Estimating HIV transmission rates with `rcolgem`

Erik M Volz

December 5, 2014

This vignette will demonstrate how to use a coalescent models as described in [2] to estimate transmission rate parameters given a pathogen genealogy.

Suppose an epidemic occurs according to a density-dependent susceptible-infected-recovered process, and a given infected individual generates a new infection at the rate βSI, where S is the number susceptible and β is the transmission rate. Furthermore, infected individuals will be removed from the population at per capita rate γ. At a single point in time, a random sample of $n = 75$ infected individuals is taken and the genealogy is reconstructed from the history of transmissions. We have simulated such a dataset using MASTER 1.10[1], which can be loaded by

```
> library(rcolgem)
> tree <- read.tree(system.file( extdata/sirModel0.nwk, package=rcolgem))
```

And, the epidemic trajectory information can be loaded by

```
> library(rjson)
> epidata <- fromJSON(file=system.file(extdata/sirModel0.json, package=rcolgem))
```

The true parameter values are given in table 1. The file used to simulate the data can be viewed by `file.show( system.file('extdata/sirModel0.xml', package=rcolgem))` .

We will fit a simple ODE model to the genealogy:

$$\dot{S} = -\beta SI \tag{1}$$

$$\dot{I} = \beta SI - \gamma I \tag{2}$$

Relevant parameters of the system are the transmission rate β, recovery rate γ, initial population size $S(0)$ and initial number infected $I(0)$. Not all parameters are identifiable from these data, so we will assume prior knowledge of $S(0)$ and γ and focus on estimating β and the nuisance parameter $I(0)$. Note that an imprecise estimate of $S(0)$ is also possible.

Create a list to store the true parameter values:

```
> parms_truth <- list( beta = .00020002, gamma = 1, S0 = 9999, t0 = 0 )
```

Note that the true value of R_0 is $\beta S(0)/\gamma = 2$.

And, create a tree with dated tips and internal nodes:

Table 1: Parameter symbols and values.

Parameter	Symbol	Value
Duration infection	$1/\gamma$	1
Transmission rate	β	2.0002e-4
Population size	$S(0)$	9999
Initial num infected	$I(0)$	1
Time of sampling	T	12

```
> sampleTimes <- rep(12, 75)
> names(sampleTimes) <- tree$tip.label
> bdt <- binaryDatedTree( tree, sampleTimes=sampleTimes)
> bdt

Phylogenetic tree with 75 tips and 74 internal nodes.

Tip labels:
        24, 7, 36, 75, 52, 38, ...

Rooted; includes branch lengths.
```

Note that the vector of sample times must have names corresponding to the taxon labels in tree.

In order to fit this model, we need to express the equations in a canonical format:

```
> births <- c( I = parms$beta * S * I )
> deaths <- c( I = parms$gamma * I )
> nonDemeDynamics <- c(S = -parms$beta * S * I)
```

The `births` vector gives the total rate that all infected generate new infections and `deaths` gives the rate that lineages are terminated. The `nonDemeDynamics` vector gives the equations for state variables that aren't directly involved in the genealogy (e.g. because a pathogen never occupies a susceptible host by definition).

Each element of the vectors is a string that will be parsed as R code and evaluated, so it is important to write it exactly as you would if you were solving the equations in R. Also note that the object `parms` is accessible to these equations, which is a list of parameters- this may include parameters to be estimated. Also note that we *must* give names to the vectors, and these names must correspond to the names of the demes.

We will use these initial conditions

```
> x0 <- c(I=1, S= unname(parms_truth$S0) )
> t0 <- bdt$maxSampleTime - max(bdt$heights) -1
```

The time of origin t_0 is chosen somewhat arbitrarily, but should occur before the root of the tree.

Now we can calculate the likelihood of the tree and assess how long it takes:

```
> print(
+ system.time(
+ print(
+   coalescent.log.likelihood(
+     bdt
+     , births, deaths, nonDemeDynamics
+     , t0, x0
+     , parms=parms_truth
+     , fgyResolution = 1000
+     , integrationMethod = rk4)
+ )))

[1] 67.75564
   user  system elapsed
  0.672   0.000   0.674
```

Note that changing the `integrationMethod` (choose 'euler'), `censorAtHeight` (only fit to part of the tree) and `fgyResolution` (set to a smaller value) options can dramatically speed up the calculation at the cost of some accuracy.

We can fit the model using maximum likelihood with the `bbmle` or `stats4` packages.

```
>         library(bbmle)
```

First, create the objective function to be minimized:

```
> obj_fun <- function(lnbeta, lnI0)
+ {
+         beta <- exp(lnbeta)
+         I0 <- exp(lnI0)
+         parms <- parms_truth
+         parms$beta <- beta
+         x0 <- c(I=unname(I0), S = unname(parms$S0) )
+         mll <- -coalescent.log.likelihood(
+                 bdt
+                 , births, deaths, nonDemeDynamics
+                 ,  t0, x0
+                 , parms=parms
+                 , fgyResolution = 1000
+                 , integrationMethod = rk4)
+         print(paste(mll, beta, I0))
+         mll
+ }
```

Note that this uses log-transformation for variables that must be positive (like rates and population sizes).

We can then fit the model by running

```
> fit <- mle2(
+   obj_fun
+   , start=list(lnbeta=log(parms_truth$beta*.75), lnI0=log(1))
+   , method=Nelder-Mead
+   , optimizer=optim
+   , control=list(trace=6, reltol=1e-8)
+ )
```

Note that we are starting the optimizer far from the true parameter values. If fitting a model to real data, it is recommended to try many different starting conditions over a large range of values. The optimizer would take a few minutes to run, so instead we will load the results:

```
>         load( system.file(extdata/sirModel0-fit.RData, package=rcolgem) )
>         AIC(fit)

[1] -145.7974

>         logLik(fit)

log Lik. 74.89871 (df=2)

>         coef(fit)

    lnbeta       lnI0
-8.4748155  0.1351695

>         exp(coef(fit))

      lnbeta         lnI0
0.0002086577 1.1447308446

>         # how biased is the estimate?
>         exp(coef(fit)[lnbeta]) - parms_truth$beta

      lnbeta
8.637689e-06
```

We can compare the fitted model to the true number of infected through time, which is shown in figure 1.

```
> beta <- exp(coef(fit)[lnbeta])
> I0 <- exp(coef(fit)[lnI0])
> parms <- parms_truth
> parms$beta <- beta
```

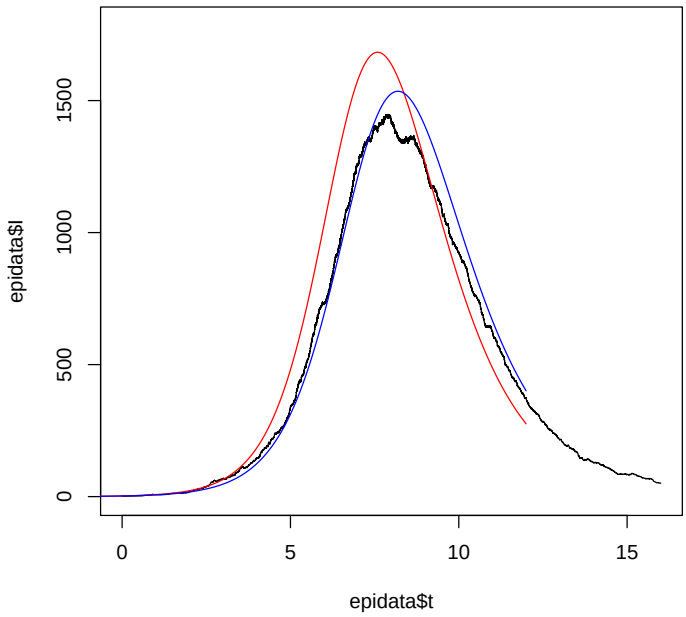

Figure 1: The actual (black) and estimated (red) number of infections through time. The blue line shows the SIR model prediction under the true parameter values.

```
> x0 <- c(I=unname(I0), S = unname(parms$S0) )
> o <- solve.model.unstructured(t0,bdt$maxSampleTime, x0
+    , births
+    ,  deaths
+    , nonDemeDynamics, parms)
> otruth <- solve.model.unstructured(t0, bdt$maxSampleTime, x0
+   , births
+   ,  deaths
+   , nonDemeDynamics, parms_truth)
> plot(epidata$t, epidata$I, type=line
+   , ylim=c(0, 100+max(max(o[,2]),max(epidata$I))))
> lines(o[,1], o[,2], col=red )
> lines(otruth[,1], otruth[,2], col=blue )
```

We can calculate a confidence interval for the transmission rate using likelihood profiles:

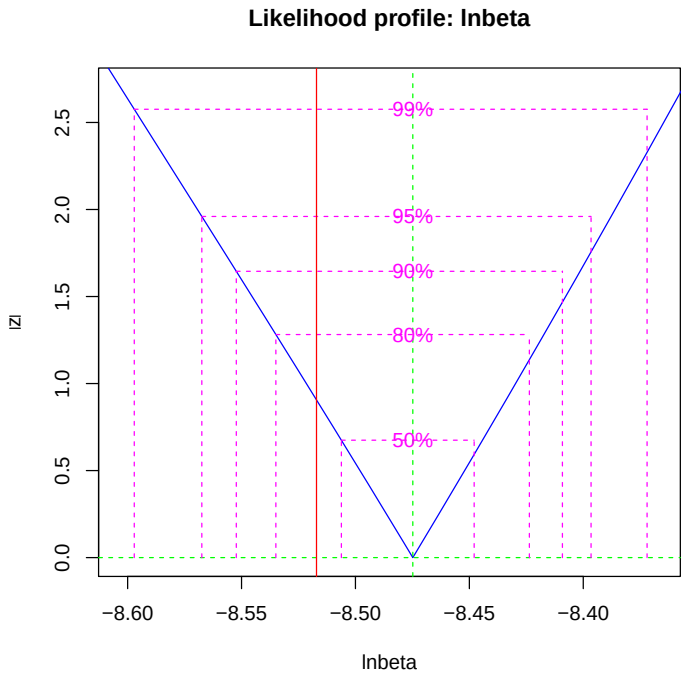

Figure 2: Likelihood profile for the transmission rate β with confidence levels. The true parameter value is indicated by the vertical red line.

```
> profbeta <- profile(fit, which=1, alpha=.05
+   , std.err=1, trace=TRUE, tol.newmin=1 )
```

This takes a few minutes, so we will load the results:

```
> load( system.file(extdata/sirModel0-profbeta.RData,  package=rcolgem) )
```

We see that the confidence interval covers the true value:

```
> c( exp( confint( profbeta ) ), TrueVal=parms_truth$beta )
```

```
       2.5 %       97.5 %      TrueVal
0.0001901721 0.0002282366 0.0002000200
```

And, we can visualize the profile (Figure 2).

```
> plot(profbeta)
> abline( v = log( parms_truth$beta) , col=red)
```

References

[1] Timothy G Vaughan and Alexei J Drummond. A stochastic simulator of birth–death master equations with application to phylodynamics. *Molecular biology and evolution*, 30(6):1480–1493, 2013.

[2] Erik M Volz. Complex population dynamics and the coalescent under neutrality. *Genetics*, 190(1):187–201, 2012.