Final

1.

(a) N_b, N_g - number of each type of fish
S_1, S_2 - size of sample at time 1 and 2
C_b - number of blue fish caught (at time 1)
T_1 - Number of fish tagged (at time 1)
T_2 - number of tagged fish caught (at time 2)

(b) $\text{Beta}(\frac{N_b}{N_g}; 1, 1)$

(c) $P(N_b, N_g|S_1, S_2, T) \propto \text{Bin}(C_b; S_1, \frac{N_b}{N_g})\text{Bin}(T_2; S_2, \frac{T_1}{N_b, N_g})\text{Beta}(\frac{N_b}{N_g}; 1, 1)$

2.

(a) The posterior $p(a, b|x, y) \propto \prod_i (\frac{(bx_i)^a}{\Gamma(a)} y_i^{a-1} e^{-bx_i y_i}) \frac{1}{\sqrt{2\pi}\log(2)a} e^{-\frac{1}{2(\log 2)^2}(\log a - \log 5)^2} \frac{1}{\sqrt{2\pi}\log(10)b} e^{-\frac{1}{2(\log 10)^2}(\log b - (\log 0.1))^2}$
Therefore the $\log p(a, b|x, y) = \sum_i (a\log(bx_i) - \log(\Gamma(a)) + (a-1)\log(y_i) - bx_i y_i) - \log(\sqrt{2\pi}\log(2)) - \log(a) - \frac{1}{2(\log 2)^2}(\log a - \log 5)^2 - \log(\sqrt{2\pi}\log(10)) - \log(b) - \frac{1}{2(\log 10)^2}(\log b - \log 0.1)^2$
gradient:
$\frac{d\log p(a,b|x,y)}{da} = \sum_i (\log bx_i - \Gamma(\alpha) + \log(y_i)) - \frac{1}{a} - \frac{1}{2(\log 2)^2}(\frac{2\log\frac{a}{5}}{a})$
$\frac{d\log p(a,b|x,y)}{db} = \sum_i (\frac{a}{b} - x_i y_i) - \frac{1}{b} - \frac{1}{2(\log 0.1)^2}(\frac{2\log\frac{b}{0.1}}{b})$

(b) First we program the gradients and check them numerically:

```
library("foreign")
library("rstan")
library("dplyr")
library("boot")
library("ggplot2")
setwd("~/Documents/BDA/Final")

log_p_th <- function(th, x, y){
  a <- th[1]
  b <- th[2]
  log_a_prior <- dlnorm(a, meanlog = log(5), sdlog = log(2), log = TRUE)
  log_b_prior <- dlnorm(b, meanlog = log(0.1), sdlog = log(10), log = TRUE)
  log_lik <- rep(NA, length(x))
  for(i in 1:n)
    log_lik[i] <- dgamma(y[i], a, rate = b * x[i], log = TRUE)
  log_likelihood <- sum(log_lik)
  return (log_likelihood + log_a_prior + log_b_prior)
}

gradient_th <- function(th, x, y){
  a <- th[1]
  b <- th[2]
  d_a_lik <- rep(NA, n)
  d_b_lik <- rep(NA, n)
  for(i in 1:n){
```

```
    d_a_lik[i] <- log(b * x[i]) - digamma(a) + log(y[i])
    d_b_lik[i] <- (a / b) - (y[i] * x[i])
  }
  d_a_prior <-  1/(2 * log(2) ^ 2) * (2 * log(a/5) / a) + (1 / a)
  d_a <- sum(d_a_lik) - d_a_prior
  d_b_prior <- 1/(2 * log(10) ^ 2) * (2 * log(b/0.1) / b) + (1 / b)
  d_b <- sum(d_b_lik) - d_b_prior
  return (c(d_a, d_b))
}

gradient_th_numerical <- function(th, x, y){
  d <- length(th)
  e <- .0001
  diff <- rep(NA, d)
  for (k in 1:d){
    th_hi <- th
    th_lo <- th
    th_hi[k] <- th[k] + e
    th_lo[k] <- th[k] - e
    diff[k] <- (log_p_th(th_hi, x, y) - log_p_th(th_lo, x, y))/(2 * e)
  }
  return (diff)
}

n <- 100
a <- 2
b <- .3
gen_fake_data <- function(n, a, b){
  x <- exp(rnorm(n))
  y <- rep(NA, n)
  for(i in 1:n)
    y[i] <- rgamma(1, a, rate = b * x[i])
  return (list(x = x, y = y))
}

data <- gen_fake_data(100, 2, .3)
x <- data$x
y <- data$y
gradient_th(c(2, .3), x, y)
```

```
## [1]  -1.745743 -22.381519
```

```
gradient_th_numerical(c(2, .3), x, y)
```

```
## [1]  -1.745743 -22.381494
```

Then setting up HMC:

```
fround <- function (x, digits) {
  format (round (x, digits), nsmall=digits)
}
```

```
hmc_iteration <- function(th, x, y, epsilon, L, M) {
  M_inv <- 1/M
  d <- length(th)
  phi <- rnorm (d, 0, sqrt(M))
  th_old <- th
  log_p_old <- log_p_th (th, x, y) - 0.5 * sum(M_inv * phi ^ 2)
  phi <- phi + 0.5 * epsilon * gradient_th(th, x, y)
  for (l in 1:L) {
    th <- th + epsilon * M_inv * phi
    phi <- phi + (if (l == L) 0.5 else 1) * epsilon * gradient_th(th, x, y)
  }
  phi <- -phi
  log_p_star <- log_p_th(th, x, y) - 0.5 * sum(M_inv * phi ^ 2)
  r <- exp(log_p_star - log_p_old)
  if(is.nan(r)) r <- 0
  p_jump <- min(r, 1)
  th_new <- if (runif(1) < p_jump) th else th_old
  return (list (th = th_new, p_jump = p_jump))
}

hmc_run <- function(starting_values, iter, epsilon_0, L_0, M) {
  chains <- nrow(starting_values)
  d <- ncol (starting_values)
  sims <- array(NA, c(iter, chains, d), dimnames = list(NULL, NULL, colnames(starting_values)))
  warmup <- 0.5 * iter
  p_jump <- array(NA, c(iter, chains))
  for (j in 1:chains) {
    th <- starting_values[j,]
    for (t in 1:iter) {
      epsilon <- runif(1, 0, 2 * epsilon_0)
      L <- ceiling(2 * L_0 * runif(1))
      temp <- hmc_iteration(th, x, y, epsilon, L, M)
      p_jump[t, j] <- temp$p_jump
      sims[t, j, ] <- temp$th
      th <- temp$th
    }
  }
  print(cat("Avg acceptance probs:", fround(colMeans(p_jump[(warmup + 1):iter,]), 2), "\n"))
  return(monitor(sims, warmup, print=FALSE))
}
```

We use a mass vector of $[\frac{1}{\log(2)^2}, \frac{1}{\log(10)^2}]$ since we only know the variance of the priors. We start with the default parameters for ϵ_0 and L_0

```
parameter_names <- c("a", "b")
d <- length(parameter_names)
chains <- 4
mass_vector <- c(1 / (log(2) ^ 2), 1 / (log(10) ^ 2))
starts <- array(NA, c(chains, d), dimnames = list(NULL, parameter_names))
for (j in 1:chains) {
  starts[j,] <- c(rlnorm(1, meanlog = log(5), sdlog = log(2)), rlnorm(1, meanlog = log(0.1), sdlog = log
}
M1 <- hmc_run(starting_values = starts, iter = 250, epsilon_0 = .05, L_0 = 20, M = mass_vector)
```

```
## Avg acceptance probs: 0.00 0.14 0.18 0.18
## NULL
```

These acceptance probabilities were too low so we adjusted ϵ_0 and L_0

```
M1 <- hmc_run(starting_values = starts, iter = 250, epsilon_0 = .01, L_0 = 100, M = mass_vector)
```

```
## Avg acceptance probs: 0.57 0.58 0.66 0.62
## NULL
```

```
knitr::kable(as.data.frame(M1), digits = 2)
```

	mean	se_mean	sd	2.5%	25%	50%	75%	97.5%	n_eff	Rhat
a	1.74	0.02	0.22	1.36	1.56	1.72	1.92	2.19	167.99	1.01
b	0.25	0.00	0.04	0.19	0.22	0.25	0.28	0.32	155.49	1.01

The r-hat values are less than 1.1 so it seems that our sampler has reasonably converged

3.

(a) Likelihood function: $P(y|\theta,\sigma^2) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{1}{2\sigma^2}(y-\theta)^2}$

Posterior: $P(\theta|y,\sigma^2) = \frac{\frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{1}{2\sigma^2}(y-\theta)^2} I_{[0,1]}}{\int_0^1 \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{1}{2\sigma^2}(y-\theta)^2} d\theta}$ where $I_{[0,1]}$ is the indicator function

(b) Here we simulate 100 ys and compute estimates of θ 20 times. The output is the number of times that the MLE had a lower mean squared error than the posterior mean

```
sigma = 30
sim_mse <- function(sigma, n) {
  theta_0 <- runif(n)
  mse_mle <- rep(NA, n)
  mse_post <- rep(NA, n)
  theta <- seq(from = 0, to = 1, length.out = 1000)
  for(i in 1:n) {
    y <- rnorm(1, mean = theta_0[i], sd = sigma)
    mle <- max(min(y, 1), 0)
    norm <- sum(dnorm(y, mean = theta, sd = sigma))
    post_mean <- sum(theta * (dnorm(y, mean = theta, sd = sigma)/norm))
    mse_mle[i] <- (mle - theta_0[i])^2
    mse_post[i] <- (post_mean - theta_0[i])^2
  }
  list(mse_mle = mse_mle, mse_post = mse_post)
}
count <- 0
for(i in 1:20){
  sims <- sim_mse(sigma, 100)
  if(mean(sims$mse_post) > mean(sims$mse_mle))
    count <- count + 1
}
count
```

```
## [1] 0
```

(c) The mean squared error of the posterior mean is $\int(\int_0^1 \theta P(\theta|y,\sigma^2)d\theta - \theta_0)^2 N(y|\theta_0,\sigma^2)dy$ However, the mean squared error of the MLE is $(\theta_0)^2 \int_{-\infty}^0 N(y|\theta_0,\sigma^2)dy + (1-\theta_0)^2 \int_1^\infty N(y|\theta_0,\sigma^2)dy + \int_0^1 (y-\theta_0)^2 N(y|\theta_0,\sigma^2)dy$ so there is always a constant term $(\theta_0)^2 \int_{-\infty}^0 N(y|\theta_0,\sigma^2)dy + (1-\theta_0)^2 \int_1^\infty N(y|\theta_0,\sigma^2)dy$ that grows as σ increases. Since the posterior mean does not have this term, there will be some sigma where the posterior mean MSE is always lower than the MLE.

4.

(a) We have a logit-binomial regression model where we try to predict the number of Democratic voters \$D_{s, m} ~ binomial $(N_{s,m}, p_{s,m})$ where s indexes state and m indexes marital status (1 if married, 0 if not). D is the number of Democratic voters in that survey category, N is the number of total respondents and p is the percentage of democratic voters. Then: $p_{s,m} = \text{logit}^{-1}(\beta_i + \beta_m x_m + \beta_s X_s + \beta_{s,m} X_s x_m)$ where x_m is a dummy variable for marital status, X_s are dummy variables for state and the corresponding βs are the regression coefficients

$\beta_i \sim N(0, \sigma_i)$
$\beta_m \sim N(0, \sigma_m)$
$\beta_s \sim N(0, \sigma_s)$ $\beta_{s,m} \sim N(0, \sigma_{s,m})$

$\sigma_i \sim N^T(0, 5)$
$\sigma_m \sim N^T(0, 5)$
$\sigma_s \sim N^T(0, 5)$ $\sigma_{s,m} \sim N^T(0, 5)$ where N^T indicates a truncated normal

(b) Preprocessing the data:

```
data <- read.dta("pew_research_center_june_elect_wknd_data.dta")
data$heat <- data$heat2
data$heat[is.na(data$heat2)] <- data$heat4[is.na(data$heat2)]
levels(data$heat)[3] <- NA
data$dem <- as.numeric(data$heat) - 1
data$married <- data$marital == "married"
state_abbs <- state.abb
state_abbs <- append(state_abbs, "DC", after = 7)
data$state_abbs <- data$state
levels(data$state_abbs) <- state_abbs
ok <- !is.na(data$dem) & !is.na(data$married) & !is.na(data$state) &
  data$state != "alaska" & data$state != "hawaii"
clean_data <- droplevels(subset(data, ok))
by_state_married <- group_by(clean_data, state, married, state_abbs)
counts_data <- summarise(by_state_married, count = n(), n_dem = sum(dem), percent = mean(dem)*100)
n_states <- nlevels(counts_data$state)
state <- as.numeric(counts_data$state)
married <- as.numeric(counts_data$married)
n_dem <- counts_data$n_dem
n_data <- dim(counts_data)[1]
count <- counts_data$count
pct_married <- rep(NA, n_states)
for(i in 1:n_states){
  pct_married[i] <- count[state==i & married==1]/
    (count[state==i & married==1]+count[state==i & married==0])
}
```

The model written out in Stan:

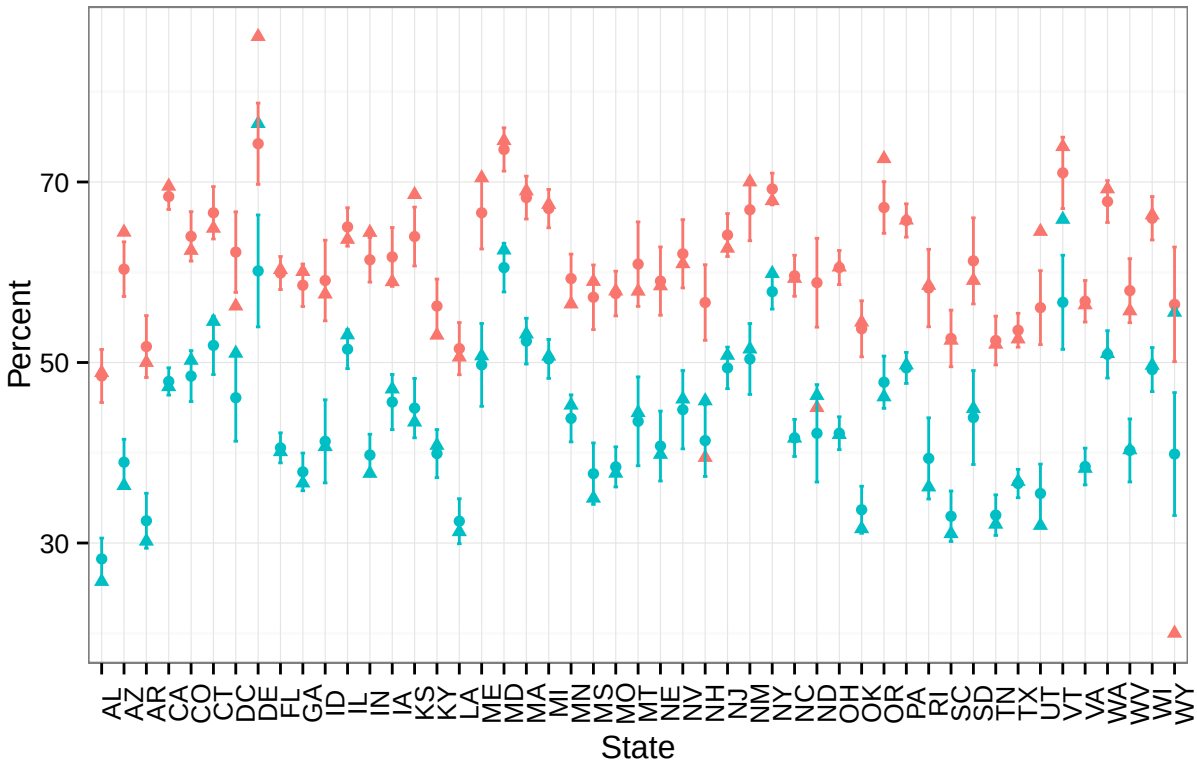
```
data {
  int<lower=0> n_data;
  int<lower=0> n_states;
  int<lower=0> state[n_data];
  int<lower=0> count[n_data];
  int<lower=0, upper=1> married[n_data];
  int<lower=0> n_dem[n_data];
  real<lower=0> pct_married[n_states];
}
parameters {
  real beta_I;
  real beta_M;
  vector[n_states] beta_state;
  vector[n_states] beta_state_married;
  real<lower=0> sigma_I;
  real<lower=0> sigma_M;
  real<lower=0> sigma_state;
  real<lower=0> sigma_state_married;
}
transformed parameters {
  vector[n_data] p;
  for (i in 1:n_data)
    p[i] <- beta_I + beta_M * married[i] + beta_state[state[i]] + beta_state_married[state[i]] * marrie
}
model {
  beta_I ~ normal(0, sigma_I);
  beta_M ~ normal(0, sigma_M);
  beta_state ~ normal(0, sigma_state);
  beta_state_married ~ normal(0, sigma_state_married);
  sigma_I ~ normal(0, 5);
  sigma_M ~ normal(0, 5);
  sigma_state ~ normal(0, 5);
  sigma_state_married ~ normal(0, 5);
  for (j in 1:n_data)
    n_dem[j] ~ binomial_logit(count[j], p[j]);
}
generated quantities {
  vector[n_data] p_pred;
  vector[n_states] p_mar;
  vector[n_states] p_not;
  vector[n_states] obama_pct;

  for (i in 1:n_data)
    p_pred[i] <- inv_logit(p[i]) * 100;
  for (i in 1:n_states){
    p_mar[i] <- beta_I + beta_M + beta_state[state[i]] + beta_state_married[state[i]];
    p_not[i] <- beta_I + beta_state[state[i]];
    obama_pct[i] <- (pct_married[i] * inv_logit(p_mar[i]) + (1 - pct_married[i]) * inv_logit(p_not[i]))
  }
}
```

Fitting the model:

```
fit <- stan("Final.stan")
monitor <- as.data.frame(monitor(fit))
```

We first plot a comparison with the true marriage gap from the survey dataset. Percentages of democratic married voters by state are in blue while non-married are in red. The ground truth from the survey dataset is in the same color but with triangles instead of circles

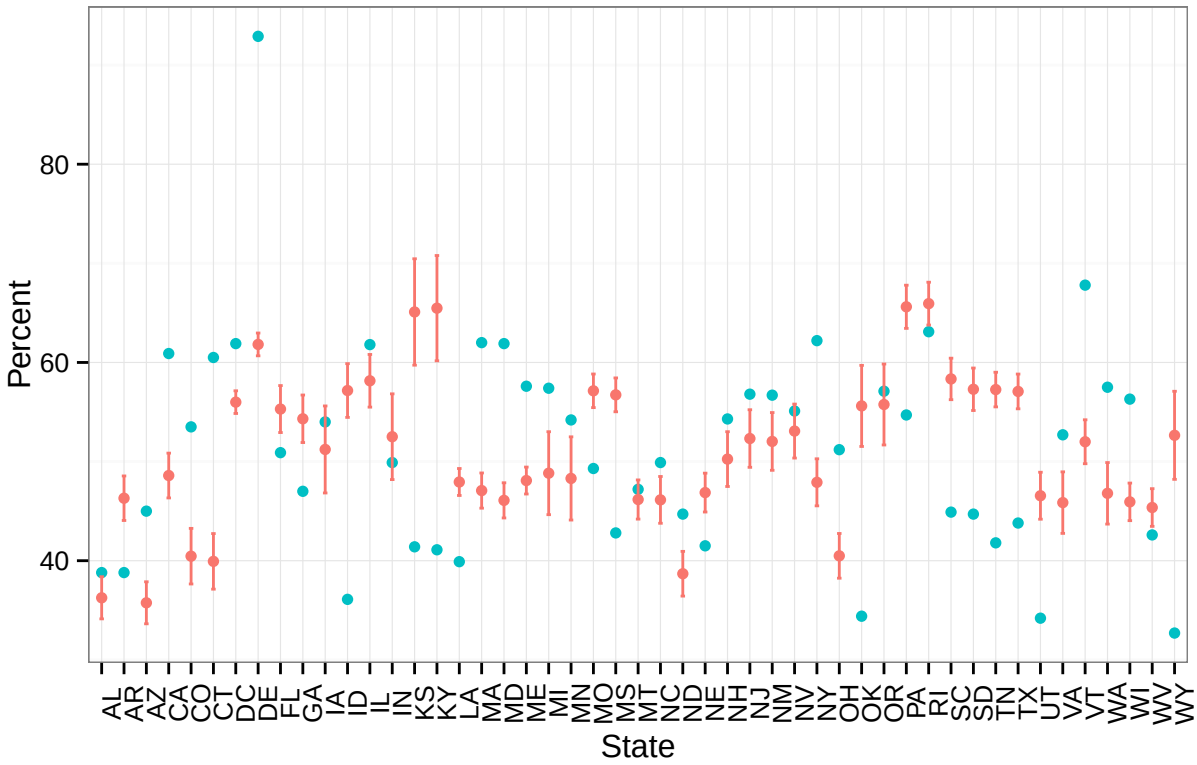
```
y_pred <- data.frame(state_abbs = counts_data$state_abbs, married = counts_data$married)
p_mean <- rep(NA, length(state))
p_sd <- rep(NA, length(state))
for(i in 1:length(state)){
  p_mean[i] <- monitor[paste("p_pred[", i, "]", sep = ""), "mean"]
  p_sd[i] <- monitor[paste("p_pred[", i, "]", sep = ""), "sd"]
}
y_pred$percent <- p_mean
y_pred$sd <- p_sd
y_pred$ground_truth <- rep(FALSE, length(y_pred$percent))
plot_data <- y_pred
plot_data$sd <- rep(NA, length(plot_data$sd))
plot_data$percent <- counts_data$percent
plot_data$ground_truth <- rep(TRUE, length(plot_data$percent))
plot_data <- rbind(y_pred, plot_data)

p <- ggplot(data=plot_data, aes(x=state_abbs, y=percent, group=interaction(married, ground_truth), colo
p + geom_point() +
  geom_errorbar(width=0.2) +
  xlab("State") +
  ylab("Percent") +
  ggtitle("") +
  theme_bw() +
  theme(legend.position="none", axis.text.x = element_text(angle=90, vjust=1))
```



Here, we project the actual democratic vote just from marriage and state parameters estimated with our model. We assume that the true proportions of married to unmarried voters are the same as in the survey for each state. The estimates are in red while the actual vote percentages are in blue. While the model could recover the marriage gap from the survey well, it seems unable to capture the real voting percentages by state.

```
results <- read.csv("~/Documents/BDA/Final/2008ElectionResult.csv")
results$state_abbs <- state_abbs
ok <- results$state != "Alaska" & results$state != "Hawaii"
clean_results<- droplevels(subset(results, ok))
obama_plot <- clean_results[, c("state_abbs", "vote_Obama_pct")]
obama_plot$sd <- rep(NA, length(obama_plot$state_abbs))
obama_plot$ground_truth <- rep(TRUE, length(obama_plot$state_abbs))
pred_mean <- rep(NA, length(obama_plot$state_abbs))
pred_sd <- rep(NA, length(obama_plot$state_abbs))
for(i in 1:length(clean_results$state_abbs)){
  pred_mean[i] <- monitor[paste("obama_pct[", i, "]", sep = ""), "mean"]
  pred_sd[i] <- monitor[paste("obama_pct[", i, "]", sep = ""), "sd"]
}
obama_preds <- data.frame(state_abbs=obama_plot$state_abbs, vote_Obama_pct=pred_mean, sd=pred_sd)
obama_preds$ground_truth <- rep(FALSE, length(obama_preds$state_abbs))
obama_plot <- rbind(obama_plot, obama_preds)

p <- ggplot(data=obama_plot, aes(x=state_abbs, y=vote_Obama_pct, group=ground_truth, color=ground_truth
p + geom_point() +
  geom_errorbar(width=0.2) +
  xlab("State") +
  ylab("Percent") +
  ggtitle("") +
  theme_bw() +
```

```
  theme(legend.position="none", axis.text.x = element_text(angle=90, vjust=1))
```



Here we estimate the model without individual state marriage gaps:

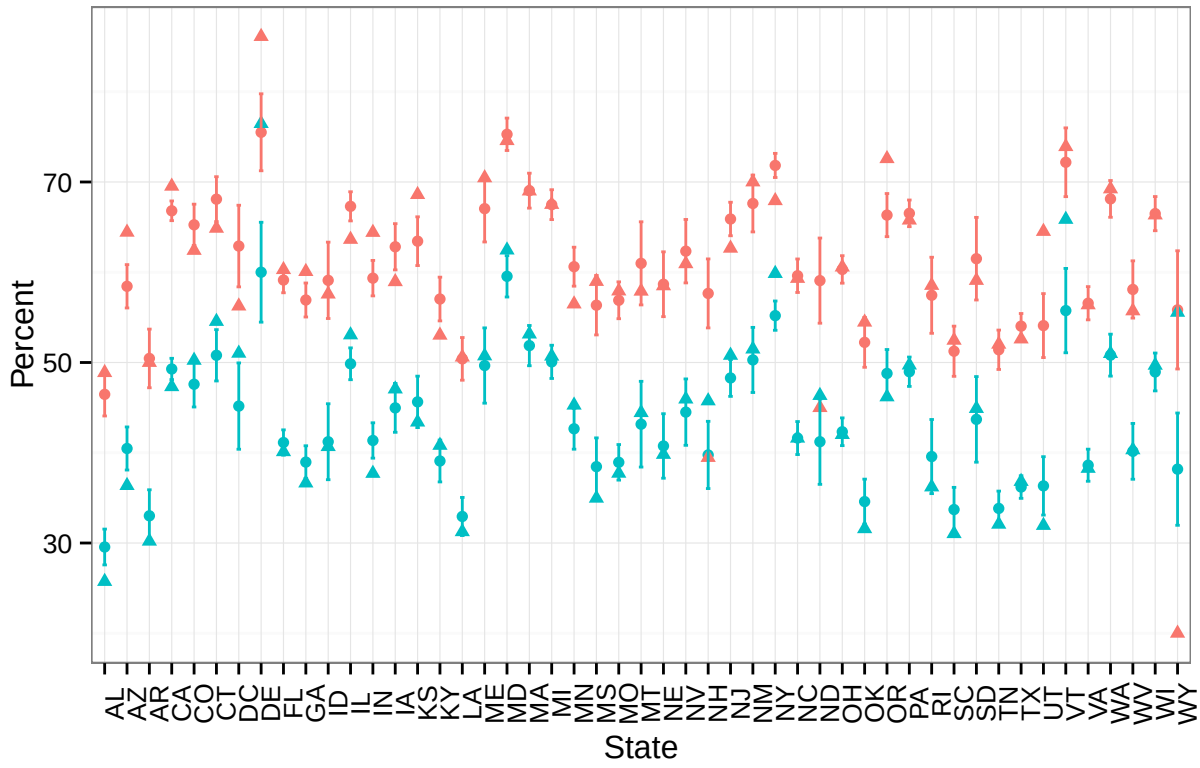
```
data {
  int<lower=0> n_data;
  int<lower=0> n_states;
  int<lower=0> state[n_data];
  int<lower=0> count[n_data];
  int<lower=0,upper=1> married[n_data];
  int<lower=0> n_dem[n_data];
  real<lower=0> pct_married[n_states];
}
parameters {
  real beta_I;
  real beta_M;
  vector[n_states] beta_state;
  real<lower=0> sigma_I;
  real<lower=0> sigma_M;
  real<lower=0> sigma_state;
}
transformed parameters {
  vector[n_data] p;
  for (i in 1:n_data)
    p[i] <- beta_I + beta_M * married[i] + beta_state[state[i]];
}
model {
  beta_I ~ normal(0, sigma_I);
  beta_M ~ normal(0, sigma_M);
```

```
  beta_state ~ normal(0, sigma_state);
  sigma_I ~ normal(0, 5);
  sigma_M ~ normal(0, 5);
  sigma_state ~ normal(0, 5);
  for (j in 1:n_data)
    n_dem[j] ~ binomial_logit(count[j], p[j]);
}
generated quantities {
  vector[n_data] p_pred;
  vector[n_states] p_mar;
  vector[n_states] p_not;
  vector[n_states] obama_pct;

  for (i in 1:n_data)
    p_pred[i] <- inv_logit(p[i]) * 100;
  for (i in 1:n_states){
    p_mar[i] <- beta_I + beta_M + beta_state[state[i]];
    p_not[i] <- beta_I + beta_state[state[i]];
    obama_pct[i] <- (pct_married[i] * inv_logit(p_mar[i]) + (1 - pct_married[i]) * inv_logit(p_not[i]))
  }
}
```

```
fit_nostate <- stan("final_nostate.stan")
monitor_nostate <- as.data.frame(monitor(fit_nostate))
```

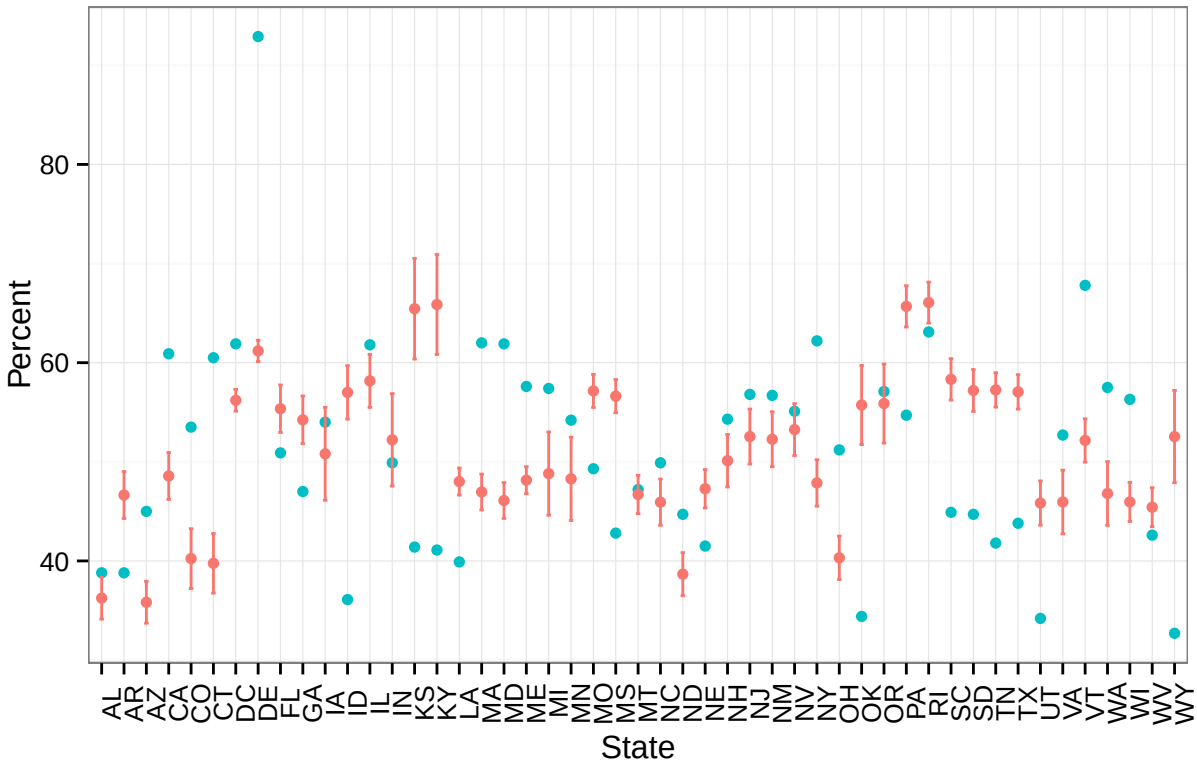
And here is the same plot as before. Despite having significantly fewer parameters, this model seems to fit almost as well since most of the marriage gaps were of similar size.

```
y_pred <- data.frame(state_abbs = counts_data$state_abbs, married = counts_data$married)
p_mean <- rep(NA, length(state))
p_sd <- rep(NA, length(state))
for(i in 1:length(state)){
  p_mean[i] <- monitor_nostate[paste("p_pred[", i, "]", sep = ""), "mean"]
  p_sd[i] <- monitor_nostate[paste("p_pred[", i, "]", sep = ""), "sd"]
}
y_pred$percent <- p_mean
y_pred$sd <- p_sd
y_pred$ground_truth <- rep(FALSE, length(y_pred$percent))
plot_data <- y_pred
plot_data$sd <- rep(NA, length(plot_data$sd))
plot_data$percent <- counts_data$percent
plot_data$ground_truth <- rep(TRUE, length(plot_data$percent))
plot_data <- rbind(y_pred, plot_data)

p <- ggplot(data=plot_data, aes(x=state_abbs, y=percent, group=interaction(married, ground_truth), colo
p + geom_point() +
  geom_errorbar(width=0.2) +
  xlab("State") +
  ylab("Percent") +
  ggtitle("") +
  theme_bw() +
  theme(legend.position="none", axis.text.x = element_text(angle=90, vjust=1))
```



And here is the same plot against the actual percentages who voted for Obama. Again, the model fails to capture much.

```
obama_plot <- clean_results[, c("state_abbs", "vote_Obama_pct")]
obama_plot$sd <- rep(NA, length(obama_plot$state_abbs))
obama_plot$ground_truth <- rep(TRUE, length(obama_plot$state_abbs))
pred_mean <- rep(NA, length(obama_plot$state_abbs))
pred_sd <- rep(NA, length(obama_plot$state_abbs))
for(i in 1:length(clean_results$state_abbs)){
  pred_mean[i] <- monitor_nostate[paste("obama_pct[", i, "]", sep = ""), "mean"]
  pred_sd[i] <- monitor_nostate[paste("obama_pct[", i, "]", sep = ""), "sd"]
}
obama_preds <- data.frame(state_abbs=obama_plot$state_abbs, vote_Obama_pct=pred_mean, sd=pred_sd)
obama_preds$ground_truth <- rep(FALSE, length(obama_preds$state_abbs))
obama_plot <- rbind(obama_plot, obama_preds)

p <- ggplot(data=obama_plot, aes(x=state_abbs, y=vote_Obama_pct, group=ground_truth, color=ground_truth
p + geom_point() +
  geom_errorbar(width=0.2) +
  xlab("State") +
  ylab("Percent") +
  ggtitle("") +
  theme_bw() +
  theme(legend.position="none", axis.text.x = element_text(angle=90, vjust=1))
```



(f) The second model is a special case of the first where the prior on the indivual state marriage gap parameters has 0 variance (i.e. complete pooling). The first model allows for some partial pooling: it leans on other states to estimate the marriage gap when data is sparse but allows the state marriage gaps to vary when the data suggests that they should. However, it seems like they did not vary very much and the second model, at least in these predictive checks, fits the data almost as well as the first model. This suggests that the marraige gap is fairly constan across states.