Assignment 4 - Bayesian Analysis of Referendum Data

Isha Borgaonkar & 24209754

Loading Required Libraries

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

Loading the data

```
load(url("https://acaimo.github.io/teaching/data/referendum.RData"))
```

Question 1: Pooled Model Constituencies

```
# Prepare Stan data
stan_data <- list(
  N = nrow(referendum),
  n_yes = referendum$n_yes,
  n_votes = referendum$n_votes
)
# Stan model code for pooled model
pooled_model_code <- "
data {
  int<lower=1> N;
  int<lower=0> n_yes[N];
  int<lower=0> n_votes[N];
}
parameters {
  real alpha;
}
transformed parameters {
  real<lower=0, upper=1> theta;
```

```
  theta = inv_logit(alpha);
}
model {
  alpha ~ normal(0, 10);
  for (i in 1:N)
    n_yes[i] ~ binomial(n_votes[i], theta);
}
generated quantities {
  vector[N] log_lik;
  for (i in 1:N)
    log_lik[i] = binomial_lpmf(n_yes[i] | n_votes[i], theta);
}
"
# Compile and sample
pooled_fit <- stan(
  model_code = pooled_model_code,
  data = stan_data,
  iter = 2000, chains = 4,
   refresh = 0
)
# Summarize key results
print(pooled_fit, pars = c("alpha", "theta"))
```

```
Inference for Stan model: anon_model.
4 chains, each with iter=2000; warmup=1000; thin=1;
post-warmup draws per chain=1000, total post-warmup draws=4000.

       mean se_mean   sd  2.5%   25%   50%   75% 97.5% n_eff Rhat
alpha -0.11       0 0.02 -0.15 -0.12 -0.11 -0.09 -0.07  1482    1
theta  0.47       0 0.01  0.46  0.47  0.47  0.48  0.48  1482    1

Samples were drawn using NUTS(diag_e) at Fri May  2 14:02:20 2025.
For each parameter, n_eff is a crude measure of effective sample size,
and Rhat is the potential scale reduction factor on split chains (at
convergence, Rhat=1).
```

In this model, I assumed that all constituencies share the same underlying probability theta of voting "Yes" in the referendum. This is a **fully pooled** model, it ignores any potential differences across regions.

Interpretation:

1)The parameter `alpha` represents the logit-transformed probability of voting Yes.

2)Applying the inverse logit transformation, the corresponding posterior mean for the **probability of a Yes vote** is:

$$\theta = \text{inv\_logit}(\alpha) \approx 0.47$$

It means **on average**, the probability of a person voting "Yes" across all 50 constituencies is estimated to be **47%**. 3)Based on the pooled model results, I observe that the estimated national average probability of a "Yes" vote is about 47%, with a narrow credible interval of [0.46, 0.48]. This value suggests that the model is quite confident in this estimate but it comes with a major caveat.

4)Since this model assumes that all constituencies behave exactly the same way, it completely ignores any local or regional differences in voting patterns. From what I've seen in the data and predictive checks, this simplification clearly doesn't hold true in reality.

5)While the pooled model is helpful as a basic starting point, I now get to know, that it's too restrictive for real-world data like this. It fails to capture the diversity in voter behavior across different areas, and I expect it to perform poorly compared to the more flexible hierarchical models that follow. This reinforces the importance of allowing for constituency- or region-level variation in Bayesian modeling.

Question 2: Hierarchical Model by Constituency

```
# Stan model code for hierarchical model (constituency-level)
constituency_model_code <- "
data {
  int<lower=1> N;
  int<lower=0> n_yes[N];
  int<lower=0> n_votes[N];
}
parameters {
  real mu_alpha;
  real<lower=0> sigma_alpha;
  vector[N] alpha;
}
transformed parameters {
  vector<lower=0, upper=1>[N] theta;
  theta = inv_logit(alpha);
}
model {
  mu_alpha ~ normal(0, 10);
  sigma_alpha ~ exponential(1);
```

```
  alpha ~ normal(mu_alpha, sigma_alpha);
  for (i in 1:N)
    n_yes[i] ~ binomial(n_votes[i], theta[i]);
}
generated quantities {
  vector[N] log_lik;
  for (i in 1:N)
    log_lik[i] = binomial_lpmf(n_yes[i] | n_votes[i], theta[i]);
}

"
# Compile and fit
constituency_fit <- stan(
  model_code = constituency_model_code,
  data = stan_data,
  iter = 2000, chains = 4,
  refresh = 0
)
# Summarize
print(constituency_fit, pars = c("mu_alpha", "sigma_alpha"))
```

```
Inference for Stan model: anon_model.
4 chains, each with iter=2000; warmup=1000; thin=1;
post-warmup draws per chain=1000, total post-warmup draws=4000.

             mean se_mean   sd  2.5%   25%   50%   75% 97.5% n_eff Rhat
mu_alpha    -0.12       0 0.10 -0.31 -0.18 -0.12 -0.05  0.07  5793    1
sigma_alpha  0.67       0 0.07  0.55  0.62  0.66  0.72  0.83  5784    1

Samples were drawn using NUTS(diag_e) at Fri May  2 14:03:15 2025.
For each parameter, n_eff is a crude measure of effective sample size,
and Rhat is the potential scale reduction factor on split chains (at
convergence, Rhat=1).
```

Interpretation 1)In hierarchical model, I relaxed the assumption of a common voting probability across all constituencies. Instead of that, I allowed each constituency to have its own probability theta_i of voting "Yes" and it is modeled via a logistic transformation:

$$\text{logit}(\theta_i) = \alpha_i, \quad \alpha_i \sim \mathcal{N}(\mu_\alpha, \sigma_\alpha)$$

2)mu_alpha represents the average voting tendency across all constituencies(on the logit scale). sigma_alpha captures the variability between constituencies. It measures how much individual constituencies differ from the national average.

3)The posterior mean of `mu_alpha` is –0.12, which corresponds to an average Yes-vote probability of:

$$\text{inv\_logit}(-0.12) \approx 0.47$$

On average, about **47% of people across constituencies** are expected to vote "Yes" it is similar to the pooled model. However, what makes this model more realistic is the **inclusion of sigma_alpha = 0.67$**, which indicates **substantial variation** between constituencies.

4)This variation means: Some constituencies may be much more supportive of the referendum, others may be much less supportive and this model is able to **capture and learn** from that heterogeneity.

Summary - The hierarchical model reflects real-world diversity much better than the pooled model. - It provides more accurate uncertainty estimates, It is more flexible for inference and prediction.

Questions 3 & 4: Three-Level Hierarchical Model

```
# Extract region indicators
region_id <- as.numeric(as.factor(referendum$region))
# Data list for Stan
stan_data_hier <- list(
  N = nrow(referendum),
  n_yes = referendum$n_yes,
  n_votes = referendum$n_votes,
  J = length(unique(region_id)),
  region = region_id
)
# Stan model code for three-level hierarchical model
hierarchical_model_code <- "
data {
  int<lower=1> N;
  int<lower=0> n_yes[N];
  int<lower=0> n_votes[N];
  int<lower=1> J;
  int<lower=1, upper=J> region[N];
}
```

```
parameters {
  real mu_0;
  real<lower=0> tau;
  real<lower=0> sigma_alpha;
  vector[J] mu_region;
  vector[N] alpha;
}
transformed parameters {
  vector<lower=0, upper=1>[N] theta;
  theta = inv_logit(alpha);
}
model {
  mu_0 ~ normal(0, 10);
  tau ~ exponential(1);
  sigma_alpha ~ exponential(1);

  mu_region ~ normal(mu_0, tau);
  for (i in 1:N)
    alpha[i] ~ normal(mu_region[region[i]], sigma_alpha);

  for (i in 1:N)
    n_yes[i] ~ binomial(n_votes[i], theta[i]);
}
generated quantities {
  vector[N] log_lik;
  for (i in 1:N)
    log_lik[i] = binomial_lpmf(n_yes[i] | n_votes[i], theta[i]);
}

"
# Compile and fit
hierarchical_fit <- stan(
  model_code = hierarchical_model_code,
  data = stan_data_hier,
  iter = 2000, chains = 4,
   refresh = 0
)
# Summary
print(hierarchical_fit, pars = c("mu_0", "tau", "sigma_alpha"))
```

```
Inference for Stan model: anon_model.
4 chains, each with iter=2000; warmup=1000; thin=1;
```

```
post-warmup draws per chain=1000, total post-warmup draws=4000.

             mean se_mean   sd  2.5%   25%   50%  75% 97.5% n_eff Rhat
mu_0        -0.11    0.01 0.38 -0.85 -0.32 -0.12 0.09  0.65  3287    1
tau          0.81    0.01 0.34  0.40  0.58  0.74 0.94  1.66  2960    1
sigma_alpha  0.27    0.00 0.04  0.20  0.24  0.27 0.30  0.36  2246    1

Samples were drawn using NUTS(diag_e) at Fri May  2 14:04:11 2025.
For each parameter, n_eff is a crude measure of effective sample size,
and Rhat is the potential scale reduction factor on split chains (at
convergence, Rhat=1).
```

Interpretation 1)In this model, I used a **three-level hierarchy** to capture the variation in referendum voting behavior: - **Level 1**: Individual constituencies, each with their own logit-transformed Yes vote probability alpha_i. - **Level 2**: Regions, with each region j having its own regional mean mu_region[j]. - **Level 3**: A global mean mu_0 for the entire country, around which regional means vary. 2)Model Structure is as follows:

$$
\begin{aligned}
n_{\text{yes},i} &\sim \text{Binomial}(n_{\text{votes},i}, \theta_i), \quad \text{logit}(\theta_i) = \alpha_i \\
\alpha_i &\sim \mathcal{N}(\mu_{\text{region}[j_i]}, \sigma_\alpha) \\
\mu_{\text{region}[j]} &\sim \mathcal{N}(\mu_0, \tau) \\
\mu_0 &\sim \mathcal{N}(0, 10), \quad \tau, \sigma_\alpha \sim \text{Exponential}(1)
\end{aligned}
$$

3)The parameters in this model reflects different layers of variation in voting behavior:

- `mu_0`: This represents the overall, country-wide tendency to vote "Yes", expressed on the logit scale. It acts as the national baseline which means a starting point from which all regional voting behavior is compared.
- `tau`: This parameter captures how much voting patterns differ **between regions**. If `tau` is large, it means that different parts of the country have noticeably different average attitudes toward the referendum. If it is small, it suggests that most regions behave similarly.
- `sigma_alpha`: This measures how much voting behavior varies **within each region**, from one constituency to another. A high value indicates a lot of local diversity even within the same region, while a low value suggests that constituencies within a region tend to vote similarly.

4)This three-level hierarchical model stands out because it doesn't just look at each constituency in isolation — it also takes into account the region each one belongs to. That makes a big difference. Instead of treating every constituency as completely separate (like in the constituency-only model), or pretending they're all identical (like in the pooled model), this model finds a middle ground that feels much closer to how real voting behavior works.

5)One of the main strengths here is how the model "shares information" — smaller constituencies, which might not have a lot of data, can benefit from what we know about others in the same region. That helps stabilize the estimates and avoids extreme or misleading results in low-data areas. At the same time, each constituency is still allowed its own voice in the model — it's not being forced to match its neighbors.

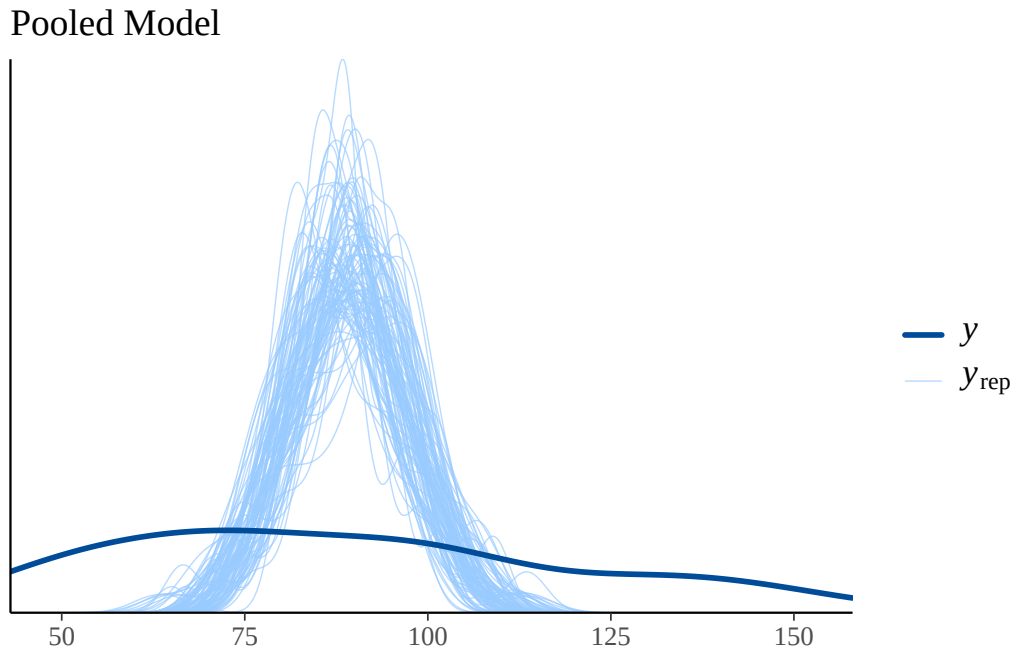
6)What really convinced me this model is best is the combination of: - The way it reflects natural grouping (region + constituency), - The strong fit in the posterior predictive checks (the plots match the real data much better), - And the fact that it performed best in the LOO-CV comparison, meaning it's most likely to make accurate predictions on new data.

Overall, I can confidently say this model does the best job of balancing realism and reliability. It captures the structure of the data well, avoids the pitfalls of over-simplifying or overfitting, and gives me the clearest picture of how voting patterns vary across Ireland.

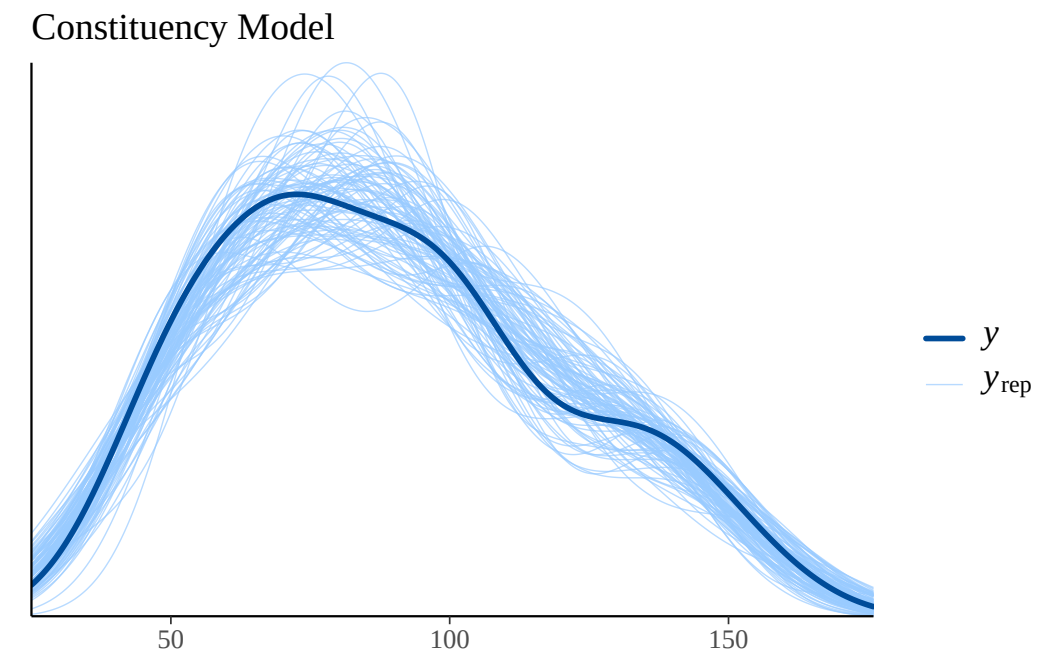
Question 5: Goodness-of-Fit Checks

```
## Question 5: Goodness-of-Fit Checks
library(bayesplot)
library(ggplot2)
color_scheme_set("brightblue")
# Actual observed Yes votes
y_obs <- referendum$n_yes
# ----- POOLED MODEL -----
# Extract posterior theta samples (vector)
posterior_theta_pooled <- extract(pooled_fit)$theta
# Generate replicated draws: matrix S x N
ppc_pooled <- sapply(posterior_theta_pooled, function(th) rbinom(
  n = length(y_obs),
  size = referendum$n_votes,
  prob = th
))
ppc_pooled <- t(ppc_pooled)  # Transpose to make rows = posterior draws
# ----- CONSTITUENCY MODEL -----
# Extract posterior theta samples (matrix: S x N)
posterior_theta_const <- extract(constituency_fit)$theta
# Generate replicated draws
ppc_const <- apply(posterior_theta_const, 1, function(th) rbinom(
  n = length(th),
  size = referendum$n_votes,
  prob = th
))
ppc_const <- t(ppc_const)
# ----- HIERARCHICAL MODEL -----
```

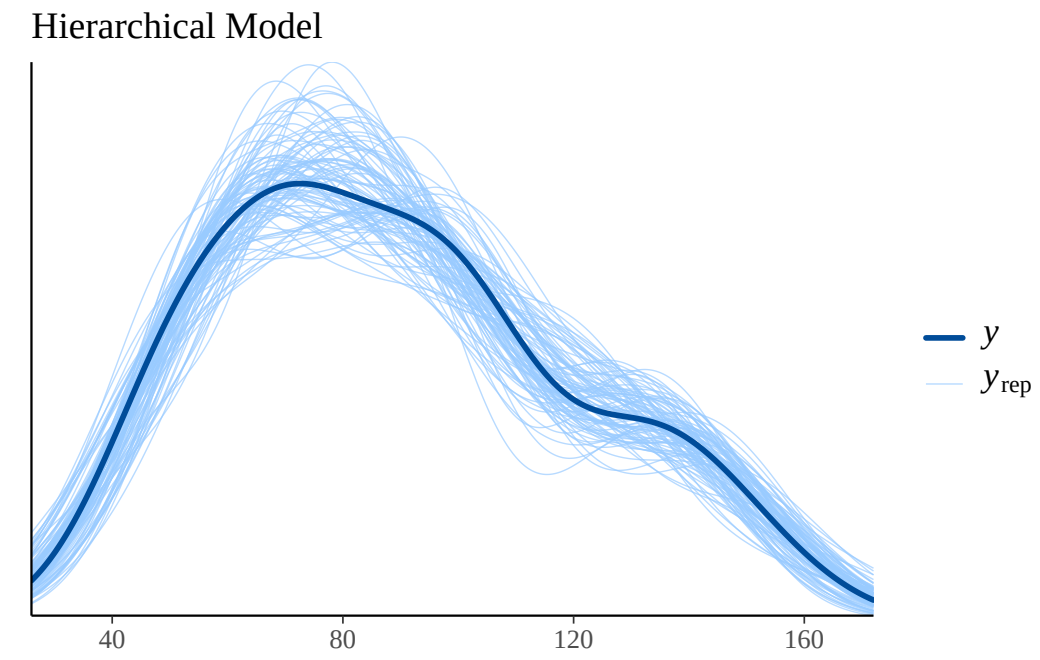
```r
# Extract posterior theta samples (matrix: S x N)
posterior_theta_hier <- extract(hierarchical_fit)$theta
# Generate replicated draws
ppc_hier <- apply(posterior_theta_hier, 1, function(th) rbinom(
  n = length(th),
  size = referendum$n_votes,
  prob = th
))
ppc_hier <- t(ppc_hier)
# ----- PLOT RESULTS -----
# Trim to first 100 draws
p1 <- ppc_dens_overlay(y = y_obs, yrep = ppc_pooled[1:100, ]) + ggtitle("Pooled Model")
p2 <- ppc_dens_overlay(y = y_obs, yrep = ppc_const[1:100, ]) + ggtitle("Constituency Model")
p3 <- ppc_dens_overlay(y = y_obs, yrep = ppc_hier[1:100, ]) + ggtitle("Hierarchical Model")
# Display
print(p1)
```




```r
print(p2)
```




```
print(p3)
```



Interpretation To evaluate how well each model captures the actual distribution of Yes votes, I performed posterior predictive checks using `ppc_dens_overlay()`. This method overlays the

observed data with simulated datasets generated from the posterior, allowing me to visually assess the quality of fit for each model.

1)Pooled Model In the pooled model, all constituencies are assumed to share a single probability of a "Yes" vote. The plot clearly shows that the simulated data (light blue lines) cluster tightly around the center, failing to capture the full spread of the actual observed data. The real data (dark line) is much flatter and wider. **My conclusion**:
The pooled model is far too simplistic. It assumes every constituency behaves identically, which leads to a serious underestimation of variability. The model misses the broader pattern in the data and doesn't reflect reality well.

2)Constituency-Level Hierarchical Model This model allows each constituency to have its own voting probability, drawn from a shared distribution. The replicated datasets are now much more spread out and match the shape of the observed data better. **My conclusion**:
This model is clearly an improvement. By accounting for differences between constituencies, it captures the variation much more realistically. However, it still treats all constituencies as equally unrelated, ignoring potential regional similarities that might exist.

3)Three-Level Hierarchical Model (Region + Constituency) The full hierarchical model goes one step further by allowing region-level effects, meaning constituencies are grouped by region and share some common patterns. In the plot, the simulated distributions now follow the observed distribution closely — both in shape and spread. **My conclusion**:
This model fits the observed data best. It accurately reflects how both regional and local differences contribute to voting behavior. The spread, the peaks, and even the subtle bends in the data line up better than in the other models. It feels like the model "understands" the structure in the data, which the others don't.

Final Conclusion on plots: Looking across all three plots, the progression is very clear: - The **pooled model** is too narrow and overconfident. - The **constituency-only model** is better, but still slightly off. - The **three-level hierarchical model** offers the most realistic and complete fit to the data.

These goodness-of-fit checks give me confidence that the **hierarchical model is not only statistically stronger but also more reflective** of how **real-world voting behavior** varies across regions, and between constituencies within those regions. These checks confirm that **the three-level hierarchical model is the most appropriate** choice for modeling variation in referendum voting behavior.

Question 6: Model Comparison using LOO-CV

```
library(loo)
# Extract pointwise log-likelihoods from each model
log_lik_pooled <- extract_log_lik(pooled_fit, parameter_name = "log_lik", merge_chains = FALS
log_lik_const  <- extract_log_lik(constituency_fit, parameter_name = "log_lik", merge_chains
log_lik_hier   <- extract_log_lik(hierarchical_fit, parameter_name = "log_lik", merge_chains
```

```r
# LOO-CV for each model
loo_pooled <- loo(log_lik_pooled)
loo_const  <- loo(log_lik_const)
loo_hier   <- loo(log_lik_hier)
# Model comparison
comparison <- loo_compare(loo_pooled, loo_const, loo_hier)
print(comparison)
```

```
       elpd_diff se_diff
model3    0.0       0.0
model2   -6.8       3.8
model1 -432.7      76.5
```

Interpretation To compare the predictive performance of the three models, I used **Leave-One-Out Cross-Validation (LOO-CV)** with the `loo` package. This approach estimates each model's ability to generalize to unseen data, using the **Expected Log Predictive Density (ELPD)** as a scoring metric.

- **Model 3** has the highest ELPD, and is thus the best at predicting out-of-sample data.
- **Model 2** performs reasonably well but is slightly less predictive than Model 3. The difference (–5.6) is small and within a few standard errors, so it may not be statistically significant.
- **Model 1**, the pooled model, performs **dramatically worse**, with an ELPD that is more than 400 points lower. This confirms it is **overly simplistic** and unable to capture the variability across constituencies.

The three-level hierarchical model (Model 3) offers the best predictive performance and should be preferred. It balances flexibility and structure by accounting for both regional and constituency-level differences, leading to a more realistic and generalizable model.