STT 465 HW5

Nate Davis

11/30/2017

Read in the data

```
data <- read.table("gout.txt")

colnames(data) <- c('sex', 'race', 'age', 'serum_urate', 'gout')

data$gout <- ifelse(data$gout == "Y", 1, 0)
```

Question 1

1.1

```
x <- as.matrix(model.matrix(~sex+age+race, data = data))[,-1]

model <- glm(gout ~ x, data = data, family = 'binomial')
summary(model)
```

```
##
## Call:
## glm(formula = gout ~ x, family = "binomial", data = data)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -0.8294  -0.4256  -0.3537  -0.2713   2.6115
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept) -8.08409    2.35356  -3.435 0.000593 ***
## xsexM        0.44915    0.38821   1.157 0.247288
## xage         0.09239    0.03644   2.536 0.011227 *
## xraceW      -0.74329    0.41914  -1.773 0.076168 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 213.11  on 399  degrees of freedom
## Residual deviance: 203.05  on 396  degrees of freedom
## AIC: 211.05
##
## Number of Fisher Scoring iterations: 5
```

1.2

From above we see that the intercept has a very large, very significant effect within the model. The other significant variables at the standard level is age – race is slighly significant but probably negligible.

1.3

We have $Y_i = 0.44915 * 1 + 0.09239 * 65 - 0.74329 * 1 - 8.08409$ which makes our linear predictor equal to about -2.37288. Using our link, we have $log(\frac{p_i}{1-p_i}) = -2.37288$, **making our predicted probability approximately 0.085**

Question 2

```
# A function to evaluate the log of the posterior density
logP=function(y,X,b,b0,varB){
     Xb=X%*%b
     theta=exp(Xb)/(1+exp(Xb))
     logLik=sum( dbinom(x=y,p=theta,size=1,log=T)  )
     logPrior=sum(  dnorm(x=b,sd=sqrt(varB),mean=b0,log=T))
     return(logLik+logPrior)
   }


logisticRegressionBayes=function(y,X,nIter=12000,V=.02,varB=rep(10000,ncol(X)),b0=rep(0,ncol(X))){

  ####### Arguments ########################
  # y  a vector with 0/1 values
  # X  incidence matrix fo effects
  # b0,varB, the prior mean and prior variance bj~N(b0[j],varB[j])
  # V the variance of the normal distribution used to generate candidates~N(b[i-1],V)
  # nIter: number of iterations of the sampler
  # Details: generates samples from the posterior distribution of a logistic regression using a
  # Metropolis algorithm
  #########################################

  # A matrix to store samples
   p=ncol(X)
   B=matrix(nrow=nIter,ncol=p)

   # Centering predictors
   meanX=colMeans(X)
   for(i in 2:p){ X[,i]=(X[,i]-meanX[i]) }


  # A vector to trace acceptancve
   accept=rep(NA,nIter)
   accept[1]=TRUE

  # Initialize
   B[1,]=0
   B[1,1]=log(mean(y)/(1-mean(y)))
   b=B[1,]
  for(i in 2:nIter){

    candidate=rnorm(mean=b,sd=sqrt(V),n=p)

    logP_current=logP(y,X,b0=b0,varB=varB,b=b)
    logP_candidate=logP(y,X,b0=b0,varB=varB,b=candidate)

    r=min(1,exp(logP_candidate-logP_current))
    delta=rbinom(n=1,size=1,p=r)

    accept[i]=delta

    if(delta==1){ b=candidate }
    B[i,]=b

  }

  B[,1]=B[,1]-B[,-1]%*%meanX[-1] # absorbing means on the intercept

  return(list(B=B,accept=accept))
}

x_new = model.matrix(~sex+age+race,data=data)

b_model <- logisticRegressionBayes(data$gout, x_new, 55000)

result <- cbind(model$coefficients,colMeans(b_model$B[-(1:5000),]))

params <- b_model$B[-(1:5000),]
colnames(params) <- c("Intercept", "Sex", "Age", "Race")
```

2.1

```
library("coda")
```

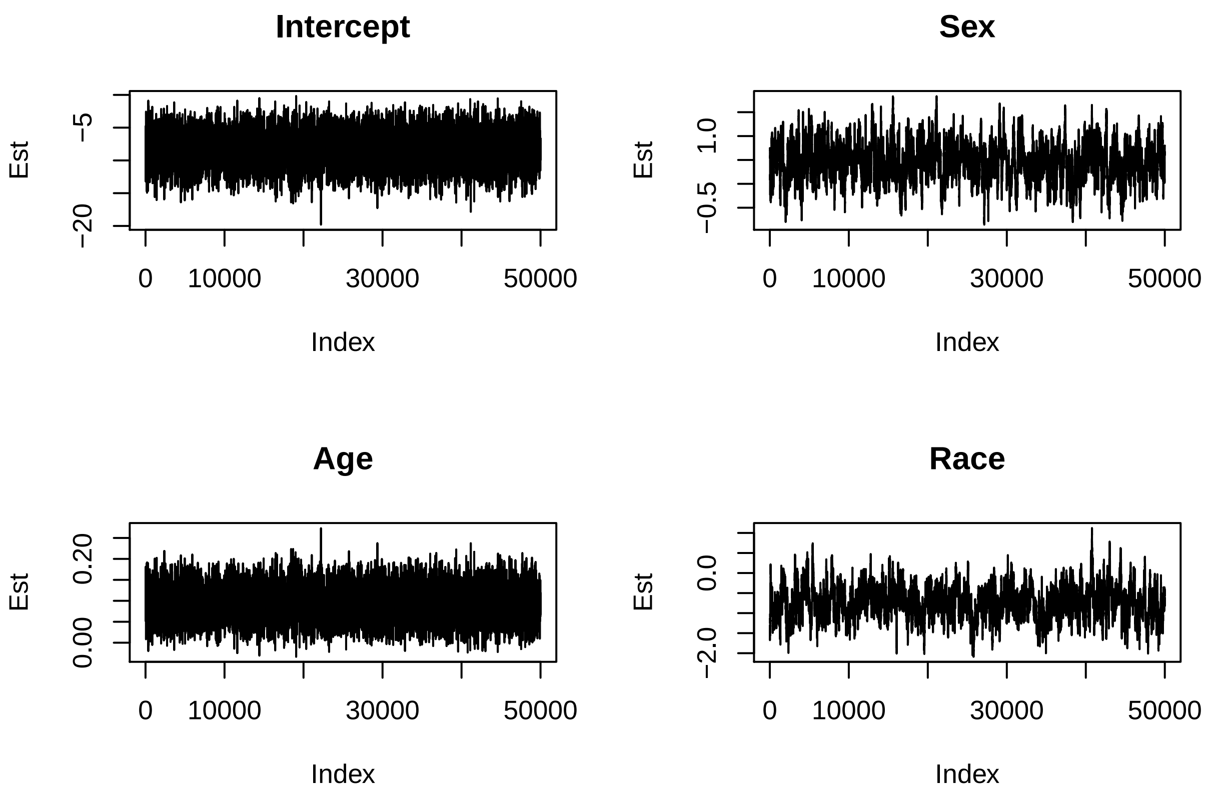
```
## Warning: package 'coda' was built under R version 3.3.2
```

```
params <- data.frame(params)

mcmc_obj <- mcmc(params)

obj_sum <- summary(mcmc_obj)

obj_sum
```

```
##
## Iterations = 1:50000
## Thinning interval = 1
## Number of chains = 1
## Sample size per chain = 50000
##
## 1. Empirical mean and standard deviation for each variable,
##    plus standard error of the mean:
##
##                Mean      SD  Naive SE Time-series SE
## Intercept -8.33091 2.37854 0.0106372       0.037958
## Sex        0.46491 0.40030 0.0017902       0.021442
## Age        0.09514 0.03694 0.0001652       0.000565
## Race      -0.73726 0.42329 0.0018930       0.024159
##
## 2. Quantiles for each variable:
##
##                2.5%     25%      50%     75%   97.5%
## Intercept -13.11548 -9.9164 -8.30724 -6.7052 -3.8026
## Sex        -0.35085  0.1954  0.46686  0.7360  1.2452
## Age         0.02378  0.0702  0.09497  0.1199  0.1686
## Race       -1.53226 -1.0322 -0.74542 -0.4608  0.1242
```

2.2

```r
par(mfrow = c(2, 2))

plot(params$Intercept, xlab = "Index", ylab = "Est", main = "Intercept", type = "l")
plot(params$Sex, xlab = "Index", ylab = "Est", main = "Sex", type = "l")
plot(params$Age, xlab = "Index", ylab = "Est", main = "Age", type = "l")
plot(params$Race, xlab = "Index", ylab = "Est", main = "Race", type = "l")
```



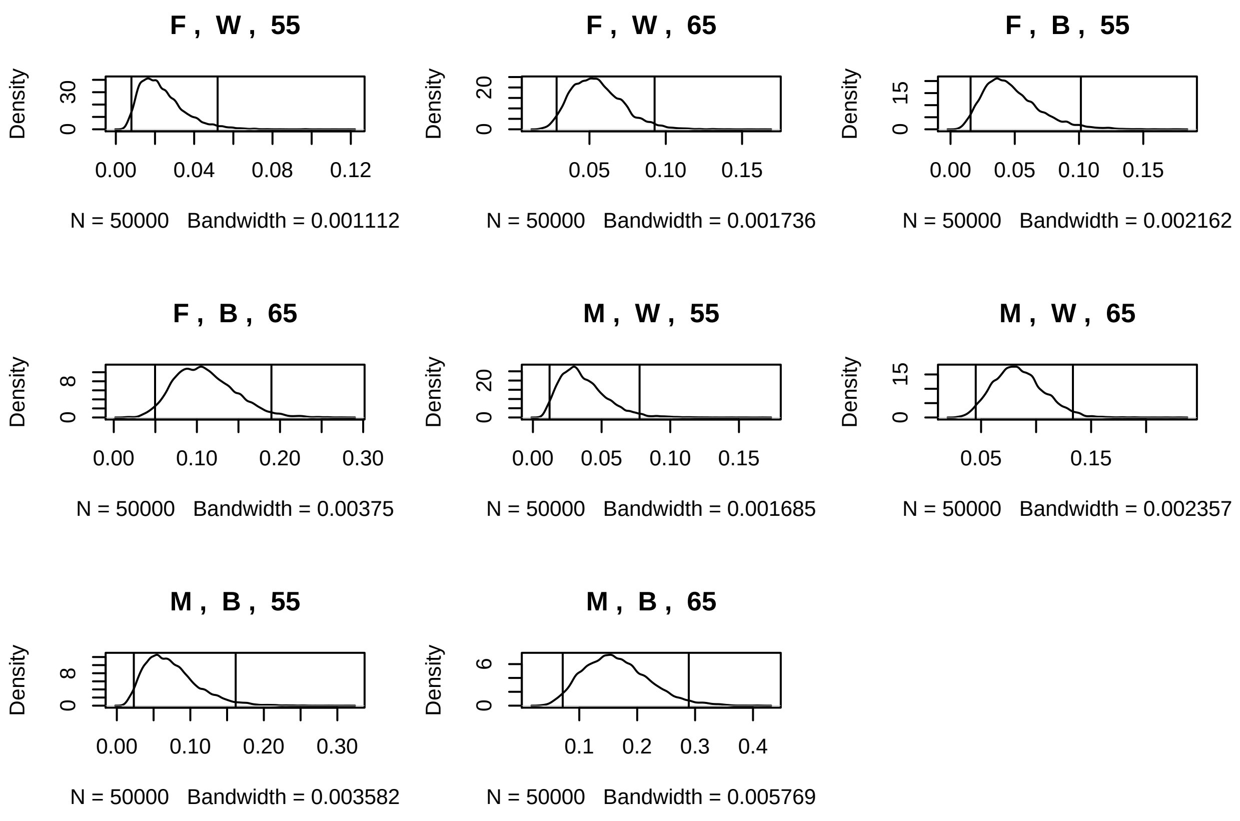
```r
size_se <- cbind(effectiveSize(mcmc_obj), obj_sum$statistics[,3])
colnames(size_se) <- c("Effective Size", "MC SE")
```

```
size_se
```

```
##           Effective Size        MC SE
## Intercept      3926.5313 0.0106371562
## Sex             348.5334 0.0017901858
## Age            4274.2771 0.0001652075
## Race            306.9899 0.0018929985
```

2.3

```
pts <- cbind(rep(c(0, 1), each = 4), rep(c(55, 65), 4), rep(c(1, 0), 2, each = 2))

par(mfrow = c(3, 3))

for(i in 1:length(pts[,1])){

    resp <- params[, 1] + params[, 2]*pts[i, 1] + params[, 3]*pts[i, 2]+ params[, 4]*pts[i, 3]

    resp <- exp(resp) / (exp(resp) + 1)

    sex <- ifelse(pts[i, 1] == 0, "F", "M")
    race <- ifelse(pts[i, 3] == 0, "B", "W")

    title <- paste(sex, ", ", race, ", ", as.character(pts[i, 2]))

    plot(density(resp), main = title)
    abline(v = quantile(resp, probs = c(0.025, 0.975)))
}
```



Question 3

3.1

```r
v_values <- c(.4, .2, .05, .001)

res <- cbind(rep(rep(NA, 4)), NA, NA)
row.names(res) <- v_values
colnames(res) <- c("Acc Rate", "Lag Corr", "Effective Size")

for(i in 1:length(v_values)){

    b_model <- logisticRegressionBayes(data$gout, model.matrix(~sex+age+race,data=data),
                                        55000, V = v_values[i])

    params <- b_model$B[-(1:5000),]
    colnames(params) <- c("Intercept", "Sex", "Age", "Race")

    res[i, 1] <- sum(b_model$accept)/length(b_model$accept)

    obj <- mcmc(params)

    res[i, 2] <- autocorr(obj[,3], lags = 100)

    res[i, 3] <- effectiveSize(obj)[3]
}

res
```

```
##         Acc Rate    Lag Corr Effective Size
## 0.4   0.01843636  0.11222517       665.3349
## 0.2   0.03587273  0.06270784      1112.9198
## 0.05  0.12898182 -0.01003891      3653.8466
## 0.001 0.70643636  0.09698724      1781.4365
```

3.2

The best value from above should be around 0.05 as it not only has the consistently smallest lag correlation value, but also the greatest effective sample size. We get more information from our algorithm when its ran with V = 0.05.