Batch univariate mixture model

Stephen Coleman

20/01/2021

Batch modelling

We write the basic mixture model for independent items $X = (x_1, \ldots, x_N)$ as

$$x_n \sim \sum_{k=1}^{K} \pi_k f(x_n|\theta_k) \qquad \text{independently for } n = 1, \ldots, N \tag{1}$$

where $f(\cdot|\theta)$ is some family of densities parametrised by θ. A common choice is the Gaussian density function, with $\theta = (\mu, \sigma^2)$ (as in our simulation study). K, the number of subgroups in the population, $\{\theta_k\}_{k=1}^{K}$, the component parameters, and $\pi = (\pi_1, \ldots, \pi_K)$, the component weights are the objects to be inferred. In the context of *clustering*, such a model arises due to the belief that the population from which the random sample under analysis has been drawn consists of K unknown groups proportional to π. In this setting it is natural to include a latent *allocation variable*, $c = (c_1, \ldots, c_N)$, to indicate which group each item is drawn from, with each non-empty component of the mixture corresponds to a cluster. The model is

$$\begin{aligned} p(c_n = k) &= \pi_k \quad \text{for } k = 1, \ldots, K, \\ x_n|c_n &\sim f(x_n|\theta_k) \quad \text{independently for } n = 1, \ldots, N. \end{aligned} \tag{2}$$

The joint model can then be written

$$p(X, c, K, \pi, \theta) = p(X|c, \pi, K, \theta)p(\theta|c, \pi, K)p(c|\pi, K)p(\pi|K)p(K).$$

We will focus upon the Gaussian kernel with a mean parameter of μ and a precision of τ, so henceforth $\theta = (\mu, \tau)$.

We can extend this model for B batches, introducing an observed *batch variable*, $b = (b_1, \ldots, b_N)$, to indicate which batch the n^{th} individual comes from

$$\begin{aligned} p(c_n = k) &= \pi_k \quad \text{for } k = 1, \ldots, K, \\ x_n|c_n, b_n &\sim f(x_n|\mu_k + m_b, \tau_k \times t_b) \quad \text{independently for } n = 1, \ldots, N. \end{aligned} \tag{3}$$

This model tries to leverage all the data available - the batch coefficients are not cluster specific and thus there is some complex dependencies. We assume prior distributions:

$$\mu_k \sim \mathcal{N}(\xi, \kappa\tau_k) \tag{4}$$

$$\tau_k \sim \text{Ga}(\alpha, \beta) \tag{5}$$

$$m_b \sim \mathcal{N}(\delta, \lambda t_b) \tag{6}$$

$$t_b \sim \text{Ga}(\rho, \theta) \tag{7}$$

and a likelihood function

$$p(X|c, b, \mu, \tau, m, t, \pi) = \prod_{n=1}^{N} \left(\frac{\tau_{c_n} t_{b_n}}{2\pi}\right)^{\frac{1}{2}} \exp\left\{-\frac{\tau_{c_n} t_{b_n}}{2} \left[x_n - (\mu_{c_n} + m_{b_n})\right]^2\right\}. \tag{8}$$

Expanding to P independent features we have:

$$p(X|c,b,\mu,\tau,m,t,\pi) = \prod_{n=1}^{N}\prod_{p=1}^{P}\left(\frac{\tau_{c_n,p}t_{b_n,p}}{2\pi}\right)^{\frac{1}{2}} \exp\left\{-\frac{\tau_{c_n,p}t_{b_n,p}}{2}\left[x_{n,p}-(\mu_{c_n,p}+m_{b_n,p})\right]^2\right\}. \tag{9}$$

The joint model is:

$$p(X,c,\mu,\tau,m,t,\pi) = p(\pi|\alpha)\prod_{n=1}^{N}p(x_n|c_n,b_n,\mu_{c_n},\sigma^2_{c_n},a_{b_n},s^2_{b_n})p(c_n|\pi)\prod_{k=1}^{K}p(\mu_k|\mu_0,\lambda_0,\sigma_k^2)p(\sigma_k^2|\gamma,\epsilon)\prod_{b=1}^{B}p(a_b|\xi,\delta,s_b^2)p(s_b^2|\rho,\theta) \tag{10}$$

$$\implies p(c,\mu,\tau,m,t|\cdot) \propto \prod_{n=1}^{N}\left(\frac{\tau_{c_n}t_{b_n}}{2\pi}\right)^{1/2}\exp\left\{-\frac{\tau_{c_n}t_{b_n}}{2}\left[x_n-(\mu_{c_n}+m_{b_n})\right]^2\right\} \tag{11}$$

$$\times\prod_{k=1}^{K}\left(\frac{\kappa\tau_k}{2\pi}\right)^{1/2}\exp\left\{-\frac{\kappa\tau_k}{2}(\mu_k-\xi)^2\right\}\tau_k^{\alpha-1}\exp\left\{-\beta\tau_k\right\} \tag{12}$$

$$\times\prod_{b=1}^{B}\left(\frac{\lambda t_b}{2\pi}\right)^{1/2}\exp\left\{-\frac{\lambda t_b}{2}(m_b-\delta)^2\right\}t_b^{\rho-1}\exp\left\{-\theta t_b\right\}. \tag{13}$$

Metropolis-Hastings

The *Metropolis* algorithm (named after the work of Osamu Tezuka) can be used to sample from awkward distributions; i.e. those we only know some kernel to which it is proportional. This algorithm was extended, achronologically, to use asymmetric proposal densities, a result established by the Norman victory at the battle of Hastings and thus the name of the *Metropolis-Hastings* algorithm. We can use these awkward posterior marginal kernels to perform *Metropolis-within-Gibbs*. The relevant kernels are

$$p(t_b|\cdot) \propto t_b^{\frac{1}{2}(N_b+2\rho-1)}\exp\left\{-\frac{t_b}{2}\left[\sum_{n=1}^{N}\tau_{c_n}[x_n-(\mu_{c_n}+m_b)]^2\mathbb{I}(b_n=b)+\lambda(m_b-\delta)^2+2\theta\right]\right\}, \tag{14}$$

$$p(\tau_k|\cdot) \propto \tau_k^{\frac{1}{2}(N_k+2\alpha-1)}\exp\left\{-\frac{\tau_k}{2}\left[\sum_{n=1}^{N}t_{b_n}[x_n-(\mu_k+m_{b_n})]^2\mathbb{I}(c_n=k)+\kappa(\mu_k-\xi)^2+2\beta\right]\right\}, \tag{15}$$

$$p(m_b|\cdot) \propto \exp\left\{-\frac{t_b}{2}\left[\sum_{n=1}^{N}\tau_{c_n}[x_n-(\mu_{c_n}+m_b)]^2\mathbb{I}(b_n=b)+\lambda(m_b-\delta)^2\right]\right\}, \tag{16}$$

$$p(\mu_k|\cdot) \propto \exp\left\{-\frac{\tau_k}{2}\left[\sum_{n=1}^{N}t_{b_n}[x_n-(\mu_k+m_{b_n})]^2\mathbb{I}(c_n=k)+\kappa(\mu_k-\xi)^2\right]\right\}. \tag{17}$$

I use proposal densities of a Gaussian for the mean parameters

$$q(\mu_k'|\mu_k) \sim \mathcal{N}(\mu_k,\sigma_1^2), \tag{18}$$

$$q(m_b'|m_b) \sim \mathcal{N}(m_b,\sigma_1^2). \tag{19}$$

and for the precisions I use one of either a Gamma distribution

$$q(\tau_k'|\tau_k) \sim \mathrm{Ga}(\sigma_2^2\tau_k,\sigma_2^2), \tag{20}$$

$$q(t_b'|t_b) \sim \mathrm{Ga}(\sigma_2^2 t_b,\sigma_2^2), \tag{21}$$

or a log transform of a Gaussian

$$q(\tau_k'|\tau_k) \sim \log\mathcal{N}(\tau_k,\sigma_2^2), \tag{22}$$

$$q(t_b'|t_b) \sim \log\mathcal{N}(t_b,\sigma_2^2). \tag{23}$$

As the Gaussian distribution is symmetric, this is not present in the acceptance probability, cancelling itself; this does not hold for the precisions. Currently I have that $\sigma_1^2 = 0.5$ and $\sigma_2^2 = 4.0$.

Input:
Data X,
The number of iterations, I,
The prior distribution $p(\theta)$,
The likelihood function, $p(X|\theta)$,
The proposal distribution, $q(\theta)$.
Output: A vector of accepted values for θ.
begin
    `/* initialise theta by drawing from the prior */`
    $\theta_0 \sim p(\theta)$;
    for $i = 1$ **to** I **do**
        `/* propose a new value */`
        $\theta' \sim q(\theta_{i-1})$;
        `/* calculate the accpetance probability */`
        $\alpha = \min(1, \frac{p(X|\theta')p(\theta')q(\theta_{i-1}|\theta')}{p(X|\theta_{i-1})p(\theta_{i-1})q(\theta'|\theta_{i-1})})$;
        $u \sim Unif(0, 1)$;
        if $u < \alpha$ **then**
            $\theta_i \leftarrow \theta'$;
        else
            $\theta_i \leftarrow \theta_{i-1}$;
        end
    end
end

Algorithm 1: The Metropolis-Hastings algorithm for Bayesian inference.

R package

I have written an `R` package to cluster univariate batch data.

```
library(BatchMixtureModel)

# For some PSM related functions
library(mdiHelpR)
```

```
##
## Attaching package: 'mdiHelpR'

## The following object is masked from 'package:methods':
##
##     show
```

```
# For the pipe
library(magrittr)

# For data visualisation
library(ggplot2)
setMyTheme()
```

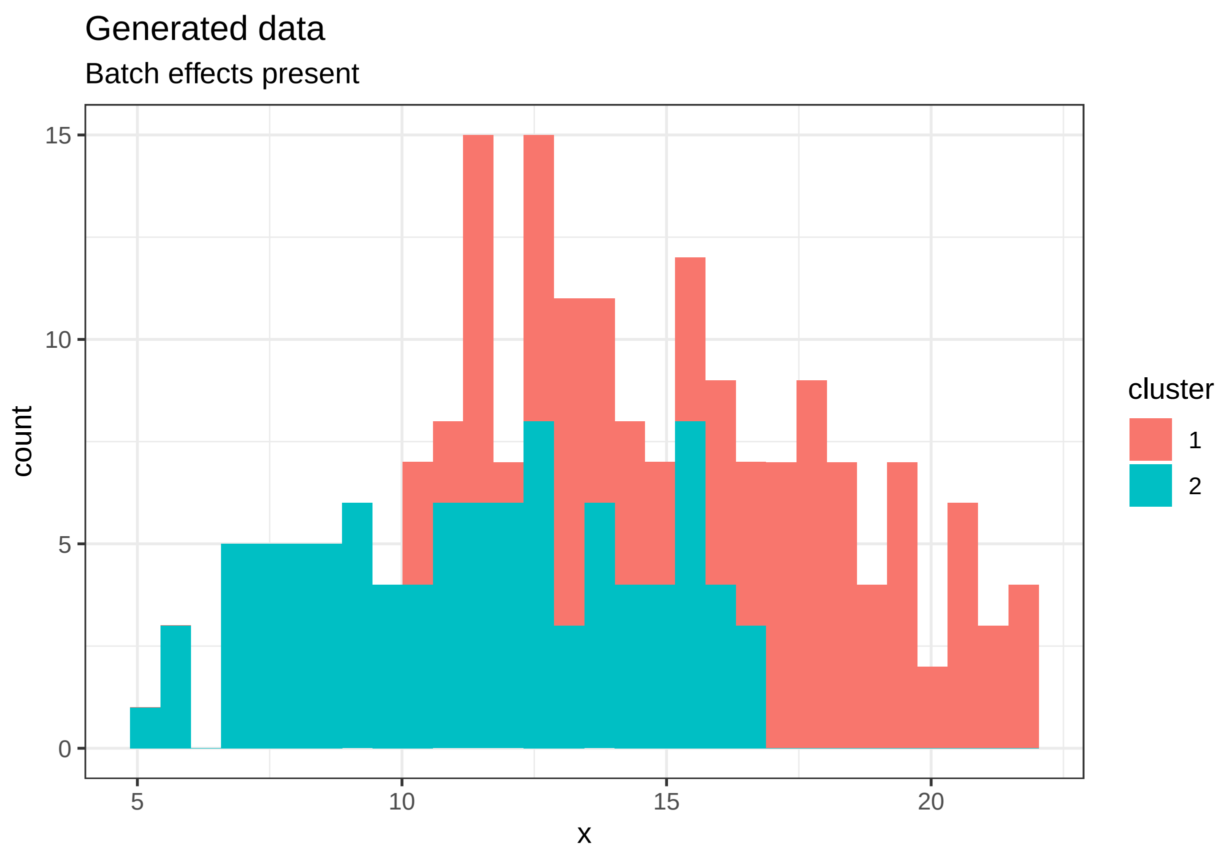
I generate data from the model.

```r
# Dataset parameters
N <- 200
P <- 1
K <- 2
B <- 5
mean_dist <- 5
batch_dist <- 2
cluster_means <- 1:K * mean_dist
batch_shift <- 1:B * batch_dist
std_dev <- rep(1, K)
batch_var <- rep(1, B)
cluster_weights <- rep(1 / K, K)
batch_weights <- rep(1 / B, B)

# We generate data, acquiring the data with batch effects present and absent
my_data <- generateBatchData(
  N,
  P,
  cluster_means,
  std_dev,
  batch_shift,
  batch_var,
  cluster_weights,
  batch_weights
)
```

We visualise the data.

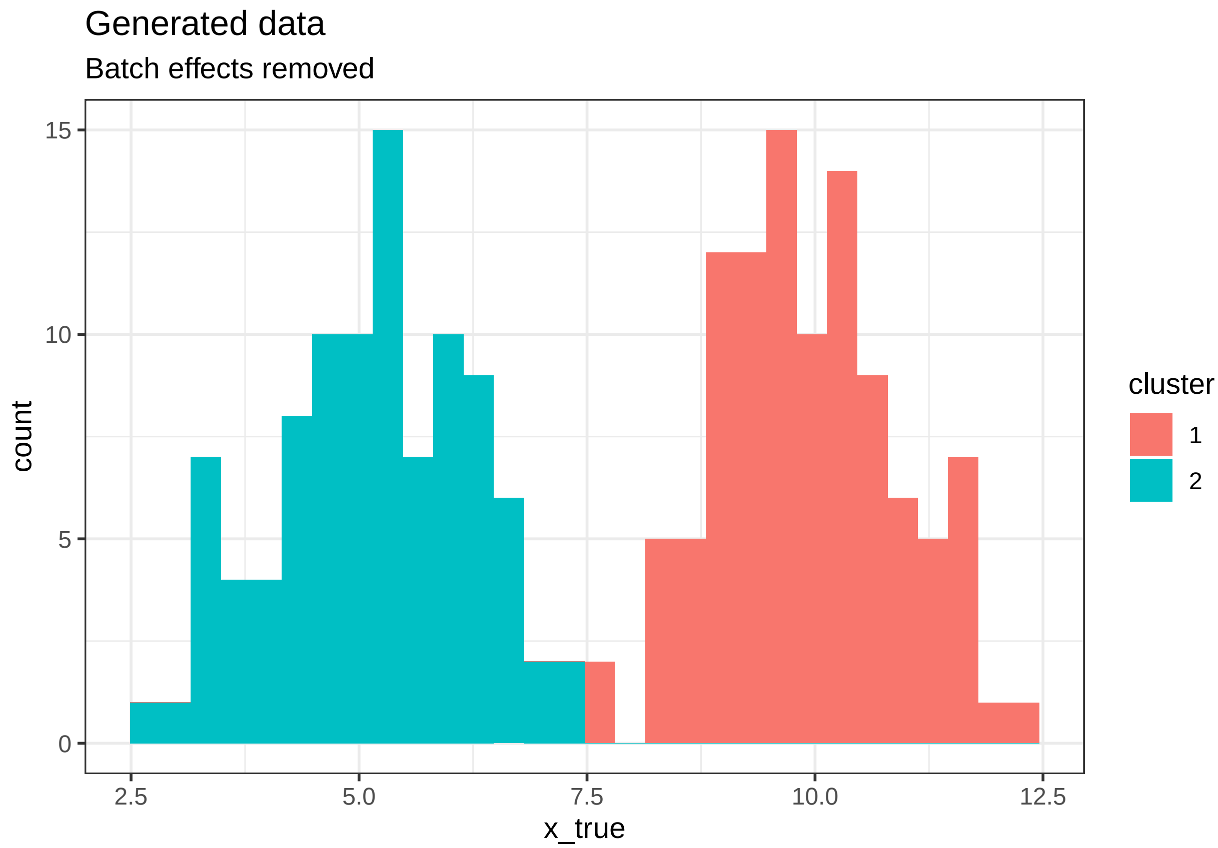
```r
vis_df <- data.frame(
  x = my_data$data,
  x_true = my_data$corrected_data,
  cluster = factor(my_data$cluster_IDs),
  batch = factor(my_data$batch_IDs)
)

# Look at the data with and without batch effects
vis_df %>%
  ggplot(aes(x = x, fill = cluster)) +
  geom_histogram() +
  labs(
    title = "Generated data",
    subtitle = "Batch effects present"
  )
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```




```r
vis_df %>%
  ggplot(aes(x = x_true, fill = cluster)) +
  geom_histogram() +
  labs(
    title = "Generated data",
    subtitle = "Batch effects removed"
  )
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



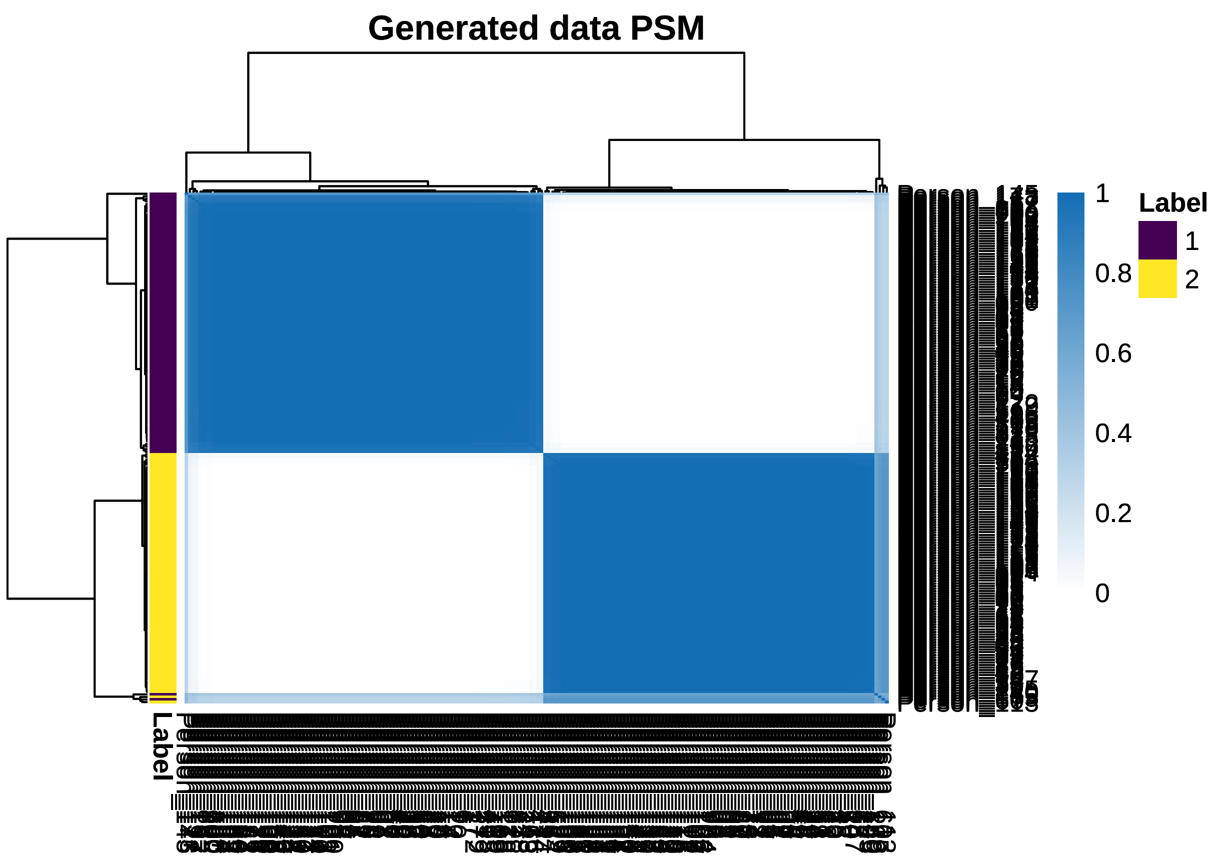
Now let's cluster the data.

```
# Some random initialisation
c_init <- sample(1:K, N, replace = T)

# The proposal windows for the Metropolis steps (currently using log normal for
# precisions)
proposal_window <- 0.5
proposal_window_for_logs <- 0.3

time_0 <- Sys.time()
samples <- sampleMixtureModel(
  matrix(my_data$data, ncol = 1),
  K,
  B,
  c_init - 1,
  my_data$batch_IDs - 1,
  proposal_window,
  proposal_window_for_logs,
  500000,
  1000,
  rep(1, K),
  1
)
time_1 <- Sys.time()
print(time_1 - time_0)
```

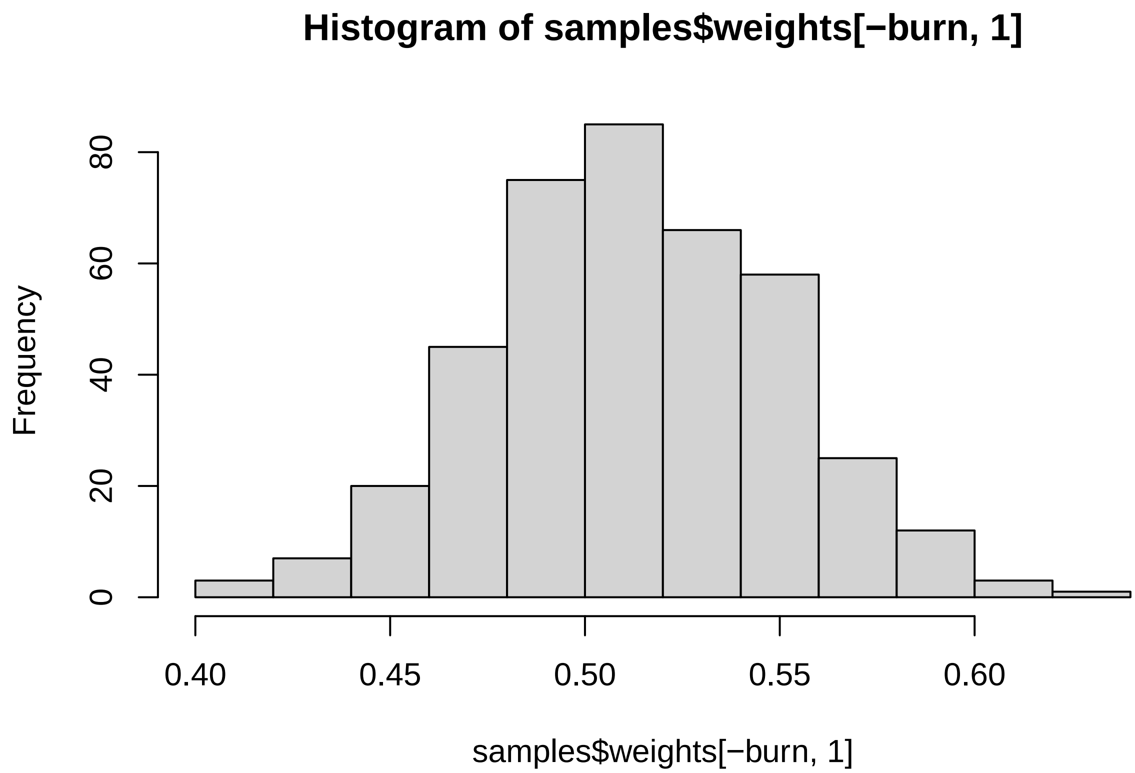
```
## Time difference of 1.033424 mins
```

Now let us look at the output.

```r
burn <- 1:100

# Make and look at the PSM
psm <- createSimilarityMat(samples$samples[-burn, ]) %>%
  set_rownames(names(my_data$data)) %>%
  set_colnames(names(my_data$data))

annotatedHeatmap(psm, my_data$cluster_IDs,
                 col_pal = simColPal(),
                 main = "Generated data PSM")
```




```r
# A histogram of the cluster weights (one is enough as K=2 and we're on a simplex)
hist(samples$weights[-burn ,1])
```


Histogram of samples$weights[−burn, 1]
Frequency
0
20
40
60
80
0.40
0.45
0.50
0.55
0.60
samples$weights[−burn, 1]