Final Report (STATS 551)
Snigdha Pakala (vpakala)
December 18, 2024

1. Introduction

Vaccinations have been an integral part of medicine for a long time, dating back to the 15th century when humans attempted to prevent disease through variolation - a process where healthy people were intentionally exposed to smallpox to protect themselves against it. Vaccines have played a critical role in the eradication and reduction of many diseases such as small pox, measles, and polio. However, despite the scientific evidence for their positive impact, much controversy surrounds vaccination. Vaccine hesitancy due to misinformation and distrust in science, have prevented high population vaccination rates.

Vaccination campaigns remain one of the most effective tools for preventing disease outbreaks, but their success depends on addressing these challenges. Understanding how effective vaccination campaigns are is necessary to optimize public health strategies. Recently, the topic of vaccines has gained increased attention and sparked widespread public debate due to the COVID-19 pandemic. This heightened awareness makes a study of vaccination campaign effectiveness particularly timely and relevant.

This study investigates the impact of a pneumococcal vaccination campaign on increasing immunization rates across U.S. communities from 2018 to 2021. We address community-level variability, using a Bayesian hierarchical model to explore how outcomes differ by geographic and demographic factors. Prior research highlights both the potential effectiveness and challenges of vaccination campaigns, but this analysis provides insights through the lens of Bayesian modeling.

2. Data

The dataset used in this analysis comes from the Behavioral Risk Factor Surveillance System (BRFSS), a comprehensive annual survey conducted by the Centers for Disease Control and Prevention (CDC). This data provide vaccination coverage estimates for adults aged 18 years and older who participated in BRFSS interviews. The survey is a critical component of the CDC's AdultVaxView platform, designed to monitor and evaluate adult immunization trends in the United States. The dataset includes vaccination coverage estimates stratified by various demographic and contextual factors, such as geographical regions, age groups, race, and survey year.

It is important to note that data availability is incomplete for all geographic regions. A subset of the data lacks coverage estimates in certain areas or for specific populations due to sample size limitations or other constraints. This variability, along with the presence of many hierarchical models in studying this research question, underscore the importance of robust statistical techniques, such as Bayesian methods, to handle the uncertainty in the dataset.

3. Method

The hierarchical model used in this study is the following:

For the likelihood, it makes sense that the vaccination counts are modeled using a Binomial distribution. Thus, for each observation y_i,

$$y_i \sim \text{Binomial}(\text{total}_i, p_i), \quad \text{logit}(p_i) = \alpha[\text{region}_i] + \beta \cdot \text{time\_period}_i + \gamma \cdot (\text{age\_group}_i - 1).$$

where p_i is the probability of vaccination for observation i.

Regions were created by grouping areas with similar vaccination rates. The four regions in the data are: New England, the most vaccinated region; Middle Atlantic, with distinct vaccination patterns compared to New England but higher than the national average; East North Central, with moderate vaccination rates and similar regional patterns; and other regions with similar rates, which have lower statistical power individually. To incorporate regional effects, each region has its own vaccination rates while sharing information through the global mean and standard deviation. So for each region r,

$$\alpha[r] \sim \text{Normal}(\mu_\alpha, \sigma_\alpha).$$

The informative priors were chosen based on CDC vaccination data to estimate parameters:

$$\mu_\alpha \sim \text{Normal}(\text{logit}(0.65), 0.5), \quad \sigma_\alpha \sim \text{Normal}(0, 0.5),$$

$$\beta \sim \text{Normal}(0, 0.5), \quad \gamma \sim \text{Normal}(0, 0.5).$$

The justification of these is as follows: Since μ_α represents the average vaccination rate across regions, it is centered at 65% with moderate uncertainty to align with the CDC-reported average rate. The logit transofrmation ensures the probabilities remain between $[0, 1]$. This combined with σ_α at Normal(0, 0.5) - which allows for moderate regional variation but penalizes excessively large variability - makes this a weekly informative prior which reflects a reasonable range for variability. These two combined create a standard normal prior for raw region-specific effects, scaled by σ_α, for $\alpha_{raw}[r]$. Next, β captures the time period effect in relation to COVID-19 (pre/post 2020), so it is normally distributed and centered at 0 for no-effect but allowing moderate deviations. Lastly, γ captures the age-group effect, and represented the same as β.

The prior predictive check shows that the prior predictions capture the general shape with some uncertainty, so the prior is reasonable for our data.

Posterior: Using Bayes' theorem, the posterior distribution is:

$$P(\mu_\alpha, \sigma_\alpha, \boldsymbol{\alpha}, \beta, \gamma \mid \mathbf{y}) \propto P(\mathbf{y} \mid \mathbf{p}) P(\mathbf{p} \mid \boldsymbol{\alpha}, \beta, \gamma) P(\boldsymbol{\alpha} \mid \mu_\alpha, \sigma_\alpha) P(\mu_\alpha) P(\sigma_\alpha) P(\beta) P(\gamma)$$

where the posterior distribution reveals the campaign effectiveness on the log-odds scale.

This hierarchical structure is the basis for the Stan model. I set adapt delta at 0.99 to reduce divergent transitions and max tree-depth at 12 to prevent the sampler from getting stuck in very long trajectories and balance computation time with exploration. The model draws 3000 iterations for each of the 4 chains to provide sufficient samples after discarding the first 1500 to allow chains to reach stable stationary distributions. The trace plots, autocorrelation and posterior predictive checks below show stable mixing and good convergence, and the parameters appear well defined. The parameters (β and γ) show great stability with quick decay in autocorrelation. All of the Rhat values being very close to 1 indicate good convergence. Lastly, the effective sample sizes are around 4000 for (β and γ) and between 1378 to 2391 for the α_{raw} parameters, indicating good sampling efficiency for parameters overall.

4. Results

Our results provide many useful implications and understandings of the nuances behind vaccination campaign effectiveness. Firstly, $\beta \approx -0.06$ with 95% credible interval $[-0.08, -0.04]$ is capturing the impact of the COVID-19 time period on vaccination rates. -0.06 suggests that on average, the odds of vaccination during and after the pandemic were approximately 6% lower than pre-COVID times. Since the interval does not include 0, these results are statistically significant. This 6% reduction is a substantial finding when considering a large population. This result highlights the importance of understanding how global events, such as a pandemic, affect public health behaviors. There was also about a 2 percentage point decrease between mean vaccination probability pre-COVID vs. post-COVID, aligning with disruptions caused by the pandemic. Acknowledging this decline allows public health officials to design more targeted interventions aimed at boosting vaccination rates before disastrous crises occur.

Next, regional variations provide much insight on strategization of effective vaccine campaigns. $\sigma_\alpha \approx 0.15$ with 95% credible interval $[0.04, 0.48]$ implies that there are statistically significant regional differences in vaccination rates. This is crucial as it suggests that localized public health interventions such as targeted outreach in under-performing regions might be an effective way to improve vaccination uptake.

Additionally, $\gamma \approx -0.001$ with 95% credible interval $[-0.99, 0.98]$ implies that there were no meaningful differences between the two age groups (18-64 years vs. >= 65 years). However, the uncertainty in this parameter could be consistent with the idea that the effect might vary regionally. This result is also implies vaccine campaigns be created without significant age-based targeting, and instead focus on strategies for particular regions.

Overall, the evidence of regional variation means campaigns can be more specifically designed for under-performing geographical areas. The decline in vaccinations during the pandemic can guide future changes of more frequent targeted campaigns in non-pandemic periods, ensuring that vaccination efforts are not hindered by future crises.

5. Limitations and Next Steps

Despite the model's overall utility, there were also limitations that could have been improved. First, as Figures 9-12 show, the model struggled with extreme values, suggesting that more flexible likelihoods (such as negative binomial) may capture outliers better in regions with high vaccination success. Second, the pre- and post-COVID categories may oversimplify temporal dynamics, potentially missing the nuanced trends visualized in Figure 1. Lastly, Figure 8 reveals correlations between the global mean and the time period effect parameters, indicating further model refinement for increased accuracy.

Along with these changes, future efforts could include incorporating additional predictors such as healthcare infrastructure, demographic details, and vaccine hesitancy, to capture the complex factors influencing vaccine uptake. Also, extending the data after 2021 to assess the robustness of the findings and provide more generalizable insights.

The evidence suggests the pneumococcal vaccination campaign had mixed effectiveness, with an overall decrease in vaccination rates during and after the COVID-19 pandemic, but significant regional variation in outcomes. The model highlights the importance of targeted interventions in under-performing regions and emphasizes the need for future campaigns to address disruptions caused by crises. By refining public health strategies, this work provides valuable insights for improving vaccination outreach and ensuring sustained coverage in the future.

6. Supplemental Details and Materials

0.0.1 Figures

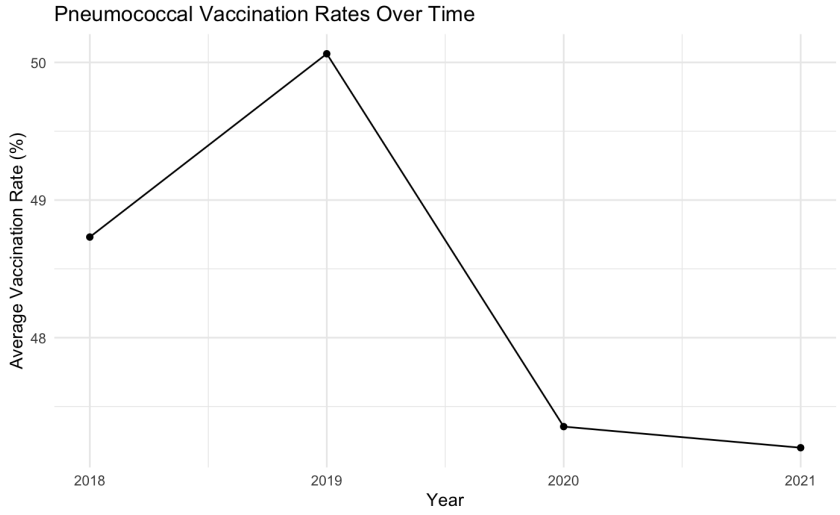


Figure 1: Trend of Pneumococcal Vaccination 2018-2021

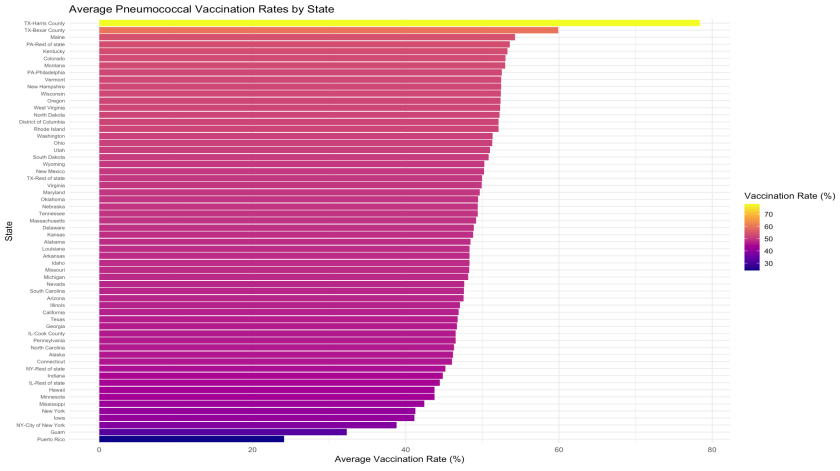


Figure 2: Pneumococcal Vaccination Rates by State

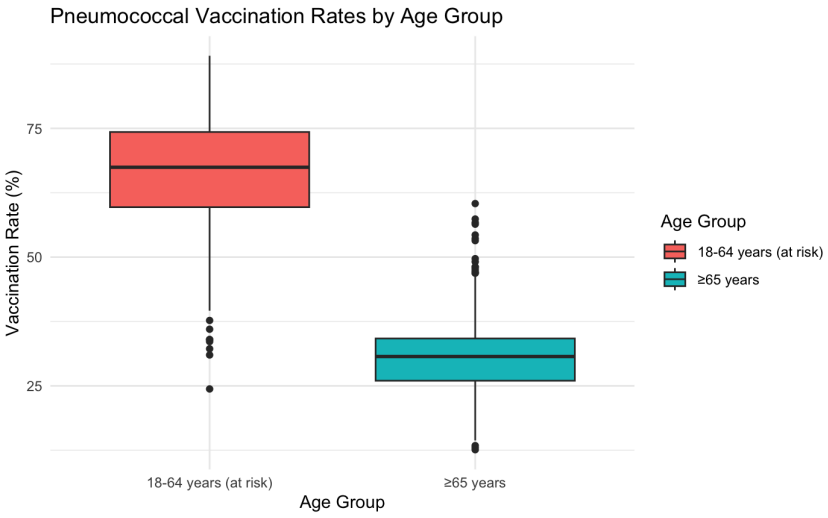


Figure 3: Age Group Variation in Vaccination Rates

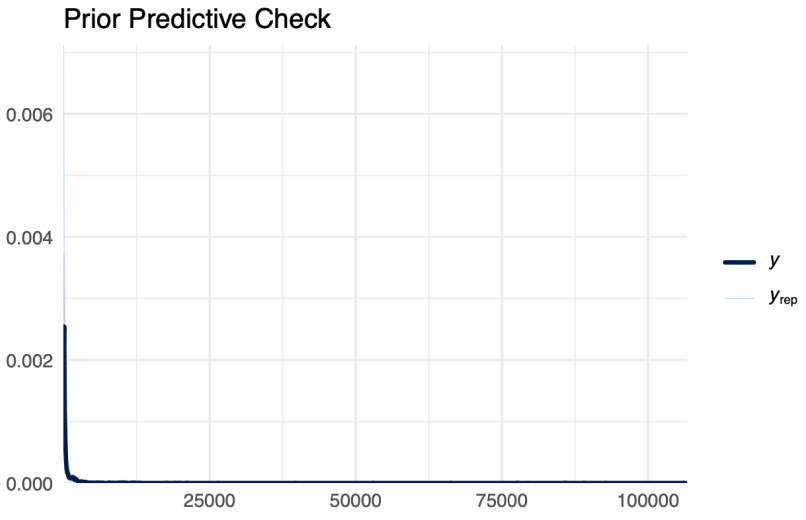


Figure 4: Prior Predictive Checking

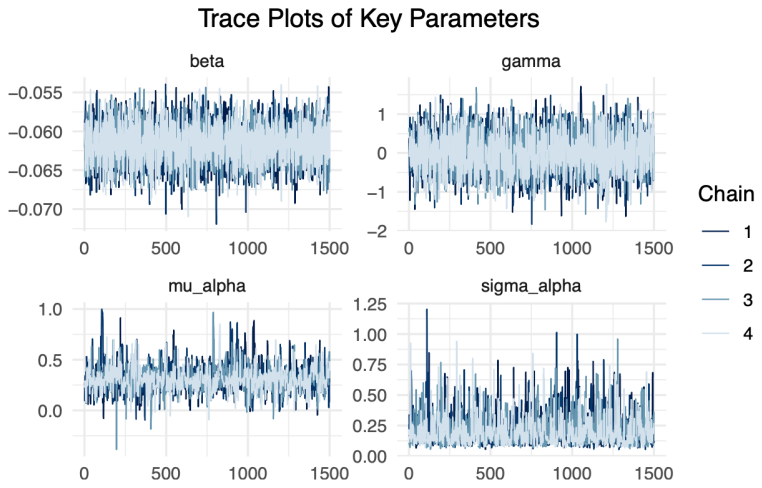


Figure 5: Trace Plots for Parameters

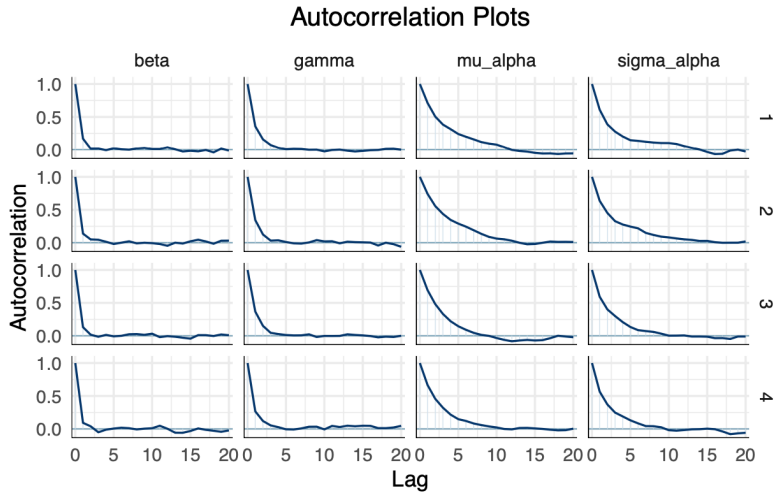


Figure 6: Autocorrelation Plots for Parameters

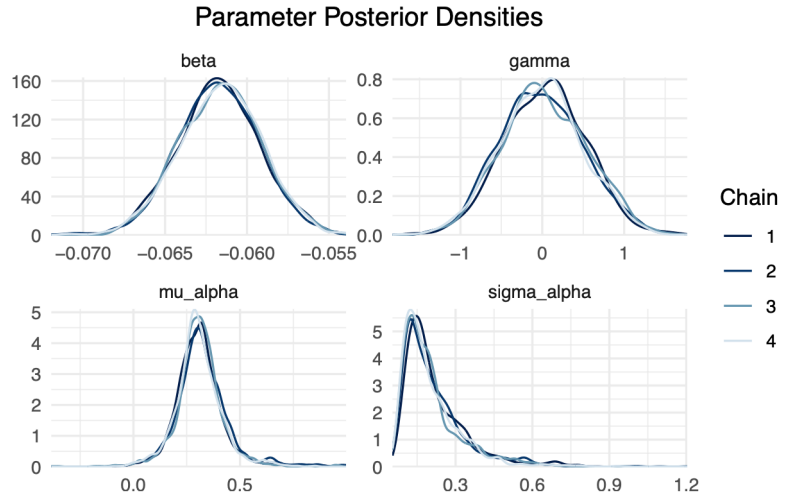


Figure 7: Parameter Posterior Densities

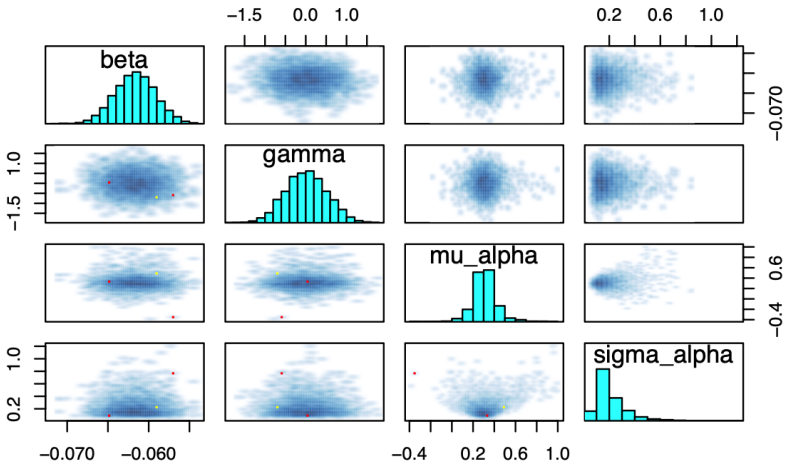


Figure 8: Parameter Pairs Plots

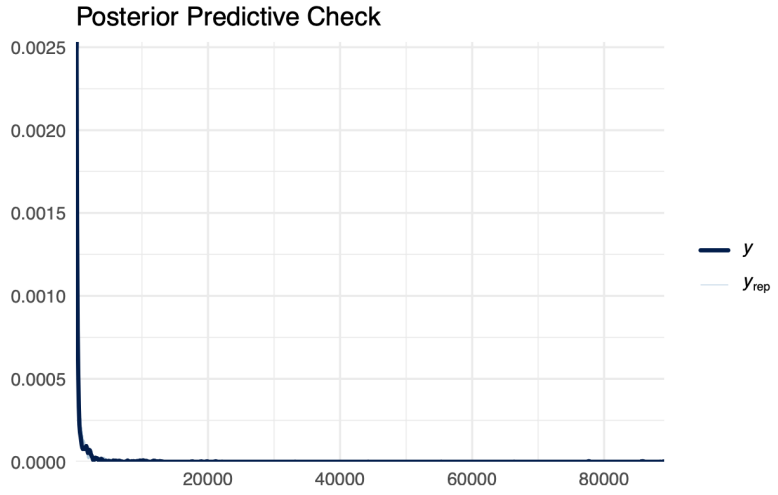


Figure 9: Posterior Predictive Check

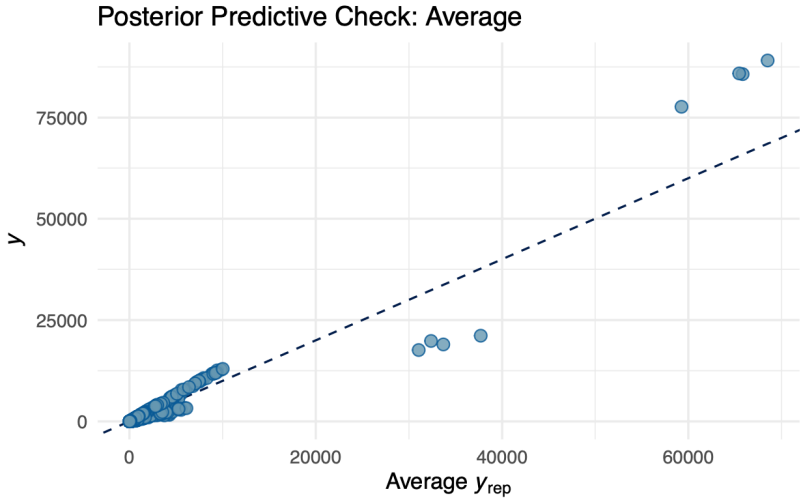


Figure 10: Posterior Predictive Check (Average)

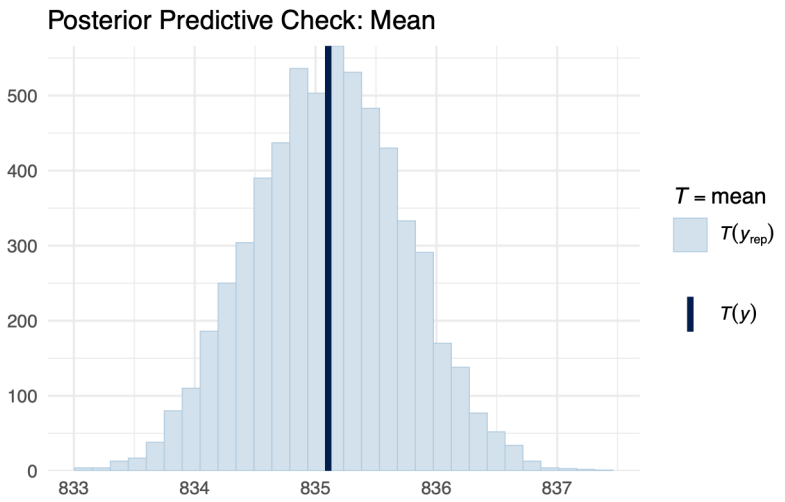


Figure 11: Posterior Predictive Check (Tstat vs. Predictive)

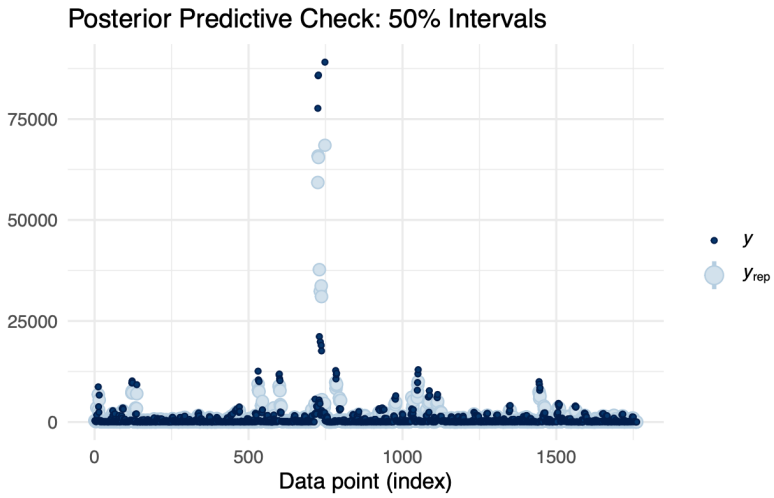


Figure 12: Posterior Predictive Check (50% Intervals)

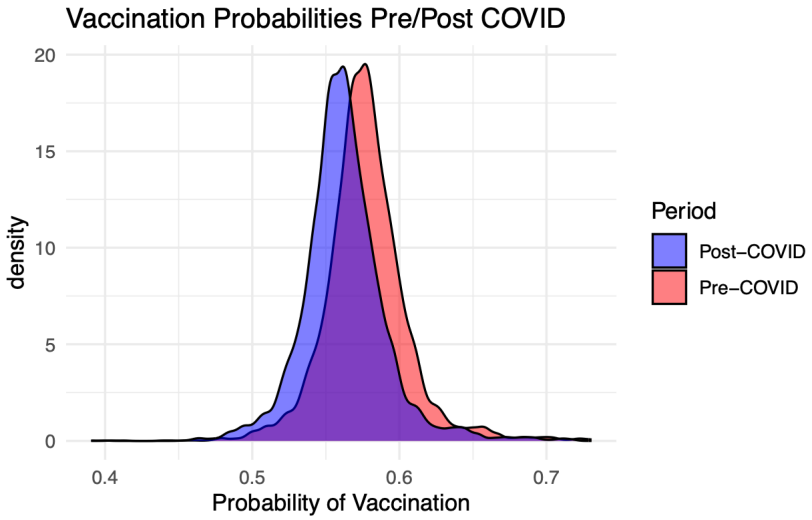


Figure 13: Campaign Effectiveness (Pre/Post COVID-19)

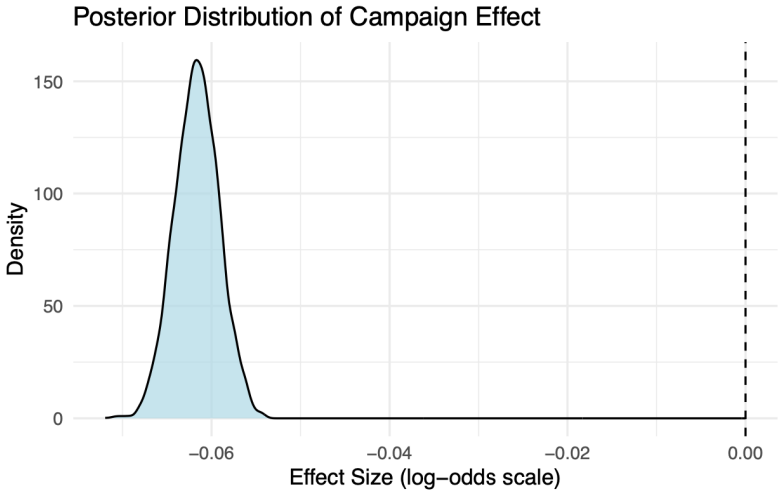


Figure 14: Campaign Effectiveness (Posterior)

```
Beta (Time Effect): Mean = -0.05954291 , 95% CI = [ -0.08326466 ,  -0.03697211 ]
Gamma (Age Group Effect): Mean = -0.00106733 , 95% CI = [ -0.9975432 ,  0.9768381 ]
Mu_alpha (Global Mean Vaccination Rate): Mean = -0.6157127 , 95% CI = [ -0.7730004 ,  -0.3224488 ]
Sigma_alpha (Regional Variation): Mean = 0.1476064 , 95% CI = [ 0.04170844 ,  0.4801429 ]
```

Figure 15: 95% Credible Intervals for Parameters

0.0.2 References

1. World Health Organization. "A Brief History of Vaccination." WHO, https://www.who.int/news-room/spotlight/history-of-vaccination/a-brief-history-of-vaccination.

2. Poland, Gregory A., et al. "The Age-Old Struggle against the Antivaccinationists." New England Journal of Medicine, vol. 364, no. 2, 2011, pp. 97–99. PubMed, https://pubmed.ncbi.nlm.nih.gov/23444591/

3. Centers for Disease Control and Prevention. "AdultVaxView: General Population." CDC, https://www.cdc.gov/adultvaxview/about/general-population.html.

4. R Core Team. "Tutorial 100: Posterior Predictive Checks." isotracer: R Package Documentation, CRAN, https://cran.r-project.org/web/packages/isotracer/vignettes/tutorial-100-posterior-predictive-checks.html

5. R Core Team. "Logistic." R Documentation, https://www.rdocumentation.org/packages/stats/versions/3.6.2/topics/Logistic.

6. UCLA Statistical Consulting Group. "Logistic Regression in R." UCLA Institute for Digital Research and Education, https://stats.oarc.ucla.edu/r/dae/logit-regression/.

7. OpenAI. "ChatGPT." ChatGPT by OpenAI, OpenAI, chat.openai.com (general sentence refinement conciseness to fit report in the 2-3 page limit)

0.0.3 Code

551 Final Project

Snigdha Pakala

Data Preparation and Cleaning

```
# Load required libraries
library(dplyr)
```

```
Attaching package: 'dplyr'

The following objects are masked from 'package:stats':

    filter, lag

The following objects are masked from 'package:base':

    intersect, setdiff, setequal, union
```

```
library(ggplot2)
library(rstan)
```

```
Loading required package: StanHeaders


rstan version 2.32.6 (Stan version 2.32.2)

For execution on a local, multicore CPU with excess RAM we recommend calling
options(mc.cores = parallel::detectCores()).
To avoid recompilation of unchanged Stan programs, we recommend calling
rstan_options(auto_write = TRUE)
For within-chain threading using `reduce_sum()` or `map_rect()` Stan functions,
change `threads_per_chain` option:
rstan_options(threads_per_chain = 1)
```

```r
library(bayesplot)
```

```
This is bayesplot version 1.11.1

- Online documentation and vignettes at mc-stan.org/bayesplot

- bayesplot theme set to bayesplot::theme_default()

   * Does _not_ affect other ggplot2 plots

   * See ?bayesplot_theme_set for details on theme setting
```

```r
# Load CDC Dataset
data <- read.csv("~/Downloads/Vaccination_Coverage_among_Adults__18__Years__20241210.csv")
names(data)
```

```
 [1] "Vaccine"        "Geography.Type" "Geography"      "FIPS"
 [5] "Survey.Year"    "Dimension.Type" "Dimension"      "Estimate...."
 [9] "X95..CI...."    "Sample.Size"
```

```r
# Initial data exploration for cleaning
data %>% distinct(Dimension)
```

```
                            Dimension
1                             Overall
2                            Hispanic
3 Other or Multiple Races, Non-Hispanic
4                   Black, Non-Hispanic
5                   White, Non-Hispanic
```

```r
# To reduce the scope of this research question, made the following data filters:
#   - Focuses only on Pneumococcal vaccination (2018-2021) to use largest data available
#   - Accounts for two distinct age groups with different recommendations:
#       1. Adults 18-64 years at increased risk
#       2. All adults  65 years
#

vax_data <- data %>%
```

```
  filter(
    Vaccine == "Pneumococcal",
    Survey.Year >= 2018,
    Survey.Year <= 2021,
    !Geography %in% c("New Jersey", "Florida"), # No data available for certain years
    !Dimension %in% c("Overall") # Avoid repetition
  ) %>%
  mutate(
    # Convert percentage and sample size
    Estimate.... = as.numeric(Estimate....),
    Sample.Size = as.numeric(Sample.Size),
    total = Sample.Size,
    vaccinated = round(Estimate.... * Sample.Size / 100),
    time_period = case_when(
      Survey.Year < 2020 ~ 0,
      Survey.Year == 2020 ~ 1,
      Survey.Year > 2020 ~ 2
    ),
    community = as.numeric(factor(Geography, levels = unique(Geography))),
    age_group = ifelse(Dimension.Type == " 65 Years", 2, 1),
    ci_bounds = strsplit(as.character(X95..CI....), " to "),
    ci_lower = as.numeric(sapply(ci_bounds, `[`, 1)),
    ci_upper = as.numeric(sapply(ci_bounds, `[`, 2)),
    ci_width = ci_upper - ci_lower) %>%
  filter(!is.na(Estimate....),
         !is.na(Sample.Size),
         Estimate.... != "NR",
         X95..CI.... != "NR")
```

```
Warning: There was 1 warning in `mutate()`.
i In argument: `Estimate.... = as.numeric(Estimate....)`.
Caused by warning:
! NAs introduced by coercion
```

```
vax_data <- vax_data %>%
  arrange(Geography) %>%  # Sort by Geography alphabetically
  mutate(
    community = dense_rank(Geography) # Use dense_rank to ensure consecutive integers
  )

vax_data %>% distinct(Geography)
```

	Geography
1	Alabama
2	Alaska
3	Arizona
4	Arkansas
5	California
6	Colorado
7	Connecticut
8	Delaware
9	District of Columbia
10	Georgia
11	Guam
12	Hawaii
13	IL-Cook County
14	IL-Rest of state
15	Idaho
16	Illinois
17	Indiana
18	Iowa
19	Kansas
20	Kentucky
21	Louisiana
22	Maine
23	Maryland
24	Massachusetts
25	Michigan
26	Minnesota
27	Mississippi
28	Missouri
29	Montana
30	NY-City of New York
31	NY-Rest of state
32	Nebraska
33	Nevada
34	New Hampshire
35	New Mexico
36	New York
37	North Carolina
38	North Dakota
39	Ohio
40	Oklahoma
41	Oregon
42	PA-Philadelphia

```
43     PA-Rest of state
44         Pennsylvania
45          Puerto Rico
46             Region 1
47            Region 10
48             Region 2
49             Region 3
50             Region 4
51             Region 5
52             Region 6
53             Region 7
54             Region 8
55             Region 9
56         Rhode Island
57       South Carolina
58         South Dakota
59      TX-Bexar County
60     TX-Harris County
61     TX-Rest of state
62            Tennessee
63                Texas
64        United States
65                 Utah
66              Vermont
67             Virginia
68           Washington
69        West Virginia
70            Wisconsin
71              Wyoming
```

```
vax_data %>% distinct(community)
```

```
   community
1          1
2          2
3          3
4          4
5          5
6          6
7          7
8          8
9          9
```

10	10
11	11
12	12
13	13
14	14
15	15
16	16
17	17
18	18
19	19
20	20
21	21
22	22
23	23
24	24
25	25
26	26
27	27
28	28
29	29
30	30
31	31
32	32
33	33
34	34
35	35
36	36
37	37
38	38
39	39
40	40
41	41
42	42
43	43
44	44
45	45
46	46
47	47
48	48
49	49
50	50
51	51
52	52

```
53        53
54        54
55        55
56        56
57        57
58        58
59        59
60        60
61        61
62        62
63        63
64        64
65        65
66        66
67        67
68        68
69        69
70        70
71        71
```

```r
# Verify data structure
print(paste("Number of observations:", nrow(vax_data)))
```

```
[1] "Number of observations: 1761"
```

```r
print("Distribution by age group:")
```

```
[1] "Distribution by age group:"
```

```r
print(table(vax_data$Dimension.Type))
```

```
                  >=65 Years 18-64 Years at Increased Risk
                         876                           885
```

```r
print("Distribution by year:")
```

```
[1] "Distribution by year:"
```

```r
print(table(vax_data$Survey.Year))
```

```
2018 2019 2020 2021
 458  457  440  406
```

```r
# Ensure there are no more NAs in data
colSums(is.na(vax_data))
```

```
       Vaccine Geography.Type      Geography           FIPS    Survey.Year
             0              0              0              0              0
Dimension.Type      Dimension   Estimate....    X95..CI....    Sample.Size
             0              0              0              0              0
         total     vaccinated    time_period      community      age_group
             0              0              0              0              0
     ci_bounds       ci_lower       ci_upper       ci_width
             0              0              0              0
```

```r
# Verify the mapping
print(length(unique(vax_data$community)))  # Should be 71
```

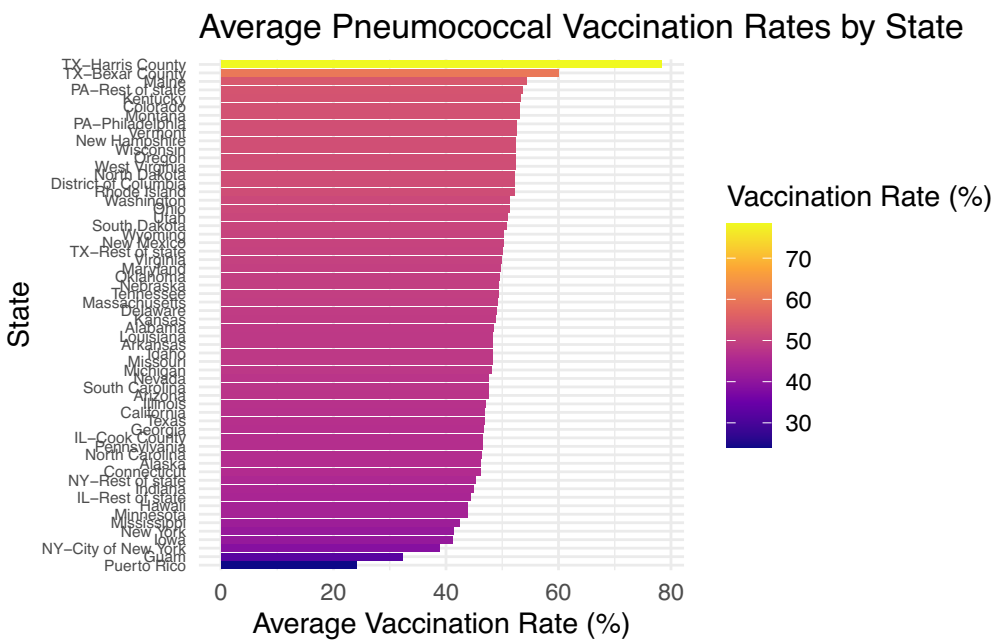
```
[1] 71
```

```r
print(max(vax_data$community))  # Should also be 71
```

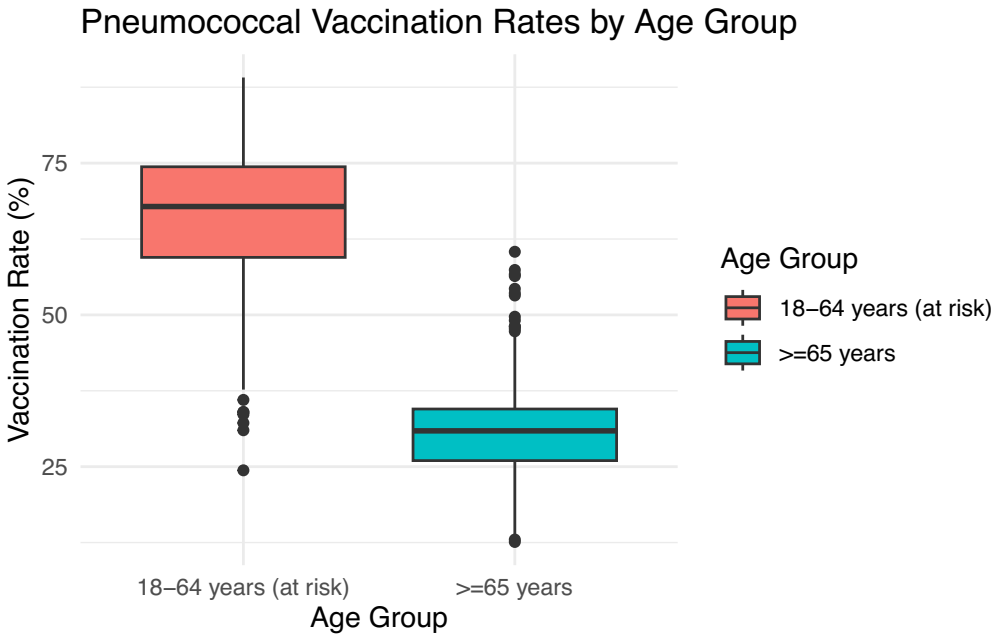
```
[1] 71
```

```r
# Initial understanding of trends

state_data <- vax_data %>%
  filter(Geography.Type == "States/Local Areas")

# Calculate average vaccination rate for each state
state_avg <- state_data %>%
  group_by(Geography) %>%
  summarize(avg_rate = mean(Estimate...., na.rm = TRUE)) %>%
  ungroup()

# Vaccination rates by state
```

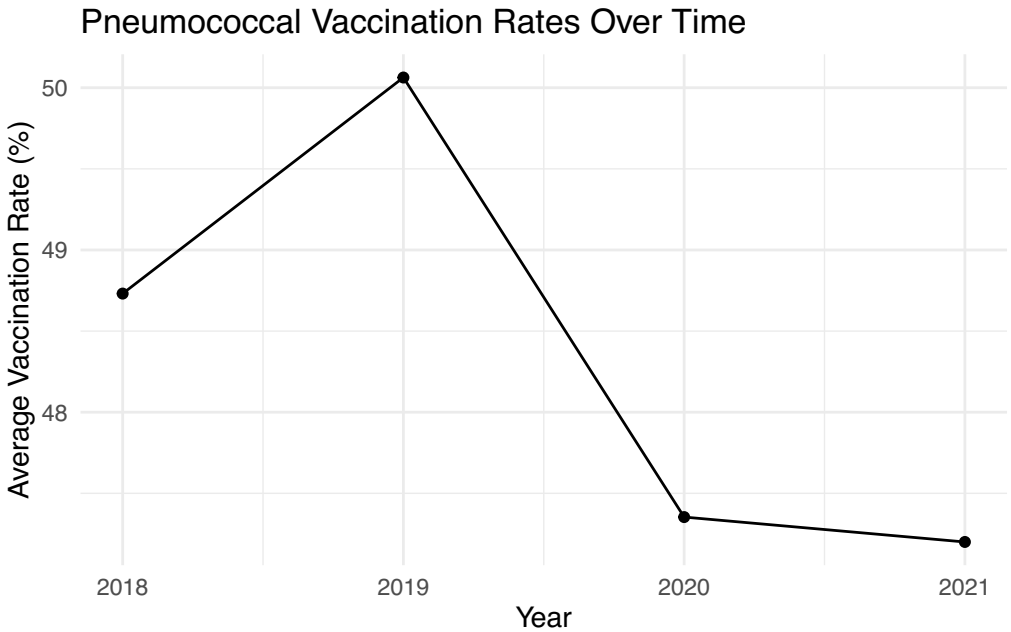
```
ggplot(state_avg, aes(x = reorder(Geography, avg_rate), y = avg_rate, fill = avg_rate)) +
  geom_bar(stat = "identity") +
  coord_flip() +
  scale_fill_viridis_c(option = "plasma") +
  labs(title = "Average Pneumococcal Vaccination Rates by State",
       x = "State",
       y = "Average Vaccination Rate (%)",
       fill = "Vaccination Rate (%)") +
  theme_minimal() +
  theme(axis.text.y = element_text(size = 6))
```




```
# Vaccination rates by age group
ggplot(state_data, aes(x = as.factor(Dimension.Type),
                       y = Estimate....,
                       fill = as.factor(Dimension.Type))) +
  geom_boxplot() +
  labs(title = "Pneumococcal Vaccination Rates by Age Group",
       x = "Age Group",
       y = "Vaccination Rate (%)",
       fill = "Age Group") +
  scale_x_discrete(labels = c("18-64 years (at risk)", " 65 years")) +
  scale_fill_discrete(labels = c("18-64 years (at risk)", " 65 years")) +
  theme_minimal()
```




```
# Vaccination rates over time
state_data %>%
  group_by(Survey.Year) %>%
  summarize(avg_rate = mean(Estimate...., na.rm = TRUE)) %>%
  ggplot(aes(x = Survey.Year, y = avg_rate)) +
  geom_line() +
  geom_point() +
  labs(title = "Pneumococcal Vaccination Rates Over Time",
       x = "Year",
       y = "Average Vaccination Rate (%)") +
  theme_minimal()
```



Model Creation

```
# Simplify data by focusing on key variables and reducing complexity
vax_data_simple <- data %>%
  filter(
    Vaccine == "Pneumococcal",
    Survey.Year >= 2018,
    Survey.Year <= 2021,
    !Geography %in% c("New Jersey", "Florida"),
    # Remove Overall category
    !Dimension %in% c("Overall")
  ) %>%
  mutate(
    vaccinated = round(as.numeric(Estimate....) * as.numeric(Sample.Size) / 100),
    total = as.numeric(Sample.Size),
    # Simplify time periods to before/after COVID
    time_period = ifelse(Survey.Year < 2020, 0, 1),
    # Keep age grouping
    age_group = ifelse(Dimension.Type == " 65 Years", 2, 1),
    # Simplify geography to regions instead of states
    region = case_when(
      Geography %in% c("Maine", "Vermont", "New Hampshire", "Massachusetts",
```

```
                      "Rhode Island", "Connecticut") ~ 1,
      Geography %in% c("New York", "Pennsylvania") ~ 2,
      Geography %in% c("Michigan", "Wisconsin", "Minnesota", "Illinois",
                      "Indiana", "Ohio") ~ 3,
      TRUE ~ 4
    )
  ) %>%
  # Remove missing values
  filter(!is.na(vaccinated),
         !is.na(total),
         total > 0)
```

```
Warning: There was 1 warning in `mutate()`.
i In argument: `vaccinated = round(as.numeric(Estimate....) *
  as.numeric(Sample.Size)/100)`.
Caused by warning:
! NAs introduced by coercion
```

```
# Prepare simplified Stan data
stan_data_simple <- list(
  N = nrow(vax_data_simple),
  R = length(unique(vax_data_simple$region)),
  y = vax_data_simple$vaccinated,
  total = vax_data_simple$total,
  region = vax_data_simple$region,
  time_period = vax_data_simple$time_period,
  age_group = vax_data_simple$age_group,
  weights = vax_data_simple$total / sum(vax_data_simple$total)
)

# Simplified Stan model
stan_code_simple <-
"data {
  int<lower=1> N;
  int<lower=1> R;
  int<lower=0> y[N];
  int<lower=1> total[N];
  int<lower=1,upper=R> region[N];
  int<lower=0,upper=1> time_period[N];
  int<lower=1,upper=2> age_group[N];
  vector[N] weights;
}
```

```
parameters {
  vector[R] alpha_raw;
  real beta;
  real gamma;
  real mu_alpha;
  real<lower=0> sigma_alpha;
}

transformed parameters {
  vector[R] alpha = mu_alpha + sigma_alpha * alpha_raw;
}

model {
  // Informative priors based on CDC data
  mu_alpha ~ normal(logit(0.65), 0.5);
  sigma_alpha ~ normal(0, 0.5);
  beta ~ normal(0, 0.5);
  gamma ~ normal(0, 0.5);
  alpha_raw ~ std_normal();

  // Likelihood
  {
    vector[N] logit_p;
    for (i in 1:N) {
      logit_p[i] = alpha[region[i]] +
                   beta * time_period[i] +
                   gamma * (age_group[i] - 1);
    }
    target += weights .* binomial_logit_lpmf(y | total, logit_p);
  }
}

generated quantities {
  vector[N] log_lik;
  vector[N] y_rep;
  {
    vector[N] logit_p;
    for (i in 1:N) {
      logit_p[i] = alpha[region[i]] +
                   beta * time_period[i] +
                   gamma * (age_group[i] - 1);
      log_lik[i] = binomial_logit_lpmf(y[i] | total[i], inv_logit(logit_p[i]));
```

```
      y_rep[i] = binomial_rng(total[i], inv_logit(logit_p[i]));
    }
  }
}
"

fit <- stan(
  model_code = stan_code_simple,
  data = stan_data_simple,
  chains = 4,
  iter = 3000,
  warmup = 1500,
  #thin = 2,
  control = list(
    adapt_delta = 0.99,
    max_treedepth = 12
  )
)
```

```
Trying to compile a simple C file

Running /Library/Frameworks/R.framework/Resources/bin/R CMD SHLIB foo.c
using C compiler: ‘Apple clang version 13.0.0 (clang-1300.0.27.3)’
using SDK: ‘’
clang -arch x86_64 -I"/Library/Frameworks/R.framework/Resources/include" -DNDEBUG   -I"/Libra
In file included from <built-in>:1:
In file included from /Library/Frameworks/R.framework/Versions/4.4-x86_64/Resources/library/S
In file included from /Library/Frameworks/R.framework/Versions/4.4-x86_64/Resources/library/R
In file included from /Library/Frameworks/R.framework/Versions/4.4-x86_64/Resources/library/R
/Library/Frameworks/R.framework/Versions/4.4-x86_64/Resources/library/RcppEigen/include/Eigen
#include <cmath>
         ^~~~~~~
1 error generated.
make: *** [foo.o] Error 1

SAMPLING FOR MODEL 'anon_model' NOW (CHAIN 1).
Chain 1:
Chain 1: Gradient evaluation took 0.002789 seconds
Chain 1: 1000 transitions using 10 leapfrog steps per transition would take 27.89 seconds.
Chain 1: Adjust your expectations accordingly!
Chain 1:
```

```
Chain 1:
Chain 1: Iteration:    1 / 3000 [  0%]  (Warmup)
Chain 1: Iteration:  300 / 3000 [ 10%]  (Warmup)
Chain 1: Iteration:  600 / 3000 [ 20%]  (Warmup)
Chain 1: Iteration:  900 / 3000 [ 30%]  (Warmup)
Chain 1: Iteration: 1200 / 3000 [ 40%]  (Warmup)
Chain 1: Iteration: 1500 / 3000 [ 50%]  (Warmup)
Chain 1: Iteration: 1501 / 3000 [ 50%]  (Sampling)
Chain 1: Iteration: 1800 / 3000 [ 60%]  (Sampling)
Chain 1: Iteration: 2100 / 3000 [ 70%]  (Sampling)
Chain 1: Iteration: 2400 / 3000 [ 80%]  (Sampling)
Chain 1: Iteration: 2700 / 3000 [ 90%]  (Sampling)
Chain 1: Iteration: 3000 / 3000 [100%]  (Sampling)
Chain 1:
Chain 1:  Elapsed Time: 1416.78 seconds (Warm-up)
Chain 1:                1041.38 seconds (Sampling)
Chain 1:                2458.16 seconds (Total)
Chain 1:

SAMPLING FOR MODEL 'anon_model' NOW (CHAIN 2).
Chain 2:
Chain 2: Gradient evaluation took 0.000641 seconds
Chain 2: 1000 transitions using 10 leapfrog steps per transition would take 6.41 seconds.
Chain 2: Adjust your expectations accordingly!
Chain 2:
Chain 2:
Chain 2: Iteration:    1 / 3000 [  0%]  (Warmup)
Chain 2: Iteration:  300 / 3000 [ 10%]  (Warmup)
Chain 2: Iteration:  600 / 3000 [ 20%]  (Warmup)
Chain 2: Iteration:  900 / 3000 [ 30%]  (Warmup)
Chain 2: Iteration: 1200 / 3000 [ 40%]  (Warmup)
Chain 2: Iteration: 1500 / 3000 [ 50%]  (Warmup)
Chain 2: Iteration: 1501 / 3000 [ 50%]  (Sampling)
Chain 2: Iteration: 1800 / 3000 [ 60%]  (Sampling)
Chain 2: Iteration: 2100 / 3000 [ 70%]  (Sampling)
Chain 2: Iteration: 2400 / 3000 [ 80%]  (Sampling)
Chain 2: Iteration: 2700 / 3000 [ 90%]  (Sampling)
Chain 2: Iteration: 3000 / 3000 [100%]  (Sampling)
Chain 2:
Chain 2:  Elapsed Time: 746.428 seconds (Warm-up)
Chain 2:                1398.32 seconds (Sampling)
Chain 2:                2144.74 seconds (Total)
Chain 2:
```

```
SAMPLING FOR MODEL 'anon_model' NOW (CHAIN 3).
Chain 3:
Chain 3: Gradient evaluation took 0.001605 seconds
Chain 3: 1000 transitions using 10 leapfrog steps per transition would take 16.05 seconds.
Chain 3: Adjust your expectations accordingly!
Chain 3:
Chain 3:
Chain 3: Iteration:    1 / 3000 [  0%]  (Warmup)
Chain 3: Iteration:  300 / 3000 [ 10%]  (Warmup)
Chain 3: Iteration:  600 / 3000 [ 20%]  (Warmup)
Chain 3: Iteration:  900 / 3000 [ 30%]  (Warmup)
Chain 3: Iteration: 1200 / 3000 [ 40%]  (Warmup)
Chain 3: Iteration: 1500 / 3000 [ 50%]  (Warmup)
Chain 3: Iteration: 1501 / 3000 [ 50%]  (Sampling)
Chain 3: Iteration: 1800 / 3000 [ 60%]  (Sampling)
Chain 3: Iteration: 2100 / 3000 [ 70%]  (Sampling)
Chain 3: Iteration: 2400 / 3000 [ 80%]  (Sampling)
Chain 3: Iteration: 2700 / 3000 [ 90%]  (Sampling)
Chain 3: Iteration: 3000 / 3000 [100%]  (Sampling)
Chain 3:
Chain 3:  Elapsed Time: 876.402 seconds (Warm-up)
Chain 3:                922.713 seconds (Sampling)
Chain 3:                1799.12 seconds (Total)
Chain 3:

SAMPLING FOR MODEL 'anon_model' NOW (CHAIN 4).
Chain 4:
Chain 4: Gradient evaluation took 0.000918 seconds
Chain 4: 1000 transitions using 10 leapfrog steps per transition would take 9.18 seconds.
Chain 4: Adjust your expectations accordingly!
Chain 4:
Chain 4:
Chain 4: Iteration:    1 / 3000 [  0%]  (Warmup)
Chain 4: Iteration:  300 / 3000 [ 10%]  (Warmup)
Chain 4: Iteration:  600 / 3000 [ 20%]  (Warmup)
Chain 4: Iteration:  900 / 3000 [ 30%]  (Warmup)
Chain 4: Iteration: 1200 / 3000 [ 40%]  (Warmup)
Chain 4: Iteration: 1500 / 3000 [ 50%]  (Warmup)
Chain 4: Iteration: 1501 / 3000 [ 50%]  (Sampling)
Chain 4: Iteration: 1800 / 3000 [ 60%]  (Sampling)
Chain 4: Iteration: 2100 / 3000 [ 70%]  (Sampling)
Chain 4: Iteration: 2400 / 3000 [ 80%]  (Sampling)
```

```
Chain 4: Iteration: 2700 / 3000 [ 90%]  (Sampling)
Chain 4: Iteration: 3000 / 3000 [100%]  (Sampling)
Chain 4:
Chain 4:  Elapsed Time: 787.742 seconds (Warm-up)
Chain 4:                439.982 seconds (Sampling)
Chain 4:                1227.72 seconds (Total)
Chain 4:
```

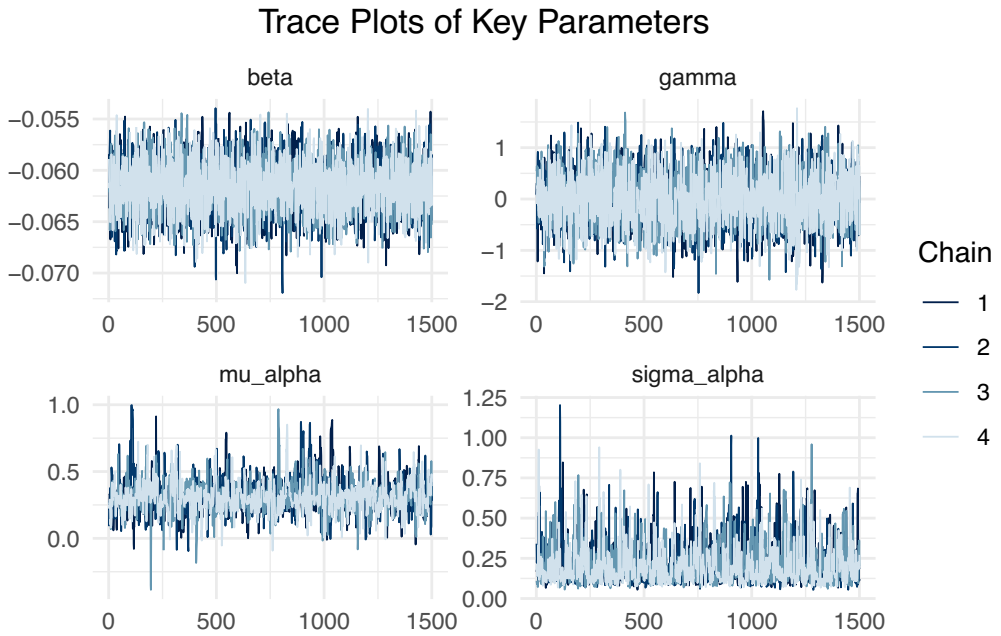
```
Warning: There were 2 divergent transitions after warmup. See
https://mc-stan.org/misc/warnings.html#divergent-transitions-after-warmup
to find out why this is a problem and how to eliminate them.
```

```
Warning: There were 1 transitions after warmup that exceeded the maximum treedepth. Increase
https://mc-stan.org/misc/warnings.html#maximum-treedepth-exceeded
```

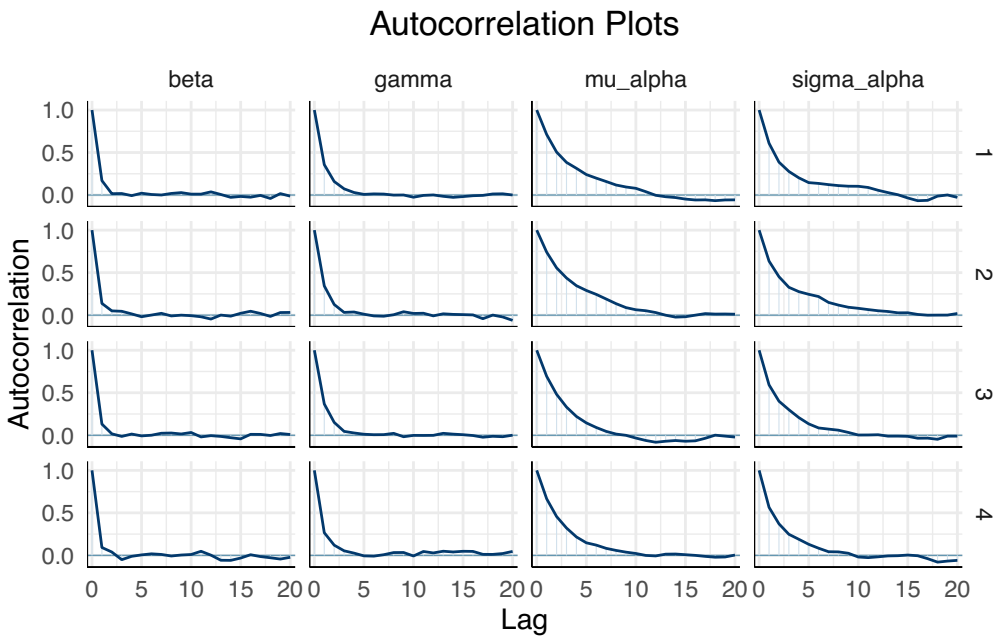
```
Warning: Examine the pairs() plot to diagnose sampling problems
```

Model Checking and Results

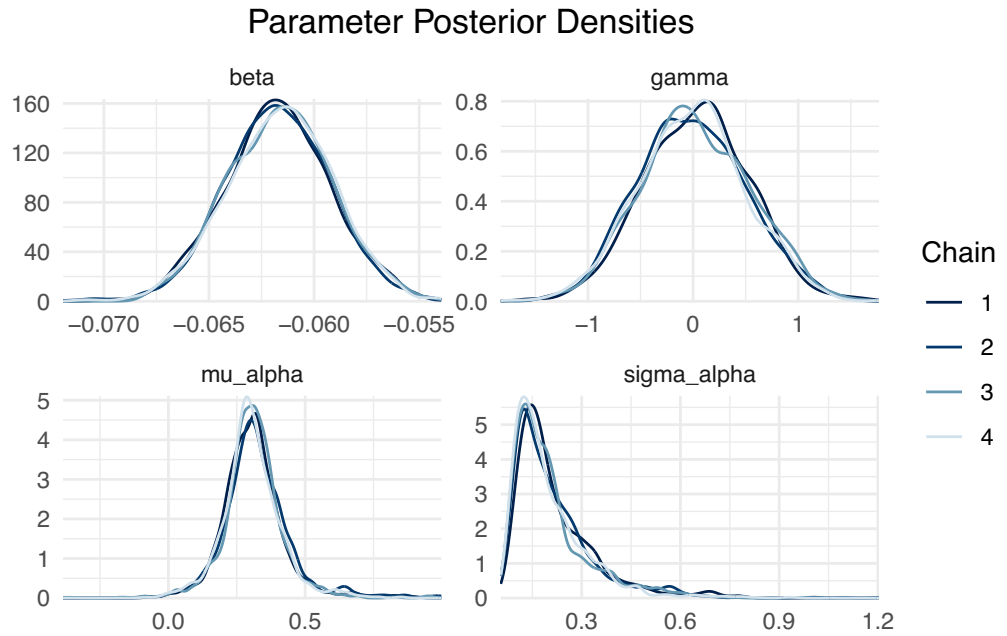
```
# Trace plots with improved aesthetics
mcmc_trace(fit,
           pars = c("beta", "gamma", "mu_alpha", "sigma_alpha"),
           facet_args = list(ncol = 2)) +
  theme_minimal() +
  ggtitle("Trace Plots of Key Parameters") +
  theme(plot.title = element_text(hjust = 0.5))
```




```
# Autocorrelation plots with more lags
mcmc_acf(fit,
         pars = c("beta", "gamma", "mu_alpha", "sigma_alpha"),
         lags = 20) +
  theme_minimal() +
  ggtitle("Autocorrelation Plots") +
  theme(plot.title = element_text(hjust = 0.5))
```

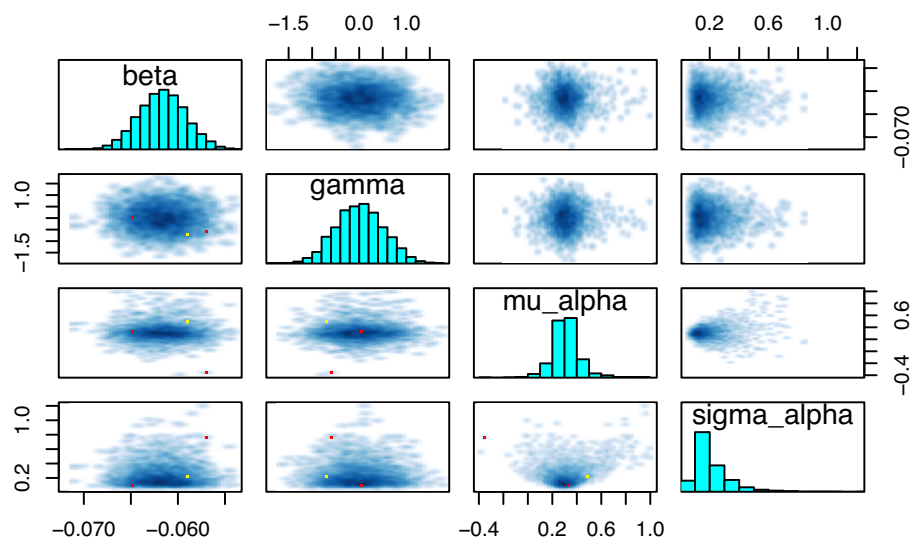



```
# Additional diagnostic plots
# Density plots
mcmc_dens_overlay(fit,
                  pars = c("beta", "gamma", "mu_alpha", "sigma_alpha")) +
  theme_minimal() +
  ggtitle("Parameter Posterior Densities") +
  theme(plot.title = element_text(hjust = 0.5))
```




```
# Pairs plot for parameter correlation
pairs(fit,
      pars = c("beta", "gamma", "mu_alpha", "sigma_alpha"))
```

```
Warning in par(usr): argument 1 does not name a graphical parameter
Warning in par(usr): argument 1 does not name a graphical parameter
Warning in par(usr): argument 1 does not name a graphical parameter
Warning in par(usr): argument 1 does not name a graphical parameter
```



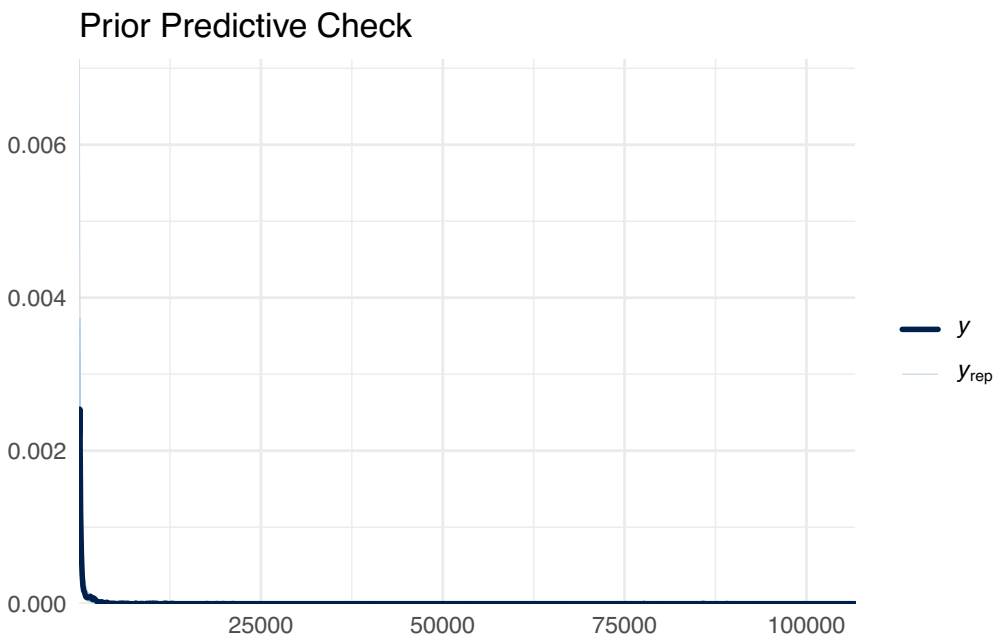
```
# Prior Predictive Check
prior_code <- "
data {
  int<lower=1> N;
  int<lower=1> R;
  int<lower=1> total[N];
}

generated quantities {
  real mu_alpha = normal_rng(logit(0.65), 0.5);
  real<lower=0> sigma_alpha = fabs(normal_rng(0, 0.5));
  real beta = normal_rng(0, 0.5);
  real gamma = normal_rng(0, 0.5);
  vector[R] alpha_raw;
  vector[R] alpha;
  vector[N] y_prior;

  // Generate alpha_raw one element at a time
  for (r in 1:R) {
    alpha_raw[r] = normal_rng(0, 1);
  }

  alpha = mu_alpha + sigma_alpha * alpha_raw;

  for (i in 1:N) {
    real logit_p = alpha[1] + beta * 0.5 + gamma * 0.5;  // Using average values
    y_prior[i] = binomial_rng(total[i], inv_logit(logit_p));
  }
}
"

prior_fit <- stan(
  model_code = prior_code,
  data = list(N = stan_data_simple$N,
              R = stan_data_simple$R,
              total = stan_data_simple$total),
  algorithm = "Fixed_param",
  iter = 1000)
```

```
Trying to compile a simple C file

Running /Library/Frameworks/R.framework/Resources/bin/R CMD SHLIB foo.c
```

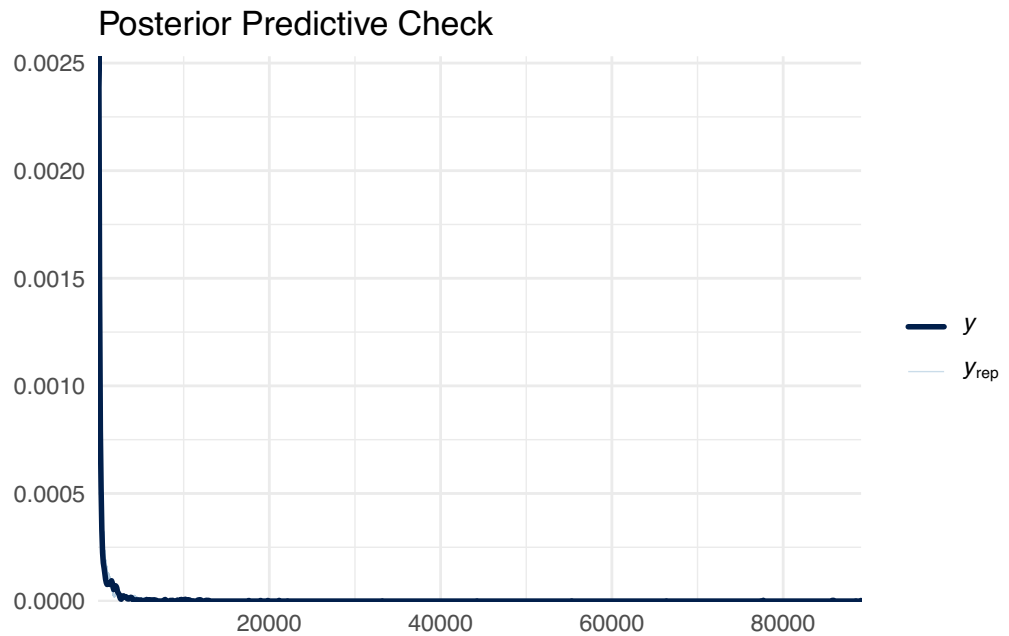
```
using C compiler: ‘Apple clang version 13.0.0 (clang-1300.0.27.3)’
using SDK: ‘’
clang -arch x86_64 -I"/Library/Frameworks/R.framework/Resources/include" -DNDEBUG   -I"/Libra
In file included from <built-in>:1:
In file included from /Library/Frameworks/R.framework/Versions/4.4-x86_64/Resources/library/S
In file included from /Library/Frameworks/R.framework/Versions/4.4-x86_64/Resources/library/R
In file included from /Library/Frameworks/R.framework/Versions/4.4-x86_64/Resources/library/R
/Library/Frameworks/R.framework/Versions/4.4-x86_64/Resources/library/RcppEigen/include/Eigen
#include <cmath>
         ^~~~~~~
1 error generated.
make: *** [foo.o] Error 1

SAMPLING FOR MODEL 'anon_model' NOW (CHAIN 1).
Chain 1: Iteration:   1 / 1000 [  0%]  (Sampling)
Chain 1: Iteration: 100 / 1000 [ 10%]  (Sampling)
Chain 1: Iteration: 200 / 1000 [ 20%]  (Sampling)
Chain 1: Iteration: 300 / 1000 [ 30%]  (Sampling)
Chain 1: Iteration: 400 / 1000 [ 40%]  (Sampling)
Chain 1: Iteration: 500 / 1000 [ 50%]  (Sampling)
Chain 1: Iteration: 600 / 1000 [ 60%]  (Sampling)
Chain 1: Iteration: 700 / 1000 [ 70%]  (Sampling)
Chain 1: Iteration: 800 / 1000 [ 80%]  (Sampling)
Chain 1: Iteration: 900 / 1000 [ 90%]  (Sampling)
Chain 1: Iteration: 1000 / 1000 [100%]  (Sampling)
Chain 1:
Chain 1:  Elapsed Time: 0 seconds (Warm-up)
Chain 1:                0.222 seconds (Sampling)
Chain 1:                0.222 seconds (Total)
Chain 1:

SAMPLING FOR MODEL 'anon_model' NOW (CHAIN 2).
Chain 2: Iteration:   1 / 1000 [  0%]  (Sampling)
Chain 2: Iteration: 100 / 1000 [ 10%]  (Sampling)
Chain 2: Iteration: 200 / 1000 [ 20%]  (Sampling)
Chain 2: Iteration: 300 / 1000 [ 30%]  (Sampling)
Chain 2: Iteration: 400 / 1000 [ 40%]  (Sampling)
Chain 2: Iteration: 500 / 1000 [ 50%]  (Sampling)
Chain 2: Iteration: 600 / 1000 [ 60%]  (Sampling)
Chain 2: Iteration: 700 / 1000 [ 70%]  (Sampling)
Chain 2: Iteration: 800 / 1000 [ 80%]  (Sampling)
Chain 2: Iteration: 900 / 1000 [ 90%]  (Sampling)
Chain 2: Iteration: 1000 / 1000 [100%]  (Sampling)
```

```
Chain 2:
Chain 2:  Elapsed Time: 0 seconds (Warm-up)
Chain 2:                0.22 seconds (Sampling)
Chain 2:                0.22 seconds (Total)
Chain 2:

SAMPLING FOR MODEL 'anon_model' NOW (CHAIN 3).
Chain 3: Iteration:   1 / 1000 [  0%]  (Sampling)
Chain 3: Iteration: 100 / 1000 [ 10%]  (Sampling)
Chain 3: Iteration: 200 / 1000 [ 20%]  (Sampling)
Chain 3: Iteration: 300 / 1000 [ 30%]  (Sampling)
Chain 3: Iteration: 400 / 1000 [ 40%]  (Sampling)
Chain 3: Iteration: 500 / 1000 [ 50%]  (Sampling)
Chain 3: Iteration: 600 / 1000 [ 60%]  (Sampling)
Chain 3: Iteration: 700 / 1000 [ 70%]  (Sampling)
Chain 3: Iteration: 800 / 1000 [ 80%]  (Sampling)
Chain 3: Iteration: 900 / 1000 [ 90%]  (Sampling)
Chain 3: Iteration: 1000 / 1000 [100%]  (Sampling)
Chain 3:
Chain 3:  Elapsed Time: 0 seconds (Warm-up)
Chain 3:                0.222 seconds (Sampling)
Chain 3:                0.222 seconds (Total)
Chain 3:

SAMPLING FOR MODEL 'anon_model' NOW (CHAIN 4).
Chain 4: Iteration:   1 / 1000 [  0%]  (Sampling)
Chain 4: Iteration: 100 / 1000 [ 10%]  (Sampling)
Chain 4: Iteration: 200 / 1000 [ 20%]  (Sampling)
Chain 4: Iteration: 300 / 1000 [ 30%]  (Sampling)
Chain 4: Iteration: 400 / 1000 [ 40%]  (Sampling)
Chain 4: Iteration: 500 / 1000 [ 50%]  (Sampling)
Chain 4: Iteration: 600 / 1000 [ 60%]  (Sampling)
Chain 4: Iteration: 700 / 1000 [ 70%]  (Sampling)
Chain 4: Iteration: 800 / 1000 [ 80%]  (Sampling)
Chain 4: Iteration: 900 / 1000 [ 90%]  (Sampling)
Chain 4: Iteration: 1000 / 1000 [100%]  (Sampling)
Chain 4:
Chain 4:  Elapsed Time: 0 seconds (Warm-up)
Chain 4:                0.227 seconds (Sampling)
Chain 4:                0.227 seconds (Total)
Chain 4:
```

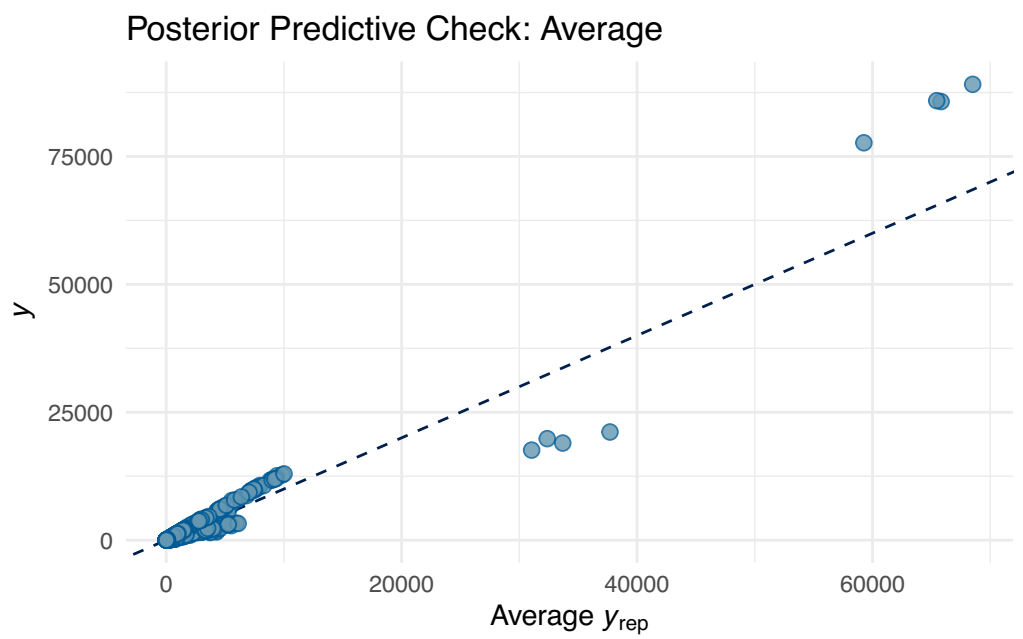
```r
# Plot prior predictive distribution
y_prior <- as.matrix(prior_fit, pars = "y_prior")

ppc_dens_overlay(y = stan_data_simple$y, yrep = y_prior[1:50,]) +
  ggtitle("Prior Predictive Check") +
  theme_minimal()
```




```r
# Posterior Predictive Check
y_rep <- as.matrix(fit, pars = "y_rep")

ppc_dens_overlay(y = stan_data_simple$y, yrep = y_rep[1:50,]) +
  ggtitle("Posterior Predictive Check") +
  theme_minimal()
```

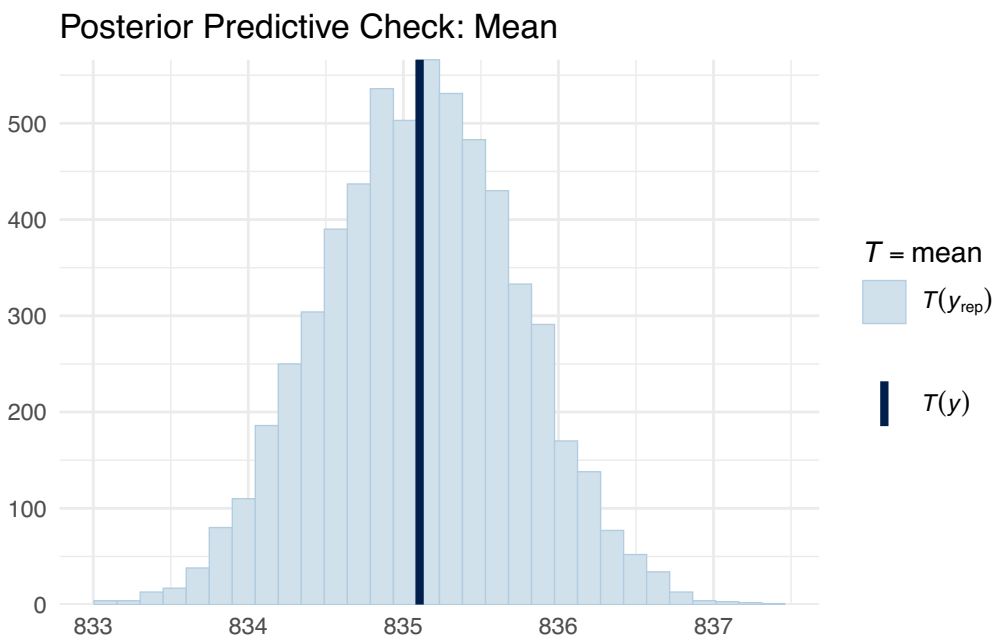



```
# Additional posterior predictive checks
ppc_scatter_avg(y = stan_data_simple$y, yrep = y_rep) +
  ggtitle("Posterior Predictive Check: Average") +
  theme_minimal()
```

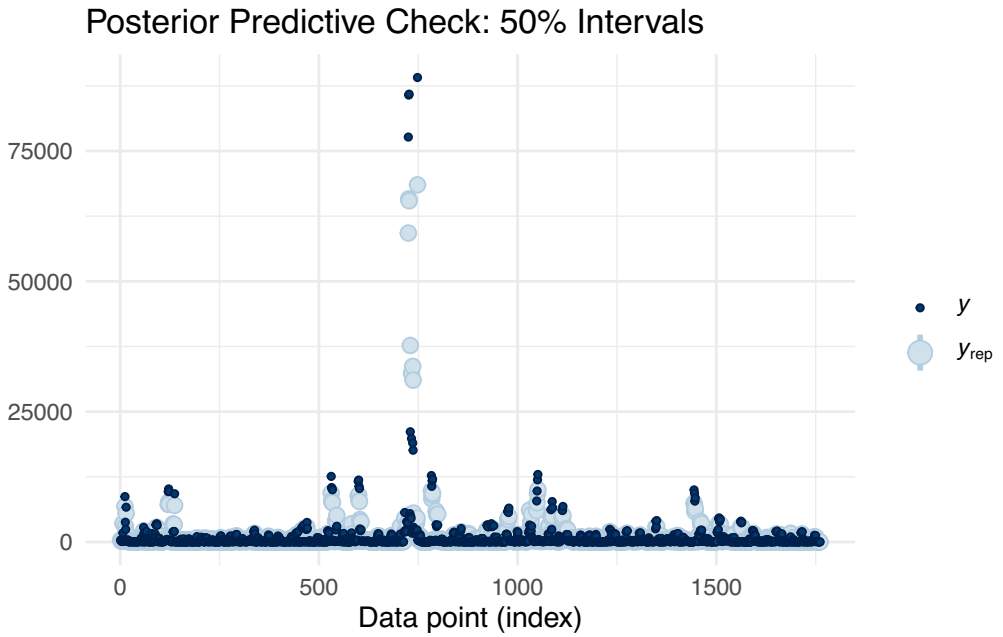


```
ppc_stat(y = stan_data_simple$y, yrep = y_rep, stat = "mean") +
  ggtitle("Posterior Predictive Check: Mean") +
  theme_minimal()
```

```
`stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```




```
ppc_intervals(y = stan_data_simple$y, yrep = y_rep, prob = 0.5) +
  ggtitle("Posterior Predictive Check: 50% Intervals") +
  theme_minimal()
```



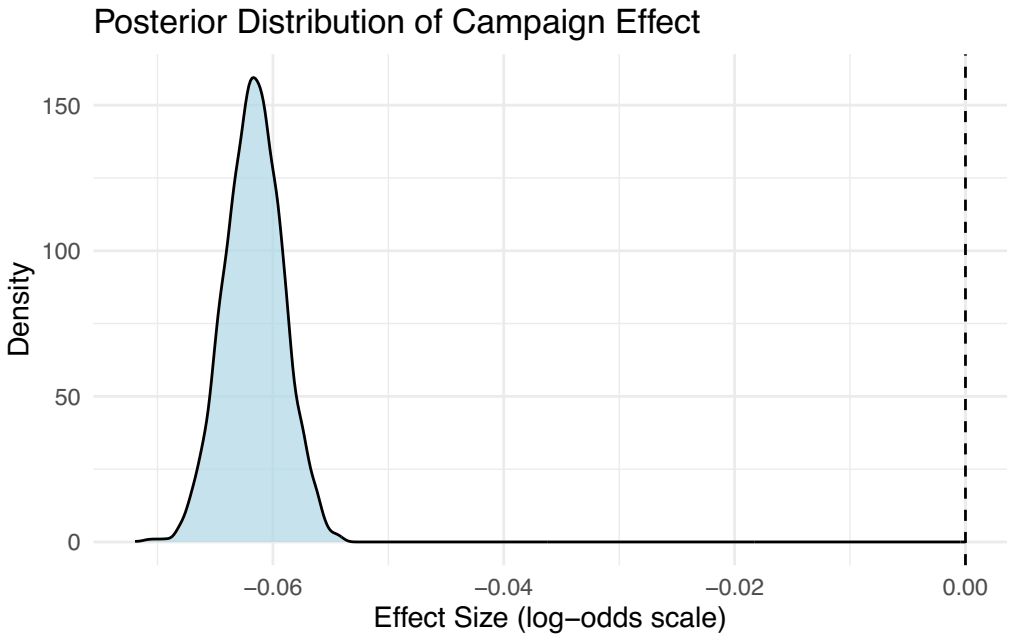

```
head(as.data.frame(summary(fit)$summary[, "n_eff"]), 20)
```

	summary(fit)$summary[, "n_eff"]
alpha_raw[1]	1038.6089
alpha_raw[2]	1224.4854
alpha_raw[3]	1161.2919
alpha_raw[4]	1139.6698
beta	4433.7297
gamma	2708.3778
mu_alpha	960.5969
sigma_alpha	1083.8380
alpha[1]	5917.4252
alpha[2]	6082.8829
alpha[3]	6235.3593
alpha[4]	4562.6382
log_lik[1]	5335.0470
log_lik[2]	4563.1902
log_lik[3]	5335.0014
log_lik[4]	4563.0259
log_lik[5]	4563.4483
log_lik[6]	5335.0016
log_lik[7]	4562.6470
log_lik[8]	4563.1952

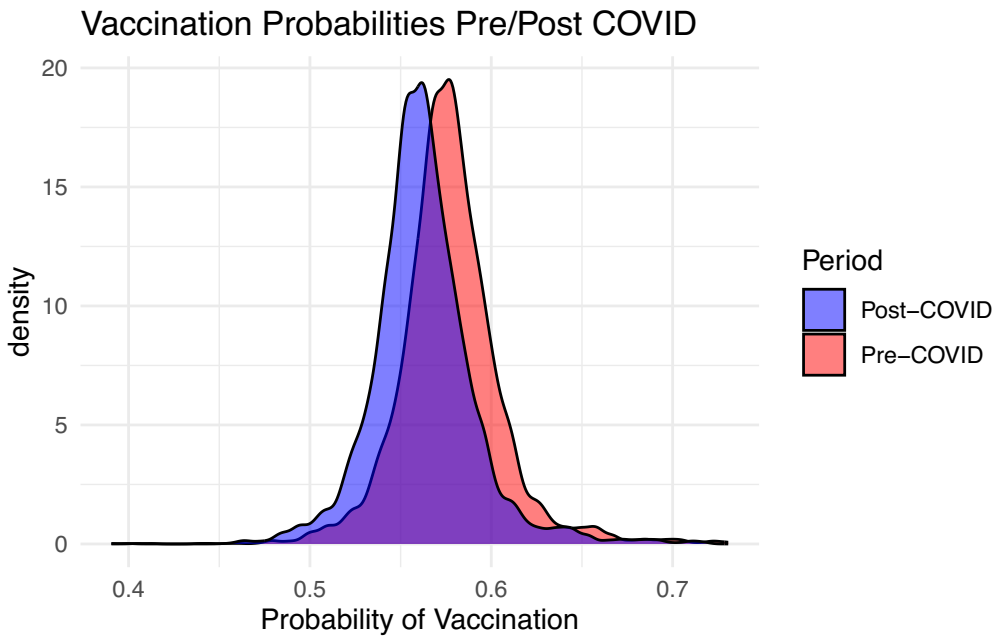
```r
as.data.frame(max(summary(fit)$summary[, "Rhat"]))
```

```
  max(summary(fit)$summary[, "Rhat"])
1                            1.005827
```

```r
# Extract posterior samples for campaign effect (beta parameter)
campaign_effect <- rstan::extract(fit, pars = "beta")$beta

# Calculate probability of negative impact
prob_negative <- mean(campaign_effect < 0)

# Calculate effect size and uncertainty
campaign_summary <- data.frame(
  mean_effect = mean(campaign_effect),
  lower_ci = quantile(campaign_effect, 0.025),
  upper_ci = quantile(campaign_effect, 0.975)
)

# Calculate predicted probabilities for pre/post COVID
predicted_probs <- function(beta_samples, mu_alpha_samples) {
  pre_covid <- plogis(mu_alpha_samples)
  post_covid <- plogis(mu_alpha_samples + beta_samples)
  return(data.frame(pre_covid, post_covid))
}

# Extract samples
mu_alpha_samples <- rstan::extract(fit, pars = "mu_alpha")$mu_alpha
probs <- predicted_probs(campaign_effect, mu_alpha_samples)

# Visualize campaign effect
ggplot(data.frame(effect = campaign_effect), aes(x = effect)) +
  geom_density(fill = "lightblue", alpha = 0.7) +
  geom_vline(xintercept = 0, linetype = "dashed") +
  theme_minimal() +
  ggtitle("Posterior Distribution of Campaign Effect") +
  xlab("Effect Size (log-odds scale)") +
  ylab("Density")
```




```
# Compare pre/post probabilities
ggplot(probs) +
  geom_density(aes(x = pre_covid, fill = "Pre-COVID"), alpha = 0.5) +
  geom_density(aes(x = post_covid, fill = "Post-COVID"), alpha = 0.5) +
  theme_minimal() +
  ggtitle("Vaccination Probabilities Pre/Post COVID") +
  xlab("Probability of Vaccination") +
  scale_fill_manual(values = c("blue", "red"), name = "Period")
```


Vaccination Probabilities Pre/Post COVID
density
20
15
10
5
0
0.4
0.5
0.6
0.7
Probability of Vaccination
Period
Post−COVID
Pre−COVID