Estimating HIV transmission rates with `rcolgem`

Erik M Volz

July 23, 2014

This vignette will demonstrate how to use structured coalescent models[1] to estimate epidemiological parameters given a pathogen genealogy and discrete-trait information about sampled individuals. We will focus on the problem of estimating HIV transmission rates by stage of infection and assuming we know the stage of infection when patients are sampled and virus is sequenced.

We will fit a very simple model of an HIV epidemic using ordinary differential equations. In this model, the infectious period will be broken into three stages of different average duration and with different transmission rates. The first stage, *early HIV infection* (EHI) is short (average duration $1/\gamma_0$=1 year), but has high transmission rate β_0. The second stage, *chronic HIV infection*, is long (average duration $1/\gamma_1$=7 years), and has small transmission rate. The last stage, AIDS, lasts $1/\gamma_2 = 2$ years on average and has an intermediate transmission rate. There are births into the susceptible state at rate bN where $N = S+I_0+I_1+I_2$. And there is mortality due to natural causes from all states at rate μ. The parameter values are listed in table 1. The model equations are as follows:

$$\dot{S} = bN - \mu S - (\beta_0 I_0 + \beta_1 I_1 + \beta_2 I_2)S/N \tag{1}$$

$$\dot{I}_0 = (\beta_0 I_0 + \beta_1 I_1 + \beta_2 I_2)S/N - (\mu + \gamma_0)I_0 \tag{2}$$

$$\dot{I}_1 = \gamma_0 I_0 - (\mu + \gamma_1)I_1 \tag{3}$$

$$\dot{I}_2 = \gamma_1 I_1 - (\mu + \gamma_2)I_2 \tag{4}$$

The model is also illustrated in figure 1.

The package is loaded by

```
> library(rcolgem)
```

Create a list of the true parameter values:

In order to fit this model, we need to express the equations in a canonical format. According to this format, we will tally birth and migration events between demes. In our example, the deme corresponds to the stage of infection that an infected host can be in, so we will refer the demes with the following names:

```
> INFECTEDNAMES <- c('I0', 'I1', 'I2')
```

Table 1: Parameter symbols and values.

Parameter	Symbol	Value
Duration EHI	$1/\gamma_0$	1 year
Duration chronic	$1/\gamma_1$	7 years
Duration AIDS	$1/\gamma_2$	2 years
Birth rate	b	0.036
Natural death rate	μ	1/30
EHI transmission rate	β_0	1.2
Chronic transmission rate	β_1	0.03
AIDS transmission rate	β_2	0.09

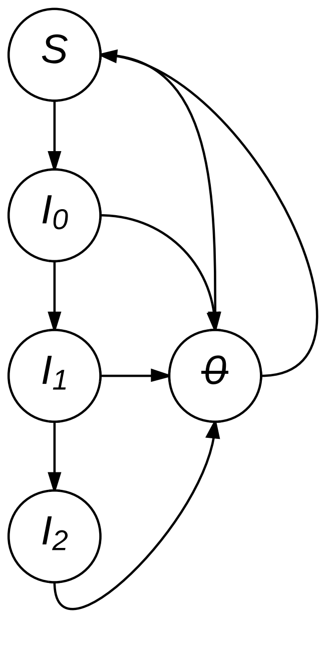

Figure 1: An illustration of the HIV compartmental model.

There are $m = 3$ demes in this model, so the birth events between demes needs to be expressed with a 3×3 matrix F. The element F_{kl} represents the rate of transmissions by a host in deme k to a host in deme l. In our example, this is the following:

```
> births <- rbind(
+   c('parms$beta0 * S * I0 / (S + I0 + I1 + I2)', '0', '0'),
+   c('parms$beta1 * S * I1 / (S + I0 + I1 + I2)', '0', '0'),
+   c('parms$beta2 * S * I2 / (S + I0 + I1 + I2)', '0', '0')
+ )
> rownames(births)=colnames(births)<- INFECTEDNAMES
```

Each element of the matrix 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 outside of colgem. Also note that the object `parms` is accessible to these equations, which is a list of parameters- this may include parameters to be estimated. Note that there are zero rates in the 2nd and third columns, since all new infected hosts start out in the first stage of infection (EHI). Also note that we *must* give row and column names to the matrix, and these names must correspond to the names of the demes.

Similarly, we must create a matrix of migrations:

```
> migrations <- rbind(
+   c('0', 'parms$gamma0 * I0', '0'),
+   c('0', '0', 'parms$gamma1 * I1'),
+   c('0', '0', '0')
+ )
> rownames(migrations)=colnames(migrations) <- INFECTEDNAMES
```

Note that this matrix tallys the stage progression from EHI to chronic and from chronic to AIDS.

We must also write a vector of expressions for events that terminate a lineage– In this model, this occurs due to natural or AIDS related mortality:

```
> deaths <- c(
+  'parms$mu*I0'
+  , 'parms$mu*I1'
+  , 'parms$mu*I2 + parms$gamma2 * I2'
+ )
> names(deaths) <- INFECTEDNAMES
```

Finally, we must write a vector of ODEs for state variables that do not correspond to demes in the coalescent model. In our example, there is only one such variable- the number of susceptibles:

```
> nonDemeDynamics <- paste(sep='',
+ '-parms$mu*S + parms$mu*(S + I0 + I1 + I2)'
+  ,'- S * (parms$beta0*I0+parms$beta1*I1+parms$beta2*I2) / (S + I0 + I1 + I2)'
+ )
> names(nonDemeDynamics) <- 'S'
```

Note well that in all cases, the expression or equation must have the corresponding name of the state variable.

The model can be fitted to a binary tree with dated tips. We will use a simulated tree such that we know the true parameter values and population size through time. We load this by:

```
> # read the tree
> tree <- read.tree(
+    system.file('extdata/hivSimulation.nwk', package='rcolgem'))
> #~ the sample times are the same, because it is a homochronous sample at 50 years
> sampleTimes <- rep(50, length(tree$tip.label))
> names(sampleTimes) <- tree$tip.label
> # create a tree with dated tips and internal nodes,
> # will infer the sample states from tip labels
> bdt <- binaryDatedTree(tree
+   , sampleTimes
+   , sampleStatesAnnotations=INFECTEDNAMES)
```

Note well that the vector of sample times must have the names of each taxon. If the state of each taxon (the stage of infection of each sample unit) is encoded at the end of taxon name, it can be loaded automatically by passing the argument `sampleStatesAnnotations`. Alternatively, that information can be passed as a matrix with row names corresponding to the tip labels of the tree.

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

```
> print(system.time( print(
+         coalescent.log.likelihood( bdt
+          , births,  deaths, nonDemeDynamics
+          , t0 = 0
+          , x0=c(I0=1, I1=.01, I2=.01, S = parms_truth$S_0)
+          , migrations = migrations
+          , parms=parms_truth
+          , fgyResolution=1000
+          , integrationMethod='euler'
+         )
+ )))

[1] -2214.567
   user  system elapsed
  1.568   0.000   1.570
```

It's a bit slow, so note that there is an alternative low-level interface that can be a bit faster (`coalescent.log.likelihood.fgy`). Also 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 by using the `bbmle` or `stats4` package.

```
> library(bbmle)
```

We will focus on estimating the three transmission rates of the system along with a nuisance parameter that controls initial conditions, t_0, which is the time of origin of the epidemic. We will assume prior knowledge of the stage progression rates γ_i mortality rates μ, and susceptible population size $S(0)$.

First, create the objective function to be minimized:

```
> obj_fun <- function(lnbeta0, lnbeta1, lnbeta2, t0)
+ {
+         parms <- parms_truth
+         parms$beta0 <- exp(lnbeta0)
+         parms$beta1 <- exp(lnbeta1)
+         parms$beta2 <- exp(lnbeta2)
+         mll <- -coalescent.log.likelihood( bdt
+                  , births, deaths, nonDemeDynamics
+                  , t0 = t0
+                  , x0=c(I0=1, I1=.01, I2=.01, S = parms$S_0)
+                  , migrations = migrations
+                  , parms=parms
+                  , fgyResolution = 1000
+                  , integrationMethod = 'rk4'
+         )
+         # track progress:
+         print(c(mll, exp(c(lnbeta0, lnbeta1, lnbeta2) ), t0) )
+         mll
+ }
```

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

We can then fit the model by running

```
> fit <- mle2(obj_fun
+   , start=list(lnbeta0=log(.6), lnbeta1=log(.2), lnbeta2=log(.05), t0=0)
+   , 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 about 10 minutes to run, so instead we will load the results:

```
> load( system.file('extdata/hivModel0-fit.RData', package='rcolgem') )
> AIC(fit)

[1] 4447.468

> logLik(fit)
```

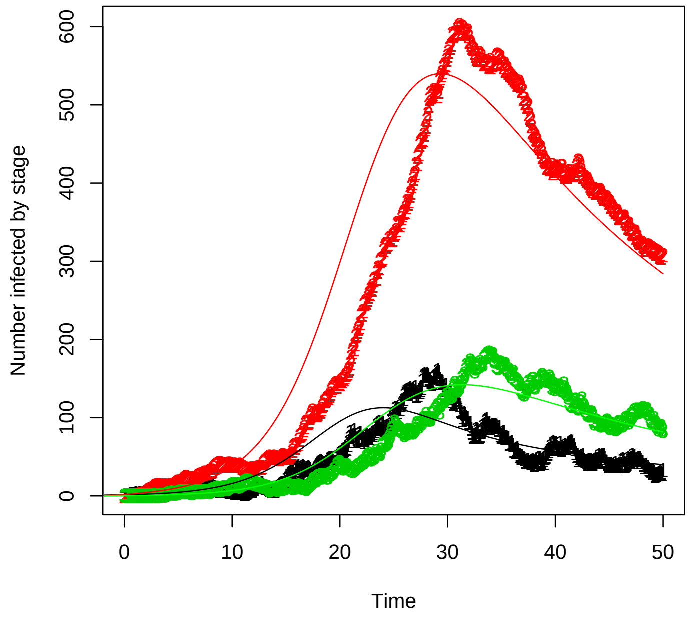

Figure 2: The actual (points) and estimated (lines) number of infections through time. Black: EHI, Red:chronic, Green:AIDS.

```
'log Lik.' -2219.734 (df=4)

> coef(fit)

   lnbeta0    lnbeta1    lnbeta2         t0
 0.1772788 -3.3862403 -2.9639203 -1.8339507

> exp(coef(fit))

   lnbeta0    lnbeta1    lnbeta2         t0
1.19396395 0.03383565 0.05161617 0.15978107
```

We can compare the fitted model to the true number of infected through time, which is shown in figure 2

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

```
> profbeta <- profile(fit, which=1, alpha=.05
+          , std.err=.5, trace=TRUE, tol.newmin=1 )
```

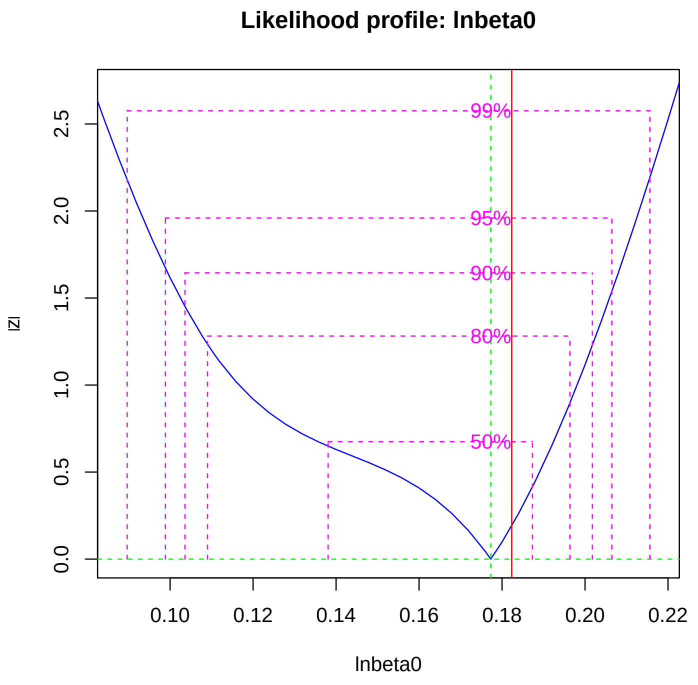

Figure 3: Likelihood profile for the transmission rate β_0 with confidence levels. The true parameter value is indicated by the vertical red line.

This takes a long time, so we will load the results:

```
> load( system.file('extdata/hivModel0-profbeta.RData',  package='rcolgem') )
```

We see that the confidence interval covers the true value:

```
> c( exp( confint( profbeta ) ), TrueVal=parms_truth$beta0 )
```

```
   2.5 %   97.5 %  TrueVal
1.093873 1.243095 1.200000
```

And, we can visualize the profile (Figure 3).

```
> plot(profbeta)
> abline( v = log( parms_truth$beta0) , col='red')
```

References

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