EXERCISE ON LOGIT AND MCMC

On the course web site is a data set called "Names.csv" that contains results from a project that sent job applications in response to numerous newspaper job postings. The applications were for fictitious people. Some were sent with "average-sounding" names, while others were sent under names chosen to make them sound "black". The idea was to see whether the applications were more likely to get a call from the employer depending on the type of name chosen for the fictitious applicant.

We're going to use a small subset of this data set, just the variables `call_back, black`, `female`, and `chicago`. These are the main ones that proved to have explanatory power. They are all variables that take values in $\{0,1\}$. You can load the data into R as a data frame with the function **`read.csv`**`()`. Below are R programs that compute the logit likelihood and that generate Metropolis MCMC draws from a logit likelihood. The programs are also available as .R text files on the course web site. You could adapt them to matlab, Julia, or python if you prefer those frameworks. These R programs use a vector `y` of dependent variables and a matrix `x` of explanatory variables. You'll need to extract them from the dataframe.

If you haven't used R and want to try it, you can download it for free from the internet. It includes an introductory tutorial.

(1) Find the maximum likelihood estimate of the coefficients in a logit model that explains the probability of **`call`**`_back = 1` with a constant and the three variables `black, female, chicago`. Calculate standard errors for the coefficients based on the second derivative matrix of the log likelihood. If you use **`nlm`**`()` for the optimization, or most other gradient-based optimization programs, you will have a numerical approximation to the second derivative of the log likelihood in the output of the converged optimization program. You could also calculate a numerical second derivative approximation by other routes.

(2) Estimate a linear probability model for the same data. Are the t-statistics on the coefficients comparable? Note that the coefficients are not expected to be of the same size. Calculate the implied $P[y = 1]$ when `black`, `female` and `chicago` are all equal to 1, both for the linear probability model and the logit model. Do the same when these three variables are all equal to zero. Is there much difference between the LPM and the logit model on these estimates?

(3) Generate MCMC draws from the posterior distribution on the coefficients, along with the corresponding log likelihoods. Check convergence (e.g. with the `effectiveSize()` function from the `coda` package.). Also check trace plots. It probably requires around 10,000 draws to get convergence, which should take just a few seconds of computer time. For the jump proposal in the MCMC, $\beta \sim N(\beta, \Sigma)$ should work,

where Σ is either .01 times the identity or .3 times the inverse hessian at the likelihood maximum.

(4) Calculate the posterior means and standard deviations from your MCMC sample and compare to those derived from the MLE and the hessian at the MLE. Construct a scatter plot of the draws for the coefficients on `female` and `chicago`.
(5) Calculate the posterior means and standard deviations of $P[y = 1 \mid x]$ when all of `black`, `female` and `chicago` are one, and also when they are all zero.
(6) Since there are just three dummy variables and a constant in this system, there are only eight possible values for the vector of explanatory variable values. We can estimate the probability of $y = 1$ for each of these eight configurations by just dividing the number of positive values for `call_back` for that x vector by the number of observations with that x vector. Make these calculations. Do they suggest any problem with the restrictions implied by the logit model? Calculate the likelihood for this "fully saturated" model and compare it to that you found for the logit model. Use a frequentist likelihood ratio test, and also the BIC criterion (which we will have discussed in class Tuesday afternoon).

```
#' Logit log likelihood
#'
#' Log likelihood of a logit model from data and parameters
#'
#' Includes the gradient as an attribute of the returned log likelihood.
#'
#' @param b The coefficients on `x`.
#' @param y The dependent variable.  A single vector of zeros and ones.
#' @param x The explanatory variable matrix.  Its first dimension matches
#'          the length of y.
#'
#' @return Since this is going to be called by a minimizer function, it
#'         returns minus the log likelihood, and minus the derivative of
#'         the log likelihood as an attribute of the returned value. `csminwelNew(
#'         can use the gradient attribute to greatly improve speed and accuracy.
logitllh <- function(b, y, x) {
    p <- x %*% b
    p <- exp(p)
    p <- p / (1 + p)
    llh <- sum(y * log(p) + (1 - y) * log(1 - p))
    if(is.nan(llh)) llh <- -1e20
    attr(llh, "gradient") <- -c(c(y - p) %*% x)
    return(-llh)
}
#' @md
#' @export

logitMCMC <- function(b0, y, x, llh, jump, nit) {
    llh0 <- llh(b0, y, x)
    draws <- matrix(0, nit, dim(x)[[2]] + 1)
    draws[1, ] <- c(b0, llh0)
    for(it in 2:nit) {
        b1 <- jump(b0)
        llh1 <- llh(b1, y, x)
        accept <- exp(llh0 - llh1) > runif(1) #llh is minus the log likelihood
        if(is.na(accept)) accept <- FALSE
        if (accept) {
            draws[it, ] <- c(b1, llh1)
            b0 <- b1
            llh0 <- llh1
        } else {
            draws[it, ] <- c(b0, llh0)
        }
    }
    return(draws)
}
```