OM386 Advanced Data Analytics in Marketing
Assignment 4

Due: April 6th, 11:59pm

Name – Mahika Bansal (mb62835)

Censored Regression and Its Bayesian Estimation

In this exercise, we will apply censored regression to the dataset "CreditCard_SOW_Data2.csv". The dataset has the following variables.

ConsumerID	ID's of the sampled consumers
SOW	The card's share of wallet in the consumer's total monthly spending
Promotion	Index of monthly promotion activity –higher index indicates more promotions
Balance	The customer's unpaid balance at the beginning of the month

1). In this data set, the share of wallet (SOW) can be 0% or 100%. Therefore, the share of wallet is considered a truncated variable at 0% (censored at 0) or 100% (censored at 1). We would like fit the following regression model

$SOW_{ij}^{*} = \beta_0 + \beta_1 \times Balance_{ij} + \beta_2 \times Promotion_{ij} + \varepsilon_{ij}$

$SOW_{ij} = SOW_{ij}^{*} \quad \text{if} \;\; 0 < SOW_{ij}^{*} < 1$

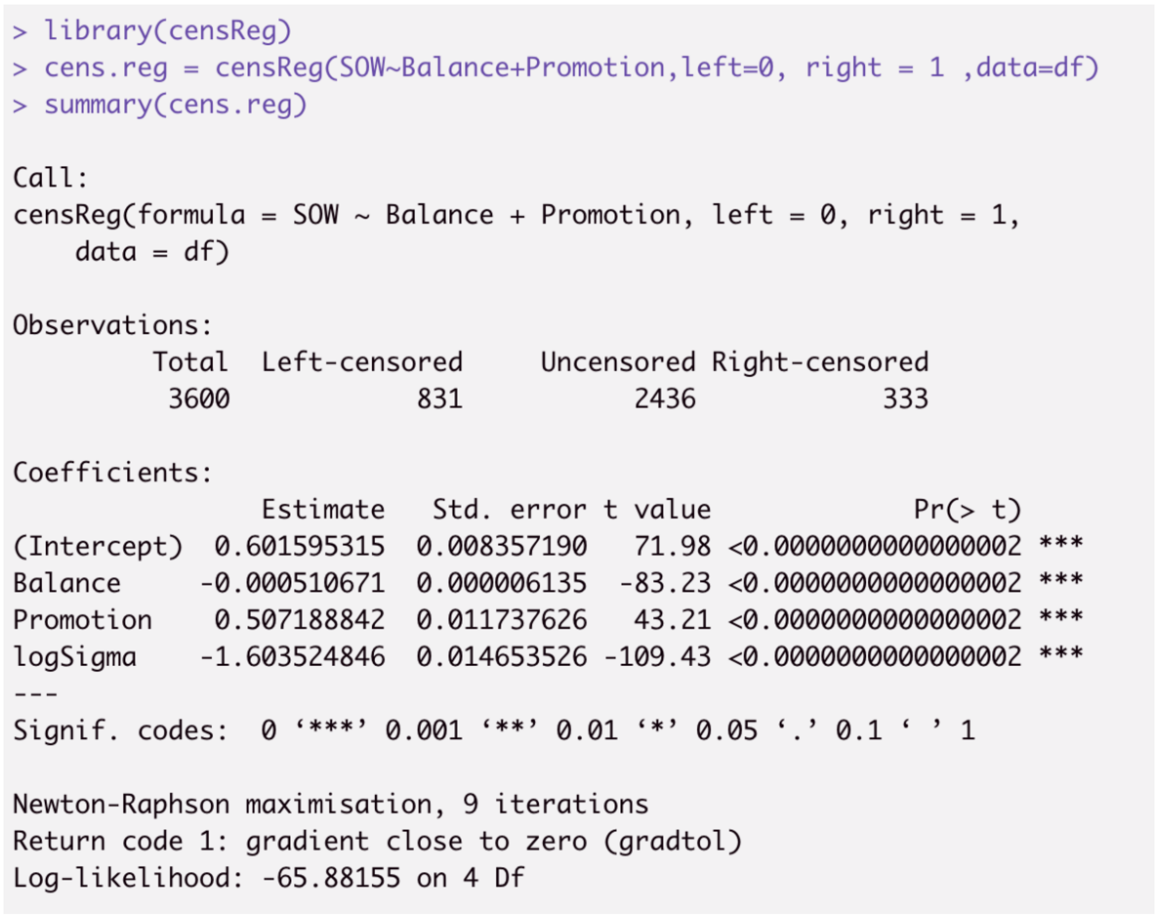
$SOW_{ij} = 0 \quad \text{if} \;\; SOW_{ij}^{*} \leq 0$

$SOW_{ij} = 1 \quad \text{if} \;\; SOW_{ij}^{*} \geq 1$

$\varepsilon_{ij} \sim N(0, \sigma^2)$ (We can reparametrize $\tau = 1/\sigma^2$, which is often named the precision)

Please use the R function censReg() in library(censReg) to fit this model by MLE. Copy and paste the summary of the results here. Interpret the parameters β_1 β_2 in the model.

Ans.



```
> library(censReg)
> cens.reg = censReg(SOW~Balance+Promotion,left=0, right = 1 ,data=df)
> summary(cens.reg)

Call:
censReg(formula = SOW ~ Balance + Promotion, left = 0, right = 1,
    data = df)

Observations:
         Total  Left-censored     Uncensored Right-censored 
          3600            831           2436            333 

Coefficients:
                Estimate   Std. error t value             Pr(> t)    
(Intercept)  0.601595315  0.008357190   71.98 <0.0000000000000002 ***
Balance     -0.000510671  0.000006135  -83.23 <0.0000000000000002 ***
Promotion    0.507188842  0.011737626   43.21 <0.0000000000000002 ***
logSigma    -1.603524846  0.014653526 -109.43 <0.0000000000000002 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Newton-Raphson maximisation, 9 iterations
Return code 1: gradient close to zero (gradtol)
Log-likelihood: -65.88155 on 4 Df
```

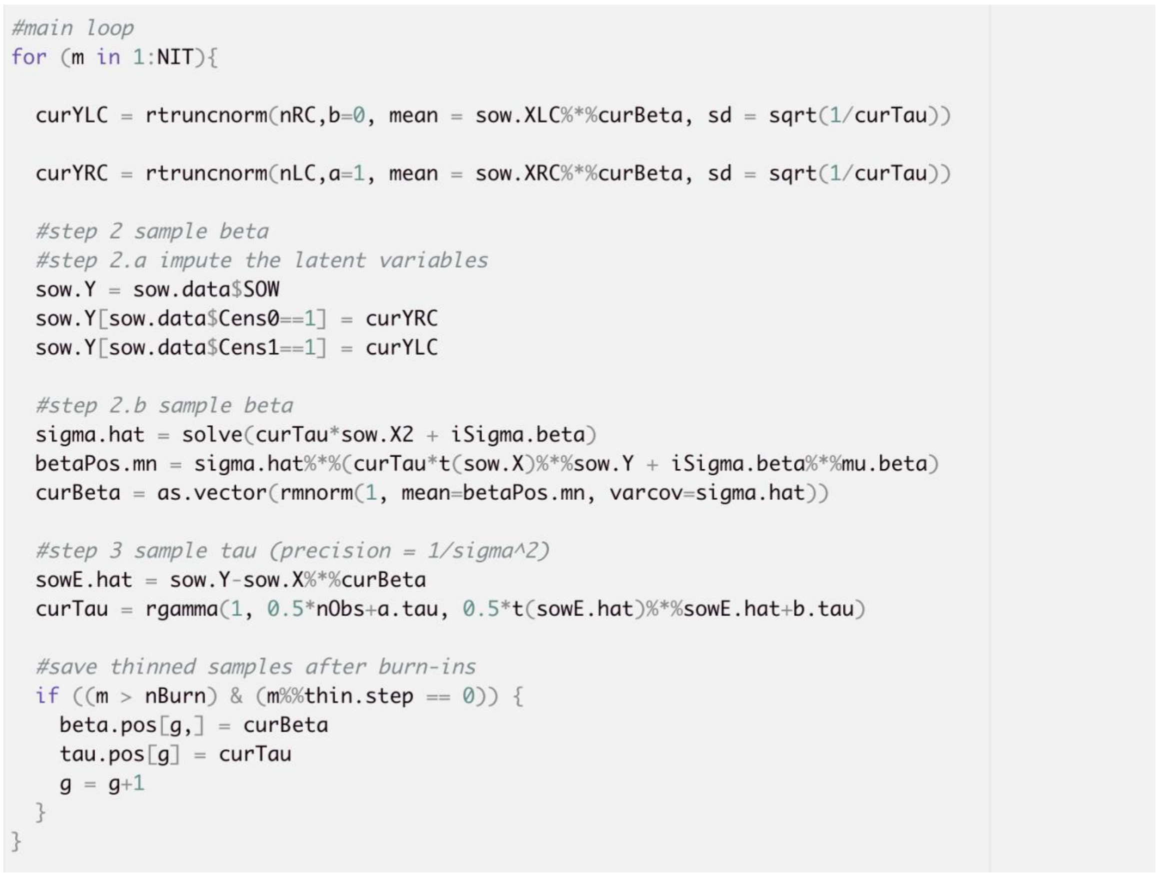
The coefficient beta1 signifies -0.0005 unit of change in SOW per unit change in balance with promotion constant. Similarly, beta2 is 0.50719 change in SOW per unit change in promotion with balance constant.

2). Next, we will fit the model above using Bayesian estimation, which involves sampling the latent SOW_{ij}^* when the observed SOW_{ij} =0 or SOW_{ij} =1. The R code using MCMC (Gibbs sampling) for this inference problem is in "Assignment-4_blankcode-2022.r". Please read the code carefully and fill in the code for sampling the latent SOW_{ij}^*, the regression coefficients and the precision (inverse of the variance) of the error. You may use the rtruncnorm( ) function in the library(truncnorm) to sample from truncated normal distributions. This method is called data augmentation, which is widely applied in Bayesian statistics for missing data.

Please run the completed code. Use the plot() function to plot the posterior sampling chains and hist() to plot posterior histograms for β_0, β_1, β_2 and τ. Copy and paste the results here. Please also calculate the 95% posterior intervals for β_0, β_1, β_2 and τ. Copy and paste the results here.

Ans.

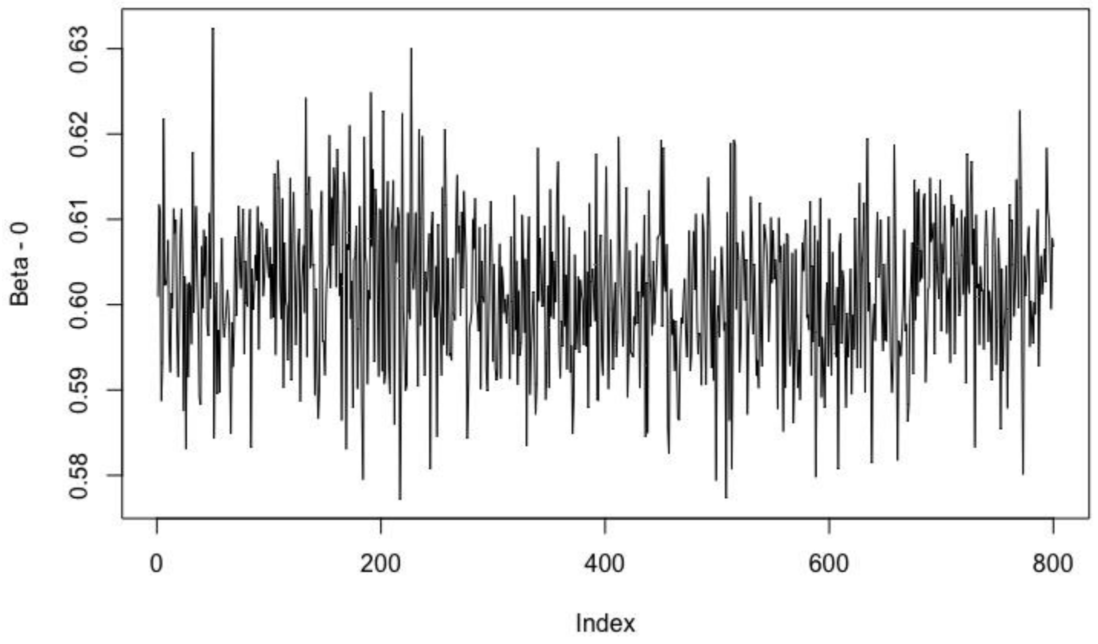
The code:



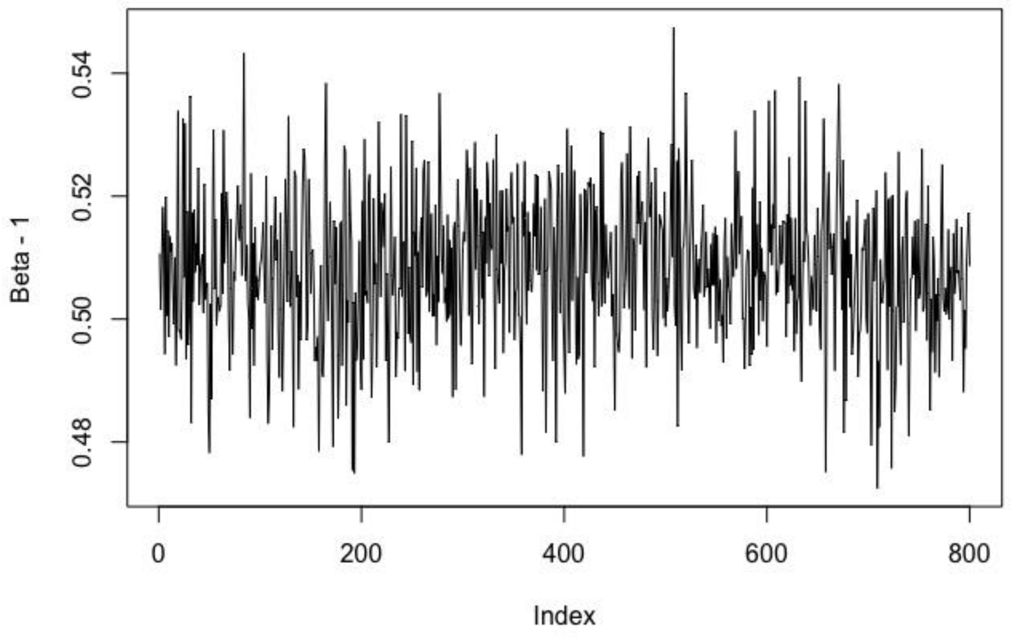
```
#main loop
for (m in 1:NIT){

  curYLC = rtruncnorm(nRC,b=0, mean = sow.XLC%*%curBeta, sd = sqrt(1/curTau))

  curYRC = rtruncnorm(nLC,a=1, mean = sow.XRC%*%curBeta, sd = sqrt(1/curTau))

  #step 2 sample beta
  #step 2.a impute the latent variables
  sow.Y = sow.data$SOW
  sow.Y[sow.data$Cens0==1] = curYRC
  sow.Y[sow.data$Cens1==1] = curYLC

  #step 2.b sample beta
  sigma.hat = solve(curTau*sow.X2 + iSigma.beta)
  betaPos.mn = sigma.hat%*%(curTau*t(sow.X)%*%sow.Y + iSigma.beta%*%mu.beta)
  curBeta = as.vector(rmnorm(1, mean=betaPos.mn, varcov=sigma.hat))

  #step 3 sample tau (precision = 1/sigma^2)
  sowE.hat = sow.Y-sow.X%*%curBeta
  curTau = rgamma(1, 0.5*nObs+a.tau, 0.5*t(sowE.hat)%*%sowE.hat+b.tau)

  #save thinned samples after burn-ins
  if ((m > nBurn) & (m%%thin.step == 0)) {
    beta.pos[g,] = curBeta
    tau.pos[g] = curTau
    g = g+1
  }
}
```

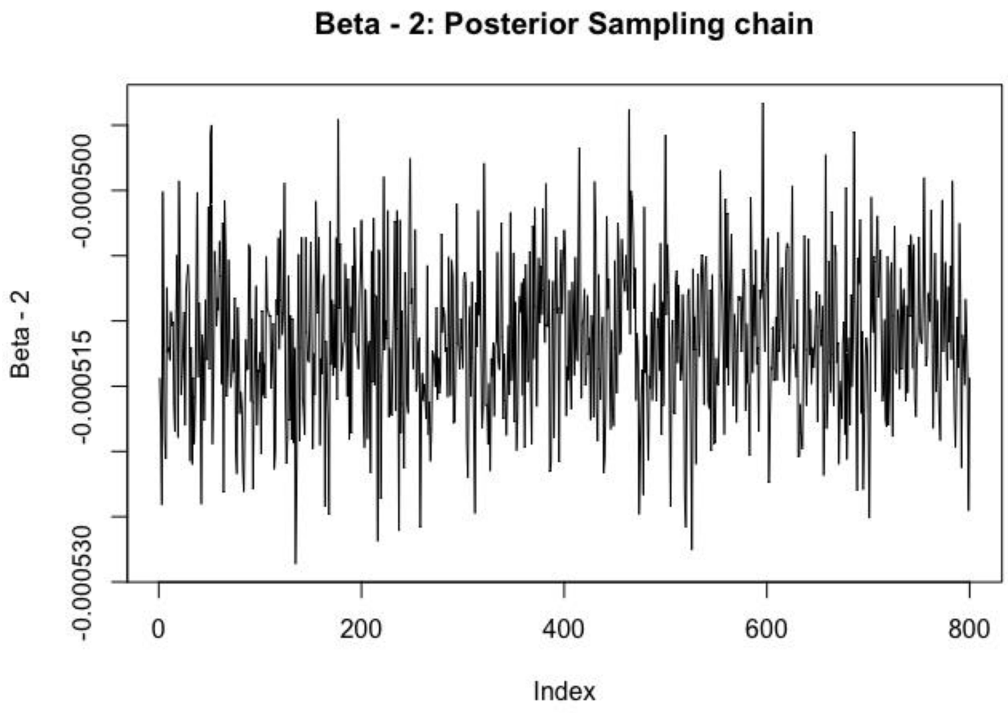
Posterior sampling chains & histograms for beta & tau:

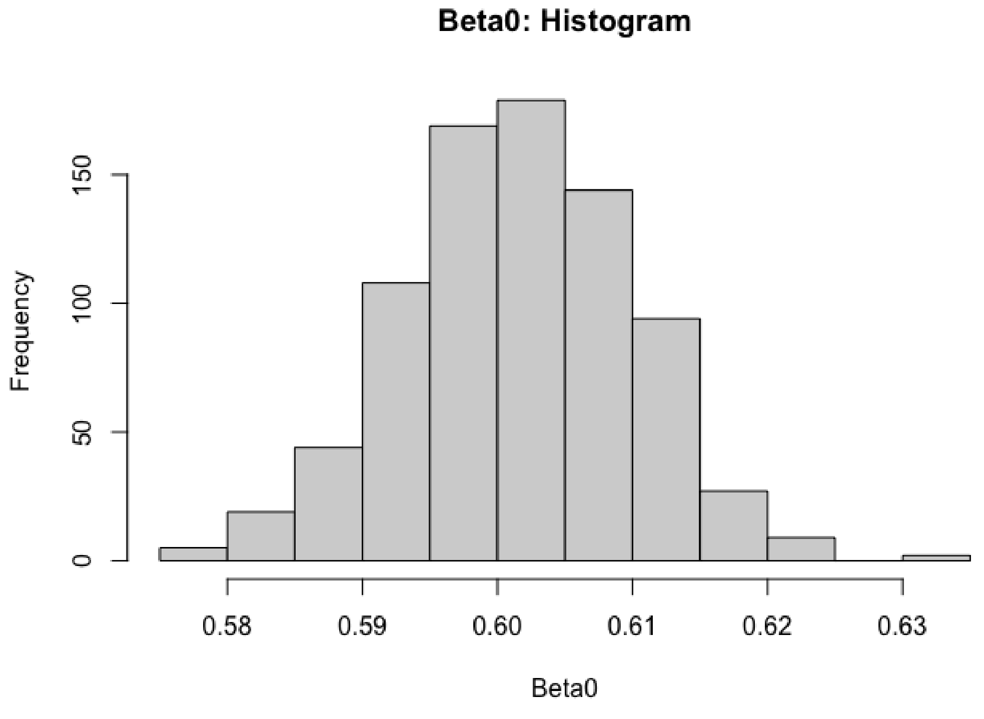
Beta - 0: Posterior Sampling chain

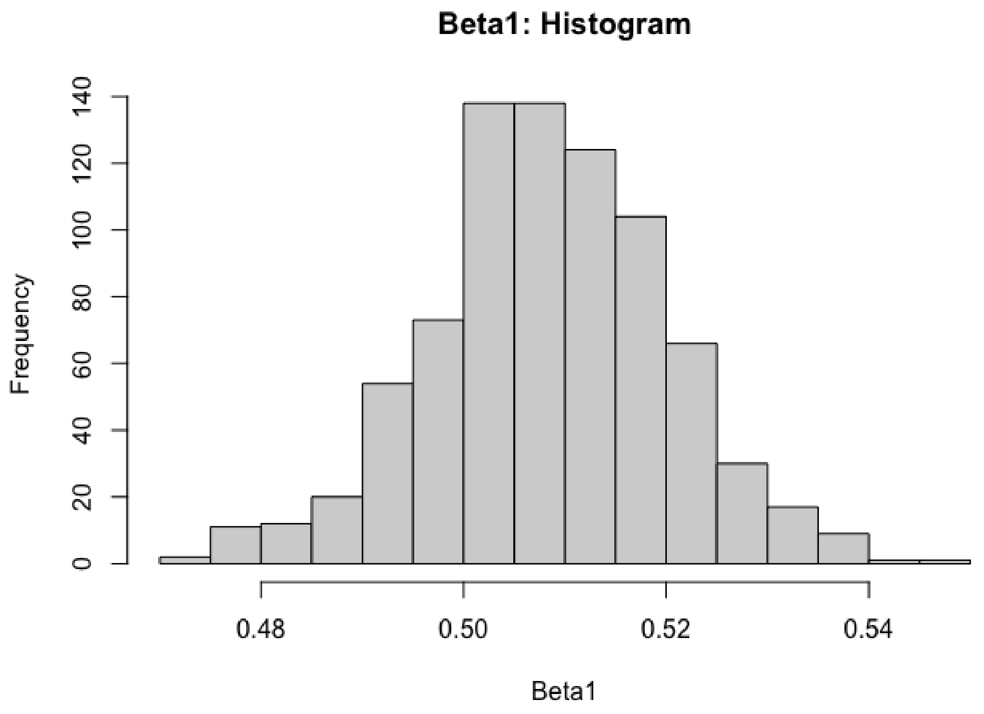


Beta - 1: Posterior Sampling chain

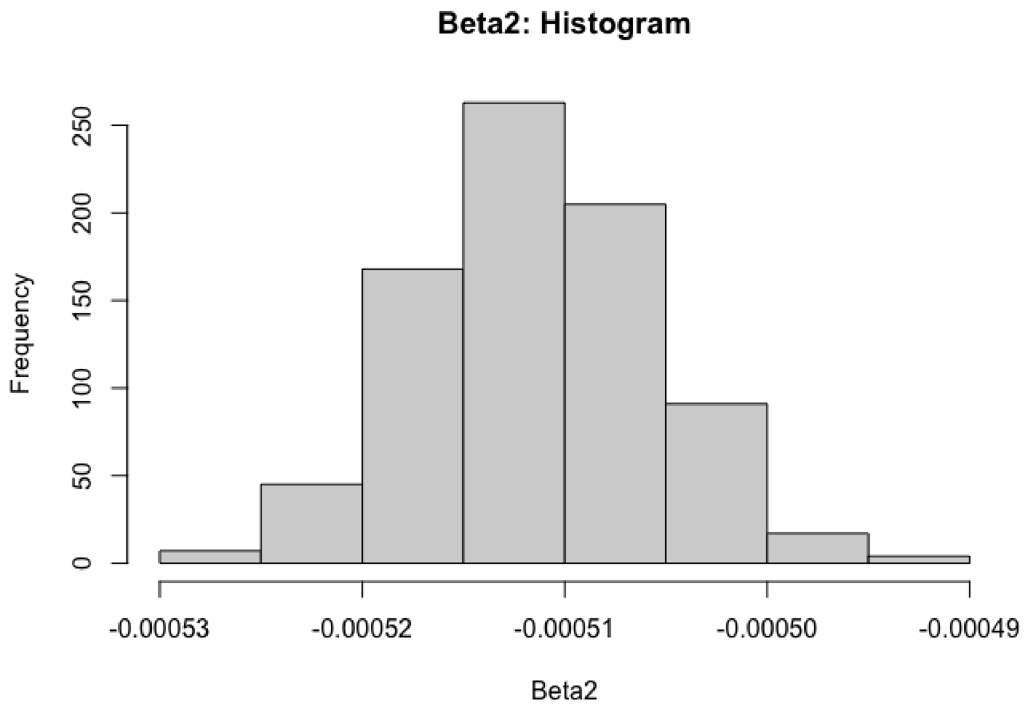



Beta - 2: Posterior Sampling chain
Beta - 2
-0.000500
-0.000515
-0.000530
0
200
400
600
800
Index

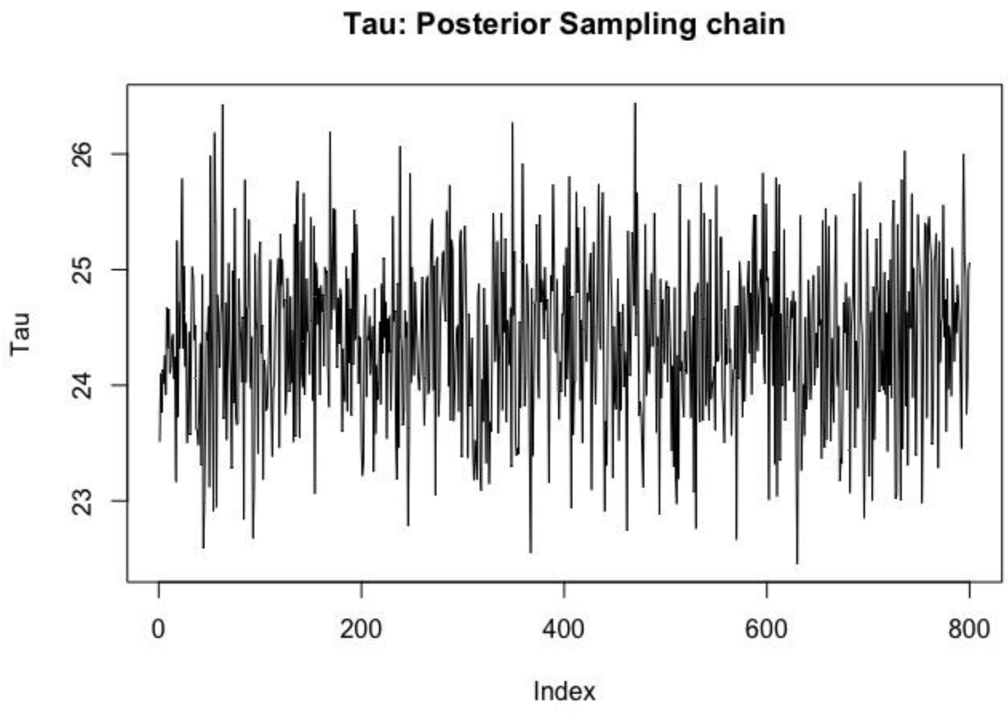

Beta0: Histogram
Frequency
0
50
100
150
0.58
0.59
0.60
0.61
0.62
0.63
Beta0

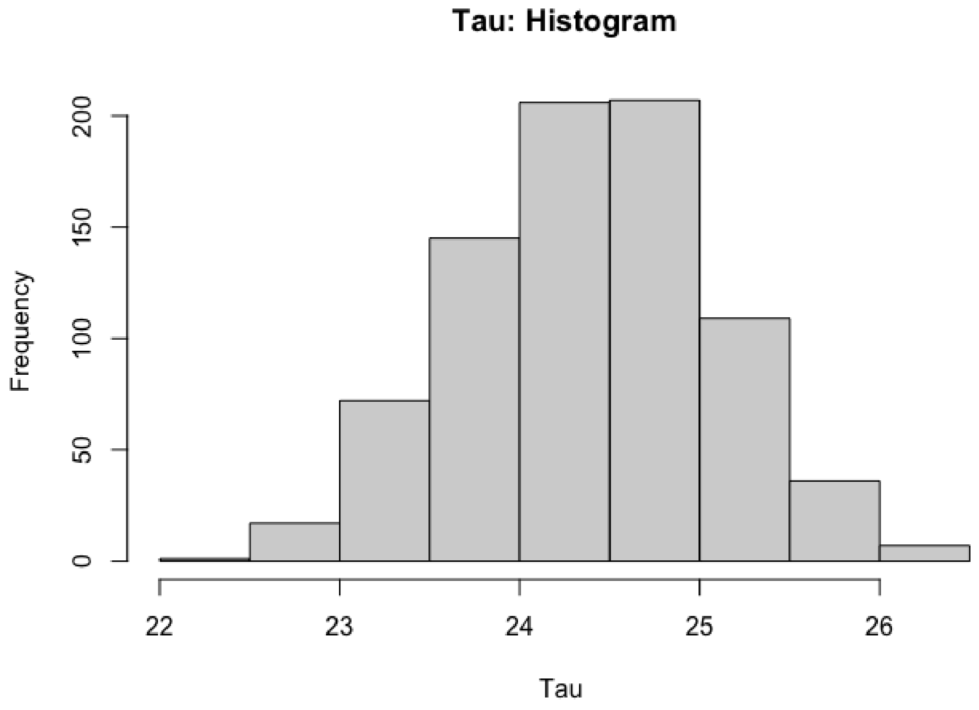


Beta1: Histogram
Frequency
0
20
40
60
80
100
120
140
0.48
0.50
0.52
0.54
Beta1

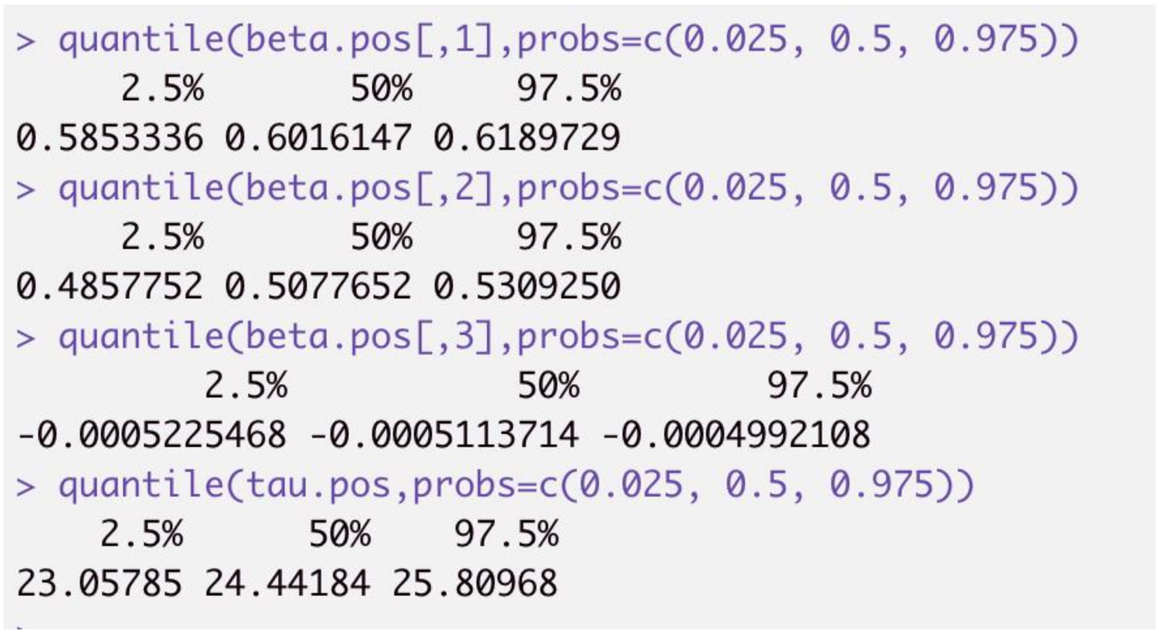


Beta2: Histogram
Frequency
0
50
100
150
200
250
-0.00053
-0.00052
-0.00051
-0.00050
-0.00049
Beta2





Quantile scores for checking significance at 95% interval:



```
> quantile(beta.pos[,1],probs=c(0.025, 0.5, 0.975))
     2.5%       50%     97.5% 
0.5853336 0.6016147 0.6189729 
> quantile(beta.pos[,2],probs=c(0.025, 0.5, 0.975))
     2.5%       50%     97.5% 
0.4857752 0.5077652 0.5309250 
> quantile(beta.pos[,3],probs=c(0.025, 0.5, 0.975))
         2.5%           50%         97.5% 
-0.0005225468 -0.0005113714 -0.0004992108 
> quantile(tau.pos,probs=c(0.025, 0.5, 0.975))
    2.5%      50%    97.5% 
23.05785 24.44184 25.80968 
```

Coefficients are significant at 95% alpha.