# 05-29-2023

Hengde Ouyang

2023-05-28

## 1. Try Another DataSet (We try the lung dataset in Survival package)

For this data set, our algorithm seem to have a lower C Statistics than cox ph model. (it's unstable because sometimes it will produce a large average C STAT, sometimes it will produce a small average C STAT)
However, when I change the eta value from n (the number of observations) to 2n, the, the value changes a lot.
Need proper choice of eta for different data set.

```r
no_na_lung = na.omit(lung)


# Input data
Y = no_na_lung$time
delta= no_na_lung$status - 1
tau = 2500
A <- model.matrix(time ~ -1+ inst+ age + sex + ph.ecog + ph.karno + pat.karno
                  +meal.cal+wt.loss,
                  data=no_na_lung)
beta0 = rep(0,dim(A)[2])
sigma0 = rep(1,dim(A)[2])

# A relatively Small data set, can increase the iteration
m = 11000
B = 1000
eta = length(Y)
Wmat_option = 0
kappa = 1
var.prop = kappa*solve(t(A)%*%A)
```

We still use cox PH model as a reference

```r
coxmodel <- coxph(Surv(time, status)~inst+ age + sex + ph.ecog
                  + ph.karno + pat.karno
                  + meal.cal+wt.loss,x=TRUE, data = no_na_lung)
summary(coxmodel)
```

```
## Call:
```

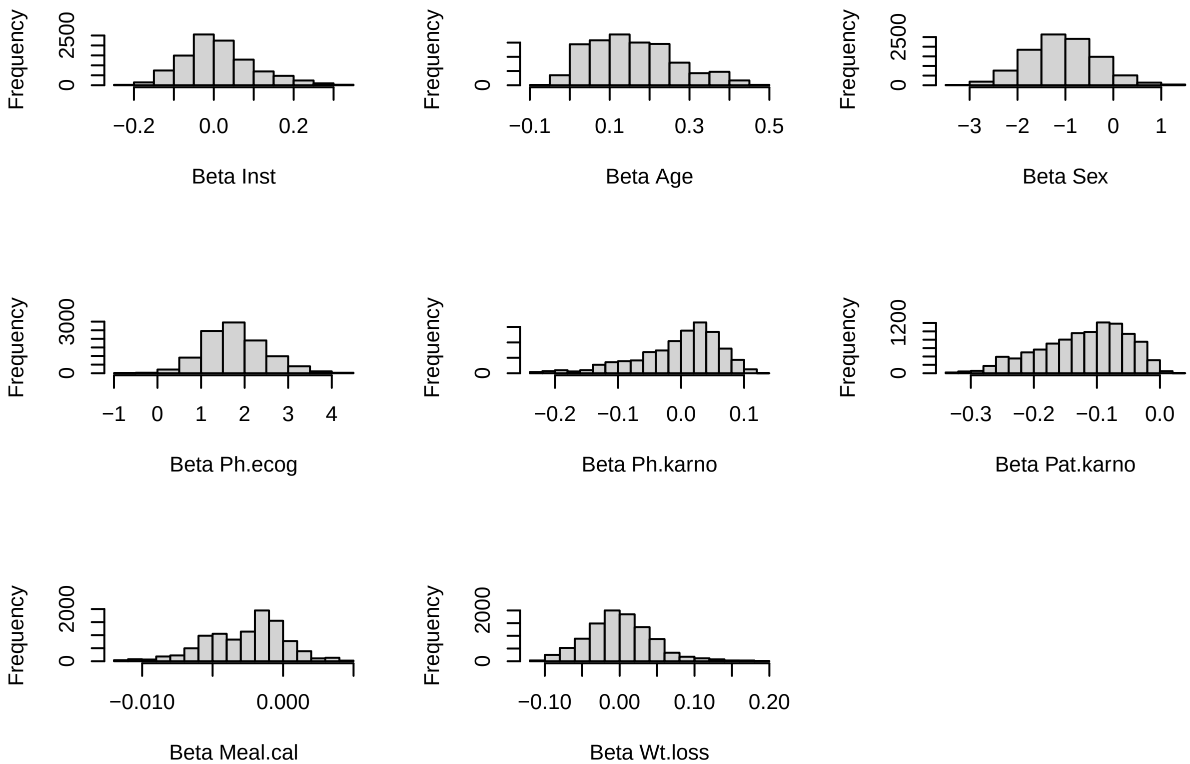
```
## coxph(formula = Surv(time, status) ~ inst + age + sex + ph.ecog +
##     ph.karno + pat.karno + meal.cal + wt.loss, data = no_na_lung,
##     x = TRUE)
##
##   n= 167, number of events= 120
##
##                 coef  exp(coef)   se(coef)      z Pr(>|z|)
## inst      -3.037e-02  9.701e-01  1.312e-02 -2.315 0.020619 *
## age        1.281e-02  1.013e+00  1.194e-02  1.073 0.283403
## sex       -5.666e-01  5.674e-01  2.014e-01 -2.814 0.004890 **
## ph.ecog    9.074e-01  2.478e+00  2.386e-01  3.803 0.000143 ***
## ph.karno   2.658e-02  1.027e+00  1.163e-02  2.286 0.022231 *
## pat.karno -1.091e-02  9.891e-01  8.141e-03 -1.340 0.180160
## meal.cal   2.602e-06  1.000e+00  2.677e-04  0.010 0.992244
## wt.loss   -1.671e-02  9.834e-01  7.911e-03 -2.112 0.034647 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
##           exp(coef) exp(-coef) lower .95 upper .95
## inst         0.9701     1.0308    0.9455    0.9954
## age          1.0129     0.9873    0.9895    1.0369
## sex          0.5674     1.7623    0.3824    0.8420
## ph.ecog      2.4778     0.4036    1.5523    3.9552
## ph.karno     1.0269     0.9738    1.0038    1.0506
## pat.karno    0.9891     1.0110    0.9735    1.0051
## meal.cal     1.0000     1.0000    0.9995    1.0005
## wt.loss      0.9834     1.0169    0.9683    0.9988
##
## Concordance= 0.648  (se = 0.03 )
## Likelihood ratio test= 33.7  on 8 df,   p=5e-05
## Wald test            = 31.72  on 8 df,   p=1e-04
## Score (logrank) test = 32.51  on 8 df,   p=8e-05
```

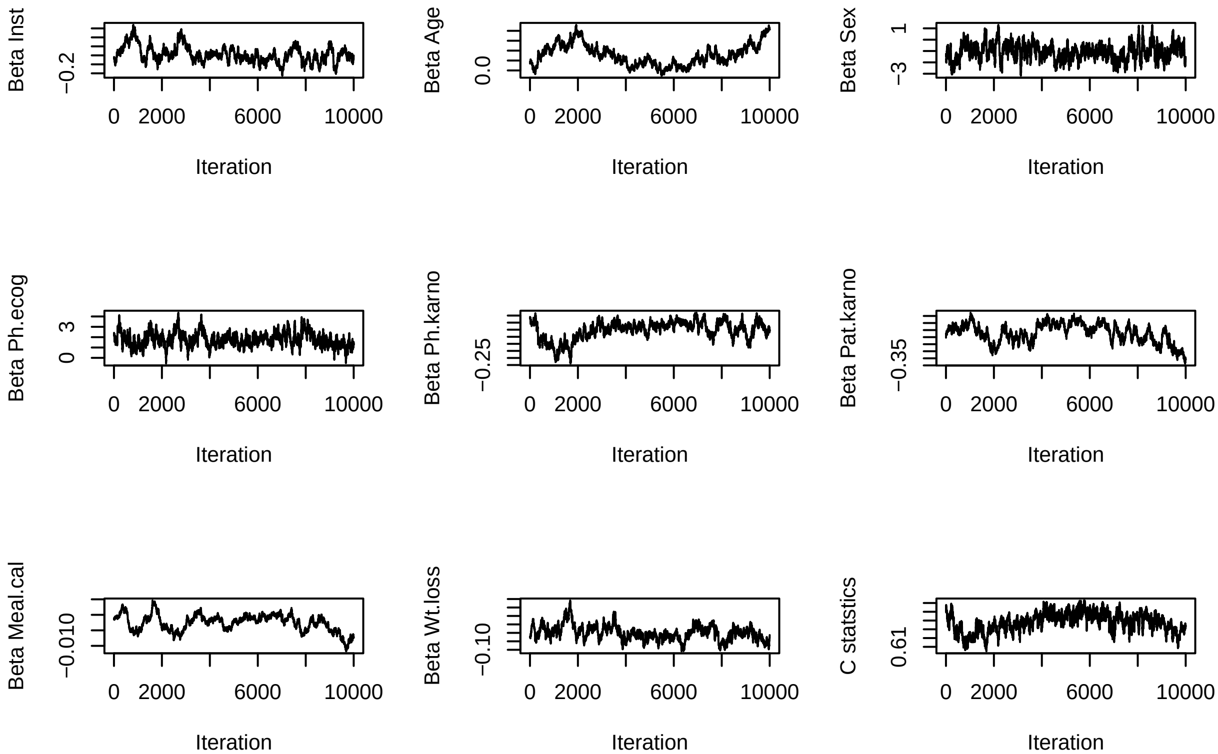
```
system.time({
  result_lung = MH_Sampling(Y,delta,tau,
                        A,beta0,sigma0,
                        var.prop,m,
                        B,eta,Wmat_option)

})
```

```
##    user  system elapsed
##   30.29    3.55   48.83
```

```
# Acceptance Rate
result_lung$accept_rate
```

```
## [1] 0.7941818
```


Frequency
0
2500
−0.2
0.0
0.2
Beta Inst
Frequency
0
−0.1
0.1
0.3
0.5
Beta Age
Frequency
0
2500
−3
−2
−1
0
1
Beta Sex
Frequency
0
3000
−1
0
1
2
3
4
Beta Ph.ecog
Frequency
0
−0.2
−0.1
0.0
0.1
Beta Ph.karno
Frequency
0
1200
−0.3
−0.2
−0.1
0.0
Beta Pat.karno
Frequency
0
2000
−0.010
0.000
Beta Meal.cal
Frequency
0
2000
−0.10
0.00
0.10
0.20
Beta Wt.loss




```
Wmat = HarrellC_Wmat(Y,delta,tau)
# Average Theta C-Stat
HarrellC(colSums(result_lung$THETA),Wmat)
```

```
## [1] 0.6549536
```

```
# Average C-Stat
mean(result_lung$C_stat)
```

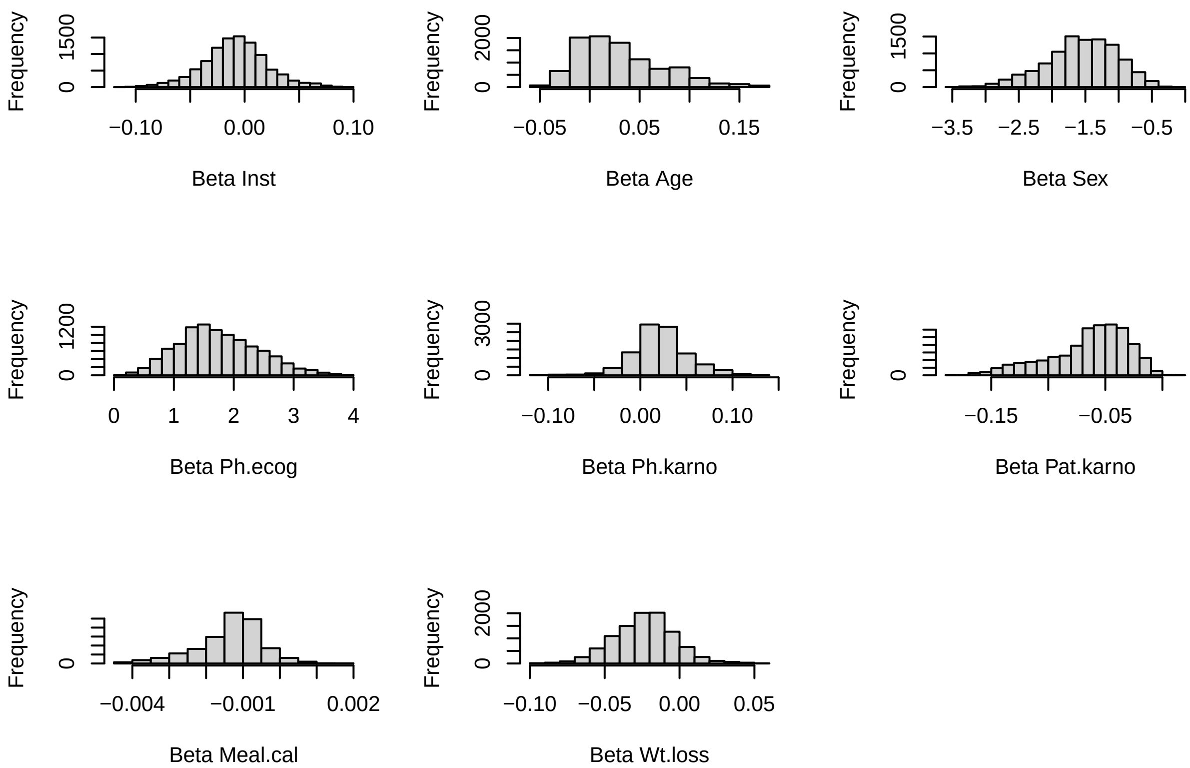
```
## [1] 0.6381377
```

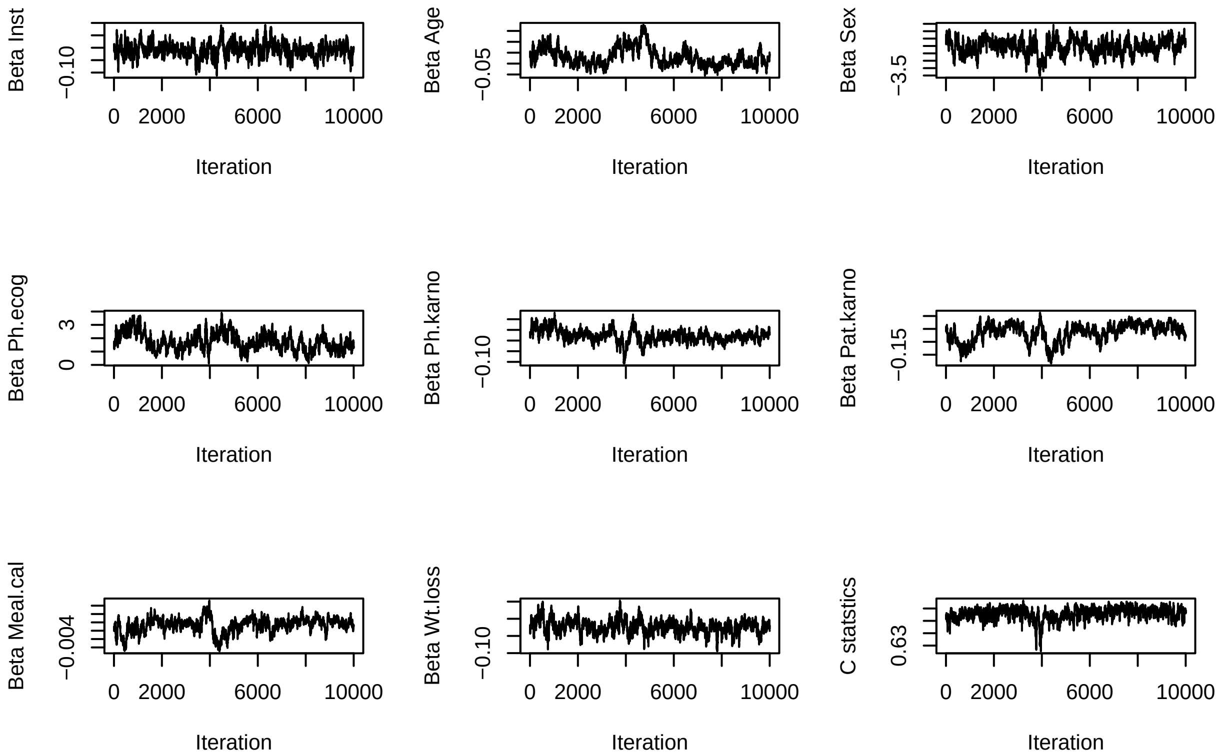
```
system.time({
  eta = 2*length(Y)
  result_lung2 = MH_Sampling(Y,delta,tau,
                        A,beta0,sigma0,
                        var.prop,m,
                        B,eta,Wmat_option)

})
```

```
##    user  system elapsed
##   51.76    5.05   82.11
```

```r
# Acceptance Rate
result_lung2$accept_rate
```

```
## [1] 0.5728182
```






```r
Wmat = HarrellC_Wmat(Y,delta,tau)
# Average Theta C-Stat
HarrellC(colSums(result_lung2$THETA),Wmat)
```

```
## [1] 0.6620572
```

```r
# Average C-Stat
mean(result_lung2$C_stat)
```

```
## [1] 0.6560137
```

## 2. Make matrix K more efficient

Mechanism:

$$K(x_i\ ,\ x_j) = exp(-\frac{1}{2}\sum_{k=1}^{p}\frac{(x_{ik}-x_{jk})^2}{\lambda_k}) = exp(-\frac{1}{2}\sum_{k=1}^{p}\frac{x_{ik}^2+x_{jk}^2-2x_{ik}x_{jk}}{\lambda_k})$$

```r
# Gbsg Data
A <- model.matrix(rfstime ~ -1+ age + meno + size + grade + er + hormon,
                  data=gbsg)
lambda0 = rep(1,dim(A)[2])
```

```r
# Simulated Data
n <- 10000  # Number of data points
d <- 6    # Number of dimensions
X <- matrix(rnorm(n * d), n, d)  # Sample data matrix
```

```r
Old_matrix_K <- function(X,lambda){
  n = dim(X)[1]
  p = dim(X)[2]
  cov_K = matrix(0,n,n)
  for (i in 1:n){
    cov_K[i,] = exp(-0.5*colSums((t(X)[,i]-t(X))^2/lambda))
  }
  return (cov_K)
}

New_matrix_K <- function(X,lambda){

  return(exp(-0.5*(outer(rowSums(t(t(X)^2/lambda)),
                         rowSums(t(t(X)^2/lambda)),'+')-
              2*(t(t(X)/sqrt(lambda)))%*%(t(X)/sqrt(lambda)))))
}
```

```r
# Compute the kernel for A
system.time({old_K_A = Old_matrix_K(A,lambda0)})
```

```
##    user  system elapsed
##    0.16    0.04    0.26
```

```r
system.time({new_K_A = New_matrix_K(A,lambda0)})
```

```
##    user  system elapsed
##    0.08    0.00    0.08
```

```r
# Compute the kernel for simulated X
system.time({old_K_X = Old_matrix_K(X,lambda0)})
```

```
##    user  system elapsed
##   45.03    9.11   63.33
```

```r
system.time({new_K_X = New_matrix_K(X,lambda0)})
```

```
##    user  system elapsed
##   24.17    1.83   31.29
```

```r
sum(new_K_A!=old_K_A)
```

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

```
sum(new_K_X!=old_K_X)
```

```
## [1] 52053505
```

Guess: some of the computation might lose precision

For example:

```
c(new_K_X[149,391],old_K_X[149,391],new_K_X[149,391] != old_K_X[149,391])
```

```
## [1] 0.1335879 0.1335879 1.0000000
```

## 3. Sample beta independently in Gaussian Process

Findings:

1.Increasing trend of C statistics?

2.Much larger average C statistics (obvious from 1) (Report the Interval)

Discussion:

In this function, I didn't use the kernel function, and I didn't sample lambda.

Reason: when computing the MH ratio, we need to compute the pdf of beta prior, which is multivariate normal with kernel covariance matrix. It's hard to compute the probability density (It's also hard to sample from MVN in this case)

Both "rmvnorm" and "dmvnorm" are inefficient.

```
# In independent case

GPMH_Sampling <- function(Y,delta,tau,
                          A,beta0,sigma0,sigma0.prop,
                          m,B,eta,
                          Wmat_option){


  accept = 0
  beta = beta0

  # What we want to record
  BETA = matrix(0,m,dim(A)[1])
  C_stat = c()


  # For safety m>B
```

```
if (B>m){
  B = 0
}


# 0 means we use Harrell C statistics
# 1 means we use Uno C statistics
if (Wmat_option==0){
  Wmat <- HarrellC_Wmat(Y, delta, tau)
}else if (Wmat_option==1){
  Wmat <- UnoC_Wmat(Y, delta, tau)
}else{  # Other Possible C index...
  Wmat <- HarrellC_Wmat(Y, delta, tau)
}


for (i in 1:m){


  ##############################################
  # Sample beta independently
  beta.p = t(rnorm(dim(A)[1],beta,sigma0.prop))
  #############################################


  # Compute C-statistics from current and last iteration
  HC.p = HarrellC(beta.p, Wmat)
  HC = HarrellC(beta, Wmat)

  # Record C-statistics from last iteration
  C_stat = c(C_stat,HC)


  # Compute log of MH ratio

  lrMH = eta*log(HC.p) +
        sum(dnorm(beta.p,beta0,sigma0,log=T))-
        eta*log(HC) -
        sum(dnorm(beta,beta0,sigma0,log=T))

    if (log(runif(1))<lrMH){
      beta = beta.p
      accept = accept + 1
    }
    BETA[i,] = beta

}

if (B == 0){
  return(list(BETA=BETA,
              accept_rate=accept/m,
              C_stat = C_stat))
}else{
```

```r
    return(list(BETA=BETA[-c(1:B),],
                accept_rate=accept/m,
                C_stat = C_stat[-c(1:B)]))
  }


}
```

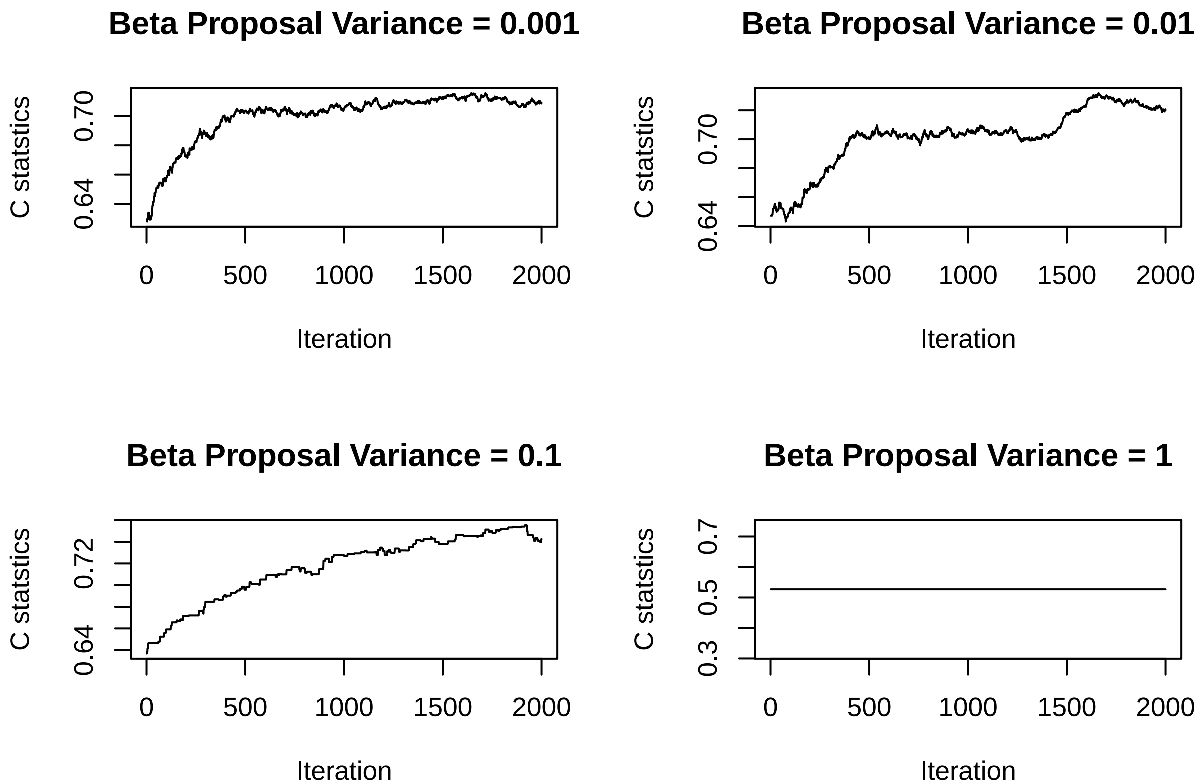
```r
m = 2200
B = 200
system.time({
  sigma0.prop = rep(0.001,dim(A)[1])
  result_GP1 = GPMH_Sampling(Y,delta,tau,
                        A,beta0,sigma0,sigma0.prop,
                        m,B,eta,
                        Wmat_option)
  sigma0.prop = rep(0.01,dim(A)[1])
  result_GP2 = GPMH_Sampling(Y,delta,tau,
                        A,beta0,sigma0,sigma0.prop,
                        m,B,eta,
                        Wmat_option)
  sigma0.prop = rep(0.1,dim(A)[1])
  result_GP3 = GPMH_Sampling(Y,delta,tau,
                        A,beta0,sigma0,sigma0.prop,
                        m,B,eta,
                        Wmat_option)
  sigma0.prop = rep(1,dim(A)[1])
  result_GP4 = GPMH_Sampling(Y,delta,tau,
                        A,beta0,sigma0,sigma0.prop,
                        m,B,eta,
                        Wmat_option)

})
```

```
##    user  system elapsed
##  424.07  213.61  829.45
```

For simplicity, the interval of C statistics is given by:

$$Average\ C\ \pm\ 1.96 * \frac{SD(C)}{\sqrt{n}}$$

```
##   Beta_Prop_Var Acceptance_Rate Average_C_STAT LB_C_STAT UB_C_STAT
## 1         0.001    0.6736363636      0.7002667 0.7002186 0.7003148
## 2         0.010    0.6672727273      0.7016817 0.7016228 0.7017405
## 3         0.100    0.0822727273      0.7168723 0.7167879 0.7169566
## 4         1.000    0.0004545455      0.5267985 0.5267985 0.5267985
```


Beta Proposal Variance = 0.001
Beta Proposal Variance = 0.01
Beta Proposal Variance = 0.1
Beta Proposal Variance = 1
C statstics
Iteration
0
500
1000
1500
2000
0.64
0.70
0.72
0.3
0.5
0.7