# Seroincidence package tutorial

*European Centre for Disease Prevention and Control (ECDC)*

*2018-06-05*

## Contents

# 1. Introduction

Antibody levels measured in a (cross–sectional) population sample can be translated into an estimate of the frequency with which seroconversions (infections) occur in the sampled population. In simple terms: the presence of many high antibody concentrations indicates that many subjects likely experienced infection recently, while mostly low concentrations indicate a low frequency of infections in the sampled population. More information about ELISA methods can be found on ECDC web site.

In order to thus interpret the measured cross-sectional antibody concentrations, characteristics of the time course of the serum antibody response must be known. For any antibody detected in the cross–sectional sample, a quantitative description of its dynamics following infection (seroconversion to an elevated concentration, followed by a gradual decrease) must be available. In published studies, this information on the time course of the serum antibody response has been obtained from follow–up data in subjects who had a symptomatic episode following infection. The onset of symptoms thus provided a proxy for the time that infection occurred. Care must be taken that longitudinal and cross–sectional measurements of antibody concentrations are calibrated on the same scale (or, preferably, expressed in identical units).

The time course of the serum antibody response to infection is assumed generic: anyone infected by the same pathogen is expected to produce a response similar to those seen in the symptomatic cohort used for the follow–up study. Thus, the strategy is to set up a longitudinal model from a follow–up study once, and use the thus obtained information to analyze many different cross–sectional data sets.

Note that it must be assumed that such variation in the time course of antibody concentrations among individuals as observed in the longitudinal cohort is the same as in the population where the cross-sectional sample was taken, where this variation cannot be observed.

Analysis of longitudinal data and estimation of the required parameters is separated from the estimation of seroincidences, because (1) there are few longitudinal data sets available and they are difficult (and expensive) to obtain, (2) such longitudinal models may be considered *generic*, presumably describing a physiological phenomenon that is similiar in any infected individual, and (3) setting up and running the longitudinal model requires additional skills, not necessary for use of the sero–incidence calculator script

# 2. The seroincidence estimator

Package **seroincidence** was designed to calculate incidence of seroconversion, by using the longitudinal seroresponse characteristics. The distribution of serum antibody concentrations in a cross–sectional population sample is calculated, as a function of the (longitudinal) seroresponse and the frequency of seroconversion (or seroincidence). Given the seroresponse, this (marginal) distribution of antibody concentrations can be fitted to the cross-sectional data, by adjusting the seroconversion frequency, thus providing a means to estimate the seroconversion frequency.

## 2.1 Time course of serum antibodies

The seroresponse, the time course of serum antibodies following seroconversion (infection) is described by means of a set of parameters describing it. These parameters may include:

a) baseline antibody concentration,
b) time to reach peak antibody concentration,
c) magnitude of the peak concentration, and
d) antibody decay parameters describing the time course (shape and rate) of antibody decay.

A model description that is used for the serocalculator is described in (Simonsen et al. 2009; Teunis et al. 2012). An alternative model, simplified but with improved interpretation of the underlying biological mechanisms, is given in (de Graaf et al. 2014) and further improved in (Teunis et al. 2016).

The current model assumes that upon exposure to an initial inoculum, pathogen concentrations increase exponentially. Presence of pathogens (antigen presented to the immune system) signals an increase in antibody production, resulting in exponentially increasing serum antibody concentrations. Increased activity of the immune system, as signalled by circulating antibodies, causes inactivation or die–off of pathogens, with a rate proportional to the serum antibody concentration. Ultimately, when all pathogens are removed, infection is over and the promotion of antibody production stops. Then the decay phase starts, and serum antibody concentrations decrease. Antibody decay in the serum compartment may start fast and slow down later, resulting in non–exponential decay. In (Teunis et al. 2016) this is modelled using a shape parameter, allowing gradual adjustment between exponential (log–linear) and strongly non–exponential decay.

## 2.2 Heterogenity

The five parameters describing the seroresponse (in round brackets names of corresponding parameters used in the package, see section 3.2.2 for details): baseline antibody concentration (`y0` and `yb`), time to peak (`t1`), peak antibody concentration (`y1`), decay rate (`alpha`) and shape factor of the decay curve (`r`) are not fixed numbers. In a population of subjects, individuals each have their own seroresponse, different from those of other subjects. Therefore, these parameters are provided as a (joint) distribution, describing their variation among individuals in the sampled population. Of course, such a distribution depends on the pathogen, the detected antibody, and even the detection assay.

Estimates for these seroresponse parameters may be obtained by fitting the model to longitudinal data. Typically, such data are collected in patients with confirmed infection, where the time of infection is detected by appearance of symptoms. Ideally, blood samples are taken at several times post infection, up to a period long enough to reliably estimate the shape of the serum antibody decay curve.

A Bayesian hierarchical framework allows for individual variation in seroresponses, so that one obtains the (posterior) joint distribution of the parameters as a (Markov chain) Monte Carlo sample, of all relevant parameters (Teunis et al. 2016).

Note that, due to the limited number of iterations in the Monte Carlo sample, the generated incidence estimates will show minor variations: if the same calculation is run twice, the outcomes may differ slightly. If this variation should be smaller, the size of the Monte Carlo sample may be increased. This however will slow down the calculations.

### 2.3 Incidence estimation

The longitudinal seroresponses are considered generic: they represent the time course of (specific) human serum antibodies against a (specific) pathogen. Therefore, with any cross–sectional sample of the same antibodies in a human population, an incidence estimate can be calculated.

Given the longitudinal response characteristics, **seroincidence** R package provide a rapid and computationally simple method for calculating seroconversion rates, as published in (Teunis et al. 2012).

The methods on which the seroincidence calculator is based were developed with publicly funded research. All procedures have been published and are open to the general public, including the core script on which the seroincidence calculator is based. To make the functionalities provided as broadly available as possible, R was chosen as the computing platform, because it is free and open source, and runs on any modern computer operating system.

### 2.4 Model variants

The package contains longitudinal models for seroincidence calculations. Four model variants are possible, depending on whether seroconversion is instantaneous ($t_1 = 0$) or not ($t_1 > 0$), and whether decay is exponential ($r = 1$) or proceeds as a power function ($r > 1$). Power function decay allows for rapid inital decay followed by a sustained period of very slow decay (Teunis et al. 2016). Available model variants are described in Table 1.

Table 1: Model variants

| Model | Time to Seroconversion ($t_1$) | Decay parameter ($r$) |
|---|---|---|
| 1 | 0 | 1 |
| 2 | 0 | $> 1$ |
| 3 | $> 0$ | 1 |
| 4 | $> 0$ | $> 1$ |

| | | | Model | | | |
|---|---|---|---|---|---|---|
| Pathogen | Antibody | Units | 1 | 2 | 3 | 4 |
| *Campylobacter jejuni* | IgG | ELISA (Delft) | * | | * | * |
| | IgM | ELISA (Delft) | * | | * | * |
| | IgA | ELISA (Delft) | * | | * | * |
| | IgG | ELISA (SSI) | * | * | | * |
| | IgM | ELISA (SSI) | * | * | | * |
| | IgA | ELISA (SSI) | * | * | | * |
| *Salmonella* | IgG | mixed ELISA (SSI) | * | * | | * |
| | IgM | mixed ELISA (SSI) | * | * | | * |
| | IgA | mixed ELISA (SSI) | * | * | | * |
| *Bordetella pertussis* | IgG-PT | IU | * | * | * | * |

## 3. How to use *seroincidence* package

This section provides step-by-step directions on the usage of the *seroincidence* package.

### 3.1. Loading the package

Functionality of the *seroincidence* package is made available to the end user only once the library is loaded into the current workspace. Assuming the package is installed already (covered in installation.pdf) loading is achieved by running the following command in the R console (bear in mind that the text after character `#` is only a comment):

```
# Load package "seroincidence"
library(seroincidence)
```

At this moment all functions required to run the seroincidence calculation are made available. It can be checked by running:

```
# List all objects (functions and data) exposed by package "seroincidence".
ls("package:seroincidence")
```

Here is the resulting output:

```
##  [1] "campylobacterDelftParams1"  "campylobacterDelftParams3"
##  [3] "campylobacterDelftParams4"  "campylobacterSSIParams1"
##  [5] "campylobacterSSIParams2"    "campylobacterSSIParams4"
##  [7] "campylobacterSimHighData"   "campylobacterSimLowData"
##  [9] "campylobacterSimMediumData" "estimateSeroincidence"
## [11] "getAdditionalData"          "pertussisIgGPTParams1"
## [13] "pertussisIgGPTParams2"      "pertussisIgGPTParams3"
## [15] "pertussisIgGPTParams4"      "pertussisSimHighData"
## [17] "pertussisSimLowData"        "pertussisSimMediumData"
## [19] "salmonellaSSIParams1"       "salmonellaSSIParams2"
## [21] "salmonellaSSIParams4"
```

We briefly describe the most important objects and functions:

a) `[DISEASE]Params[MODEL_TYPE]`: Monte Carlo sample of longitudinal response parameters per antibody for various diseases: Campylobacter, Pertussis, Salmonella. Object class: `list`.
b) `[DISEASE]Sim[Low|Medium|High]Data`: Example simulated antibody levels data measured in a cross-sectional population for three values of lambda (incidence): 0.036 ("Low"), 0.21 ("Medium") and 1.15 ("High") (1/yr). Object class: `data.frame`.
c) `getAdditionalData`: Utility function for downloading additional longitudinal response parameters from an online repository. Files available for download are `coxiellaIFAParams4.zip` and `yersiniaSSIParams4.zip`. Object class: `function`.
d) `estimateSeroincidence`: Main function of the package. Estimates seroincidence based on supplied cross-section antibody levels data and longitudinal response parameters. Object class: `function`.

### 3.2. Specifying input data

There are two sets of inputs to the serology calculator that must always be specified:

a) Simulated antibody levels measured in cross-sectional population sample (see `campylobacterSimLowData`, `campylobacterSimMediumData`, etc. above).
b) Longitudinal response characteristic as set of parameters `(y1, alpha, yb, r, y0, mu1, t1)` (see `campylobacterDelftParams1`, `campylobacterSSIParams2`, etc. above).

We start with loading antibody levels measured in cross-sectional population sample.

#### 3.2.1. Specifying cross-sectional antibody levels data

The user has a few options of providing this data to the calculator:

**i) Using supplied sample data**

Data set `campylobacterSimMediumData` provides an example of a cross-sectional antibody levels data for Campylobacter. It is loaded into the workspace as soon as the package is loaded, therefore it can be referenced right away. Here is a small exempt from this data set:

```
# Show three first rows of "campylobacterSimMediumData" data.frame.
head(campylobacterSimMediumData, 3)
```

```
##         Age       IgG        IgM          IgA
## 1  4530.743 0.2242790 0.05442598 1.138591e-02
## 2  5173.256 0.8498734 0.10667042 2.093677e-06
## 3 24187.164 0.1849355 1.24958865 1.652755e-01
```

Let us create the serology data `serologyData` based on this data set:

```
# Assign data.frame "campylobacterDelftData" to object named "serologyData".
serologyData <- with(campylobacterSimMediumData,
                     data.frame(
                       Age = Age,
                       IgG = IgG,
                       IgM = IgM,
                       IgA = IgA,
                       AgeCat = cut(Age, 15, labels = FALSE)))

# Print a few first observations.
head(serologyData)
```

```
##         Age       IgG          IgM          IgA AgeCat
## 1  4530.743 0.2242790 0.0544259797 1.138591e-02      3
## 2  5173.256 0.8498734 0.1066704172 2.093677e-06      3
## 3 24187.164 0.1849355 1.2495886495 1.652755e-01     14
```

```
## 4 21427.836 1.1743485 0.0001590331 3.122555e-02     12
## 5 12560.687 1.0656224 0.0013906805 8.723256e-04      7
## 6 17083.219 0.1115373 0.4976598547 2.321977e-01     10
```

You can view the data also in a tabular form:

```
View(serologyData)
```

We shall pass this object later to the calculator. Object `serologyData` must contain at least one column named `IgG`, `IgM`, or `IgA`. Additionally, column `Age` is used in the calculations if it is present. Otherwise, it is created on-the-fly and initialized to `NA` (not available). Column `Age`, if supplied, is assumed to represent age in days.

Please, note that the cross-sectional data set can include extra columns allowing performing the seroincidence calculations stratified per factors available in those data. In the example above column `AgeCat` can be used as a *stratum* variable. This particular variable is created by mapping the age in days (column `Age`) into 15 categories, category 1 grouping the youngest subjects and category 15 grouping the oldest subjects.

**ii) Loading from external files**

The user can load her/his own data into an R session. Details on loading specific file types can be found in online R manual on R Data Import/Export.

It is important that the input data is in a tabular form with each column containing levels measured for a single antibody type. Column name should indicate name of the measured antibody. For instance, a comma separated file with the following content:

```
IgG, IgM, IgA
5.337300, 0.4414653, 0.3002395
3.534118, 0.3888226, 0.3543486
2.144549, 0.3320178, 0.3884205
2.957854, 0.4967764, 0.3472340
```

is a valid cross-sectional data set. Suppose this file is named `c:\cross-sectional-data.csv`. Then it can be loaded into R like this:

```
# Read content of file "C:\cross-sectional-data.csv" into object named
# "serologyData".
serologyData <- read.csv(file = "C:\\cross-sectional-data.csv")
```

### 3.2.2. Specifying longitudinal response parameters

The longitudinal response parameters set consists of the following items:

- `y1`: antibody peak level (ELISA units)
- `alpha`: antibody decay rate (1/days for the current longitudinal parameter sets)
- `yb`: baseline antibody level at $t$ approaching infinity ($y(t- > inf)$)

- `r`: shape factor of antibody decay
- `y0`: baseline antibody level at $t = 0$ ($y(t = 0)$)
- `mu1`: initial antibody growth rate
- `t1`: duration of infection

Please, refer to vignette methodology.pdf for more details on the underlying methodology.

End user has three options for loading the response parameters into R session:

**i) Using supplied data**

There are a few longitudinal response parameter data sets provided by the package including Monte-Carlo sample of longitudinal response parameters, like `campylobacterDelftParams4`. This particular object is a list containing three data.frame objects named `IgG`, `IgM` and `IgA`. Let's have a pick into each of these two data sets separately (notice character `$` indicating a sub-object of the parent object):

```
# Show first rows of data.frame "IgG" in list "campylobacterDelftParams4".
head(campylobacterDelftParams4$IgG)
```

```
##          y1        alpha yb        r          y0       mu1       t1
## 1 27.861427 0.0051607511  0 1.019014 0.07791154 0.7296875 8.057456
## 2  3.306758 0.0011913102  0 1.099056 0.05077868 0.8435221 4.950963
## 3 18.791581 0.0006869688  0 1.010531 0.09212040 0.6116469 8.694670
## 4  5.866064 0.0006929407  0 1.019739 0.07694126 0.5232645 8.282421
## 5 14.364930 0.0035222413  0 1.092096 0.05110497 0.9445758 5.969519
## 6  5.790477 0.0003697673  0 1.010602 0.09127834 0.7696968 5.391807
```

```
# Show first rows of data.frame "IgM" in list "campylobacterDelftParams4".
head(campylobacterDelftParams4$IgM)
```

```
##          y1        alpha yb        r          y0       mu1       t1
## 1 3.0218848 0.0068170529  0 1.073372 0.09900034 0.3980767 8.587573
## 2 1.4605254 0.0004345318  0 1.042603 0.07462244 0.4741715 6.272224
## 3 2.1067968 0.0043837064  0 1.068606 0.10507351 0.4748134 6.314615
## 4 1.1187181 0.0019154929  0 1.071340 0.09971466 0.3124635 7.737306
## 5 0.4222472 0.0005651563  0 1.000181 0.12935880 0.5040532 2.346977
## 6 4.5818356 0.0158849155  0 1.071558 0.10262796 0.6777377 5.605037
```

```
# Show first rows of data.frame "IgA" in list "campylobacterDelftParams4".
head(campylobacterDelftParams4$IgA)
```

```
##          y1        alpha yb        r          y0       mu1        t1
## 1 2.7471594 0.0235920904  0 2.773443 0.06452524 0.2737697 13.702269
## 2 0.3361973 0.0001511561  0 1.006182 0.13241137 0.6774068  1.375517
## 3 1.3402570 0.0072306458  0 1.640734 0.12395023 0.3883815  6.129892
## 4 7.8335091 0.0028906244  0 2.768401 0.06512216 0.3482461 13.754356
## 5 0.3069594 0.0029306951  0 1.050954 0.07870162 0.4218624  3.226293
## 6 1.6315826 0.0044291436  0 1.740029 0.12264517 0.5924527  4.368299
```

Let us assign object `campylobacterDelftParams4` to variable `responseParams`:

```
responseParams <- campylobacterDelftParams4
```

#### ii) Loading from local external files

Alternatively, this data can be loaded from an external file. Bear in mind, that most likely your data is organized into three files, separately for antibody `IgG`, `IgM` and `IgA`. Assuming that the data is saved into csv files named `IgG.csv`, `IgM.csv` and `IgA.csv` the end user would run the following code in order to create a valid input to the calculator:

```
IgGData <- read.csv(file = "IgG.csv")
IgMData <- read.csv(file = "IgM.csv")
IgAData <- read.csv(file = "IgA.csv")

# Create a list named "responseData" containing objects named "IgG", "IgM" and
# "IgA".
responseParams <- list(IgG = IgGData, IgM = IgMData, IgA = IgAData)
```

Note that object `responseParams` is a list with three dataframes named `IgG`, `IgM` and `IgA` as required.

#### iii) Loading from an external repository

Finally, response parameters can be obtained from an online repository set up for this package. The repository is hosted on ECDC servers. At the time of publishing this package two files are available:

a) `coxiellaIFAParams4.zip`: longitudinal response parameters per antibody for Coxiella.
b) `yersiniaSSIParams4.zip`: longitudinal response parameters per antibody for Yersinia.

Simply use the supplied utility function `getAdditionalData` to download this data:

```
responseParams <- getAdditionalData("coxiellaIFAParams4.zip")
```

Internet connection is needed for this function to work.

### 3.3. Estimating seroincidence

Main calculation function provided by package *seroincidence* is called `estimateSeroincidence`. It takes several arguments:

- `data`: A data frame with the cross-sectional data (see variable `serologyData` above). This may be the whole data set loaded from file, but can also be a subset, e.g. `data = serologyData[1, ]` to use only the first row. In this tutorial this is the data loaded into variable `serologyData`. It is a required input.

- `antibodies`: A list of antibodies to be used. This can be a triplet (`antibodies = c("IgG", "IgM", "IgA")`) or any single antibody (`antibodies = c("IgG")`) or a combination of any two antibodies. It is a required input.
- `strata`: A list of categories to stratify data. The column `AgeCat` in the example data set can be used as an example. If `strata = ""` (default), then the whole data set is treated as a single stratum; if each sample is assigned a unique identifier (e.g. `strata = "AgeCat"`) there are as many strata as there are samples and an incidence estimate is calculated for each sample. If not specified, then `strata` is initialized to `""`.
- `params`: A list with the longitudinal parameters for all antibodies specified. In this tutorial this is the data loaded into variable `responseParams`. It is a required input.
- `censorLimits`: A list of cutoff levels, one for each antibody used. These are antibody levels below which observations should be treated as censored. It is a required input.
- `par0`: List of parameters for the (lognormal) distribution of antibody concentrations for true seronegatives (i.e. those who never seroconverted), by named antibody type (corresponding to `data`).
- `start` A starting value for `log(lambda)`. Value of -6 corresponds roughly to 1 day (`log(1/365.25)`), value of -4 corresponds roughly to 1 week (`log(7/365.25)`). Users are adivised to experiment with this value to confirm that convergence to the same estimate is obtained. Default is -6.
- `numCores` Number of processor cores to use for calculations when computing by `strata`. If set to more than 1 and R package `parallel` is available (installed by default), then the computations by stratum are executed in parallel. Default is 1, ie. no execution in parallel.

Here is an example of the *Campylobacter* seroincidence calculation for three antibodies measured, stratified per `AgeCat`, and with measurements below `0.1` removed:

```
# Prepare input data.
serologyData <- with(campylobacterSimMediumData,
                     data.frame(
                       Age = rep(NA, nrow(campylobacterSimMediumData)),
                       IgG = IgG,
                       IgM = IgM,
                       IgA = IgA,
                       AgeCat = cut(Age, 15, labels = FALSE)))

responseParams <- campylobacterDelftParams4

cutOffs <- list(IgG = 0.1, IgM = 0.1, IgA = 0.1)

# Baseline distributions: the distributions of antibody concentrations in
# subjects who have never seroconverted.
baseLine <- list(IgG = c(log(0.05), 1),
                 IgM = c(log(0.005), 1),
                 IgA = c(log(0.005), 1))

# Assign output of function "estimateSeroincidence" to object named
# "seroincidenceData". Use all available processor cores.
seroincidenceData <- estimateSeroincidence(
```

```
  data = serologyData,
  antibodies = c("IgG", "IgM", "IgA"),
  strata = "AgeCat",
  params = responseParams,
  censorLimits = cutOffs,
  par0 = baseLine,
  start = -4,
  numCores = parallel::detectCores())

# Show content of the output variable...
print(seroincidenceData)
# ...or simply type in the console: 'seroincidenceData' (without "'") and
# press ENTER.
```

The following text should be printed.

```
## Seroincidence object estimated given the following setup:
## a) Antibodies   : IgG, IgM, IgA
## b) Strata       : AgeCat
## c) Censor limits:  IgG = 0.1, IgM = 0.1, IgA = 0.1
##
## This object is a list containing the following items:
## Fits         - List of outputs of "optim" function per stratum.
## Antibodies   - Input parameter antibodies of function "estimateSeroincidence".
## Strata       - Input parameter strata of function "estimateSeroincidence".
## CensorLimits - Input parameter censorLimits of function "estimateSeroincidence".
##
## Call summary function to obtain output results.
```

The most important output is subobject `Fits` containing raw results on the output incidence. Translation of these raw results is provided by a custom function `summary` explained in the following section.

The cutoff argument is based on censoring of the observed serum antibody measurements (Strid et al. 2001). Cut-off levels must always be specified when calling function `estimateSeroincidence` (argument `censorLimits`). Value `0` should be set for antibody measurements where no censoring is needed, for instance

```
censorLimits <- list(IgG = 0, IgM = 0, IgA = 0)
```

Using different censoring levels will produce different results. Choosing higher cut-off causes the estimates of lambda to decrease. Results should be published together with the chosen cut-off values. The cut-off level of `0.1` set in the example above is a typical value, but users should set it to value reflecting the censoring level in the data.

### 3.4. Getting a summary of seroincidence

Output of the calculator should be passed in to function `summary` calculating output results:

```
summary(seroincidenceData)
```

```
## Seroincidence estimated given the following setup:
## a) Antibodies   : IgG, IgM, IgA
## b) Strata       : AgeCat
## c) Censor limits: IgG = 0.1, IgM = 0.1, IgA = 0.1
## d) Quantiles    : 0.025, 0.975
##
##  Seroincidence estimates:
##    Lambda.est Lambda.lwr Lambda.upr  Deviance Convergence Stratum
## 1  0.09721890 0.06397004 0.14774909 105.29815           0       1
## 2  0.05497303 0.03573489 0.08456817 181.90627           0       2
## 3  0.04456096 0.02845903 0.06977327 199.99357           0       3
## 4  0.02944038 0.01715302 0.05052964 141.32558           0       4
## 5  0.02569781 0.01436524 0.04597053 123.33213           0       5
## 6  0.02565297 0.01587275 0.04145941 187.41060           0       6
## 7  0.03458616 0.02012722 0.05943207 128.84486           0       7
## 8  0.02776873 0.02640621 0.02920154 133.57166          52       8
## 9  0.04016261 0.02133209 0.07561541  94.99389           0       9
## 10 0.01996477 0.01898409 0.02099612 104.30992          52      10
## 11 0.03356017 0.01917771 0.05872886 141.20168           0      11
## 12 0.05287648 0.03263923 0.08566140 181.68655           0      12
## 13 0.05081097 0.02946963 0.08760729 123.66670          52      13
## 14 0.04813104 0.02945312 0.07865371 186.92218           0      14
## 15 0.03421042 0.02210895 0.05293569 215.78684           0      15
```

The `summary` function returns a list of objects as well. It contains sub-objects `Antibodies`, `Strata`, `CensorLimits`, `Quantiles` and `Results`.

The most important sub-object of the output of the `summary` functions is `Results`, containing estimates of `Lambda` (annual incidence) together with their lower and upper bounds (`Lambda.lwr` and `Lambda.upr`, respectively). The default lower bound is 2.5% and the upper bound is 97.5% of the distribution of `Lambda`. These values can be changed (see Other examples).

Optionally, the summary function returns also `Deviance` (negative log likelihood at the estimated `Lambda`) and `Convergence` (indicator returned by function `stats::optim`; value of 0 indicates convergence).

It may be beneficial to assign output of this function to a variable for further analysis:

```
# Compute seroincidence summary and assign to object "seroincidenceSummary".
seroincidenceSummary <- summary(seroincidenceData)

# Show the results.
seroincidenceSummary$Results
```

```
##    Lambda.est Lambda.lwr Lambda.upr  Deviance Convergence Stratum
## 1  0.09721890 0.06397004 0.14774909 105.29815           0       1
## 2  0.05497303 0.03573489 0.08456817 181.90627           0       2
## 3  0.04456096 0.02845903 0.06977327 199.99357           0       3
## 4  0.02944038 0.01715302 0.05052964 141.32558           0       4
## 5  0.02569781 0.01436524 0.04597053 123.33213           0       5
## 6  0.02565297 0.01587275 0.04145941 187.41060           0       6
## 7  0.03458616 0.02012722 0.05943207 128.84486           0       7
## 8  0.02776873 0.02640621 0.02920154 133.57166          52       8
## 9  0.04016261 0.02133209 0.07561541  94.99389           0       9
## 10 0.01996477 0.01898409 0.02099612 104.30992          52      10
## 11 0.03356017 0.01917771 0.05872886 141.20168           0      11
## 12 0.05287648 0.03263923 0.08566140 181.68655           0      12
## 13 0.05081097 0.02946963 0.08760729 123.66670          52      13
## 14 0.04813104 0.02945312 0.07865371 186.92218           0      14
## 15 0.03421042 0.02210895 0.05293569 215.78684           0      15
```

As one may note it is a dataframe with six columns that can be later post-processed with custom calculations.

## 4. Other examples

The following examples show the standard procedure for estimating seroincidence.

```
# 1. Define cross-sectional data.
serologyData <- with(campylobacterSimMediumData,
                     data.frame(
                       IgG = IgG.ratio.new,
                       IgM = IgM.ratio.new,
                       IgA = IgA.ratio.new))

# 2. Define longitudinal response data
responseParams <- campylobacterDelftParams3

# 3. Define cut-offs
cutOffs <- list(IgG = 0.25, IgM = 0.25, IgA = 0.25)

# 4. Baseline distributions
baseLine <- list(IgG = c(log(0.05), 1),
                 IgM = c(log(0.005), 1),
                 IgA = c(log(0.005), 1))

# 4a. Calculate a single seroincidence rate for all serum samples...
seroincidenceData <- estimateSeroincidence(
  data = serologyData,
  antibodies = c("IgG", "IgM", "IgA"),
```

```
  strata = "",
  params = responseParams,
  censorLimits = cutOffs,
  par0 = baseLine)

# 4b. ...or calculate a single seroincidence rate for a single serum sample
# (triplet of titres)...
seroincidenceData <- estimateSeroincidence(
  data = serologyData[1, ],
  antibodies = c("IgG", "IgM", "IgA"),
  strata = "",
  params = responseParams,
  censorLimits = cutOffs,
  par0 = baseLine)

# 4c. ...or calculate a single seroincidence rate for all serum samples (only IgG)
seroincidenceData <- estimateSeroincidence(
  data = serologyData,
  antibodies = c("IgG"),
  strata = "",
  params = responseParams,
  censorLimits = cutOffs,
  par0 = baseLine)

# 5a. Produce summary of the results with 2.5% and 97.5% bounds...
summary(seroincidenceData)

# 5b. ...or produce summary of the results with 5% and 95% bounds, do not show
# convergence...
summary(seroincidenceData, quantiles = c(0.05, 0.95), showConvergence = FALSE)

# 5c. ...or produce summary and assign to an object...
seroincidenceSummary <- summary(seroincidenceData)
# ...and work with the results object from now on (here: display the results).
seroincidenceSummary$Results
```

## Acknowledgements

Project funded by European Centre for Disease Prevention and Control (ECDC).

## References

de Graaf, W. F., M. E. E. Kretzschmar, P. F. M. Teunis, and O. Diekmann. 2014. "A Two-Phase Within-Host Model for Immune Response and Its Application to Serological Profiles of Pertussis." *Epidemics* 9 (December): 1–7. https://doi.org/10.1016/j.epidem.2014.08.002.

Simonsen, J., K. Mølbak, G. Falkenhorst, K. A. Krogfelt, A. Linneberg, and P. F. Teunis. 2009. "Estimation of Incidences of Infectious Diseases Based on Antibody Measurements." *Statistics in Medicine* 28 (14): 1882–95. https://doi.org/10.1002/sim.3592.

Strid, M. A., J. Engberg, L. B. Larsen, K Begtrup, K Mølbak, and K. A. Krogfelt. 2001. "Antibody Responses to Campylobacter Infections Determined by an Enzyme–Linked Immunosorbent Assay: 2-Year Follow-up Study of 210 Patients." *Clinical and Vaccine Immunology.* https://doi.org/10.1128/CDLI.8.2.314-319.2001.

Teunis, P. F., J. C. van Eijkeren, C. W. Ang, Y. T. van Duynhoven, J. B. Simonsen, M. A. Strid, and W. van Pelt. 2012. "Biomarker Dynamics: Estimating Infection Rates from Serological Data." *Statistics in Medicine* 31 (208). https://doi.org/10.1002/sim.5322.

Teunis, P. F., J. C. van Eijkeren, W. F. de Graaf, A. Bonačić Marinović, and M. E. E. Kretzschmar. 2016. "Linking the Seroresponse to Infection to Within-Host Heterogeneity in Antibody Production." *Epidemics* 16 (September). https://doi.org/10.1016/j.epidem.2016.04.001.