Assessment 3

Waikei Lau (Z5349878)

1. [2 points] are given if the submitted R code works with no error.
2. [2 points] Read the dataset into R. Check if there are missing values (NA) and, in case there are, remove them.

```
wines <- read.csv("D:\\General\\Bayesian Inference\\winequality-red.csv")

wines <- na.omit(wines)
```

3. [2 points] 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).

```
wines$Good <- ifelse(wines$quality >= 6.5 , 1 , 0)
```

Created new column where wines with score greater than or equal to 6.5 are 1 otherwise 0.

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

```
obj_log <- glm(wines$Good ~ . - quality, family = binomial(link="logit"), data=wines)

summary(obj_log)
```

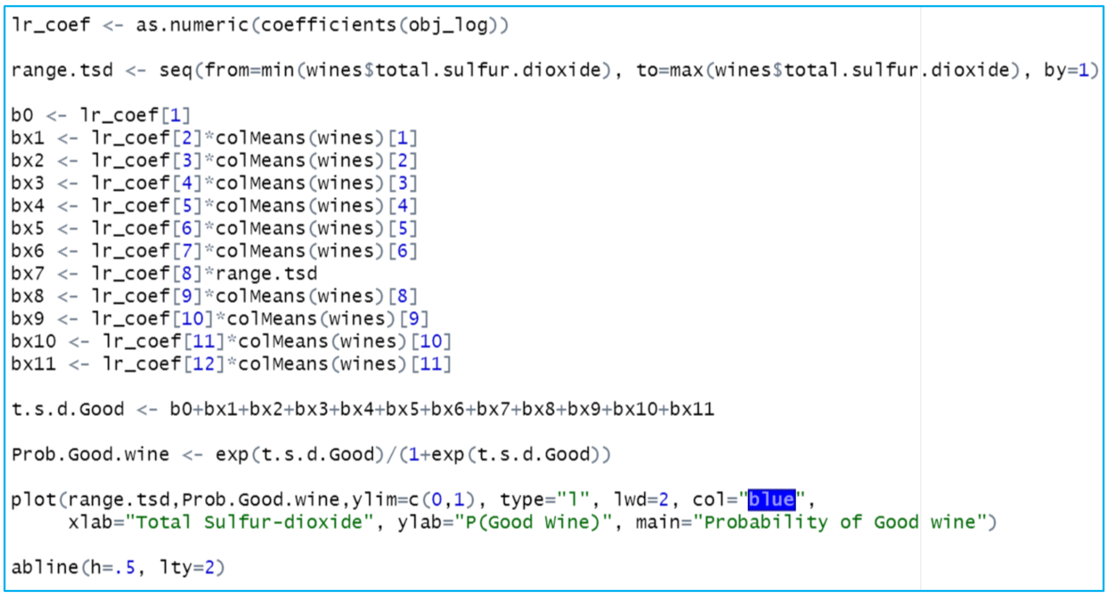
Using the glm function, implemented a logistic regression model with wine quality as response variable and all other coefficients included in model.

```
Call:
glm(formula = wines$Good ~ . - quality, family = binomial(link = "logit"),
    data = wines)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-2.9878  -0.4351  -0.2207  -0.1222   2.9869

Coefficients:
                       Estimate Std. Error z value Pr(>|z|)
(Intercept)           2.428e+02  1.081e+02   2.247 0.024660 *
fixed.acidity         2.750e-01  1.253e-01   2.195 0.028183 *
volatile.acidity     -2.581e+00  7.843e-01  -3.291 0.000999 ***
citric.acid           5.678e-01  8.385e-01   0.677 0.498313
residual.sugar        2.395e-01  7.373e-02   3.248 0.001163 **
chlorides            -8.816e+00  3.365e+00  -2.620 0.008788 **
free.sulfur.dioxide   1.082e-02  1.223e-02   0.884 0.376469
total.sulfur.dioxide -1.653e-02  4.894e-03  -3.378 0.000731 ***
density              -2.578e+02  1.104e+02  -2.335 0.019536 *
pH                    2.242e-01  9.984e-01   0.225 0.822327
sulphates             3.750e+00  5.416e-01   6.924 4.39e-12 ***
alcohol               7.533e-01  1.316e-01   5.724 1.04e-08 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 1269.92  on 1598  degrees of freedom
Residual deviance:  870.86  on 1587  degrees of freedom
AIC: 894.86

Number of Fisher Scoring iterations: 6
```

At α=0.01, the significant coefficients are:

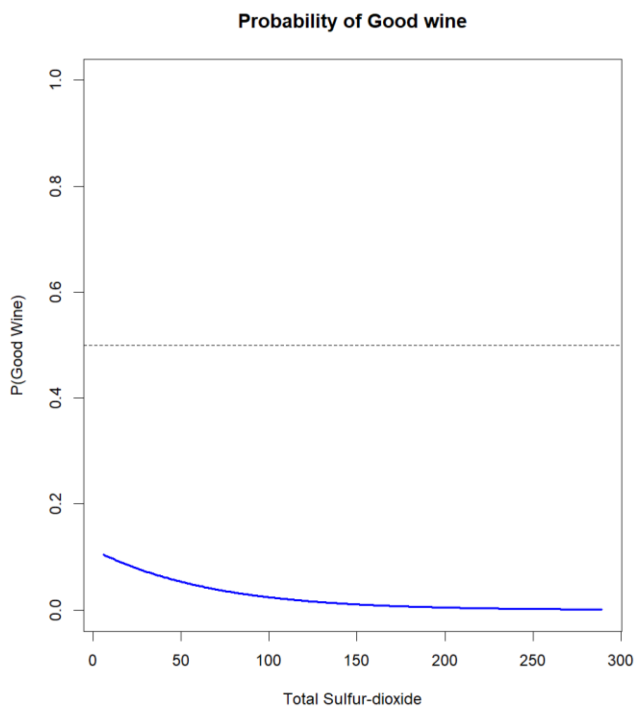
Intercept | fixed acidity | volatile acidity | residual sugar | chlorides | total sulfur dioxide | density | sulphates | alcohol

5. [5 points] 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.



```r
lr_coef <- as.numeric(coefficients(obj_log))

range.tsd <- seq(from=min(wines$total.sulfur.dioxide), to=max(wines$total.sulfur.dioxide), by=1)

b0 <- lr_coef[1]
bx1 <- lr_coef[2]*colMeans(wines)[1]
bx2 <- lr_coef[3]*colMeans(wines)[2]
bx3 <- lr_coef[4]*colMeans(wines)[3]
bx4 <- lr_coef[5]*colMeans(wines)[4]
bx5 <- lr_coef[6]*colMeans(wines)[5]
bx6 <- lr_coef[7]*colMeans(wines)[6]
bx7 <- lr_coef[8]*range.tsd
bx8 <- lr_coef[9]*colMeans(wines)[8]
bx9 <- lr_coef[10]*colMeans(wines)[9]
bx10 <- lr_coef[11]*colMeans(wines)[10]
bx11 <- lr_coef[12]*colMeans(wines)[11]

t.s.d.Good <- b0+bx1+bx2+bx3+bx4+bx5+bx6+bx7+bx8+bx9+bx10+bx11

Prob.Good.wine <- exp(t.s.d.Good)/(1+exp(t.s.d.Good))

plot(range.tsd,Prob.Good.wine,ylim=c(0,1), type="l", lwd=2, col="blue",
     xlab="Total Sulfur-dioxide", ylab="P(Good Wine)", main="Probability of Good wine")

abline(h=.5, lty=2)
```

Fixing all covariates at mean levels, and multiplying by the corresponding MLE from logistic regression, we see that the probability of a good wine increases to ~10% as Total Sulfur dioxide decreases.

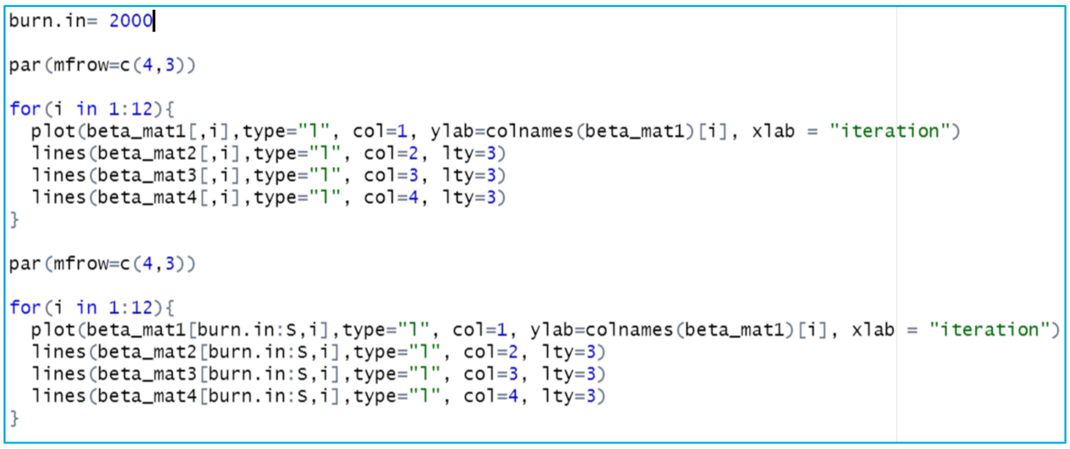


6. [15 points] Perform a Bayesian analysis of the logistic model for the dataset, i.e. approximate the posterior distributions of the regression coefficients, following these steps:

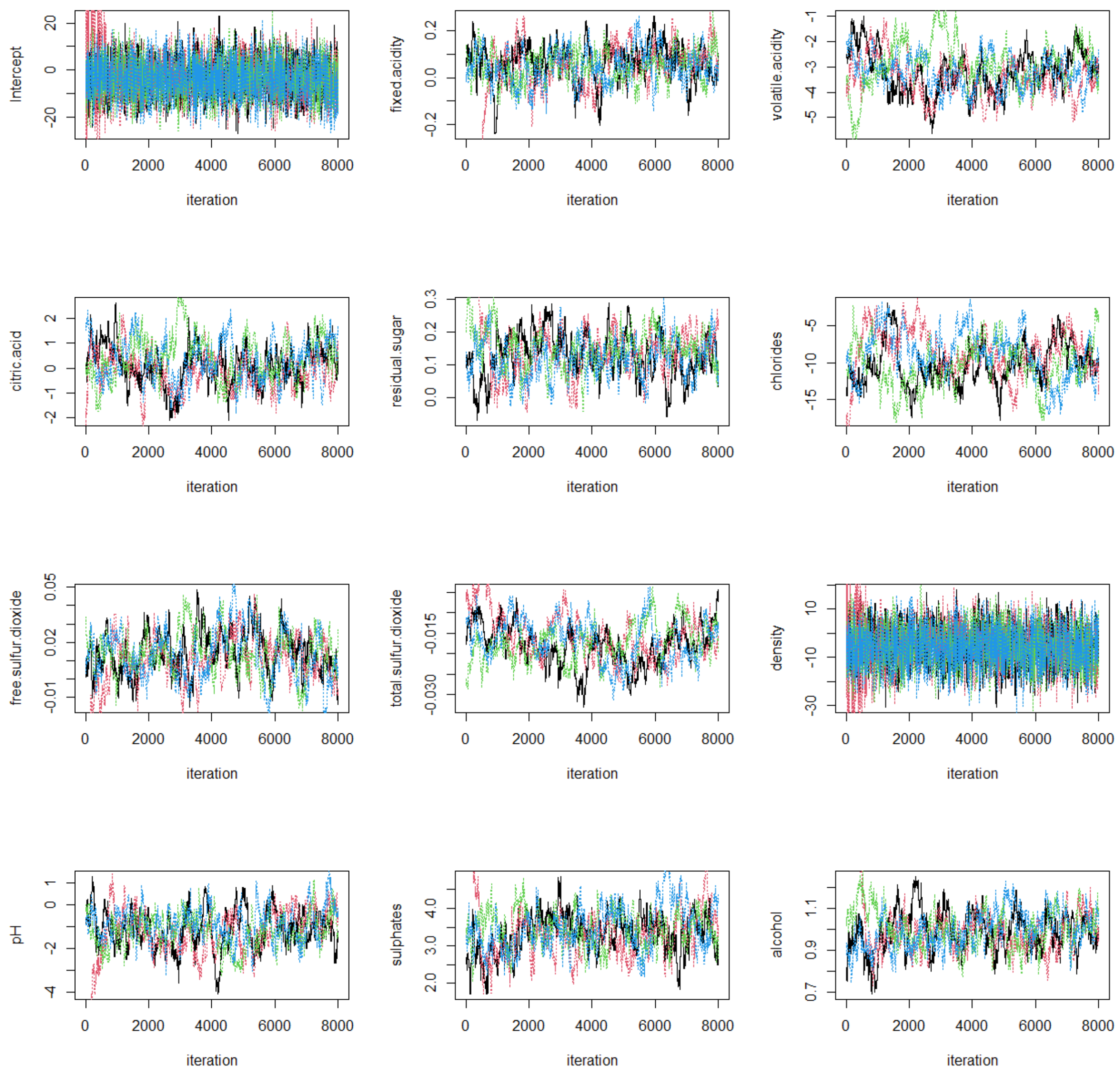
- Write an R function for the log posterior distribution.
- Fix the number of simulation at 10^4.
- Choose 4 different initialisations for the coefficients.
- For each initialisation, run a Metropolis–Hastings algorithm.
- Plot the chains for each coefficients (the 4 chains on the same plot) and comment.

```r
# Log-posterior distribution
lpost.LR <- function(beta,x,y){
  eta <- as.numeric(x %*% beta)
  logp <- eta - log(1+exp(eta))
  logq <- log(1-exp(logp))
  logl <- sum(logp[y==1]) + sum(logq[y==0])
  lprior <- sum(dnorm(beta,0,10,log=T))
  return(logl + lprior)
}

S <- 10^4

X = as.matrix(cbind(rep(1,nrow(wines)), wines[,1:11]))
colnames(X)[1] <- "Intercept"
y = wines[,"Good"]


k <- ncol(X)
acc <-0
Omega_prop <- solve(t(X) %*% X)

library(mvtnorm)

beta_mat1 <- beta_mat2 <- beta_mat3 <- beta_mat4 <- matrix(NA, nrow=S, ncol= ncol(X))
colnames(beta_mat1) <- colnames(beta_mat2) <- colnames(beta_mat3) <- colnames(beta_mat4) <- colnames(X)

beta_mat1[1,] <- lr_coef
beta_mat2[1,] <- c(10,0.5,-3, 1.2,0.3,-10,0.05,-0.018,-10,0.3,4,0.9)
beta_mat3[1,] <- c(-10,-0.1,-5,-1,-0.35,-12,0.05,-0.01,10,0,0,0)
beta_mat4[1,] <- c(rep(0,12))

tuning = 0.85
acc=0
```

```r
for(i in 2:S){

  # 1. Propose a new set of values
  beta_star1 <- rmvnorm(1,beta_mat1[i-1,],tuning*Omega_prop)
  beta_star2 <- rmvnorm(1,beta_mat2[i-1,],tuning*Omega_prop)
  beta_star3 <- rmvnorm(1,beta_mat3[i-1,],tuning*Omega_prop)
  beta_star4 <- rmvnorm(1,beta_mat4[i-1,],tuning*Omega_prop)

  # 2. Compute the posterior density on the proposed value and on the old value
  newpost1=lpost.LR(t(beta_star1),X,y)
  newpost2=lpost.LR(t(beta_star2),X,y)
  newpost3=lpost.LR(t(beta_star3),X,y)
  newpost4=lpost.LR(t(beta_star4),X,y)

  oldpost1=lpost.LR(matrix(beta_mat1[i-1,],ncol=1),X,y)
  oldpost2=lpost.LR(matrix(beta_mat2[i-1,],ncol=1),X,y)
  oldpost3=lpost.LR(matrix(beta_mat3[i-1,],ncol=1),X,y)
  oldpost4=lpost.LR(matrix(beta_mat4[i-1,],ncol=1),X,y)

  # 3. Acceptance step
  if(runif(1,0,1)>exp(newpost1-oldpost1)){
    beta_mat1[i,]=beta_mat1[i-1,]
  } else {
    beta_mat1[i,]=beta_star1
    acc=acc+1
  }
  if(runif(1,0,1)>exp(newpost2-oldpost2)){
    beta_mat2[i,]=beta_mat2[i-1,]
  } else {
    beta_mat2[i,]=beta_star2
    acc=acc+1
  }
  if(runif(1,0,1)>exp(newpost3-oldpost3)){
    beta_mat3[i,]=beta_mat3[i-1,]
  } else {
    beta_mat3[i,]=beta_star3
    acc=acc+1
  }
  if(runif(1,0,1)>exp(newpost4-oldpost4)){
    beta_mat4[i,]=beta_mat4[i-1,]
  } else {
    beta_mat4[i,]=beta_star4
    acc=acc+1
  }

  # 4. Print the stage of the chain
  if(i%%1000==0){print(c(i,acc/(i*4)))}

}
```



```
burn.in= 2000

par(mfrow=c(4,3))

for(i in 1:12){
  plot(beta_mat1[,i],type="l", col=1, ylab=colnames(beta_mat1)[i], xlab = "iteration")
  lines(beta_mat2[,i],type="l", col=2, lty=3)
  lines(beta_mat3[,i],type="l", col=3, lty=3)
  lines(beta_mat4[,i],type="l", col=4, lty=3)
}

par(mfrow=c(4,3))

for(i in 1:12){
  plot(beta_mat1[burn.in:S,i],type="l", col=1, ylab=colnames(beta_mat1)[i], xlab = "iteration")
  lines(beta_mat2[burn.in:S,i],type="l", col=2, lty=3)
  lines(beta_mat3[burn.in:S,i],type="l", col=3, lty=3)
  lines(beta_mat4[burn.in:S,i],type="l", col=4, lty=3)
}
```



The log posterior distribution is:

$$\log L(\boldsymbol{\beta};\mathbf{y},\mathbf{x}) = \log\left[\prod_{i=1}^{n}\left(\frac{\exp(\mathbf{x}_i\boldsymbol{\beta})}{1+\exp(\mathbf{x}_i\boldsymbol{\beta})}\right)^{y_i}\left(1-\frac{\exp(\mathbf{x}_i\boldsymbol{\beta})}{1+\exp(\mathbf{x}_i\boldsymbol{\beta})}\right)^{1-y_i}\right]$$

Four different initialisations were chosen:

- Using MLE from logistic regression model
- Two randomly selected starting points based on deviations from MLE and standard deviations.
- All coefficients starting from zero.

A tuning parameter of 0.85 was selected for an acceptance rate close to 23%.

A burn-in was selected at 2,000 out of 10,000 – where in most coefficients appear to have achieved convergence.

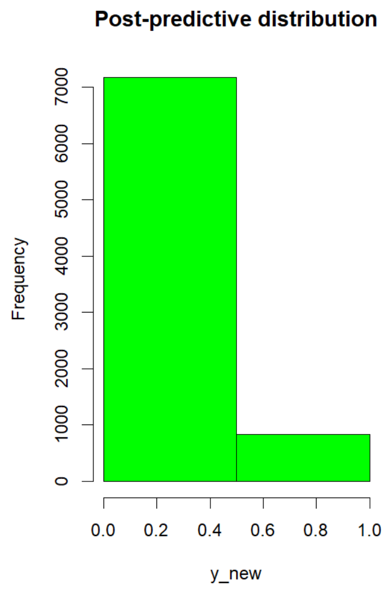
Observations on the intercept and the density coefficient that deviated from the MLE of 242.8 and -257.8 respectively. Convergence was observed around -3.29 and -5.1 respectively with no autocorrelation detected.

7. [5 points] Approximate the posterior predictive distribution of an unobserved variable characterised by:

- fixed acidity: 7.5 | volatile acidity: 0.6 | citric acid: 0.0 | residual sugar: 1.70 | chlorides: 0.085 | free sulfur dioxide: 5 | total sulfur dioxide: 45 | density: 0.9965 | pH: 3.40 | sulphates: 0.63 | alcohol: 12

Plot the approximate posterior predictive distribution.

```
y_new <- c(1)
x_new <- c(1,7.5,0.6,0,1.7,0.085,5,45,0.9965,3.4,0.63,12)
par(mfrow=c(1,1))

for(i in burn.in:S){

  # 5. Prediction
  p_new <- exp(sum(beta_mat1[i,] * x_new) ) / (1 + exp(sum(beta_mat1[i] * x_new) ))
  y_new[i] <- rbinom(1,1,prob=p_new)

}

hist(y_new, main="Post-predictive distribution",
     breaks = 2, col="green")
```



Implemented predictive X values in combination with the beta matrix to derive probability of success. This was incorporated into Bernoulli (binomial) samples on the response variable y.

Total accepted: 831
Total rejected: 7171
Acceptance rate: 10.38%

8. [5 points] 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. (Here a visual comparison of the chains is enough to get full mark).

```r
# Log-posterior distribution
lpost.metr <- function(x,y)function(beta){
  eta <- as.numeric(x %*% beta)
  logp <- eta - log(1+exp(eta))
  logq <- log(1-exp(logp))
  logl <- sum(logp[y==1]) + sum(logq[y==0])
  lprior <- sum(dnorm(beta,0,10,log=T))
  return(logl + lprior)
}

lpost <- lpost.metr(X, y)

burn.in= 2000
tuning = 0.25

out1 <- metrop(lpost, beta_mat1[1,], S, scale=tuning*Omega_prop)
out2 <- metrop(lpost, beta_mat2[1,], S, scale=tuning*Omega_prop)
out3 <- metrop(lpost, beta_mat3[1,], S, scale=tuning*Omega_prop)
out4 <- metrop(lpost, beta_mat4[1,], S, scale=tuning*Omega_prop)

out1$accept
out2$accept
out3$accept
out4$accept

par(mfrow=c(4,3))

axis <- rbind(c(-25,25),c(-0.5,0.8),c(-6,1),c(-1.8,2),c(-0.5,0.5),c(-15,1),
              c(-0.02,0.07),c(-0.025,0.005),c(-25,25),c(-5,5),c(-1,5),c(-0.2,1.2))


for(i in 1:12){
  plot(out1$batch[burn.in:S,i],type="l", col=1, ylab=colnames(beta_mat1)[i], xlab = "iteration", ylim=axis[i,])
  lines(out2$batch[burn.in:S,i],type="l", col=2, lty=3)
  lines(out3$batch[burn.in:S,i],type="l", col=3, lty=3)
  lines(out4$batch[burn.in:S,i],type="l", col=4, lty=3)
}
```

Using the metrop() function, both the intercept and density coefficients appear to have achieved the stationary distribution where all four initialisations converged. However, other coefficients did not achieve convergence within 10,000 iterations using the metrop() function.

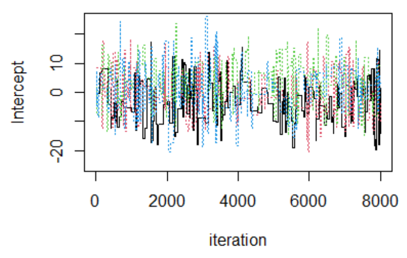
As can be seen form the plots after a burn in of 2,000 observations was applied, the plots of various initialisations did not converge. This was after using the four initialisations from earlier:

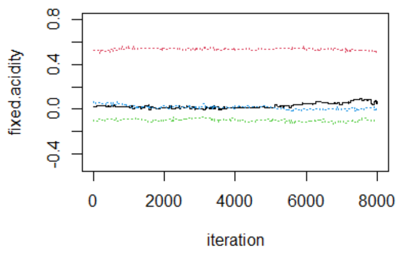
- Using MLE from logistic regression model
- Two randomly selected starting points based on deviations from MLE and standard deviations.
- All coefficients starting from zero.

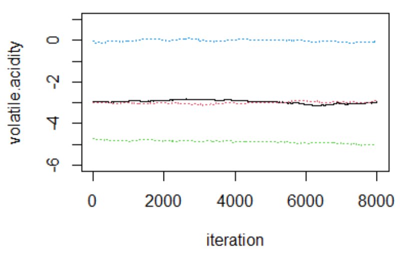
A smaller tuning parameter of 0.25 was required to achieve an acceptance rate close to 23%, which was gradually adjusted to 0.9 to test for convergence. The plots are provided below showing separation between coefficients at both 25% and 90% tuning.

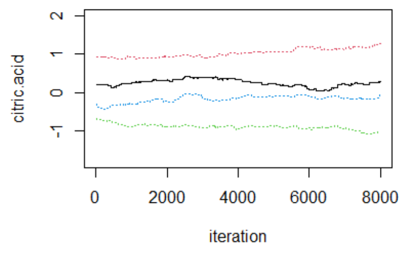
Additionally there appears to be autocorrelation as the coefficients follow a relatively flat trend for consecutive iterations indicating multiple rejections.

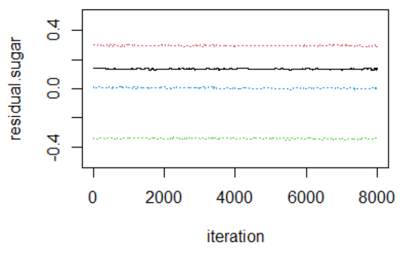
Tuning at 25%

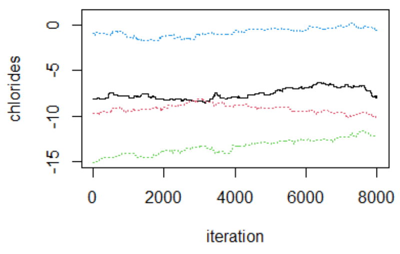

Intercept
-20
0
10
0
2000
4000
6000
8000
iteration

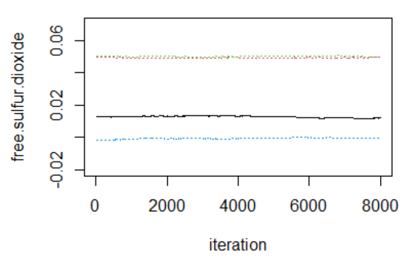

fixed.acidity
-0.4
0.0
0.4
0.8
0
2000
4000
6000
8000
iteration

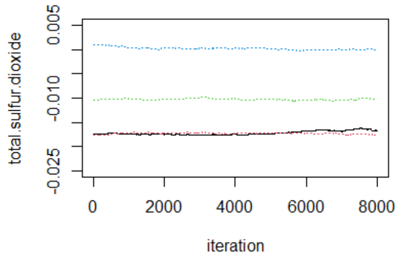

volatile.acidity
-6
-4
-2
0
0
2000
4000
6000
8000
iteration

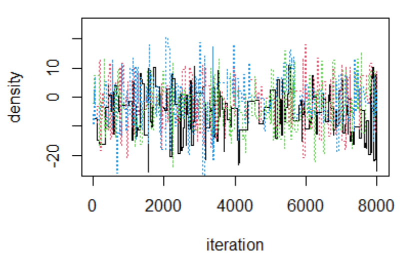

citric.acid
-1
0
1
2
0
2000
4000
6000
8000
iteration

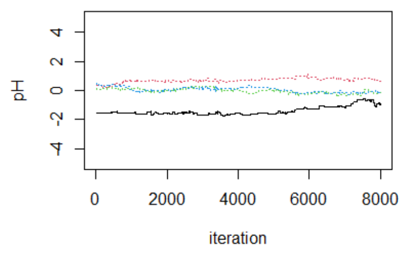

residual.sugar
-0.4
0.0
0.4
0
2000
4000
6000
8000
iteration

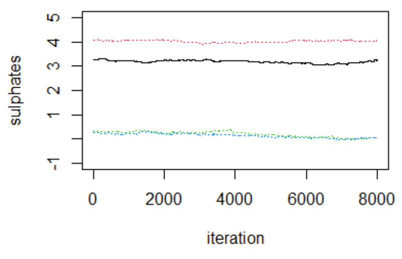

chlorides
-15
-10
-5
0
0
2000
4000
6000
8000
iteration

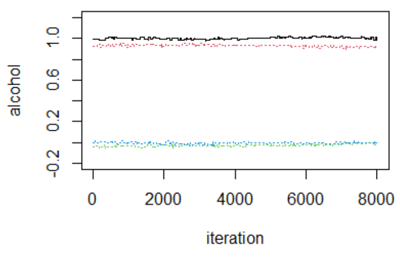

free.sulfur.dioxide
-0.02
0.02
0.06
0
2000
4000
6000
8000
iteration


total.sulfur.dioxide
-0.025
-0.010
0.005
0
2000
4000
6000
8000
iteration

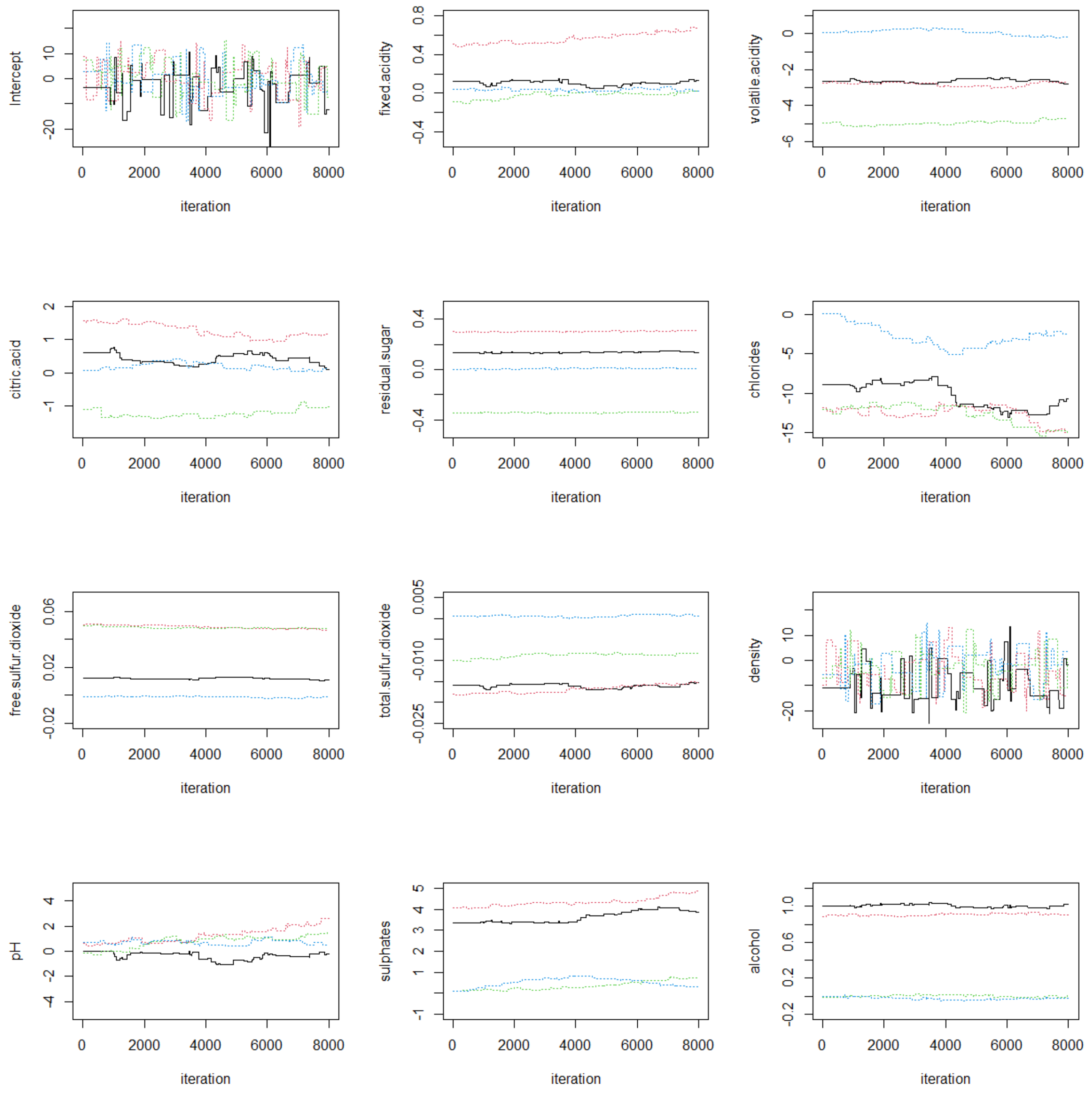

density
-20
0
10
0
2000
4000
6000
8000
iteration


pH
-4
-2
0
2
4
0
2000
4000
6000
8000
iteration


sulphates
-1
1
2
3
4
5
0
2000
4000
6000
8000
iteration


alcohol
-0.2
0.2
0.6
1.0
0
2000
4000
6000
8000
iteration

Tuning at 90%


Intercept
fixed.acidity
volatile.acidity
citric.acid
residual.sugar
chlorides
free.sulfur.dioxide
total.sulfur.dioxide
density
pH
sulphates
alcohol
iteration