NAME: Micah Botkin-Levy
DATE: 4/18/19
Final Project Technical Review
STAT/CS 387

The majority of the code for this project is contained within two python files, 'Code\run_ess.py' and 'Code\ess_functions.py'. 'run_ess.py' is essentially a wrapper script that runs the project and 'ess_functions.py' contains all the functions that the wrapper calls. The other piece of code is a custom package called gym-ess which uses the gym package to create the custom energy storage system environment and object.

In order to solve the energy arbitrage problem, I solved an optimization problem and a reinforcement learning. These were chosen to contrast each other. In terms of software tools used, I implemented the solution using python in pycharm. I used pandas to store the data as a dataframe and matplotlib to produce the plots. To solve the optimization problem, I formulated it using cvxpy which is a modeling language package and used the solver Gurobi which has a free academic license. For the reinforcement learning, I customized a environment for modeling the energy storage system using the gym package.

Looking back over the life-cycle of this project, it mostly did not change from original scope. Originally, I was hoping to explore a few more reinforcement learning. Additionally, if I had more time I would have liked to test a deep Q learning algorithm. Due to some pieces taking longer than expected, I didn't not get to test as many variations of reinforcement learning algorithm as I would have liked.

The biggest challenged I faced was creating the custom gym environment to model the energy storage system. As I had not created a python package or used gym before this was new to me. However there were some useful online resources that made this process easier. Other challenges that I faced included needing to cap the power when the energy storage system was close to full or empty to keep the energy storage system within its feasible physical limits. Additionally, I had to round the energy state of the energy storage system to prevent the system using incrementally smaller discharge powers as it got closer to empty. Finally, working with dates is always a headache and I had to deal with reading these in and making the X axis labels on the plots readable but pandas is relatively helpful for this. I didn't have any major failures or dead ends but I did start by downloading LMP data for all locations which was time and memory consuming so I changed to only download data associated with the location Id of Burlington.

While I did not necessarily follow all of the best practices rules from the articles, I did try to use scientific computational best practices where possible. First, I used git and github to version control my code which was hosted at https://github.com/mbotkinl/DS2_Final_Project. I included comments and docstrings for functions in my code to increase readability for others. In terms of code structure, I functionalized as much as possible including the data import, algorithms, and plotting. Keeping the plotting separate from my algorithm computations was also a conscious decision. Where possible I used existing packages and tools such as pandas and gym to assist with my code functionality. With more time I would include a readme, examples, and some error checking code.