
sim_pop_data_multi(
num_cores = n_cores,
lambdas = lambdas,
nclus = nclus,
sample_sizes = nrep,
age_range = lifespan,
antigen_isos = antibodies,
renew_params = FALSE,
add_noise = TRUE,
curve_params = dmcmc,
noise_limits = dlims,
format = "long"
)
cond <- tibble::tibble(
antigen_iso = c("HlyE_IgG", "HlyE_IgA"),
nu = c(0.5, 0.5), # Biologic noise (nu)
eps = c(0, 0), # M noise (eps)
y.low = c(1, 1), # low cutoff (llod)
y.high = c(5e6, 5e6)
)
ests <-
est_seroincidence_by(
pop_data = sim_df,
sr_params = dmcmc,
noise_params = cond,
num_cores = n_cores,
strata = c("lambda.sim", "sample_size", "cluster"),
curve_strata_varnames = NULL,
noise_strata_varnames = NULL,
verbose = FALSE,
build_graph = FALSE, # slows down the function substantially
antigen_isos = c("HlyE_IgG", "HlyE_IgA")
)

ests |>
summary() |>
analyze_sims() |>
autoplot()
```

`autoplot.summary.seroincidence.by` *Plot method for* `summary.seroincidence.by` *objects*

Description

Plot method for `summary.seroincidence.by` objects

Usage

```
## S3 method for class 'summary.seroincidence.by'
autoplot(object, type, ...)
```

Arguments

object	a summary.seroincidence.by object (generated by applying the summary() method to the output of est_seroincidence_by()).
type	character string indicating which type of plot to generate. The implemented options are: • "scatter": calls strat_ests_scatterplot() to generate a scatterplot • "bar": calls strat_ests_barplot() to generate a barplot
...	Arguments passed on to strat_ests_scatterplot, strat_ests_barplot xvar the name of a stratifying variable in object alpha transparency for the points in the graph (1 = no transparency, 0 = fully transparent) shape shape argument for geom_point() dodge_width width for jitter CIs logical, if TRUE, add CI error bars color_var character which variable in object to use to determine point color group_var character which variable in object to use to connect points with lines (NULL for no lines) yvar the name of a stratifying variable in object. title a title for the final plot. xlab a label for the x-axis of the final plot. ylab a label for the y-axis of the final plot. fill_lab fill label. color_palette optional color palette for bar color.

Value

a ggplot2::ggplot() object

Examples

```
library(dplyr)
library(ggplot2)

xs_data <-
  sees_pop_data_pk_100

curve <-
  typhoid_curves_nostrat_100 %>%
  filter(antigen_iso %in% c("HlyE_IgA", "HlyE_IgG"))

noise <-
```

```
  example_noise_params_pk

est2 <- est_seroincidence_by(
  strata = c("catchment", "ageCat"),
  pop_data = xs_data,
  sr_params = curve,
  noise_params = noise,
  curve_strata_varnames= NULL,
  noise_strata_varnames = NULL,
  antigen_isos = c("HlyE_IgG", "HlyE_IgA"),
  num_cores = 2 # Allow for parallel processing to decrease run time
)

est2sum <- summary(est2)

est2sum |> autoplot(
    type ="scatter",
    xvar = "ageCat",
    color_var = "catchment",
    CIs = TRUE,
    group_var = "catchment")

est2sum |> autoplot(
    type = "bar",
    yvar = "ageCat",
    color_var = "catchment",
    CIs = TRUE)
```

`check_pop_data` *Check the formatting of a cross-sectional antibody survey dataset.*

Description

Check the formatting of a cross-sectional antibody survey dataset.

Usage

```
check_pop_data(pop_data, verbose = FALSE)
```

Arguments

`pop_data`	dataset to check
`verbose`	whether to print an "OK" message when all checks pass

Value

NULL (invisibly)

Examples

```
library(magrittr)

xs_data <-
  serocalculator_example("example_pop_data.csv") |>
  read.csv() |>
  as_pop_data()

check_pop_data(xs_data, verbose = TRUE)
```

`compare_seroincidence` *Compare seroincidence rates between two groups*

Description

Perform a two-sample z-test to compare seroincidence rates between two groups. Since we use maximum likelihood estimation (MLE) for each seroincidence estimate and estimates from different strata or data sets are uncorrelated, we can use a simple two-sample z-test using the Gaussian distribution. The standard error for the difference is computed by adding the estimated variances and taking the square root.

Usage

```
compare_seroincidence(x, y = NULL, coverage = 0.95, verbose = FALSE, ...)

## S3 method for class 'seroincidence'
compare_seroincidence(x, y = NULL, coverage = 0.95, verbose = FALSE, ...)

## S3 method for class 'seroincidence.by'
compare_seroincidence(x, y = NULL, coverage = 0.95, verbose = FALSE, ...)
```

Arguments

`x`	A `"seroincidence"` object from `est_seroincidence()` or a `"seroincidence.by"` object from `est_seroincidence_by()`
`y`	A `"seroincidence"` object from `est_seroincidence()` (optional if `x` is a `"seroincidence.by"` object)
`coverage`	Desired confidence interval coverage probability (default = 0.95)
`verbose`	Logical indicating whether to print verbose messages (default = FALSE)
`...`	Additional arguments (currently unused)

Details

When comparing two single `"seroincidence"` objects, this function performs a two-sample z-test and returns results in the standard `htest` format.

When applied to a `"seroincidence.by"` object (stratified estimates), the function compares all pairs of strata and returns a nicely formatted table with point estimates for the difference in seroincidence, p-values, and confidence intervals.

The test statistic is computed as:

$$z = \frac{\lambda_1 - \lambda_2}{\sqrt{SE_1^2 + SE_2^2}}$$

where λ_1 and λ_2 are the estimated incidence rates, and SE_1 and SE_2 are their standard errors.

Value

- When comparing two `"seroincidence"` objects: An object of class `"htest"` containing the test statistic, p-value, confidence interval, and estimates.
- When applied to a `"seroincidence.by"` object: A `tibble::tibble()` with columns for each pair of strata, the difference in incidence rates, standard error, z-statistic, p-value, and confidence interval bounds.

Methods (by class)

- `compare_seroincidence(seroincidence)`: Compare two single seroincidence estimates
- `compare_seroincidence(seroincidence.by)`: Compare all pairs of stratified seroincidence estimates

Examples

```
## Not run:
# See inst/examples/exm-compare_seroincidence.R for complete examples

## End(Not run)
```

`est_seroincidence` *Find the maximum likelihood estimate of the incidence rate parameter*

Description

This function models seroincidence using maximum likelihood estimation; that is, it finds the value of the seroincidence parameter which maximizes the likelihood (i.e., joint probability) of the data.

Usage

```
est_seroincidence(
  pop_data,
  sr_params,
  noise_params,
  antigen_isos = get_biomarker_names(pop_data),
  lambda_start = 0.1,
  stepmin = 1e-08,
  stepmax = 3,
  verbose = FALSE,
  build_graph = FALSE,
  print_graph = build_graph & verbose,
  cluster_var = NULL,
  stratum_var = NULL,
  sampling_weights = NULL,
  ...
)
```

Arguments

pop_data	a data.frame with cross-sectional serology data per antibody and age, and additional columns
sr_params	a `data.frame()` containing MCMC samples of parameters from the Bayesian posterior distribution of a longitudinal decay curve model. The parameter columns must be named: • `antigen_iso`: a `character()` vector indicating antigen-isotype combinations • `iter`: an `integer()` vector indicating MCMC sampling iterations • `y0`: baseline antibody level at $t=0$ ($y(t=0)$) • `y1`: antibody peak level (ELISA units) • `t1`: duration of infection • `alpha`: antibody decay rate (1/days for the current longitudinal parameter sets) • `r`: shape factor of antibody decay
noise_params	a `data.frame()` (or `tibble::tibble()`) containing the following variables, specifying noise parameters for each antigen isotype: • `antigen_iso`: antigen isotype whose noise parameters are being specified on each row • `nu`: biological noise • `eps`: measurement noise • `y.low`: lower limit of detection for the current antigen isotype • `y.high`: upper limit of detection for the current antigen isotype
antigen_isos	Character vector with one or more antibody names. Must match `pop_data`
lambda_start	starting guess for incidence rate, in events/year.
stepmin	A positive scalar providing the minimum allowable relative step length.

`stepmax` a positive scalar which gives the maximum allowable scaled step length. `stepmax` is used to prevent steps which would cause the optimization function to overflow, to prevent the algorithm from leaving the area of interest in parameter space, or to detect divergence in the algorithm. `stepmax` would be chosen small enough to prevent the first two of these occurrences, but should be larger than any anticipated reasonable step.

`verbose` logical: if TRUE, print verbose log information to console

`build_graph` whether to graph the log-likelihood function across a range of incidence rates (lambda values)

`print_graph` whether to display the log-likelihood curve graph in the course of running `est_seroincidence()`

`cluster_var` optional name(s) of the variable(s) in `pop_data` containing cluster identifiers for clustered sampling designs (e.g., households, schools). Can be a single variable name (character string) or a vector of variable names for multi-level clustering (e.g., `c("school", "classroom")`). When provided, standard errors will be adjusted for within-cluster correlation using cluster-robust variance estimation.

`stratum_var` optional name of the variable in `pop_data` containing stratum identifiers. Used in combination with `cluster_var` for stratified cluster sampling designs.

`sampling_weights`
optional data.frame containing sampling weights with columns for cluster/stratum identifiers and their sampling probabilities. Currently not implemented; reserved for future use.

`...` Arguments passed on to `stats::nlm`

`typsize` an estimate of the size of each parameter at the minimum.

`fscale` an estimate of the size of `f` at the minimum.

`ndigit` the number of significant digits in the function `f`.

`gradtol` a positive scalar giving the tolerance at which the scaled gradient is considered close enough to zero to terminate the algorithm. The scaled gradient is a measure of the relative change in `f` in each direction `p[i]` divided by the relative change in `p[i]`.

`iterlim` a positive integer specifying the maximum number of iterations to be performed before the program is terminated.

`check.analyticals` a logical scalar specifying whether the analytic gradients and Hessians, if they are supplied, should be checked against numerical derivatives at the initial parameter values. This can help detect incorrectly formulated gradients or Hessians.

Value

a `"seroincidence"` object, which is a `stats::nlm()` fit object with extra metadata attributes `lambda_start`, `antigen_isos`, and `ll_graph`

Examples

```
library(dplyr)

xs_data <-
```

```
  sees_pop_data_pk_100

sr_curve <-
  typhoid_curves_nostrat_100 |>
  filter(antigen_iso %in% c("HlyE_IgA", "HlyE_IgG"))

noise <-
  example_noise_params_pk

# Basic usage without clustering
est1 <- est_seroincidence(
  pop_data = xs_data,
  sr_params = sr_curve,
  noise_params = noise,
  antigen_isos = c("HlyE_IgG", "HlyE_IgA"),
)

summary(est1)

# Usage with clustered sampling design
# Standard errors will be adjusted for within-cluster correlation
est2 <- est_seroincidence(
  pop_data = xs_data,
  sr_params = sr_curve,
  noise_params = noise,
  antigen_isos = c("HlyE_IgG", "HlyE_IgA"),
  cluster_var = "cluster"
)

summary(est2)

# With both cluster and stratum variables
est3 <- est_seroincidence(
  pop_data = xs_data,
  sr_params = sr_curve,
  noise_params = noise,
  antigen_isos = c("HlyE_IgG", "HlyE_IgA"),
  cluster_var = "cluster",
  stratum_var = "catchment"
)

summary(est3)
```

`est_seroincidence_by` *Estimate Seroincidence*

Description

Function to estimate seroincidences based on cross-sectional serology data and longitudinal response model.

Usage

```
est_seroincidence_by(
  pop_data,
  sr_params,
  noise_params,
  strata,
  curve_strata_varnames = strata,
  noise_strata_varnames = strata,
  antigen_isos = unique(pull(pop_data, "antigen_iso")),
  lambda_start = 0.1,
  build_graph = FALSE,
  num_cores = 1L,
  verbose = FALSE,
  print_graph = FALSE,
  cluster_var = NULL,
  stratum_var = NULL,
  ...
)
```

Arguments

pop_data	a data.frame with cross-sectional serology data per antibody and age, and additional columns corresponding to each element of the `strata` input
sr_params	a `data.frame()` containing MCMC samples of parameters from the Bayesian posterior distribution of a longitudinal decay curve model. The parameter columns must be named: • `antigen_iso`: a `character()` vector indicating antigen-isotype combinations • `iter`: an `integer()` vector indicating MCMC sampling iterations • `y0`: baseline antibody level at $t=0$ ($y(t=0)$) • `y1`: antibody peak level (ELISA units) • `t1`: duration of infection • `alpha`: antibody decay rate (1/days for the current longitudinal parameter sets) • `r`: shape factor of antibody decay
noise_params	a `data.frame()` (or `tibble::tibble()`) containing the following variables, specifying noise parameters for each antigen isotype: • `antigen_iso`: antigen isotype whose noise parameters are being specified on each row • `nu`: biological noise • `eps`: measurement noise • `y.low`: lower limit of detection for the current antigen isotype • `y.high`: upper limit of detection for the current antigen isotype
strata	a character vector of stratum-defining variables. Values must be variable names in `pop_data`.

curve_strata_varnames
: A subset of strata. Values must be variable names in curve_params. Default = "".

noise_strata_varnames
: A subset of strata. Values must be variable names in noise_params. Default = "".

antigen_isos
: Character vector with one or more antibody names. Must match pop_data

lambda_start
: starting guess for incidence rate, in events/year.

build_graph
: whether to graph the log-likelihood function across a range of incidence rates (lambda values)

num_cores
: Number of processor cores to use for calculations when computing by strata. If set to more than 1 and package **parallel** is available, then the computations are executed in parallel. Default = 1L.

verbose
: logical: if TRUE, print verbose log information to console

print_graph
: whether to display the log-likelihood curve graph in the course of running est_seroincidence()

cluster_var
: optional name(s) of the variable(s) in pop_data containing cluster identifiers for clustered sampling designs (e.g., households, schools). Can be a single variable name (character string) or a vector of variable names for multi-level clustering (e.g., c("school", "classroom")). When provided, standard errors will be adjusted for within-cluster correlation using cluster-robust variance estimation.

stratum_var
: optional name of the variable in pop_data containing stratum identifiers. Used in combination with cluster_var for stratified cluster sampling designs.

...
: Arguments passed on to est_seroincidence, stats::nlm

  stepmin A positive scalar providing the minimum allowable relative step length.

  sampling_weights optional data.frame containing sampling weights with columns for cluster/stratum identifiers and their sampling probabilities. Currently not implemented; reserved for future use.

  stepmax a positive scalar which gives the maximum allowable scaled step length. stepmax is used to prevent steps which would cause the optimization function to overflow, to prevent the algorithm from leaving the area of interest in parameter space, or to detect divergence in the algorithm. stepmax would be chosen small enough to prevent the first two of these occurrences, but should be larger than any anticipated reasonable step.

  typsize an estimate of the size of each parameter at the minimum.

  fscale an estimate of the size of f at the minimum.

  ndigit the number of significant digits in the function f.

  gradtol a positive scalar giving the tolerance at which the scaled gradient is considered close enough to zero to terminate the algorithm. The scaled gradient is a measure of the relative change in f in each direction p[i] divided by the relative change in p[i].

  iterlim a positive integer specifying the maximum number of iterations to be performed before the program is terminated.

  check.analyticals a logical scalar specifying whether the analytic gradients and Hessians, if they are supplied, should be checked against numerical derivatives at the initial parameter values. This can help detect incorrectly formulated gradients or Hessians.

Details

If `strata` is left empty, a warning will be produced, recommending that you use `est_seroincidence()` for unstratified analyses, and then the data will be passed to `est_seroincidence()`. If for some reason you want to use `est_seroincidence_by()` with no strata instead of calling `est_seroincidence()`, you may use `NA`, `NULL`, or `""` as the `strata` argument to avoid that warning.

Value

- if `strata` has meaningful inputs: An object of class `"seroincidence.by"`; i.e., a list of `"seroincidence"` objects from `est_seroincidence()`, one for each stratum, with some meta-data attributes.
- if `strata` is missing, `NULL`, `NA`, or `""`: An object of class `"seroincidence"`.

Examples

```
library(dplyr)

xs_data <-
  sees_pop_data_pk_100

curve <-
  typhoid_curves_nostrat_100 |>
  filter(antigen_iso %in% c("HlyE_IgA", "HlyE_IgG"))

noise <-
  example_noise_params_pk

est2 <- est_seroincidence_by(
  strata = "catchment",
  pop_data = xs_data,
  sr_params = curve,
  noise_params = noise,
  antigen_isos = c("HlyE_IgG", "HlyE_IgA"),
  # num_cores = 8 # Allow for parallel processing to decrease run time
  iterlim = 5 # limit iterations for the purpose of this example
)
print(est2)
summary(est2)
```

---

example_noise_params_pk

Small example of noise parameters for typhoid

Description

A subset of noise parameter estimates from the SEES study, for examples and testing, for Pakistan

Usage

```
example_noise_params_pk
```

Format

`example_noise_params_pk`**:**

A `curve_params` object (from `as_sr_params()`) with 4 rows and 7 columns:

antigen_iso which antigen and isotype are being measured (data is in long format)

Country Location for which the noise parameters were estimated

y.low Lower limit of detection

eps Measurement noise, defined by a CV (coefficient of variation) as the ratio of the standard deviation to the mean for replicates. Note that the CV should ideally be measured across plates rather than within the same plate.

nu Biological noise: error from cross-reactivity to other antibodies. It is defined as the 95th percentile of the distribution of antibody responses to the antigen-isotype in a population with no exposure.

y.high Upper limit of detection

Lab Lab for which noise was estimated.

Source

https://osf.io/rtw5k

---

example_noise_params_sees

Small example of noise parameters for typhoid

Description

A subset of noise parameter estimates from the SEES study, for examples and testing.

Usage

```
example_noise_params_sees
```

Format

`example_noise_params_pk`**:**

A `curve_params` object (from `as_sr_params()`) with 4 rows and 7 columns:

antigen_iso which antigen and isotype are being measured (data is in long format)

Country Location for which the noise parameters were estimated

y.low Lower limit of detection

eps Measurement noise, defined by a CV (coefficient of variation) as the ratio of the standard deviation to the mean for replicates. Note that the CV should ideally be measured across plates rather than within the same plate.

nu Biological noise: error from cross-reactivity to other antibodies. It is defined as the 95th percentile of the distribution of antibody responses to the antigen-isotype in a population with no exposure.

y.high Upper limit of detection

Lab Lab for which noise was estimated.

Source

https://osf.io/rtw5k

`get_biomarker_levels` *Extract biomarker levels*

Description

Extract biomarker levels

Usage

```
get_biomarker_levels(object, ...)
```

Arguments

`object`	a `pop_data` object
`...`	unused

Value

the biomarker levels in `object`

Examples

```
sees_pop_data_100 |> get_biomarker_levels()
```

`get_biomarker_names_var`
Get biomarker variable name

Description

Get biomarker variable name

Usage

```
get_biomarker_names_var(object, ...)
```

Arguments

object	a pop_data object
...	unused

Value

a character string identifying the biomarker names column in object

Examples

```
sees_pop_data_100 |> get_biomarker_names_var()
```

---

get_values *Get antibody measurement values*

Description

Get antibody measurement values

Usage

```
get_values(object, ...)
```

Arguments

object	a pop_data object
...	unused

Value

a numeric vector of antibody measurement values

Examples

```
sees_pop_data_100 |> get_values()
```

get_values_var *Extract antibody measurement values*

Description

Extract antibody measurement values

Usage

```
get_values_var(object, ...)
```

Arguments

object	a pop_data object
...	unused

Value

the name of the column in object specified as containing antibody abundance measurements

Examples

```
sees_pop_data_100 |> get_values_var()
```

graph.curve.params *Graph estimated antibody decay curves*

Description

Graph estimated antibody decay curves

Usage

```
graph.curve.params(
  object,
  antigen_isos = unique(object$antigen_iso),
  verbose = FALSE,
  quantiles = c(0.1, 0.5, 0.9),
  alpha_samples = 0.3,
  chain_color = TRUE,
  log_x = FALSE,
  log_y = TRUE,
  n_curves = 100,
  iters_to_graph = head(unique(object$iter), n_curves),
  ...
)
```

Arguments

object	a data.frame() containing MCMC samples of antibody decay curve parameters
antigen_isos	antigen isotypes to analyze (can subset object)
verbose	verbose output
quantiles	Optional numeric vector of point-wise (over time) quantiles to plot (e.g., 10%, 50%, and 90% = c(0.1, 0.5, 0.9)). If NULL, no quantile lines are shown.
alpha_samples	alpha parameter passed to ggplot2::geom_line (has no effect if iters_to_graph is empty)
chain_color	logical: if TRUE (default), MCMC chain lines are colored by chain. If FALSE, all MCMC chain lines are black.
log_x	should the x-axis be on a logarithmic scale (TRUE) or linear scale (FALSE, default)?
log_y	should the Y-axis be on a logarithmic scale (default, TRUE) or linear scale (FALSE)?
n_curves	how many curves to plot (see details).
iters_to_graph	which MCMC iterations in curve_params to plot (overrides n_curves).
...	not currently used

Details

n_curves and iters_to_graph:

In most cases, object will contain too many rows of MCMC samples for all of these samples to be plotted at once.

- Setting the n_curves argument to a value smaller than the number of rows in curve_params will cause this function to select the first n_curves rows to graph.
- Setting n_curves larger than the number of rows in ' will result all curves being plotted.
- If the user directly specifies the iters_to_graph argument, then n_curves has no effect.

Value

a ggplot2::ggplot() object showing the antibody dynamic kinetics of selected antigen/isotype combinations, with optional posterior distribution quantile curves.

Examples

```
# Load example dataset
curve <- typhoid_curves_nostrat_100 |>
  dplyr::filter(antigen_iso %in% c("HlyE_IgA", "HlyE_IgG"))

# Plot quantiles without showing all curves
plot1 <- graph.curve.params(curve, n_curves = 0)
print(plot1)

# Plot with additional quantiles and show all curves
plot2 <- graph.curve.params(
  curve,
```

```
  n_curves = Inf,
  quantiles = c(0.1, 0.5, 0.9)
)
print(plot2)

# Plot with MCMC chains in black
plot3 <- graph.curve.params(
  curve,
  n_curves = Inf,
  quantiles = c(0.1, 0.5, 0.9),
  chain_color = FALSE
)
print(plot3)
```

`graph_loglik` *Graph log-likelihood of data*

Description

Graph log-likelihood of data

Usage

```
graph_loglik(
  pop_data,
  curve_params,
  noise_params,
  antigen_isos = pop_data %>% get_biomarker_levels(),
  x = 10^seq(-3, 0, by = 0.1),
  highlight_points = NULL,
  highlight_point_names = "highlight_points",
  log_x = FALSE,
  previous_plot = NULL,
  curve_label = paste(antigen_isos, collapse = " + "),
  ...
)
```

Arguments

`pop_data`	a `data.frame()` with cross-sectional serology data by antibody and age, and additional columns
`curve_params`	a `data.frame()` containing MCMC samples of parameters from the Bayesian posterior distribution of a longitudinal decay curve model. The parameter columns must be named: • `antigen_iso`: a `character()` vector indicating antigen-isotype combinations • `iter`: an `integer()` vector indicating MCMC sampling iterations

- `y0`: baseline antibody level at $t=0$ ($y(t=0)$)
- `y1`: antibody peak level (ELISA units)
- `t1`: duration of infection
- `alpha`: antibody decay rate (1/days for the current longitudinal parameter sets)
- `r`: shape factor of antibody decay

`noise_params` a `data.frame()` (or `tibble::tibble()`) containing the following variables, specifying noise parameters for each antigen isotype:

- `antigen_iso`: antigen isotype whose noise parameters are being specified on each row
- `nu`: biological noise
- `eps`: measurement noise
- `y.low`: lower limit of detection for the current antigen isotype
- `y.high`: upper limit of detection for the current antigen isotype

`antigen_isos` Character vector listing one or more antigen isotypes. Values must match `pop_data`.

`x` sequence of lambda values to graph

`highlight_points` a possible highlighted value

`highlight_point_names` labels for highlighted points

`log_x` should the x-axis be on a logarithmic scale (`TRUE`) or linear scale (`FALSE`, default)?

`previous_plot` if not NULL, the current data is added to the existing graph

`curve_label` if not NULL, add a label for the curve

`...` Arguments passed on to `log_likelihood`

`verbose` logical: if TRUE, print verbose log information to console

Value

a `ggplot2::ggplot()`

Examples

```
library(dplyr)
library(tibble)

# Load cross-sectional data
xs_data <-
  sees_pop_data_pk_100

# Load curve parameters and subset for the purposes of this example
curve <-
  typhoid_curves_nostrat_100 %>%
  filter(antigen_iso %in% c("HlyE_IgA", "HlyE_IgG"))

# Load noise parameters
```

```
cond <- tibble(
  antigen_iso = c("HlyE_IgG", "HlyE_IgA"),
  nu = c(0.5, 0.5),                          # Biologic noise (nu)
  eps = c(0, 0),                             # M noise (eps)
  y.low = c(1, 1),                           # Low cutoff (llod)
  y.high = c(5e6, 5e6))                      # High cutoff (y.high)

# Graph the log likelihood
lik_HlyE_IgA <- # nolint: object_name_linter
  graph_loglik(
    pop_data = xs_data,
    curve_params = curve,
    noise_params = cond,
    antigen_isos = "HlyE_IgA",
    log_x = TRUE
)

lik_HlyE_IgA # nolint: object_name_linter
```

`load_noise_params` *Load noise parameters*

Description

Load noise parameters

Usage

```
load_noise_params(file_path, antigen_isos = NULL)
```

Arguments

`file_path`	path to an RDS file containing biologic and measurement noise of antibody decay curve parameters `y.low`, `eps`, `nu`, and `y.high`, stored as a `data.frame()` or tibble::tbl_df
`antigen_isos`	`character()` vector of antigen isotypes to be used in analyses

Value

a `noise` object (a tibble::tbl_df with extra attribute `antigen_isos`)

Examples

```
noise <- load_noise_params(
  serocalculator_example("example_noise_params.rds")
)
print(noise)
```

load_pop_data *Load a cross-sectional antibody survey data set*

Description

Load a cross-sectional antibody survey data set

Usage

```
load_pop_data(file_path, ...)
```

Arguments

file_path	path to an RDS file containing a cross-sectional antibody survey data set, stored as a data.frame() or tibble::tbl_df
...	Arguments passed on to as_pop_data data a data.frame() or tibble::tbl_df antigen_isos character() vector of antigen isotypes to be used in analyses age a character() identifying the age column id a character() identifying the id column value a character() identifying the value column standardize a logical() to determine standardization of columns

Value

a pop_data object (a tibble::tbl_df with extra attributes)

Examples

```
xs_data <- load_pop_data(serocalculator_example("example_pop_data.rds"))

print(xs_data)
```

load_sr_params *Load longitudinal seroresponse parameter samples*

Description

Load longitudinal seroresponse parameter samples

Usage

```
load_sr_params(file_path, antigen_isos = NULL)
```

Arguments

file_path	path to an RDS file containing MCMC samples of antibody seroresponse parameters y0, y1, t1, alpha, and r, stored as a data.frame() or tibble::tbl_df
antigen_isos	character() vector of antigen isotypes used in analyses

Value

a curve_params object (a tibble::tbl_df with extra attribute antigen_isos)

Examples

```
curve <- load_sr_params(serocalculator_example("example_curve_params.rds"))

print(curve)
```

---

log_likelihood *Calculate log-likelihood*

Description

Calculates the log-likelihood of a set of cross-sectional antibody response data, for a given incidence rate (lambda) value.

Usage

```
log_likelihood(
  lambda,
  pop_data,
  curve_params,
  noise_params,
  antigen_isos = get_biomarker_levels(pop_data),
  verbose = FALSE,
  ...
)
```

Arguments

lambda	a numeric vector of incidence parameters, in events per person-year
pop_data	a data.frame() with cross-sectional serology data by antibody and age, and additional columns
curve_params	a data.frame() containing MCMC samples of parameters from the Bayesian posterior distribution of a longitudinal decay curve model. The parameter columns must be named: • antigen_iso: a character() vector indicating antigen-isotype combinations

- `iter`: an `integer()` vector indicating MCMC sampling iterations
- `y0`: baseline antibody level at $t=0$ ($y(t=0)$)
- `y1`: antibody peak level (ELISA units)
- `t1`: duration of infection
- `alpha`: antibody decay rate (1/days for the current longitudinal parameter sets)
- `r`: shape factor of antibody decay

`noise_params` a `data.frame()` (or `tibble::tibble()`) containing the following variables, specifying noise parameters for each antigen isotype:

- `antigen_iso`: antigen isotype whose noise parameters are being specified on each row
- `nu`: biological noise
- `eps`: measurement noise
- `y.low`: lower limit of detection for the current antigen isotype
- `y.high`: upper limit of detection for the current antigen isotype

`antigen_isos` Character vector listing one or more antigen isotypes. Values must match `pop_data`.

`verbose` logical: if TRUE, print verbose log information to console

`...` additional arguments passed to other functions (not currently used).

Value

the log-likelihood of the data with the current parameter values

Examples

```
library(dplyr)
library(tibble)

# Load cross-sectional data
xs_data <-
  sees_pop_data_pk_100

# Load curve parameters and subset for the purposes of this example
curve <-
  typhoid_curves_nostrat_100 %>%
  filter(antigen_iso %in% c("HlyE_IgA", "HlyE_IgG"))

# Load noise params
cond <- tibble(
  antigen_iso = c("HlyE_IgG", "HlyE_IgA"),
  nu = c(0.5, 0.5), # Biologic noise (nu)
  eps = c(0, 0), # M noise (eps)
  y.low = c(1, 1), # low cutoff (llod)
  y.high = c(5e6, 5e6)
) # high cutoff (y.high)

# Calculate log-likelihood
ll_AG <- log_likelihood(
```

```
  pop_data = xs_data,
  curve_params = curve,
  noise_params = cond,
  antigen_isos = c("HlyE_IgG", "HlyE_IgA"),
  lambda = 0.1
) %>% print()
```

`sees_pop_data_100` *Small example cross-sectional data set*

Description

A subset of data from the SEES data, for examples and testing.

Usage

```
sees_pop_data_100
```

Format

`sees_pop_data_pk_100`**:**

A `pop_data` object (from `as_pop_data()`) with 200 rows and 8 columns:

id Observation ID

Country Country where the participant was living

cluster survey sampling cluster

catchment survey catchment area

age participant's age when sampled, in years

antigen_iso which antigen and isotype are being measured (data is in long format)

value concentration of antigen isotype, in ELISA units

Source

https://osf.io/n6cp3

sees_pop_data_pk_100 *Small example cross-sectional data set*

Description

A subset of data from the SEES data, for examples and testing, data from Pakistan only.

Usage

```
sees_pop_data_pk_100
```

Format

`sees_pop_data_pk_100`**:**
A `pop_data` object (from `as_pop_data()`) with 200 rows and 8 columns:

id Observation ID
Country Country where the participant was living
cluster survey sampling cluster
catchment survey catchment area
age participant's age when sampled, in years
antigen_iso which antigen and isotype are being measured (data is in long format)
value concentration of antigen isotype, in ELISA units

Source

https://osf.io/n6cp3

sees_typhoid_ests_strat *Example* `"seroincidence.by"` *object*

Description

Typhoid seroconversion rate estimates by country and age category from the SEES study.

Usage

```
sees_typhoid_ests_strat
```

Format

An object of class `seroincidence.by` (inherits from `list`) of length 9.

Source

`serocalculator/data-raw/sees_typhoid_ests_strat.R`

serocalculator_example *Get path to an example file*

Description

The serocalculator package comes bundled with a number of sample files in its `inst/extdata` directory. This `serocalculator_example()` function make those sample files easy to access.

Usage

```
serocalculator_example(file = NULL)
```

Arguments

`file`	Name of file. If `NULL`, the example files will be listed.

Details

Adapted from `readr::readr_example()` following the guidance in https://r-pkgs.org/data.html#sec-data-example-path-helper.

Value

a character string providing the path to the file specified by `file`, or a vector or available files if `file = NULL`.

Examples

```
serocalculator_example()
serocalculator_example("example_pop_data.csv")
```

sim_pop_data *Simulate a cross-sectional serosurvey with noise*

Description

Makes a cross-sectional data set (age, y(t) set) and adds noise, if desired.

Usage

```
sim_pop_data(
  lambda = 0.1,
  n_samples = 100,
  age_range = c(0, 20),
  age_fixed = NA,
 antigen_isos = intersect(get_biomarker_levels(curve_params), rownames(noise_limits)),
  n_mcmc_samples = 0,
  renew_params = FALSE,
  add_noise = FALSE,
  curve_params,
  noise_limits,
  format = "wide",
  verbose = FALSE,
  ...
)
```

Arguments

lambda	a numeric() scalar indicating the incidence rate (in events per person-years)
n_samples	number of samples to simulate
age_range	age range of sampled individuals, in years
age_fixed	specify the curve parameters to use by age (does nothing at present?)
antigen_isos	Character vector with one or more antibody names. Values must match curve_params.
n_mcmc_samples	how many MCMC samples to use: • when n_mcmc_samples is in 1:4000 a fixed posterior sample is used • when n_mcmc_samples = 0, a random sample is chosen
renew_params	whether to generate a new parameter set for each infection • renew_params = TRUE generates a new parameter set for each infection • renew_params = FALSE keeps the one selected at birth, but updates baseline y0
add_noise	a logical() indicating whether to add biological and measurement noise
curve_params	a data.frame() containing MCMC samples of parameters from the Bayesian posterior distribution of a longitudinal decay curve model. The parameter columns must be named: • antigen_iso: a character() vector indicating antigen-isotype combinations • iter: an integer() vector indicating MCMC sampling iterations • y0: baseline antibody level at $t=0$ ($y(t=0)$) • y1: antibody peak level (ELISA units) • t1: duration of infection • alpha: antibody decay rate (1/days for the current longitudinal parameter sets) • r: shape factor of antibody decay

noise_limits	biologic noise distribution parameters
format	a character() variable, containing either: • "long" (one measurement per row) or • "wide" (one serum sample per row)
verbose	logical: if TRUE, print verbose log information to console
...	Arguments passed on to simcs.tinf, ldpar, ab, mk_baseline age age at infection nmc mcmc sample to use npar number of parameters t numeric vector of elapsed times since start of infection par numeric matrix of model parameters: • rows are parameters • columns are biomarkers kab integer indicating which row to read from blims n number of observations blims range of possible baseline antibody levels

Value

a tibble::tbl_df containing simulated cross-sectional serosurvey data, with columns:

- age: age (in days)
- one column for each element in the antigen_iso input argument

Examples

```
# Load curve parameters
dmcmc <- typhoid_curves_nostrat_100

# Specify the antibody-isotype responses to include in analyses
antibodies <- c("HlyE_IgA", "HlyE_IgG")

# Set seed to reproduce results
set.seed(54321)

# Simulated incidence rate per person-year
lambda <- 0.2
# Range covered in simulations
lifespan <- c(0, 10)
# Cross-sectional sample size
nrep <- 100

# Biologic noise distribution
dlims <- rbind(
  "HlyE_IgA" = c(min = 0, max = 0.5),
  "HlyE_IgG" = c(min = 0, max = 0.5)
)
```

```
# Generate cross-sectional data
csdata <- sim_pop_data(
  curve_params = dmcmc,
  lambda = lambda,
  n_samples = nrep,
  age_range = lifespan,
  antigen_isos = antibodies,
  n_mcmc_samples = 0,
  renew_params = TRUE,
  add_noise = TRUE,
  noise_limits = dlims,
  format = "long"
)
```

`sim_pop_data_multi` *Simulate multiple data sets*

Description

Simulate multiple data sets

Usage

```
sim_pop_data_multi(
  nclus = 10,
  sample_sizes = 100,
  lambdas = c(0.05, 0.1, 0.15, 0.2, 0.3),
  num_cores = 2L,
  rng_seed = 1234,
  verbose = FALSE,
  ...
)
```

Arguments

`nclus`	number of clusters
`sample_sizes`	sample sizes to simulate
`lambdas`	incidence rate, in events/person*year
`num_cores`	number of cores to use for parallel computations
`rng_seed`	starting seed for random number generator, passed to `rngtools::RNGseq()`
`verbose`	whether to report verbose information
`...`	Arguments passed on to `sim_pop_data` `lambda` a `numeric()` scalar indicating the incidence rate (in events per person-years) `n_samples` number of samples to simulate

`age_range` age range of sampled individuals, in years

`age_fixed` specify the curve parameters to use by age (does nothing at present?)

`antigen_isos` Character vector with one or more antibody names. Values must match `curve_params`.

`n_mcmc_samples` how many MCMC samples to use:

- when `n_mcmc_samples` is in `1:4000` a fixed posterior sample is used
- when `n_mcmc_samples = 0`, a random sample is chosen

`renew_params` whether to generate a new parameter set for each infection

- `renew_params = TRUE` generates a new parameter set for each infection
- `renew_params = FALSE` keeps the one selected at birth, but updates baseline y0

`add_noise` a `logical()` indicating whether to add biological and measurement noise

`noise_limits` biologic noise distribution parameters

`format` a `character()` variable, containing either:

- `"long"` (one measurement per row) or
- `"wide"` (one serum sample per row)

`curve_params` a `data.frame()` containing MCMC samples of parameters from the Bayesian posterior distribution of a longitudinal decay curve model. The parameter columns must be named:

- `antigen_iso`: a `character()` vector indicating antigen-isotype combinations
- `iter`: an `integer()` vector indicating MCMC sampling iterations
- `y0`: baseline antibody level at $t=0$ ($y(t=0)$)
- `y1`: antibody peak level (ELISA units)
- `t1`: duration of infection
- `alpha`: antibody decay rate (1/days for the current longitudinal parameter sets)
- `r`: shape factor of antibody decay

Value

a `tibble::tibble()`

Examples

```
# Load curve parameters
dmcmc <- typhoid_curves_nostrat_100

# Specify the antibody-isotype responses to include in analyses
antibodies <- c("HlyE_IgA", "HlyE_IgG")

# Set seed to reproduce results
set.seed(54321)

# Simulated incidence rate per person-year
lambdas = c(.05, .1, .15, .2, .3)
```

```
# Range covered in simulations
lifespan <- c(0, 10);

# Cross-sectional sample size
nrep <- 100

# Biologic noise distribution
dlims <- rbind(
  "HlyE_IgA" = c(min = 0, max = 0.5),
  "HlyE_IgG" = c(min = 0, max = 0.5)
)

sim_data <- sim_pop_data_multi(
  curve_params = dmcmc,
  lambdas = lambdas,
  sample_sizes = nrep,
  age_range = lifespan,
  antigen_isos = antibodies,
  n_mcmc_samples = 0,
  renew_params = TRUE,
  add_noise = TRUE,
  noise_limits = dlims,
  format = "long",
  nclus = 10,
  num_cores = 2)

sim_data
```

`strata` *Extract* `Strata` *metadata from an object*

Description

Generic method for extracting strata metadata from objects. See `strata.default()`

Usage

```
strata(x)
```

Arguments

`x`	an object

Value

the strata metadata of `x`

summary.seroincidence *Summarizing fitted seroincidence models*

Description

This function is a summary() method for seroincidence objects. When the model was fit with clustered data (using the cluster_var parameter in est_seroincidence()), this function automatically computes cluster-robust standard errors to account for within-cluster correlation.

Usage

```
## S3 method for class 'seroincidence'
summary(object, coverage = 0.95, verbose = TRUE, ...)
```

Arguments

object	a list() outputted by stats::nlm() or est_seroincidence()
coverage	desired confidence interval coverage probability
verbose	whether to produce verbose messaging
...	unused

Value

a tibble::tibble() containing the following:

- est.start: the starting guess for incidence rate
- ageCat: the age category we are analyzing
- incidence.rate: the estimated incidence rate, per person year
- SE: standard error of the incidence rate estimate
- CI.lwr: lower limit of confidence interval for incidence rate
- CI.upr: upper limit of confidence interval for incidence rate
- se_type: type of standard error used ("standard" or "cluster-robust")
- coverage: coverage probability
- log.lik: log-likelihood of the data used in the call to est_seroincidence(), evaluated at the maximum-likelihood estimate of lambda (i.e., at incidence.rate)
- iterations: the number of iterations used
- antigen_isos: a list of antigen isotypes used in the analysis
- nlm.convergence.code: information about convergence of the likelihood maximization procedure performed by nlm() (see "Value" section of stats::nlm(), component code); codes 3-5 indicate issues:
  - 1: relative gradient is close to zero, current iterate is probably solution.
  - 2: successive iterates within tolerance, current iterate is probably solution.

- 3: Last global step failed to locate a point lower than x. Either x is an approximate local minimum of the function, the function is too non-linear for this algorithm, or `stepmin` in `est_seroincidence()` (a.k.a., `steptol` in `stats::nlm()`) is too large.
- 4: iteration limit exceeded; increase `iterlim`.
- 5: maximum step size `stepmax` exceeded five consecutive times. Either the function is unbounded below, becomes asymptotic to a finite value from above in some direction, or `stepmax` is too small.

Examples

```
library(dplyr)

xs_data <-
  sees_pop_data_pk_100

curve <-
  typhoid_curves_nostrat_100 |>
  filter(antigen_iso %in% c("HlyE_IgA", "HlyE_IgG"))

noise <-
  example_noise_params_pk

est1 <- est_seroincidence(
  pop_data = xs_data,
  sr_params = curve,
  noise_params = noise,
  antigen_isos = c("HlyE_IgG", "HlyE_IgA")
)

summary(est1)
```

---

summary.seroincidence.by

Summary Method for "seroincidence.by" Objects

Description

Calculate seroincidence from output of the seroincidence calculator `est_seroincidence_by()`.

Usage

```
## S3 method for class 'seroincidence.by'
summary(
  object,
  confidence_level = 0.95,
  show_deviance = TRUE,
  show_convergence = TRUE,
  verbose = FALSE,
  ...
)
```

Arguments

object	A dataframe containing output of est_seroincidence_by().
confidence_level	desired confidence interval coverage probability
show_deviance	Logical flag (FALSE/TRUE) for reporting deviance (-2*log(likelihood) at estimated seroincidence. Default = TRUE.
show_convergence	Logical flag (FALSE/TRUE) for reporting convergence (see help for optim() for details). Default = FALSE.
verbose	a logical scalar indicating whether to print verbose messages to the console
...	Additional arguments affecting the summary produced.

Value

A summary.seroincidence.by object, which is a tibble::tibble, with the following columns:

- incidence.rate maximum likelihood estimate of lambda (seroincidence)
- CI.lwr lower confidence bound for lambda
- CI.upr upper confidence bound for lambda
- Deviance (included if show_deviance = TRUE) Negative log likelihood (NLL) at estimated (maximum likelihood) lambda)
- nlm.convergence.code (included if show_convergence = TRUE) Convergence information returned by stats::nlm()

The object also has the following metadata (accessible through base::attr()):

- antigen_isos Character vector with names of input antigen isotypes used in est_seroincidence_by()
- Strata Character with names of strata used in est_seroincidence_by()

Examples

```
library(dplyr)

xs_data <-
  sees_pop_data_pk_100

curve <-
  typhoid_curves_nostrat_100 |>
  filter(antigen_iso %in% c("HlyE_IgA", "HlyE_IgG"))

noise <-
  example_noise_params_pk

# estimate seroincidence
est2 <- est_seroincidence_by(
  strata = c("catchment"),
  pop_data = xs_data,
  sr_params = curve,
```

```
  noise_params = noise,
  antigen_isos = c("HlyE_IgG", "HlyE_IgA"),
  # num_cores = 8 # Allow for parallel processing to decrease run time
)

# calculate summary statistics for the seroincidence object
summary(est2)
```

`typhoid_curves_nostrat_100` *Small example of antibody response curve parameters for typhoid*

Description

A subset of data from the SEES study, for examples and testing.

Usage

```
typhoid_curves_nostrat_100
```

Format

`typhoid_curves_nostrat_100`**:**

A `curve_params` object (from `as_sr_params()`) with 500 rows and 7 columns:

antigen_iso which antigen and isotype are being measured (data is in long format)

iter MCMC iteration

y0 Antibody concentration at t = 0 (start of active infection)

y1 Antibody concentration at t = `t1` (end of active infection)

t1 Duration of active infection

alpha Antibody decay rate coefficient

r Antibody decay rate exponent parameter

Source

https://osf.io/rtw5k

Index