Joseph High
Hopkins ID: 9E1FDC

HW 3 R Code and Output

Problem 1

R Code:

```
#Problem 1
n = 10000;
x= numeric(n);
f= numeric(n);
g= numeric(n);
x= rnorm(n);
g= dnorm(x);
for (i in 1:n) {
  y= runif(1, 0, 1);
  if (y<= 0.3)
  {
    f[i] = dbeta(x[i], 5, 2);
  }
  else
  {
    f[i] = dbeta(x[i], 2, 8);
  }
}
expected.value <- sum(x*f/g)/sum(f/g)

#Computing the probability that the random variable is in the interval
(0.35,0.55)
for (i in 1:n) {
  temp <- runif(1, 0, 1);
  if (temp <= 0.3) {
    f[i] = rbeta(1, 5, 2);
  } else {
    f[i] = rbeta(1, 2, 8);
  }
}
prob <- sum(f >= 0.35 & f <= 0.55)/n
```

Results:

```
> expected.value
[1] 0.3637159
> prob
[1] 0.1205
>
```

Problem 3

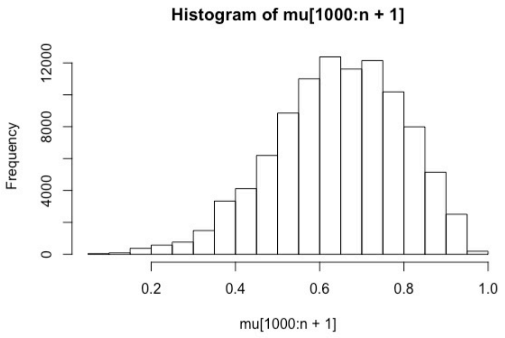
R Code:

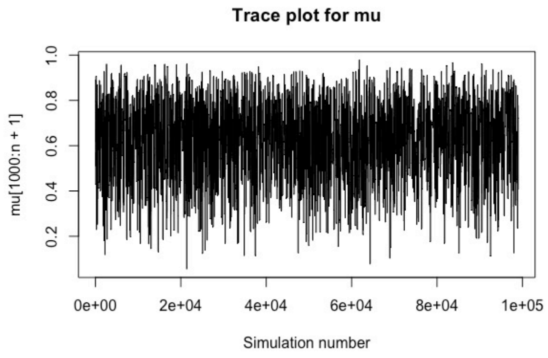
```
#Problem 3
x=c(2.3656491, 2.4952035, 1.0837817, 0.7586751, 0.8780483, 1.2765341, 1.4598699,
      0.1801679, -1.0093589, 1.4870201, -0.1193149, 0.2578262)
n = 100000
set.seed(123)
mu= numeric(n)
tau= numeric(n)
p=numeric(n)
mu[1] = rbeta(1, 2, 2)
tau[1] = rlnorm(1, 1, 10)
p[1] = 1/(sqrt(tau[1]*2*pi)^length(x))*exp(-sum((x-mu[1])^2)/(2*tau[1]))*dbeta(mu[1],
2, 2)*dlnorm(tau[1], 1, 10);
for (i in 1:n) {
  mu_star = rbeta(1, 2, 2)
  tau_star = rlnorm(1, 1, 10);
  p_star = 1/(sqrt(tau_star*2*pi)^length(x))*exp(-sum((x-
mu_star)^2)/(2*tau_star))*dbeta(mu_star, 2, 2)*dlnorm(tau_star, 1, 10)
  theta = min(p_star/p[i], 1)
  U = runif(1)
  if(U < theta) {
    mu[i+1] = mu_star
    tau[i+1] = tau_star
    p[i+1] = p_star
    }
  else {
    mu[i+1] = mu[i]
    tau[i+1] = tau[i]
    p[i+1] = p[i]
    }
  }
#posterior prob of mu>=0.5
sum(mu >= 0.5)/n
hist(mu[1000:n+1])
plot(mu[1000:n+1], type='l', main="Trace plot for mu", xlab='Simulation number')
hist(tau[1000:n+1])
plot(tau[1000:n+1],type='l', main="Trace plot for tau", xlab='Simulation number')
plot(mu[1000:n+1], tau[1000:n+1], type='l', main='Trace plot for mu and tau')
acf(mu)
acf(tau)
```

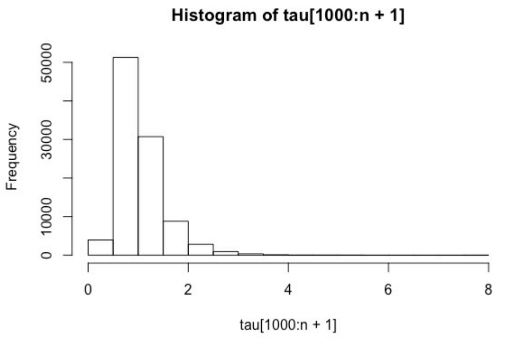
Result:

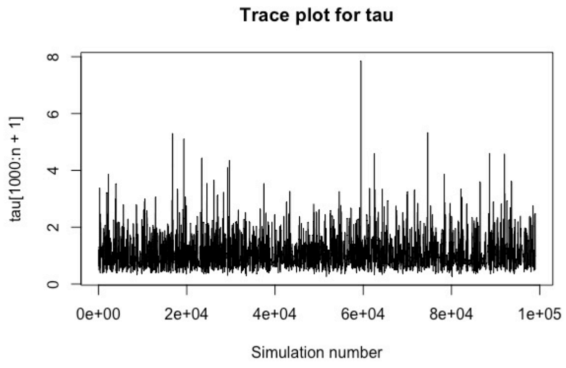
```
[1] 0.82797
```

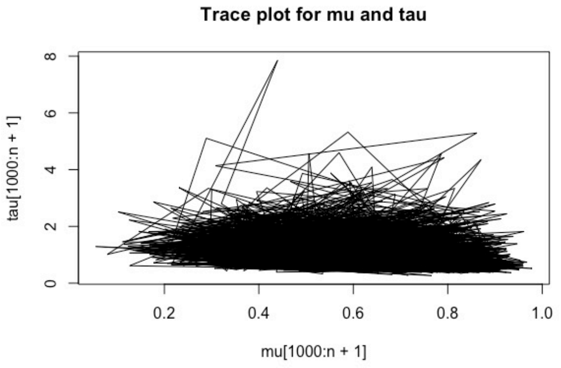
Problem 3 Graphics:

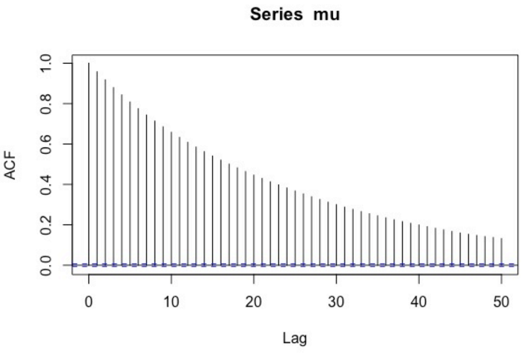


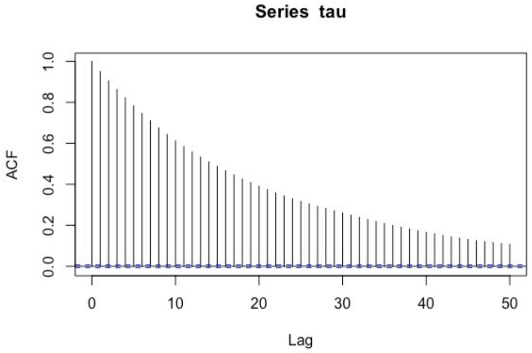












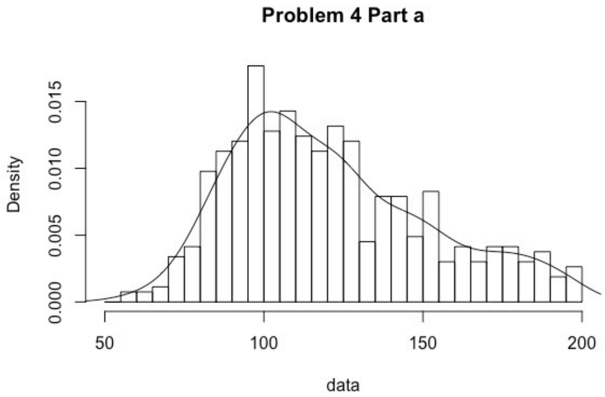
Problem 4

Part a

R Code:

```
#part a
glucose = read.table("glucose.dat", header = FALSE);
data=as.matrix(glucose)
data=as.numeric(data)
hist(data,breaks=seq(50,200,5),freq=FALSE,main="Problem 4 Part a")
lines(density(data))
```

Result/Graphic:



Part c

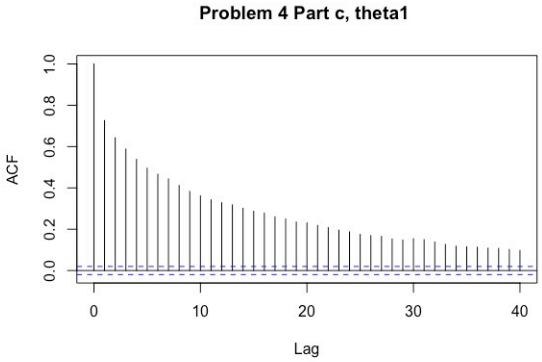
R Code:

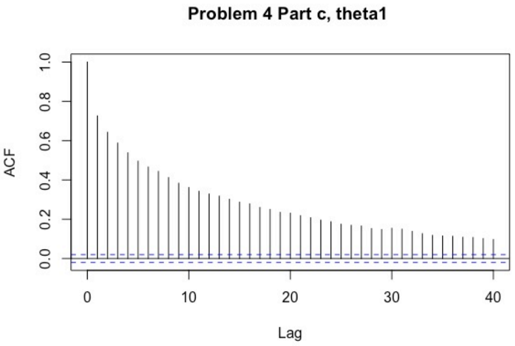
```
#part c
y=data
set.seed(0)
n=length(y)
iter=10000
a=1
b=1
mu0=120
tao0.sq=200
sigma0.sq=1000
nu0=10
x=matrix(0,iter, n)
p=numeric(iter)
theta1=numeric(iter)
theta2=numeric(iter)
sigma1.sq=numeric(iter)
sigma2.sq=numeric(iter)
p[1]=rbeta(1, a, b)
x[1,]=rbinom(n,1,p[1])
theta1[1]=rnorm(1,mu0,sqrt(tao0.sq))
theta2[1]=rnorm(1,mu0,sqrt(tao0.sq))
sigma1.sq[1]=1/rgamma(1, nu0/2, nu0*sigma0.sq/2)
sigma2.sq[1]=1/rgamma(1, nu0/2, nu0*sigma0.sq/2)
for (i in 2:iter)
{
  for (j in 1:n)
  {
    y1=dnorm(y[j], theta1[i-1], sqrt(sigma1.sq[i-1]))
    y2=dnorm(y[j], theta2[i-1], sqrt(sigma2.sq[i-1]))
    x[i,j]=rbinom(1,1,(p[i-1]*y1)/(p[i-1]*y1+(1-p[i-1])*y2))
  }
  c=sum(x[i,])
  p[i]=rbeta(1,a+c,b+n-c)
  y_1.bar=mean(y[x[i,]==1])
  mu_n=(mu0/tao0.sq+c*y_1.bar/sigma1.sq[i-1])/(1/tao0.sq+c/sigma1.sq[i-1])
  tao2_n=1/(1/tao0.sq+c/sigma1.sq[i-1])
  theta1[i]=rnorm(1, mu_n, sqrt(tao2_n))

  nu_n=nu0+c
  s2_n=sum((y[x[i,] == 1]-theta1[i])^2)/c
  sigma2_n=(nu0*sigma0.sq+c*s2_n)/nu_n
  sigma1.sq[i]=1/rgamma(1,nu_n/2,nu_n*sigma2_n/2)

  y_2.bar=mean(y[x[i,] == 0])
  mu_n=(mu0/tao0.sq+(n-c)*y_2.bar/sigma2.sq[i-1])/(1/tao0.sq+(n-c)/sigma2.sq[i-1])
  tao2_n=1/(1/tao0.sq+(n-c)/sigma2.sq[i-1])
  theta2[i]=rnorm(1, mu_n, sqrt(tao2_n))

  nu_n=nu0+(n-c)
  s2_n=sum((y[x[i,] == 0]-theta2[i])^2)/(n-c)
  sigma2_n=(nu0*sigma0.sq+(n-c)*s2_n)/nu_n
  sigma2.sq[i]=1/rgamma(1,nu_n/2,nu_n*sigma2_n/2)
}
theta_1s=rep(0,iter)
theta_2s=rep(0,iter)
for (i in 1:iter)
{
  theta_1s[i]=min(theta1[i], theta2[i])
  theta_2s[i]=max(theta1[i], theta2[i])
}
acf(theta_1s,main="Problem 4 Part c, theta1 ")
acf(theta_2s,main="Problem 4 Part c, theta2 ")
effectiveSize(theta_1s)
effectiveSize(theta_2s)
```

Results:

```
> effectiveSize(theta_1s)
    var1
418.4169
> effectiveSize(theta_2s)
    var1
230.2658
>
```

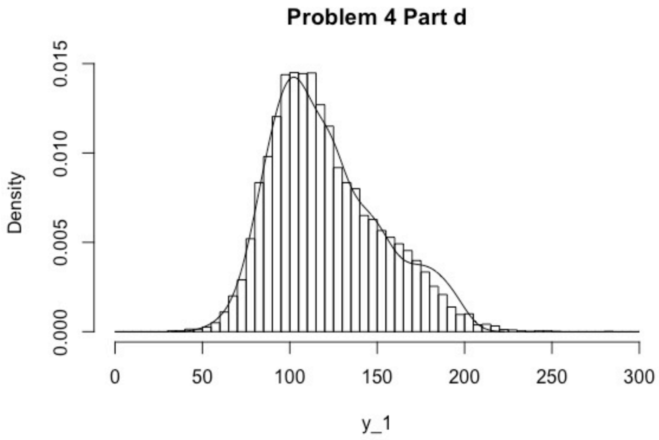
Graphics:





```
#part d
x_1=rbinom(length(p), 1, p)
y_1=numeric(iter)
for (i in 1:iter)
{
  if (x_1[i] == 1)
  {
    y_1[i]=rnorm(1, theta1[i], sqrt(sigma1.sq[i]))
  }
  else
  {
   y_1[i]=rnorm(1, theta2[i], sqrt(sigma2.sq[i]))
  }
}
hist(y_1,breaks=seq(0,300,5),freq=FALSE,main="Problem 4 Part d")
lines(density(y))
```

Graphic:

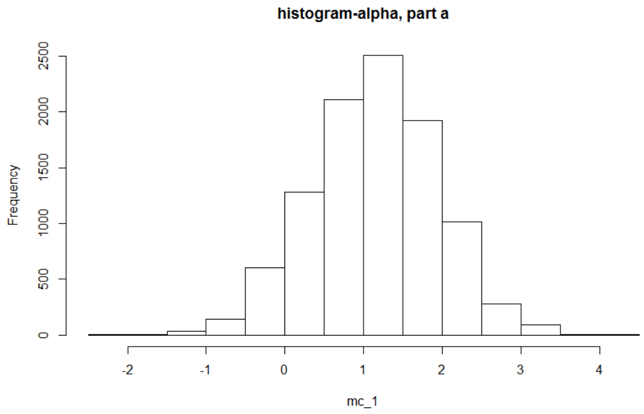


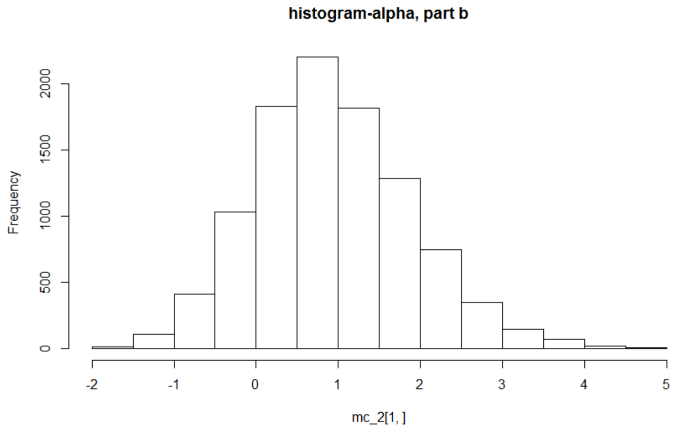
Problem 5

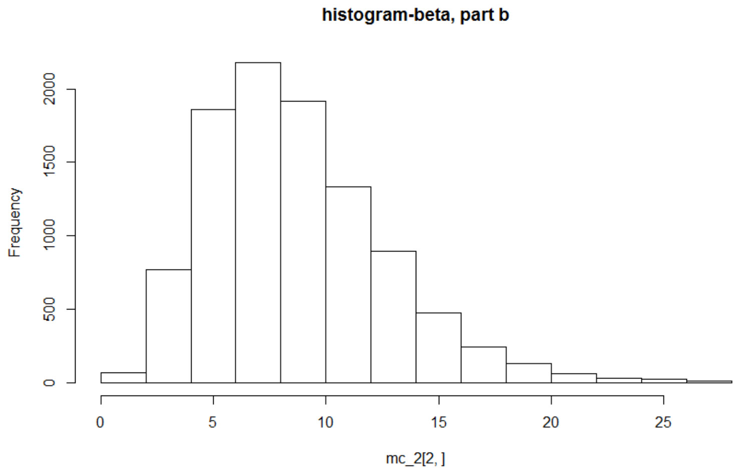
```
y=c(0,1,3,5)
n=c(5, 5, 5,5 )
x=c(-0.86,-0.30,-0.05,0.73)

#part 1 functions
part1post=function(alpha,beta,y,x,n){
  prior=dnorm(alpha,0,10)
  like=1
  for(i in 1:4){
    like=like*(inv.logit(alpha+beta*x[i]))^y[i]*(1-inv.logit(alpha+beta*x[i]))^(n[i]-
y[i])
  }
  post=prior*like
  return(post)
}

part1iter=function(alpha,beta=10,y,x,n,alphasd){
  U=runif(1,0,1)
  star=rnorm(1,alpha,alphasd)
  p=min(part1post(star,beta,y,x,n)/part1post(alpha,beta,y,x,n),1)
  if(U <= p)
  {alpha=star}
  else {alpha=alpha}
  return(alpha)
}
#part 2 functions
part2post=function(alpha,beta,y,x,n){
  prior=dnorm(alpha,0,10)*dnorm(beta,0,10)
  like=1
  for(i in 1:4){
    like=like*(inv.logit(alpha+beta*x[i]))^y[i]*(1-inv.logit(alpha+beta*x[i]))^(n[i]-
y[i])
  }
  post=prior*like
  return(post)
}

part2iter=function(param,y,x,n,sd_param){
    star1=rnorm(1,param[1],sd_param[1])
    star2=rnorm(1,param[2],sd_param[2])

    p1=min(part2post(star1,param[2],y,x,n)/part2post(param[1],param[2],y,x,n),1)
    U=runif(1,0,1)
    if(U <= p1)
    {param[1]=star1}
    else {param[1]=param[1]}
    p2=min(part2post(param[1],star2,y,x,n)/part2post(param[1],param[2],y,x,n),1)
    U=runif(1,0,1)
    if(U <= p2)
    {param[2]=star2}
    else {param[2]=param[2]}
  return(param)
}
#part 3 functions
part3post=function(param){
  prior=dnorm(param[1],0,10)*dnorm(param[2],0,10)
  like=1
  for(i in 1:4){
    like=like*(((inv.logit(param[1]+param[2]*x[i]))^y[i])*((1-
inv.logit(param[1]+param[2]*x[i]))^(n[i]-y[i])))
  }
  post=prior*like
  return(post)
}

part3iter=function(param,y,x,n,cov_mat=diag(1,2,2)){
  star=mvrnorm(1,param,cov_mat)
  U=runif(1,0,1)
  p=min(part3post(star)/part3post(param),1)
  if(U <= p)
  {param=star}
  else {param=param}
  return(param)
}
#part 4 functions
part4iter=function(param,y,x,n,cov,delta){
  normal_mean=param+delta*(theta_hat-
param)/as.numeric(dist(rbind(param,theta_hat),method = "euclidean"))
  star=mvrnorm(1,normal_mean,cov)
  U=runif(1,0,1)
  p=min((part3post(star)/part3post(param)),1)
  if(U <= p)
  {param=star}
  else {param=param}
  return(param)
}
```

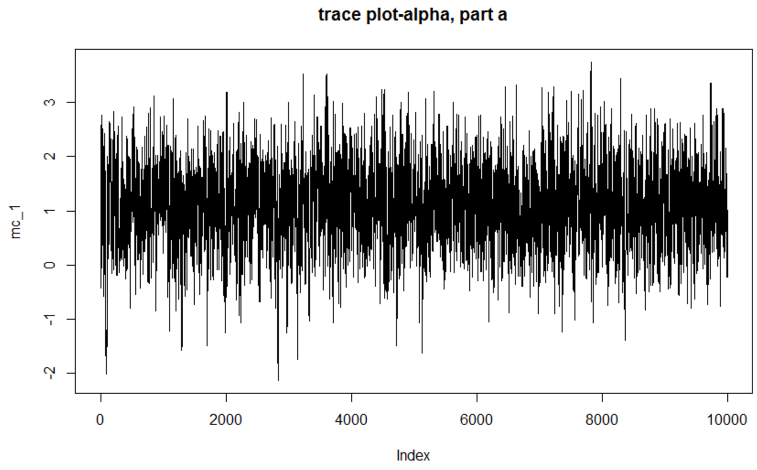
```
alphasd=1
sd_param=c(1,2)

cov=diag(1,2,2)
delta=1
N=10000
post3n=function(param){
  return(-part3post(param))
}
theta_hat=optim(c(1,1),post3n)$par

mc_1=numeric(N)
mc_2=array(NA,dim = c(2,N))
mc_3=array(NA,dim = c(2,N))
mc_4=array(NA,dim = c(2,N))
mc_1[1]=0
mc_2[,1]=c(1,10)
mc_3[,1]=c(1,10)
mc_4[,1]=c(1,10)

for (iter in 2:N){
  alpha1=part1iter(mc_1[iter-1],beta=10,y,x,n,alphasd)
  mc_1[iter]=alpha1
  param2=part2iter(mc_2[,iter-1],y,x,n,sd_param)
  mc_2[,iter]=param2
  param3=part3iter(mc_3[,iter-1],y,x,n,cov_mat=diag(1,2,2))
  mc_3[,iter]=param3
  param4=part4iter(mc_4[,iter-1],y,x,n,cov,delta)
  mc_4[,iter]=param4
}

hist(mc_1, main='histogram-alpha, part a')
hist(mc_2[1,], main='histogram-alpha, part b')
hist(mc_2[2,], main='histogram-beta, part b')
plot(mc_1, type='l')
plot(mc_2[1,], type='l')
plot(mc_2[2,], type='l')
plot(mc_3[1,], type='l')
plot(mc_3[2,], type='l')
plot(mc_4[1,], type='l')
plot(mc_4[2,], type='l')
acf(mc_1)
acf(mc_2[1,])
acf(mc_2[2,])
acf(mc_3[1,])
acf(mc_3[2,])
acf(mc_4[1,])
acf(mc_4[2,])
```

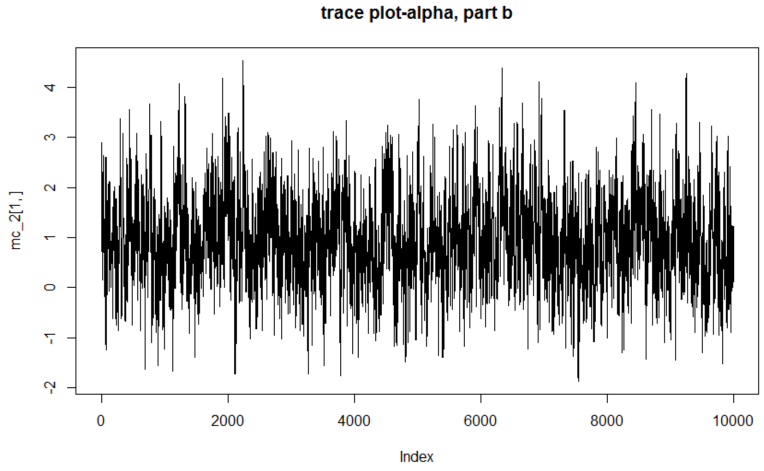
Results & Graphics:

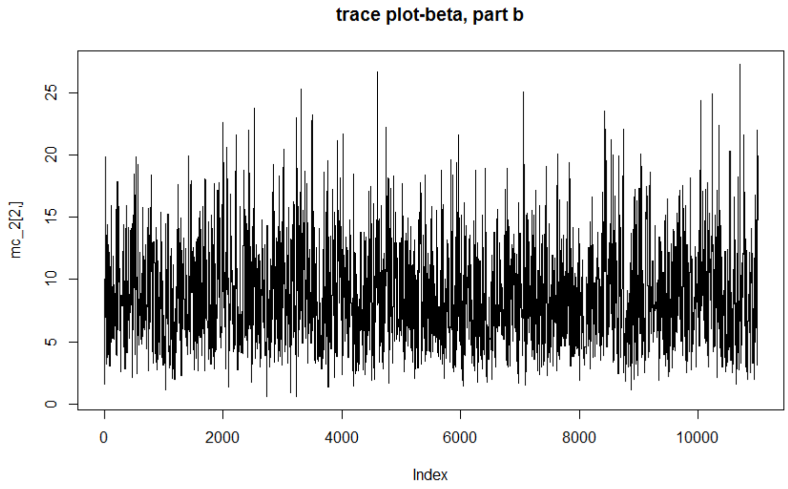




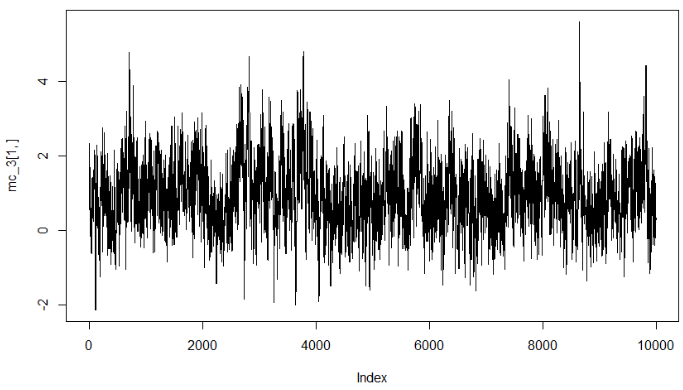



trace plot-alpha, part a
mc_1
-2
-1
0
1
2
3
0
2000
4000
6000
8000
10000
Index

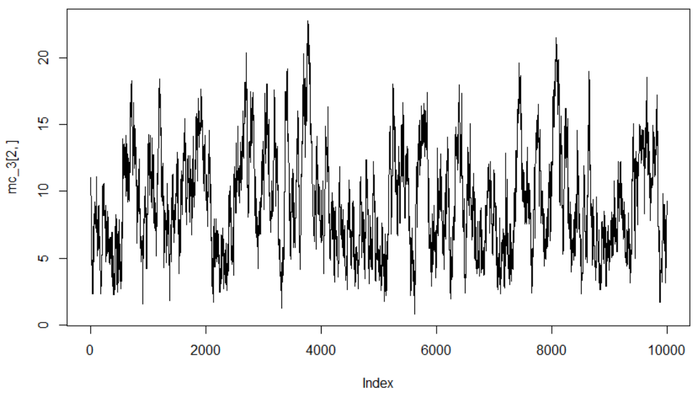

trace plot-alpha, part b
mc_2[1,]
-2
-1
0
1
2
3
4
0
2000
4000
6000
8000
10000
Index


trace plot-beta, part b
mc_2[2,]
0
5
10
15
20
25
0
2000
4000
6000
8000
10000
Index

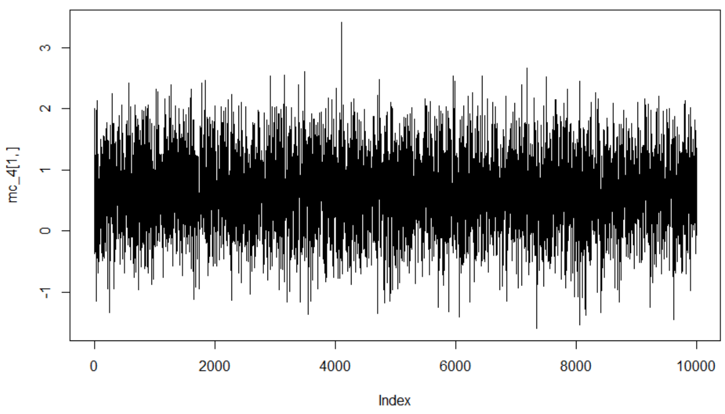
trace plot-alpha, part c



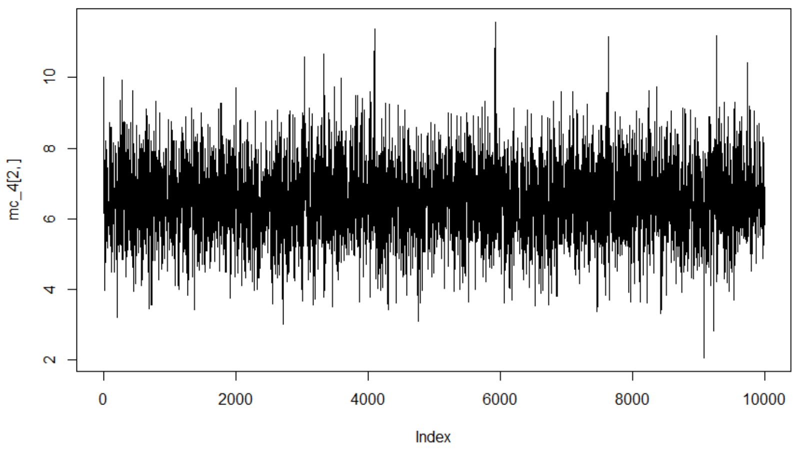
trace plot-beta, part c



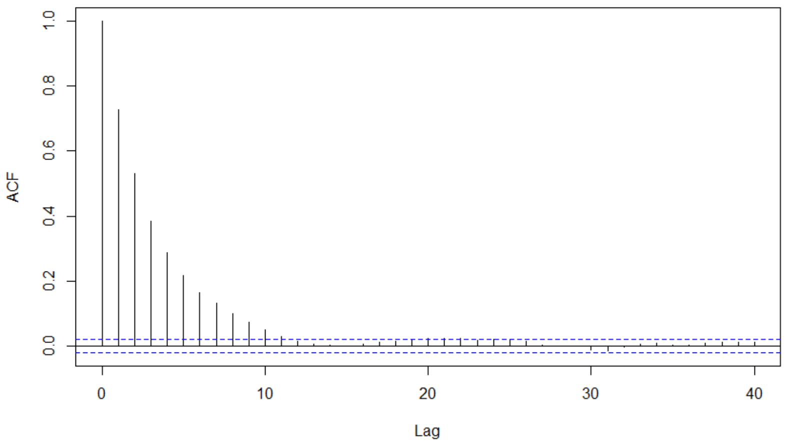
trace plot-alpha, part d



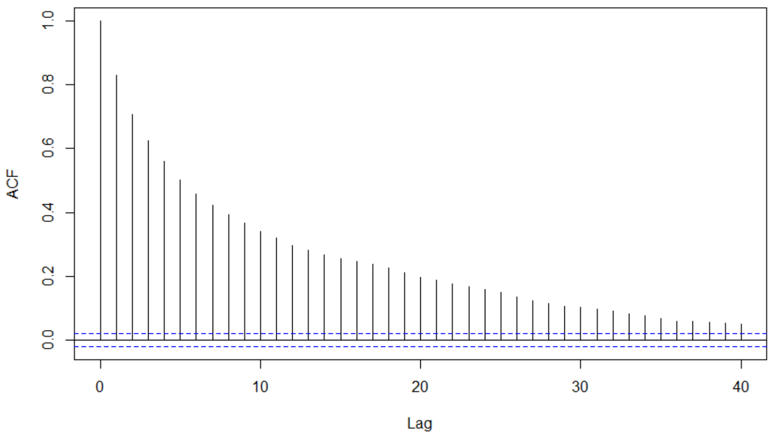
trace plot-beta, part d



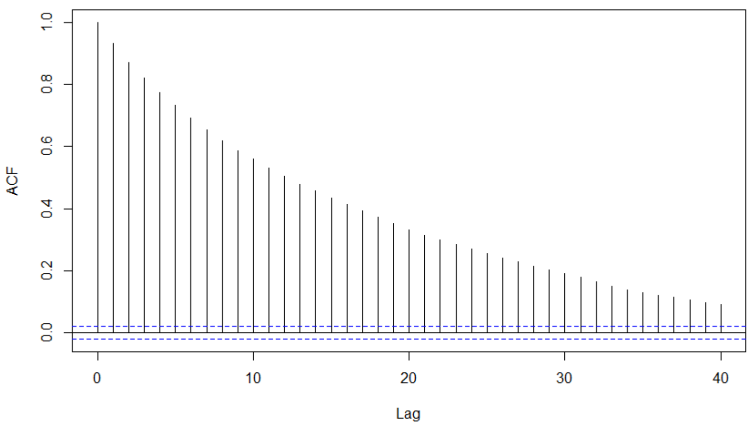
acf plot-alpha, part a



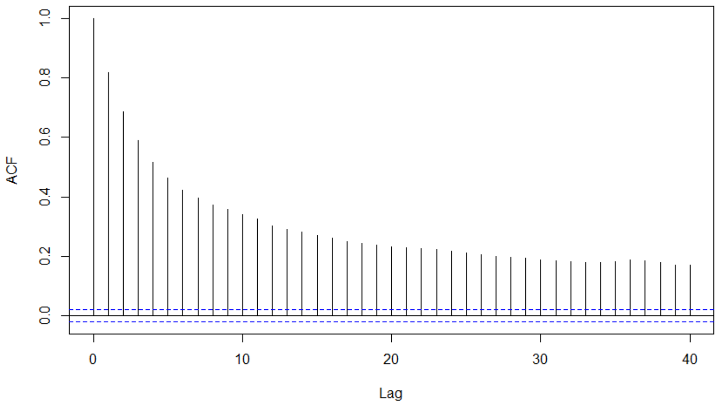
acf plot-alpha, part b



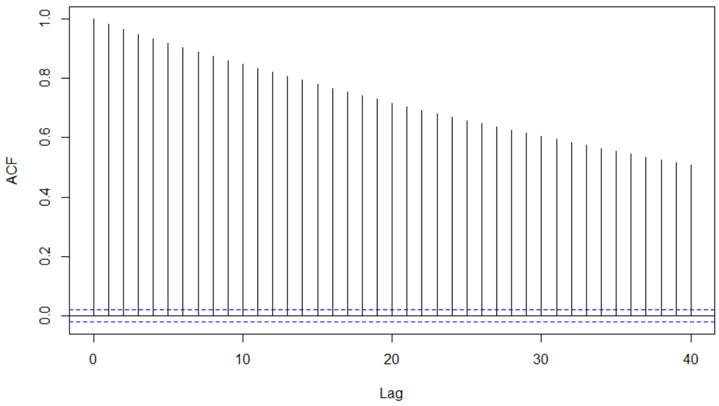
acf plot-beta, part b

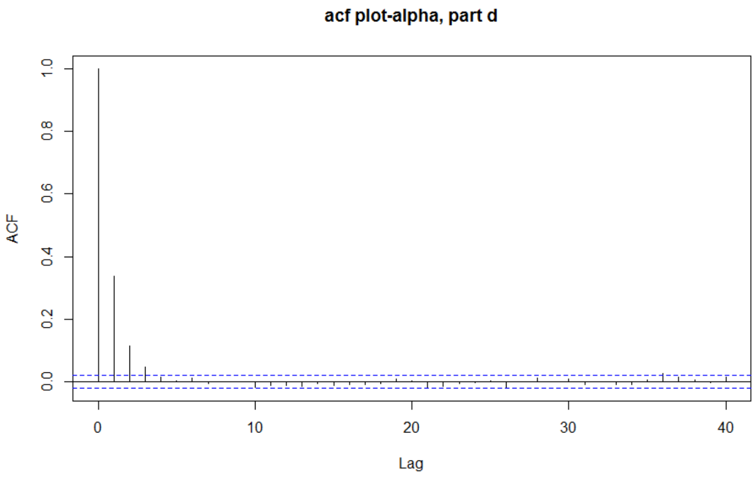


acf plot-alpha, part c

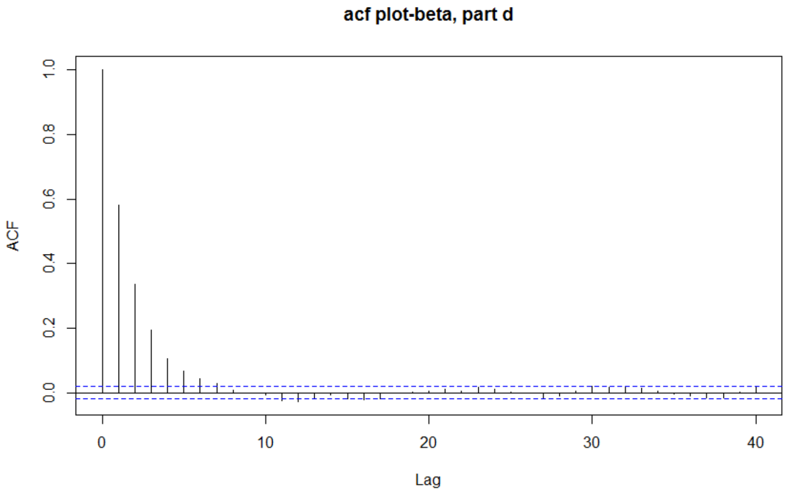


acf plot-beta, part c





acf plot-alpha, part d



acf plot-beta, part d