ThetaMater: Rapid and scalable Bayesian estimation of theta from genomic data

Rich Adams, Drew Schield, Daren Card, Andrew Corbin, and Todd Castoe

last updated: 2017-05-08

Table of Contents

## The population size parameter

The population size parameter reflects the effects of genetic drift and mutation on patterns of genetic variation within a diploid population ( for a haploid population) with an effective size of individuals and a mutation rate o f per site per generation. If two homologous sequences are sampled at random from a population, describes the expected number of mutations between these two sequences. is a fundamental measure of genetic diversity in populations and is thus an informative parameter used in many population genetic models. The R package ThetaMater provides a Bayesian framework to estimate both and (shape of among-locus rate variation) parameters from a variety of genetic datasets, including haploid or diploid genomic data from single or multiple individuals, reduced-representation genomic data (e.g., RADseq, sequence capture), and single or multilocus Sanger sequence data (and variations of these datasets). ThetaMater implements three different functions that can be used to estimate these parameters within a Bayesian framework:

The population size parameter reflects the effects of genetic drift and mutation on patterns of genetic variation within a population with an effective size of individuals and a mutation rate of per site per generation. If two homologous sequences are sampled at random from a population, describes the expected number of mutations between these two sequences. is a fundamental parameter of many population genetic models and is a fundamental measure of genetic diversity within a population. The R package ThetaMater provides a Bayesian framework to estimate both and (shape of among-locus rate variation)from a variety of genetic datasets, including single or multiple genomes (haploid or diploid), RADseq, sequence capture, and single or multilocus sanger sequence data (and variations of these datasets). ThetaMater implements three different functions that can be used to estimate these parameters within a Bayesian framework:  
\* ThetaMater.M1: estimate without among-locus variation  
\* ThetaMater.M2: estimate with a fixed parameter of rate variation and a user-defined number of rate classes  
\* ThetaMater.M3: estimate both and the shape parameter and a user-defined number of rate classes

## The likelihood function implemented by ThetaMater: Probability of observing mutations given a locus of length , a sample size of individuals, and a vaue

The three functions (ThetaMater.M1, ThetaMater.M2, ThetaMater.M3) simulate posterior distributions of and/or parameters for a given dataset. These functions employ the likelihood function of XXX et. al 1995 to compute the probability of observing k mutations in a sample size of n from a locus with length l. These methods compute the likelihood of a given dataset as a summation of the log-likelihood probabilities across all loci, each with respective lengths, mutation counts, and sample sizes.For detailed information on the infinite sites model of mutation that is implemented by these functions. See the following papers and books for more information about this model, its derivation, applications, and other similar models:  
\* Watterson,G.A. (1975) On the number of segregating sites in genetical models without recombination. Theor. Popul. Biol.  
\* Wakeley, John. "Coalescent theory." Roberts & Company (2009).  
\* Hein, Jotun, Mikkel Schierup, and Carsten Wiuf. Gene genealogies, variation and evolution: a primer in coalescent theory. Oxford University Press, USA, 2004.  
\* Takahata, Naoyuki, and Yoko Satta. "Evolution of the primate lineage leading to modern humans: phylogenetic and demographic inferences from DNA sequences." Proceedings of the National Academy of Sciences 94.9 (1997): 4811-4815.  
\* Takahata, Naoyuki, Yoko Satta, and Jan Klein. "Divergence time and population size in the lineage leading to modern humans." Theoretical population biology 48.2 (1995): 198-221. \* Tavaré, Simon. "Line-of-descent and genealogical processes, and their applications in population genetics models." Theoretical population biology 26.2 (1984): 119-164.  
\* Yang, Ziheng. "On the estimation of ancestral population sizes of modern humans." Genetical research 69.02 (1997): 111-116.

Below is the formula for the likelihood function described in this papers that is at the heart of ThetaMater:

For a dataset consist of loci, each with mutation count , number of bases , and number of sequences sampled , we can sum the likelihoods of the individual loci to get the likelihood of the entire dataset under a given value of :

As estimates from any one locus entails significant uncertainity, ThetaMater allows researchers to take full advantage of large, diverse datasets when estimating and providing a distribution of plausible values while accounting for uncertainty.

## Step 1: install R package ThetaMater from github

The R package ThetaMater is freely available to download and distribute from github <https://github.com/radamsRHA/ThetaMater/>. To install and load ThetaMater, you must first install the R packages devtools, MCMCpack, phangornand ape.

# download dependencies  
install.packages("devtools")  
install.packages("MCMCpack")  
install.packages("ape")  
install.packages("phangorn")

Now using devtools we can install ThetaMater from github:

library(devtools)  
install\_github("radamsRHA/ThetaMater")

Next, load the dependency packages for ThetaMater into the R working environment with the following code:

library(ThetaMater) # Load package  
library(MCMCpack) # Load dependency phybase  
library(ape) # Load dependency ape  
library(phangorn) # Load dependency phangorn

## Step 2: Functions to read input data formats and convert into infinite-sites data used by ThetaMater

ThetaMater currently includes a set of 5 functions (more to come) to read in the following data formats and convert these into the infinite sites format used by ThetaMater:  
\* Fasta alignments: a directory containing a set of fasta alignments  
\* Nexus alignments: a directory containing a set of nexus alignments  
\* pYRAD output alignments: a single, multilocus *alleles file produced by the pyRAD pipeline*  
 Interleaved fasta alignments: a single, multilocus fasta file comprising multiple independent loci (i.e., similar to stacks output)  
\* Diploid genome fasta alignments: a single fasta file representing a diploid sequence alignment in which SNPs are coded as ambiquities

Please contact Rich Adams ([radams@uta.edu](mailto:radams@uta.edu)) to request additional formats that are not currently supported (usually a short example file will be used to build a custom function) The format for the input conversion functions arguments are as follows:

Read.FastaDir(fasta.dir)  
\* fasta.dir: path to the directory of fasta alignments, each with a suffix of .fasta or .fa

Read.NexusDir(nexus.dir)  
\* nexus.dir: path to the directory of nexus alignments, each with a suffix of .nexus or .nex

Read.AllelesFile(alleles.file)  
\* alleles.file: path to the .alleles file provided by the pyRAD pipeline

Read.InterleavedFasta(fasta.file)  
\* fasta.file: path to an 'interleaved' fasta file (i.e., stacks output)

Read.DiploidFasta(genome.fasta.file)  
\* genome.fata.file: path to a diploid genome fasta alignment (ambiquities code for SNPs)

Below we read one of the example datasets used in this tutorial:

# Load the example data provided with the package  
data(example.dat,package= "ThetaMater")  
  
# Let's look at the data  
example.dat$k.vec # mutation counts

## [1] 0 1 2 3 4 0 1 2 3 6

example.dat$l.vec # locus lengths

## [1] 100 100 100 100 100 100 100 100 100 100

example.dat$n.vec # number of samples

## [1] 5 5 5 5 5 7 7 7 7 7

example.dat$c.vec # number of observations

## [1] 350 121 27 12 2 318 126 38 5 1

library(ThetaMater)  
# To use the function 'Read.AllelesFile' you can try loading the raw file included with this package  
file.loc <- system.file("example.alleles", package="ThetaMater")  
example.dat <- Read.AllelesFile(alleles.file = file.loc)

## [1] "Reading /Library/Frameworks/R.framework/Versions/3.3/Resources/library/ThetaMater/example.alleles for all 1000 loci..."  
## [1] "ALL DONE! returning a list of k.vec, n.vec, l.vec, locus.nums for 1000 loci"

Here, the object 'example.dat' returns a list with the following vectors: \* k.vec = vector of mutation counts \* l.vec = vector of locus lengths \* n.vec = vector of sample counts \* c.vec = vector of unique pattern counts

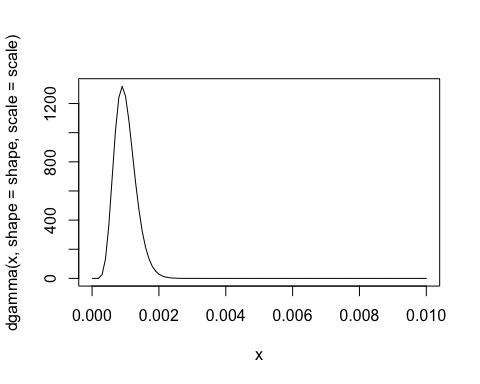
## Step 3: Setting the prior distributions for and/or

ThetaMater uses gamma distributions to model the prior probabilty distributions for both and . The prior gamma distribution for and are described by two parameters (shape and scale, with expectation[parameter] = shape\*scale). These parameters should be set to reflect prior knowledge about and before analyzing the observed dataset. In practice, we find that most theta values are within the range 0.00001-0.01. You can use the below code to view the gamma distribution prior to running ThetaMater:

# Lets see some gamma distribution settings for theta.   
  
# Here we have a relatively peaked prior with the e expectation (shape \* scale) = 0.001  
shape = 10  
scale = 0.0001  
  
# See expected theta  
E.theta = shape\*scale   
E.theta

## [1] 0.001

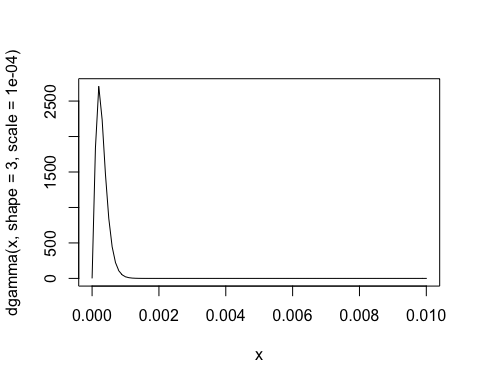
# Now let's plot this distribution  
curve( dgamma(x,shape = shape,scale = scale), xlim=c(0,.01) )



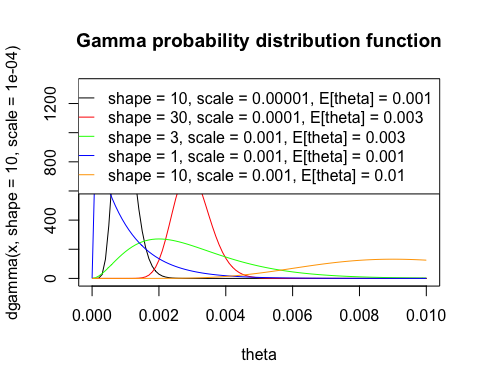
# Here's a prior for ~order of magnitude smaller than before  
shape = 3  
scale = 0.0001  
  
# See expected theta  
E.theta = shape\*scale   
E.theta

## [1] 3e-04

curve( dgamma(x,shape = 3,scale = 0.0001), xlim=c(0,.01) )



# Finally, here's an array of different settings for the prior  
curve( dgamma(x,shape = 10,scale = 0.0001), xlim=c(0,0.01), xlab = "theta")  
curve( dgamma(x,shape = 30, scale =0.0001), add=T, col='red' )  
curve( dgamma(x,shape = 3,scale = 0.001), add=T, col='green' )  
curve( dgamma(x,shape = 1,scale = 0.001), add=T, col='blue' )  
curve( dgamma(x,shape = 10, scale = 0.001), add=T, col='orange' )  
title(main="Gamma probability distribution function")  
legend(par('usr')[2], par('usr')[4], xjust=1,  
 c('shape = 10, scale = 0.00001, E[theta] = 0.001', 'shape = 30, scale = 0.0001, E[theta] = 0.003',   
 'shape = 3, scale = 0.001, E[theta] = 0.003', 'shape = 1, scale = 0.001, E[theta] = 0.001', 'shape = 10, scale = 0.001, E[theta] = 0.01'),  
 lwd=1, lty=1,  
 col=c(par('fg'), 'red', 'green', 'blue', 'orange') )



The shape and scale parameters can be set for in a similar manner to reflect prior knowledge about the distributon of among-locus rate variation in your dataset.

***IMPORTANT: Setting the prior distribution is critical for ThetaMater analyses. This prior distribution is designed to reflect prior knowledge about the parameters before viewing the dataset. It is often recommended to try a set of prior values to determine the sensitivty of the posterior to the prior and to evaluate the results under different settings***

## Step 4.1: ThetaMater.M1: function to simulate a posterior distribution of theta without among-locus rate variation

Here we will estimate Theta for the given dataset using ThetaMater.M1. The input arguments for the function ThetaMater.M1 are as follows:

ThetaMater.M1(k.vec, l.vec, n.vec, c.vec, ngens, burnin, thin, theta.shape, theta.scale)  
\* k.vec: vector of mutation counts  
\* l.vec: vector of locus lengths  
\* n.vec: vector of sample counts  
\* c.vec: vector of unique pattern counts  
\* ngens: number of generations to run the MCMC simulation  
\* burnin: number of generations to discard as burnin  
\* thin: number of generations between recorded MCMC samples  
\* theta.shape: shape parameter of the prior gamma distribution on theta (See Step 2)  
\* theta.scale: scale parameter of the prior gamma distribution on theta (See Step 2)

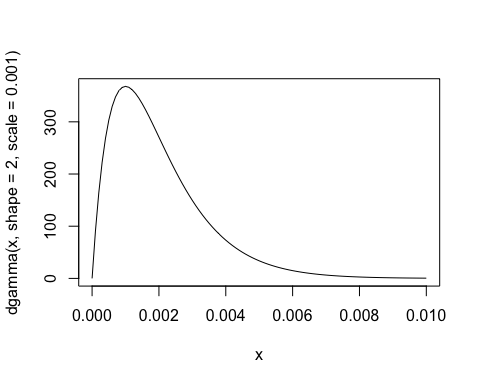
Before estimating theta with ThetaMater.M1, let's first view and set the prior distribution for theta that we will use in this example analysis. These data were simulated using theta = 0.002, and thus we set a prior distribution with an expectation of 0.002 for this example population. For empirical analyses, the values of these parameters (theta.shape, theta.scale) will be ideally set to reflect prior knowledge about a given population under study. For this analysis will set theta.shape and theta.scale as the following:

shape = 2  
scale = 0.001  
E.theta = shape\*scale   
E.theta

## [1] 0.002

Let's go ahead and plot this distribution to visualize our prior (See Step 3 for more information on setting priors):

curve( dgamma(x,shape = 2,scale = 0.001), xlim=c(0,.01) )



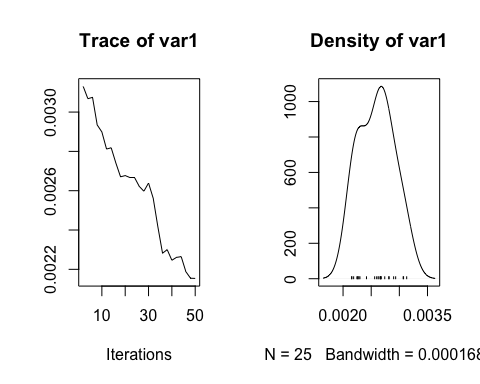
Now we are ready to estimate We can load the example data and run the MCMC for ThetaMater.M1 using the code below:

data(example.dat,package= "ThetaMater")  
  
example.MCMC <- ThetaMater.M1(k.vec = example.dat$k.vec, l.vec = example.dat$l.vec, n.vec = example.dat$n.vec, c.vec = example.dat$c.vec, ngens = 50, burnin = 1, theta.shape = shape, theta.scale = scale, thin = 2)

## MCMCmetrop1R iteration 1 of 51   
## function value = -934.35152  
## theta =   
## 0.00321  
## Metropolis acceptance rate = 0.00000  
##   
##   
##   
## @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@  
## The Metropolis acceptance rate was 0.66667  
## @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@

We didn't run the mcmc very long, and thus the MCMC output does not appear at stationarity. See trace and density plot of the posterior below:

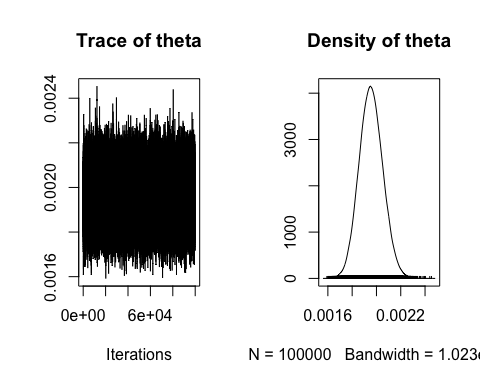
plot(example.MCMC)



Let's look at the results from a much longer run that was executed using the below command:

example.MCMC <- ThetaMater.M1(k.vec = example.dat$k.vec, l.vec = example.dat$l.vec, n.vec = example.dat$n.vec, c.vec = example.dat$c.vec,ngens = 1000000, burnin = 1000, thin = 10, theta.shape = shape, theta.scale = scale)

file.loc <- system.file("example.MCMC.M1.csv", package="ThetaMater")  
plot(as.mcmc(read.csv(file = file.loc)))



As you can see from the trace and density plot, the MCMC has reached stationarity (represented by the classic "fuzzy caterpillar" shape). Also, we can look at the mean and variance of the posterior distribution of using this code below:

file.loc <- system.file("example.MCMC.M1.csv", package="ThetaMater")  
mean(as.mcmc(read.csv(file = file.loc))) # close to the simulated value of 0.002

## [1] 0.00195588

sd(as.mcmc(read.csv(file = file.loc)))

## [1] 9.646917e-05

summary(as.mcmc(read.csv(file = file.loc)))

##   
## Iterations = 1:1e+05  
## Thinning interval = 1   
## Number of chains = 1   
## Sample size per chain = 1e+05   
##   
## 1. Empirical mean and standard deviation for each variable,  
## plus standard error of the mean:  
##   
## Mean SD Naive SE Time-series SE   
## 1.956e-03 9.647e-05 3.051e-07 4.375e-07   
##   
## 2. Quantiles for each variable:  
##   
## 2.5% 25% 50% 75% 97.5%   
## 0.001772 0.001890 0.001954 0.002020 0.002150

***IMPORTANT: Don't panic if you see this error when running ThetaMater: "initial value in 'vmmin' is not finite". This just means that the likelihood of the data is very small under the current prior settings and thus 'infinite' likelihood values may arise. Thus, the likelihood function used by ThetaMater may react pooly to badly specified prior values for . If you see this error when running ThetaMater, try different prior settings and multiple runs. See below commands for a demonstration***

# Here's a poorly specified prior that is far from the true value (this will give the error)  
example.MCMC <- ThetaMater.M1(k.vec = example.dat$k.vec, l.vec = example.dat$l.vec, n.vec = example.dat$n.vec, c.vec = example.dat$c.vec,ngens = 500, burnin = 1, thin = 1, theta.shape = 10, theta.scale = 10)

## Error in optim(theta.init.0, maxfun, control = optim.control, lower = optim.lower, : initial value in 'vmmin' is not finite

# Let's try another prior setting that is closer to the true value of theta  
example.MCMC <- ThetaMater.M1(k.vec = example.dat$k.vec, l.vec = example.dat$l.vec, n.vec = example.dat$n.vec, c.vec = example.dat$c.vec,ngens = 500, burnin = 1, thin = 1, theta.shape = 1, theta.scale = 1)

## MCMCmetrop1R iteration 1 of 501   
## function value = -19582.01378  
## theta =   
## 1.50501  
## Metropolis acceptance rate = 1.00000  
##   
## MCMCmetrop1R iteration 501 of 501   
## function value = -19578.89136  
## theta =   
## 1.50405  
## Metropolis acceptance rate = 0.92615  
##   
##   
##   
## @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@  
## The Metropolis acceptance rate was 0.92615  
## @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@

It works! No error this time. With better prior settings that are close to the true value, this likelihood function will not misbehave and the error will not occur. As always, contact the author Rich Adams ([radams@uta.edu](mailto:radams@uta.edu)) if you have any questions and/or receive this error message.

## Step 4.2: ThetaMater.M2: function to simulate a posterior distribution of with a fixed shape parameter of among-locus rate variation

Here we will estimate the posterior distribution of using a fixed parameter describing the distribution of among-locus rate variation within our dataset. The input arguments for the function ThetaMater.M2 are as follows:

ThetaMater.M2(k.vec, l.vec, n.vec, c.vec, alpha, K.classes, ngens, burnin, thin, theta.shape, theta.scale) \* k.vec: vector of mutation counts  
\* l.vec: vector of locus lengths  
\* n.vec: vector of sample counts  
\* c.vec: vector of unique pattern counts  
\* ngens: number of generations to run the MCMC simulation  
\* alpha: fixed alpha parameter describing the shape of the distribution of among-locus rate variation  
\* K.classes: number of disctinct classes to approximate the gamma distribution (4-20 are commonly used for datasets) \* burnin: number of generations to discard as burnin  
\* thin: number of generations between recorded MCMC samples  
\* theta.shape: shape parameter of the prior gamma distribution on (See Step 2)  
\* theta.scale: scale parameter of the prior gamma distribution on (See Step 2)

The following example data were simulated using = 0.002 and = 0.1 (using 4 rate classes to approximate the gamma distribution)

data(example.M2.dat, package= "ThetaMater")  
# Let's look at the data  
example.M2.dat$k.vec # mutation counts  
example.M2.dat$l.vec # locus lengths  
example.M2.dat$n.vec # number of samples  
example.M2.dat$c.vec # number of observations

It is well-established that failure to account for among-locus rate variation can result in inadequate estimates of sequence divergence. So, let's see what happens when we do not account for among-locus rate variation (i.e., model misspecification). Here the data were generated under ThetaMater.M2, but we will first simulate posterior distributions of theta using ThetaMater.M1 (does not account for rate variation)

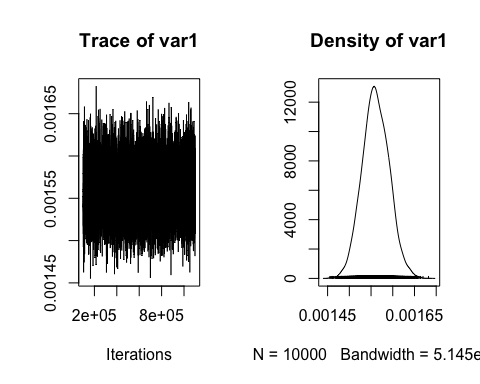
shape = 2  
scale = 0.001  
  
example.MCMC.M1.M2 <- ThetaMater.M1(k.vec = example.M2.data$k.vec, l.vec = example.M2.data$l.vec, n.vec = example.M2.data$n.vec, c.vec = example.M2.data$c.vec, theta.shape = shape, theta.scale = scale, ngens = 1000000, burnin = 100000, thin = 100)

The results from this 'model-misspecification' analysis are shown below:

data(example.MCMC.M2,package= "ThetaMater")  
mean(example.MCMC.M2)

## [1] 0.001560757

plot(as.mcmc(example.MCMC.M2))



In the above case, using M1 instead of the correct M2 (or M3) lead to substantially lower estimates of with a posterior distribution that is tightly peaked at theta = 0.0015.  
Alright, so let's use M2 to infer the posterior distribution of given = 0.10 and k = 4

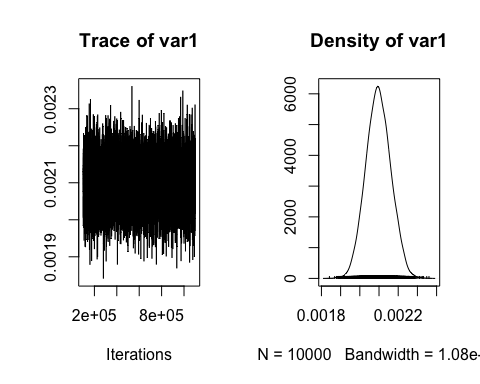
shape = 2  
scale = 0.001  
  
example.MCMC.M2 <- ThetaMater.M2(k.vec = example.M2.data$k.vec, l.vec = example.M2.data$l.vec, n.vec = example.M2.data$n.vec, c.vec = example.M2.data$c.vec, theta.shape = shape, theta.scale = scale, ngens = 1000000, burnin = 100000, thin = 100, K = 4, alpha.param = 0.1)

And here are the results from this analysis:

data(example.MCMC.M2.M2,package= "ThetaMater")  
mean(example.MCMC.M2.M2)

## [1] 0.002096711

plot(as.mcmc(example.MCMC.M2.M2))



## Step 4.3: ThetaMater.M3: function to simulate a posterior distribution of and

Finally, we use ThetaMater.M3 to estimate the joint posterior distribution of and for our dataset. The input arguments for the function ThetaMater.M3 are as follows:

ThetaMater.M3(k.vec, l.vec, n.vec, c.vec, alpha, K.classes, ngens, burnin, thin, theta.shape, theta.scale, alpha.shape, alpha.scale)  
\* k.vec: vector of mutation counts  
\* l.vec: vector of locus lengths  
\* n.vec: vector of sample counts  
\* c.vec: vector of unique pattern counts  
\* ngens: number of generations to run the MCMC simulation  
\* K.classes: number of diiscrete classes to approximate the gamma distribution (4-20 are commonly used)  
\* burnin: number of generations to discard as burnin  
\* thin: number of generations between recorded MCMC samples  
\* theta.shape: shape parameter of the prior gamma distribution on (See Step 2)  
\* theta.scale: scale parameter of the prior gamma distribution on (See Step 2)  
\* alpha.shape: shape parameter of the prior gamma distribution on (See Step 2)  
\* alpha.scale: scale parameter of the prior gamma distribution on (See Step 2)

Notice: here we will set the prior distributions for both and (theta.shape, theta.scale, alpha.shape, alpha.scale). Let's load the data from the previous analyses (Step 4.2, M2)

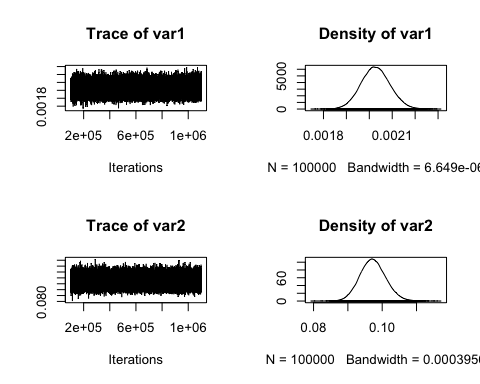
data(example.M2.dat, package= "ThetaMater")  
# Let's look at the data  
example.M2.dat$k.vec # mutation counts  
example.M2.dat$l.vec # locus lengths  
example.M2.dat$n.vec # number of samples  
example.M2.dat$c.vec # number of observations

Let's run these analysis using the command below:

example.MCMC.M3 <- ThetaMater.M3(k.vec = example.M2.dat$k.vec, l.vec = example.M2.dat$l.vec, n.vec = example.M2.dat$n.vec, c.vec = example.M2.dat$c.vec, K = 4, ngens = 1000000, burnin = 100000, thin = 10, theta.shape = theta.shape, theta.scale = theta.scale, alpha.shape = alpha.shape, alpha.scale = alpha.scale)

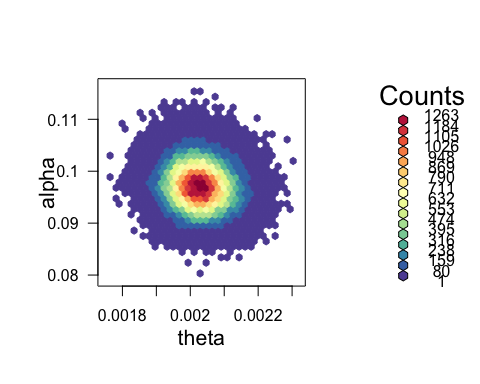
And here are the results from this run, with the posterior of on top and on bottom

data(example.MCMC.M3,package= "ThetaMater")  
plot(as.mcmc(example.MCMC.M3))



We can also make a nice 3D hexbin plot with colors indicating the number of MCMC steps in that state (i.e., colors showing higher posterior probability):

# See instructions at http://www.everydayanalytics.ca/2014/09/5-ways-to-do-2d-histograms-in-r.html  
library(hexbin)  
library(RColorBrewer)  
rf <- colorRampPalette(rev(brewer.pal(11,'Spectral')))  
h <- hexbin(example.MCMC.M3)  
plot(h, colramp=rf, xlab = "theta", ylab = "alpha")



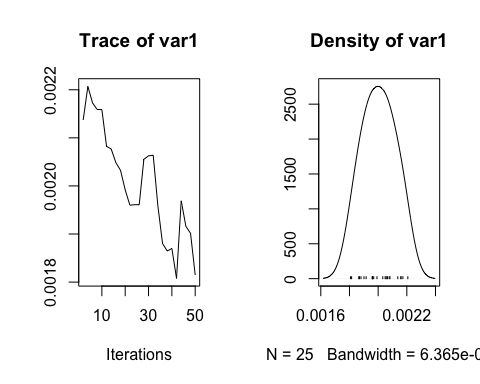
## Step 5: Evaluating the results of a ThetaMater analysis

We can visualize/assess the results of MCMC runs under each model as using the mcmc plotting function provided by MCMCpack. As discussed earlier, it is important to assess the mcmc mixing behavior to convergence on the posterior distribution. In this sense, we want to make sure the posterior sampled sufficient steps to reach the posterior distribution. Roughly speaking, we are looking for a 'fuzzy caterpillar' shape of the mcmc trace shown in the plots. These plots can be used to decide how many generations should be discarded as burnin; these are steps that are correlated with the iniatial state, and may not be accurate approximations to the true posterior. For example, the below MCMC analyses has yet to reach convergence and does not show the "fuzzy caterpillar" shape that is indicative of convergence to the posterior distribution.

data(example.dat,package= "ThetaMater")  
  
example.MCMC <- ThetaMater.M1(k.vec = example.dat$k.vec, l.vec = example.dat$l.vec, n.vec = example.dat$n.vec, c.vec = example.dat$c.vec, ngens = 50, burnin = 1, theta.shape = shape, theta.scale = scale, thin = 2)

## MCMCmetrop1R iteration 1 of 51   
## function value = -881.20219  
## theta =   
## 0.00219  
## Metropolis acceptance rate = 1.00000  
##   
##   
##   
## @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@  
## The Metropolis acceptance rate was 0.72549  
## @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@

plot(example.MCMC)



Here's a much better mcmc run with many more steps sampled and adequate burnin, demonstrating the "fuzzy caterpillar" shape:

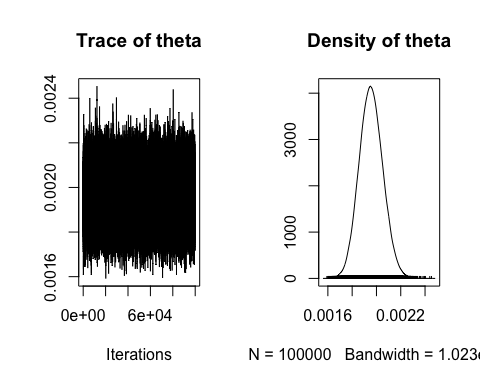
file.loc <- system.file("example.MCMC.M1.csv", package="ThetaMater")  
mean(as.mcmc(read.csv(file = file.loc))) # close to the simulated value of 0.002

## [1] 0.00195588

sd(as.mcmc(read.csv(file = file.loc)))

## [1] 9.646917e-05

plot(as.mcmc(read.csv(file = file.loc)))



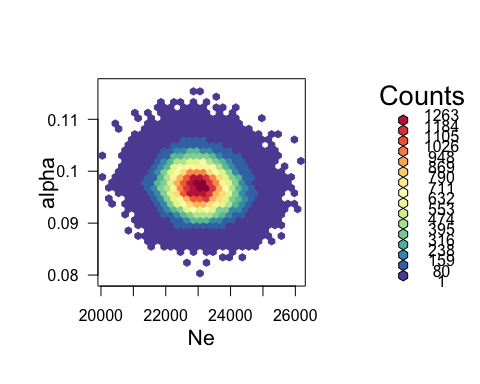
Use the argument ngens to run the MCMC chain longer if your analysis has not yet converged to stationarity.

## Step 6: (Optional) Convert estimates into estimates of effective population size

It is often desirable to convert estimates of into estimates of the true effective population size . Giving an estimate of the mutation rate , we can convert the posterior distriburion of into a posterior distribution of population

For example, let's assume our given population evolved under a mutation rate of 2.2\*10^-8 (similar to human estimates). We simply take the results from ThetaMater and divide the vector of by this mutation rate and a factor of 4 (or 2 for haploid)

# load the results from a ThetaMater analysis  
data(example.MCMC.M3,package= "ThetaMater")  
mutation.rate = 2.2\*10^-8  
example.MCMC.M3.Ne <- example.MCMC.M3  
example.MCMC.M3.Ne[,1] = example.MCMC.M3.Ne[,1]/(mutation.rate\*4)  
h <- hexbin(example.MCMC.M3.Ne)  
plot(h, colramp=rf, xlab = "Ne", ylab = "alpha")



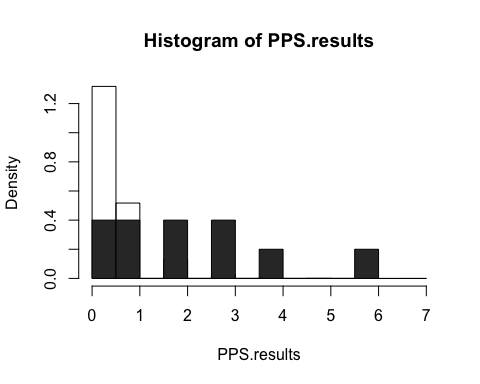
## Step 7: (Optional) conduct posterior predictive simulation to remove loci with evidence of unlikely mutation counts (i.e., potential paralogs)

Finally, we can leverage the posterior distribution of that is estimated by ThetaMater to simulate posterior predictive simulated (PPS) distributions of mutation counts (k.vec) using the function ThetaMaterPPS. We can leaverage this PPS distribution filter out loci with unexpected mutation counts (i.e., likely potential paralogs). The commands for using the function ThetaMater.PPS are as follows:

ThetaMater.PPS(theta.MCMC, l.vec, n.vec)  
\* l.vec: vector of locus lengths  
\* n.vec: vector of sample counts  
\* theta.MCMC: Posterior distribution of theta inferred via ThetaMater

In this example, the PPS and the observed distribution overlap considerably, and thus there is no need to filter out loci based on mutation counts alone

data(example.dat,package= "ThetaMater")  
file.loc <- system.file("example.MCMC.M1.csv", package="ThetaMater")  
mcmc.results <- read.csv(file = file.loc) # close to the simulated value of  
PPS.results <- ThetaMater.PPS(theta.MCMC = mcmc.results[,1], l.vec = example.dat$l.vec, n.vec = example.dat$n.vec)  
hist(PPS.results, freq = F, breaks = 10)  
hist(example.dat$k.vec, col="gray20",add=T, freq = F, breaks = 10)



Now, in this next sample we have a simulated dataset in which 4 paralogous loci have been erronously placed into the same alignment (i.e., assumed to be homologous). Using the code below, we will load the 'example.SeqError.alles' data into R and first estimate theta using this unfiltered dataset.

# Let's look at the mutation count vector  
data(example.SeqError.alles,package= "ThetaMater")  
example.SeqError.alles$k.vec

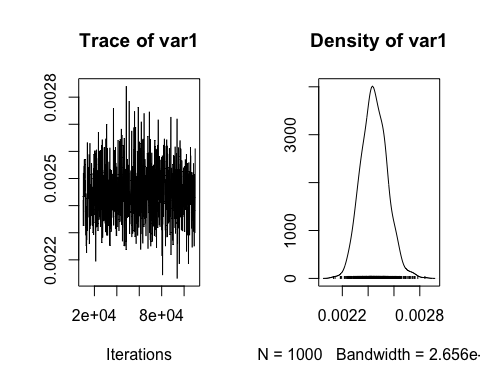
## [1] 0 1 2 23 28 3 31 38 4 5

Next, we estimated using ThetaMater.M1 on this unfiltered dataset:

mcmc.seq.error <- ThetaMater.M1(k.vec= example.SeqError.alles$k.vec, l.vec = example.SeqError.alles$l.vec, n.vec = example.SeqError.alles$n.vec, c.vec = example.SeqError.alles$c.vec, ngens = 100000, burnin = 10000, thin = 100, theta.shape = 2, theta.scale = 0.001)

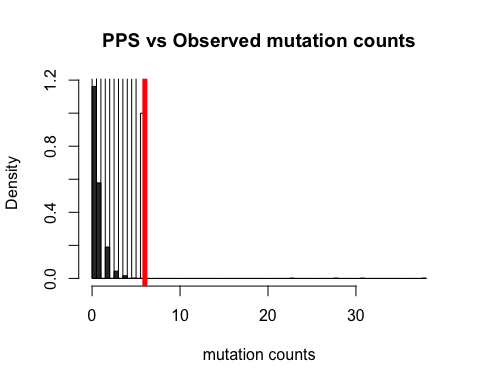
And here are the results:

data(example.MCMC.SequenceError, package = "ThetaMater")  
plot(as.mcmc(example.MCMC.SequenceError))



Now let's conduct PPS using ThetaMater.PPS and overlay the distributions to see if we can identiy the 4 outlier loci:

# run PPS analysis using this command:  
example.PPS.results <- ThetaMater.PPS(theta.MCMC = example.MCMC.SequenceError[,1], l.vec = example.SeqError.alles$l.vec, n.vec = example.SeqError.alles$n.vec)  
hist(rep(example.SeqError.alles$k.vec, example.SeqError.alles$c.vec), col="gray20",add=F, freq = F, breaks = 100, main = "PPS vs Observed mutation counts", xlab = "mutation counts")  
hist(example.PPS.results, freq = T, breaks = 10, add = T)  
abline(v=6, col="red", lwd = 5)



Here the red line shows the maximum number of mutations observed in the PPS data. So, let's remove the four extreme loci that are beyond this value and reestimate after filtering using this PPS distribution

max(example.PPS.results)

## [1] 6

example.Filtered <- FilterData.PPS(dataset = example.SeqError.alles, threshold = 6)

Now, let's estimate after PPS filtering using the following commands:

# Now let's estimate theta after PPS filtering  
example.MCMC.PostFilter <- ThetaMater.M1(k.vec = example.Filtered$k.vec, l.vec = example.Filtered$l.vec, n.vec = example.Filtered$n.vec, c.vec = example.Filtered$c.vec, ngens = 100000, burnin = 10000, thin = 100, theta.shape = 2, theta.scale = 0.001)

And let's load these results below:

data(example.MCMC.PostFilter, package = "ThetaMater")  
plot(as.mcmc(example.MCMC.PostFilter))

