Problem Set 1

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.

Before starting, you will need to install and load the following packages:

```
JLD

Random

LinearAlgebra

Statistics

CSV

DataFrames

FreqTables

Distributions
```

0. GitHub setup
   (a) Create a GitHub account
   (b) Fork the class repository to your account
1. Initializing variables and practice with basic matrix operations
   (a) Create the following four matrices of random numbers, *setting the seed to* '1234'[1]. Name the matrices and set the dimensions as noted
      i. $A_{10\times7}$ - random numbers distributed $U[-5,10]$
      ii. $B_{10\times7}$ - random numbers distributed $N(-2,15)$ [st dev is 15]

[1] To set the seed, type the following code (after loading the `Random` package): `Random.seed!(1234)`

iii. $C_{5\times7}$ - the first 5 rows and first 5 columns of A and the last two columns and first 5 rows of B

iv. $D_{10\times7}$ - where $D_{i,j} = A_{i,j}$ if $A_{i,j} \leq 0$, or 0 otherwise

(b) Use a built-in Julia function to list the number of elements of A

(c) Use a series of built-in Julia functions to list the number of *unique* elements of D

(d) Using the `reshape()` function, create a new matrix called E which is the 'vec' operator[2] applied to B. Can you find an easier way to accomplish this?

(e) Create a new array called F which is 3-dimensional and contains A in the first column of the third dimension and B in the second column of the third dimension

(f) Use the `permutedims()` function to twist F so that it is now $F_{2\times10\times7}$ instead of $F_{10\times7\times2}$. Save this new matrix as F.

(g) Create a matrix G which is equal to $B \otimes C$ (the Kronecker product of B and C). What happens when you try $C \otimes F$?

(h) Save the matrices A, B, C, D, E, F and G as a .jld file named `matrixpractice`.

(i) Save only the matrices A, B, C, and D as a .jld file called `firstmatrix`.

(j) Export C as a .csv file called `Cmatrix`. You will first need to transform C into a `DataFrame`.

(k) Export D as a tab-delimited .dat file called `Dmatrix`. You will first need to transform D into a `DataFrame`.

(l) Wrap a function definition around all of the code for question 1. Call the function `q1()`. The function should have 0 inputs and should output the arrays A, B, C and D. At the very bottom of your script you should add the code `A,B,C,D = q1()`.

2. Practice with loops and comprehensions

(a) Write a loop or use a comprehension that computes the element-by-element product of A and B. Name the new matrix AB. Create a matrix called $AB2$ that accomplishes this task without a loop or comprehension.

(b) Write a loop that creates a column vector called `Cprime` which contains only the elements of C that are between -5 and 5 (inclusive). Create a vector called `Cprime2` which does this calculation without a loop.

(c) Using loops or comprehensions, create a 3-dimensional array called X that is of dimension $N \times K \times T$ where $N = 15,169$, $K = 6$, and $T = 5$. For all t, the columns of X should be (in order):

- an intercept (i.e. vector of ones)
- a dummy variable that is 1 with probability $.75 * (6-t)/5$
- a continuous variable distributed normal with mean $15+t-1$ and standard deviation $5(t-1)$

[2] See `http://en.wikipedia.org/wiki/Vectorization_(mathematics)`

- a continuous variable distributed normal with mean $\pi(6-t)/3$ and standard deviation $1/e$
- a discrete variable distributed "discrete normal" with mean 12 and standard deviation 2.19. (A discrete normal random variable is properly called a binomial random variable. The distribution described above can be implemented by choosing binomial parameters n and p where $n = 20$ and $p = 0.6$. Use the following code (after loading Julia's `Distributions` package) to generate this vector of X: `rand(Binomial(20,0.6),N)`, where N is the length of the vector
- a discrete variable distributed binomial with $n = 20$ and $p = 0.5$

i.e., let columns 1, 5 and 6 remain stationary over time.

(d) Use comprehensions to create a matrix β which is $K \times T$ and whose elements evolve across time in the following fashion:

- $1, 1.25, 1.5, ...$
- $\ln(t)$
- $-\sqrt{t}$
- $e^t - e^{t+1}$
- t
- $t/3$

(e) Use comprehensions to create a matrix Y which is $N \times T$ defined by $Y_t = X_t\beta_t + \varepsilon_t$, where $\varepsilon_t \overset{iid}{\sim} N(0, \sigma = .36)$

(f) Wrap a function definition around all of the code for question 2. Call the function `q2()`. The function should have take as inputs the arrays A, B and C. It should return nothing. At the very bottom of your script you should add the code `q2(A,B,C)`. Make sure `q2()` gets called after `q1()`!

3. Reading in Data and calculating summary statistics

(a) Import the file `nlsw88.csv` into Julia as a `DataFrame`. Make sure you appropriately convert missing values and variable names. Save the result as `nlsw88_processed.csv`.

(b) What percentage of the sample has never been married? What percentage are college graduates?

(c) Use the `freqtable()` function to report what percentage of the sample is in each race category

(d) Use the `describe()` function to create a matrix called `summarystats` which lists the mean, median, standard deviation, min, max, number of unique elements of the data frame. How many `grade` observations are missing?

(e) Show the joint distribution of industry and occupation using a cross-tabulation.

(f) Tabulate the mean wage over industry and occupation categories. Hint: you should first subset the data frame to only include the columns industry, occupation and wage. You should then follow the "split-apply-combine" directions `here`.

(g) Wrap a function definition around all of the code for question 3. Call the function `q3()`. The function should have no inputs and no outputs. At the very bottom of your script you should add the code `q3()`.

4. Practice with functions

   (a) Load `firstmatrix.jld`.

   (b) Write a function called `matrixops` that takes as inputs the matrices A and B from question (a) of problem 1 and has three outputs: *(i)* the element-by-element product of the inputs, *(ii)* the product $A'B$, and *(iii)* the sum of all the elements of $A+B$.

   (c) Starting on the line above the function, write a docstring that explains what `matrixops` does. Follow Julia best practices here.

   (d) Evaluate `matrixops()` using A and B from question (a) of problem 1

   (e) Just after the function definition line of `matrixops()`, write an if statement which gives an error if the two inputs are not the same size. Have the error say "inputs must have the same size." You may also elect to add a type check to the inputs of the function, e.g. `function matrixops(A::Array{Float64},B::Array{Float64})`.

   (f) Evaluate `matrixops()` using C and D from question (a) of problem 1. What happens?

   (g) Now evaluate `matrixops()` using `ttl_exp` and `wage` from `nlsw88_processed.csv`. Hint: before doing this, you will need to convert the data frame columns to Arrays. e.g. `convert(Array,nlsw88.ttl_exp)`, depending on what you called the data frame object [I called it `nlsw88`].

   (h) Wrap a function definition around all of the code for question 4. Call the function `q4()`. The function should have no inputs or outputs. At the very bottom of your script you should add the code `q4()`.

5. Write unit tests for each of the functions you've created and run them to verify that they work as expected. AIs are great at writing unit tests for you. 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:

   - `PS1_LastName_source.jl`
   - `PS1_LastName_script.jl`
   - `PS1_LastName_tests.jl`

6. Turn in your files as a commit to the `ProblemSets/PS1-julia-intro/` folder on your GitHub fork. You can do this by simply clicking "upload files" in the appropriate folder, or you can use the GitHub desktop app, RStudio, VS Code, or the command line to stage, commit and push the files. If you are having trouble, AIs are great at git.