Markov Chain Monte Carlo

Wei Deng

Oct.10, 2017

Theory

Consider observations $y_1, ..., y_n | N, \theta \overset{\text{i.i.d.}}{\sim} Binomial(N, \theta)$, where both N and θ are unknown parameters that are to be inferred. Raftery (1988) describes one Bayesian approach for this inference. Instead of directly specifying a joint prior on (N, θ), he first specifies $N|\theta, \mu \sim Poisson(\mu)$, then defines $\lambda = \mu\theta$, and finally specifies a prior on (λ, θ). Your task in the Theory and Computation problems is to develop and implement this Bayesian approach.

1. One suggested prior on (λ, θ) is $p(\lambda, \theta) \propto \lambda^{-1}$. Describe the rationale for this choice (hint: transform and determine the induced distribution on (N, θ)).

$p(\lambda, \theta) \propto \frac{1}{\lambda}$

$p(N|\lambda, \theta) = \frac{(\frac{\lambda}{\theta})^N e^{-\frac{\lambda}{\theta}}}{N!}$

$$\begin{aligned} p(N, \theta) &= \int_0^\infty p(N, \theta, \lambda) d\lambda \\ &= \int_0^\infty p(N|\lambda, \theta) p(\lambda, \theta) d\lambda \\ &\propto \frac{1}{\theta^N N!} \int_0^\infty \lambda^{N-1} exp(-\frac{\lambda}{\theta}) d\lambda \\ &\propto \frac{\Gamma(N)\theta^N}{N!\theta^N} \\ &\propto \frac{1}{N} \end{aligned}$$

The rationale to specify the prior distribution in terms of (λ, θ) rather than (μ, θ) is that it would be easier to formulate the prior information about λ, the unconditional expectation of the observations, than about μ, the mean of the unobserved quantity N. The prior information about λ would be more precise than that about μ or θ, so that it may be more reasonable to assume λ and θ independent a priori than μ and θ.

2. Is the suggested prior distribution proper? Why or why not?

The suggested prior is not proper, since $\int_0^1 \int_0^\infty \frac{1}{\lambda} d\lambda d\theta = ln(\lambda)|_0^\infty = \infty$.

3. Is the corresponding posterior density function $p(\lambda, \theta | y_1, ..., y_n)$ proper? Why or why not?

$$\begin{aligned} p(\lambda, \theta | y_1, ..., y_n) &\propto p(y_1, ..., y_n | \lambda, \theta) p(\lambda, \theta) \\ &\propto p(\lambda, \theta) \sum_{N=Y}^\infty p(y_1, ..., y_n | N, \theta) p(N|\lambda, \theta) \ \#\ denote\ max\{y_i\}\ as\ Y \\ &\propto \frac{1}{\lambda} \sum_{N=Y}^\infty \{\prod_{i=1}^n (C_N^{y_i} \theta^{y_i} (1-\theta)^{N-y_i}) \frac{(\frac{\lambda}{\theta})^N e^{-\frac{\lambda}{\theta}}}{N!}\} \ \#\ denote\ sum\{y_i\}\ as\ S \\ &\propto \frac{1}{\lambda} \theta^S (1-\theta)^{nN-S} e^{-\frac{\lambda}{\theta}} \sum_{N=Y}^\infty \frac{(\frac{\lambda}{\theta})^N}{N!} \ \#\ taylor\ expansion \\ &\propto \frac{1}{\lambda} \theta^S (1-\theta)^{nN-S} e^{-\frac{\lambda}{\theta}} (e^{\frac{\lambda}{\theta}} - C) \ \#\ C > 1\ if\ not\ all\ y_i\ are\ 0 \\ &\propto \frac{1}{\lambda} \theta^S (1-\theta)^{nN-S} (1 - Ce^{-\frac{\lambda}{\theta}}) \end{aligned}$$

$$\begin{aligned}
\int_0^\infty \int_0^1 p(\lambda,\theta|y_1,...,y_n)d\lambda d\theta &= \int_0^1 \theta^S(1-\theta)^{nN-S}\int_0^\infty \frac{1}{\lambda}(1-Ce^{-\frac{\lambda}{\theta}})d\lambda d\theta \ \#\ choose\ b\ s.t.\ C\ e^{-\frac{\lambda}{\theta}}=1 \\
&< \int_0^1 \theta^S(1-\theta)^{nN-S}(\int_0^b \frac{1}{\lambda}(1-Ce^{-\frac{\lambda}{\theta}})+\int_b^\infty \frac{1}{\lambda}(1-Ce^{-\frac{\lambda}{\theta}}))d\lambda d\theta \\
&< \int_0^1 \theta^S(1-\theta)^{nN-S}(0+\int_b^\infty \frac{1}{\lambda})d\lambda d\theta \\
&< \int_0^1 \theta^S(1-\theta)^{nN-S}Cd\theta \ \#\ C\ is\ a\ constant \\
&< \infty
\end{aligned}$$

Therefore, the posterior density function $p(\lambda,\theta|y_1,...,y_n)$ is proper.

Computation

```
set.seed(666)
setwd("C:/Users/Wei/Documents/Purdue STAT 695 Bayesian Data Analysis/HW3")
```

1. Develop a MCMC algorithm to sample from $p(N,\theta|y_1,...,y_n)$ using the previously suggested prior. Implement 10 chains of your algorithm on both the waterbuck and impala data sets provided by Raftery (1988, p. 226), and contained in the respective files *impala.txt* and *waterbuck.txt*. Note that for both of these data sets, n = 5. From your chains, construct trace and ACF plots of N and θ, and calculate the Gelman-Rubin statistics and effective sample sizes of N and θ.

Load data first

```
Y1 = read.csv(file="impala.txt", header=FALSE)
Y2 = read.csv(file="waterbuck.txt", header=FALSE)
```

From previous section, derive our posterior density

$$\begin{aligned}
p(N,\theta|y_1,...,y_n) &\propto p(y_1,...,y_n|N,\theta)p(N,\theta) \\
&\propto \frac{1}{N}\prod_{i=1}^n (C_N^{y_i}\theta^{y_i}(1-\theta)^{N-y_i})
\end{aligned}$$

Build our posterior density function

```
posterior = function(pars, Y) { # pars: N, theta, Y
  sum(dbinom(t(Y), pars[1], pars[2], log=TRUE)) - log(pars[1])
}
```

Set up the proposal distribution to generate random samples and compute transition probaiblity

```
library(KScorrect)
proposalFunc = function(par) {
  return(c(rpois(1, par[1]), # poisson distribution
           rbeta(1, 1, 1 / par[2] - 1))) # beta distribution with mean par[2]
}

proposal_p = function(new, par) { # p(new|par)
  return(dpois(new[1], par[1]) * dbeta(new[2], 1, 1 / par[2] - 1))
}
```

Build our **Metropolis Hastings** algorithm

```
library(coda)
```

```
## Warning: package 'coda' was built under R version 3.4.2
```

```
MCMC = function(initialValue, Y, burnIN, iterations) {
  chains = array(dim=c(iterations + 1, 2))
  chains[1, ] = initialValue
```

```
  for (i in 1: iterations) {
    proposal = proposalFunc(chains[i, ])
    accept_p = exp(posterior(proposal, Y) + proposal_p(chains[i, ], proposal)
                   - posterior(chains[i, ], Y) - proposal_p(proposal, chains[i, ]))
    # take care that accept_p = p1 / p2, thus p2 should not equal 0
    if (runif(1) < accept_p)
      chains[i+1, ] = proposal
    else
      chains[i+1, ] = chains[i, ]
  }
  #acceptance = 1 - mean(duplicated(chains)) # hide this information
  #print(paste0("acceptance rate: ", round(acceptance * 100, 4), " %"))
  chains = chains[-(1: burnIn), ]
  chains = data.frame(N=chains[, 1], theta=chains[, 2])
  return(mcmc(chains))
}
```

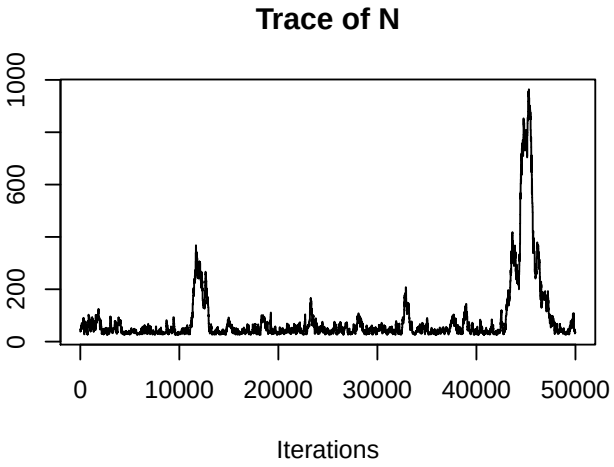
Generate our first markov chain for data *impala.txt*.

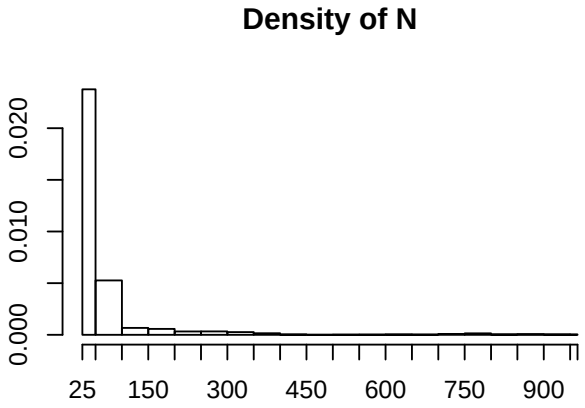
```
initialValue = c(50, 0.5)
iterations = 100000
burnIn = 50000
chains1 = MCMC(initialValue, Y1, burnIn, iterations)
summary(chains1)
```

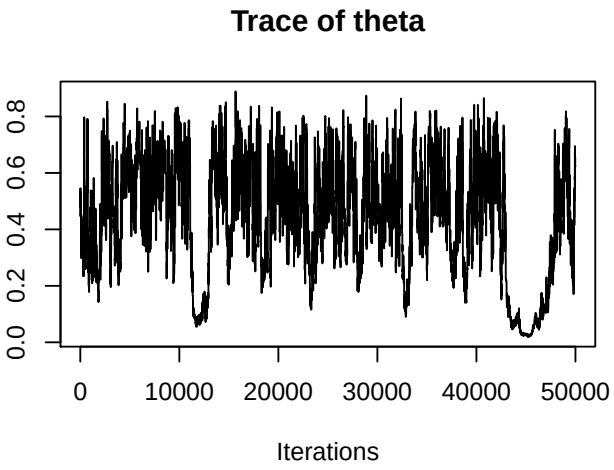
```
##
## Iterations = 1:50001
## Thinning interval = 1
## Number of chains = 1
## Sample size per chain = 50001
##
## 1. Empirical mean and standard deviation for each variable,
##    plus standard error of the mean:
##
##          Mean      SD  Naive SE Time-series SE
## N     83.8523 127.545 0.5703951       47.01528
## theta  0.4536   0.207 0.0009255        0.02017
##
## 2. Quantiles for each variable:
##
##           2.5%     25%     50%     75%    97.5%
## N     27.00000 34.0000 44.0000 68.0000 515.0000
## theta  0.04337  0.3084  0.4739  0.6237   0.7831
```

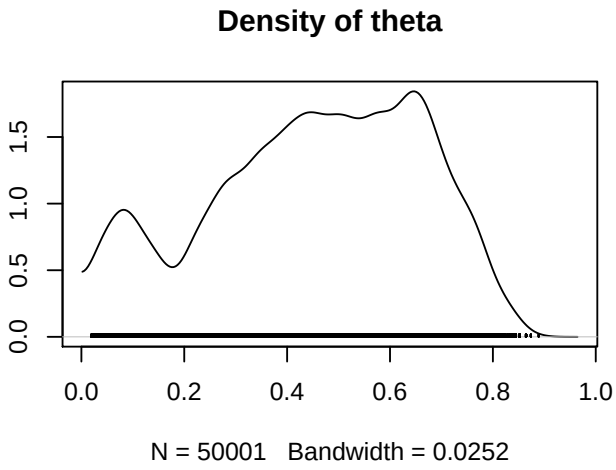
Construct the trace plots

```
plot(chains1)
```



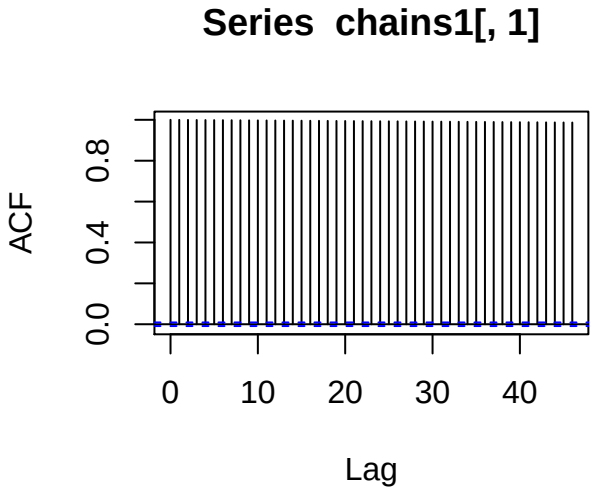


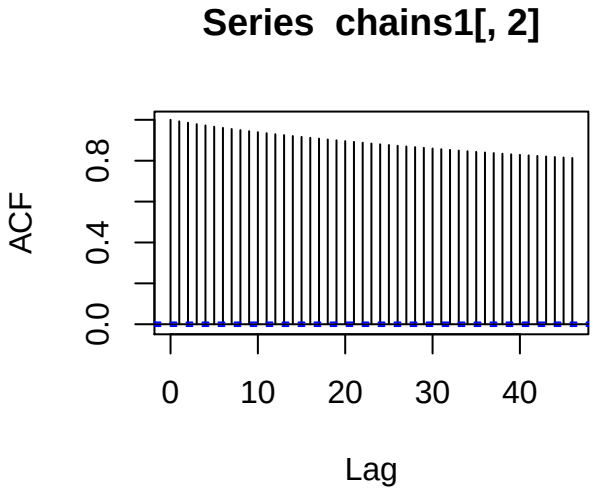




ACF plot: a good MCMC algorithm should generate highly autocorrelated parameter values to traverse the whole sample space of the parameters, which shows a **slow geometric decay**.

```
par(mfrow=c(1,2))
acf(chains1[, 1])
acf(chains1[, 2])
```

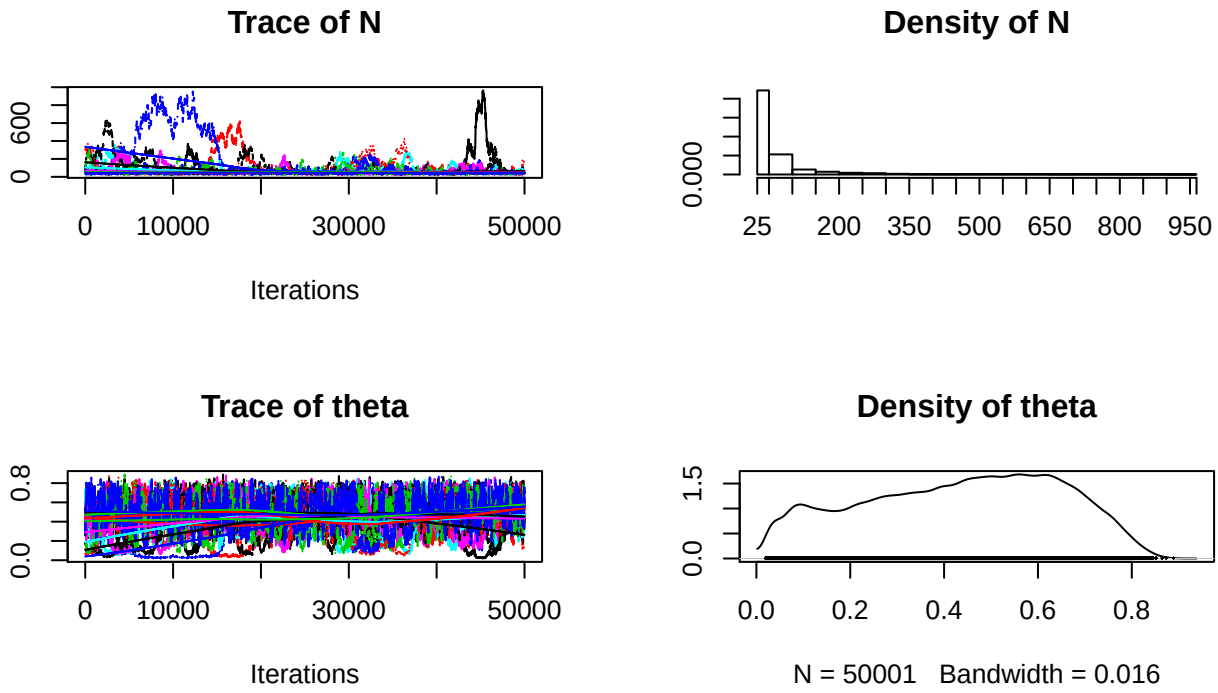




From Gelman-Rubin statistics, it is better for our **Potential scale reduction factor** to be close to 1. We see from Gelman's

diagonisis that the result is acceptable.

```
LIST1 = list(chains1)
for (i in 2: 10) {
  initialValue = c(sample(30: 60, 1), runif(1, 0.1, 0.9))
  chains = MCMC(initialValue, Y1, burnIn, iterations)
  LIST1[[i]] = chains
}
MCMC_list = mcmc.list(LIST1)
plot(MCMC_list)
```




```
gelman.diag(MCMC_list)
```

```
## Potential scale reduction factors:
##
##       Point est. Upper C.I.
## N           1.18       1.36
## theta       1.01       1.03
##
## Multivariate psrf
##
## 1.06
```

```
#gelman.plot(MCMC_list)
```

Large effective sample size doesn't guarantee sufficient/representative MCMC samples, but lower effective sample size does imply insufficient samples. In our case, the effective sample size is not too small which is safe for us to use.

```
effectiveSize(MCMC_list)
```

```
##         N     theta
##  268.6021 1017.3263
```

Repeat the procedure for *waterbuck.txt*.

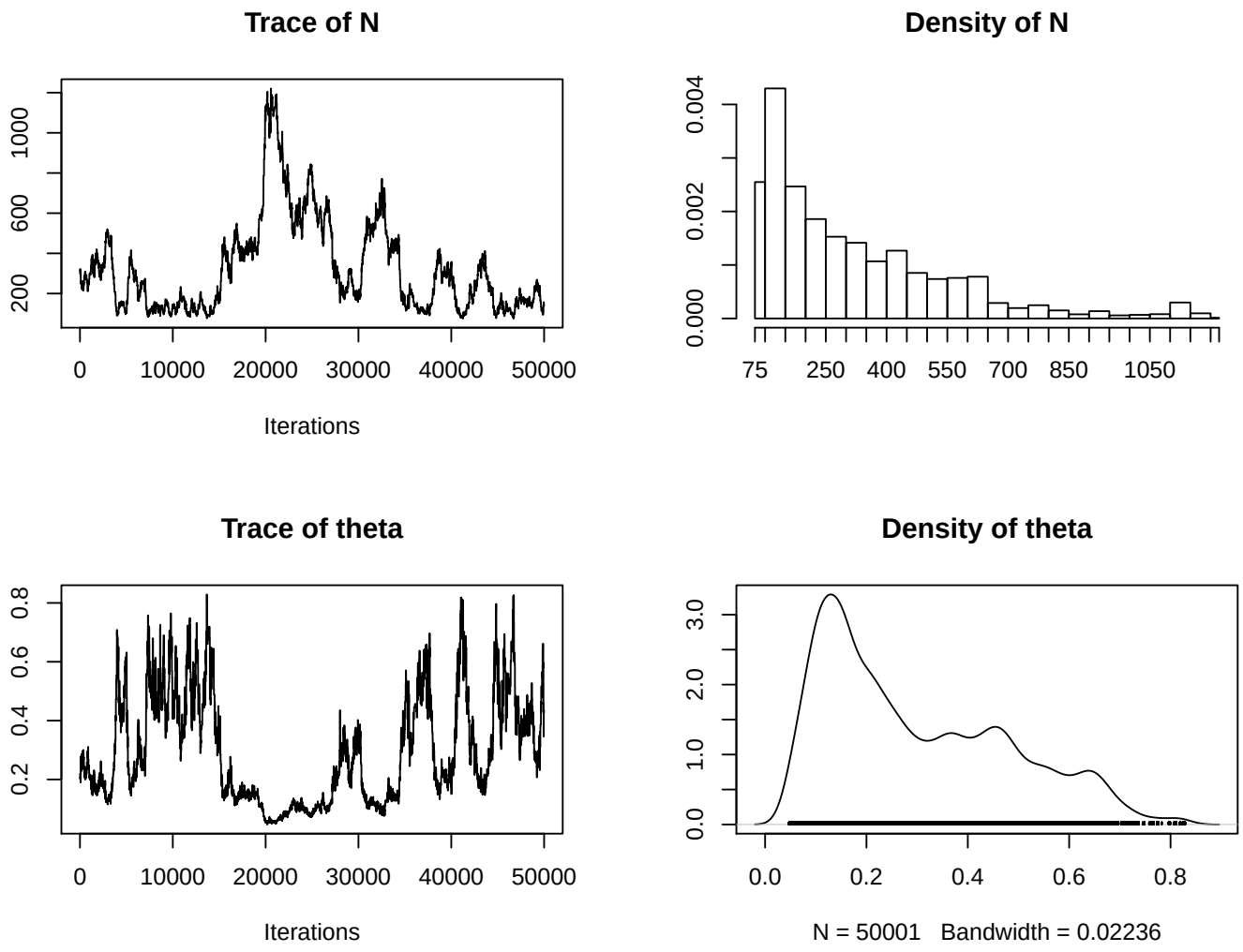
```
initialValue = c(150, 0.5)
iterations = 100000
burnIn = 50000
chains2 = MCMC(initialValue, Y2, burnIn, iterations)
```

```
summary(chains2)
```

```
##
## Iterations = 1:50001
## Thinning interval = 1
## Number of chains = 1
## Sample size per chain = 50001
##
## 1. Empirical mean and standard deviation for each variable,
##    plus standard error of the mean:
##
##           Mean       SD  Naive SE Time-series SE
## N     327.4570 234.6330 1.0493002      107.17574
## theta   0.2978   0.1837 0.0008213        0.04452
##
## 2. Quantiles for each variable:
##
##           2.5%      25%      50%      75%     97.5%
## N     93.00000 144.0000 253.0000 440.0000 1048.0000
## theta  0.06065   0.1429   0.2463   0.4397    0.6821
```

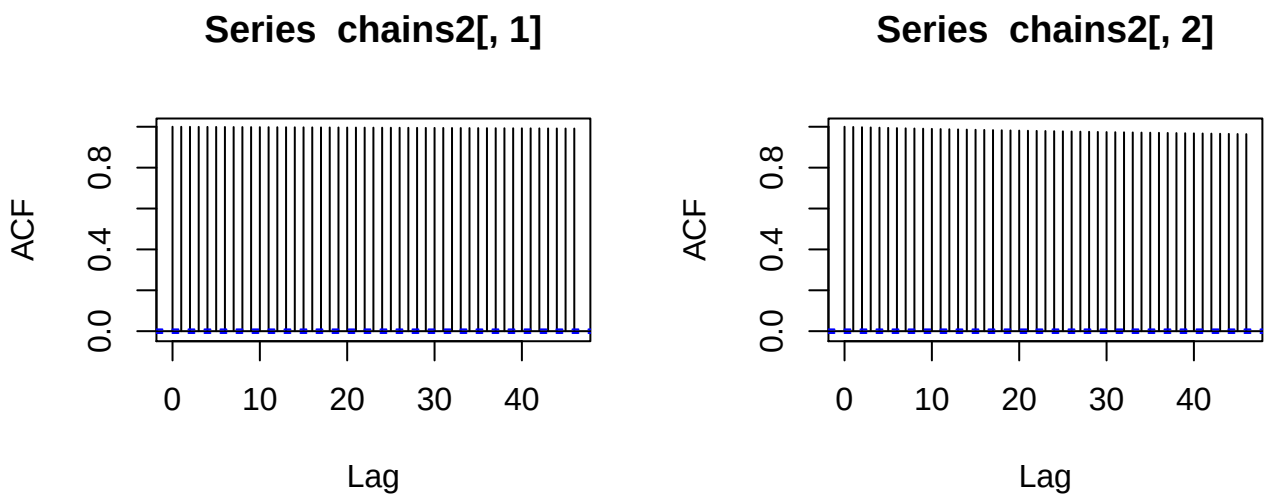
Construct the trace plots

```
plot(chains2)
```



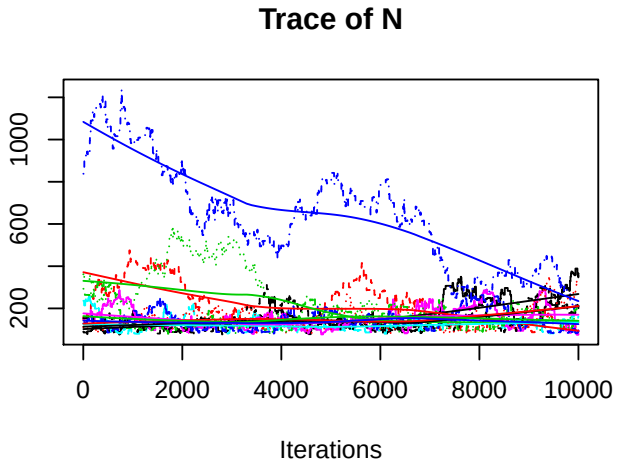
The ACF plot shows a **slow geometric decay** and matches our expectation.

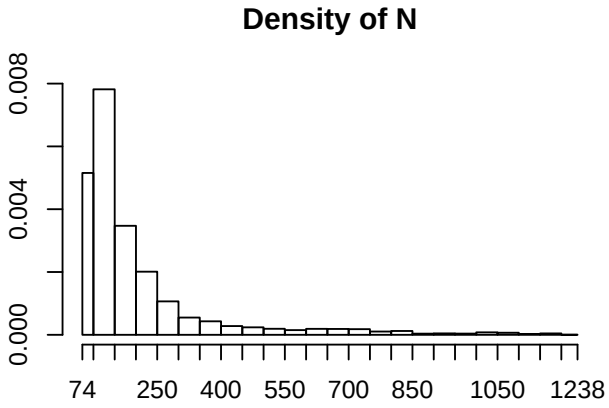
```
par(mfrow=c(1,2))
acf(chains2[, 1])
acf(chains2[, 2])
```

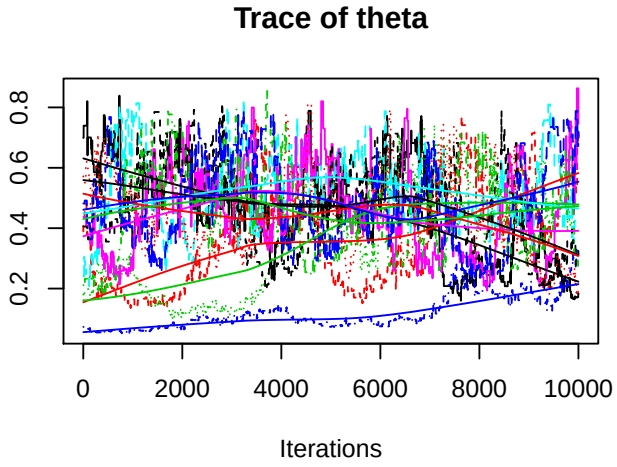


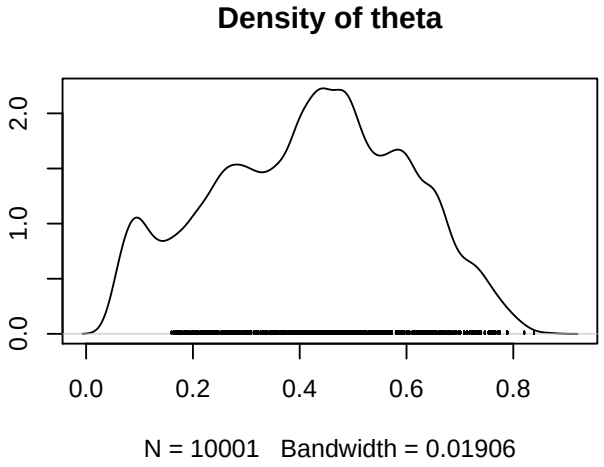
We see from Gelman's diagonisis that the result is acceptable since it is close to 1. But we can no longer achieve a low gelman diagonosis using a small amount of samples.

```
LIST2 = list(chains2)
for (i in 2: 10) {
  initialValue = c(sample(70: 300, 1), runif(1, 0.1, 0.9))
  chains = MCMC(initialValue, Y2, burnIn, iterations)
  LIST2[[i]] = chains
}
MCMC_list = mcmc.list(LIST2)
plot(MCMC_list)
```










```
gelman.diag(MCMC_list)
```

```
## Potential scale reduction factors:
##
##       Point est. Upper C.I.
## N           1.93       3.54
## theta       1.30       1.59
##
## Multivariate psrf
##
## 1.95
```

In this dataset, the effective sample size is not too small.

```
effectiveSize(MCMC_list)
```

```
##         N     theta
##  87.40742 159.35606
```

2. For each of the two data sets, construct a scatterplot of all of your posterior draws of (N, θ), with posterior contours overlaid.

Prepare the basic settings.

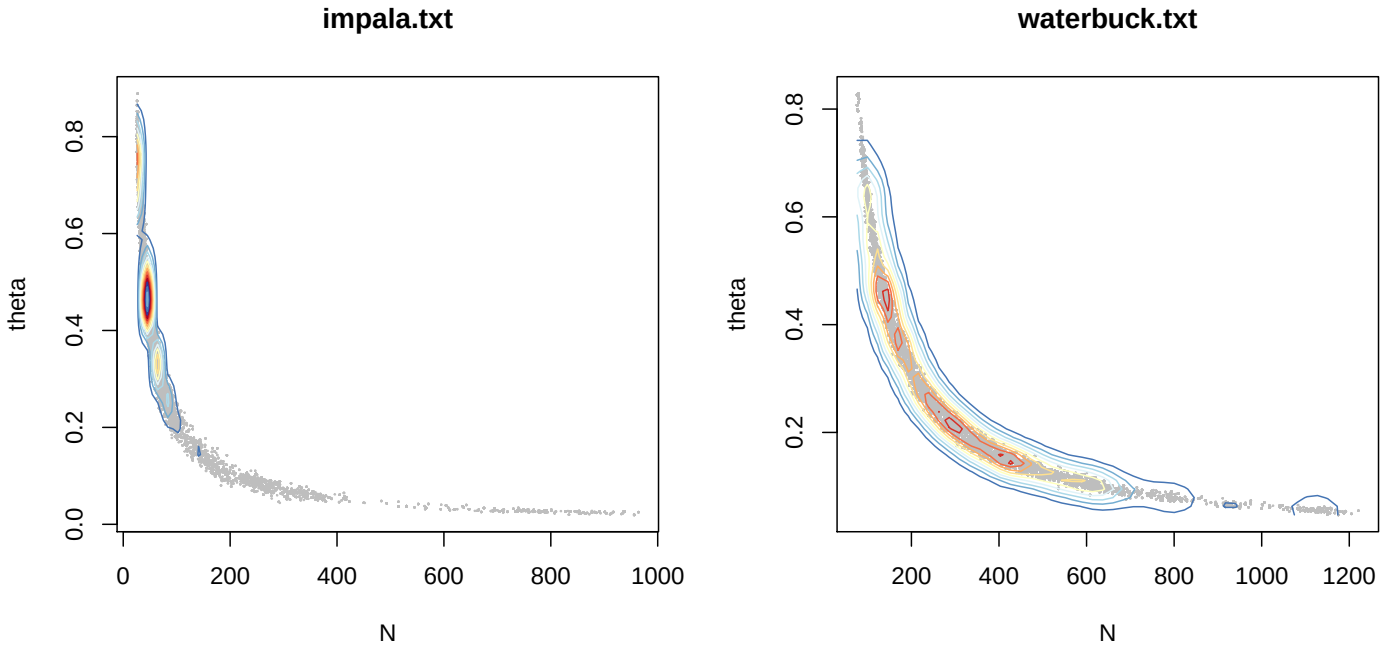
```
library(MASS)
library(RColorBrewer)
k = 11
```

```
my.cols = rev(brewer.pal(k, "RdYlBu"))
```

Posterior draws of (N, θ) for *impala.txt*(left) and *waterbuck.txt*(right)

```
par(mfrow=c(1,2))
z = kde2d(chains1[, 1], chains1[, 2], n=50)
p_posterior = sum(dbinom(t(Y1), chains1[, 1], chains1[, 2], log=TRUE)) - log(chains1[, 1])
plot(as.matrix(chains1), col="grey", cex = .1, main="impala.txt")
contour(z, drawlabels=FALSE, nlevels=k, col=my.cols, add=TRUE)

z = kde2d(chains2[, 1], chains2[, 2], n=50)
plot(as.matrix(chains2), col="grey", cex = .1, main="waterbuck.txt")
contour(z, drawlabels=FALSE, nlevels=k, col=my.cols, add=TRUE)
```



3. Derive the marginal posterior distribution for N under the previously suggested prior, and find its normalizing constant (you will need to compute probabilities using this marginal distribution later in this problem). Calculate the value of the normalizing constants for the impala and waterbuck data sets. You can perform these calculations numerically if you wish.

We already know from #1 that $p(N, \theta|y_1, ..., y_n) \propto \frac{1}{N}\prod_{i=1}^{n}(C_N^{y_i})\theta^{\sum_{i=1}^{n} y_i}(1-\theta)^{nN-\sum_{i=1}^{n} y_i}$, since $\int_0^\infty \theta^{\sum_{i=1}^{n} y_i}(1-\theta)^{nN-\sum_{i=1}^{n} y_i} = \frac{\Gamma(\sum_{i=1}^{n} y_i+1)\Gamma(nN-\sum_{i=1}^{n} y_i+1)}{\Gamma(nN+2)}$, therefore, the marginal posterior distribution for N is

$$
\begin{aligned}
p(N|prior) &\propto \frac{1}{N}\prod_{i=1}^{n}(C_N^{y_i})\frac{\Gamma(\sum_{i=1}^{n} y_i+1)\Gamma(nN-\sum_{i=1}^{n} y_i+1)}{\Gamma(nN+2)} \\
&\propto \frac{1}{N}\prod_{i=1}^{n}(C_N^{y_i})\frac{(nN-\sum_{i=1}^{n} y_i)!}{(nN+1)!} \\
&\propto \prod_{i=1}^{n}(C_N^{y_i})\frac{1}{N\prod_{j=1+nN-\sum_{i=1}^{n} y_i}^{nN+1} j}
\end{aligned}
$$

Build the marginal posterior function for every possible N, and numerically compute the normalizing constant, here we suppose N ranges from 1 to 1000, since when the marginal probability decays extremly fast as N increases.

```
# work with number that overflows in R, e.g. as.brob: log(as.brob(3)^100000) -> 109861.2
library("Brobdingnag")
```

```
##
## Attaching package: 'Brobdingnag'
## The following objects are masked from 'package:base':
##
##     max, min, prod, range, sum
```

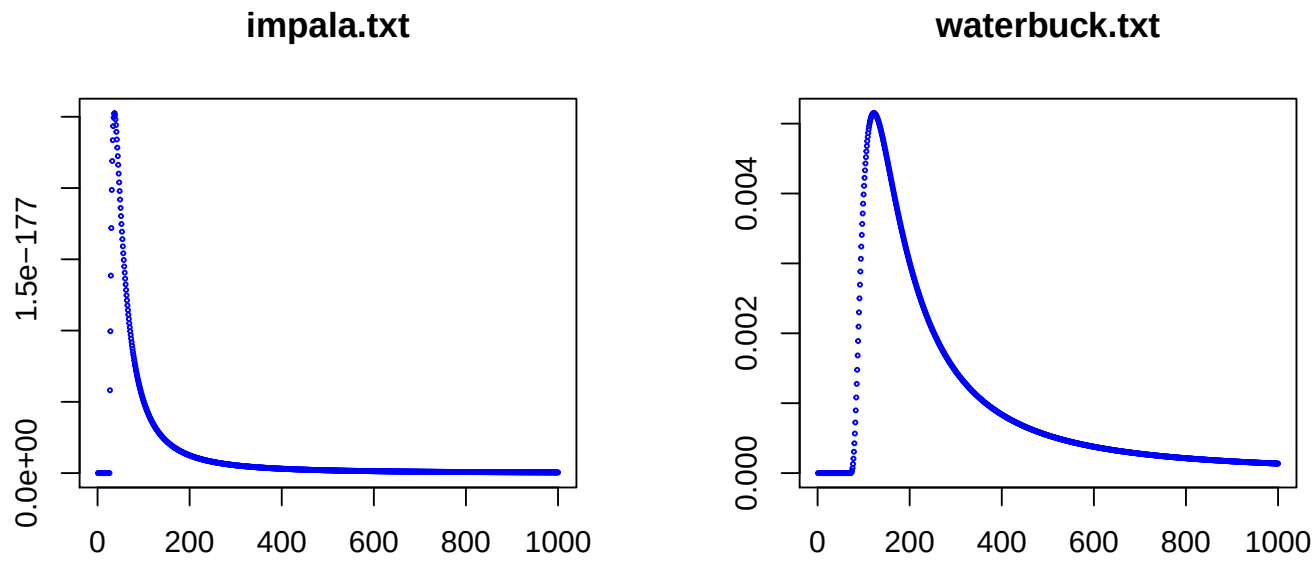
```
log_marginal_posterior = function(N, Y) {
  if (N <= max(Y)) return(-Inf)
  denominor_list = mapply(as.brob, seq(5*N - sum(Y) + 1, 5 * N + 1))
  numerator_list = mapply(as.brob, choose(N, t(Y)))
  denominor = N
  numerator = 1
  for (i in 1: length(denominor_list)) denominor = denominor * denominor_list[[i]]
  for (i in 1: 5) numerator = numerator * numerator_list[[i]]
  return(log(numerator / denominor))
}
```

Show the density functions of N for the two datasets.

```
par(mfrow=c(1,2))
Normal_constant1 = exp(mapply(log_marginal_posterior, seq(1, 1000), Y1)) # not overflow, use exp instead
Normal_constant2 = mapply(log_marginal_posterior, seq(1, 1000), Y2)

Normal_constant2 = Normal_constant2 - max(Normal_constant2)
Normal_constant2 = exp(Normal_constant2) / sum(exp(Normal_constant2))


plot(Normal_constant1, main="impala.txt", xlab="", ylab="", col="blue", cex = .3)
plot(Normal_constant2, main="waterbuck.txt", xlab="", ylab="", col="blue", cex = .3)
```



4. For both the impala and waterbuck data sets, compare the posterior probability that N > 100 obtained from your MCMC algorithm with the probability obtained analytically (or possibly using numerical integration), separately for each of your 10 chains. Then compute the mean and standard deviation of the 10 posterior probability estimates obtained using MCMC, for each of the two data sets. Comment on your findings.

The posterior probability of $N > 100$ obtained analytically for the two datasets.

```
prob_impala = sum(Normal_constant1[101: 1000]) / sum(Normal_constant1) # impala.txt
prob_waterbuck = sum(Normal_constant2[101: 1000]) / sum(Normal_constant2) # waterbuck.txt
prob_impala
```

```
## [1] 0.3107036
```

```
prob_waterbuck
```

```
## [1] 0.9518959
```

The posterior probability of $N > 100$ obtained from 10 chains of MCMC samples for the two datasets, respectively.

```
prob_mcmc = array(dim=c(10, 2))
for (i in 1: 10) {
  prob_mcmc[i, 1] = length(LIST1[[i]][LIST1[[i]][, 1] > 100, 1]) / length(LIST1[[i]][, 1])
  prob_mcmc[i, 2] = length(LIST2[[i]][LIST2[[i]][, 1] > 100, 1]) / length(LIST2[[i]][, 1])
}
mean(prob_mcmc[, 1]); sd(prob_mcmc[, 1])
```

```
## [1] 0.1805144
```

```
## [1] 0.08172237
```

```
mean(prob_mcmc[, 2]); sd(prob_mcmc[, 2])
```

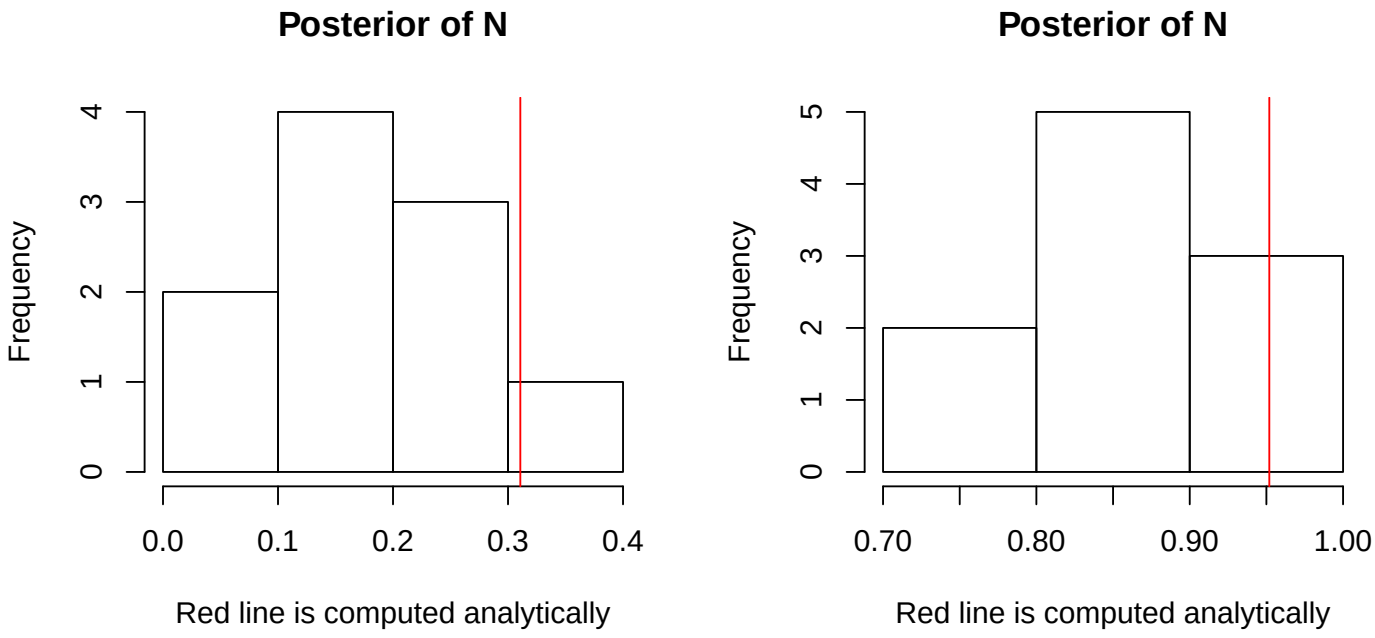
```
## [1] 0.8659834
```

```
## [1] 0.07619652
```

From the result, we see that the result computed analytically falls within the range of simulated MCMC samples, which suggests a good match. The waterbuck example has a very long tail, which may be related to the instability of the dataset.

```
par(mfrow = c(1, 2))
hist(prob_mcmc[, 1], nclass=3, main="Posterior of N", xlab="Red line is computed analytically" )
abline(v = prob_impala, col="red" )

hist(prob_mcmc[, 2], nclass=3, main="Posterior of N", xlab="Red line is computed analytically" )
abline(v = prob_waterbuck, col="red" )
```



Applied

The New York Department of Education (DoE) conducted a randomized experiment to assess the effects of the following five interventions (and their possible interactions) on the performance of 232 high schools.

(QR) Quality Review: The Quality Review is a two- or three-day visit by an experienced educator to the school. During the review, the educator visits classrooms, talks with school leaders, and utilizes a rubric to evaluate how well the school is organized to educate its students. After the visit, the school receives both a Quality Review score and a report that is published on the DoE website. This document provides the school's community with evidence-based information about the school's development, and serves as a source of feedback for the school's leadership to improve the school's support for student performance.

(PA) Periodic Assessments: Students take Periodic Assessments several times throughout the school year to give teachers more information about what the students have learned. Teachers use these assessments, along with other school work and what they observe in class, to learn where students need more help, and to plan targeted instruction.

(IT) Inquiry Team: The school creates a broad-based Inquiry Team consisting of the principal and several teachers. This team uses a multistep cycle of inquiry to identify skill gaps for "target" low-performing students, and to address their learning needs through instruction and changes in systems that inhibit their success.

(SPBP) School-wide performance bonus program: This program features a school-wide reward structure over which school-based committees have authority. It utilizes multiple measures of teacher effectiveness.

(ARIS) This program provides education and support to promote the usage of ARIS, an online database for educators that enables teachers to view student information, explore instructional resources, and collaborate with teachers in their own school and across New York.

Each school can receive either the "treatment" or "control" level of each of these factors, with the restriction that no school will receive the control levels of all of the factors (each school expects to be able to try at least one new intervention, otherwise they won't participate in the study). The DoE conducted a fractional factorial experiment on the schools. The treatment-level combinations assigned to these schools are contained in the file *ny_doe_treatment_assignments.csv*. Each row corresponds to one school, and the columns contain the levels of the five factors, where 1 denotes the treatment level and -1 denotes the control level.

The performance measure of a school is a cumulative score on a 1-100 scale in the school's Progress Report, which is issued annually near the start of the school year. This score is based on four components..

I School Environment (15% of overall score): Measures "pre-conditions" for learning, specifically, student attendance and other crucial aspects of the school's environment (e.g., high expectations, engagement, safety, respect, and communication). Attendance is measured directly, and the other aspects of school environment are measured by surveys of parents, students, and teachers.

II Student Performance (25% of overall score): Measures the percentage of students at a school who have graduated within four to six years.

III Student Progress (60% of overall score): Measures the ability of a school to help students progress toward the eventual goal of earning a diploma. The measure focuses on the capacities students develop as a result of attending the school, not the capacities they bring with them on the first day. Attention is given to all students in each school, and particular emphasis is given to the one-third of students who entered high school at the lowest performance level.

IV Schools can earn additional credit in the Exemplary Student Outcomes category. The cumulative scores for the schools are contained in the file *ny_doe_treatment_assignments.csv*. Data on the following 12 school-level covariates have also been collected by the DoE.

x1: Total number of students in the school.

x2: Ratio of numbers of female and male students.

x3: Percentage of African American students.

x4: Percentage of Asian students.

x5: Percentage of attendance.

x6: Poverty rate as percentage of total enrollments.

x7: Total number of staff and teachers in the school.

x8: Percentage of teachers with more than 5 years teaching experiences.

x9: Percentage of teachers with masters or doctoral degrees.

x10: Previous year's school performance score (1-25) obtained from the Progress Report.

x11: Previous year's school environment score (1-15) obtained from the Progress Report.

x12: Previous year's student progress score (1-60) obtained from the Progress Report.

The values of these covariates for each of the schools are contained in the file *ny_doe_treatment_assignments.csv*. You have been tasked with analyzing this experiment, and presenting your final conclusions on the significant main effects and interactions of the five interventions to representatives from the New York DoE. The representatives want to know in particular which interventions can help schools improve their performance, how much gain is expected from such interventions, and how much uncertainty is associated with your conclusions.

Important requirements for your analysis follow below.

- You must identify the experimental units.

- You must clearly state and justify the assumptions underlying your analysis. For example, if you believe that all third- and higher-order interactions are sufficiently negligible so as to be omitted from consideration, you must provide your reasoning as to why you believe this to be the case. If relevant, briey discuss whether your assumptions could be violated.

- You must identify the estimands, i.e., the quantities to be estimated from this experiment. Explain your notation clearly.

- You must state whether any main effects or interactions of interest are confounded in this design.

- You must state whether this experimental design guarantees that the treatment comparisons are fair, in the sense that the covariates are reasonably balanced over the interventions.

- You must clearly describe your methodology.

- You must perform posterior predictive checks and/or use predictive accuracy measures to assess the fit of your model.

- You must present your final results in a form that can be understood by the DoE representatives.

To find significant main effects and interactions of the five interventions to representatives from the New York DoE, I will first use probit regression to do variable selection and proceed by using ridge regression to quantitatively analyze the effect these interventions.

Load data first

```
set.seed(666)
setwd("C:/Users/Wei/Documents/Purdue STAT 695 Bayesian Data Analysis/HW3")

data_covariates = read.csv("ny_doe_covariates.csv", header=TRUE)
data_assignments = read.csv("ny_doe_treatment_assignments.csv", header=TRUE)
data_grades = read.csv("ny_doe_cumulative_grades.csv", header=TRUE)
```

Normalize the input

```
library(standardize)
```

```
## Warning: package 'standardize' was built under R version 3.4.2
```

```
colNames = colnames(data_assignments)
data_assignments = scale(data_assignments)
```

From viewpoint of experiment design, we know that the pricinples of effect hierarchy, effect heredity and effect sparsity are quite related to model selection, first order effects typically have larger influence than second order effects, namely interactions. Before we take care of interactions, let's first consider the input with only the first order effects and intercept.

```
X = cbind(rep(1, length(data_grades)), data_assignments)
colnames(X)[1] = "Intercept"
```

Compare the grades in last year and this year to determine if the first order effects have made a difference. 1 represents improvement in grade, 0 otherwise.

```
past_grades = data_covariates[11: 13]
past_grades = rowSums(past_grades)
Y = as.vector(data_grades > past_grades)
Y[Y==TRUE] = 1
Y[Y==FALSE] = 0
```

Probit regression with MH sampling

Build the posterior density function, since we have no prior information, scaled Cauchy prior is applied.

```
log_posterior = function(corr) {
  return(sum(dcauchy(corr, scale=2.5, log=TRUE)) +
         sum(Y * pnorm(X %*% corr, log.p=TRUE) +
         (1 - Y) * pnorm(X %*% corr, lower.tail=FALSE, log.p=TRUE)))
}
```

Before sampling coefficients using MCMC, we first estimate the optimal parameters to give us a robust initial point to start.

```
optim_pars = optim(rep(0, ncol(X)), log_posterior,
                   method="BFGS", control=list(fnscale=-1), hessian=TRUE)
```

```
initialValue = optim_pars$par
Hessian = optim_pars$hessian / (2.4 / sqrt(ncol(X))) # BDA pg. 290
burnIn = 10000
iterations = 20000
```

Set up the MCMC sampler with symmetric proposal distribution

```
MCMC_MH = function(X, initialValue, Hessian) {
  chains = array(dim=c(iterations + 1, ncol(X)))
  chains[1, ] = initialValue
  Rchol = chol(-Hessian)
  BarProgress = txtProgressBar(min = 1, max = nrow(chains), style = 3)
  for (i in 1: iterations) {
    # better avoid saving the inverse of a matrix, compute them instead
    proposal = chains[i, ] + backsolve(Rchol, rnorm(ncol(X)))

    # write exp(num) as num to avoid overflow; symmetric proposal
    log_acceptance_prob = log_posterior(proposal) - log_posterior(chains[i, ])

    chains[i + 1, ] = chains[i, ]
    if (log(runif(1)) < log_acceptance_prob)
      chains[i + 1, ] = proposal
    #setTxtProgressBar(BarProgress, i) # not suitable in rmd, useful in other cases
  }
  close(BarProgress)
  return(chains[-(1: burnIn), ])
}
```

Compute and visualize the result. From the quantile of parameters bettwen 2.5% to 97.5%, we see that QR, PA and SPBP are significant different from 0, the rest of the coefficients, such as intercept, IT and ARIS, are not that clear.

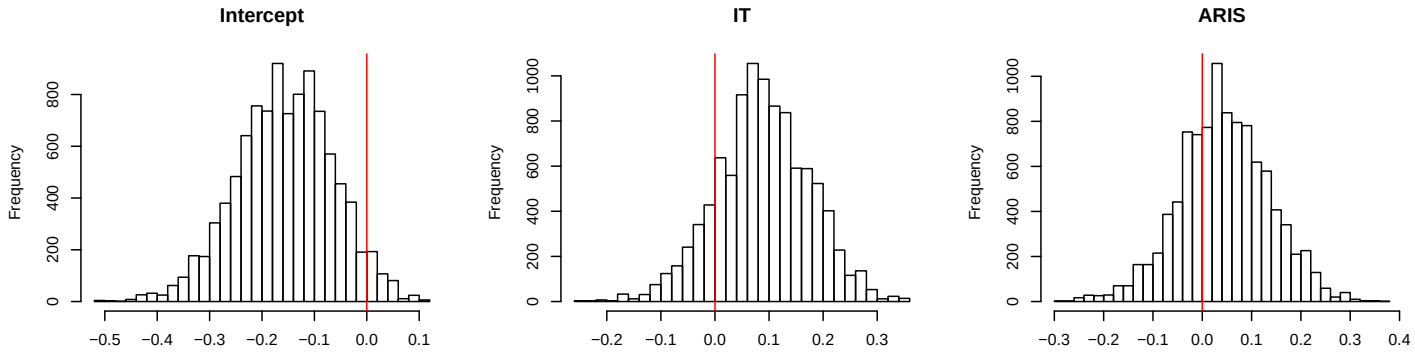
```
pars_draws = MCMC_MH(X, initialValue, Hessian)
```

```
par(mfrow=c(1, 3))
for (i in 1: ncol(X)) {
  if (i == 1) print("Significant effects")
  interval = quantile(pars_draws[ , i], probs=c(0.025, 0.975))
  if (interval[1] * interval[2] < 0) { # coefficient not significantly differs from 0
    hist(pars_draws[ , i], nclass=30, main=colnames(X)[i], xlab="" )
    abline(v = 0, col="red")
  }
  else {
    print(colnames(X)[i])
  }
}
```

```
## [1] "Significant effects"
```

```
## [1] "QR"
## [1] "PA"
```

```
## [1] "SPBP"
```



Next, let's consider two-factor interactions items. We will evaluate the first order effects of QR, PA, SPBP with all second order effects containing one of these main factors. Moreover, we exclude the intercept item since it is not significant.

```
QR = X[, 2]; PA = X[, 3]; IT = X[, 4]; SPBP = X[, 5]; ARIS = X[, 6]
XX = cbind(QR, PA, SPBP, QR * PA, QR * SPBP, PA * SPBP,
           QR * IT, QR * ARIS, PA * IT, PA * ARIS, SPBP * IT, SPBP * ARIS)
colnames(XX) = c("QR", "PA", "SPBP", "QR*PA", "QR*SPBP", "PA*SPBP",
                 "QR*IT", "QR*ARIS", "PA*IT", "PA*ARIS", "SPBP*IT", "SPBP*ARIS")
```

Rewrite the posterior density function to adapt to the new input, solve the optimal parameter again to sample with a robust starting point.

```
log_posterior = function(corr) {
  return(sum(dcauchy(corr, scale=2.5, log=TRUE)) +
         sum(Y * pnorm(XX %*% corr, log.p=TRUE) +
         (1 - Y) * pnorm(XX %*% corr, lower.tail=FALSE, log.p=TRUE)))
}

optim_pars = optim(rep(0, ncol(XX)), log_posterior,
                   method="BFGS", control=list(fnscale=-1), hessian=TRUE)

initialValue = optim_pars$par
Hessian = optim_pars$hessian / (2.4 / sqrt(ncol(XX))) # BDA pg. 290
```

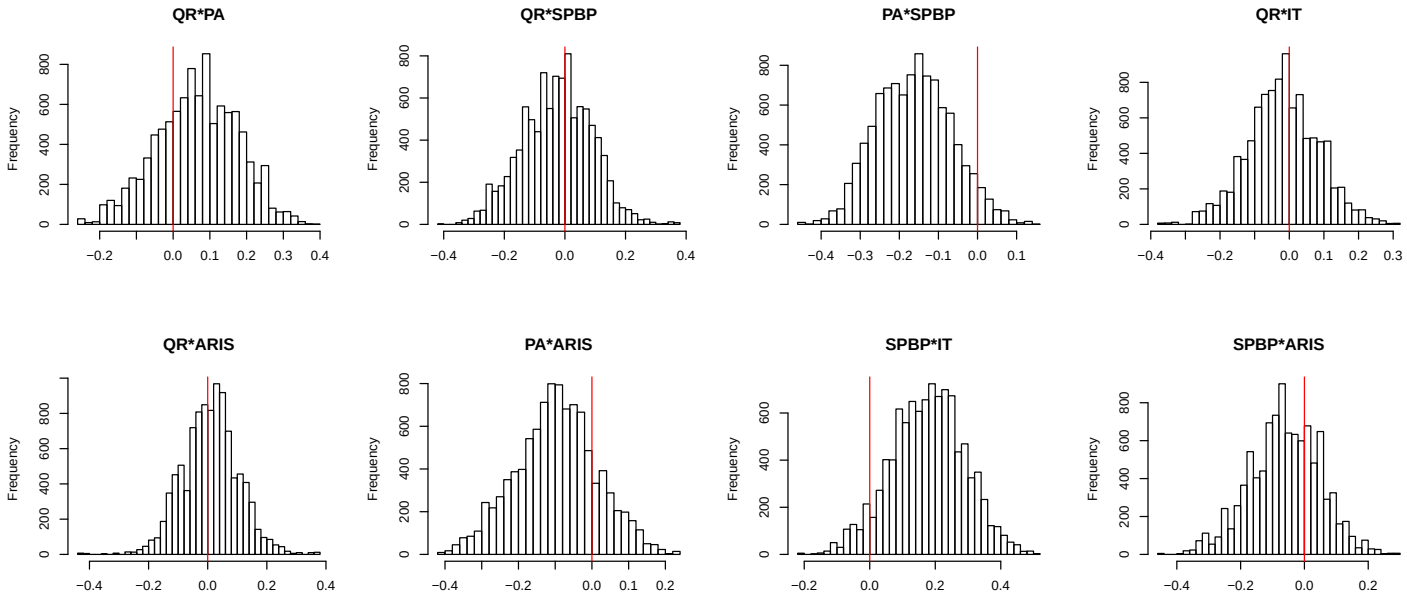
Compute and visualize the result. We see that we can ignore most of the second order effects other than $PA \times IT$.

```
pars_draws = MCMC_MH(XX, initialValue, Hessian)
```

```
par(mfrow=c(2, 4))
for (i in 1: ncol(XX)) {
  if (i == 1) print("Significant effects")
  interval = quantile(pars_draws[ , i], probs=c(0.025, 0.975))
  if (interval[1] * interval[2] < 0) { # coefficient not significantly differs from 0
    hist(pars_draws[ , i], nclass=30, main=colnames(XX)[i], xlab="" )
    abline(v = 0, col="red")
  }
  else {
    print(colnames(XX)[i])
  }
}
```

```
## [1] "Significant effects"
## [1] "QR"
## [1] "PA"
## [1] "SPBP"
```

```
## [1] "PA*IT"
```



Since second order effects already make a difference, let's figure out if the third order effects should also be included. Same procedure again.

```
XX = cbind(QR, PA, SPBP, PA * IT,
           QR*PA*IT, QR*PA*SPBP, QR*PA*ARIS, QR*IT*SPBP, QR*IT*ARIS, QR*SPBP*ARIS,
           PA*IT*SPBP, PA*IT*ARIS, PA*SPBP*ARIS,
           IT*SPBP*ARIS)
colnames(XX) = c("QR", "PA", "SPBP", "PA*IT", "QR*PA*IT",
                 "QR*PA*SPBP", "QR*PA*ARIS", "QR*IT*SPBP",
                 "QR*IT*ARIS", "QR*SPBP*ARIS", "PA*IT*SPBP",
                 "PA*IT*ARIS", "PA*SPBP*ARIS", "IT*SPBP*ARIS")

log_posterior = function(corr) {
  return(sum(dcauchy(corr, scale=2.5, log=TRUE)) +
         sum(Y * pnorm(XX %*% corr, log.p=TRUE) +
         (1 - Y) * pnorm(XX %*% corr, lower.tail=FALSE, log.p=TRUE)))
}

optim_pars = optim(rep(0, ncol(XX)), log_posterior,
                   method="BFGS", control=list(fnscale=-1), hessian=TRUE)

initialValue = optim_pars$par
Hessian = optim_pars$hessian / (2.4 / sqrt(ncol(XX))) # BDA pg. 290
```

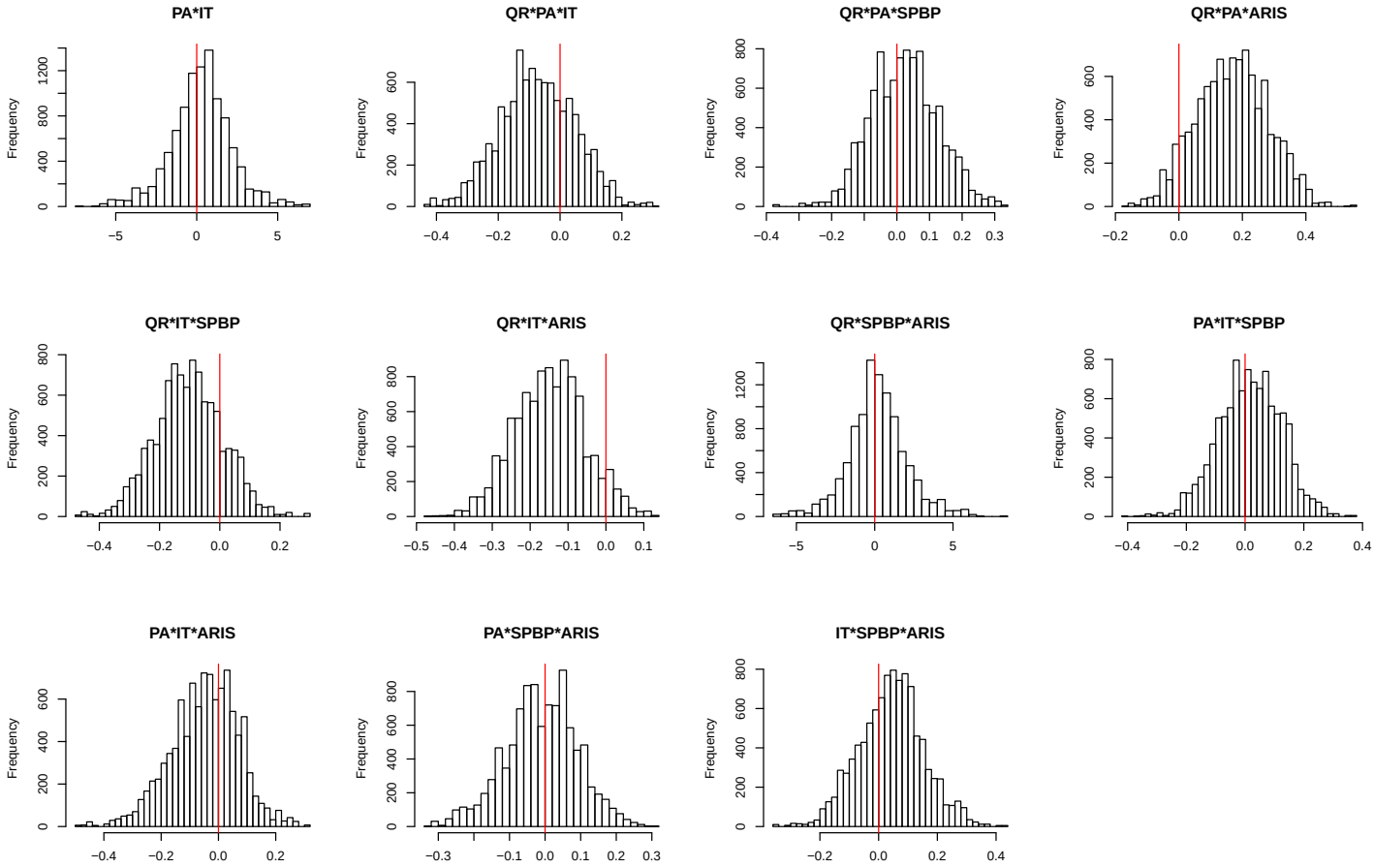
Compute and visualize the result. We find that all the higher order effects don't have any too much impact to our model. We will then only focus on interventions of QR, PA, SPBP, $PA \times IT$

```
pars_draws = MCMC_MH(XX, initialValue, Hessian)
```

```
colnames(pars_draws) = colnames(XX)
par(mfrow=c(3, 4))
for (i in 1: ncol(XX)) {
  if (i == 1) print("Significant effects")
  interval = quantile(pars_draws[ , i], probs=c(0.025, 0.975))
  if (interval[1] * interval[2] < 0) { # coefficient not significantly differs from 0
    hist(pars_draws[ , i], nclass=30, main=colnames(XX)[i], xlab="" )
    abline(v = 0, col="red")
  }
  else {
    print(colnames(XX)[i])
```

```
  }
}
```

```
## [1] "Significant effects"
## [1] "QR"
## [1] "PA"
## [1] "SPBP"
```



Evaluate the effects of QR, PA, SPBP, PA*IT. Genrate input and adapt the log posterior data to the new dataset.

```
XX = cbind(QR, PA, SPBP, PA * IT)
colnames(XX) = c("QR", "PA", "SPBP", "PA*IT")

log_posterior = function(corr) {
  return(sum(dcauchy(corr, scale=2.5, log=TRUE)) +
         sum(Y * pnorm(XX %*% corr, log.p=TRUE) +
         (1 - Y) * pnorm(XX %*% corr, lower.tail=FALSE, log.p=TRUE)))
}
```

Same procedure again. We see from the quantile statistics that all the effects are significant in our model.

```
optim_pars = optim(rep(0, ncol(XX)), log_posterior,
                   method="BFGS", control=list(fnscale=-1), hessian=TRUE)

initialValue = optim_pars$par
Hessian = optim_pars$hessian / (2.4 / sqrt(ncol(XX))) # BDA pg. 290
pars_draws = MCMC_MH(XX, initialValue, Hessian)
```

```
colnames(pars_draws) = colnames(XX)
pars_draws = data.frame(pars_draws)
```

We can see that QR (Quality Review) has a negative impact to the performance

```
quantile(pars_draws$QR, probs=c(0.025, 0.975))
```

```
##        2.5%       97.5%
## -0.37940459 -0.03391224
```

PA (Periodic Assessments), SPBP(School-wide performance bonus program) and PA-IT (Periodic Assessments with Inquiry Team) can help improve the performance significantly.

```
quantile(pars_draws$PA, probs=c(0.025, 0.975))
```

```
##      2.5%     97.5%
## 0.3267412 0.7079752
```

```
quantile(pars_draws$SPBP, probs=c(0.025, 0.975))
```

```
##      2.5%     97.5%
## 0.3241203 0.6926350
```

```
quantile(pars_draws$PA.IT, probs=c(0.025, 0.975))
```

```
##      2.5%     97.5%
## 0.2029513 0.5753790
```

Posterior checks

Let's move to the second part to see if model makes sense, it follows by posterior checks. Set up the variables first

```
XX = data.frame(XX)
group = list(QR=XX$QR>0, PA=XX$PA>0, SPBP=XX$SPBP>0, PA.IT=XX$PA.IT>0)
mat = matrix(NA, nrow=nrow(pars_draws), ncol=2)
posterior_check = list(QR=mat, PA=mat, SPBP=mat, PA.IT=mat)
```

We use the sampling parameters to simulate the input and compare them with the real probability (since we have scaled the features, they all have 0 mean)

```
#BarProgress = txtProgressBar(min=1, max=nrow(pars_draws), style=3)
for (i in 1: nrow(pars_draws)) {
    diff = rep(NA, nrow(XX))
    for (j in 1: nrow(XX)) {
        prediction = pnorm(sum(XX[j, ] * pars_draws[i, ]))
        diff[j] = Y[j] - prediction
    }
    # posterior_check$element = [treatment group, control group]
    posterior_check$QR[i, ] = c(mean(diff[group$QR]), mean(diff[!group$QR]))
    posterior_check$PA[i, ] = c(mean(diff[group$PA]), mean(diff[!group$PA]))
    posterior_check$SPBP[i, ] = c(mean(diff[group$SPBP]), mean(diff[!group$SPBP]))
    posterior_check$PA.IT[i, ] = c(mean(diff[group$PA.IT]), mean(diff[!group$PA.IT]))
    #setTxtProgressBar(BarProgress, i) # not suitable in rmd
}
#close(BarProgress)
```

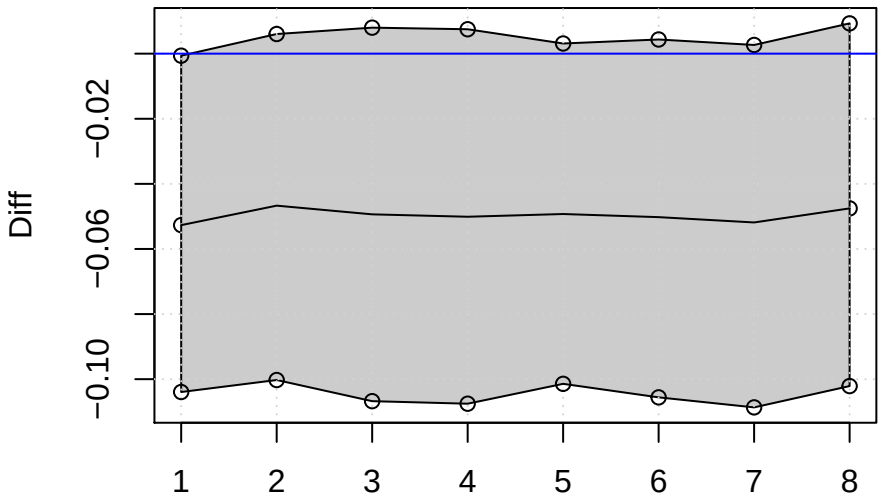
Let's try to evaluate the result using quantile statistics.

```
mt = cbind(posterior_check$QR, posterior_check$PA, posterior_check$SPBP, posterior_check$PA.IT)
up_quantile = function(vec) quantile(vec, prob=0.975)
down_quantile = function(vec) quantile(vec, prob=0.025)

evals = data.frame(mean=apply(mt, 2, mean),
                   up=apply(mt, 2, up_quantile),
                   down=apply(mt, 2, down_quantile))
```

From the figure below, we see that the mean values 0 are all located in the credit interval, which means the model does a fair job. For further problems like why the differences seem a bit of biased, due to time limit, we will not discuss it here.

```
plot(evals$mean, ylim=c(min(evals$down), max(evals$up)), xaxt="n", ylab="Diff", xlab="")
polygon(c(1: 8, rev(1: 8)),c(evals$down, rev(evals$up)),col=gray(0.8))
lines(evals$mean, pch=15, panel.first = grid())
points(evals$up)
points(evals$down)
abline(h=0, col="blue")
axis(1, at=1:8, labels=rownames(evals))
```



Ridge regression

We have finished variable selection, let's proceed by analyzing the magnitude of the effect of interventions.

```
XX = cbind(rep(1, nrow(XX)), QR, PA, SPBP, PA * IT)
colnames(XX) = c("Intercept", "QR", "PA", "SPBP", "PA*IT")
Y = (data_grades - past_grades)[1: 232, ]
```

Before we move forward, let's analyze the correlation matrix of features. It appears that the correlations are small enough, which makes sure that we can safely use ridge regression

```
round(cor(XX[, 2:5]), 4)
```

```
##            QR      PA    SPBP   PA*IT
## QR     1.0000  0.0000  0.0082 -0.0003
## PA     0.0000  1.0000 -0.0259  0.0000
## SPBP   0.0082 -0.0259  1.0000 -0.0436
## PA*IT -0.0003  0.0000 -0.0436  1.0000
```

```
coef = function(k) {
    R = t(XX) %*% XX
    inv = solve(R + k * diag(5)) # when k is large, inverse suffers less numerical problem

    mu = inv %*% t(XX) %*% Y
    sd = sqrt(diag(inv %*% R %*% inv))

    df = data.frame(mean=mu, sd=sd, '0.025'=mu-2*sd, '0.975'=mu+2*sd)
```

```
    round(df, 4)
}
```

We tested three different choices of k, which is 0.01, 1, 100, corresponding to the prior distributions following $N(0, 10^2)$, $N(0, 1^2)$ and $N(0, 0.1^2)$, it's beyond our interest to seek for all possible priors because the score only ranges from 0 ~ 100.

```
coef(0.01)
```

```
##               mean     sd  X0.025  X0.975
## Intercept -1.3130 0.0657 -1.4443 -1.1817
## QR        -0.4107 0.0658 -0.5423 -0.2791
## PA         5.3510 0.0658  5.2194  5.4826
## SPBP       5.9791 0.0659  5.8474  6.1109
## PA*IT      3.4220 0.0660  3.2900  3.5540
```

```
coef(1)
```

```
##               mean     sd  X0.025  X0.975
## Intercept -1.3071 0.0654 -1.4379 -1.1764
## QR        -0.4087 0.0655 -0.5398 -0.2777
## PA         5.3275 0.0655  5.1964  5.4586
## SPBP       5.9523 0.0656  5.8211  6.0835
## PA*IT      3.4061 0.0657  3.2747  3.5375
```

```
coef(100)
```

```
##               mean     sd  X0.025  X0.975
## Intercept -0.9044 0.0459 -0.9961 -0.8126
## QR        -0.2762 0.0459 -0.3680 -0.1844
## PA         3.7008 0.0459  3.6090  3.7926
## SPBP       4.1090 0.0459  4.0172  4.2009
## PA*IT      2.3234 0.0460  2.2314  2.4153
```

From the three sets of results above, we can see with no doubt that QR (Quality Review) has a slighted negative impact to the performancce, with reducing roughtly -0.4 ~ -0.2 score to the overall perforamance. As a contrast, PA (Periodic Assessments), SPBP (School-wide performance bonus program) and PA-IT (Periodic Assessments with Inquiry Team) can significantly improve the annual performance by no less than 3.6, 4.0, 2.2 points respectively.

References

Raftery, A. E. (1988). Inference for the Binomial N parameter: A hierarchical Bayes approach. Biometrika 75 (2), 223-228.

A simple Metropolis-Hastings MCMC in R

MCMC chain analysis and convergence diagnostics with coda in R

Ridge regression