Seroincidence package tutorial

European Centre for Disease Prevention and Control (ECDC)

2015-12-29

Contents

1. Introduction

Antibody levels measured by ELISA in a (cross-sectional) population sample can be translated into an estimate of the frequency with which seroconversions (infections) occur in the sampled population. Formulated simply: the presence of many high titres indicates that many subjects likely experienced infection recently, while low titres indicate a low frequency of infections in the sampled population. More information about ELISA methods can be found at 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 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 similar in any infected individual, and (3) setting up and running the longitudinal model requires additional skills, not necessary for use of the sero-incidence estimator package.

2. The seroincidence estimator

Package **seroincidence** was designed to use the longitudinal response characteristics by means of a set of parameters characterizing the longitudinal response of the selected serum antibodies. These parameters are peak concentrations and decay rates (describing the maximum antibody concentration reached after seroconversion and the rate with which the antibody concentration decreases, assuming first order, or exponential, kinetics).

Peak concentrations and decay rates may be provided as a set of parameters describing parametric distributions (a **gamma** distribution for the peak concentrations and an **inverse gamma** distribution for the decay rates, denoted by A and k, respectively), that characterize the heterogeneity of these response characteristics: peak concentrations and decay rates show considerable variation among individuals. This individual variation in antibody responses is accounted for by generating a Monte Carlo sample (default number of observations $n = \mathit{500}$) from these parametric distributions and calculating the resulting incidence (as in vignette methodology.pdf, equation (7)). Alternatively, these peak concentrations and decay rates may be provided as sets of Monte-Carlo samples, as can be obtained from longitudinal models (analyzed by *MCMC* methods). Note that, due to the limited number of parameter pairs, 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 undesired, the numbers of parameter pairs may be increased. This however will slow down the calculations.

Given these longitudinal response characteristics, the R-scripts delivered for the ECDC project provide a rapid and computationally simple method for calculating seroconversion rates, as published in (Teunis et al. 2012).

Given these longitudinal response characteristics, this package provide a rapid and computationally simple method for calculating seroconversion rates, as published in (Feller 1968).

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 function 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 operating system.

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 it is achieved by running the following command in the R console (bear in mind that the text after character `#` is only a comment):

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

At this moment you should have all functions required to run the seroincidence calculation available. You can check that by running:

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

Here is the resulting output:

```
## [1] "campylobacterResponseParams"      "campylobacterSerologyData"
## [3] "estimateSeroincidence"            "salmonellaSerologyData"
## [5] "simulateSalmonellaResponseParams" "simulateSerologyData"
```

We briefly describe the objects:

```
[1] "campylobacterResponseParams": Monte Carlo sample of longitudinal response
        parameters `A` and `k` per antibody for Campylobacter collected with
        SSI ELISA procedure.
        Object class: `list`.

[2] `campylobacterSerologyData`: Example antibody levels data measured in
        cross-sectional population sample for Campylobacter (Delft ELISA).
        Object class: `data.frame`.

[3] `estimateSeroincidence`: Main function of the package. Estimates
        seroincidence based on supplied cross-section antibody levels data
        and longitudinal response parameters.
        Object class: `function`.

[4] `salmonellaSerologyData`: Example antibody levels data measured in
        cross-sectional population sample for Salmonella (SSI Mixed ELISA).
        Object class: `data.frame`.

[5] `simulateSalmonellaResponseParams`: Function to simulate the longitudinal
        response parameters `A` (peak level) and `k` (decay rate) per
        antibody for Salmonella.
        Object class: `function`.

[6] `simulateSerologyData`: Function to simulate cross-sectional serology
        data for three antibodies: `IgG`, `IgM`, `IgA`.
        Object class: `function`.
```

3.2. Specifying input data

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

a) Antibody levels measured in cross-sectional population sample (see `salmonellaSerologyData`, `campylobacterSerologyData`, `simulateSerologyData` above).
b) Longitudinal response characteristic as pair of peak concentrations and decay parameters `(A, k)` (see `simulateSalmonellaResponseParams`, `campylobacterResponseParams` 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 many options of providing this data to the calculator. Here are a few:

a) Using supplied sample data

Data sets `salmonellaSerologyData` and `campylobacterSerologyData` provide examples of cross-sectional antibody levels data for Salmonella and Campylobacter, respectively. They are loaded into the workspace as soon as the package is loaded, therefore they can be referenced right away. Here are small exempts from these data sets:

```r
# Show first rows of "salmonellaSerologyData" data.frame
head(salmonellaSerologyData)
```

```
##        id  IgG  IgM   IgA sex age
## 1 1800002 0.13 0.09 0.050   1  35
## 2 1800005 0.36 0.14 0.160   2  47
## 3 3500002 0.22 0.20 0.050   1  54
## 4 1800003 0.70 0.21 0.001   1  28
## 5 3500003 0.26 0.22 0.080   1  38
## 6 2100005 0.12 0.16 0.040   2  37
```

```r
# Show first rows of "campylobacterSerologyData" data.frame
head(campylobacterSerologyData)
```

```
##        IgG      IgM      IgA
## 1 0.960035 0.588600 0.302515
## 2 0.320800 0.191485 0.283095
## 3 0.720960 0.718775 0.307855
## 4 0.896630 0.303260 0.242810
## 5 0.351110 1.223450 0.515610
## 6 1.061600 0.710760 0.534545
```

Let us assign one of the sets to variable `serologyData`:

```r
# Assign data.frame "salmonellaSerologyData" to object named "serologyData"
serologyData <- salmonellaSerologyData
```

In this particular example `sex = 1` denotes “male” and `sex = 2` denotes “female”.

We shall pass this object later to the calculator. Of course, object `salmonellaSerologyData` can be used as an argument of the calculator directly as well.

b) Simulating data from parametric model

Another option is to generate cross-sectional data from the supplied Monte-Carlo simulation function `simulateSerologyData`. This function only exists for testing/training, the simulations do not represent outcomes of any seroconversion process. The underlying parametric models are log-Normal distributions with parameters:

- IgG: `mu = 1.246, sd = 0.757`
- IgM: `mu = -0.764, sd = 0.413`
- IgA: `mu = -1.13, sd = 0.318`

By default, 500 observations are generated if no argument `n` is supplied (`n`=number of observations). In the following example we override the default number of generated observations and set it to `300`:

```r
# Assign output of function "simulateSerologyData" to object named "serologyData"
serologyData <- simulateSerologyData(n = 300)

# Show first rows of object "serologyData"
head(serologyData)
```

```
##         IgG       IgM       IgA
## 1  2.697145 0.4907698 0.2634563
## 2  4.520141 0.8373940 0.6295903
## 3  1.641992 0.3637384 0.3139299
## 4  8.014495 0.3505074 0.5222766
## 5  1.666772 0.9689420 0.3486882
## 6 24.879401 0.5259398 0.2968051
```

c) Loading from external files

Finally, 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] (http://cran.r-project.org/doc/manuals/r-release/R-data.html).

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:

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

Please, note that the cross-sectional data set can include extra columns allowing performing the seroincidence calculations stratified per factors available in those data. For instance, data set `salmonellaSerologyData` includes factors `sex` and `age`.

3.2.2. Specifying longitudinal response parameters (`A, k`)

The longitudinal response parameters set consists of two items:

- `A` = antibody peak level (ELISA units)
- `k` = antibody decay rate (1/days for the current longitudinal parameter sets)

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:

a) Using supplied data

For Campylobacter there is a data set provided by the package including Monte-Carlo sample of longitudinal response parameters, `campylobacterResponseParams`. This object is a list containing two data.frame objects named `A` and `k`. Let's have a pick into each of these two data sets separately (notice character `$` indicating a subobject of the parent object):

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

```
##         IgG       IgM       IgA
## 1 1.5841532 0.8079065 0.7003339
## 2 1.8657514 0.4662612 0.4881885
## 3 1.3126431 0.9829273 0.7607791
## 4 1.1131536 0.1463502 1.5342800
## 5 1.4794378 0.1940006 0.2454033
## 6 0.9348555 0.7127425 0.9043005
```

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

```
##           IgG          IgM         IgA
## 1 0.001312580 0.0012365599 0.022609111
## 2 0.001438733 0.0010868782 0.021309170
## 3 0.001351047 0.0013366391 0.016486080
## 4 0.001048270 0.0009022539 0.014820666
## 5 0.001274966 0.0005063741 0.005397104
## 6 0.001437053 0.0013184843 0.017098452
```

Such data set is provided only for Campylobacter. Let us assign object `campylobacterResponseParams` to variable `responseParams`:

```
responseParams <- campylobacterResponseParams
```

b) Simulating data from parametric model

Alternatively, for Salmonella an example of the `A` and `k` parameters can be generated using function `simulateSalmonellaResponseParams` provided by the package:

```
responseParams <- simulateSalmonellaResponseParams()
```

Similarly to function `simulateSerologyData` this function also accepts argument `n` specifying the number of observations to simulate. The default is `n = 500`.

The underlying parametric models for peak levels `A` are Gamma distributions with parameters:

- IgG: `shape = 1.1175, scale = 0.848`
- IgM: `shape = 1.337, scale = 0.902`
- IgA: `shape = 1.205, scale = 0.651`

The underlying parametric models for decay rates `k` are Inverse Gamma distributions with parameters:

- IgG: `shape = 0.869, scale = 1/728.5`

- IgM: `shape = 0.731, scale = 1/514.6`
- IgA: `shape = 1.759, scale = 1/132/5`

c) Loading from external files

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

```r
AData <- read.csv(file = "A.csv")
kData <- read.csv(file = "k.csv")

# Create a list named "responseData" containing objects named "A" and "k"
responseParams <- list(A = AData, k = kData)
```

Note that object `responseParams` is a list with two dataframes named `A` and `k` as required.

3.3. Estimating seroincidence for human Salmonella and Campylobacter infections

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

- `data`: A data frame with the cross-sectional data. 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 `age` in the example data set can be used as an example. If `strata = ""` the whole data set is treated as a single stratum; if each sample is assigned a unique identifier (e.g. `strata = "id"`) 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 `""`.
- `Ak`: 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. It is a required input.
- `showProgress`: A Boolean input indicating whether or not to show progress of the calculation. It is an optional input. If not specified then it is initialized to `FALSE`.

Here is an example of the Salmonella seroincidence calculation for three antibodies measured, stratified per sex, and with measurements below 0.25 removed:

```r
# Assign output of function "estimateSeroincidence" to object named "seroincidenceData"
serologyData <- salmonellaSerologyData
responseParams <- simulateSalmonellaResponseParams()
seroincidenceData <- estimateSeroincidence(data = serologyData,
                                           antibodies = c("IgG", "IgM", "IgA"),
                                           strata = "sex",
                                           Ak = responseParams,
                                           censorLimits = list(IgG = 0.25, IgM = 0.25, IgA = 0.25))
```

```
# 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:         sex
## c) Censor limits:  IgG = 0.25, IgM = 0.25, IgA = 0.25
##
## 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). 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.

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)
```

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:         sex
## c) Censor limits:  IgG = 0.25, IgM = 0.25, IgA = 0.25
## d) Quantiles:      0.025, 0.975
## e) Seroincidence estimates
##      Lambda Lambda.lwr Lambda.upr Deviance Convergence Stratum
## 1 0.1218136 0.10788284  0.1375432 2144.977           0       1
## 2 0.1063245 0.09367877  0.1206772 1908.507           0       2
```

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).

In this particular example `lambda=0.12` for stratum `1` means that yearly 12% of people are infected based on the sample.

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

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

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

# Show the results
seroincidenceSummary$Results
```

```
##      Lambda Lambda.lwr Lambda.upr Deviance Convergence Stratum
## 1 0.1218136 0.10788284  0.1375432 2144.977           0       1
## 2 0.1063245 0.09367877  0.1206772 1908.507           0       2
```

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.

```r
# Calculate seroincidence rates for Salmonella

# 1. Define cross-sectional data
serologyData <- salmonellaSerologyData

# 2. Define longitudinal response data (simulate 1000 observations)
responseParams <- simulateSalmonellaResponseParams(n = 1000)

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

# 4a. Calculate seroincidence rates by age (triplet of titres)...
seroincidenceData <- estimateSeroincidence(data = serologyData,
                        antibodies = c("IgG", "IgM", "IgA"),
                        strata = "age",
                        Ak = responseParams,
                        censorLimits = cutoffs,
                        showProgress = TRUE)

# 4b. ...or calculate a single seroincidence rate for all serum samples (triplet of titres)...
seroincidenceData <- estimateSeroincidence(data = serologyData,
                        antibodies = c("IgG", "IgM", "IgA"),
                        strata = "",
                        Ak = responseParams,
                        censorLimits = cutoffs)
```

```r
# 4c. ...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 = "",
                            Ak = responseParams,
                            censorLimits = cutoffs)

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

# 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

Feller, W. 1968. *An Introduction to Probability Theory and Its Applications. John Wiley & Sons*. Vol. 2.

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*. doi: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 (20): 2240–48. doi:10.1002/sim.5322.