TECH2 mandatory assignment

Deadline: October 6, 2024 at 23:59

Standard deviation of a sequence of numbers

You are having a discussion with your boss and a colleague regarding the fastest method to calculate the standard deviation of a sequence of numbers.

The standard deviation σ characterizes the dispersion of a sequence of data $(x_1, x_2, \ldots, x_N)$ around its mean $\overline{x}$. A high σ indicates that many of the values in the sequence are far away from the mean, whereas a low value indicates that most values are close to the mean.

The standard deviation is computed as the square root of the variance σ^2, defined as

$$\sigma^2 = \frac{1}{N} \sum_{i=1}^{N} \Big(x_i - \overline{x} \Big)^2$$

where N is the number of elements, and the mean $\overline{x}$ is defined as

$$\overline{x} = \frac{1}{N} \sum_{i=1}^{N} x_i$$

The above formula for the variance can be rewritten as

$$\sigma^2 = \left(\frac{1}{N} \sum_{i=1}^{N} x_i^2 \right) - \overline{x}^2$$

This suggests the following algorithm to compute the standard deviation:

1. Compute the mean $\overline{x} = \frac{1}{N} \sum_{i=1}^{N} x_i$
2. Compute the mean of squares $S = \frac{1}{N} \sum_{i=1}^{N} x_i^2$
3. Compute the variance $\sigma^2 = S - \overline{x}^2$
4. Compute the standard deviation $\sigma = \sqrt{\sigma^2}$

There are several ways to compute the standard deviation of a sequence of values in Python:

1. Your colleague thinks that the fastest solution is to compute the mean and sum of the squares in the algorithm above by using solely `for` loops, and that no functions are required.
2. Your boss believes that `for` loops are slow and should therefore be avoided. He instead suggests using Python's built-in functions such as `sum()` and `len()` to reduce the number of loops.
3. You, however, know that NumPy has a function called `std()` that can be used to calculate the standard deviation of a sequence of numbers. You believe that it is generally better to use pre-existing Python functions instead of generating your own functions.

Your task is to compute the standard deviation of a sequence of numbers using all three approaches.

Part A

Write a Python script that uses all three approaches above to compute the standard deviation of a sequence of values.

For this purpose, the file `part_A.py` contains the headers of the following two functions:

[1]:
```python
def std_loops(x):
    """
    Compute standard deviation of x using loops.

    Parameters
    ----------
    x: Sequence of numbers

    Returns
    -------
    sd : float
        Standard deviation of the list of numbers.
    """

def std_builtin(x):
    """
    Compute standard deviation of x using the built-in functions sum()
    and len().

    Parameters
    ----------
    x: Sequence of numbers

    Returns
    -------
    sd : float
        Standard deviation of the list of numbers.
    """
```

Provide an implementation for each function, the first using loops and the second using `sum()` and `len()`.

Write code to demonstrate that these two functions and `std()` from NumPy calculate the same standard deviations for the following list of numbers:

```
num_lst = [1, 2, 3, 4, 5]
```

Hint: You can use the built-in `sqrt()` function from the math module to compute the square root in step (4) of the algorithm outlined above:

```
from math import sqrt
```

Part B

Your boss is not convinced that `std()` from NumPy can compute the standard deviation faster than the two other approaches. He therefore gives you a file containing randomly generated data, and he asks you to compare the run time of all three approaches.

The file `data.csv` contains the following three columns:

- Column 1: sequence of 100 numbers between 0 and 1
- Column 2: sequence of 1,000 numbers between 0 and 1
- Column 3: sequence of 10,000 numbers between 0 and 1

This is a comma-separated text file where the values in each row are separated by a comma. He knows that you are not familiar with this file format, so he suggests that you look at this link to see how you can import the data in the file by using Python's built-in `open()` function.

Use the empty notebook `part_B.ipynb` in this repository to complete the following tasks:

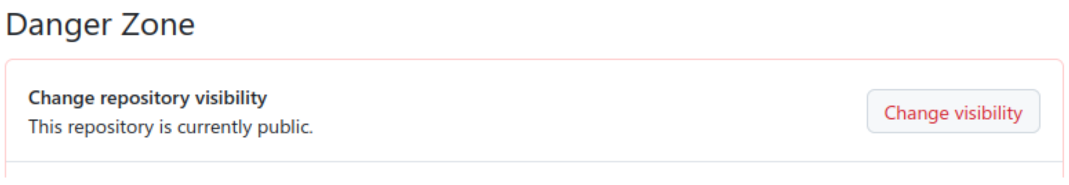
1. Import the file and store each column of values in a list.
2. Compute the standard deviation of all three sequences using the three different approaches
3. Record the run-time of each approach for each sequence. Recall that you can use the magic function `%timeit` to time the execution of a function in a Jupyter notebook.

Summarize your conclusion for your boss. Which approach computes the standard deviation the fastest? Is one approach always faster or does it depend on the length of the sequence? Try to explain your findings.

Assessment

The requirements to pass the assignment are as follows:

- You have submitted within the deadline.
- Your submission is in a GitHub repository. All commits in this repository must be prior to the deadline. The repository must be publicly accessible (set visibility to public in the repository settings):



Danger Zone

Change repository visibility
This repository is currently public.

- The repository contains a Python script that attempts to calculate the standard deviation using all three approaches using the list of numbers (part A).
- You must complete a peer-review of the assignments of two other students on Canvas (deadline: Friday, October 11, 2024 at 14:00).

In addition, the following are recommended, but not required to pass the assignment:

- Your create and use the Anaconda environment defined in `environment.yml`.
- Your Python script use functions to solve part A of the assignment.
- Your code is well documented.
- Your repository contains a Jupyter notebook that solves part B of the assignment.
- You use git to manage your repository.