1. Tabulation of Execution Times for Python:

By analyzing the execution times of three different methods for computing distances in Python. The first method used a for-loop, where the program iterates through each element of the data and computes the distance one at a time. The second method optimized the for-loop by using iterrows(), which allows for better handling of data. The third approach was vectorized, using Pandas to apply the computation across all data elements at once. Below are the execution times for each approach:

Approach	**Execution Time (seconds)**
For-loop based approach	0.0038
Optimized for-loop using iterrows()	0.0031
Vectorized implementation using Pandas	0.0021

As shown in the table, the vectorized approach was the fastest, followed by the optimized for-loop approach.

2. Replication of Approaches in R:

Approach	**User Time (seconds)**	**System Time (seconds)**	**Total Time (seconds)**
For-loop based approach	0.004	0.000	0.003
Efficient approach using apply()	0.003	0.000	0.002
Fully vectorized approach	0.001	0.000	0.001

Next, they executed these methods in R and measured the run times; they tested three methods: the original for-loop, an apply() based approach, and a fully vectorized approach. The execution times for each method in R are as follows:From the table, it's clear that the fully vectorized approach was the fastest in R as well, with a total execution time of 0.001 seconds.

3. On the basis of Computational Efficiency and Coding Time: In comparing Python and R based on computational efficiency (run time), the fully vectorized approach was the fastest in both languages. However, when considering coding time, R had an advantage. The vectorized approach in R is quite easy to use with built in functions such as apply() and vectorized operations, while Python needs to use external libraries like Pandas to handle the data effectively. Taking into consideration these factors the fully vectorized approach of R is the best approach for both computational efficiency and coding ease. Not only is this method the fastest, but it is also the easiest to code, thus making it the best for this task.

Conclusion: Therefore, the best approach for this task would be R's fully vectorized approach, as it is the fastest in terms of execution time and is easier to implement as compared to Python. But the choice between Python and R depends on the nature of the project. For statistical analysis and data visualization R is ideal. For general purpose programming or integration with other systems, Python may be more appropriate. Both environments have their strengths, and the decision ultimately depends on the specific requirements of the task in question.