Lab 6: Generate & Analyze Spatial random variables.

Overall Deliverables & Rubric:

● Complete set of code and figures (one notebook per exercise) concatenated as 1 PDF file, uploaded to CANVAS.

● Total Points 50

0. Computer Programming Environment

All the exercises involve Python and can be done from your web browser using Google's Colab.

1. Exercise 1: exponential and unfiform random variables

You will read the numpy documentation about the random number generator for exponential random variables (RVs). Draw/generate some RVs and compute summary statistics on them.
https://numpy.org/doc/stable/reference/random/generated/numpy.random.Generator.exponential.html

Code Block 1:
Draw 100 exponential RV (independent realizations) with a scale parameter of 1/200

Figure 1: Using hist function for Matplotlib make a histogram. Make sure to choose appropriate bins if the default are not.

Answer 1. Using NumPy's quantile, calculate the 95% confidence interval.

Code Block 2, Figure 2:
Plot the probability density function as a curve. Choose a set of points to evaluate the function so that the curve is smooth and has an appropriate limit to show the shape.

Code Block 3:
Generate N=200 realizations from a uniform RV on [0,1).
Calculate the distance between each point and its neighbor to the right (N-1).
Figure 3. Plot the histogram of the realization.
Figure 4. Plot the histogram of the differences.
Text block 1: Write a discussion of your insights from this exercise. Did you find a connection between the uniform and exponential random variables?

Deliverable & Rubric:

Make this Exercise 1 in one standalone Jupyter Notebook and print it.
2 points for each figure. 5 points for the discussion and code. (25 points in total).

2. Exercise 2: Geospatial queries

Upload the GDS_lab6_exercise_2.ipynb to Google Colab or use your own machine.
The location of households, parks, and grocery stores are given as Shapely objects.
A GeoPandasData frame is created with some columns.

This exercise is based on existing tutorials:
https://geopandas.org/en/stable/docs/user_guide/aggregation_with_dissolve.html
https://geographicdata.science/book/notebooks/06_spatial_autocorrelation.html

Figure 1:
Using the data, make a scatter plot of the distance to park versus satisfaction index.
Label the axes.

Figure 2:
Make a histogram of satisfaction index.
Label the axes.

Figure 3:
Make a histogram of household size.
Label the axes.

Code Block 2:
The existing code uses NumPy's histogram2d to create the bin edges, but then does spatial joins to aggregate the data.
Using the merged/dissolved data, you will make a new histogram and compare it to Figure 2.

Figure 4:
Make a histogram of spatially averaged satisfaction index.
Label the axes.

Code Block 3:
The analysis found both population and satisfaction index were significantly spatially correlated. Repeat the analysis for the spatial analysis of the population and satisfaction index for the randomly distributed households.

Report Moran I's and the p-values for population and satisfaction.

Deliverable & Rubric:

Make this Exercise 2 in one standalone Jupyter Notebook and print it.
5 points for each figure. 5 points for the code block 3 and correctly calculating Moran I using the packages. (25 points in total).

In the future, this is an example rubric (from Prof. Dani Arribas-Bel) for your code submission
0-15%: the code does not run and there is no documentation to follow it.

16-39%: the code does not run, or runs but it does not produce the expected outcome. There is some documentation explaining its logic.

40-49%: the code runs and produces the expected output. There is some documentation explaining its logic.

50-59%: the code runs and produces the expected output. There is extensive documentation explaining its logic.

60-69%: the code runs and produces the expected output. There is extensive documentation, properly formatted, explaining its logic.

70-79%: all as above, plus the code design includes clear evidence of skills presented in advanced sections of the course (e.g. custom methods, list comprehensions, etc.).

80-100%: all as above, plus the code contains novel contributions that extend/improve the functionality the student was provided with (e.g. algorithm optimizations, novel methods to perform the task, etc.).