HIV Senegal Model

Fabricia F. Nascimento

2018-05-04

Introduction

This vignette will demonstrate how we estimated HIV transmission rates using DNA sequences from Senegal. This also provides a guidance on how analysis described in Nascimento *et al.* 2018 (In Prep.) was carried out. We analysed HIV-1 sequences from subtypes B, C and 02_AG.

Basic requirements

This vignette assumes that you know the basics of R and have the following packages already installed:

- ape: for phylogenetic trees
- akima: necessary for interpolation of data (used for the calculation of the likelihood by the phydynR package)
- BayesianTools: package for Bayesian inference
- devtools: useful for installing packages directly from github repository, for example.
- phydynR: implements the coalescent simulation and likelihood function for phylodynamics analysis
- treedater: fits a molecular clock to a phylogenetic tree.

Load the necessary packages:

```
library(ape)
library(akima)
library(BayesianTools)
library(phydynR)
library(treedater)
```

The Model

The model we fit is based on the structured coalescent models (Volz 2012). These models are used to estimate epidemiological parameters using a phylogenetic tree and information on states of each tip of the tree. These states are discrete-trait information representing each sequences.

In our mathematical model we have 4 different discrete-traits associated to each DNA sequence:

- gpf = HIV sample from the general population – females;
- gpm = HIV sample from the general population – males;
- msm = HIV sample from men that have sex with other men;
- src = source sample, which are HIV samples from individuals that are from other countries and not from Senegal.

Stage of infection

We fit the HIV epidemic in Senegal using ordinary differential equations (ODE) and only 1 stage of infection. This means that infected individuals would die and not recover from the infection. In our model we represented it as γ rate. We used 1 stage of infection, because the metadata available for the Senegal sequences did not have information that we could use to determine the stage of HIV infection at the time the samples were collected.

How transmissions were modelled?

- An infected msm (I_{msm}) could transmite to another msm with probability $p_{msm2msm}$
- An infected msm (I_{msm}) could transmit to a gpf with probability $(1 - p_{msm2msm})$
- An infected gpf (I_{gpf}) could transmit to a gpm with probability $p_{gpf2gpm}$
- An infected gpf (I_{gpf}) could transmit to a msm with probability $(1 - p_{gpf2gpm})$
- An infected gpm could also transmit to a gpf. For this event, we used the risk ratio of a male to transmite to a female, and fixed it to 2.0. This is the parameter $male_x$ of our model.

See Figure 1 for a schematic representation of the transmission model for HIV in Senegal. In this figure gpf, gpm and msm represent the infected individuals.

How about HIV incidence rate?

We also modelled the HIV incidence rate as a funtion of time (t) in msm and the gp (general population) as different spline functions (Eilers and Marx 1996), that in our ODEs are represented by $\lambda(t)$ and $\mu(t)$, respectively.

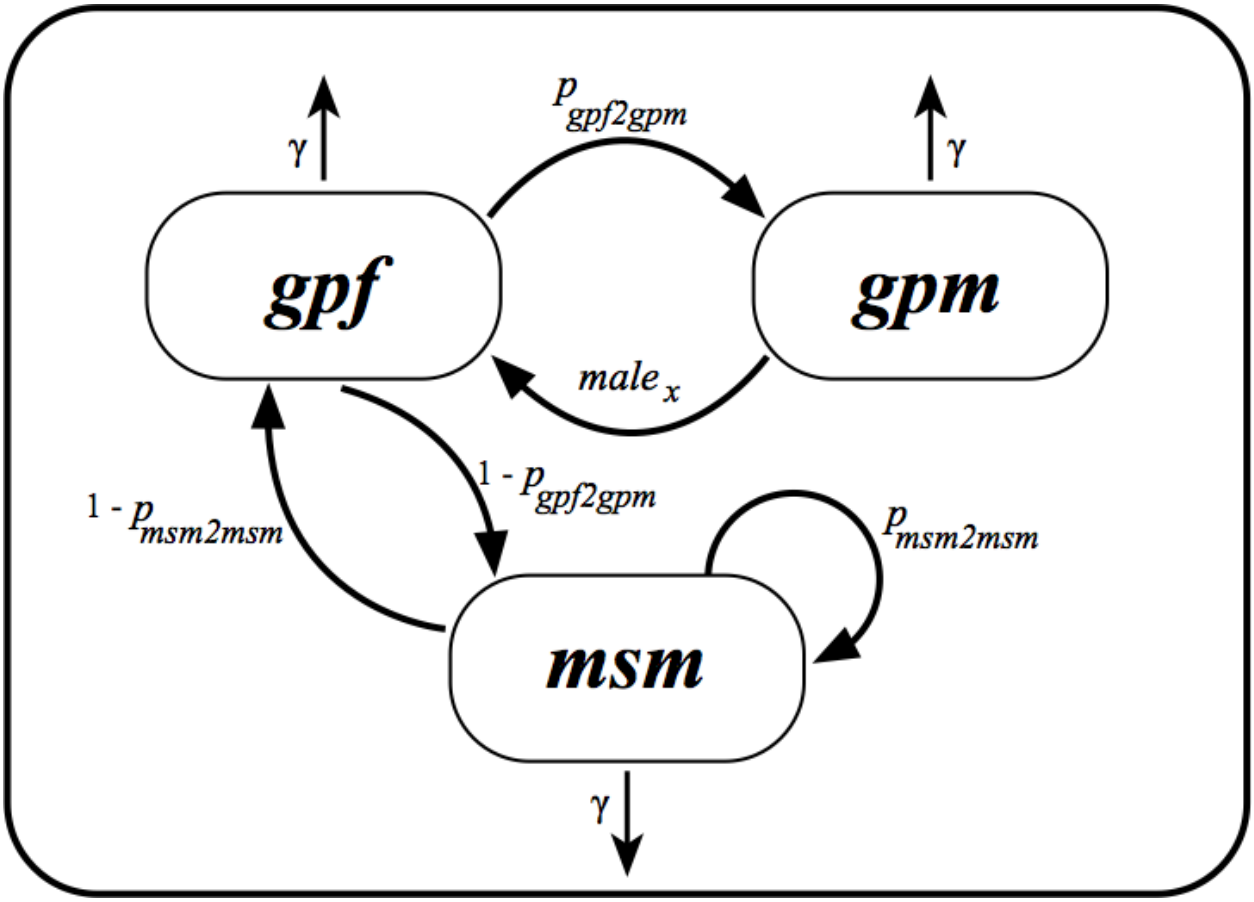

Figure 1: Transmission model for HIV in Senegal. gpf, gpm and msm represent infected individuals.

The *source* compartment

Finally, to model the HIV epidemic in Senegal, we also added an additional compartment named "source" (*src*), that represents the rate in which HIV lineages are imported to Senegal from other countries. We modelled this as a constant efective population size rate with two parameters to be estimeted – $srcNe$: the effective source population size; and the *import* rate. Because the number of imported HIV balances the number of exported HIV, the infected *src* individuals along time are not represented in the ODEs.

The ODEs or mathematical model equations

Based on **Figure 1** and the parameters we would like to estimate, the ordinary differential equations (ODEs) of our model are:

$$\dot{I}_{gpf} = male_x \mu(t) I_{gpm} + (1 - p_{msm2msm})\lambda(t) I_{msm} - \gamma I_{gpf}$$

$$\dot{I}_{gpm} = p_{gpf2gpm} \mu(t) I_{gpf} - \gamma I_{gpm}$$

$$\dot{I}_{msm} = (1 - p_{gpf2gpm})\mu(t) I_{gpf} + p_{msm2msm} \lambda(t) I_{msm} - \gamma I_{msm}$$

How to express the mathematical model in R?

In our model, we are interested in HIV transmissions in the *gp* and in the *msm* risk group. But also remember that we have to consider the imported HIV, which is in the *src* compartment. Based on that, *gpf*, *gpm*, *msm*, and *src* are the demes of our model, and are represented as a vector in R as:

```
demes <- c("gpm", "gpf", "msm", "src")
```

Because we use spline functions to determine the shape of the curve for the transmission rates in *gp* and *msm*, we should provide the initial `T0` and final `T1` times for our simulations.

```
T0 <- 1978
T1 <- 2014
```

We can also go ahead and set the value for the stage of infection, that in our model is just one stage. We assume we know this value.

```
GAMMA <- 1/10
```

We should also create a list to set up a template for the parameter values of our model. You can set this to values that you think are appropriate. Remember that the majority of parameter values in this list will be estimated. In R, this can be created following:

```
THETA <- list(gpsp0 = 6/10,
              gpsp1 = 4/10,
              gpsp2 = 1/10,
```

```
              gpsploc = 1987,
              msmsp0 = 4/10,
              msmsp1 = 4/10,
              msmsp2 = 2/10,
              msmsploc = 1995,
              maleX = 2.0,
              import = 1/20,
              srcNe = 1/10,
              gpspline = function(t, parms){
                if (t < T0) return(parms$gpsp0)
                if (t > T1) return (parms$gpsp2)
                with(parms, aspline(x = c(T0, gpsploc, T1),
                                    y=c(gpsp0, gpsp1, gpsp2),
                                    xout = t)$y)
              },
              msmspline  = function(t, parms){
                if (t < T0) return(parms$msmsp0)
                if (t > T1) return (parms$msmsp2)
                with(parms, aspline(x = c(T0, msmsploc, T1),
                                    y=c(msmsp0, msmsp1, msmsp2),
                                    xout = t)$y)
              },
             pmsm2msm = 0.85,
             pgpf2gpm = 0.85,
             initmsm = 1,
             initgp = 1)
```

Note that parameters *gpsp*0, *gpsp*1, *gpsp*2, and *gpsploc* are necessary to estimate the spline function for the *gp* (*gpspline* in R or $\mu(t)$ in the ODE). And parameters *msmsp*0, *msmsp*1, *msmsp*2, *msmsploc* are necessary to estimate the spline function for the *msm* risk group (*msmspline* in R or $\lambda(t)$ in the ODE).

We also need to setup some initial conditions of our model. We should set an arbitrary large number for the *src* population,

```
SRCSIZE <<- 1e5
X0 <- c(gpm = unname(THETA$initgp/2),
        gpf = unname(THETA$initgp/2),
        msm = unname(THETA$initmsm),
        src = SRCSIZE)
```

Setting up the birth, migration and death rates

The calculation of births and migrations of our model are expressed as 4×4 matrices, which represent a transmission or movement from one deme to another deme. Lineages also die at

Table 1: Parameter definition, values and symbols used in the R script

Parameter	Symbol in R	Initial values
Spline shape gp0	gpsp0	6/10
Spline shape gp1	gpsp1	4/10
Spline shape gp2	gpsp2	1/10
Spline interval gp	gpsploc	1987
Spline shape msm0	msmsp0	4/10
Spline shape msm1	msmsp1	4/10
Spline shape msm2	msmsp2	2/10
Spline interval msm	msmsploc	1995
Infectiouness ratio from male to female	maleX	2.0
Importation rate	import	1.20
Effective population size of src	srcNe	1.10
Probability of infected msm to infect another msm	pmsm2msm	0.85
Probability of infected gpf to infect a gpm	pgpf2gpm	0.85
Initial number of infected msm	initmsm	1.0
Initial number of infected gp	initgp	1.0

Table 2: Illustration of allowed transmissions between demes.

	gpm	gpf	msm	src
gpm	0.	$gpm \rightarrow gpf$	0.	0.
gpf	$gpf \rightarrow gpm$	0.	$gpf \rightarrow msm$	0.
msm	0.	$msm \rightarrow gpf$	$msm \rightarrow msm$	0.
src	0.	0.	0.	$src \rightarrow src$

a same rate.

First, we have to setup the components of the model. This can be done using the `setup.model.equations` function in this research compendium.

```
eqns <- senegalHIVmodel::setup.model.equations(demes)
attach(eqns)
```

Setting up the birth matrix

We should set up the birth matrix to allow transmissions from one deme to another deme as in **Figure 1**.

To set up these transmissions between demes, each element in the matrix is a string that will be passed as R code.

```
births['msm', 'msm'] <- "parms$msmspline(t, parms) * msm * parms$pmsm2msm"
births['msm', 'gpf'] <- "parms$msmspline(t, parms) * msm * (1-parms$pmsm2msm)"

births['gpm', 'gpf'] <- "parms$gpspline(t, parms) * gpm * parms$maleX"
births['gpf', 'gpm'] <- "parms$gpspline(t, parms) * gpf * parms$pgpf2gpm"
births['gpf', 'msm'] <- "parms$gpspline(t, parms) * gpf * (1-parms$pgpf2gpm)"

# f = (1/2)*(Y^2)/Ne
births['src', 'src'] <- "0.5 * SRCSIZE^2 / parms$srcNe"
```

Setting up the migration matrix

Similar to the birth matrix, we also allow migrations from gpf, gpm, or msm to the src; and from src to gpf, gpm, or msm. This is modelled as a constant effective population size.

Table 3: Illustration of allowed migrations between demes.

	gpm	gpf	msm	src
gpm	0.	0.	0.	$gpm \rightarrow src$
gpf	0.	0.	0.	$gpf \rightarrow src$
msm	0.	0.	0.	$msm \rightarrow src$
src	$src \rightarrow gpm$	$src \rightarrow gpf$	$src \rightarrow msm$	0.

We also set up migrations between demes where each element in the matrix is a string that will be passed as R code.

```
migs['src', 'gpm'] <- "parms$import * gpm"
migs['src', 'gpf'] <- "parms$import * gpf"
migs['src', 'msm'] <- "parms$import * msm"

migs['gpm', 'src'] <- "parms$import * gpm"
migs['gpf', 'src'] <- "parms$import * gpf"
migs['msm', 'src'] <- "parms$import * msm"
```

Setting up the vector for the death rates

Similarly to the birth and migration matrices, we also set up death rates where each element is a string that will be passed as R code.

```
deaths['msm'] <- "GAMMA * msm"
deaths['gpf'] <- "GAMMA * gpf"
deaths['gpm'] <- "GAMMA * gpm"
deaths['src'] <- "0.5 * SRCSIZE^2 / parms$srcNe"
```

The demographic model

After setting up all the components of the mathematical model, we can build the demographic process using the function `build.demographic.process` from the `phydynR` pachage. The `dm` output can be used as imput to coalescent models for the calculation of the likelihood, when fitting the model using a Markov chain Monte Carlo (MCMC), for example.

```
dm <- build.demographic.process(births = births,
                                deaths = deaths,
                                migrations = migs,
                                parameterNames = names(THETA),
                                rcpp = FALSE,
                                sde = FALSE)
```

For more information on the the input data for the `build.demographic.process` funtion, you should see its R documentaion.

Load additional data

After setting up all the equations of the model in a way that R can understand, the model can now be fitted to a binary and dated phylogenetic tree. In our specific case, we estimated a phylogenetic tree using RAxML-NG for each subtype, as we are analysing HIV-1 subtypes B, C and 02_AG.

Each tree had a relaxed clock fitted using the R package treedater. And after that, the trees were merged into a single tree using the R script *merge_trees.R* (it can be found in the directory `analysis/scripts`). The final tree did not contain sequences from children, and from which risk group or sex was NA.

We load the binary dated tree in R using the following:

```
tree.all <- read.tree(
  system.file("data/bindTree_CGR_GTR+Gp12+3_droppedTip.tre",
  package = "senegalHIVmodel"))
```

After, we should load all the information for the metadata. This aims to create a discrete-trait data for all tips in the phylogenetic tree. Remember that our discrete-trait data in this model are the *gpf*, *gpm*, *msm*, and *src*.

First we need to read in R all metadata as below:

```
# Reads metadata for the CGR (close global reference) sequences
# CGRs are referred in the mathematical model as src (source) data
# src are HIV sequences that are from other countries and not from Senegal
```

```
all.data.cgr <- read.csv(
  system.file("data/HIV_subtypes_summary_CGR.csv",
  package = "senegalHIVmodel"))

# Reads all metadata for HIV sequences from Senegal
all.data.SN <- read.csv(
  system.file("data/HIV_subtypes_summary_SENEGAL_noDups.csv",
  package = "senegalHIVmodel"))

# organize metadata in 2 columns.
# the 1st column is the sequence names
# the 2nd colum is the state (gpf, gpm, msm, or src) of each sequence
# metadata from children, or risk group or sex = NA will also be removed.
all_data <- senegalHIVmodel::organize_metadata(all.data.cgr, all.data.SN)
```

After organing all metadata, we further organize it in a way that the R package `phydynR` will understand it. For that, we create a matrix to receive the information on states (discrete-traits) for each tip of the tree

```
gpm <- gpf <- msm <- src <- rep(0, length(tree.all$tip.label))

# Adds 1 to where states matches "gpm", and so on.
gpm[all_data$States == "gpm"] <- 1
gpf[all_data$States == "gpf"] <- 1
msm[all_data$States == "msm"] <- 1
src[all_data$States == "src"] <- 1

sampleStates <- cbind(gpm, gpf, msm, src)
rownames(sampleStates) <- all_data$tip.name
```

Now, we need to read the estimated times (in calendar units) for each sequence in the phylogenetic tree

```
times <- readRDS(
  system.file("data/bindTree_CGR_GTR+Gp12+3_droppedTip_sts.RDS",
  package = "senegalHIVmodel"))
```

Finaly, we create an an object of `DatedTree` [phydynR package]. This is the tree that should be used in the calculation of the likelihood to estimate parameter values using `phydynR`

```
dated.tree <- phydynR::DatedTree(phylo = tree.all,
                                 sampleTimes = times,
                                 sampleStates = sampleStates,
                                 minEdgeLength = 2/52,
                                 tol = 0.1)
```

Calculation of the likelihood

After setting up the mathematical model and the data, we have the following:

- `dated.tree` = a phylogenetic tree of class DatedTree
- `THETA` = template for parameter values
- `dm` = the demographic process
- `X0` = initial conditions

The above objects will be used in the calculation of the likelihoood using the `colik` function in the phydynR package.

```
phydynR::colik(tree = dated.tree,
               theta = THETA,
               demographic.process.model = dm,
               x0 = X0,
               t0 = 1978,
               res = 1e3,
               timeOfOriginBoundaryCondition = FALSE,
               AgtY_penalty = 1,
               maxHeight = 41)
```

Note that for the calculation of the likelihood you can provide a value for maximum height `maxHeight`. This parameter "tells" the function up to which point back in time the calculation of the likelihood should be done. If computer resources are not a problem, you can leave this as the default. Using the whole tree for the calculation of the likelihood can make it slower than when using just a portion back in time.

For the Senegal data, we used a `maxHeight` = 41. This means that it will go back in time in the tree at approximatelly 1973. This was merely chosen to put the HIV epidemics in Senegal around this time, for estimation of the parameters in our model.

Estimation of epidemiological parameters

Now that we are happy with everything, we are ready to estimate the parameters of our model. For the Senegal HIV model, we chose to estimate the parameters using a Markov chain Monte Carlo (MCMC) as implemented in the R package BayesianTools.

Which parameters to estimate?

For this example, we decided to estimate the following parameters:

Paramters for estimating the spline function for the *gp*:

- *gpsp0*

- *gpsp1*
- *gpsp2*
- *gpsploc*

Parameters for estimating the spline function for the *msm*:

- *msmsp0*
- *msmsp1*
- *msmsp2*
- *msmsploc*

Parameters that controls the *src*:

- *import*
- *srcNe*

Probability of certain events to occour:

- *pmsm2msm*
- *pgpf2gpm*

To estimate these parameters we should set up an object function. This object function will receive the proposals of the MCMC. The reason of using an object function is to make it easier to change the values of the parameters to be estimated in THETA (our parameter template as explained before). Note that not all parameters listed in THETA will be estimated.

```
obj_fun <- function(parameters){
  # we use unname here because "parameters" can be a vector or matrix, and
  # sometimes it comes with column names, which I chose to remove these names
  # in here.
  parameters <- unname(parameters)

  # add the values of THETA to a new variable named THETA.new
  THETA.new <- THETA

  # change the values in THETA.new to the new proposals that will be evaluated
  THETA.new$gpsp0 <- parameters[1]
  THETA.new$gpsp1 <- parameters[2]
  THETA.new$gpsp2 <- parameters[3]
  THETA.new$gpsploc <- parameters[4]
  THETA.new$msmsp0 <- parameters[5]
  THETA.new$msmsp1 <- parameters[6]
  THETA.new$msmsp2 <- parameters[7]
  THETA.new$msmsploc <- parameters[8]
  THETA.new$import <- parameters[9]
  THETA.new$srcNe <- parameters[10]
  THETA.new$pmsm2msm <- parameters[11]
  THETA.new$pgpf2gpm <- parameters[12]
```

```
  # After changing the parameter values to the new proposals, a likelihood is
  # calculated with the funtion colik.
  # Note that this function uses several global variables, such as,
  # dated.tree, dm, and X0
  mll <- phydyn::colik(tree = dated.tree,
                theta = THETA.new,
                demographic.process.model = dm,
                x0 = X0,
                t0 = 1978,
                res = 1e3,
                timeOfOriginBoundaryCondition = FALSE,
                AgtY_penalty = 1,
                maxHeight = 41)

  return(mll)

}
```

We can now estimate these parameters using the MCMC. For that we need to decide the priors – our best knowledge on the parameter value before the data is analysed – on each parameter that will be estimated.

Because we are using the BayesianTools R package, we need to specify the density function (that represent our prior) for each parameter as below:

```
# Specify a density function to be used in the
# prior especification (see below)
densities <-  function(par){
  # d1 to d3 and d5 to d7 I am using a gamma distribution with
  #    mean = R0 = 1.1 and sigma = 1
  # d4, d8 uniform distribution between the start time and
  # the final time of our simualtions (t0 = 1978, and t1 = 2014)
  # d9 exponential distribution with mean around 1/30
  # d10 exponential distribution with mean around 1/20
  d1 = dgamma(par[1], shape = 3, rate = 3/1.1, log = TRUE) #gpsp0
  d2 = dgamma(par[2], shape = 3, rate = 3/1.1, log = TRUE) #gpsp1
  d3 = dgamma(par[3], shape = 3, rate = 3/1.1, log = TRUE) #gpsp2
  d4 = dunif(par[4], min = 1978, max = 2014, log = TRUE) #gpsploc
  d5 = dgamma(par[5], shape = 3, rate = 3/1.1, log = TRUE) #msmsp0
  d6 = dgamma(par[6], shape = 3, rate = 3/1.1, log = TRUE) #msmsp1
  d7 = dgamma(par[7], shape = 3, rate = 3/1.1, log = TRUE) #msmsp2
  d8 = dunif(par[8], min = 1978, max = 2014, log = TRUE) #msmsploc
  d9 = dexp(par[9], rate = 30, log = TRUE) #import
  d10 = dexp(par[10], rate = 20, log = TRUE) #srcNe
  d11 = dbeta(par[11], shape1 = 16, shape2 = 4, log = TRUE) #pmsm2msm
```

```
  d12 = dbeta(par[12], shape1 = 16, shape2 = 4, log = TRUE) #pgpf2gpm

  return(d1 + d2 + d3 + d4 + d5 + d6 + d7 + d8 + d9 + d10 + d11 + d12)
}
```

Now, we can provide a sampler, which is optional as described in the BayesianTools package, as below:

```
# Create sampling, this is optional.
#The MCMCs can automatically generate starting
# conditions if a sampler is provided
sampler <-  function(n=1){
  d1 = rgamma(n, shape = 3, rate = 3/1.1) #gpsp0
  d2 = rgamma(n, shape = 3, rate = 3/1.1) #gpsp1
  d3 = rgamma(n, shape = 3, rate = 3/1.1) #gpsp2
  d4 = runif(n, min = 1978, max = 2014) #gpsploc
  d5 = rgamma(n, shape = 3, rate = 3/1.1) #msmsp0
  d6 = rgamma(n, shape = 3, rate = 3/1.1) #msmsp1
  d7 = rgamma(n, shape = 3, rate = 3/1.1) #msmsp2
  d8 = runif(n, min = 1978, max = 2014) #msmsploc
  d9 = rexp(n, rate = 30) #import
  d10 = rexp(n, rate = 20) #srcNe
  d11 = rbeta(n, shape1 = 16, shape2 = 4) #pmsm2msm
  d12 = rbeta(n, shape1 = 16, shape2 = 4) #pgpf2gpm

  return(cbind(d1, d2, d3, d4, d5, d6, d7, d8, d9, d10, d11, d12))
}
```

After setting up the densities and sampler funtions we can now set up our prior by using the following:

```
# Create prior (necessary for the BayesianTools package)
prior <- createPrior(density = densities,
                     sampler = sampler,
                     lower = c(0.01, 0.01, 0.01,
                               1978, 0.01, 0.01,
                               0.01, 1978, 0,
                               0.0001, 0.3, 0.3),
                     upper = c(3, 3, 3,
                               2014, 3, 3,
                               3, 2014, 0.15,
                               0.15, 1, 1))
```

We now ready to estimating the parameter values using the MCMC. We chose the DEzs sampler. DEzs stands for Differential-Evolution MCMC zs and this is beyoud what will cover in this vingnette. However, if you can check Ter Braak and Vrugt (2008) for more

information.

```
settings = list(iterations = 18000, nrChains = 1, thin = 1)
out <- runMCMC(bayesianSetup = bayesianSetup,
               sampler = "DEzs",
               settings = settings)
```

In our paper, we have also extended this analysis by providing a z-matrix. First we run a MCMC as above, and after having an ok run, we used that run to provided a z-matrix as explained here. For details on how this was done, you can see our script *mcmc.R* in the directory `analysis/scripts`.

References

Eilers, Paul H. C., and Brian D. Marx. 1996. "Flexible smoothing with B-splines and penalties." *Statistical Science* 11 (2):89–121. https://doi.org/10.1214/ss/1038425655.

Ter Braak, Cajo J.F., and Jasper A. Vrugt. 2008. "Differential Evolution Markov Chain with snooker updater and fewer chains." *Statistics and Computing* 18 (4):435–46. https://doi.org/10.1007/s11222-008-9104-9.

Volz, Erik M. 2012. "Complex population dynamics and the coalescent under neutrality." *Genetics* 190 (1):187–201. https://doi.org/10.1534/genetics.111.134627.