**Gelsa: GPU-accelerated local similarity analysis toolkit**

Yang Li1,Li Charlie Xia1

1School of Mathematics, South China University of Technology, Guangzhou 510641, China

**#** Corresponding author: Li Charlie Xia: [lcxia@scut.edu.cn](mailto:lcxia@scut.edu.cn)

**Abastract**

We present novel parallel computing method called GeLSA(GPU-accelerated local similarity analysis), designed to accelerate analysis the time series data, particularly in the fields of biology and environmental science. The GeLSA enhances computational efficiency enabling the processing of large-scale data in a short amount of time, parallelizes both the alignment and dispatch algorithms, and further accelerates the computation by ~144x on a GPU equipped PC.Futhermore, GeLSA integrates a range of algorithms designed for local correlation analysis, including theo and perm, incorporating MBBLSA, DDLSA, and STLTA, which are developed to address the complexities associated with analyzing non-independent and identically distributed time series data. GeLSA not only maintains the same level of accuracy as traditional methods but also significantly outperforms them in terms of efficiency and speed on experimental results on simulated datasets.Additionally,, the conducted on real biological datasets underscores GeLSA's proficiency in precisely pinpointing local correlations and temporal dependencies within time series data. The GeLSA holds significant value, as it contributes to the acceleration of big data analysis processes and enhances the overall efficiency of data mining endeavors.

***Keywords*:** Local similarity analysis; Local trend analysis; Statistical significance; Accelerated local correlation computations;Time series data

**Availability:** An open-source python-C++ implementation of GeLSA is available at: [https://github.com/labxscut/GeLSA](https://github.com/labxscut/OmiHier)

**Contact:** [lcxia@scut.edu.cn](mailto:lcxia@scut.edu.cn)

**1 Introduction**

In fields such as biology and environmental science, analyzing the interactions among multiple factors and their impacts on ecosystems or biological populations is often essential. Time series analysis serves as an effective method to understand the patterns and trends of these factors over time. In traditional research, most methods consider the overall correlation among factors over the entire time interval. However, real biological data often exhibit more complex relationships and dynamic changes, including localized and lagged correlations observed in various fields such as microbiology[1-4], molecular biology[5,6], and functional neuroscience[7,8],methods based on global similarity analysis may fail to detect these relationships. To address the limitations of global correlation, researchers have introduced many methods, among which local similarity analysis (LSA)[10-13]stands out.LSA is a locally aligned method that identifies optimized alignment configurations between two time series, thereby detecting paired correlations at local and potential time delays. A related method, called local trend analysis (LTA)[14-16], is a variant of LSA. Through these analyses of local similarity, we can gain deeper insights into the relationships between factors in different time periods and their mechanisms of action. This is crucial for formulating strategies for ecological conservation, predicting trends in biological population changes, and more.

LSA is not only widely used in many subfields but has also undergone significant theoretical and practical improvements by numerous researchers. Qian et al. initially proposed a method for analyzing gene expression profiles[10],based on this method, LSA has been extended to analyze microbiome data, including studies by Ruan et al. on operational taxonomic unit (OTU) analysis[13] and by Xia et al. on metagenomic data analysis[12]. Methodological advances in LSA also include support for replicated series data (eLSA)[12], statistical significance approximation (p-values) reported by Xia et al.[11] and Durno et al.[9], and more advanced methods for computing p-values such as moving block bootstrap LSA (MBBLSA)[17] and data-driven LSA (DDLSA)[18]. Xia et al.[16] proposed a theoretical approximation of the statistical significance of LTA, later further improved by Shan et al. as steady-state theory local trend analysis (STLTA)[19]. These three methods address the limitation of traditional LSA, which can only analyze independent and identically distributed data, thus enabling LSA to perform well on autocorrelated data.

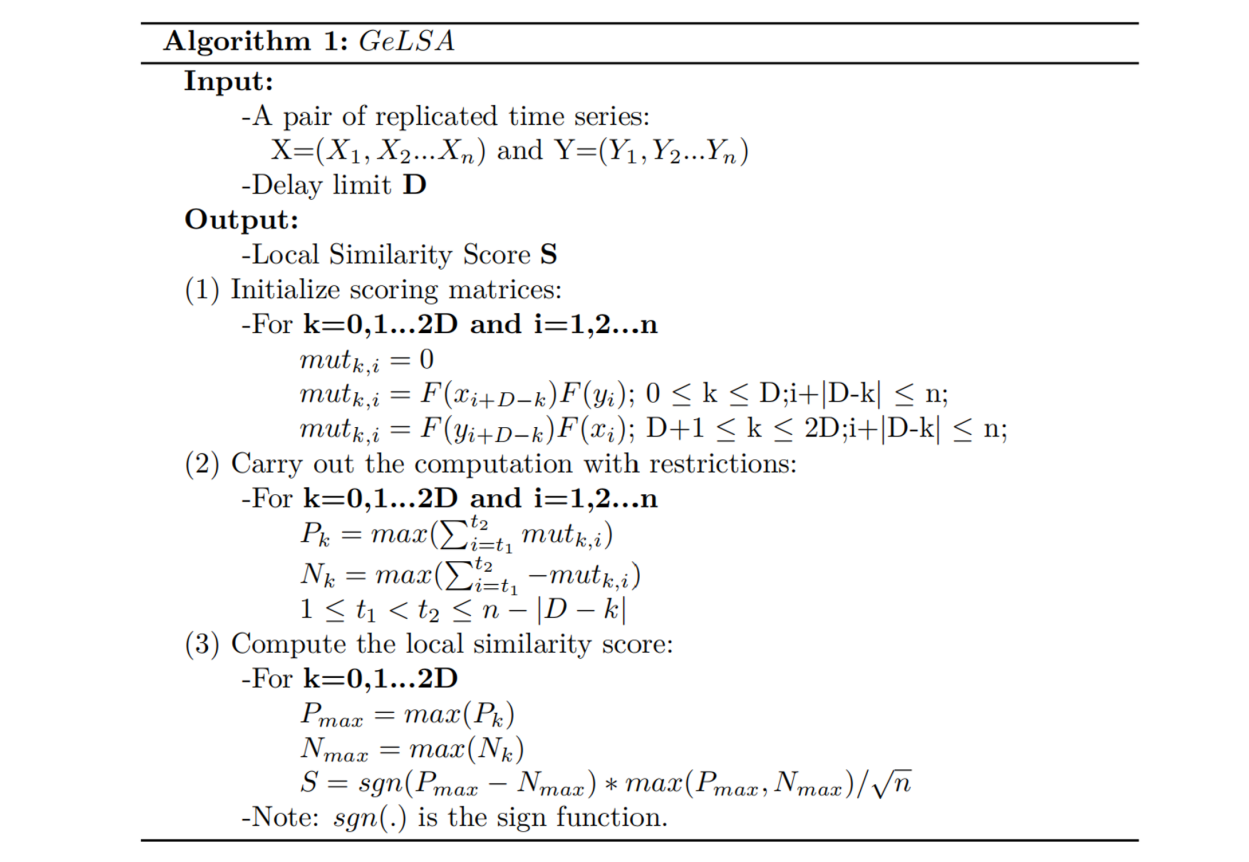
With the continuous deepening of research in this field and the ongoing development of data mining techniques, traditional LSA computing software faces more challenges. On one hand, previous computational methods for LSA incur significant computational costs, requiring quadratic comparisons for n data points. On the other hand, novel theoretical algorithms for LSA, (MBBLSA, DDLSA, and STLTA)[17-19], have expanded the scope of LSA and LTA research to analyze autocorrelated time series data. However, previous solutions have failed to meet the demands for rapid and efficient analysis of large-scale time series data. We propose a parallel computing method to accelerate the analysis of time series data. This method not only speeds up the computation process but also improves the LSA computing core, further reducing the time and space complexity of computations. We compare the efficiency of GeLSA with the LSA computing core on a single core. Through experiments on simulated datasets, we compare the overall software performance of GeLSA with eLSA. We find that the analysis of large-scale time series data, which previously required more time to complete, can now be accomplished in significantly shorter timeframes, greatly enhancing the efficiency and accuracy of time series data analysis in fields such as biology and environmental science. Additionally, we integrate the aforementioned expanded algorithms for LSA (MBBLSA, DDLSA, STLTA)[17-19], enabling effective analysis even of autocorrelated time series data.

The structure of this paper is as follows: Section 2, we introduce the GeLSA algorithm, and compare the computational core efficiency of CPU and GPU parallel computing, verifie the new algorithm, and demonstrate the feasibility and reliability of parallel computing. Section 3, we demonstrate how to test the overall efficiency of software by using simulated data and perform biological time series calculations on the continuous 72-hour dataset from Daya Bay, drawing the computational analysis results into correlation networks. Section 4 we evaluate the efficiency of the new method.

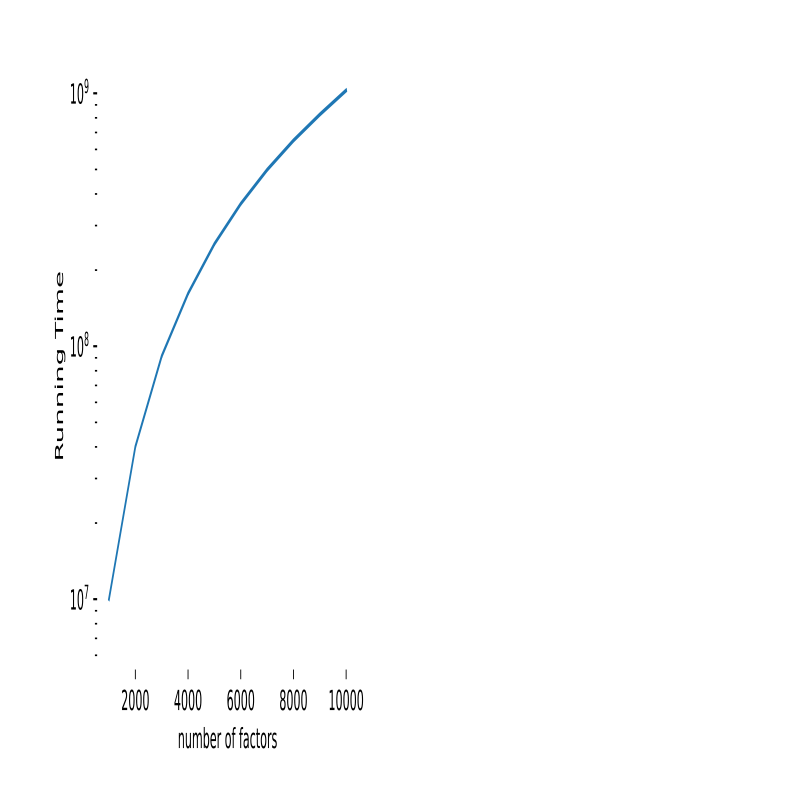
**2 Materials and methods**

**2.1 Algorithm**

The first step is to underwent redesign the traditional LSA algorithm, then applying it to GPU computing cores. The effectiveness of the proposed method is clearly demonstrated through the comparison of runtime graphs.

****

In theory, for a pair of time series data, the new LSA algorithm, compared to the traditional LSA algorithm, reduces the spatial complexity from 3\*n^2 to (2D+1)\*n, where D is the time series offset, n is the length of the time series, and D is much smaller than n. Therefore, the newly designed algorithm will achieve better performance both on a single CPU core and on a GPU.



On one hand, in Figure 2A, we compared the computational performance of the new LSA algorithm and the traditional LSA on a single CPU core. The experiments show that the new LSA computing core has improved performance compared to the traditional LSA computing core. For example, when m = 1000, new LSA = 6775000s, LSA = 10023300s, the runtime ratio = 1.47;when m = 5000, new LSA = 171716081s, LSA = 253681000s, the runtime ratio = 1.48;and when m = 10000, new LSA = 693567567s, LSA = 1026480000s, the runtime ratio = 1.48.

On the other hand, in Figure 2B, we compared the computational performance of the traditional LSA with the GeLSA computing core. Theoretically, the GeLSA algorithm has a significant improvement in time complexity compared to the LSA algorithm. The time complexity of GeLSA is O(T), while the time complexity of LSA is O(t\*m^2), where T is the time complexity of new LSA,and t is the time complexity of LSA, m is the number of time series. Meanwhile, the experiments show that the GeLSA computing core has improved performance compared to the traditional LSA computing core. For example, when m = 1000, GeLSA = 69606.25s, LSA = 10023300s, the runtime ratio = 144; when m = 5000, GeLSA = 1761673.61s, LSA = 253681000s, the runtime ratio = 143; when m = 10000, GeLSA = 7128333s, LSA = 1026480000s, the runtime ratio = 144.145.

**2.2 Simulation studies and real datasets**

To evaluate the runtime efficiency of the GeLSA software, we conducted analysis computations by using both GeLSA and the eLSA software on the same simulated dataset. We compared the runtime of these two software packages and generated 8 performance comparison graphs.

Firstly, for given (n, m) parameters, we generated n pairs of independent and identically distributed standard normal random variable pairs (Xi, Yi), where each pair represents observations on the time series, and each time series has a length of m. Xi and Yi are mutually independent.

Next, on one hand, keeping the number of data points (m) fixed, we varied the number of data points (n) and conducted experiments to obtain the runtime of both software packages as the length of the time series data changed; on the other hand, keeping the number of data points (n) fixed, we varied the number of data points (m) and conducted experiments to obtain the runtime of both software packages as the number of the time series data changed. Each experiment generated a simulated dataset, and the corresponding runtime was got. Under this comparison method, we obtained the runtime of both software packages theoretically (GeLSA theo, DDLSA, STLTA, and eLSA theo) and experimentally (GeLSA perm, MBBLSA, and eLSA perm). Each algorithm was run 5 times on the experimental dataset, and the mean runtime was calculated to obtain a sufficient and reliable amount of data for evaluating the performance of GeLSA and eLSA software.

Finally, based on the runtime of the above software, we plotted line graphs of the runtime to visually demonstrate the performance differences between the two software packages.

We applied the GeLSA method to analyze the Daya Bay dataset. We performed GeLSA calculations on this dataset and then plotted correlation networks based on it.The Daya Bay dataset consists of ASV (Amplicon Sequence Variants) abundance data sampled at high frequency over 72 hours. We selected 400 abundant ASVs from the Daya Bay dataset, with a minimum relative abundance of 1%, covering up to 97.0% of all ASVs. For the time series, we examined time-lagged correlations with delays of (0 hours, 6 hours, 12 hours, 18 hours, 24 hours, and 48 hours). P-value comparisons for local similarity analysis were conducted in supplementary Table (SX).

**3 Results**

**3.1 performence comparation**

There are a total of 8 subgraphs in Figure 4, and the line graph data in each subplot are obtained by averaging the results of 5 experimental tests. Overall, it can be clearly seen that each algorithm in GeLSA (MBBLSA, perm, DDLSA, STLTA) exhibits significantly improved efficiency compared to eLSA.

It is evident that the parallel algorithms, which designed based on theoretical formulas, can fully utilize the hardware resources of the computer, resulting in a considerable improvement in runtime efficiency of GeLSA relative to eLSA in subgraph A1. For instance, in theoretical calculations, both for autocorrelated time series data (GeLSA DDLSA) and independent and identically distributed time series data (GeLSA theo), there is a substantial reduction in runtime compared to eLSA theo under similar computational loads. For example, when m=100, GeLSA theo=13.37s, GeLSA DDLSA=15.54s, while eLSA theo=188.49s, indicating a speedup of 14.09 times. Similarly, conclusive results can be derived from subplot A2 regarding the LTA theoretical algorithm.

****

It is apparent that for permutation-based experimental computations, GeLSA also exhibits significantly improved runtime efficiency compared to eLSA, in subgraph B1. The computational performance improvement is especially valuable for analyses where theoretical calculations (n<20) are not feasible. Both for autocorrelated time series data (GeLSA MBBLSA) and independent and identically distributed time series data (GeLSA perm), there is a substantial reduction in runtime compared to eLSA perm under similar computational loads. For example, when m=10, GeLSA perm=51.67s, GeLSA MBBLSA=125.33s, while eLSA perm=704.16s, indicating a speedup of 13.89 times. Similar conclusions can be drawn from subplot B2 regarding the LTA permutation algorithm.

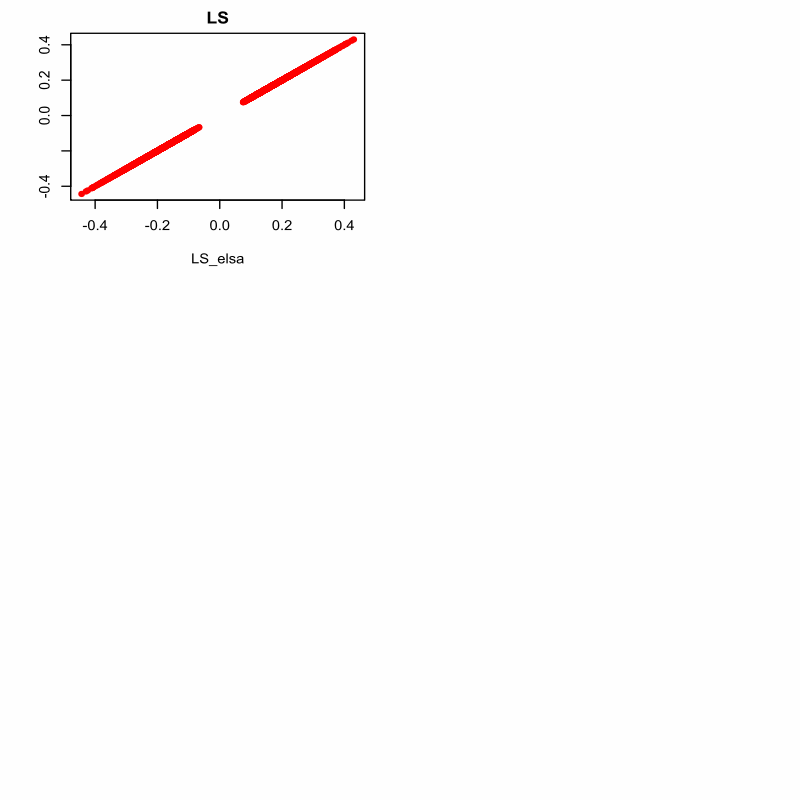
Although only the length of the time series data is varied, in subgraph C1, it can be observed that GeLSA significantly outperforms eLSA in theoretical computations, similar to the findings in subplot A1. There is a considerable reduction in runtime compared to eLSA theo under similar computational loads. For instance, when n=20, GeLSA theo=10.75s, GeLSA DDLSA=11.42s, while eLSA theo=164.04s, indicating a speedup of 16.09 times. Similar conclusions can be drawn from subplot C2 regarding the LTA theoretical algorithm. Additionally, it can be observed that the runtime of GeLSA DDLSA is almost identical to GeLSA theo, which is consistent with the findings of Zhang F et al.[18], indicating a negligible difference of 0.12% on average for n=20, 60, and 100.

It is evident that GeLSA significantly outperforms eLSA in permutation algorithm, in subgraph D1, similar to the findings in subplot B1. Both for autocorrelated time series data (GeLSA MBBLSA) and independent and identically distributed time series data (GeLSA perm), there is a considerable reduction in runtime compared to eLSA perm under similar computational loads. For example, when n=20, GeLSA perm=384.13s, GeLSA MBBLSA=1957.38s, while eLSA perm=25669.11s, indicating a speedup of 65 times. Similar conclusions can be drawn from subplot D2 regarding the LTA theoretical algorithm. Additionally, it's evident that the runtime of GeLSA MBBLSA is significantly slower than GeLSA perm, consistent with the findings of Zhang F et al.[17], indicating a 1.5 times slower runtime for GeLSA MBBLSA compared to GeLSA perm for n=20, 60, and 100.

It's important to note that the perfromance of software acceleration mainly involve two aspects: the number of data points (m) and the method of calculating p-values. The number of data points determines whether the computer CPU hardware is fully utilized, while the p-value calculation method determines whether the computer GPU hardware is fully utilized. Therefore, when analyzing and optimizing software acceleration effects, it is necessary to consider the influence of both factors.

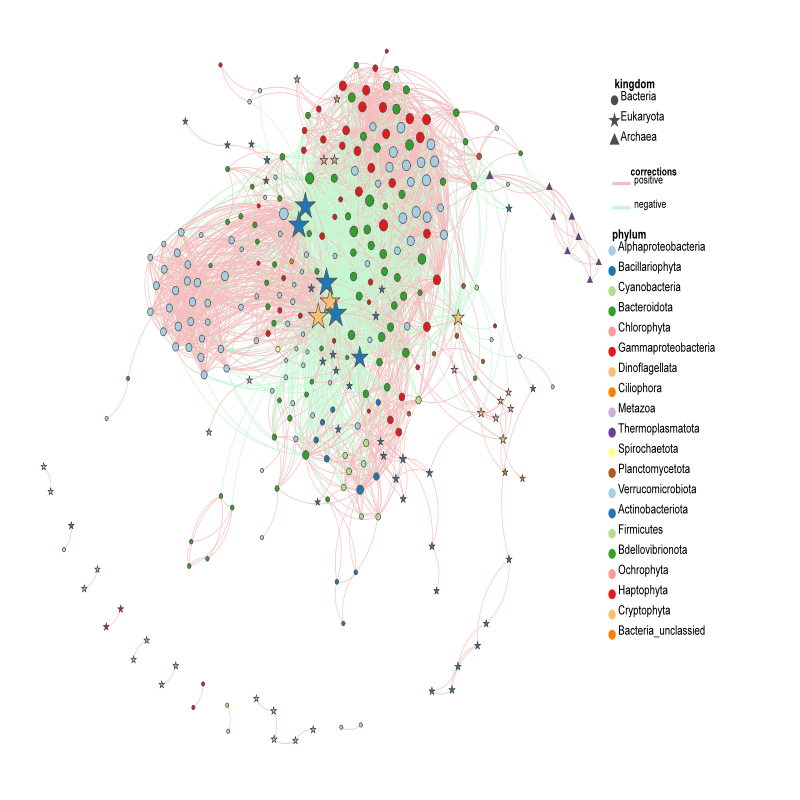
**3.2 Algorithm tidely**

We conducted a comparison of the results obtained from running GeLSA and eLSA software using the simulated dataset generated in section 2.2 "Simulation Studies and Real Datasets". Specifically, the settings were configured as follows: m=100, n=50 and d=0. The outcomes are presented in the following Figure 5. There are a total of 6 subplots. Overall, it is evident from the plots that there is no deviation between the GeLSA's "theo" algorithm and the traditional eLSA software's "theo" algorithm when analyzing a dataset comprising 100 pairs of sequences which gengerated 4950 pieces of data.

****

It is clear that, in subplot A, the results of GeLSA(theo) and eLSA(theo) for LS (local similarity) calculations are consistently distributed along the diagonal line, forming a straight line composed of closely aligned points. This suggests that GeLSA performs LS calculations accurately.In subplots B, C, and D, the results of GeLSA(theo) and eLSA(theo) for the starting positions (xs, ys) of local maxima, as well as for the subsequence lengths (len), are consistently distributed along the diagonal lines, forming straight lines composed of closely aligned points. Similarly, in subplot E, the results of GeLSA(theo) and eLSA(theo) for p-values are consistently distributed along the diagonal line, indicating accurate calculations. This observation is consistent with the conclusion drawn from subplot A, further corroborating the software's accuracy.In subplot F, it is evident that this comparative experiment of software performance was conducted with a delay of 0.

**3.3 The Daya Bay time series dataset**



In the 72-hour time series data from Daya Bay, potential microbial interactions between phytoplankton and prokaryotes were identified using GeLSA. These interactions include symbiosis, cross-nutrition, competition, parasitism, predation, and allelopathy. It was found that several significant time-lagged correlations (Spearman's |R| > 0.70, P < 0.01) exist between major phytoplankton taxa and specific prokaryotes in the Daya Bay time series. Notably, significant correlations were observed between certain diatoms and members of the Alpha-, Gamma-, and Betaproteobacteria.

Furthermore, time-lagged correlations were also observed between dominant MGII (Marine Group II) archaea and diatoms such as Chaetoceros (Bacillariophyta) and Skeletonema (Dinophyta). These findings provide insights into the interactions between traditional phytoplankton and prokaryotes, offering a higher level of phylogenetic and temporal resolution.

Figure x illustrates the microbial association network in the 72-hour time series of Daya Bay, showing that the local similarity correlations with time-lags associated with major phytoplankton taxa often involve bacteria and archaea, indicating temporal delays.

**4 Discussion and Conclusions**

The computational workload, which previously took nearly a month using the traditional eLSA theoretical method, can be completed in just one day by fully leveraging hardware resources with the GeLSA software. This advancement significantly enhances the efficiency of data analysis, allowing us to process massive datasets rapidly. This breakthrough not only accelerates workflow but also aids in a deeper understanding and utilization of data. The Python-C++ implementation of GeLSA can be found at https://github.com/labxscut/GeLSA.

**References**

1. Caporaso JG, Lauber CL, Costello EK, *et al.* Moving pictures of the human microbiome. *Genome Biol* 2011;**12**:R50.

2. Cram JA, Xia LC, Needham DM, *et al.* Cross-depth analysis of marine bacterial networks suggests downward propagation of temporal changes. *ISME J* 2015;**9**:2573–86.

3. Steele JA, Countway PD, Xia L, *et al.* Marine bacterial, archaeal and protistan association networks reveal ecological linkages. *ISME J* 2011;**5**:1414–25.

4. Shade A, McManus PS, Handelsman J. Unexpected diversity during community succession in the apple f lower microbiome. *MBio* 2013;**4**:e00602–12.

5. Cho RJ, Campbell MJ, Winzeler EA, *et al.* A genome-wide transcriptional analysis of the mitotic cell cycle. *Mol Cell* 1998;**2**: 65–73.

6. Spellman PT, Sherlock G, Zhang MQ, *et al.* Comprehensive identification of cell cycle-regulated genes of the yeast Saccharomyces cerevisiae by microarray hybridization. *Mol Biol Cell*

1998;**9**:3273–97.

7. Amar D, Yekutieli D, Maron-Katz A, *et al.* A hierarchical Bayesian model for f lexible module discovery in three-way time-series data. *Bioinformatics* 2015;**31**:i17–26.

8. Vaisvaser S, Lin T, Admon R, *et al.* Neural traces of stress: cortisol related sustained enhancement of amygdala-hippocampal functional connectivity. *Front Hum Neurosci* 2013;**7**:313.

9. Durno WE, Hanson NW, Konwar KM, Hallam SJ. Expanding the boundaries of local similarity analysis. *BMC Genomics* 2013;**14**(Suppl 1):S3.

10. Qian J, Dolled-Filhart M, Lin J, *et al.* Beyond synexpression relationships: local clustering of time-shifted and inverted gene expression profiles identifies new. Biologically relevant interactions. *J Mol Biol* 2001;**314**:1053–66.

11. Xia LC, Ai D, Cram J, *et al.* Efficient statistical significance approximation for local similarity analysis of high-throughput time series data. *Bioinformatics* 2013;**29**:230–7.

12. Xia LC, Steele JA, Cram JA, *et al.* Extended local similarity analysis (eLSA) of microbial community and other time series data with replicates. *BMC Syst Biol* 2011;**5**:S15.

13. Ruan Q, Dutta D, Schwalbach MS, *et al.* Local similarity analysis reveals unique associations among marine bacterioplankton species and environmental factors. *Bioinformatics* 2006;**22**: 2532–8.

14. He F, Zeng AP. In search of functional association from timeseries microarray data based on the change trend and level of gene expression. *BMC Bioinformatics* 2006;**7**:69.

15. Ji L, Tan KL. Identifying time-lagged gene clusters using gene expression data. *Bioinformatics* 2005;**21**:509–16.

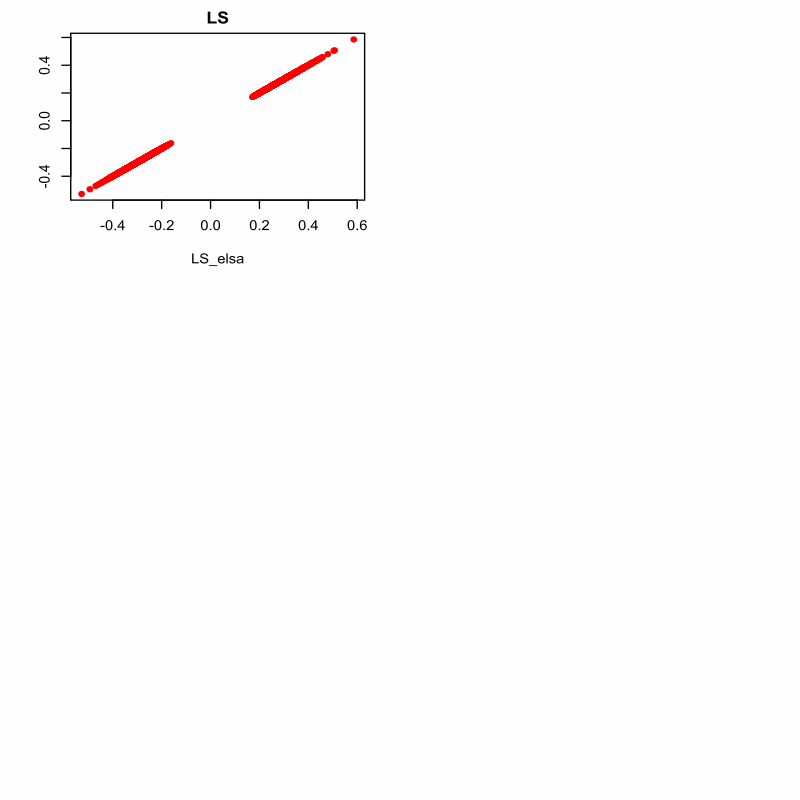
16. Xia LC, Ai D,Cram JA, *et al.* Statistical significance approximation in localtrend analysis of high-throughputtime-series data using the theory of Markov chains. *BMC Bioinformatics* 2015;**16**:301.

17. Zhang F, Shan A, Luan Y. A novel method to accurately calculate statistical significance of local similarity analysis for high-throughput time series. *Stat Appl Genet Mol Biol* 2018; **17**:20180019.

18. Zhang F, Sun F, Luan Y. Statistical significance approximation for local similarity analysis of dependent time series data. *BMC Bioinformatics* 2019;**20**:53.

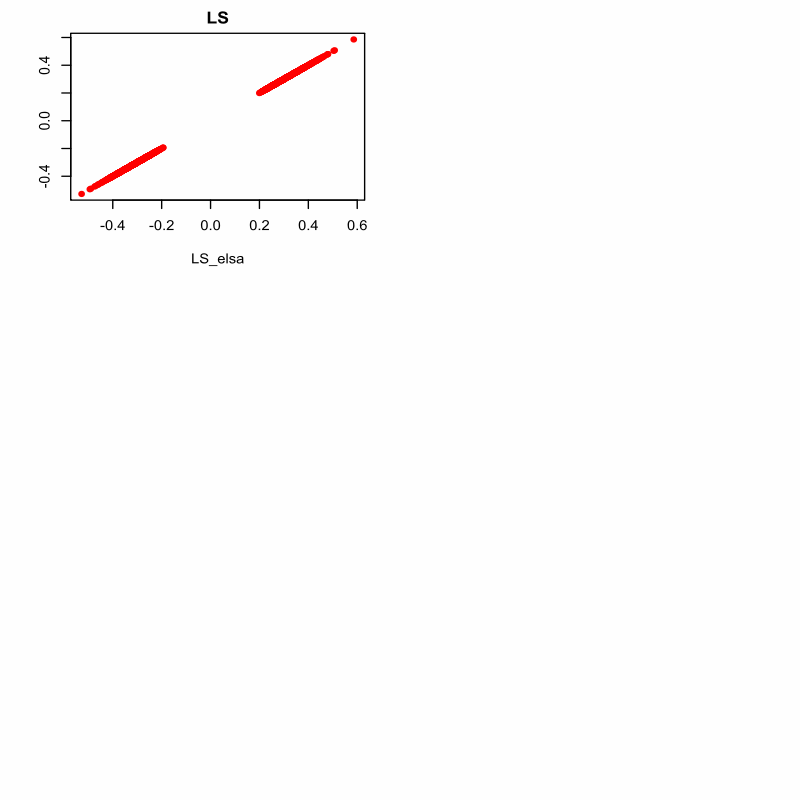
19. Shan A, Zhang F, Luan Y. Efficient approximation of statistical significance in local trend analysis of dependent time series. *Front Genet* 2022;**13**:729011

**附录**



There are a total of 6 subplots. Overall, it is evident from the plots that there is no deviation between the GeLSA's "theo" algorithm and the traditional eLSA software's "theo" algorithm when analyzing a dataset comprising 100 pairs of sequences which gengerated 4950 pieces of data.

It is clear that, in subplot A, the results of GeLSA(theo) and eLSA(theo) for LS (local similarity) calculations are consistently distributed along the diagonal line, forming a straight line composed of closely aligned points. This suggests that GeLSA performs LS calculations accurately.In subplots B, C, and D, the results of GeLSA(theo) and eLSA(theo) for the starting positions (xs, ys) of local maxima, as well as for the subsequence lengths (len), are consistently distributed along the diagonal lines, forming straight lines composed of closely aligned points. Similarly, in subplot E, the results of GeLSA(theo) and eLSA(theo) for p-values are consistently distributed along the diagonal line, indicating accurate calculations. This observation is consistent with the conclusion drawn from subplot A, further corroborating the software's accuracy.In subplot F, it is evident that this comparative experiment of software performance was conducted with a delay of 5.



There are a total of 6 subplots. Overall, it is evident from the plots that there is no deviation between the GeLSA's "theo" algorithm and the traditional eLSA software's "theo" algorithm when analyzing a dataset comprising 100 pairs of sequences which gengerated 4950 pieces of data.

It is clear that, in subplot A, the results of GeLSA(theo) and eLSA(theo) for LS (local similarity) calculations are consistently distributed along the diagonal line, forming a straight line composed of closely aligned points. This suggests that GeLSA performs LS calculations accurately.In subplots B, C, and D, the results of GeLSA(theo) and eLSA(theo) for the starting positions (xs, ys) of local maxima, as well as for the subsequence lengths (len), are consistently distributed along the diagonal lines, forming straight lines composed of closely aligned points. Similarly, in subplot E, the results of GeLSA(theo) and eLSA(theo) for p-values are consistently distributed along the diagonal line, indicating accurate calculations. This observation is consistent with the conclusion drawn from subplot A, further corroborating the software's accuracy.In subplot F, it is evident that this comparative experiment of software performance was conducted with a delay of 0.