Lisa Over
HW 9
Newton-Raphson
Logistic Regression with Metropolis

Newton-Raphson Method

Code

```
nextX <- function(x) {
 m=3
 c=5
 x = x*(1 - 1/m + c/(m*x^m))
 return(x)
}

newt_raphson <- function(x.i, x.next, condition) {
 x.i = x.next
 x.next = nextX(x.i)
 if(abs(x.i - x.next) <= condition) {
	return(x.next)
 }
 else {
	newt_raphson(x.i, x.next, condition)
 }
}

x0 = 5
condition = 0.000000001
root = newt_raphson(x0, nextX(x0), condition)
```

Results

```
#From Newton-Raphson
root
#[1] 1.709976


5^(1/3)
#[1] 1.709976
```

Lisa Over
HW 9
Newton-Raphson
Logistic Regression with Metropolis

Logistic Regression with Metropolis

Code

#printPDF function receives two vectors and a filename. It prints three plots for each vector to the specified filename: ts plot, hist, acf.

```
printPDF <- function(v1,v2,filename) {

  pdf(filename)
   #PLOTS to PDF
        par(mfrow=c(3,2)) #split plotting window into 3 rows and 2 columns
        ts.plot(v1,xlab="Iterations")
        ts.plot(v2,xlab="Iterations")
        hist(v1,probability=T, cex.lab=1.5, cex.axis=1.5)
        hist(v2,probability=T, cex.lab=1.5, cex.axis=1.5)
        acf(v1, lag.max=500)
   acf(v2, lag.max=500)
  dev.off()

}
```

#metro function receives nine parameters: two vectors "x.s" and “y.s” for the data values of interest, "a0" for the initial alpha, "b0" for the initial beta, "k" for the alpha interval, "c" for the beta interval, "N" for number of independent random normal realizations, "lag" for determining how many realizations to skip between saves, and "burnin" for determining how many realizations to skip before starting to save.

```
metro <- function(x.s,y.s,a0,b0,k,c,N,lag,burnin) {

        #Set N to be N*lag+burnin
        N <- N*lag + burnin

        #Initialize vectors to hold alpha (a.v) and beta (b.v) values
        a.s <- NULL
        b.s <- NULL

        #store the acceptance rate for alpha (a.cnt) and beta (b.cnt)
        a.cnt = 0
        b.cnt = 0

 for(i in 1:N) {
```

Lisa Over
HW 9
Newton-Raphson
Logistic Regression with Metropolis

```
#Generate an alpha star 'a.star' from the proposal density (Normal) using a0 as mu
and k as std dev
	a.star = rnorm(1,a0,k)

#Compute probability for alpha using Normal priors with mean (0), variance (100) -
- Use full conditional (from joint posterior which is
likelihood*prior(alpha)*prior(beta)) -- compute ratio of full conditional given a.star
to the full conditional given a0
	numerator = sum(y.s*a.star) - sum(log(1 + exp(a.star + b0*x.s))) -
a.star^2/200
	denominator = sum(y.s*a0) - sum(log(1 + exp(a0 + b0*x.s))) - a0^2/200
	target.ratio = exp(numerator - denominator)

	#Use target.ratio to determine if a.star should be accepted
	param = 0
	accept.code = 0
	if(target.ratio < 1) {
		if(runif(1,0,1) < target.ratio) {
			param = a.star
			accept.code = 1
		}
		else {
			param = a0
			accept.code = 0
		}
	}
	else {
		param = a.star
		accept.code = 1
	}

#if i is greater than burnin and if i is a multiple of the lag, store value
#add accept.code to total a.cnt - accept.code will be 1 if a.star was accepted and 0 ow
#add accepted value to vector - a0 or a.star - stored as param
	if(i > burnin) {
	if(i %% lag == 0) {
		a.cnt = a.cnt + accept.code
		a.s = c(a.s, param)
		#set a0 equal to param (parameter that was stored in vector -- a.star
or a0) for next iteration
		a0 = param
  }
```

```
  }

#Reset target.ratio, numerator, denominator
numerator = 0
denominator = 0
target.ratio = 0
#Generate a beta star 'b.star' from the proposal density (Normal) using b0 as mu
and c as std dev
	b.star = rnorm(1,b0,c)

#Compute probability for beta using Normal priors with mean (0), variance (100) --
Use full conditional (from joint posterior which is
likelihood*prior(alpha)*prior(beta)) -- compute ratio of full conditional given b.star
to the full conditional given b0
	numerator = sum(y.s*b.star*x.s) - sum(log(1 + exp(a.star + b.star*x.s))) -
b.star^2/200
	denominator = sum(y.s*b0*x.s) - sum(log(1 + exp(a.star + b0*x.s))) -
b0^2/200
	target.ratio = exp(numerator - denominator)

	#Use target.ratio to determine if b.star should be accepted
	param = 0
	accept.code = 0
	if(target.ratio < 1) {
		if(runif(1,0,1) < target.ratio) {
			param = b.star
			accept.code = 1
		}
		else {
			param = b0
			accept.code = 0
		}
	}
	else {
		param = b.star
		accept.code = 1
	}

#if i is greater than burnin and if i is a multiple of the lag, store value
#add accept.code to total b.cnt - accept.code will be 1 if b.star was accepted and 0
ow
#add accepted value to vector - b0 or b.star - stored as param
	if(i > burnin) {
```

Lisa Over
HW 9
Newton-Raphson
Logistic Regression with Metropolis

```
    if(i %% lag == 0) {
            b.cnt = b.cnt + accept.code
            b.s = c(b.s, param)
            #set b0 equal to param (parameter that was stored in vector -- b.star
or b0) for next iteration
            b0 = param
   }
  }
}

 filename = sprintf("Documents/R-FILES/HW9-lag-%s-k-%s-c-%s.pdf",lag,k,c)
 printPDF(a.s,b.s,filename)

 vectors <- list("alpha" = a.s, "aCount" = a.cnt, "beta" = b.s, "bCount" = b.cnt)
 return(vectors)
}

#Set metro parameters
N = 5000
lag = 500
burnin = 0
k=.5
c=.011
a0 = -3.82
b0 = .086
x.s = c(21,24,25,26,28,31,33,34,35,37,43,49,51,55,25,29,43,44,46,46,51,55,56,58)
y.s = c(0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,1,1,1,1,1,1,1,1)
#Obtain alpha and beta from data to use for a0 and b0
#output = summary(glm(y.s~x.s,family=binomial))

vectors = metro(x.s,y.s,a0,b0,k,c,N,lag,burnin)

#Obtain acceptance probabilities
alpha.accept = vectors$aCount/N
beta.accept = vectors$bCount/N
alpha.accept
#[1] 0.6864
beta.accept
#[1] 0.6946

#Obtain credible intervals for alpha and beta
quantile(vectors$alpha, 0.025)
#-9.030698
```

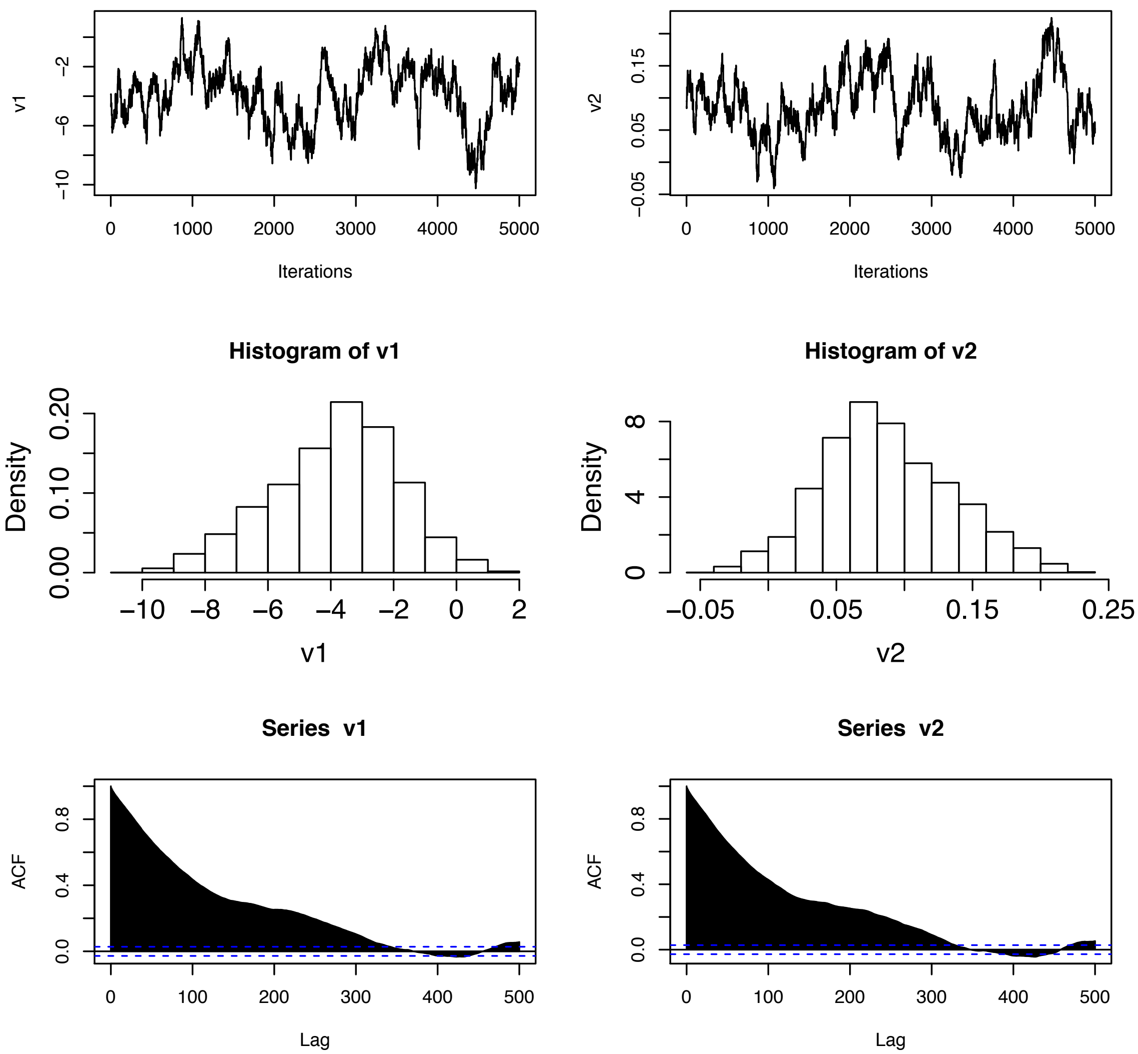
Lisa Over
HW 9
Newton-Raphson
Logistic Regression with Metropolis

```
quantile(vectors$alpha, 0.975)
#-0.1419091
quantile(vectors$beta, 0.025)
#-0.0021883
quantile(vectors$beta, 0.975)
#0.2096534

#Produce a scatterplot of x and y data with posterior mean curve and credible
bounds superposed on the plot
x = seq(min(x.s),max(x.s),length=250)
m <- matrix(, nrow = 5000, ncol = 250)
for(i in 1:250){
  vector = exp(vectors$alpha + vectors$beta*x[i])/(1+exp(vectors$alpha +
vectors$beta*x[i]))
  m[, i] <- vector
}
m.mean = apply(m, 2, mean)
q97.5 = apply(m, 2, quantile, probs=0.975)
q2.5 = apply(m, 2, quantile, probs=0.025)

plot(x.s,y.s)
lines(x, m.mean)
lines(x, q97.5)
lines(x, q2.5)
```

Lisa Over
HW 9
Newton-Raphson
Logistic Regression with Metropolis

Results

- $\pi\left(\alpha, \beta \middle| (\vec{x}, \vec{y})\right) = \frac{1}{200\pi} e^{\sum_{i=1}^{n} y_i \alpha + \sum_{i=1}^{n} y_i \beta x_i - \sum_{i=1}^{n} \ln\left(1 + e^{\alpha + \beta x_i}\right) - \frac{\alpha^2}{200} - \frac{\beta^2}{200}}$
- Trace Plots



Lisa Over
HW 9
Newton-Raphson
Logistic Regression with Metropolis

- Credible Intervals

  Credible Interval for α
  (-9.030698, -0.1419091)

  Credible Interval for β
  (-0.0021883, 0.2096534)

- Scatterplot with posterior mean curve and credible bounds superposed