Week 3 – Execution Time Profiling in Python and R

Student: Shreya Kurani

Course: High Performance Computing

Assignment:- Week 3 Assignment

Q1. Tabulate the execution times of each of the individual approaches for computing distance in Python.

Python was stable across methods. Although the base loop was easy and reasonably fast, the iterrows() method provided a slight speed advantage but is not scalable. The apply() method turned out to be the slowest because of its increased overhead. NumPy vectorization had strong results through fast array operations for bulk computation.

Method	Mean Execution Time	Standard Deviation
Base For Loop	2.15 ms	0.18 ms
Iterrows Haversine	1.76 ms	0.25 ms
Pandas Apply	4.92 ms	0.27 ms
Vectorized (NumPy Arrays)	2.37 ms	0.11 ms

Q2. Next, replicate the for-loop based approach (the first one) and two different ways to make that version more efficient, in R. Profile these three approaches, and tabulate the results.

In R, the manually nested distance loop was slowest as it was expected because of repetitive calculations. The use of `apply()` decreased the run time by a lot. The matrix-based Haversine implementation was superior to the remaining two methods by a huge margin, leveraging R's internal vector optimizations for numerical tasks.

Method	Min Time	Mean Time	Max Time
Manual Nested `sapply()` Loop	3.51 sec	3.73 sec	4.11 sec
`apply()` Function	350 µs	541 µs	1120 µs
Matrix Vectorized Calculation	28 µs	46.1 µs	465 µs

Q3. Based on the computational efficiency of implementations in Python and R, which one would you prefer? Based on a consideration of implementation (i.e., designing and implementing the code), which approach would you prefer? Taking both of these (run time and coding time), which approach would you prefer?

Coming from my experience, R was a little bit ahead in raw performance due to vectorized matrix math. However, Python was much easier to work with because of its cleaner syntax, in-built methods and error messages which helped me to debug faster. It was more rigid and took longer to write the same logic in R.

Therefore, R is faster but Python is my favorite language. For the perfect balance between speed and development time, I would still go for Python because it scales and fits well into the different tools and workflows that I use.

Q4. Identify and describe one or two other considerations, in addition to these two, in determining which of the two environments – Python or R – is preferable to you.

Apart from run-time and coding experience, there were two other considerations that came to my mind:

1. Library Ecosystem – There is a massive variety of ML, web, and automation libraries in Python (e.g., `pandas`, `scikit-learn`, `Flask`), and it is not just a data analysis tool. R is very domain-specific but powerful.

2. Industry Use and Career Goals – Python is more sought in the tech and healthcare positions. R has a role to play in academic research and bioinformatics, but Python is king in applied positions.

Since I hope to be doing health data science jobs that require real-world application and production environments, I feel that Python is the better long-term investment. However, I appreciate R's strengths and will develop competence in both.