MATH 559 - ASSIGNMENT 5 - SOLUTIONS

This project considers robust regression using Bayesian methods. The linear model

$$Y_i = \beta_0 + \beta_1 x_i + \epsilon_i \qquad i = 1, \ldots, n$$

*when $\epsilon_1, \ldots, \epsilon_n$ are independent and identically but **not Normally distributed** may provide a way of introducing robustness into the estimation of the regression parameters (β_0, β_1). This makes inference about the parameters robust to **outliers**.*

One robustifying assumption is to assume $Cauchy \equiv Student(1)$ errors; data generated under this assumption are depicted below.

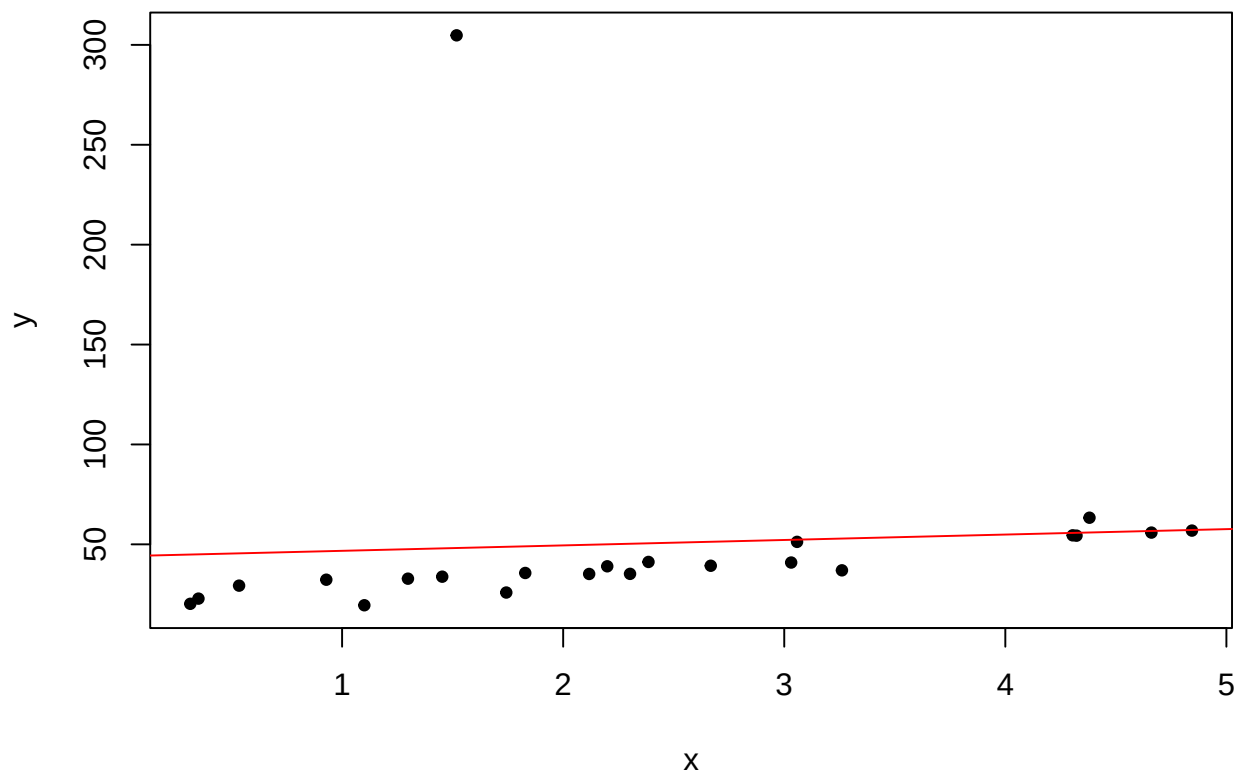
```
n<-23
set.seed(81642)
be0<-20
be1<-8
sig<-4
x<-sort(runif(n,0,5))
y<-be0+be1*x+rt(n,1)*sig
cbind(x,y)
```

```
+                x         y
+  [1,] 0.3132980  20.23565
+  [2,] 0.3502758  22.83761
+  [3,] 0.5340795  29.33431
+  [4,] 0.9285782  32.30061
+  [5,] 1.1006034  19.50876
+  [6,] 1.2979742  32.81472
+  [7,] 1.4529997  33.81671
+  [8,] 1.5180004 304.76622
+  [9,] 1.7426959  25.88330
+ [10,] 1.8285557  35.66872
+ [11,] 2.1175934  35.18131
+ [12,] 2.1994464  39.01334
+ [13,] 2.3025292  35.23135
+ [14,] 2.3859049  41.18901
+ [15,] 2.6675895  39.28907
+ [16,] 3.0313951  40.89708
+ [17,] 3.0580300  51.24407
+ [18,] 3.2609863  36.99516
+ [19,] 4.3048254  54.50957
+ [20,] 4.3220233  54.34492
+ [21,] 4.3806682  63.33452
+ [22,] 4.6609183  55.88011
+ [23,] 4.8442350  56.88866
```

```
summary(lm(y~x))$coef
```

```
+             Estimate Std. Error   t value   Pr(>|t|)
+ (Intercept) 44.045757  24.079428 1.8291862 0.08160871
+ x            2.712521   8.773261 0.3091805 0.76023093
```

```
par(mar=c(4,4,2,2))
plot(x,y,pch=19,cex=0.75)
abline(coef(lm(y~x)),col='red')
```



Problem: *For the data generated above, construct and implement a Bayesian analysis (using a suitably chosen prior) for the parameters $(\beta_0, \beta_1, \sigma^2)$ in the Cauchy regression model. Recall that the Cauchy assumption means that the density for the observed data takes the form*

$$f_{Y|X}(y|x;\beta_0,\beta_1,\sigma^2) = \frac{1}{\pi}\frac{1}{\sigma}\left\{1+\left(\frac{y-\beta_0-\beta_1 x}{\sigma}\right)^2\right\}^{-1} \qquad y \in \mathbb{R}$$

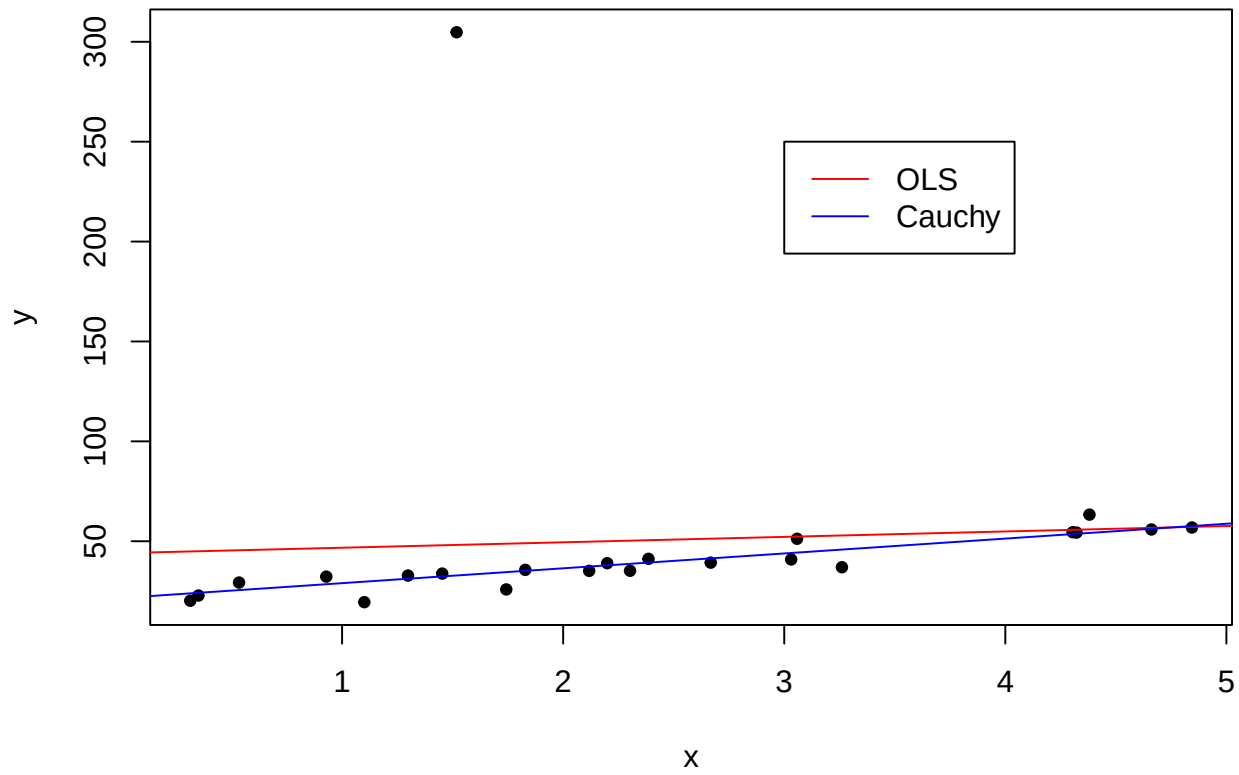
where β_0, β_1 are real-valued parameters, and $\sigma^2 > 0$, and where the x values may be treated as fixed constants.

As well as mathematical derivations that define the form of the posterior distribution, you should include in your solution annotated computational code and demonstrate that it is producing Bayesian inference. You may re-use code from the course if it is annotated appropriately.

20 MARKS

Solutions: First, we can compute the maximum likelihood estimates, implementing the optimization using `optim`, and parameterizing using $\log\sigma$ for the scale parameter.

```
log.like<-function(xp,yv,xv){
        lpv<-yv-xp[1]-xp[2]*xv
        return(sum(dcauchy(lpv,0,exp(xp[3]),log=T)))
}

fit1<-optim(c(0,0,1),log.like,yv=y,xv=x,control=list(fnscale=-1),hessian=T)
fit1$par

+ [1] 21.5814215  7.4354977  0.9001732

fit1$value

+ [1] -78.16744

par(mar=c(4,4,2,2))
plot(x,y,pch=19,cex=0.75)
abline(coef(lm(y~x)),col='red')
abline(fit1$par[1:2],col='blue')
legend(3,250,c('OLS','Cauchy'),lty=1,col=c('red','blue'))
```



From this fit with the Cauchy model, we have

$$\widehat{\beta}_0 = 21.58142 \qquad \widehat{\beta}_1 = 7.43550 \qquad \log\widehat{\sigma} = 0.90017$$

implying that $\widehat{\sigma} = 2.46003$.

For the Bayesian analysis, we begin by specifying the prior. Here there is no possible conjugate analysis, so the simplest choice is to specify independent priors, say

$$\pi_0(\beta_0) \equiv Normal(0, M_{00}) \qquad \pi_0(\beta_1) \equiv Normal(0, M_{01}) \qquad \pi_0(\sigma^2) \equiv InvGamma(a_0, b_0)$$

and then we have for the posterior

$$\pi_n(\beta_0, \beta_1, \sigma^2) \propto \left\{ \prod_{i=1}^{n} \frac{1}{\sigma} \left\{ 1 + \left(\frac{y_i - \beta_0 - \beta_1 x_i}{\sigma} \right)^2 \right\}^{-1} \right\} \pi_0(\beta_0)\pi_0(\beta_1)\pi_0(\sigma^2).$$

In this analysis, we will specify a fairly diffuse prior, with $M_{00} = M_{01} = 400$, and $a_0 = 4, b_0 = 40$.
We have several options for implementing Bayesian inference.

1. **Rejection sampling:** Rejection sampling requires the specification of a proposal distribution. A typical choice is based on a Gaussian approximation to the posterior; we retain the $\beta_0, \beta_1, \psi = \log \sigma$ parameterization to implement this. We have

   $$\pi_0(\sigma^2) \propto \left(\frac{1}{\sigma^2} \right)^{a_0+1} \exp\left\{ -\frac{b_0}{\sigma^2} \right\}$$

   and as $\sigma^2 = e^{2\psi}$, we have by the usual transformation result

   $$\pi_0(\psi) \propto \left(\frac{1}{e^{2\psi}} \right)^{a_0+1} \exp\left\{ -\frac{b_0}{e^{2\psi}} \right\} \times e^{2\psi}.$$

   as the Jacobian is $d\sigma^2/d\psi = 2e^{2\psi}$.

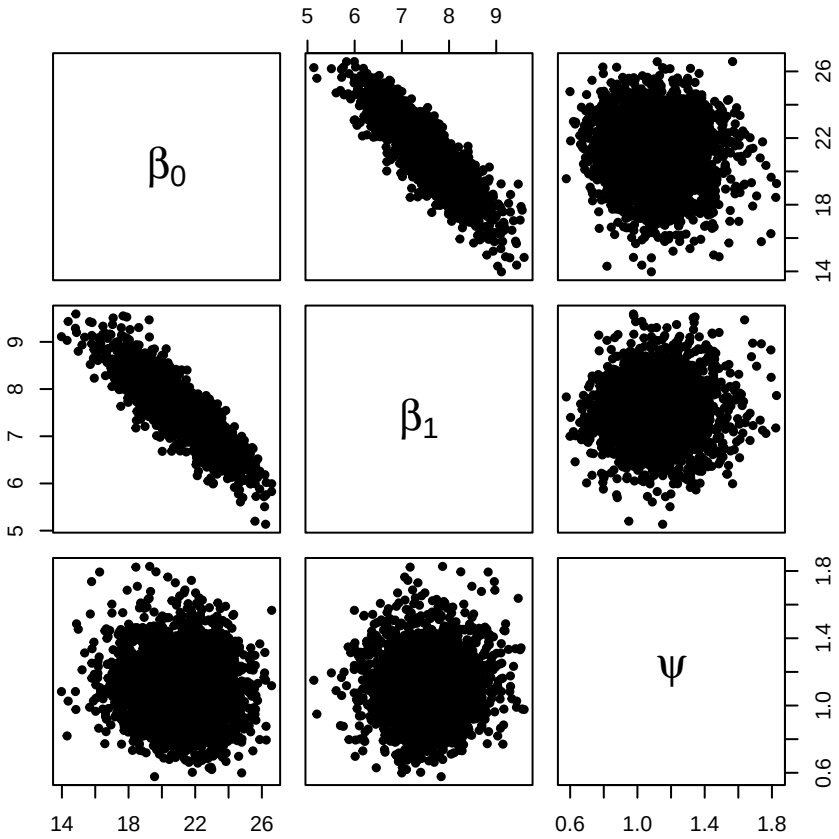
```
log.post<-function(xp,yv,xv,m0v,m1v,a0v,b0v){
        lpv<-yv-xp[1]-xp[2]*xv

        psi<-xp[3]
        sig<-exp(psi)

        llike<-sum(dcauchy(lpv,0,sig,log=T))
        lp1<-dnorm(xp[1],0,sqrt(m0v),log=T)
        lp2<-dnorm(xp[2],0,sqrt(m1v),log=T)
        lp3<--(a0v+1)*(2*psi)-b0v/exp(2*psi)+(2*psi)
        return(llike+lp1+lp2+lp3)
}
#Approximate the posterior
fit2<-optim(c(0,0,1),log.post,yv=y,xv=x,m0v=400,m1v=400,a0v=4,b0v=40,
        control=list(fnscale=-1),hessian=T)
fit2$par  #Posterior mode

+ [1] 21.204758  7.522465  1.050959

fit2$value

+ [1] -100.0674

prop.mean<-fit2$par
prop.Sig<-solve(-fit2$hessian)
```

   We will propose from the three dimensional multivariate Student-t density with 3 degrees of freedom, with mean vector and variance-covariance matrix chosen by approximating the posterior. Recall that we propose from $f_0(x)$, and then accept points with probability

   $$\frac{f(x)}{M f_0(x)}$$

   evaluated at the sampled x values, where M is an upper bound of $f(x)/f_0(x)$.

```
#Rejection sampling
library(mvnfast)
log.rat<-function(xp,yv,xv,m0v,m1v,a0v,b0v,prmu,prSig){  #f(x)/f0(x) on the log scale
    num<-log.post(xp,yv,xv,m0v,m1v,a0v,b0v)
    den<-dmvt(xp,prmu,prSig,df=3,log=T)
    return(num-den)
}
fit3<-optim(c(0,0,1),log.rat,yv=y,xv=x,m0v=400,m1v=400,a0v=4,b0v=40,
        prmu=prop.mean,prSig=prop.Sig,control=list(fnscale=-1),hessian=T)
max.rat<-fit3$value  #Find the bound of log f(x) - log f0(x)

nprop<-10000
th<-rmvt(nprop,prop.mean,prop.Sig,df=3)  #Propose points
rat.vec<-apply(th,1,log.rat,yv=y,xv=x,m0v=400,m1v=400,a0v=4,b0v=40,
        prmu=prop.mean,prSig=prop.Sig)

ival<-log(runif(nprop)) < rat.vec-max.rat   #Accept/Reject
sum(ival)

+ [1] 3060

par(mar=c(4,4,2,2),pty='s')
pairs(th[ival,],pch=19,cex=0.75,
    labels=c(expression(beta[0]),expression(beta[1]),expression(psi)))
```



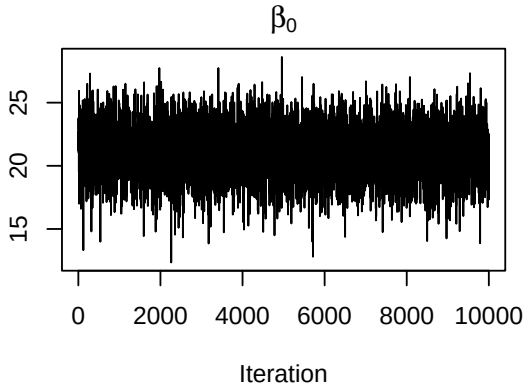
The acceptance rate of rejection sampling is 0.30600, which is reasonable.

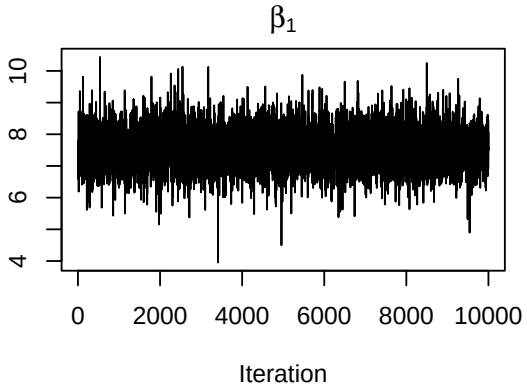
```
#Collect sample of size 10000
N<-50000
th<-rmvt(N,prop.mean,prop.Sig,df=3)  #Propose points
rat.vec<-apply(th,1,log.rat,yv=y,xv=x,m0v=400,m1v=400,a0v=4,b0v=40,
        prmu=prop.mean,prSig=prop.Sig)

ival<-log(runif(N)) < rat.vec-max.rat   #Accept/Reject
sum(ival)

+ [1] 15751

th<-th[ival,]
th<-th[1:nprop,]
dim(th)

+ [1] 10000     3
```

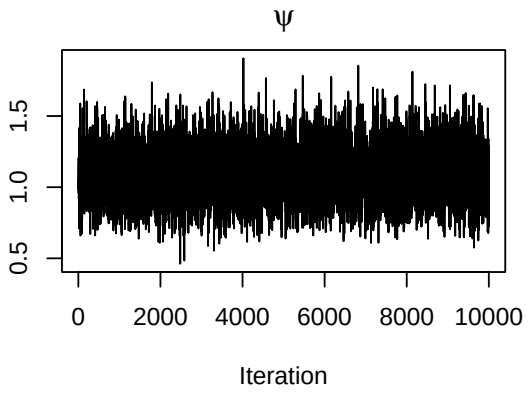
2. **Metropolis-Hastings:** We may implement a joint Metropolis-Hastings algorithm, proposing *independently* from the same Student-t distribution used in rejection sampling.

```
#Metropolis-Hastings
old.th<-prop.mean
old.post<-log.post(old.th,y,x,400,400,4,40)
old.prop<-dmvt(old.th,prop.mean,prop.Sig,df=3,log=T)
th.samp1<-matrix(0,nrow=nprop,ncol=3)
ico<-0
for(iter in 1:nprop){
    new.th<-rmvt(1,prop.mean,prop.Sig,df=3)
    new.post<-log.post(new.th,y,x,400,400,4,40)
    new.prop<-dmvt(new.th,prop.mean,prop.Sig,df=3,log=T)
    if(log(runif(1)) < (new.post-new.prop)-(old.post-old.prop)){ #Accept proposal
            ico<-ico+1
            old.th<-new.th
            old.post<-new.post
            old.prop<-new.prop
    }
    th.samp1[iter,]<-old.th
}
sum(ico)   #Number of accepted points in MH

+ [1] 7164
```

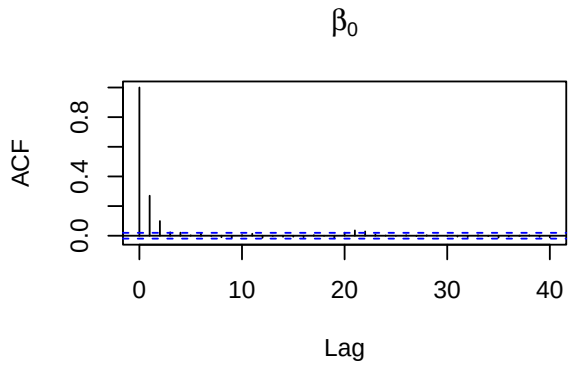
```
par(mar=c(4,4,2,2),mfrow=c(2,2),pty='m')
plot(th.samp1[,1],type='l',xlab='Iteration',ylab='');title(expression(beta[0]))
plot(th.samp1[,2],type='l',xlab='Iteration',ylab='');title(expression(beta[1]))
plot(th.samp1[,3],type='l',xlab='Iteration',ylab='');title(expression(psi))
```

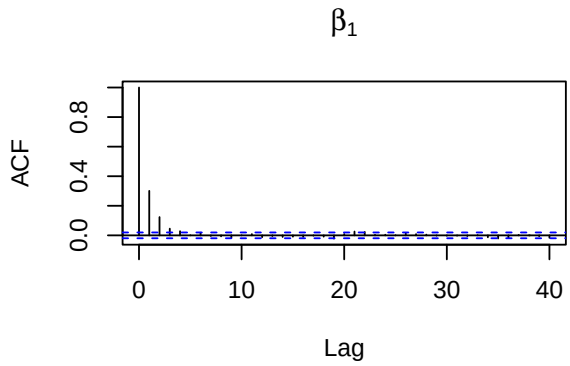


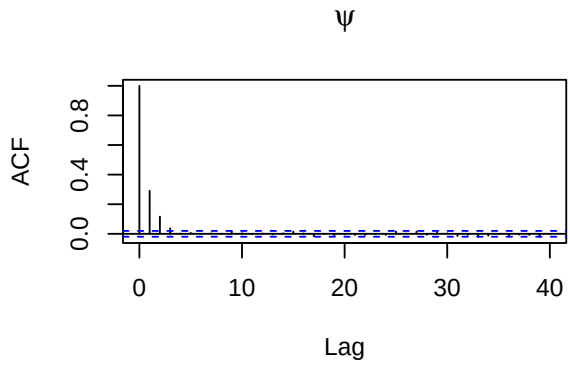





```
par(mar=c(4,4,4,2),mfrow=c(2,2),pty='m')
acf(th.samp1[,1],main=expression(beta[0]))
acf(th.samp1[,2],main=expression(beta[1]))
acf(th.samp1[,3],main=expression(psi))
```







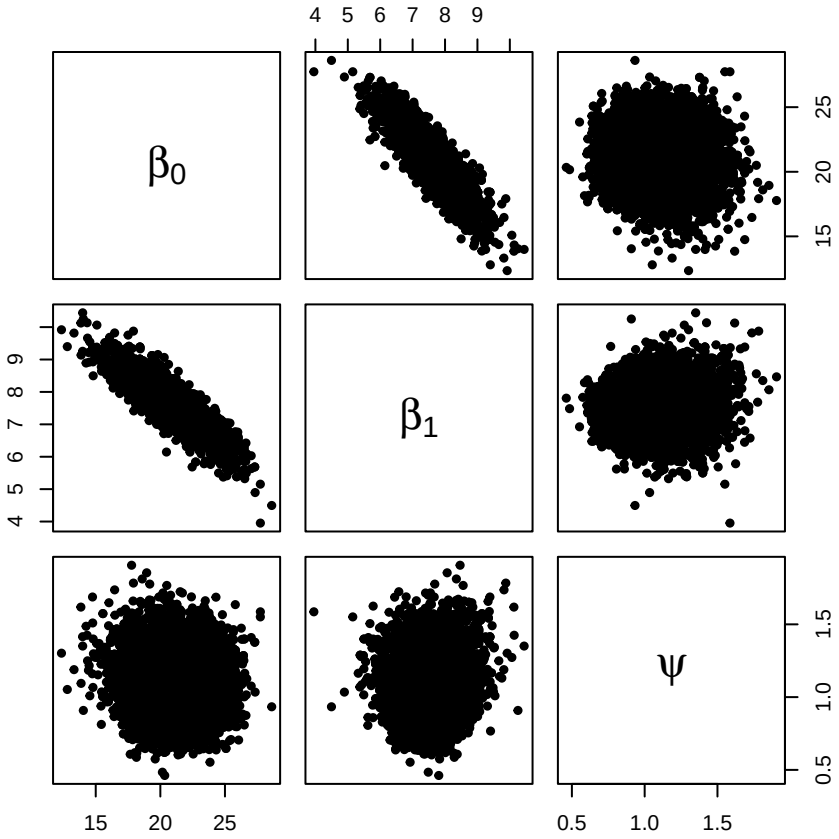
The performance of the Metropolis-Hastings sampler seems to be very good, and we can check this using the effective sample size

```
library(coda)
effectiveSize(th.samp1)

+     var1     var2     var3
+ 5431.373 4962.017 5123.771
```

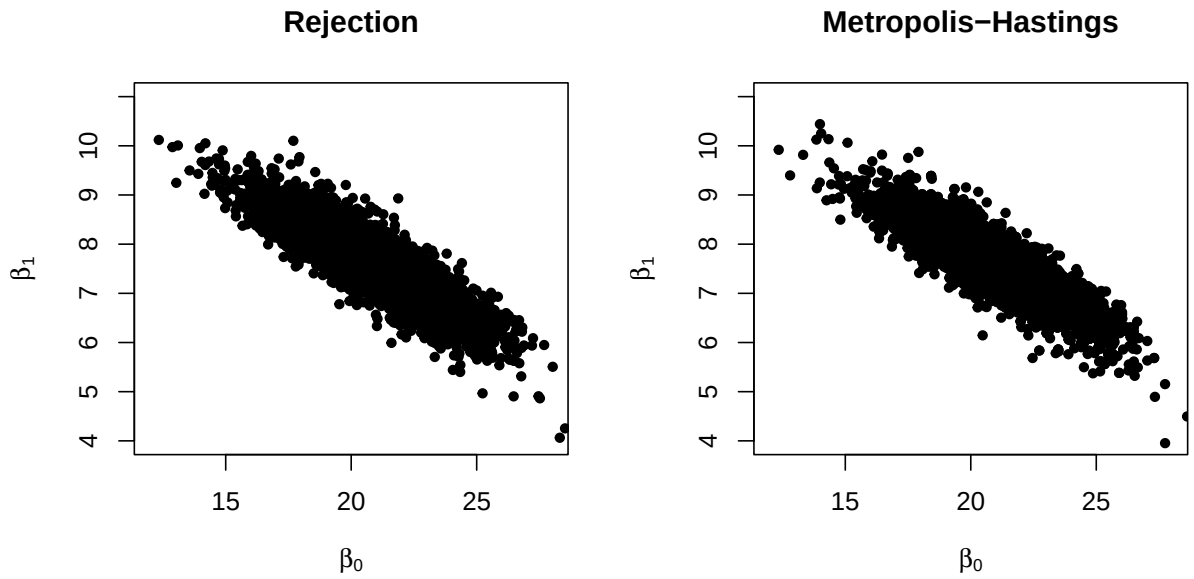
The effective sample sizes (from a run-length of $N = 10000.00000$ are high, in fact higher than the acceptance rate of rejection sampling with the same proposal. If we compare summary statistics from the posterior samples, we see a close correspondence in the means and variances

```
mean.compare<-rbind(apply(th,2,mean),apply(th.samp1,2,mean))
var.compare<-rbind(apply(th,2,var),apply(th.samp1,2,var))
rownames(mean.compare)<-rownames(var.compare)<-c('Rej.','MH')
colnames(mean.compare)<-colnames(var.compare)<-c('be0','be1','psi')
mean.compare

+           be0      be1      psi
+ Rej. 21.07373 7.548435 1.101752
+ MH   21.07886 7.551352 1.100625

var.compare

+           be0       be1        psi
+ Rej. 3.796244 0.3756354 0.03376821
+ MH   3.747083 0.3645685 0.03380689
```

```
par(mar=c(4,4,2,2),pty='s')
pairs(th.samp1,pch=19,cex=0.75,
    labels=c(expression(beta[0]),expression(beta[1]),expression(psi)))
```




```
par(mar=c(4,4,4,2),mfrow=c(1,2))
plot(th[,1:2],pch=19,cex=0.75,xlim=range(12,28),ylim=range(4,11),
    xlab=expression(beta[0]),ylab=expression(beta[1]))
    title('Rejection')
plot(th.samp1[,1:2],pch=19,cex=0.75,xlim=range(12,28),ylim=range(4,11),
    xlab=expression(beta[0]),ylab=expression(beta[1]))
    title('Metropolis-Hastings')
```



3. **Metropolis-within-Gibbs:** It is also possible to implement a Gibbs sampler, updating components of the parameter vector in turn. However, the posterior correlations reveal that the ψ is almost uncorrelated with β_0 and β_1, but that β_0 and β_1 are highly negatively correlated.

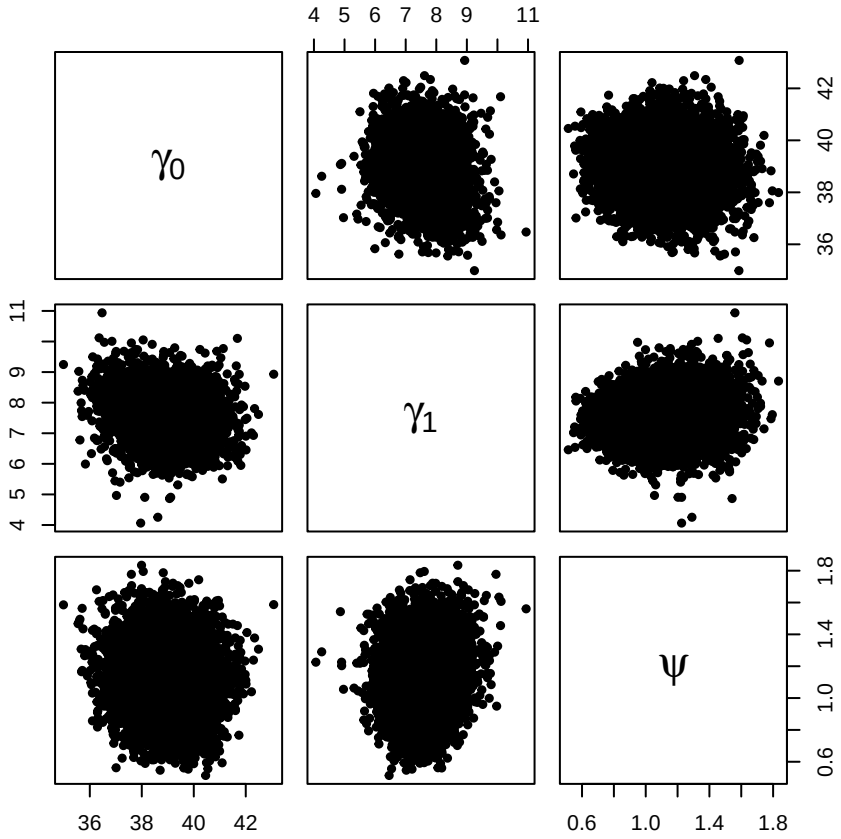
```
cor(th)        #From rejection sampling

+             [,1]        [,2]        [,3]
+ [1,]  1.0000000 -0.88161093 -0.11376016
+ [2,] -0.8816109  1.00000000  0.09745812
+ [3,] -0.1137602  0.09745812  1.00000000

cor(th.samp1)  #From MH

+              [,1]        [,2]        [,3]
+ [1,]  1.00000000 -0.87717489 -0.09227257
+ [2,] -0.87717489  1.00000000  0.08401623
+ [3,] -0.09227257  0.08401623  1.00000000
```

This means that a single parameter update for these parameters is likely to be inefficient. This can be partly solved using centering of the covariate, which effectively induces a re-parameterization. That is, in the updated model

$$Y_i = \gamma_0 + \gamma_1(x_i - \overline{x}) + \epsilon_i \qquad i = 1, \ldots, n$$

where $\beta_0 = \gamma_0 - \gamma_1\overline{x}$, $\beta_1 = \gamma_1$.

```
gam<-th
gam[,1]<-th[,1]+th[,2]*mean(x)
par(mar=c(4,4,2,2),pty='s')
pairs(gam,pch=19,cex=0.75,
    labels=c(expression(gamma[0]),expression(gamma[1]),expression(psi)))
```




```
cor(gam)

+              [,1]        [,2]        [,3]
+ [1,]  1.00000000 -0.27467176 -0.08348625
+ [2,] -0.27467176  1.00000000  0.09745812
+ [3,] -0.08348625  0.09745812  1.00000000
```

Thus we have two options to implement the Metropolis-within-Gibbs:

- update (β_0, β_1) and ψ from their full conditionals;
- update γ_0, γ_1 and ψ from their full conditionals.