The Rose Garden Event: A Hierarchical Bayesian Approach to Modeling Positive Coronavirus Tests

Jedidiah Harwood* Eric A. Suess†

Abstract

We present an example of a Bayesian hierarchical model, for an advanced undergraduate Bayesian Statistics class, with a focus on explaining the variability in coronavirus testing. This demonstration was applied to data from various Trump events throughout 2020, where testing and social distancing were not enforced. Using R and the JAGS software for Gibbs Sampling, we developed predictive, posterior distributions for the total number of people who would have tested positive for coronavirus at each (and every) event. Additionally, we developed posterior distributions for the coronavirus test sensitivity and specificity. The MCMC chains appeared to properly converge to the posterior distributions, and was further suggested so, by the Gelman-Rubin statistic.

Key Words: Bayesian, Hierarchical model, Statistics Education, MCMC, Gibbs Sampling

1. Introduction

On September 26, 2020, then-President Trump held a party in the White House Rose Garden, to celebrate the nomination of Amy Coney Barrett to the Supreme Court. It would later become known as a "super spreader" event.

Over 300 people were in attendance, and every guest tested negative for coronavirus before entry. Subsequently, numerous attendees contracted coronavirus. This poses the following question: If every attendee at the Rose Garden event tested negative, then why did a subsequent coronavirus outbreak occur?

The answer lies in the fact that the coronavirus tests are imperfect. This, in essence, is the motivation for our example.

To assess this situation, we built a Bayesian hierarchical model to produce posterior estimates for the coronavirus test *sensitivity* and *specificity*, the underlying probability of testing positive, the underlying prevalence of coronavirus in the population, a *predictive distribution* for the number of people who may have tested positive for coronavirus, given a set of event sizes and the number of people in each event who tested positive for the coronavirus. These posterior estimates were found by using a Gibbs Sampler. To apply our model, we gathered data on the events that Trump had held during the height of the pandemic. However, these events did not enforce coronavirus testing. As a result, we implemented a simulation to estimate the number of people who would have tested positive at these events, in order to generate our posterior estimates.

After generating our predictive and posterior estimates, we utilized trace plots, as well as the Gelman-Rubin statistic to demonstrate that the MCMC chains had properly converged.

This model can serve as an example of a Bayesian hierarchical model for an advanced undergraduate Bayesian Statistics class, and demonstrates the versatility of Bayesian models to overcome issues that would have led to difficulties in a frequentist approach, in particular through the use of *latent variables*.

*Department of Statistics, University of California, Davis, 1 Shields Ave, Davis, CA 95616

†Department of Statistics and Biostatistics, California State University, East Bay, 25800 Carlos Bee Blvd, Hayward, CA 94542

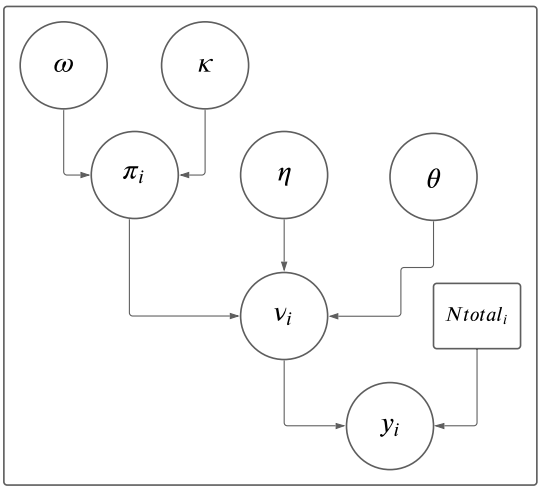

Figure 1: Diagram of Model Parameters

2. What is the Model?

2.1 Parameters

To represent the number of people who had tested positive for the coronavirus at each event, *i*, we used y_i.

$$y_i \sim binomial(\nu_i, Ntotal_i)$$

We modeled y_i with a binomial distribution, under the assumption that there was some underlying, varying, probability of testing positive (ν_i), at each event, *i*. Associated with each underlying probability with testing positive, is the event's size ($Ntotal_i$). This binomial distribution serves as the likelihood function for our model parameters.

An important aspect of diagnostic testing, is the accuracy of the test, which are accounted for through the sensitivity and specificity of the test. *Sensitivity* represents the probability for one to test positive for the disease, given that they actually have contracted the disease. *Specificity* represents the probability for one to test negative for the disease, given that they have not contracted the disease. Here, we represent the *sensitivity* with η, and the *specificity* with θ. We used the following prior distributions.

$$\eta \sim beta(910, 90)$$

$$\theta \sim beta(950, 50)$$

For the model, we placed beta prior distributions on η and θ, due to its simplicity and convenience. The prior distribution for the sensitivity had a mode of 91%. The prior distribution for the specificity had a mode of 95%.

For each of the Trump events in 2020, we assumed that there was a varying, underlying *prevalence* for the coronavirus in the population. To represent the coronavirus prevalence at each event, i, we used π_i.

$$\pi_i \sim beta(\omega(\kappa - 2) + 1, (1 - \omega)(\kappa - 2) + 1)$$

Rather than use the typical α and β shape parameters for the prior placed upon π_i, we used the mode of the distribution (ω), and the concentration of the distribution (κ) so as to more easily specify our prior beliefs on the prevalence of the coronavirus. Using ω and κ, we can reparameterize with respect to α and β, as shown in the equation above.

In our model, we placed prior distributions on ω and κ. As a result, π_i is dependent on both ω and κ.

To represent the underlying probability of testing positive at each event, we used ν_i. With π_i, η, θ, and the law of total probability, we formulated the prior distribution of ν_i.

$$\nu_i = \eta\pi_i + (1-\theta)(1-\pi_i)$$

Due to the formulation of ν_i, its prior distribution is dependent on π_i, η, and θ.

As mentioned earlier, we used ω and κ to represent the mode and concentration of the prior distribution put upon π_i.

$$\omega \sim beta(6, 95)$$

$$(\kappa - 2) \sim gamma(5.8, 0.48)$$

We placed a beta prior on ω, with the mode of the distribution centered at around 6%. Instead of placing a prior distribution on strictly κ, we placed a gamma prior on $(\kappa-2)$, and specified in our implementation of the model, the relationship between the κ and $(\kappa - 2)$.[1]

2.2 Predictive, Posterior Distributions

Our model produces predictive, posterior distributions for the number of people who would test positive for coronavirus – within each and throughout all of the events. To represent the number of positive tests within each event, we used Py_i. To represent the number of positive tests throughout all the events, we used Py_{tot}.

$$Py_i \sim binomial(\nu_i, Ntotal_i)$$

$$Py_{tot} = \sum_{i=1}^{N} Py_i$$

2.3 Limitations

We assumed that for every event, the same type of test was used. We also made the assumption that the underlying coronavirus prevalence distribution, was the same for each event. If we were to apply this model to an event where social distancing (or other precautionary measures) took place, it is possible that the underlying prevalence distribution would be different from that of a non-socially distant event.

3. Implementation

3.1 The Data

To apply our model, we collected data on 73 different Trump events (including the Rose Garden event). These events varied in size. Presented in Table 1 is a numerical summary of the event sizes. The smallest of the 73 events had 110 participants, while the largest had 30000 participants.

Table 1: Summary of Event Sizes

Min	Q_1	Median	Q_3	Max
110	1200	5000	7000	30000

[1] Figure 1 does not explicitly show this relationship in the hierarchy.

Unfortunately, as these events did not enforce coronavirus testing, there was no data on the number of people who tested positive at these events. Subsequently, we simulated the number of people who would have tested positive at these events, using some reasonable assumptions, and a binomial distribution.

To simulate the data, we randomly generated an underlying probability of testing positive for each event. These probabilities were randomly sampled from a beta distribution, with a mode of 5%. From there, we used each event's simulated probability of testing positive and size, to create a binomial distribution from which we simulated the data.

$$P(+)_i \sim beta(5.9, 94.1)$$

$$y_i \sim binomial(P(+)_i, Ntotal_i)$$

Using R, we implemented the simulation.

```
# Reading in Data
covid19trump <- read.csv("whitehouse_covid_events.csv")

# Number of Events
k <- nrow(na.omit(covid19trump))

# Generating P(+)
prob_nu <- rbeta(k, 5.9, 94.1)

# Loading different event sizes
n <- na.omit(covid19trump$Number_of_Participants)

# Simulating the Number of Positive Cases in Each Event
positive_cases_sim <- function(x){
  rbinom(1,x, prob_nu)
}
y <- lapply(n, positive_cases_sim)
```

3.2 JAGS Implementation

To compute our posterior estimates, we used JAGS, accessed through R packages *rjags* and *runjags*.[2]

In order to specify the model to JAGS, we used a text file that contained the structure of our model. From there, we specified the initial values of our model parameters.[3]

```
modelString <- "
model {
  for (i in 1:k){
    # Likelihood in Code
    y[i] ~ dbin(nu[i], Ntotal[i])

    # Probability of Testing Positive
    nu[i] = eta*pi[i] + (1-theta)*(1-pi[i])

    # Prior for Prevalence
```

[2]We recommend using *runjags*, as it allows parallel computing.

[3]We set initial values of $\eta = \theta = .95$, $\omega = 0.05$, and $\kappa = 10$.

```
    pi[i] ~ dbeta(omega*(kappa-2) + 1,
                  (1-omega)*(kappa-2) + 1)

    # Predictive Distribution
    Py[i] ~ dbin(nu[i], Ntotal[i])
  }
  # Mode of the Beta Prior on pi[i]
  omega ~ dbeta(6, 95)

  # Concentration for the Beta Prior on pi[i]
  kappa = kappaMinusTwo+2

  # (kappa - 2) Cannot be Negative
  kappaMinusTwo ~ dgamma(5.8, .48)

  # Priors for Test Parameters
  eta ~ dbeta(910, 90)
  theta ~ dbeta(950, 50)

  # Predictive Distribution
  Py_tot = sum(Py)
}
"

# Write to File
writeLines(modelString, con="model.txt")

# Implement using runjags
library(runjags)
runJagsOut <- run.jags( method="parallel",
 model = "model.txt",
 monitor = c("nu","pi","omega","kappa",
               "eta","theta","Py", "Py_tot"),
 data = dataList,
 inits = initsList,
 n.chains = nChains,
 adapt = nAdaptSteps,
 burnin = nBurninSteps,
 sample = ceiling(nUseSteps/nChains),
 thin = nThinSteps,
 summarise = FALSE,
 plots = FALSE)

# Convert to MCMC List Object
codaSamples = as.mcmc.list(runJagsOut)
```

Although not explicitly shown, we supplied *runjags* initial values for our model's parameters, in the form of a list object. For more information on how to use *runjags*, we recommend viewing the package's R documentation.

4. Posterior Estimates

4.1 Highest Density Intervals

After running the Gibbs Sampler, we used the R packages *ggplot2* and *tidybayes* to asses our results. Presented in Table 2 is a 95% Highest Density Interval (HDI) for the mode of our posterior estimates. It is important to note, that this 95% probability interval has a different, more intuitive interpretation than a frequentist confidence interval.

From the 95% HDI for the sensitivity η, we can see that there is a 95% probability, that the test sensitivity fell between 88.8% and 92.43%. This implies that it is plausible, for upwards of 10% of all tests to return false negative results. With such an imperfect test sensitivity, it is of no question, that an event such as the Rose Garden event would be possible – where every attendee tested negative, yet may have had coronavirus.

Table 2: 95% HDI Intervals for Mode of Posterior Estimates

Posterior	Estimate	Lower HDI	Upper HDI
η	0.9076	0.8880	0.9243
θ	0.9510	0.9489	0.9539
ω	0.0025	0.0008	0.0053
κ	76.9259	57.9097	99.0559
Py_{tot}	24,989	24,543	25,400
Py_1	33	20	49
ν_1	0.0671	0.0515	0.0893
π_1	0.0218	0.00247	0.0470
.	.	.	.
.	.	.	.
Py_{73}	543	478	599
ν_{73}	0.0540	0.0501	0.0582
π_{73}	0.0059	0.0009	0.0118

In addition to possible false negatives, we can see that the tests were also prone to false positives. According to the HDI for θ, it is likely that upwards of 6% of tests were false positives.

The *latent variables*, π_i, the underlying true *prevalence* of the population at the time of event i, and v_i, the underlying probability of a randomly selected person from the population testing positive for event i were estimated for each event. The values for $i = 1$ and $i = 73$ are included in Table 2.

4.2 Density Plots

In Figure 2 is a density plot of the posterior distribution of Py_{tot}. The golden bar in the plot represents the 95% HDI for the mode of the distribution. We can see that around 11,600 people would have tested positive for the coronavirus, as a result of all the Trump events. However, this number includes false positive results. Based on the 95% HDI, we can conclude that it was 95% likely that between $24,543$ and $25,400$ people would have tested positive.

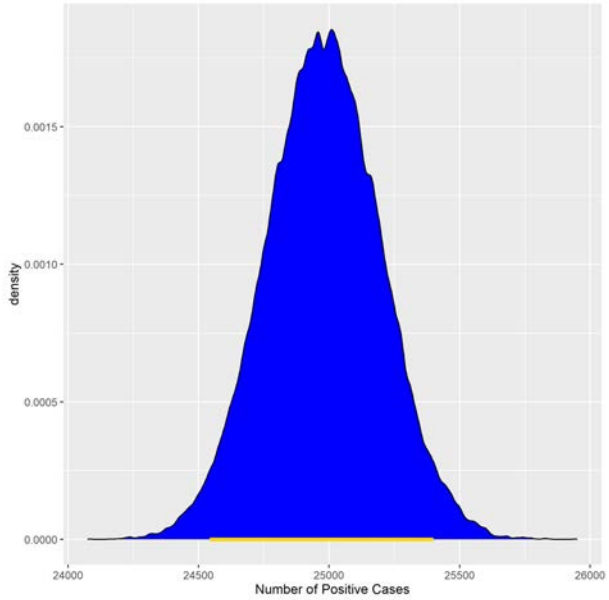

Figure 2: Density Plot of Py_{tot}

As evident from Figure 3, there is some variability in the number of people who would have tested positive, between the different events. This variability is mainly due to the differing event sizes. For example: You may notice that the distribution of π_7 is significantly shifted to the right of the others. This is because the seventh event had significantly more attendees than the others – and therefore, more people who could test positive.

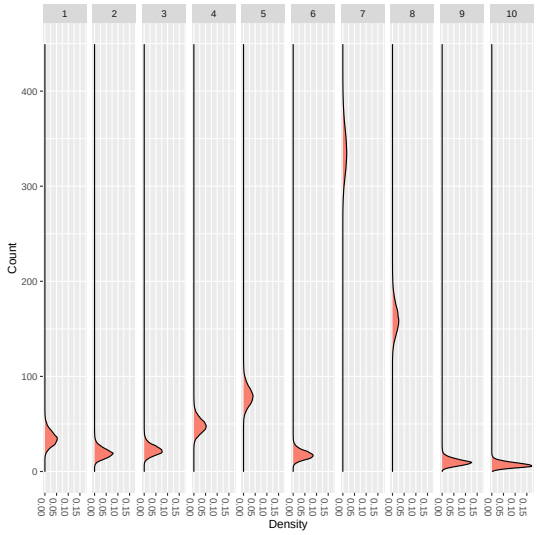

Figure 3: Density Plots for Py_i

From Figure 4, we can see the scope of the imperfections in the coronavirus tests used. Due to the amount of false positives, the distribution of ν_i is significantly shifted to the right of π_i.

As shown in the density plot, the posterior distribution for θ is shifted to the right of η. We can gather that the posterior distribution of η has wider range of credible values than θ.

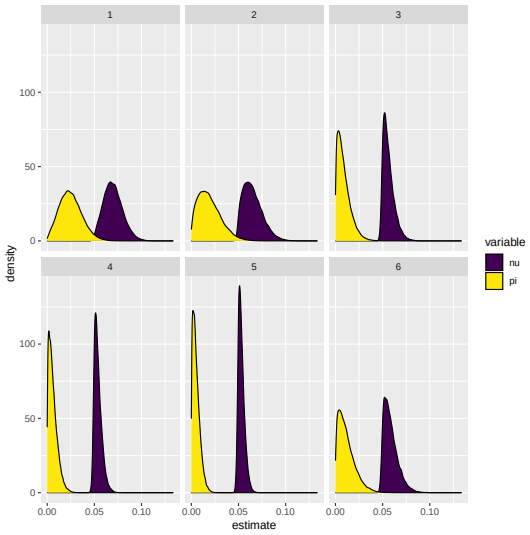

Figure 4: Overlaid Densities of π_i and ν_i

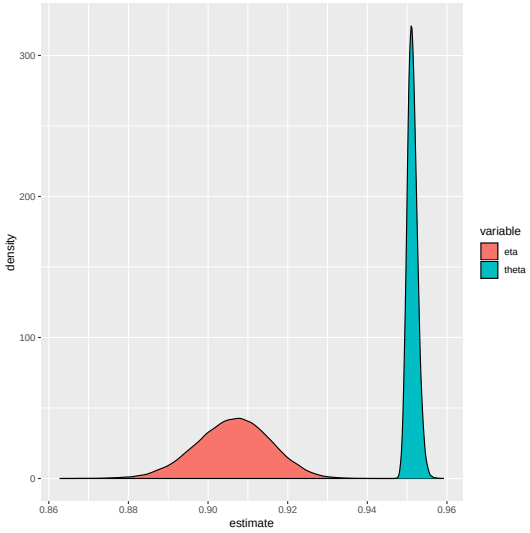

Figure 5: Overlaid Densities of η and θ

5. Diagnostics

5.1 Trace Plots

In order to assess if the Gibbs Sampler converged to the posterior distributions, we used the R packages *ggplot2* and *tidybayes*. Using the trace plots, we can determine if there are any abnormalities in the MCMC sampling algorithm.

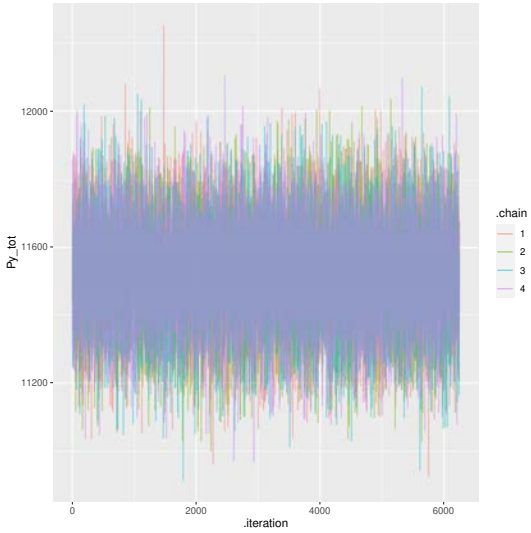

Figure 6: Trace Plot for $Pytot$

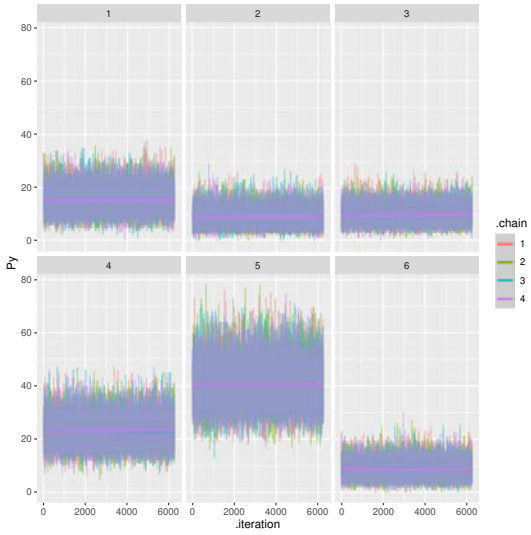

Figure 7: Trace Plot for Py_i

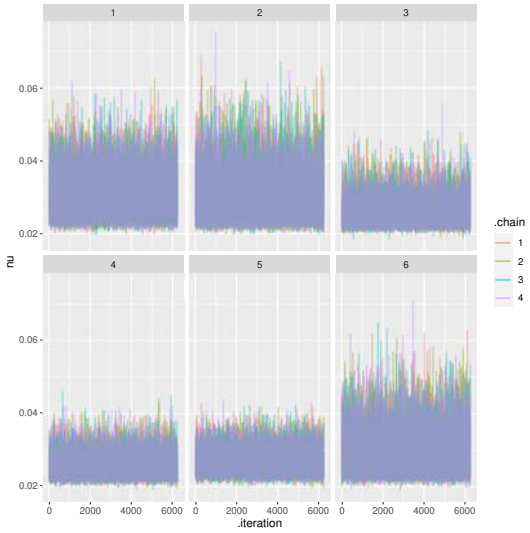

Figure 8: Trace Plot for ν_i

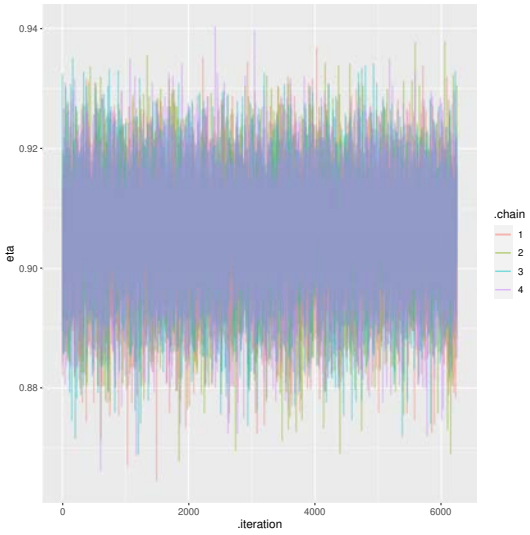

Figure 9: Trace Plot for η

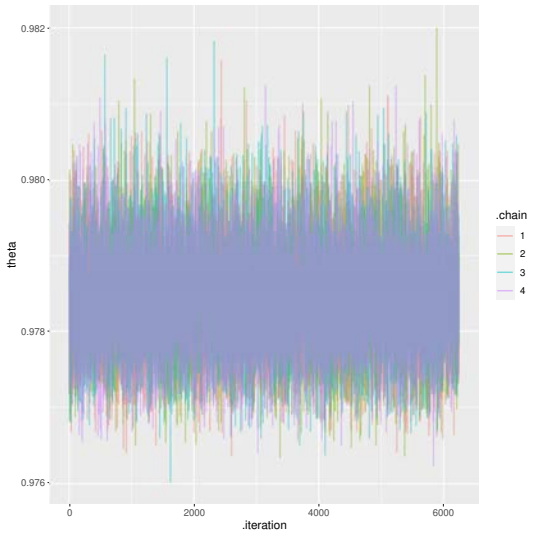

Figure 10: Trace Plot for θ

As evident from the trace plots, it appears that all the MCMC chains have successfully converged, for each of the parameters. While these trace plots can be helpful to assess for the MCMC chains' convergence, it is oftentimes more useful to confirm with a statistic and hypothesis test.

5.2 Gelman-Rubin Statistic

The Gelman-Rubin statistic is often used to assess the convergence of MCMC chains. It can be thought of as a sort of ANOVA F-Ratio, to compare the variance between each chain to within each chain. Similar to an ANOVA, if the Gelman-Rubin statistic is near 1, it suggests that there is no significant difference between the chains – implying that the chains have successfully converged to the posterior distributions.

Posterior	Estimate	Upper C.I.
Py_{tot}	1.0000076	1.0001058
ω	0.9999569	1.0000408
κ	1.0000918	1.0003094
η	0.9999740	1.0000254
θ	1.0001809	1.0004417

Table 3: Gelman-Rubin Statistics for Non-Event Specific Parameters

Posterior	Estimate	Upper C.I.
Py_1	1.0000912	1.0003353
ν_1	1.0002477	1.0009390
π_1	1.0006337	1.0013397
.	.	.
.	.	.
Py_{73}	1.0001174	1.0004958
ν_{73}	1.0000770	1.0002103
π_{73}	1.0007099	1.0008744

Table 4: Gelman-Rubin Statistic for Event-Specific Parameters

In Tables 3 and 4, the Gelman-Rubin statistics for all of the posterior distributions are around 1. Additionally, none of the upper confidence limits for the Gelman-Rubin statistics are significantly above 1. This shows that the variance between each chain, is (almost) the same as the variance within each change. Therefore, we can conclude that the random sampling algorithms successfully converged to the posterior distributions.

6. Conclusion

Through this model, we were able to simulate and model the diagnostic testing data with an understanding of the limitation of the imperfect sensitivity and specificity of tghe test used to screen the attendants of the Rose Garden event which resulted in many coronavirus cases. Additionally, we were able to create predictive posterior distributions for the number of people who would have tested positive, at each of former-President Trump's events held throughout the height of the pandemic. Using diagnostics such as the Gelman-Rubin statistic, we were able to conclude that our MCMC chains had successfully converged to the posterior distributions of interest.

REFERENCES

Brooks, S. P., Gelman, A. (1998). General Methods for Monitoring Convergence of Iterative Simulations. Journal of Computational and Graphical Statistics. Volume 7(4), Pages 434-455.

Kruschke, J. (2015). Doing Bayesian Data Analysis: A Tutorial with R, Jags, and Stan (2nd ed.). Academic Press/Elsevier.

Mandavilli, A. (2020, October 02). The White House relied on a rapid test, but used it in a way it was not intended. Retrieved from URL: https://www.nytimes.com/2020/10/02/us/elections/the-white-house-relied-on-a-rapid-test-but-used-it-in-a-way-it-was-not-intended.html

Plummer, Martyn (2019). rjags: Bayesian Graphical Models using MCMC. R package version 4-10. URL: https://CRAN.R-project.org/package=rjags

Plummer, Martyn. (2003). JAGS: A Program for Analysis of Bayesian Graphical Models using Gibbs Sampling. 3rd International Workshop on Distributed Statistical Computing (DSC 2003); Vienna, Austria. 124.

R Core Team (2021). R: A language and environment for statistical computing. R Foundation for Statistical Computing, Vienna, Austria. URL: www.R-project.org/

Suess, Eric A., Gardner, I.A., Johnson, W.O. (2002). Hierarchical Bayesian model for prevalence inferences and determination of a country's status for an animal pathogen. Preventive Veterinary Medicine, Volume 55(3), Pages 155-171