Stat 651 Project

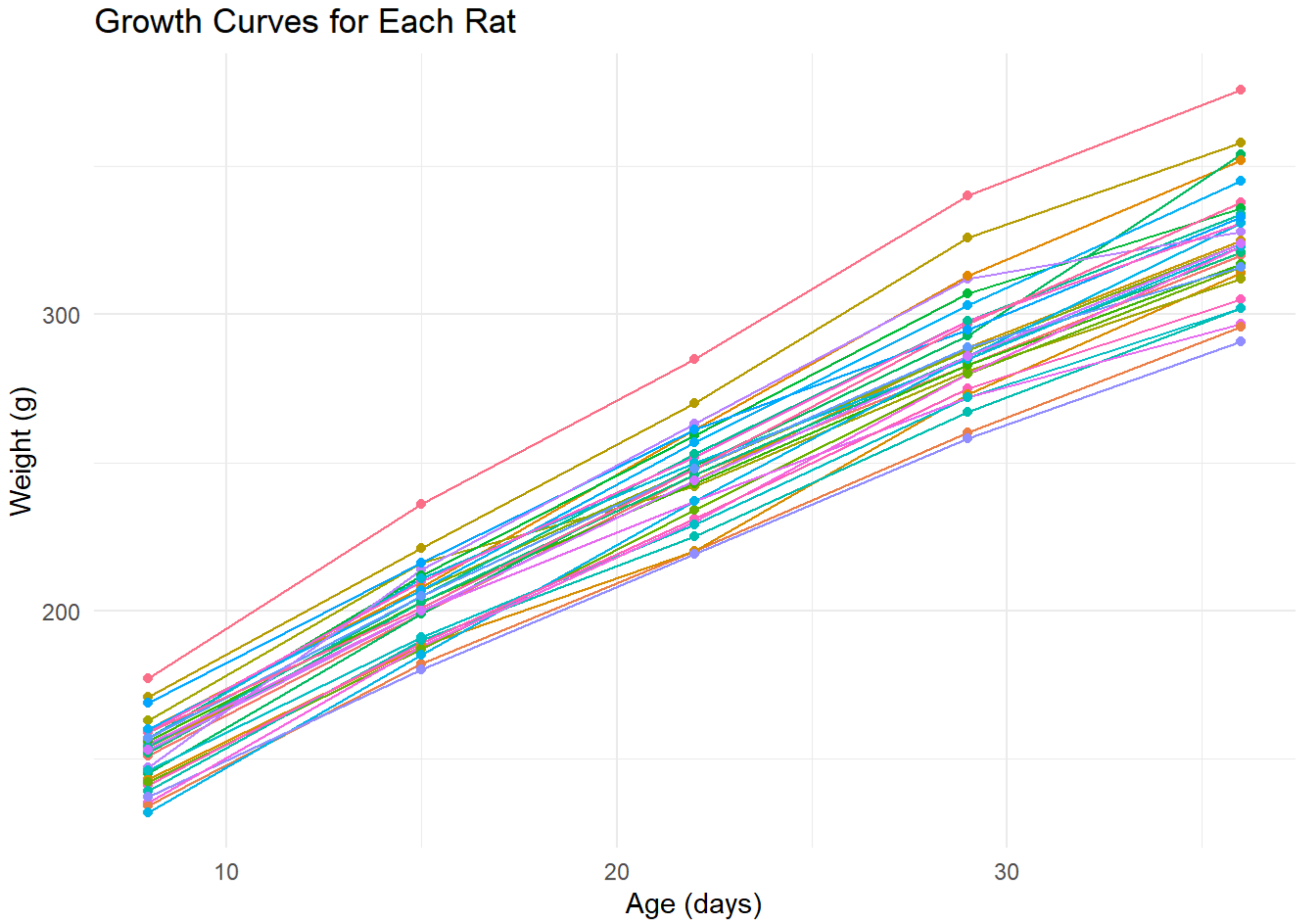
Talmage Hilton

2025-04-08

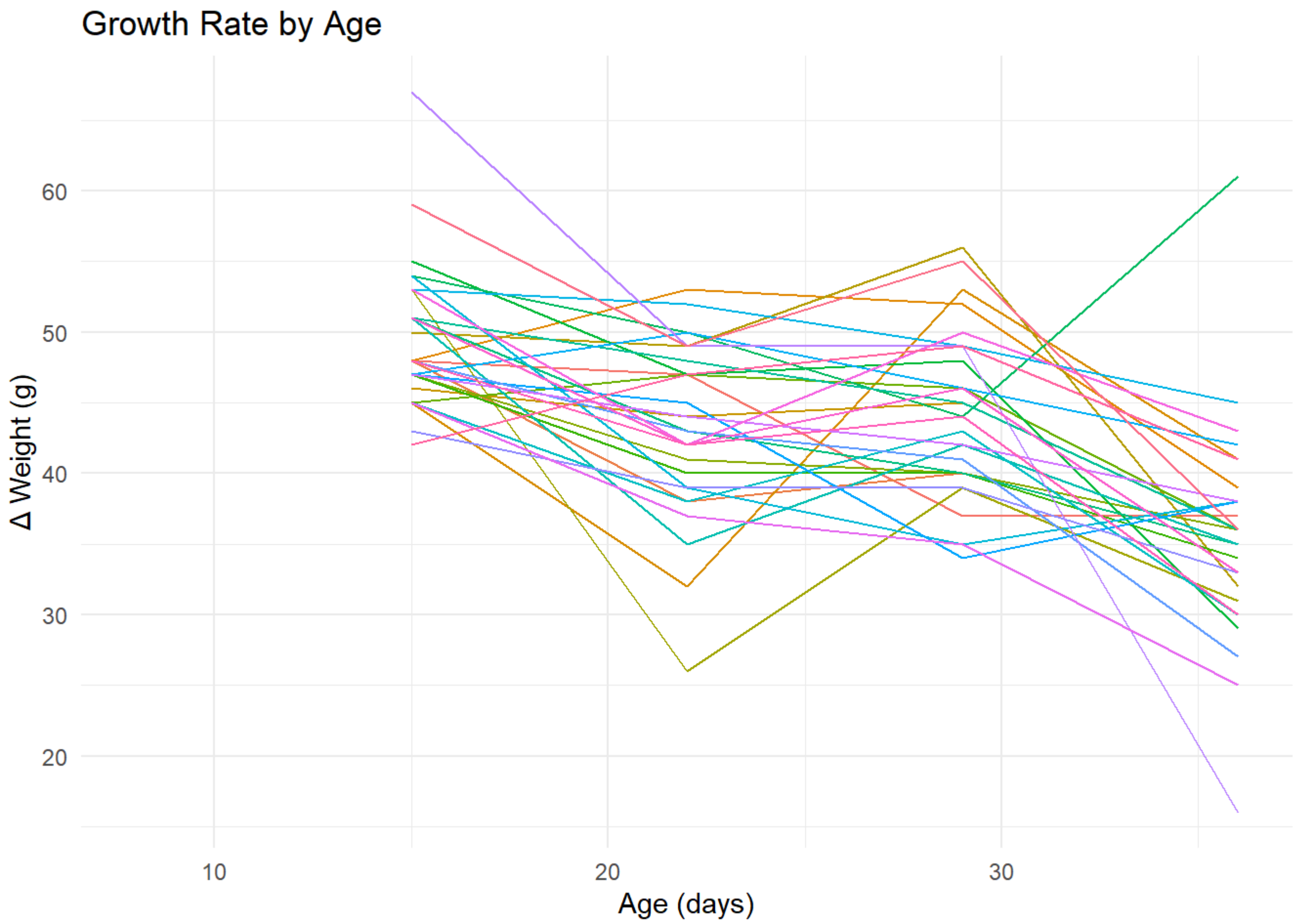
```
# Read in data
ratdata <- read_excel("ratdata.xlsx")

# Pivot longer to get tidy data
rat_long <- ratdata %>%
  pivot_longer(cols = starts_with("rat"), names_to = "rat", values_to = "weight")
```

EDA

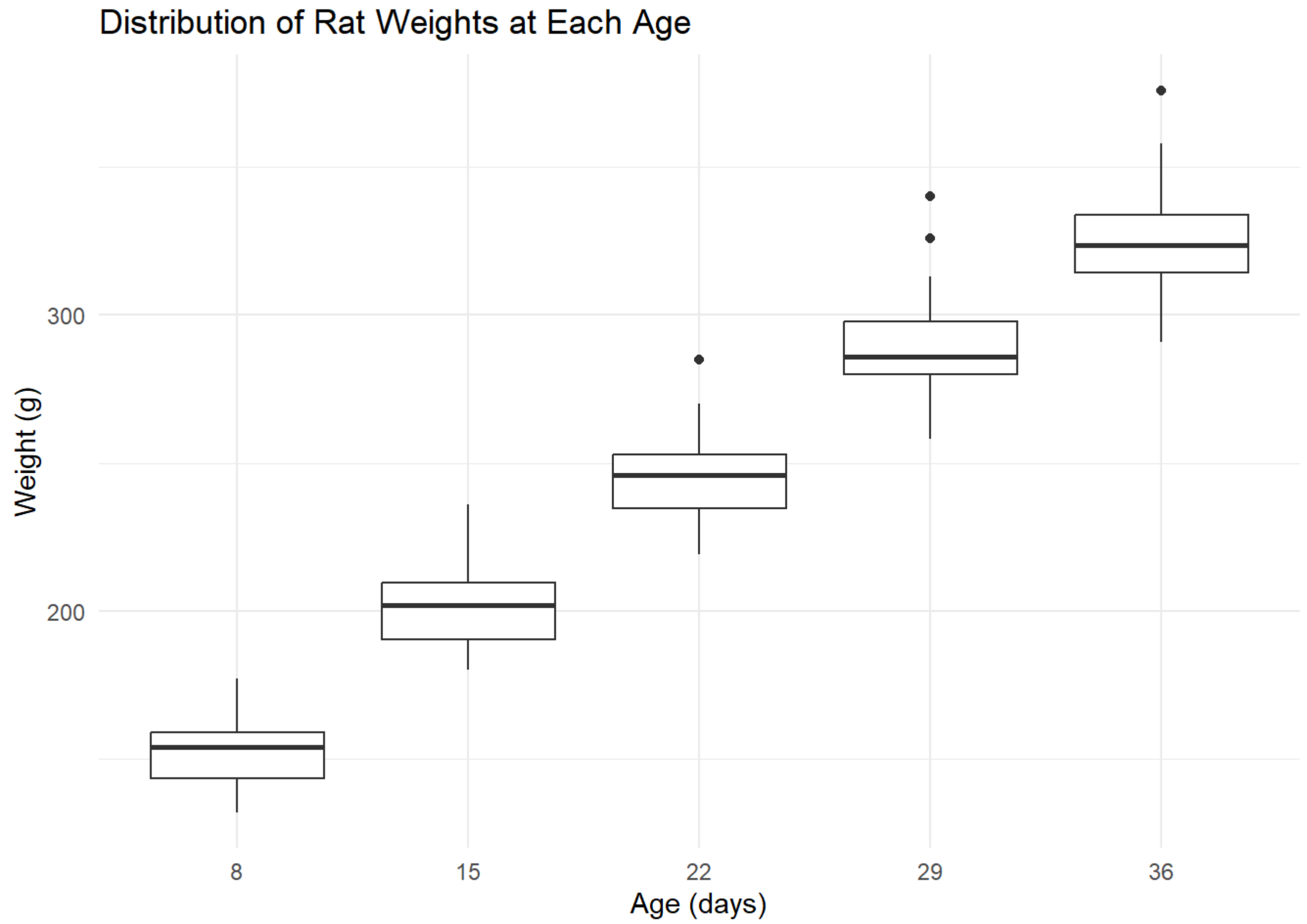
```
# Weights of each rat
ggplot(rat_long, aes(x = age, y = weight, group = rat, color = rat)) +
  geom_line() +
  geom_point() +
  labs(title = "Growth Curves for Each Rat", x = "Age (days)", y = "Weight (g)") +
  theme_minimal() +
  theme(legend.position = "none")
```



```
# Growth rate for each rat
rat_diff <- ratdata %>%
  dplyr::select(-age) %>%
  mutate(across(everything(), ~ c(NA, diff(.)))) %>%
  mutate(age = ratdata$age)

rat_diff_long <- rat_diff %>%
  pivot_longer(cols = -age, names_to = "rat", values_to = "growth_rate")

ggplot(rat_diff_long, aes(x = age, y = growth_rate, group = rat, color = rat)) +
  geom_line() +
  labs(title = "Growth Rate by Age", x = "Age (days)", y = "Δ Weight (g)") +
  theme_minimal() +
  theme(legend.position = "none")
```




```
# Boxplots at each age
ggplot(rat_long, aes(x = factor(age), y = weight)) +
  geom_boxplot() +
  labs(title = "Distribution of Rat Weights at Each Age", x = "Age (days)", y = "Weight (g)")
+
  theme_minimal()
```



The growth curves show that the rats grow in a mostly linear fashion. Obviously there is some variability, but overall I'd say that it's pretty linear. The growth rates are not very consistent (lots of ups and downs), but overall they're all pretty similar. For the most part the growth rate slows down between days 15-22, then speed up between days 22-29, and then decrease again from days 29-36. However, there are a few rate that increase in growth rate during the last time period. There are some exceptions to the rule here, but overall I'd say that the Normal model for Y_{ij} is fairly reasonable.

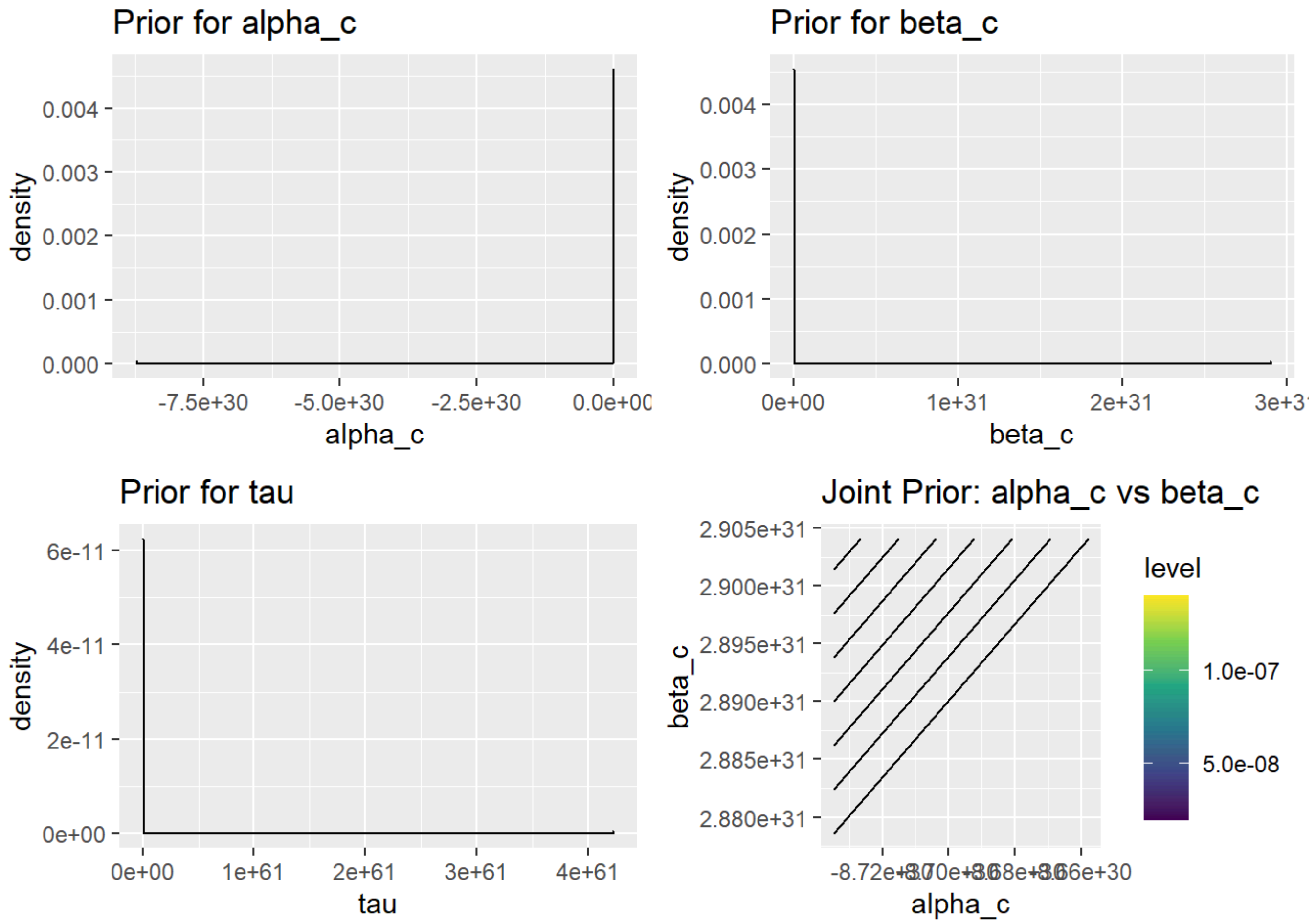
```
# Justify the prior
library(MASS)       # For mvrnorm
library(ggplot2)    # For prettier plots
library(reshape2)   # For melt
library(gridExtra)  # For side-by-side plots
```

```
##
## Attaching package: 'gridExtra'
```

```
## The following object is masked from 'package:dplyr':
##
##     combine
```

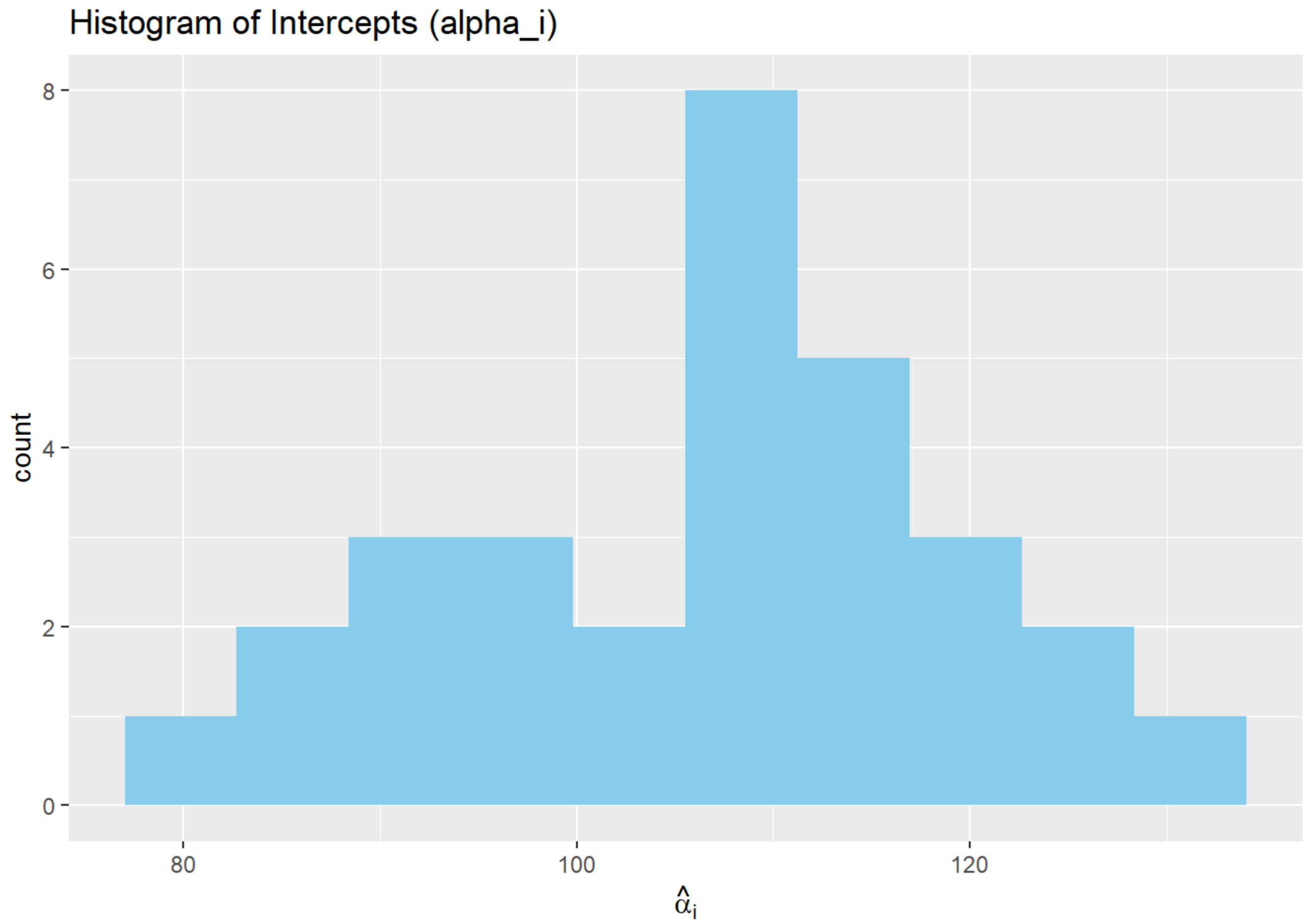
```
# Prior parameters
eta <- c(0, 0)
C <- diag(5, 2)
lambda_0 <- 0.1
nu_0 <- 0.1

# Sample from prior
set.seed(123)
n_samples <- 100

# Sample tau ~ Inv-Gamma
# Inv-Gamma(a, b) → 1 / rgamma(n, shape = a, rate = b)
tau <- 1 / rgamma(n_samples, shape = nu_0 / 2, rate = (nu_0 * lambda_0) / 2)

# For each tau, sample alpha_c and beta_c ~ N2(eta, tau * C)
alphas_betas <- t(sapply(tau, function(t) {
  mvrnorm(1, mu = eta, Sigma = t * C)
}))

# Organize samples
df <- data.frame(alpha_c = alphas_betas[,1],
                 beta_c = alphas_betas[,2],
                 tau = tau)

# Marginal density plots
p1 <- ggplot(df, aes(x = alpha_c)) + geom_density(fill = "skyblue") + ggtitle("Prior for alph
a_c")
p2 <- ggplot(df, aes(x = beta_c)) + geom_density(fill = "orange") + ggtitle("Prior for beta_
c")
p3 <- ggplot(df, aes(x = tau)) + geom_density(fill = "green") + ggtitle("Prior for tau")

# Contour plot of joint (alpha_c, beta_c)
p4 <- ggplot(df, aes(x = alpha_c, y = beta_c)) +
  stat_density_2d(aes(fill = ..level..), geom = "polygon", color = "black") +
  scale_fill_viridis_c() + ggtitle("Joint Prior: alpha_c vs beta_c")

# Plot all together
grid.arrange(p1, p2, p3, p4, ncol = 2)
```

```
## Warning: The dot-dot notation (`..level..`) was deprecated in ggplot2 3.4.0.
## ℹ Please use `after_stat(level)` instead.
## This warning is displayed once every 8 hours.
## Call `lifecycle::last_lifecycle_warnings()` to see where this warning was
## generated.
```

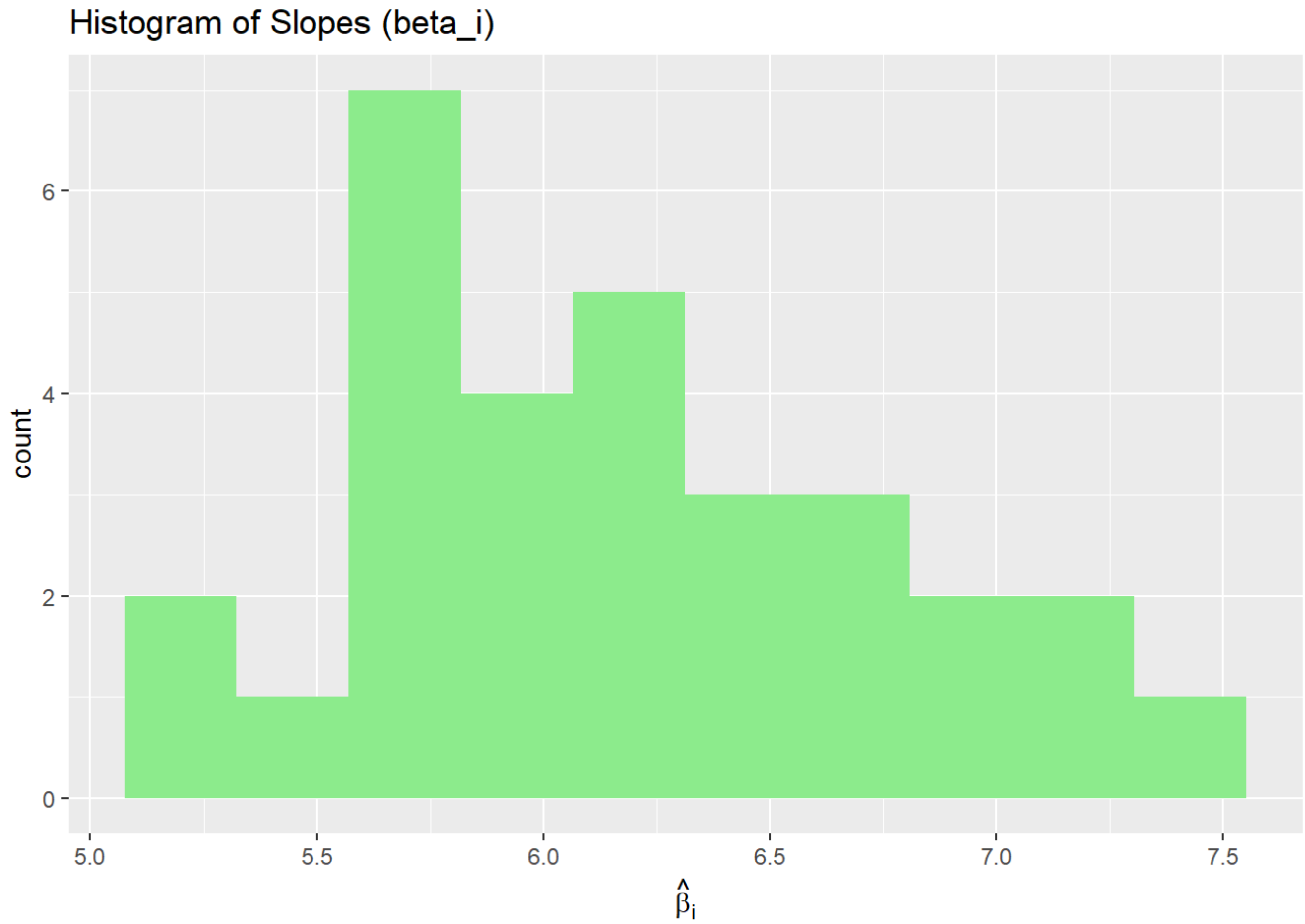


Question 2

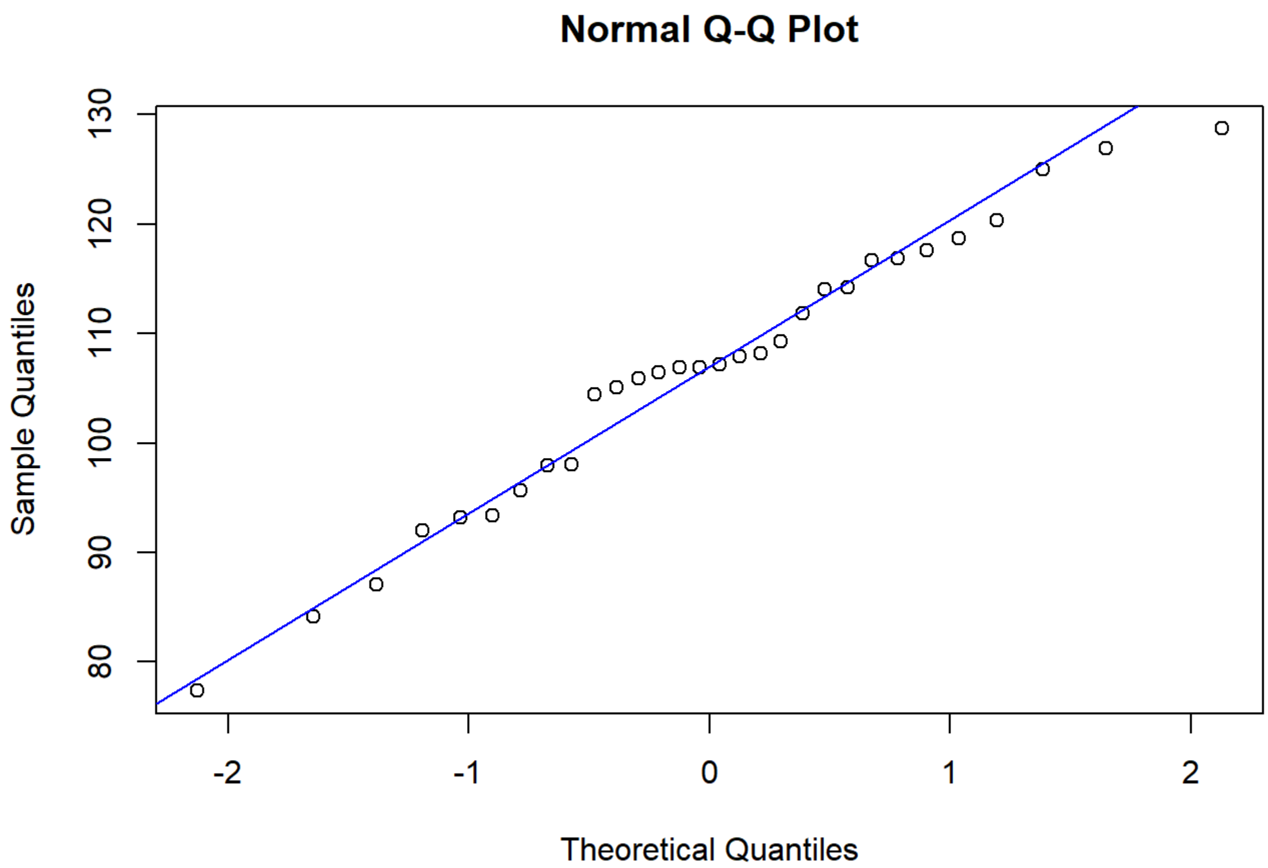
```
# Nest the data by rat
rat_nested <- rat_long %>%
  group_by(rat) %>%
  nest()

# Fit regression to each nested dataset
rat_models <- rat_nested %>%
  mutate(model = map(data, ~lm(weight ~ age, data = .)))

# Extract coefficients from each model
rat_coefs <- rat_models %>%
  mutate(coefs = map(model, broom::tidy)) %>%
  unnest(coefs) %>%
  dplyr::select(rat, term, estimate) %>%
  pivot_wider(names_from = term, values_from = estimate) %>%
  rename(alpha_hat = `(Intercept)`, beta_hat = age)


# Check Normality of thetas

# Histogram of intercepts
ggplot(rat_coefs, aes(x = alpha_hat)) +
  geom_histogram(bins = 10, fill = "skyblue") +
  labs(title = "Histogram of Intercepts (alpha_i)", x = expression(hat(alpha)[i]))
```



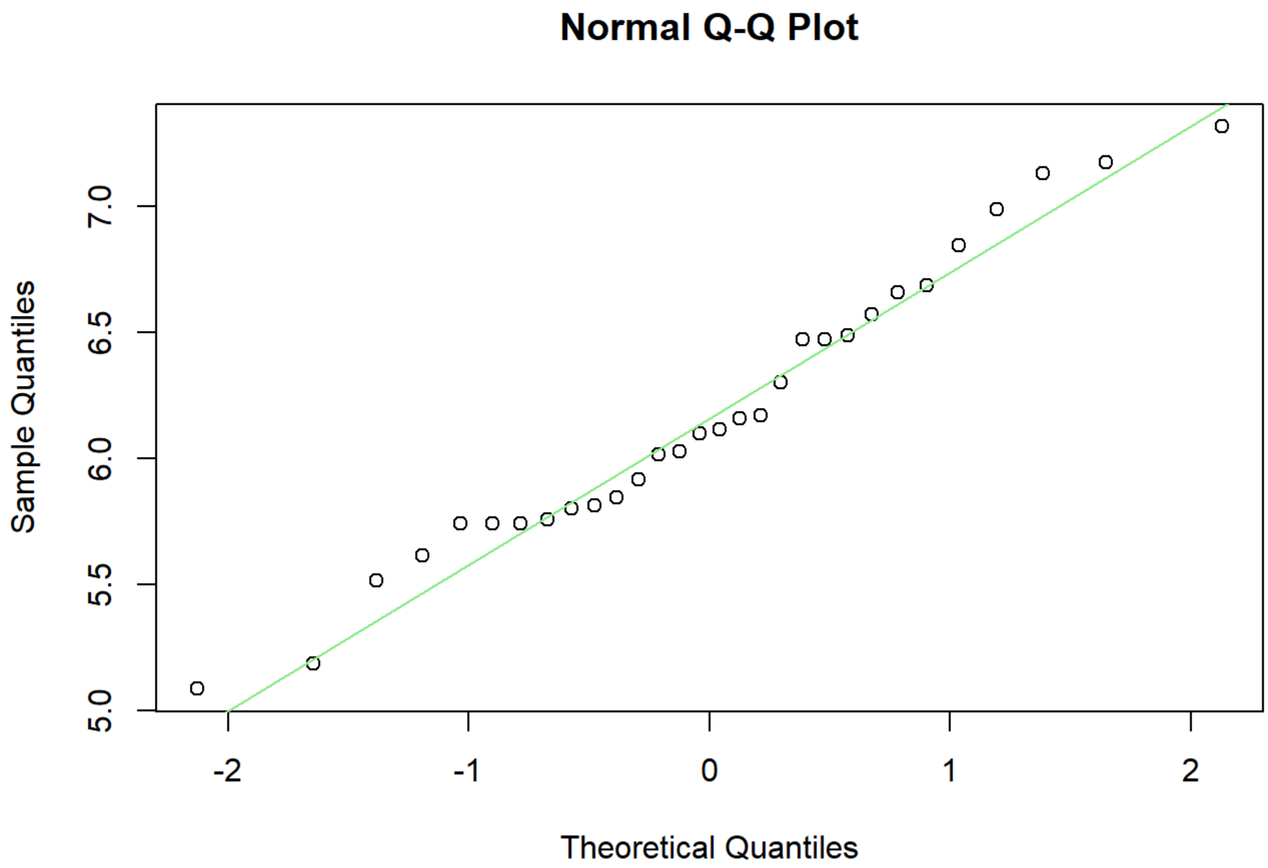

```
# Histogram of slopes
ggplot(rat_coefs, aes(x = beta_hat)) +
  geom_histogram(bins = 10, fill = "lightgreen") +
  labs(title = "Histogram of Slopes (beta_i)", x = expression(hat(beta)[i]))
```




```
# QQ plots
qqnorm(rat_coefs$alpha_hat); qqline(rat_coefs$alpha_hat, col = "blue")
```



```
qqnorm(rat_coefs$beta_hat); qqline(rat_coefs$beta_hat, col = "lightgreen")
```



I would say that the normality assumption is reasonable here. The distribution of intercepts is approximately normal. The distribution of slopes is a little right skewed, but I wouldn’t say it’s too egregious. The Q-Q plots also aren’t too terrible.

Gibbs Sampler

```
# Reshape data
rats <- melt(ratdata, id.vars = c("age"))
colnames(rats) <- c("age",'id','weight')

# Priors
lambda0 <- 0.1
nu0 <- 0.1
eta <- matrix(c(0,0), nrow=2)
Sigma <- matrix(c(10, 0, 0, 10), nrow=2, ncol=2)
Sigma_inv <- solve(Sigma)
C <- matrix(c(5, 0, 0, 5), nrow=2, ncol=2)
C_inv <- solve(C)
V <- solve(30 * Sigma_inv + C_inv)


set.seed(123)

# Prepare data
age <- ratdata$age
n <- 30  # number of rats
t <- 5  # number of time points = 5

# Design matrix
rat_data <- lapply(unique(rats$id), function(j) {
    that_rat <- rats[rats$id == j, ]
    that_rat$weight          # Response variable
})
Xi <- cbind(rep(1, 5), c(8, 15, 22, 29, 36))  # Define X_i once
XtX <- t(Xi) %*% Xi
Y_matrix <- do.call(cbind, rat_data)  # Combine all Y_i into a matrix (5 x 30)

# Hyperparameters
n_iter <- 100000
burn_in <- 10000
thin <- 10

# Storage
alphas = matrix(nrow=n_iter,ncol=30)
betas = matrix(nrow=n_iter,ncol=30)
mu_cs = matrix(nrow=n_iter,ncol=2)
taus = matrix(nrow=n_iter,ncol=1)
sses = rep(0,n_iter)

# Initialize
alpha = rep(0,30)
beta = rep(0,30)
mu_c = rep(0,2)
tau = 1

# Gibbs sampler
for (i in 1:n_iter) {

  # --- 1. Sample theta_i | Y_i, mu, tau ---
  # Compute D_i_inv for all groups (same for all 30 rats)
  Di_inv <- (1/tau) * XtX + Sigma_inv
```

```
  Di <- solve(Di_inv)

  # Compute means (2 × 30 matrix)
  means <- Di %*% ((1/tau) * (t(Xi) %*% Y_matrix) + Sigma_inv %*% matrix(mu_c, ncol=30, nrow=
2, byrow=FALSE))

  # Draw samples for all 30 groups at once (each row is a rat)
  new_params <- t(rmvnorm(n = 30, mean = rep(0, 2), sigma = Di)) + means  # Ensuring correct
shape

  # Extract alpha and beta
  alphas[i, ] <- alpha <- new_params[1, ]  # First row is alpha
  betas[i, ] <- beta <- new_params[2, ]   # Second row is beta


  # --- 2. Sample mu | theta, tau ---
  ## Update the group parameters
  theta_bar = matrix(c(mean(alpha), mean(beta)), nrow=2)
  mu_c = rmvnorm(n = 1,
                mean = V %*% (30 * Sigma_inv %*% theta_bar + C_inv %*% eta),
                sigma = V)
  mu_c = as.vector(mu_c)
  mu_cs[i,] = mu_c


  # --- 3. Sample tau | theta ---
  # Compute SSE efficiently
  residuals <- Y_matrix - Xi %*% new_params
  sse <- sum(residuals^2)
  sses[i] <- sse  # Store SSE

  # Update tau
  tau <- rinvgamma(n=1,
                  shape = (nu0 + 150)/ 2,
                  scale = (1/2) * (nu0 * lambda0 + sse))
  taus[i] <- tau
}



# POSTERIOR INFERENCE

# Remove burn-in
alpha_post <- alphas[(burn_in+1):n_iter, ]
beta_post <- betas[(burn_in+1):n_iter, ]
alpha_c_post <- mu_cs[(burn_in+1):n_iter, 1]
beta_c_post <- mu_cs[(burn_in+1):n_iter, 2]
tau_post <- taus[(burn_in+1):n_iter]

# Thin
alpha_post <- alpha_post[seq(1, nrow(alpha_post), by=thin), ]
beta_post <- beta_post[seq(1, nrow(beta_post), by=thin), ]
alpha_c_post <- alpha_c_post[seq(1, length(alpha_c_post), by=thin)]
beta_c_post <- beta_c_post[seq(1, length(beta_c_post), by=thin)]
tau_post <- tau_post[seq(1, length(tau_post), by=thin)]
```

```
# Compute posterior means, variances, and credible intervals
alpha_est <- apply(alpha_post, 2, mean)
alpha_var <- apply(alpha_post, 2, var)
alpha_ci <- apply(alpha_post, 2, function(x) quantile(x, probs = c(0.025, 0.975)))

beta_est <- apply(beta_post, 2, mean)
beta_var <- apply(beta_post, 2, mean)
beta_ci <- apply(beta_post, 2, function(x) quantile(x, probs = c(0.025, 0.975)))

alpha_c_est <- mean(alpha_c_post)
alpha_c_var <- var(alpha_c_post)
alpha_c_ci <- quantile(alpha_c_post, probs = c(0.025, 0.975))
beta_c_est <- mean(beta_c_post)
beta_c_var <- var(beta_c_post)
beta_c_ci <- quantile(beta_c_post, probs = c(0.025, 0.975))

tau_est <- mean(tau_post)
tau_var <- var(tau_post)
tau_ci <- quantile(tau_post, probs = c(0.025, 0.975))

# Print posterior estimates
cat("Posterior Mean of alpha:\n", alpha_est, "\n")
```

```
## Posterior Mean of alpha:
##  6.667316 6.630745 6.778173 6.779697 6.533948 6.725573 6.655685 6.732996 6.788809 6.633324
6.703732 6.607456 6.652874 6.718473 6.770628 6.750904 6.674162 6.76905 6.710859 6.70431 6.734
694 6.582309 6.697096 6.768374 6.600971 6.776939 6.737182 6.754197 6.611198 6.702432
```

```
cat("Posterior Variance of alpha:\n", alpha_var, "\n")
```

```
## Posterior Variance of alpha:
##  15.63594 15.45511 15.43135 14.88998 15.12935 15.19445 15.33134 15.48413 15.62299 15.3057
15.10144 15.38704 15.43006 15.0637 15.01497 15.33964 15.12333 15.09833 15.16042 15.39388 15.8
0234 15.50716 15.28125 15.16023 15.3075 15.21455 15.17751 15.00546 15.04692 15.20299
```

```
cat("95% CI for alpha:\n", alpha_ci, "\n\n")
```

```
## 95% CI for alpha:
##  -1.048659 14.42171 -1.104099 14.25037 -0.971798 14.52742 -0.8354788 14.32131 -1.144158 1
4.12933 -0.818209 14.40971 -1.197021 14.22993 -0.9412614 14.49281 -0.8589824 14.64125 -0.8530
033 14.47356 -0.8307167 14.34078 -1.026804 14.38797 -0.9369168 14.54719 -0.8896057 14.1826 -
0.8173413 14.39218 -1.002791 14.50161 -0.8390374 14.25726 -0.8782033 14.40379 -0.9388665 14.3
6312 -0.8931014 14.45864 -1.162896 14.45107 -1.159827 14.34182 -1.032787 14.34 -0.8284084 14.
4285 -1.032591 14.1381 -0.8332164 14.57909 -0.9704972 14.30392 -0.5907241 14.33717 -0.9984552
14.2894 -1.067173 14.26645
```

```
cat("Posterior Mean of beta:\n", beta_est, "\n")
```

```
## Posterior Mean of beta:
##  9.783375 10.27388 10.32268 9.367298 9.601935 10.16816 9.363423 10.18162 11.48696 8.984793 10.60972 9.372273 9.908306 10.93497 9.763672 9.935697 9.550815 9.751102 10.37209 9.843294 10.18381 9.217273 9.303531 9.924102 9.751282 10.39854 10.28911 9.868462 8.910079 9.849183
```

```
cat("Posterior Variance of beta:\n", beta_var, "\n")
```

```
## Posterior Variance of beta:
##  9.783375 10.27388 10.32268 9.367298 9.601935 10.16816 9.363423 10.18162 11.48696 8.984793 10.60972 9.372273 9.908306 10.93497 9.763672 9.935697 9.550815 9.751102 10.37209 9.843294 10.18381 9.217273 9.303531 9.924102 9.751282 10.39854 10.28911 9.868462 8.910079 9.849183
```

```
cat("95% CI for beta:\n", beta_ci, "\n\n")
```

```
## 95% CI for beta:
##  8.139502 11.40018 8.59136 11.95068 8.657094 11.96094 7.679782 10.97746 7.952186 11.26588 8.528952 11.83996 7.713505 11.05325 8.552631 11.84143 9.815865 13.15352 7.352503 10.64765 8.914892 12.28516 7.739273 10.98631 8.268249 11.56366 9.261959 12.58416 8.108349 11.39903 8.310479 11.64418 7.925098 11.21899 8.092801 11.37382 8.701472 12.06636 8.173262 11.4954 8.511383 11.84849 7.589857 10.89919 7.678441 10.92656 8.290863 11.59992 8.091827 11.40905 8.726808 12.04698 8.641312 11.94331 8.239374 11.51634 7.271488 10.5886 8.228946 11.50582
```

```
cat("Posterior Mean of alpha_c:\n", alpha_c_est, "\n")
```

```
## Posterior Mean of alpha_c:
##  6.281169
```

```
cat("Posterior Variance of alpha_c:\n", alpha_c_var, "\n")
```

```
## Posterior Variance of alpha_c:
##  5.262789
```

```
cat("95% CI for alpha_c:\n", alpha_c_ci, "\n\n")
```

```
## 95% CI for alpha_c:
##  1.859327 10.82475
```

```
cat("Posterior Mean of beta_c:\n", beta_c_est, "\n")
```

```
## Posterior Mean of beta_c:
##  9.285139
```

```
cat("Posterior Variance of beta_c:\n", beta_c_var, "\n")
```

```
## Posterior Variance of beta_c:
##  0.3478579
```

```
cat("95% CI for beta_c:\n", beta_c_ci, "\n\n")
```

```
## 95% CI for beta_c:
##  8.112458 10.42863
```

```
cat("Posterior Mean of tau:\n", tau_est, "\n")
```

```
## Posterior Mean of tau:
##  2162.16
```

```
cat("Posterior Variance of tau:\n", tau_var, "\n")
```

```
## Posterior Variance of tau:
##  89883.6
```

```
cat("95% CI for tau:\n", tau_ci, "\n")
```

```
## 95% CI for tau:
##  1647.776 2826.678
```

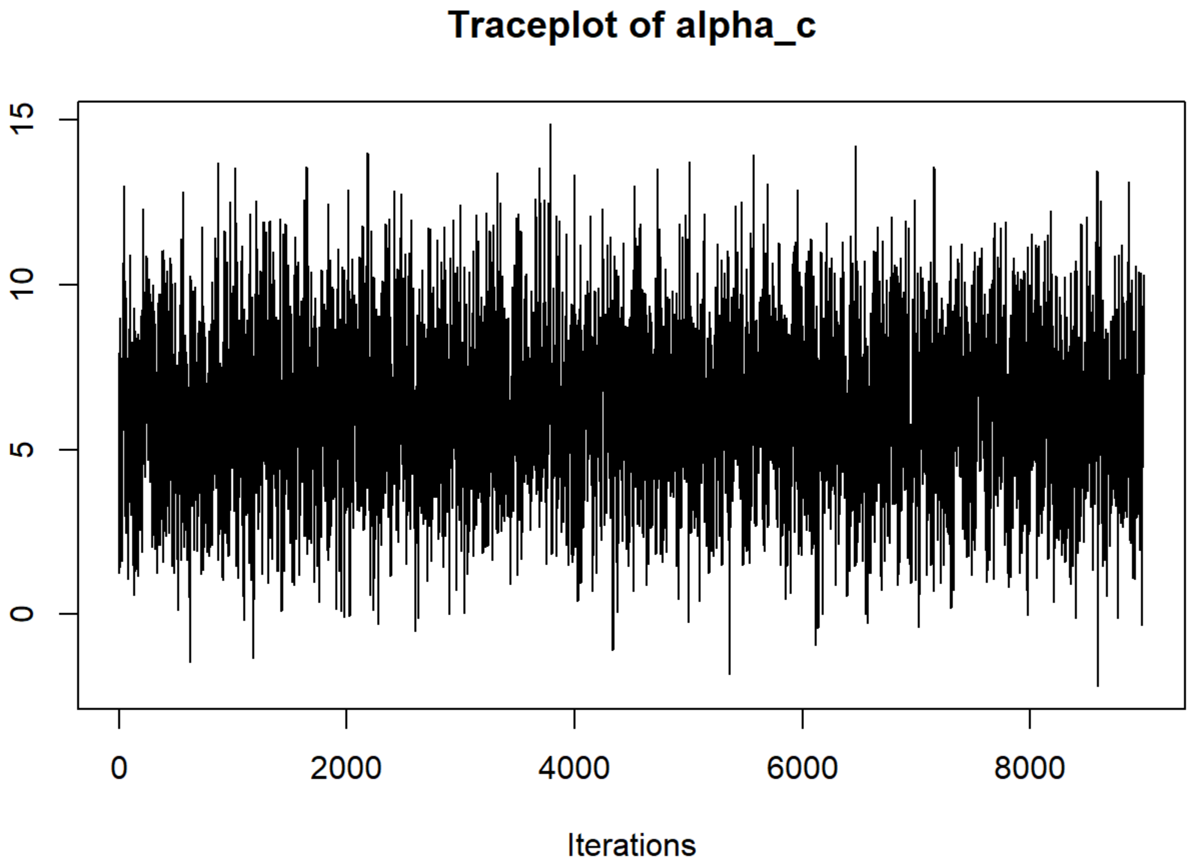
```
# Create a data frame to store the results
ci_table <- data.frame(
  Alpha_Mean = alpha_est,           # Posterior mean of alpha
  Alpha_Lower = alpha_ci[1, ],    # Lower bound of alpha CI
  Alpha_Upper = alpha_ci[2, ],    # Upper bound of alpha CI
  Beta_Mean = beta_est,             # Posterior mean of beta
  Beta_Lower = beta_ci[1, ],      # Lower bound of beta CI
  Beta_Upper = beta_ci[2, ]       # Upper bound of beta CI
)

# View the table
print(ci_table)
```

```
##    Alpha_Mean Alpha_Lower Alpha_Upper Beta_Mean Beta_Lower Beta_Upper
## 1    6.667316  -1.0486585    14.42171  9.783375   8.139502   11.40018
## 2    6.630745  -1.1040991    14.25037 10.273876   8.591360   11.95068
## 3    6.778173  -0.9717980    14.52742 10.322685   8.657094   11.96094
## 4    6.779697  -0.8354788    14.32131  9.367298   7.679782   10.97746
## 5    6.533948  -1.1441581    14.12933  9.601935   7.952186   11.26588
## 6    6.725573  -0.8182090    14.40971 10.168160   8.528952   11.83996
## 7    6.655685  -1.1970214    14.22993  9.363423   7.713505   11.05325
## 8    6.732996  -0.9412614    14.49281 10.181621   8.552631   11.84143
## 9    6.788809  -0.8589824    14.64125 11.486964   9.815865   13.15352
## 10   6.633324  -0.8530033    14.47356  8.984793   7.352503   10.64765
## 11   6.703732  -0.8307167    14.34078 10.609716   8.914892   12.28516
## 12   6.607456  -1.0268038    14.38797  9.372273   7.739273   10.98631
## 13   6.652874  -0.9369168    14.54719  9.908306   8.268249   11.56366
## 14   6.718473  -0.8896057    14.18260 10.934973   9.261959   12.58416
## 15   6.770628  -0.8173413    14.39218  9.763672   8.108349   11.39903
## 16   6.750904  -1.0027908    14.50161  9.935697   8.310479   11.64418
## 17   6.674162  -0.8390374    14.25726  9.550815   7.925098   11.21899
## 18   6.769050  -0.8782033    14.40379  9.751102   8.092801   11.37382
## 19   6.710859  -0.9388665    14.36312 10.372089   8.701472   12.06636
## 20   6.704310  -0.8931014    14.45864  9.843294   8.173262   11.49540
## 21   6.734694  -1.1628959    14.45107 10.183812   8.511383   11.84849
## 22   6.582309  -1.1598271    14.34182  9.217273   7.589857   10.89919
## 23   6.697096  -1.0327869    14.34000  9.303531   7.678441   10.92656
## 24   6.768374  -0.8284084    14.42850  9.924102   8.290863   11.59992
## 25   6.600971  -1.0325908    14.13810  9.751282   8.091827   11.40905
## 26   6.776939  -0.8332164    14.57909 10.398543   8.726808   12.04698
## 27   6.737182  -0.9704972    14.30392 10.289108   8.641312   11.94331
## 28   6.754197  -0.5907241    14.33717  9.868462   8.239374   11.51634
## 29   6.611198  -0.9984552    14.28940  8.910079   7.271488   10.58860
## 30   6.702432  -1.0671729    14.26645  9.849183   8.228946   11.50582
```

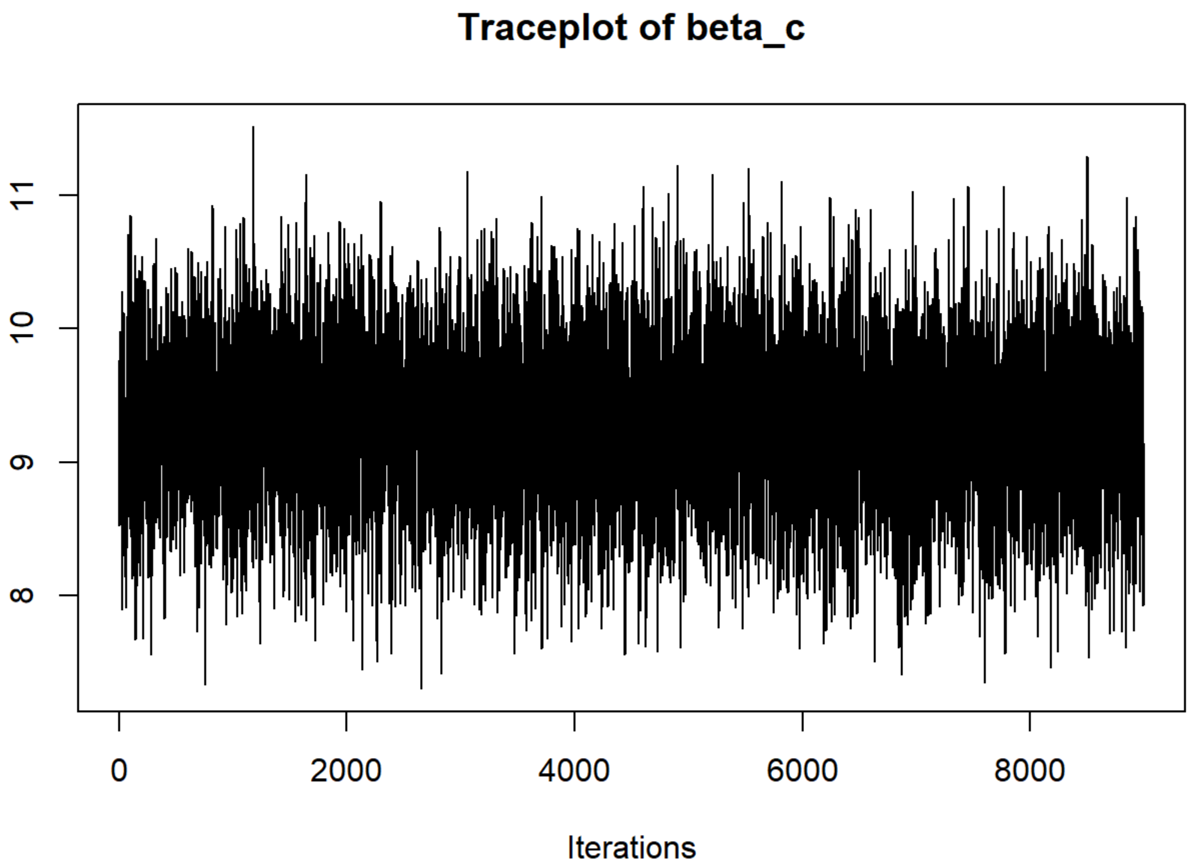
```
# Covariance matrix of alpha_c, beta_c, tau
joint <- cbind(alpha_c_post, beta_c_post, tau_post)
cov(joint)
```

```
##              alpha_c_post beta_c_post     tau_post
## alpha_c_post    5.2627888  -0.1978537  -218.555290
## beta_c_post    -0.1978537   0.3478579     7.908453
## tau_post     -218.5552896   7.9084527 89883.603609
```

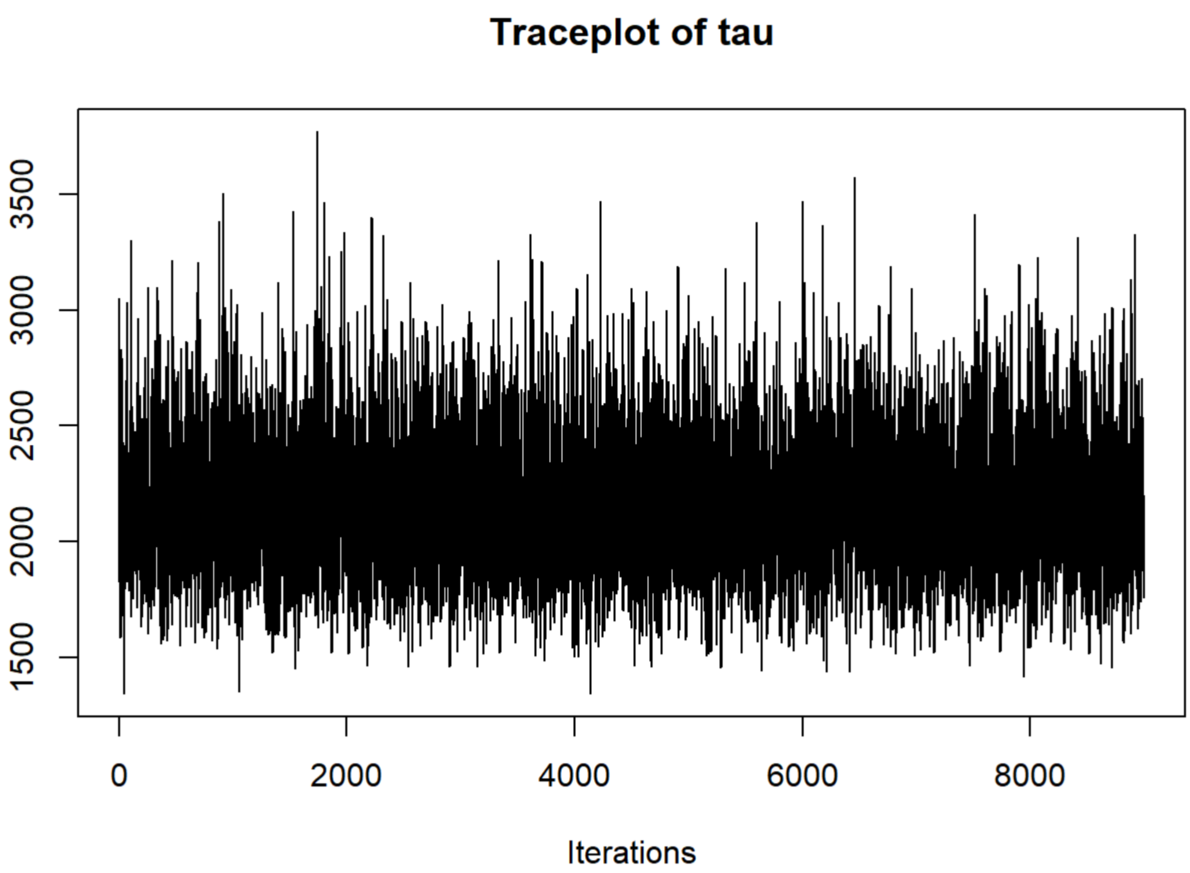
```
# CONVERGENCE GRAPHICAL DIAGNOSTICS

# Convert to mcmc objects
mcmc_alpha <- mcmc(alpha_post)
mcmc_beta <- mcmc(beta_post)
mcmc_alpha_c <- mcmc(alpha_c_post)
mcmc_beta_c <- mcmc(beta_c_post)
mcmc_tau <- mcmc(tau_post)

# Trace plots
par(mfrow = c(1,1))
traceplot(mcmc_alpha_c, main = "Traceplot of alpha_c")
```




```
traceplot(mcmc_beta_c, main = "Traceplot of beta_c")
```



```
traceplot(mcmc_tau, main = "Traceplot of tau")
```



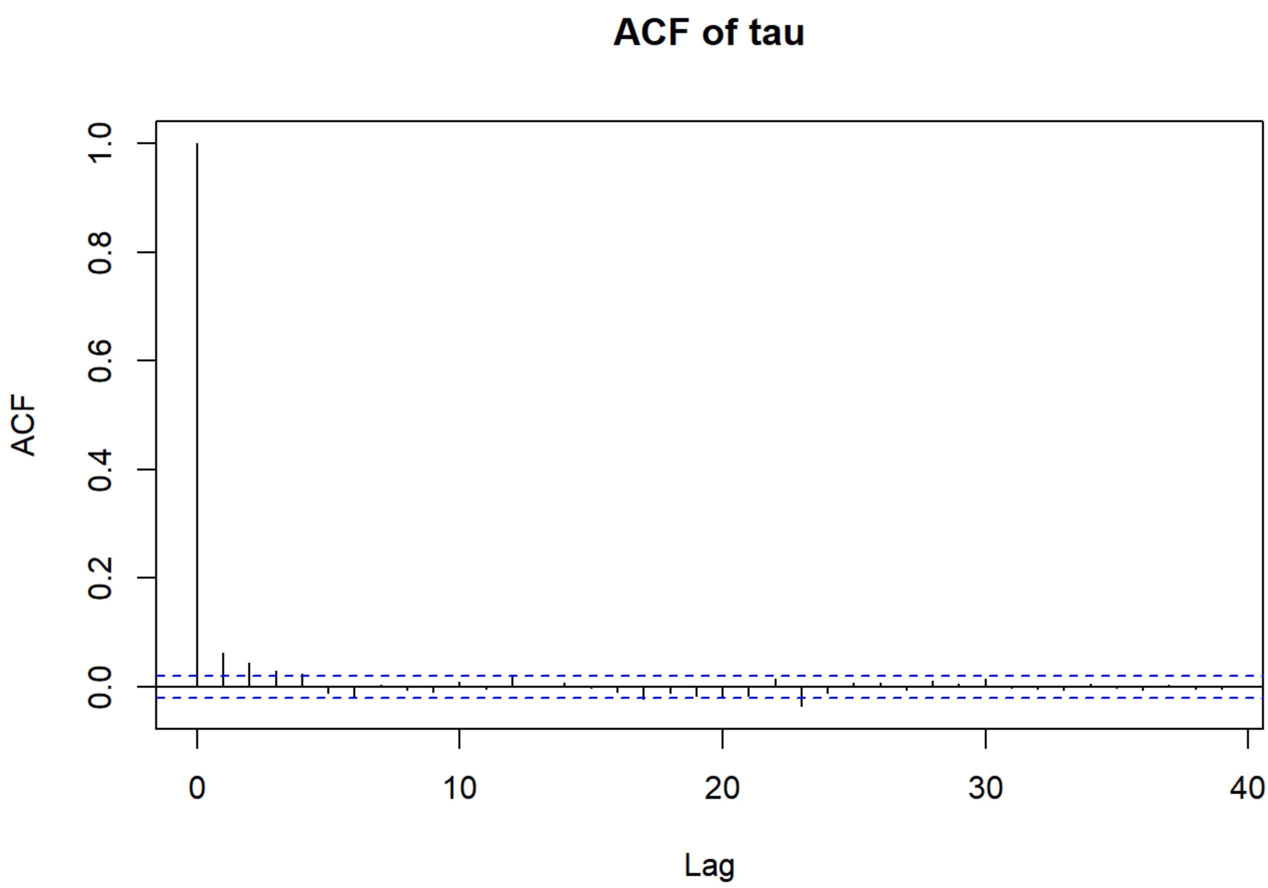

```
# Trace plots for alpha
png("alpha_trace_plots.png", width=1500, height=1000)
par(mfrow=c(6,5))
for (i in 1:30) {
  # Plot trace for alpha (i-th observation)
  plot(1:length(alpha_post[,1]), alpha_post[, i], type = "l",
       main = paste("Trace plot for Alpha", i),
       xlab = "Iteration", ylab = "Alpha Value", col = "black")
}
dev.off()
```

```
## png
##   2
```

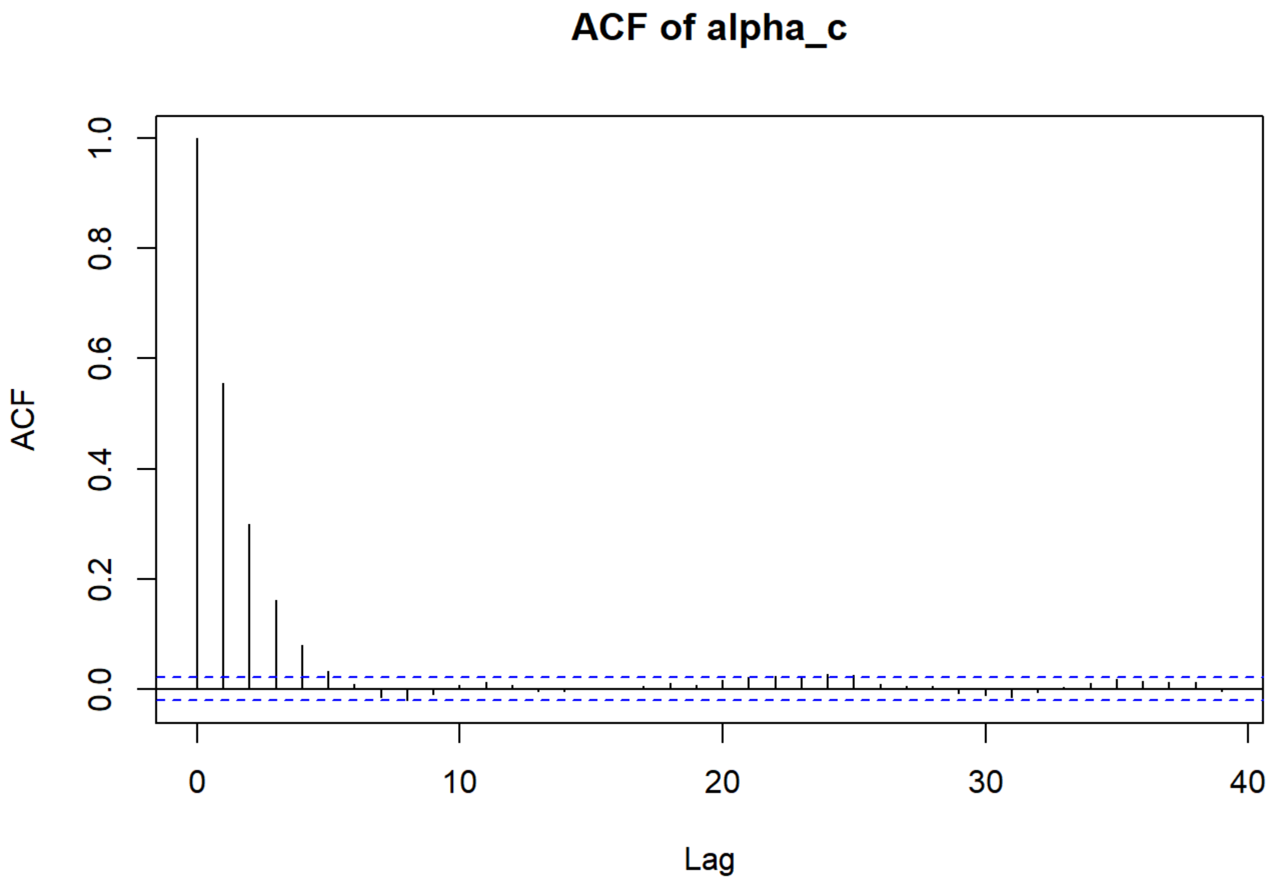
```
# Trace plots for beta
png("beta_trace_plots.png", width=1500, height=1000)
par(mfrow=c(6,5))
for (i in 1:30) {
  # Plot trace for beta (i-th observation)
  plot(1:length(beta_post[,1]), beta_post[, i], type = "l",
       main = paste("Trace plot for Beta", i),
       xlab = "Iteration", ylab = "Beta Value", col = "black")
}

dev.off()
```

```
## png
##   2
```

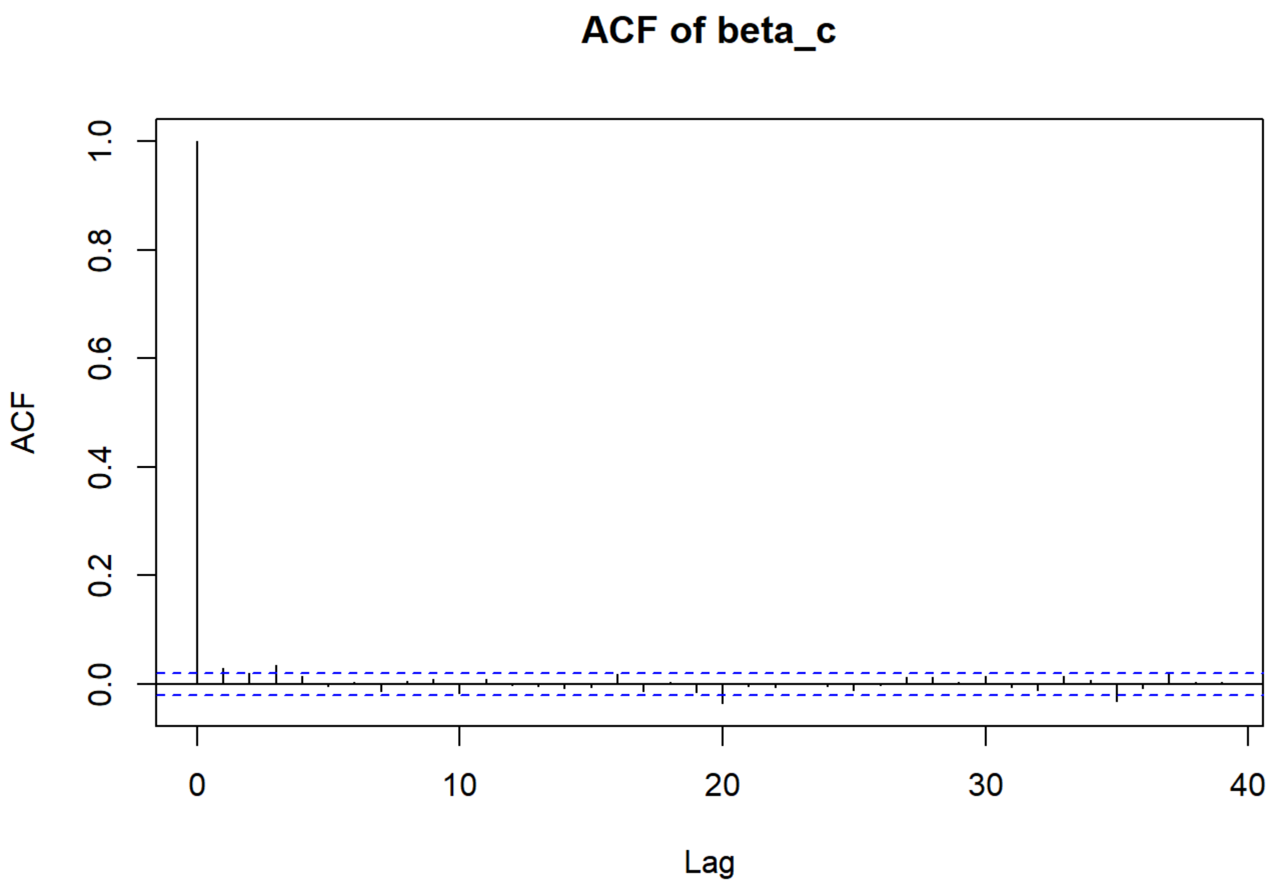
```
# Autocorrelation
par(mfrow=c(1,1))
acf(mcmc_tau, main = "ACF of tau")
```




```
acf(mcmc_alpha_c, main = "ACF of alpha_c")
```




```
acf(mcmc_beta_c, main = "ACF of beta_c")
```



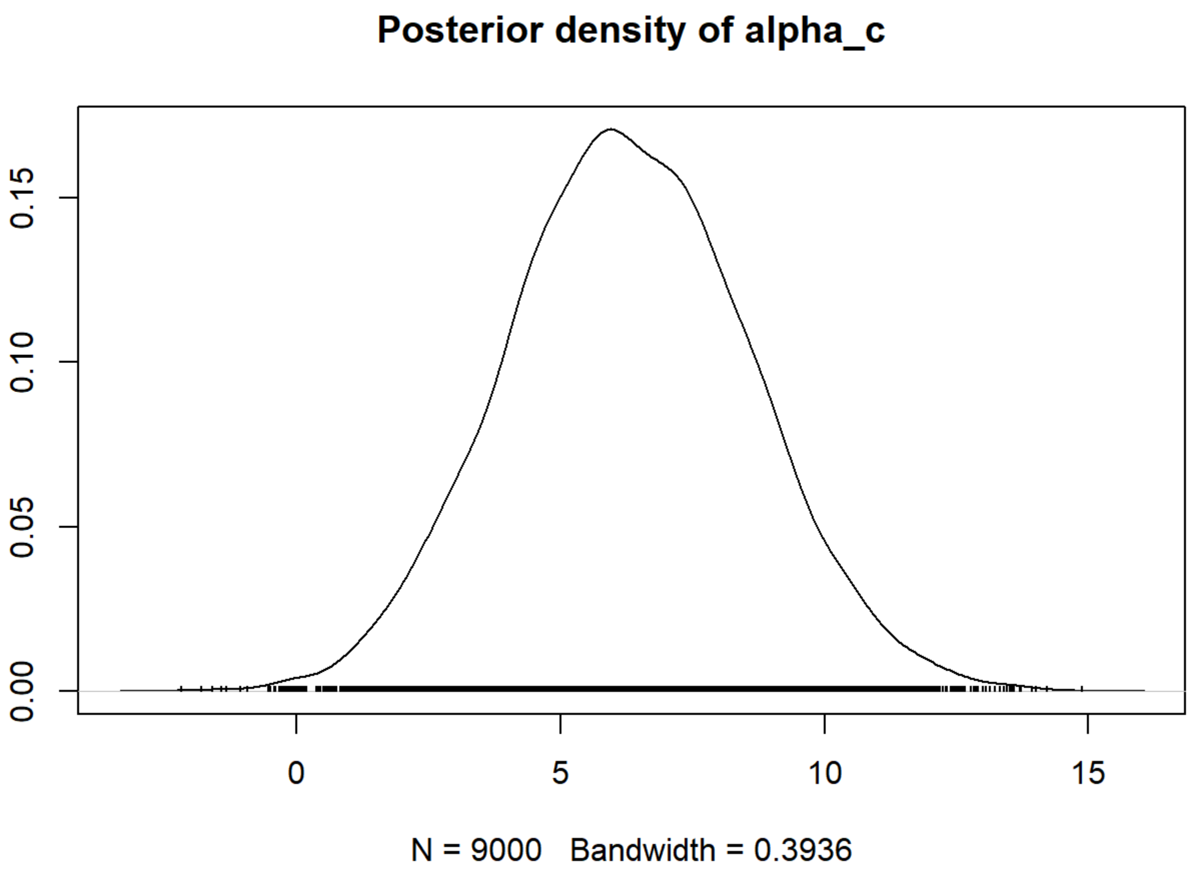
```
# ACF plots for alpha
png("alpha_acf_plots.png", width=1500, height=1000)
par(mfrow=c(6,5))
for (i in 1:30) {
  # ACF for alpha (i-th observation)
  acf(alpha_post[, i], main = paste("ACF for Alpha", i))
}
dev.off()
```

```
## png
##   2
```

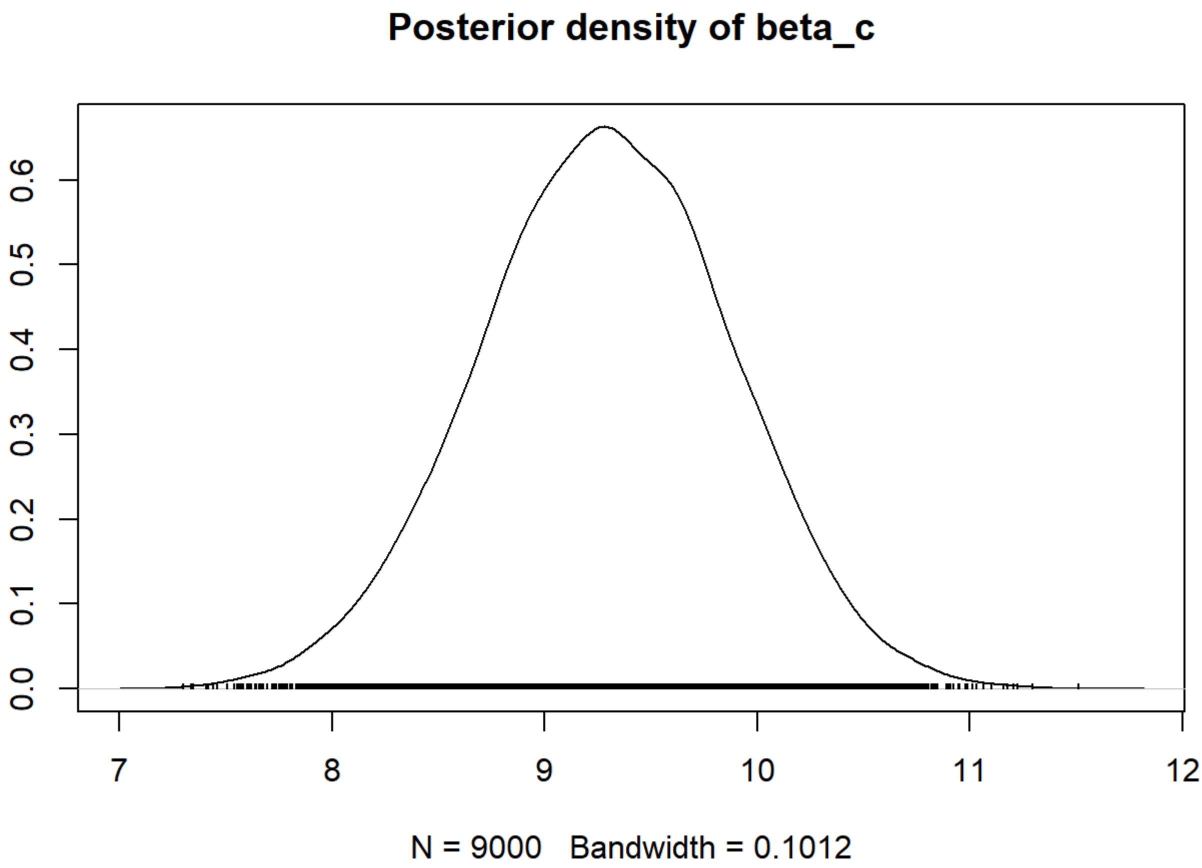
```
# ACF plots for beta
png("beta_acf_plots.png", width=1500, height=1000)
par(mfrow=c(6,5))
for (i in 1:30) {
  # ACF for beta (i-th observation)
  acf(beta_post[, i], main = paste("ACF for Beta", i))
}
dev.off()
```

```
## png
##   2
```

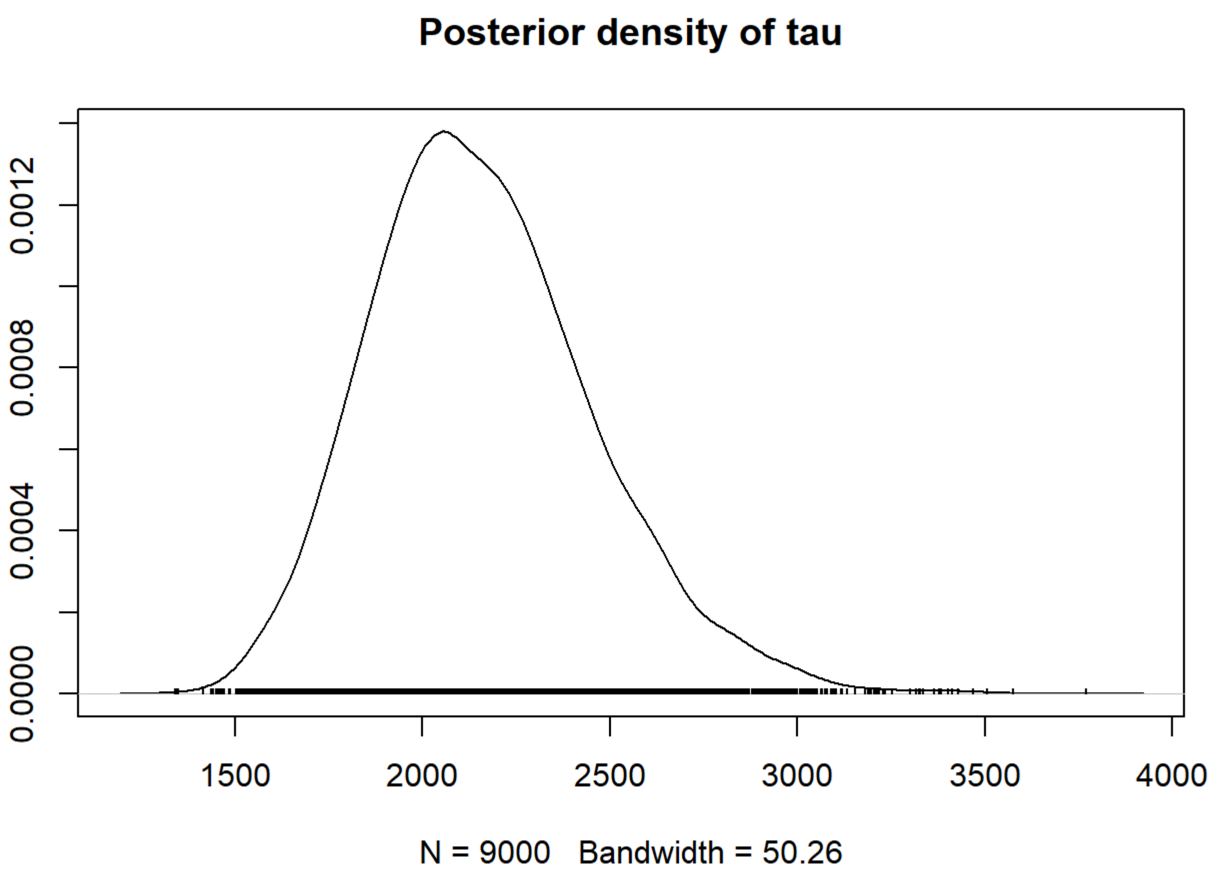
```
# Posterior densities
par(mfrow=c(1,1))
densplot(mcmc_alpha_c, main = "Posterior density of alpha_c")
```




```
densplot(mcmc_beta_c, main = "Posterior density of beta_c")
```



```
densplot(mcmc_tau, main = "Posterior density of tau")
```




```
# Posterior densities for alpha
png("alpha_density_plots.png", width=1500, height=1000)
par(mfrow=c(6,5))
for (i in 1:30) {
  # Density Plot for alpha (i-th observation)
  densplot(mcmc_alpha[, i], main = paste("Density Plot for Alpha", i))
}
dev.off()
```

```
## png
##   2
```

```
# Posterior densities for beta
png("beta_density_plots.png", width=1500, height=1000)
par(mfrow=c(6,5))
for (i in 1:30) {
  # Density Plot for beta (i-th observation)
  densplot(mcmc_beta[, i], main = paste("Density Plot for Beta", i))
}
dev.off()
```

```
## png
##   2
```

```
# Geweke Tests for each parameter

geweke_results_alpha <- apply(alpha_post, 2, function(chain) {
  geweke.diag(mcmc(chain))$z
})

geweke_results_beta <- apply(beta_post, 2, function(chain) {
  geweke.diag(mcmc(chain))$z
})

geweke_results_alpha_c <- geweke.diag(mcmc_alpha_c)$z

geweke_results_beta_c <- geweke.diag(mcmc_beta_c)$z

geweke_results_tau <- geweke.diag(mcmc_tau)$z

print(list(geweke_results_alpha, geweke_results_beta, geweke_results_alpha_c, geweke_results_
beta_c, geweke_results_tau))
```

```
## [[1]]
##  [1] -0.3810342  0.3757453 -0.6466500 -1.7067673 -0.9068367 -1.4256248
##  [7] -1.1590214 -1.6364037 -0.6551472 -0.9101827 -1.8100018 -0.8062032
## [13] -1.1829874 -1.5179295 -0.8851523 -1.1514061 -0.3335064 -1.2697546
## [19] -0.7802281 -1.8412930 -1.5494567 -0.5081327 -0.9544819 -0.5983633
## [25] -0.4354705 -1.9139924 -0.6176586 -0.5264184 -1.3628785  0.1671091
##
## [[2]]
##  [1]  0.716810922  0.403106702  1.726454116  0.761766330  1.879849369
##  [6]  0.615711328  0.256272308 -0.006404438  1.605324468  0.018395054
## [11] -0.349733177  0.614962349  0.882148372 -0.127280873 -1.933425679
## [16]  0.437498473  0.272254332  1.165727064  1.977643040  0.668673325
## [21] -2.157728573  0.423928164 -0.665050906  1.067509748  1.789676462
## [26] -0.972874585  0.342931200  1.519293720  0.370981263 -1.009252332
##
## [[3]]
##      var1
## -1.257055
##
## [[4]]
##     var1
## 2.056386
##
## [[5]]
##       var1
## -0.3676366
```

None of them are far away enough from 0 to worry me too much, so I'm happy to conclude that all these have done a good job of converging!