Math 640 Final Paper

Max Kearns and Tommy Jones

May 10, 2018

1 Introduction

Analyzing text data is of interest to frequentists and Bayesians. Text stores information that is not easily evaluated with well-understood statistical methods. Textual data is not as well understood as numerical data. In this paper, we explore two alternatives for modeling text.

Word frequencies may be modeled as a multinomial distribution with a Dirichlet prior and (possibly) a uniform hyperprior. This specification models uncertainty on the probability of words, $\vec{\theta}$, and uncertainty on the Dirichlet prior parameter, $\vec{\alpha}$. The uniform hyperprior is totally non-informative. Yet an empirical law of language, Zipf's law, hints at an informative prior. We hope to use Zipf's law to provide principled uncertainty on $\vec{\alpha}$ in the hopes of improving the model.

Zipf's law is an empirical propety of natural language. It states that the word frequencies of any corpus of text follows a power law distribution, regardless of context or language. The most common word will be twice as frequent as the second most common word, and n times as frequent as the nth most common word.

Informed by Zipf's law, we place a $Pareto(\gamma, \beta)$ prior on $\vec{\alpha}$. We assume γ is fixed at the small value of 0.01. We hypothesize that the elements of $\vec{\alpha}$ are proportional to the data, $\vec{y}$, based on the law of total expectation. This is shown in figure 1. The Pareto distribution is a power-law that should provide some uncertainty on $\vec{\alpha}$ that mirrors this inherent property of language.

In order to determine whether this prior merits further research on more complex models, we model a small data set using a simple model that features a multinomial likelihood, a Dirichlet prior, a Pareto hyper-prior, with a non-informative Jeffery's hyper-prior . We compare this model to a control that does not allow for uncertainty on $\vec{\alpha}$. Our data set is 100 randomly sampled NIH grant abstracts from 2014.

$$
\begin{aligned}
E[\vec{y}] &= E\left[E\left[\vec{y}|\vec{\theta}\right]\right] \\
E[\vec{y}] &= E\left[n\vec{\theta}\right] \\
E[\vec{y}] &= nE[\vec{\theta}] \\
E[\vec{y}] &= n\frac{\vec{\alpha}}{\sum_k \alpha_k} \\
E[\vec{y}] &\propto \vec{\alpha}
\end{aligned}
$$

Figure 1: α is roughly proportional to a power law.

2 Methods

2.1 Models

For both the control and the experimental model, we assume that the word count $\vec{y}$ is Multinomial, so has the following likelihood.

$$\vec{y} \sim multinom(n, \vec{\theta}) \implies \mathcal{L}(\vec{y}|\vec{\theta}, \vec{\alpha}, \beta) \propto \prod_k \theta_k^{y_k} \tag{1}$$

On $\vec{\theta}$, the vector of word probabilities, we place a Dirichlet prior.

$$\vec{\theta} \sim Dir(\vec{\alpha}) \implies \pi(\vec{\theta}) = \mathcal{B}(\vec{\alpha}) \prod_k \theta_k^{\alpha_k - 1} \tag{2}$$

The control model places a uniform hyperprior on each α_k. The control model then has the simple form $\vec{\theta}|\vec{y}, \vec{\alpha} \sim Dir(\vec{y} + \vec{\alpha})$, and an unknown distribution for $\vec{\alpha}|\vec{y}, \vec{\beta}$; shown below (the full derivation can be found in Appendix B). We will sample from $\vec{\alpha}|\vec{y}, \vec{\beta}$ using Metropolis-Hastings with an Inverse-Gaussian proposal.

$$p(\alpha_k|\theta_k, y_k) \propto \theta_k^{\alpha_k} \tag{3}$$

The candidate model, however, will assume a $Pareto(\gamma, \beta)$ prior on α_k, with Jeffery's prior on β.

$$\alpha_k \sim Pareto(\gamma, \beta) \implies \pi(\vec{\alpha}) = \prod_k \gamma^\beta \beta \alpha_k^{-(\beta+1)} \tag{4}$$

$$\pi(\beta) \propto \frac{1}{\beta} \tag{5}$$

This results in the below posterior. A full derivation of the model can be found in Appendix B.

$$P(\vec{\theta}, \vec{\alpha}, \beta|\vec{y}) \propto \beta^{(K-1)} \gamma^{k\beta} \mathcal{B}(\vec{\alpha}) \prod_k \theta_k^{y_k+\alpha_k-1} \alpha_k^{-(\beta+1)} \tag{6}$$

The full conditional posteriors are below.

$$P(\vec{\theta}|\vec{\alpha}, \beta, \vec{y}) \propto \prod_k \theta_k^{y_k+\alpha_k-1} \tag{7}$$

$$P(\vec{\alpha}|\vec{\theta}, \beta, \vec{y}) \propto \mathcal{B}(\vec{\alpha}) \prod_k \theta_k^{y_k+\alpha_k-1} \alpha_k^{-(\beta+1)} \tag{8}$$

$$P(\beta|\vec{\theta}, \vec{\alpha}, \vec{y}) \propto \beta^{K-1} \exp\left[-\beta\left(\sum_k \log(\alpha_k) - k\log(\gamma)\right)\right] \tag{9}$$

$$\tag{10}$$

Of these conditional posteriors, only $\vec{\alpha}|\vec{\theta}, \beta, \vec{y}$ is an unknown distribution. $\vec{\theta}|\vec{\alpha}, \beta, \vec{y}$ is a $Dir(\vec{y} + \vec{\alpha})$, and $\beta|\vec{\theta}, \vec{\alpha}, \vec{y}$ is a Gamma$(k, \sum_k \log(\alpha_k) - k\log(\gamma))$. These may be sampled using the Gibbs algorithm. We use a Metropolis-Hastings algorithm with an Inverse-Gaussian proposal to sample $\vec{\alpha}|\vec{\theta}, \beta, \vec{y}$.

2.2 Sampling

2.2.1 Data

The NIH provides grant abstracts to the public, and these documents provide a perfect data set upon which to test our model. The data set contains 5,542 unique words. Figure 2 shows the properties of Zipf's law in the data set. In the histogram, it is evident just how much mass is in the right tail of the distribution. It shows that the overwhelming majority of the words occur less than 100 times, despite the fact that the most frequent word, 'the', occurs 1,928 times. The right side of figure 2 shows a log transformation of the word count and the rank. When graphed in this way, power laws appear as a straight line. This shows that our data set does indeed follow a power law, despite some noise toward the lower end of the distribution. This is a common result, and given a larger sample, this noise would be reduced.

2.2.2 Samplers

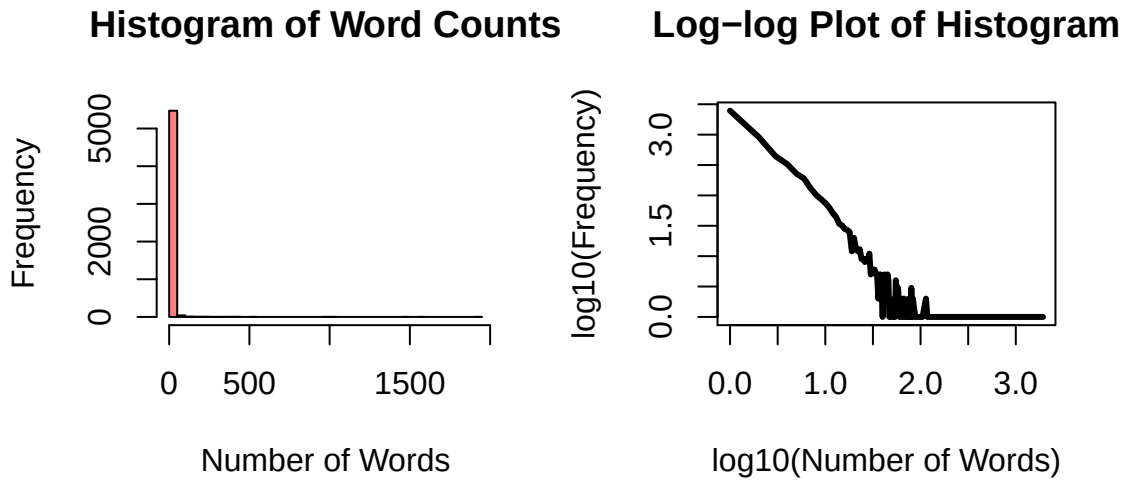


Figure 2: Histogram of Word counts (left) and Log-log plot of word counts, ordered by rank (right).

As discussed above, to sample from these conditional distributions, our samplers have both Gibbs and Metropolis-Hastings components. For both the control and the man model, we used an Inverse-Gaussian to sample from the conditional posterior on α. For the control model, the Inverse-Gaussian parameters are $\mu = 0.1$ and $\lambda = 0.01$. For the main model, the Inverse-Gaussian parameters are $\mu = 1$ and $\lambda = 0.01$. We ran four chains of 20,000 iterations each.

Table 1: Acceptance Rates

	Main	Control
min.	0.09	0.28
25%	0.18	0.47
50%	0.19	0.49
75%	0.21	0.51
max.	0.41	0.51

Table 1 provides a summary of the distributions of the acceptance rates for each element of $\vec{\alpha}$. While the acceptance rates are diverse for each element of $\vec{\alpha}$, they all fall within an acceptable range for both models.

We use the Geweke statistic on the log posterior liklihood of each model to assess convergence. The control model shows convergence with this statistic (p=.02), but the main model is well outside the typical range for a convergent sample(p=-2.5). Convergence for this sampler is extremely difficult to achieve, and may be impossible. We attempt many proposal distributions, and none allowed this sample to converge within a reasonable number of iterations. This is one of the flaws of our main model. We also inspect some visual convergence diagnostics (Appendix C), but they all indicate that the control has converged and the main model has not.

2.3 Comparison

For the comparison of the models, we calculate the deviance information criterion (DIC) for each estimate of α, creating a distribution of DICs. This distribution is shown in figure 3. We also provide the plot of the densities of the log-likelihood of each estimate of α (Figure 3).

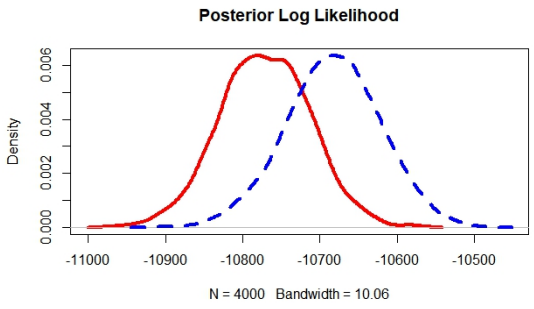


Figure 3: Distribution of log-likelihoods.

3 Results

We find mixed evidence that the main model is an improvement. In figure 3, the log posterior liklihood corresponding to the control model is markedly higher than the main model. In figure 4, the DIC corresponding to the main model is lower. (With DIC, lower is better.) Given

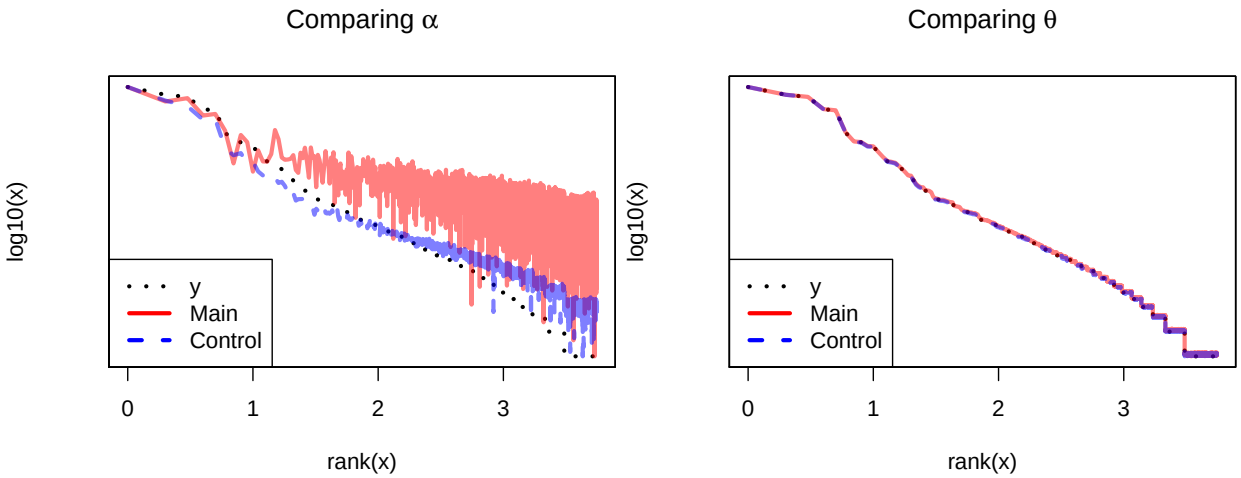


Figure 5: Comparison of the log-log transformation of the models and the data (left) and comparison of the probabilty estimates of the models and the data (right).

this mixed evidence together with the increased complexity of the main model and its lack of convergence, we do not believe the main model improves upon the control.

4 Discussion

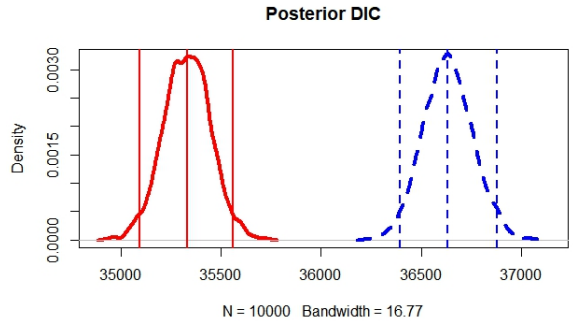


Figure 4: Distribution of DICs; vertical lines indicate the median and credible intervals.

The main model did not significantly improve upon the control. Indeed, the main model drastically increased the complexity of the process, without conclusively improving the estimation. The lesson of this experiment seems to be that increased complexity does not necessarily mean better estimation, even when using a well-informed prior.

Figure 5 shows the log-log plot of the data, compared to the two models. Our hypothesis that α is roughly proportional to the data is somewhat borne out by this plot in the control model. The experimental model also has a similar slope, although the lack of convergence is displaying noise on this plot as well.

In figure 5, we also compare the two models' estimates of θ. The two models are nearly identical, and this result makes sense based on inspection of the conditional distribution of θ. The model is dominated by the data, so changes in α have little impact on the estimation of θ. This result confirms that even with convergence of alpha in the experimental model would not provide a meaningful improvement in estimation.

5 References

Meyers, G. (1994). Quantifying the Uncertainty in Claim Severity Estimates for an Excess Layer When Using the Single Parameter Pareto. In Proceedings of the Casualty Actuarial Society (Vol. 81, pp. 91-122).

Zipf, G. K. (2016). Human behavior and the principle of least effort: An introduction to human ecology. Ravenio Books.

Manaris, B. Z., Pellicoro, L., Pothering, G., & Hodges, H. (2006, February). Investigating Esperanto's Statistical Proportions Relative to other Languages using Neural Networks and Zipf's Law. In Artificial Intelligence and Applications (pp. 102-108).

National Institutes of Health. (2014). NIH ExPORTER. Retrieved from http://exporter.nih.gov/ExPORTER_Catalog.aspx (January 2015)

6 Appendix A

```
knitr::opts_chunk$set(echo = FALSE, cache = TRUE)
# load libraries
library(textmineR)
library(gtools)
library(mcmcplots)
library(EnvStats)
library(statmod)
library(coda)
library(knitr)
library(kableExtra)

# load and prepare the data
dtm <- CreateDtm(doc_vec = nih_sample$ABSTRACT_TEXT, # nih_sample from textmineR
                 doc_names = nih_sample$APPLICATION_ID,
                 ngram_window = c(1,1),
                 stopword_vec = c(),
                 verbose = FALSE)

y <- colSums(dtm) # nih_sample_dtm from the textmineR package

y <- y[order(y, decreasing = TRUE)]

# Histogram of y
hist(y, breaks = 50, col = rgb(1,0,0,0.5),
     main = "Histogram of Word Counts",
     ylab = "Frequency",
     xlab = "Number of Words")

# log-log plot of y
f <- as.data.frame(table(y), stringsAsFactors = FALSE)

plot(log10(as.numeric(f$y)), log10(f$Freq), type = "l", lwd = 3,
     main = "Log-log Plot of Histogram",
     xlab = "log10(Number of Words)", ylab = "log10(Frequency)")

# declare a function for the control sampler
```

```
control_sampler <- function(y, B, seed, theta0, alpha0) {

  # set up sampling functions for M-H
  f_alpha_k <- function(alpha_k, theta_k) {
    theta_k ^ alpha_k
  }

  j_alpha <- function(mu = .1, prob = FALSE, n = NULL, x = NULL, rate = 0.01) {
    # if prob is false, draw a sample, otherwise find p(x)
    if (! prob) {
      # out <- rexp(n = n, rate = rate)
      out <- rinvgauss(n = n, mean = mu, shape = rate)
      # out <- rpareto(n = n, location = mu, shape = rate)
    } else {
      # out <- dexp(x, rate)
      out <- dinvgauss(x, mean = mu, shape = rate)
      # out <- dpareto(x, location = mu, shape = rate)
    }

    out
  }

  # set up constants
  k <- length(y)

  theta <- matrix(0, nrow = B, ncol = k)

  theta[1,] <- theta0

  alpha <- matrix(0, nrow = B, ncol = k)

  alpha[1,] <- alpha0

  acc_alpha <- numeric(ncol(alpha))

  # run the sampler
  for (j in 2:B) {
    # sample theta
    theta[j,] <- rdirichlet(1, y + alpha[j-1,])

    # sample alpha
    alpha_star <- j_alpha(n = k)

    r <- (f_alpha_k(alpha_k = alpha_star, theta_k = theta[j-1,]) /
            f_alpha_k(alpha[j-1,],theta[j-1,])) /
      (j_alpha(prob = TRUE, x = alpha_star) /
         j_alpha(prob = TRUE, x = alpha[j-1,]))

    u <- runif(1)

    keep <- pmin(r,1) > u

    alpha[j,keep] <- alpha_star[keep]
```

```
    alpha[j,!keep] <- alpha[j-1,!keep]

    # acc_alpha[j] <- sum(keep) / k

    acc_alpha <- acc_alpha + keep

  }

  acc_alpha <- acc_alpha / B

  # return the result
  list(theta = theta, alpha = alpha, acc_alpha = acc_alpha, seed = seed,
       theta0 = theta0, alpha0 = alpha0)
}

# run 4 chains of 20,000 iterations each

B <- 20000

control_chains <- list(list(seed = 1020,
                            theta0 = y / sum(y),
                            alpha0 = rep(0.01, length(y))),
                       list(seed = 6,
                            theta0 = c(100, rep(1, length(y) - 1)) /
                              sum(c(100, rep(1, length(y) - 1))),
                            alpha0 = rep(0.1, length(y))),
                       list(seed = 74901,
                            theta0 = rep(1/length(y), length(y)),
                            alpha0 = rep(0.5, length(y))),
                       list(seed = 481,
                            theta0 = c(rep(100, 100), rep(10, length(y) - 100)) /
                               sum(c(rep(1, 100), rep(0, length(y) - 100))),
                            alpha0 = rep(0.9, length(y))))

control_chains <- TmParallelApply(control_chains, function(x){
  # run sampler
  result <- control_sampler(y, B, seed = x$seed, theta0 = x$theta0, alpha0 = x$alpha0)

  # remove 50% burn-in itertions and check convergence
  result$alpha <- result$alpha[(B/2):B,]

  result$theta <- result$theta[(B/2):B,]

  result$geweke_alpha <- apply(result$alpha,2,function(k){
    out <- try(geweke.diag(k)$z)

    if (class(out) == "try-error")
      return(NA)

    out
  })

  result$geweke_theta <- apply(result$theta,2,function(k){
```

```
    out <- try(geweke.diag(k)$z)

    if (class(out) == "try-error")
      return(NA)

    out
  })

  # thin every 10th interation
  result$alpha <- result$alpha[seq(10, nrow(result$alpha), by=10),]

  result$theta <- result$theta[seq(10, nrow(result$theta), by=10),]

  # return full result
  result

}, cpus = 2, export = c("y", "B", "control_sampler"),
   libraries = c("gtools", "EnvStats", "coda", "statmod"))

# check acceptance rate
control_alpha_acc <- sapply(control_chains, function(x) mean(x$acc_alpha))

# check convergence with Geweke statistic
control_theta_conv <- sapply(control_chains, function(x){
  mean(abs(x$geweke_theta) >= 1.96 | is.na(x$geweke_theta))
})

control_alpha_conv <- sapply(control_chains, function(x){
  mean(abs(x$geweke_alpha) >= 1.96 | is.na(x$geweke_alpha))
})

# declare a sampling function for the main model

main_sampler <- function(y, B, seed, theta0, alpha0, beta0, gamma) {

  # set up functions for M-H sampler for alpha

  # these functions are just for a single alpha_k
  f_alpha_k <- function(alpha_k, theta_k, beta) {
    alpha_k ^ (-beta - 1) * theta_k ^ alpha_k
  }

  j_alpha <- function(mu = 1, prob = FALSE, n = NULL, x = NULL, rate = 0.1) {
    # if prob is false, draw a sample, otherwise find p(x)
    if (! prob) {
      # out <- rexp(n = n, rate = rate)
      out <- rinvgauss(n = n, mean = mu, shape = rate)
      # out <- rpareto(n = n, location = mu, shape = rate)
      # out <- rhalfcauchy(n = n, scale = rate)
    } else {
      # out <- dexp(x, rate)
      out <- dinvgauss(x, mean = mu, shape = rate)
      # out <- dpareto(x, location = mu, shape = rate)
```

```
    # out <- dhalfcauchy(x, scale = rate)
  }

  out
}

# set up sampler
k <- length(y)

theta <- matrix(0, nrow = B, ncol = k)

theta[ 1, ] <- theta0

alpha <- matrix(0, nrow = B, ncol = k)

alpha[ 1, ] <- alpha0

acc_alpha <- numeric(ncol(alpha))

beta <- numeric(B)

beta[ 1 ] <- 2

a <- 0; b <- 0 # reduces to non-informative prior for beta

g <- gamma

# run the sampler
# run sampler
for (j in 2:B) {

  # sample theta
  theta[j,] <- rdirichlet(1, y + alpha[j-1,])

  # sample beta
  beta[j] <- rgamma(1, k + a, sum(log(alpha[j-1,]) - k * log(g)) + b)

  # sample alpha
  alpha_star <- j_alpha(n = k)

  r <- (f_alpha_k(alpha_k = alpha_star, theta_k = theta[j-1,], beta = beta[j-1]) /
          f_alpha_k(alpha[j-1,],theta[j-1,],beta[j-1])) /
    (j_alpha(prob = TRUE, x = alpha_star) / j_alpha(prob = TRUE, x = alpha[j-1,]))

  u <- runif(1)

  keep <- pmin(r,1) > u

  alpha[j,keep] <- alpha_star[keep]

  alpha[j,!keep] <- alpha[j-1,!keep]

  # acc_alpha[j] <- sum(keep) / k
```

```
    acc_alpha <- acc_alpha + keep

  }

  acc_alpha <- acc_alpha / B

  # return the result
  list(theta = theta, alpha = alpha, acc_alpha = acc_alpha, beta = beta,
       seed = seed, theta0 = theta0, alpha0 = alpha0)

}
# run 4 chains of 20,000 iterations each

B <- 20000

main_chains <- list(list(seed = 1020,
                         theta0 = y / sum(y),
                         alpha0 = rep(0.01, length(y)),
                         beta0 = 2),
                    list(seed = 6,
                         theta0 = c(100, rep(1, length(y) - 1)) /
                           sum(c(100, rep(1, length(y) - 1))),
                         alpha0 = rep(0.1, length(y)),
                         beta0 = 1),
                    list(seed = 74901,
                         theta0 = rep(1/length(y), length(y)),
                         alpha0 = rep(0.5, length(y)),
                         beta0 = 0.01),
                    list(seed = 481,
                         theta0 = c(rep(100, 100), rep(10, length(y) - 100)) /
                           sum(c(rep(1, 100), rep(0, length(y) - 100))),
                         alpha0 = rep(0.9, length(y)),
                         beta0 = 0.5))

main_chains <- TmParallelApply(main_chains, function(x){
  # run sampler
  result <- main_sampler(y, B, seed = x$seed, theta0 = x$theta0,
                         alpha0 = x$alpha0, beta0 = x$beta0, gamma = 0.01)

  # remove 50% burn-in itertions and check convergence
  result$alpha <- result$alpha[(B/2):B,]

  result$theta <- result$theta[(B/2):B,]

  result$beta <- result$beta[(B/2):B]

  result$geweke_alpha <- apply(result$alpha,2,function(k){
    out <- try(geweke.diag(k)$z)

    if (class(out) == "try-error")
      return(NA)

    out
```

```
  })

  result$geweke_theta <- apply(result$theta,2,function(k){
    out <- try(geweke.diag(k)$z)

    if (class(out) == "try-error")
      return(NA)

    out
  })

  result$geweke_beta <- try(geweke.diag(result$beta)$z)


  # thin every 10th interation
  result$alpha <- result$alpha[seq(10, nrow(result$alpha), by=10),]

  result$theta <- result$theta[seq(10, nrow(result$theta), by=10),]

  result$beta <- result$beta[seq(10, length(result$beta), 10)]

  # return full result
  result

}, cpus = 2, export = c("y", "B", "main_sampler"),
   libraries = c("gtools", "EnvStats", "coda", "statmod"))

# check acceptance rate
main_alpha_acc <- sapply(main_chains, function(x) mean(x$acc_alpha))

# check convergence with Geweke statistic
main_theta_conv <- sapply(main_chains, function(x){
  mean(abs(x$geweke_theta) >= 1.96 | is.na(x$geweke_theta))
})

main_alpha_conv <- sapply(main_chains, function(x){
  mean(abs(x$geweke_alpha) >= 1.96 | is.na(x$geweke_alpha))
})

main_beta_conv <- sapply(main_chains, function(x) x$geweke_beta)

save.image('.RData')
# Combine the results from the chains
control_posterior <- list(theta = do.call(rbind, lapply(control_chains, function(x) x$theta)),
                          alpha = do.call(rbind, lapply(control_chains, function(x) x$alpha)))

main_posterior <- list(theta = do.call(rbind, lapply(main_chains, function(x) x$theta)),
                       alpha = do.call(rbind, lapply(main_chains, function(x) x$alpha)),
                       beta = do.call(c, lapply(main_chains, function(x) x$beta)))

# check acceptance rates
main_alpha_acc <- sapply(main_chains, function(x) x$acc_alpha)
```

```r
control_alpha_acc <- sapply(control_chains, function(x) x$acc_alpha)

alpha_acc_table <- data.frame(Main = quantile(rowMeans(main_alpha_acc), c(0,0.25,0.5,0.75,1)),
                              Control = quantile(rowMeans(control_alpha_acc), c(0,0.25,0.5,0.75,1)),
                              stringsAsFactors = FALSE)

rownames(alpha_acc_table) <- c("min.", "25%", "50%", "75%", "max.")

kable_styling(kable(alpha_acc_table, format = "latex", caption = "Acceptance Rates", digits = 2),positi

conv <- parallel::mclapply(list(main = main_posterior, control = control_posterior),
                           function(x){
                             apply(x$theta,1,function(z) dmultinom(y, prob = z, log = T))
                           })

par(mfrow = c(2,1), mar = c(2.1,4.1,2.1,2.1))
g <- lapply(conv, geweke.diag)

g <- data.frame(Main = g[[1]]$z, Control = g[[2]]$z)

rownames(g) <- ""

kable(data.frame(g),digits = 2, format = 'latex', caption = "Geweke statistic of log likelihoods")

# log posterior comparison
main_p <- apply(main_posterior$theta,1,function(x) dmultinom(y, prob = x, log = TRUE))

control_p <- apply(control_posterior$theta,1,function(x) dmultinom(y, prob = x, log = TRUE))

d1 <- density(main_p)
d2 <- density(control_p)

plot(d1, col = "red", lwd = 4,
     main = "Posterior Log Likelihood",
     xlim = range(c(d1$x, d2$x)))
lines(d2, col = "blue", lwd = 4, lty = 2)

# calculate DIC for each model
calc_dic <- function(y, theta_mat, B = NULL) {

  llik <- apply(theta_mat, 1, function(x) dmultinom(y, prob = x, log = TRUE))

  pdic <- 2 * var(llik)


  if (! is.null(B)) {
    dic <- sapply(seq_len(B), function(th){
      -2 * dmultinom(y, prob = theta_mat[ sample(seq_len(nrow(theta_mat)), 1) , ], log = TRUE)
    })

    dic <- dic + 2 * pdic
  } else {
    dic <- -2 * dmultinom(y, prob = colMeans(theta_mat), log = TRUE) + 2 * pdic
```

```
  }

  dic

}

main_dic <- calc_dic(y, main_posterior$theta, B = 10000)

control_dic <- calc_dic(y, control_posterior$theta, B = 10000)

d1 <- density(main_dic)

d2 <- density(control_dic)

plot(d1, col = "red", lwd = 4,
     main = "Posterior DIC",
     xlim = range(c(d1$x, d2$x)))
lines(d2, col = "blue", lwd = 4, lty = 2)
abline(v = quantile(main_dic, probs = c(0.025, 0.5, 0.975)), col = "red", lwd = 2)
abline(v = quantile(control_dic, probs = c(0.025, 0.5, 0.975)), col = "blue", lwd = 2, lty = 2)


# barplot(c(Main = main_dic, Control = control_dic), col = c("red", "blue"),
#         density = c(-1,25))

# log-log plots of alpha and theta vs. y

# alpha
plot(log10(seq_along(y)), log10(y),
     lwd = 3, yaxt = "n", ylab = "log10(x)", xlab = "rank(x)", type = "l",
     lty = 3, main = expression(paste("Comparing ", alpha)))
par(new = TRUE)
plot(log10(seq_along(y)), log10(colMeans(main_posterior$alpha)),
     lwd = 3, yaxt = "n", ylab = "", xaxt = "n", xlab = "", type = "l", lty = 1,
     col = rgb(1,0,0,0.5))
par(new = TRUE)
plot(log10(seq_along(y)), log10(colMeans(control_posterior$alpha)),
     lwd = 3, yaxt = "n", ylab = "", xaxt = "n", xlab = "", type = "l", lty = 2,
     col = rgb(0,0,1,0.5))
legend("bottomleft", legend = c("y", "Main", "Control"),
       lwd = 3, lty = c(3,1,2), col = c("black", "red", "blue"))

# theta
plot(log10(seq_along(y)), log10(y),
     lwd = 3, yaxt = "n", ylab = "log10(x)", xlab = "rank(x)", type = "l",
     lty = 3, main = expression(paste("Comparing ", theta)))
par(new = TRUE)
plot(log10(seq_along(y)), log10(colMeans(main_posterior$theta)),
     lwd = 3, yaxt = "n", ylab = "", xaxt = "n", xlab = "", type = "l", lty = 1,
     col = rgb(1,0,0,0.5))
par(new = TRUE)
plot(log10(seq_along(y)), log10(colMeans(control_posterior$theta)),
     lwd = 3, yaxt = "n", ylab = "", xaxt = "n", xlab = "", type = "l", lty = 2,
```

```
     col = rgb(0,0,1,0.5))
legend("bottomleft", legend = c("y", "Main", "Control"),
       lwd = 3, lty = c(3,1,2), col = c("black", "red", "blue"))
# MCMC plots

# get indices of max, median, 3rd quartile words
q <- quantile(y, c(1,0.95, 0.75))

ind <- sapply(q, function(x) which(y == x)[1])

# look at ACF/mcmcplot

capture <- lapply(ind, function(k){
  a <- matrix(main_posterior$alpha[,k], ncol = 1)
  colnames(a) <- "alpha"
  mcmcplot1(a, greek = TRUE, col = "red")

  a <- matrix(control_posterior$alpha[,k], ncol = 1)
  colnames(a) <- "alpha"
  mcmcplot1(a, greek = TRUE)

})

capture <- lapply(ind, function(k){
  a <- matrix(main_posterior$theta[,k], ncol = 1)
  colnames(a) <- "theta"
  mcmcplot1(a, greek = TRUE, col = "red")

  a <- matrix(control_posterior$theta[,k], ncol = 1)
  colnames(a) <- "theta"
  mcmcplot1(a, greek = TRUE)
})

b <- matrix(main_posterior$beta, ncol = 1)
colnames(b) <- "beta"
mcmcplot1(b, greek = TRUE, col = "red")

# check convergence
plot(conv$main, type = "l",
     main = "Main model", col = "red", xaxt = "n", xlab = "",
     ylab = expression(paste("log[P(y|",theta,")]")))
abline(v = c(1000,2000,3000))
plot(conv$control, type = "l",
     main = "Control model", col = "blue", xaxt = "n", xlab = "",
     ylab = expression(paste("log[P(y|",theta,")]")))
abline(v = c(1000,2000,3000))
```

7 Appendix B

7.1 Control Posterior Derivations

$$P(\vec{\theta}, \vec{\alpha}|\vec{y}) \propto \left[\prod_k \theta_k^{y_k}\right]\left[\mathcal{B}(\vec{\alpha})\prod_k \theta_k^{\alpha_k - 1}\right] \times 1 \tag{11}$$

$$= \left[\prod_k \theta_k^{y_k}\right]\left[\mathcal{B}(\vec{\alpha})\prod_k \theta_k^{\alpha_k - 1}\right] \tag{12}$$

$$P(\vec{\theta}|\vec{\alpha}, \vec{y}) \propto \prod_k \theta_k^{y_k + \alpha_k - 1} \tag{13}$$

$$\implies \vec{\theta}|\vec{\alpha}, \vec{y} \sim Dir(\vec{y} + \vec{\alpha}) \tag{14}$$

$$P(\vec{\alpha}|\vec{\theta}, \vec{y}) \propto \mathcal{B}(\vec{\alpha})\prod_k \theta_k^{\alpha_k} \tag{15}$$

$$P(\alpha_k|\theta_k, y_k) \propto \theta_k^{\alpha_k} \tag{16}$$

7.2 Main Posterior Derivations

$$P(\vec{\theta}, \vec{\alpha}, \beta|\vec{y}) \propto \left[\prod_k \theta_k^{y_k}\right]\left[\mathcal{B}(\vec{\alpha})\prod_k \theta_k^{\alpha_k - 1}\right]\left[\prod_k \gamma^\beta \beta \alpha_k^{-(\beta+1)}\right] \tag{17}$$

$$= \beta^{K-1}\gamma^{\beta k}\mathcal{B}(\vec{\alpha})\prod_k \theta_k^{y_k + \alpha_k - 1}\alpha_k^{-(\beta+1)} \tag{18}$$

$$P(\vec{\theta}|\vec{\alpha}, \beta, \vec{y}) \propto \prod_k \theta_k^{y_k + \alpha_k - 1} \tag{19}$$

$$\implies \vec{\theta}|\vec{\alpha}, \beta, y \sim Dir(\vec{y} + \vec{\alpha}) \tag{20}$$

$$P(\vec{\alpha}|\vec{\theta}, \beta, \vec{y}) \propto \mathcal{B}(\vec{\alpha})\prod_k \theta_k^{y_k + \alpha_k - 1}\alpha_k^{-(\beta+1)} \tag{21}$$

$$\implies \text{unknown distribution} \tag{22}$$

$$P(\beta|\vec{\theta}, \vec{\alpha}, \vec{y}) \propto \beta^{K-1}\gamma^{\beta k}(\prod_k \alpha_k)^{-(\beta+1)} \tag{23}$$

$$\propto \beta^{K-1}\gamma^{\beta k}(\prod_k \alpha_k)^{-\beta} \tag{24}$$

$$\propto \beta^{K-1}\exp\left[-\beta\left(\sum_k \log(\alpha_k) - k\log(\gamma)\right)\right] \tag{25}$$

$$\implies \beta|\vec{\theta}, \vec{\alpha}, \vec{y} \sim Gamma\Big(k, \sum_k \log(\alpha_k) - k\log(\gamma)\Big) \tag{26}$$

7.3 Jeffrey's prior on β

$$p(\beta) \propto \prod_k \beta \alpha_k^{-(\beta+1)} \tag{27}$$

$$p(\beta) \propto \beta^k (\prod_k \alpha_k)^{-(\beta+1)} \tag{28}$$

$$\log\big(p(\beta)\big) \propto k \log(\beta) - \beta \log(\prod_k \alpha_k) - \log(\prod_k \alpha_k) \tag{29}$$

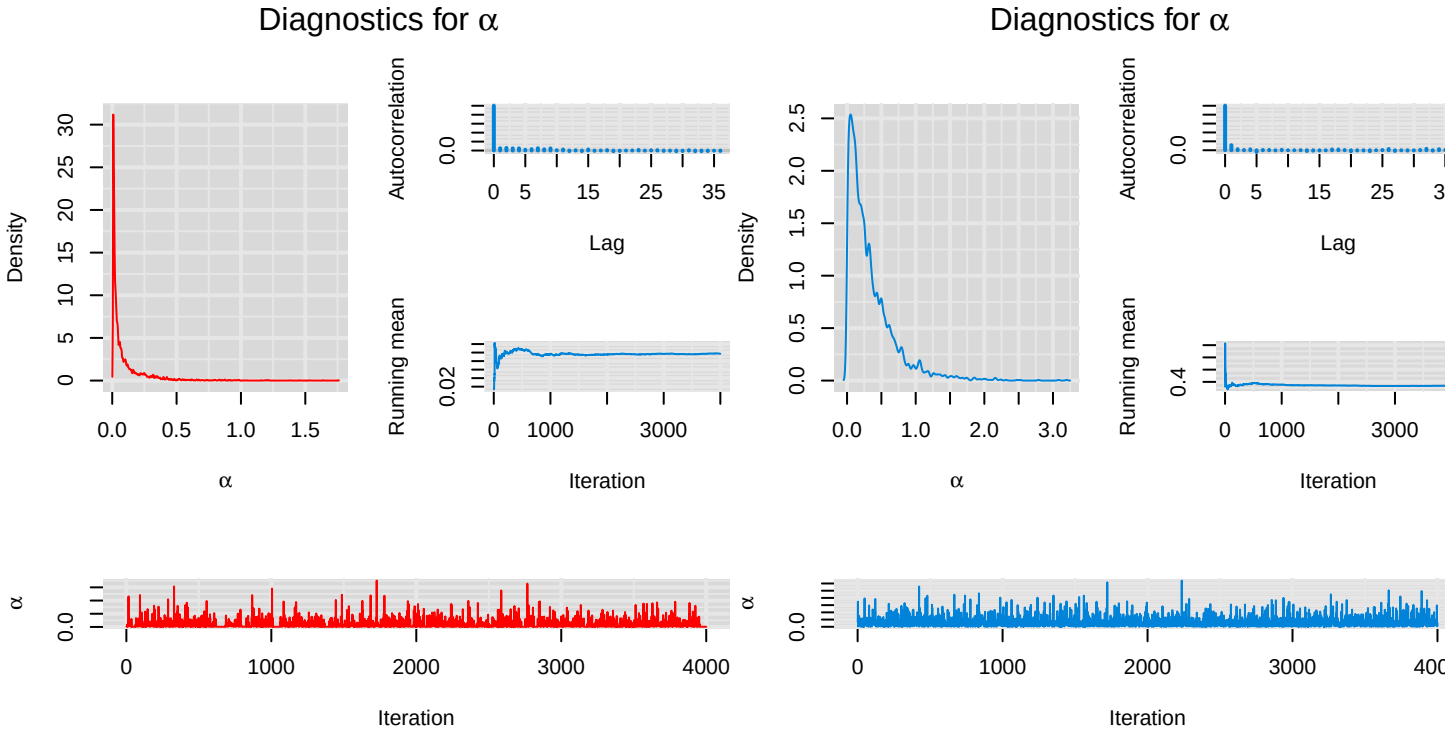
$$\frac{\partial}{\partial \beta} \log\big(p(\beta)\big) \propto \frac{k}{\beta} - \log(\prod_k \alpha_k) \tag{30}$$

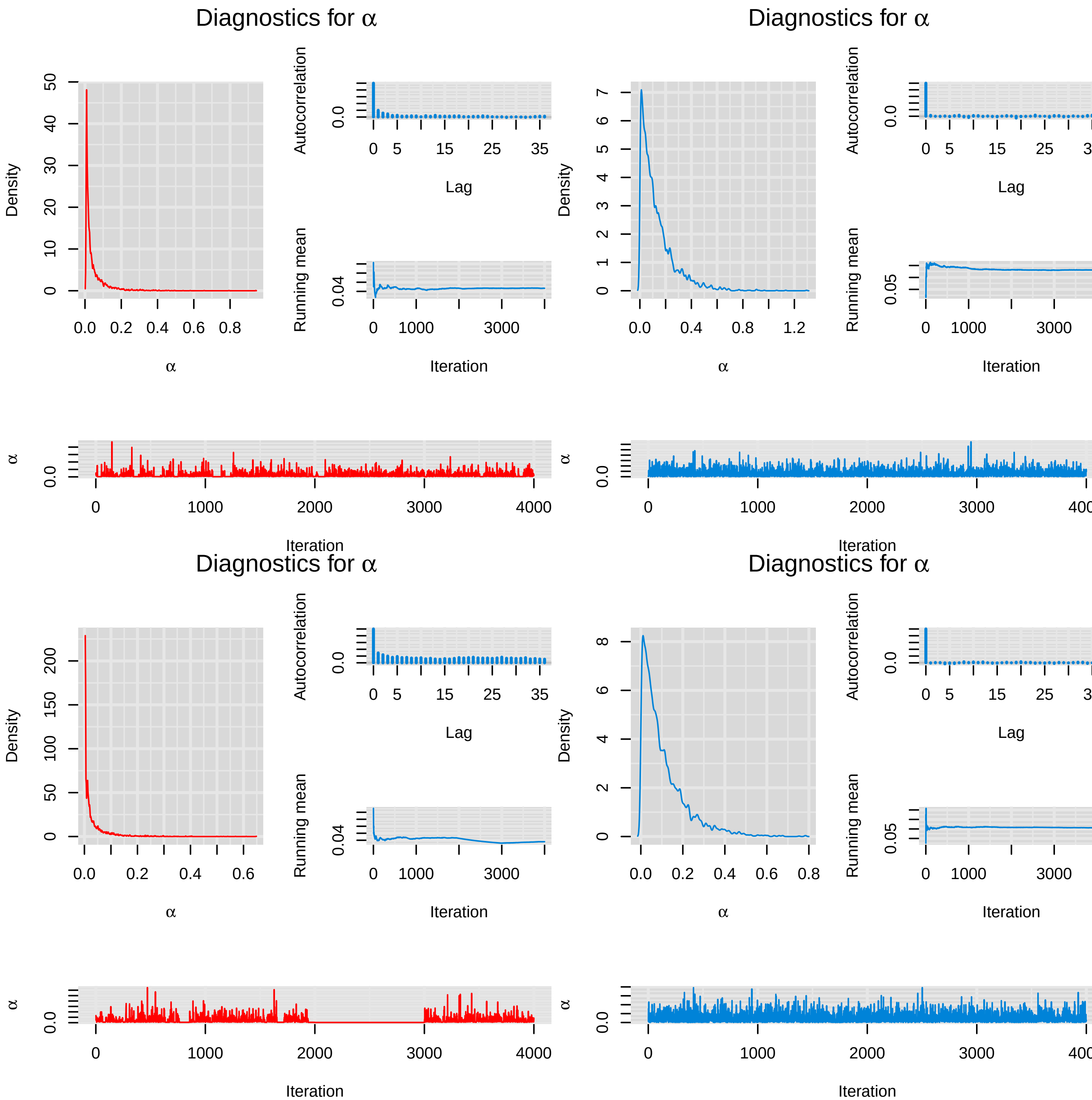
$$\frac{\partial^2}{\partial \beta^2} \log\big(p(\beta)\big) \propto \frac{-k}{\beta^2} \tag{31}$$

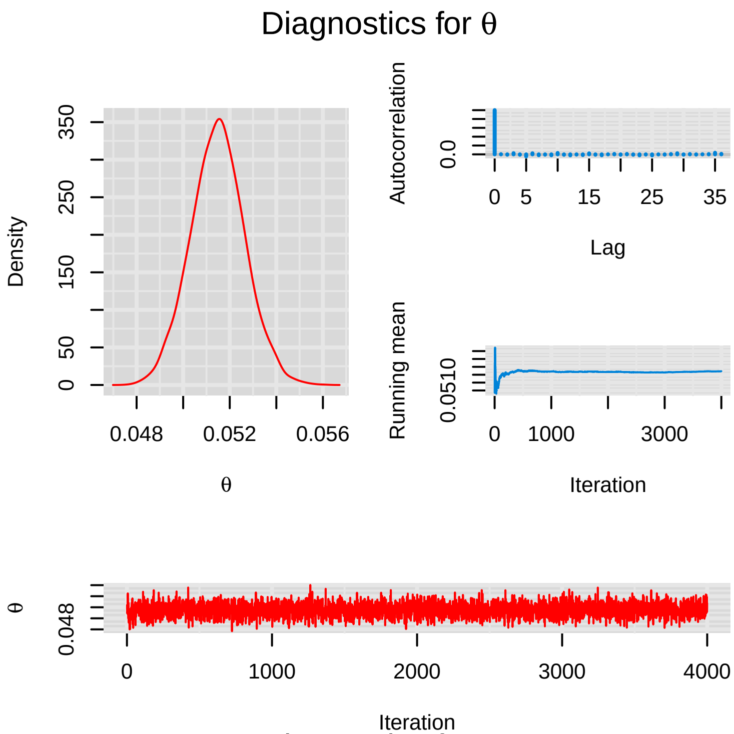
$$-E\Big[\frac{\partial^2}{\partial \beta^2} \log\big(p(\beta)\big)\Big] \propto \frac{k}{\beta^2} \tag{32}$$

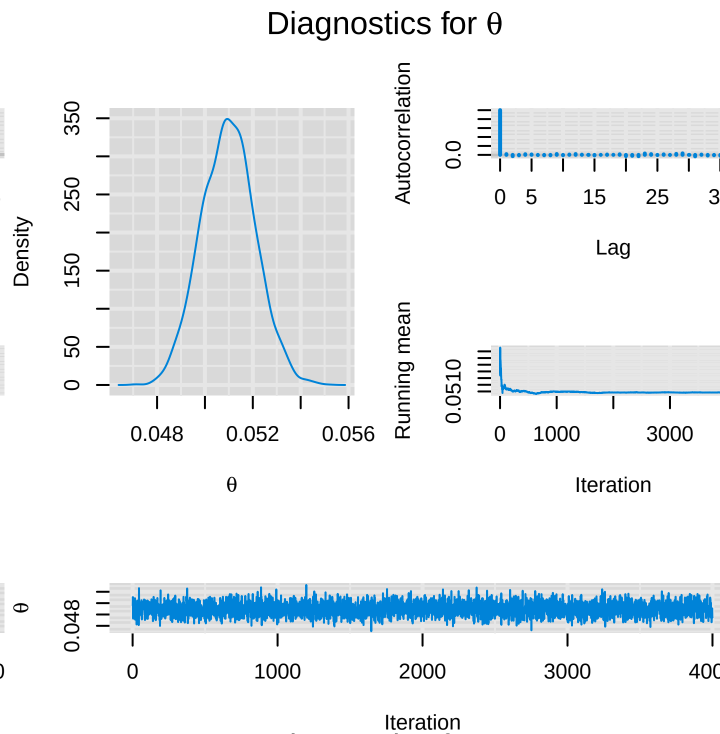
$$\pi(\beta) \propto \frac{1}{\beta} \tag{33}$$

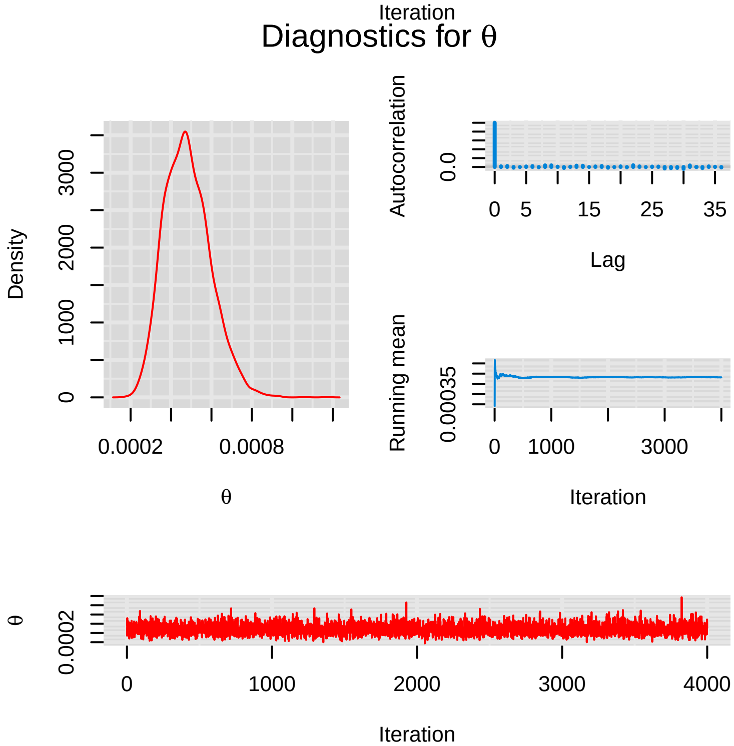
8 Appendix C

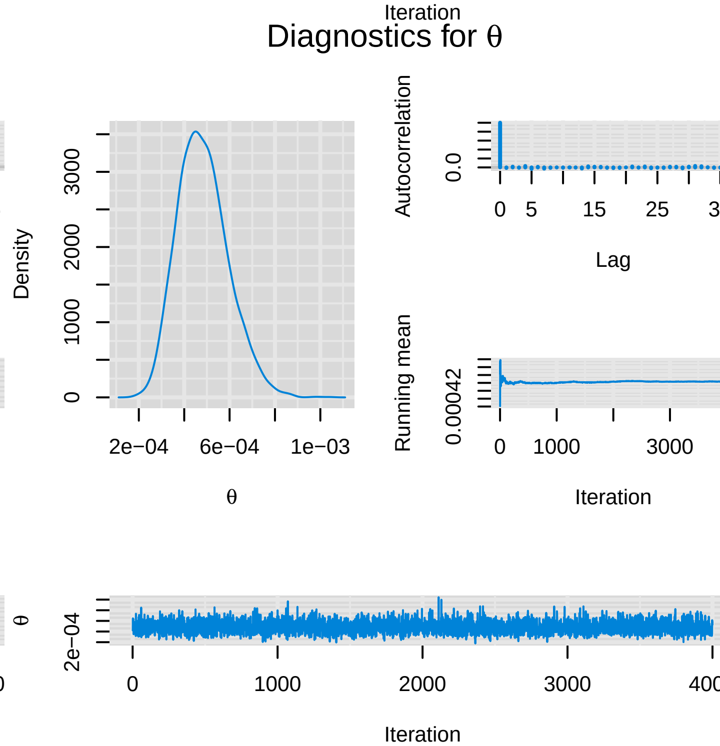


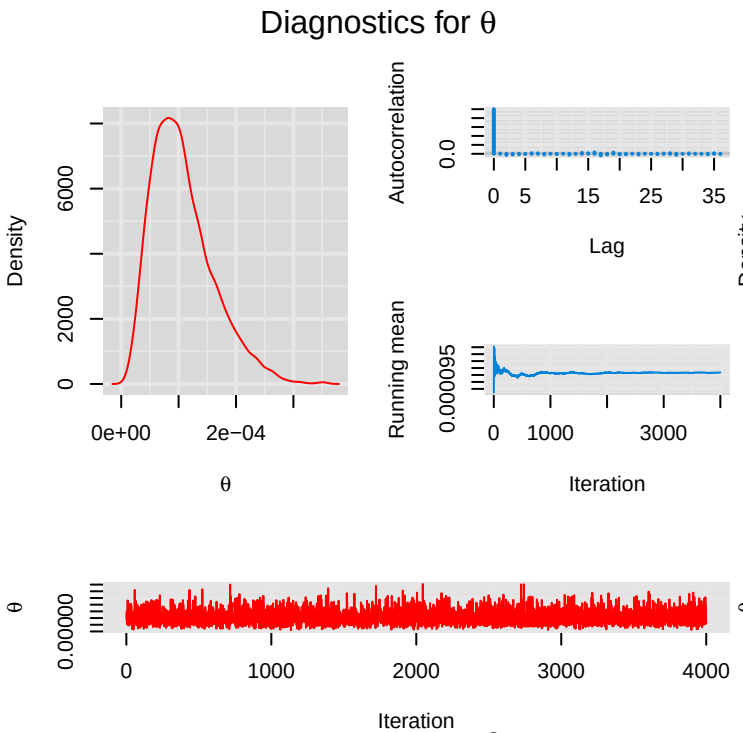

Diagnostics for α
Density
α
Autocorrelation
Lag
Running mean
Iteration
Diagnostics for α
Density
α
Autocorrelation
Lag
Running mean
Iteration
Diagnostics for α
Density
α
Autocorrelation
Lag
Running mean
Iteration
Diagnostics for α
Density
α
Autocorrelation
Lag
Running mean
Iteration

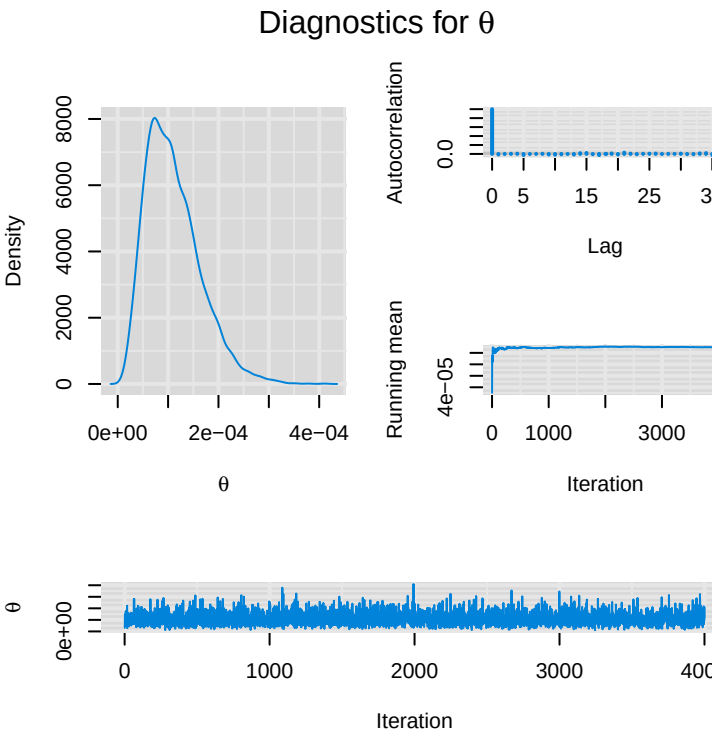

Diagnostics for θ
Density
θ
0.048
0.052
0.056
Autocorrelation
Lag
Running mean
Iteration
Iteration

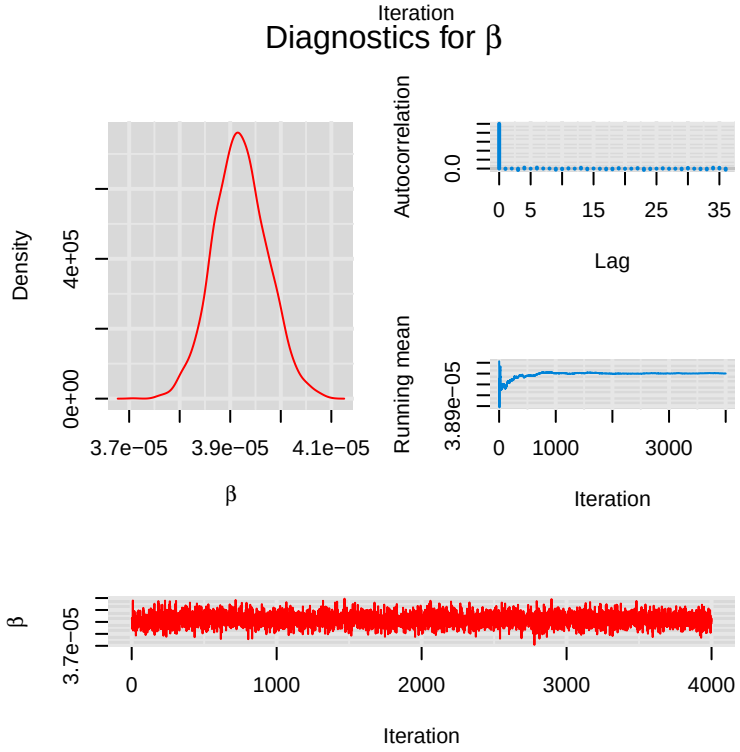

Diagnostics for θ
Density
θ
0.048
0.052
0.056
Autocorrelation
Lag
Running mean
Iteration
Iteration

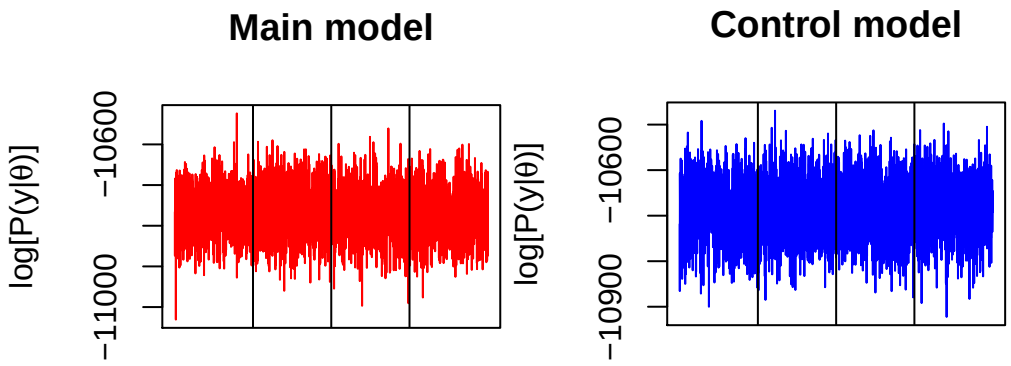

Diagnostics for θ
Density
θ
0.0002
0.0008
Autocorrelation
Lag
Running mean
Iteration
Iteration


Diagnostics for θ
Density
θ
2e−04
6e−04
1e−03
Autocorrelation
Lag
Running mean
Iteration
Iteration


Diagnostics for θ
Density
θ
Autocorrelation
Lag
Running mean
Iteration
θ
Iteration


Diagnostics for θ
Density
θ
Autocorrelation
Lag
Running mean
Iteration
θ
Iteration


Diagnostics for β
Density
β
Autocorrelation
Lag
Running mean
Iteration
β
Iteration


Main model
Control model
log[P(y|θ)]
−10600
−11000
log[P(y|θ)]
−10600
−10900