Problem Set 4

ECON 6343: Econometrics III
Prof. Tyler Ransom
University of Oklahoma

Due: September 27, 9:00 AM

Directions: Answer all questions. Each student must turn in their own copy, but you may work in groups. Clearly label all answers. Show all of your code. Turn in jl-file(s), output files and writeup via GitHub. Your writeup may simply consist of comments in jl-file(s). If applicable, put the names of all group members at the top of your writeup or jl-file.

You may need to install the following package:

```
Distributions
```

You will need to load the following packages:

```
Optim

HTTP

GLM

LinearAlgebra

Random

Statistics

DataFrames

CSV

FreqTables
```

1. Estimate a multinomial logit (with alternative-specific covariates Z) on the following data set, which is a panel form of the same data as Problem Set 3. You should be able to simply re-use your code from Problem Set 3. However, I would ask that you use automatic differentiation to speed up your estimation, and to obtain the standard errors of your estimates.

   Note: this took my machine about 30 minutes to estimate using random starting values. You might consider using the estimated values from Question 1 of PS3.

```
using DataFrames
using CSV
using HTTP
url = "https://raw.githubusercontent.com/OU-PhD-Econometrics/fall-2022/
master/ProblemSets/PS4-mixture/nlsw88t.csv"
df = CSV.read(HTTP.get(url).body, DataFrame)
X = [df.age df.white df.collgrad]
Z = hcat(df.elnwage1, df.elnwage2, df.elnwage3, df.elnwage4,
         df.elnwage5, df.elnwage6, df.elnwage7, df.elnwage8)
y = df.occ_code
```

The choice set is identical to that of Problem Set 3 and represents possible occupations and is structured as follows.

1 Professional/Technical
2 Managers/Administrators
3 Sales
4 Clerical/Unskilled
5 Craftsmen
6 Operatives
7 Transport
8 Other

2. Does the estimated coefficient $\hat{\gamma}$ make more sense now than in Problem Set 3?

3. Now we will estimate the mixed logit version of the model in Question 1, where γ is distributed $N\left(\tilde{\gamma}, \sigma_\gamma^2\right)$. Following the notes, the formula for the choice probabilities will be

$$P_{ijt} = \begin{cases} \displaystyle\int \frac{\exp\left(X_{it}\beta_j + \tilde{\gamma}(Z_{itj} - Z_{itJ})\right)}{1 + \sum_{k=1}^{J-1}\exp\left(X_{it}\beta_k + \tilde{\gamma}(Z_{itk} - Z_{itJ})\right)} f\left(\tilde{\gamma}\right) d\tilde{\gamma}, & j = 1, \ldots, J-1 \\ \displaystyle\int \frac{1}{1 + \sum_{k=1}^{J-1}\exp\left(X_{it}\beta_k + \tilde{\gamma}(Z_{itk} - Z_{itJ})\right)} f\left(\tilde{\gamma}\right) d\tilde{\gamma}, & j = J \end{cases}$$

and the log likelihood function will be

$$\ell\left(X, Z; \beta, \mu_\gamma, \sigma_\gamma\right) = \sum_{i=1}^{N}\sum_{t=1}^{T} \log\left\{\int \prod_j \left[\frac{\exp\left(X_{it}\left(\beta_j - \beta_J\right) + \tilde{\gamma}\left(Z_{itj} - Z_{itJ}\right)\right)}{\sum_k \exp\left(X_{it}\left(\beta_k - \beta_J\right) + \tilde{\gamma}\left(Z_{itk} - Z_{itJ}\right)\right)}\right]^{d_{itj}} f\left(\tilde{\gamma}\right) d\tilde{\gamma}\right\} \tag{1}$$

While this looks daunting, we can slightly modify the objective function from Question 1.

The first step is to recognize that we will need to *approximate* the integral in (1). There are many ways of doing this, but we will use something called Gauss-Legendre quadrature.[1] We can rewrite the integral in (1) as a (weighted) discrete summation:

$$\ell\left(X,Z;\beta,\mu_{\gamma},\sigma_{\gamma}\right)=\sum_{i=1}^{N}\sum_{t=1}^{T}\log\left\{\sum_{k}\omega_{r}\prod_{j}\left[\frac{\exp\left(X_{it}\left(\beta_{j}-\beta_{J}\right)+\xi_{r}\left(Z_{itj}-Z_{itJ}\right)\right)}{\sum_{k}\exp\left(X_{it}\left(\beta_{k}-\beta_{J}\right)+\xi_{r}\left(Z_{itk}-Z_{itJ}\right)\right)}\right]^{d_{itj}}f\left(\xi_{r}\right)\right\} \tag{2}$$

where ω_r are the quadrature weights and ξ_r are the quadrature points. $\tilde{\gamma}$ and σ_γ are parameters of the distribution function $f(\cdot)$.

(a) Before we dive in, let's learn how to use quadrature. In the folder where this problem set is posted on GitHub, there is a file called `lgwt.jl`. This is a function that returns the ω's and ξ's for a given choice of K (number of quadrature points) and bounds of integration $[a,b]$.

Let's practice doing quadrature using the density of the Normal distribution.

```julia
using Distributions
include("lgwt.jl") # make sure the function gets read in

# define distribution
d = Normal(0,1) # mean=0, standard deviation=1
```

Once we have the distribution defined, we can do things like evaluate its density or probability, take draws from it, etc.

We want to verify that $\int\phi(x)dx$ is equal to 1 (i.e. integrating over the density of the support equals 1). We also want to verify that $\int x\phi(x)dx$ is equal to μ (which for the distribution above is 0).

When using quadrature, we should try to pick a number of points and bounds that will minimize computational resources, but still give us a good approximation to the integral. For Normal distributions, $\pm 4\sigma$ will get us there.

```julia
# get quadrature nodes and weights for 7 grid points
nodes, weights = lgwt(7,-4,4)

# now compute the integral over the density and verify it's 1
sum(weights.*pdf.(d,nodes))

# now compute the expectation and verify it's 0
sum(weights.*nodes.*pdf.(d,nodes))
```

(b) To get some more practice, I'd like you to use quadrature to compute the following integrals:

[1] Another popular method of approximating the integral is by simulation.

- $\int_{-5\sigma}^{5\sigma} x^2 f(x)\,dx$ where $f(\cdot)$ is the pdf of a $N(0,2)$ distribution using 7 quadrature points
- The same as above, but with 10 quadrature points
- The above integrals are the variance of the distribution f. Comment on how well the quadrature approximates the true value.

(c) An alternative to quadrature is Monte Carlo integration. Under this approach, we approximate the integral of f by averaging over a function of many random numbers. Formally, we have that

$$\int_a^b f(x)\,dx \approx (b-a)\frac{1}{D}\sum_{i=1}^{D} f(X_i) \tag{3}$$

where D is the number of random draws and where each X_i is drawn from a $U[a,b]$ interval

- With $D = 1{,}000{,}000$, use the formula in (3) to approximate $\int_{-5\sigma}^{5\sigma} x^2 f(x)\,dx$ where $f(\cdot)$ is the pdf of a $N(0,2)$ and verify that it is (very) close to 4
- Do the same for $\int_{-5\sigma}^{5\sigma} x f(x)\,dx$ where $f(\cdot)$ and verify that it is very close to 0
- Do the same for $\int_{-5\sigma}^{5\sigma} f(x)\,dx$ and verify that it is very close to 1
- Comment on how well the simulated integral approximates the true value when $D = 1{,}000$ compared to when $D = 1{,}000{,}000$.

(d) Note the similarity between quadrature and Monte Carlo. With quadrature, we approximate the integral with

$$\int_a^b f(x)\,dx \approx \omega_i \sum_{i=1}^{D} f(\xi_i)$$

where ω_i is the quadrature weight and ξ_i the quadrature node.

With Monte Carlo, we approximate the integral with

$$\int_a^b f(x)\,dx \approx \frac{(b-a)}{D}\sum_{i=1}^{D} f(X_i)$$

So the "quadrature weight" in Monte Carlo integration is the same $\frac{(b-a)}{D}$ at each node, and the "quadrature node" is a $U[a,b]$ random number

4. Try to modify (but not run!) your code from Question 1 to optimize the likelihood function in (2)

   (This took about 4 hours to estimate on my machine, given the multinomial logit starting values. "Don't try this at home," as they say.)

5. Try to modify (but not run!) your code from Question 1 to optimize the likelihood function in (1), where the integral is approximated by Monte Carlo. Your program will take basically the same form as under Quadrature, but the weights will be slightly different.

   (This takes even longer because, instead of 7 quadrature points, we need to use many, many simulation draws.)

6. Wrap all of your code above into a function and then call that function at the very bottom of your script. Make sure you add `println()` statements after obtaining each set of estimates so that you can read them.