Problem Set 3

Directions: Answer all questions. Clearly label all answers. Show all of your code. Compute standard errors for all parameter estimates. Turn in the following to me via your dropbox (in a folder labeled 'MatlabPS2.3') in Sakai by 11:59 p.m. on Thursday, August 9, 2012:

- m-file(s)
- .sh shell script file that executes your m-file(s)[1]
- a log file (from off the cluster)
- shell_name.oXXXXX file
- pdf version of your writeup with its LaTeXsource code

Put the names of all group members at the top of your writeup (each student must turn in his/her own materials). As usual, set the seed for `rand` and `randn` to 1234.

1. Simulation of a definite integral with no closed form. In this question, compute the integral of $f(x)$ from -10 to 1, where

$$f(x) = \exp\left(-x^2\right)$$

   This integral has no closed form solution, so we will approximate its value by two methods:

   (a) Use one of the quadrature formulas in Matlab to approximate the solution. Note: it's easiest (but not necessary) to use a function handle to define the function.

   (b) Simulate the solution to the integral using the notes from the lecture slide entitled "general steps for simulating an integral." Compare your solution with different numbers of draws. How many draws are required to get close to the solution that the quadrature gave? How long did each procedure take?

2. Bootstrapping, using the dataset `nlsw88.mat` (from problem set 1 of the first Matlab module).

   (a) Bootstrap the sample mean of the variable `ttl_exp` using Matlab's `bootstrp` function. Do 10,000 bootstrap iterations. How does the boostrapped mean compare to the population mean? How does the standard error of the bootstrapped mean compare to its population analog?[2]

[1] Name your job something other than "matsub," and have it send you an email upon completion.

[2] Recall that the standard error of the mean of a variable is $\sigma/\sqrt{n}$, where σ is the standard deviation.

(b) Bootstrap the sample median of the natural logarithm of the variable `wage` without using Matlab's `bootstrp` function. Do 10,000 bootstrap iterations. How does the boostrapped median compare to the population median? How does the standard error of the bootstrapped median compare to its population analog?[3]

(c) Estimate the following model using OLS (same as problem set 2 of the first Matlab module). Feel free to use any estimation method available to you.

$$\begin{aligned}\ln(wage_i) &= \beta_1 + \beta_2 age_i + \beta_3 black_i + \beta_4 other_i + \beta_5 collgrad_i + \\ &\quad \beta_6 grade_i + \beta_7 married_i + \beta_8 south_i + \beta_9 c\_city_i + \\ &\quad \beta_{10} union_i + \beta_{11} ttl\_exp_i + \beta_{12} tenure_i + \beta_{13} age_i^2 + \\ &\quad \beta_{14} hours_i + \beta_{15} never\_married_i + \varepsilon_i\end{aligned}$$

(Be sure to drop observations for all variables where any of the variables are missing.) Obtain bootstrapped standard errors for 10,000 replications for this simple OLS model using Matlab's `bootstrp` function. How do your bootstrapped β's and standard errors compare to the β's and standard errors from the regular OLS?

(d) Repeat (c) but this time code the bootstrapping by hand. Do not use loops over individual observations (or else your bootstrap will never finish).

3. Examining heterogeneity methods in a binomial probit model. Using the dataset `nlsy97.mat` (from problem set 1 of this module), consider the following model (similar to before)

$$\begin{aligned}d_{it} &= \beta_1 + \beta_2 male_i + \beta_3 AFQT_i + \beta_4 Mhgc_i + \beta_5 hgc_{it} + \beta_6 exper_{it} + \\ &\quad \beta_7 Diploma_{it} + \beta_8 AA_{it} + \beta_9 BA_{it} + \alpha_i + \varepsilon_{it}\end{aligned}$$

where d_{it} is 1 if $activity_{it} = 2$ and 0 otherwise. Assume a random effect: $\alpha_i \sim Bernoulli(\pi)$, where $\alpha_i = 0$ with probability $1-\pi$, $\alpha_i = 1$ with probability π, and $\varepsilon_{it} \sim N(0,1)$.

(a) First estimate the model with no heterogeneity using `glmfit`, i.e. assume $\alpha_i = 0$ for all i.

(b) Estimate the model with unobserved heterogeneity using `fminunc`.[4] Use the `glmfit` estimates + random noise as your starting values. How do your parameter estimates change? Report the standard errors for each of your parameters. What is $\hat{\pi}$? How would you interpret this coefficient estimate?

(c) Now assume that the random effect follows a normal distribution; i.e. $\alpha_i \sim N\left(0, \sigma_\alpha^2\right)$. Estimate the parameters β and σ_α^2 using the Composite Simpson's rule[5] over discretized values of α_i (e.g. from -3 to 3). In order to integrate out the random effects, your log likelihood function should look something like

$$\ell = \sum_i \ln\left(\int_{-3}^{3} g_i(\alpha_i) f(\alpha_i) d\alpha_i\right)$$

[3] The standard error of the median of a variable is $1.253\sigma/\sqrt{n}$, where σ is the standard deviation.

[4] You may need to use `fminsearch`, then start `fminunc` at the `fminsearch` answers. This is an example of when optimization can be more of an art than a science.

[5] See "Composite Simpson's rule" in Wikipeida for further details.

where

$$g_i(\alpha_i) = \prod_t \Phi(X_{it}\beta + \alpha_i)^{y_{it}} \left(1 - \Phi(X_{it}\beta + \alpha_i)\right)^{1-y_{it}}$$

and $f(\alpha_i)$ is the pdf of the normal distribution with mean 0 and variance σ_α^2, evaluated at α_i. The Composite Simpson's rule for quadrature states that the integral can be approximated as

$$\begin{aligned}\int_{-3}^{3} g_i(\alpha_i) f(\alpha_i)\, d\alpha_i \;&\approx\; \frac{h}{3}\left[g_i(-3) f(-3) + 4g_i(-3+h) f(-3+h) + 2g_i(-3+2h) f(-3+2h)\right.\\ &\quad \left. + \cdots + 4g_i(3-h) f(3-h) + g_i(3) f(3)\right],\end{aligned}$$

where h is the step size of the discretized grid. Hence, the log likelihood function should look like

$$\ell = \sum_{i=1}^{n} \ln\left(\sum_{g=1}^{G}\left\{\prod_{t=1}^{T} \Phi(X_{it}\beta + \alpha_g)^{y_{it}} \left(1 - \Phi(X_{it}\beta + \alpha_g)\right)^{1-y_{it}}\right\} s_g w_g\right)$$

where

$$\alpha_g = \{-3, -3+h, -3+2h, \ldots, 3-2h, 3-h, 3\},\; g = 1, \ldots G,$$

$$s_g = \frac{h}{3}\{1, 2, 4, 2, \ldots, 2, 4, 1\},\; g = 1, \ldots G,$$

and

$$w_g = f(\alpha_g).$$

Notice that the "Simpson's weights" s_g are 2 if the element of the grid is even, and 4 if odd, with 1 as the first and last element.

Notes: This is quite computationally intensive, so if you can't get estimates for the parameters, don't worry. I want to see you code this. Of course, if you code it successfully, you should estimate it; but it will take awhile, even on the cluster. Avoid loops at all costs, because they go very slowly inside of Matlab's optimizers. It takes a lot of thinking (and use of 3-dimensional matrices), but this problem can be coded with no loops.