Math 640 Final Paper

Max Kearns and Tommy Jones

May 10, 2018

1 Introduction

The analysis of text data is an area of vital research in both frequentist and Bayesian statistics. Text can, and indeed does, store a vast amount of information that is not easily evaluated with well-understood statistical methods. While text analysis is used throughout our economy, it does not have nearly as much research and knowledge behind it as does numerical data. This paper attempts to slightly further the bank of techniques for text analysis, in the hopes that text data will someday be as understood as numerical data is today.

One method that researchers currently use to model text frequencies is with a Dirichlet prior on a Multinomial likelihood. The prior provides uncertainty on $\vec{\theta}$, which is the vector of word probabilities. The usual model then assumes a non-informative uniform prior on the Dirichlet parameter ($\vec{\alpha}$). In a Bayesian setting, however, this approach seems overly simplistic, and MCMC methods provide a simple solution to sample from a more complex distribution. This research intends to start to answer the question as to whether more uncertainty on $\vec{\alpha}$ would improve the model. Zipf's law provides a basis for how to vary $\vec{\alpha}$ in a way that is consistent with knowledge about human language.

Zipf's law is an empirical property of natural language. It states that the word frequencies of any corpus of text follows a power law distribution, regardless of context or language. This means that 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. (citations needed) Based on what Zipf's law dictates, this research tests the viability of placing a Pareto(γ, β) prior on $\vec{\alpha}$, where γ is fixed at a sufficiently small value. This 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 will begin by modeling a small data set using a simple model that features a multinomial likelihood, a Dirichlet prior, a Pareto hyper-prior, with a non-informative Jeffrey's hyper-prior . We will compare this model to a control that does not allow for uncertainty on $\vec{\alpha}$. The data set is 100 randomly sampled NIH grant abstracts from 2014.

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 will stop here, and assume a uniform distibution on 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 $\vec{\alpha}|\vec{y}, \vec{\beta}$ 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 Jeffrey'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 unknown posterior below, and a full derivation of the model can be found in Appendix B.

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

This prior is unrecognizable, so we will proceed by making a Gibbs sampler of the full conditional posteriors, which can be found 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 $\text{Gamma}\left(k, \sum_k \log(\alpha_k) - k\log(\gamma)\right)$. The other two conditionals will be sampled using a Metropolis-Hastings algorithm with another Inverse-Gaussian proposal.

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 1 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 1 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.

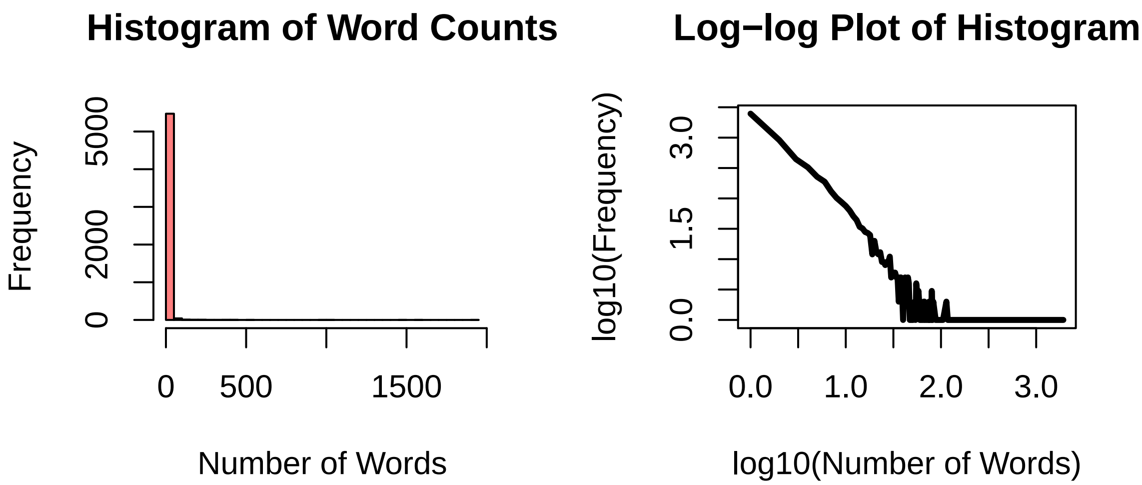


Figure 1:Histogram of Word counts (left) and Log-log plot of word counts, ordered by rank (right).

2.2.2 Samplers

As discussed above, to sample from these conditional distributions, we had to create Gibb's samplers using metropolis-hastings components. For both the control and the man model, we used an Inverse-Gaussian to sample from the condtional posterior on α, and sampled directly from the conditional posteriors of the other paramters. This is where

Table 1: Table 1: Acceptance Rates

	Main	Control
min.	0.1009625	0.3991625
25%	0.1843125	0.5299656
50%	0.1996875	0.5485250
75%	0.2156688	0.5849531
max.	0.3779250	0.6833500

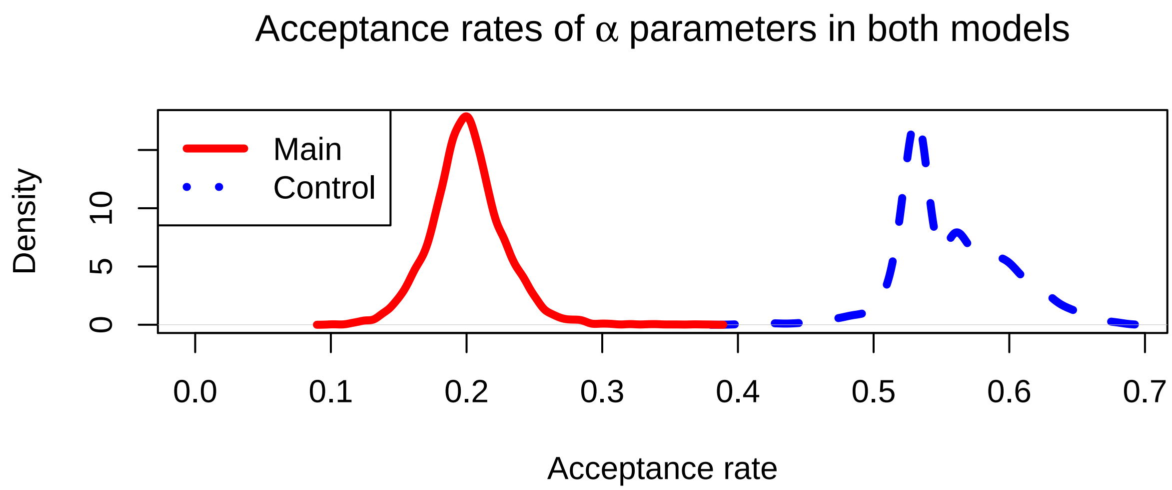


Figure 2: Plot of acceptance rates of α.

Convergence is based on log likelihood of both models

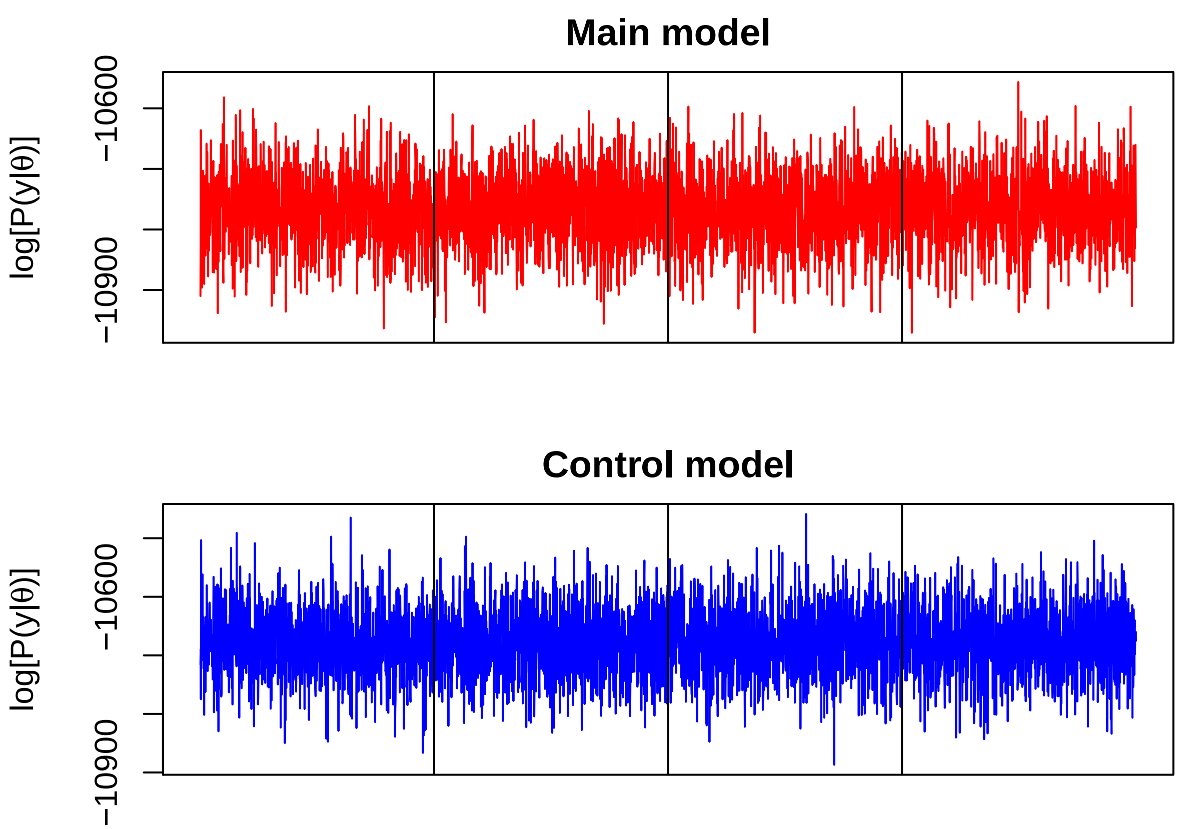


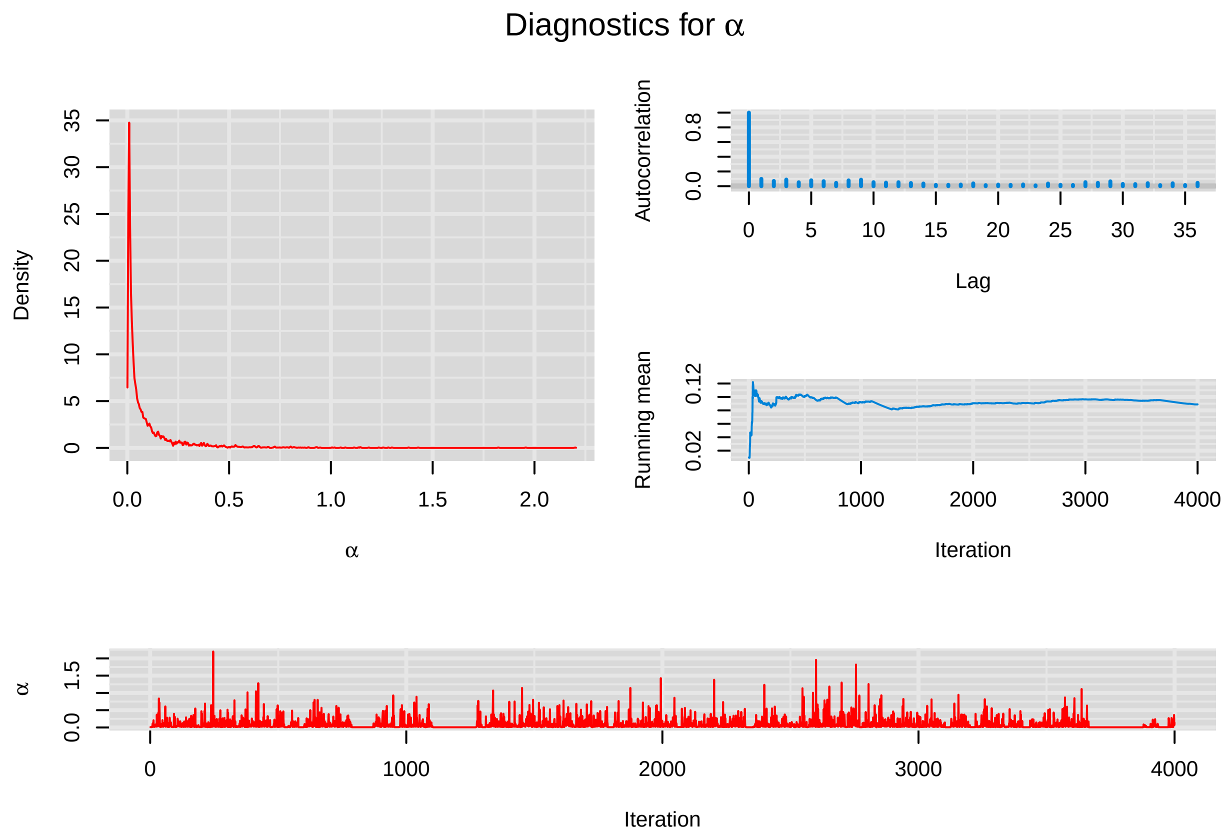
Table 2: Geweke statistic of log likelihoods

Main	Control
1.63	0.53

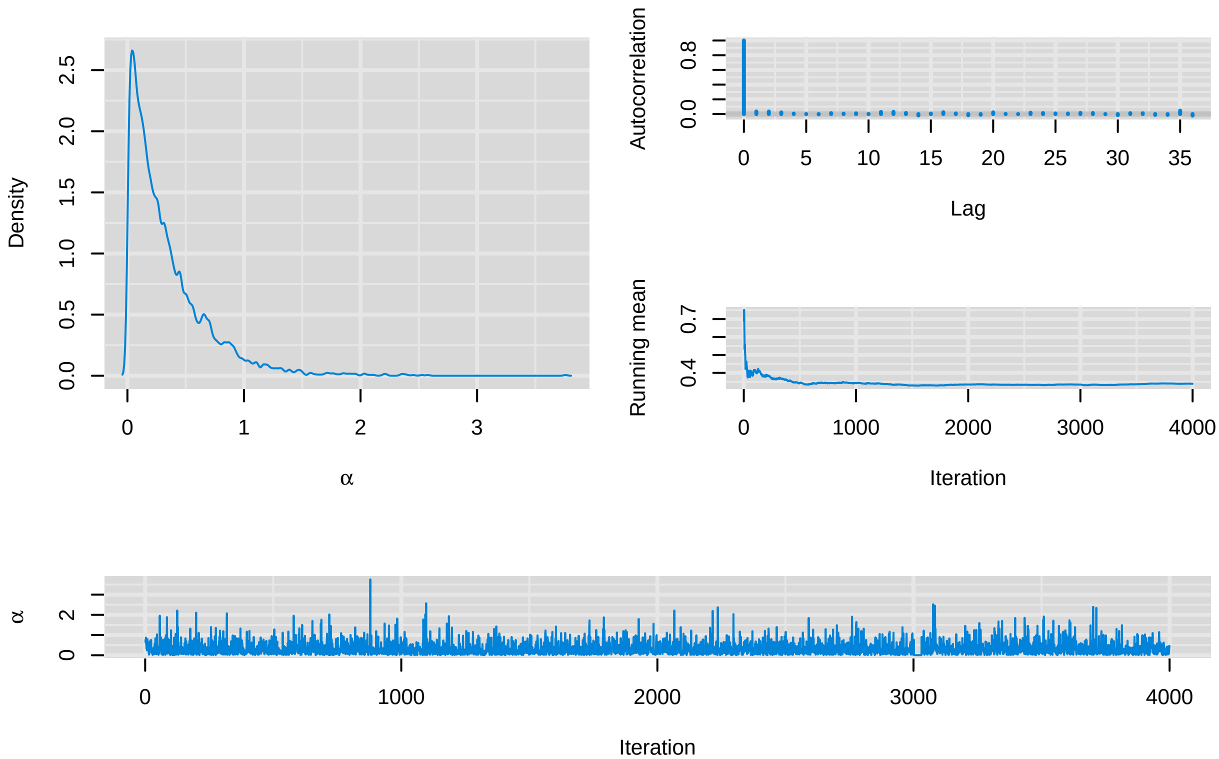
	Main	Control
##Compa		rison

We should put some posterior plots here, maybe

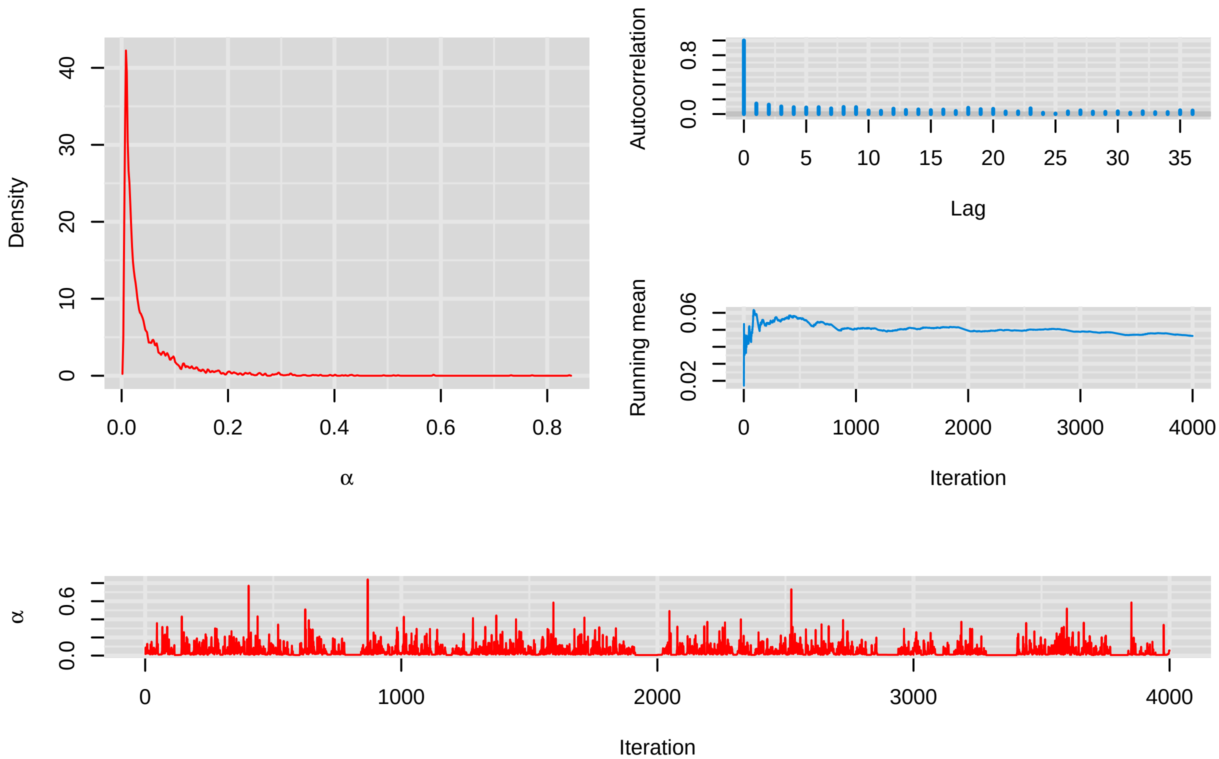
MCMC plots of selected words

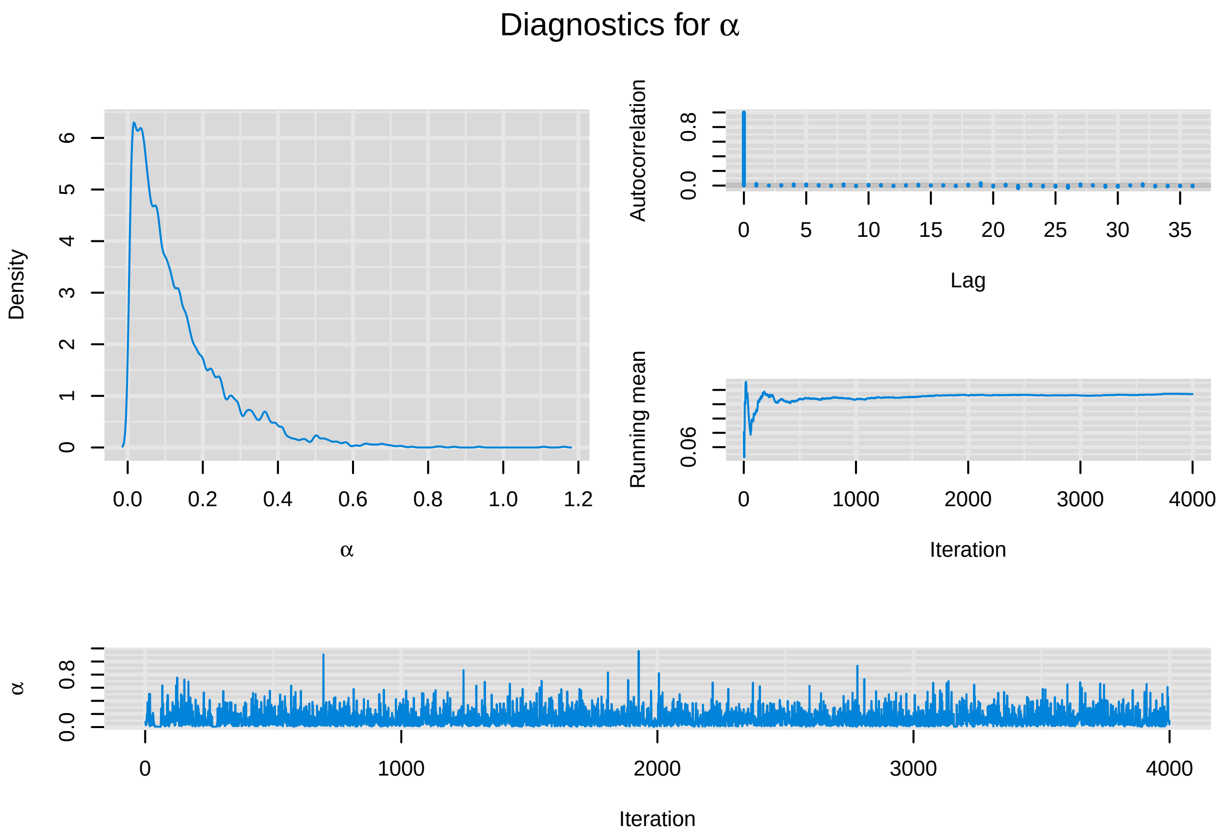


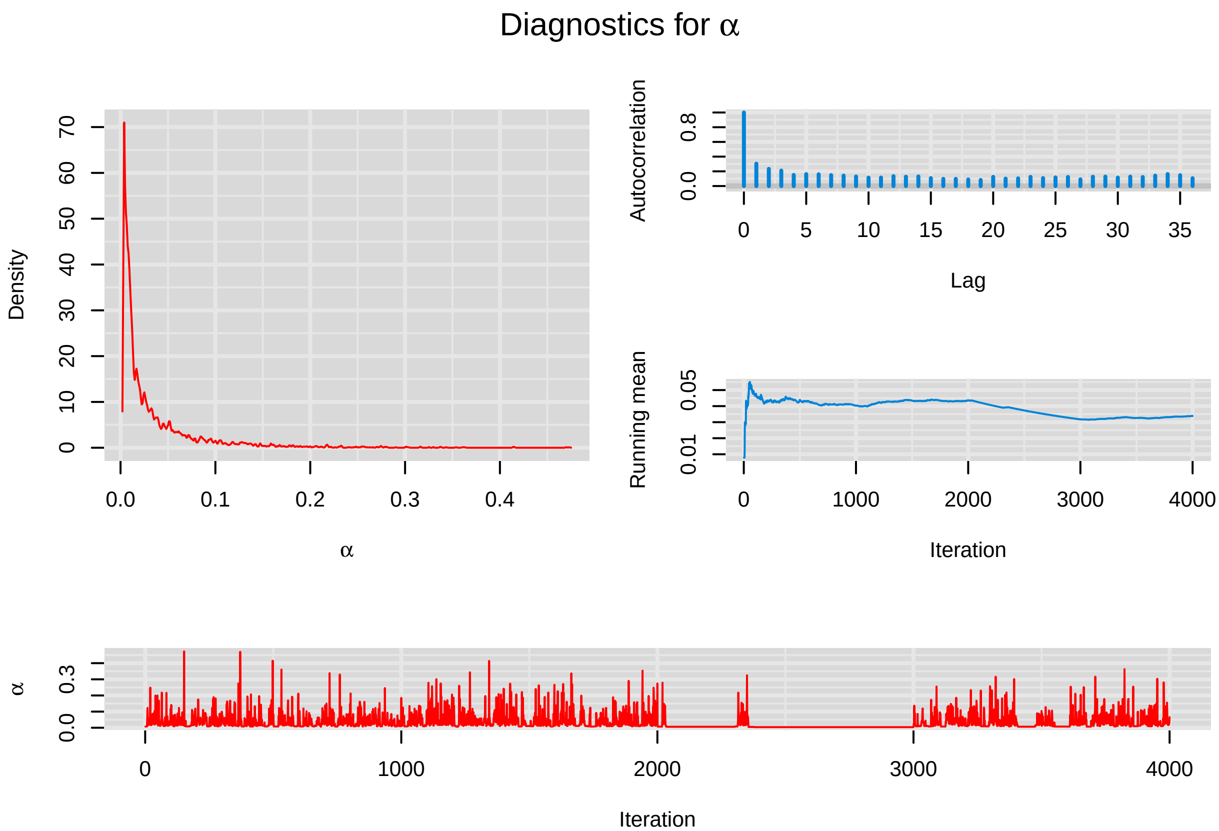
Diagnostics for α



Diagnostics for α

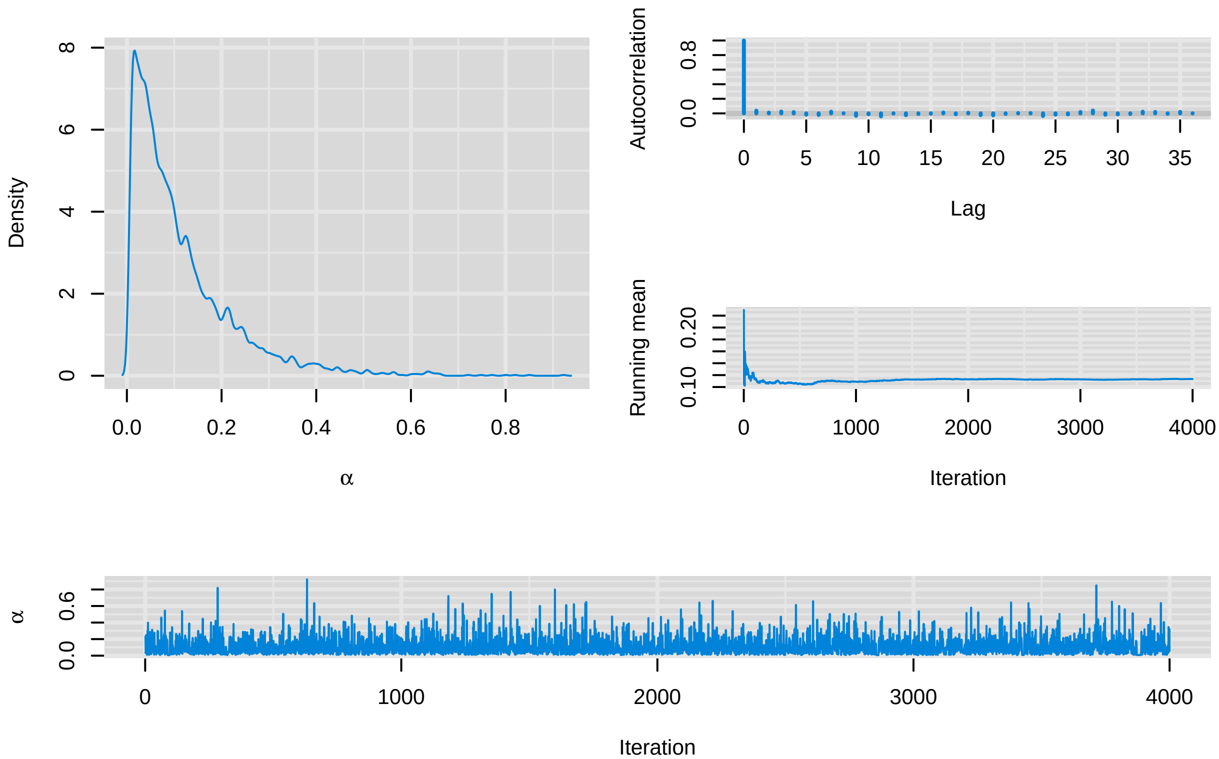


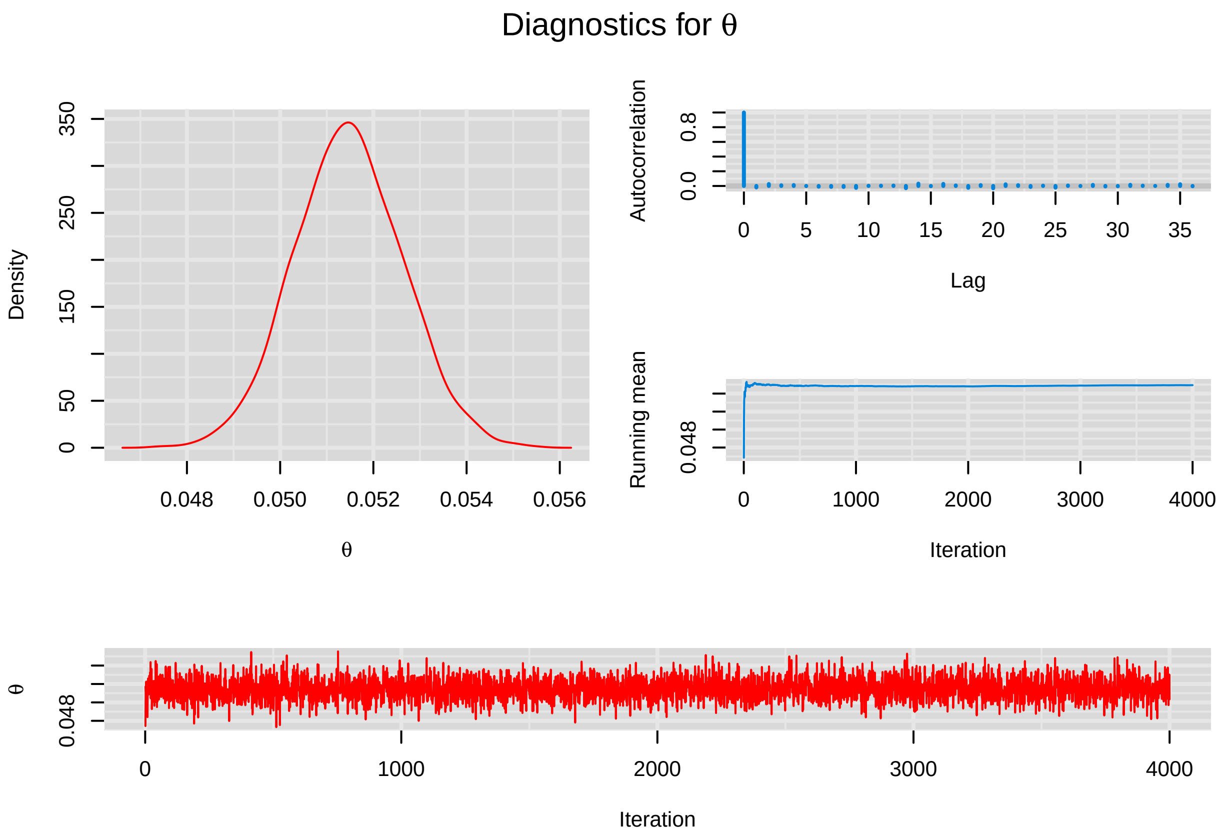

Diagnostics for α
Density
α
Autocorrelation
Lag
Running mean
Iteration
α
Iteration

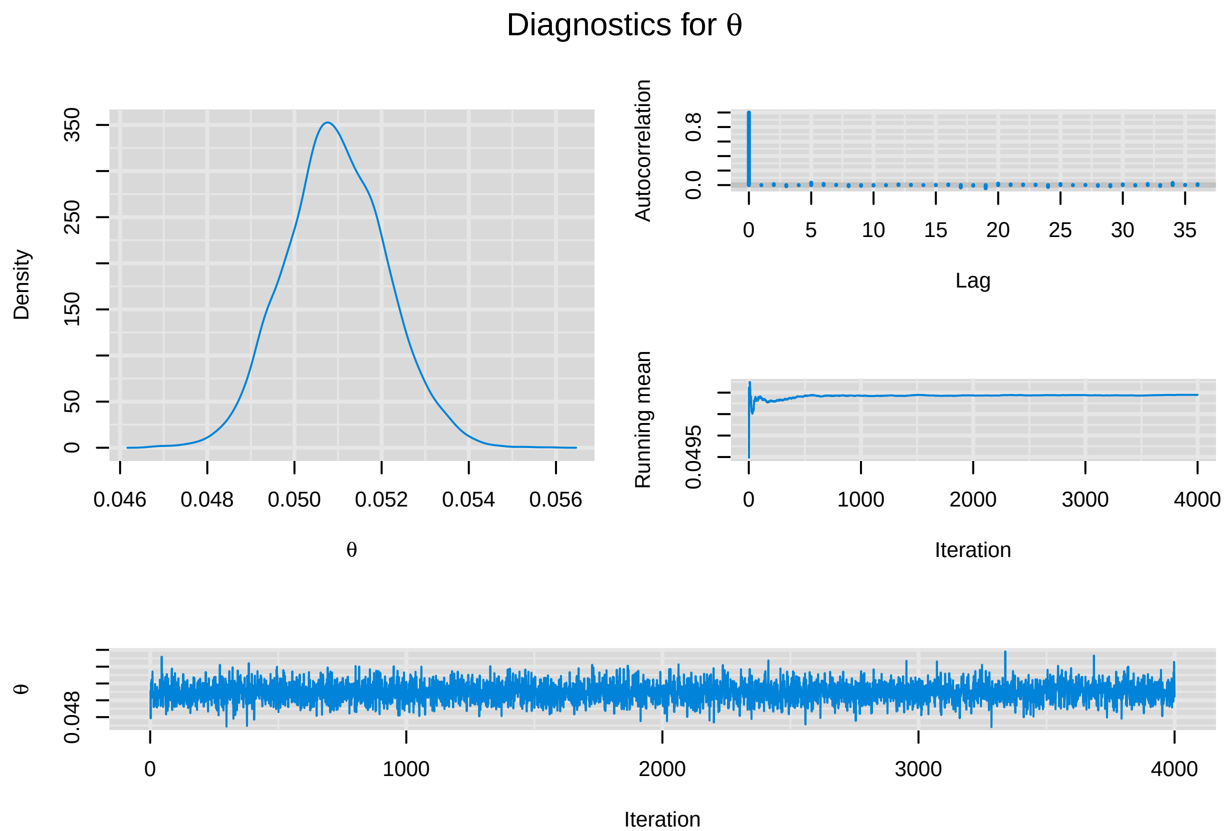


Diagnostics for α
Density
α
Autocorrelation
Lag
Running mean
Iteration
α
Iteration

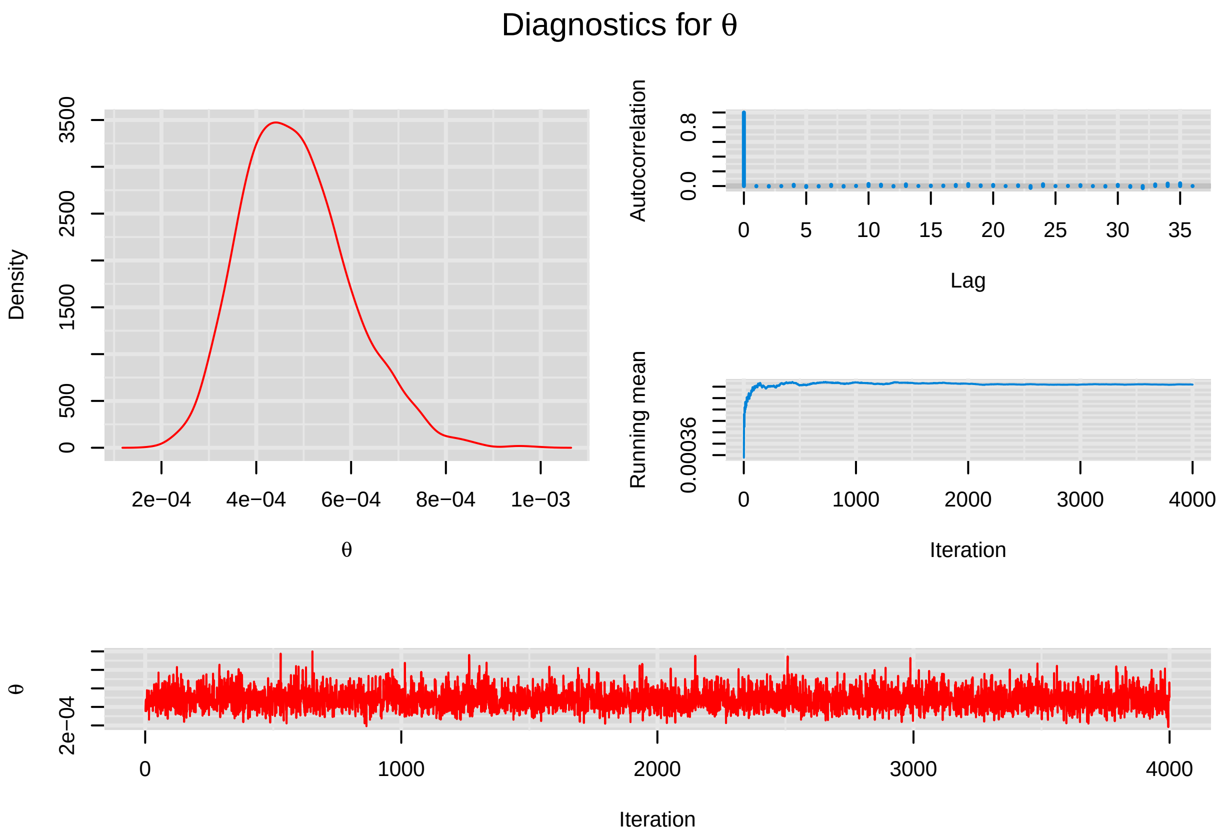
Diagnostics for α




Diagnostics for θ
Density
350
250
150
50
0
0.048
0.050
0.052
0.054
0.056
θ
Autocorrelation
0.8
0.0
0
5
10
15
20
25
30
35
Lag
Running mean
0.048
0
1000
2000
3000
4000
Iteration
θ
0.048
0
1000
2000
3000
4000
Iteration

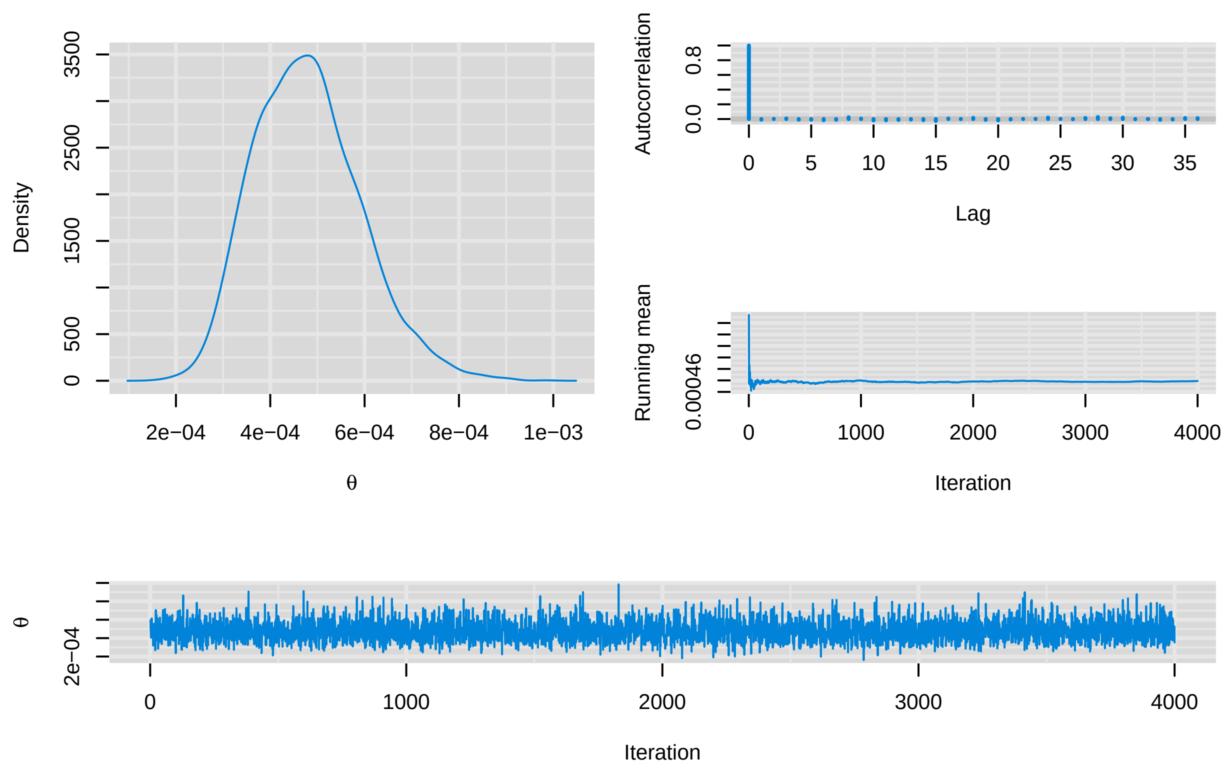


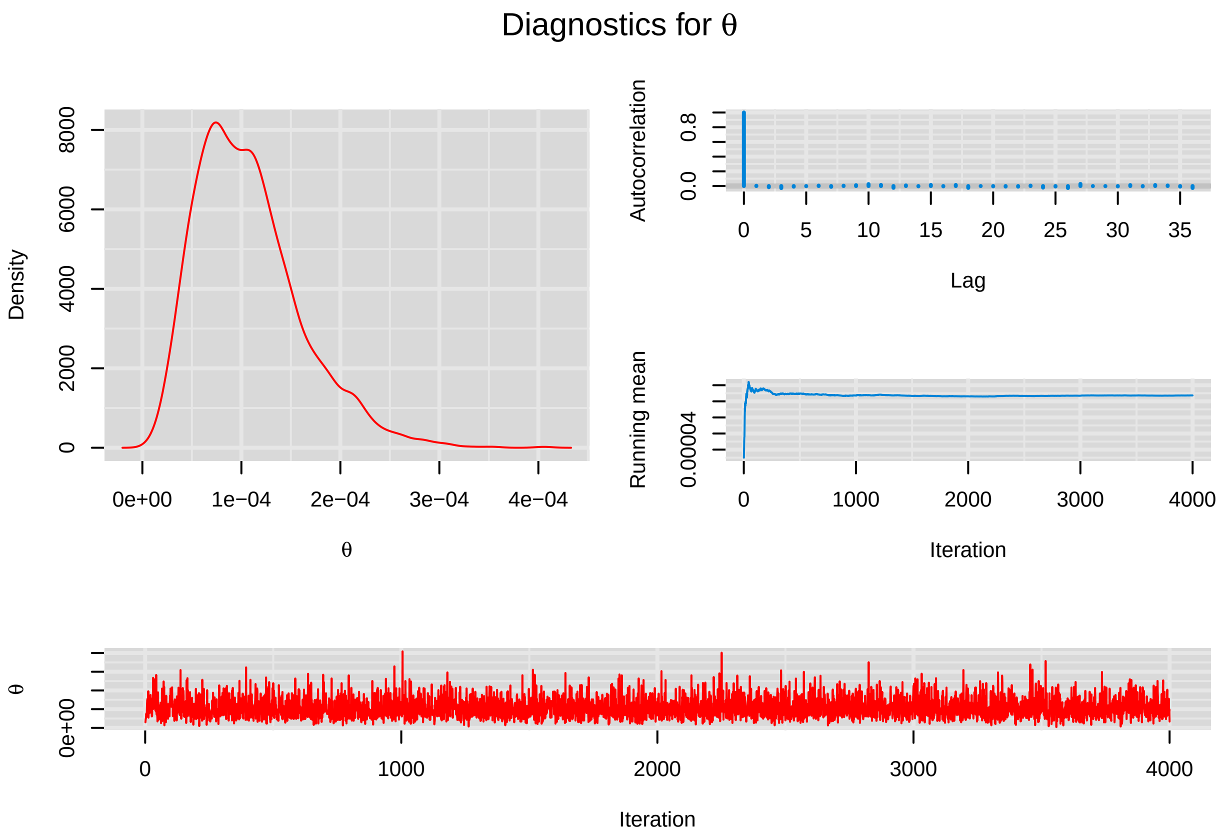
Diagnostics for θ
Density
350
250
150
50
0
0.046
0.048
0.050
0.052
0.054
0.056
θ
Autocorrelation
0.8
0.0
0
5
10
15
20
25
30
35
Lag
Running mean
0.0495
0
1000
2000
3000
4000
Iteration
θ
0.048
0
1000
2000
3000
4000
Iteration



Diagnostics for θ
Density
3500
2500
1500
500
0
2e−04
4e−04
6e−04
8e−04
1e−03
θ
Autocorrelation
0.8
0.0
0
5
10
15
20
25
30
35
Lag
Running mean
0.00036
0
1000
2000
3000
4000
Iteration
θ
2e−04
0
1000
2000
3000
4000
Iteration

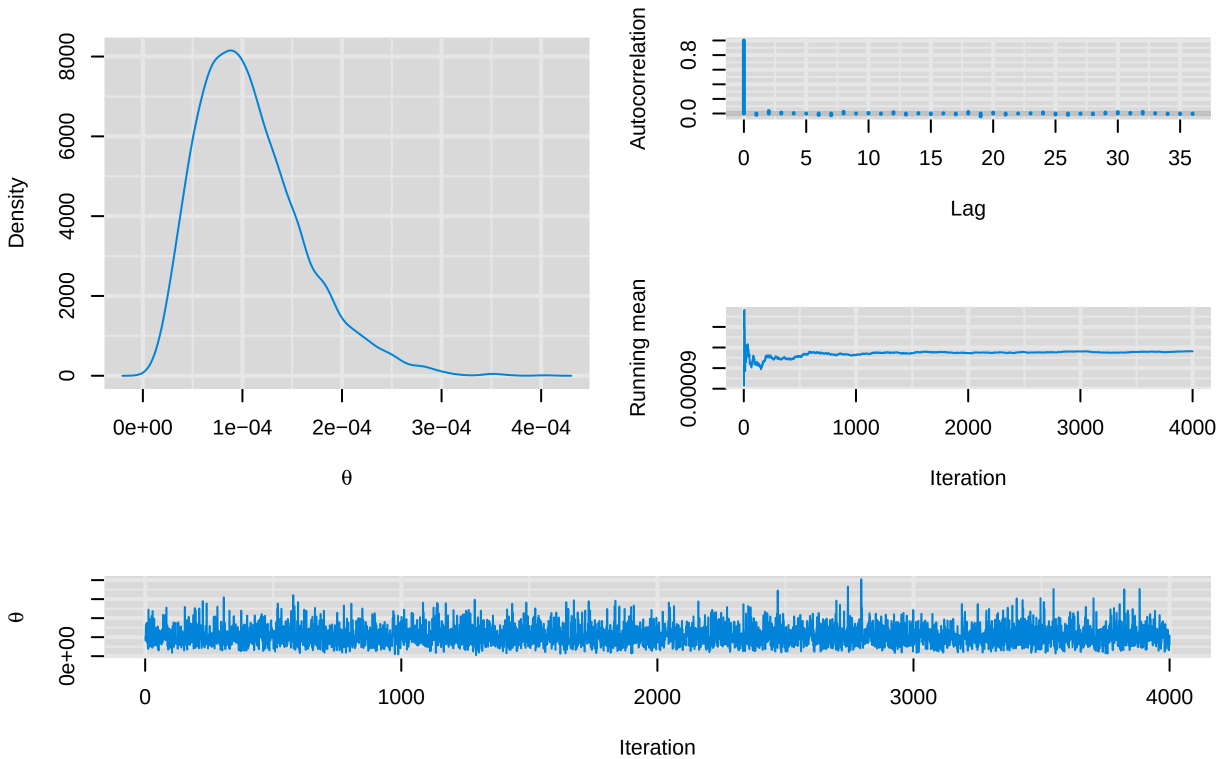
Diagnostics for θ

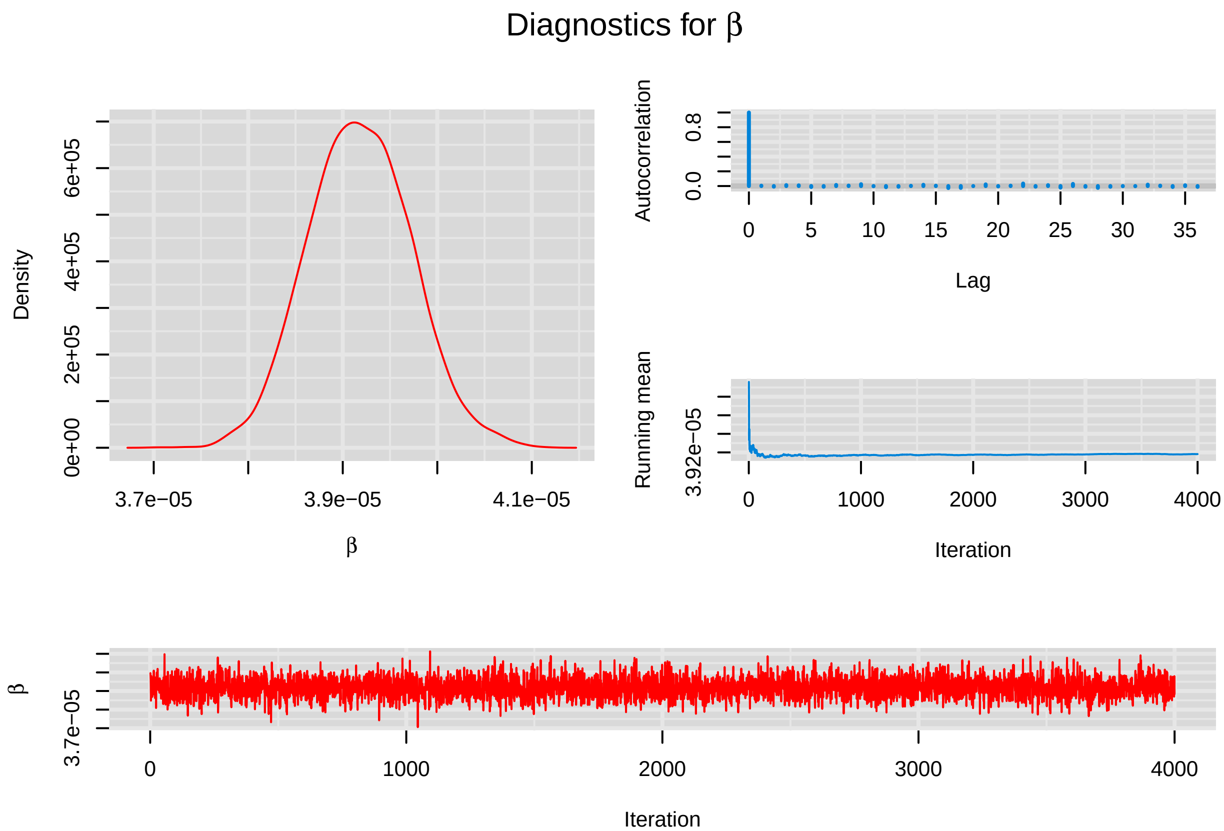




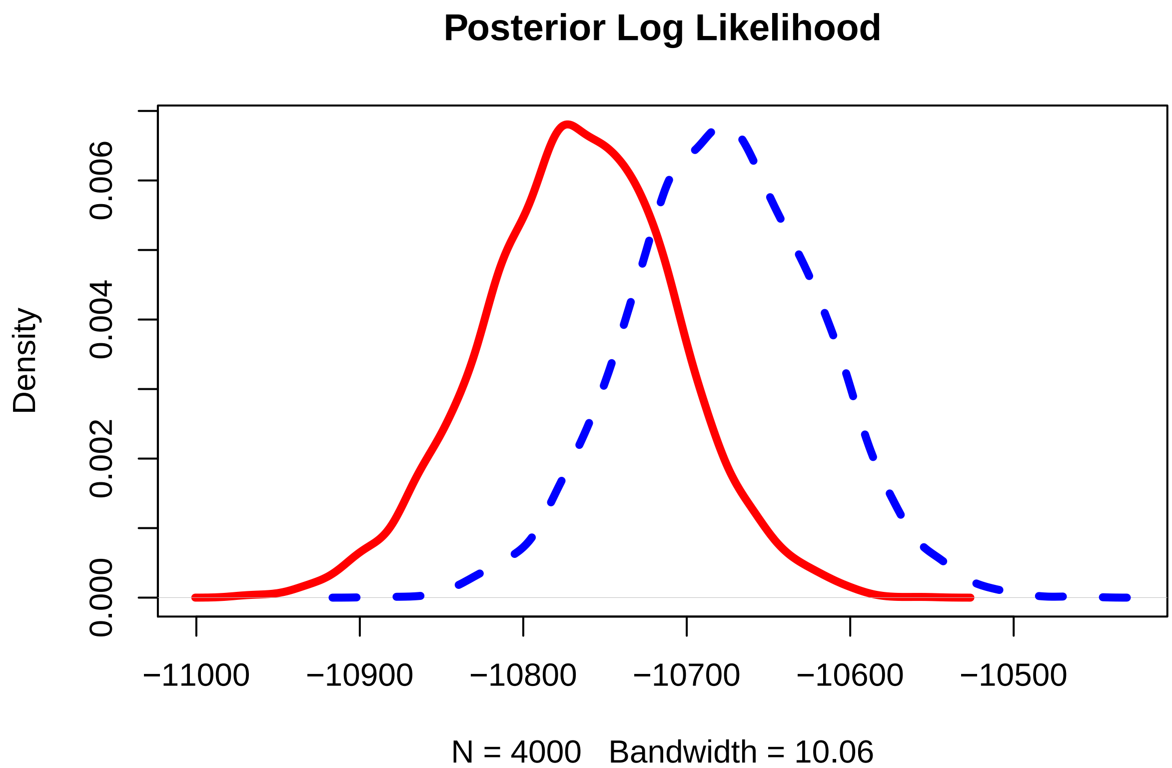
Diagnostics for θ
Density
8000
6000
4000
2000
0
0e+00
1e−04
2e−04
3e−04
4e−04
θ
Autocorrelation
0.8
0.0
0
5
10
15
20
25
30
35
Lag
Running mean
0.00004
0
1000
2000
3000
4000
Iteration
θ
0e+00
0
1000
2000
3000
4000
Iteration

Diagnostics for θ

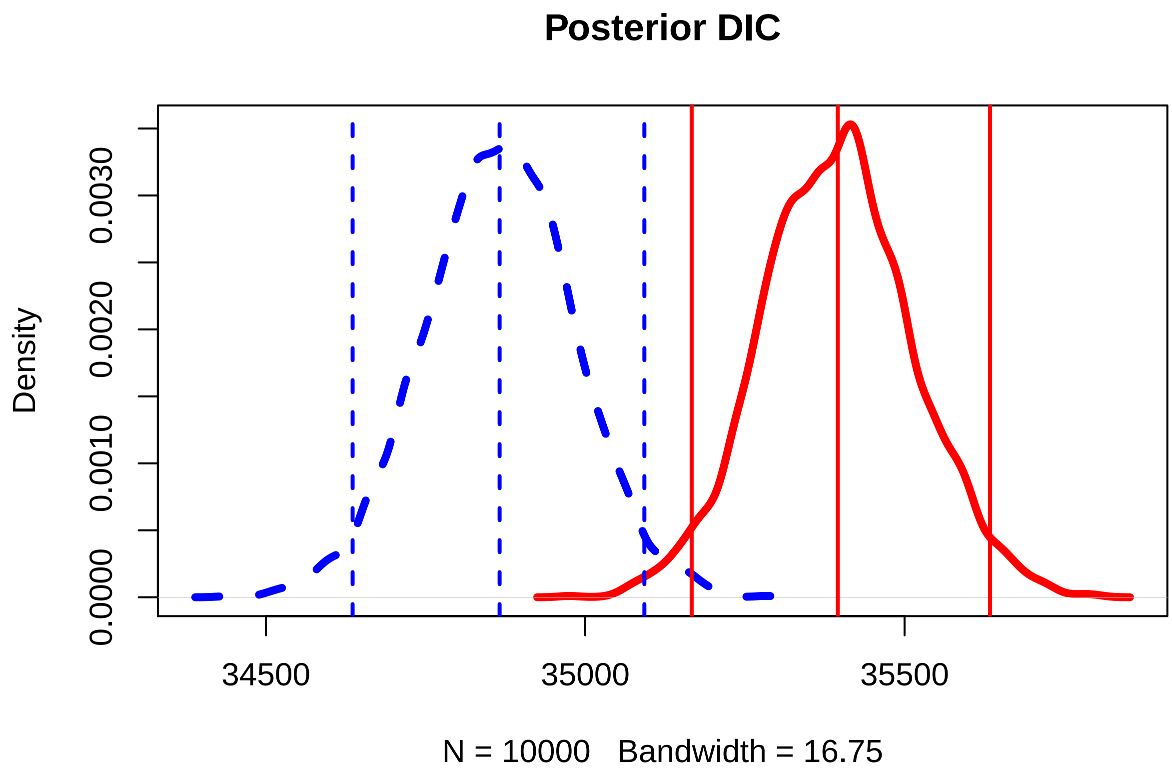




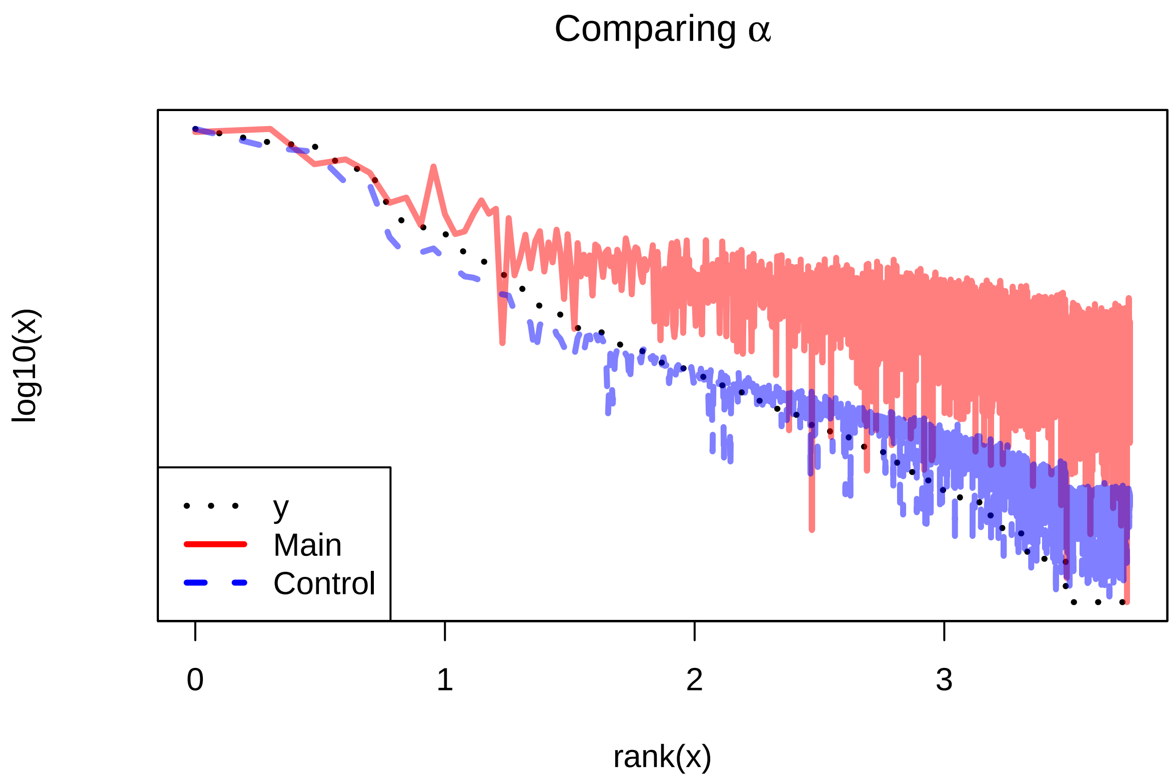
log posterior

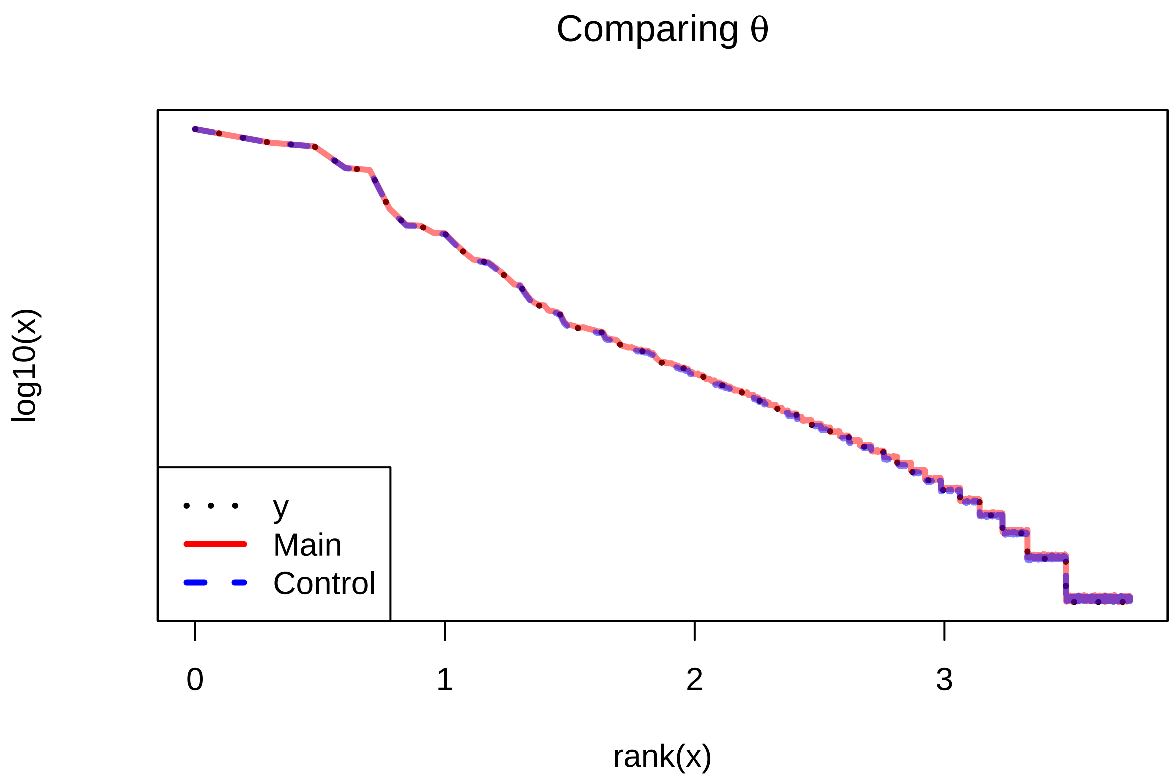


DIC, now in distribution form!



full circle: zipfs law and our models


Comparing α
log10(x)
rank(x)
0
1
2
3
y
Main
Control



3 Results

4 Discussion

5 References

5.1 make sure to cite the data source

5.2 Cite info on Zipf's law

6 Appendix A

```
# 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
```

```r
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")
```

Describe Zipf's law plot here

```r
# 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)")
```

6.0.1 Sampling the control model

```r
# 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 = 0.1) {
    # 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)
    } 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
```

```r
  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))),
```

```r
                              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))
})
```

6.0.2 Sampling the main model

```
# 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
```

```r
  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
```

```r
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')
```

6.1 Comparison

We should put some posterior plots here, maybe

```r
# 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)))
```

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

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

plot(density(rowMeans(control_alpha_acc)), lwd = 4, col = "blue", lty = 2,
     main = expression(paste("Acceptance rates of ", alpha, " parameters in both models")),
     xlab = "Acceptance rate",
     xlim = range(control_alpha_acc, main_alpha_acc))
lines(density(rowMeans(main_alpha_acc)), lwd = 4, lty = 1, col = "red")
legend("topright", legend = c("Main", "Control"), lwd = 4, lty = c(1,3),
       col = c("red", "blue"))

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.")

knitr::kable(alpha_acc_table, digits = 2)
```

Convergence is based on log likelihood of both models

```r
# check convergence

conv <- parallel::mclapply(list(main = main_posterior, control = control_posterior),
                           function(x){
```

```r
                              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))
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))

g <- lapply(conv, geweke.diag)

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

rownames(g) <- ""

knitr::kable(data.frame(g),digits = 2, caption = "Geweke statistic of log likelihoods")
```

MCMC plots of selected words

```r
# 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")
```

log posterior

```
# 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)
```

DIC, now in distribution form!

```
# 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))
```

full circle: zipfs law and our models

```
# 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"))
```

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

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

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

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

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

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

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

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

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

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

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

7.3 Jeffrey's prior on β

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

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

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

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

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

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

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