NAR Genomics and Bioinformatics, 2024, **6**, lqae177
https://doi.org/10.1093/nargab/lqae177
Advance access publication date: 18 December 2024
Computational Methods for Transcriptome Analysis

PyNetCor: a high-performance Python package for large-scale correlation analysis

Shibin Long [1,†], Yan Xia [1,2,†], Lifeng Liang [1,†], Ying Yang [1], Hailiang Xie [1,*] and Xiaokai Wang [1,*]

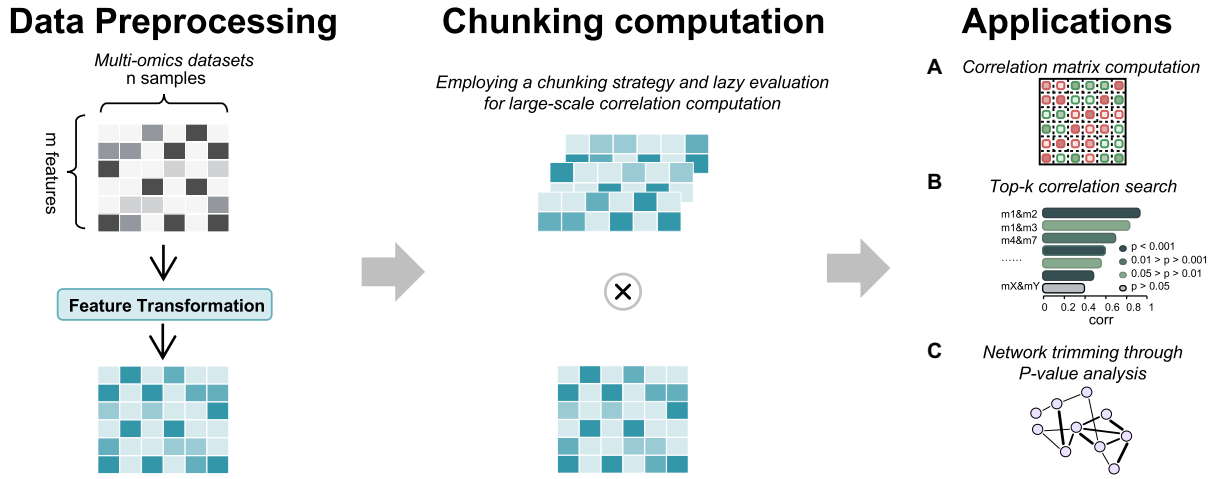
[1]Department of Data Science, 01Life Institute, Shenzhen 518000, China
[2]State Key Laboratory of Pharmaceutical Biotechnology, Chemistry and Biomedicine Innovation Center (ChemBIC), School of Life Sciences, Nanjing University, Nanjing 210023, China

[*]To whom correspondence should be addressed. Tel: +86 0755 86549757; Email: wangxiaokai@01lifetech.com
Correspondence may also be addressed to Hailiang Xie. Email: xiehailiang@01lifetech.com
[†]The first three authors should be regarded as Joint First Authors.

Abstract

The development of multi-omics technologies has generated an abundance of biological datasets, providing valuable resources for investigating potential relationships within complex biological systems. However, most correlation analysis tools face computational challenges when dealing with these high-dimensional datasets containing millions of features. Here, we introduce pyNetCor, a fast and scalable tool for constructing correlation networks on large-scale and high-dimensional data. PyNetCor features optimized algorithms for both full correlation coefficient matrix computation and top-k correlation search, outperforming other tools in the field in terms of runtime and memory consumption. It utilizes a linear interpolation strategy to rapidly estimate *P*-values and achieve false discovery rate control, demonstrating a speedup of over 110 times compared to existing methods. Overall, pyNetCor supports large-scale correlation analysis, a crucial foundational step for various bioinformatics workflows, and can be easily integrated into downstream applications to accelerate the process of extracting biological insights from data.

Graphical abstract



Introduction

Correlation analysis is widely employed in omics data to infer potential interaction networks within biological systems. In transcriptomics, researchers construct gene regulatory networks by analyzing the coordinated expression of different genes, enabling the identification of critical transcriptional regulatory mechanisms and functional modules (1–3). In proteomics, correlation quantifies the strength of interactions between proteins, facilitating the discovery of novel protein complexes and their functions through the analysis of protein activity or expression levels (4–6). In microbiomics, microbial co-occurrence relationships can be detected based on relative abundance, revealing complex interactions between microbial communities (7–9). Beyond applications in single omics data, correlation analysis can integrate multi-omics data to provide a comprehensive view of complex biological systems (10–12).

Over the past decade, the rapid development of multi-omics technologies has generated vast amounts of

Received: June 27, 2024. **Revised:** November 19, 2024. **Editorial Decision:** November 25, 2024. **Accepted:** November 28, 2024

biological data (13), enabling comprehensive analyses of biological processes and diseases. Many large-scale projects have collected multi-omics datasets containing over millions of measurements (14,15). The high dimensionality and complexity of such data pose significant computational challenges for correlation analysis.

Numerous powerful software packages have been developed for correlation analysis, including Numpy (16), Pandas (17), WGCNA (18), Cupy (19), RAPIDS (https://rapids.ai) and CorALS (20). Numpy and Pandas are standard computational libraries in data analysis. NumPy uses the C language for efficient multidimensional array operations and provides the numpy.corrcoef function to calculate the Pearson correlation matrix. Pandas implements a series of user-friendly statistical functions for tabular data structures. WGCNA focuses on the biomedical field and has been specifically optimized for Pearson correlation coefficient calculation. Additionally, Cupy, RAPIDS and CorALS have been developed to address large-scale data analysis. Cupy and RAPIDS utilize GPUs (Graphics Processing Units) to accelerate correlation computations but require dedicated hardware support and are subject to memory limitations. CorALS adopts a Numpy-based framework, combining dedicated vector projection with spatial partitioning techniques to achieve faster correlation matrix computation and top-k correlation approximation. While these software packages perform well in specific scenarios, they still have limitations, such as high memory consumption and dependence on hardware support.

To address these limitations, we developed pyNetCor, a high-performance and scalable correlation analysis tool for large-scale multi-omics data. PyNetCor provides two core modules: the correlation analysis module, which utilizes vector transformations and parallel matrix multiplication routines to enhance computational efficiency and incorporates a heap sort algorithm to achieve accurate top-k search, and the significance testing module, which employs a linear interpolation-based approximation algorithm to rapidly generate approximate *P*-values and enable false discovery rate (FDR) control without calculating all *P*-values. Furthermore, to enhance scalability for handling large-scale data, pyNetCor implements an out-of-core algorithm for chunked matrix, enabling partial processing of datasets and preventing memory exhaustion from constructing the entire correlation matrix simultaneously. Subsequently, we evaluate pyNetCor's performance on real-world metagenomic datasets with varying dimensions, and demonstrate its application potential by analyzing microbe-metabolite correlations in an inflammatory bowel disease (IBD) multi-omics database.

Materials and methods

Correlation coefficient matrix

PyNetCor supports the computation of Pearson, Spearman and Kendall correlation coefficients. Employing appropriate vector transformations, we can transform the computation of Pearson correlation coefficients between all feature pairs in a matrix to matrix multiplication (20). This transformation method not only streamlines the computation process but also enables the utilization of efficient linear algebra routines, such as OpenBLAS, effectively taking advantage of modern CPUs' multi-core processors and SIMD (Single Instruction Multiple Data) instructions, further enhancing computational efficiency. On this basis, pyNetCor arranges the data points within the feature in ascending order and assigns them rankings accordingly. Subsequently, it employs the optimized Pearson correlation technique to compute the Spearman correlation coefficient, thereby enhancing the efficiency of diverse correlation assessments across a broader spectrum.

The implementation for Kendall's tau correlation coefficient in pyNetCor is based on the cor.fk function from pcaPP-2.0.3 (21). This function simplifies the computation process by employing the maximum likelihood estimator, optimizing the algorithm's time complexity from $\mathcal{O}(n^2)$ to $\mathcal{O}(n \log n)$, leading to significant improvements in performance for high-dimensional data correlation analysis.

Approximate calculation of *P*-values and adjusted *P*-values for multiple hypothesis testing

The calculation of P-values for Pearson, Spearman and Kendall correlation coefficients relies on the relationship between the correlation statistic and the cumulative distribution function (CDF). Let's focus on the Pearson correlation coefficient r for a pair of features with n samples. The *P*-value associated with r can be determined using the t-distribution CDF (pt) as shown in Equation (1) (22):

$$P = pt\left(|\, t \,|,\ n-2\right) \tag{1}$$

where the t-statistic is derived using $\mathrm{t} = \frac{r\cdot\sqrt{n-2}}{\sqrt{1-r^2}}$.

To accelerate the computation of complex CDFs, we construct a pre-computed mapping table between *P*-values and t-statistics. This involved L (>10 000) *P*-values ($\frac{1}{L}$, ..., 1) and obtaining their associated square of correlations (r_1^2, ..., r_L^2). Given a new square of correlations, an approximate *P*-value can be estimated through linear interpolation within this table.

$$\mathrm{P}' = \begin{cases} \frac{r_{i+1}^2 - r^2}{r_{i+1}^2 - r_i^2} P_i + \frac{r^2 - r_i^2}{r_{i+1}^2 - r_i^2} P_{i+1}, & r_{i+1}^2 \le r^2 < r_i^2,\ 1 \le i < L \\ b + \frac{1-r^2}{1-r_1^2}\left(P_1 - b\right), & r^2 \ge r_1^2 \end{cases} \tag{2}$$

Where $0 < b \ll \frac{1}{L}$.

Building on these findings, pyNetCor further employs a similar approach for optimizing multiple hypothesis testing correction of *P*-values. It provides five correction methods: Bonferroni, Holm (23), Hochberg (24), Benjamini-Hochberg (BH) (25) and Benjamini-Yekutieli (BY) (26). Specifically, it randomly selects M (>500 000) tests, calculates the *P*-values and sorts them as ($P_{(1)} \le \ldots \le P_{(M)}$). Subsequently, the corresponding approximate corrected *P*-values ($P'_{c(1)} \le \ldots \le P'_{c(M)}$) are obtained for each original $P_{(i)}$ using a chosen *P*-value correction method. It is worth noting that this approach approximates the population size N using the smaller sample size M. Most methods require an adjustment to the *P*-value correction method. The Bonferroni, Holm's and Hochberg's methods necessitate multiplying the *P*-value by $\frac{N}{M+1}$ before correction, whereas the BH method requires multiplying the *P*-value by $\frac{M+1}{M}$. Furthermore, the BY method entails multiplying the *P*-value by $\frac{M+1}{N}$ and considers N as the number of tests for the calculation. Similar to the previous method, linear interpolation can be used to compute an approximate corrected *P*-value for any new given true *P*-value.

$$P_c = \begin{cases} \frac{P}{P_{(1)}} P'_{c(1)}, & if\ P \le P_{(1)} \\ \frac{P_{(i+1)} - P}{P_{(i+1)} - P_{(i)}} P'_{c(i)} + \frac{P - P_{(i)}}{P_{(i+1)} - P_{(i)}} P'_{c(i+1)}, & if\ P_{(i)} < P \le P_{(i+1)}, 1 \le i \le M \\ P'_{c(M)}, & if\ P > P_{(M)} \end{cases} \tag{3}$$

Top-k correlation and differential correlation selection

In high-dimensional datasets, numerous weak correlations can arise between features due to random noise. Consequently, we prioritize focusing on the core top-k strongest correlations. The workflow of top-k correlation selection is shown in Figure 1. We employ a high-performance, memory-efficient chunking algorithm to iterate through all correlations (Supplementary Method S1) and utilize a min-heap of size k to maintain the top-k largest correlations encountered thus far. Subsequently, we sort these top-k correlations using the heap-sort algorithm (27). The average time complexity of the top-k search process is $\mathcal{O}(n \log k)$, and the space complexity is $\mathcal{O}(k)$.

Differential correlation analysis examines changes in correlation between feature pairs across two states or time periods, where the most significant correlation patterns may be implicated in key biological processes. Here, we calculate the correlation matrices for both conditions in chunks before computing their difference. The remaining top-k search and sorting methods are identical to the top-k correlation calculation method.

Canopy-based clustering

Clustering algorithms can automatically uncover distribution patterns in data, reducing the scope of correlation computation. This facilitates the understanding of functional modules and interactions within biological networks. PyNetCor seamlessly integrates an optimized C++ implementation of the mgs-canopy-algorithm (28). Building upon this foundation, pyNetCor employs efficient correlation algorithms (see 'Materials and methods' section) to accelerate the calculation of similarity between clustered entities.

Missing value processing

Traditional correlation methods are inapplicable to datasets with missing values. To address this, pyNetCor provides two missing value handling strategies for analyzing datasets with only a small number of missing values. These strategies are as follows: (i) replacing missing values with the mean or median of the non-missing samples for the respective feature. (ii) Sample pairs containing missing values are excluded from the correlation calculation. Only sample pairs with complete data are used to compute the correlation coefficient. By default, pyNetCor automatically selects the appropriate missing value handling strategy based on the selected correlation method, thereby preventing anomalies caused by missing values.

Datasets

To assess pyNetCor's performance, we employed two real-world datasets: an in-house metagenomic dataset and 16S rRNA gene sequencing data from the Earth Microbiome Project (EMP). Due to the limitations of certain analytical tools in handling missing values, we excluded features with missing data. Comprehensive dataset statistics can be found in Supplementary Table S1.

The metagenomic HUMAnN3 gene families (MHGF) dataset was generated using HUMAnN3 (29) to quantify the functional profiles of microbial communities present in over 500 metagenomic samples. This process involves the mapping of DNA reads to the UniProt Reference Clusters database (UniRef90). Due to the plenty of zeros in metagenomic data, we excluded features with zero variance or zero values exceeding 95% to alleviate the influence of zero inflation on the data.

The EMP dataset comprises 96 studies from diverse global environments, encompassing 23 828 samples (30). Through 16S rRNA gene sequencing and amplicon data analysis, researchers generated an operational taxonomic unit (OTU) table containing 307 572 features. The table is accessible at ftp://ftp.microbio.me/emp/release1/otu_tables. We utilized all samples to conduct experiments and evaluate pyNetCor's performance with large sample sizes.

To demonstrate the application of pyNetCor, we obtained an IBD dataset from the Integrative Human Microbiome Project (HMP2 or iHMP) through the Inflammatory Bowel Disease Multi'omics Database (IBDMDB) website (https://ibdmdb.org) (31). The dataset contains microbiome and metabolome profiles of 181 Crohn's disease (CD) samples, 102 ulcerative colitis (UC) samples and 105 non-IBD control samples.

Performance evaluation

We compared the runtime and memory consumption of pyNetCor-0.1.0, CorALS-0.1.6, NumPy-1.24.0, pandas-1.5.2 and WGCNA-1.71. All experiments were conducted on a CentOS server with an AMD EPYC 9754 CPU @ 2.3GHz, 32 cores and 128GB RAM. Additionally, we evaluated pyNetCor's performance in resource-constrained environments, with tests performed on a personal computer with an Intel (R) Core (TM) i7-10700 CPU @ 2.9GHz under Windows 11 operating system, 8 cores and 32GB RAM. Each test was executed either 3 or 10 times, and the median running time and peak memory usage were recorded as the definitive outcomes.

Results

Memory-efficient construction of large-scale correlation networks

Constructing correlation networks for million-scale datasets requires over 1 TB of storage space, and memory consumption increases quadratically with the number of features, far exceeding the storage capacity of high-performance computing clusters. To address this challenge, pyNetCor optimizes memory efficiency using chunked matrix multiplication and lazy computation techniques, enabling large-scale computations on small-memory machines and even personal computers. We used six high-dimensional datasets downsampled from the MHGF dataset, with feature numbers ranging from 0.05 to 1.6 million. The Pearson correlation matrix was calculated using the chunked method on a personal computer equipped with 8 cores and 32 GB RAM (Table 1). The results demonstrated that pyNetCor enables users to construct large-scale correlation networks without relying on expensive high-performance computing resources. Specifically, when processing a dataset with 1.6 million features, computing the full correlation matrix theoretically requires over 20TB of memory. However, using pyNetCor with a single core, only 22 GB of memory is required, and the calculation can be completed within 9 h. Furthermore, in multi-threaded mode, pyNetCor can further enhance computational efficiency, reducing the runtime to 3.5 h.

Since the chunked technique to divide a large matrix into smaller matrices of a specified size, the correlation of each

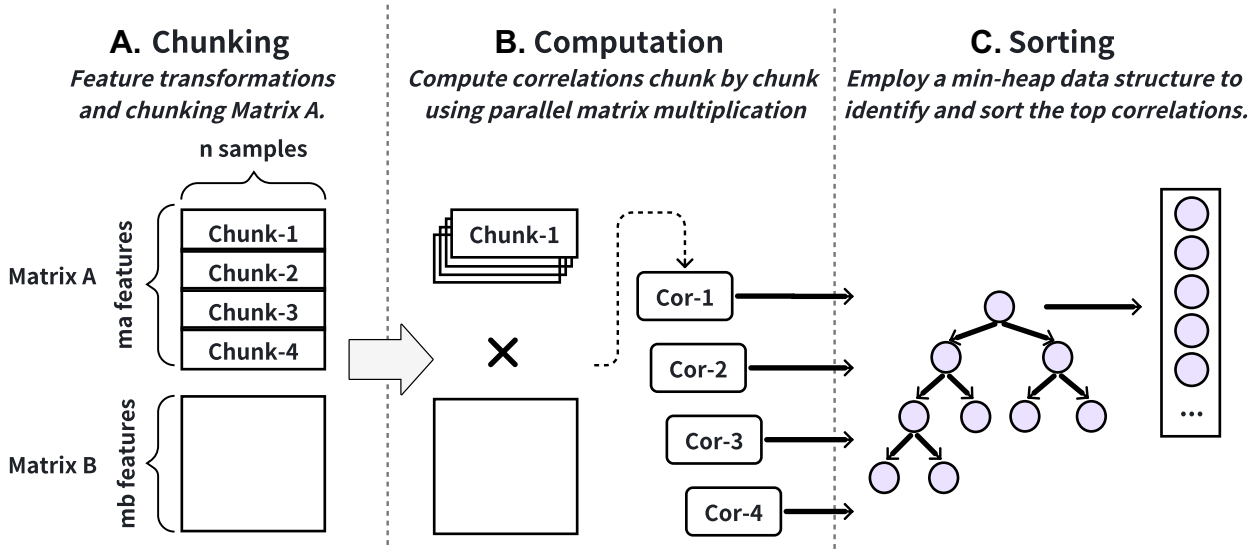


Figure 1. PyNetCor workflow for top-k correlation selection. Matrix A and B represent the input data, with rows corresponding to features (e.g. genes and species) and columns to samples. Matrix B may be identical to matrix A. (**A**) Preprocess and transform the input matrices, dividing matrix A into multiple smaller chunks along the feature dimension while maintaining matrix B intact. (**B**) Compute correlations using one sub-matrix of A and the complete matrix B at a time, sequentially processing the remaining sub-matrices to optimize memory usage. (**C**) Finally, a min-heap structure is used to store the highest-scoring correlation results, and heap sort is employed to obtain the top-k correlations.

Table 1. Runtime and memory usage for Pearson correlation on personal computer

Features	Chunked Time (min)	Chunked_parallel Time (min)	Chunked Memory (GB)	Chunked_parallel Memory (GB)
50 000	0.54	0.26	0.9	0.9
100 000	2.15	1.02	1.61	1.6
200 000	8.45	3.75	2.99	2.99
400 000	33.11	13.68	5.79	5.78
800 000	134.37	52.61	11.37	11.37
1 600 000	542.7	207.97	22.51	22.53

small matrix is calculated and the result returned only when requested, thus achieving controllable memory usage. Therefore, the runtime may increase in chunked computation mode. To evaluate the performance of chunked computation, we measured its runtime and that of the conventional method using varying numbers of threads. As illustrated in Figure 2, the runtime of chunked computation is ∼0.6–24.2% higher than that of the conventional method, primarily due to the additional overhead of multiple memory allocations and deallocations. However, these minor runtime increases are acceptable given the significant reduction in memory consumption achieved by this method.

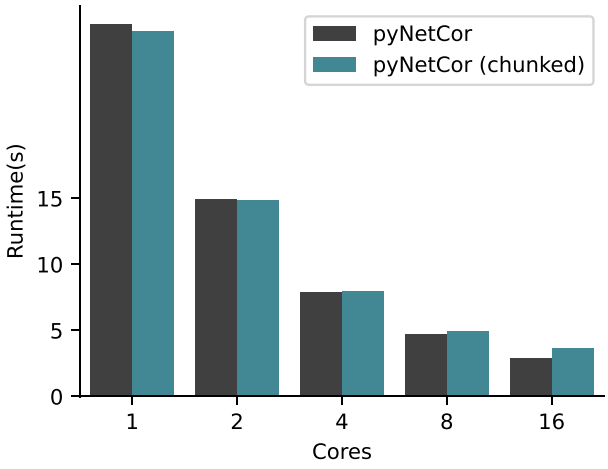


Figure 2. Runtime experiments for correlation computation using the chunked method. Compare the runtime of the pyNetCor conventional method and the chunked method for calculating the Pearson correlation matrix of a metagenomic dataset with 50 000 features and 500 samples. Runtime measurements were performed using 1–16 cores.

Efficient and robust correlation matrix computation

To validate pyNetCor's performance in correlation analysis, we compared it against existing tools, including CorALS, Numpy, Pandas and WGCNA. As the single-core implementations of pyNetCor and CorALS exhibited superior performance compared to the other tools, we further extended the comparison to multi-threaded environments utilizing 32 cores. Figure 3 illustrates the runtime for calculating Spearman correlation matrices using various tools when processing datasets with different numbers of features and samples (Supplementary Tables S3 and S5). For the MHGF dataset (70 000 features), only pyNetCor and CorALS can complete the computation within 10 min, with pyNetCor consistently demonstrating superior performance. In multi-threaded mode, pyNetCor requires only 21.1 s, >3 times faster than CorALS and 260 times faster than Pandas. It is worth noting that these performance gaps are expected to widen further as the sample size increases. For instance, on the EMP dataset (70 000 features, >20 000 samples), pyNetCor was 4.3 times faster than CorALS and 1816 times faster than Pandas. Additionally, pyNetCor supports the computation of Pearson correlation matrices, with the running time analysis presented in Supplementary Figure S1 (Supplementary Tables S2 and S4). While pyNetCor and NumPy show comparable runtimes on large-sample datasets (>1000 samples), pyNetCor performs slightly better on small samples (<500 samples).

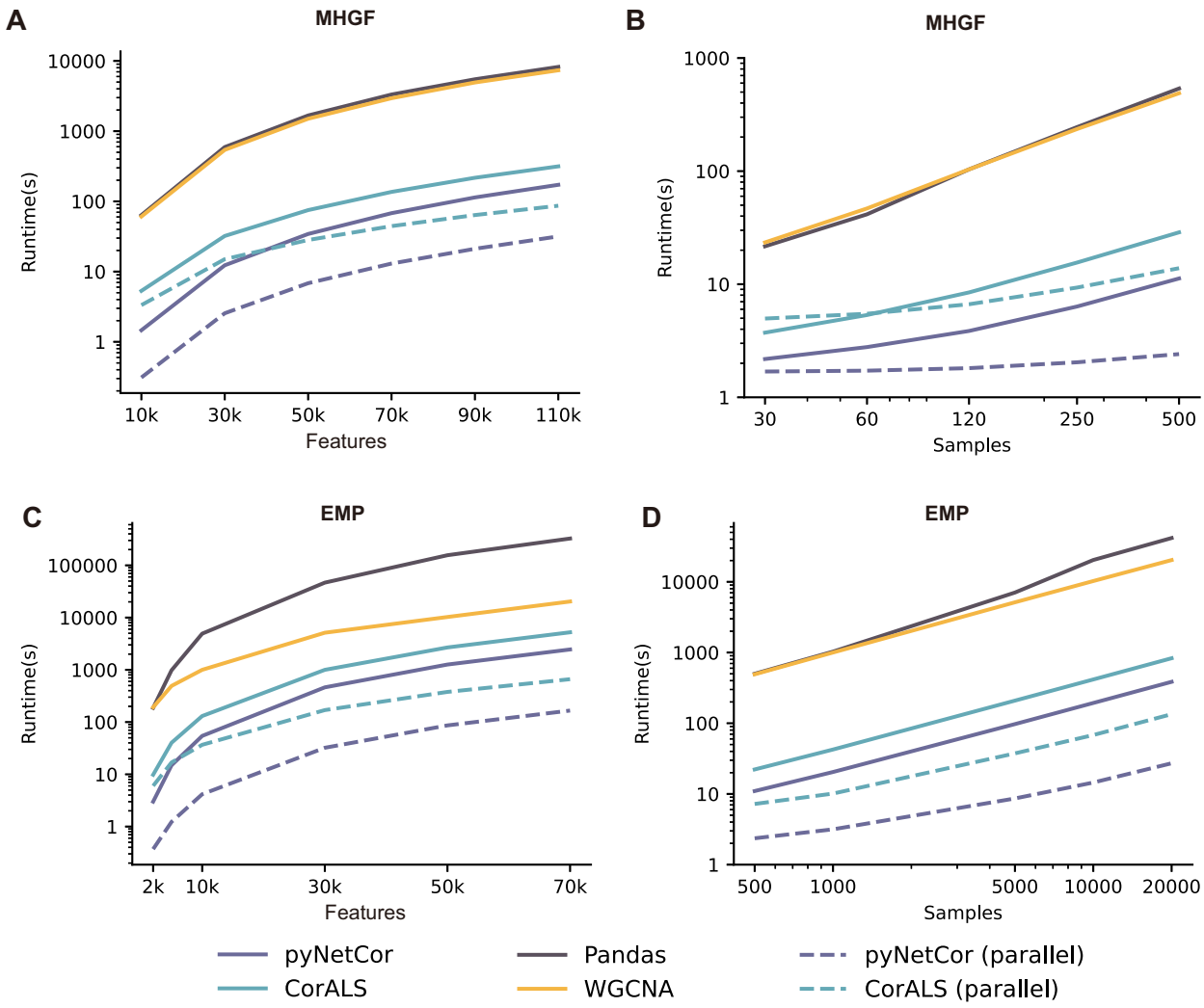


Figure 3. Benchmarking of Spearman correlation matrix computation. Runtime comparisons of pyNetCor and other tools for calculating Spearman correlation matrices on (**A** and **B**) MHGF and (**C** and **D**) EMP datasets. Subfigures show performance with (A and C) varying feature numbers and (B and D) varying sample numbers. Dashed lines represent parallel computations using 32 threads, while solid lines indicate computations using a single thread.

Efficient and accurate top-k correlation search

Complete correlation matrices are often too large for comprehensive examination. PyNetCor employs top-k analysis to search for the k pairs of associations most likely to possess statistical significance. Next, we further evaluate pyNetCor's performance in top-k correlation search. We directly implemented the top-k selection method using Numpy and WGCNA, that is, first calculate the full correlation matrix, and then obtain the top-k correlations through an efficient sorting algorithm. Due to the substantial memory consumption of this approach, we were unable to run the direct implementation based on NumPy and WGCNA on datasets with >70 000 features (Figure 4B and Supplementary Table S7). Therefore, pyNetCor provides a top-k correlation selection implementation that uses chunked computation and a partial sorting algorithm based on heaps, while CorALS extracts approximate top-k correlations through feature vector projection and ball trees. On the MHGF dataset, pyNetCor required only 150 s of runtime and 2.3 GB of memory to search the top 1% of correlations among 70 000 features using a single thread (Figure 4A and B; Supplementary Tables S6 and S7), which is 34 times faster than CorALS, and the maximum memory usage was only 39% of CorALS. A similar trend was observed on the EMP dataset (Figure 4C and D; Supplementary Tables S8 and S9). It is worth noting that even in the case of multi-core parallelization to accelerate the calculation, pyNetCor maintains a very small memory consumption.

Efficient approximation for *P*-value and multiple test correction

Constructing large correlation networks often requires hypothesis testing of correlation coefficients to determine the reliability of correlation relationships. Existing tools primarily focus on the computational performance of correlation matrix calculation, resulting in low efficiency for *P*-value calculation and multiple testing correction. Therefore, pyNetCor utilizes linear interpolation to estimate *P*-values and adjusted *P*-values, thereby achieving FDR control without full *P*-values computation and demonstrably enhancing performance. To compare the runtime of the approximate *P*-value method implemented by pyNetCor and the classical *P*-value method, we processed the Pearson correlation matrix for the MHGF dataset and created six new datasets (with sizes of 10, 30, 50, 70, 90 and 110 million correlation coefficients) by down-

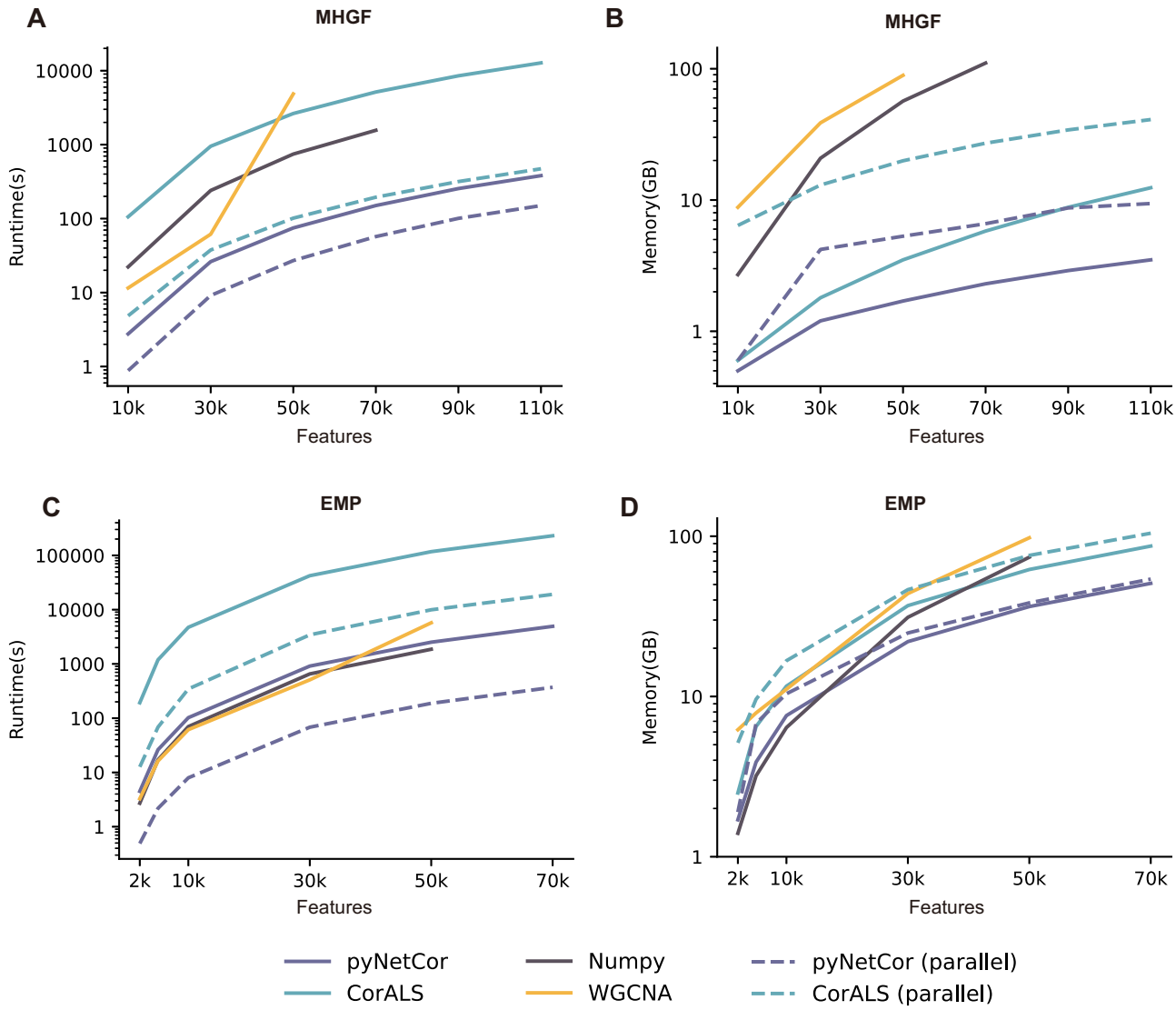


Figure 4. Benchmarking of top-k correlation search. Comparisons of pyNetCor and other tools for searching the top 1% correlations on (**A** and **B**) MHGF and (**C** and **D**) EMP metagenomic datasets with varying feature numbers. Subfigures show (A and C) runtime and (B and D) peak memory usage. Missing lines indicate program crashes due to exceeding system memory limits. Dashed lines represent parallel computations using 32 threads, while solid lines indicate computations using a single thread.

sampling. We used derive_pvalues implemented in CorALS as the classic *P*-value approach for comparison. With an efficient parallelized linear interpolation engine, pyNetCor can perform 110 million approximate *P*-value computations in 3.36 s, which is 118 times faster than the CorALS implementation that does not support multithreading (Figure 5A). Beyond its exceptional performance, the approximate algorithm also demonstrates high accuracy. On the same dataset, the absolute error between the approximate *P*-values calculated by pyNetCor for Pearson, Spearman and Kendall correlation coefficients and the true *P*-values consistently remains below 1e-8 (Figure 5B). Furthermore, the absolute error of the approximate corrected *P*-values is <5e-4 (Figure 5C).

PyNetCor facilitates multi-omics integrated analysis in IBD

IBD is a chronic, relapsing inflammatory condition of the intestine, primarily comprising UC and CD (32). The genetic basis of IBD remains poorly understood. In recent years, multiple studies have utilized multi-omics integration analysis to explore the pathogenesis of IBD. To validate the application of pyNetCor in multi-omics data correlation analysis, we analyzed the microbiome and metabolome data from the IBDMDB database. Using non-IBD samples as controls, we calculated the differential Spearman correlation between all microbial–metabolite pairs compared to disease samples using pyNetCor and extracted the top 1% differentially abundant microbes and metabolites for visualization (Figure 6A). The results revealed that the correlation structure of 10 microbes was significantly altered in IBD patients. Among them, *Roseburia inulinivorans* and *Faecalibacterium prausnitzii* are known butyrate-producing bacteria (33,34). Butyrate, a short-chain fatty acid with anti-inflammatory properties, has been implicated in maintaining intestinal barrier function and suppressing IBD development (35).

To identify the metabolites most relevant to the 10 differential microbes, we explored the relationships between

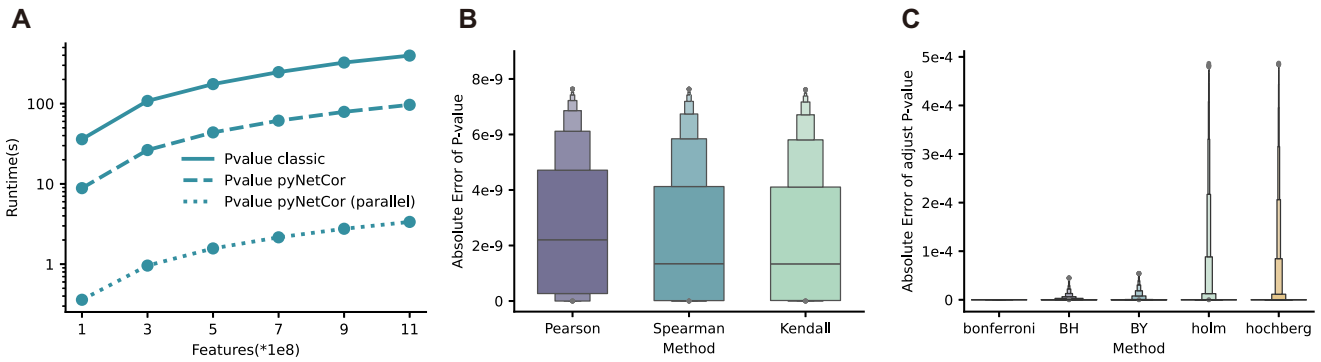


Figure 5. Evaluation of *P*-value Approximation and Multiple Testing Correction in pyNetCor. (**A**) Runtime comparison between the pyNetCor's *P*-value approximation method and the CorALS's classic method across correlation coefficient datasets of varying sizes. Parallel computations using 32 threads are shown, with pyNetCor being the only method capable of parallel computation for *P*-values. (**B**) Distribution of absolute errors for the approximate *P*-values computed by pyNetCor for Pearson, Spearman and Kendall correlations, with all absolute errors <1e-8. (**C**) Distribution of absolute errors for multiple testing correction using the approximate methods provided by pyNetCor: Bonferroni, BH, BY, Holm and Hochberg. No significant error is observed for the approximate Bonferroni correction, and all other methods have absolute errors <5e-4.

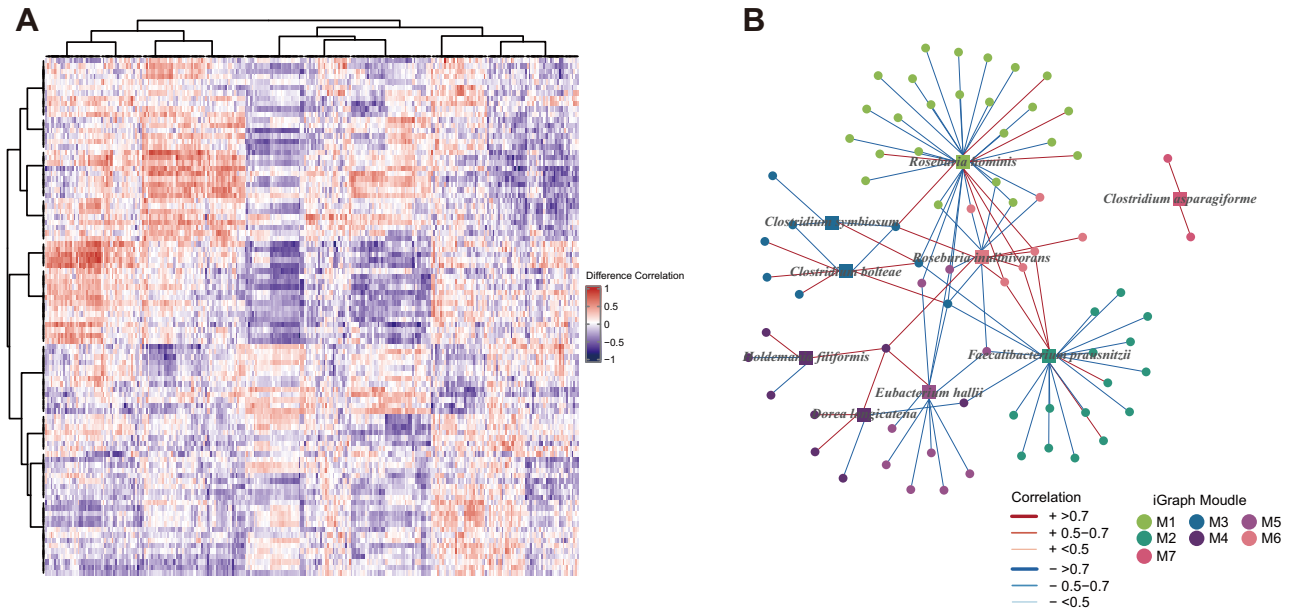


Figure 6. Network analysis of microbe-metabolite interactions in IBD. (**A**) Heatmap of differences in pairwise Spearman correlations between disease and control samples for omics data (microbiome and metabolome) from the IBDMDB database. Top 1% microbe-metabolite pairs with the largest correlation differences are shown, identifying 10 differential microbes. Red indicates increased differential correlation in disease conditions compared to controls, while blue indicates decreased differential correlation. (**B**) Significant association network (FDR < 0.05) for the 10 differential microbes and metabolites. Rectangular nodes represent microbes, and circular nodes represent metabolites. Red edges denote positive correlations, and blue edges indicate negative correlations, with deeper colors representing stronger associations.

these microbes and metabolites using Spearman correlation analysis. The analysis results revealed that a total of 103 microbial–metabolite pairs exhibited significant correlation (FDR < 0.05). As illustrated in Figure 5B, more than fifteen acylcarnitines were significantly correlated with *Roseburia hominis*. In particular, the negative correlation between C18:1-OH carnitine, C16-OH carnitine, C14:1 carnitine and *Roseburia hominis* was the most significant. It is worth noting that previous studies have observed a decreased abundance of *Roseburia hominis* in patients with IBD (31,36). This suggests that acylcarnitines may be key metabolites in IBD and have the potential to become novel therapeutic targets for this disease, providing important clues for further research.

Discussion

In this study, we describe pyNetCor, a framework for correlation and network analysis of multi-omics data. PyNetCor implements efficient correlation calculations and a top-k correlation network construction algorithm, significantly outperforming existing tools in terms of analysis time and computational cost. By employing a specialized chunked strategy for memory optimization, pyNetCor can process datasets with 1.6 million features using only 22GB of memory in 3.5 h.

Correlation analysis often involves hypothesis testing of correlation coefficients to assess the strength of the correlation relationship. However, most methods lack explicit control over the FDR, leading to reduced statistical power with an in-

creasing number of tests and possibly yielding no significant findings. Therefore, pyNetCor adopts an innovative linear interpolation strategy to quickly estimate *P*-values, achieving a speed improvement of one order of magnitude over traditional algorithms. Similarly, pyNetCor utilizes approximation methods to achieve effective FDR control before computing all *P*-values, which is crucial for maintaining statistical power and avoiding false discoveries in large-scale data analysis.

Integrative analysis of multi-omics data can provide a more comprehensive biological perspective, revealing cross-scale molecular regulatory mechanisms. Currently, pyNetCor only supports correlation analysis between two omics datasets. For complex studies involving multiple high-dimensional data types, manual combination can extend it to multiple correlated datasets, as in (31). Future iterations of pyNetCor may integrate this method and apply it to top-k and differential correlation network analysis, providing users with a fully automated multi-omics data integration solution.

At present, pyNetCor supports analysis based on Pearson, Spearman and Kendall correlation coefficients. However, for certain special data types, introducing other types of correlation coefficients may be necessary to quantify the strength of relationships between features. For example, MIC can broadly capture both linear and nonlinear associations between variables and provides a score similar to the coefficient of determination (R^2) in regression (37–39). It is commonly used to identify potential relationships in large datasets. SparCC is designed for the compositional nature of microbiome data (40–42). It utilizes log-ratio transformations and iterative calculations to infer correlations, effectively correcting spurious correlations. Extending to more advanced correlation methods can further enhance pyNetCor's analytical performance and application range.

Furthermore, Table 1 illustrates pyNetCor's superior performance in handling high-dimensional datasets. By implementing a chunked strategy, pyNetCor maintains linear memory usage as the feature count increases, significantly improving the algorithm's scalability. However, practical applications are constrained by computational resources. For datasets comprising tens of millions of features, users may need to manually adjust the chunk size based on available RAM. In cases where a single machine's memory is insufficient, distributed computing methods can be employed to ensure optimal computational performance on ultra-large datasets.

Finally, due to the simplicity and computational efficiency of pyNetCor, it can be easily integrated into complex network models and downstream applications, providing underlying computational support. One potential application involves combining pyNetCor's correlation matrix calculations with random matrix theory to construct molecular ecological networks, thereby augmenting the performance of MENA (43,44). We expect that pyNetCor will evolve into a modularly integratable fundamental correlation analysis tool in the future, ultimately facilitating in-depth exploration of complex biological systems.

Data availability

The pyNetCor package and analyzed data can be found at https://github.com/01life/pyNetCor and https://doi.org/10.5281/zenodo.14177447.

Supplementary data

Supplementary Data are available at NARGAB Online.

Acknowledgements

Author contributions: S.L. and L.L., under the supervision of X.W., Y.X. and H.X., designed and developed PyNetCor. Data curation was handled by L.L. and Y.Y. S.L. drafted the manuscript, with Y.X. providing critical feedback on the overall structure and arguments. Contributions to the writing were also made by L.L., H.X. and X.W.

Funding

Shenzhen 01 Life Technology Co., Ltd.

Conflict of interest statement

None declared.

References

1. Stuart,J.M., Segal,E., Koller,D. and Kim,S.K. (2003) A gene-coexpression network for global discovery of conserved genetic modules. *Science*, **302**, 249–255.
2. Yang,Y., Han,L., Yuan,Y., Li,J., Hei,N. and Liang,H. (2014) Gene co-expression network analysis reveals common system-level properties of prognostic genes across cancer types. *Nat. Commun.*, **5**, 3231.
3. van Dam,S, Võsa,U, van der Graaf,A, Franke,L and de Magalhães,J.P. (2018) Gene co-expression analysis for functional classification and gene–disease predictions. *Brief. Bioinf.*, **19**, 575–592.
4. Stelzl,U., Worm,U., Lalowski,M., Haenig,C., Brembeck,F.H., Goehler,H., Stroedicke,M., Zenkner,M., Schoenherr,A., Koeppen,S., *et al.* (2005) A human protein-protein interaction network: a resource for annotating the proteome. *Cell*, **122**, 957–968.
5. Sharan,R., Ulitsky,I. and Shamir,R. (2007) Network-based prediction of protein function. *Mol. Syst. Biol.*, **3**, 88.
6. Huttlin,E.L., Bruckner,R.J., Navarrete-Perea,J., Cannon,J.R., Baltier,K., Gebreab,F., Gygi,M.P., Thornock,A., Zarraga,G., Tam,S., *et al.* (2021) Dual proteome-scale networks reveal cell-specific remodeling of the human interactome. *Cell*, **184**, 3022–3040.
7. Arumugam,M., Raes,J., Pelletier,E., Le Paslier,D., Yamada,T., Mende,D.R., Fernandes,G.R., Tap,J., Bruls,T., Batto,J.-M., *et al.* (2011) Enterotypes of the human gut microbiome. *Nature*, **473**, 174–180.
8. Faust,K., Sathirapongsasuti,J.F., Izard,J., Segata,N., Gevers,D., Raes,J. and Huttenhower,C. (2012) Microbial co-occurrence relationships in the human microbiome. *PLoS Comput. Biol.*, **8**, e1002606.
9. Ma,B., Wang,Y., Ye,S., Liu,S., Stirling,E., Gilbert,J.A., Faust,K., Knight,R., Jansson,J.K., Cardona,C., *et al.* (2020) Earth microbial co-occurrence network reveals interconnection pattern across microbiomes. *Microbiome*, **8**, 82.
10. Hasin,Y., Seldin,M. and Lusis,A. (2017) Multi-omics approaches to disease. *Genome Biol.*, **18**, 83.
11. do Valle,Í.F., Menichetti,G., Simonetti,G., Bruno,S., Zironi,I., Durso,D.F., Mombach,J.C.M., Martinelli,G., Castellani,G. and Remondini,D. (2018) Network integration of multi-tumour omics data suggests novel targeting strategies. *Nat. Commun.*, **9**, 4514.
12. Vandereyken,K., Sifrim,A., Thienpont,B. and Voet,T. (2023) Methods and applications for single-cell and spatial multi-omics. *Nat. Rev. Genet.*, **24**, 494–515.

13. Shendure,J., Balasubramanian,S., Church,G.M., Gilbert,W., Rogers,J., Schloss,J.A. and Waterston,R.H. (2017) DNA sequencing at 40: past, present and future. *Nature*, **550**, 345–353.
14. Huttenhower,C., Gevers,D., Knight,R., Abubucker,S., Badger,J.H., Chinwalla,A.T., Creasy,H.H., Earl,A.M., FitzGerald,M.G., Fulton,R.S., *et al.* (2012) Structure, function and diversity of the healthy human microbiome. *Nature*, **486**, 207–214.
15. Li,J., Jia,H., Cai,X., Zhong,H., Feng,Q., Sunagawa,S., Arumugam,M., Kultima,J.R., Prifti,E., Nielsen,T., *et al.* (2014) An integrated catalog of reference genes in the human gut microbiome. *Nat. Biotechnol.*, **32**, 834–841.
16. Harris,C.R., Millman,K.J., van der Walt,S.J., Gommers,R., Virtanen,P., Cournapeau,D., Wieser,E., Taylor,J., Berg,S., Smith,N.J., *et al.* (2020) Array programming with NumPy. *Nature*, **585**, 357–362.
17. McKinney,W. (2011) pandas: a foundational Python library for data analysis and statistics. *Python High Perform. Sci. Comput.*, **14**, 1–9.
18. Langfelder,P. and Horvath,S. (2008) WGCNA: an R package for weighted correlation network analysis. *BMC Bioinf.*, **9**, 559.
19. Okuta,R., Unno,Y., Nishino,D., Hido,S. and Loomis,C. (2017) CuPy: a NumPy-compatible library for NVIDIA GPU calculations. In: *Proceedings of workshop on machine learning systems(LearningSys) in the thirty-first annual conference on neural information processingsystems (NIPS)*. Long Beach, CA, USA,. Vol. **151**.
20. Becker,M., Nassar,H., Espinosa,C., Stelzer,I.A., Feyaerts,D., Berson,E., Bidoki,N.H., Chang,A.L., Saarunya,G., Culos,A., *et al.* (2023) Large-scale correlation network construction for unraveling the coordination of complex biological systems. *Nat. Comput. Sci.*, **3**, 346–359.
21. Todorov,V. and Filzmoser,P. (2013) Comparing classical and robust sparse PCA. In: Kruse,R., Berthold,M.R., Moewes,C., Gil,M.Á., Grzegorzewski,P. and Hryniewicz,O. (eds.) *Synergies of Soft Computing and Statistics for Intelligent Data Analysis*, Advances in Intelligent Systems and Computing. Springer, Berlin, Heidelberg, pp. 283–291.
22. Rahman,N. (1968) In: *A Course in Theoretical Statistics*. Charles Griffin and Company, London.
23. Holm,S. (1979) A simple sequentially rejective multiple test procedure. *Scand. J. Stat.*, **6**, 65–70.
24. HOCHBERG,Y. (1988) A sharper Bonferroni procedure for multiple tests of significance. *Biometrika*, **75**, 800–802.
25. Benjamini,Y. and Hochberg,Y. (1995) Controlling the false discovery rate: a practical and powerful approach to multiple testing. *J. R. Stat. Soc. Series B Stat. Methodol.*, **57**, 289–300.
26. Benjamini,Y. and Yekutieli,D. (2001) The control of the false discovery rate in multiple testing under dependency. *Ann. Stat.*, **29**, 1165–1188.
27. Carlsson,S. (1987) Average-case results on heapsort. *BIT Num. Mathem.*, **27**, 2–17.
28. Nielsen,H.B., Almeida,M., Juncker,A.S., Rasmussen,S., Li,J., Sunagawa,S., Plichta,D.R., Gautier,L., Pedersen,A.G., Le Chatelier,E., *et al.* (2014) Identification and assembly of genomes and genetic elements in complex metagenomic samples without using reference genomes. *Nat. Biotechnol.*, **32**, 822–828.
29. Beghini,F., McIver,L.J., Blanco-Míguez,A., Dubois,L., Asnicar,F., Maharjan,S., Mailyan,A., Manghi,P., Scholz,M., Thomas,A.M., *et al.* (2021) Integrating taxonomic, functional, and strain-level profiling of diverse microbial communities with bioBakery 3. *eLife*, **10**, e65088.
30. Thompson,L.R., Sanders,J.G., McDonald,D., Amir,A., Ladau,J., Locey,K.J., Prill,R.J., Tripathi,A., Gibbons,S.M., Ackermann,G., *et al.* (2017) A communal catalogue reveals Earth's multiscale microbial diversity. *Nature*, **551**, 457–463.
31. Lloyd-Price,J., Arze,C., Ananthakrishnan,A.N., Schirmer,M., Avila-Pacheco,J., Poon,T.W., Andrews,E., Ajami,N.J., Bonham,K.S., Brislawn,C.J., *et al.* (2019) Multi-omics of the gut microbial ecosystem in inflammatory bowel diseases. *Nature*, **569**, 655–662.
32. Plichta,D.R., Graham,D.B., Subramanian,S. and Xavier,R.J. (2019) Therapeutic opportunities in inflammatory bowel disease: mechanistic dissection of host-microbiome relationships. *Cell*, **178**, 1041–1056.
33. Ananthakrishnan,A.N., Luo,C., Yajnik,V., Khalili,H., Garber,J.J., Stevens,B.W., Cleland,T. and Xavier,R.J. (2017) Gut microbiome function predicts response to anti-integrin biologic therapy in inflammatory bowel diseases. *Cell Host Microbe*, **21**, 603–610.
34. Lenoir,M., Martín,R., Torres-Maravilla,E., Chadi,S., González-Dávila,P., Sokol,H., Langella,P., Chain,F. and Bermúdez-Humarán,L.G. (2020) Butyrate mediates anti-inflammatory effects of *Faecalibacterium prausnitzii* in intestinal epithelial cells through Dact3.*Gut Microbes*, **12**, 1826748.
35. Recharla,N., Geesala,R. and Shi,X.-Z. (2023) Gut microbial metabolite butyrate and its therapeutic role in inflammatory bowel disease: a literature review. *Nutrients*, **15**, 2275.
36. Tilg,H. and Danese,S. (2014) *Roseburia hominis*: a novel guilty player in ulcerative colitis pathogenesis? *Gut*, **63**, 1204–1205.
37. Steuer,R., Kurths,J., Daub,C.O., Weise,J. and Selbig,J. (2002) The mutual information: detecting and evaluating dependencies between variables. *Bioinformatics*, **18**, S231–S240.
38. Reshef,D.N., Reshef,Y.A., Finucane,H.K., Grossman,S.R., McVean,G., Turnbaugh,P.J., Lander,E.S., Mitzenmacher,M. and Sabeti,P.C. (2011) Detecting novel associations in large data sets. *Science*, **334**, 1518–1524.
39. Maurice,C.F., Haiser,H.J. and Turnbaugh,P.J. (2013) Xenobiotics shape the physiology and gene expression of the active human gut microbiome. *Cell*, **152**, 39–50.
40. Friedman,J. and Alm,E.J. (2012) Inferring correlation networks from genomic survey data. *PLoS Comput. Biol.*, **8**, e1002687.
41. McHardy,I.H., Goudarzi,M., Tong,M., Ruegger,P.M., Schwager,E., Weger,J.R., Graeber,T.G., Sonnenburg,J.L., Horvath,S., Huttenhower,C., *et al.* (2013) Integrative analysis of the microbiome and metabolome of the human intestinal mucosal surface reveals exquisite inter-relationships. *Microbiome*, **1**, 17.
42. Carr,A., Diener,C., Baliga,N.S. and Gibbons,S.M. (2019) Use and abuse of correlation analyses in microbial ecology. *ISME J.*, **13**, 2647–2655.
43. Zhou,J., Deng,Y., Luo,F., He,Z. and Yang,Y. (2011) Phylogenetic molecular ecological network of soil microbial communities in response to elevated CO2. *mBio*, **2**, e00122-11.
44. Deng,Y., Jiang,Y.-H., Yang,Y., He,Z., Luo,F. and Zhou,J. (2012) Molecular ecological network analyses. *BMC Bioinf.*, **13**, 113.

Received: June 27, 2024. **Revised:** November 19, 2024. **Editorial Decision:** November 25, 2024. **Accepted:** November 28, 2024