Stat 570 Midterm

Dongyang Wang

2022-11-10

Q1

a

The log likelihood for Binomial distribution is $l = \log(nC_y) + y\log(p) + (n-y)\log(1-p)$. Taking derivative, we have $\frac{y}{p} - \frac{N-y}{1-p} = 0$. Solving, we have $\hat{p} = \frac{y}{N}$.

By the invariance of MLE, $\hat{\pi} = \frac{\hat{p}-(1-\gamma)}{\delta+\gamma-1} = \frac{y-N(1-\gamma)}{N(\delta+\gamma-1)}$.

b

Since the score function for binomial distribution is $\frac{y}{p} - \frac{N-y}{1-p}$, we take derivative wrt p and get $-\frac{y}{p^2} - \frac{N-y}{(1-p)^2}$ and the FIM is $I = -E(-\frac{y}{p^2} - \frac{N-y}{(1-p)^2}) = \frac{N}{p(1-p)}$. By asymptotic distribution of MLE, $Var(\hat{p}) = \frac{\hat{p}(1-\hat{p})}{N}$.The variance for $\hat{\pi}$ is therefore $Var(\hat{\pi}) = Var(\frac{\hat{p}-(1-\gamma)}{\delta+\gamma-1}) = Var(\frac{\hat{p}}{\delta+\gamma-1}) = \frac{1}{(\delta+\gamma-1)^2}Var(\hat{p}) = \frac{1}{(\delta+\gamma-1)^2}\frac{\hat{p}(1-\hat{p})}{N} = \frac{\hat{p}(1-\hat{p})}{N(\delta+\gamma-1)^2}$. By asymptotics of MLE, $I_{\hat{\pi}} = \frac{N(\delta+\gamma-1)^2}{\hat{p}(1-\hat{p})}$.

From the question, we know $\pi_0 = \frac{p+\gamma-1}{\delta+\gamma-1}$ As for the Wald test, the 90% Wald CI for $\hat{p}$ by definition will be $[\hat{p} + z_{0.05} * \sqrt{\frac{\hat{p}(1-\hat{p})}{N}}, \hat{p} + z_{0.95} * \sqrt{\frac{\hat{p}(1-\hat{p})}{N}}]$.

As for $\hat{\pi}$, the 90% CI will be $[\hat{\pi} + z_{0.05} * \sqrt{\frac{\hat{p}(1-\hat{p})}{N(\delta+\gamma-1)^2}}, \hat{\pi} + z_{0.95} * \sqrt{\frac{\hat{p}(1-\hat{p})}{N(\delta+\gamma-1)^2}}]$.

c

Based on information from a and b, we can calculate the MLE as $\hat{\pi} = \frac{y-N(1-\gamma)}{N(\delta+\gamma-1)}$ and based on calculation in r it is 0.0125975.

```
## [1] 0.0125975
```

We can also calculate the 90% confidence interval for $\hat{\pi}$ and obtain [0.00823721, 0.01695780].

```
## [1] 0.00823721 0.01695780
```

d

Also by the invariance of MLE, $\hat{\theta} = \log(\frac{\hat{\pi}}{1-\hat{\pi}}) = \log(\frac{y-N(1-\gamma)}{N\delta+2N\gamma-2N-y})$

$\frac{d\pi}{d\theta} = \frac{e^{\theta}}{(e^{\theta}+1)^2}$.

Using the delta method on FIN, we obtain $I_{\hat{\theta}} = I_{\hat{\pi}} * |\frac{d\pi}{d\theta}|^2 = \frac{N(\delta+\gamma-1)^2}{\hat{p}(1-\hat{p})} * \frac{e^{2\hat{\theta}}}{(e^{\hat{\theta}}+1)^4}$.

```
## [1] -4.361579
```

```
## [1] 22.01813
```

Also, by numerical calculation, we have $\hat{\theta} = -4.361579$ and the information matrix being the inverse of variance and is 22.01813.

e

Yes, we might fit a Beta Binomial model to overcome over-dispersion. As we will see in Q2, adding the Beta distribution to the Binomial distribution, we have a compound distribution that can help with identifying the extra part of overdispersion in the variance expression, as shown in Jon's book Section 3.7.

Q2

a

Since the pdf of π is Beta(1,1), it is equivalent to Unif(0,1). and f(π) = 1 on [0,1]. Therefore, by Jacobian, $\pi(\theta) = \frac{e^\theta}{(e^\theta+1)^2}$.

Plugging in the values for π and θ,

$p(y|\theta) = \binom{n}{y}((\delta+\gamma-1)\frac{e^\theta}{e^\theta+1} - \gamma + 1)^y(\gamma - (\delta+\gamma-1)\frac{e^\theta}{e^\theta+1})^{n-y}$

$p(\theta|y) = \frac{p(y|\theta)\pi(\theta)}{p(y)} \propto p(y|\theta)\pi(\theta) = \binom{n}{y}((\delta+\gamma-1)\frac{e^\theta}{e^\theta+1} - \gamma + 1)^y(\gamma - (\delta+\gamma-1)\frac{e^\theta}{e^\theta+1})^{n-y}\frac{e^\theta}{(e^\theta+1)^2}$.

b

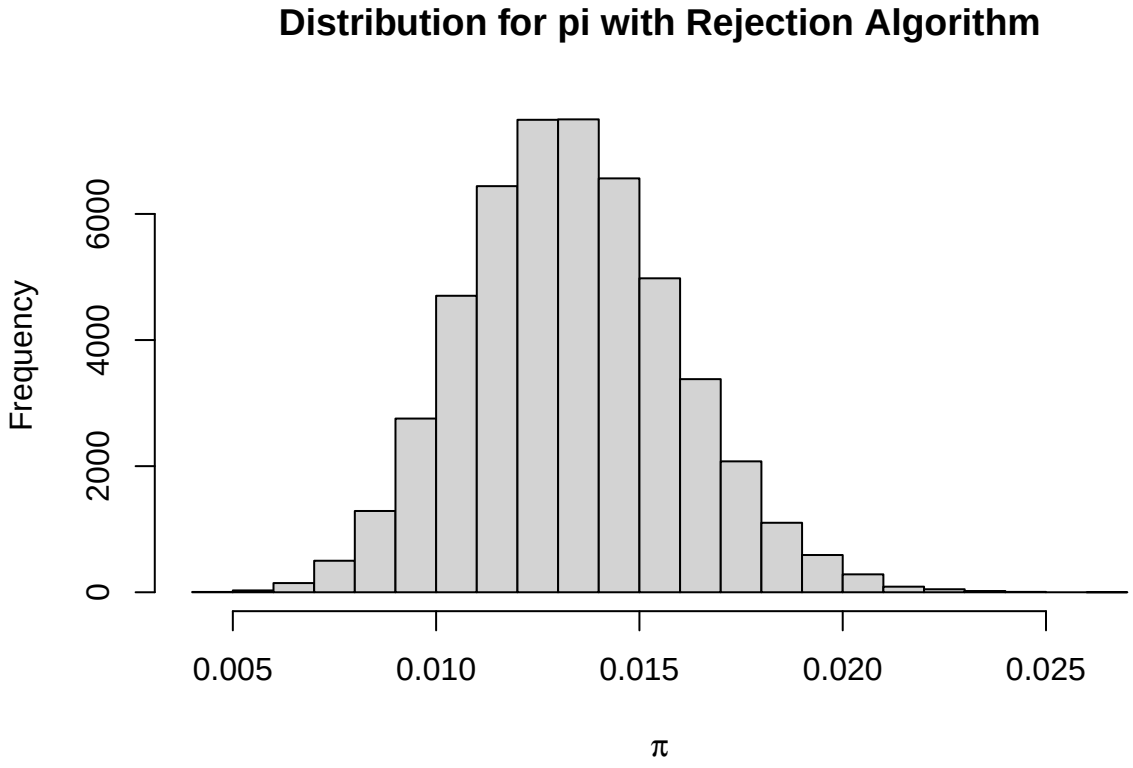
I will use the MLE estimate and variance as the the center and scale parameters, since they provide a good estimate of the true distribution. As we can observe below, the choice of m = 4 and m =5 will render nearly identical results.

```
## G-H no of 4 pts p(y):  0.0003776999

## G-H no of 4 pts mean:  -4.353621

## G-H no of 4 pts variance:  0.04419235

## G-H no of 5 pts p(y):  0.0003776999

## G-H no of 5 pts mean:  -4.353644

## G-H no of 5 pts variance:  0.04587609
```

c

By the rejection algorithm, the results are obtained as follows:

```
##                                  Estimate
## Normalizing Constant Rejection -0.02983475
## Posterior Mean Rejection       -4.31048637
## Posterior Variance Rejection    0.04227743
```



d

Applying the mean and variance for t distribution as specified we will obtain the following metrics.

```
##
## Attaching package: 'metRology'
```

```
## The following objects are masked from 'package:base':
##
##     cbind, rbind
```

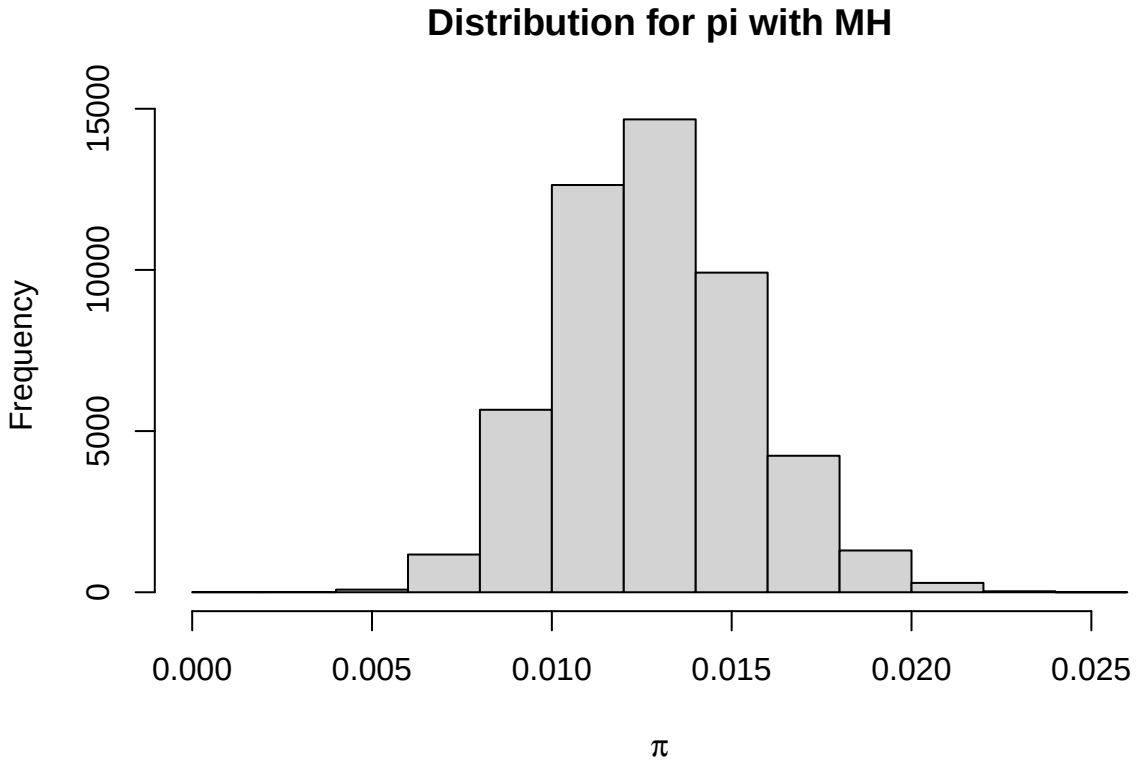
```
##                           Estimate  Lower 90% CI  Upper 90% CI
## Normalizing Constant  0.0003775609  0.0003752091  0.0003799127
## Posterior Mean       -4.3545188794 -4.4429666881 -4.2660710706
## Posterior Variance    0.0460716114 -0.0001660423  0.0923092652
```

The results are shown in the above table.

e

I chose the proposal variance to be the theta MLE variance since it seems a good estimate.

```
##                            Estimate
## Posterior Mean MH     -4.35164360
## Posterior Variance MH  0.04745552
```



f

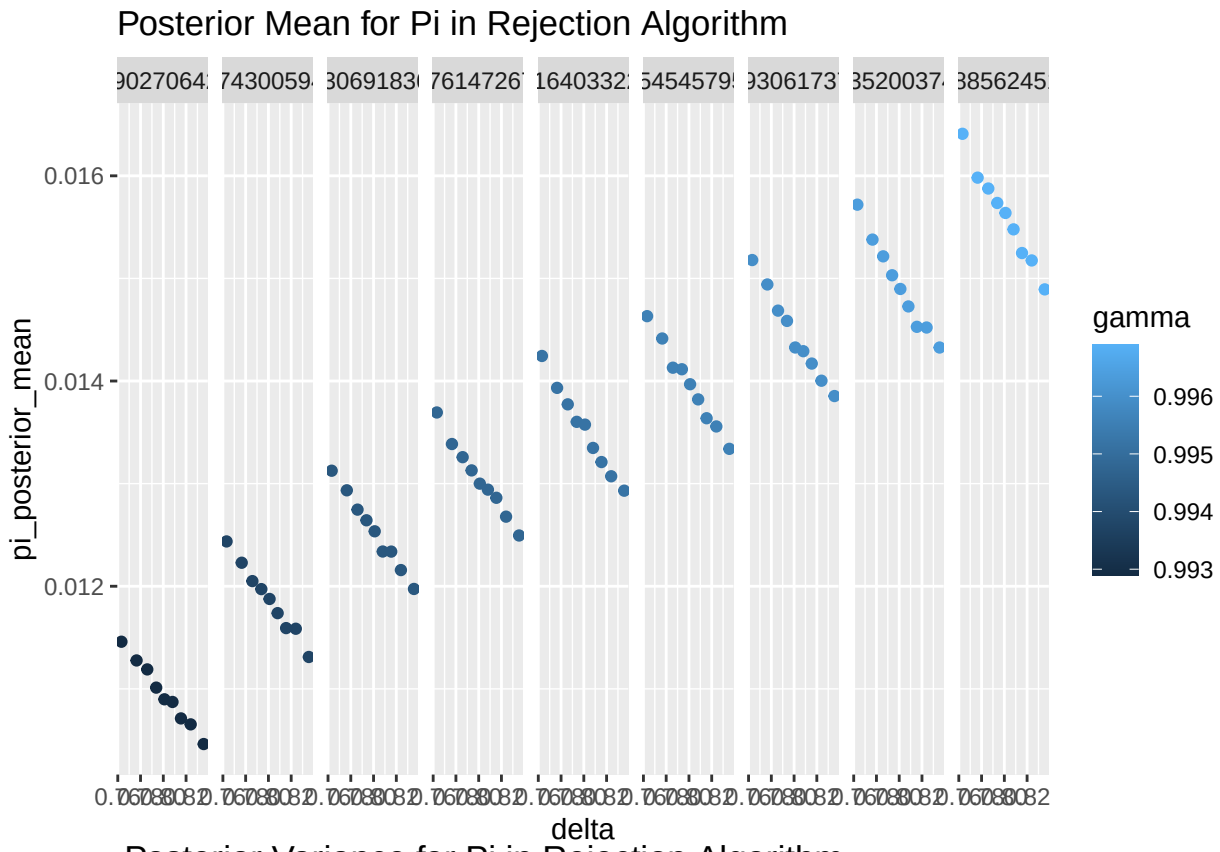
In this section, I have edited the function get_loglik to include gamma and delta as variables. I have generated the gammas and deltas from 10th quantile to 90th quantile, increasing by 10 and obtaining 9 unique values based on the distributions. Therefore, i obtain 9x9=81 pairs of values for experimentation with the rejection algorithm and the MH algorithm, respectively, for investigating the impact of gamma and delta being variables.

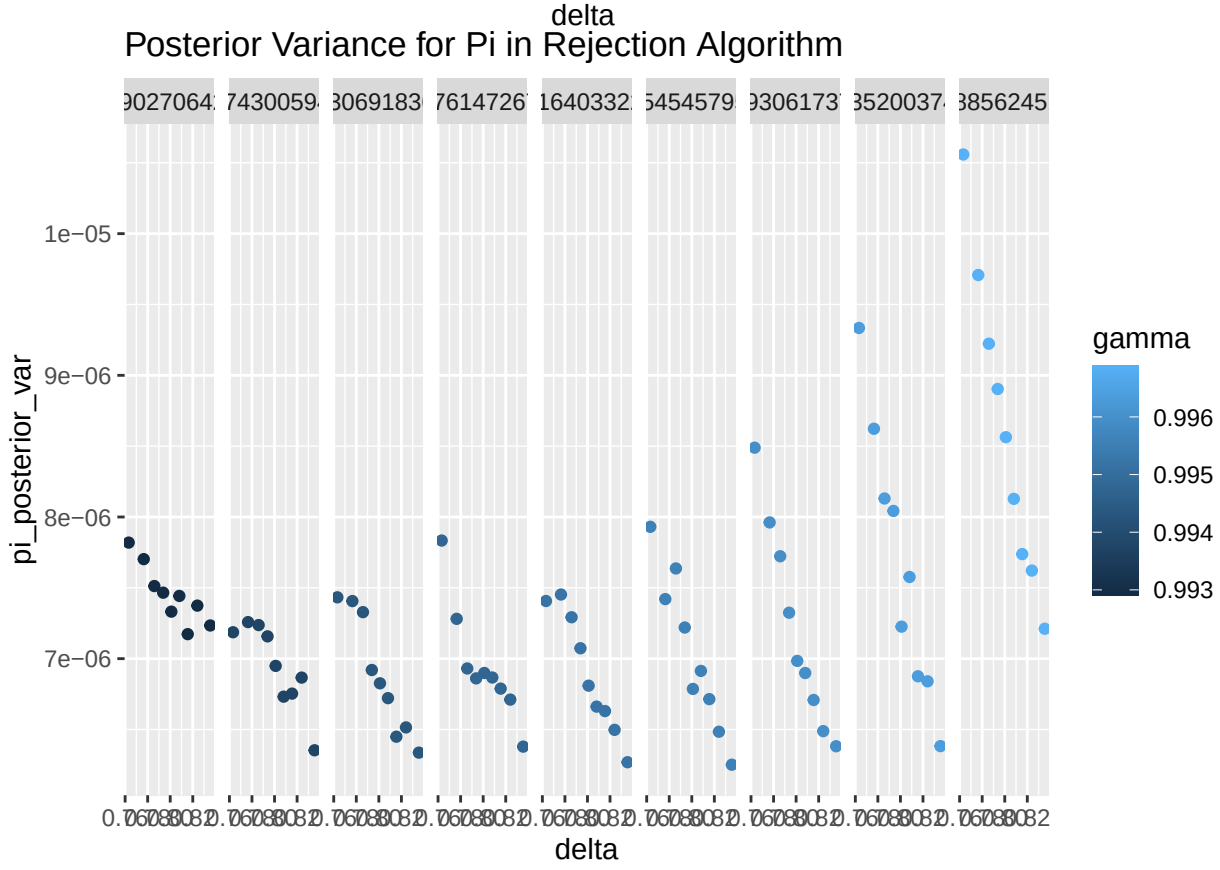
The plots below will show the changes.

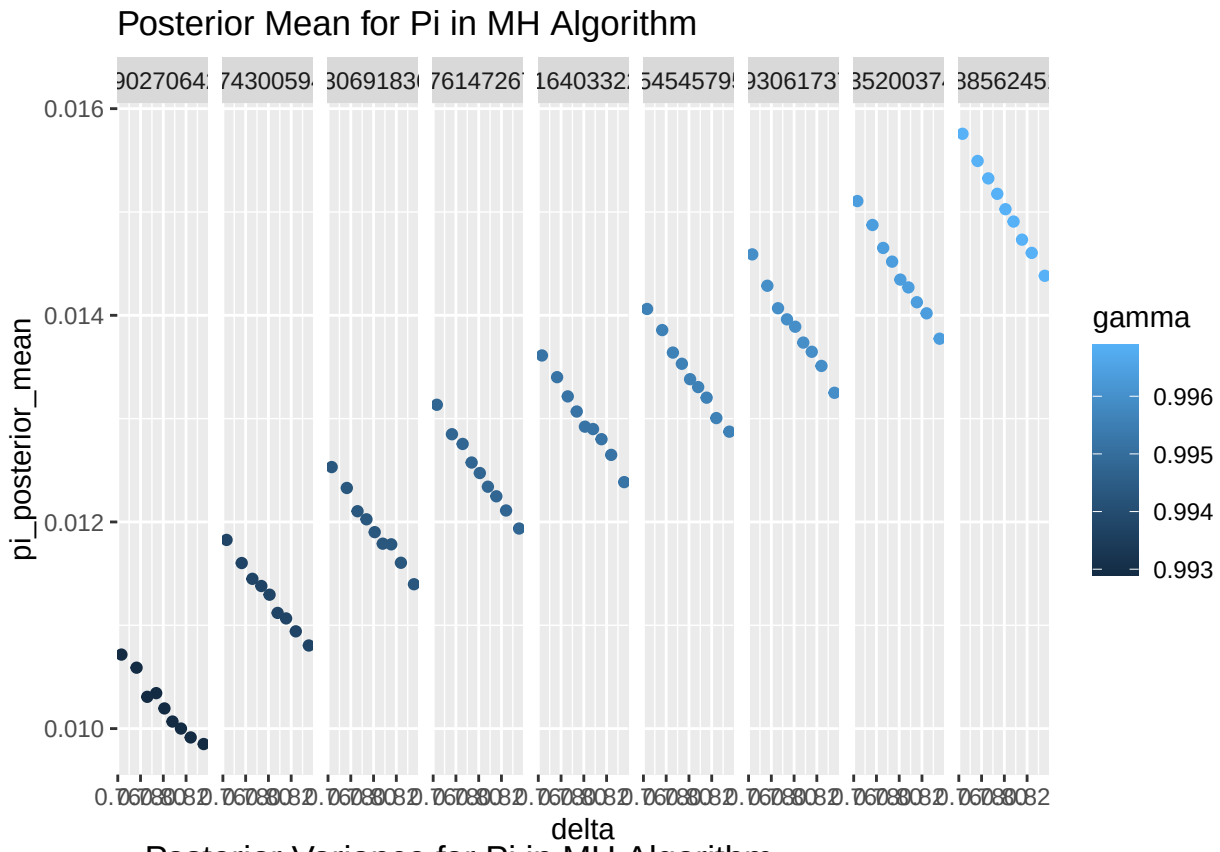
Based on the results below, we can conclude that the distribution of the posterior for π is indeed associated with the values of γ, δ. To be specific, as γ increases, the posterior mean of π will decrease and the posterior variance of π will decrease first but the results show an uptick when we increase γ too much. Also, as δ increases, both the posterior mean of π and the posterior variance of π will decrease.

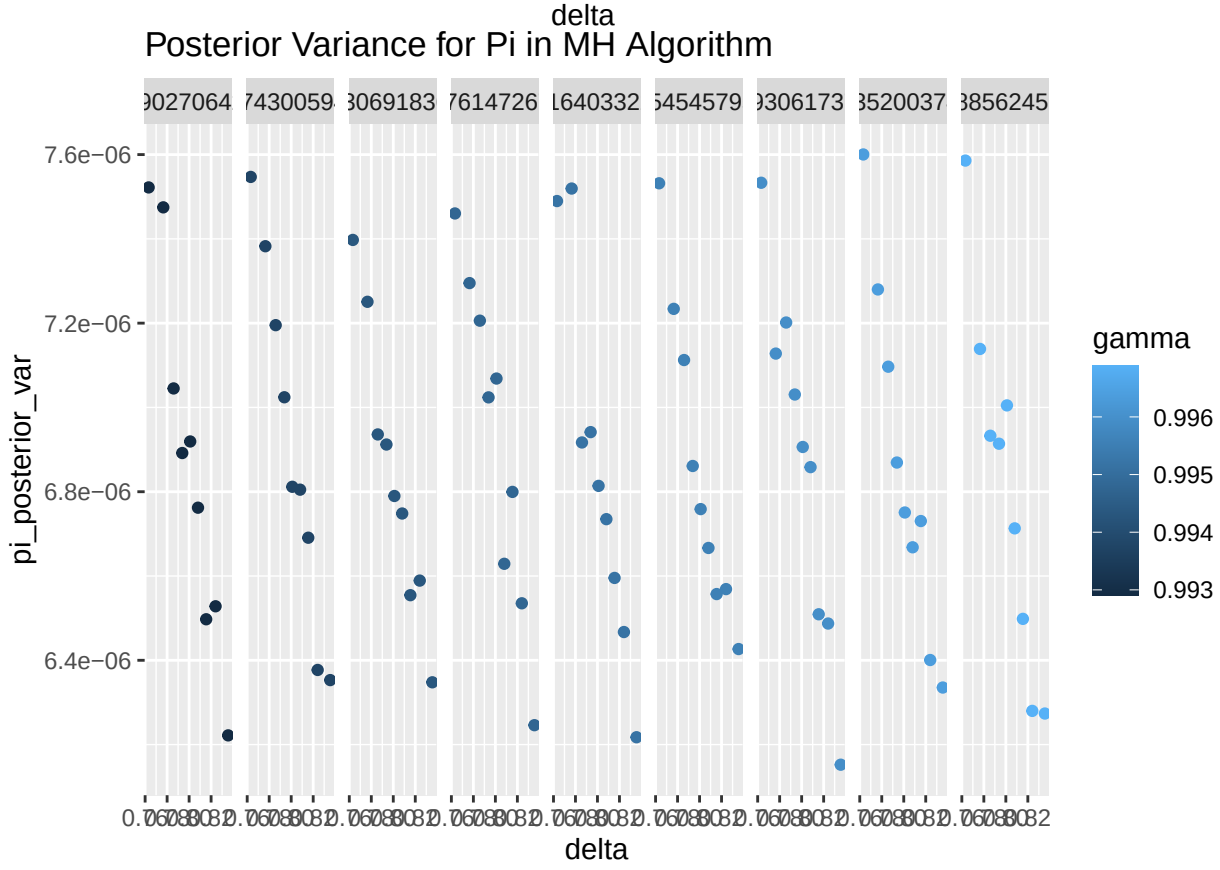
Across the two algorithms, the results show similar trends but also exhibit some differences. For the rejection algorithm, we can observe that the variance of the posterior variance decreases as γ gets large. And there are quite a few points, when γ is small, where the variance gets higher compared with other points. The MH algorithm does not show this trend. One possible explanation is that smaller γ and δ will make the rejection more likely, biasing the result. By contrast, the variance of the posterior variance in the MH algorithm does not change much.

```
## -- Attaching packages --------------------------------------- tidyverse 1.3.2 --
## v ggplot2 3.4.0      v purrr   0.3.5
## v tibble  3.1.8      v dplyr   1.0.10
## v tidyr   1.2.1      v stringr 1.4.1
## v readr   2.1.3      v forcats 0.5.2
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```


Posterior Mean for Pi in Rejection Algorithm
pi_posterior_mean
0.016
0.014
0.012
delta
gamma
0.996
0.995
0.994
0.993


Posterior Variance for Pi in Rejection Algorithm
pi_posterior_var
1e−05
9e−06
8e−06
7e−06
delta
gamma
0.996
0.995
0.994
0.993


Posterior Mean for Pi in MH Algorithm
pi_posterior_mean
0.016
0.014
0.012
0.010
delta
gamma
0.996
0.995
0.994
0.993


Posterior Variance for Pi in MH Algorithm
pi_posterior_var
7.6e−06
7.2e−06
6.8e−06
6.4e−06
delta
gamma
0.996
0.995
0.994
0.993

Appendix

Q1

```
rm(list = ls())
delta = 0.8
gamma = 0.995
N = 3330
y = 50

# calculate mle and var for pi and theta for future use
p_mle = y/N
p_var = (1-p_mle)*p_mle/N

pi_mle = (y-N*(1-gamma))/((delta+gamma-1)*N)
pi_var = p_mle*(1-p_mle) /((delta+gamma-1)^2*N)

theta_mle = log(pi_mle/(1-pi_mle))
theta_var = pi_var * ((exp(theta_mle) + 1)^4/exp(2*theta_mle))

pi_mle

lower_pi_ci = pi_mle +qnorm(0.05)*sqrt(pi_var)
upper_pi_ci = pi_mle +qnorm(0.95)*sqrt(pi_var)
c(lower_pi_ci, upper_pi_ci)
```

Q2

```
################ Q2.b ################
get_loglik <- function(theta){
  c <- choose(N,y)
  part1 <- (gamma + delta - 1)*exp(theta)/(exp(theta)+1) -gamma +1
  part2 <- 1 - part1
  part3 <- exp(theta)
  part4 <- exp(theta)+1
  res <- log(c) + y*log(part1) + (N-y) * log(part2) + log(part3) - 2*log(part4)
  return(res)
}

library(statmod)
m <- 4 # no of numerical integration points to use
quad <- gauss.quad(m,kind="hermite")
ghest0 <- 0
ghest1 <- 0
ghest2 <- 0

for (i in 1:m){
    theta <- theta_mle + sqrt(2*theta_var)*quad$nodes[i]
    ghest0 <- ghest0 +
                  quad$weights[i]*(1/dnorm(theta,theta_mle,sqrt(theta_var)))*
                  exp(get_loglik(theta))/sqrt(pi)
    ghest1 <- ghest1 +
                  quad$weights[i]*(1/dnorm(theta,theta_mle,sqrt(theta_var)))*
                  exp(get_loglik(theta))*theta/sqrt(pi)
```

```
    ghest2 <- ghest2 +
                  quad$weights[i]*(1/dnorm(theta,theta_mle,sqrt(theta_var)))*
                  exp(get_loglik(theta))*theta^2/sqrt(pi)
}
cat("G-H no of 4 pts p(y): ",ghest0,"\n")
ghmean <- ghest1/ghest0
cat("G-H no of 4 pts mean: ", ghmean,"\n")
cat("G-H no of 4 pts variance: ",ghest2/ghest0-ghmean^2,"\n")

m2 <- 5 # no of numerical integration points to use
quad <- gauss.quad(m2,kind="hermite")
m2_ghest0 <- 0
m2_ghest1 <- 0
m2_ghest2 <- 0

for (i in 1:m2){
    theta <- theta_mle + sqrt(2*theta_var)*quad$nodes[i]
    m2_ghest0 <- m2_ghest0 +
                  quad$weights[i]*(1/dnorm(theta,theta_mle,sqrt(theta_var)))*
                  exp(get_loglik(theta))/sqrt(pi)
    m2_ghest1 <- m2_ghest1 +
                  quad$weights[i]*(1/dnorm(theta,theta_mle,sqrt(theta_var)))*
                  exp(get_loglik(theta))*theta/sqrt(pi)
    m2_ghest2 <- m2_ghest2 +
                  quad$weights[i]*(1/dnorm(theta,theta_mle,sqrt(theta_var)))*
                  exp(get_loglik(theta))*theta^2/sqrt(pi)
}
cat("G-H no of 5 pts p(y): ",ghest0,"\n")
m2_ghmean <- m2_ghest1/m2_ghest0
cat("G-H no of 5 pts mean: ",m2_ghmean,"\n")
cat("G-H no of 5 pts variance: ",m2_ghest2/m2_ghest0-m2_ghmean^2,"\n")

############### Q2.c ###############

set.seed(42)

get_loglik <- function(theta){
  c <- choose(N,y)
  part1 <- (gamma + delta - 1)*exp(theta)/(exp(theta)+1) -gamma +1
  part2 <- 1 - part1
  part3 <- exp(theta)
  part4 <- exp(theta)+1
  res <- log(c) + y*log(part1) + (N-y) * log(part2) + log(part3) - 2*log(part4)
  return(res)

  #return(c*part1^y*part2^(N-y)*part3)
}

theta_loglik <- get_loglik(theta_mle)

accept <- 0
counter <- 0
n <- 50000 # No of samples to generate from the posterior
```

```r
thetasamp <- rep(0,n)

while (accept < n) {

  counter <- counter + 1

  u <- runif(1,0,1)
  pi <- rbeta(1,1,1)
  theta <- log(pi/(1-pi))

  test <- get_loglik(theta) - (theta_loglik)

    if (log(u) < test ) {
        accept <- accept + 1
        thetasamp[accept] <- theta
    }
}

rej_norm_constant <- n/counter * theta_mle
rej_posterior_mean <- mean(thetasamp)
rej_posterior_var <- sd(thetasamp)^2

result_c = matrix(c(rej_norm_constant, rej_posterior_mean, rej_posterior_var),
                  nrow = 3, byrow = T)
colnames(result_c) = c('Estimate')
rownames(result_c) = c('Normalizing Constant Rejection',
                       'Posterior Mean Rejection', 'Posterior Variance Rejection')
result_c

rej_pi <- exp(thetasamp)/(1+exp(thetasamp))^2
hist(rej_pi, main="Distribution for pi with Rejection Algorithm",
     xlab=expression(pi))

############### Q2.d ###############

library(metRology)

importance_sampling <- function(n = 5000){
  n <- 5000 # pts to use to evaluate integrals
  df = 4
  thsampn <- rt.scaled(n, df, mean = theta_mle, sd = sqrt(theta_var))

  n0vals <- exp(get_loglik(thsampn))/
    dt.scaled(thsampn, df, mean = theta_mle, sd = sqrt(theta_var))
  n1vals <- thsampn*exp(get_loglik(thsampn))/
    dt.scaled(thsampn, df, mean = theta_mle, sd = sqrt(theta_var))
  n2vals <- thsampn^2*exp(get_loglik(thsampn))/
    dt.scaled(thsampn, df, mean = theta_mle, sd = sqrt(theta_var))

  impn0est <- mean(n0vals)
  varn0est <- var(n0vals)/n
  impn1est <- mean(n1vals)
  varn1est <- var(n1vals)/n
```

```
  covn0n1est <- cov(n0vals,n1vals)/n
  covn0n2est <- cov(n0vals,n2vals)/n
  covn1n2est <- cov(n1vals,n2vals)/n

  #cat("p(y) Imp samp = ", impn0est, ":  95% Interval: ",
  #impn0est-1.96*sqrt(varn0est),impn0est+1.96*sqrt(varn0est),"\n")

  pmeannest <- impn1est/impn0est
  pmeannse <- sqrt( varn0est*impn1estˆ2/impn0estˆ4 +
                      varn1est/impn0estˆ2 -   2*covn0n1est*impn1est/impn0estˆ3 )

  #cat("Post mean Imp samp = ", pmeannest, ":  95% Interval for r=1: ",pmeannest-1.96*pmeannse,pmeannes

  impn2est <- mean(n2vals)
  varn2est <- var(n2vals)/n
  pvarnest <- impn2est/impn0est - impn1estˆ2/impn0estˆ2

  # Derviatives for delta method on var of var
  deriv0 <- -impn2est/impn0estˆ2  + 2*impn1estˆ2/impn0estˆ3
  deriv1 <- -2*impn1est/impn0estˆ2
  deriv2 <- 1/impn0est

  pvarnse <- sqrt( deriv0ˆ2*varn0est + deriv1ˆ2*varn1est +
                     deriv2ˆ2*varn2est +2*deriv0*deriv1*covn0n1est +
                     2*deriv0*deriv2*covn0n2est + 2*deriv1*deriv2*covn1n2est )

  #cat("Post var Imp samp = ", pvarnest, ":  95% Interval for r=2:",
  #pvarnest-1.96*pvarnse,pvarnest+1.96*pvarnse,"\n")

  return (matrix(c(impn0est, impn0est+qnorm(0.05)*sqrt(varn0est),
                   impn0est+qnorm(0.95)*sqrt(varn0est),
                   pmeannest,pmeannest+qnorm(0.05)*sqrt(pmeannse),
                   pmeannest+qnorm(0.95)*sqrt(pmeannse),
                   pvarnest, pvarnest+qnorm(0.05)*sqrt(pvarnse),
                   pvarnest+qnorm(0.95)*sqrt(pvarnse)), nrow = 1))
}

res = replicate(1000, importance_sampling(5000))
result_d = apply(res, 2, mean)
result_d = matrix(result_d, nrow = 3, byrow = T)
colnames(result_d) = c('Estimate', 'Lower 90% CI', 'Upper 90% CI')
rownames(result_d) = c('Normalizing Constant', 'Posterior Mean',
                       'Posterior Variance')
result_d

############### Q2.e ###############

set.seed(42)
n <- 50000 # no of iterations
thetaMH <- NULL

thetaMH[1] <- theta_loglik
accept <- 0
```

```
for (i in 2:n){
       thetaMH[i] <- thetaMH[i-1]
       thetaprop <- rnorm(1,m=thetaMH[i-1],sd=sqrt(theta_var))
       log_test <- get_loglik(thetaprop) - get_loglik(thetaMH[i-1])

       if (log(runif(1)) < log_test){
               thetaMH[i] <- thetaprop;
               accept <- accept+1
        }
}

MH_posterior_mean <- mean(thetaMH)
MH_posterior_var <- sd(thetaMH)^2

MH_pi <- exp(thetaMH)/(1+exp(thetaMH))^2

result_e = matrix(c(MH_posterior_mean, MH_posterior_var), nrow = 2, byrow = T)
rownames(result_e) = c('Posterior Mean MH', 'Posterior Variance MH')
colnames(result_e) = c('Estimate')
result_e

hist(MH_pi, main="Distribution for pi with MH",xlab=expression(pi))

############### Q2.f ###############

set.seed(42)
gammas = deltas = seq(0.1, 0.9, 0.1)

get_loglik <- function(theta, gamma, delta){
  c <- choose(N,y)
  part1 <- (gamma + delta - 1)*exp(theta)/(exp(theta)+1) -gamma +1
  part2 <- 1 - part1
  part3 <- exp(theta)
  part4 <- exp(theta)+1
  res <- log(c) + y*log(part1) + (N-y) * log(part2) + log(part3) - 2*log(part4)
  return(res)
}

rejection <- function(gamma, delta){

  theta_loglik <- get_loglik(theta_mle, gamma, delta)

  accept <- 0
  counter <- 0
  n <- 5000 # No of samples to generate from the posterior
  thetasamp <- rep(0,n)

  while (accept < n) {

    counter <- counter + 1

    u <- runif(1,0,1)
    pi <- rbeta(1,1,1)
```

```
    theta <- log(pi/(1-pi))

    test <- get_loglik(theta, gamma, delta) - theta_loglik

    if (log(u) < test ) {
        accept <- accept + 1
        thetasamp[accept] <- theta
    }
  }

  rej_norm_constant <- n/counter * theta_mle
  rej_posterior_mean <- mean(thetasamp)
  rej_posterior_var <- sd(thetasamp)^2

  rej_pi <- exp(thetasamp)/(1+exp(thetasamp))^2

  data_temp = data.frame(matrix(c(1), nrow = 1))
  data_temp['delta'] = delta
  data_temp['gamma'] = gamma
  data_temp['pi_posterior_mean'] = mean(rej_pi)
  data_temp['pi_posterior_var'] = var(rej_pi)

  data_temp = data_temp[,-1]

  return(data_temp)
}

MH <- function(gamma, delta){
  n <- 50000 # no of iterations
  thetaMH <- NULL

  theta_loglik <- get_loglik(theta_mle, gamma, delta)
  thetaMH[1] <- theta_loglik
  accept <- 0

  for (i in 2:n){
     thetaMH[i] <- thetaMH[i-1]
     thetaprop <- rnorm(1,m=thetaMH[i-1],sd=sqrt(theta_var))
     log_test <- get_loglik(thetaprop, gamma, delta) -
       get_loglik(thetaMH[i-1], gamma, delta)

     if (log(runif(1)) < log_test){
           thetaMH[i] <- thetaprop;
           accept <- accept+1
      }
  }

  MH_posterior_mean <- mean(thetaMH)
  MH_posterior_var <- sd(thetaMH)^2
  MH_pi <- exp(thetaMH)/(1+exp(thetaMH))^2

  data_temp = data.frame(matrix(c(1), nrow = 1))
  data_temp['delta'] = delta
```

```
  data_temp['gamma'] = gamma
  data_temp['pi_posterior_mean'] = mean(MH_pi)
  data_temp['pi_posterior_var'] = var(MH_pi)

  data_temp = data_temp[,-1]

  return(data_temp)
}

data_f_rej = data.frame()
data_f_MH = data.frame()

for (i in 1:9){
  delta_var = qbeta(deltas[i],160,40)
  for (j in 1:9){
     gamma_var = qbeta(gammas[j],1990,10)
     data_f_rej = rbind(data_f_rej, rejection(gamma_var, delta_var))
     data_f_MH = rbind(data_f_MH, MH(gamma_var, delta_var) )

  }
}

#MH(gamma_var, delta_var)
#data_f_rej

library(tidyverse)

ggplot(data=data_f_rej, aes(x= delta, y=pi_posterior_mean, color= gamma)) +
  geom_point() +
  facet_grid(.~gamma,scales="free") +
  labs(title = "Posterior Mean for Pi in Rejection Algorithm")

ggplot(data=data_f_rej, aes(x= delta, y=pi_posterior_var, color= gamma)) +
  geom_point() +
  facet_grid(.~gamma,scales="free") +
  labs(title = "Posterior Variance for Pi in Rejection Algorithm")

ggplot(data=data_f_MH, aes(x= delta, y=pi_posterior_mean, color= gamma)) +
  geom_point() +
  facet_grid(.~gamma,scales="free") +
  labs(title = "Posterior Mean for Pi in MH Algorithm")

ggplot(data=data_f_MH, aes(x= delta, y=pi_posterior_var, color= gamma)) +
  geom_point() +
  facet_grid(.~gamma,scales="free") +
  labs(title = "Posterior Variance for Pi in MH Algorithm")
```