APPLIED MODEL CRITICISM: THE MINIMUM WAGE IN NEW JERSEY

In a famous article, David Card and Alan Krueger studied employment in fast food restaurants in New Jersey and Pennylvania before and after New Jerssey set a minimum wage at \$5.05, which was above the starting salary at some of the restaurants at the time. They concluded that it appeared that those restaurants for which the wage minimum was binding *increased* their hiring, rather than decreasing it as some simple theories suggest. Because the studies were somewhat politically charged, and because they contradicted what many people thought they had been taught in ECO101, the article drew a lot of attention and criticism. In this exercise, you, somewhat belatedly, join in. You should read the article before starting work with the data.

1. The Data

The complete data set was made public, and a copy of it is available on a link through our course web site, or directly at http://sims.princeton.edu/yftp/cardKrueger. This directory contains both the file that Card and Krueger provided (`public.dat`), which is formatted in a way that is not convenient for humans to read and requires use of a "code book" to be interpreted or read in to a computer, and an R data frame, `ckdata`, which has in it all the information in the data file and the code book. The R data frame has variables labeled with their original names, plus variable descriptions accessible as a character array `attr(ckdata,"varlabels")`. To get started with this dataset in R, give the command `load("ckdata")`, assuming your working directory contains the file. Otherwise either change your working directory with `setwd("<path to directory>")` or give the full path name of the file in the `load()` command. After it's loaded, to see what's in the dataframe (or in any other R object), use `str(ckdata)`.

In R, it will probably be convenient to `attach(ckdata)` after loading it. This lets you use variable names in `ckdata` without a prefix — e.g. `EMPFT` rather than `ckdata$EMPFT`. Another point about this dataset in R is that it contains variable names (e.g. `WAGE_ST`) that include an underscore. In older versions of R, an underscore was an abbreviation for the `<-` assignment operator, and if you use R with emacs or xemacs and ess, every typed underscore gets automatically converted to `<-`. To work around this, precede the underscore with a quotation mark, type the underscore, then backspace to erase the quote mark. You might want to create copies of such variables with less inconvenient names.

In any program, you will need to recognize that like most survey data this data set contains missing values. In `ckdata` these are coded in the standard R fashion as "NA". Many statistical commands in R will automatically drop observations for which any of the data being used contain NA's, unless the argument `na.rm=FALSE` is invoked. Other commands (e.g. `sum()`, `max()`) have the opposite default behavior, returning NA if there are any NA's in their arguments, unless `na.rm=TRUE` is an explicit argument. If you work directly from `public.dat`, the original data file, you will have to take account of the fact that NA's are literally missing. The data for a given variable show up in a given set of columns, and missing data show up as blanks.

2. Reproduce some of the CK results

Begin by reproducing (at least to a close numerical approximation) the original Card-Krueger results by finding a least squares fit explaining change in employment by a constant and `gap`, a constant and a state dummy, and a constant and both `gap` and a state dummy. Note that you will have to construct employment before and after the minimum wage from the separate data on full time and part time employment that appears in the file, using the same conversion formula that CK used. Also, the `gap` variable has to be constructed from initial starting wage data, using the formula CK give in their article. In R, the command to estimate a linear regression equation is `lm()`. You will want to give a command like `lmout <- lm(demp~gap)` (for a regression of change in employment on the gap, say). You can then get useful summary statistics from `summary(lmout)` and some diagnostic plots from `plot(lmout)`. Some of the diagnostic plots are hard to interpret, but the `qq` plot to check for non-normality and the plot of residuals against predicted values, which may help detect nonlinearity, are fairly easy to understand.

3. Variants needing MCMC

Note that, though there is specific guidance below for dealing with the variant models in R, you can use other languages. Lancaster explains how to handle many models in WinBUGS, a program that runs nicely in Windows and hides the details of how it is implementing its MCMC algorithms. (The hidden implementation is a convenience, so long as the algorithm works, but not otherwise.) WinBUGS is available for free with a time-limited license. For this exercise, it may not be easier to use than R, because R code to handle specific models is provided with this exercise, and you would need to use some other language (e.g. Matlab or, most conveniently, the `coda` package in R) to analyze the WinBUGS output for convergence in any case.

(a) *t* **errors:** Display a qq plot of the residuals from the regression of change in employment on the gap variable. Probably this suggests fatter-than-normal tails. Whether or not it does, proceed to extend the model by estimating it with i.i.d. t_8 residuals using MCMC and data augmentation. This is done automatically in the program `tshock.R` available on the course web site with this exercise. You can also implement the same, or any other correct, algorithm for estimation and inference with this model, using the code in `tshock.R` as a guide, or not. To make `tshock()` available, you can give the command `source("tshock.R")`, assuming `tshock.R` is in your working directory. `tshock()` generates Monte Carlo draws from the posterior for this model assuming a flat prior on the coefficients and a $d\sigma_0^2/\sigma_0^2$ prior on the overall scale of the shocks.

Note that `tshock.R` does not take a formula (like `y` $\sim$ `gap`) as the specification for the regression. Instead it wants a right-hand-side matrix `X` as first argument and a left-hand side vector `y` as second argument. While the `lm()` commmand takes formulas as arguments and automatically deletes rows with NA's, the `tshock()` command requires that you eleminate NA's in advance. In R this is conveniently done by, e.g.,

```
df <- model.frame(demp~gap)
y <- model.response(df)
X <- model.matrix(demp~gap,data=df)
```

The `X` created this way has a column of ones corresponding to the constant term as its first column. This works on the assumption that the `na.action` global option is left at the default setting of `omit`.

A reasonable number of iterations to use here is 10,000. Plot the draws of the coefficients against iteration number and compute effective sample size. (This can be done either by comparing between-group to full-sample variance, as discussed in class, or by invoking `effectiveSize(mcmc(tout$B))`, if `MCMCpack` is available to you. If `MCMCpack` is available on your machine, it will be loaded by `library("MCMCpack")` without any error messages. It is possible to download and install `MCMCpack` and its companion packages `MASS` and `coda` on your own directory if they are not installed globally. If you are lucky, just issuing the command `install.packages(c(MASS,coda,MCMCpack),lib="MyRpackDir")` will install them (assuming you have created the `MyRpackDir` directory). For this exercise, having `MCMCpack` available is only a minor convenience, not at all a necessity. If you have it available and have the return value from `tshock()` in the variable `tout`, then the single command `plot(mcmc(tout$B))` shows the requested plot of coefficent against iteration number, plus a smoothed density estimate, for each coefficient. But producing these graphs with separate commands (e.g. `plot(tout$B[,2])` and `plot(density(tout$B[,2]))`) is not much extra work.

From the MCMC output calculate posterior mean and standard error for each coefficient and compare the results to the OLS results. Also calculate the posterior probability (under the implicit flat prior) that the coefficients on `gap`, `STATE`, or *both* (in the regression containing both) are negative. Compare these to the corresponding probabilities computed under flat priors for the original standard linear regression model. (Note that calculating the probability that both `STATE` (with New Jersey positive) and `gap` have negative coefficients is not a simple linear restriction or set of linear restrictions. So calculating it for the SNLM is non-trivial and you don't have to do it.)

(b) **Probit:** It could be that a few firms are hiring a lot of new workers despite the increased minimum wage, but that the probability that a firm will increase employment at all is reduced by the minimum wage. To check this, we can replace the change in employment as the dependent variable by a 0-1 dummy variable that is 1 if employment increased and 0 otherwise. We can then use the same variables as above as right-hand-side variables in a probit model and again look at the posterior probability that the effect of `gap` or a New Jersey dummy is negative.

As discussed in class, it is easy to get posterior probabilities for a probit model by a Gibbs sampler with data augmentation. Lancaster describes how to do it in his book, giving R code, and essentially his setup is implemented in the R function file `mcprobit.R` on the course web site. To use this file you of course have to construct the appropriate dummy dependent variable. Otherwise proceed as in part (a). This program generates MCMC draws from the posterior assuming a flat prior on the coefficients.

(c) **Logit:** We can use the idea of data augmentation for logit, but because the disturbances in the implied linear equation are logistic distributed, the posterior on β, the linear coefficients, given a draw of the unobservable y^*, is not a standard distribution. In drawing β, therefore, one has to do a Metropolis or Metropolis-Hastings step. The file `mclogit.R` on the course website implements a data-augmentation algorithm with "Metropolis-Hastings within Gibbs". It is not so slow and ill-behaved as to be unusable, but it provides a nice illustration of the possibility that effective sample size can be much smaller than the number of draws and convergence can be slow and doubtful. This program generates MCMC draws from the posterior assuming a flat prior on the coefficients.

If you have access to `MCMCpack`, you can use its `MCMClogit()` function, which is more than 20 times faster than `mclogit` (because written in C) and has better convergence properties. It does not use data augmentation and uses a straight Metropolis algorithm. Even this algorithm has a much worse ratio of effective sample size to numbers of draws than do the algorithms for the models in parts (a) and (b). So for this model your assessment of the needed number of draws and of whether there is convergence is important.

For this model also, carry out the same sorts of analysis as in part (b).