takehome midterm

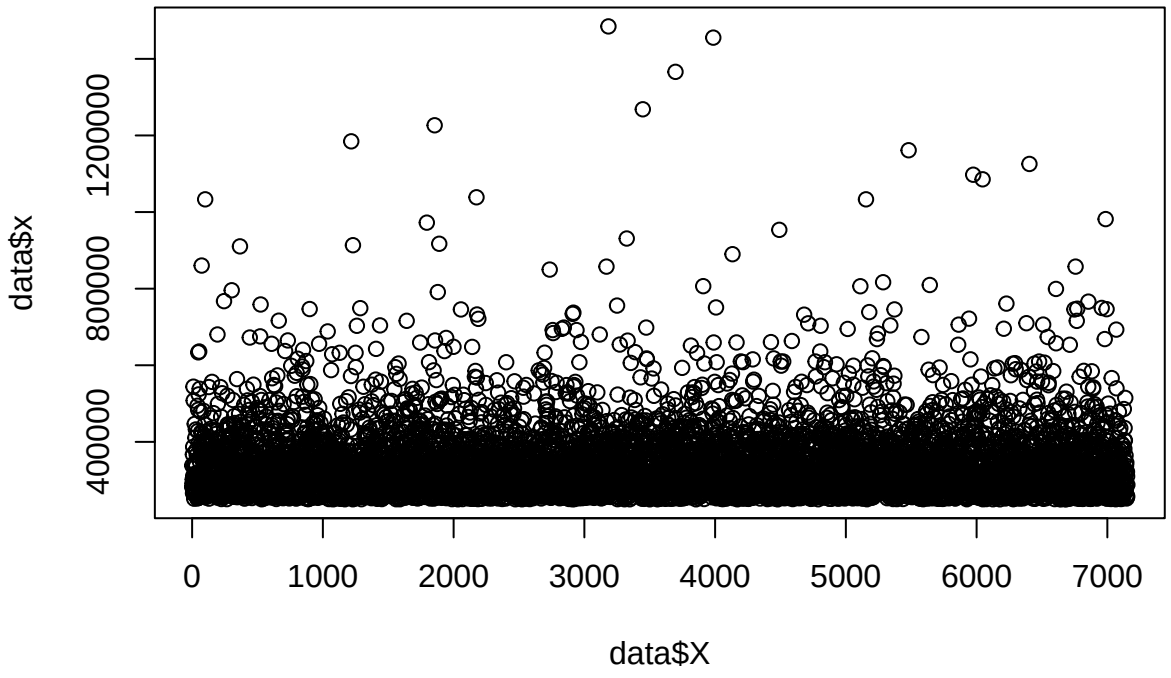
Michael Peña

2024-11-02

Question 1

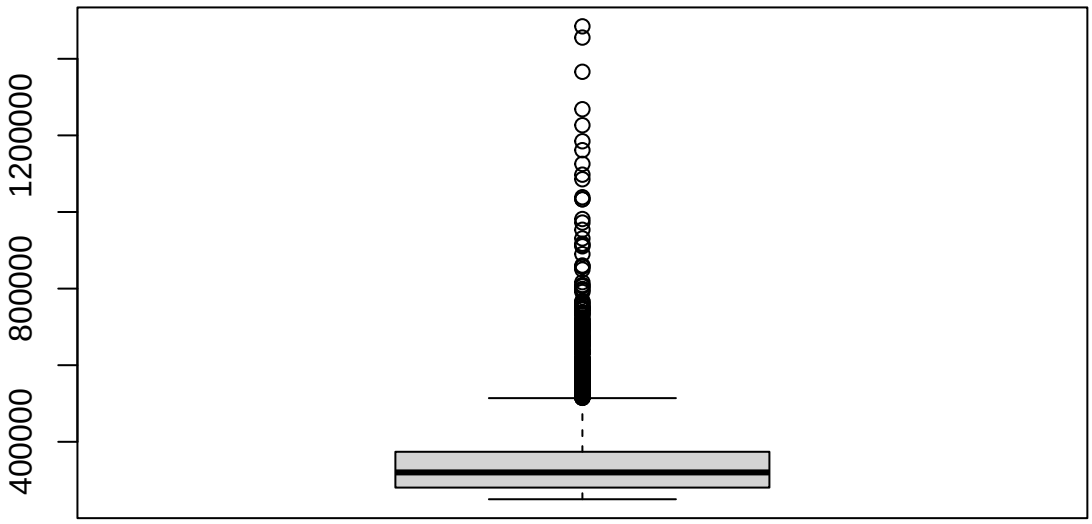
```
data <- read.csv("income20train-2.csv", header = T)
data[,2] -> y
```

```
plot(data$X,data$x)
```

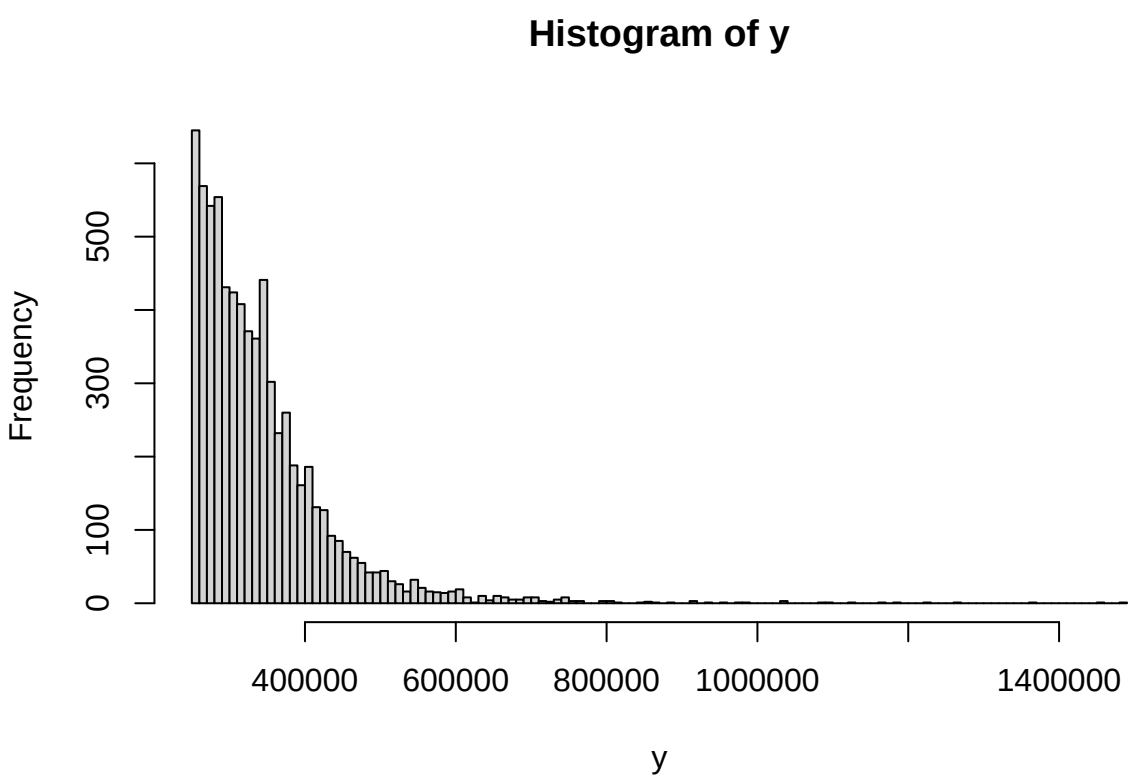


(a)

```
boxplot(y)
```




```
hist(y, breaks = 100)
```




```
summary(y)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  250034  280433  320084  341938  373952 1484705
```

```
mean(y)
```

```
## [1] 341938.1
```

Important to note that the incomes tend to be dense towards the minimum rather than being in the middle; these numbers skew heavily to the right. We can try to fit a Pareto distribution to this later to see how is models the data and run our test set against it. We have a median of 320084 and a mean of 341938.1 while he have a max of 1484705. Also note the amount of outliers in the boxplot and where these outliers are. Also note how these outliers range wider than that of the not unusual data. This shows support to the phenomenon of American income inequality (even among the 20% richest in the nation).

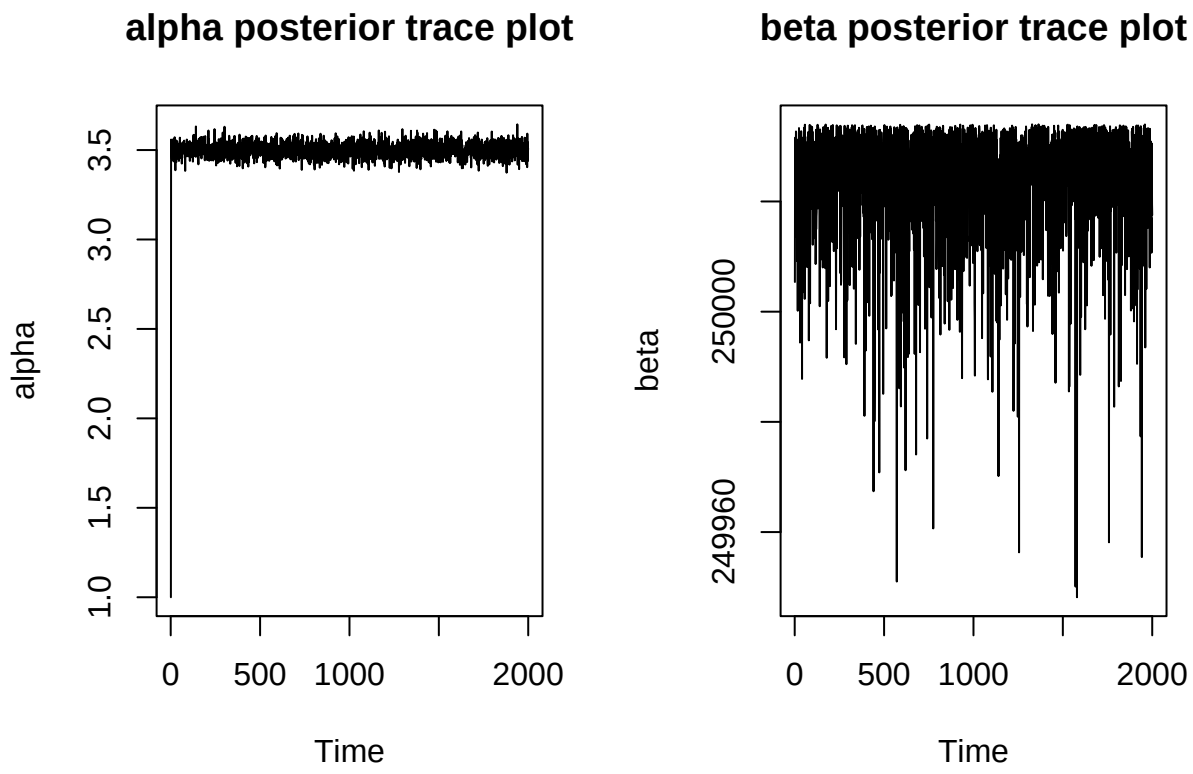
(b)

- consider an initial α_{b-1}
- sample β_{b-1} from $Mono(n\alpha_{b-1}+1, min(y_1, ..., y_n))$
- enter loop for B amount of times
  - sample α_b from $\Gamma(n+1, \sum_{i=1}^{n}[ln(y_i)] - nln(\beta_{b-1}))$
  - sample β_b from $Mono(n\alpha_b+1, min(y_1, ..., y_n))$
  - β_b is set to β_{b-1}
- these α's and β's are stored in a vector.

```
# prep the mono sample function
```

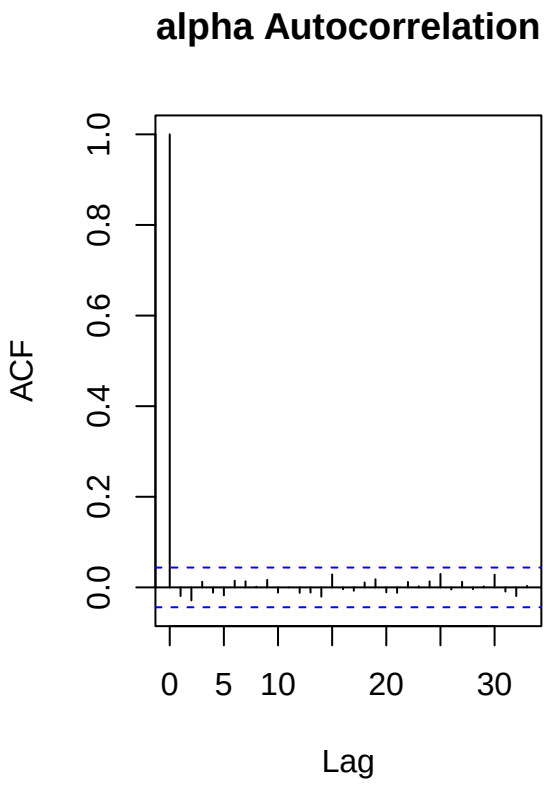
```r
rmono = function(n,alpha,beta){
  u = runif(n)
  x = exp(log(beta) + (log(u)/alpha))
  return(x)
}
# make this the last this
GIBBS1 <- function(B,data,alpha.init){
  set.seed(90909)
  # define n
  n = length(data)
  # make vector spaces
  alpha <- beta <- rep(0,B)
  # apply initials
  alpha[1] = alpha.init
  beta[1] = rmono(1,n*alpha.init + 1, min(data))

  #loop
  for(b in 2:B){
    rgamma(1,n+1,sum(log(data)) - n*log(beta[b-1])) -> alpha[b]
    rmono(1,n*alpha[b] + 1, min(data)) -> beta[b]
  }
  # output
  theta = cbind(alpha,beta)
  return(theta)
}
GIBBS1(2000,y,1) -> theta
```

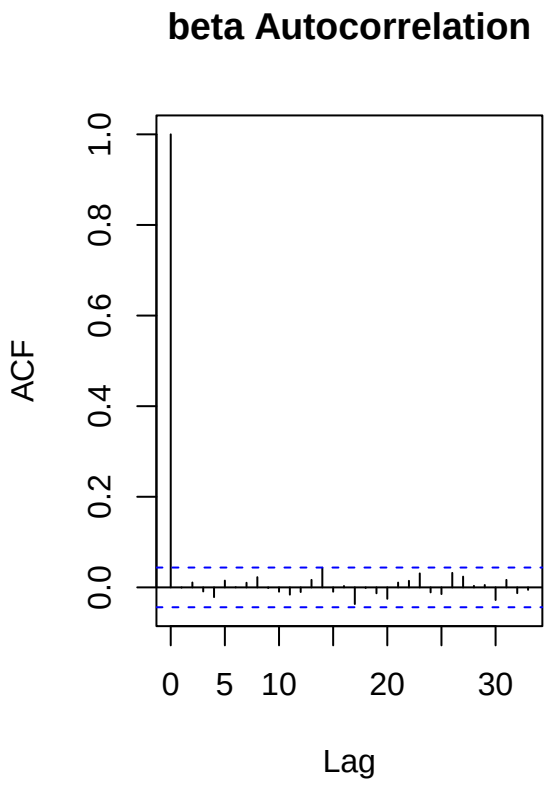
```r
par(mfrow = c(1,2))
plot.ts(theta[,1], main = "alpha posterior trace plot", ylab = "alpha")
plot.ts(theta[,2], main = "beta posterior trace plot", ylab = "beta")
```



(c)

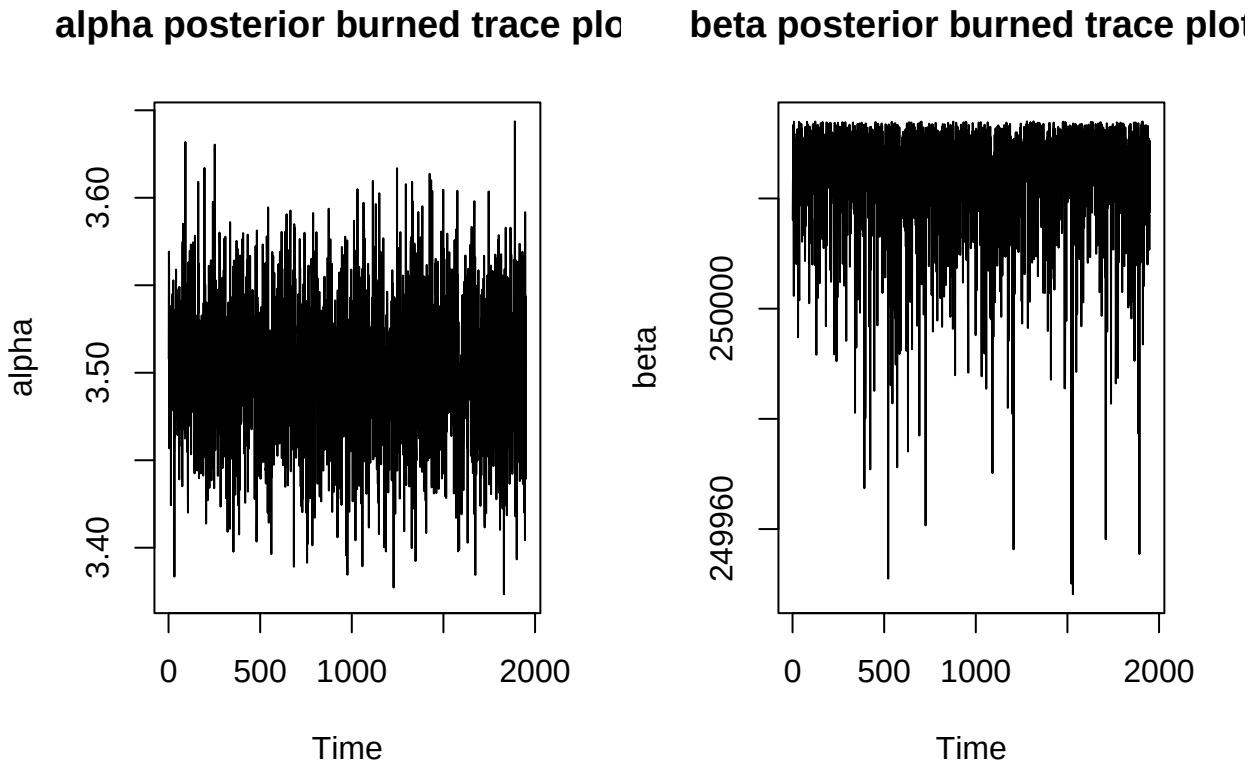
```
acf(theta[,1], main = "alpha Autocorrelation")
acf(theta[,2], main = "beta Autocorrelation")
```



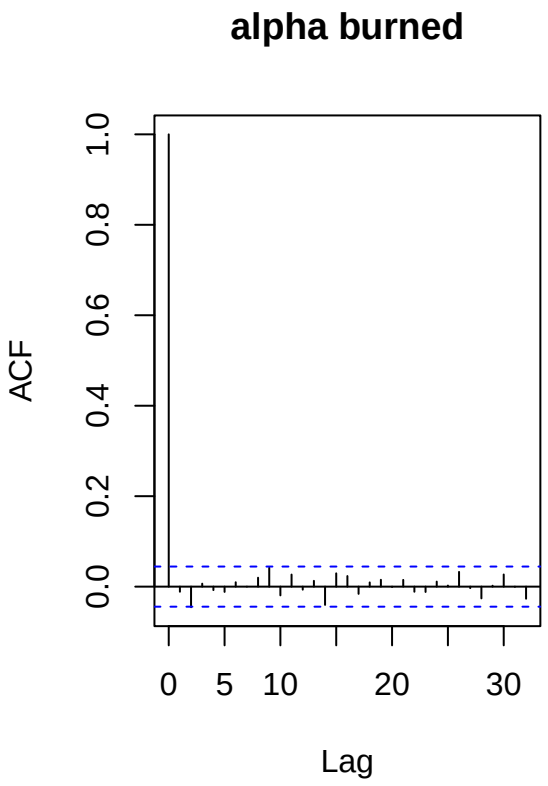


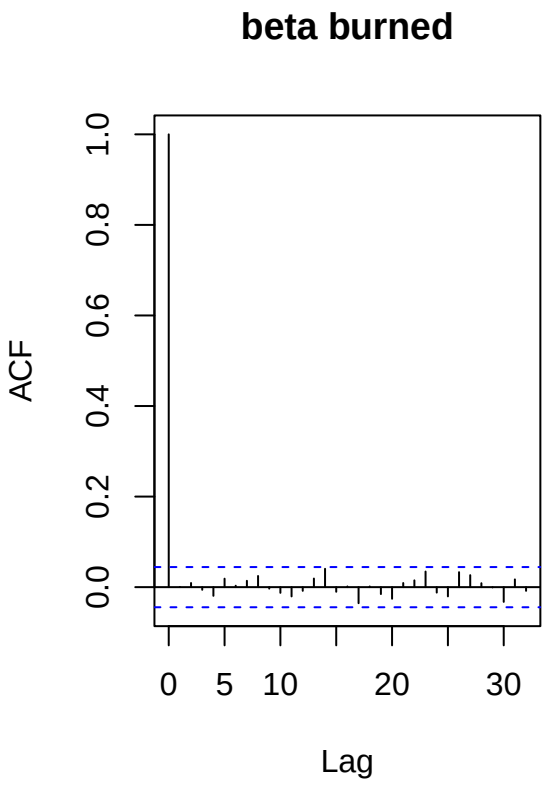
There seems to be clear stabilization with β and α doesn't need much to be burnt off. I will remove the the first 49.

```
# burning
B = length(theta[,1])
theta.burn = theta[50:B,]
# visuals
par(mfrow = c(1,2))
plot.ts(theta.burn[,1], main = "alpha posterior burned trace plot", ylab = "alpha")
plot.ts(theta.burn[,2], main = "beta posterior burned trace plot", ylab = "beta")
```

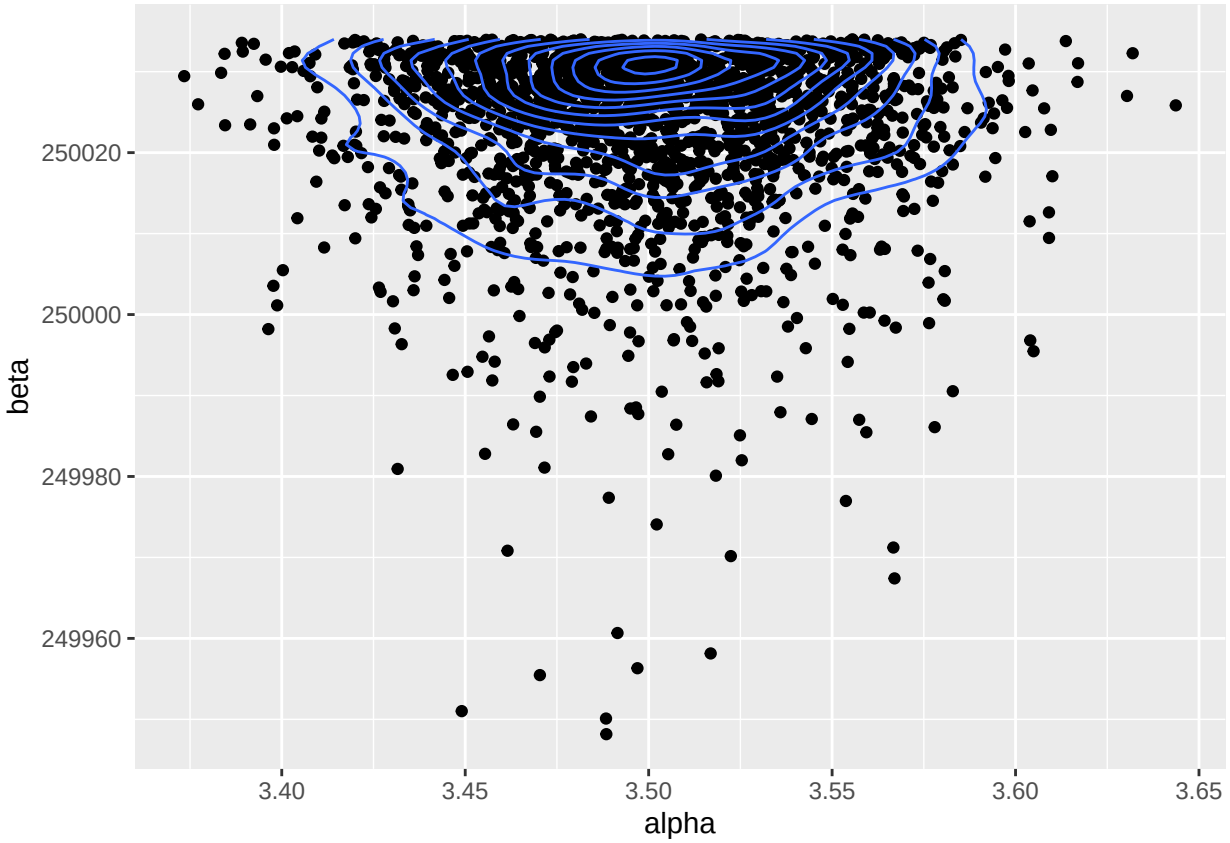



```
acf(theta.burn[,1], main = "alpha burned")
acf(theta.burn[,2], main = "beta burned")
```

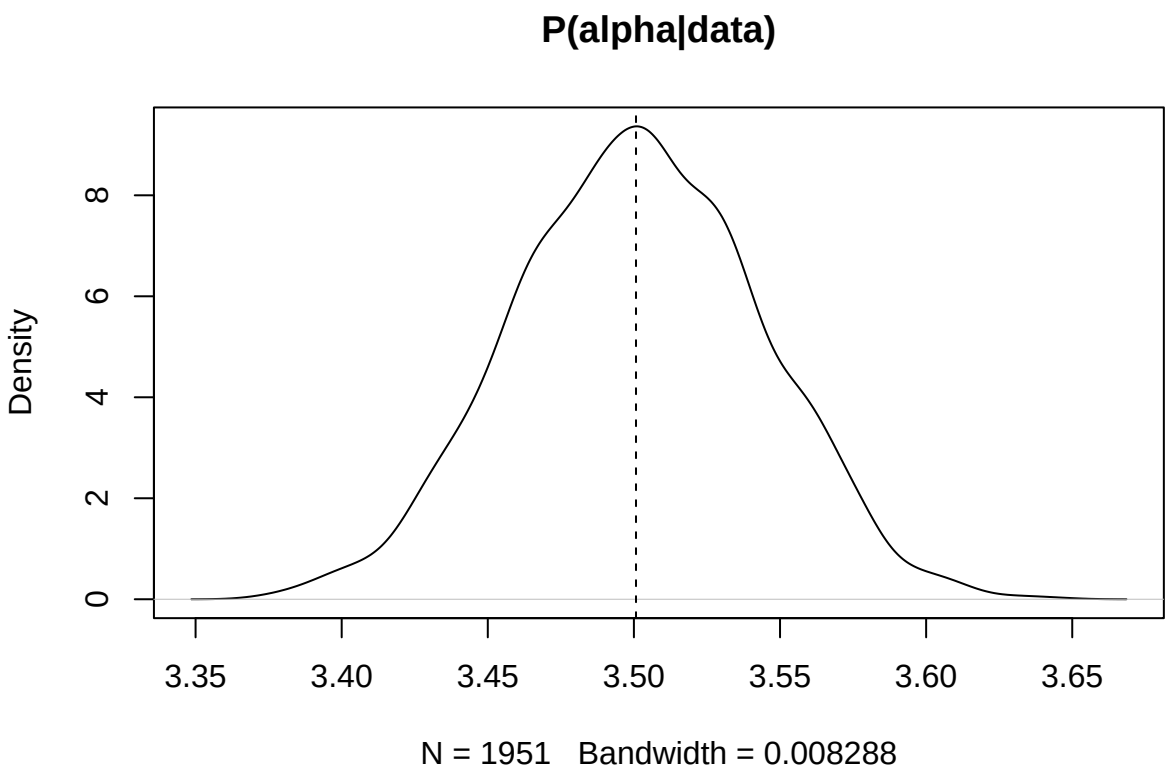





```
# generate bivariate scatterplot
post <- data.frame(theta.burn)
ggplot(post, aes(x = alpha, y = beta)) + geom_point() + geom_density2d()
```




```
# plot alpha|data
plot(density(theta.burn[,1]),
     main = "P(alpha|data)")
dens = density(theta.burn[,1])
point.est.a = dens$x[dens$y == max(dens$y)]
abline(v = point.est.a,lty=2)
```

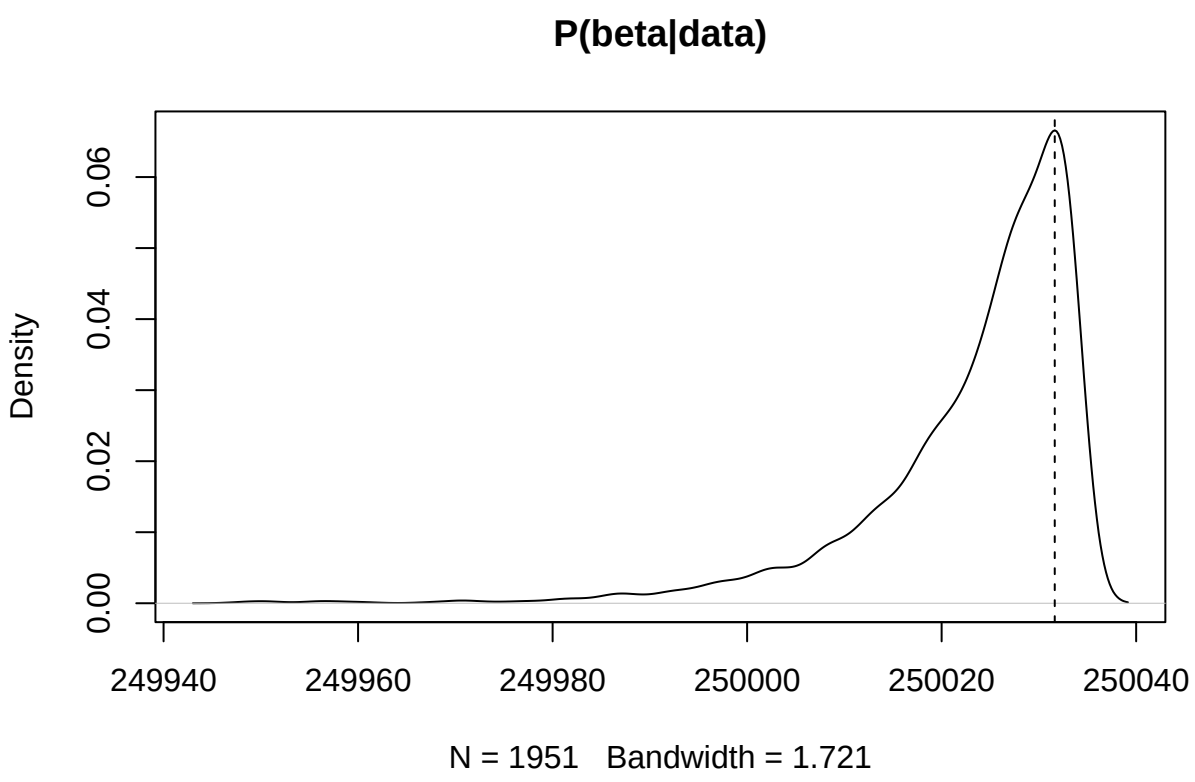


(d)

```
# credible interval
c(quantile(theta.burn[,1],.025),quantile(theta.burn[,1],.975))
```

```
##     2.5%    97.5%
## 3.420024 3.580153
```

```
# plot beta|data
plot(density(theta.burn[,2]),
     main = "P(beta|data)")
dens = density(theta.burn[,2])
point.est.b = dens$x[dens$y == max(dens$y)]
abline(v = point.est.b,lty=2)
```



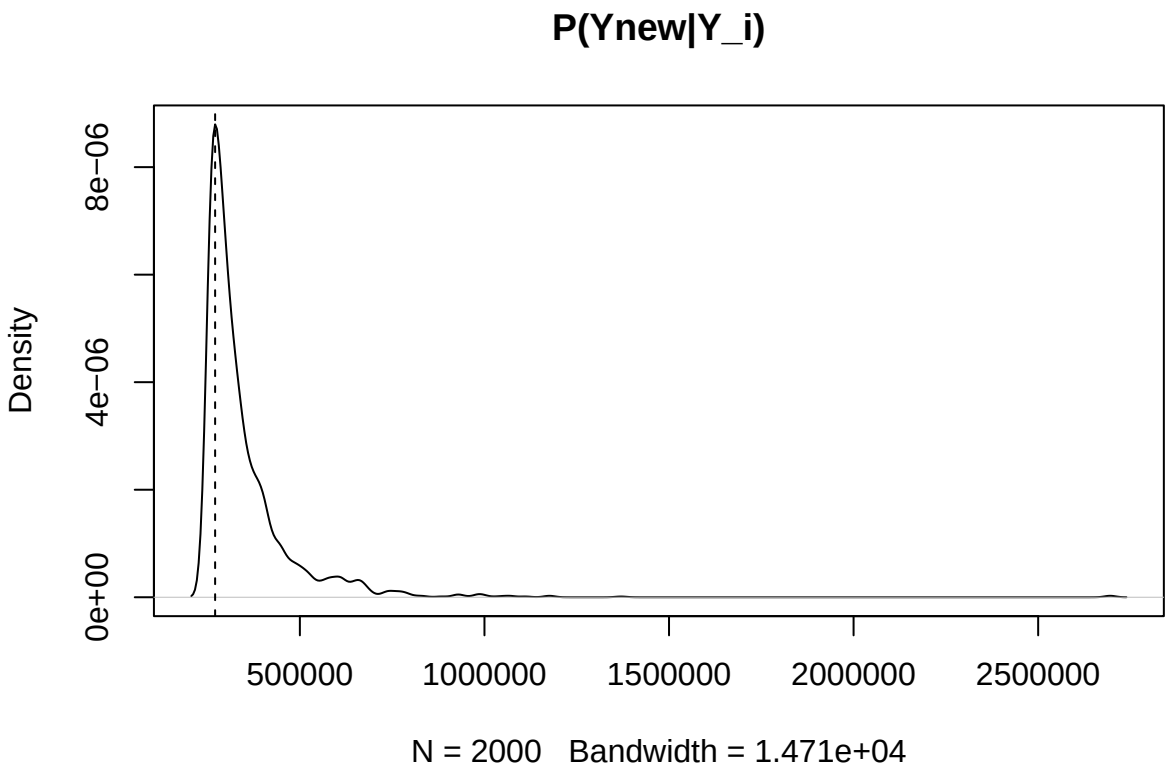

```
# credible interval
c(quantile(theta.burn[,2],.025),quantile(theta.burn[,2],.975))
```

```
##     2.5%    97.5%
## 249994.6 250033.7
```

param.	CI lower	mode	CI upper
α	3.420024	3.50072	3.580153
β	249994.6	250031.6	250033.7

Note that the β parameter can be interpreted as an average lower end of the income data. Our most dense point for this parameter is 250031.6 and this is very close to our data's actual minimum of 250034.

```
set.seed(90909)
# get the the 2000 samples
ynew = rpareto(1951,location = theta.burn[,2], shape = theta.burn[,1])
ynew = sample(ynew, size = 2000, rep = T)
# visuals
dens = density(ynew)
plot(dens, main = "P(Ynew|Y_i)")
point.est.y = dens$x[dens$y ==max(dens$y)]
abline(v=point.est.y, lty =2)
```



(d)

```
# credible interval
lowr = quantile(ynew,.025)
uppr = quantile(ynew,.975)
sprintf("Mode : %0.2f",point.est.y)
```

```
## [1] "Mode : 270397.53"
```

```
sprintf("Credible Interval : ( %0.2f , %0.2f )",lowr,uppr)
```

```
## [1] "Credible Interval : ( 252707.12 , 665193.97 )"
```

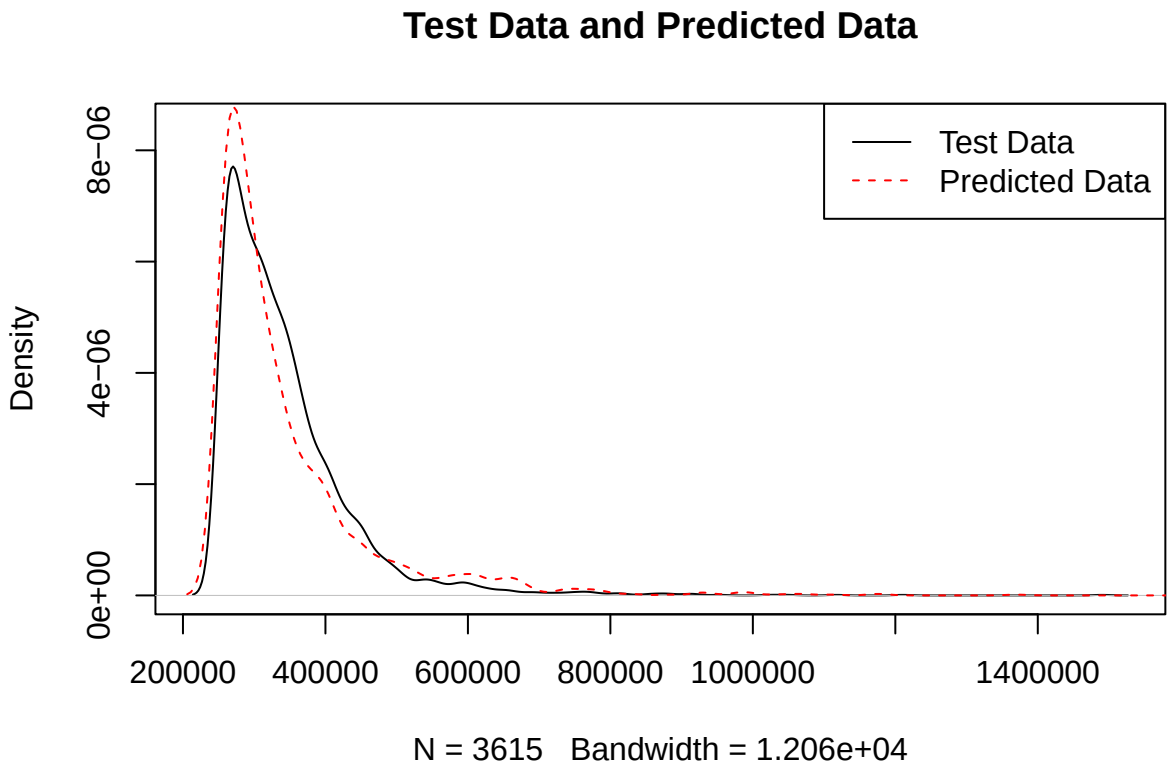
```
# test data
y.test <- as.vector(read.csv("income20test-2.csv", header = T)[,2])
# count how much falls into the credible interval
I = as.numeric(y.test > lowr & y.test < uppr)
sum(I)/length(I)
```

```
## [1] 0.9634855
```

The Pareto distribution seems to be an accurate tool for modeling the top 20% of the incomes of US adults. Our credible interval captures more than 95% of the data as we see in the above script. We can also overly the predictive model over or actual test data and observe the similarities in the test data and predicted data; they almost follow each other exactly. The modes are almost similar too; they are presented below. At this current seed, they present only difference of 15.3 dollars which is arguably negligible considering these high end salaries.

```
# overlay prediction on test data
dens.test = density(y.test)
plot(dens.test,
```

```
     ylim = c(0,8.5e-06),
     main = "Test Data and Predicted Data")
lines(dens,lty = 2, col = 'red')
legend("topright", legend = c("Test Data", "Predicted Data"), col = c("black", "red"), lty = c(1, 2))
```




```
# modes
point.est.y
```

```
## [1] 270397.5
```

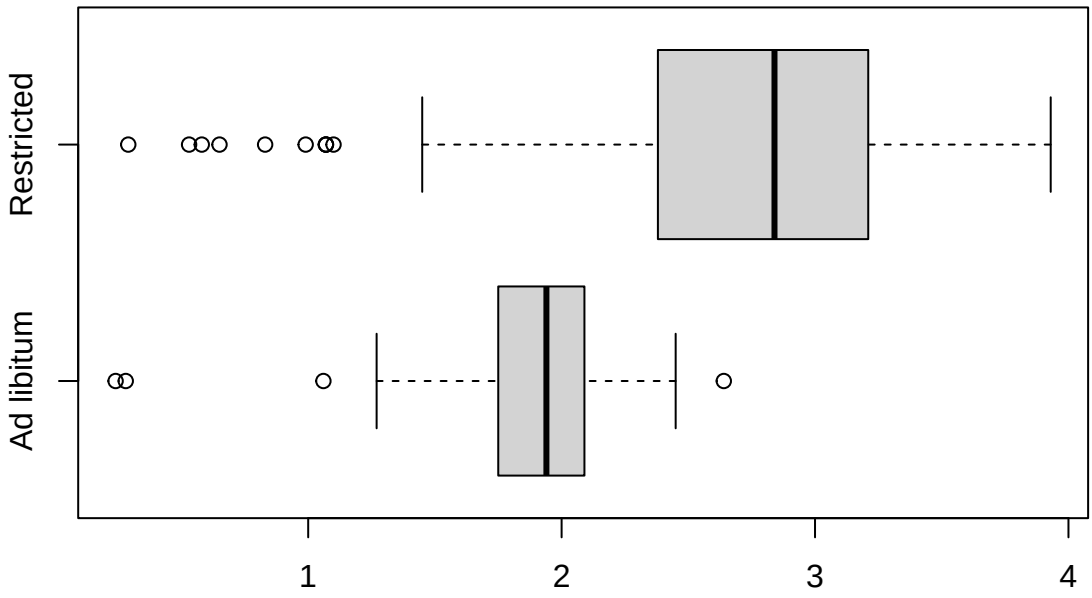
```
dens.test$x[dens.test$y == max(dens.test$y)]
```

```
## [1] 270382.2
```

Problem 2

```
# get the rat data
rat = as.matrix(read.csv("rat-3.csv"))
```

```
# EDA
# boxplot
adlib  = rat[rat[,2] == 0,1]
restr = rat[rat[,2] == 1,1]
boxplot(adlib,restr,names = c("Ad libitum", "Restricted"), horizontal = T)
```



problem (a)

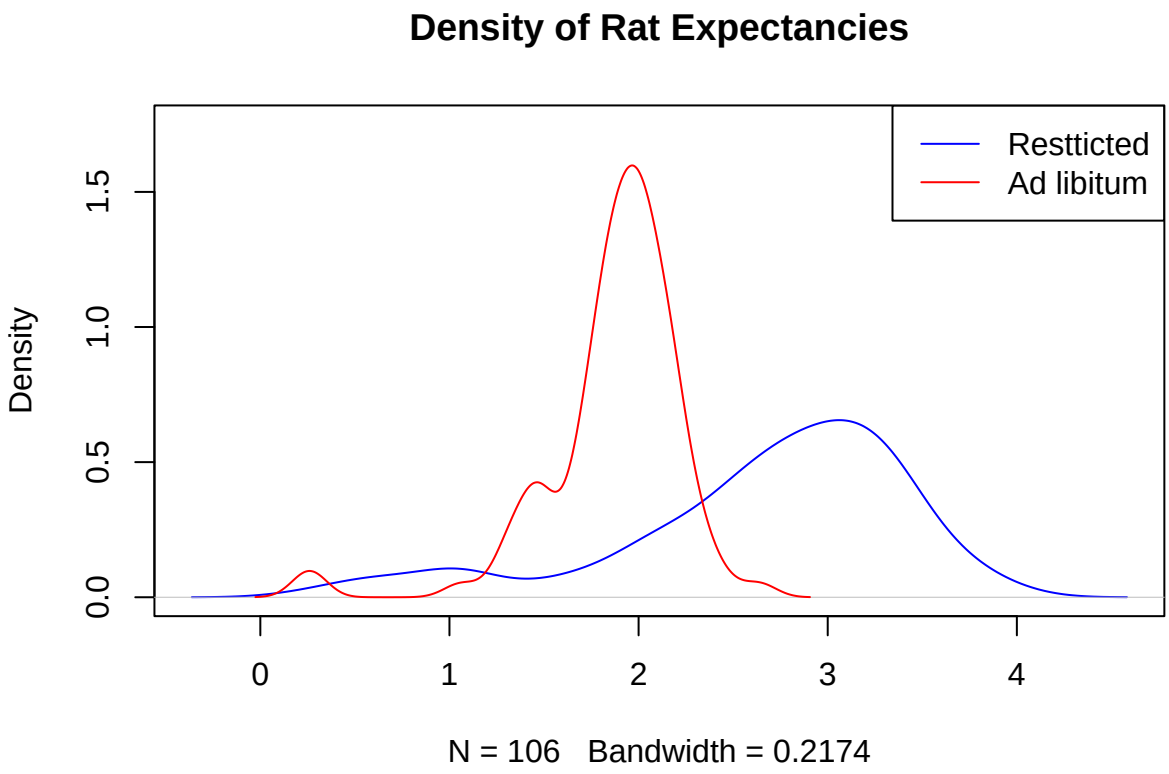
```
# 5p summaries
summary(restr)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   0.290   2.382   2.840   2.654   3.205   3.930
```

```
summary(adlib)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   0.240   1.758   1.940   1.869   2.083   2.640
```

```
plot(density(restr), ylim = c(0,1.75), col = "blue", main = "Density of Rat Expectancies")
lines(density(adlib),col = 'red')
legend("topright", legend = c("Restticted", "Ad libitum"), col = c("blue", "red"), lty = c(1,1))
```



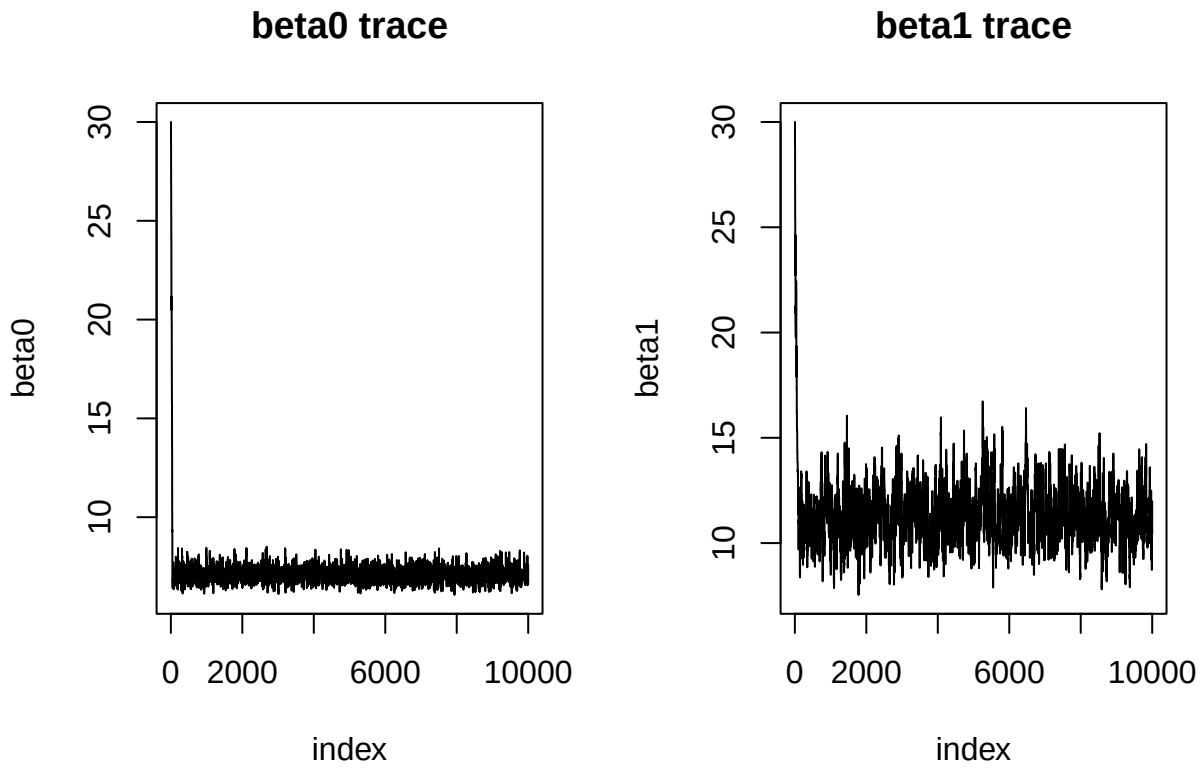
We can observe that the the *Ad libitum* group tends to have a lower average expectancy that the Restricted group. There can support the speculation that rats eat themselves to death. The span for *Ad libitum* is also smaller than the span for the Restricted rats; the Restricted deaths varies more in time.

```
# establish functions
# prior
prior = function(beta0,beta1){1}

# likelihood
like = function(X,Y,beta0,beta1){
  sigma = log(beta0 + beta1*X)
  D = dweibull(Y,shape = 4, scale = sigma)
  D = D[!is.na(D) == 1]
  prod(D)
}

# log-likelihood
llike = function(X,Y,beta0,beta1){
  sigma = log(beta0 + beta1*X)
  D = log(dweibull(Y,shape = 4, scale = sigma))
  return(sum(D))
}

# proposal
rPro = function(beta0,beta1,s,S){
```

```
  rmvnorm(1,mean = c(beta0,beta1), sigma = s*S)
}

MH2 <- function(data,group,B = 10000,beta.init = c(1,1), tuning.param = 1, cov.mat = diag(2)){
  # initialize
  accept = 0
  s = tuning.param # tuning param
  S = cov.mat
  y = data
  x = group

  # posterior params.
  beta0.post <- beta1.post <- numeric()
  beta0.post[1] = beta.init[1]
  beta1.post[1] = beta.init[2]

  # loop
  for(b in 2:(B+1)){
    # proposal
    beta.star = rPro(beta0.post[b-1],beta1.post[b-1],s,S)

    # Ratio of Densities
    # r.num = like(x,y,beta.star[1],beta.star[2]) * prior(beta.star[1],beta.star[2])
    # r.den = like(x,y,beta0.post[b-1],beta1.post[b-1]) * prior(beta0.post[b-1],beta1.post[b-1])
    # r = r.num/r.den

    # Ratio of Densities log method
    # because the prior is uniform, we can just disregard it
    alpha = llike(x,y,beta.star[1],beta.star[2]) - llike(x,y,beta0.post[b-1],beta1.post[b-1])
    r = exp(alpha)

    # Accept/Reject
    if(runif(1) < min(1, r)){
     beta0.post[b] = beta.star[1]
     beta1.post[b] = beta.star[2]
     accept = accept + 1
    }else{
     beta0.post[b] = beta0.post[(b-1)]
     beta1.post[b] = beta1.post[(b-1)]
    }

  }
  # return
  beta0 = beta0.post
  beta1 = beta1.post
  beta.post = cbind(beta0,beta1)
  list("beta" = beta.post,"AR" = accept/B)
}

# running the program
MH2(rat[,1],rat[,2],beta.init = c(30,30),tuning.param = 1) -> mh.obj
beta.post = mh.obj$beta
ar = mh.obj$AR
```

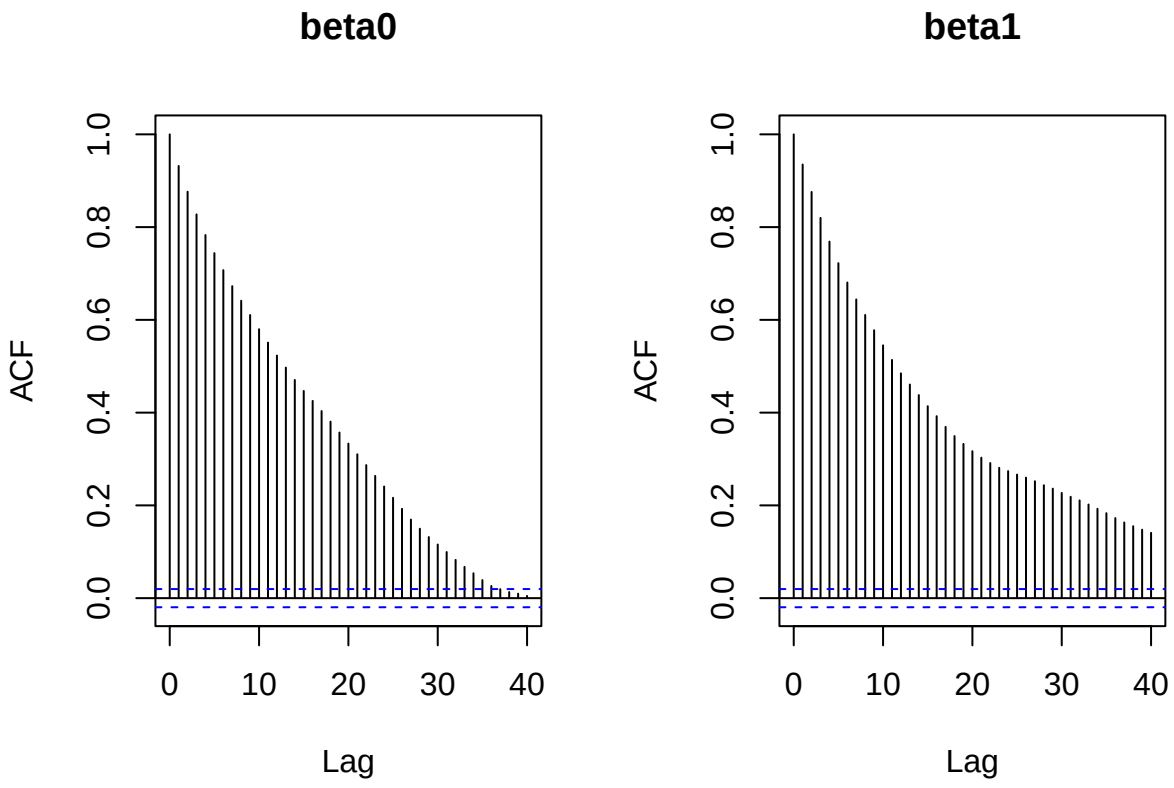
(b)

```
#traceplots
par(mfrow = c(1,2))
ts.plot(beta.post[,1], main = "beta0 trace", xlab = "index", ylab = "beta0")
ts.plot(beta.post[,2], main = "beta1 trace", xlab = "index", ylab = "beta1")
```



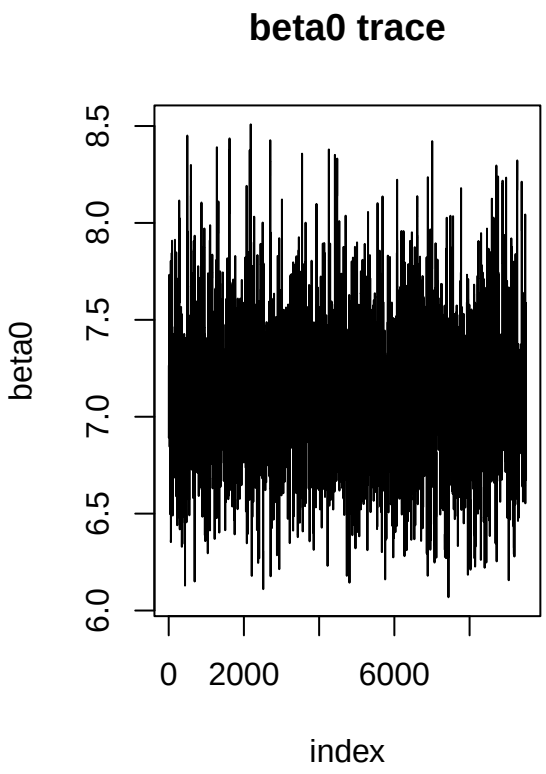
(c)

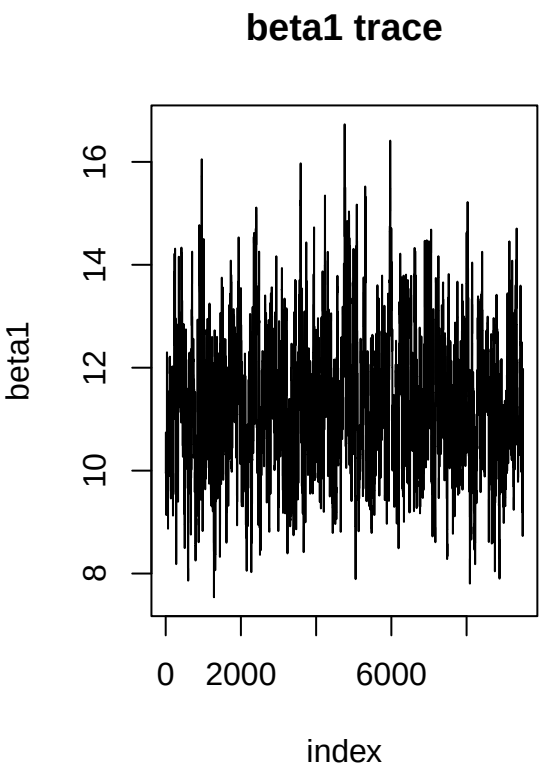
```
# autocorrelation
acf(beta.post[,1], main = "beta0")
acf(beta.post[,2], main = "beta1")
```



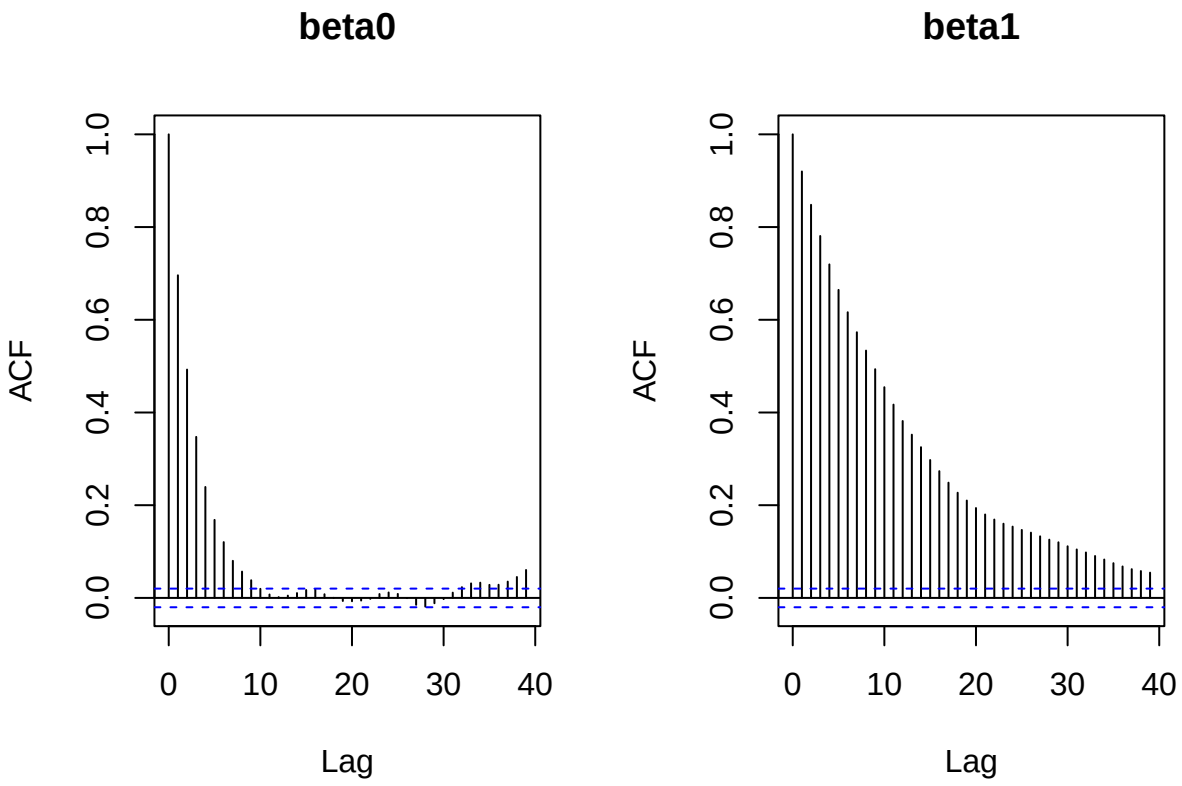
Going to burn 499.

```
burn = 500
beta.burn = beta.post[burn:10001,]
#traceplots
par(mfrow = c(1,2))
ts.plot(beta.burn[,1], main = "beta0 trace", xlab = "index", ylab = "beta0")
ts.plot(beta.burn[,2], main = "beta1 trace", xlab = "index", ylab = "beta1")
```



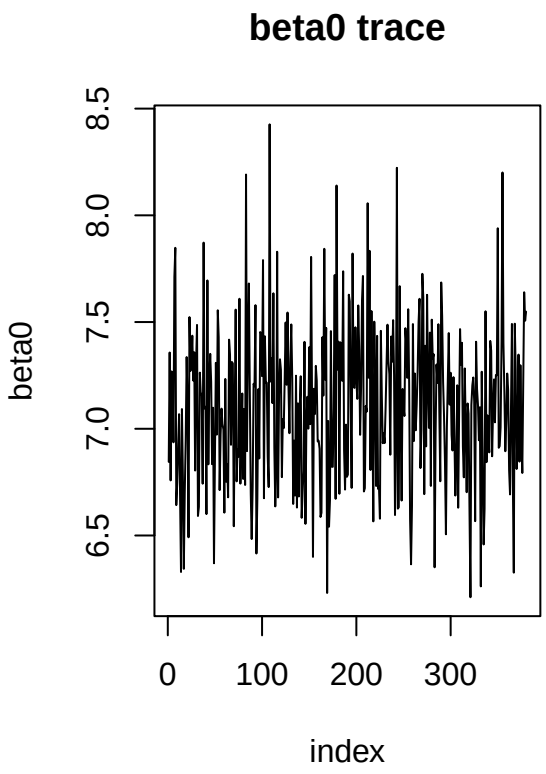


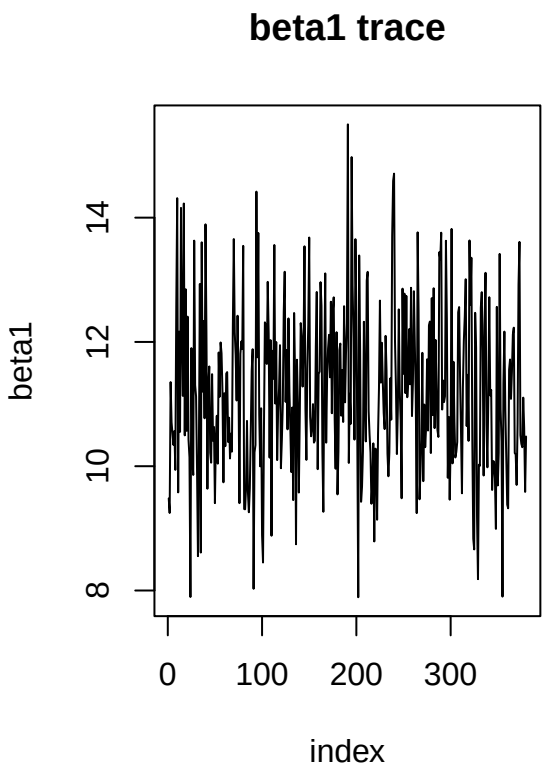

```
# autocorrelation
acf(beta.burn[,1], main = "beta0")
acf(beta.burn[,2], main = "beta1")
```



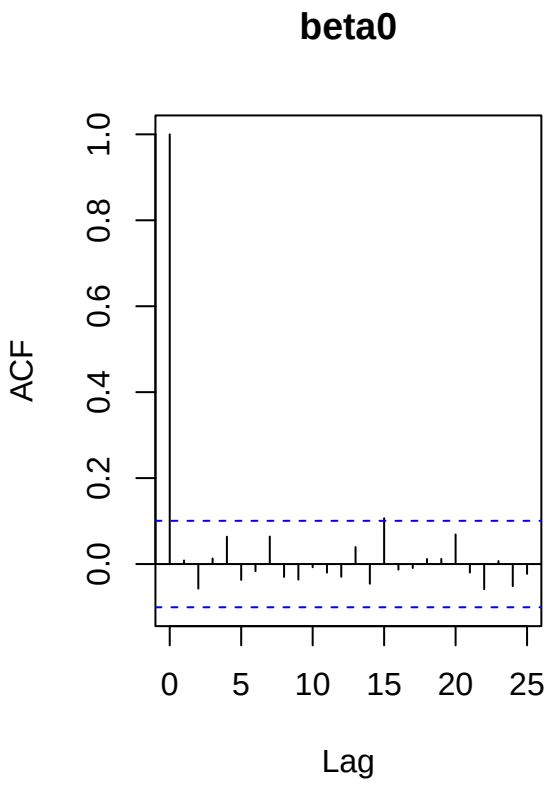
Going to thin by taking the 25th lag

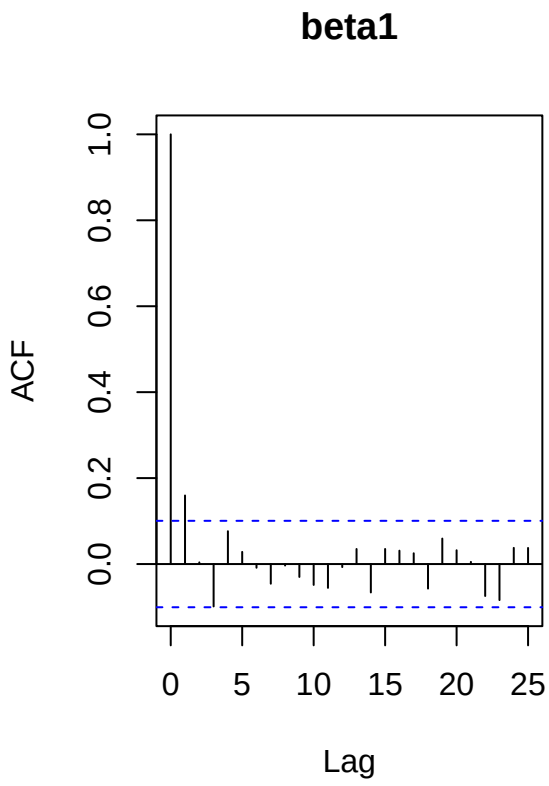
```
#thinning
k = 25*(1:9502)
k = k[k < 9502]
beta.thin = beta.burn[k,]
#traceplots
par(mfrow = c(1,2))
ts.plot(beta.thin[,1], main = "beta0 trace", xlab = "index", ylab = "beta0")
ts.plot(beta.thin[,2], main = "beta1 trace", xlab = "index", ylab = "beta1")
```





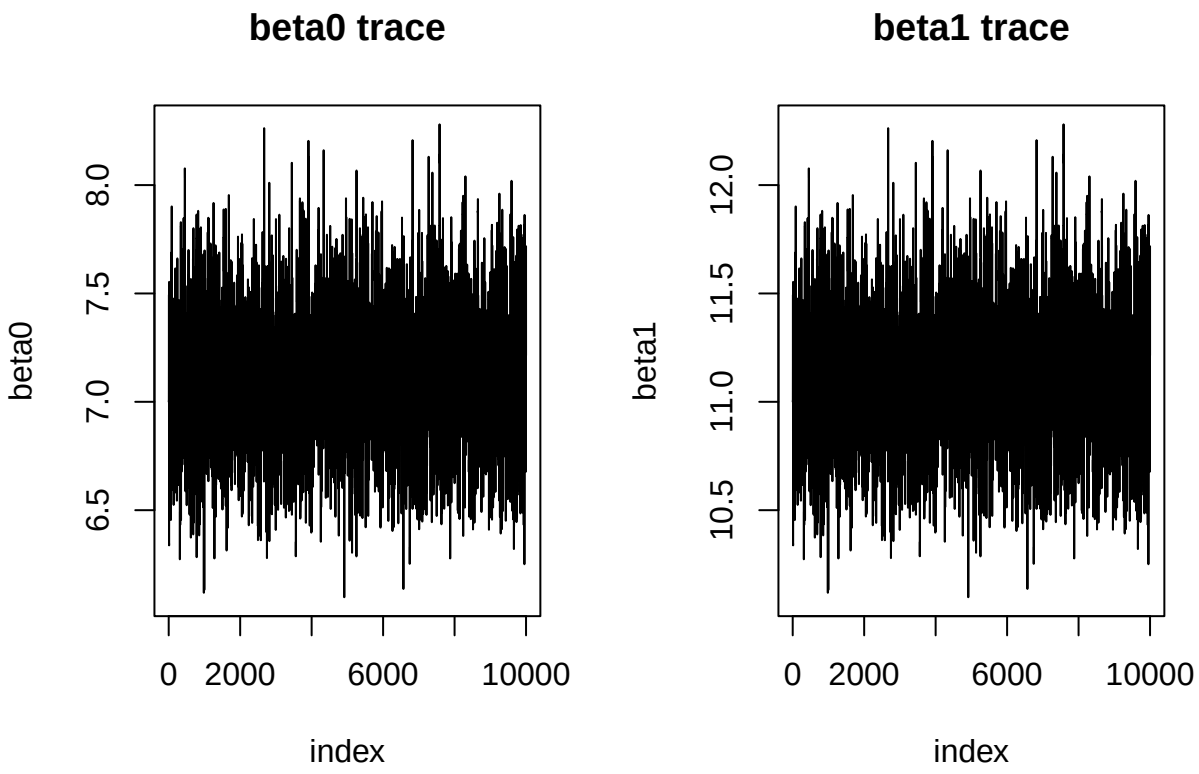

```
# autocorrelation
acf(beta.thin[,1], main = "beta0")
acf(beta.thin[,2], main = "beta1")
```



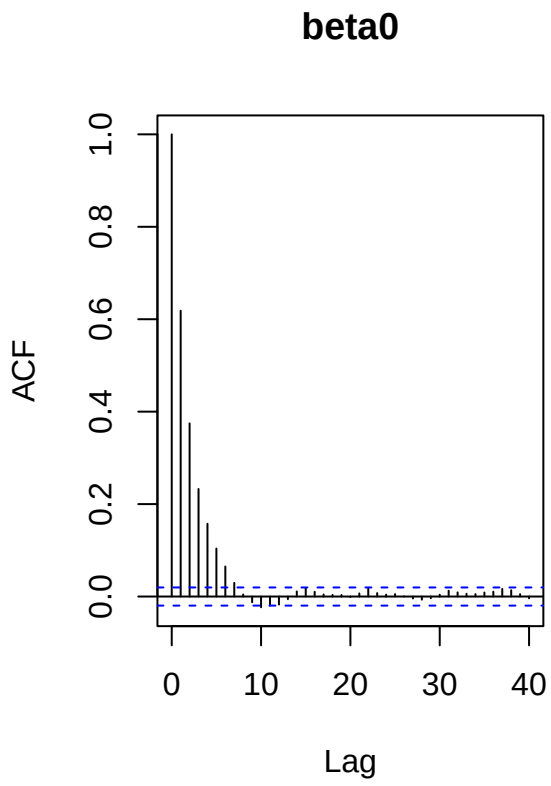


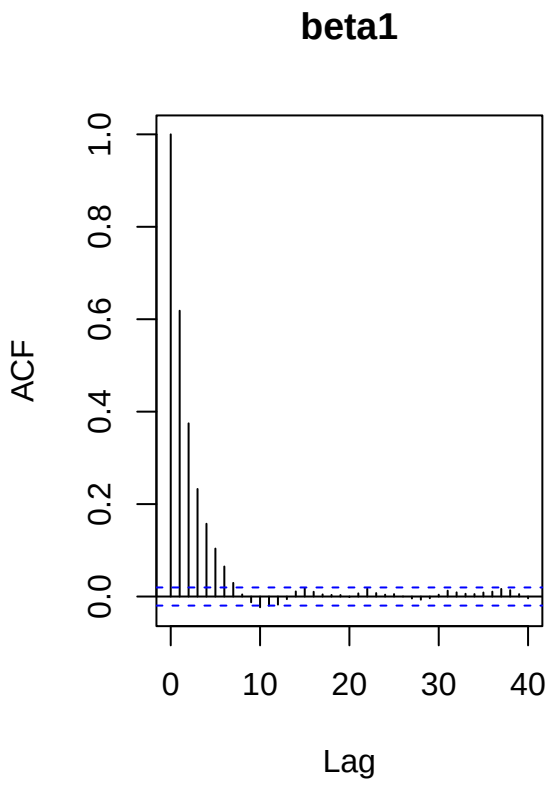
Tuning to get good mixing (use initial 7,11)

```
tuncov = matrix(c(1,1,1,1),2,2)
MH2(rat[,1],rat[,2],beta.init = c(7,11),tuning.param = .84, cov.mat =tuncov ) -> mh.obj
beta.post = mh.obj$beta
ar = mh.obj$AR
#traceplots
par(mfrow = c(1,2))
ts.plot(beta.post[,1], main = "beta0 trace", xlab = "index", ylab = "beta0")
ts.plot(beta.post[,2], main = "beta1 trace", xlab = "index", ylab = "beta1")
```




```
# autocorrelation
acf(beta.post[,1], main = "beta0")
acf(beta.post[,2], main = "beta1")
```





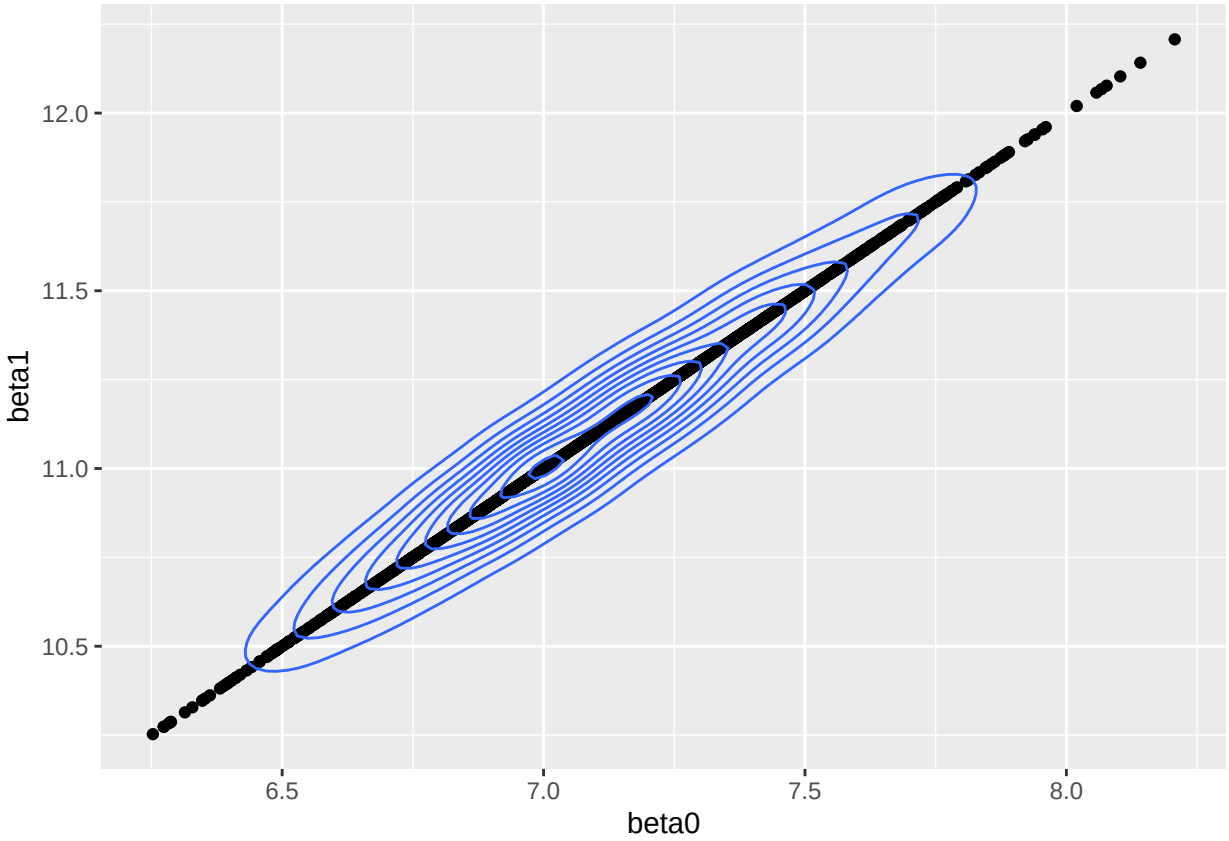

```
if(ar > .75){print("YAY!")}
```

This tuning parameter .82 with a tuning One's matrix gives us an eventually matrix of

$$\begin{bmatrix} 0.82 & 0.82 \\ 0.82 & 0.82 \end{bmatrix}$$

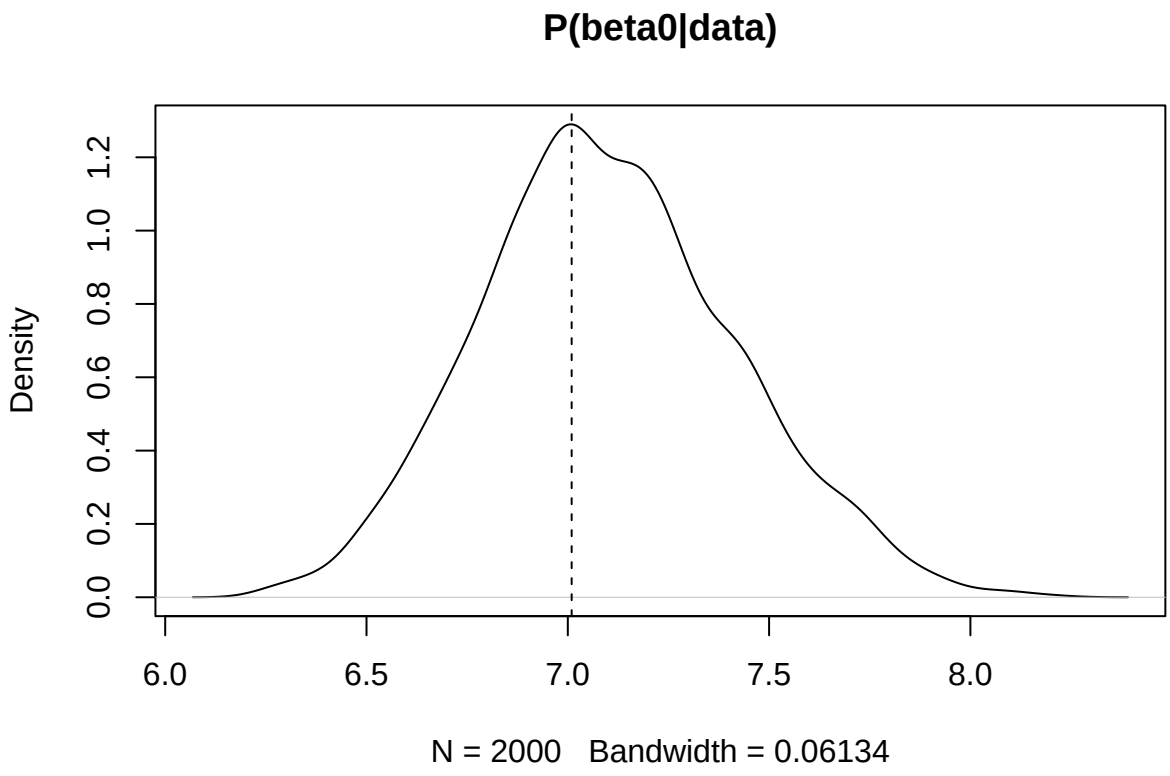
This gives an acceptance rate of around 39% and rather low autocorrelation without giving us NaNs or singularities.

```
# generate bivariate scatterplot
samp = sample(1:length(beta.post[,1]), size = 2000, rep = F)
beta.post = beta.post[samp,]
post <- data.frame(beta.post)
ggplot(post, aes(x = beta0, y = beta1)) + geom_jitter() + geom_density2d()
```

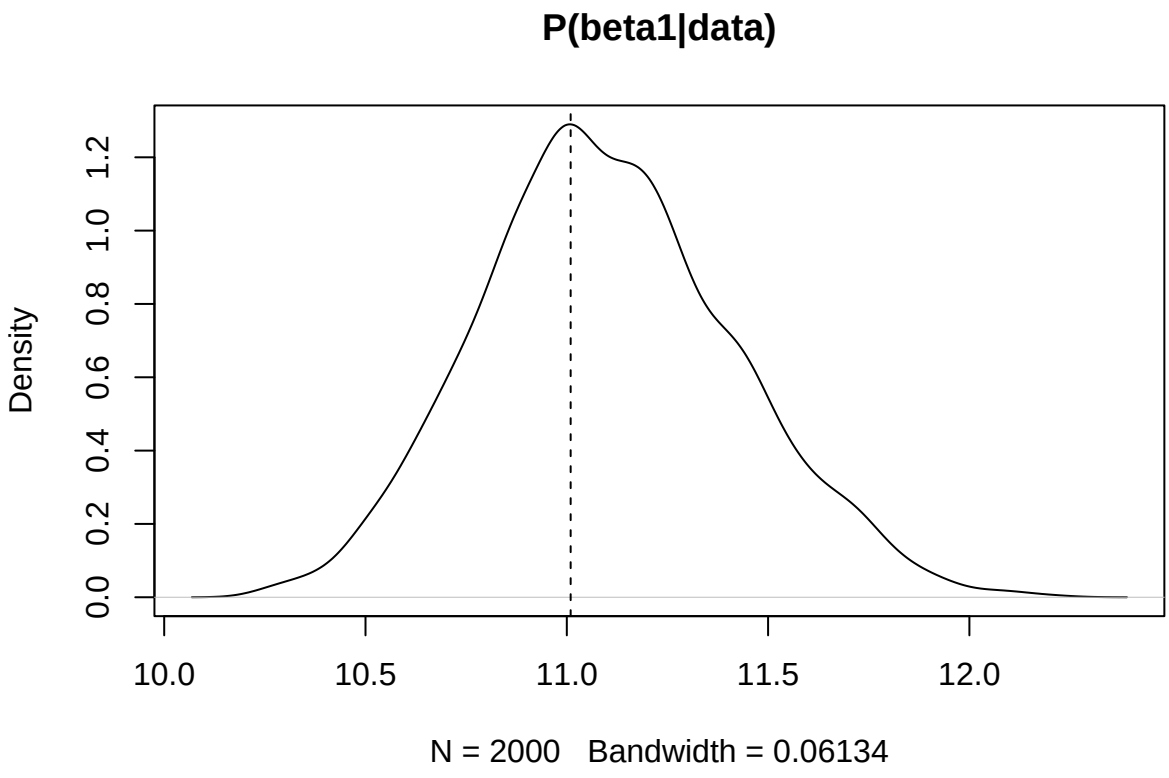


(d)

```
# plot alpha|data
plot(density(beta.post[,1]),
     main = "P(beta0|data)")
dens = density(beta.post[,1])
point.est.0 = dens$x[dens$y == max(dens$y)]
abline(v = point.est.0,lty=2)
```




```
# plot beta/data
plot(density(beta.post[,2]),
     main = "P(beta1|data)")
dens = density(beta.post[,2])
point.est.1 = dens$x[dens$y == max(dens$y)]
abline(v = point.est.1,lty=2)
```




```
# credible interval/MAP
sprintf("Credible Interval for beta_0: ( %f , %f )",quantile(beta.post[,1],.025),quantile(beta.post[,1]
```

```
## [1] "Credible Interval for beta_0: ( 6.512799 , 7.753556 )"
```

```
sprintf("MAP : %f",point.est.0)
```

```
## [1] "MAP : 7.009553"
```

```
sprintf("Credible Interval for beta_1 : ( %f , %f )",quantile(beta.post[,2],.025),quantile(beta.post[,2
```

```
## [1] "Credible Interval for beta_1 : ( 10.512799 , 11.753556 )"
```

```
sprintf("MAP : %f",point.est.1)
```

```
## [1] "MAP : 11.009553"
```

I want to look at this in context of the σ_i

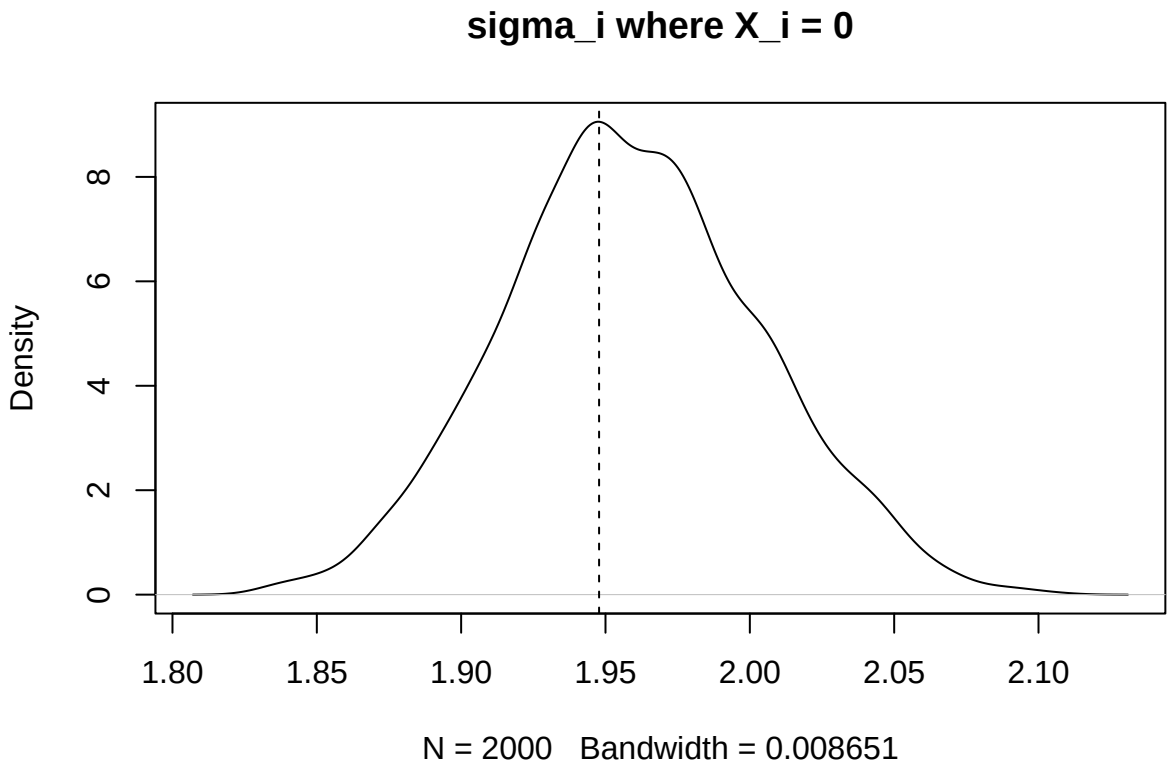
```
# sigmas
sig0 = log(beta.post[,1])
sig1 = log(beta.post[,1] + beta.post[,2])
summary(sig0)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   1.833   1.929   1.958   1.959   1.988   2.105
```

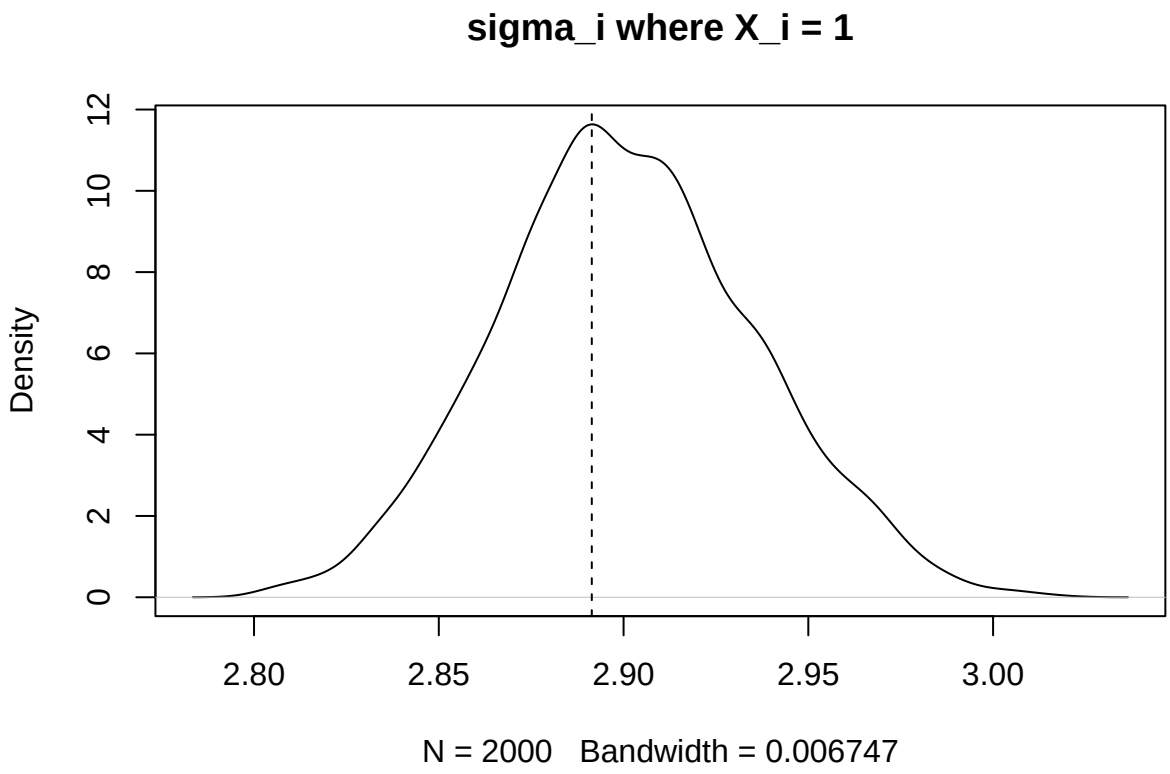
```
summary(sig1)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   2.804   2.877   2.900   2.901   2.923   3.016
```

```r
# plot sigma where x = 0
plot(density(sig0),
     main = "sigma_i where X_i = 0")
dens = density(sig0)
point.est.0 = dens$x[dens$y == max(dens$y)]
abline(v = point.est.0,lty=2)
```




```r
# plot sigma where x = 1
plot(density(sig1),
     main = "sigma_i where X_i = 1")
dens = density(sig1)
point.est.1 = dens$x[dens$y == max(dens$y)]
abline(v = point.est.1,lty=2)
```




```
# credible interval + MAP
sprintf("Credible Interval for sigma when x = 0 : ( %f , %f )",quantile(sig0,.025),quantile(sig0,.975))
```

```
## [1] "Credible Interval for sigma when x = 0 : ( 1.873769 , 2.048152 )"
```

```
sprintf("MAP : %f",point.est.0)
```

```
## [1] "MAP : 1.947783"
```

```
sprintf("Credible Interval for sigma when x = 1 : ( %f , %f )",quantile(sig1,.025),quantile(sig1,.975))
```

```
## [1] "Credible Interval for sigma when x = 1 : ( 2.834718 , 2.970779 )"
```

```
sprintf("MAP : %f",point.est.1)
```

```
## [1] "MAP : 2.891399"
```

When we apply β_0 and β_1 to into standard deviation, there is clearly a difference that the value of x_i plays on the spread survival time. Note how the two credible intervals do not even intersect; this implies the a significant affect that group the rat is in will have on its survival.

(e) Our model's HR can be expressed as

$$\frac{h(y|4,\sigma(x=1))}{h(y|4,\sigma(x=0))} = (\frac{\sigma_{x=0}}{\sigma_{x=1}})^4$$

```
# HR function
HazRat = function(sigma0,sigma1){
(sigma0/sigma1)^4
```

```
}
# get credible interval
HazRat(sig0,sig1) -> HRvec
dens = density(HRvec)
hrmap = dens$x[dens$y == max(dens$y)]
sprintf("Credible Interval for HR : ( %f , %f )",quantile(HRvec,.025),quantile(HRvec,.975))
```

```
## [1] "Credible Interval for HR : ( 0.190908 , 0.225926 )"
```

```
sprintf("MAP : %f",hrmap)
```

```
## [1] "MAP : 0.205705"
```

HR is definitely below 1. This implies that the group (x = 0) in the denominator of our HR expression is experiencing more less time to event that the group (x = 1) represented in the numerator. For context of the problem, we have implied that *Ad libitum* group has a significantly higher risk of failure as the credible interval does not include 1 (nor does it include 0.5). Regarding the research question, The MAP and credible interval of the Hazard Ratio does imply that there is strong evidence support the claim that "rats tend to eat themselves to an earlier grave."

(f) we know that EV(Z|0,1) $= e^{z-e^z}$

we also know that $log(Y) = \beta_0 + \beta_1 x + \tau Z$

which implies $Z = \frac{1}{\tau}(log(Y) - \beta_0 - \beta_1 x) = g^{-1}(y)$ when we define Z in terms of Y.

The Jacobian is given by

$$J = |\frac{d}{dy}(\frac{1}{\tau}(log(Y) - \beta_0 - \beta_1 x))| = |\frac{1}{\tau y}|$$

Thus the transform can be expressed as follows

$$f_Z(z) = e^{\frac{1}{\tau}(\log(y) - \beta_0 - \beta_1 x) - e^{\frac{1}{\tau}(log(y) - \beta_0 - \beta_1 x)}} \cdot |\frac{1}{\tau y}|$$

notice that $e^{\frac{1}{\tau}(\log(y) - \beta_0 - \beta_1 x)} = y^{\frac{1}{\tau}} e^{-\frac{1}{\tau}(\beta_0 + \beta_1 x)}$

so then the transform then becomes

$$\begin{aligned} f_Z(z) &= y^{\frac{1}{\tau}} e^{-\frac{1}{\tau}(\beta_0 + \beta_1 x) - y^{\frac{1}{\tau}} e^{-\frac{1}{\tau}(\beta_0 + \beta_1 x)}} \cdot |\frac{1}{\tau y}| \\ &= y^{\frac{1}{\tau}} e^{-\frac{1}{\tau}(\beta_0 + \beta_1 x)} e^{-y^{\frac{1}{\tau}} e^{-\frac{1}{\tau}(\beta_0 + \beta_1 x)}} \cdot |\frac{1}{\tau y}| \\ &= \frac{1}{\tau} y^{\frac{1}{\tau} - 1} e^{-\frac{1}{\tau}(\beta_0 + \beta_1 x)} e^{-y^{\frac{1}{\tau}} e^{-\frac{1}{\tau}(\beta_0 + \beta_1 x)}} \\ &= \frac{1}{\tau} [e^{\frac{1}{\tau}(\beta_0 + \beta_1 x)}]^{-1} y^{\frac{1}{\tau} - 1} e^{-y^{\frac{1}{\tau}} [e^{\frac{1}{\tau}(\beta_0 + \beta_1 x)}]^{-1}} \end{aligned}$$

This reveals that in for Weibull distributed Y that

- for shape, $\alpha = \frac{1}{\tau}$
- for scale,
  - when x = 1, $\sigma = e^{\frac{1}{\tau}(\beta_0 + \beta_1)}$
  - when x = 0, $\sigma = e^{\frac{1}{\tau}\beta_0}$