SIMPLE MCMC EXERCISE

Using the jump distribution in the example code, effective sample sizes calculated by `coda` for 1000, 10000, and 100000 draws are

```
> effectiveSize(mcmc(lacl1[1:1000, ]))
var1      var2      var3
6.460910  6.357981 18.115986
> effectiveSize(mcmc(lacl1[1:10000, ]))
var1      var2      var3
40.10646  53.98901 217.29245
> effectiveSize(mcmc(lacl1))
var1      var2      var3
439.8908  453.0319 1890.3198
>
```

The 1000 draw result is clearly unusable, the 10000 draw result is marginal, and the 10000 draw result looks respectable. The fact that the effective sample size increases roughly linearly with the actual number of draws suggests that we are getting convergence to the target distribution. The remaining question is whether we have the accuracy we need. Since any effective size measure is based on calculating the standard deviation of the mean, allowing for the serial correlation in the draws, we can calculate the standard deviation of the mean as the standard deviation of the MCMC draws divided by the square root of the effective sample size. For the 100000 draw case, we get as means and standard deviations

```
> rbind(eacl, apply(lacl1, 2, sd)/es)
             var1          var2          var3
eacl 1.110909e-01 -1.0069262267 -1.311276e+02
6.441952e-05  0.0006879993  5.276176e-04
```

So we have accuracy on the two parameters α and γ to four significant figures, which for most purposes is plenty.

With 10000 draws, we get instead

```
            var1        var2          var3
[1,] 0.109531254 -0.99284379 -131.18720834
[2,] 0.004032652  0.04208352    0.06566187
```

Here the MCMC standard errors are small relative to the coefficient values, and much smaller than the standard deviations of the parameters themselves (thus having only modest effects on *t*-statistics). But if (as in this model) the MCMC runs are not too costly, one would probably want to use longer runs.

As mentioned in the problem set, one could use an initial run of 1000 draws to estimate a normal approximation to the likelihood. Taking the covariance matrix of the 100000 draws

Date: April 14, 2018.

from the initial run, scaled by about a factor of .3, as the covariance matrix of a normal jump distribution, we get effective sample sizes of about 400 with just 10000 draws, and effective sizes of 5000 with 100000 draws. In other words, we increase the efficiency of the MCMC steps by about a factor of 10.

Below is code that does both the normal and the original independent uniform jump version of the MCMC. It is mostly the same as the example code with the original exercise, except that that code assumed that alph and bet had been set in the global environment and was in the form of script rather than a function. In this code, `W`, the Cholesky decomposition of the covariance matrix, is assumed set in the global environment when the normal-jump version is used.

At the end below are plots (generated by `plot(mcmc(laclN))` using R with the `coda` package) of the uniform jump 1000 draw case and the normal, 100,000 draw case. In the former we see an initial trend in the first few hundred draws, plus highly serially correlated oscillations thereafter, so the lack of convergence is evident.

```
llhgit <- function(y, alph, cnst, x) {
    sum(y * (alph * x + cnst)) - sum(log(1 + exp(alph * x + cnst)))
}
plot(mcmc(laclN))
x <- 20*runif(200)
y <- runif(200) > 1/(1 + exp(alph0 * x + cnst0))
y <- as.numeric(y)
acl <- matrix(0, nit, 3)
acl[1, 3] <- llhgit(y, 0, 0, x)
for(it in 2:nit) {
    ## ----- Original independent uniform jump distribution
    ## newac <- acl[it - 1, 1:2] +c(.02 * (runif(1) - .5), .2 * (runif(1) - .5))
    ##-----------------------------------
    ## Normal jump distribution.  W calculated in global environment.  Note .55^2
    ne\end{document}
    wac <- acl[it - 1, 1:2] + crossprod(W, rnorm(2)) * .55
    ##-----------------------------------------
    lpnew <- llhgit(y, newac[1], newac[2], x)
    if (exp(lpnew - acl[ it -1, 3]) > runif(1)) {
        acl[it, 1:2] <- newac
        acl[it, 3] <- lpnew
        } else {
        acl[it, ] <- acl[it-1, ]
    }
}
return(acl)
}
```

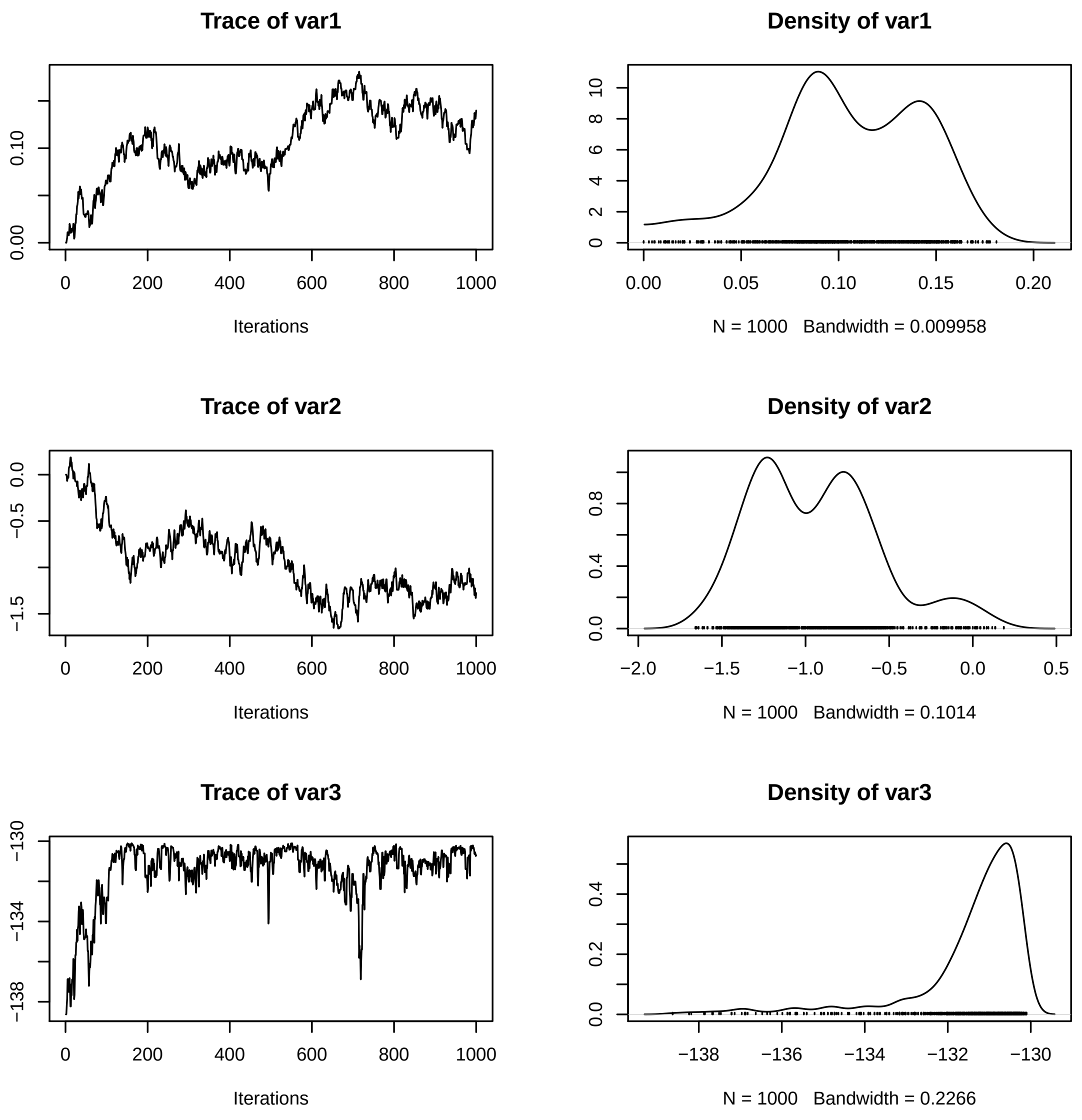

FIGURE 1. 1000 draws from indendent uniform jump MCMC

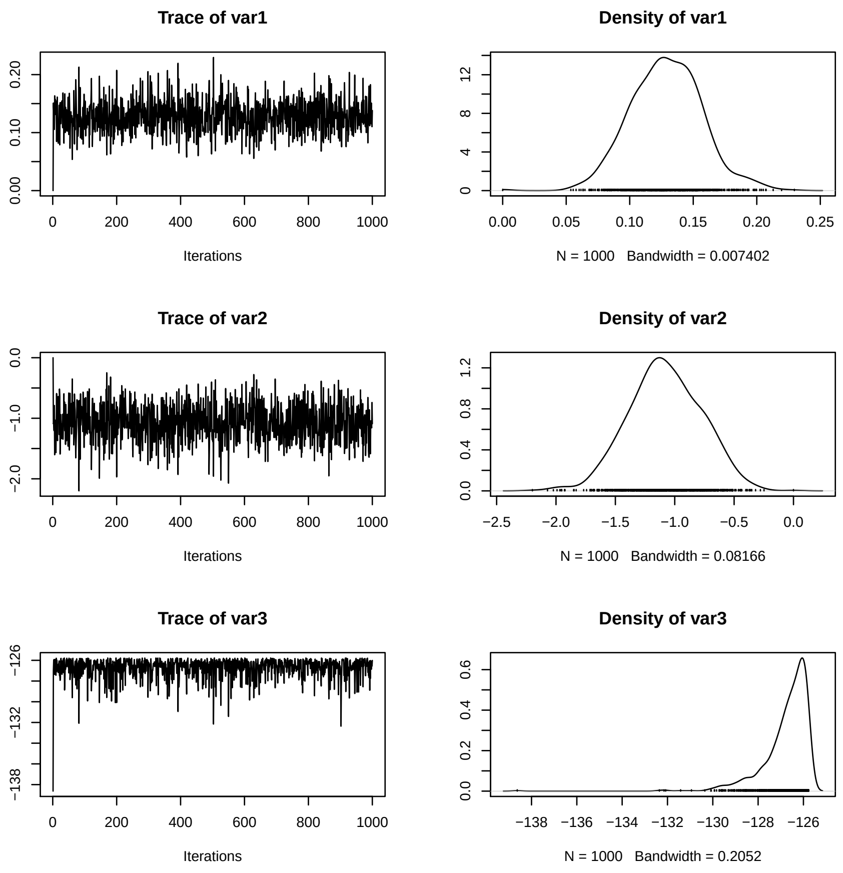

FIGURE 2. 100,000 draws from normal jump MCMC, thinned by factor of 100