Pset 07
18.S096

Thank you L.N.

Spring 2018

1.

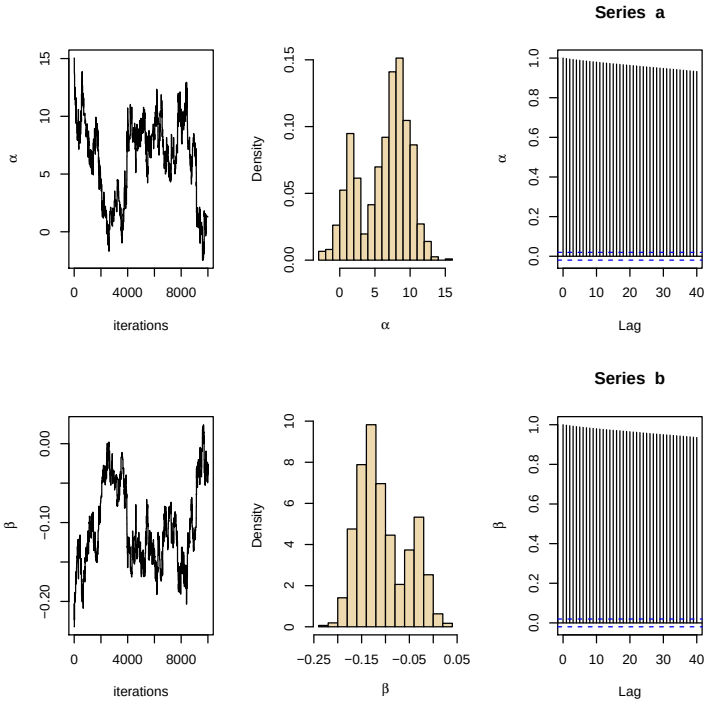
a.

```
> library(mcsm)
> data("challenger")
> fit.logistic<-glm(oring ~ temp, data=challenger,family=binomial(link="logit"))
> fit.logistic

Call:  glm(formula = oring ~ temp, family = binomial(link = "logit"),
    data = challenger)

Coefficients:
(Intercept)         temp
    15.0429      -0.2322

Degrees of Freedom: 22 Total (i.e. Null);  21 Residual
Null Deviance:            28.27
Residual Deviance: 20.32         AIC: 24.32

> summary(fit.logistic)

Call:
glm(formula = oring ~ temp, family = binomial(link = "logit"),
    data = challenger)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-1.0611  -0.7613  -0.3783   0.4524   2.2175

Coefficients:
            Estimate Std. Error z value Pr(>|z|)
(Intercept)  15.0429     7.3786   2.039   0.0415 *
temp         -0.2322     0.1082  -2.145   0.0320 *
---
Signif. codes:  0

> fit.logistic$coefficients

(Intercept)        temp
 15.0429016  -0.2321627

> confint(fit.logistic) #note default is .95% confidence interval

                2.5 %      97.5 %
(Intercept)  3.3305848 34.34215133
temp        -0.5154718 -0.06082076
```

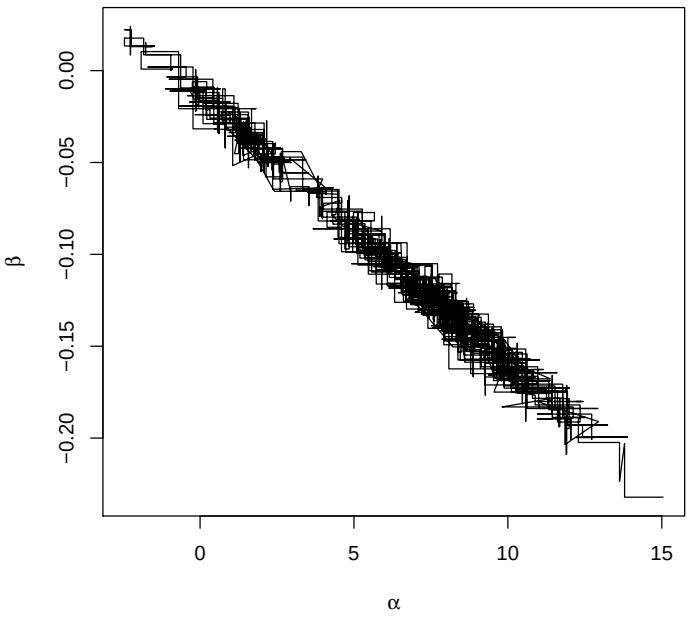
b

```
> set.seed(1)
> #implement Metropolis Hastings using a[t-1] +Laplace(0,ascal) and b[t-1] +Laplace(0,bscal)
> Nsim = 10^4
> x=challenger$temp
> y=challenger$oring
> sigmaa=5 ; sigmab=5/sd(x)
> lpost=function(a,b){
+   sum(y*(a+b*x)-log(1+exp(a+b*x)))+ dnorm(a,sd=sigmaa,log=TRUE)+dnorm(b,sd=sigmab,log=TRUE
+ }
> # Initialize a and b to equal the MLEs
> beta=as.vector(fit.logistic$coefficients)
> #As scale for the proposal densities consider the square root of the
> # cov.unscaled from the ml fit to the logistic model
> MH_logisitc <- function(beta, logistic.data, cand.dist, scale.mult = 1){
+   a=b=rep(0,Nsim)
+   #get first values for a,b
+   a[1]=beta[1]
+   b[1]=beta[2]
+   fit.logistic.summary<-summary(logistic.data)
+   scala=sqrt(fit.logistic.summary$cov.unscaled[1,1])*scale.mult
+   scalb=sqrt(fit.logistic.summary$cov.unscaled[2,2])*scale.mult
+   for (t in 2:Nsim){
+     #get MH prob for a
+     propa=a[t-1]+sample(c(-1,1),1)*rexp(1)*scala
+     if (log(runif(1))<cand.dist(propa,b[t-1])- cand.dist(a[t-1],b[t-1])){
+       a[t]=propa
+     }else{a[t]=a[t-1]}
+     #get MH prob for b
+     propb=b[t-1]+sample(c(-1,1),1)*rexp(1)*scalb
+     if (log(runif(1))<cand.dist(a[t],propb)- cand.dist(a[t],b[t-1])){
+       b[t]=propb
+     }else{ b[t]=b[t-1]}
+   }
+   return(list('a' = a, 'b'= b))
+ }
> result = MH_logisitc(beta, fit.logistic, lpost)
> a = result$a; b =result$b
> #acceptance rate of a and b
> length(unique(a))/Nsim

[1] 0.1001

> length(unique(b))/Nsim

[1] 0.0987
```

```
> par(mfrow=c(2,3))
> plot(a,type="l",xlab="iterations",ylab=expression(alpha))
> hist(a,prob=TRUE,col="wheat2",xlab=expression(alpha),main="")
> acf(a,ylab=expression(alpha))
> plot(b,type="l",xlab="iterations",ylab=expression(beta))
> hist(b,prob=TRUE,col="wheat2",xlab=expression(beta),main="")
> acf(b,ylab=expression(beta))
> c(mean(a), sd(a)); c(mean(b), sd(b))

[1] 6.294067 3.497769

[1] -0.10533114  0.05123078
```

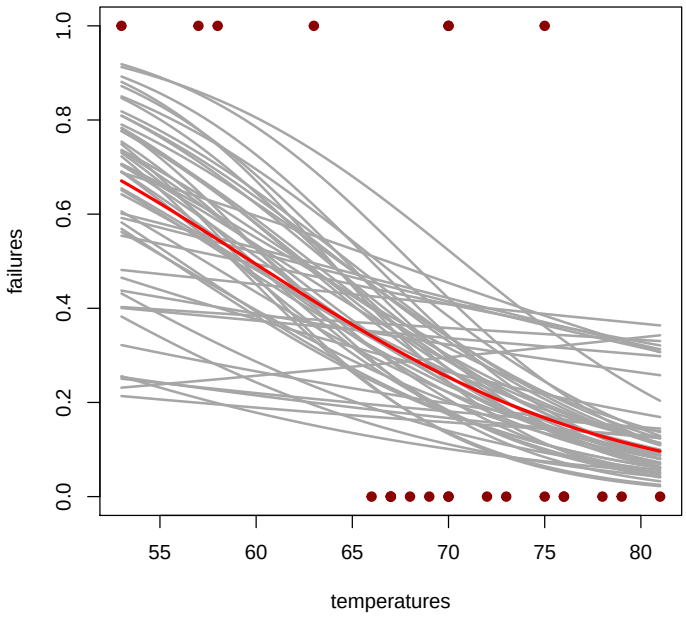


```
> par(mfcol = c(1,1))
> plot(a,b,type = "l", xlab = expression(alpha), ylab =expression(beta))
```



This shows α and β are inversely proportional. This makes sense as we are trying to maximize our loglikelihood and having a large α and small β does that.

```
> plot(challenger$temp,challenger$oring,pch=19,col="red4",
+      xlab="temperatures",ylab="failures")
> for (t in seq(1000,Nsim,le=50))
+   curve(1/(1+exp(-a[t]-b[t]*x)),add=TRUE,col="grey65",lwd=2)
> curve(1/(1+exp(-mean(a)-mean(b)*x)),add=TRUE,col="red",lwd=2.5)
> for (x0 in c(60,50,40,30)){
+ + print(c(x0,
+ + mean(1/(1+exp(-a-b*x0))),
+ + sd(1/(1+exp(-a-b*x0)))))
+   }

[1] 60.0000000  0.4944781  0.1431809
[1] 50.0000000  0.7012076  0.1959169
[1] 40.0000000  0.8134438  0.2077082
[1] 30.0000000  0.8673593  0.1983737
```



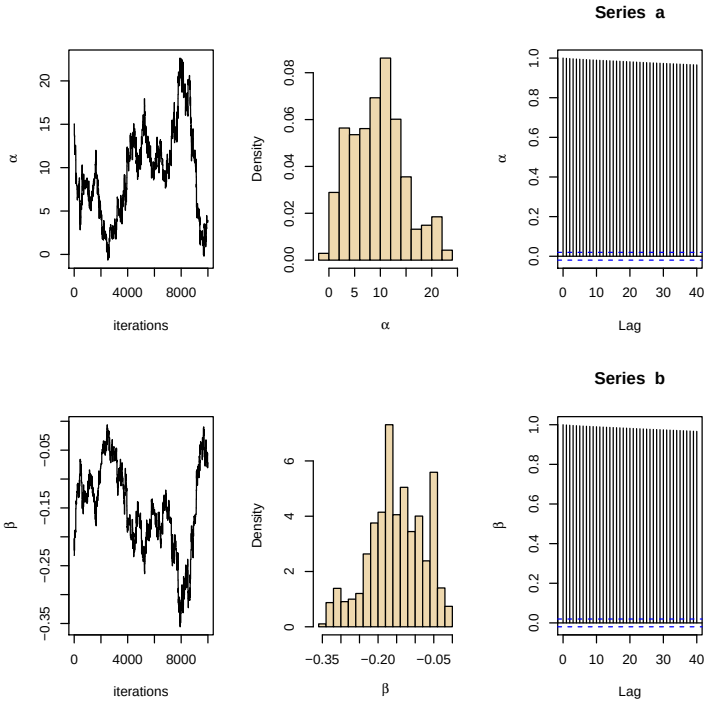
In general it seems as the temperature goes up the faliure rate goes down.

c.

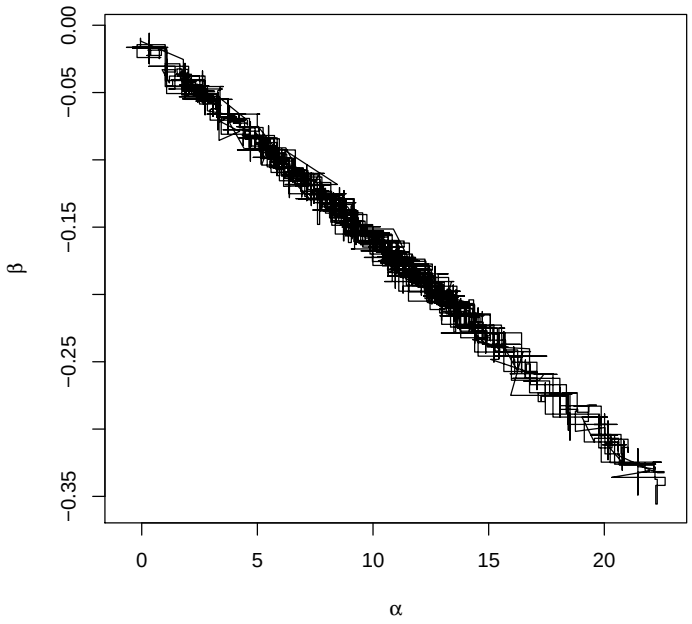
Note I repeat all the comparisons at the end of the problem

```
> set.seed(1)
> #implement Metropolis Hastings using a[t-1] +Laplace(0,ascal) and b[t-1] +Laplace(0,bscal)
> Nsim = 10^4
> x=challenger$temp
> y=challenger$oring
> #multiply be 10 to increase variance by 100
> sigmaa=5*10 ; sigmab=5*10/sd(x)
> lpost=function(a,b){
+   sum(y*(a+b*x)-log(1+exp(a+b*x)))+dnorm(a,sd=sigmaa,log=TRUE)+dnorm(b,sd=sigmab,log=TRUE)
+ }
> beta=as.vector(fit.logistic$coefficients)
> result = MH_logisitc(beta, fit.logistic, lpost)
> a = result$a; b =result$b
> #acceptance rate of a and b
> length(unique(a))/Nsim

[1] 0.1062

> length(unique(b))/Nsim

[1] 0.1039
```

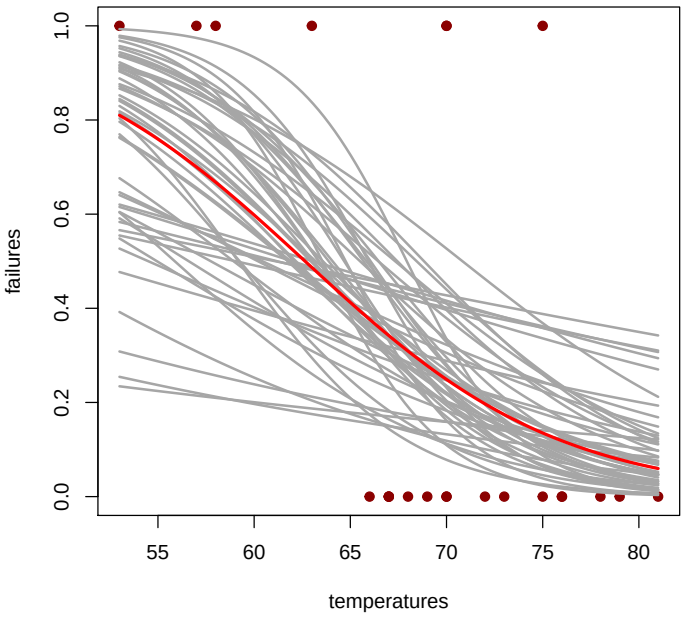
```
> par(mfrow=c(2,3))
> plot(a,type="l",xlab="iterations",ylab=expression(alpha))
> hist(a,prob=TRUE,col="wheat2",xlab=expression(alpha),main="")
> acf(a,ylab=expression(alpha))
> plot(b,type="l",xlab="iterations",ylab=expression(beta))
> hist(b,prob=TRUE,col="wheat2",xlab=expression(beta),main="")
> acf(b,ylab=expression(beta))
> c(mean(a), sd(a)); c(mean(b), sd(b))

[1] 9.414120 5.119201

[1] -0.1502712  0.0746917
```



```
> #path monte carlo distribution
> par(mfcol = c(1,1))
> plot(a,b,type = "l", xlab = expression(alpha), ylab =expression(beta))
```




```
> #probability of faliure
> plot(challenger$temp,challenger$oring,pch=19,col="red4",
+      xlab="temperatures",ylab="failures")
> for (t in seq(1000,Nsim,le=50)) curve(1/(1+exp(-a[t]-b[t]*x)),
+                                       add=TRUE,col="grey65",lwd=2)
> curve(1/(1+exp(-mean(a)-mean(b)*x)),add=TRUE,col="red",lwd=2.5)
```


failures
0.0
0.2
0.4
0.6
0.8
1.0
55
60
65
70
75
80
temperatures

```
> #estimate probablity of faliure
> for (x0 in c(60,50,40,30)){
+ + print(c(x0,
+ + mean(1/(1+exp(-a-b*x0))),
+ + sd(1/(1+exp(-a-b*x0)))))
+ }

[1] 60.0000000  0.5847252  0.1688639
[1] 50.0000000  0.7995651  0.1822004
[1] 40.0000000  0.8898374  0.1559213
[1] 30.0000000  0.9311312  0.1273459

> #from b
> # 60.0000000  0.4944781  0.1431809
> # 50.0000000  0.7012076  0.1959169
> # 40.0000000  0.8134438  0.2077082
> # 30.0000000  0.8673593  0.19837375
>
> #acceptance rate of a and b
> length(unique(a))/Nsim #from b 0.1001

[1] 0.1062

> length(unique(b))/Nsim #from b 0.09873

[1] 0.1039

> #mean and sd of a
> c(mean(a), sd(a))#from b. mean(a) = 6.294067; sd(a) = 3.497769

[1] 9.414120 5.119201

> c(mean(b), sd(b))#from b. mean(b) = -0.10533114; sd(b) =  0.05123078

[1] -0.1502712  0.0746917
```

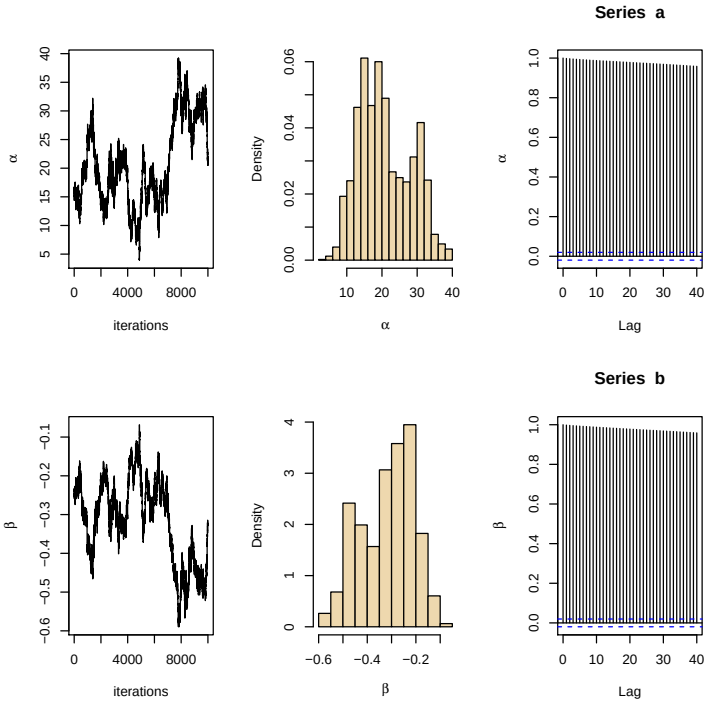
The acceptance rate of α and β both increase slightly. For α both the mean and sd increase by about 50 percent. For β the magnitude of both the mean and sd increase about 50 percent though note the mean is negative. At 30 Degrees Fahrenheit the mean and sd of faliure increase by about 0.7.

d.

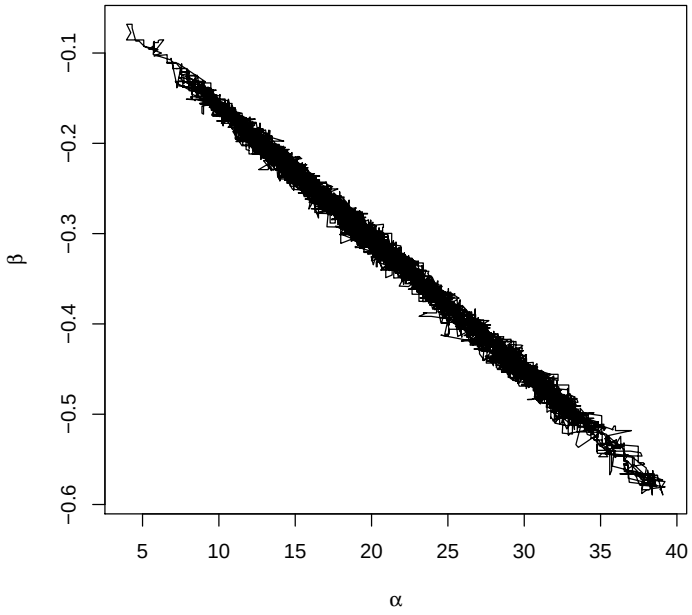
Note like c. I repeat all the comparisons at the end of the problem

```
> set.seed(1)
> #implement Metropolis Hastings using a[t-1] +Laplace(0,ascal) and b[t-1] +Laplace(0,bscal)
> Nsim = 10^4
> x=challenger$temp
> y=challenger$oring
> #multiply be 10 to increase variance by 100 to stay similar to c.
> sigmaa=5*10 ; sigmab=5*10/sd(x)
> lpost=function(a,b){
+   sum(y*(a+b*x)-log(1+exp(a+b*x)))+dnorm(a,sd=sigmaa,log=TRUE)+dnorm(b,sd=sigmab,log=TRUE)
+ }
> beta=as.vector(fit.logistic$coefficients)
> #set scale.mult = .1 to reduce the scale by 1/10th
> result = MH_logisitc(beta, fit.logistic, lpost, scale.mult = .1)
> a = result$a; b =result$b
> #acceptance rate of a and b
> length(unique(a))/Nsim

[1] 0.6104

> length(unique(b))/Nsim

[1] 0.6141
```

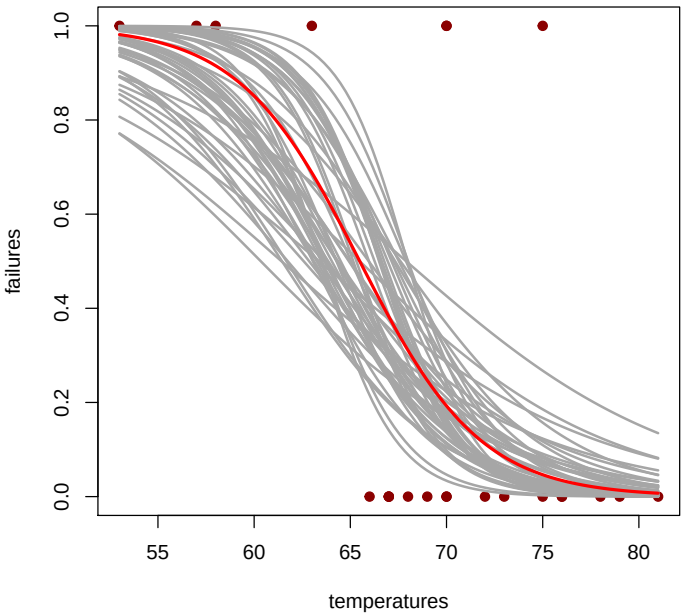
```
> par(mfrow=c(2,3))
> plot(a,type="l",xlab="iterations",ylab=expression(alpha))
> hist(a,prob=TRUE,col="wheat2",xlab=expression(alpha),main="")
> acf(a,ylab=expression(alpha))
> plot(b,type="l",xlab="iterations",ylab=expression(beta))
> hist(b,prob=TRUE,col="wheat2",xlab=expression(beta),main="")
> acf(b,ylab=expression(beta))
> c(mean(a), sd(a)); c(mean(b), sd(b))

[1] 20.807518  7.358899

[1] -0.3177214  0.1082698
```



```
> #path monte carlo distribution
> par(mfcol = c(1,1))
> plot(a,b,type = "l", xlab = expression(alpha), ylab =expression(beta))
```




```
> #probability of faliure
> plot(challenger$temp,challenger$oring,pch=19,col="red4",
+      xlab="temperatures",ylab="failures")
> for (t in seq(1000,Nsim,le=50)) curve(1/(1+exp(-a[t]-b[t]*x)),
+                                       add=TRUE,col="grey65",lwd=2)
> curve(1/(1+exp(-mean(a)-mean(b)*x)),add=TRUE,col="red",lwd=2.5)
```


failures
0.0
0.2
0.4
0.6
0.8
1.0
55
60
65
70
75
80
temperatures

```
> #estimate probablity of faliure
> for (x0 in c(60,50,40,30)){
+ + print(c(x0,
+ + mean(1/(1+exp(-a-b*x0))),
+ + sd(1/(1+exp(-a-b*x0)))))
+ }

[1] 60.0000000  0.8100354  0.1353225
[1] 50.00000000  0.97234977  0.04588713
[1] 40.00000000  0.99508344  0.01506382
[1] 30.000000000  0.998901915  0.005897289

> #from b
> # 30.0000000  0.8673593  0.19837375
> #from c
> # 30.0000000  0.9311312  0.1273459
>
> #acceptance rate of a and b
> length(unique(a))/Nsim #from b 0.1001; from c 0.1062

[1] 0.6104

> length(unique(b))/Nsim #from b 0.09873; from c 0.1039

[1] 0.6141

> #mean and sd of a
> c(mean(a), sd(a))#from b. mean(a) = 6.294067; sd(a) = 3.497769

[1] 20.807518  7.358899

> #from c. mean(a) = 9.414120; sd(a) =  5.119201
> c(mean(b), sd(b))#from b. mean(b) = -0.10533114; sd(b) =  0.05123078

[1] -0.3177214  0.1082698

> #from c. mean(b) = -0.1502712; sd(b) =  0.0746917
```

The acceptance rate of α and β both increase by a large amount compared to both ba nd c (around 6 times), which makes sense as we are decreasing the variance of the candiate distribution so we get less exreme values. For α both the mean and sd increase by a large amount compared to both b and c, with the mean increasing by a larger percent. Similar to α, the magnitude of both the mean and sd of β increase by a large amount with the magnitude of the mean increasing by more than sd. At 30 Degrees Fahrenheit the mean of faliure rate increased to nearly 1 and the sd is about 0, which means that faliure is basically guarnteed where in b,c it was only around 80-90 percent.

2.

a.

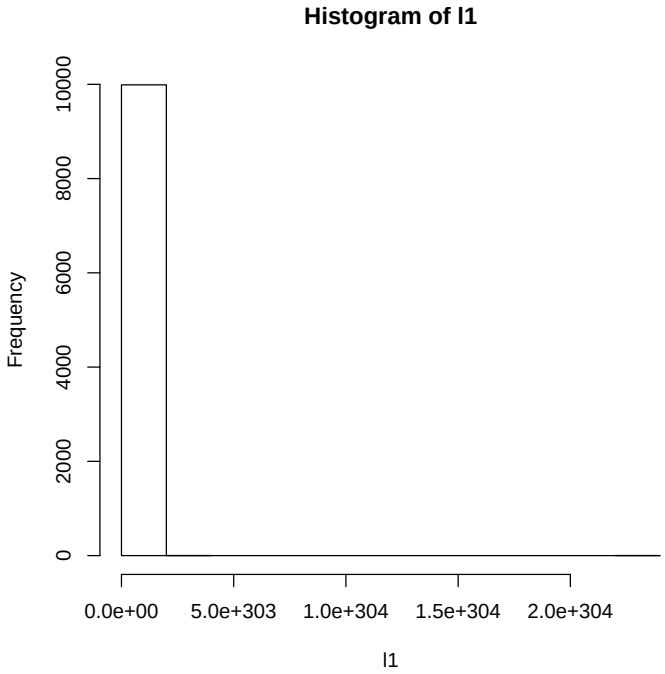
$\Pi_{i=1}^{10}((\lambda_i t_i)^{x_i} e^{-\lambda_i t_i} \lambda_i^{\alpha-1} e^{-\beta\lambda_i}) \beta^{10\alpha} \beta^{\gamma-1} e^{-\delta\beta}$
$\propto \Pi_{i=1}^{10}(\lambda_i^{x_i+\alpha-1} e^{-\lambda_i(t_i+\beta)}) \beta^{10\alpha+\gamma-1} e^{-\delta\beta} *$
$\propto \Pi_{i=1}^{10}(\lambda_i^{x_i+\alpha-1} e^{-\lambda_i t_i}) \beta^{10\alpha+\gamma-1} e^{-\beta(\delta+\Sigma_{i=1}^{10}\lambda_i)} **$

b.

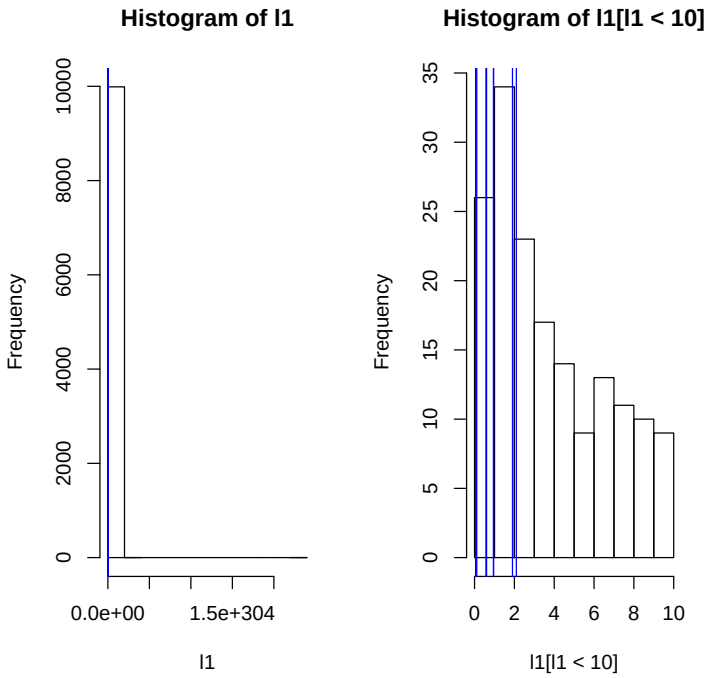
We know that $\propto x^{\alpha-1} e^{-\beta x} \sim G(\alpha, \beta)$ so:
(from * in a)
$\pi(\lambda_i|\beta, t_i, x_i) i = 1, ..., 10 \propto (\lambda_i^{x_i+\alpha-1} e^{-\lambda_i(t_i+\beta)}) \beta^{10\alpha+\gamma-1} e^{-\delta\beta}$
$\propto \lambda_i^{x_i+\alpha-1} e^{-\lambda_i(t_i+\beta)} \sim G(x_i+\alpha, t_i+\beta)$
(from ** in a)
$\pi(\beta|\lambda_1, ..., \lambda_{10}) \propto \Pi_{i=1}^{10}(\lambda_i^{x_i+\alpha-1} e^{-\lambda_i t_i}) \beta^{10\alpha+\gamma-1} e^{-\beta(\delta+\Sigma_{i=1}^{10}\lambda_i)}$
$\propto \beta^{10\alpha+\gamma-1} e^{-\beta(\delta+\Sigma_{i=1}^{10}\lambda_i)} \sim G(10\alpha+\gamma, \delta+\Sigma_{i=1}^{10}\lambda_i)$

c.

```
> a = 1.8; y = .01 ; d = 1 ; n = 10000; b = rgamma(n,y,d)
> l1 <- rgamma(n, a, b)
> hist(l1)
> mean(l1);sd(l1);median(l1)

[1] Inf

[1] NaN

[1] 2.215405e+31

> pump = c(5,1,5,14,3,19,1,1,4,22); time = c(94.32, 15.72, 62.88, 125.76, 5.24, 31.44, 1.05,
> l.mle = pump/time; l.mle

 [1] 0.05301103 0.06361323 0.07951654 0.11132316 0.57251908 0.60432570
 [7] 0.95238095 0.95238095 1.90476190 2.09923664
```




```
> l.mle.talbe <- as.table(l.mle)
> names(l.mle.talbe) <- c("l1", "l2", "l3", "l4", "l5", "l6", "l7","l8", "l9", "l10")
> par(mfcol = c(1,2))
> hist(l1)
> abline(v = l.mle, col = c("blue"))
> hist(l1[l1 <10])
> abline(v = l.mle, col = c("blue"))
> percentile.l1 <- function(x){
+   return(sum(l1<x)/length(l1))
+ }
> apply(l.mle.talbe,1,percentile.l1)

    l1     l2     l3     l4     l5     l6     l7     l8     l9    l10
0.0000 0.0001 0.0001 0.0001 0.0013 0.0015 0.0026 0.0026 0.0057 0.0062
```



The prior distribution is really bad. First because some β are zero or very close to zero, the proir allows α to become unreasonablly large (even infinity in some cases). Second if we try to fit an MLE estimate based on the values we've seen from the actual pumps, every λ_i is below the the first percentile, which is very unlikely.

d.

```
> n = 10000; num_l = 10;l = matrix(rep(0,n*num_l), nrow = n, ncol = num_l); b = rep(0,n)
> l[1,] = l.mle
> for (i in 2:n){
+   for (j in 1: num_l){
+     l[i,j] = rgamma(1, shape = pump[j]+a,rate = time[j]+b[i-1])
+     b[i] = rgamma(1, shape = y+ 10*a, rate = d + sum(l[i,1:10]))
+   }
+ }
> l[n,1:10]

 [1] 0.12518810 0.17929022 0.06286322 0.14322893 1.31062742 0.61602438
 [7] 0.89926389 2.46938117 1.58476829 2.16481007

> b[n]

[1] 1.882487
```

e.

```
> #get credible by equal-tailed interval -- equal probability
> quantile(l[,1], c(.025,.975))

     2.5%     97.5%
0.0269573 0.1314081

> quantile(l[,2], c(.025,.975))

      2.5%      97.5%
0.02824752 0.38399400

> quantile(l[,3], c(.025,.975))

      2.5%      97.5%
0.04216767 0.19596485

> quantile(l[,4], c(.025,.975))

     2.5%     97.5%
0.0702907 0.1888561

> quantile(l[,5], c(.025,.975))

     2.5%     97.5%
0.1977972 1.3271647

> quantile(l[,6], c(.025,.975))
```

```
     2.5%     97.5%
0.3778729 0.9088502

> quantile(l[,7], c(.025,.975))

     2.5%     97.5%
0.1527129 2.1915975

> quantile(l[,8], c(.025,.975))

     2.5%     97.5%
0.1482992 2.1506231

> quantile(l[,9], c(.025,.975))

     2.5%     97.5%
0.4240203 2.6387069

> quantile(l[,10], c(.025,.975))

    2.5%    97.5%
1.173706 2.708014

> quantile(b, c(.025,.975))

    2.5%    97.5%
1.325300 4.080276

> #get creidible inteval with highest posterior density interval -- narrowest region
> cred_interval <- function(param, size = .95){
+   sorted_param <- sort(param)
+   interval_size <- as.integer(length(param)* size)
+   #get size of widths of intervals
+   width_sizes <- sorted_param[(1+interval_size): length(param)] -
+     sorted_param[1:(length(param)-interval_size)]
+   smalled_interval_index = which.min(width_sizes)
+   return(c(sorted_param[smalled_interval_index],
+            sorted_param[smalled_interval_index+interval_size]))
+ }
> cred_interval(l[, 1])

[1] 0.02161627 0.12232638

> cred_interval(l[, 2])

[1] 0.01299276 0.33633875

> cred_interval(l[, 3])
```

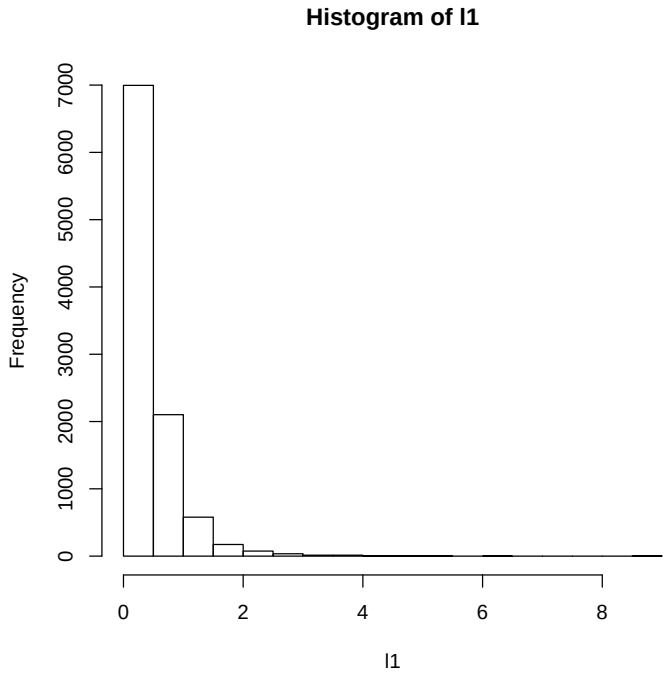
```
[1] 0.03308214 0.18116603

> cred_interval(l[, 4])

[1] 0.06633748 0.18221267

> cred_interval(l[, 5])

[1] 0.126335 1.197708

> cred_interval(l[, 6])

[1] 0.3661171 0.8902785

> cred_interval(l[, 7])

[1] 0.05451808 1.85873968

> cred_interval(l[, 8])

[1] 0.06880289 1.88389957

> cred_interval(l[, 9])

[1] 0.3151314 2.4116750

> cred_interval(l[, 10])

[1] 1.127439 2.633033

> cred_interval(b)

[1] 1.165207 3.864610
```

For this problem let the upper bound on the 95 percent below 1 be good, below 2 be ok, below 3 be bad and anything else be really bad.
Based on a equal-tailed cred interval pumps 1-4,6 are really good, 5,7,8 are ok (below 2), 9 is bad, and 10 is really bad (being around 4).
Based on a creidible inteval with highest posterior density interval 1-4,6 are good, 5,7,8 are ok and 9,10 are bad.

f.

Change mentioned in code below

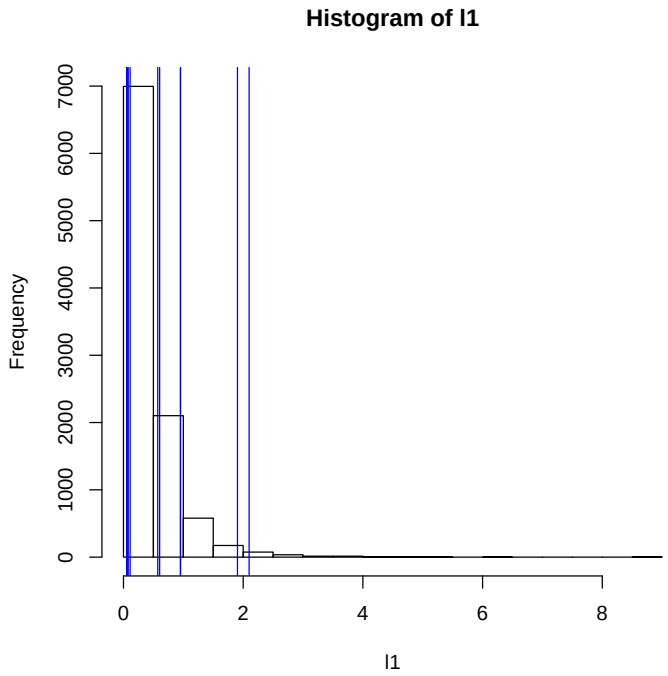
```
> #change y to 5 as in previous problem beta ended up being around 5 with a credible interva
> #from 1-10 so setting y = 5 and keeping d = 1 gives up and expected value of 5 and var of
> a = 1.8; y = 5 ; d = 1 ; n = 10000; b = rgamma(n,y,d)
> l1 <- rgamma(n, a, b)
> hist(l1)
> mean(l1);sd(l1);median(l1)

[1] 0.4511198

[1] 0.4644829

[1] 0.3216859

> pump = c(5,1,5,14,3,19,1,1,4,22); time = c(94.32, 15.72, 62.88, 125.76, 5.24, 31.44, 1.05,
> l.mle = pump/time; l.mle

 [1] 0.05301103 0.06361323 0.07951654 0.11132316 0.57251908 0.60432570
 [7] 0.95238095 0.95238095 1.90476190 2.09923664
```




```
> l.mle.talbe <- as.table(l.mle)
> names(l.mle.talbe) <- c("l1", "l2", "l3", "l4", "l5", "l6", "l7","l8", "l9", "l10")
```

```
> hist(l1)
> abline(v = l.mle, col = c("blue"))
> percentile.l1 <- function(x){
+   return(sum(l1<x)/length(l1))
+ }
> apply(l.mle.talbe,1,percentile.l1)
```

```
    l1     l2     l3     l4     l5     l6     l7     l8     l9    l10
0.0496 0.0668 0.0932 0.1483 0.7509 0.7705 0.9010 0.9010 0.9835 0.9871
```



This new prior seems reasonable. It doesn't have the same issue with β being too small. The MLE of λ values are pretty reasonable based on the sample distribution, with some lying toward the lower percentile, some higher and some towards the middle. It is a little worrying that 6 of the MLEs lie either in the 10th percentile or above the 90th percentile but this seems consitent with the credible intervals above and below where the same pumps were the more extreme credible intervals. The exception to this being pumps 7,8 who are now in the 90th percentile but whose credible intervals weren't that extreme making me think that maybe the prior has a β with a slightly lower expected value (which is also consistent with the β calculated below).

```
> n = 10000; num_l = 10;l = matrix(rep(0,n*num_l), nrow = n, ncol = num_l); b = rep(0,n)
> l[1,] = l.mle
> for (i in 2:n){
+   for (j in 1: num_l){
+     l[i,j] = rgamma(1, shape = pump[j]+a,rate = time[j]+b[i-1])
+     b[i] = rgamma(1, shape = y+ 10*a, rate = d + sum(l[i,1:10]))
+   }
+ }
> l[n,1:10]

 [1] 0.0430443 0.2077238 0.0471020 0.1146184 0.7769673 0.4676458 1.4618501
 [8] 1.4256963 0.9260060 1.7778235

> b[n]

[1] 2.222627

> #get credible by equal-tailed interval -- equal probability
> quantile(l[,1], c(.025,.975))

      2.5%      97.5%
0.02790454 0.12823285

> quantile(l[,2], c(.025,.975))

      2.5%      97.5%
0.02819363 0.35540636

> quantile(l[,3], c(.025,.975))

     2.5%     97.5%
0.0408982 0.1930084

> quantile(l[,4], c(.025,.975))

      2.5%      97.5%
0.07013801 0.18704932

> quantile(l[,5], c(.025,.975))

     2.5%     97.5%
0.1705647 1.1632500

> quantile(l[,6], c(.025,.975))

     2.5%     97.5%
0.3648971 0.8699032

> quantile(l[,7], c(.025,.975))
```

```
     2.5%     97.5%
0.1128432 1.6769010

> quantile(l[,8], c(.025,.975))

     2.5%     97.5%
0.1144116 1.6349888

> quantile(l[,9], c(.025,.975))

     2.5%     97.5%
0.3446151 2.1630220

> quantile(l[,10], c(.025,.975))

    2.5%    97.5%
1.058485 2.482353

> quantile(b, c(.025,.975))

    2.5%    97.5%
2.047414 5.657192

> #get creidible inteval with highest posterior density interval -- narrowest region
> cred_interval <- function(param, size = .95){
+   sorted_param <- sort(param)
+   interval_size <- as.integer(length(param)* size)
+   #get size of widths of intervals
+   width_sizes <- sorted_param[(1+interval_size): length(param)] -
+     sorted_param[1:(length(param)-interval_size)]
+   smalled_interval_index = which.min(width_sizes)
+   return(c(sorted_param[smalled_interval_index],
+            sorted_param[smalled_interval_index+interval_size]))
+ }
> cred_interval(l[, 1])

[1] 0.02196033 0.11977306

> cred_interval(l[, 2])

[1] 0.01401167 0.31817090

> cred_interval(l[, 3])

[1] 0.03285703 0.17782387

> cred_interval(l[, 4])

[1] 0.0667559 0.1822241
```

```
> cred_interval(l[, 5])

[1] 0.1182216 1.0533949

> cred_interval(l[, 6])

[1] 0.3526913 0.8516245

> cred_interval(l[, 7])

[1] 0.05240427 1.45295727

> cred_interval(l[, 8])

[1] 0.04829719 1.42228919

> cred_interval(l[, 9])

[1] 0.2865932 2.0225275

> cred_interval(l[, 10])

[1] 1.006304 2.407497

> cred_interval(b)

[1] 1.941606 5.471098
```

Like above, for this problem let the upper bound on the 95 percent below 1 be good, below 2 be ok, below 3 be bad and anything else be really bad.
Based on a equal-tailed cred interval pumps 1-4,6 are really good, 5,7,8 are ok (below 2), 9,10 are.
Based on a creidible inteval with highest posterior density interval 1-4,6 are good, 5,7,8,9 are ok (with 5 being a little above 1) and 10 is bad.