Problem Set 6

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

Directions: Answer all questions. Each student must turn in their own copy, but you may work in groups. You are encouraged to use any and all Artificial Intelligence resources available to you to complete this problem set. 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 will need to load the following previously installed packages:

```
Optim

HTTP

GLM

LinearAlgebra

Random

Statistics

DataFrames

DataFramesMeta

CSV
```

In this problem set, we will repeat the estimation of the simplified version of the Rust (1987, *Econometrica*) bus engine replacement model. Rather than solve the model by backwards recursion, we will exploit the renewal property of the replacement decision and estimate the model using conditional choice probabilities (CCPs).

1. Follow the directions from PS5 to read in the data (the second CSV file you read in as part of PS5) and reshape to "long" panel format, calling your long dataset `df_long`.

2. Estimate a flexible logit model where the dependent variable is the replacement decision and the right hand side is a fully interacted set of the following variables:

   - Mileage
   - Mileage^2
   - Route Usage
   - Route Usage^2
   - Branded
   - Time period
   - Time period^2

   Hint: "Fully interacted" means that all terms from 1st order to 7th order (e.g. Odometer$^2\times$ RouteUsage$^2\times$ Branded $\times$ time^2.

   Hint: Julia's `GLM` package allows you to easily accomplish this by specifying the interacted variables with asterisks in between them. e.g. `Odometer * RouteUsage` estimates a model that includes Odometer, Route Usage and the product of the two.

Dynamic estimation with CCPs

We will use the flexible logit parameters to generate CCPs which we can use to compute the future value term as alternative to the backwards recursion we did in PS5.

Recall the model from PS5, where the differenced conditional value function for running the bus (relative to replacing it) was

$$v_{1t}(x_t,b) - v_{0t}(x_t,b) = \theta_0 + \theta_1 x_{1t} + \theta_2 b + \beta \int V_{t+1}(x_{t+1},b)\,\mathrm{d}F(x_{t+1}|x_t) \tag{1}$$

and where V_{t+1} is the value function and the integral is over transitions in the mileage states x_t.

By exploiting the renewal property of the decision property, we can express V_{t+1} instead as $v_{0t+1} - \log p_{0t+1}$. And since v_{0t+1} corresponds to the renewal action, we know that it is equivalent to

Thus, our value function formulation can be simplified to

$$v_{1t}(x_t,b) - v_{0t}(x_t,b) = \theta_0 + \theta_1 x_{1t} + \theta_2 b - \beta \int \log p_{0t+1}(x_{t+1},b)\,\mathrm{d}F(x_{t+1}|x_t) \tag{2}$$

and by discretizing the integral, we can simplify this even further to be

$$v_{1t}(x_t,b) - v_{0t}(x_t,b) = \theta_0 + \theta_1 x_{1t} + \theta_2 b - \beta \sum_{x_{1,t+1}} \log p_{0t+1}(x_{t+1},b)\left[f_1(x_{1,t+1}|x_{1,t},x_2) - f_0(x_{1,t+1}|x_{1,t},x_2)\right] \quad (3)$$

where the f_j's are defined identically as in PS5.

3. Estimate the θ's using (3) and assuming a discount factor of $\beta = 0.9$. I will walk you through specific steps for how to do this:

   (a) **Construct the state transition matrices** using the exact same code as in this step of PS5.

   (b) **Compute the future value terms** for all possible states of the model. Basically, what we want is $-\log p_{0t+1}$ evaluated at every possible state of the model $(t,b,x_{1,t},x_2)$. The easiest way to do this is to adjust the data that we feed into a `predict()` function using the flexible logit coefficients from question number 2.

   - First, create a data frame that has four variables:
     - Odometer reading (equals `kron(ones(zbin),xval)`)
     - Route usage (equals `kron(ones(xbin),zval)`)
     - Branded (equals 0s of the same size as Odometer and Route usage)
     - time (equals 0s of the same size as Branded)
   - Now write a function that reads in this data frame, the flexible logit estimates, and the other state variables (`Xstate`, `Zstate`, `xtran`, etc.)
   - Initialize the future value array, which should be a 3-dimensional array of zeros. The size of the first dimension should be the total number of grid points (i.e. the number of rows of `xtran`). The second dimension should be 2, which is the possible outcomes of `:Branded`. The third dimension should be $T+1$. Note that the number of rows of the future value array should equal the number of rows of the state data frame.
   - Now write two nested `for` loops:
     - Loop over `t` from 2 to T
     - Loop over the two possible brand states $\{0,1\}$
   - Inside all of the for loops, make the following calculations
     - Update your state data frame so that the `:time` variable takes on the value of `t` and the `:Branded` variable takes on the value of `b`
     - Compute `p0` using the `predict()` function applied to your updated data frame and the flexible logit estimates
     - Store in the FV array the value of $-\beta \log p_0$. Remember that every row of the data frame corresponds to the rows in the state transition matrix, so you can vectorize this calculation.

- Now multiply the state transitions by the future value term. This requires writing another for loop that goes over the rows in the original data frame (the one that you read in at the very beginning of this problem set). In other words, loop over `i` and `t`. To get the actual rows of the state transition matrix (since we don't need to use all possible rows), you should re-use the similar code from PS5; something like this:

```
FVT1[i,t] = (xtran[row1,:].-xtran[row0,:])'*
            FV[row0:row0+xbin-1,B[i]+1,t+1]
```

  The purpose of this loop is to map the CCPs from the each-possible-state-is-a-row data frame to the actual data frame we used to estimate the flexible logit in question 2.
- Your function should return `FVT1` in "long panel" format. I used `FVT1'[:]` to make this conversion, but you should double check that your i and t indexing of your original data frame matches.

(c) **Estimate the structural parameters.**

- Add the output of your future value function as a new column in the original "long panel" data frame. The easiest way to do this is `df_long = @transform(df_long, fv = fvt1)`
- Now use the `GLM` package to estimate the structural model. Make use of the "offset" function to add the future value term as another regressor whose coefficient is restricted to be 1. That is:

```
theta_hat_ccp_glm = glm(@formula(Y ~ Odometer + Branded),
                    df_long, Binomial(), LogitLink(),
                    offset=df_long.fv)
```

(d) Optionally, you can write your own function to estimate a binary logit where you restrict the offset term to have a coefficient of 1. (I will include this code in my solutions.)

(e) Wrap all of your code in an empty function as you've done with other problem sets. Prepend your wrapper function call (at the very end of the script) with `@time` so that you can time how long everything takes. (On my machine, everything took under 20 seconds.)

(f) Glory in the power of CCPs!

4. Have an AI write unit tests for each of the functions you've created (or components of each) and run them to verify that they work as expected. Best practice is to provide unit tests in a separate script that first reads in the source code before running the tests. Thus, I would like you to turn in three files:

   - `PS6_LastName_source.jl`
   - `PS6_LastName_script.jl`
   - `PS6_LastName_tests.jl`