Problem Set 5

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 may need to install and load the following package:

```
DataFramesMeta
```

You will also need to load the following previously installed packages:

```
Optim

HTTP

GLM

LinearAlgebra

Random

Statistics

DataFrames

CSV
```

In this problem set, we will explore a simplified version of the Rust (1987, *Econometrica*) bus engine replacement model. Let's start by reading in the data.

```
using DataFrames
using CSV
using HTTP
url = "https://raw.githubusercontent.com/OU-PhD-Econometrics/fall-2024/
master/ProblemSets/PS5-ddc/busdataBeta0.csv"
df = CSV.read(HTTP.get(url).body, DataFrame)
```

Static estimation

1. Reshape the data into "long" panel format, calling your long dataset `df_long`. I have included code on how to do this in the `PS5starter.jl` file that accompanies this problem set.

2. The model we would like to estimate is Harold Zurcher's decision to run buses in his fleet. Zurcher's flow utility of running (i.e. not replacing) a bus is

$$u_1(x_{1t}, b) = \theta_0 + \theta_1 x_{1t} + \theta_2 b \tag{1}$$

   where x_{1t} is the mileage on the bus's odometer (in 10,000s of miles) and b is a dummy variable indicating whether the bus is branded (meaning its manufacturer is high-end). The choice set is $\{0,1\}$ where 0 denotes replacing the engine.

   Estimate the θ parameters assuming Zurcher is completely myopic. This amounts to estimating a simple binary logit model. (**Note:** you may estimate this any way you wish. I would recommend using the `GLM` package, but you may also use `Optim` with your own log likelihood function.)

Dynamic estimation

Now I will walk you through how to estimate the dynamic version of this model using backwards recursion. With discount factor β, the differenced conditional value function for running the bus (relative to replacing it) is

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

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

We will approximate the integral with a summation, which means that we will specify a discrete mass function for $f(x_{t+1}|x_t)$. This probability mass function depends on the current odometer reading (x_{1t}), whether the engine is newly replaced (i.e. $d_{t-1} = 0$), and on the value of another state variable x_2 which measures the usage intensity of the bus's route (i.e. high values of x_2 imply a low usage intensity and vice versa).

We discretize the mileage transitions into 1,250-mile bins (i.e. 0.125 units of x_{1t}). We specify x_2 as a discrete uniform distribution ranging from 0.25 to 1.25 with 0.01 unit increments.

Formally, we are discretely (but not discreetly!) approximating an exponential distribution:

$$f_j\left(x_{1,t+1}|x_{1,t},x_2\right)=\begin{cases}e^{-x_2(x_{1,t+1}-x_{1t})}-e^{-x_2(x_{1,t+1}+0.125-x_{1t})} & \text{if } j=1 \text{ and } x_{1,t+1}\geq x_{1,t}\\ e^{-x_2(x_{1,t+1})}-e^{-x_2(x_{1,t+1}+0.125)} & \text{if } j=0 \text{ and } x_{1,t+1}\geq 0\\ 0 & \text{otherwise}\end{cases} \tag{3}$$

You will not need to program (3); I will provide code for this part.

Under this formulation, (2) can be written as

$$\begin{aligned}v_{1t}\left(x_t,b\right)-v_{0t}\left(x_t,b\right)=\theta_0+\theta_1 x_{1t}+\theta_2 b+\\ \beta\sum_{x_{1,t+1}}V_{t+1}\left(x_{t+1},b\right)\left[f_1\left(x_{1,t+1}|x_{1,t},x_2\right)-f_0\left(x_{1,t+1}|x_{1,t},x_2\right)\right]\end{aligned} \tag{4}$$

Finally, we can simplify (4) since we know that $V_{t+1}=\log\left(\sum_k \exp\left(v_{k,t+1}\right)\right)$ when we assume that unobserved utility is drawn from a T1EV distribution (as we do here):

$$\begin{aligned}v_{1t}\left(x_t,b\right)-v_{0t}\left(x_t,b\right)=\theta_0+\theta_1 x_{1t}+\theta_2 b+\\ \beta\sum_{x_{1,t+1}}\log\left\{\exp\left(v_{0,t+1}\left(x_{t+1},b\right)\right)+\exp\left(v_{1,t+1}\left(x_{t+1},b\right)\right)\right\}\times\\ \left[f_1\left(x_{1,t+1}|x_{1,t},x_2\right)-f_0\left(x_{1,t+1}|x_{1,t},x_2\right)\right]\end{aligned} \tag{5}$$

Estimation of our dynamic model now requires two steps:

Solving the model First, we need to solve the value functions for a given value of our parameters θ. The way we do this is by backwards recursion. We know that $V_{t+1}=0$ in our final period (i.e. when $t=T$). Then we work backwards to obtain the future value at every possible state in our model. This will include many states that do not actually show up in our data.

Estimating the model Second, once we've solved the value functions, we use maximum likelihood to estimate the parameters θ. The log likelihood function in this case is simply

$$\ell=\sum_{i=1}^{N}\sum_{j=0}^{1}\sum_{t=1}^{T}d_{ijt}\log P_{ijt} \tag{6}$$

where

$$\begin{aligned}P_{i1t}&=\frac{\exp\left(v_{1t}-v_{0t}\right)}{1+\exp\left(v_{1t}-v_{0t}\right)}\\ P_{i0t}&=1-P_{i1t}\end{aligned} \tag{7}$$

3. Now estimate the θ's assuming that Zurcher discounts the future with discount factor $\beta=0.9$. I will walk you through specific steps for how to do this:

(a) **Read in the data** for the dynamic model. This can be found at the same URL as listed at the top of p. 2, but remove the `"Beta0"` from the CSV filename.

Rather than reshaping the data to "long" format as in question 1, we want to keep the data in "wide" format. Thus, columns `:Y1` through `:Y20` should be converted to an array labeled `Y` which has dimension 1000×20 where $N = 1000$ and $T = 20$. And similarly for columns starting with `:Odo` and `:Xst`. Variables `:Xst*` and `:Zst` keep track of which discrete bin of the f_j's the given observation falls into.

(b) **Construct the state transition matrices**, which are the f_j's in (3). To do so, simply run the following code:

```
zval,zbin,xval,xbin,xtran = create_grids()
```

`zval` and `xval` are the grids defined at the bottom of p. 2, which respectively correspond to the route usage and odometer reading. `zbin` and `xbin` are the number of bins in `zval` and `xval`, respectively. `xtran` is a (`zbin*xbin`)$\times$`xbin` Markov transition matrix[1] that gives the probability of falling into each $x_{1,t+1}$ bin given values of $x_{1,t}$ and x_2, according to the formula in (3).

(c) **Compute the future value terms** for all possible states of the model.

- First, 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 $T + 1$.
- Now write four nested `for` loops over each of the possible states:
  - Loop backwards over `t` from T to 1
  - Loop over the two possible brand states $\{0, 1\}$
  - Loop over the possible permanent route usage states (i.e. from 1 to `zbin`)
  - Loop over the possible odometer states (i.e. from 1 to `xbin`)
- Inside all of the for loops, make the following calculations
  - Create an object that marks the row of the transition matrix that we need to be looking at (based on the loop values of the two gridded state variables). This will be `x + (z-1)*xbin` (where `x` indexes the mileage bin and `z` indexes the route usage bin), given how the `xtran` matrix was constructed in the `create_grids()` function.
  - Create the conditional value function for driving the bus (v_{1t}) based on the values of the state variables in the loop (not the values observed in the data). For example, for the mileage (x_{1t}), you should plug in `xval[x]` rather than `:Odo`.

    The first component of v_{1t} should be the flow utility, which is listed in (1).

    The difficult part of v_{1t} is the discrete summation over the state transitions. For this, you need to grab the appropriate row (and all columns) of the `xtran`

[1] A Markov transition matrix is a matrix where each row sums to 1 and moving from e.g. column 1 to column 4 within a row gives the probability of moving from state 1 to state 4. Check out the Wikipedia page for more information

matrix, and then take the dot product of that with all possible x_{1t} rows of the FV matrix for the given value of x_2.
You should end up with something like

```
xtran[row,:]'*FV[(z-1)*xbin+1:z*xbin,b+1,t+1]
```

where `b` indexes the branded dummy and `t` indexes time periods.[2]

- Now create the conditional value function for replacing the engine (v_{0t}). For this, we repeat the same process as with v_{1t} except the θ's are normalized to be 0. The code for the expected future value is the same as for v_{1t} with the exception that mileage resets to 0 after replacement, so instead of grabbing `xtran[row,:]` we want `xtran[1+(z-1)*xbin,:]`.
- Finally, update the future value array in period t by storing $\beta \log(\exp(v_{0t}) + \exp(v_{1t}))$ in the tth slice of the 3rd dimension of the array. This will be the new future value term for period $t-1$. Remember to set $\beta = 0.9$

(d) **Construct the log likelihood** using the future value terms from the previous step and only using the observed states in the data. This will entail a `for` loop over buses and time periods.

- Initialize the log likelihood value to be 0. (We will iteratively add to it as we loop over observations in the data)
- Create a variable that indexes the state transition matrix rows for the case where the bus has been replaced. This will be the same `1+(z-1)*xbin` as in the conditional value function v_{0t} above. However, we need to plug in `:Zst` from the data rather than a hypothetical value `z`.
- Create a variable that indexes the state transition matrix rows for the case where the bus has not been replaced. This will be the same `x + (z-1)*xbin` as in v_{1t} above, except we substitute `:Xst` and `:Zst` for `x` and `z`.
- Now create the flow utility component of $v_{1t} - v_{0t}$ using the actual observed data on mileage and branding.
- Next, we need to add the appropriate discounted future value to round out our calculation of $v_{1t} - v_{0t}$. Here, we can difference the f_j's as in (5). You should get something like

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

- Finally, create the choice probabilities for choosing each option as written in (7) and then create the log likelihood according to the summation in (6).

(e) Wrap all of the code you wrote in (c) and (d) into a function and set up the function so that it can be passed to `Optim`. For example, you will need to return the negative of the log likelihood and you will need to have the first argument be the θ vector that we are trying to estimate

[2] We need to index by `b+1` because `b` takes on values 0 or 1, but in Julia it is illegal to reference the 0th element of an array, so element 1 of the index corresponds to `b=0` while element 2 of the index corresponds to `b=1`.

(f) On the same line as the function, prepend the function declaration with the macros so that your code says `@views @inbounds function myfun()` rather than `function myfun()`. This will give you more performant code. On my machine, it cut the computation time in half.

(g) Wrap all of your code in an empty function as you've done with other problem sets

(h) Try executing your script to estimate the likelihood function. This took about 4 minutes on my machine when I started from the estimates of the static model in Question 2.

(i) Pat yourself on the back and grab a beverage of your choice, because that was a lot of work!

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:

   - `PS5_LastName_source.jl`
   - `PS5_LastName_script.jl`
   - `PS5_LastName_tests.jl`