# Package: serocalculator (via r-universe)

August 21, 2024

**Type** Package

**Title** Estimating Infection Rates from Serological Data

**Version** 1.0.3

**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. Methods originally published in Simonsen et al. (2009) <doi:10.1002/sim.3592> and Teunis et al. (2012) <doi:10.1002/sim.5322>, and further developed in subsequent publications by de Graaf et al. (2014) <doi:10.1016/j.epidem.2014.08.002>, Teunis et al. (2016) <doi:10.1016/j.epidem.2016.04.001>, and Teunis et al. (2020) <doi:10.1002/sim.8578>.

**Depends** R (>= 4.1.0)

**License** GPL-3

**Imports** doParallel, dplyr (>= 1.1.1), foreach, ggplot2, ggpubr, lifecycle, magrittr, mixtools, Rcpp, rlang, rngtools, scales, stats, tibble, tidyr, utils, cli

**Suggests** parallel, knitr, rmarkdown, tidyverse, fs, testthat (>= 3.0.0), readr, bookdown, ggbeeswarm, DT

**Encoding** UTF-8

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

**RoxygenNote** 7.3.1

**NeedsCompilation** yes

**LinkingTo** Rcpp

**Language** en-US

**Config/testthat/edition** 3

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

**Maintainer** Kristina Lai <kwlai@ucdavis.edu>

**Repository** CRAN

**Date/Publication** 2024-07-21 15:40:02 UTC

# Contents

## ab *kinetics of the antibody (ab) response (power function decay)*

### Description

kinetics of the antibody (ab) response (power function decay)

### Usage

```
ab(t, par, ...)
```

### Arguments

| | |
|---|---|
| `t` | age at infection? |
| `par` | parameters |
| `...` | arguments passed to `baseline()` |

### Value

a `matrix()`

## `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,
  antigen_isos = unique(object$antigen_iso),
  ncol = min(3, length(antigen_isos)),
  ...
)
```

## Arguments

| | |
|---|---|
| object | a data.frame() of curve parameters (one or more MCMC samples) |
| antigen_isos | antigen isotypes to analyze (can be used to subset curve_params) |
| ncol | how many columns of subfigures to use in panel plot |
| ... | Arguments passed on to plot_curve_params_one_ab<br>verbose verbose output<br>xlim range of x values to graph<br>n_curves how many curves to plot (see details).<br>n_points Number of points to interpolate along the x axis (passed to ggplot2::geom_function())<br>rows_to_graph which rows of curve_params to plot (overrides n_curves).<br>alpha (passed to ggplot2::geom_function()) how transparent the curves should be:<br>• 0 = fully transparent (invisible)<br>• 1 = fully opaque<br>log_x should the x-axis be on a logarithmic scale (TRUE) or linear scale (FALSE, default)?<br>log_y should the Y-axis be on a logarithmic scale (default, TRUE) or linear scale (FALSE)? |

## Details

rows_to_graph:

Note that if you directly specify rows_to_graph when calling this function, the row numbers are enumerated separately for each antigen isotype; in other words, for the purposes of this argument, row numbers start over at 1 for each antigen isotype. There is currently no way to specify different row numbers for different antigen isotypes; if you want to do that, you will could call plot_curve_params_one_ab() directly for each antigen isotype and combine the resulting panels yourself. Or you could subset curve_params manually, before passing it to this function, and set the n_curves argument to Inf.

## Value

a ggplot2::ggplot() object

## Examples

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

curve = load_curve_params("https://osf.io/download/rtw5k/") %>%
  filter(antigen_iso %in% c("HlyE_IgA", "HlyE_IgG")) %>%
  slice(1:100, .by = antigen_iso)  %>% # Reduce dataset for the purposes of this example
  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)

xs_data <- "https://osf.io/download//n6cp3/" %>%
  load_pop_data() %>%
  clean_pop_data()

xs_data %>% autoplot(strata = "Country", type = "density")
xs_data %>% autoplot(strata = "Country", 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.incidence()`) |
| `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 <- load_pop_data("https://osf.io/download//n6cp3/") %>%
  clean_pop_data()

curve <- load_curve_params("https://osf.io/download/rtw5k/") %>%
  filter(antigen_iso %in% c("HlyE_IgA", "HlyE_IgG")) %>%
  slice(1:100, .by = antigen_iso) # Reduce dataset for the purposes of this example

noise <- load_noise_params("https://osf.io/download//hqy4v/")

est1 <- est.incidence(
  pop_data = xs_data %>% filter(Country == "Pakistan"),
  curve_param = curve,
  noise_param = noise %>% filter(Country == "Pakistan"),
  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.incidence.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

an object of class "ggarrange", which is a `ggplot2::ggplot()` or a `list()` of `ggplot2::ggplot()`s.

## Examples

```
library(dplyr)
library(ggplot2)

xs_data <- "https://osf.io/download//n6cp3/" %>%
  load_pop_data() %>%
  clean_pop_data

curve <- load_curve_params("https://osf.io/download/rtw5k/") %>%
  filter(antigen_iso %in% c("HlyE_IgA", "HlyE_IgG")) %>%
  slice(1:100, .by = antigen_iso) # Reduce dataset for the purposes of this example

noise <- load_noise_params("https://osf.io/download//hqy4v/")


est2 <- est.incidence.by(
  strata = c("catchment"),
  pop_data = xs_data %>% filter(Country == "Pakistan"),
  curve_params = curve,
```

```
  noise_params = noise %>% filter(Country == "Pakistan"),
  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.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, xvar, alpha = 0.7, shape = 1, width = 0.001, ...)
```

## Arguments

| | |
|---|---|
| `object` | a `summary.seroincidence.by` object (generated by applying the `summary()` method to the output of `est.incidence.by()`). |
| `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()` |
| `width` | width for jitter |
| `...` | unused |

## Value

a `ggplot2::ggplot()` object

## Examples

```
library(dplyr)
library(ggplot2)


xs_data <- load_pop_data("https://osf.io/download//n6cp3/") %>%
  clean_pop_data()
```

```
curve <- load_curve_params("https://osf.io/download/rtw5k/") %>%
  filter(antigen_iso %in% c("HlyE_IgA", "HlyE_IgG")) %>%
  slice(1:100, .by = antigen_iso) # Reduce dataset for the purposes of this example

noise <- load_noise_params("https://osf.io/download//hqy4v/")

est2 <- est.incidence.by(
  strata = c("catchment"),
  pop_data = xs_data %>% filter(Country == "Pakistan"),
  curve_params = curve,
  noise_params = noise %>% filter(Country == "Pakistan"),
  antigen_isos = c("HlyE_IgG", "HlyE_IgA"),
  #num_cores = 8 #Allow for parallel processing to decrease run time
)

est2sum <- summary(est2)

autoplot(est2sum, "catchment")
```

---

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)
```

## Arguments

| | |
|---|---|
| pop_data | dataset to check |

## Value

NULL (invisibly)

## Examples

```
library(dplyr)

# Import cross-sectional data from OSF and rename required variables
xs_data <- load_pop_data("https://osf.io/download//n6cp3/") %>%
  clean_pop_data() %>%
  check_pop_data()
```

`clean_pop_data` *Reformat a cross-sectional antibody survey dataset*

## Description

Reformat a cross-sectional antibody survey dataset

## Usage

```
clean_pop_data(pop_data)
```

## Arguments

| | |
|---|---|
| `pop_data` | a `data.frame()` containing the following columns:<br>• `index_id`: a `character()` variable identifying multiple rows of data from the same person<br>• `antigen_isos`: a `character()` variable indicating the antigen-isotype measured<br>• `result`: the measured antibody concentration<br>• `Age`: age of the individual whose serum has been assayed, at the time of blood sample collection. |

## Value

A data.frame (or tibble::tbl_df) containing the following columns:

- `id`: a `character()` variable identifying multiple rows of data from the same person
- `antigen_isos`: a `character()` variable indicating the antigen-isotype measured
- `value`: the measured antibody concentration
- `age`: age of the individual whose serum has been assayed, at the time of blood sample

## Examples

```
xs_data <- load_pop_data("https://osf.io/download//n6cp3/")

clean_pop_data(xs_data)
```

## df_to_array *Convert a data.frame (or tibble) into a multidimensional array*

### Description

Convert a data.frame (or tibble) into a multidimensional array

### Usage

```
df_to_array(df, dim_var_names, value_var_name = "value")
```

### Arguments

| | |
|---|---|
| df | a data.frame() (or tibble::tibble()) in long format (each row contains one value for the intended array) |
| dim_var_names | a character() vector of variable names in df. All of these variables should be factors, or a warning will be produced. |
| value_var_name | a character() variable containing a variable name from df which contains the values for the intended array. |

### Value

an array() with dimensions defined by the variables in df listed in dim_var_names

## est.incidence *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.incidence(
  pop_data,
  curve_params,
  noise_params,
  antigen_isos = pop_data$antigen_iso %>% unique(),
  lambda_start = 0.1,
  stepmin = 1e-08,
  stepmax = 3,
  verbose = FALSE,
  build_graph = FALSE,
  print_graph = build_graph & verbose,
  ...
)
```

## Arguments

| | |
|---|---|
| pop_data | data.frame() with cross-sectional serology data per 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:<br>• antigen_iso: a character() vector indicating antigen-isotype combinations<br>• iter: an integer() vector indicating MCMC sampling iterations<br>• y0: baseline antibody level at $t = 0$ ($y(t = 0)$)<br>• y1: antibody peak level (ELISA units)<br>• t1: duration of infection<br>• alpha: antibody decay rate (1/days for the current longitudinal parameter sets)<br>• 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:<br>• antigen_iso: antigen isotype whose noise parameters are being specified on each row<br>• nu: biological noise<br>• eps: measurement noise<br>• y.low: lower limit of detection for the current antigen isotype<br>• y.high: upper limit of detection for the current antigen isotype |
| antigen_isos | Character vector with one or more antibody names. Values must match pop_data |
| lambda_start | starting guess for incidence rate, in years/event. |
| 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.incidence() |
| ... | Arguments passed on to stats::nlm<br>typsize an estimate of the size of each parameter at the minimum.<br>fscale an estimate of the size of f at the minimum.<br>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 meta-data attributes `lambda_start`, `antigen_isos`, and `ll_graph`

## Examples

```
library(dplyr)

xs_data <- load_pop_data("https://osf.io/download//n6cp3/") %>%
  clean_pop_data()

curve <- load_curve_params("https://osf.io/download/rtw5k/") %>%
  filter(antigen_iso %in% c("HlyE_IgA", "HlyE_IgG")) %>%
  slice(1:100, .by = antigen_iso) # Reduce dataset for the purposes of this example

noise <- load_noise_params("https://osf.io/download//hqy4v/")


est1 <- est.incidence(
  pop_data = xs_data %>% filter(Country == "Pakistan"),
  curve_param = curve,
  noise_param = noise %>% filter(Country == "Pakistan"),
  antigen_isos = c("HlyE_IgG", "HlyE_IgA")
)

summary(est1)
```

---

`est.incidence.by` *Estimate Seroincidence*

---

## Description

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

## Usage

```
est.incidence.by(
  pop_data,
  curve_params,
  noise_params,
  strata,
  curve_strata_varnames = strata,
  noise_strata_varnames = strata,
  antigen_isos = pop_data %>% pull("antigen_iso") %>% unique(),
  lambda_start = 0.1,
  build_graph = FALSE,
  num_cores = 1L,
  verbose = FALSE,
  print_graph = FALSE,
  ...
)
```

## Arguments

| | |
|---|---|
| pop_data | data.frame() with cross-sectional serology data per antibody and age, and additional columns to identify possible strata. |
| 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:<br>• antigen_iso: a character() vector indicating antigen-isotype combinations<br>• iter: an integer() vector indicating MCMC sampling iterations<br>• y0: baseline antibody level at $t = 0$ ($y(t = 0)$)<br>• y1: antibody peak level (ELISA units)<br>• t1: duration of infection<br>• alpha: antibody decay rate (1/days for the current longitudinal parameter sets)<br>• 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:<br>• antigen_iso: antigen isotype whose noise parameters are being specified on each row<br>• nu: biological noise<br>• eps: measurement noise<br>• y.low: lower limit of detection for the current antigen isotype<br>• y.high: upper limit of detection for the current antigen isotype |
| strata | 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. Values must match pop_data

lambda_start
: starting guess for incidence rate, in years/event.

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.incidence()

...
: Arguments passed on to est.incidence, stats::nlm

  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.

  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.incidence() for unstratified analyses, and then the data will be passed to est.incidence(). If for some reason you want to use est.incidence.by() with no strata instead of calling est.incidence(), 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.incidence(), one for each stratum, with some metadata attributes.
- if strata is missing, NULL, NA, or "": An object of class "seroincidence".

## Examples

```
library(dplyr)

xs_data <- load_pop_data("https://osf.io/download//n6cp3/") %>%
  clean_pop_data()

curve <- load_curve_params("https://osf.io/download/rtw5k/") %>%
  filter(antigen_iso %in% c("HlyE_IgA", "HlyE_IgG")) %>%
  slice(1:100, .by = antigen_iso) # Reduce dataset for the purposes of this example

noise <- load_noise_params("https://osf.io/download//hqy4v/")


est2 <- est.incidence.by(
  strata = c("catchment"),
  pop_data = xs_data %>% filter(Country == "Pakistan"),
  curve_params = curve,
  noise_params = noise %>% filter(Country == "Pakistan"),
  antigen_isos = c("HlyE_IgG", "HlyE_IgA"),
  #num_cores = 8 # Allow for parallel processing to decrease run time
)

summary(est2)
```

---

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

---

## Description

Graph estimated antibody decay curve

## Usage

```
graph.curve.params(
  curve_params,
  antigen_isos = unique(curve_params$antigen_iso),
  verbose = FALSE
)
```

## Arguments

| | |
|---|---|
| curve_params | a data.frame() containing MCMC samples of antibody decay curve parameters |
| antigen_isos | antigen isotypes |
| verbose | verbose output |

## Value

a ggplot2::ggplot() object

## Examples

```
curve_params <- readRDS(url("https://osf.io/download/rtw5k/"))

plot1 <- graph.curve.params(curve_params)

print(plot1)
```

---

# graph.loglik *Graph log-likelihood of data*

---

## Description

Graph log-likelihood of data

## Usage

```
graph.loglik(
  pop_data,
  curve_params,
  noise_params,
  antigen_isos,
  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 per 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:<br>• antigen_iso: a character() vector indicating antigen-isotype combinations<br>• iter: an integer() vector indicating MCMC sampling iterations<br>• 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 <- load_pop_data("https://osf.io/download//n6cp3/") %>%
  clean_pop_data()

# Load curve parameters and subset for the purposes of this example
dmcmc <- load_curve_params("https://osf.io/download/rtw5k/") %>%
  filter(antigen_iso %in% c("HlyE_IgA", "HlyE_IgG")) %>%
  slice(1:100, .by = antigen_iso)

# 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 <- graph.loglik(
  pop_data = xs_data,
  curve_params = dmcmc,
  noise_params = cond,
  antigen_isos = "HlyE_IgA",
  log_x = TRUE
)

lik_HlyE_IgA
```

---

`load_curve_params` *Load antibody decay curve parameter samples*

---

## Description

Load antibody decay curve parameter samples

## Usage

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

## Arguments

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

## Value

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

## Examples

```
curve <- load_curve_params("https://osf.io/download/rtw5k/")

print(curve)
```

## 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("https://osf.io/download//hqy4v/")

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, antigen_isos = NULL)
```

### 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 |
| antigen_isos | character() vector of antigen isotypes to be used in analyses |

## Value

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

## Examples

```
xs_data <- load_pop_data("https://osf.io/download//n6cp3/")

print(xs_data)
```

---

# 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,
  antigen_isos,
  curve_params,
  noise_params,
  verbose = FALSE,
  ...
)
```

## Arguments

| | |
|---|---|
| lambda | numeric() incidence parameter, in events per person-year |
| pop_data | a data.frame() with cross-sectional serology data per antibody and age, and additional columns |
| antigen_isos | Character vector listing one or more antigen isotypes. Values must match pop_data. |
| 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:<br>• antigen_iso: a character() vector indicating antigen-isotype combinations<br>• iter: an integer() vector indicating MCMC sampling iterations<br>• y0: baseline antibody level at $t = 0$ ($y(t = 0)$)<br>• 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

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 in longitudinal parameters
dmcmc = load_curve_params("https://osf.io/download/rtw5k")

xs_data <- "https://osf.io/download//n6cp3/" %>%
load_pop_data() %>%
clean_pop_data()

#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 = dmcmc,
  noise_params = cond,
  antigen_isos = c("HlyE_IgG","HlyE_IgA"),
  lambda = 0.1) %>% print()
```

## mk_baseline *generate random sample from baseline distribution*

### Description

generate random sample from baseline distribution

### Usage

```
mk_baseline(kab, n = 1, blims, ...)
```

### Arguments

| | |
|---|---|
| kab | index for which row of antibody baseline limits to read from blims |
| n | number of observations |
| blims | range of possible baseline antibody levels |
| ... | not currently used |

### Value

a numeric() vector

## plot_curve_params_one_ab *Graph an antibody decay curve model*

### Description

Graph an antibody decay curve model

### Usage

```
plot_curve_params_one_ab(
  object,
  verbose = FALSE,
  alpha = 0.4,
  n_curves = 100,
  n_points = 1000,
  log_x = FALSE,
  log_y = TRUE,
  rows_to_graph = 1:min(n_curves, nrow(object)),
  xlim = c(10^-1, 10^3.1),
  ...
)
```

## Arguments

| | |
|---|---|
| object | a data.frame() of curve parameters (one or more MCMC samples) |
| verbose | verbose output |
| alpha | (passed to ggplot2::geom_function()) how transparent the curves should be:<br>• 0 = fully transparent (invisible)<br>• 1 = fully opaque |
| n_curves | how many curves to plot (see details). |
| n_points | Number of points to interpolate along the x axis (passed to ggplot2::geom_function()) |
| 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)? |
| rows_to_graph | which rows of curve_params to plot (overrides n_curves). |
| xlim | range of x values to graph |
| ... | Arguments passed on to ggplot2::geom_function<br>mapping Set of aesthetic mappings created by aes(). If specified and inherit.aes = TRUE (the default), it is combined with the default mapping at the top level of the plot. You must supply mapping if there is no plot mapping.<br>data Ignored by stat_function(), do not use.<br>stat The statistical transformation to use on the data for this layer. When using a geom_*() function to construct a layer, the stat argument can be used the override the default coupling between geoms and stats. The stat argument accepts the following:<br>• A Stat ggproto subclass, for example StatCount.<br>• A string naming the stat. To give the stat as a string, strip the function name of the stat_ prefix. For example, to use stat_count(), give the stat as "count".<br>• For more information and other ways to specify the stat, see the layer stat documentation.<br>position A position adjustment to use on the data for this layer. This can be used in various ways, including to prevent overplotting and improving the display. The position argument accepts the following:<br>• The result of calling a position function, such as position_jitter(). This method allows for passing extra arguments to the position.<br>• A string naming the position adjustment. To give the position as a string, strip the function name of the position_ prefix. For example, to use position_jitter(), give the position as "jitter".<br>• For more information and other ways to specify the position, see the layer position documentation.<br>na.rm If FALSE, the default, missing values are removed with a warning. If TRUE, missing values are silently removed.<br>show.legend logical. Should this layer be included in the legends? NA, the default, includes if any aesthetics are mapped. FALSE never includes, and TRUE always includes. It can also be a named logical vector to finely select the aesthetics to display. |

`inherit.aes` If `FALSE`, overrides the default aesthetics, rather than combining with them. This is most useful for helper functions that define both data and aesthetics and shouldn't inherit behaviour from the default plot specification, e.g. `borders()`.

## Details

`n_curves` **and** `rows_to_graph`**:**

In most cases, `curve_params` 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 `rows_to_graph` argument, then `n_curves` has no effect.

## Value

a `ggplot2::ggplot()` object

## Examples

```
library(dplyr) # loads the `%>%` operator and `dplyr::filter()`

load_curve_params("https://osf.io/download/rtw5k/") %>%
  dplyr::filter(antigen_iso == "HlyE_IgG") %>%
  serocalculator::plot_curve_params_one_ab()
```

---

# `print.seroincidence` *Print Method for* `seroincidence` *Object*

---

## Description

Custom `print()` function to show output of the seroincidence calculator `est.incidence()`.

## Usage

```
## S3 method for class 'seroincidence'
print(x, ...)
```

## Arguments

| | |
|---|---|
| `x` | A list containing output of function `est.incidence.by()`. |
| `...` | Additional arguments affecting the summary produced. |

## Value

A `list` containing the output of the function `est.incidence()`.

## Examples

```
library(tidyverse)

xs_data <- load_pop_data("https://osf.io/download//n6cp3/") %>%
  clean_pop_data()

curve <- load_curve_params("https://osf.io/download/rtw5k/") %>%
  filter(antigen_iso %in% c("HlyE_IgA", "HlyE_IgG")) %>%
  slice(1:100, .by = antigen_iso) # Reduce dataset for the purposes of this example

noise <- load_noise_params("https://osf.io/download//hqy4v/")


est1 <- est.incidence(
  pop_data = xs_data %>% filter(Country == "Pakistan"),
  curve_param = curve,
  noise_param = noise %>% filter(Country == "Pakistan"),
  antigen_isos = c("HlyE_IgG", "HlyE_IgA")
)

print(est1)
```

---

# print.seroincidence.by

*Print Method for* seroincidence.by *Object*

---

## Description

Custom print() function to show output of the seroincidence calculator est.incidence.by().

## Usage

```
## S3 method for class 'seroincidence.by'
print(x, ...)
```

## Arguments

| | |
|---|---|
| x | A list containing output of function est.incidence.by(). |
| ... | Additional arguments affecting the summary produced. |

## Value

A list containing the output of the function est.incidence.by().

## Examples

```
library(tidyverse)

xs_data <- load_pop_data("https://osf.io/download//n6cp3/") %>%
  clean_pop_data()

curve <- load_curve_params("https://osf.io/download/rtw5k/") %>%
  filter(antigen_iso %in% c("HlyE_IgA", "HlyE_IgG")) %>%
  slice(1:100, .by = antigen_iso) # Reduce dataset for the purposes of this example

noise <- load_noise_params("https://osf.io/download//hqy4v/")


est2 <- est.incidence.by(
  strata = c("catchment"),
  pop_data = xs_data %>% filter(Country == "Pakistan"),
  curve_params = curve,
  noise_params = noise %>% filter(Country == "Pakistan"),
  antigen_isos = c("HlyE_IgG", "HlyE_IgA"),
  #num_cores = 8 # Allow for parallel processing to decrease run time
)

print(est2)
```

---

# print.summary.seroincidence.by

*Print Method for Seroincidence Summary Object*

---

## Description

Custom `print()` function for "summary.seroincidence.by" objects (constructed by `summary.seroincidence.by()`)

## Usage

```
## S3 method for class 'summary.seroincidence.by'
print(x, ...)
```

## Arguments

| | |
|---|---|
| x | A "summary.seroincidence.by" object (constructed by `summary.seroincidence.by()`) |
| ... | Additional arguments affecting the summary produced. |

## Value

A `list` containing the summary statistics for output of the function `est.incidence.by()`.

## Examples

```
library(tidyverse)

xs_data <- load_pop_data("https://osf.io/download//n6cp3/") %>%
  clean_pop_data()

curve <- load_curve_params("https://osf.io/download/rtw5k/") %>%
  filter(antigen_iso %in% c("HlyE_IgA", "HlyE_IgG")) %>%
  slice(1:100, .by = antigen_iso) # Reduce dataset for the purposes of this example

noise <- load_noise_params("https://osf.io/download//hqy4v/")


est2 <- est.incidence.by(
  strata = c("catchment"),
  pop_data = xs_data %>% filter(Country == "Pakistan"),
  curve_params = curve,
  noise_params = noise %>% filter(Country == "Pakistan"),
  antigen_isos = c("HlyE_IgG", "HlyE_IgA"),
  #num_cores = 8 # Allow for parallel processing to decrease run time
)

summary(est2) %>%
print()
```

---

# row_longitudinal_parameter

*extract a row from longitudinal parameter set*

---

## Description

take a random sample from longitudinal parameter set given age at infection, for a list of antibodies

## Usage

```
row_longitudinal_parameter(age, antigen_isos, nmc, npar, ...)
```

## Arguments

| | |
|---|---|
| age | age at infection |
| antigen_isos | antigen isotypes |
| nmc | mcmc sample to use |
| npar | number of parameters |
| ... | passed to simpar() |

### Value

an array of parameters: c(y0,b0,mu0,mu1,c1,alpha,shape)

---

`serocalculator` *Estimating Infection Rates from Serological Data*

---

### Description

This package translates antibody levels measured in a (cross-sectional) population sample into an estimate of the frequency with which seroconversions (infections) occur in the sampled population.

The API for this package includes the following functions:

| Function Name | Purpose |
|---|---|
| `load_pop_data()` | loading cross-sectional antibody survey data |
| `clean_pop_data()` | cleaning antibody data |
| `check_pop_data()` | checking antibody data |
| `summary.pop_data()` | numerical summaries of antibody data |
| `autoplot.pop_data()` | graphs of antibody data distributions |
| `load_curve_params()` | loading antibody decay curve models |
| `autoplot.curve_params()` | graphing antibody decay curves |
| `log_likelihood()` | computing log-likelihoods |
| `graph.loglik()` | graphing log-likelihood functions |
| `autoplot.seroincidence()` | graphing log-likelihood functions |
| `autoplot.seroincidence.by()` | graphing log-likelihood functions |
| `est.incidence()` | estimating incidence rates |
| `est.incidence.by()` | estimating incidence rates by strata |
| `summary.seroincidence.by()` | summarizing stratified incidence rate estimates |
| `autoplot.summary.seroincidence.by()` | graphing incidence rate estimates |
| `sim.cs()` | simulating cross-sectional population antibody data using longitudinal seroresp |

### Details

_PACKAGE

### Author(s)


- Peter Teunis <p.teunis@emory.edu>
- Doug Ezra Morrison <demorrison@ucdavis.edu>
- Kristen Aiemjoy <kaiemjoy@ucdavis.edu>
- Kristina Lai <kwlai@ucdavis.edu>

## References

### Methods for estimating seroincidence


- Teunis, P. F. M., and J. C. H. van Eijkeren. "Estimation of seroconversion rates for infectious diseases: Effects of age and noise." Statistics in Medicine 39, no. 21 (2020): 2799-2814.
- Teunis, P. F. M., J. C. H. van Eijkeren, W. F. de Graaf, A. Bonačić Marinović, and M. E. E. Kretzschmar. "Linking the seroresponse to infection to within-host heterogeneity in antibody production." Epidemics 16 (2016): 33-39.


### Applications


- Aiemjoy, Kristen, Jessica C. Seidman, Senjuti Saha, Sira Jam Munira, Mohammad Saiful Islam Sajib, Syed Muktadir Al Sium, Anik Sarkar et al. "Estimating typhoid incidence from community-based serosurveys: a multicohort study." The Lancet Microbe 3, no. 8 (2022): e578-e587.
- Aiemjoy, Kristen, John Rumunu, Juma John Hassen, Kirsten E. Wiens, Denise Garrett, Polina Kamenskaya, Jason B. Harris et al. "Seroincidence of enteric fever, Juba, South Sudan." Emerging infectious diseases 28, no. 11 (2022): 2316.
- Monge, S., Teunis, P. F., Friesema, I., Franz, E., Ang, W., van Pelt, W., Mughini-Gras, L. "Immune response-eliciting exposure to Campylobacter vastly exceeds the incidence of clinically overt campylobacteriosis but is associated with similar risk factors: A nationwide serosurvey in the Netherlands" Journal of Infection, 2018, 1–7, doi:10.1016/j.jinf.2018.04.016
- Kretzschmar, M., Teunis, P. F., Pebody, R. G. "Incidence and reproduction numbers of pertussis: estimates from serological and social contact data in five European countries" PLoS Medicine 7, no. 6 (June 1, 2010):e1000291. doi:10.1371/journal.pmed.1000291.
- Simonsen, J., Strid, M. A., Molbak, K., Krogfelt, K. A., Linneberg, A., Teunis, P. "Sero-epidemiology as a tool to study the incidence of Salmonella infections in humans" Epidemiology and Infection 136, no. 7 (July 1, 2008): 895–902. doi:10.1017/S0950268807009314
- Simonsen, J., Teunis, P. F., van Pelt, W., van Duynhoven, Y., Krogfelt, K. A., Sadkowska-Todys, M., Molbak K. "Usefulness of seroconversion rates for comparing infection pressures between countries" Epidemiology and Infection, April 12, 2010, 1–8. doi:10.1017/S0950268810000750.
- Falkenhorst, G., Simonsen, J., Ceper, T. H., van Pelt, W., de Valk, H., Sadkowska-Todys, M., Zota, L., Kuusi, M., Jernberg, C., Rota, M. C., van Duynhoven, Y. T., Teunis, P. F., Krogfelt, K. A., Molbak, K. "Serological cross-sectional studies on salmonella incidence in eight European countries: no correlation with incidence of reported cases" BMC Public Health 12, no. 1 (July 15, 2012): 523–23. doi:10.1186/1471-2458-12-523.
- Teunis, P. F., Falkenhorst, G., Ang, C. W., Strid, M. A., De Valk, H., Sadkowska-Todys, M., Zota, L., Kuusi, M., Rota, M. C., Simonsen, J. B., Molbak, K., Van Duynhoven, Y. T., van Pelt, W. "Campylobacter seroconversion rates in selected countries in the European Union" Epidemiology and Infection 141, no. 10 (2013): 2051–57. doi:10.1017/S0950268812002774.
- de Melker, H. E., Versteegh, F. G., Schellekens, J. F., Teunis, P. F., Kretzschmar, M. "The incidence of Bordetella pertussis infections estimated in the population from a combination of serological surveys" The Journal of Infection 53, no. 2 (August 1, 2006): 106–13. doi:10.1016/j.jinf.2005.10.020


### Quantification of seroresponse

- de Graaf, W. F., Kretzschmar, M. E., Teunis, P. F., Diekmann, O. "A two-phase within-host model for immune response and its application to serological profiles of pertussis" Epidemics 9 (2014):1–7. doi:10.1016/j.epidem.2014.08.002.
- Berbers, G. A., van de Wetering, M. S., van Gageldonk, P. G., Schellekens, J. F., Versteegh, F. G., Teunis, P.F. "A novel method for evaluating natural and vaccine induced serological responses to Bordetella pertussis antigens" Vaccine 31, no. 36 (August 12, 2013): 3732–38. doi:10.1016/j.vaccine.2013.05.073.
- Versteegh, F. G., Mertens, P. L., de Melker, H. E., Roord, J. J., Schellekens, J. F., Teunis, P. F. "Age-specific long-term course of IgG antibodies to pertussis toxin after symptomatic infection with Bordetella pertussis" Epidemiology and Infection 133, no. 4 (August 1, 2005): 737–48.
- Teunis, P. F., van der Heijden, O. G., de Melker, H. E., Schellekens, J. F., Versteegh, F. G., Kretzschmar, M. E. "Kinetics of the IgG antibody response to pertussis toxin after infection with B. pertussis" Epidemiology and Infection 129, no. 3 (December 10, 2002):479. doi:10.1017/S0950268802007896.


---

`sim.cs` *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.cs(
  lambda = 0.1,
  n.smpl = 100,
  age.rng = c(0, 20),
  age.fx = NA,
  antigen_isos,
  n.mc = 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.smpl` | number of samples to simulate |
| `age.rng` | age range of sampled individuals, in years |

| | |
|---|---|
| age.fx | 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.mc | how many MCMC samples to use:<br>• when n.mc is in 1:4000 a fixed posterior sample is used<br>• when n.mc = 0, a random sample is chosen |
| renew.params | whether to generate a new parameter set for each infection<br>• renew.params = TRUE generates a new parameter set for each infection<br>• 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:<br>• antigen_iso: a character() vector indicating antigen-isotype combinations<br>• iter: an integer() vector indicating MCMC sampling iterations<br>• y0: baseline antibody level at $t = 0$ ($y(t = 0)$)<br>• y1: antibody peak level (ELISA units)<br>• t1: duration of infection<br>• alpha: antibody decay rate (1/days for the current longitudinal parameter sets)<br>• r: shape factor of antibody decay |
| noise_limits | biologic noise distribution parameters |
| format | a character() variable, containing either:<br>• "long" (one measurement per row) or<br>• "wide" (one serum sample per row) |
| verbose | logical: if TRUE, print verbose log information to console |
| ... | additional arguments passed to simcs.tinf() |

## 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 <- load_curve_params("https://osf.io/download/rtw5k")

# 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.cs(
  curve_params = dmcmc,
  lambda = lambda,
  n.smpl = nrep,
  age.rng = lifespan,
  antigen_isos = antibodies,
  n.mc = 0,
  renew.params = TRUE,
  add.noise = TRUE,
  noise_limits = dlims,
  format = "long"
)
```

---

`sim.cs.multi` *Simulate multiple data sets*

---

## Description

Simulate multiple data sets

## Usage

```
sim.cs.multi(
  nclus = 10,
  lambdas = c(0.05, 0.1, 0.15, 0.2, 0.3),
  num_cores = max(1, parallel::detectCores() - 1),
  rng_seed = 1234,
  renew.params = TRUE,
  add.noise = TRUE,
  verbose = FALSE,
  ...
)
```

## Arguments

| | |
|---|---|
| nclus | number of clusters |
| 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() |
| renew.params | whether to generate a new parameter set for each infection<br>• renew.params = TRUE generates a new parameter set for each infection<br>• 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 |
| verbose | whether to report verbose information |
| ... | Arguments passed on to sim.cs<br>lambda a numeric() scalar indicating the incidence rate (in events per person-years)<br>n.smpl number of samples to simulate<br>age.rng age range of sampled individuals, in years<br>age.fx specify the curve parameters to use by age (does nothing at present?)<br>antigen_isos Character vector with one or more antibody names. Values must match curve_params.<br>n.mc how many MCMC samples to use:<br>• when n.mc is in 1:4000 a fixed posterior sample is used<br>• when n.mc = 0, a random sample is chosen<br>noise_limits biologic noise distribution parameters<br>format a character() variable, containing either:<br>• "long" (one measurement per row) or<br>• "wide" (one serum sample per row)<br>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:<br>• antigen_iso: a character() vector indicating antigen-isotype combinations<br>• iter: an integer() vector indicating MCMC sampling iterations<br>• y0: baseline antibody level at $t = 0$ ($y(t = 0)$)<br>• y1: antibody peak level (ELISA units)<br>• t1: duration of infection<br>• alpha: antibody decay rate (1/days for the current longitudinal parameter sets)<br>• r: shape factor of antibody decay |

## Value

a tibble::tibble()

`simcs.tinf` *collect cross-sectional data*

## Description

output: (age, y(t) set)

## Usage

```
simcs.tinf(
  lambda,
  n.smpl,
  age.rng,
  age.fx = NA,
  antigen_isos,
  n.mc = 0,
  renew.params = FALSE,
  ...
)
```

## Arguments

| | |
|---|---|
| `lambda` | seroconversion rate (in events/person-day) |
| `n.smpl` | number of samples n.smpl (= nr of simulated records) |
| `age.rng` | age range to use for simulating data, in days |
| `age.fx` | age.fx for parameter sample (age.fx = NA for age at infection) |
| `antigen_isos` | Character vector with one or more antibody names. Values must match `curve_params`. |
| `n.mc` | • when `n.mc` is in 1:4000 a fixed posterior sample is used<br>• when n.mc = 0 a random sample is chosen |
| `renew.params` | • `renew.params` = `TRUE` generates a new parameter set for each infection<br>• `renew.params` = `FALSE` keeps the one selected at birth, but updates baseline y0 |
| `...` | arguments passed to `simresp.tinf()` |

## Value

an `array()`

`simresp.tinf` *simulate antibody kinetics of y over a time interval*

## Description

simulate antibody kinetics of y over a time interval

## Usage

```
simresp.tinf(
  lambda,
  t.end,
  age.fx,
  antigen_isos,
  n.mc = 0,
  renew.params,
  predpar,
  ...
)
```

## Arguments

| | |
|---|---|
| `lambda` | seroconversion rate (1/days), |
| `t.end` | end of time interval (beginning is time 0) in days(?) |
| `age.fx` | parameter estimates for fixed age (age.fx in years) or not. when age.fx = NA then age at infection is used. |
| `antigen_isos` | antigen isotypes |
| `n.mc` | a posterior sample may be selected (1:4000), or not when n.mc = 0 a posterior sample is chosen at random. |
| `renew.params` | At infection, a new parameter sample may be generated (when `renew.params = TRUE`). Otherwise (when `renew.params = FALSE`), a sample is generated at birth and kept, but baseline y0 are carried over from prior infections. |
| `predpar` | an `array()` with dimensions named:<br>• `antigen_iso`<br>• `parameter`<br>• `obs` |
| `...` | Arguments passed on to `row_longitudinal_parameter`, `ab`, `mk_baseline`<br>`age` age at infection<br>`nmc` mcmc sample to use<br>`npar` number of parameters<br>`t` age at infection?<br>`par` parameters<br>`kab` index for which row of antibody baseline limits to read from `blims`<br>`n` number of observations<br>`blims` range of possible baseline antibody levels |

### Value

This function returns a `list()` with:

- t = times (in days, birth at day 0),
- b = bacteria level, for each antibody signal (not used; probably meaningless),
- y = antibody level, for each antibody signal
- smp = whether an infection involves a big jump or a small jump
- t.inf = times when infections have occurred.

---

## strata *Extract strata from an object*

---

### Description

Generic method for extracting strata from objects. See `strata.seroincidence.by()`

### Usage

```
strata(x)
```

### Arguments

| | |
|---|---|
| x | an object |

### Value

the strata of x

---

## strata.seroincidence.by *Extract the* Strata *attribute from an object, if present*

---

### Description

Extract the Strata attribute from an object, if present

### Usage

```
## S3 method for class 'seroincidence.by'
strata(x)
```

### Arguments

| | |
|---|---|
| x | any R object |

### Value

- a tibble::tibble() with strata in rows, or
- NULL if x does not have a "strata" attribute

---

summary.pop_data *Summarize a cross-sectional antibody survey data set*

---

### Description

This function is a summary() method for pop_data objects

### Usage

```
## S3 method for class 'pop_data'
summary(object, ...)
```

### Arguments

| | |
|---|---|
| object | a pop_data object |
| ... | unused |

### Value

a list containing two summary tables: one of age and one of value, stratified by antigen_iso

### Examples

```
library(dplyr)

xs_data <- load_pop_data("https://osf.io/download//n6cp3/") %>%
  clean_pop_data()

summary(xs_data)
```

`summary.seroincidence` *Summarizing fitted seroincidence models*

## Description

This function is a `summary()` method for `seroincidence` objects.

## Usage

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

## Arguments

| | |
|---|---|
| `object` | a `list()`, outputted by `stats::nlm()` or `est.incidence()` |
| `coverage` | desired confidence interval coverage probability |
| `...` | 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
- `CI.lwr`: lower limit of confidence interval for incidence rate
- `CI.upr`: upper limit of confidence interval for incidence rate
- `coverage`: coverage probability
- `log.lik`: log-likelihood of the data used in the call to `est.incidence()`, 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.incidence()` (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 <- load_pop_data("https://osf.io/download//n6cp3/") %>%
  clean_pop_data()

curve <- load_curve_params("https://osf.io/download/rtw5k/") %>%
  filter(antigen_iso %in% c("HlyE_IgA", "HlyE_IgG")) %>%
  slice(1:100, .by = antigen_iso) # Reduce dataset for the purposes of this example

noise <- load_noise_params("https://osf.io/download//hqy4v/")

est1 <- est.incidence(
  pop_data = xs_data %>% filter(Country == "Pakistan"),
  curve_param = curve,
  noise_param = noise %>% filter(Country == "Pakistan"),
  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.incidence.by().

## Usage

```
## S3 method for class 'seroincidence.by'
summary(
  object,
  confidence_level = 0.95,
  showDeviance = TRUE,
  showConvergence = TRUE,
  ...
)
```

## Arguments

| | |
|---|---|
| object | A dataframe containing output of function est.incidence.by(). |
| confidence_level | desired confidence interval coverage probability |
| showDeviance | Logical flag (FALSE/TRUE) for reporting deviance (-2*log(likelihood) at estimated seroincidence. Default = TRUE. |

showConvergence
: Logical flag (FALSE/TRUE) for reporting convergence (see help for optim() for details). Default = FALSE.

...
: 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 showDeviance = TRUE) Negative log likelihood (NLL) at estimated (maximum likelihood) lambda)
  - nlm.convergence.code (included if showConvergence = 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.incidence.by()
- Strata Character with names of strata used in est.incidence.by()

## Examples

```
library(dplyr)

xs_data <- load_pop_data("https://osf.io/download//n6cp3/")%>%
  clean_pop_data()

curve <- load_curve_params("https://osf.io/download/rtw5k/") %>%
  filter(antigen_iso %in% c("HlyE_IgA", "HlyE_IgG")) %>%
  slice(1:100, .by = antigen_iso) # Reduce dataset for the purposes of this example

noise <- load_noise_params("https://osf.io/download//hqy4v/")
# estimate seroincidence#'
est2 <- est.incidence.by(
  strata = c("catchment"),
  pop_data = xs_data %>% filter(Country == "Pakistan"),
  curve_params = curve,
  noise_params = noise %>% filter(Country == "Pakistan"),
  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)
```

## warn.missing.strata *Warn about missing stratifying variables in a dataset*

### Description

Warn about missing stratifying variables in a dataset

### Usage

```
warn.missing.strata(data, strata, dataname)
```

### Arguments

| | |
|---|---|
| data | the dataset that should contain the strata |
| strata | a data.frame() showing the strata levels that are expected to be in the dataset |
| dataname | the name of the dataset, for use in warning messages if some strata are missing. |

### Value

a character() vector of the subset of stratifying variables that are present in pop_data

## [.seroincidence.by *Extract or replace parts of a* seroincidence.by *object*

### Description

Extract or replace parts of a seroincidence.by object

### Usage

```
## S3 method for class 'seroincidence.by'
x[i, ...]
```

### Arguments

| | |
|---|---|
| x | the object to subset/replace elements of |
| i | the indices to subset/replace |
| ... | passed to [.list |

### Value

the subset specified

# Index