Shawn Rice
4/29/18
Modeling & Simulation with R & Python
Prof. Dale Frakes

Exercise 1

Problem 1 – Random numbers and Probability Distributions

Ex1_Prob1_Python

I am not much of a mathematician, so I took this opportunity to familiarize myself with the similarities, differences, and applications for different probability distributions. I took the chance as well to learn about page design within python as well. I had otherwise used the code from the instructor and notated as much as possible to understand the code syntax and logic flow decision ordinance. I did play with x & y labels, colors, character traces vs raster\vector traces and other design choices.

Ex1_Prob1_R

Again, my focus here was on the commonalities and differences between Python and R. I have found that the designers axiom holds true; form has followed function. Python, being a bit more of a general-purpose tool, is more verbose and circumspect to get many mathematic purposes accomplished. I have found that many elements come pre-designed. For example, x & y axis labeled themselves. Scales were also added automatically. Consequently, inserting my design choices is more verbose and circumspect. I did find the language elements familiar though. Matlab, Wolfram, NetBeans and Perl all have elements in common with R's syntax, page design commands, and nomenclature.

draw at least 2 sets of random samples from the distribution
–try varying N (maybe one with a small N and one with a large N)
I have run N & n as side-by-side graphs.
Run 1 (N = 1000, n = 100):
(N) y_fit = [0:0.4]; x_fit = [-3:3]
(n) y_fit = [0:1.25}; x_fit = [-0.5:0.5]; scale [-3:3]
Run 2 (N = 5000, n = 300):
(N) y_fit = [0:0.4]; x_fit = [-4:4]
(n) y_fit = [0:1.4}; x_fit = [-0.5:0.5]; scale [-4:4]

–try varying the parameters (e.g. mean, standard deviation, rate, min, max, mode, shape, etc.)
These all had the expected results.

Problem 2 – Approximating the Binomial Distribution

Ex1_Prob2_Python & R

I had a difficult time understanding the application for this, and so had a difficult time understanding what the results were displaying. I again copied the code and notated it until I fully understood the function of this script. It took a few times of running it on both Python and R; then cross-referencing internet lookups surrounding "probability distributions"+"Monte Carlo method". I understand the function to be in one usage a count of probability of a predicted flip within x number of attempts. In another usage, this can be used to check uniform odds by altering the "Total_Flips"

variable. Another usage allows for non-uniform odds by creating classes, allotting probability by class, and dropping individual agents into their respective classes.

Problem 3 – Approximating the Geometric Distribution

Ex1_Prob3_Python & R

In this model, I used the instructor's code to investigate the Python Data Analysis Library (pandas). Speaking to the statement earlier about form following function; I found it interesting that Python had to import two libraries and call to several modules in order to accomplish, in this case, what R did in only a few lines of code and with only one call to a module which didn't need to be imported. Again, though, it was fairly easy to left-justify the text inside the data frame this time. I went back and left justified the data frame output in Prob2_R.

Problem 4 – The Monte Hall Problem

Ex1_Prob4_Python

In this one I flew solo. I started by drawing up the pseudocode which is still in the code as comments. From there I just started entering code and it worked after some minor syntactical cleanup. I am impressed by the ease and familiarity Python offers. It took very few lookups, as the language is very high-level. I found the community to be quite active and there are an astounding number of procedural walkthroughs available. The "man pages" are easy to locate information in, and easy to digest once the information is located. While many tasks in Python are more circumspect than in R, it appears to be more "all-purpose" and thus the attention to intelligibility is quite noticeable.

A Word on Lists and Arrays in Python 3

The paradigmatic shift between Python "Lists" and traditional "Arrays" is both inconsequential in description and maddening in execution. The simple entry style is greatly appreciated; but retrieving the data from a construct with no locations or enumeration more than makes up for the easy creation. It will take some getting used to, but I can see the logic behind it. It appears to be driven by the focus on utilitarianism.