Tasks 2020

Fundamentals of Data Analysis

Due: last commit on or before December 18th, 2020

These are the instructions for the Tasks assessment for Fundamentals of Data Analysis in 2020. The assessment is worth 50% of the marks for the module. Please read the *Using git for assessments* [3] document on the Moodle page which applies here. As always, you must also follow the code of student conduct and the policy on plagiarism [2].

Instructions

Four tasks will be listed here at different times during the semester. You should complete all tasks in a single jupyter notebook. This, along with relevant files like a README, should be in a single git repository synced with a hosting provider like GitHub [1]. That URL should then be submitted using the link on the Moodle page.

1. **October 5th, 2020:** Write a Python function called `counts` that takes a list as input and returns a dictionary of unique items in the list as keys and the number of times each item appears as values. So, the input `['A', 'A', 'B', 'C', 'A']` should have output `{'A': 3, 'B': 1, 'C': 1}`. Your code should not depend on any module from the standard library[1] or otherwise. You should research the task first and include a description with references of your algorithm in the notebook.

2. **November 2nd, 2020:** Write a Python function called `dicerolls` that simulates rolling dice. Your function should take two parameters: the number of dice k and the number of times to roll the dice n. The function should simulate randomly rolling k dice n times, keeping track of each total face value. It should then return a dictionary with the number of times each possible total face value occurred. So, calling the function as `diceroll(k=2, n=1000)` should return a dictionary like:

   ```
   {2:19,3:50,4:82,5:112,6:135,7:174,8:133,9:114,10:75,11:70,12:36}
   ```

   You can use any module from the Python standard library you wish and you should include a description with references of your algorithm in the notebook.

[1] By the standard library, we mean the modules and packages that come as standard with Python. Anything built-in that can be used without an `import` statement can be used.

3. **November 16th, 2020:** The `numpy.random.binomial` function can be used to simulate flipping a coin with a 50/50 chance of heads or tails. Interestingly, if a coin is flipped many times then the number of heads is well approximated by a bell-shaped curve. For instance, if we flip a coin 100 times in a row the chance of getting 50 heads is relatively high, the chances of getting 0 or 100 heads is relatively low, and the chances of getting any other number of heads decreases as you move away from 50 in either direction towards 0 or 100. Write some python code that simulates flipping a coin 100 times. Then run this code 1,000 times, keeping track of the number of heads in each of the 1,000 simulations. Select an appropriate plot to depict the resulting list of 1,000 numbers, showing that it roughly follows a bell-shaped curve. You should explain your work in a Markdown cell above the code.

4. **November 30th, 2020:** Simpson's paradox is a well-known statistical paradox where a trend evident in a number of groups reverses when the groups are combined into one big data set. Use numpy to create four data sets, each with an `x` array and a corresponding `y` array, to demonstrate Simpson's paradox. You might create your `x` arrays using `numpy.linspace` and create the `y` array for each `x` using notation like `y = a * x + b` where you choose the `a` and `b` for each `x`, `y` pair to demonstrate the paradox. You might see the Wikipedia page for Simpson's paradox for inspiration.

Marking scheme

The following marking scheme will be used to mark your submission out of 100%, which will then be scaled to 50%. The examiner's overall impression of your submission may influence marks in each individual component. It is important that your submission provides direct evidence of each of the items listed in each category. For instance, your commit history should demonstrate and provide evidence that you had a pragmatic attitude to completing the assessment. Likewise, your submission should have references in it to demonstrate that you considered the literature and the work of others.

25%	**Research**	Evidence of research performed on topic; submission based on referenced literature, particularly academic literature; evidence of understanding of the documentation for any software or libraries used.
25%	**Development**	Environment can be set up as described; code works without tweaking and as described; code is efficient, clean, and clear; evidence of consideration of standards and conventions appropriate to code of this kind.
25%	**Consistency**	Evidence of planning and project management; pragmatic attitude to work as evidenced by well-considered commit history; commits are of a reasonable size; consideration of how commit history will be perceived by others.
25%	**Documentation**	Clear documentation of how to create an environment in which any code will run, how to prepare the code for running, how to run the code including setting any options or flags, and what to expect upon running the code. Concise descriptions of code in comments and README.

References

[1] GitHub Inc., "GitHub," https://github.com/.

[2] GMIT, "Quality Assurance Framework," https://www.gmit.ie/general/quality-assurance-framework.

[3] I. McLoughlin, "Using git for assessments," https://github.com/ianmcloughlin/using-git-for-assessments/.