Final Exam

Ziyue Wang

Problem 1

(a)

Define the number of blue fish in a size-100 simply random sample as $y \in \{0, \ldots, 100\}$ and the number of green fish is $100 - y$ which is completely determined by y. Let y_1 be the first batch and let y_2 be the second batch so $y = (y_1, y_2)$. Define I to be the inclusion variable that means whether the y_i is observed or not. In our case $I = (1, 0)$. Define the number of tagged fish in the second or later capture of 100 fish to be $z \in \{0, \ldots, 100\}$. The complete data model given (N_b, N_g) and thus given $N := N_b + N_g$ is

$$p(y, z, I, |N_b, N_g) = p(y|N_b, N_g)p(z|N_b, N_g)p(I),$$

where

$$p(y|N_b, N_g) = \begin{cases} \prod_{i=1}^{2} \frac{\binom{N_b}{y_i}\binom{N_g}{100-y_i}}{\binom{N}{100}} & \text{if } y_i \leq N_b \text{ and } 100 - y_i \leq N_g \\ 0 & \text{otherwise} \end{cases}$$

and

$$p(z|N_b, N_g) = \begin{cases} \frac{\binom{100}{z}\binom{N-100}{100-z}}{\binom{N}{100}} & \text{if } z \leq 100 \text{ and } 100 - z \leq N - 100 \\ 0 & \text{otherwise} \end{cases}$$

For the missing mechanism, since knocking over the fish bucket is completely out of our control and is obviously not determined by the number of fish in the pond, we would assume that this knocking over is completely at random and that $p(I)$ is some unknown distribution on $\{1, 0\}^2$ that does not need to be estimated. Please note that only y can be missing and we always have complete data for z.

For the model for observed data y_{obs}, since we assume that $p(I)$ does not depend on y or N_b, N_g we have

$$\begin{aligned} p(y_{\text{obs}}, z, I, |N_b, N_g) &= p(z|N_b, N_g) \int p(y_{\text{obs}}, y_{\text{mis}}|N_b, N_g)p(I)\, dy_{\text{mis}} \\ &= p(z|N_b, N_g)p(I) \int p(y_{\text{obs}}, y_{\text{mis}}|N_b, N_g)\, dy_{\text{mis}} \\ &= p(z|N_b, N_g)p(I)p(y_{\text{obs}}|N_b, N_g). \end{aligned}$$

So the missing of the second bucket fish does not harm the estimation of (N_b, N_g).

(b)

We will use an weakly informative prior for (N_b, N_g) and assuming that N_b is independent of N_g. It will depends on the observed data $y_{\text{obs}} := y_1$ and z but it is still a valid prior not a conditional distribution.

We use Poisson distribution for the priors, however it is assigned to N_b and N_g with a mean shift such that it is consistent with what we observed in y_1 and z:

$$p(N_b) = \text{Poisson}(N_b - y_1 - (200 - z) \mid \lambda_b),$$

and

$$p(N_g) = \text{Poisson}(N_g - (100 - y_1) \mid \lambda_g).$$

This prior guarantees that (N_b, N_g) matches the reality and it is reasonable to use observed data to define prior. If you capture 100 fish and observe that y_1 of them are blue, then release the fish to the pond and capture 100 fish again and observe that z of them are from the first batch, this immediately implies that the total number of fish in the pond is greater than $200 - z$, total number of blue fish is greater than y_1 and total number of green fish is greater than $100 - y_1$. To reflect these facts in the prior, we let N_b always greater than $y_1 + (200 - z)$ and N_g always greater than $(100 - y_1)$. Please note that we give all the z numbers to N_b, this is a relatively compromised way to enforce that $N_b + N_g \geq 200 - z$ but should not affect the estimation of N_b and N_g in the end.

After setting the deterministic lower bound we make the fluctuation follows a Poisson distribution with parameter λ_b and λ_g respectively.

(c)

The joint posterior given $y_{\text{obs}} := y_1$, z and I is

$$\begin{aligned}
p(N_b, N_g | y_{\text{obs}}, z, I) &\propto p(N_b)p(N_g)p(y_{\text{obs}}, z, I, | N_b, N_g) \\
&= p(N_b)p(N_g)p(I)p(z|N_b, N_g)p(y_{\text{obs}}|N_b, N_g) \\
&\propto p(N_b)p(N_g)p(z|N_b, N_g)p(y_{\text{obs}}|N_b, N_g) \qquad \text{Since } p(I) \text{ is constant for } N_b \text{ and } N_g \\
&= \frac{\lambda_b^{N_b - y_1 - (200-z)} e^{-\lambda_b}}{(N_b - y_1 - (200 - z))!} \frac{\lambda_g^{N_b - (100 - y_1)} e^{-\lambda_g}}{(N_b - (100 - y_1))!} \frac{\binom{100}{z}\binom{N-100}{100-z}}{\binom{N}{100}} \frac{\binom{N_b}{y_{\text{obs}}}\binom{N_g}{100-y_{\text{obs}}}}{\binom{N}{100}}
\end{aligned}$$

Please note that in above expression $y_{\text{obs}} := y_1$ so we equivalently used both notation in the formula. If in the future more than y_1 is observed then for the prior needs to be redefined to use the information in the new observed data.

Problem 2

We know that for $a > 0$

$$p(a) = p(\log a)\left|\frac{1}{a}\right| = \frac{p(\log a)}{a},$$

and $p(a) = 0$ otherwise. However for convenience we are going to sample from $\log a$ and $\log b$ and all the following calculation are on $\log a$ and $\log b$.

(a) Log posterior and the derivatives

Denote $\log a$ by α and $\log b$ by β. The log joint posterior is

$$\begin{aligned}\log p(\alpha, \beta|x, y) &\propto \log p(\alpha) + \log p(\alpha) + \log p(y|\alpha, \beta, x)\\ &= -\log(2\pi) - \log(\log 2) - \log(\log 10) + \log p(\alpha) + \log p(\beta) + \log p(y|\alpha, \beta, x)\\ &= C - \frac{(\alpha - \log 5)^2}{2(\log 2)^2} - \frac{(\beta - \log 10)^2}{2(\log 10)^2} + \sum_{i=1}^{n} \{\exp(\alpha)(\beta + \log x_i) - \log(\Gamma(\exp(\alpha))) + (\exp(\alpha) - 1)\log y_i - \exp(\beta)\end{aligned}$$

Denote the log posterior as ℓ, the partial derivatives are

$$\frac{d\ell}{d\alpha} = \frac{d\ell}{da} \times \frac{da}{d\alpha} = \exp(\alpha)\left\{-\frac{\log(a/5)}{a(\log 2)^2} + \sum_{i=1}^{n}\{\log(bx_i) - \psi(a) + logy_i\}\right\},$$

where we use the logrithme derivative of Gamma function $\psi(x)$ which has a built-in function in R, for the formula see https://en.wikipedia.org/wiki/Digamma_function and page 3 of http://scipp.ucsc.edu/~haber/archives/physics116A10/psifun_10.pdf.

$$\frac{d\ell}{d\beta} = \frac{d\ell}{db} \times \frac{db}{d\beta} = \exp(\beta)\left\{-\frac{\log(10b)}{b(\log 10)^2} - \frac{1}{b} + \sum_{i=1}^{n}\left\{\frac{a}{b} - x_i y_i\right\}\right\}.$$

Now we are ready to use HMC.

(b) Sample from the posterior

First we simulate a size-100 dataset including x and y.

```r
set.seed(2021)

n <- 100

x <- exp(rnorm(n))
a <- 2
b <- .3

y <- rgamma(n, shape=a, rate=b*x)
```

Next we use HMC to sample from the posterior. Let's define the gradient and the log posterior in R.

```r
grad <- function(a, b) {
  # Input variable a is alpha not the parameter a. Same for variable b.
  a <- exp(a); b <- exp(b)
  grad_a <- a*(-log(a/5)/a/(log(2))^2 - 1/a - n*digamma(a) + sum(log(b*x) + log(y)))
  grad_b <- b*(-log(10*b)/b/(log(10))^2 - 1/b + n*a/b - sum(x*y))

  c(grad_a, grad_b)
}
```

```r
logp <- function(a, b) {
  # Input variable a is alpha not the parameter a. Same for variable b.
  a <- exp(a); b <- exp(b)
  -log(2*pi) - log(log(2)) - log(log(10)) - log(a) - log(b) -
    (log(a/5)^2)/(2*(log(2))^2) -
    (log(10*b)^2)/(2*(log(10))^2) -
    n*lgamma(a) +
    sum(a*log(b*x) + (a-1)*log(y) - b*x*y)
}
```

Next we check the gradient with the finite-difference approximation at $(1, 2)$:

```r
grad(1, -2)
```

```
## [1] -296.5505  170.9224
```

```r
c((logp(1+1e-5, -2) - logp(1, -2))/1e-5, (logp(1, -2+1e-5) - logp(1, -2))/1e-5)
```

```
## [1] -296.5537  170.9219
```

Now we are sure that the gradient is correct and we are ready to implement HMC.

```r
hmc <- function(iter, init) {
  Theta <- matrix(0, nrow=iter+1, ncol=2)
  Theta[1, ] <- init

  M <- matrix(c(5, 0, 0, 5), 2, 2)
  eps <- .008
  L <- 10
  Accept <- rep(NA, iter)

  for (i in 1:iter) {
  # Leapfrog
  Theta0 <- Theta[i, ]
  phi <- rnorm(2, mean=0, sd=diag(M))
  phi0 <- phi
  phi <- phi + .5*eps*grad(Theta0[1], Theta0[2])
  Theta0 <- Theta0 + eps*t(phi)%*%solve(M)
  for (j in 1:(L-1)) {
    phi <- phi + eps*grad(Theta0[1], Theta0[2])
    Theta0 <- Theta0 + eps*t(phi)%*%solve(M)
  }
  phi <- phi + .5*eps*grad(Theta0[1], Theta0[2])

  #Accept-reject sampling
  log_r <- logp(Theta0[1], Theta0[2]) - logp(Theta[i, 1], Theta[i, 2]) +
    sum(dnorm(phi, sd=diag(M), log=T)) - sum(dnorm(phi0, sd=diag(M), log=T))
  Accept[i] <- (log(runif(1)) < log_r)
  Theta[i+1, ] <- ifelse(rep(Accept[i], 2),
                         Theta0,
                         Theta[i, ])
  }
  return(list(chain=Theta, accept=Accept))
}

cat(paste("Acceptance rate is" ,mean(hmc(1e4, c(0, 1))$accept)))
```

```
## Acceptance rate is 0.6312
```

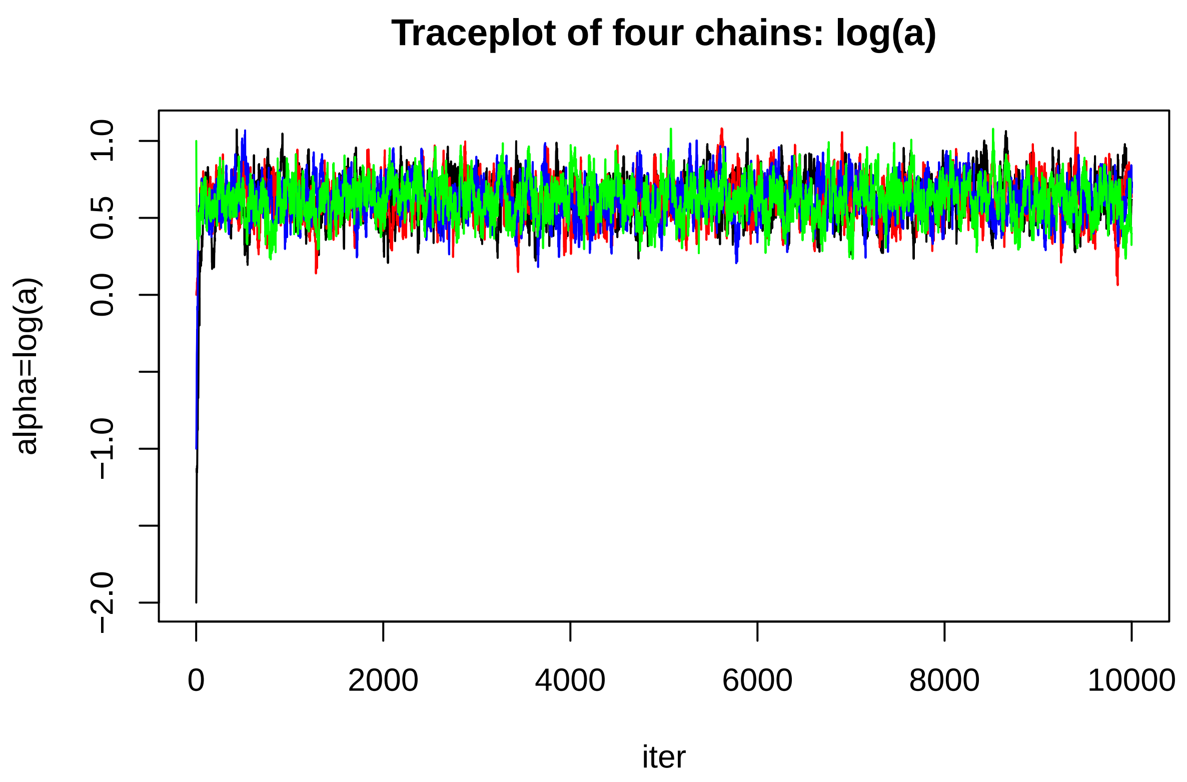
We use $L = 10$ and tune the M and ϵ such that acceptance rate is roughly 2/3. In this case $M = 5I_2$ and $\epsilon = 0.008$. Next we check the convergence of our HMC program.

(c) Convergence diagnostics

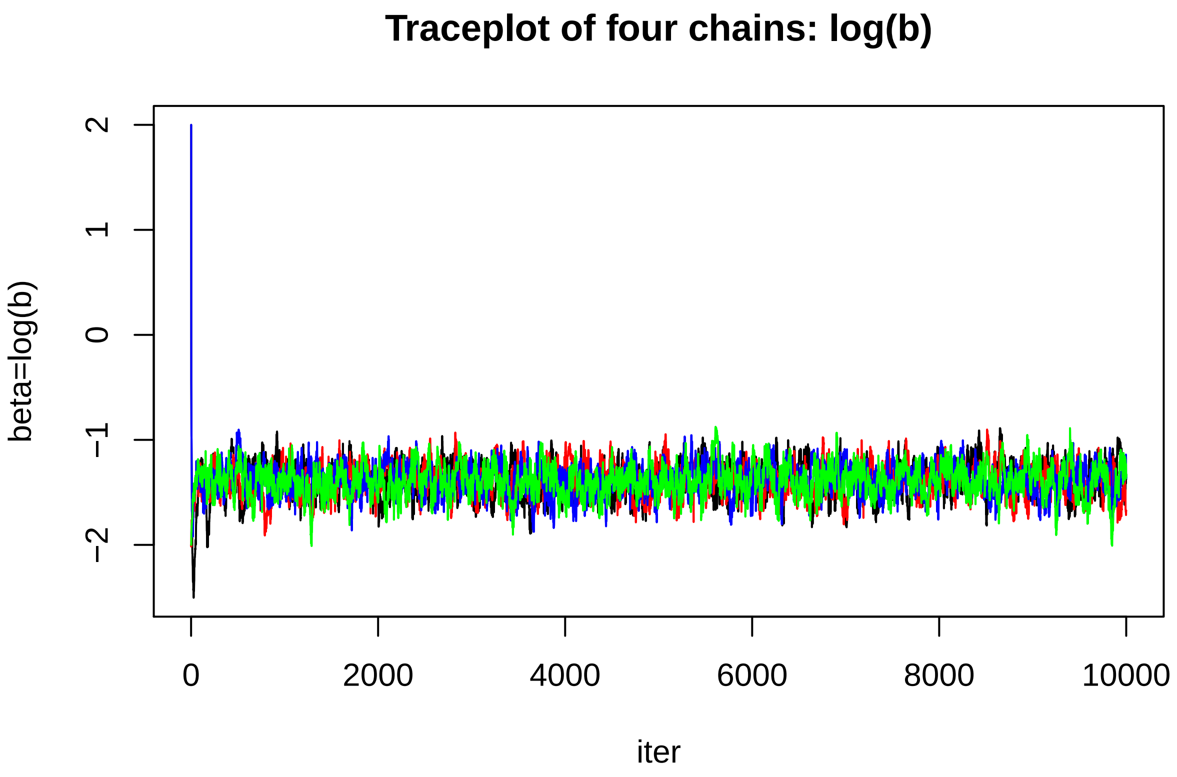
We run four chains with dispersed initial values and check the mixing status.

```
chain1 <- hmc(1e4, c(0, -2))$chain
chain2 <- hmc(1e4, c(1, -2))$chain
chain3 <- hmc(1e4, c(-1, 2))$chain
chain4 <- hmc(1e4, c(-2, 2))$chain

chains_loga <- cbind(chain1[, 1], chain2[, 1], chain3[, 1], chain4[, 1])
chains_logb <- cbind(chain1[, 2], chain2[, 2], chain3[, 2], chain4[, 2])
```

```
plot(chains_loga[, 4], type="l",
     main="Traceplot of four chains: log(a)",
     xlab="iter",
     ylab="alpha=log(a)")
lines(chains_loga[, 1], col="red")
lines(chains_loga[, 3], col="blue")
lines(chains_loga[, 2], col="green")
```



```
plot(chains_logb[, 4], type="l",
     main="Traceplot of four chains: log(b)",
     xlab="iter",
     ylab="beta=log(b)")
lines(chains_logb[, 2], col="red")
lines(chains_logb[, 3], col="blue")
lines(chains_logb[, 1], col="green")
```



The traceplot looks good. Although the initial points are far from each other, the four chains eventually mixed together and becomes indistinguishable. From this we are confident that the chains approximately converged. However it does no harm to further check some statistics like the PSRF and effective sample size to numerically access the convergence.

```
burn_in <- function(chain) {
  tail(chain, -.5*length(chain))
}

 my_diff <- function(t, x) {
    mean(diff(x, lag = t)^2, na.rm = TRUE)
 }

n_eff <- function(chains) {
  require(purrr)
  m <- ncol(chains)
  n <- nrow(chains)
  # Within-chain var
  W <- 1/m*sum(apply(chains, 2, var))

  means <- colMeans(chains)
  B <- n * var(means)

  V_plus <- (n-1)/n * W + 1/n * B
  Rhat <- sqrt(V_plus/W)

  V <- sapply(seq_len(nrow(chains) - 1), my_diff, x=chains[, 1]) + sapply(seq_len(nrow(chains) - 1), my_

  V <- V/m
  rho <- head_while(1 - V / (2*V_plus), ~. >0)

  n_hat_eff <- m*n / (1 + 2*sum(rho))
  return(list(n_hat_eff, Rhat))
}

chains_loga_burn <- apply(chains_loga, 2, burn_in)
res <- n_eff(chains_loga_burn)
```

```
## Loading required package: purrr
```

```
cat(paste("log(a): The PSRF is", res[[2]]))
```

```
## log(a): The PSRF is 1.00172068524053
```

```
cat(paste("\nThe estimated effective sample size is", res[[1]]))
```

```
##
## The estimated effective sample size is 616.64218064808
```

```
chains_logb_burn <- apply(chains_logb, 2, burn_in)
res <- n_eff(chains_logb_burn)
cat(paste("\n\nlog(b): The PSRF is", res[[2]]))
```

```
##
##
## log(b): The PSRF is 1.00293577326083
```

```
cat(paste("\nThe estimated effective sample size is", res[[1]]))
```

```
##
## The estimated effective sample size is 592.733958242492
```

Looks like the PSRF is close enough to 1, and the estimated effective sample size is not as large as expected but is not bad. Combining these results with what we observed from the traceplot we conclude that the chains obtained from our HMC program are approximately converged.

(d) Posterior inference

For the part (d) we will use the first chain in part (c) to do the posterior inference. Please note that our chains give sample of $\log(a)$ and $\log(b)$ so before we proceed we need to calculate their exponent.

Below is the summary of quantiles and mean of posterior samples.

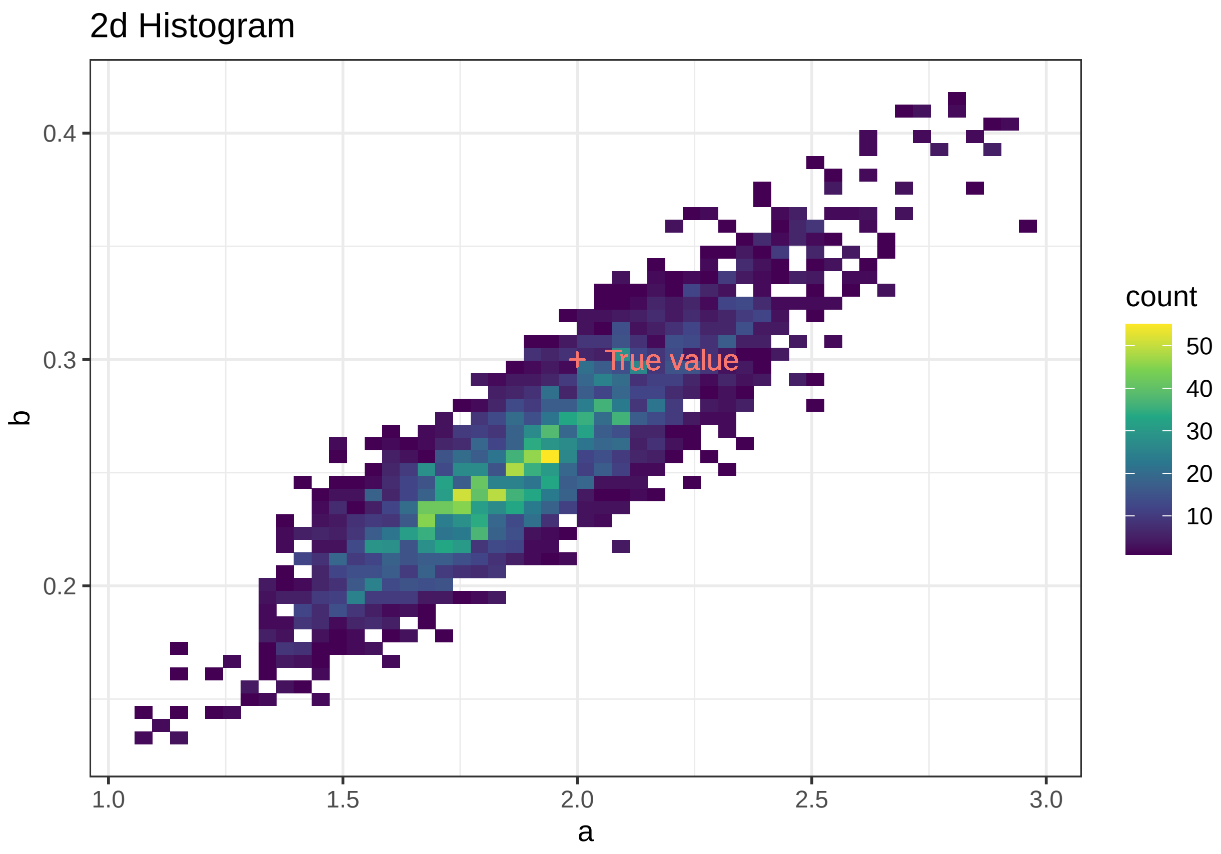
```
chain1_exp <- exp(chain1)
colnames(chain1_exp) <- c("a", "b")
summary(chain1_exp)
```

```
##         a                b
##  Min.   :1.000   Min.   :0.1338
##  1st Qu.:1.699   1st Qu.:0.2263
##  Median :1.856   Median :0.2490
##  Mean   :1.869   Mean   :0.2518
##  3rd Qu.:2.019   3rd Qu.:0.2748
##  Max.   :2.951   Max.   :0.4168
```

So the posterior mean of a is 1.869 and that of b is 0.2518. The mode is about the same. Both are close to the true value $(2, 0.3)$ but there are gaps. As the sample size of x and y increases the difference in posterior mean and true value will become smaller and smaller.

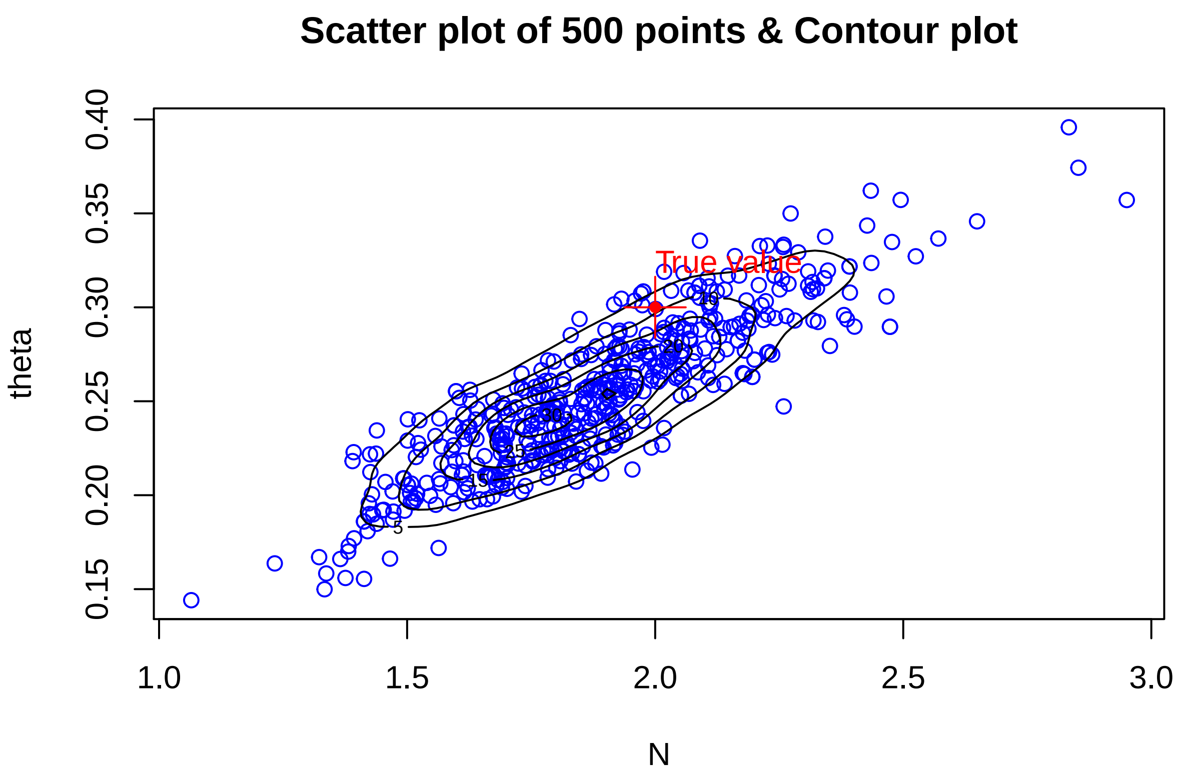
Let's take a look at a 2d Histogram. It looks better comparing to the scatterplot and contour plot.

```r
burn_in <- function(chain) {
  tail(chain, -.5*dim(chain)[1])
}
library(ggplot2)
res <- burn_in(chain1_exp) # remove burn-in
plt_data <- as.data.frame(res)
colnames(plt_data) <- c('a', 'b')
ggplot(plt_data, aes(x=a, y=b) ) +
  geom_bin2d(bins = 50) +
  scale_fill_continuous(type = "viridis") +
  theme_bw() + ggtitle('2d Histogram') +
  geom_point(aes(x=2, y=.3, col='red'), shape=3) +
  geom_text(aes(x=2, y=.3, label="True value", col="red"), hjust=-.2) +
  guides(color = FALSE)
```



Scatterplot and contour plot It is not reasonable to do a 5000 size scatterplot so I just randomly picked 500 points from the chain to draw the scatterplot. The coutour plot is using the entire chain.

```
library(MASS)
ind <- sample(nrow(res), 500)
plot(res[ind, 1], res[ind, 2], col='blue',
     xlab='N', ylab='theta', main='Scatter plot of 500 points & Contour plot')
contour(kde2d(res[, 1], res[, 2], n=100), add=TRUE)
points(x=2, y=0.3, pch=3, col="red", cex=3)
points(x=2, y=0.3, pch=20, col="red")
text(x=2, y=0.3, "True value", col="red", adj=c(0, -1.5))
```



From the plot it is again confirmed that the posterior mode is close to the true value, but there is an error which is reasonable since we only have 100 y data points.

Problem 3

Let's first take a look at the data.

```
dat <- read.csv("GSS2006_abridged.csv")

sum(is.na(dat))
```

```
## [1] 4
```

```
sum(is.null(dat))
```

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

```
sum(is.na(dat$AGE))
```

```
## [1] 4
```

```
dat <- dat[-which(is.na(dat$AGE)), ]
```

It seems that there are 4 missing values in AGE variable. We drop those 4 rows and we will explain this later.

(a) (b) and (c)

Let θ be the probability that a person in the population knows someone gay which is equivalent to defining θ to be the percentage of people in the population that knows someone gay. Let x be the covariates and y be a binary variable indicating whether a person knows someone gay.

We first specify the structure of θ. We let

$$\theta = \text{logistic}(\alpha + x^T\beta)$$

where the logistic function is $1/(1 + \exp(-x))$ and (α, β) are parameters that need to be estimated. We assign normal priors to α and each β_j, $j \in \{1, \dots, 6\}$. The mean of the prior is given based on the existing knowledge and the variance is fixed.

By the naive, but never meant to be biased or of discrimination, knowledge obtained by the simple summary of the data, we give variable age, being male, favoring republican negative impact by assigning negative mean on their corresponding $beta_j$ priors. We assign positive mean priors to $beta_j$ of variables being female, being white and favoring democrats. Then we give a negative impact for people of other race and political views.

The full generative model is

$$p(y, \alpha, \beta|x) = p(\alpha)\prod_{j=1}^{6} p(\beta_j)p(y|x, \alpha, \beta),$$

where $p(\alpha)$ and $p(\beta_j)$ are all normal with mean $(-5, -5, -2, 2, 0, 3, -3)$ being a fixed hyperparameter and variance 1. Also, for $p(y|x, \alpha, \beta)$ we set

$$p(y|x, \alpha, \beta) = \text{Bernoulli}(y \,|\, p = \text{logistic}(\alpha + x^T\beta)).$$

Finally, the posterior is

$$P(\alpha, \beta|x, y) \propto \frac{1}{\sqrt{2\pi}} \exp\left(-\frac{(\alpha - \mu_0)^2}{2}\right) \prod_{j=1}^{6} \frac{1}{\sqrt{2\pi}} \exp\left(-\frac{(\beta_j - \mu_j)^2}{2}\right) \qquad \times \prod_{i=1}^{n} \text{logistic}^{y_i}(\alpha + x_i^T\beta)(1 - \text{logistic}(\alpha + x_i^T\beta))^{1-y_i}.$$

This complete our Bayesian logistic regression model with simple regressors. If we want to further improve the model, adding interaction terms and other regressors could help.

Please note that we assume that the missing of variable age is completely at random. Just like in problem 1 the observed likelihood is the same function as the complete likelihood so the posterior remains in the same form. As a result, although we have 4 missing values in age variable, it is safe to discard those 4 rows and proceed the modeling.

(d) Fit the model in Stan

Below is the Stan code used for our Bayesian logistic model with simple linear regressors.

```
stan_code <- "data {
  int<lower=0> N; // sample size
  matrix[N, 6] x; // covariates
  int<lower=0,upper=1> y[N]; // response
}
parameters {
  real alpha; // Intercept
  vector[6] beta;  // Coeff.
}


model {
  alpha ~ normal(-5, 1);
  beta[1] ~ normal(-5, 1);
  beta[2] ~ normal(-2, 1);
  beta[3] ~ normal(2, 1);
  beta[4] ~ normal(0, 1);
  beta[5] ~ normal(3, 1);
  beta[6] ~ normal(-3, 1);
  // Specify priors one by one Looks awkward
  // but I guess there is no better way
  y ~ bernoulli_logit(alpha +  x * beta);
}

generated quantities {
  vector[N] yrep; // replications from posterior predictive distr.
  for (i in 1:N) {
    yrep[i] = bernoulli_rng(inv_logit(alpha + x[i] * beta));
  }
}

"
```

Next we pass the code to Stan compiler using the default setting and get four chains each with 1000 samples after burn-in. The log information generated by Stan is hidden for simplicity of the report.

```
library(rstan)
```

```
## Loading required package: StanHeaders
```

```
## rstan (Version 2.21.1, GitRev: 2e1f913d3ca3)
```

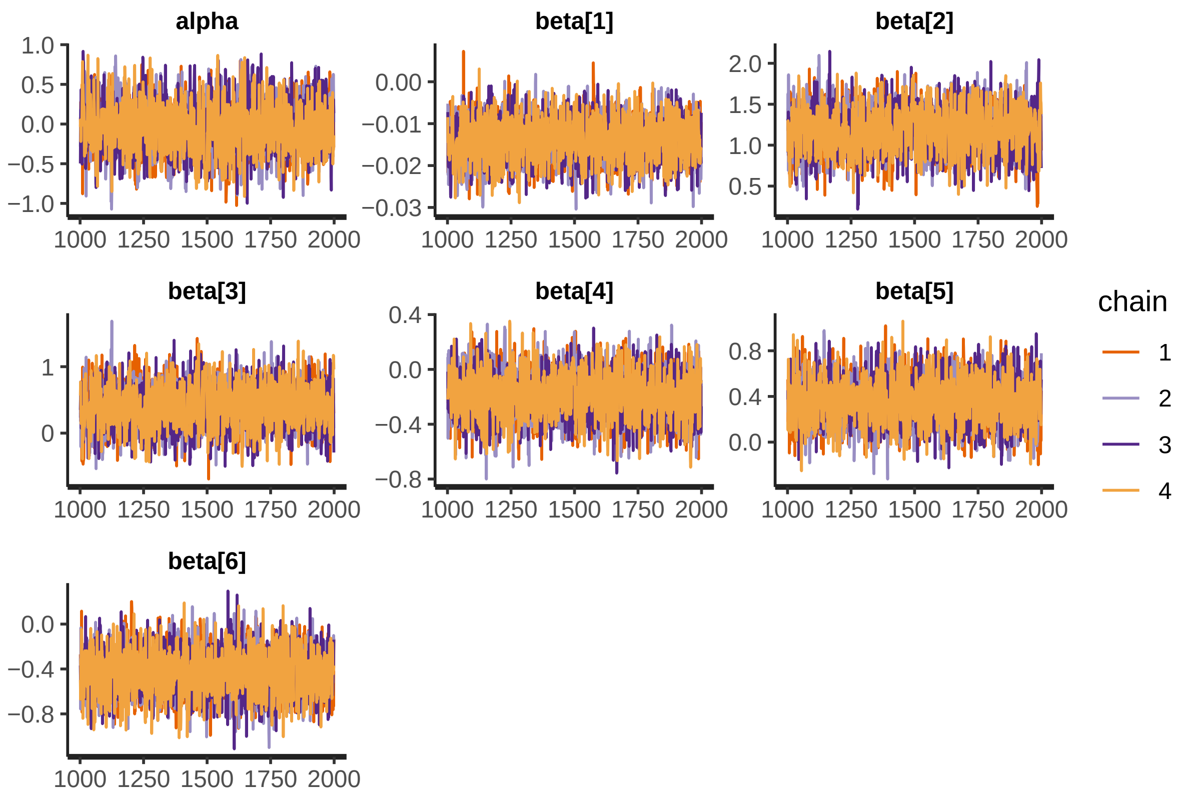
```
## For execution on a local, multicore CPU with excess RAM we recommend calling
## options(mc.cores = parallel::detectCores()).
## To avoid recompilation of unchanged Stan programs, we recommend calling
## rstan_options(auto_write = TRUE)
```

```r
options(mc.cores = parallel::detectCores())

dat <- list(N=dim(dat)[1], x=dat[, -7], y=dat$GAYACQ)
res <- stan(model_code=stan_code, data=dat, seed=2021)
```

```
## Trying to compile a simple C file
```

To make sure that it converged, we first take a look at the traceplot.

```r
traceplot(res, pars=c("alpha", "beta"))
```



Looks like the chains are indistinguishable. Then we check the estimated effective sample size and the PSVF statistics.

```r
summary(res)$summary[1:7, c("n_eff", "Rhat")]
```

```
##              n_eff     Rhat
## alpha   1962.278 1.000759
## beta[1] 3141.804 1.001258
## beta[2] 1983.992 1.001388
## beta[3] 2064.650 1.001658
## beta[4] 3511.022 1.001084
## beta[5] 2944.103 1.000144
## beta[6] 2943.231 1.000176
```

The PSVF is pretty close to 1 and estimated effective sample size is large enough. We are confident that the chains approximately converged.

(e) Posterior predictive checks

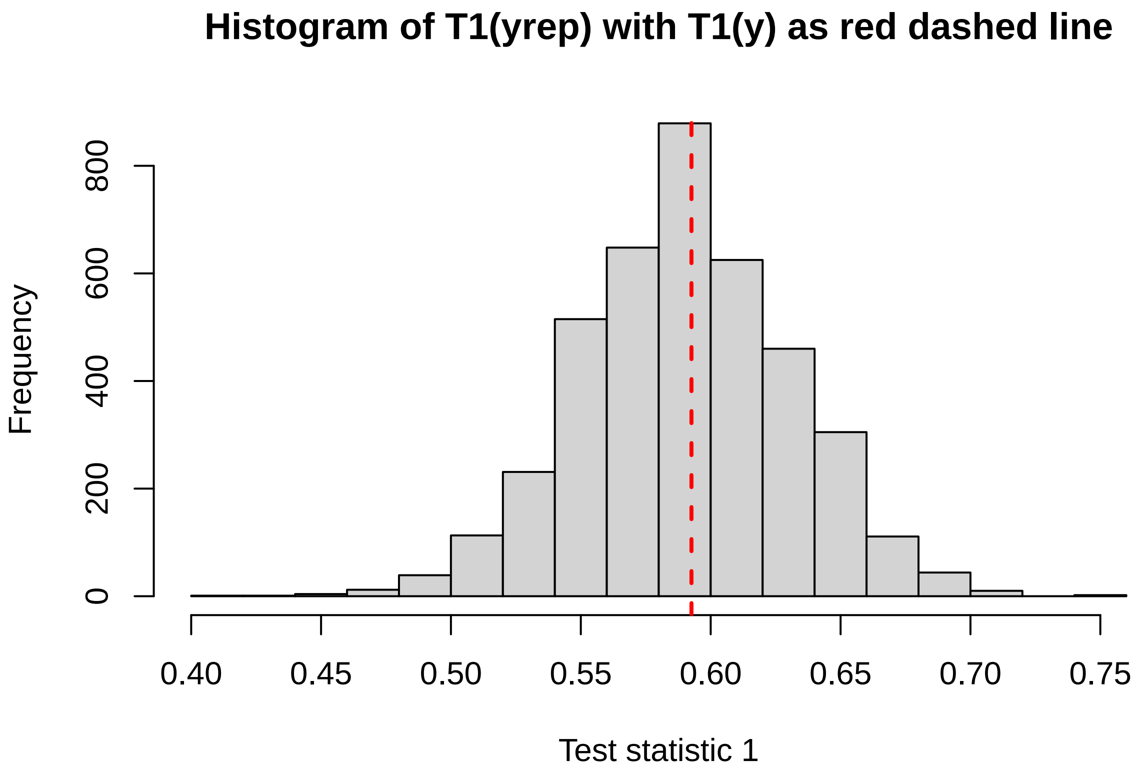
Consider three statistics T_1, T_2, T_3 where T_1 is the percentage of white female knowing someone gay, T_2 is percentage of overall population that knows someone gay and T_3 is the percentage of people of other races who does not support Dem or GOP that knows someone gay.

The replicated y matrix is already generated and is included in our Stan output.

```r
yrep <- extract(res)$yrep

T1 <- function(y) {
  # Percentage of white female knowing someone gay
  mean(y[which(dat$x$RACE_WHITE == 1 & dat$x$SEX == 0)])
}

mean(T1(dat$y) <= apply(yrep, 1, T1))
```

```
## [1] 0.50425
```

```r
hist(apply(yrep, 1, T1),
     xlab = "Test statistic 1",
     main = "Histogram of T1(yrep) with T1(y) as red dashed line",)
abline(v=T1(dat$y), lwd=2, lty=2, col="red")
```

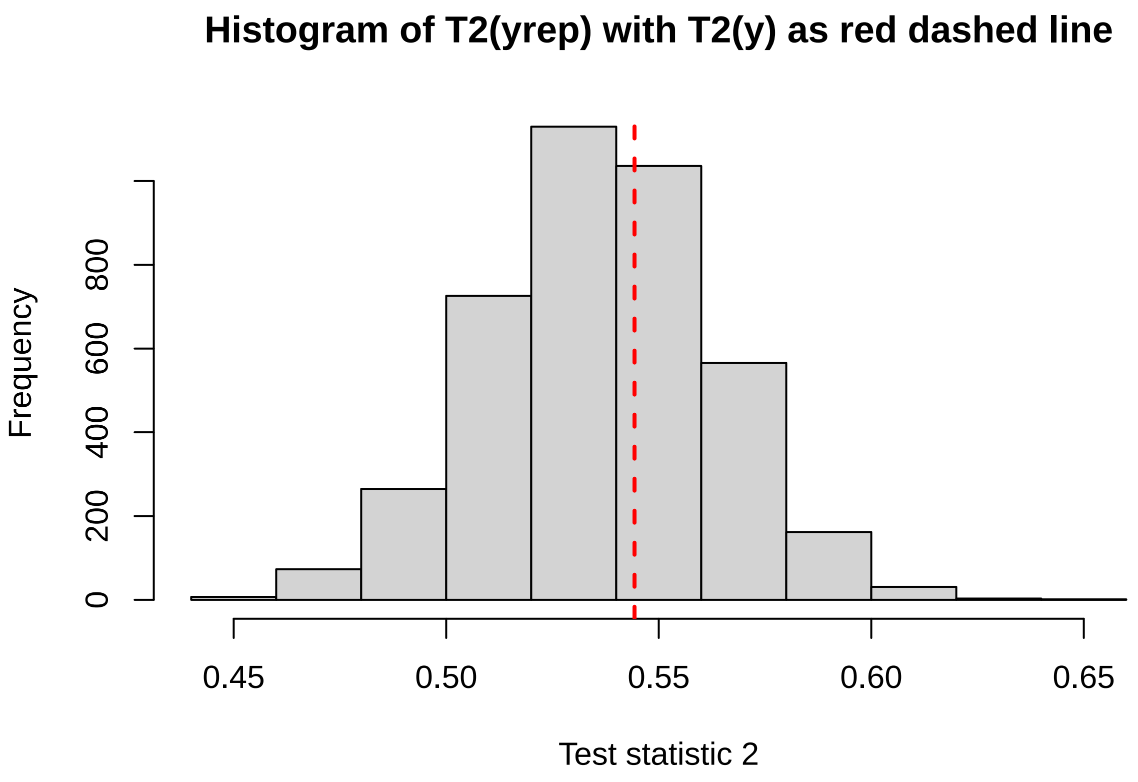


Although the posterior check p-value is pretty close to 50 and the histogram looks well because $T_1(y)$ roughly sits at the mode of $T_1(y_rep)$. Everything looks perfect.

```r
T2 <- function(y) {
  # Percentage of overall population that knows someone gay
  mean(y)
}

mean(T2(dat$y) <= apply(yrep, 1, T2))
```

```
## [1] 0.403
```

```r
hist(apply(yrep, 1, T2),
     xlab = "Test statistic 2",
     main = "Histogram of T2(yrep) with T2(y) as red dashed line",)
abline(v=T2(dat$y), lwd=2, lty=2, col="red")
```

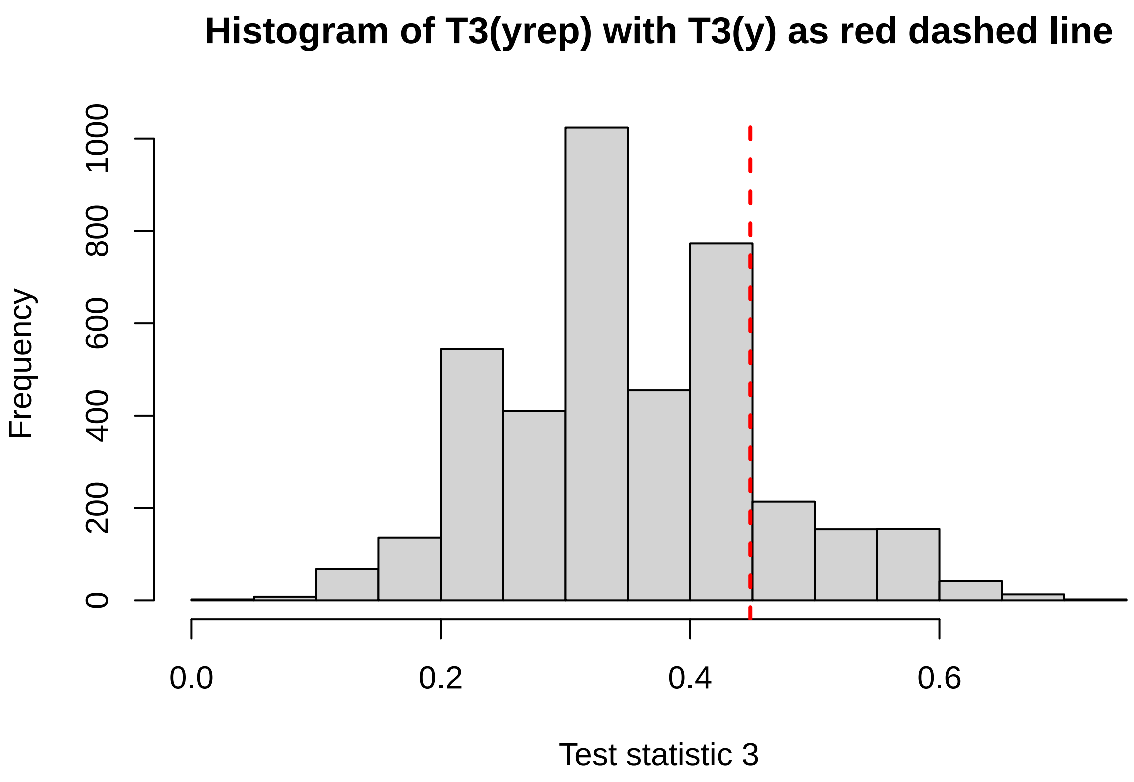


The posterior check p-value is okay, the histogram also looks well because $T_2(y)$ roughly sits at the mode of $T_2(y_rep)$.

```r
T3 <- function(y) {
  # Percentage of people of other races who does not support Dem or GOP...
  # ...that knows someone gay
  mean(y[which( (dat$x$RACE_WHITE + dat$x$RACE_BLACK) == 0 &
                (dat$x$DEMOCRAT + dat$x$REPUBLICAN) == 0 )])

}

mean(T3(dat$y) <= apply(yrep, 1, T3))
```

```
## [1] 0.23
```

```r
hist(apply(yrep, 1, T3),
     xlab = "Test statistic 3",
     main = "Histogram of T3(yrep) with T3(y) as red dashed line",)
abline(v=T3(dat$y), lwd=2, lty=2, col="red")
```



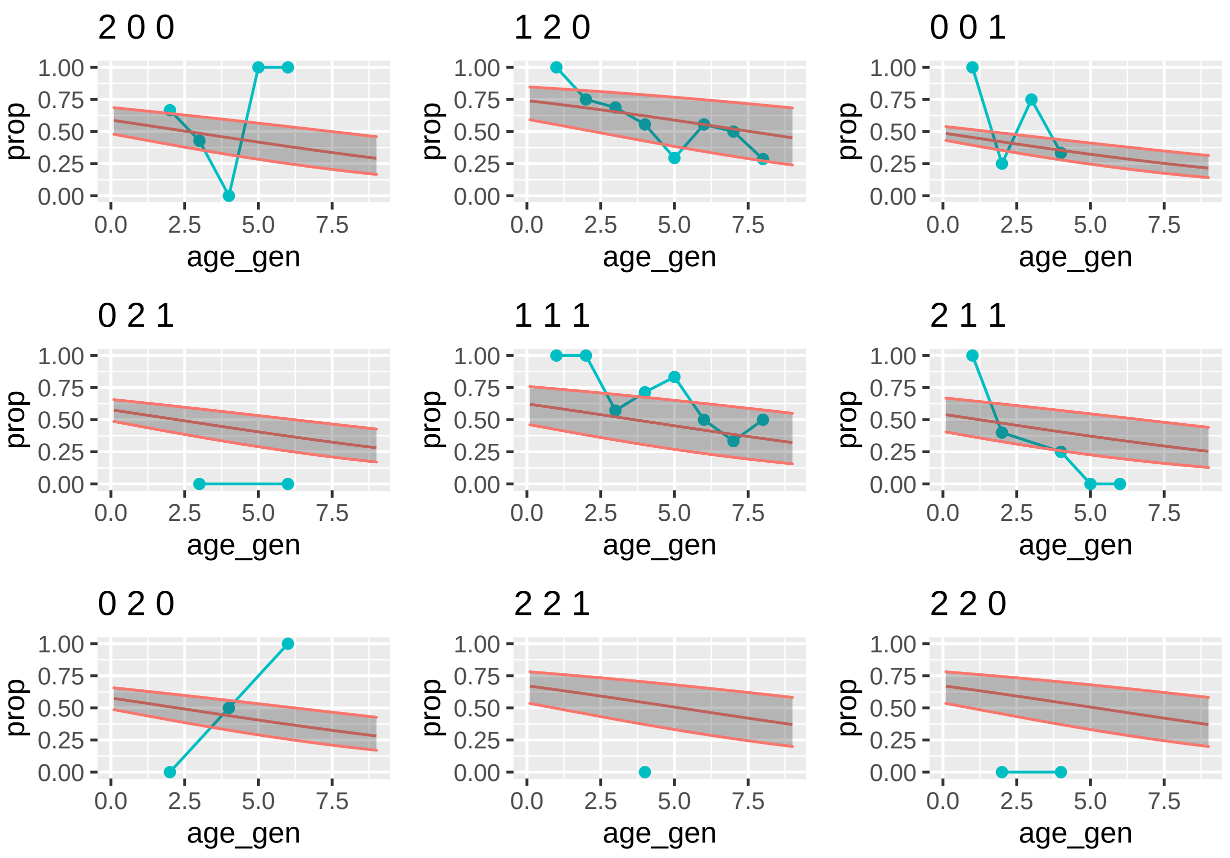
The posterior check p-value is okay, but the histogram shows that $T_2(y)$ are away from the mode of $T_2(y_rep)$ so our model is not good capturing the information of the minority population.

Overall the model fits the data well. If we want to further improve it we may consider adding interaction terms and also other regressors like $1/x$ or $log(x)$.

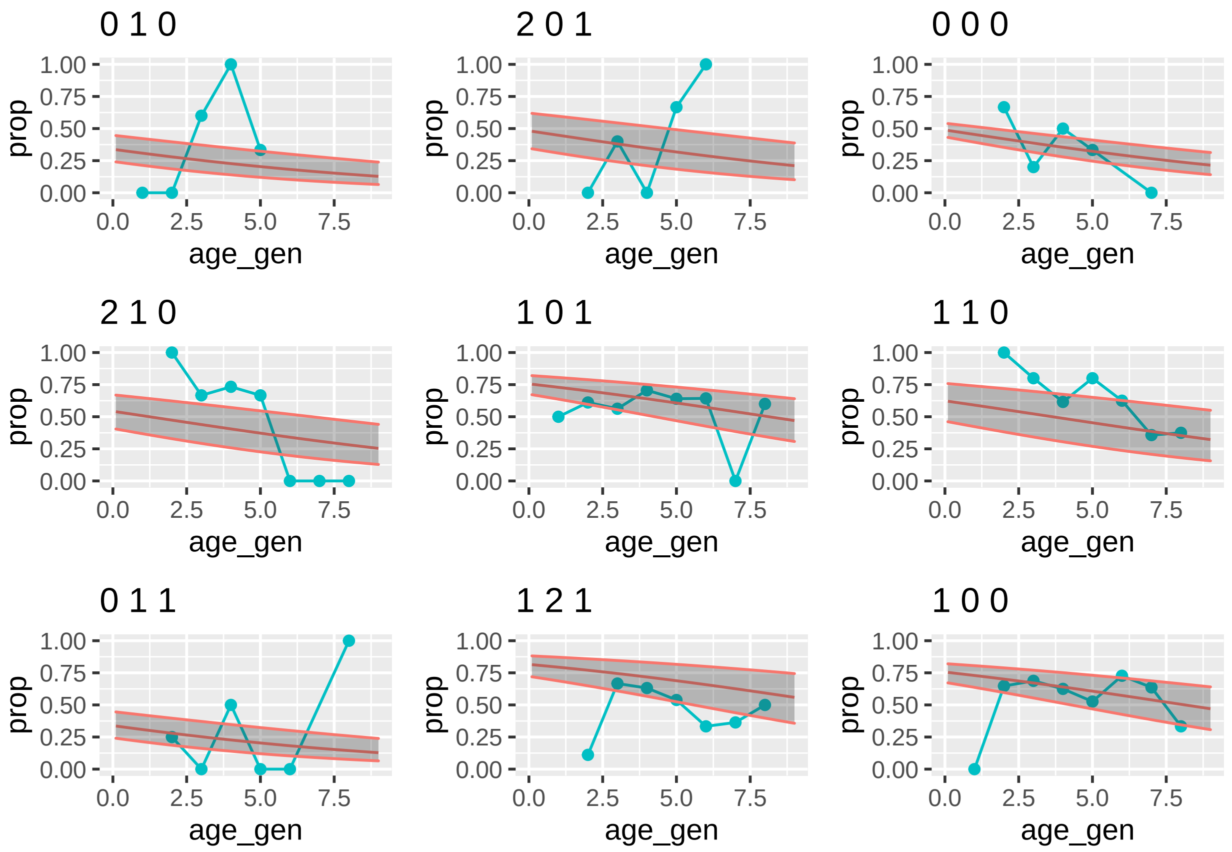
(f)

We are going to plot the estimated proportion and the data by age for each of the 18 cells defined by different combination of levels of discrete covariates sex, race and political views.

```r
param <- summary(res)$summary[1:7, "mean"]
param_lwr_upr <- summary(res)$summary[1:7, c("25%", "75%")]

dat2 <- cbind(dat$x, y=dat$y)

library(tidyverse)

Plot_func <- function(df, race, politic, sex) {
  age <- seq(1, 90, .1)
  esti <- plogis(param[1] + age*param[2] +
                   param[3:7] %*% t(df[1, 2:6]))
  lwr <- plogis(param_lwr_upr[1, 1] + age*param_lwr_upr[2, 1] +
                 param_lwr_upr[3:7, 1] %*% t(df[1, 2:6]))
  upr <- plogis(param_lwr_upr[1, 2] + age*param_lwr_upr[2, 2] +
                 param_lwr_upr[3:7, 2] %*% t(df[1, 2:6]))
  temp_dat <- data.frame(age_gen=age/10, prop=esti, upr=upr, lwr=lwr)

  plt_dat <- df %>% group_by(age_gen) %>% summarise(prop = mean(y))
  ggplot(plt_dat, aes(x=age_gen, y=prop, color="red")) +
   geom_line() + geom_point() +
   ylim(0, 1) +
   xlim(0, 9) +
   geom_line(data=temp_dat, aes(x=age_gen, y=esti, color="black")) +
   geom_ribbon(data=temp_dat, aes(ymin=lwr, ymax=upr, color="black"), alpha=0.3) +
   ggtitle(paste(race, politic, sex)) +
  guides(color = FALSE)
}

plot_dat <- dat2 %>%
  mutate(race = RACE_WHITE + 2*RACE_BLACK,
         politic = DEMOCRAT + 2*REPUBLICAN,
         age_gen = floor(AGE/10)) %>%
  group_by(race, politic, SEX) %>%
  nest() %>%
  mutate(Plot = map(data, Plot_func, race, politic, sex=SEX))

library(gridExtra)
grid.arrange(grobs=tail(plot_dat$Plot, 9),
             nrow=3, ncol=3)
```



```r
grid.arrange(grobs=head(plot_dat$Plot, 9),
             nrow=3, ncol=3)
```



In above plot, the age_gen variable is the age divided by 10 then floored. The proportion is calculated using the data within each bin of age_gen. The size of data is not large enough and we have so many cells for different combination of levels of discrete covariates and so many bins of age_gen so the data proportion looks pretty unstable and does not have a reasonable trend.

The titles 022, 120, . . . represents the levels of covariates of the group that we are plotting. The first argument is race with 0 being others, 1 being white and 2 being black. The second argument is political view with 2 being Republican, 1 being democarts and 0 being others. The third argument is the gender with 1 being male and 0 being female.

Above this layer we plot the posterior mean of proportions together with a 50 credible interval. It is true that the estimated trend is too simple because the model is not that complex, however the inconsistency between estimation and data is mostly by the fact that the data size is not large enough to compute proportion for each cell.

(g)

We first compute cases (i) and (ii).

```
param_intv <- summary(res)$summary[1:7, c("25%","mean", "75%")]

f_dat <- data.frame(age=c(30, 50),
                    race_white = c(1, 0),
                    race_black = c(0, 1),
                    sex = c(1, 1),
                    dem = c(1, 0),
                    gop = c(0, 1))
```

```r
prop_lwr <- plogis(as.matrix(f_dat) %*% param_intv[-1, 1] + param_intv[1, 1])
prop_upr <- plogis(as.matrix(f_dat) %*% param_intv[-1, 3] + param_intv[1, 3])
prop_est <- plogis(as.matrix(f_dat) %*% param_intv[-1, 2] + param_intv[1, 2])
```

```r
cat(paste("The posterior mean for the first case is", prop_est[1]))
```

```
## The posterior mean for the first case is 0.706787759521148
```

```r
cat(paste("The 50% credible interval for the first case is", prop_lwr[1], prop_upr[1]))
```

```
## The 50% credible interval for the first case is 0.53463416567914 0.835235663420717
```

```r
cat(paste("The posterior mean for the second case is", prop_est[2]))
```

```
## The posterior mean for the second case is 0.277807551213953
```

```r
cat(paste("The 50% credible interval for the second case is", prop_lwr[2], prop_upr[2]))
```

```
## The 50% credible interval for the second case is 0.142368855015276 0.471841161817894
```

For the third case we have to integrate from age 70 to the "max" age we assume which is set to be 110. This is done by taking the average.

```r
f_dat <- data.frame(age=seq(70, 110, .1),
                    race_white = 1,
                    race_black = 0,
                    sex = 0,
                    dem = 0,
                    gop = 0)
prop_lwr <- mean(plogis(as.matrix(f_dat) %*% param_intv[-1, 1] + param_intv[1, 1]))
prop_upr <- mean(plogis(as.matrix(f_dat) %*% param_intv[-1, 3] + param_intv[1, 3]))
prop_est <- mean(plogis(as.matrix(f_dat) %*% param_intv[-1, 2] + param_intv[1, 2]))
```

```r
cat(paste("The posterior mean for the third case is", prop_est))
```

```
## The posterior mean for the third case is 0.470286859111071
```

```r
cat(paste("The 50% credible interval for the third case is", prop_lwr, prop_upr))
```

```
## The 50% credible interval for the third case is 0.309091464800977 0.639961689013043
```

Problem 4

(a)

Under H_1 We first compute $p(y|H_1)$ directly. Actually you can use the fact that y_j is distributed as $\theta_j + \epsilon_j$ where ϵ_j is independent mean zero Gaussian with variance σ_j^2 to easily get the marginal distribution of y_j without any calculation. But we will do the calculation anyway.

$$p(y|H_1) = \int \prod_{j=1}^{J} \left\{ N(y_j|\theta_j, \sigma_j^2) N(\theta_j|0, A^2) \right\} \, d\vec{\theta}$$

where $\vec{\theta} = (\theta_1, \ldots, \theta_J)$.

Continuing the calculation, we have $p(y|H_1)$ equals

$$\begin{aligned}
&= \prod_{j=1}^{J} \int N(y_j|\theta_j, \sigma_j^2) N(\theta_j|0, A^2) \, d\theta_j \\
&= \prod_{j=1}^{J} \int \frac{1}{2\pi} \frac{1}{A\sigma_j} \exp\left(-\frac{\theta_j^2}{2A^2} - \frac{(\theta_j - y_j)^2}{2\sigma_j^2} \right) d\theta_j \\
&= \prod_{j=1}^{J} \int \frac{1}{2\pi} \frac{1}{A\sigma_j} \exp\left(-\frac{(A^2+\sigma_j^2)\theta_j^2 - 2A^2 y_j \theta_j + \frac{A^4 y_j^2}{A^2+\sigma_j^2}}{2A^2\sigma_j^2} \right) \exp\left(\frac{\frac{A^4 y_j^2}{A^2+\sigma_j^2} - A^2 y_j^2}{2A^2\sigma_j^2} \right) d\theta_j \\
&= \prod_{j=1}^{J} \int \frac{1}{2\pi} \frac{1}{A\sigma_j} \exp\left(-\frac{(\theta_j - \frac{A^2 y_j}{A^2+\sigma_j^2})^2}{\frac{2A^2\sigma_j^2}{A^2+\sigma_j^2}} \right) \exp\left(-\frac{y_j^2}{2(A^2+\sigma_j^2)} \right) d\theta_j \\
&= \prod_{j=1}^{J} \frac{1}{\sqrt{2\pi}} \frac{1}{\sqrt{A^2+\sigma_j^2}} \exp\left(-\frac{y_j^2}{2(A^2+\sigma_j^2)} \right) \int \frac{1}{\sqrt{2\pi}} \frac{\sqrt{A^2+\sigma_j^2}}{A\sigma_j} \exp\left(-\frac{(\theta_j - \frac{A^2 y_j}{A^2+\sigma_j^2})^2}{\frac{2A^2\sigma_j^2}{A^2+\sigma_j^2}} \right) d\theta_j \\
&= \prod_{j=1}^{J} \frac{1}{\sqrt{2\pi(A^2+\sigma_j^2)}} \exp\left(-\frac{y_j^2}{2(A^2+\sigma_j^2)} \right).
\end{aligned}$$

Under H_2 Next we calculate $p(y|H_2)$.

$$p(y|H_2) = \int N(\theta|0, A^2) \prod_{j=1}^{J} N(y_j|\theta, \sigma_j^2) \, d\theta$$

Since the density of $N(\theta|0, A^2)$ equals to the density of $N(0|\theta, A^2)$, for convenience let's denote $y_0 = 0$ and $\sigma_0 = A$ and we have $p(y|H_2)$ equals

$$\begin{aligned}
&= \int \prod_{j=0}^{J} N(y_j|\theta, \sigma_j^2) \, d\theta \\
&= \left(\prod_{j=0}^{J} \frac{1}{\sqrt{2\pi}\sigma_j} \right) \int \exp\left(-\sum_{j=0}^{J} \frac{(\theta - y_j)^2}{2\sigma_j^2} \right) d\theta
\end{aligned}$$

Similarly, completing the square we have

$$= \left(\prod_{j=0}^{J} \frac{1}{\sqrt{2\pi}\sigma_j}\right) \int \exp\left(-\frac{\left(\theta - (\sum_{j=0}^{J} \frac{y_j}{2\sigma_j^2})/(\sum_{j=0}^{J} \frac{1}{2\sigma_j^2})\right)^2}{1/\sum_{j=0}^{J} \frac{1}{2\sigma_j^2}}\right) \exp\left(\frac{(\sum_{j=0}^{J} \frac{y_j}{2\sigma_j^2})^2}{\sum_{j=0}^{J} \frac{1}{2\sigma_j^2}} - \sum_{j=0}^{J} \frac{y_j^2}{2\sigma_j^2}\right) d\theta$$

$$= \left(\prod_{j=0}^{J} \frac{1}{\sqrt{2\pi}\sigma_j}\right) \sqrt{\frac{2\pi}{\sum_{j=0}^{J} \frac{1}{\sigma_j^2}}} \exp\left(\frac{(\sum_{j=0}^{J} \frac{y_j}{2\sigma_j^2})^2}{\sum_{j=0}^{J} \frac{1}{2\sigma_j^2}} - \sum_{j=0}^{J} \frac{y_j^2}{2\sigma_j^2}\right) \int \sqrt{\frac{\sum_{j=0}^{J} \frac{1}{\sigma_j^2}}{2\pi}} \exp\left(-\frac{\left(\theta - (\sum_{j=0}^{J} \frac{y_j}{2\sigma_j^2})/(\sum_{j=0}^{J} \frac{1}{2\sigma_j^2})\right)^2}{1/\sum_{j=0}^{J} \frac{1}{2\sigma_j^2}}\right) d\theta$$

$$= \left(\prod_{j=0}^{J} \frac{1}{\sqrt{2\pi}\sigma_j}\right) \sqrt{\frac{2\pi}{\sum_{j=0}^{J} \frac{1}{\sigma_j^2}}} \exp\left(\frac{(\sum_{j=0}^{J} \frac{y_j}{2\sigma_j^2})^2}{\sum_{j=0}^{J} \frac{1}{2\sigma_j^2}} - \sum_{j=0}^{J} \frac{y_j^2}{2\sigma_j^2}\right)$$

Notice that $y_0 = 0$ so we can write it as

$$\left(\prod_{j=0}^{J} \frac{1}{\sqrt{2\pi}\sigma_j}\right) \sqrt{\frac{2\pi}{\sum_{j=0}^{J} \frac{1}{\sigma_j^2}}} \exp\left(\frac{(\sum_{j=0}^{J} \frac{y_j}{2\sigma_j^2})^2}{\sum_{j=0}^{J} \frac{1}{2\sigma_j^2}} - \sum_{j=0}^{J} \frac{y_j^2}{2\sigma_j^2}\right)$$

Notice that inside the exponential is the weighted sample variance, we can simplify it to the following formula,

$$\left(\prod_{j=0}^{J} \frac{1}{\sqrt{2\pi}\sigma_j}\right) \sqrt{\frac{2\pi}{\sum_{j=0}^{J} \frac{1}{\sigma_j^2}}} \exp\left(-\sum_{j=0}^{J} w_j (y_j - \bar{y})^2\right)$$

where $w_j = \frac{1}{2\sigma_j^2}$ are the weights, $\bar{y} = \frac{\sum_{j=0}^{J} w_j y_j}{\sum_{j=0}^{J} w_j}$ is the weighted average.

The Bayes factor Now we compute the Bayes factor,

$$\frac{p(y|H_1)}{p(y|H_2)} = \left\{\prod_{j=1}^{J} \frac{1}{\sqrt{2\pi(A^2+\sigma_j^2)}} \exp\left(-\frac{y_j^2}{2(A^2+\sigma_j^2)}\right)\right\} \left\{\left(\prod_{j=0}^{J} \frac{1}{\sqrt{2\pi}\sigma_j}\right) \sqrt{\frac{2\pi}{\sum_{j=0}^{J} \frac{1}{\sigma_j^2}}} \exp\left(-\sum_{j=0}^{J} w_j (y_j - \bar{y})^2\right)\right\}^{-1}$$

Since $y_0 = 0$ and $\sigma_0^2 = A^2$ we can simplify it,

$$= \sqrt{\frac{1}{A \sum_{j=1}^{J} \frac{1}{\sigma_j^2}}} \left(\prod_{j=1}^{J} \frac{1}{\sqrt{A^2\sigma_j^2 + \sigma_j^4}}\right) \exp\left\{-\sum_{j=0}^{J} \left(\frac{y_j^2}{2(A^2+\sigma_j^2)} - w_j (y_j - \bar{y})^2\right)\right\}.$$

There might be a way to further simplify but I think this is good enough for the analysis.

(b)

For a fixed y and σ, evaluate

$$\lim_{A\to\infty} \sqrt{\frac{1}{A \sum_{j=1}^{J} \frac{1}{\sigma_j^2}}} \left(\prod_{j=1}^{J} \frac{1}{\sqrt{A^2\sigma_j^2 + \sigma_j^4}}\right) \exp\left\{-\sum_{j=0}^{J} \left(\frac{y_j^2}{2(A^2+\sigma_j^2)} - w_j (y_j - \bar{y})^2\right)\right\}.$$

Notice that

$$\lim_{A\to\infty} \exp\left\{-\sum_{j=0}^{J}\left(\frac{y_j^2}{2(A^2+\sigma_j^2)}\right)\right\} = 1$$

$$\lim_{A\to\infty} \sqrt{\frac{1}{A\sum_{j=1}^{J}\frac{1}{\sigma_j^2}}}\left(\prod_{j=1}^{J}\frac{1}{\sqrt{A^2\sigma_j^2+\sigma_j^4}}\right) = 0$$

and

$$\lim_{A\to\infty} \exp\left\{-\sum_{j=0}^{J} w_j(y_j-\bar{y})^2\right\} = \exp\left\{\lim_{A\to\infty} -\sum_{j=0}^{J} w_j(y_j-\bar{y})^2\right\} = \exp(-\sum_{j=1}^{J} w_j(y_j-\bar{y}_{1,\dots,J})^2) = C < \infty.$$

The last one by the fact that for a weighted variance if the weight of one of the term shrinks to zero then it converges to the weighted variance without that term. Here $\bar{y}_{1,\dots,J}$ represents the weighted average of only y_1 to y_J without y_0.

With above terms evaluated, we conclude that $\lim_{A\to\infty}\frac{p(y|H_1)}{p(y|H_2)} = 0$. So the Bayes factor favors H_2.

(c)

With the simplification that $\sigma_1 = \cdots = \sigma_J = \sigma$, the Bayes factor

$$\frac{p(y|H_1)}{p(y|H_2)} = \left\{\prod_{j=1}^{J}\frac{1}{\sqrt{2\pi(A^2+\sigma_j^2)}}\exp\left(-\frac{y_j^2}{2(A^2+\sigma_j^2)}\right)\right\}\left\{\left(\prod_{j=0}^{J}\frac{1}{\sqrt{2\pi}\sigma_j}\right)\sqrt{\frac{2\pi}{\sum_{j=0}^{J}\frac{1}{\sigma_j^2}}}\exp\left(-\sum_{j=0}^{J}w_j(y_j-\bar{y})^2\right)\right\}^{-1}$$

can be further simplified. The (weighted variance) formula we used for the analysis when A goes to infinity is no longer convenient, so we return back to the original formula and simplify the terms outside the exponent, now the Bayes factor is

$$\begin{aligned}&(AJ)^{-1/2}\sigma^{-1}(A^2\sigma^2+\sigma^4)^{-J/2}\exp\left(-\frac{\sum_{j=1}^{J}y_j^2}{2(A^2+\sigma^2)}\right)\exp\left(\frac{(\sum_{j=1}^{J}\frac{y_j}{2\sigma_j^2})^2}{\sum_{j=1}^{J}\frac{1}{2\sigma_j^2}+\frac{1}{2A^2}}-\sum_{j=1}^{J}\frac{y_j^2}{2\sigma_j^2}\right)\\ =\ &(AJ)^{-1/2}\sigma^{-1}(A^2\sigma^2+\sigma^4)^{-J/2}\exp\left(-\frac{\sum_{j=1}^{J}y_j^2}{2(A^2+\sigma^2)}\right)\exp\left(\frac{(\sum_{j=1}^{J}y_j)^2}{J+\frac{\sigma^2}{A^2}}-\frac{\sum_{j=1}^{J}y_j^2}{2\sigma^2}\right).\end{aligned}$$

The sample mean and variance remains the same when J increases, thus $\sum_{j=1}^{J}y_j$ does not change as J increases, and $\sum_{j=1}^{J}y_j^2$ increases as J increases because of the $1/(J-1)$ factor in the formula of sample variance. Since $(AJ)^{-1/2}$, $(A^2\sigma^2+\sigma^4)^{-J/2}$, $\exp(1/(J+\sigma^2/A^2))$ and $\exp(-\sum_{j=1}^{J}y_j^2)$ are all decreasing function of J and other terms remains unchanged, we have that the whole term decreases as J increases.

This means that when we have more and more groups but keep observing stable sample mean and sample variance, the H_2 complete pooling model is more and more preferred by the Bayes factor. According to this result. when we have less cells/groups it might better to use no pooling model, each group has its own independent mean. When the group number is large it is better to use complete pooling model where we decide that all groups share the same mean parameter.