Homework 2 PUBH 7440

Jake Wittman

2/17/2020

1a) From Gelman 3.2, the posterior mean given a non-informative uniform prior on $(\mu, log\sigma)$ is

$p(u|y_i) \sim t_{n-1}(\bar{y}, \frac{s^2}{n})$

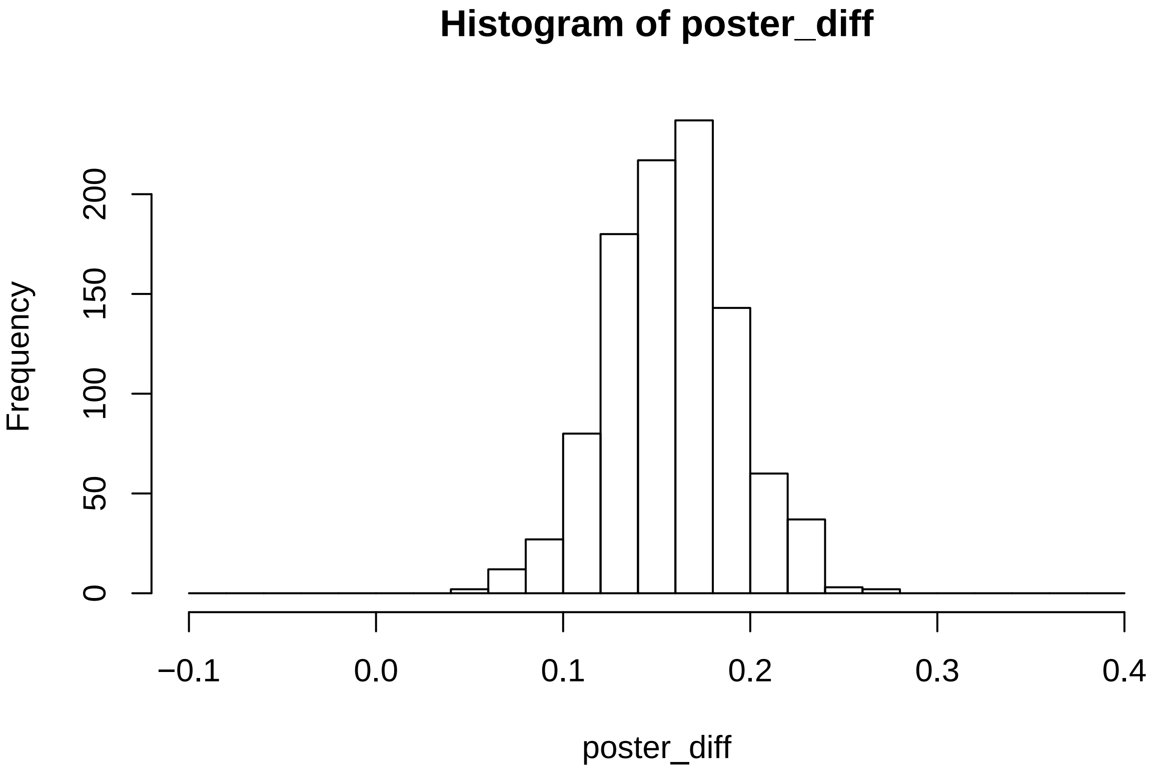
so $p(u_c|y_i) \sim t_{31}(1.013, \frac{0.025^2}{32})$ and

$p(u_t|y_i) \sim t_{35}(1.173, \frac{0.20^2}{36})$

1b)

```r
mu.c <- 1.013 + (0.025/sqrt(32))*rt(1000,31)
mu.t <- 1.173 + (0.20/sqrt(36))*rt(1000,35)

poster_diff <- mu.t - mu.c
hist(poster_diff, breaks = seq(-0.1, 0.4, 0.02))
```



The 95% credible interval is [0.0949057, 0.2279495]

2)

$$p\left(\mu,\sigma^2|y\right)\propto p\left(y|\mu,\sigma^2\right)p\left(\mu,\sigma^2\right)$$
$$\propto\left(\sigma^2\right)^{-n/2}\exp\left(-\frac{(n-1)s^2+n(\mu-\bar{y})^2}{2\sigma^2}\right)\sigma^{-1}\left(\sigma^2\right)^{-(\nu_0/2+1)}\exp\left(-\frac{\nu_0\sigma_0^2+\kappa_0(\mu-\mu_0)^2}{2\sigma^2}\right)$$
$$\propto\sigma^{-1}\left(\sigma^2\right)^{-((v_0+n)/2+1)}\exp\left(-\frac{\nu_0\sigma_0^2+(n-1)s^2+\frac{n\kappa_0(\bar{y}-\mu_0)^2}{n+\kappa_0}+(n+\kappa_0)\left(\mu-\frac{\mu_0\kappa_0+n\bar{y}}{n+\kappa_0}\right)^2}{2\sigma^2}\right)$$

Because of the above,

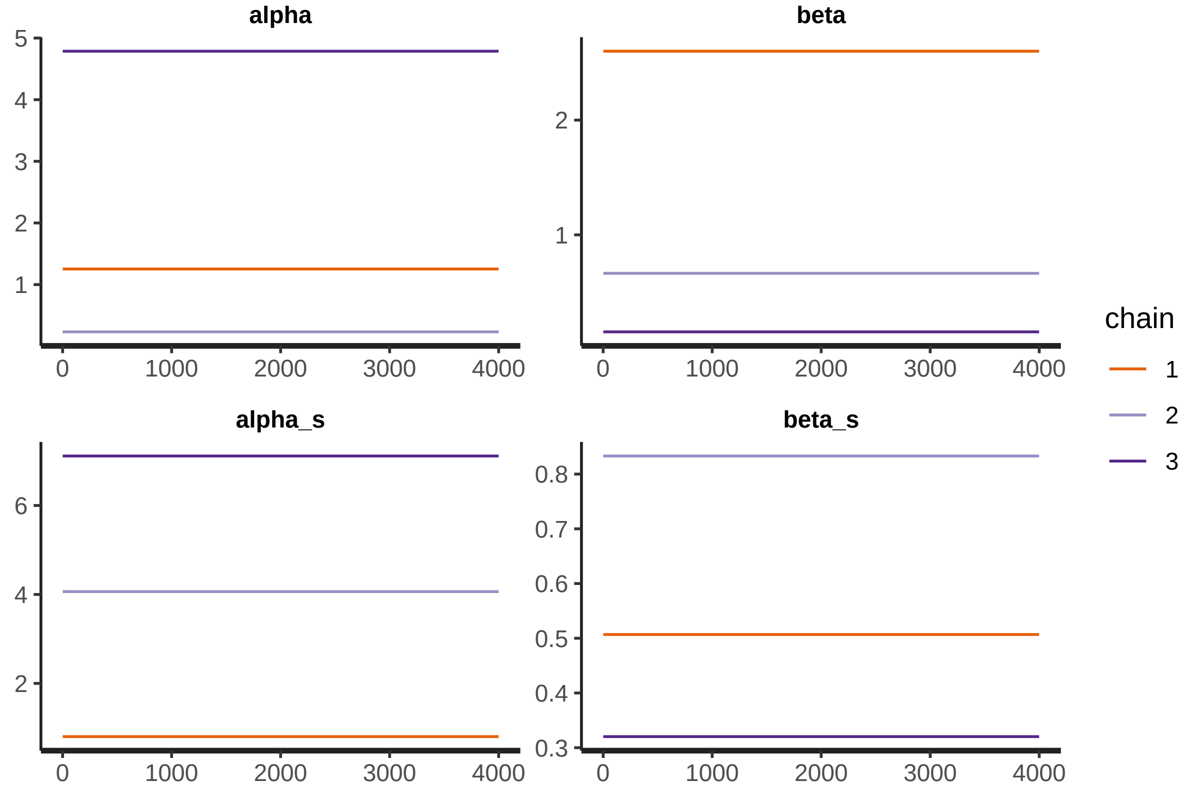
$$\mu,\sigma^2|y\sim\mathrm{N}-\mathrm{Inv}-\chi^2\left(\frac{\mu_0\kappa_0+n\bar{y}}{n+\kappa_0},\frac{\sigma_n^2}{n+\kappa_0};n+\nu_0,\sigma_n^2\right)$$

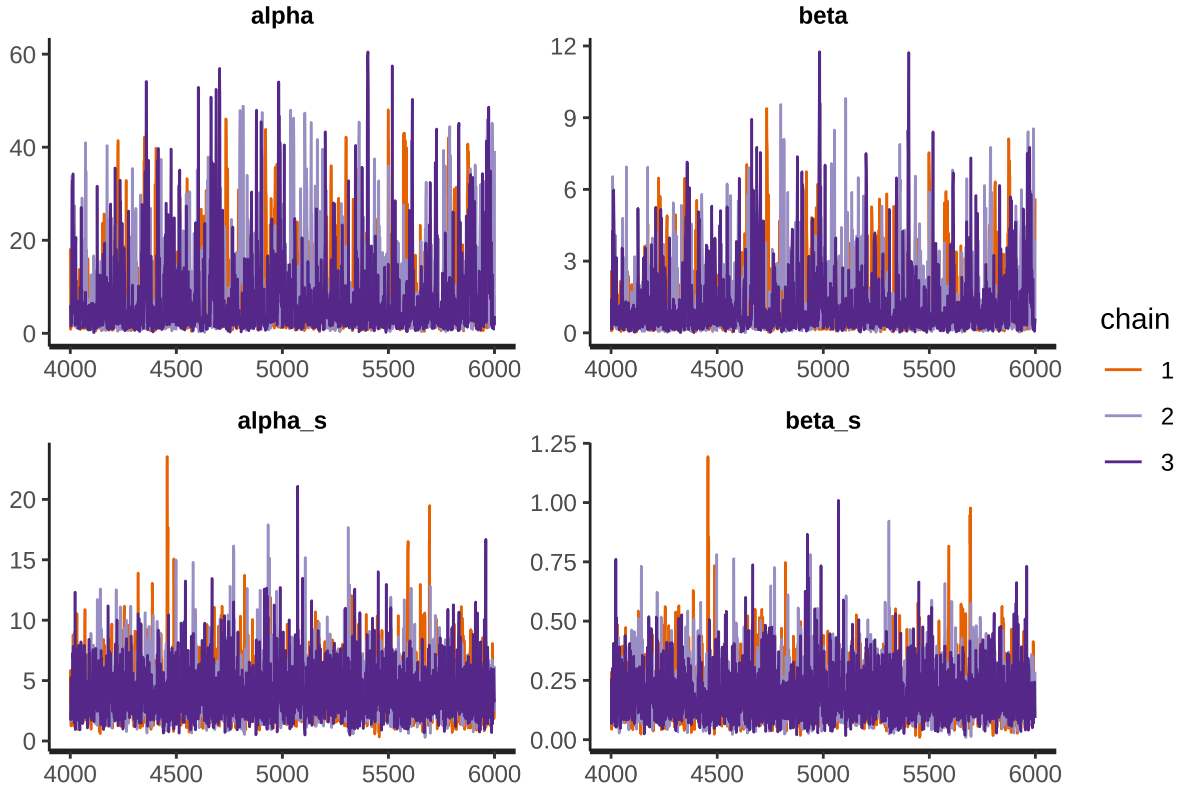
with

$$\sigma_n^2=\frac{\nu_0\sigma_0^2+(n-1)s^2+\frac{n\kappa_0(\bar{y}-\mu_0)^2}{n+\kappa_0}}{n+\nu_0}$$

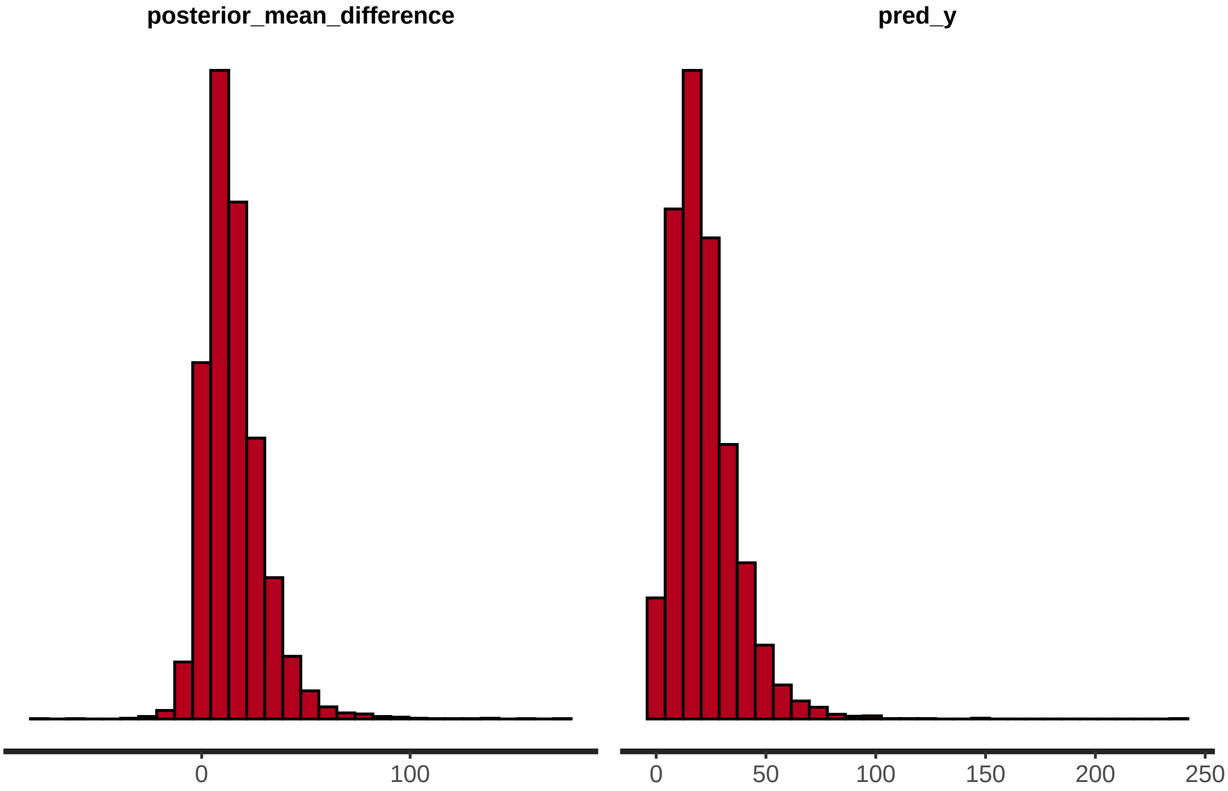
3a) Stan code used to fit the models is in the code appendix at the end of this assignment

Plots of the chains

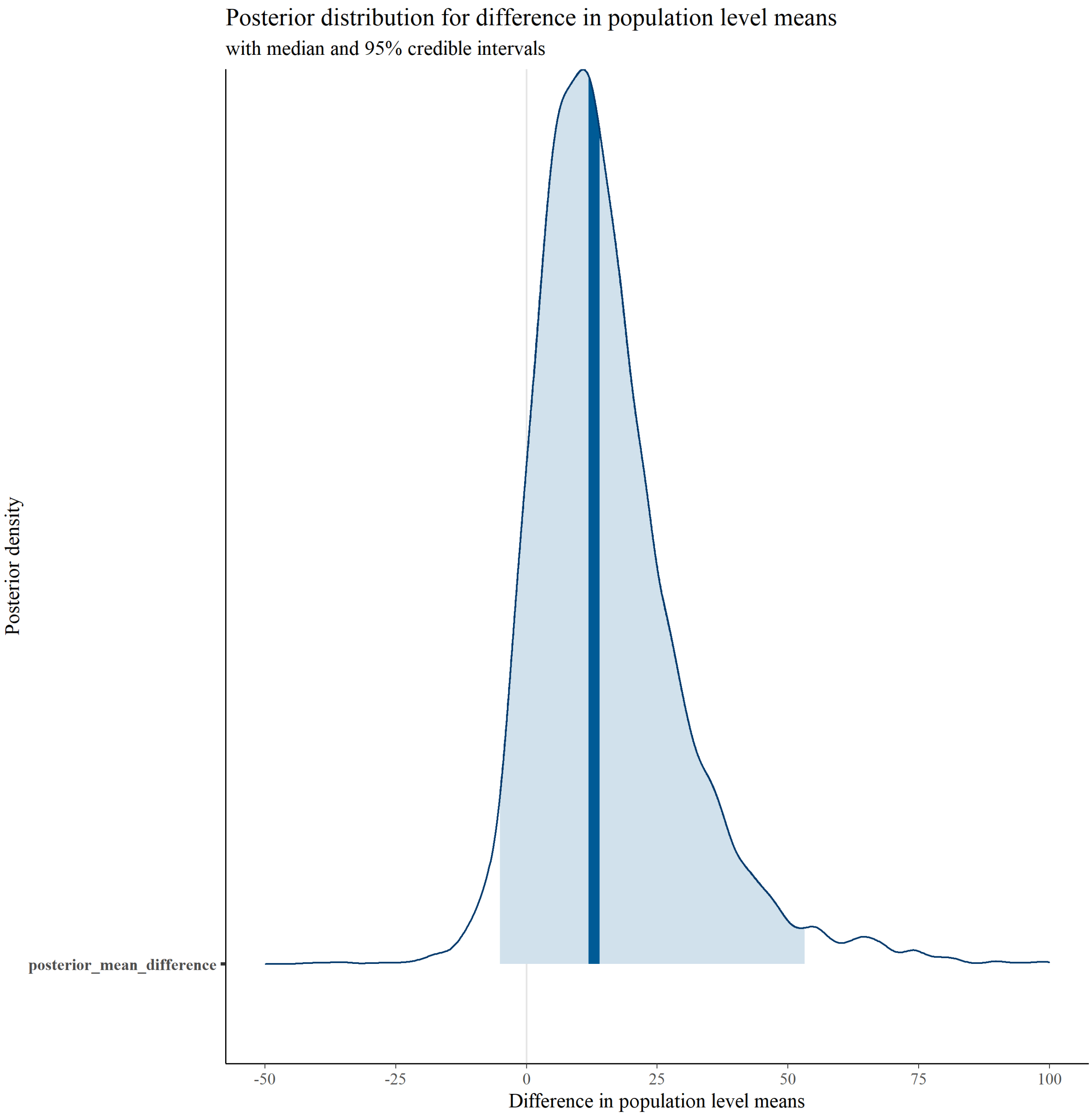


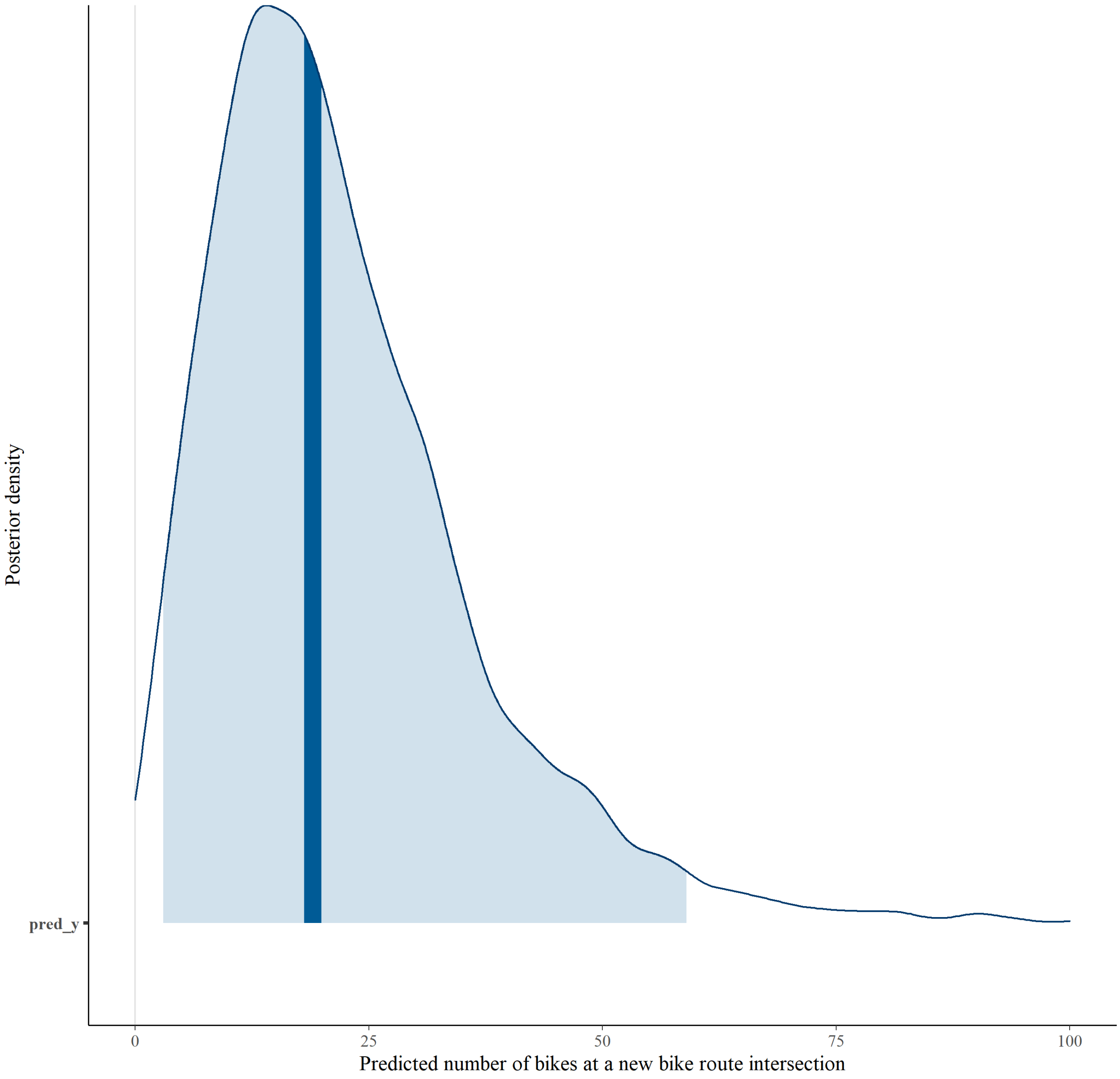


Posterior plots for the difference in the population means and the new predicted y value.



Posterior plots with 95% credible intervals.

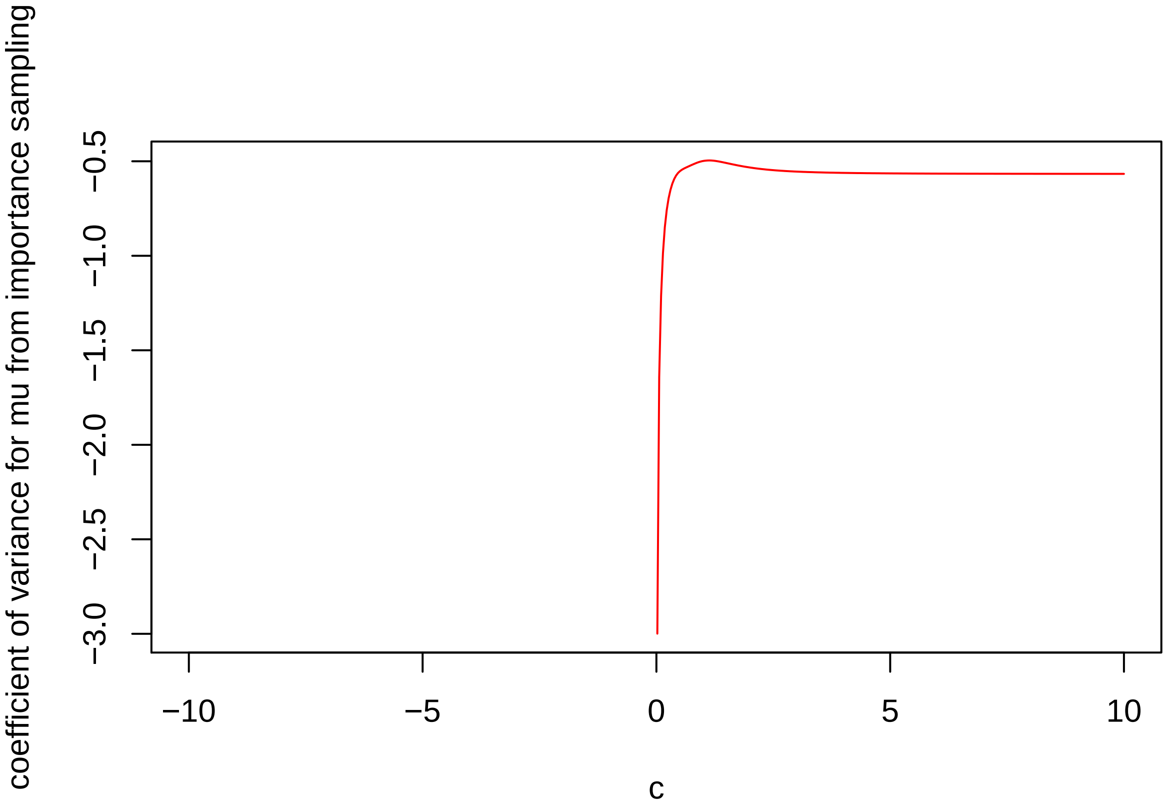

Posterior distribution for difference in population level means
with median and 95% credible intervals
Posterior density
posterior_mean_difference
-50
-25
0
25
50
75
100
Difference in population level means

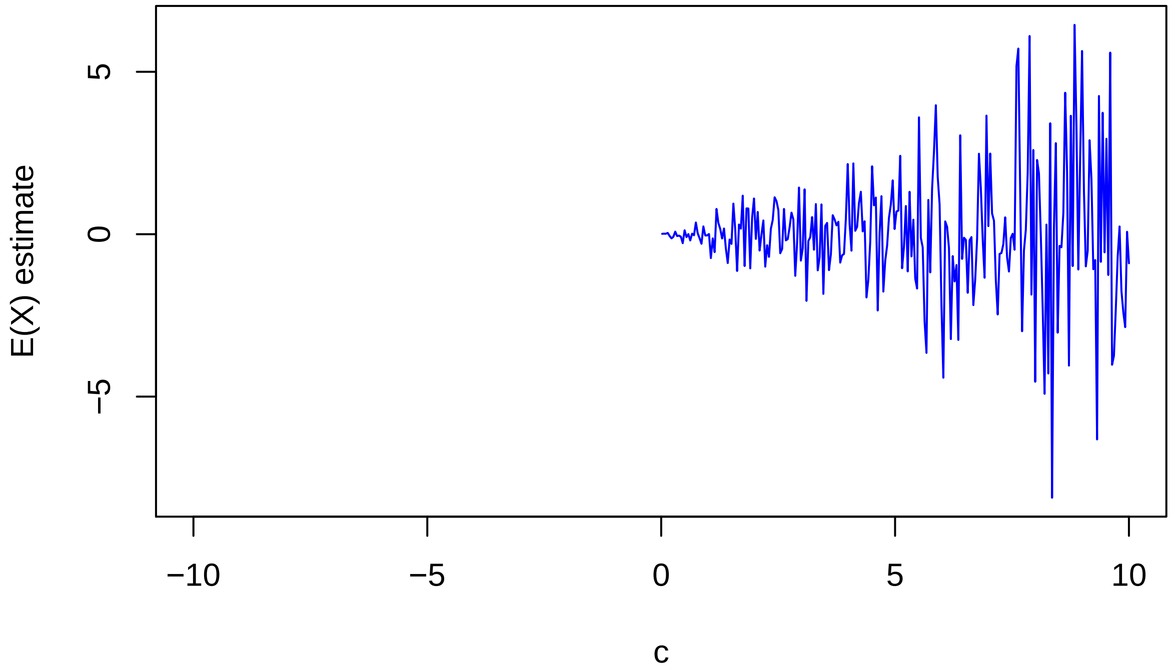


Predictive distribution for the number of bikes on a new bike route intersection

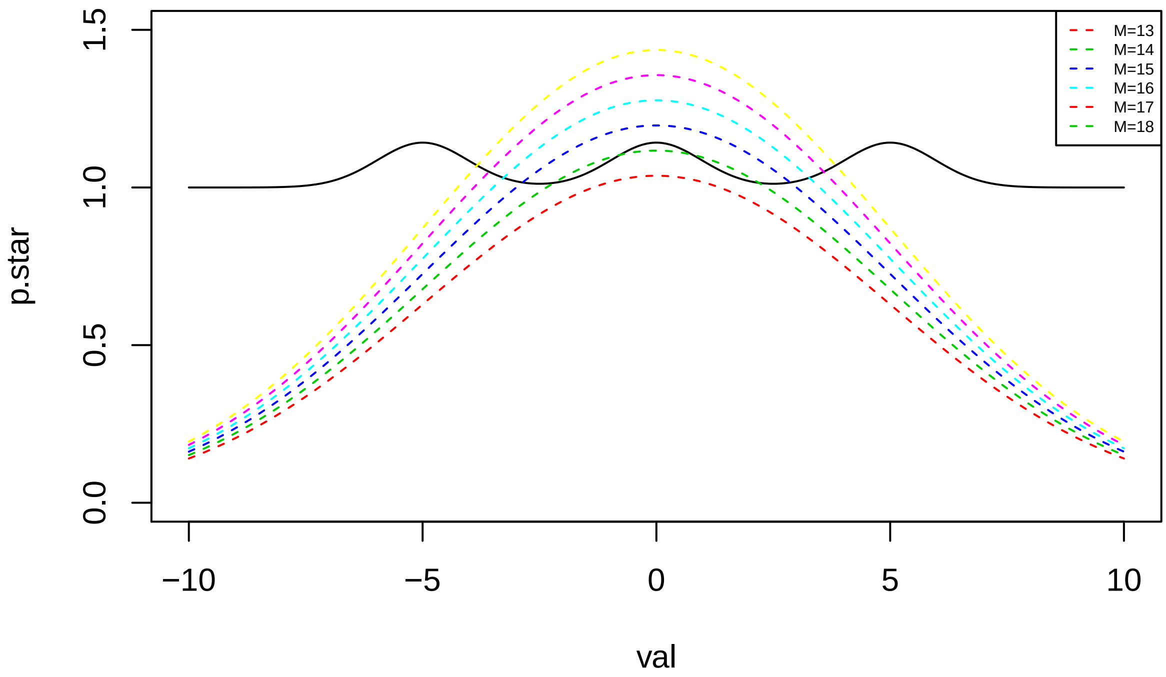
with median and 95% credible intervals

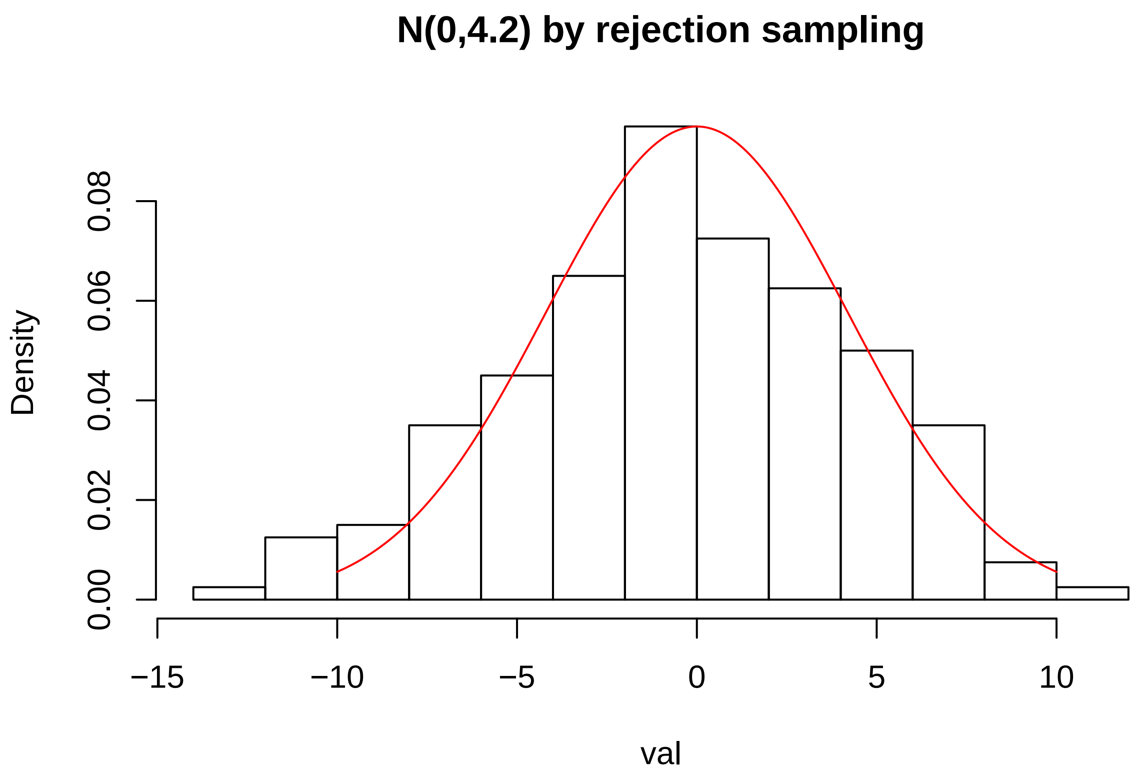
4)


coefficient of variance for mu from importance sampling
−0.5
−1.0
−1.5
−2.0
−2.5
−3.0
−10
−5
0
5
10
c



The final estimate for μ from importance sampling is 0.0035294. But I don't think I did this part right…


p.star
0.0
0.5
1.0
1.5
−10
−5
0
5
10
val
M=13
M=14
M=15
M=16
M=17
M=18



Code Appendix

```
knitr::opts_chunk$set(echo = FALSE)
mu.c <- 1.013 + (0.025/sqrt(32))*rt(1000,31)
mu.t <- 1.173 + (0.20/sqrt(36))*rt(1000,35)

poster_diff <- mu.t - mu.c
hist(poster_diff, breaks = seq(-0.1, 0.4, 0.02))
data {
  // Define data in this block

  //data for bike route intersections
  int<lower=0> n_s;// sample size for bike route intersections
  int<lower=0> y_s[n_s]; // bike route data

    //data for non-bike route intersections
  int<lower=0> n;// sample size for streets intersections
  int<lower=0> y[n]; // streets data

}


parameters {
  // Define parameters in this block
  // parameters for bike route intersection
  real<lower = 0> lambda_s[n_s]; //lambda for streets
  real<lower = 0> alpha_s; // alpha for streets
  real<lower = 0> beta_s; // beta for streets

    // parameters for non-bike route intersection
  real<lower = 0> lambda[n]; //lambda for streets
  real<lower = 0> alpha; // alpha for streets
  real<lower = 0> beta; // beta for streets
}

// The model
model {
  // Define model in this block

  // Model for bike route intersections
  target += gamma_lpdf(alpha_s | 0.01, 0.01); // gamma hyperprior on alpha
  target += gamma_lpdf(beta_s | 0.01, 0.01); // gamma hyperprior on beta
  target += gamma_lpdf(lambda_s | alpha_s, beta_s); // gamma prior on lambda
  target += poisson_lpmf(y_s | lambda_s); // distribution of response given lambda

    // Model for non-bike route intersections
  target += gamma_lpdf(alpha | 0.01, 0.01); // gamma hyperprior on alpha
  target += gamma_lpdf(beta | 0.01, 0.01); // gamma hyperprior on beta
  target += gamma_lpdf(lambda | alpha, beta); // gamma prior on lambda
  target += poisson_lpmf(y | lambda); // distribution of response given lambda

}

generated quantities{
  // Define other generated quantities in this block

  // Difference in lambdas
 real posterior_mean_difference;
 real pred_lambda;
 real pred_y;


 posterior_mean_difference = gamma_rng(alpha_s, beta_s) - gamma_rng(alpha, beta);

 pred_lambda = gamma_rng(alpha_s, beta_s);
 pred_y = poisson_rng(pred_lambda);


}

# Load libraries ----------------------------------------------------------
pacman::p_load(tidyverse,
               rstan,
               brms,
               bayesplot)
# data --------------------------------------------------------------------
intersections <- c(1:18)
```

```r
streets <- c(rep(1, 10), rep(0, 8))
bikes <- c(16, 9, 10, 13, 19, 20, 18, 17, 35, 55, 12, 1, 2, 4, 9, 7, 9, 8)

bike <- data.frame(
  intersections = intersections,
  streets = streets,
  bikes = bikes
)

bike_street <- bike %>% dplyr::filter(streets == 1)
bike_no_streets <- bike %>% dplyr::filter(streets == 0)


# Stan model --------------------------------------------------------------
bike_model <- rstan::stan_model(file = here::here("homework/hw2.stan"))
# 3a ----------------------------------------------------------------------
set.seed(1)
out_model1_streets <- sampling(
  object = bike_model,
  data = list(y_s = bike_street$bikes,
              n_s = nrow(bike_street),
              y = bike_no_streets$bikes,
              n = nrow(bike_no_streets)),
  warmup = 0,
  iter = 4000,
  chains = 3,
  cores = 4,
  control = list(adapt_delta = 0.9),
  show_messages = FALSE
)

traceplot(out_model1_streets, par = c("alpha", "beta", "alpha_s", "beta_s"))


# 3b ----------------------------------------------------------------------

set.seed(1)
out_model1_streets <- sampling(
  object = bike_model,
  data = list(y_s = bike_street$bikes,
              n_s = nrow(bike_street),
              y = bike_no_streets$bikes,
              n = nrow(bike_no_streets)),
  warmup = 4000,
  iter = 6000,
  chains = 3,
  cores = 4,
  control = list(adapt_delta = 0.9),
  show_messages = FALSE
)

traceplot(out_model1_streets, par = c("alpha", "beta", "alpha_s", "beta_s"))
stan_hist(out_model1_streets, pars = c("posterior_mean_difference", "pred_y"))
```

```
mcmc_areas(out_model1_streets,
           pars = c("posterior_mean_difference"),
           prob = 0.95) +
  labs(title = "Posterior distribution for difference in population level means",
       subtitle = "with median and 95% credible intervals",
       x = "Difference in population level means",
       y = "Posterior density") +
  scale_x_continuous(limits = c(-50, 100),
                     labels = seq(-50, 100, 25),
                     breaks = seq(-50, 100, 25))

mcmc_areas(out_model1_streets,
           pars = c("diff_ys"),
           prob = 0.95) +
  labs(title = "Predictive distribution for the number of bikes on a new bike route intersection",
       subtitle = "with median and 95% credible intervals",
       x = "Predicted number of bikes at a new bike route intersection",
       y = "Posterior density") +
  scale_x_continuous(limits = c(0, 100),
                     labels = seq(0, 100, 25),
                     breaks = seq(0, 100, 25))

# Question 4 ------------------------------------------------------------------


# Importance sampling ---------------------------------------------------------
posterior <- function(x) {

  post <- (1/3)*(1/sqrt(2*pi))*(exp((-1/2)*(x+5)^2) + exp((-1/2)*x^2) + exp((-1/2)*(x-5)^2))
  return(post)
}



# prep2: define the function to calculate log(weights)=log(p*(theta|x))-log(g(theta))
weight <- function(mu_j, c_mean, c_sd){

    posterior(mu_j) - dnorm(mu_j, mean = c_mean, sd = c_sd)
}


IS <- function(X, c, N) {

    # first calculate all the parameters
    n <- length(X)
    c_mean <- c * 0
    c_sd <- c * 4.2
    mu_j <- X

    # step 1: generate N samples from the importance function g(theta)
    theta <- rnorm(N, c_mean, c_sd)
    # step 2: use the defined functions to calculate log of weights
    # We work in the log scale for the sake of computational stability
    lw <- weight(mu_j, c_mean, c_sd)


    # The following step is to avoid inifite values when taking exponential of lw
    # Using the step we have weights:  w = exp(lw) = exp(w.scaled)*exp(max(lw))
    # Since w are in both the nominator and the denominator, exp(max(lw)) will be canceled out
    lw.scaled <- lw - max(lw)


    ## Finally, calculate the importance sampling estimate
    theta.hat <- mean(theta * lw.scaled) / mean(lw.scaled)


    ## Now, evaluate the importance function using the coefficient of variance
    c.v <- sd(lw.scaled)/mean(lw.scaled)


    return(c(theta.hat,c.v))
}


## consider a grid of c values
c.candidate <- seq(-10, 10, length=500)


## specify vectors to store theta.hat and c.v for each value of c.candidate
hats <- rep(NA, 500)
cvs_hats <- rep(NA, 500)


## calculate estimate theta.hat and coef of var for each value of c.candidate
set.seed(12345)
vals <- seq(-10, 10, length.out = 500)
for(i in 1:500) {

    OUT <- IS(X = vals, c = c.candidate[i], 200)
    hats[i] <- OUT[1]
    cvs_hats[i] <- OUT[2]

}


## Examine where coef. of variance is low

plot(c.candidate, cvs_hats, ylab="coefficient of variance for mu from importance sampling", xlab="c", c


## target mean estimator

plot(c.candidate, hats, xlab="c", ylab="E(X) estimate", col=4, type="l")
```

```r
set.seed(12345)
N <- 200

# Rejection sampling

val <- seq(-10,10,by=0.01)
p.star <- exp(posterior(val))
plot(val, p.star, type = 'l', ylim = c(0,1.5), xlim = c(-10,10))

M.cand <- c(13:18)
for(i in 1:length(M.cand)){
  lines(val,dnorm(val, mean = 0, sd = 5)*M.cand[i],col=i+1,lty=2)
}
legend('topright',legend=paste0('M=',as.character(M.cand)),col=2:5,lty=2,cex=0.5)

RS <- function(M,N) {

  val.samples <- NULL

  while(length(val.samples) < N) {

    val.star <- rnorm(200, mean = 0, sd = 5)
    u <- runif(1)
    ratio <- posterior(val.star) / (M * dnorm(val.star, mean = 0, sd = 5))

    if(u<ratio) val.samples <- c(val.samples,val.star)
  }

  return(val.samples)
}


M <- 15
val.samples <- suppressWarnings(RS(M, N))


hist(val.samples, freq = FALSE, main = "N(0,4.2) by rejection sampling",xlab = expression(val))
lines(val, dnorm(val, mean = 0, sd = 4.2), col = 2)
```