Case Study 1

Parker Johnson

10/23/2022

Introduction

Test scores are measurements that institutions place lots of value on to try and quantify how well a student has learned the material. Therefore, having an understanding of how well students are collectively learning is important for professors to know if they need to shift anything in their teaching. In this study, we estimate the true mean and standard deviation of test scores in a Stat 120 class and predict a future test score by using the Gibbs sampler in an effort to understand how the class is performing.

Methodology

For this estimation, we applied the Two-stage Gibbs sampler. This estimation seemed appropriate for the study, because the Gibbs sampler can efficiently sample from 2D posterior distributions, so we can estimate both the mean and standard deviation that are in this model.

Before implementing the Gibbs sampler, we needed to choose the model and prior distributions. Since we are estimating the mean and standard deviation, it follows that our model would be a normal distribution with mean μ and standard deviation σ. It made sense that μ would be normally distributed, since test scores likely are distributed symmetrically and could have a decent spread. We will also assume that the mean and standard deviation are independent. The mean for μ, μ_0 was chosen to be 85, because this gives a B average on the test which seems reasonable, and since I am around 95% confident that the mean test score is between 71 and 99, this approximately means that $71 \pm 2\sigma_0 = [71, 99]$. Therefore, the standard deviation for μ, σ_0, could reasonably be 7. Therefore, the prior distribution for the mean was chosen to be $\mu \sim N(85, 7)$.

Since I didn't have any prior knowledge on the standard deviation, a weakly uninformative prior made sense to be chosen. I will work with precision, which is $\phi = \frac{1}{\sigma^2}$, within this model to allow for easier implementation before transferring this back to standard deviations. Since a gamma distribution is commonly used for precisions due to covering all values greater than 0, the weakly uninformative prior distribution for the precision was chosen to be $\phi \sim Gamma(.01, .01)$. Therefore, the full model we are working with is as follows:

$$Y_i|\mu, \phi \overset{\text{iid}}{\sim} N(\mu, \sigma)$$
$$\mu \sim N(\mu_0, \sqrt{1/\phi_0})$$
$$\phi = \frac{1}{\sigma^2} \sim Gamma(a, b)$$

Where $\mu_0 = 85$, $\phi_0 = \frac{1}{7^2}$, $a = .01$, and $b = .01$. Using this model specification, the posterior distribution is given by the following:

$$\pi(\mu, \phi|y_1, ..., y_n) \propto \pi(\mu)\pi(\phi) * \prod_{i=1}^{n} f(y_i|\mu, \sigma^2) \propto exp[-\frac{\phi_0}{2}(\mu - \mu_0)^2] * \phi^{a-1} exp[-b\phi] * \prod_{i=1}^{n} \phi^{1/2} exp[-\frac{\phi}{2}(y_i - \mu)^2]$$

With the full model, we can now implement the Gibbs sampler. The process for sampling from the joint distribution of μ and ϕ using the Gibbs sampler is as follows:

1. Choose initial values for $\mu^{(0)}$ and $\phi^{(0)}$.
2. Simulate $\mu^{(j)}$ from the conditional posterior distribution of μ, which is $\pi(\mu^{(j)}|\phi^{(j)}, y_1, ..., y_n) \propto exp[-\frac{\phi_0+n\phi^{(j)}}{2} * (\mu - \frac{\mu_0\phi_0+n\bar{y}\phi^{(j)}}{\phi_0+n\phi^{(j)}})^2]$ in this case.
3. Given the current simulated value $\mu^{(j)}$, simulate $\phi^{(j+1)}$ from the conditional posterior distribution of ϕ, which is $\pi(\phi^{(j+1)}|\mu^{(j)}, y_1, ..., y_n) \propto \phi^{(n/2+a)-1} exp[-\phi(\frac{1}{2}\sum_{i=1}^{n}(y_i - \mu^{(j)})^2 + b)]$ in this case.
4. Repeat 2-3 S times.

With this process, the posterior distribution of μ and ϕ can be estimated, which can in turn be used to estimate the posterior distribution of σ and produce a prediction interval for a randomly selected future exam score. The prediction interval was found by calculating the 0.05 and 0.95 quantiles from a simulated normal distribution with the estimated μ and σ from above. I chose to find a 90% prediction interval for the future test score, as this is a commonly used prediction interval and will give us a good sense of what are probable values for the future score.

Results

The results of the simulated draws from the Gibbs sampler can be seen in Figure 1. From these simulated draws, the largest estimated mean was 95.2, and the largest estimated standard deviation was 17.2. The smallest estimated mean was 80.3, and the smallest estimated standard deviation was 6.07.

From these draws, the average exam score of these simulated draws is approximately 87.1, with a standard deviation of approximately 8.79. Given the 29 exam scores, there is a 90% chance that a randomly selected future score is between 72.9 and 100 (assuming that students cannot score above 100).

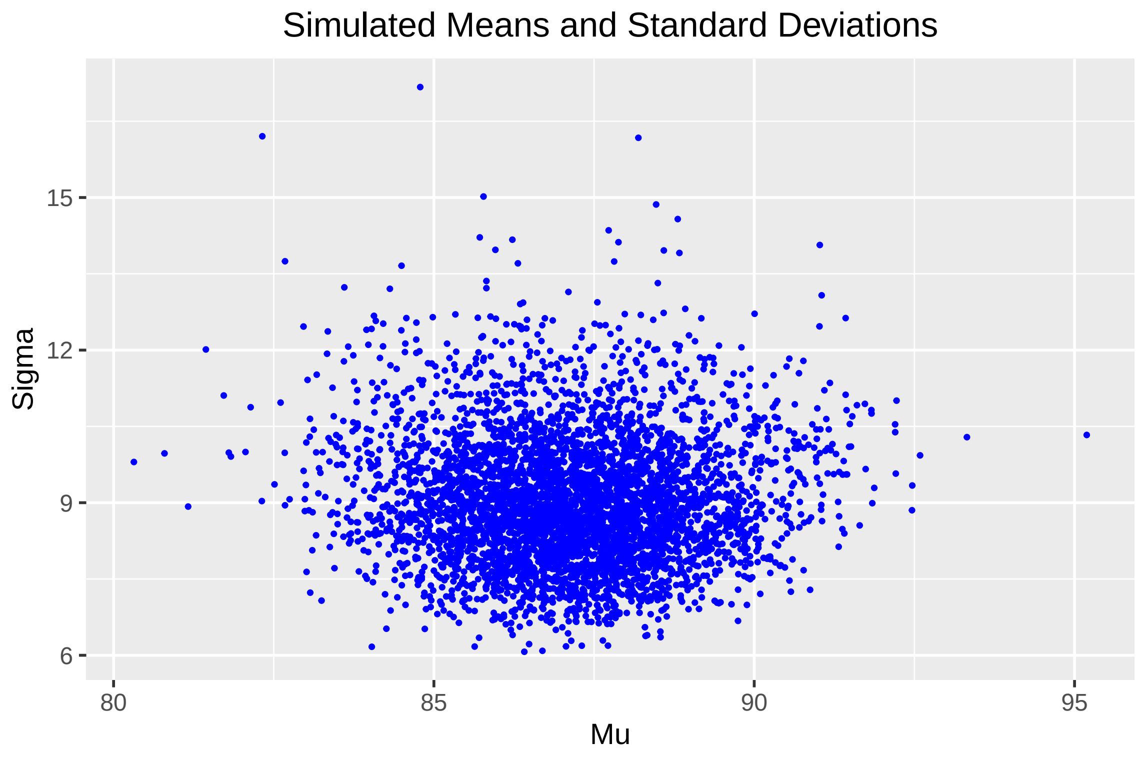


Figure 1: Scatterplot of each pair of simulated draws.

Discussion

We estimate that the average exam score is 87.1, with a standard deviation of 8.79. We also estimate that there's a 90% chance that the exam test score for a randomly selected future Stat 120 student is between

72.9 and 100. This means that the average score on the test was a B+, and a randomly selected student in the future has a 90% chance of scoring at least a 72.9.

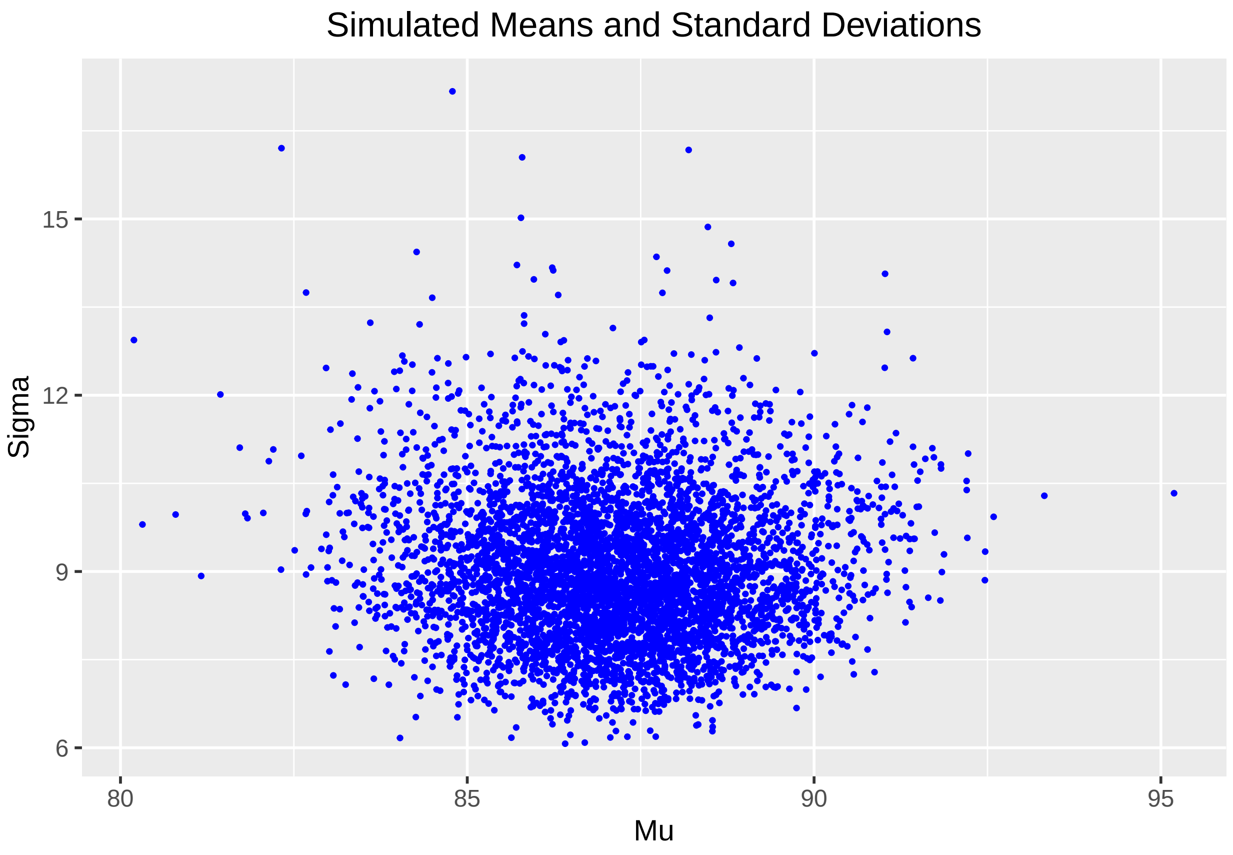
One drawback of using a Gibbs sampler is that it can only approximate the posterior distribution. While the diagnostics (see appendix) indicate that this is a close approximation to the posterior distribution, it isn't exactly equal to the posterior distribution. Therefore, the values found aren't perfectly accurate to the true values, just approximations. If this test were to be conducted again, I would run more iterations of the Gibbs sampler to get as close as possible to the exact distribution as I could.

The main drawback to this design is the possibility for inaccuracies to arise when justifying the prior distribution and model. While a weakly informative prior does not have a large impact on the posterior distribution, it does sway it slightly, so an incorrect prior distribution will result in a sightly inaccurate posterior distribution and interpretations. Therefore, perhaps more research could have been conducted to find more confident prior distributions for both μ and ϕ; for example, perhaps 7 was too narrow for the standard deviation of μ's distribution, and it's possible that a gamma distribution was not the correct distribution to choose for ϕ. More research would need to be done, especially with regards to ϕ, to find accurate prior distributions.

Technical Appendix

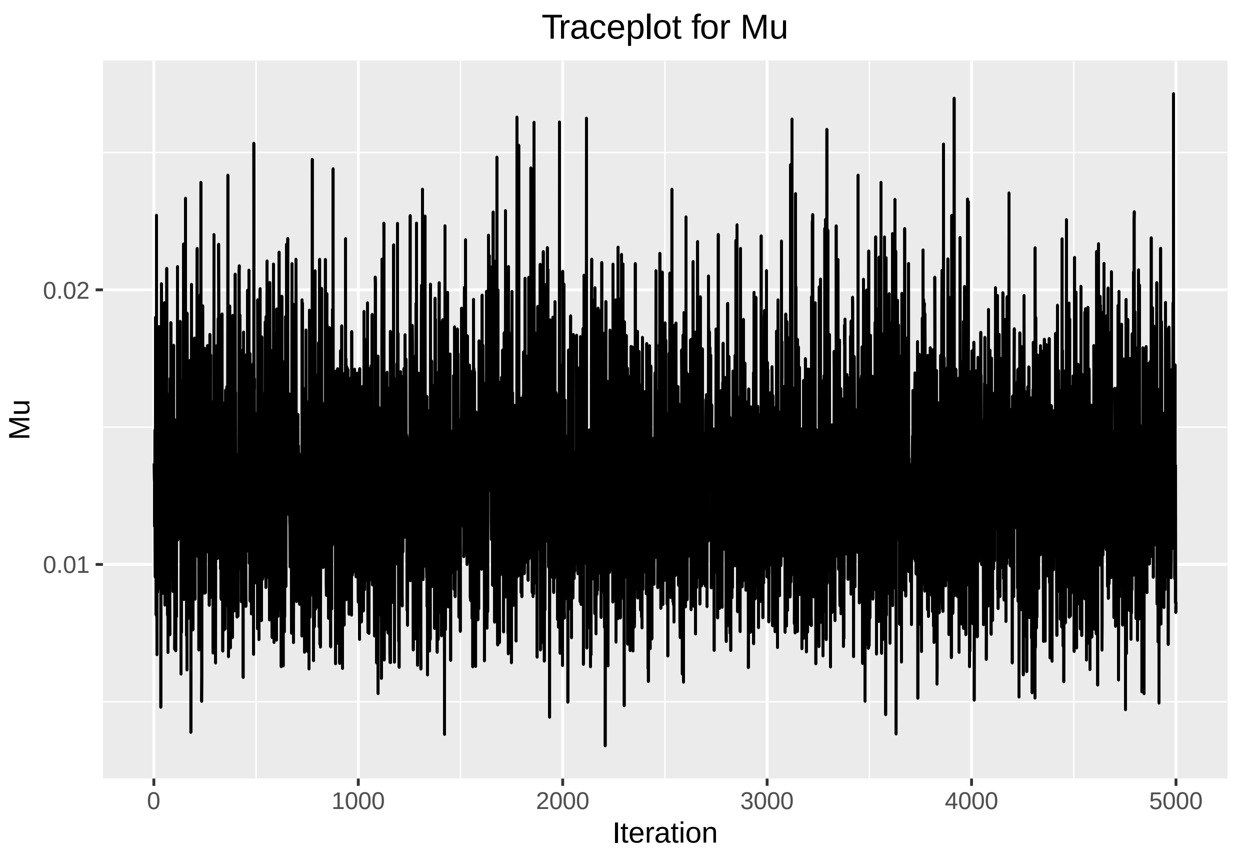
After specifying the prior distributions, I implemented the Gibbs sampler based on the normal sampling example found in Probability and Bayesian Modeling (Albert and Hu, 2020). The plot is a scatterplot of each simulated draw.

```r
scores <- read.csv("http://aloy.rbind.io/data/test_scores.csv")

gibbs_normal <- function(s, phi = 0.002, iter = 1000){
  ybar <- mean(s$y) #Data
  n <- length(s$y)
  mu0 <- s$mu0      #Prior specifications
  phi0 <- s$phi0
  a <- s$a
  b <- s$b
  x <- matrix(0, iter, 2) #Empty matrix for draws
  for(k in 1:iter){
   mun <- (phi0 * mu0 + n * phi * ybar) /
      (phi0 + n * phi) #Mean for first conditional density
    sigman <- sqrt(1 / (phi0 + n * phi)) #Std for first conditional density
    mu <- rnorm(1, mean = mun, sd = sigman) #Draw for mu^i
    an <- n / 2 + a
    bn <- sum((s$y - mu) ^ 2) / 2 + b
    phi <- rgamma(1, shape = an, rate = bn) #Draw for phi^i
    x[k, ] <- c(mu, phi) #Adding the pair of draws to the matrix
  }
  x
}

#Adding data and priors
s <- list(y = scores$score, mu0 = 85, phi0 = 1/7^2, a = .01, b = .01)
set.seed(597)
out <- data.frame(gibbs_normal(s, iter=5000)) #Making the sampler a data frame
names(out) <- c("Mu", "Phi")

#Scatterplot
ggplot(out, aes(x = Mu, y = 1/sqrt(Phi))) +
  geom_point(color = "blue", size = 0.5) +
  labs(x = "Mu", y = "Sigma", title = "Simulated Means and Standard Deviations") +
  theme(plot.title = element_text(hjust = 0.5))
```

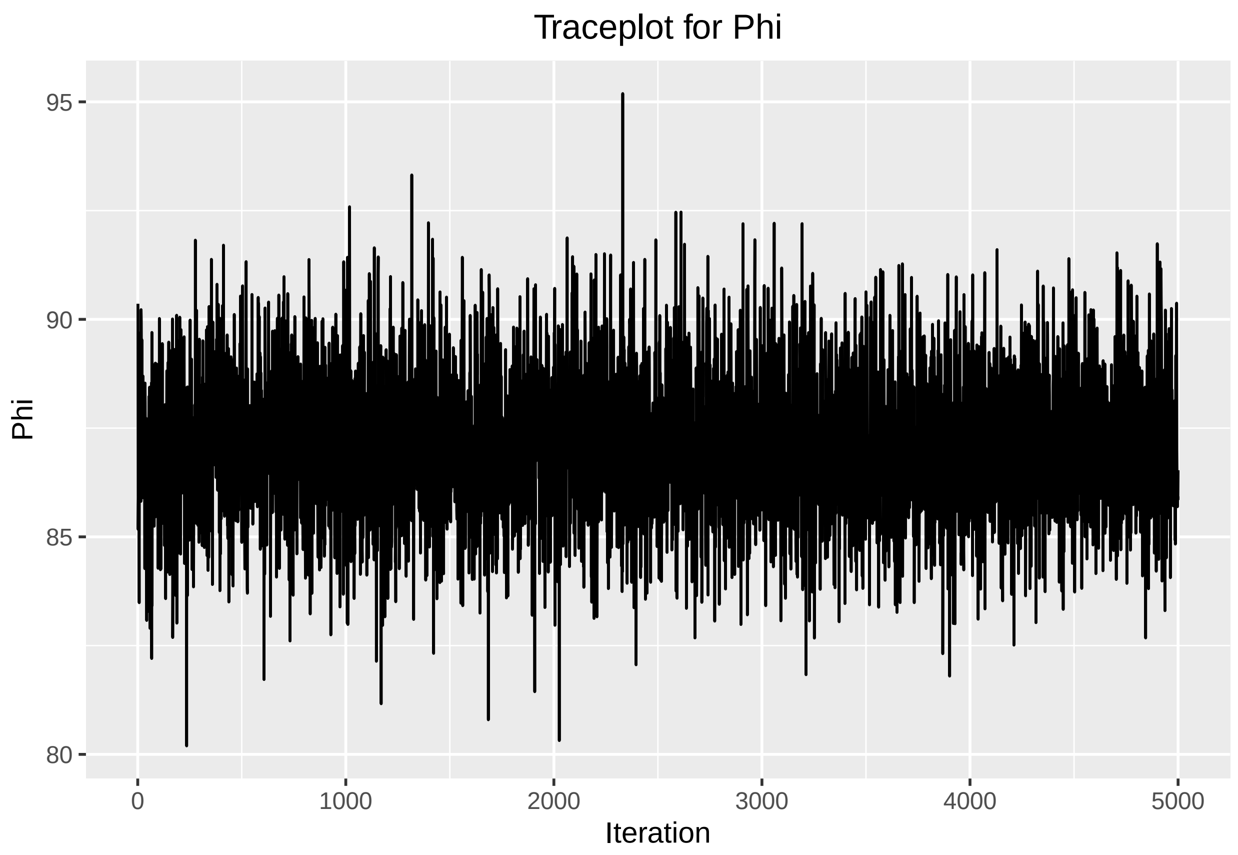


Diagnostics needed to be conducted to ensure that the Gibbs sampler properly converged to a stationary mean. The trace plots indicate that the Gibbs sampler converged fairly quickly to white noise, and the ACF plots indicate that there isn't high correlation since it tapers off quickly. Therefore, this Gibbs sampler is sufficient for analysis.

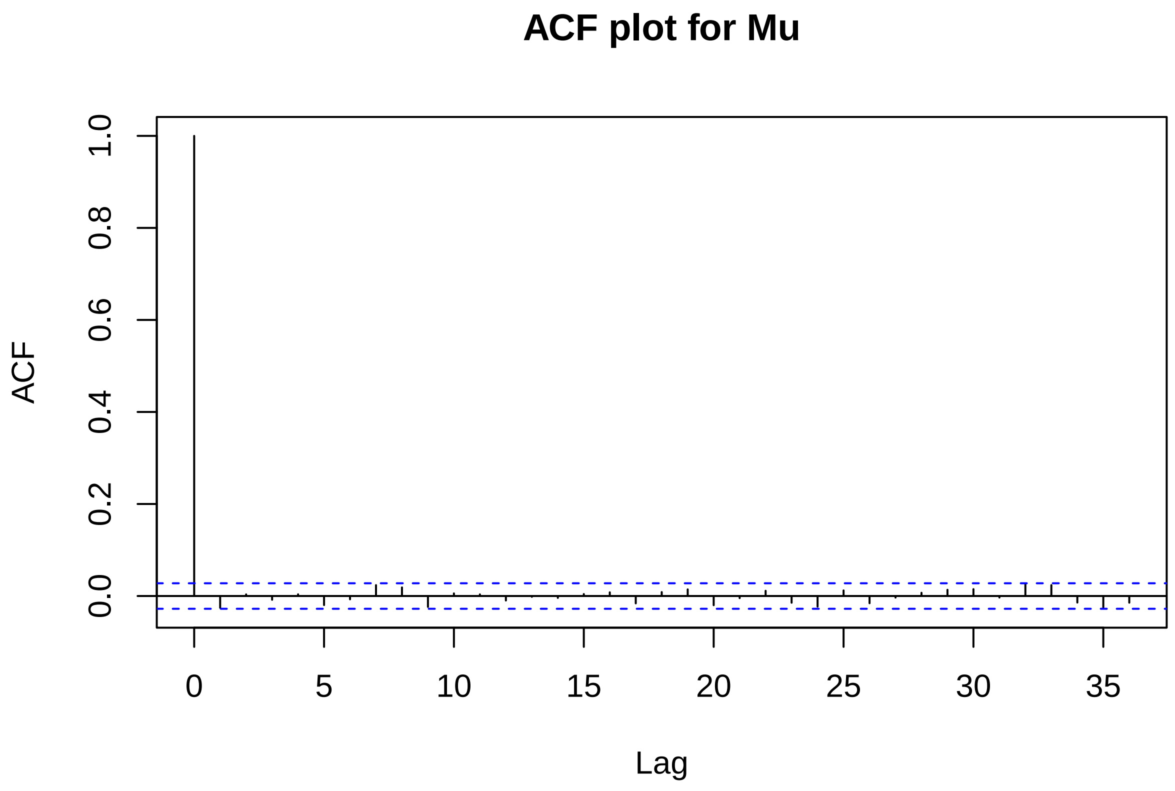
```
#Traceplots
ggplot() +
  geom_line(aes(x = seq_along(out$Phi), y = out$Phi)) +
  labs(x = "Iteration", y = "Mu", title = "Traceplot for Mu") +
  theme(plot.title = element_text(hjust = 0.5))
```



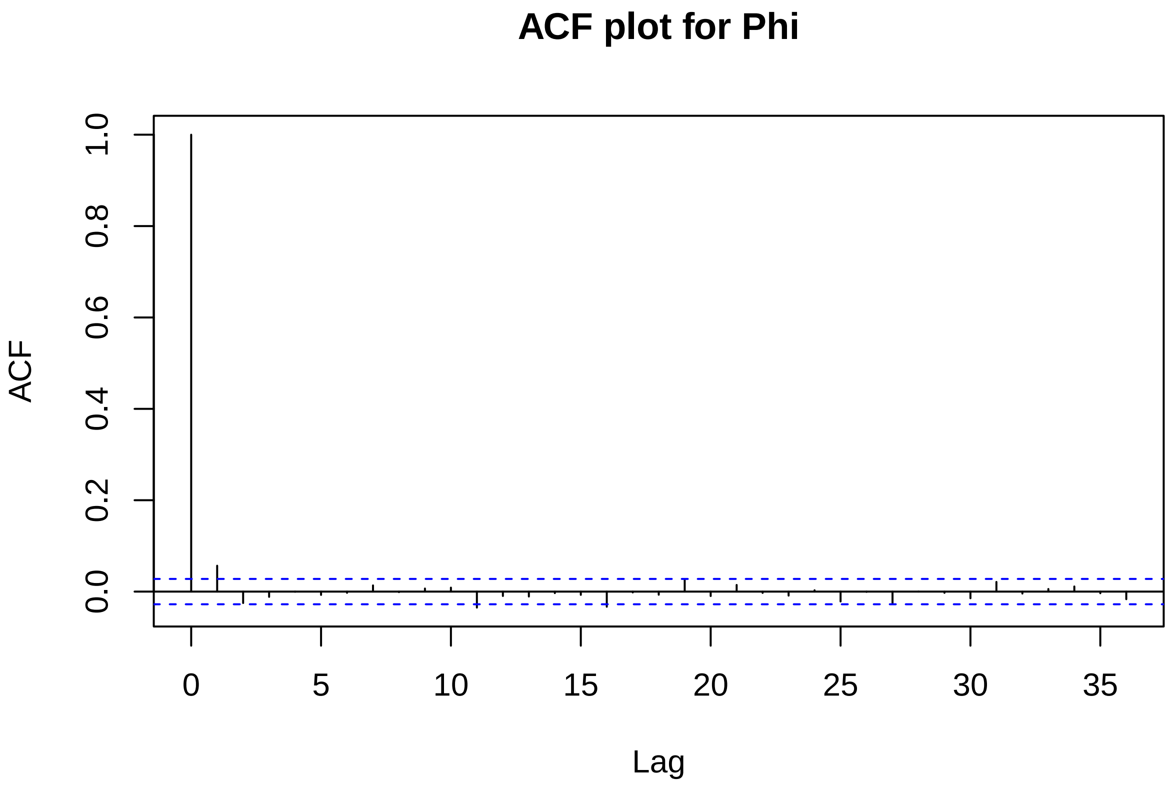
```
ggplot() +
  geom_line(aes(x = seq_along(out$Mu), y = out$Mu)) +
  labs(x = "Iteration", y = "Phi", title = "Traceplot for Phi") +
  theme(plot.title = element_text(hjust = 0.5))
```



```r
#ACF plots
acf(out$Mu, main = "ACF plot for Mu")
```

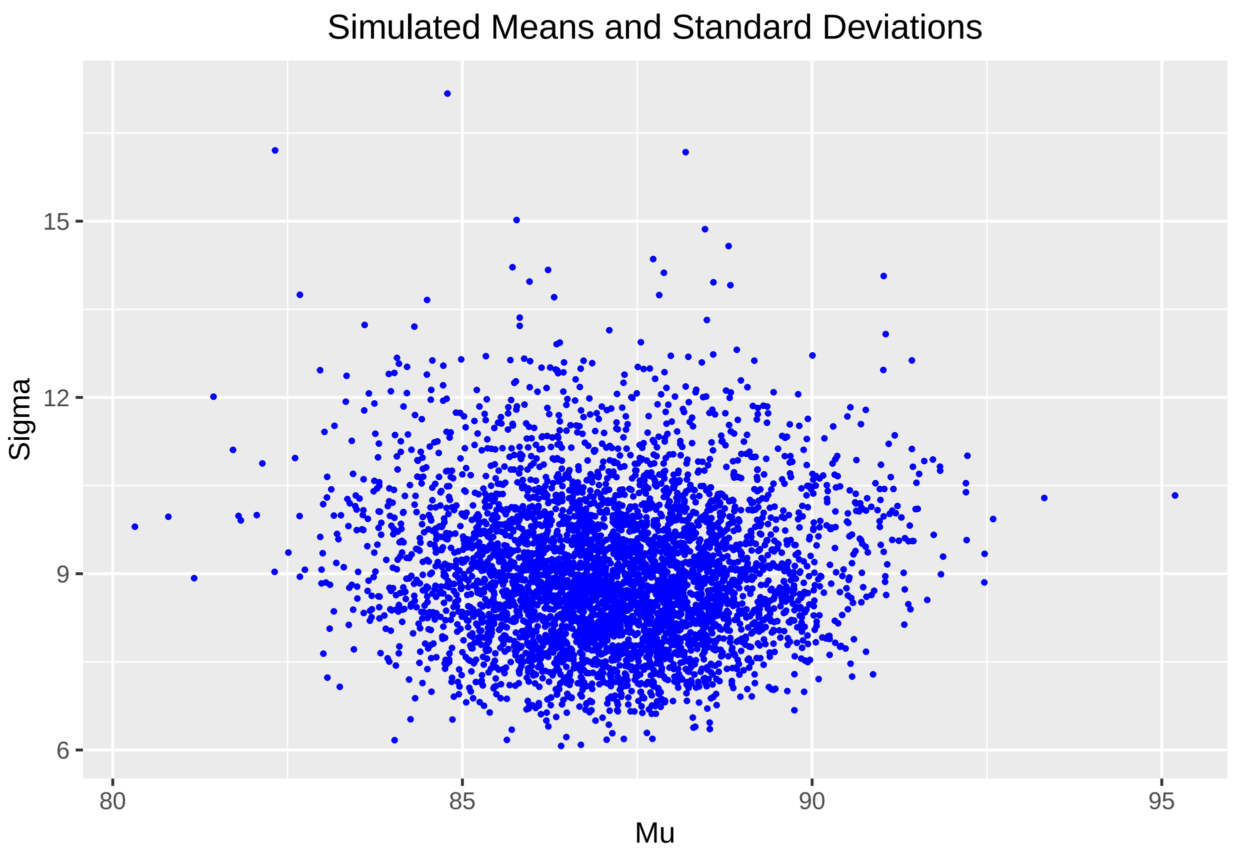


```r
acf(out$Phi, main = "ACF plot for Phi")
```



While this sampler converged quickly, the first 500 values were thrown out to ensure that the burn in period is fully removed. The scatterplot is shown below that does not include the burn in period.

```r
out_burnout <- out[-c(1:500), ] #Removing burn in period

#Scatterplot without burn in period
ggplot(out_burnout, aes(x = Mu, y = 1/sqrt(Phi))) +
  geom_point(color = "blue", size = 0.5) +
  labs(x = "Mu", y = "Sigma", title = "Simulated Means and Standard Deviations") +
  theme(plot.title = element_text(hjust = 0.5))
```



To fully check the adequacy of this model, we need to check how well the data matches the model. This can be done through posterior predictive checking, in which we simulate a number of replicated datasets from the posterior predictive distribution and see how the observed sample compares to the replications. If the observed data resembles the replications, we can say that the observed data is consistent with predicted data from the model.

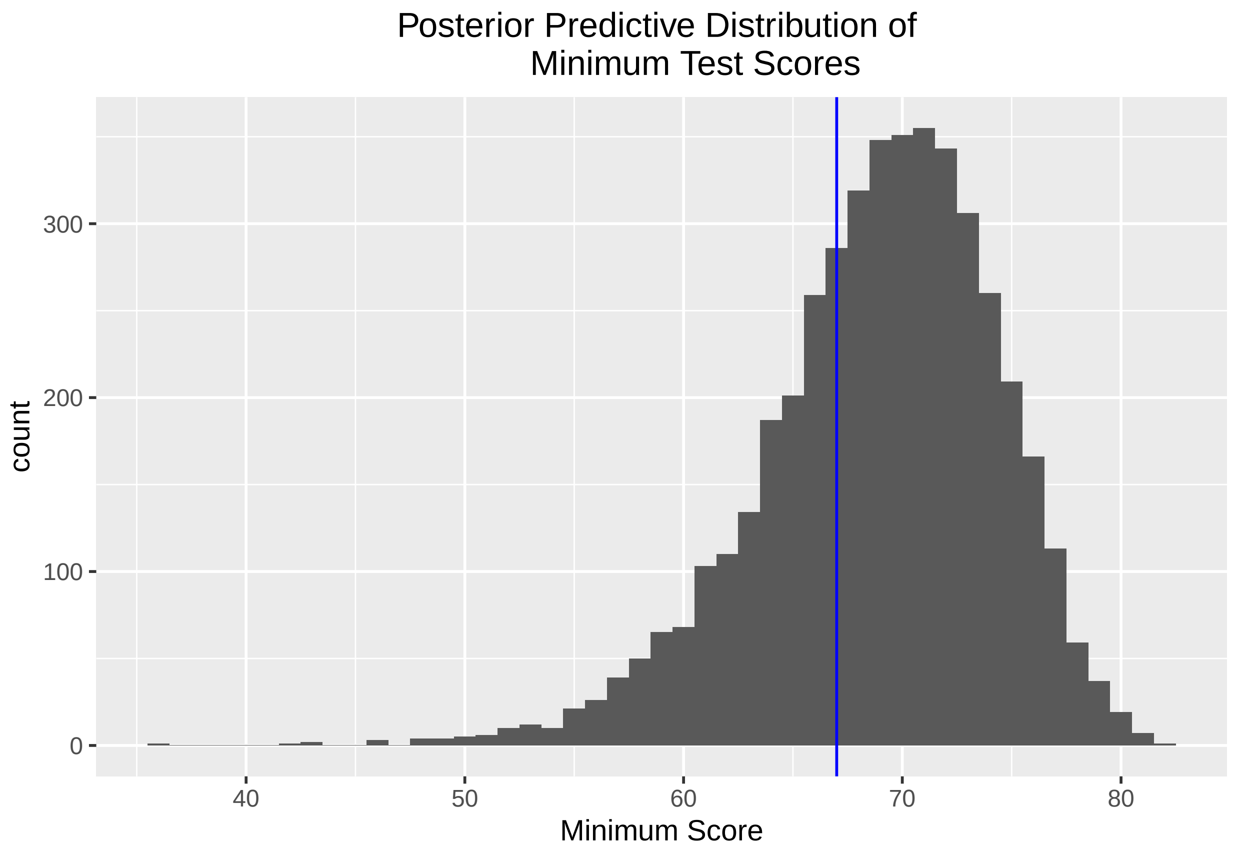
These are the steps for posterior predictive checking:

1. Draw a set of parameter values, in this case μ^* and σ^*, from the posterior distribution of (μ^*, σ^*).
2. Given these parameter values, simulate $\tilde{y}_1, ..., \tilde{y}_{27}$ from the normal sampling density with mean μ^* and standard deviation σ^*.
3. Repeat this multiple times.

We then check some function that will help us detect possible discrepancies between the observed data and the simulated prediction samples. Since the maximum of the simulated prediction samples could be over 100 while students couldn't score over 100 in actuality, we can instead use the minimum value as the checking function. The histogram is shown below of the observed minimum in blue versus the minimum of each simulated prediction samples. The observed minimum of our data could plausibly have been generated from this distribution, so the model appears to be adequate for our data.

```
postpred_sim <- function(j){
  rnorm(29, mean = out_burnout[j, "Mu"],      #Drawing 29 mu*s
        sd = 1/sqrt(out_burnout[j, "Phi"]))  #Drawing 29 sigma*s
}

set.seed(597)
ypred <- t(sapply(1:4500, postpred_sim)) #Doing this 4500 times
postpred_min <- apply(ypred, 1, min)
#Finding the minimum value for each replicated dataset
```

```r
#Plotting the observed mimimum versus the mimimum from each replicated dataset
ggplot(data = NULL, aes(x = postpred_min)) +
  geom_histogram(binwidth = 1) +
  geom_vline(xintercept = min(scores$score), color = "blue") +
  labs(x = "Minimum Score", title = "Posterior Predictive Distribution of
       Minimum Test Scores") +
  theme(plot.title = element_text(hjust = 0.5))
```



This is calculating the minimum and maximum simulated μ and σ values. While this wasn't necessary, these were calculated to better understand the spread of values from the simulated posterior distribution.

```r
max(out_burnout$Mu)
```

```
## [1] 95.19024
```

```r
1 / sqrt(min(out_burnout$Phi))
```

```
## [1] 17.17155
```

```r
min(out_burnout$Mu)
```

```
## [1] 80.31724
```

```r
1 / sqrt(max(out_burnout$Phi))
```

```
## [1] 6.069325
```

These are the estimated values for μ and σ.

```
mean(out_burnout$Mu)
```

```
## [1] 87.13441
```

```
1 / sqrt(mean(out_burnout$Phi))
```

```
## [1] 8.791931
```

This is the distribution and calculation of the 90% prediction interval for the future exam score of a randomly selected student.

```
pred_int <- rnorm(5000, 87.13441, 8.791931)
quantile(pred_int, probs = c(0.05, 0.95))
```

```
##        5%       95%
##  72.29433 101.95860
```