Tutorial for siestim

Kurnia Susvitasari

2023-08-25

siestim is an R package that estimates the serial interval distribution in incomplete sampling data. The input is a vector (or a list of vectors) of the serial interval data. Given the data, siestim will estimate not only the parameters (mean and standard deviation) of the serial interval distribution, but also the parameters that explain the data incompleteness, e.g. the success probability of sampling the secondary case in a transmission chain.

The package is created as a complementary for paper “A Method to Estimate Serial Interval Distribution Under Partially-sampled Data”, which is co-author by Kurnia Susvitasari, Paul Tupper, Jessica Stockadale, and Caroline Colijn.

To install the package, you can run the following R command

```
remotes::install_github("ksusvita92/siestim")
```

Once installed, you should be able to call the package using:

```
library(siestim)
```

This tutorial consists of two parts. In the first half, we will simulate a data set of observed serial intervals directly from the Compound Geometric Gamma (CGG) distribution and the Folded Gamma Difference (FGD) distribution; see [1]. Then, we will estimate the following parameters of interest:

- mean (mu) of the serial interval distribution,
- standard deviation (sigma) of the serial interval distribution
- success probability (pi) of sampling the secondary case
- proportion of non-coprimary transmission (w); see [1].

In the second half, we will simulate an SIR outbreak and then we will estimate the same parameters of interest as above. This section is to show that siestim can also be implemented in a data set that contains stochastic uncertainty.

Distribution-based Simulation

Under incomplete sampling, the infector-infectee pairs may not represent direct transmissions. We differentiate two types of transmission: coprimary and non-coprimary. Coprimary transmission is defined as a pair in which both the infector and infectee were directly infected by the same unsampled case; non-coprimary transmission includes both direct transmission and inderect transmission (a pair in which the infector and infectee is separated by at least one unsampled cases).

We model the serial intervals that come from coprimary transmission using FGD distribution and those come from non-coprimary transmission using CGG distribution. The idea of siestim is to use mixture model to estimate the parameters of interest.

Folded Gamma Difference (FGD) Distribution

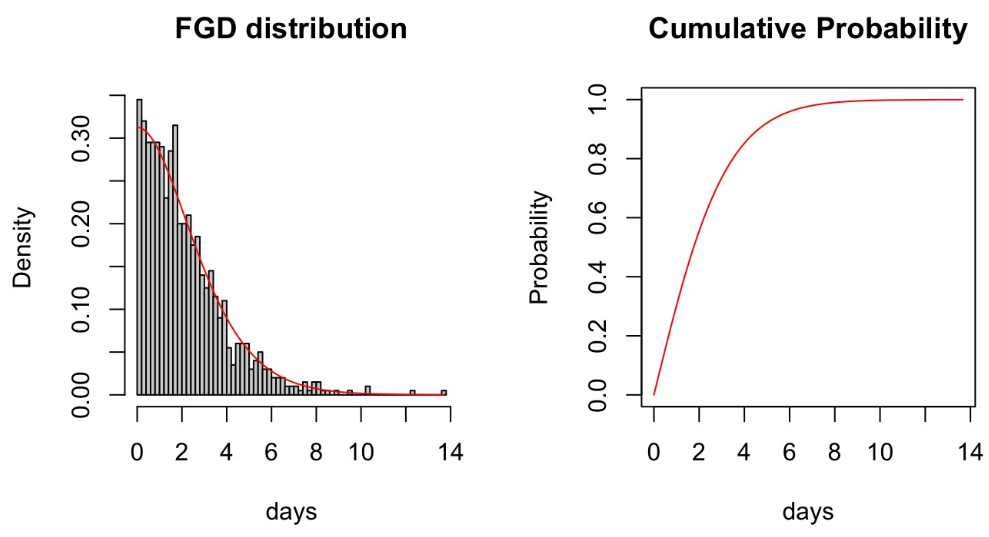
Suppose X_{xi} and X_{xj} represent the serial intervals between transmission pair $x \to i$ and $x \to j$ where x is an unsampled case. Then, a random variable

$$Y = |X_{xi} - X_{xj}|$$

represents the symptom onset difference between i and j. In this case, the transmission pair $i \to j$ is referred to coprimary transmission and thereby, is modeled by the FGD distribution.

To generate random samples from FGD distribution of size 1000 with mean 4 days and standard deviation 2 days, you can run the following commands

```
set.seed(12)
x <- rfgd(1000, 4, 2)

# Fitting the histogram with the density
par(mfrow = c(1,2))
hist(x, breaks = 50, freq = F, xlab = "days", main = "FGD distribution")
lines(sort(x), dfgd(sort(x), 4, 2), col = "red")
base::plot(sort(x), pfgd(sort(x), 4, 2), xlab = "days", ylab = "Probability", main = "Cumulative Probability", ty
pe = "l", col = "red")
```



siestim also provides density, cumulative probability, and quantile functions of a FGD distribution; see `?fgd` .

Compound Geometric Gamma (CGG) Distribution

Suppose that the transmission pair $i \to j$ is separated by $m = 1, \ldots, M$ unsampled cases; if $m = 0$, it means that $i \to j$ is a direct transmission.

$$i \to person_1 \to \ldots \to person_M \to j$$

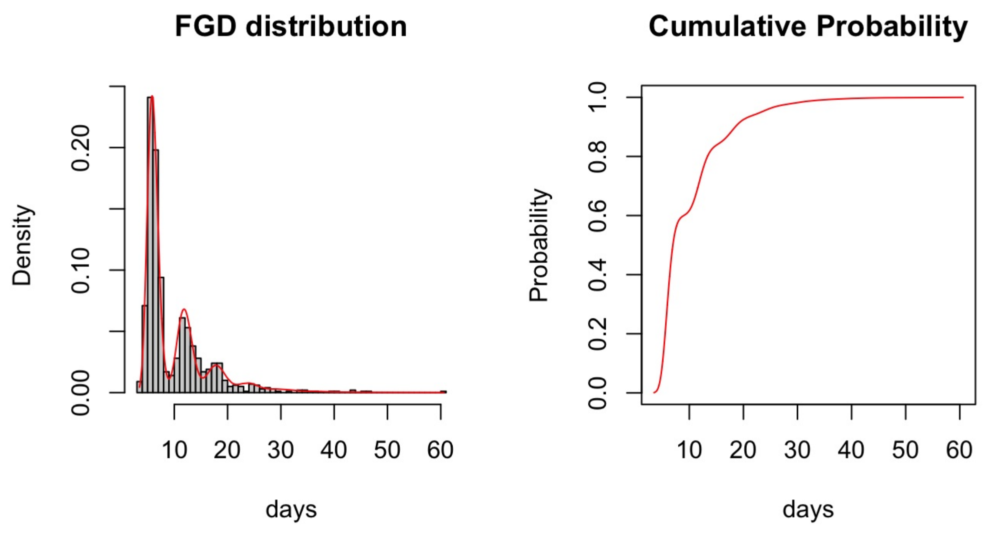
Assuming the serial intervals in a transmission chain are identically distributed and independent, the symptom onset difference between case i and j is the convolution over all underlying serial intervals in the transmission pair $i \to j$, that is

$$Y = X_{iperson_1} + \ldots + X_{person_M j}$$

In this case, Y is modeled by CGG distribution having parameters (μ, σ, π); π is the success probability of sampling the secondary case.

To generate random samples from CGG distribution of size 1000 with mean 6 days, standard deviation 1 days, and success probability 0.6, you can run the following commands

```
set.seed(12)
x <- rcgg(1000, 6, 1, .6)

# Fitting the histogram with the density
par(mfrow = c(1,2))
hist(x, breaks = 50, freq = F, xlab = "days", main = "FGD distribution")
lines(sort(x), dcgg(sort(x), 6, 1, .6), col = "red")
base::plot(sort(x), pcgg(sort(x), 6, 1, .6), xlab = "days", ylab = "Probability", main = "Cumulative Probability"
, type = "l", col = "red")
```



siestim also provides density, cumulative probability, and quantile functions of a FGD distribution; see `?cgg` .

Parameter Estimation

To perform parameter estimation, we need to generate a collection of serial interval data first. You can generate random samples of size 500 with parameters:

- $\mu = 4$ days
- $\sigma = 2$ days
- $\pi = .7$
- $w = .6$

by running these following commands

```
# true parameters
mu <- 4; sigma <- 2; pi <- .75; w <- .6

set.seed(12)
n <- rbinom(1, 500, w)
x <- c(rcgg(n, mu, sigma, pi), rfgd(500-n, mu, sigma))
```

To estimate the parameters of interest, you can run

```
iv <- c(5,3,.5,.5) # initial values of the estimates
lb <- rep(0, 4) # lower bound of the estimates
ub <- c(10, 5, 1, 1) # upper bound of the estimates

myest <- siestim(x, iv, lb, ub)
myest
#>
#>  ----- Serial Interval Estimation -----
#> ============================================
#>
#> Parameter estimations:
#>    mu sigma    pi     w
#> 4.156 2.055 0.789 0.547
#>
#> Standard error:
#>    mu sigma    pi     w
#> 0.475 0.375 0.064 0.100
#>
#> Convergence message:
#> [1] "Successful convergence"
```

As well, you can estimate the 95% confidence intervals for each estimate by running the following command

```
getci(myest, .95)
#>           lower     upper
#> mu    3.2243807 5.0876834
#> sigma 1.3202107 2.7890577
#> pi    0.6638437 0.9133689
#> w     0.3519951 0.7427887
```

SIR Simulation

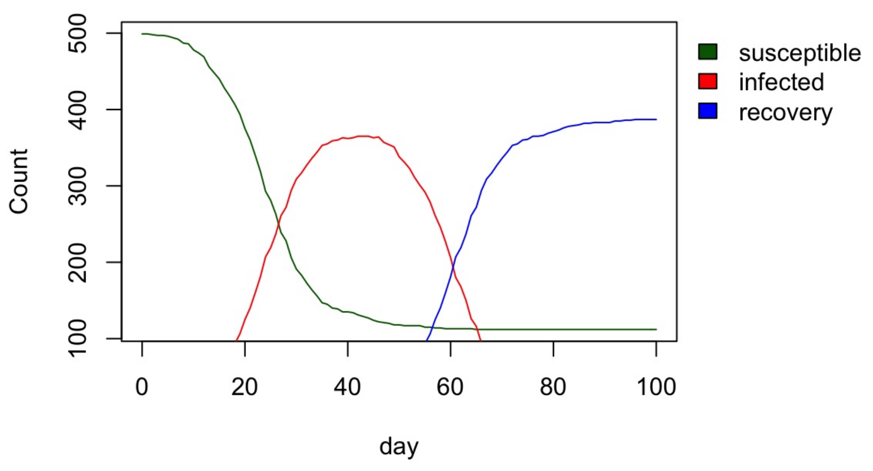
You can generate an influenza-like outbreak using siestim using a function called `simOutbreak`. The function generates an SIR outbreak together with, for every infected case, the DNA sequences and epidemiological data (who infected whom, time of infection and recovery).

Suppose you want to generate an outbreak with these parameters:

- population size: 500
- epidemic duration: 100 days
- initial infected case: 1
- reproduction number, R_0: 2
- generation interval: $\Gamma(4, 2)$
- importation rate: 0.01
- transversion rate: 5×10^{-5}
- transition rate: 10^{-4}

Then, you can run

```
set.seed(12)
myoutbreak <- simOutbreak(500, c(4,2))
#> Attempt 1 : Generate outbreak with ncases = 388
#>
#> Attaching package: 'dplyr'
#> The following objects are masked from 'package:stats':
#>
#>     filter, lag
#> The following objects are masked from 'package:base':
#>
#>     intersect, setdiff, setequal, union

par(oma=c(0, 0, 0, 5))
base::plot(0:100, myoutbreak$dynam$nsus, type = "l", col = "darkgreen", xlab = "day", ylab = "Count")
lines(0:100, myoutbreak$dynam$ninf, col = "red")
lines(0:100, myoutbreak$dynam$nrec, col = "blue")
legend(par('usr')[2], par('usr')[4], bty = 'n', xpd = NA, legend = c("susceptible", "infected", "recovery"), fill
= c("darkgreen", "red", "blue"))
```

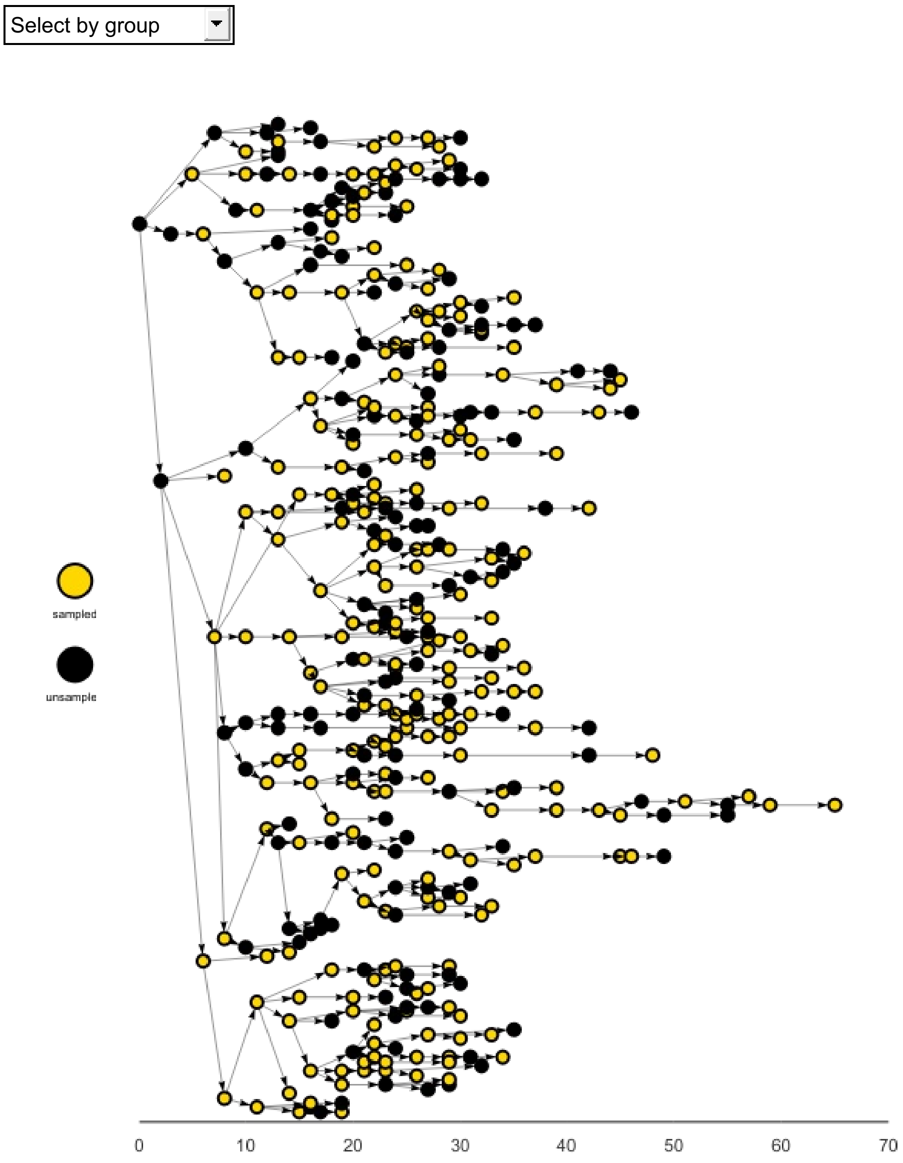


Reconstruct a Transmission Cloud

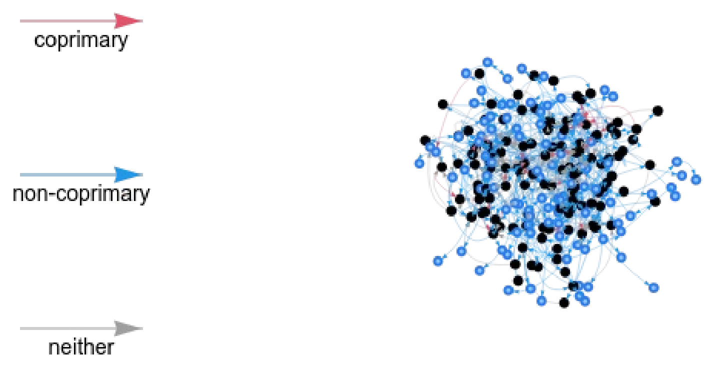
Assuming that we sample only p proportion of infected cases and we do not know the true transmission pairs. In this case, we need to reconstruct all plausible transmission pairs from the genomic data. We say that case i and j are linked as an infector-infectee pair if their genomic distance is less than a non-negative threshold ε. We refer all plausible transmission pairs as *transmission cloud*.

You can sample $p = .6$ of the infected cases to reconstruct the transmission cloud with the genomic distance cutoff $\varepsilon = 18$ SNPs using

```
mytc <- createTC(myoutbreak, .6, 18, seed = 12)

# true transmission tree
## plot using epicontacts
library(epicontacts)
tt <- make_epicontacts(mytc$tt,
    mytc$tt %>%
      filter(!is.na(inf.source)),
    id = "inf.ID",
    from = "inf.source",
    to = "inf.ID",
    directed = T)

adegenet::plot(tt,
     thin = F,
     x_axis = "inf.times",
     node_color = "group",
     node_value = 1,
     col_pal = c(unsampled = "black", sampled = "gold"),
     arrow_size = .5,
     node_size = 5,
     edge_width = .5,
     node_width = .5,
     height = 800,
     width = 600,
     label = F)
```



```
# plot transmission cloud
## plot using epicontacts
tc <- make_epicontacts(mytc$tt %>% filter(group == "sampled"),
    mytc$tc,
    id = "inf.ID",
    from = "inf.source",
    to = "inf.ID",
    directed = T)
tc$contacts$si <- tc$contacts$si+1 # edge's length proportional to the serial interval

adegenet::plot(tc,
    thin = T,
    label = F,
    node_color = NULL,
    #
    edge_color = "type",
    edge_col_pal = c(coprimary = "#DF536B", `non-coprimary` = "#2297E6", neither = "#9E9E9E"),
    edge_width = "si",
    #
    height = 800,
    width = 600,
    legend_width = .2)
```



Parameter Estimation

Sometimes we may be uncertain about which cases are truly linked. For a given case, there may be multiple other cases that may have infected them (see the transmission cloud above). To handle this extra source of uncertainty, we sample a collection of transmission trees consistent with it, and then estimate the parameters of interest over all sampled trees.

To sample, say, 100 transmission trees from given transmission cloud, you can run

```
mytt <- lapply(1:100, function(i){
  mytc$tc %>%
    group_by(inf.ID) %>%
    slice_sample(n = 1) %>%
    pull(si)
})
```

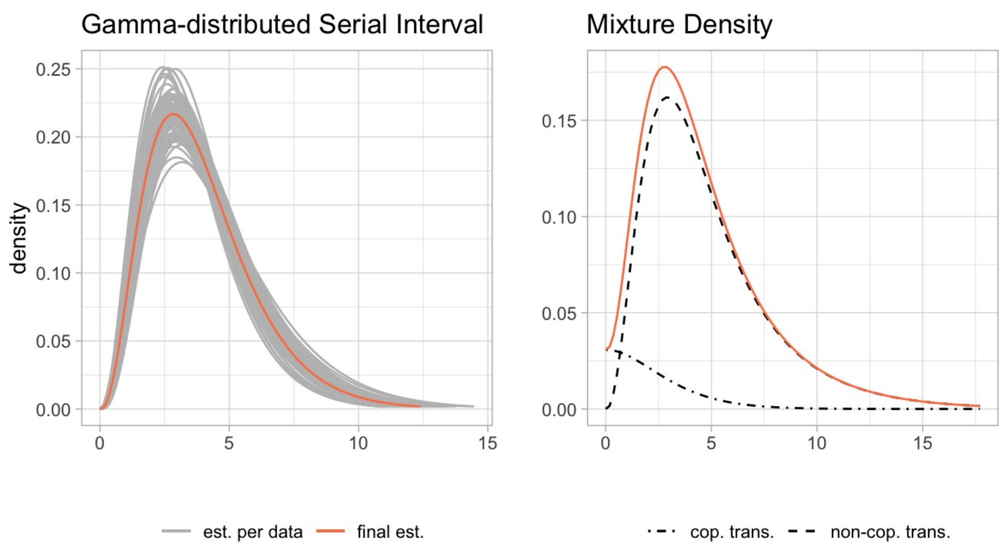
We estimate the serial interval distribution using the collection of serial interval data sets above. The following command allows you to run siestim in parallel; see `?parallel::detectCores()` for detail.

```
myest <- siestim(mytt, iv, lb, ub, list(ncore = parallel::detectCores()))
```

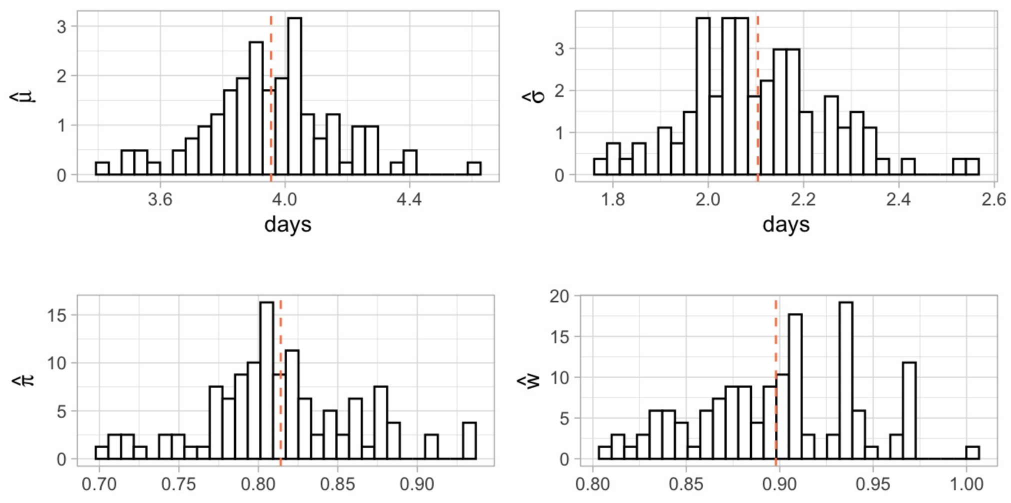
```
myest
#>
#>  ----- Serial Interval Estimation -----
#> ============================================
#>
#> Parameter estimations:
#>    mu sigma    pi     w
#> 3.955 2.104 0.814 0.898
#>
#> Standard error:
#>    mu sigma    pi     w
#> 0.536 0.433 0.108 0.072
```

To plot the siestim output, you can use

```
siestim::plot(myest)
#> $density
```



```
#>
#> $est_hist
```



You can also estimate the 95% confidence interval using

```
getci(myest, .95)
#>             lower    upper
#> mu    2.9040124 5.006162
#> sigma 1.2556631 2.952586
#> pi    0.6022847 1.000000
#> w     0.7563758 1.000000
```

The function siestim allows you to include prior distributions on each parameter μ, σ, π, w. For more detail, see `?siestim`.

1

1. Stockdale, J.E., Susvitasari, K., Tupper, P. et al. Genomic epidemiology offers high resolution estimates of serial intervals for COVID-19. *Nat Commun* **14**, 4830 (2023).↩

Processing math: 100%