HW7

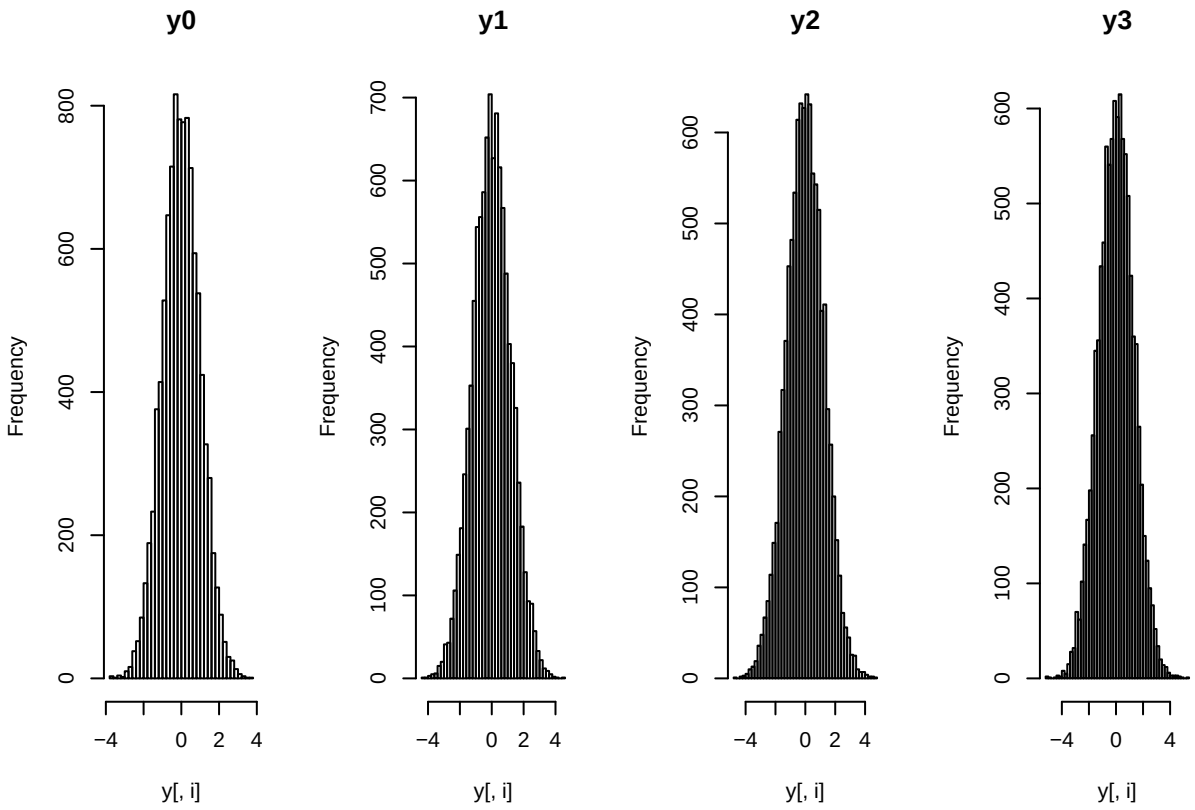
Zach White

April 10, 2017

Problem 2

Part B

```
T = 3
rho = .7
sigma2 = 1

n.iter = 10000
burnin = 1000
y = matrix(0,ncol = T+1,nrow = n.iter + burnin)
y[1,] = 0

for(i in 2:n.iter+ burnin){
  y[i,1] = rnorm(1,(rho*y[i-1,2]) / (1+rho^2), sqrt(sigma2 / (1+rho^2)) )
  y[i,2] = rnorm(1,rho*(y[i,1] + y[i-1,3]) / (1+rho^2), sqrt(sigma2 / (1+rho^2)))
  y[i,3] = rnorm(1,rho*(y[i,2] + y[i-1,4]) / (1+rho^2),sqrt(sigma2 / (1+rho^2)))
  y[i,4] = rnorm(1,rho*y[i,2],sqrt(sigma2))
}
y = y[-c(1:1000),]
par(mfrow = c(1,4))
ys = c("y0","y1","y2","y3")
for(i in 1:4){
  hist(y[,i],breaks = 50,main = ys[i])
}
```



The variance of the histogram might increase for the values of y_3, but it isn't very drastic in this case.

Part C

In order for this to be constrained as strictly increasing, I will use metropolis-hastings algorithm. I will write a general function for this Metropolis-Hastings because we will run this for $T = 3, 5, 10$.

```
cand.v = 1
MH2c = function(T,reps){
  burn = 1e3
  y = matrix(0,ncol=(T+1),nrow=(reps+burn))
  y[1,] = rep(0,T+1)
  acc = numeric(T+1)
  for(i in 2:(reps+burn)){
    cand = rnorm(1, y[i-1,1], cand.v)
    y[i,1] = y[i-1,1]
    if(cand < y[i-1,2]){
      rat = dnorm(cand ,(rho* y[i-1,2])/(1 + rho^2 ) , sqrt(sigma2/(1 + rho^2 )) ,log=T) - dnorm(y[i-1,
    if(rat > log(runif(1)) ){ y[i,1] = cand ; acc[1] = acc[1] + 1     }
  }
  for(j in 2:T){
    cand = rnorm(1, y[i-1,j], cand.v)
    y[i,j] = y[i-1,j]
    if(cand > y[i,j-1] & cand < y[i-1,j+1]){
      rat = dnorm(cand ,rho*(y[i,j-1] + y[i-1,j+1]) /(1 + rho^2 ) , sqrt(sigma2/(1 + rho^2)) ,log=T) -
    if(rat > log(runif(1)) ){ y[i,j] = cand ; acc[j] = acc[j] + 1     }
```

```
  }
}

  cand = rnorm(1, y[i-1,T+1], cand.v)
  y[i,T+1] = y[i-1,T+1]
  if(cand > y[i,T]){
    rat = dnorm(cand ,rho* y[i,T] , sqrt( sigma2) ,log=T) - dnorm(y[i-1,T+1] , rho* y[i,T] , sqrt( sigma
  if(rat > log(runif(1)) ){ y[i,T+1] = cand ; acc[T+1] = acc[T+1] + 1 }
  }
}
  return(list(y=y[-(1:burn),],accept = acc/(reps + burn)))
}
out3 = MH2c(3,1e5)
out5 = MH2c(5,1e5)
out10 = MH2c(10,1e5)

# We also use a rejection sampler
cov.ar = function(T,rho =0.7,sig2=1){
  prec = matrix(0,ncol=(T+1),nrow=(T+1))
  diag(prec) = c(rep(rho^2+1,T),1)
  prec[1,2] = -rho
  prec[T+1,T] = -rho
  for(i in 2:(T)){
    prec[i,c(i-1,i+1)] = -rho
  }
  return(solve(prec))
}

co3 = cov.ar(3)
co5 = cov.ar(5)
co10 = cov.ar(10)
samp3 = mvrnorm(1e5,rep(0,4),co3)
samp5 = mvrnorm(1e5,rep(0,6),co5)
samp10 = mvrnorm(1e5,rep(0,11),co10)
```

Part D

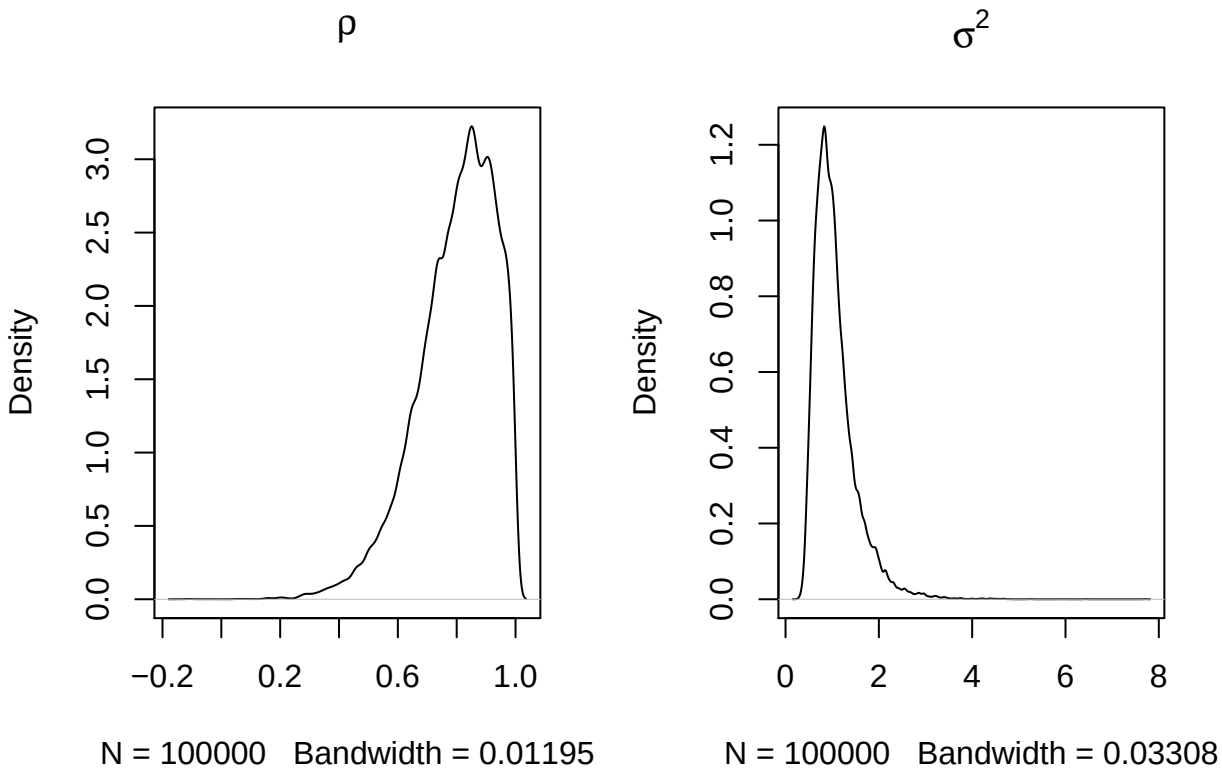
I will now simulate a sequence from the AR(1) process. We set a prior on ρ and σ^2. I don't know much about ρ, but I do know that it is between -1 and 1, and thus I will set a uniform prior $\rho \sim U(-1, 1)$ and for $\frac{1}{\sigma^2} = \phi = \text{Gamma}(2, 2)$.

```
reps = 1e5
burn = 1e3
rhos = numeric(reps + burn)
phi = numeric(reps + burn)
acc= c(0,0)
phi[1] = 1
rhos[1] = .5
y.real = samp10[2,]
#plot(0:10,y.real,xlab="t",ylab="y",type="o")
p.rho = function(x) dunif(x,-1,1,log=TRUE)
p.phi = function(x) dgamma(x,2,2,log=TRUE)
log.like = function(y=y.real,rho,sig2){
```

```
dnorm(y[1],0,sqrt(sig2),log=TRUE) + sum(dnorm(y[2:11],rho*y[1:10],sqrt(sig2),log=TRUE))
}
log.post= function(rho,sig2) log.like(y.real,rho,sig2) + p.rho(rho) + p.phi(1/sig2)
s.r = 1
s.p = 2
for( i in 2:(reps+burn)){
cand = rnorm(1, rhos[i-1], s.r)
rhos[i] = rhos[i-1]
if( cand > -1 & cand <1){
rat = log.post(cand,1/phi[i-1]) - log.post(rhos[i-1],1/phi[i-1])
if(rat > log(runif(1)) ){ rhos[i] = cand ; acc[1] = acc[1] + 1 }
}
cand = rnorm(1, phi[i-1], s.p)
phi[i] = phi[i-1]
if( cand > 0){
rat = log.post(rhos[i],1/cand) - log.post(rhos[i],1/phi[i-1])
if(rat > log(runif(1)) ){ phi[i] = cand ; acc[2] = acc[2] + 1 }
}
}
```

The following are posterior distribtions for both ρ and σ^2.

```
fin.rhos = rhos[-(1:burnin)]
fin.sigma2 = 1 / phi[-(1:burnin)]
mean.rho = mean(fin.rhos)
quants.rho = quantile(fin.rhos, c(.025,.975))
mean.sigma = mean(fin.sigma2)
quants.sigma = quantile(fin.sigma2,c(.025,.975))

par(mfrow =c(1,2))
plot(density(fin.rhos),main = expression(rho))
plot(density(fin.sigma2),main = expression(sigma^2))
```



The posterior expectation for $\rho = 0.7991126$, and the credible interval is 0.4900101, 0.987496.

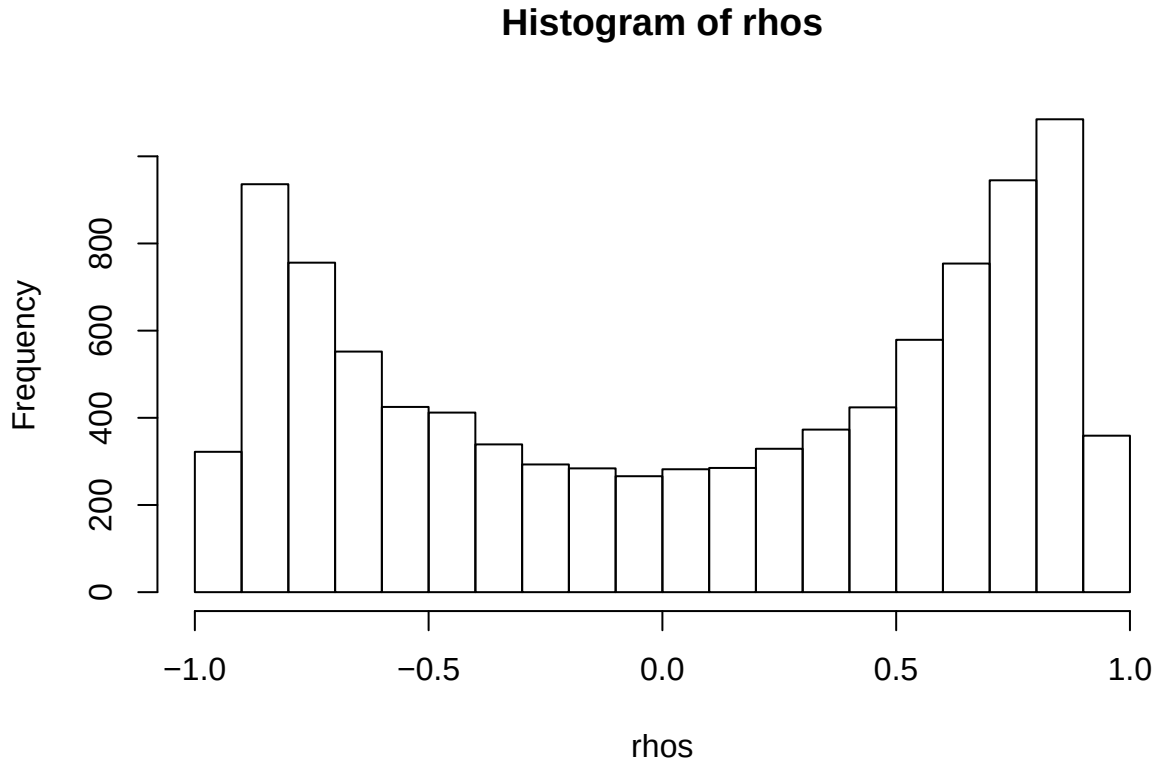
The posterior expectation for $\sigma^2 = 1.0561592$, and the credible interval is 0.489427, 2.2089722.

Problem 4

Part A

```
y1 = c(1,1,-1,-1,2,2,-2,-2,NA,NA,NA,NA)
y2 = c(1,-1,1,-1,NA,NA,NA,NA,2,2,-2,-2)
y = cbind(y1,y2)
n = nrow(y)

n.iter = 10000
burn.in = 1000

dcomp = function(mat){
  return(mat[1,2] / sqrt(mat[1,1] * mat[2,2]))
}

mvn.cond = function(start = 0,n.iter = 10000){
  rhos = numeric(n.iter + burn.in)
  rhos[1] =start
  S = matrix(c(1.01,start,start,1.01),ncol =2)
```

```
  for(i in 2:(n.iter+burn.in  )){
    y.inc = y

    for(j in 5:8){
      y.inc[j,2] = rnorm(1,S[2,1]*y.inc[j,1]/S[1,1] , sqrt(S[2,2] - S[2,1]^2 /S[1,1]))
    }
    for(j in 9:12){
      y.inc[j,1] = rnorm(1,S[2,1]*y.inc[j,2] / S[2,2],sqrt(S[1,1] - S[2,1]^2 / S[2,2]))
    }
    Sy = (n-1) *cov(y.inc)
    S = riwish(n,Sy)
    rhos[i] = dcomp(S)
  }
  return(rhos)
}
rhos = mvn.cond(n.iter = 10000)[-(1:burn.in)]
hist(rhos)
```



The variance of this methodology is significantly lower than that of the importance sampling methodology.

Part B

```
vals = c(-.99,-.50,-.25,0,.25,.50,.99)
mat.rhos = matrix(0,nrow = 10000,ncol = 7)
rho.list = list()
```

```
for(i in 1:7){
  mat.rhos[,i] = mvn.cond(start = vals[i], 10000)[-(1:burn.in)]
  test = mcmc(mat.rhos[,i])
  rho.list[[i]] = test
}
mcmc.rho = mcmc.list(rho.list)
gelman.diag(mcmc.rho)
```

```
## Potential scale reduction factors:
##
##      Point est. Upper C.I.
## [1,]          1       1.01
```

According to this, the Upper CI is 1.1, which would mean that the chain has likely converged, or at least there isn't evidence of lack of convergence.

Problem 5

Part A

```
x = c(4, 4, 7, 7, 8, 9, 10, 10, 10, 11, 11, 12, 12, 12, 12, 13, 13, 13, 13, 14, 14, 14, 14, 15, 15, 15,
y = c(2, 10, 4, 22, 16, 10, 18, 26, 34, 17, 28, 14, 20, 24, 28, 26, 34, 34, 46, 26, 36, 60, 80, 20, 26,

dat = as.data.frame(cbind(x,y))
N = nrow(dat)
x2 = dat$x^2
init.model = lm(y~ x + I(x^2), data = dat)
sum.mod = summary(init.model)
```

Part B

Use the estimates to select a candidate distribution.

We have estimates for a, b, and c, and thus we will find candidate distributions for the parameters. I will simply use normals centered around the maximum likelihood estimates, and then I will toy around with the candidate variances to make sure that the acceptance ratios are reasonable. But before I toy around with those variances, I will I just use the standard error from the linear model we have already fit. For the prior of σ^2, I use an inverse-gamma. This requires metropolis-hastings and not just metropolis because our proposal distribution are not symmetric

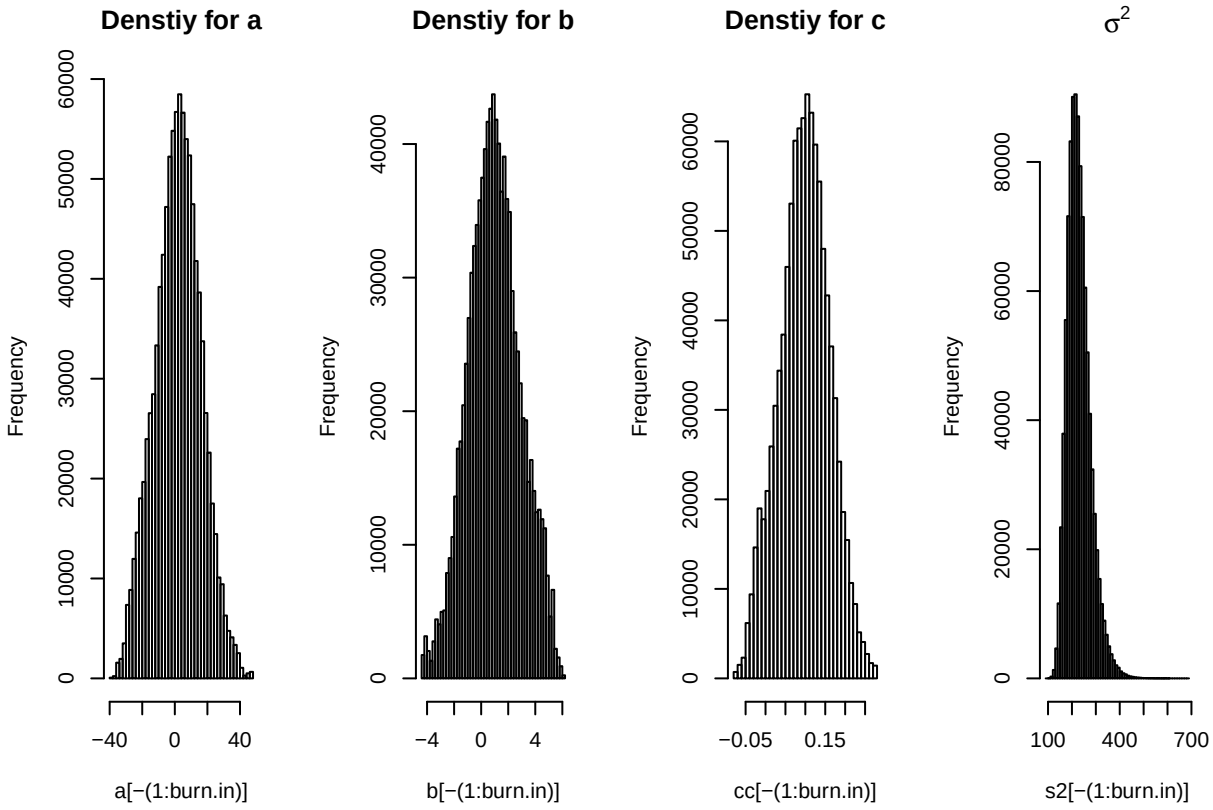
$$\sigma^2 \sim \text{IG}(N/2, N/2 \times (\hat{\sigma}^2))$$

```
est = c(summary(init.model)$coef[,1],summary(init.model)$sigma)
sds = c(summary(init.model)$coef[,2])
acc = rep(0,4)
reps = 1e6
burn = 1e3
a = b = cc = s2 = numeric(reps + burn)
a[1] = est[1]
b[1] = est[2]
cc[1] = est[3]
s2[1] = est[4]^2
```

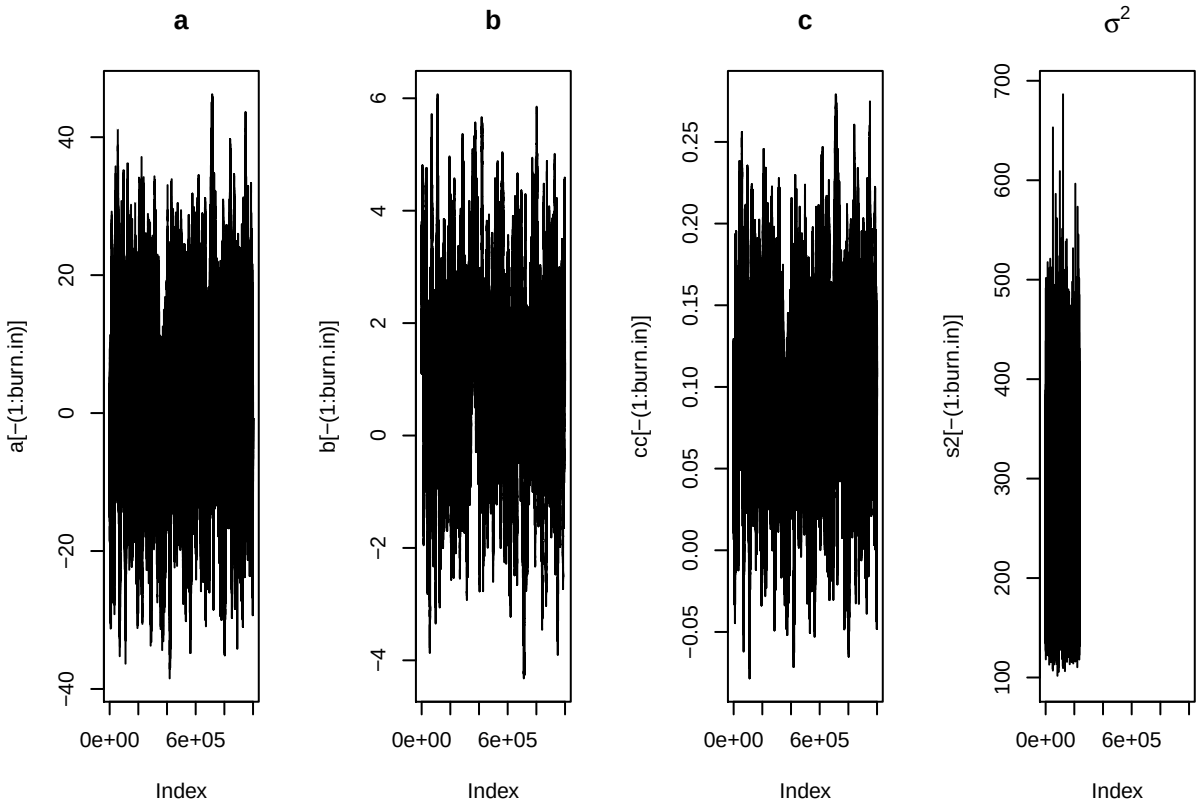
```
log.like = function(a,b,c,s2) -N/2 * log(s2) - 1/(2*s2) * sum((y - a - b*x - c*x2)^2)

alp = (N/2)/15
bet= (1/2 * N*est[4]^2)/15

for(i in 2:(reps + burn)){
  cand = rnorm(1,est[1],sds[1])
  a[i] = a[i-1]
  rat = log.like(cand,b[i-1],cc[i-1],s2[i-1]) - dnorm(cand,est[1],sds[1],log=T) -
  log.like(a[i-1],b[i-1],cc[i-1],s2[i-1]) + dnorm(a[i-1],est[1],sds[1],log=T)
  if(rat > log(runif(1)) ){ a[i] = cand; acc[1] = acc[1] + 1}
    cand = rnorm(1,est[2],sds[2])
    b[i] = b[i-1]
    rat = log.like(a[i],cand,cc[i-1],s2[i-1]) - dnorm(cand,est[2],sds[2],log=T) -
    log.like(a[i],b[i-1],cc[i-1],s2[i-1]) + dnorm(b[i-1],est[2],sds[2],log=T)
    if(rat > log(runif(1)) ){
      b[i] = cand ; acc[2] = acc[2] + 1
    }
  cand = rnorm(1,est[3],sds[3])
  cc[i] = cc[i-1]
  rat = log.like(a[i],b[i],cand,s2[i-1]) - dnorm(cand,est[3],sds[3],log=T) -
  log.like(a[i],b[i],cc[i-1],s2[i-1]) + dnorm(cc[i-1],est[3],sds[3],log=T)
  if(rat > log(runif(1)) ){
    cc[i] = cand
    acc[3] = acc[3] + 1}

  cand = 1/ rgamma(1,alp,bet)
  s2[i] = s2[i-1]
  rat = log.like(a[i],b[i],cc[i],cand) - dgamma(1/cand,alp,bet,log=T) -
  log.like(a[i],b[i],cc[i],s2[i-1]) + dgamma(1/s2[i-1],alp,bet,log=T)
  if(rat > log(runif(1)) ){
    s2[i] = cand;
    acc[4] = acc[4] + 1}
}
acc/(reps+burn)
```

```
## [1] 0.13122078 0.05924975 0.09549351 0.30095205
```

```
par(mfrow = c(1,4))
hist(a[-(1:burn.in)],breaks = 50, main = "Denstiy for a")
hist(b[-(1:burn.in)],breaks = 50, main = "Denstiy for b")
hist(cc[-(1:burn.in)],breaks = 50, main = "Denstiy for c")
hist(s2[-(1:burn.in)],breaks = 50, main = expression(sigma^2))
```




```
par(mfrow = c(1,4))
plot(a[-(1:burn.in)],type = "l",main = "a")
plot(b[-(1:burn.in)],type = "l",main = "b")
plot(cc[-(1:burn.in)],type = "l",main = "c")
plot(s2[-(1:burn.in)],type = "l",main = expression(sigma^2))
```



Part D

We now use the robust forumlation using a t-distribution. However, I there is typo in the formulation in the text. It should be as follows.

$$\frac{1}{\sigma^2}\left[1+\frac{1}{\nu}\left(\frac{y-\mu}{\sigma}\right)^2\right]$$

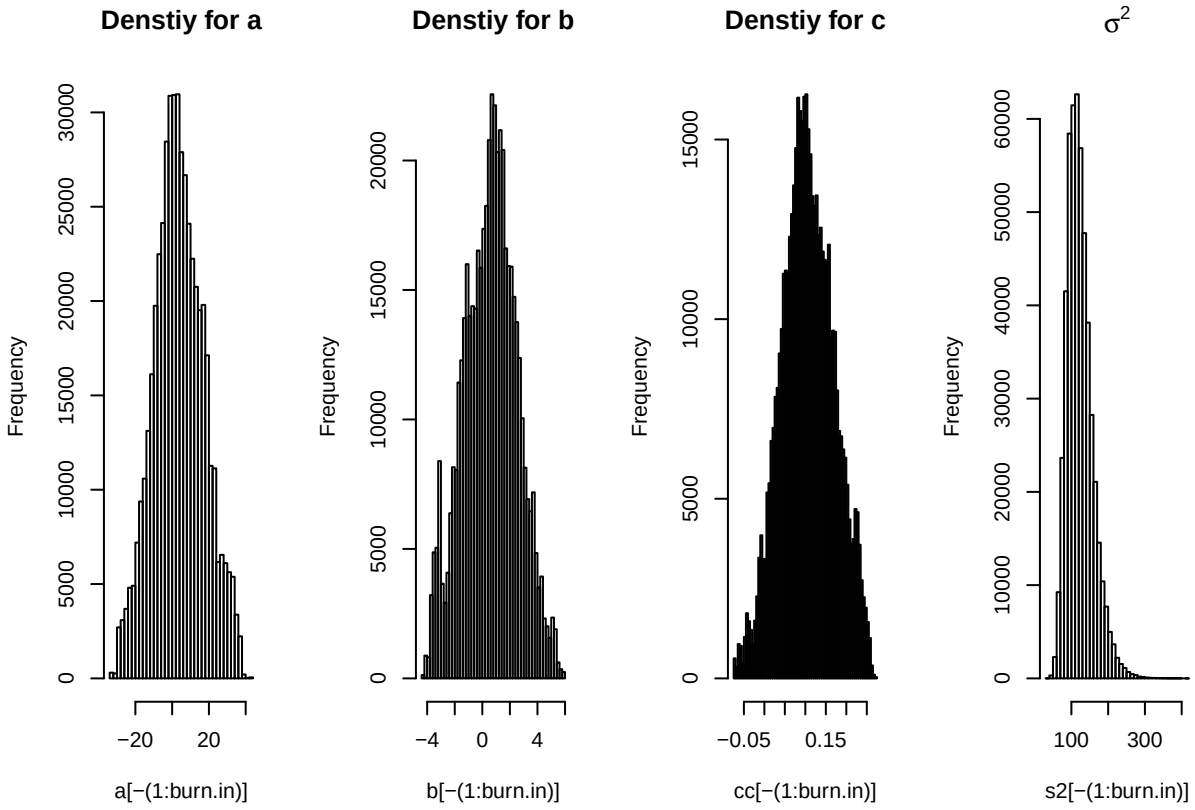
I use the same candidates as in b, and they provide reasonableacceptance ratios . This new prior obviously gives a different like likelihood

```
log.like = function(a,b,c,s2){
  -N/2 * log(s2) - 5/2 * sum( log(1 + (y-a-b*x -c*x2)^2/(4*s2)))
  }
acc = rep(0,4)
reps = 5e5
burn = 1e3
sds = c(summary(init.model)$coef[,2])
a = b = cc = s2 = numeric(reps + burn)
a[1] = est[1]
b[1] = est[2]
cc[1] = est[3]
s2[1] = 200

for(i in 2:(reps + burn)){
```

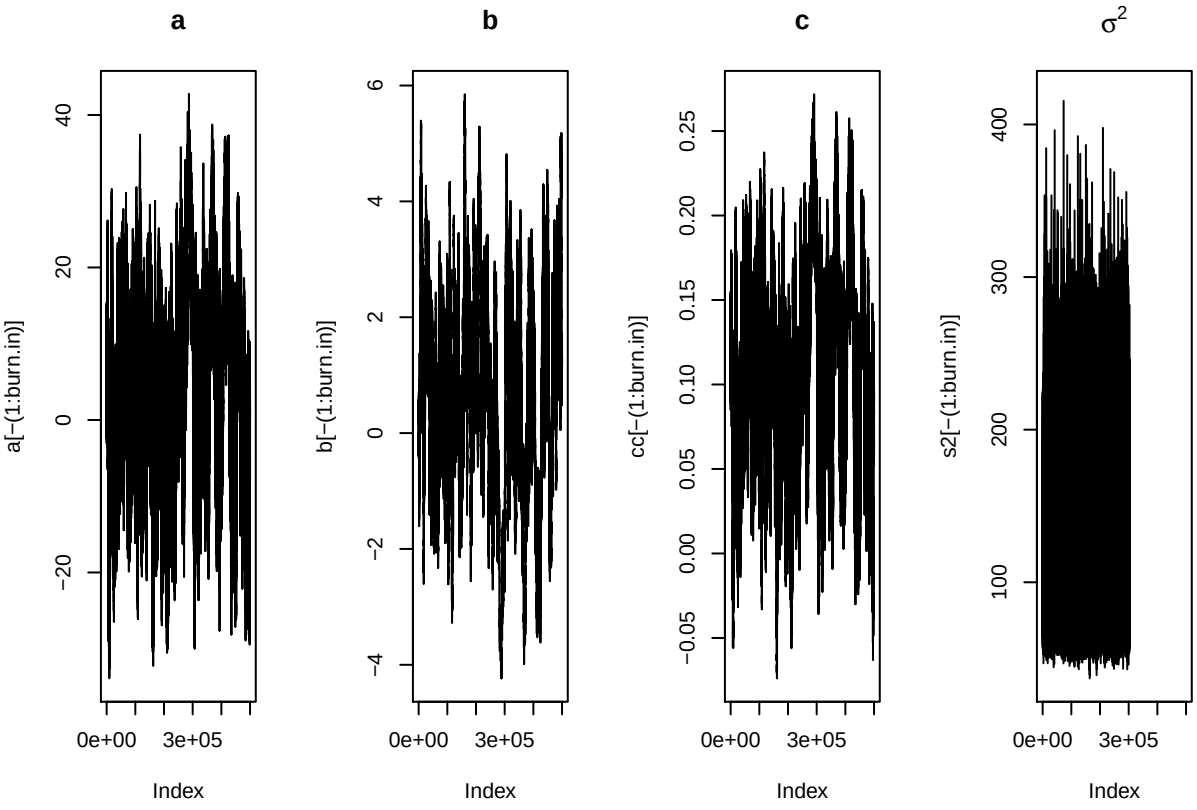
```
  cand = rnorm(1,est[1],sds[1])
  a[i] = a[i-1]
  rat = log.like(cand,b[i-1],cc[i-1],s2[i-1]) - dnorm(cand,est[1],sds[1],log=T) -
  log.like(a[i-1],b[i-1],cc[i-1],s2[i-1]) + dnorm(a[i-1],est[1],sds[1],log=T)
  if(rat > log(runif(1)) ){
    a[i] = cand
    acc[1] = acc[1] + 1
  }
  cand = rnorm(1,est[2],sds[2])
  b[i] = b[i-1]
  rat = log.like(a[i],cand,cc[i-1],s2[i-1]) - dnorm(cand,est[2],sds[2],log=T) -
  log.like(a[i],b[i-1],cc[i-1],s2[i-1]) + dnorm(b[i-1],est[2],sds[2],log=T)
  if(rat > log(runif(1)) ){
    b[i] = cand
    acc[2] = acc[2] + 1
  }

  cand = rnorm(1,est[3],sds[3])
  cc[i] = cc[i-1]
  rat = log.like(a[i],b[i],cand,s2[i-1]) - dnorm(cand,est[3],sds[3],log=T) -
  log.like(a[i],b[i],cc[i-1],s2[i-1]) + dnorm(cc[i-1],est[3],sds[3],log=T)
  if(rat > log(runif(1)) ){
    cc[i] = cand
    acc[3] = acc[3] + 1
  }

  cand = 1/ rgamma(1,alp,bet)
  s2[i] = s2[i-1]
  rat = log.like(a[i],b[i],cc[i],cand) - dgamma(1/cand,alp,bet,log=T) -
  log.like(a[i],b[i],cc[i],s2[i-1]) + dgamma(1/s2[i-1],alp,bet,log=T)
  if(rat > log(runif(1)) ){
    s2[i] = cand
    acc[4] = acc[4] + 1
  }
}
acc / reps + burn
```

```
## [1] 1000.116 1000.054 1000.088 1000.230
```

```
par(mfrow = c(1,4))
hist(a[-(1:burn.in)],breaks = 50, main = "Denstiy for a")
hist(b[-(1:burn.in)],breaks = 50, main = "Denstiy for b")
hist(cc[-(1:burn.in)],breaks = 50, main = "Denstiy for c")
hist(s2[-(1:burn.in)],breaks = 50, main = expression(sigma^2))
```




```
par(mfrow = c(1,4))
plot(a[-(1:burn.in)],type = "l",main = "a")
plot(b[-(1:burn.in)],type = "l",main = "b")
plot(cc[-(1:burn.in)],type = "l",main = "c")
plot(s2[-(1:burn.in)],type = "l",main = expression(sigma^2))
```


a
b
c
σ2
a[−(1:burn.in)]
b[−(1:burn.in)]
cc[−(1:burn.in)]
s2[−(1:burn.in)]
Index
0e+00
3e+05