MH Examples: Logistic Regression

Chang Tu

```
###MH Examples: Logistic Regression

library(MASS)
library(mvtnorm)      #mutivariate normal density function and random variate generat
ion
library(LearnBayes)  #laplace function
library(arm)          #logit, invlogit, display and bayesglm functions
```

```
## Loading required package: Matrix
```

```
## Loading required package: lme4
```

```
##
## arm (Version 1.12-2, built: 2021-10-15)
```

```
## Working directory is /work/files/workspace/Baysian-inference/CODE
```

```
#################################################################################
##########       Logistic regression example #####################################
##################################################################################

##If not using a Rstudio project, unhash the line below and change to the
##name of the folder you are using.

#setwd("~/Patrick/Stat314-2018")

##read in additional functions to help  with logistic
##regression
##The code below assumes a folder structure with sub-folders "R" for code
##and "data". If you do not have this set-up you will need to modify
##the lines below appropriately.

source("~/workspace/assignment4/logisticregression_functions.r")
readdata <- read.csv("~/workspace/assignment4/nzis11-cart-surf.csv",header=TRUE)
str(readdata)
```

```
## 'data.frame':    29471 obs. of  8 variables:
##  $ ethnicity    : int  1 1 1 3 1 1 1 1 1 2 ...
##  $ lgr          : int  2 2 10 2 2 5 10 10 10 1 ...
##  $ sex          : int  2 2 2 2 2 2 1 1 1 2 ...
##  $ agegrp       : int  45 65 40 15 30 65 20 50 40 45 ...
##  $ qualification: int  3 1 3 2 4 1 2 4 3 3 ...
##  $ occupation   : int  2 10 2 10 2 7 3 1 2 4 ...
##  $ hours        : num  30 0 15 0 37.5 16 0 45 40 40 ...
##  $ income       : int  644 0 2671 458 1019 1033 0 2405 576 540 ...
```

```r
##To check functions have been read in:
logpost_logit_norm  #code should appear in the console
```

```
## function (beta, prior_mean, prior_variance, model)
## {
##     log_posterior <- loglike_logit(beta = beta, model = model) +
##         dmvnorm(x = beta, mean = prior_mean, sigma = prior_variance,
##             log = TRUE)
##     return(log_posterior)
## }
```

```r
mylaplace           #code should appear in the console
```

```
## function (logpost, mode, maxiter = 1000, ...)
## {
##     options(warn = -1)
##     fit = optim(mode, logpost, gr = NULL, ..., hessian = TRUE,
##         control = list(fnscale = -1, maxit = maxiter))
##     options(warn = 0)
##     mode = fit$par
##     h = -solve(fit$hessian)
##     p = length(mode)
##     int = p/2 * log(2 * pi) + 0.5 * log(det(h)) + logpost(mode,
##         ...)
##     stuff = list(mode = mode, var = h, int = int, converge = fit$convergence ==
##         0, code = fit$convergence, counts = fit$counts)
##     return(stuff)
## }
```

```r
###set-up variables

attach(readdata)
highincome <- as.numeric(readdata$income > 1250)
str(highincome)
```

```
##  num [1:29471] 0 0 1 0 0 0 0 1 0 0 ...
```

```r
mean(highincome)
```

```
## [1] 0.145974
```

```r
agefactor <- relevel(as.factor(readdata$agegrp),"40")
regfactor <-  relevel(as.factor(readdata$lgr),"2")  #Auckland as ref
sexfactor <- relevel(as.factor(readdata$sex),"1")  #male as ref
qualfactor <- relevel(as.factor(readdata$qualification),"2") #school quals as ref
table(qualfactor)
```

```
## qualfactor
##    2    1    3    4    5
## 7064 6891 8435 5223 1858
```

```r
occfactor <- relevel(as.factor(readdata$occupation),"1")
table(occfactor)
```

```
## occfactor
##     1     2     3     4     5     6     7     8     9    10
##  3162  4540  2377  1734  2126  1688  1049  2154    24 10617
```

```r
obshours_centred <- readdata$hours-mean(readdata$hours)
obshours_cen40  <- readdata$hours-40
hourscut <- cut(readdata$hours,breaks=c(0,10,30,40,50,168),
                include.highest=TRUE,right=FALSE)
table(hourscut)
```

```
## hourscut
##   [0,10)  [10,30)  [30,40)  [40,50) [50,168)
##    14637     2799     2920     7468     1646
```

```r
hoursfactor <- relevel(hourscut,"[0,10)")
table(hoursfactor)
```

```
## hoursfactor
##   [0,10)  [10,30)  [30,40)  [40,50) [50,168)
##    14637     2799     2920     7468     1646
```

```r
### Fit a basic  version of your model
#assuming uniform prors for beta parameter.  You need to replace
#the ... after ~ with your chosen model for the covariates. Check
# the glm syntax if you need to (?glm). If your outcome variable is something
#Other than highincome then that also needs to be replaced in the call to glm

#
###basic models  assuming uniform prors for beta parameter
logitmodel <- glm(highincome ~ hoursfactor + sexfactor +
                    qualfactor,family=binomial(link="logit") )

summary(logitmodel)
```

```
##
## Call:
## glm(formula = highincome ~ hoursfactor + sexfactor + qualfactor,
##     family = binomial(link = "logit"))
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.4466  -0.5945  -0.3321  -0.2685   2.6118
##
## Coefficients:
##                      Estimate Std. Error z value Pr(>|z|)
## (Intercept)          -3.02244    0.05531 -54.642  < 2e-16 ***
## hoursfactor[10,30)    0.50728    0.07794   6.508 7.61e-11 ***
## hoursfactor[30,40)    1.57825    0.05934  26.595  < 2e-16 ***
## hoursfactor[40,50)    1.66099    0.04661  35.633  < 2e-16 ***
## hoursfactor[50,168)   2.66743    0.06307  42.294  < 2e-16 ***
## sexfactor2           -0.35466    0.03734  -9.499  < 2e-16 ***
## qualfactor1           0.07248    0.05959   1.216    0.224
## qualfactor3           0.32169    0.05271   6.103 1.04e-09 ***
## qualfactor4           0.96864    0.05390  17.971  < 2e-16 ***
## qualfactor5           0.42156    0.07983   5.281 1.29e-07 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 24500  on 29469  degrees of freedom
## Residual deviance: 20900  on 29460  degrees of freedom
##   (1 observation deleted due to missingness)
## AIC: 20920
##
## Number of Fisher Scoring iterations: 5
```

```
display(logitmodel)
```

```
## glm(formula = highincome ~ hoursfactor + sexfactor + qualfactor,
##     family = binomial(link = "logit"))
##                      coef.est coef.se
## (Intercept)          -3.02     0.06
## hoursfactor[10,30)    0.51     0.08
## hoursfactor[30,40)    1.58     0.06
## hoursfactor[40,50)    1.66     0.05
## hoursfactor[50,168)   2.67     0.06
## sexfactor2           -0.35     0.04
## qualfactor1           0.07     0.06
## qualfactor3           0.32     0.05
## qualfactor4           0.97     0.05
## qualfactor5           0.42     0.08
## ---
##   n = 29470, k = 10
##   residual deviance = 20899.7, null deviance = 24499.6 (difference = 3599.9)
```

```
##see what the covariance matrix looks like
vcov(logitmodel)
```

```
##                      (Intercept) hoursfactor[10,30) hoursfactor[30,40)
## (Intercept)           0.003059595      -1.229195e-03      -1.249327e-03
## hoursfactor[10,30)  -0.001229195       6.075317e-03       1.479753e-03
## hoursfactor[30,40)  -0.001249327       1.479753e-03       3.521736e-03
## hoursfactor[40,50)  -0.001428943       1.388649e-03       1.390640e-03
## hoursfactor[50,168) -0.001540927       1.349618e-03       1.355418e-03
## sexfactor2          -0.000648049      -2.840374e-04      -2.492331e-04
## qualfactor1         -0.001822761       1.098990e-04       1.055694e-04
## qualfactor3         -0.001695912      -7.916709e-05      -6.247676e-05
## qualfactor4         -0.001611610      -1.811269e-04      -1.715305e-04
## qualfactor5         -0.001714780      -1.687519e-05      -3.747820e-05
##                     hoursfactor[40,50) hoursfactor[50,168)    sexfactor2
## (Intercept)              -1.428943e-03       -1.540927e-03 -6.480490e-04
## hoursfactor[10,30)        1.388649e-03        1.349618e-03 -2.840374e-04
## hoursfactor[30,40)        1.390640e-03        1.355418e-03 -2.492331e-04
## hoursfactor[40,50)        2.172871e-03        1.437798e-03  1.615546e-04
## hoursfactor[50,168)       1.437798e-03        3.977579e-03  2.687971e-04
## sexfactor2                1.615546e-04        2.687971e-04  1.394000e-03
## qualfactor1               9.675295e-05        1.193966e-04  3.058381e-05
## qualfactor3              -1.111677e-04       -3.141968e-05  6.280221e-05
## qualfactor4              -1.386672e-04        9.785551e-06 -6.896867e-05
## qualfactor5              -4.346861e-05       -1.087340e-06  1.649586e-05
##                       qualfactor1   qualfactor3   qualfactor4   qualfactor5
## (Intercept)         -1.822761e-03 -1.695912e-03 -1.611610e-03 -1.714780e-03
## hoursfactor[10,30)   1.098990e-04 -7.916709e-05 -1.811269e-04 -1.687519e-05
## hoursfactor[30,40)   1.055694e-04 -6.247676e-05 -1.715305e-04 -3.747820e-05
## hoursfactor[40,50)   9.675295e-05 -1.111677e-04 -1.386672e-04 -4.346861e-05
## hoursfactor[50,168)  1.193966e-04 -3.141968e-05  9.785551e-06 -1.087340e-06
## sexfactor2           3.058381e-05  6.280221e-05 -6.896867e-05  1.649586e-05
## qualfactor1          3.550481e-03  1.728839e-03  1.723570e-03  1.730422e-03
## qualfactor3          1.728839e-03  2.778333e-03  1.735850e-03  1.735070e-03
## qualfactor4          1.723570e-03  1.735850e-03  2.905267e-03  1.734567e-03
## qualfactor5          1.730422e-03  1.735070e-03  1.734567e-03  6.373007e-03
```

```
bayesmodel1 <- bayesglm(highincome ~ hoursfactor + sexfactor +
                          qualfactor,family=binomial(link="logit") )

display(bayesmodel1)
```

```
## bayesglm(formula = highincome ~ hoursfactor + sexfactor + qualfactor,
##     family = binomial(link = "logit"))
##                     coef.est coef.se
## (Intercept)         -3.02     0.06
## hoursfactor[10,30)   0.50     0.08
## hoursfactor[30,40)   1.58     0.06
## hoursfactor[40,50)   1.66     0.05
## hoursfactor[50,168)  2.66     0.06
## sexfactor2          -0.35     0.04
## qualfactor1          0.07     0.06
## qualfactor3          0.32     0.05
## qualfactor4          0.97     0.05
## qualfactor5          0.42     0.08
## ---
## n = 29470, k = 10
## residual deviance = 20899.7, null deviance = 24499.6 (difference = 3599.9)
```

```r
##re-fit  controlling  for occupation
logitmodel2 <- glm(highincome ~ hoursfactor + sexfactor +
                     qualfactor + occfactor,
                   family=binomial(link="logit") )

#summary(logitmodel2)
display(logitmodel2)
```

```
## glm(formula = highincome ~ hoursfactor + sexfactor + qualfactor +
##     occfactor, family = binomial(link = "logit"))
##                     coef.est coef.se
## (Intercept)         -1.31     0.07
## hoursfactor[10,30)  -0.46     0.08
## hoursfactor[30,40)   0.62     0.07
## hoursfactor[40,50)   0.71     0.05
## hoursfactor[50,168)  1.72     0.07
## sexfactor2          -0.29     0.04
## qualfactor1          0.16     0.06
## qualfactor3          0.20     0.05
## qualfactor4          0.44     0.06
## qualfactor5          0.31     0.08
## occfactor2          -0.11     0.05
## occfactor3          -0.87     0.07
## occfactor4          -1.30     0.09
## occfactor5          -0.93     0.08
## occfactor6          -1.01     0.08
## occfactor7          -0.99     0.09
## occfactor8          -1.52     0.08
## occfactor9          -1.48     0.63
## occfactor10         -2.96     0.10
## ---
##   n = 29470, k = 19
##   residual deviance = 19296.4, null deviance = 24499.6 (difference = 5203.2)
```

```r
###illustrate how glm deals with factors
##extract model matrix -- internal glm representation of the model structure
chkX <- model.matrix(logitmodel)
head(chkX)
```

```
##   (Intercept) hoursfactor[10,30) hoursfactor[30,40) hoursfactor[40,50)
## 1           1                  0                  1                  0
## 2           1                  0                  0                  0
## 3           1                  1                  0                  0
## 4           1                  0                  0                  0
## 5           1                  0                  1                  0
## 6           1                  1                  0                  0
##   hoursfactor[50,168) sexfactor2 qualfactor1 qualfactor3 qualfactor4
## 1                   0          1           0           1           0
## 2                   0          1           1           0           0
## 3                   0          1           0           1           0
## 4                   0          1           0           0           0
## 5                   0          1           0           0           1
## 6                   0          1           1           0           0
##   qualfactor5
## 1           0
## 2           0
## 3           0
## 4           0
## 5           0
## 6           0
```

```
##Should really control for age also; income changes with age and
#age is also #related to quals

##re-fit  controlling  for occupation and age
logitmodel3 <- glm(highincome ~ hoursfactor + sexfactor +
                     qualfactor + occfactor +agefactor,
                   family=binomial(link="logit") )

summary(logitmodel3)
```

```
##
## Call:
## glm(formula = highincome ~ hoursfactor + sexfactor + qualfactor +
##     occfactor + agefactor, family = binomial(link = "logit"))
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.7130  -0.5904  -0.2187  -0.1199   3.4078
##
## Coefficients:
##                      Estimate Std. Error z value Pr(>|z|)
## (Intercept)         -0.891298   0.084653 -10.529  < 2e-16 ***
## hoursfactor[10,30)  -0.370929   0.083712  -4.431 9.38e-06 ***
## hoursfactor[30,40)   0.718255   0.066755  10.760  < 2e-16 ***
## hoursfactor[40,50)   0.789141   0.053998  14.614  < 2e-16 ***
## hoursfactor[50,168)  1.813919   0.071013  25.543  < 2e-16 ***
## sexfactor2          -0.317168   0.040876  -7.759 8.54e-15 ***
## qualfactor1         -0.006148   0.064174  -0.096  0.92367
## qualfactor3         -0.007392   0.056604  -0.131  0.89610
## qualfactor4          0.282402   0.061270   4.609 4.04e-06 ***
## qualfactor5          0.084274   0.084499   0.997  0.31860
## occfactor2          -0.090545   0.055106  -1.643  0.10036
## occfactor3          -0.854089   0.068823 -12.410  < 2e-16 ***
## occfactor4          -1.315746   0.087537 -15.031  < 2e-16 ***
## occfactor5          -0.978108   0.076708 -12.751  < 2e-16 ***
## occfactor6          -0.992900   0.084085 -11.808  < 2e-16 ***
## occfactor7          -1.010010   0.093423 -10.811  < 2e-16 ***
## occfactor8          -1.521534   0.085242 -17.850  < 2e-16 ***
## occfactor9          -1.418897   0.636817  -2.228  0.02587 *
## occfactor10         -2.824018   0.103850 -27.193  < 2e-16 ***
## agefactor15         -1.764882   0.174267 -10.127  < 2e-16 ***
## agefactor20         -1.407822   0.097738 -14.404  < 2e-16 ***
## agefactor25         -0.898490   0.083843 -10.716  < 2e-16 ***
## agefactor30         -0.463247   0.078323  -5.915 3.33e-09 ***
## agefactor35         -0.011822   0.074081  -0.160  0.87321
## agefactor45         -0.095715   0.073016  -1.311  0.18990
## agefactor50         -0.034945   0.073904  -0.473  0.63632
## agefactor55         -0.225836   0.079658  -2.835  0.00458 **
## agefactor60         -0.170303   0.082538  -2.063  0.03908 *
## agefactor65         -0.122871   0.091067  -1.349  0.17726
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 24500  on 29469  degrees of freedom
## Residual deviance: 18745  on 29441  degrees of freedom
##   (1 observation deleted due to missingness)
## AIC: 18803
##
## Number of Fisher Scoring iterations: 7
```

```
display(logitmodel3)
```

```
## glm(formula = highincome ~ hoursfactor + sexfactor + qualfactor +
##     occfactor + agefactor, family = binomial(link = "logit"))
##                     coef.est coef.se
## (Intercept)         -0.89     0.08
## hoursfactor[10,30)  -0.37     0.08
## hoursfactor[30,40)   0.72     0.07
## hoursfactor[40,50)   0.79     0.05
## hoursfactor[50,168)  1.81     0.07
## sexfactor2          -0.32     0.04
## qualfactor1         -0.01     0.06
## qualfactor3         -0.01     0.06
## qualfactor4          0.28     0.06
## qualfactor5          0.08     0.08
## occfactor2          -0.09     0.06
## occfactor3          -0.85     0.07
## occfactor4          -1.32     0.09
## occfactor5          -0.98     0.08
## occfactor6          -0.99     0.08
## occfactor7          -1.01     0.09
## occfactor8          -1.52     0.09
## occfactor9          -1.42     0.64
## occfactor10         -2.82     0.10
## agefactor15         -1.76     0.17
## agefactor20         -1.41     0.10
## agefactor25         -0.90     0.08
## agefactor30         -0.46     0.08
## agefactor35         -0.01     0.07
## agefactor45         -0.10     0.07
## agefactor50         -0.03     0.07
## agefactor55         -0.23     0.08
## agefactor60         -0.17     0.08
## agefactor65         -0.12     0.09
## ---
##   n = 29470, k = 29
##   residual deviance = 18745.1, null deviance = 24499.6 (difference = 5754.5)
```

```
###Specify priors:
####################################################################
##Analysis with an informative prior
###################################################################




##obtain priors for parameters of logitmodel
##using the makepriorfunctions makepriorb0() and makepriorbreg() read in from
#logisticregression_functions.r

##Intercept - specify prior interval on the probability scale
b0prior <- makepriorb0(low=0.01,high=0.10,credibility=0.95)
b0prior
```

```
## [1] -3.3961722  0.6117192
```

```
##logistic regression parameters
##specify prior on odds ratio scale; makepriorreg converts to prior
#on the beta scale
bfemale   <- makepriorbreg(low=0.5,high=1.1,credibility=0.95)
bqualnone <- makepriorbreg(low=0.4,high=2,credibility=0.95)
bqualtrade <- makepriorbreg(low=0.7,high=3,credibility=0.95)
bqualuni <- makepriorbreg(low=0.7,high=4,credibility=0.95)
bqualother <- makepriorbreg(low=0.1,high=10,credibility=0.95)
bhours10_30 <- makepriorbreg(low=1,high=5,credibility=0.9)
bhours30_40 <- makepriorbreg(low=1,high=5,credibility=0.9)
bhours40_50 <- makepriorbreg(low=1,high=5,credibility=0.9)
bhours50pl <- makepriorbreg(low=1.1,high=10,credibility=0.9)

###add lines to set priors for any additional  parameters
###check  a few of these
bfemale
```

```
## [1] -0.2989185  0.2011408
```

```
bqualuni
```

```
## [1] 0.5148097 0.4446432
```

```
##combine prior means into a vector and construct a
#diagional prior variance matrix
##based on the prior  standard deviations prior.matrix

#First put everything in matrix - column1= priormeans,
#column 2 = prior sd
prior.matrix<- rbind(b0prior,bhours10_30,bhours30_40,bhours40_50,
                     bhours50pl,bfemale,bqualnone,bqualtrade,
                     bqualuni,bqualother)
prior.matrix
```

```
##                      [,1]      [,2]
## b0prior     -3.396172e+00 0.6117192
## bhours10_30  8.047190e-01 0.4892344
## bhours30_40  8.047190e-01 0.4892344
## bhours40_50  8.047190e-01 0.4892344
## bhours50pl   1.198948e+00 0.6709639
## bfemale     -2.989185e-01 0.2011408
## bqualnone   -1.115718e-01 0.4105784
## bqualtrade   3.709687e-01 0.3712536
## bqualuni     5.148097e-01 0.4446432
## bqualother   2.220446e-16 1.1748099
```

```
prior_mean_inf <- prior.matrix[,1]    #prior mean vector
prior_variance_inf <- diag((prior.matrix[,2])^2)  #prior variance matrix

prior_mean_inf
```

```
##       b0prior   bhours10_30   bhours30_40   bhours40_50    bhours50pl
## -3.396172e+00  8.047190e-01  8.047190e-01  8.047190e-01  1.198948e+00
##       bfemale     bqualnone    bqualtrade      bqualuni    bqualother
## -2.989185e-01 -1.115718e-01  3.709687e-01  5.148097e-01  2.220446e-16
```

```
prior_variance_inf
```

```
##            [,1]      [,2]      [,3]      [,4]      [,5]       [,6]      [,7]
##  [1,] 0.3742004 0.0000000 0.0000000 0.0000000 0.0000000 0.00000000 0.0000000
##  [2,] 0.0000000 0.2393503 0.0000000 0.0000000 0.0000000 0.00000000 0.0000000
##  [3,] 0.0000000 0.0000000 0.2393503 0.0000000 0.0000000 0.00000000 0.0000000
##  [4,] 0.0000000 0.0000000 0.0000000 0.2393503 0.0000000 0.00000000 0.0000000
##  [5,] 0.0000000 0.0000000 0.0000000 0.0000000 0.4501926 0.00000000 0.0000000
##  [6,] 0.0000000 0.0000000 0.0000000 0.0000000 0.0000000 0.04045761 0.0000000
##  [7,] 0.0000000 0.0000000 0.0000000 0.0000000 0.0000000 0.00000000 0.1685747
##  [8,] 0.0000000 0.0000000 0.0000000 0.0000000 0.0000000 0.00000000 0.0000000
##  [9,] 0.0000000 0.0000000 0.0000000 0.0000000 0.0000000 0.00000000 0.0000000
## [10,] 0.0000000 0.0000000 0.0000000 0.0000000 0.0000000 0.00000000 0.0000000
##            [,8]      [,9]    [,10]
##  [1,] 0.0000000 0.0000000 0.000000
##  [2,] 0.0000000 0.0000000 0.000000
##  [3,] 0.0000000 0.0000000 0.000000
##  [4,] 0.0000000 0.0000000 0.000000
##  [5,] 0.0000000 0.0000000 0.000000
##  [6,] 0.0000000 0.0000000 0.000000
##  [7,] 0.0000000 0.0000000 0.000000
##  [8,] 0.1378292 0.0000000 0.000000
##  [9,] 0.0000000 0.1977076 0.000000
## [10,] 0.0000000 0.0000000 1.380178
```

```
#############################################################
#Next step is build an approximation to the posterior
#which we can use to generate starting values for the Markov Chains
#We can also base the variance of the jumping density on an
#approximation to the posterior variance
#We will use a multivariate normal approximation centred on the
# posterior model with variance determined by the curvature of the
#log-posterior at the mode. The posterior mode and approximate variance
#can be obtained using either the laplace() function or
#bayesglm(), given appropriate specification of the prior in both cases )

##test out the log posterior function
bmle <- coef(logitmodel)  #For convenience evaluate the unnormalised
#log posterior at the mle of the basic model
#called logitmodel
testpost <- logpost_logit_norm(beta=bmle,
                               prior_mean=prior_mean_inf,
                               prior_variance=prior_variance_inf,
                               model=logitmodel)
testpost
```

```
## [1] -10458.15
```

```
##use laplace  to find the posterior mode and a variance approximation
## first argument passed to laplace is the name of the log-posterior function,
#then an initial guess at the mode --suggest using the mle from the glm);
#then all the arguments required by the log posterior function need to be
#explicitly passed to laplace()

laplace_post <- laplace(logpost_logit_norm,mode=bmle,
                        prior_mean=prior_mean_inf,
                        prior_variance=prior_variance_inf,
                        model=logitmodel)
##in the above call to laplace() the parameter mode just specified
##the starting values for the optimisation procedure; ideally an
##informed guessfor the posterior mode would be supplied
laplace_post$converge      ##This should equal TRUE otherwise
```

```
## [1] TRUE
```

```
##there is a problem.  For a complex model the
##solution may simply be to increase the number
##of iterations in laplace.  This can achieved
##using the mylaplace() function in place of
##laplace.This has the same syntax, except
##for an additional
# parameter, maxiter. The default maximum number of iterations
#in mylaplace is set to
#1000, in contrast to laplace in which the maximium
#number of  iterations is set to 500.

mean_approx <- laplace_post$mode
var_approx <- laplace_post$var
sd_approx <- sqrt(diag(var_approx))
###display mode and  approximation to variance
cbind(mean_approx,sd_approx)
```

```
##                      mean_approx  sd_approx
## (Intercept)         -3.00997705 0.05412000
## hoursfactor[10,30)   0.50787063 0.07655976
## hoursfactor[30,40)   1.56615282 0.05865690
## hoursfactor[40,50)   1.65350144 0.04603601
## hoursfactor[50,168)  2.65393987 0.06248255
## sexfactor2          -0.35184188 0.03665130
## qualfactor1          0.06360094 0.05850864
## qualfactor3          0.31373853 0.05171067
## qualfactor4          0.96093295 0.05297908
## qualfactor5          0.41351140 0.07919749
```

```
##Obtain over-dispersed approx to variance

vstart <- 2*var_approx

#Obtain jumping variance
jumpcov <- (2.4^2/length(bmle)) * var_approx   ## as per Gelman et al
###Set-up the M-H procedure

###decide on number of chains and length of the chains

nchains <- 3 ##number of chains must be greater 1 but setting it too
##too high will slow computations between 3 and 5 is usually
##sufficient

nsim <- 1250 ##simulation sample size

###Specify  array to store sampled values

storebeta <- array(NA,dim=c(length(mean_approx),nchains,nsim))
##For each parameter we have a nchains by nsims matrix
##To pull out the simulated points for a parameter k use storebeta[k,,];
#i.e storek <- storebeta[k,,].

#nchains by nsim matrix of acceptance indicators
##use code like bksims <- storebeta[k,,]
storeaccept <- matrix(NA,nrow=nsim,ncol=nchains)

###Start Metropolis-Hastings computations
system.time (  #not necessary to enclose the M-H within a system.time
  #call but is of interest in the set-up phase
  for (j in 1:nchains) {
    ##draw starting value for this chain e.g use rmvnorm to draw from
    #a multivariate normal approximation
    ##

    oldbeta <- rmvnorm(n=1,mean=mean_approx,sigma=vstart)

    for (i in 1:nsim) {
      storebeta[,j,i] <- oldbeta  ##assuming the current  draw is called oldbeta;

      #generate a proposal from the jumping distribution - make this symmetric e,g
      #rmvnorm, with variance set to jumpcov

      newbeta <- rmvnorm(n=1,mean=oldbeta,sigma=jumpcov)

      ##if we are using a symmetric proposal distribution so the acceptance ratio is
 just
      ## the ratio of the posterior at newbeta to the posterior at oldbeta
      #on the logscale this is just the difference in the log posterior

      ##evaluate unnormalized posterior at proposal


      logpost_prop <- logpost_logit_norm(beta=newbeta,
                                         prior_mean=prior_mean_inf,
                                         prior_variance=prior_variance_inf,
                                         model=logitmodel)
```

```
      ##evaluate unnormalized posterior at current value


      logpost_old <- logpost_logit_norm(beta=oldbeta,
                                        prior_mean=prior_mean_inf,
                                        prior_variance=prior_variance_inf,
                                        model=logitmodel)

      ##compute  MH acceptance ratio or log MH acceptance ratio

      logrMH <- logpost_prop - logpost_old

      #decide whether to accept the proposal: create an acceptance indicator
      #called accept
      logU <- log(runif(1))
      accept   <- (logU < logrMH)

      #Assuming acceptance ratio is called accept
      storeaccept[i,j] <- as.numeric(accept)

      ###If jump accepted update  oldbeta to newbeta
      if (accept) {
        oldbeta <- newbeta
      }
      ##else just stay at curent value of oldbeta
    } ###end loop over simulations

  } ###end loop over chains
) #end(system.time)
```

```
##    user  system elapsed
##  26.504   0.204  26.713
```

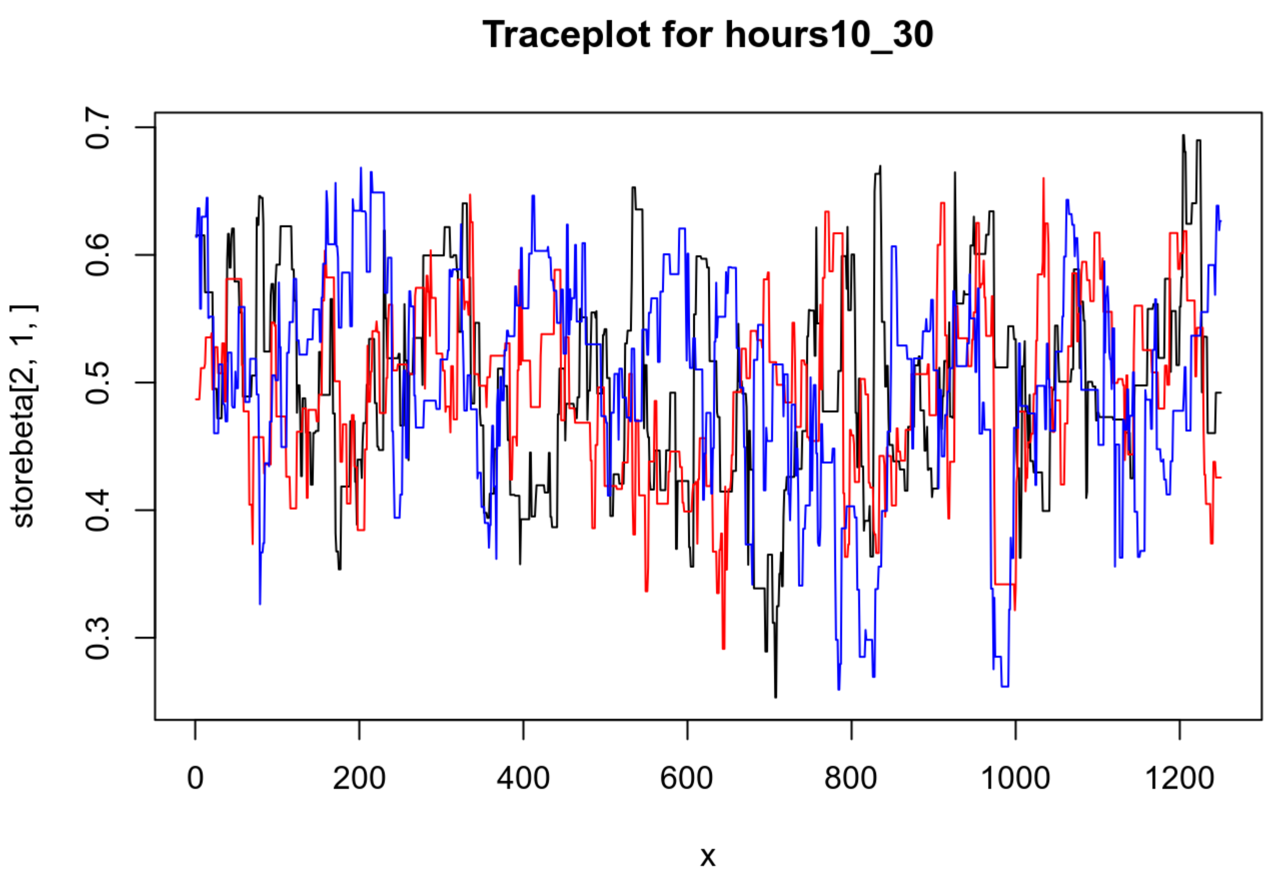
```r
#Rhat statistics

##Simply loop over the parameters and carry out the computations
# as per Gelman et al

#decide on burn-in period

burnin <- 250
npos <- nsim - burnin  ##size of retained posterior sample in each chain

#set-up a vector for storing the Rhat statistics
rhat <- vector(mode="numeric",length=length(mean_approx))

#set-up a vector for storing the effective sample sizes
neff <- vector(mode="numeric",length=length(mean_approx))

for (k in 1:length(bmle)) {
  ##subsets out values for ith parameter

  betak <- storebeta[k,1:nchains,((burnin+1):nsim)]  ##subsets out post burnin sample
for kth parameter
  chainmeans <- rowMeans(betak)
  withinsd <- apply(betak,MARGIN=1,FUN="sd")
  betweensd <- sd(chainmeans)
  B = npos*betweensd^2
  W = mean(withinsd^2)
  varplus <- ((npos-1)/npos) * W + (1/npos)*B
  rhat[k] <- sqrt(varplus/W)
  neff[k] <- nchains*npos*(varplus/B)
}
rhat
```

```
##  [1] 1.0073349 0.9996836 1.0150601 1.0020650 1.0208741 1.0236054 1.0279172
##  [8] 1.0712192 1.0268253 1.0434332
```

```r
neff  ##Some of these are concerningly low - we would probably want to
```

```
##  [1]  193.60475 8163.18265   98.60730  586.72998   72.40098   64.45433
##  [7]   55.01874   23.18039   57.12667   36.39167
```

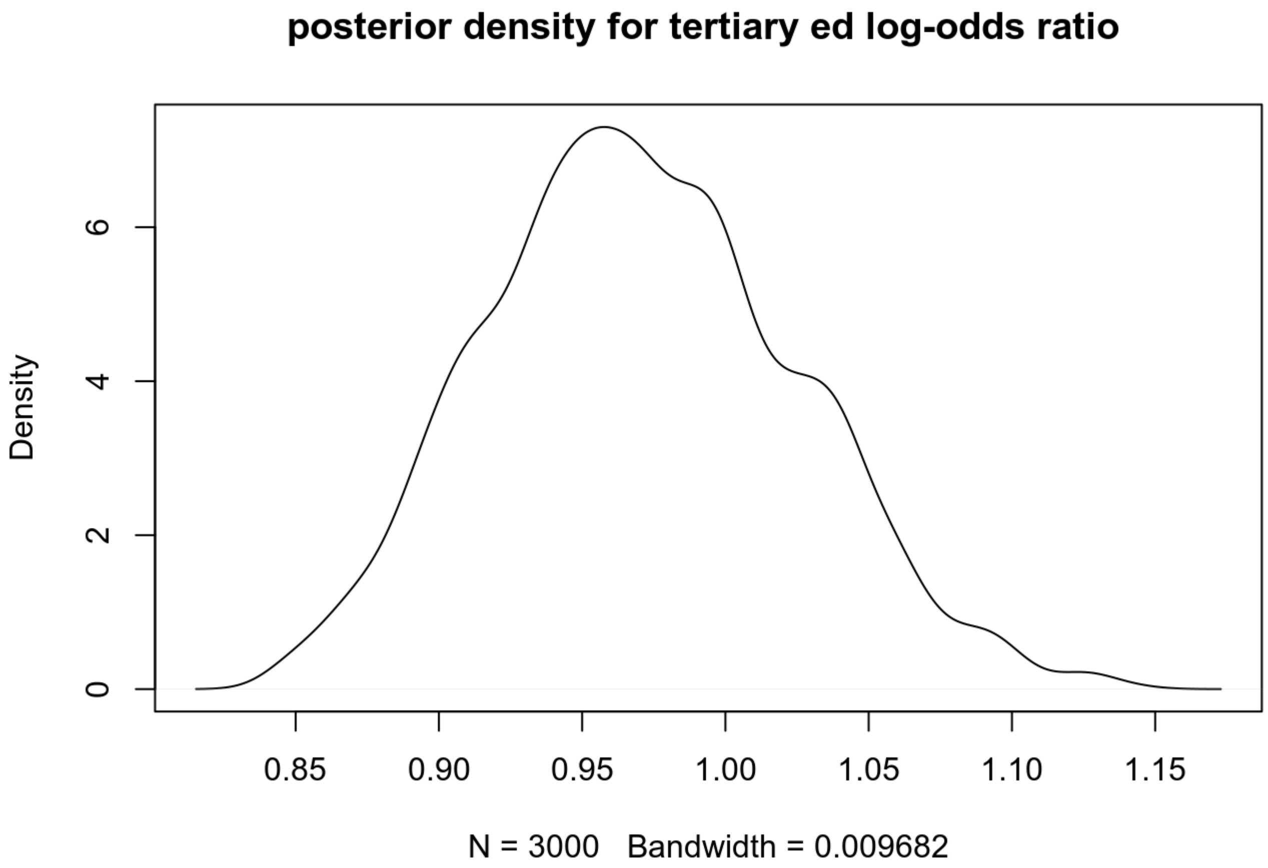
```
##increase the posterior sample size in reality


##Traceplots
##useful but optional for assignment   use code like this
##Assumes 3 chains and looking at just the 2nd  parameter
##Really needs to be repeated for each parameter.

##You may also  want to  exclude the  burnin
x <- seq(from=1,to=nsim,by=1)

plot(x,storebeta[2,1,],col="black",type="l",
     main="Traceplot for hours10_30")
lines(x,storebeta[2,2,],col="red")
lines(x,storebeta[2,3,],col="blue")
```



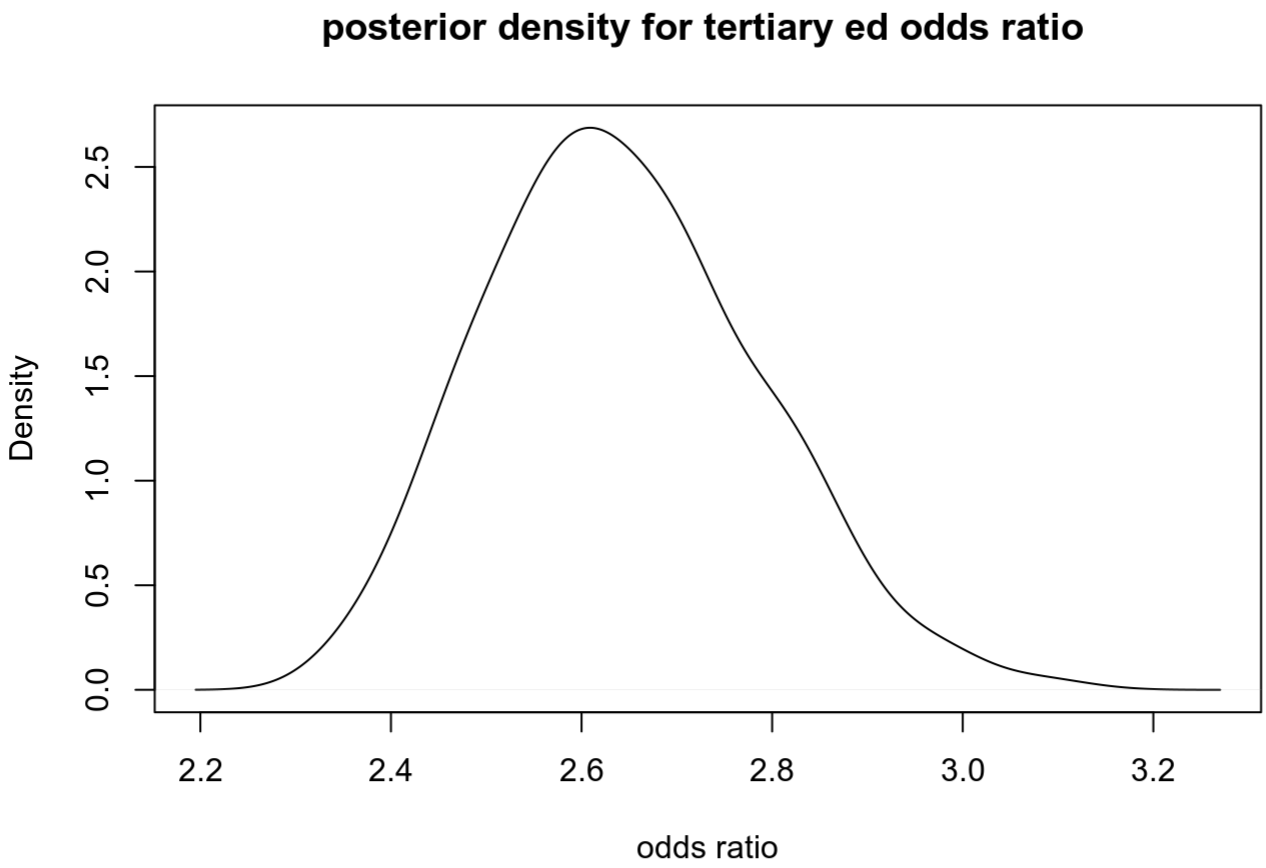
```
###If satisfied about convergence  use the post burn-in sample for posterior inferenc
e

possamp <- storebeta[,,(burnin+1):nsim]  # this is our sample from the posterior

##produce posterior density plot for at least one of the logistic regression coeffici
ents
##credible interval; posterior probability > 0.
##To subset out the  posterior for a single
##parameters use code like  pos_betak <- possamp[k,,];
#Functions like summary() and quantile()  can then be applied
##to pos_betak and work happily even though it is an
#array rather than a vector

##Up to the user to know which parameters are of particular interest
pos_tertiary <- possamp[9,,]

mean(pos_tertiary > 0)
```

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

```
plot(density(pos_tertiary),main="posterior density for tertiary ed log-odds ratio ")
```




```
##back on the odds ratio scale
plot(density(exp(pos_tertiary),adjust=1.7),
     main="posterior density for tertiary ed odds ratio ",
     xlab="odds ratio")
```




```
pos_tert <- possamp[9,,]
####90% credible interval for tertiary education odds ratio
quantile(exp(pos_tertiary),probs=c(0.05,0.5,0.95))
```

```
##       5%      50%      95%
## 2.433730 2.631385 2.883247
```

```
##clearly seems to be a fairly strong tertiary education effect - but recall this
##is without adjusting for age or occupation.
```