# Assignment 3

**2. Read the dataset into R. Check if there are missing values (NA) and, in case there are, remove them.**

```
> #Reading the csv File
> wine<- read.csv(("winequality-red.csv"))
>
> #Looking for Null or Missing values in data
> which(is.na(wine))
integer(0)
```

The dataset does not contain any missing values (NA).

**3. We want to implement a logistic regression; therefore, we want a response variable which assume values either 0 or 1. Suppose we consider "good" a wine with quality above 6.5 (included).**

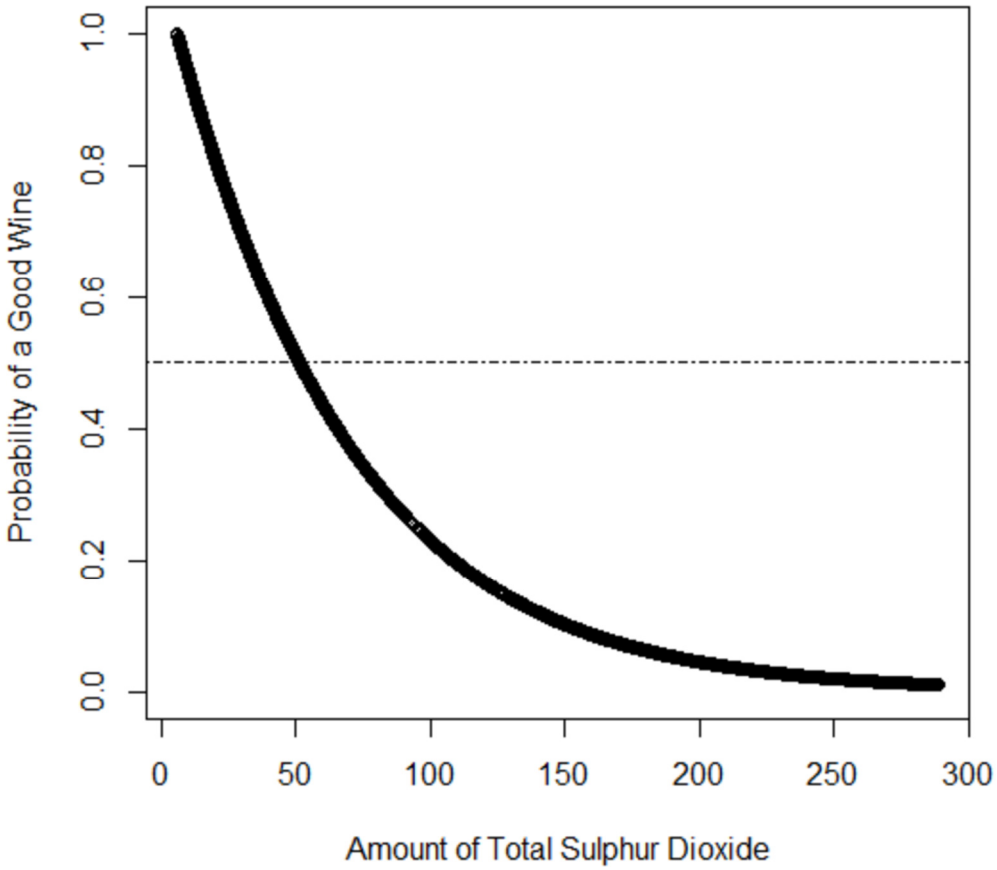
```
> wine['logqua'] = ifelse(wine$quality >= 6.5 ,1,0)
```

**4. Run a frequentist analysis on the logistic model, using the glm() function. What are the significant coefficients?**

```
> mmm<- glm(wine$logqua ~ wine$fixed.acidity + wine$volatile.acidity +  wine$citric.acid + wine$residual.sugar + wine$chlorides + wine$free.sulfur.dioxide + wine$to
tal.sulfur.dioxide + wine$density + wine$pH + wine$sulphates + wine$alcohol, data = wine, family = binomial(link = "logit"))
> mmm$coefficients
             (Intercept)        wine$fixed.acidity     wine$volatile.acidity          wine$citric.acid       wine$residual.sugar            wine$chlorides
            242.76251933               0.27495289              -2.58100211               0.56779433               0.23946420              -8.81636544
 wine$free.sulfur.dioxide wine$total.sulfur.dioxide             wine$density                  wine$pH           wine$sulphates             wine$alcohol
              0.01082060              -0.01653061            -257.79757874               0.22418522               3.74987886               0.75333905
```

**5. Estimate the probabilities of having a "success": fix each covariate at its mean level, and compute the probabilities for a wine to score "good" varying total.sulfur.dioxide, and plot the results.**

```
> tsulr<- seq(from = min(wine$total.sulfur.dioxide), to = max(wine$total.sulfur.dioxide), by = 0.1 )
> newmod<- mmm$coefficients[1] + mmm$coefficients[2]*mean(wine$fixed.acidity) + mmm$coefficients[3]*mean(wine$volatile.acidity) + mmm$coefficients[4]*mean(wine$citr
ic.acid) + mmm$coefficients[5]*mean(wine$residual.sugar) + mmm$coefficients[6]*mean(wine$chlorides) + mmm$coefficients[7]*mean(wine$free.sulfur.dioxide) + mmm$coeff
icients[8]*tsulr + mmm$coefficients[9]*mean(wine$density) + mmm$coefficients[10]*mean(wine$pH) + mmm$coefficients[11]*mean(wine$sulphates)+ mmm$coefficients[12]*mea
n(wine$alcohol)
> newmodp<- exp(newmod)/(1 + exp(newmod))
> plot(tsulr,newmodp/max(newmodp), ylim = c(0,1), xlab = "Amount of Total Sulphur Dioxide", ylab = "Probability of a Good Wine")
> abline(h = 0.5, lty = 10)
```




**6. Perform a Bayesian analysis of the logistic model for the dataset, i.e. approximate the posterior distributions of the regression coefficients, following these steps:**

- **Write an R function for the log posterior distribution.**

```
> # Log-posterior distribution
> logpost <- function(beta,x,y)
+ {
+   asd <- as.numeric(x %*% beta)
+   loga <- asd - log(1+exp(asd))
+   logb <- log(1-exp(loga))
+   logl <- sum(loga[y==1]) + sum(logb[y==0])
+   lprior <- sum(dnorm(beta,0,10,log=T))
+   return(logl + lprior)
+ }
```

- **Fix the number of simulations at $10^4$.**

```
> S <- 10^4
```

- **Choose 4 different initialisations for the coefficients.**

  The following 4 different initialisations for the coefficients were chosen.

  1. coefficients x 0.2
  2. coefficients x 0.7
  3. coefficients x 1.2
  4. coefficients x 1.7

```
> S <- 10^4
> X1=cbind(rep(1,nrow(wine)),wine$fixed.acidity,wine$volatile.acidity,wine$citric.acid,wine$residual.sugar,wine$chlorides,wine$free.sulfur.dioxide,wine$total.sulfu
r.dioxide,wine$density,wine$sulphates,wine$pH,wine$alcohol)
> X = X1[1,]
> X <- rbind(X,X1)
> y =wine$logqua[1]
> y <- c(y,wine$logqua)
```

- **For each initialisation, run a Metropolis–Hastings algorithm.**

  **1st initialisation**

```
> #First initialisation *0.2
> d1 <- mmm$coefficients*0.2
> asd_mat1 <- matrix(NA,nrow=S,ncol=ncol(X))
> asd_mat1[1,] <- as.numeric(d1)
> sigma <- solve(t(X) %*% X)
> k <- ncol(asd_mat1)
> acc <- 1
> for(iter in 2:S)
+ {
+   bt1 <- rmvnorm(1,asd_mat1[iter-1,],sigma)
+   newpost=logpost(t(bt1),X,y)
+   oldpost=logpost(matrix(asd_mat1[iter-1,],ncol=1),X,y)
+
+   if(runif(1,0,1)>exp(newpost-oldpost)){
+     asd_mat1[iter,]=asd_mat1[iter-1,]
+   } else{
+     asd_mat1[iter,]=bt1
+     acc=acc+1
+   }
+
+   if(iter%%1000==0){print(c(iter,acc/iter))}}
[1] 1000.000    0.269
[1] 2000.0000    0.2405
[1] 3000.0000000    0.2263333
[1] 4000.000    0.218
[1] 5000.0000    0.2126
[1] 6000.0000000    0.2098333
[1] 7000.0000000    0.2111429
[1] 8000.000    0.208
[1] 9000.0000000    0.2067778
[1] 1.000e+04 2.049e-01
```

**2nd initialisation**

```
> #Second initialisation *0.7
> d2 <- (mmm$coefficients*0.7)
> asd_mat2 <- matrix(NA,nrow=S,ncol=ncol(X))
> asd_mat2[1,] <- as.numeric(d2)
> sigma <- solve(t(X) %*% X)
> k <- ncol(asd_mat2)
> acc <- 1
> for(iter in 2:S)
+ {
+   bt2 <- rmvnorm(1,asd_mat2[iter-1,],sigma)
+   newpost=logpost(t(bt2),X,y)
+   oldpost=logpost(matrix(asd_mat2[iter-1,],ncol=1),X,y)
+
+   if(runif(1,0,1)>exp(newpost-oldpost)){
+     asd_mat2[iter,]=asd_mat2[iter-1,]
+   } else{
+     asd_mat2[iter,]=bt2
+     acc=acc+1
+   }
+
+   if(iter%%1000==0){print(c(iter,acc/iter))}}
[1] 1000.000    0.333
[1] 2000.000    0.268
[1] 3000.000    0.242
[1] 4000.00000    0.22975
[1] 5000.000    0.225
[1] 6000.0000000    0.2206667
[1] 7000.0000000    0.2161429
[1] 8000.000000    0.212375
[1] 9000.0000000    0.2115556
[1] 1.000e+04 2.097e-01
```

**3rd initialisation**

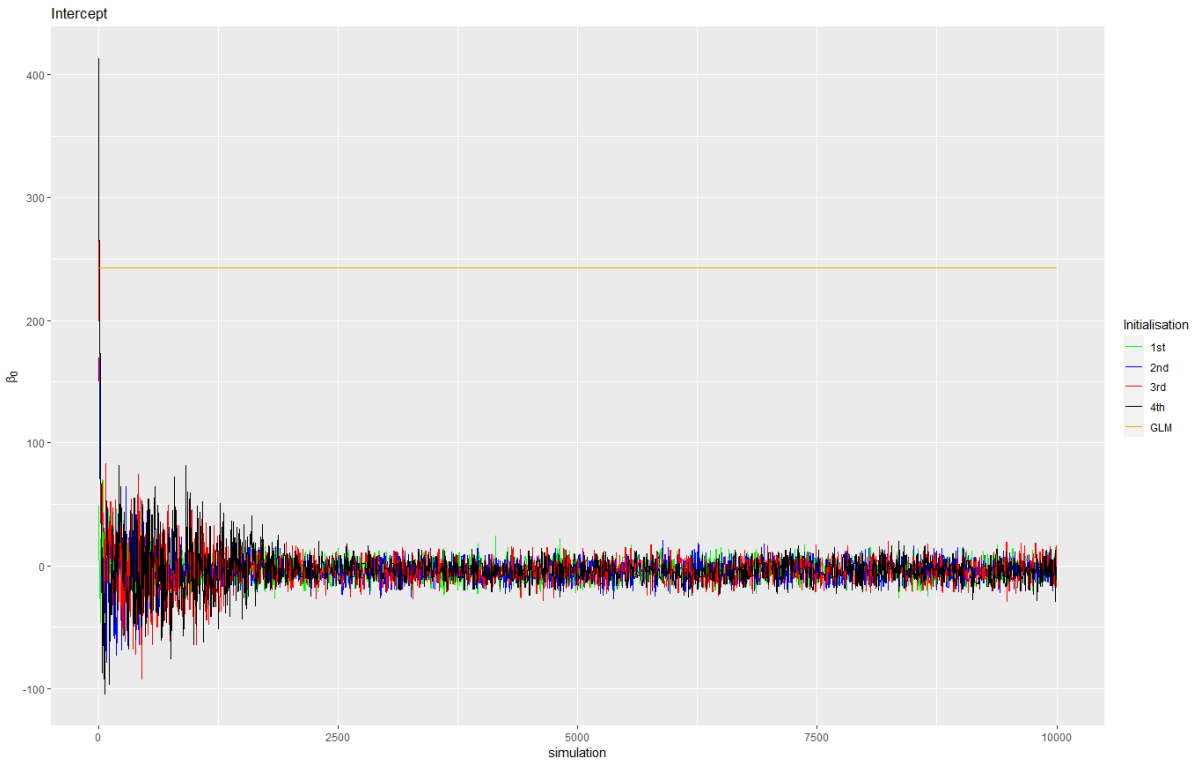
```
>   #Third initialisation *1.2
>   d3 <- (mmm$coefficients*1.2)
>   asd_mat3 <- matrix(NA,nrow=S,ncol=ncol(X))
>   asd_mat3[1,] <- as.numeric(d3)
>   sigma <- solve(t(X) %*% X)
>   k <- ncol(asd_mat3)
>   acc <- 1
>   for(iter in 2:S)
+   {
+     bt3 <- rmvnorm(1,asd_mat3[iter-1,],sigma)
+     newpost=logpost(t(bt3),X,y)
+     oldpost=logpost(matrix(asd_mat3[iter-1,],ncol=1),X,y)
+
+     if(runif(1,0,1)>exp(newpost-oldpost)){
+       asd_mat3[iter,]=asd_mat3[iter-1,]
+     } else{
+       asd_mat3[iter,]=bt3
+       acc=acc+1
+     }
+
+     if(iter%%1000==0){print(c(iter,acc/iter))}}
[1] 1000.000    0.387
[1] 2000.0000    0.3155
[1] 3000.0000000    0.2776667
[1] 4000.00000    0.26225
[1] 5000.0000    0.2498
[1] 6000.000    0.245
[1] 7000.0000000    0.2407143
[1] 8000.000    0.235
[1] 9000.0000000    0.2328889
[1] 1.00e+04 2.33e-01
```

**4th initialisation**

```
>   #Fourth initialisation *1.7
>   d4 <- (mmm$coefficients*1.7)
>   asd_mat4 <- matrix(NA,nrow=S,ncol=ncol(X))
>   asd_mat4[1,] <- as.numeric(d4)
>   sigma <- solve(t(X) %*% X)
>   k <- ncol(asd_mat4)
>   acc <- 1
>   for(iter in 2:S)
+   {
+     bt4 <- rmvnorm(1,asd_mat4[iter-1,],sigma)
+     newpost=logpost(t(bt4),X,y)
+     oldpost=logpost(matrix(asd_mat4[iter-1,],ncol=1),X,y)
+
+     if(runif(1,0,1)>exp(newpost-oldpost)){
+       asd_mat4[iter,]=asd_mat4[iter-1,]
+     } else{
+       asd_mat4[iter,]=bt4
+       acc=acc+1
+     }
+
+     if(iter%%1000==0){print(c(iter,acc/iter))}}
[1] 1000.000    0.428
[1] 2000.0000    0.3775
[1] 3000.0000000    0.3186667
[1] 4000.0000    0.2865
[1] 5000.0000    0.2692
[1] 6000.0000    0.2545
[1] 7000.0000000    0.2441429
[1] 8000.000    0.238
[1] 9000.0000000    0.2354444
[1] 1.00e+04 2.33e-01
```

- **Plot the chains for each coefficient (the 4 chains on the same plot) and comment.**
  The table after every graph contains values for the coefficients generated by the four initialisations.

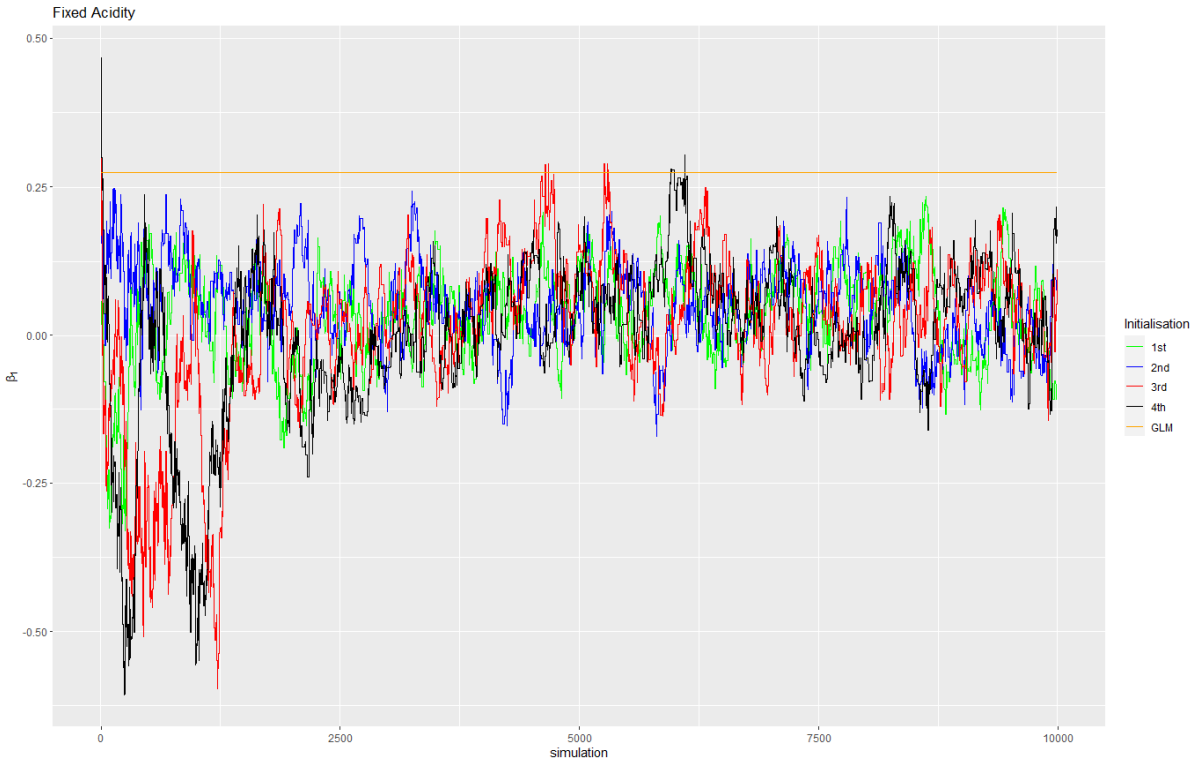
**Intercept**



| 1st initialisation | 2nd initialisation | 3rd initialisation | 4th initialisation | Frequentist |
|---|---|---|---|---|
| -6.808631 | 5.939533 | 1.942699 | -5.492696 | 242.762519 |

**Comment -** All the 4 initialisation chains converge after about 1500 simulations to similar values. These values do not converge at the MLE as well as show at a large difference with value of intercept generated by the GLM model. This is due the fact this model is very sensitive to tuning parameters.

**Fixed Acidity**




| 1st initialisation | 2nd initialisation | 3rd initialisation | 4th initialisation | Frequentist |
|---|---|---|---|---|
| 0.15402250 | 0.14897802 | 0.03447363 | 0.14404377 | 0.27495289 |

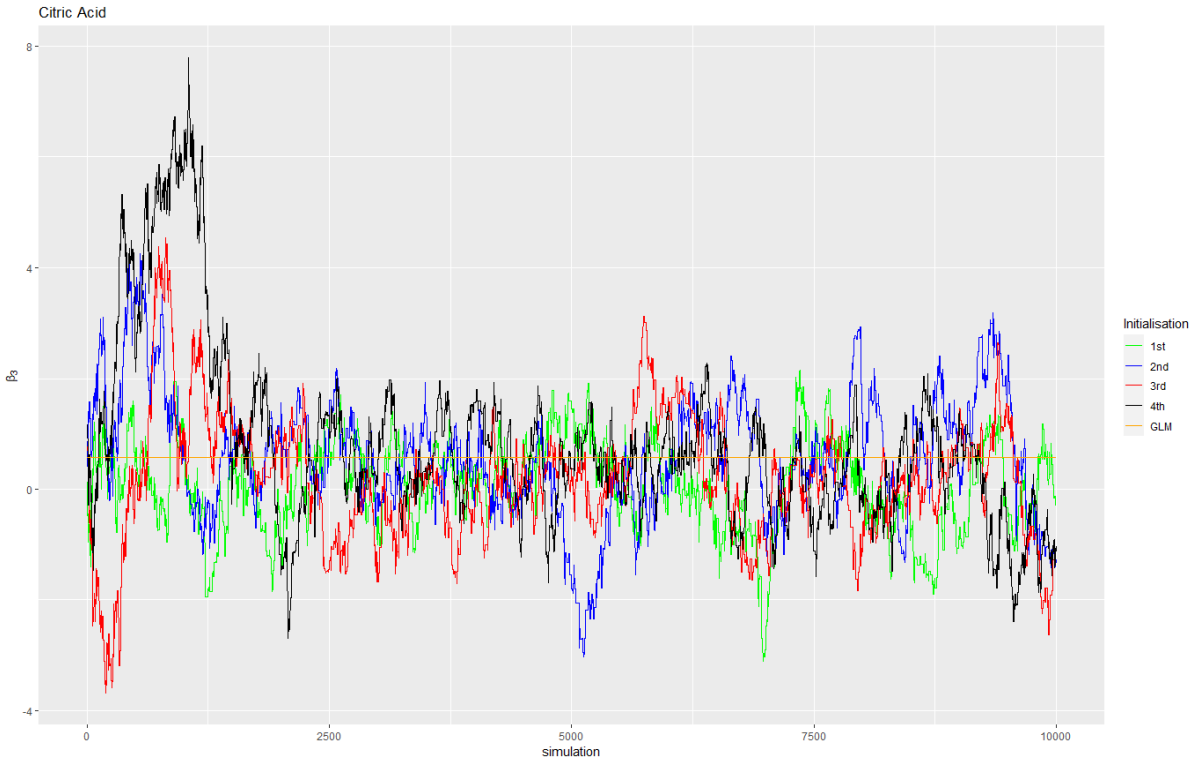
**Comment –** All the 4 initialisation chains converge after about 1250 simulations to an approximated value of 0.14. This value is half the value of the coefficient generated by the GLM model.

## Volatile Acidity


| 1st initialisation | 2nd initialisation | 3rd initialisation | 4th initialisation | Frequentist |
|---|---|---|---|---|
| -3.162759 | -4.087140 | -2.795408 | -4.437245 | -2.581002 |

**Comment –** All the 4 initialisation chains converge after about 2500 simulations to an approximated value less than the value of the coefficient generated by the GLM model.

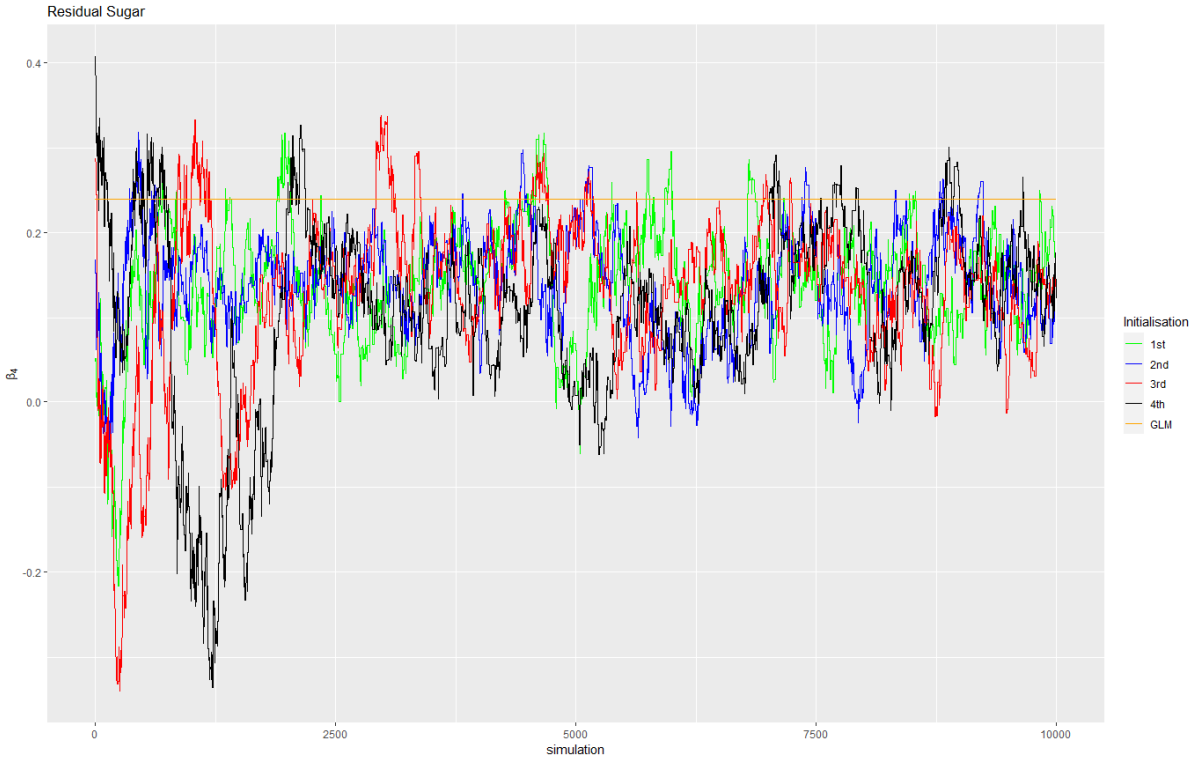
## Citric Acid




| 1st initialisation | 2nd initialisation | 3rd initialisation | 4th initialisation | Frequentist |
|---|---|---|---|---|
| -0.2902753 | -1.2835568 | -1.3524843 | -1.0363209 | 0.5677943 |

**Comment –** All the 4 initialisation chains converge after about 2500 simulations to an approximated value less than the value of the coefficient generated by the GLM model.

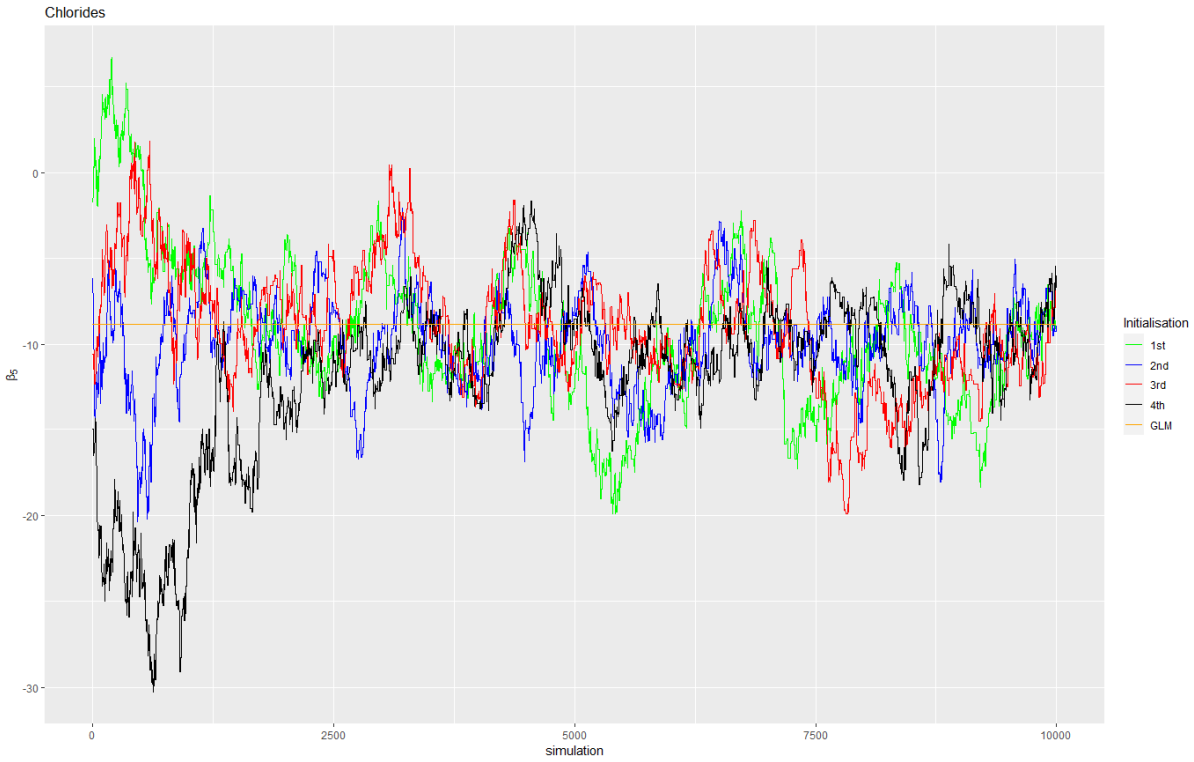
**Residual Sugar**




| 1st initialisation | 2nd initialisation | 3rd initialisation | 4th initialisation | Frequentist |
|---|---|---|---|---|
| 0.1747075 | 0.1333657 | 0.1368356 | 0.1387273 | 0.2394642 |

**Comment –** All the 4 initialisation chains converge after about 2000 simulations to an approximated value of 0.13 The first initialisation chain converges to a value of 0.17. This value is lower than the value of the coefficient generated by the GLM model.

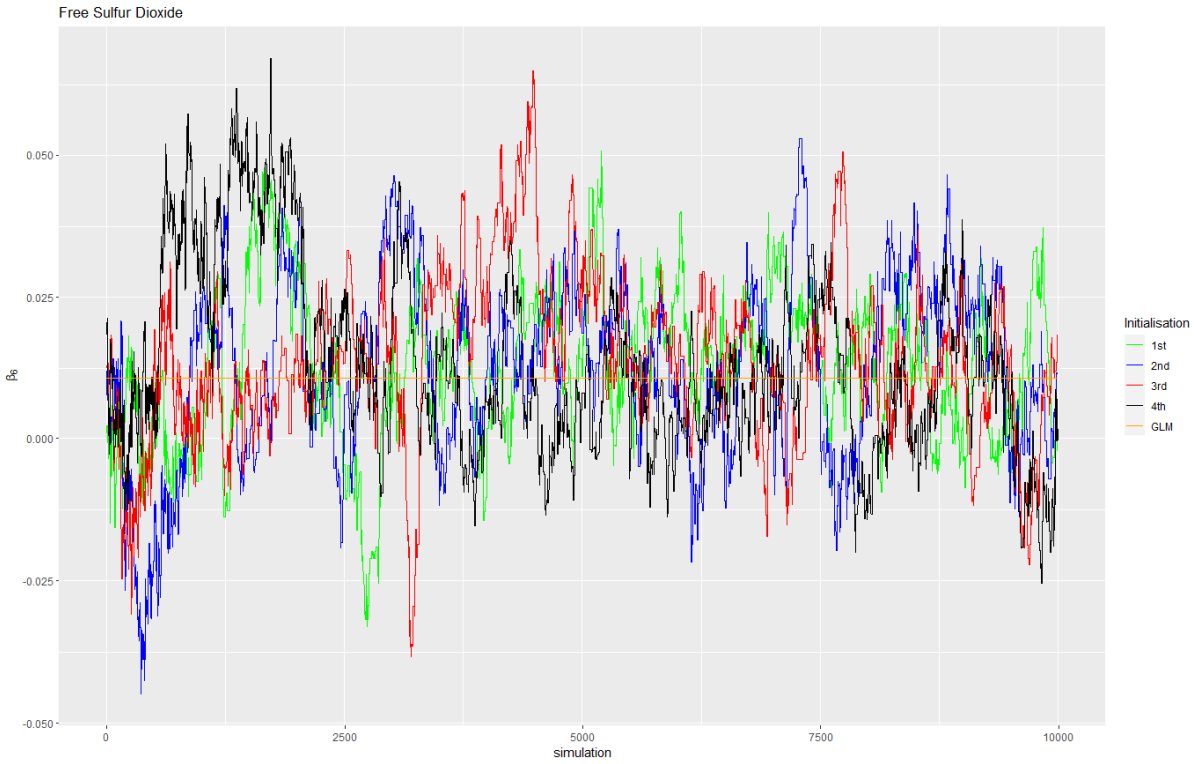
## Chlorides




| 1st initialisation | 2nd initialisation | 3rd initialisation | 4th initialisation | Frequentist |
|---|---|---|---|---|
| -8.649588 | -8.942966 | -6.726945 | -6.753570 | -8.816365 |

**Comment –** All the 4 initialisation chains converge after about 2000 simulations The first two which had higher initial values higher than coefficient value by the GLM model converges to the GLM value for the coefficient. The other two initialisations converge at a lower value.

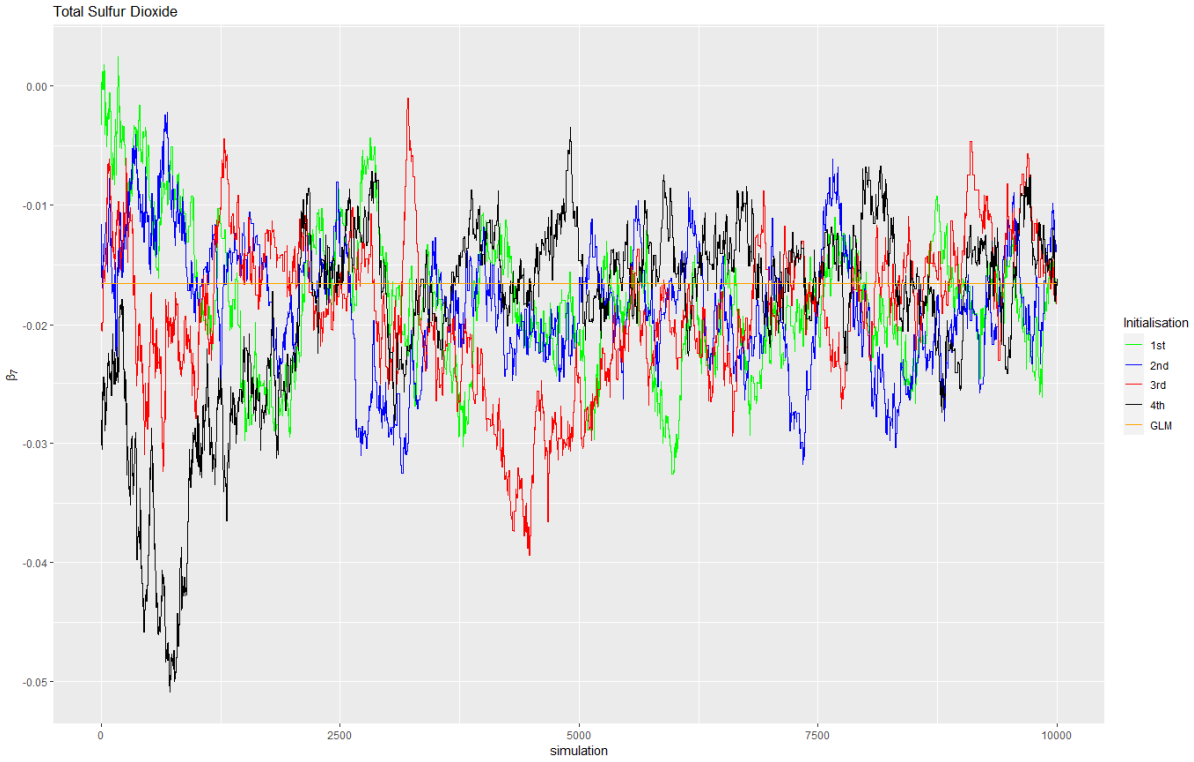
## Free Sulfur Dioxide




| 1st initialisation | 2nd initialisation | 3rd initialisation | 4th initialisation | Frequentist |
|---|---|---|---|---|
| -0.002135599 | 0.0115758628 | 0.0183495620 | -0.000467929 | 0.010820601 |

**Comment –** All the 4 initialisation chains do not converge in the given number of simulations. A higher number of simulations may be necessary for convergence. The end value of all chains is close to value of coefficient generated by the GLM model.

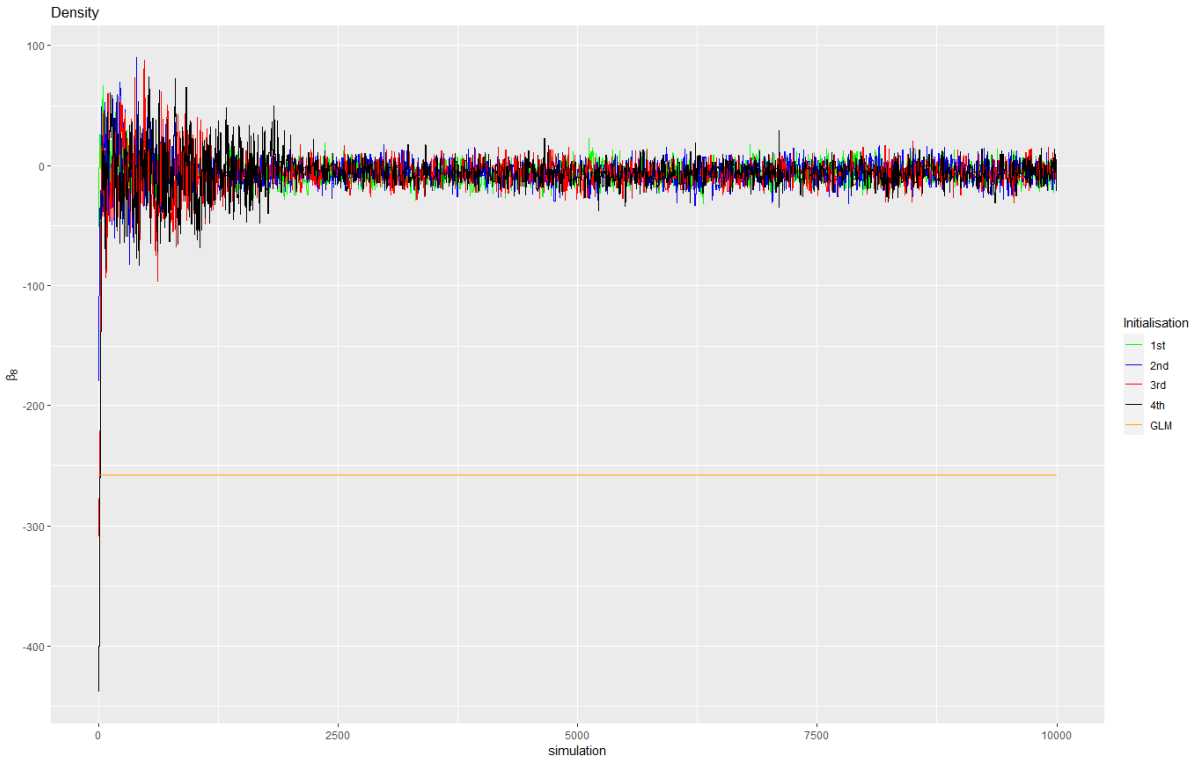
**Total Sulfur Dioxide**




| 1st initialisation | 2nd initialisation | 3rd initialisation | 4th initialisation | Frequentist |
|---|---|---|---|---|
| -0.01641886 | -0.01325836 | -0.01822679 | -0.01615706 | -0.01653061 |

**Comment –** All the 4 initialisation chains converge to a value of -0.01 after about 5000 simulations. Al the chains converge close to the value of the coefficient generated by the GLM model.

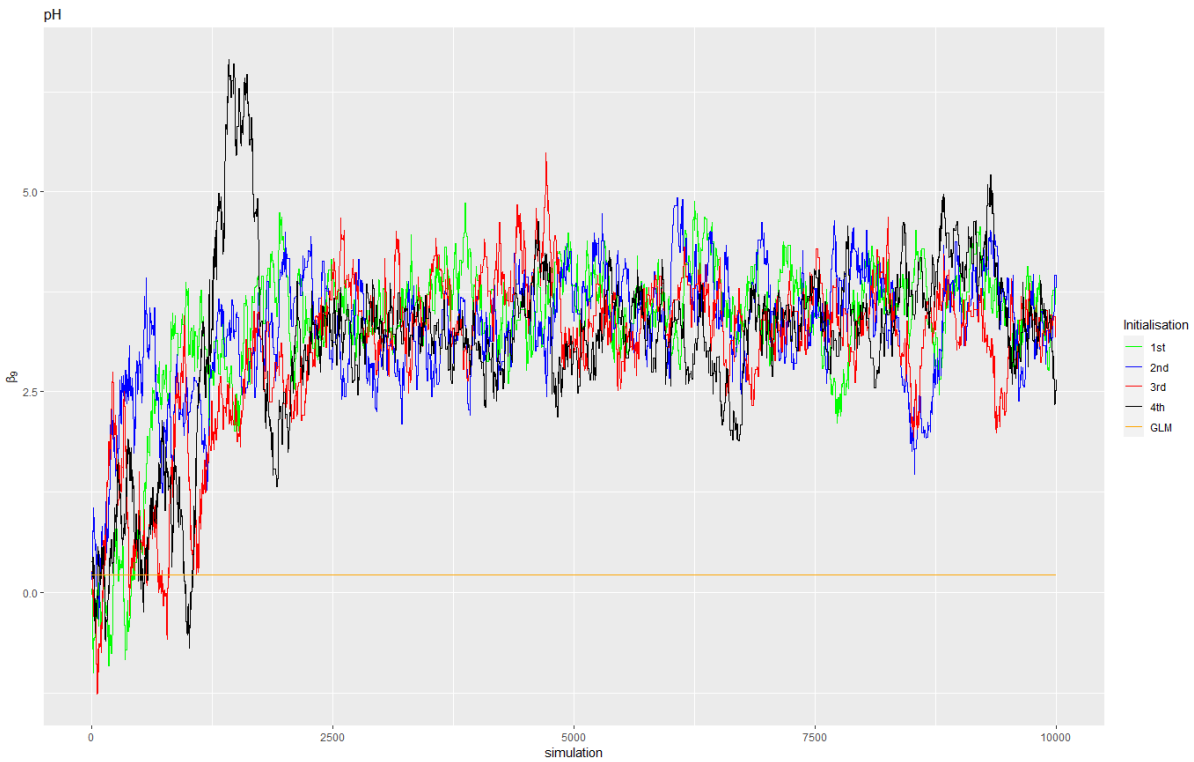
## Density




| 1st initialisation | 2nd initialisation | 3rd initialisation | 4th initialisation | Frequentist |
|---|---|---|---|---|
| -3.960263 | -15.369208 | -6.346984 | -2.623045 | -257.797579 |

**Comment –** All the 4 initialisation chains converge after about 1800 simulations to similar values. These values do not converge at the MLE as well as show at a large difference with value of coefficient for density generated by the GLM model. This is due the fact this model is very sensitive to tuning parameters.

## pH




| 1st initialisation | 2nd initialisation | 3rd initialisation | 4th initialisation | Frequentist |
|---|---|---|---|---|
| 3.8097271 | 3.8026114 | 3.1911742 | 2.6600167 | 0.2241852 |

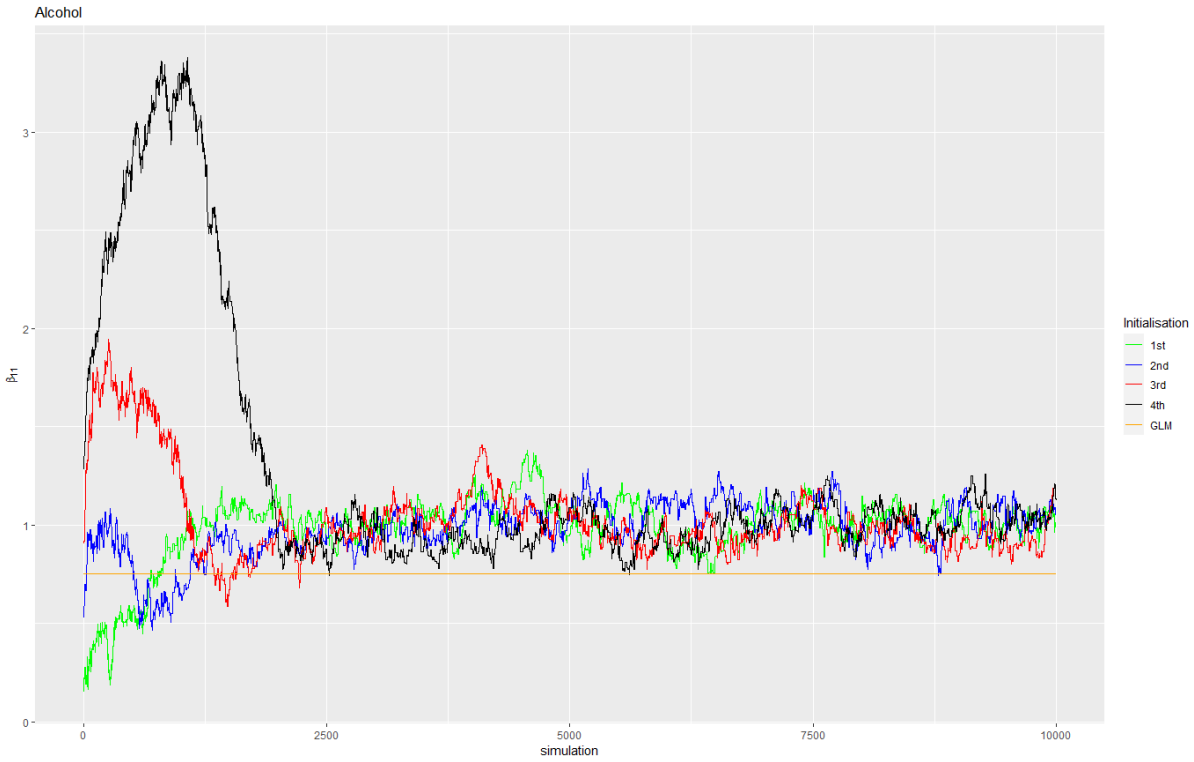
**Comment –** All the 4 initialisation chains converge after about 2000 simulations. The first three initialisation chains converge to a similar value. The fourth initialisation converges to a lower value. All the chains converge to a value higher than the value of the coefficient generated by the GLM model.

## Sulphates


| 1st initialisation | 2nd initialisation | 3rd initialisation | 4th initialisation | Frequentist |
|---|---|---|---|---|
| -1.058831 | -1.475224 | -3.225674 | -1.830579 | 3.749879 |

**Comment –** All the 4 initialisation chains converge after about 2500 simulations. The first, second and fourth initialisation chains converge to a similar value. The third initialisation converges to a lower value. All the chains converge to a value higher than the value of the coefficient generated by the GLM model.

## Alcohol



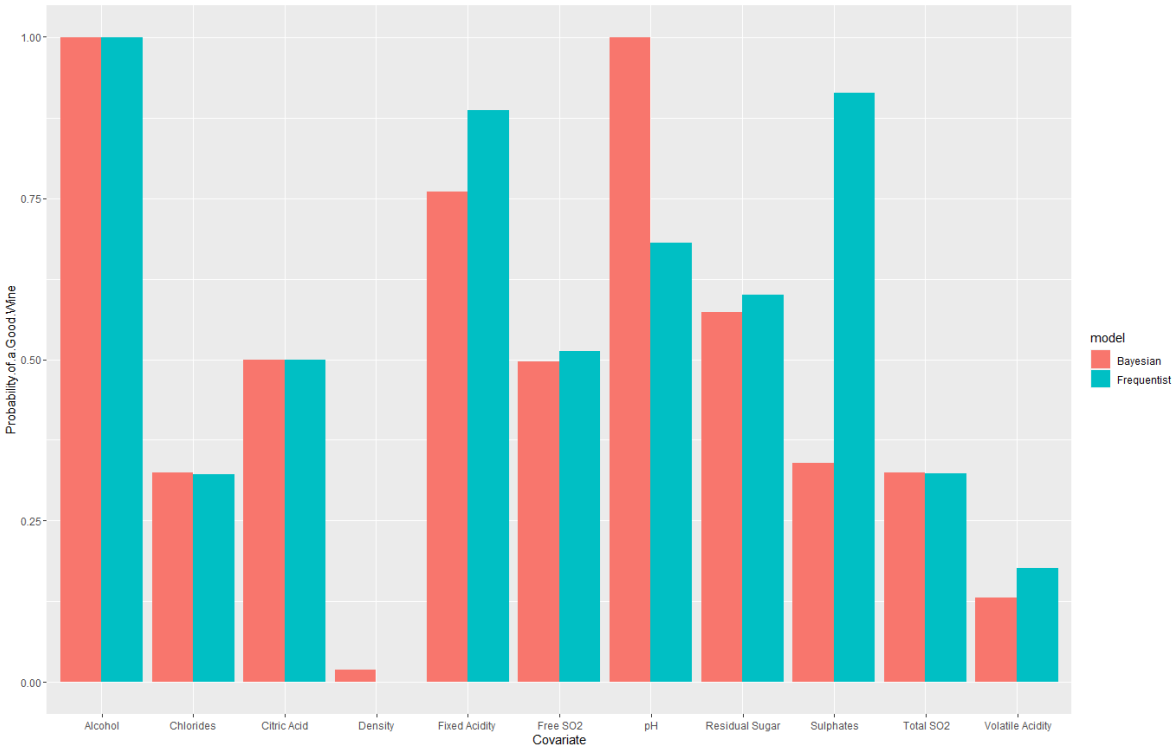

| 1st initialisation | 2nd initialisation | 3rd initialisation | 4th initialisation | Frequentist |
|---|---|---|---|---|
| 1.0097784 | 1.0538146 | 1.2053650 | 1.1215988 | 0.7533391 |

**Comment –** All the 4 initialisation chains converge after about 2200 simulations. The first third and fourth initialisation chains converge to a similar value. All the four initialisation chains converge to a similar value which is lesser than the value of coefficient generated by the GLM model.

**7. Approximate the posterior predictive distribution of an unobserved variable characterised by.**

```
> pre<- c(asd_mat1[iter,], asd_mat2[iter,], asd_mat3[iter,], asd_mat4[iter,])
>
> #Prediction
> x_new <-c(7.5,0.6,0,1.7,0.085,5,45,0.9965,3.40,0.63,12)
> p_new1 <- exp(mmm$coefficients[2:12] * x_new) / (1 + exp(mmm$coefficients[2:12] * x_new))
> p_new2 <- exp(pre[2:12] * x_new) / (1 + exp(pre[2:12] * x_new))
```

**Plot the approximate posterior predictive distribution.**

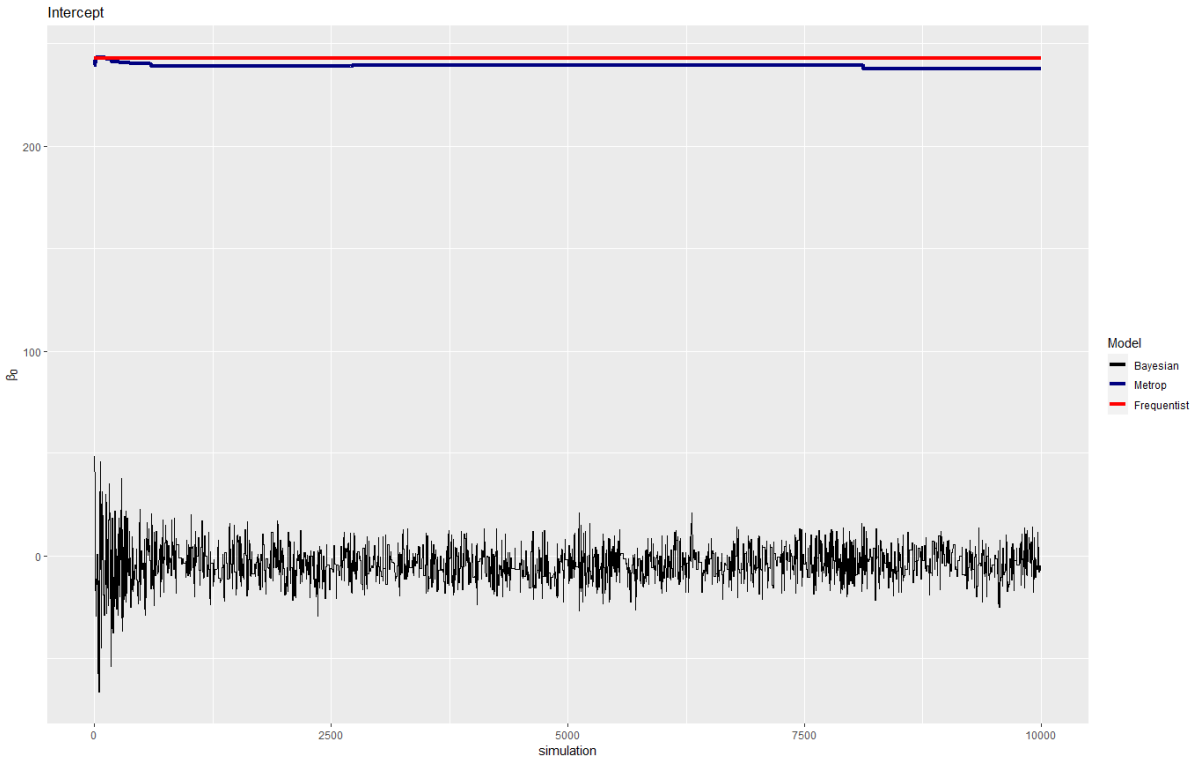



**8. Use the metrop() function available in the mcmc package to perform the same analysis on the posterior distribution you have approximated for Question 6. Choose again $10^4$ simulations and compare the results with the results obtained with your code**.

```
> #Metrop Analysis
> lat <- function(x, y) function(beta)
+ {
+   asd <- as.numeric(x %*% beta)
+   loga <- asd - log(1+exp(asd))
+   logb <- log(1-exp(loga))
+   logl <- sum(loga[y==1]) + sum(logb[y==0])
+   lprior <- sum(dnorm(beta,0,10,log=T))
+   return(logl + lprior)
+ }
> lut <- lat(X1, mmm$y)
> coff <- as.numeric(coefficients(mmm))
> zxc <- metrop(lut, coff, S,blen = 1,nspac = 10,debug = TRUE)
```

The table after every graph contains values for the coefficients generated by the three mentioned models.

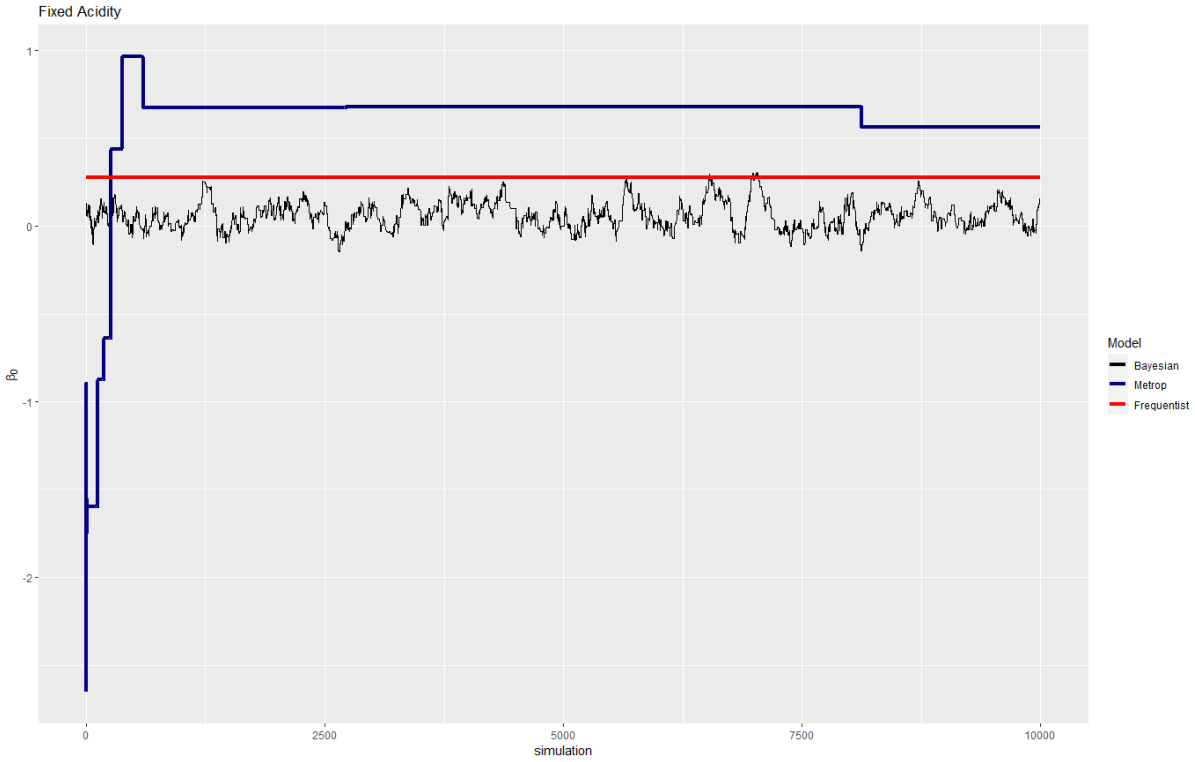
## Intercept




| Bayesian MHA | MCMC Metrop | Frequentist GLM |
|---|---|---|
| -6.808631 | 237.551494 | 242.762519 |

**Comment –** The value of coefficient generated by the metrop function and the GLM model are very similar which is significantly higher thank the value generated by the Bayesian model. These values do not converge at the MLE as well as show at a large difference with value of coefficient for density generated by the GLM model. This is due the fact this model is very sensitive to tuning parameters.

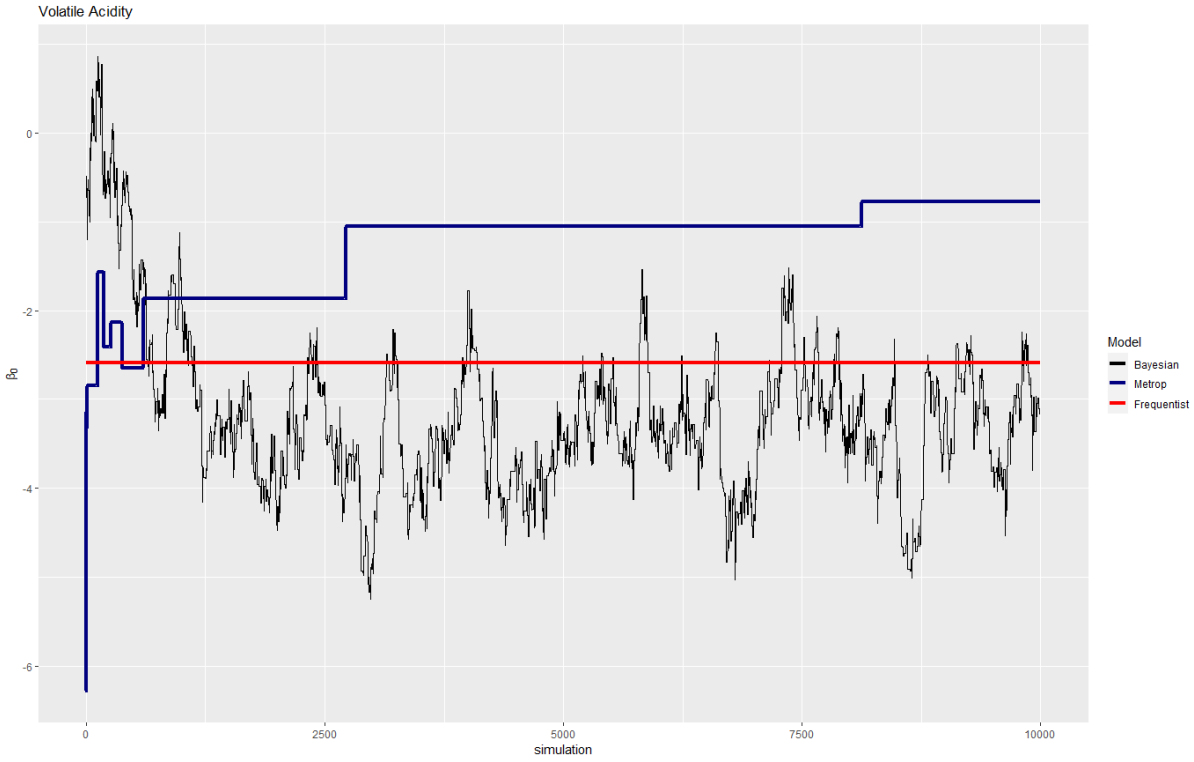
## Fixed Acidity



| Bayesian MHA | MCMC Metrop | Frequentist GLM |
|---|---|---|
| 0.1540225 | 0.5617872 | 0.2749529 |

**Comment –** The value of coefficient generated by the metrop function is higher than other two models. The Bayesian model generates the lowest value for the coefficient among the three models.

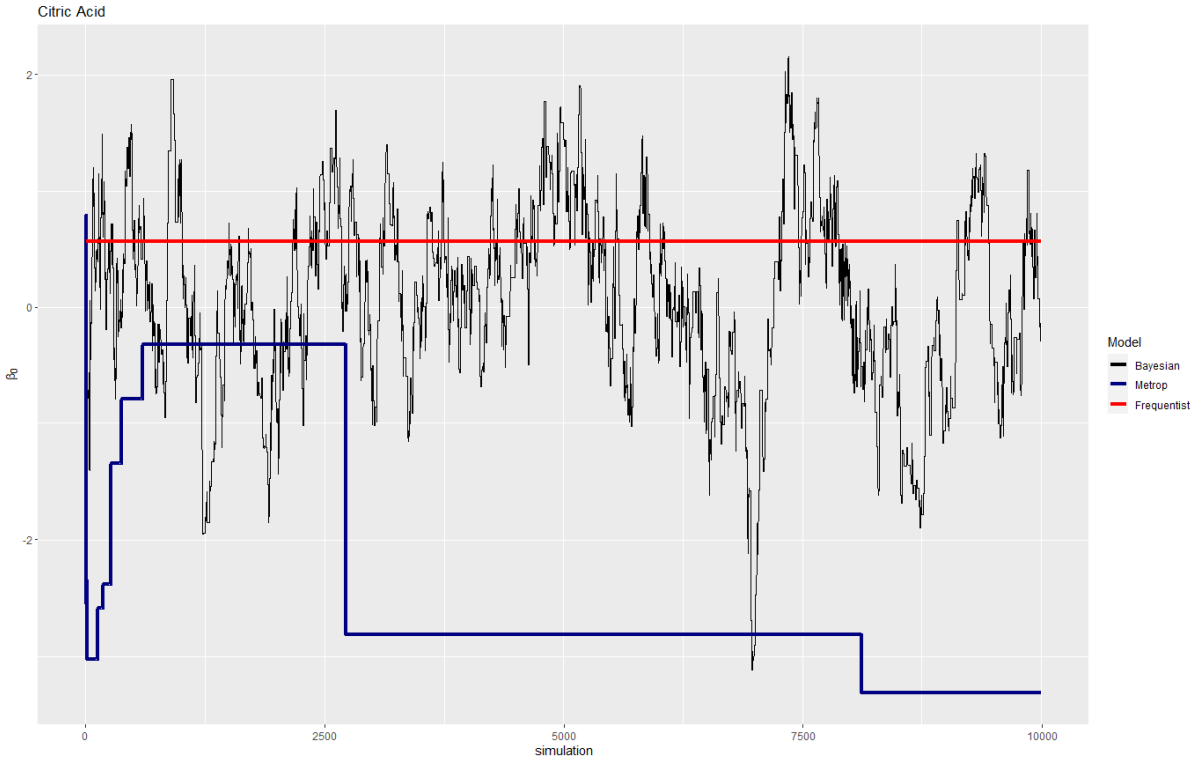
## Volatile Acidity




| Bayesian MHA | MCMC Metrop | Frequentist GLM |
|---|---|---|
| -3.1627588 | -0.7673774 | -2.5810021 |

**Comment –** The value of coefficient generated by the metrop function is higher than the other two models. The value of the coefficient generated by the GLM model is very similar to the value of the coefficient generated by the Bayesian model.

## Citric Acid




| Bayesian MHA | MCMC Metrop | Frequentist GLM |
|---|---|---|
| -0.2902753 | -3.3114508 | 0.5677943 |

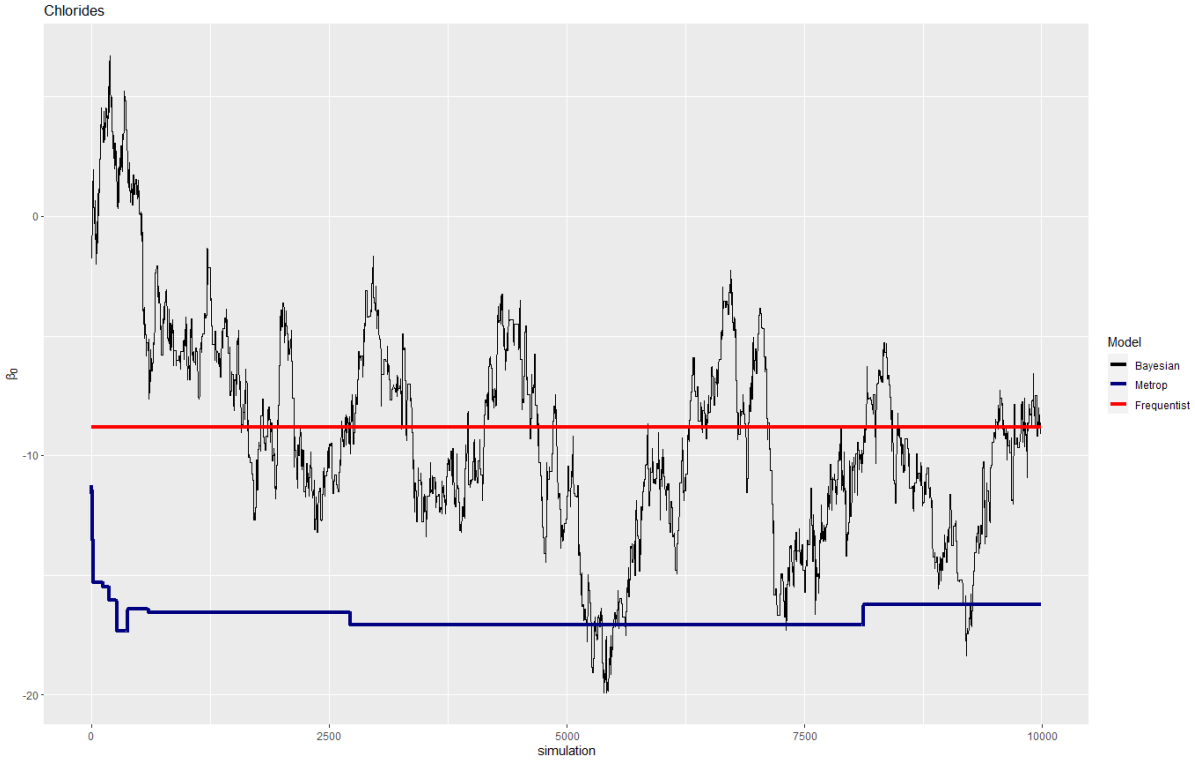
**Comment –** The value of coefficient generated by the metrop function is lower than the other two models. The value of the coefficient generated by the GLM model is closer to the value of the coefficient generated by the Bayesian model.

## Residual Sugar


| Bayesian MHA | MCMC Metrop | Frequentist GLM |
|---|---|---|
| 0.1747075 | 0.4196317 | 0.2394642 |

**Comment –** The value of coefficient generated by the metrop function oscillates between the values of coefficient generated by the other two models and is eventually higher than the other two models. The value of the coefficient generated by the GLM and Bayesian models do not differ significantly.

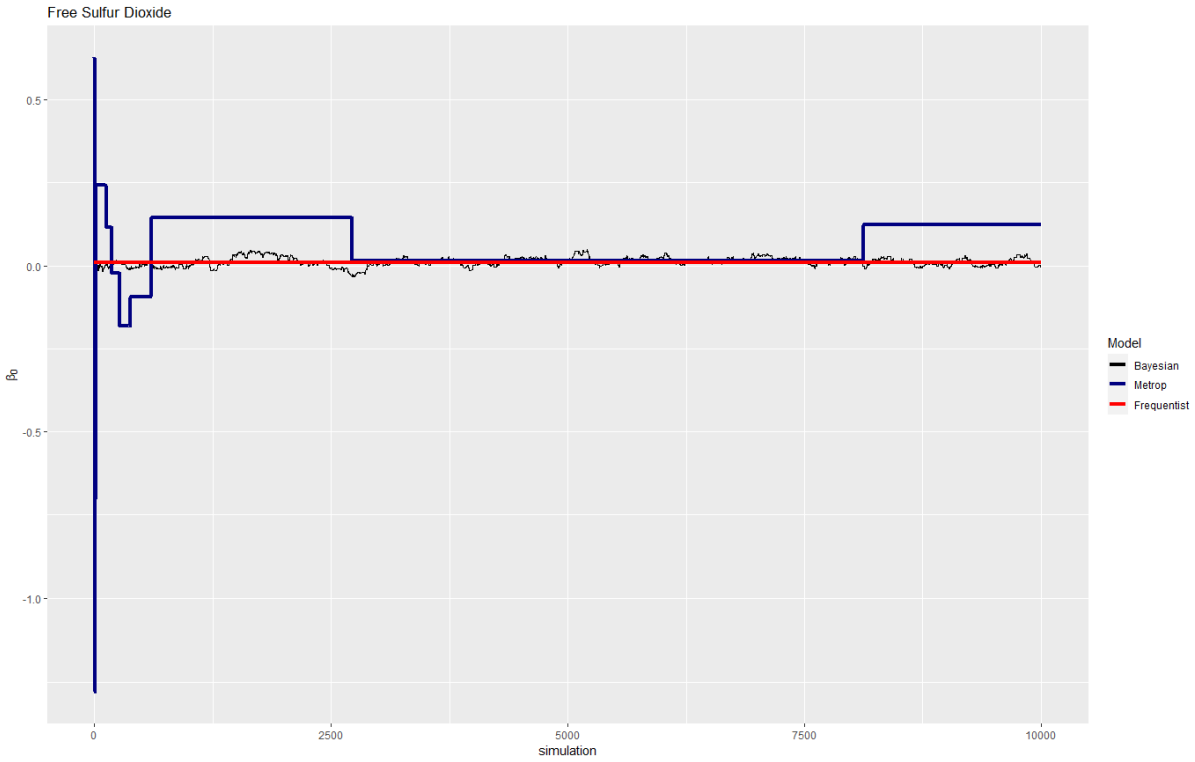
**Chlorides**




| Bayesian MHA | MCMC Metrop | Frequentist GLM |
|---|---|---|
| -8.649588 | -16.227976 | -8.816365 |

**Comment –** The value of coefficient generated by the metrop function is lower than the values of coefficient generated by the other two models. The value of the coefficient generated by the GLM and Bayesian models do not differ significantly.

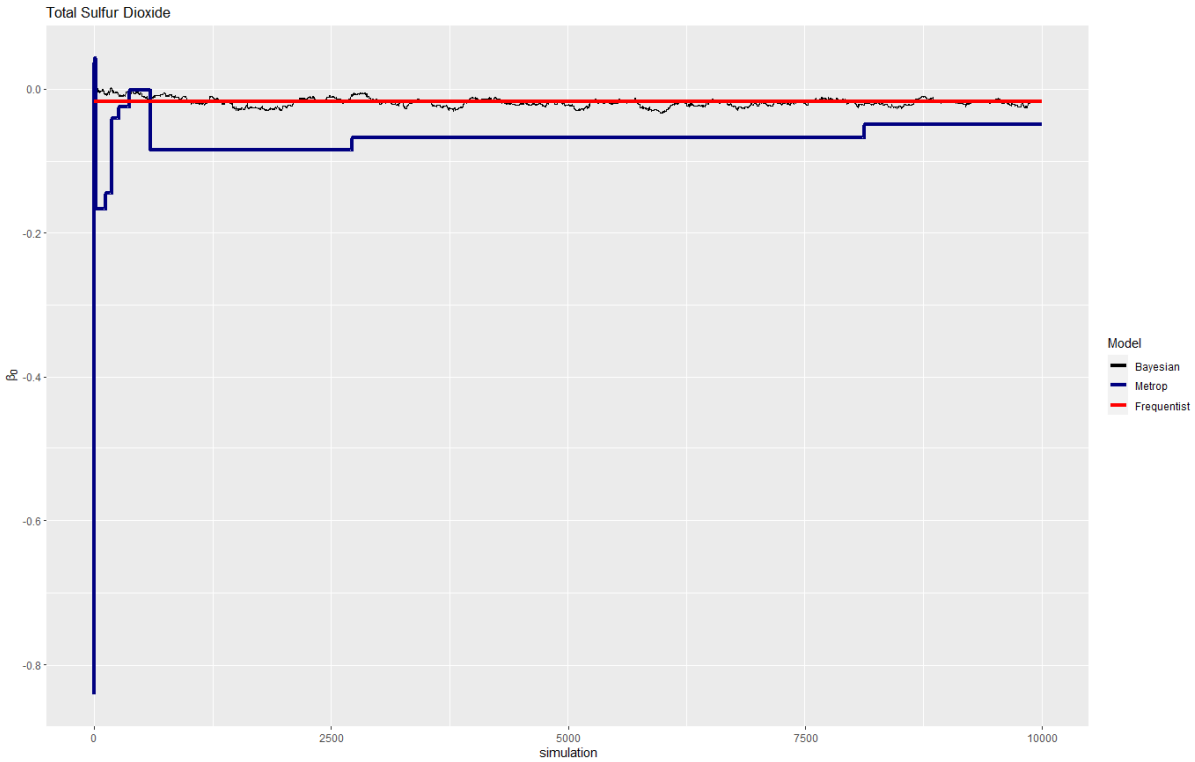
## Free Sulfur Dioxide




| Bayesian MHA | MCMC Metrop | Frequentist GLM |
|---|---|---|
| -0.002135599 | 0.122592311 | 0.010820602 |

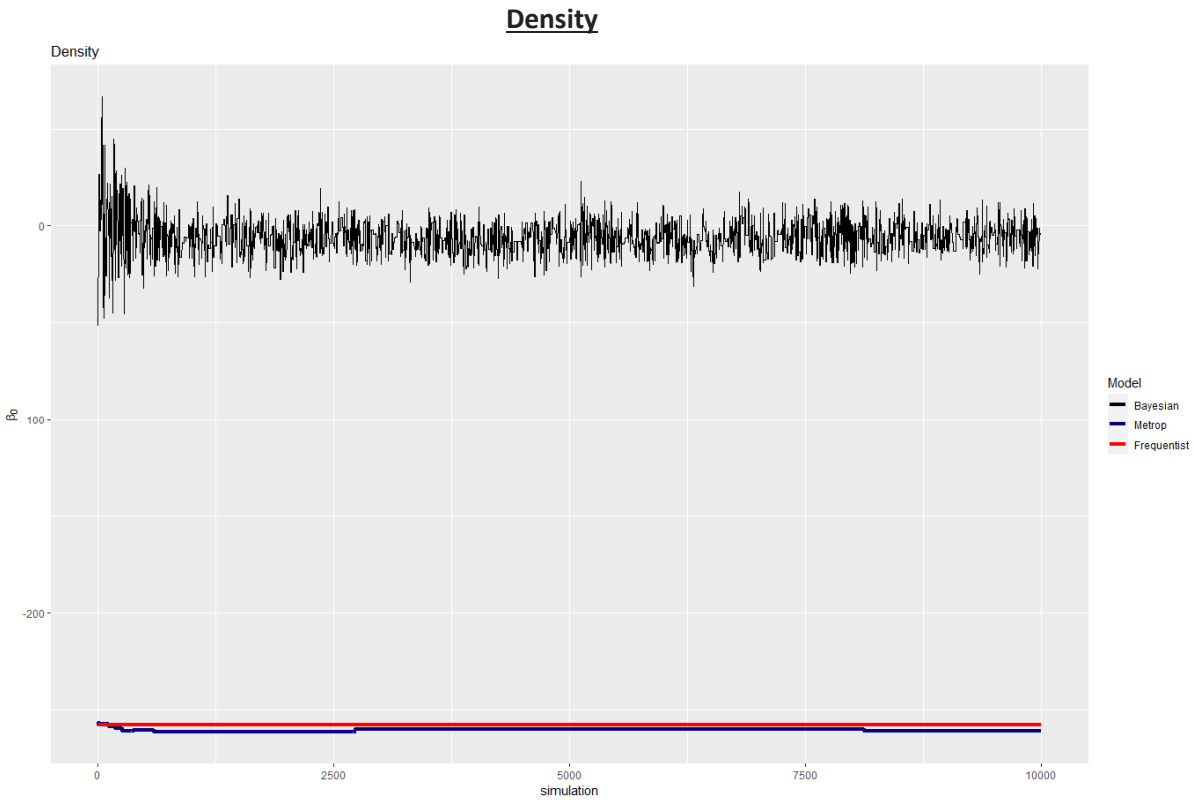
**Comment –** The value of coefficient generated by the metrop function is higher than the values of coefficient generated by the other two models. The value of the coefficient generated by the GLM and Bayesian models do not differ significantly.

## Total Sulfur Dioxide




| Bayesian MHA | MCMC Metrop | Frequentist GLM |
|---|---|---|
| -0.01641886 | -0.04932476 | -0.01653061 |

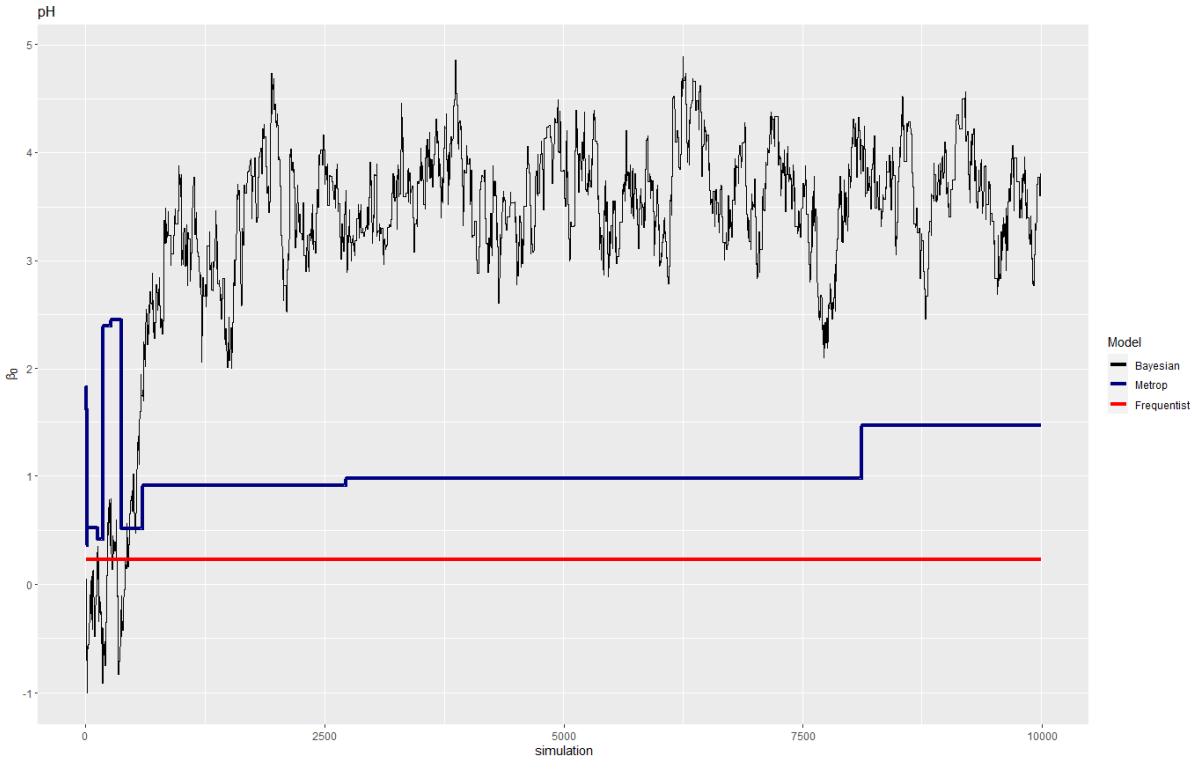
**Comment –** The value of coefficient generated by the metrop function is lower than the values of coefficient generated by the other two models. The value of the coefficient generated by the GLM and Bayesian models do not differ significantly.




| Bayesian MHA | MCMC Metrop | Frequentist GLM |
|---|---|---|
| -3.960263 | -260.886888 | -257.797579 |

**Comment –** The value of coefficient generated by the metrop function and the GLM model are very similar which is significantly higher than the value generated by the Bayesian model. These values do not converge at the MLE as well as show at a large difference with value of coefficient for density generated by the GLM model. This is due the fact this model is very sensitive to tuning parameters.

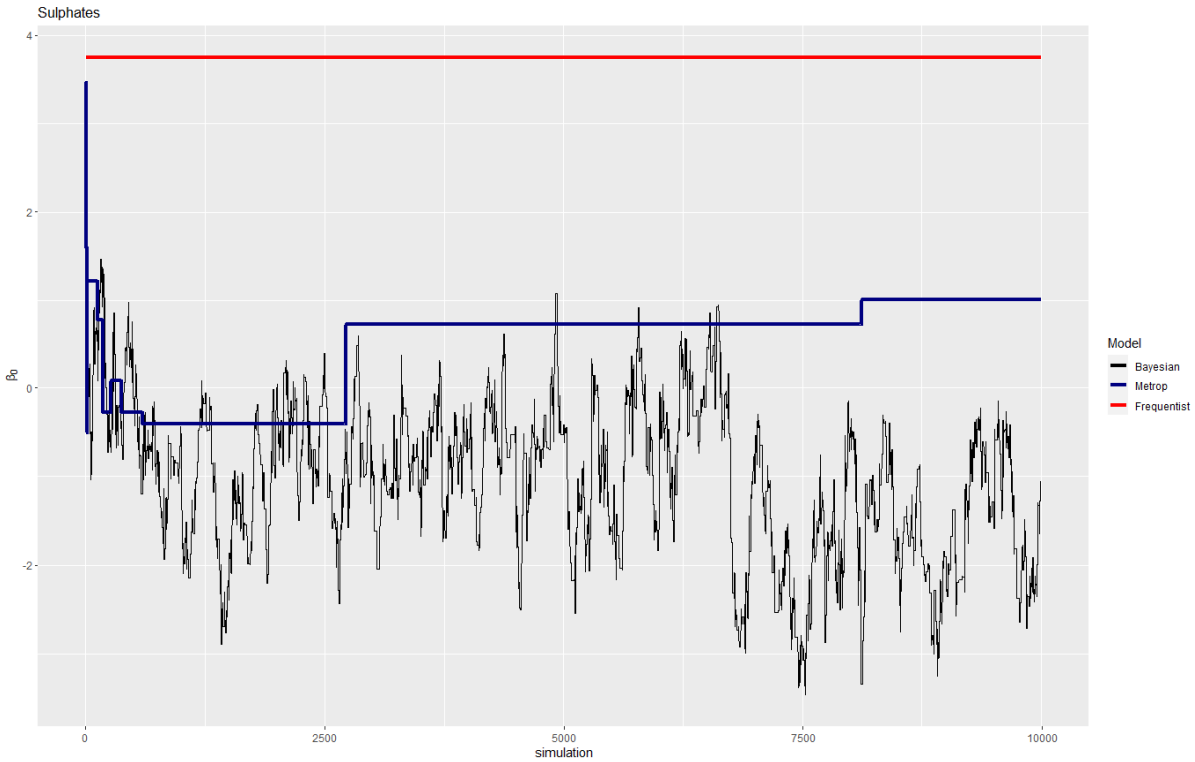
## pH




| Bayesian MHA | MCMC Metrop | Frequentist GLM |
|---|---|---|
| 3.8097271 | 1.4675141 | 0.2241852 |

**Comment –** The value of coefficient generated by the metrop function is lower the values of coefficient generated by the Bayesian model. The value of the coefficient generated by the metrop function is higher than the values of coefficient generated by the GLM model.

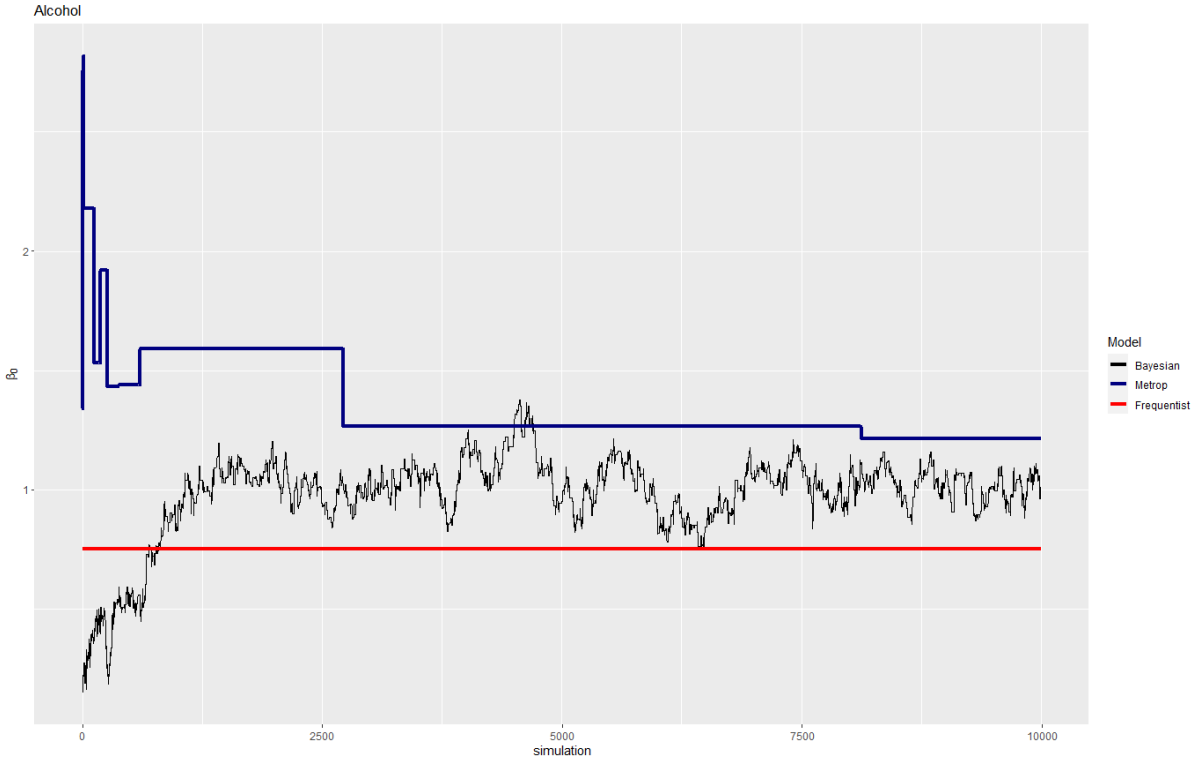
## Sulphates




| Bayesian MHA | MCMC Metrop | Frequentist GLM |
|---|---|---|
| -1.058831 | 1.010802 | 3.749879 |

**Comment –** The value of coefficient generated by the metrop function is lower the values of coefficient generated by the GLM model. The value of the coefficient generated by the metrop function is closer but higher than the values of coefficient generated by the Bayesian model.

**Alcohol**




| Bayesian MHA | MCMC Metrop | Frequentist GLM |
|---|---|---|
| 1.0097784 | 1.2147102 | 0.7533391 |

**Comment –** The value of coefficient generated by the metrop function is higher the values of coefficient generated by the GLM model. The value of the coefficient generated by the Bayesian function is higher than the values of coefficient generated by the GLM model.