scientific reports

OPEN

A python library for the fast and scalable computation of biologically meaningful individual specific networks

Giada Lalli[1]✉, Zuqi Li[1,5], Federico Melograna[1,5], James Collier[2], Yves Moreau[3], Daniele Raimondi[3] & Kristel Van Steen[1,4]

Individual Specific Networks (ISNs) are a tool used in computational biology to infer Individual Specific relationships between biological entities from omics data. ISNs provide insights into how the interactions among these entities affect their respective functions. To address the scarcity of solutions for efficiently computing ISNs on large biological datasets, we present ISN-tractor, a data-agnostic, highly optimized Python library to build and analyse ISNs. ISN-tractor demonstrates superior scalability and efficiency in generating Individual Specific Networks (ISNs) when compared to existing methods such as LionessR, both in terms of time and memory usage, allowing ISNs to be used on large datasets. We show how ISN-tractor can be applied to real-life datasets, including The Cancer Genome Atlas (TCGA) and HapMap, showcasing its versatility. ISN-tractor can be used to build ISNs from various -omics data types, including transcriptomics, proteomics, and genotype arrays, and can detect distinct patterns of gene interactions within and across cancer types. We also show how Filtration Curves provided valuable insights into ISN characteristics, revealing topological distinctions among individuals with different clinical outcomes. Additionally, ISN-tractor can effectively cluster populations based on genetic relationships, as demonstrated with Principal Component Analysis on HapMap data.

Keywords Individual specific networks, Network analysis, Biological networks

Individual Specific Networks (ISNs) are a data analysis approach for -omics data that capture the unique relation patterns between biological entities (e.g., genes) within each individual in a population[1]. They provide a detailed understanding of heterogeneity and individuality among samples[2], offering advantages over traditional analysis approaches[3] and they can therefore provide new opportunities for stratified medicine. For example, ISNs can give a comprehensive view of within-samples relationships between biological entities such as genes[4], leading to a more complete understanding of complex biological systems and diseases compared to traditional methods[5].

ISNs have been successfully applied to the analysis of transcriptomics[6], metabolomics, and single-cell data[7]. Studies on COVID-19 have highlighted the importance of ISNs in understanding individual variability and guiding personalised treatment approaches[8]. Some ISN construction approaches use a perturbation-based method to estimate the impact of individual samples on network dynamics[2,9], while others employ a density-based method to evaluate the significance of interactions but are limited to binary (0–1) edges[10,11].

Regardless of their specific approach, all these methods face severe limitations in their applicability to large -omics datasets due to the computational burden of computing ISNs over densely connected biological networks. Existing ISN-building methods, such as LionessR[12], are indeed limited to compute ISNs containing less than 2000 genes due to computation time and memory limitation, thereby hindering the comprehensive *global* analyses of biological network perturbations of current biological datasets.

To address these limitations and extend the applicability of ISNs to the analysis of large multi-omics datasets, here we introduce ISN-tractor, a Pytorch-based[13] Python library that overcomes the scalability issues of conventional ISNs construction methods. With respect to the existing methods, ISN-tractor brings four major

[1]BIO3 - Systems Medicine Lab, Department of Human Genetics, KU Leuven, Leuven, Belgium. [2]VIB Technologies, VIB, Ghent, Belgium. [3]ESAT-STADIUS, KU Leuven, Leuven, Belgium. [4]BIO3 - Systems Genetics Lab, GIGA-R Medical Genomics, University of Liège, Liège, Belgium. [5]These authors contributed equally: Zuqi Li and Federico Melograna. ✉email: giada.lalli@kuleuven.be

novelties, both in terms of implementation and conceptual algorithmic improvements, to the conventional perturbation-based computation of ISNs.

First, ISN-tractor allows the parallelization of the ISN computation on CPUs and GPUs, providing a significant speed increase. Second, it is currently the only tool that can construct ISNs from any omics data source, including gene expression, proteomics, and genotyping arrays.

Third, ISN-tractor drastically reduces the required computational resources by allowing the user to choose to build sparse ISNs. These ISNs use an underlying graph representation that mirrors existing biological networks, such as the human interactome[14] or any user-defined gene-gene or protein-protein interaction network. By restricting the inference of ISN weights to knowledge-based small-world biological networks instead of conventional dense ISN graphs, the computational requirements are drastically reduced, further improving the speed and scalability of ISN-tractor.

The fourth novelty consists of a more efficient algorithm for the incremental computation of the Pearson correlation in the context of the ISNs. This new approach leverages the iterative nature of perturbation-based ISN computation[12] to minimize redundant operations. Consequently, the computational complexity is reduced from $\Theta(Np^2)$ to $\Theta(p^2)$, where N is the number of samples and p is the number of nodes in the ISN. This leads to a substantial speedup in ISN computation.

Methods

The perturbation-based computation of ISNs

A network $G := (V, E)$ consists of a set V of nodes v, and a set of edges $E \subseteq V \times V$, such that each edge e_{ij} connects a pair of nodes $(v_i, v_j) \in V$. ISN-tractor utilizes the perturbation-based approach to ISNs proposed by Ref.[2]. Each Individual Specific edge e^q_{ij}, for the q^{th} individual is computed as:

$$e^q_{ij} = N\, e^\alpha_{ij} - (N-1)\, e^{\alpha-q}_{ij} \tag{1}$$

where N is the sample size, e^α_{ij} is the edge computed on every individual in the dataset, and $e^{\alpha-q}_{ij}$ is the same edge computed without considering the q^{th} sample. The network calculated on every individual is called population network, while the one with an individual removed leave-one-out network. The edge values can be computed in several ways. One possibility is to compute a correlation score, like the Pearson correlation, between the values associated with each pair of nodes. Hence, the resulting network is weighted and undirected. This formula capitalizes on the contrast between two networks, differing solely in the presence or absence of a specific individual q, to infer insights regarding the impact on network topology when an individual is added or removed. Furthermore, the rationale behind this is the desire to construct ISNs such that their average would be close to the network obtained by pooling all samples together. In Ref.[2], the authors show that for large sample sizes and under specific linearity assumptions (see Suppl. 5.2 for an extensive discussion), the average of the ISNs across individuals is equal to the network computed on the population. Moreover, the final result is not influenced by the sample size. While larger sample sizes lead to a lower difference between e^α_{ij} and $e^{\alpha-q}_{ij}$, this is countered by the multiplicative factor N (sample size). Hence, the estimated ISN is driven by the deviation obtained by removing/adding individual q, and not by the sample size.

A novel algorithm for the Incremental ISNs computation drastically reduces the computational complexity

When ISNs are built using Pearson correlation, Eq. (1) involves several computations of Pearson between all the samples e^α_{ij} and all the sample with the exception on the q-th target sample, indicated as $e^{\alpha-q}_{ij}$. However, this means that a nearly identical operation is repeated N times. For example, the algorithm used in the LionessR ISNs package[2,12], suffers from this redundancy in the computation, since it removes the target sample q at each iteration (i.e., for each individual). This is very inefficient since most of the operations repeated for each sample could be avoided instead. To overcome this issue, we propose an incremental calculation of the Pearson correlation that dramatically speeds up the calculation by avoiding all the redundant operations. In our approach, we store summary statistics of our population and, through addition and subtraction, we calculate the Pearson correlation of the population minus the target sample, without re-running the computation on the entire dataset. Furthermore, we vectorize the procedure to make it more efficient and better suited for GPU computations.

Explaining the intuition behind the incremental Pearson correlation algorithm

In this section, we describe the reasoning and the mathematical steps behind the Incremental Pearson correlation algorithm, focused on one single edge. In the Supplementary (Section 1.2.1) we expand it to a vectorized version where an entire network is calculated in one single step. Finally, we discuss the advantages of the proposed incremental Pearson.

Given a $N \times p$ matrix X, where N is the number of samples and p is the number of features (i.e. genes), the Pearson correlation between columns (genes) j and k is computed as:

$$r^\alpha_{jk} = \frac{\Sigma^n_{q=1}(x_{qj} - \bar{x}_j)(x_{qk} - \bar{x}_k)}{\sqrt{\Sigma^n_{q=1}(x_{qj} - \bar{x}_j)^2}\sqrt{\Sigma^n_{q=1}(x_{qk} - \bar{x}_k)^2}} \tag{2}$$

Here, we follow the notation used in Ref.[2], indicating with the apex α the Pearson correlation calculated on the whole population (all the N individuals). Moreover,

- x_{qj} and x_{qk} are the individual gene values for genes j and k in individual q,
- $\bar{x}_j = \frac{1}{n}\Sigma_{q=1}^{n} x_{qj}$, is the arithmetic mean of gene j across all the individuals and, specularly, $\bar{x}_k$ is the mean across all the individuals for gene k.

For big datasets, i.e., high sample size and feature set, recalculating Pearson correlation for each leave-one-out step of the LIONESS procedure can be very time-consuming. To reduce the computation time, we extract *sufficient statistics* from the population network, ensuring that to calculate the leave-one-out edges we only need to add or subtract quantities calculated from them. Sufficient statistics are quantities that summarize all the needed information from the samples to build the Pearson correlation. For example, in this context sufficient statistics calculate the sum of the contribution of each sample, such as the gene abundance, and thus, the contribution of each sample can be easily singled out.

Hence, we rearranged the Eq. (2) to highlight these sufficient statistics. The resulting formula is as follows:

$$r_{jk}^{\alpha} = \frac{NA_{jk} - \hat{x}_j\hat{x}_k}{\sqrt{N\hat{a}_k - (\hat{x}_k)^2}\sqrt{N\hat{a}_j - (\hat{x}_j)^2}} \tag{3}$$

The sufficient statistics are listed below. The size of the corresponding vectors and matrices are indicated as pedix:

- $\hat{x}_{p\times 1}$, with element $\hat{x}_j = \sum_{i=1}^{n} x_{ij}$, the sum of j-th gene in all individuals.
- $A_{p\times p}$, with element $A_{jk} = \sum_{i=1}^{n} x_{ij}x_{ik}$ the sum over all individuals of two genes j and k (dot product).
- $\hat{a}_{p\times 1}$, with element $\hat{a}_j = \sum_{i=1}^{n} x_{ij}^2$, the sum of the square j-th gene in all individuals. (The diagonal of the dot product).

Since our goal is to calculate the Pearson correlation for a population composed of all-but-one sample (leave-one-out, indicated with $\alpha - q$ as apex), we subtract the contribution given by observation q from each sufficient statistic. In detail, for a generic q-th individual:

- x_{qj}; the value of gene j in individual q subtracted from $\hat{x}_j = \sum_{i=1}^{n} x_{ij}$ in both numerator and denominator; (similarly x_{qk} subtracted from $\hat{x}_k$).
- $x_{qj}x_{qk}$; the product of genes j and k in individual q subtracted from $A_{jk} = \sum_{i=1}^{n} x_{ij}x_{ik}$.
- x_{qj}^2; the squared value of gene j in individual q subtracted from $\hat{a}_j = \sum_{i=1}^{n} x_{ij}^2$; (similarly x_{qk}^2 subtracted from $\hat{a}_k$).
- 1 from N, since the sample size is reduced by 1.

Such differences constitute the contribution of individual q to the population Pearson correlation; by eliminating them, the result is the leave-one-out network, based on the population without the individual q

Hence, the formula to calculate the leave-one-out edge (correlation) between genes j and k, without individual q, with in bold the contribution subtracted, is:

$$r_{jk}^{\alpha-q} = \frac{(N\mathbf{-1})(A_{jk}\mathbf{-x_{qj}x_{qk}}) - (\hat{x}_j\mathbf{-x_{qj}})(\hat{x}_k\mathbf{-x_{qk}})}{\sqrt{(N\mathbf{-1})(\hat{a}_k\mathbf{-x_{qk}^2}) - (\hat{x}_k\mathbf{-x_{qk}})^2}\sqrt{(N\mathbf{-1})(\hat{a}_j - \mathbf{x_{qj}^2}) - (\hat{x}_j - \mathbf{x_{qj}})^2}} \tag{4}$$

With the formula above, we calculate the Pearson correlation for the leave-one-out network directly from the sufficient statistic of the entire population. Hence, without the need to re-compute the Pearson correlation, but exploiting the knowledge of the population network.

Thus, having both r_{jk}^{α} and $r_{jk}^{\alpha-q}$, as calculated in Eqs. (3), and (4), the LIONESS formula (Eq. (1)), for an ISN edge between genes j and k of individual q, is as follows:

$$r_{jk}^{q} = Nr_{jk}^{\alpha} - (N-1)r_{jk}^{\alpha-q} \tag{5}$$

A summarization of the various steps to achieve the incremental Pearson correlation algorithm is provided in Fig. S4.

Speeding up further the ISN computation by exploiting the sparsity of biological networks

Existing methods for constructing ISNs typically assume an underlying dense network. This poses challenges when the goal is to construct genome- or proteome-wide ISNs, since the number of edges scales quadratically with the number of genes/proteins considered. In contrast, the library we propose, ISN-tractor, introduces a novel way to reduce this computational burden in a biologically meaningful manner. If the user needs to build particularly large ISNs, they can choose to reduce the computation burden by defining a sparse network as *scaffold* underlying the computed ISNs. This underlying sparse network can be selected from established biological networks, from databases such as STRING[15], KEGG[16], consensuspathDB[17], or any other source. In this way, ISN-tractor can build ISNs considering only user-defined edges, thereby leveraging the inherent sparsity of biological networks[18] to reduce the size and the computational cost of the ISNs. The integration of prior biological knowledge reduces the number of edges to be inferred in a biologically meaningful way, ideally reducing the network size while preserving the most significant interactions that are likely to contribute meaningful insights. For instance, in the context of proteomics or gene expression data analysis, ISN-tractor can build ISNs based on known gene or protein interactions extracted from the Human Reference Interactome (HuRI)[14], which is one of the most comprehensive protein-protein interaction network available to date. This allows researchers to construct proteome-scale ISNs in a scalable way, tailoring the network underlying the ISN to their specific analysis objectives, dataset characteristics, or organism of interest. While drastically improving the scalability of the computation, this biologically meaningful sparsification enables researchers to integrate prior knowledge into the network construction.

ISN-tractor can compute ISNs from genotype array data

ISN-tractor is designed to compute ISNs from a variety of -omics data, including but not limited to genotype array data, transcriptomics, and proteomics. This versatility allows ISN-tractor to be applied across different -omics datasets, ensuring its utility in a wide range of biological and clinical research applications. For instance, when applied to genotype array data, the observed Single Nucleotide Polymorphisms (SNPs) are first mapped to their respective user-defined genes or genomic regions.

Each of these genomic regions is associated with a network node. The edge-weights between nodes are inferred by evaluating the correlation between the sets of SNPs mapped on each node V_i. In case of large SNP-array data encompassing many genes, the number of edges to be computed can be reduced by sparsifying the expected network of connections between nodes, as explained in Section "Speeding up further the ISN computation by exploitingthe sparsity of biological networks". For instance, if each node represents a gene, the user may opt to instruct ISN-tractor to permit edges exclusively between genes known to interact, as evidenced by biological networks such as HuRI[14].

To determine the weight assigned to each edge, ISN-tractor computes the correlations among the sets of SNPs mapped on each pair of nodes V_i and V_j that are connected by an edge, obtaining two $N \times m_i$ and $N \times m_j$ matrices respectively, where N is the number of samples and m_* indicates the number of SNPs mapped on each node V_*. In the case of SNP-array data, ISN-tractor computes the correlation between every SNP mapped on V_i with every SNP mapped on V_j, thus obtaining a $m_i \times m_j$ matrix C_{ij}. To compute correlations, ISN-tractor provides 3 commonly used similarity metrics (i.e., Pearson, Spearman, or Dot Product) and allows any other user-defined metric (see Sect. "Clustering HapMap data with ISN-tractor" for an example).

Finally, in order to obtain a scalar value that can be used as edge weight between the V_i and V_j nodes in the ISN, the user can choose between several pooling strategies to *summarize* the matrix C_{ij}. The simplest ones are to take the maximum or the average value of the elements in C_{ij}. The computation of this additional matrix is necessary to accomodate the SNPs mapped on genomic regions to the ISN nodes in a flexible way. From this point on, the ISN can be computed conventionally with Eq. (1), similarly to an ISN derived from expression or proteomic data.

Results

ISN-tractor is currently the most scalable and efficient way to generate ISNs

We conducted a comprehensive benchmarking analysis to evaluate the time and memory efficiency of ISN-tractor in constructing ISNs on synthetic data, gradually increasing the number of samples and features. Figure 1A provides a visual comparison with LionessR[12], showing the superior performance of ISN-tractor in both time and memory usage across various sample sizes. A notable advantage is that ISN-tractor is not confined by the 2000-node limitation of LionessR, enabling the analysis of modern proteome and genome-wide datasets.

In terms of computation time (Fig. 1A), for a dataset with 200 samples and 1000 features, LionessR takes 43 s to compute ISNs, while ISN-tractor achieves this task 36 times faster on CPU (1.2 s) and 478 times faster on GPU (0.09 s). Similarly, with a dataset of 500 samples and 2000 features, LionessR consumes 1491 s, while ISN-tractor completes the computation 135 times faster on CPU (11 s) and 12425 times faster on GPU (0.12 s).

While LionessR is restricted to CPU-based computations for ISNs, ISN-tractor can harness the power of GPU acceleration, allowing ISN-tractor to efficiently handle large datasets. For example, given a dataset with 2000 samples by 10000 features (i.e. genes), ISN-tractor takes 1107 seconds to build an ISN on CPU, while only 6.8 seconds are required on GPU. In contrast, LionessR's maximum processing capacity is constrained to either 200 samples by 3000 features (completed in 948 seconds) or 2000 samples by 1000 features (completed in 3275 seconds). Considering memory usage, ISN-tractor's efficiency is striking.

From the perspective of RAM memory usage, unlike LionessR, which loads the entire dataset each time, ISN-tractor employs precomputed summary statistics and mini-batching for the ISN computation. For a dataset of 200 samples and 1000 features, LionessR utilizes over 2 GiB of memory, whereas ISN-tractor averages a mere 0.33 GiB (see right panel in Fig. 1A). Extending this comparison to a larger dataset of 500 samples by 1000 features, LionessR's memory usage hits almost 14 GiB, while ISN-tractor remains consistently below 0.37 GiB. These memory optimizations further highlight ISN-tractor's practical advantage in handling large-scale data analyses efficiently.

Analysis of TCGA data with ISN-tractor

To illustrate the analysis that ISN-tractor can enable on real-life datasets such as The Cancer Genome Atlas (TCGA)[19], we selected 3024 patients from 6 cancer types, including 731 breast invasive carcinoma (BRCA), 408 thyroid carcinoma (THCA), 387 urothelial bladder carcinoma (BLCA), 297 lung squamous cell carcinoma (LUSC), 412 head and neck squamous cell carcinoma (HNSC), and 312 kidney renal clear cell carcinoma (KIRC). For every patient we downloaded their transcriptomic profiles of 58456 hg38 genes with TCGAbiolinks[20] and

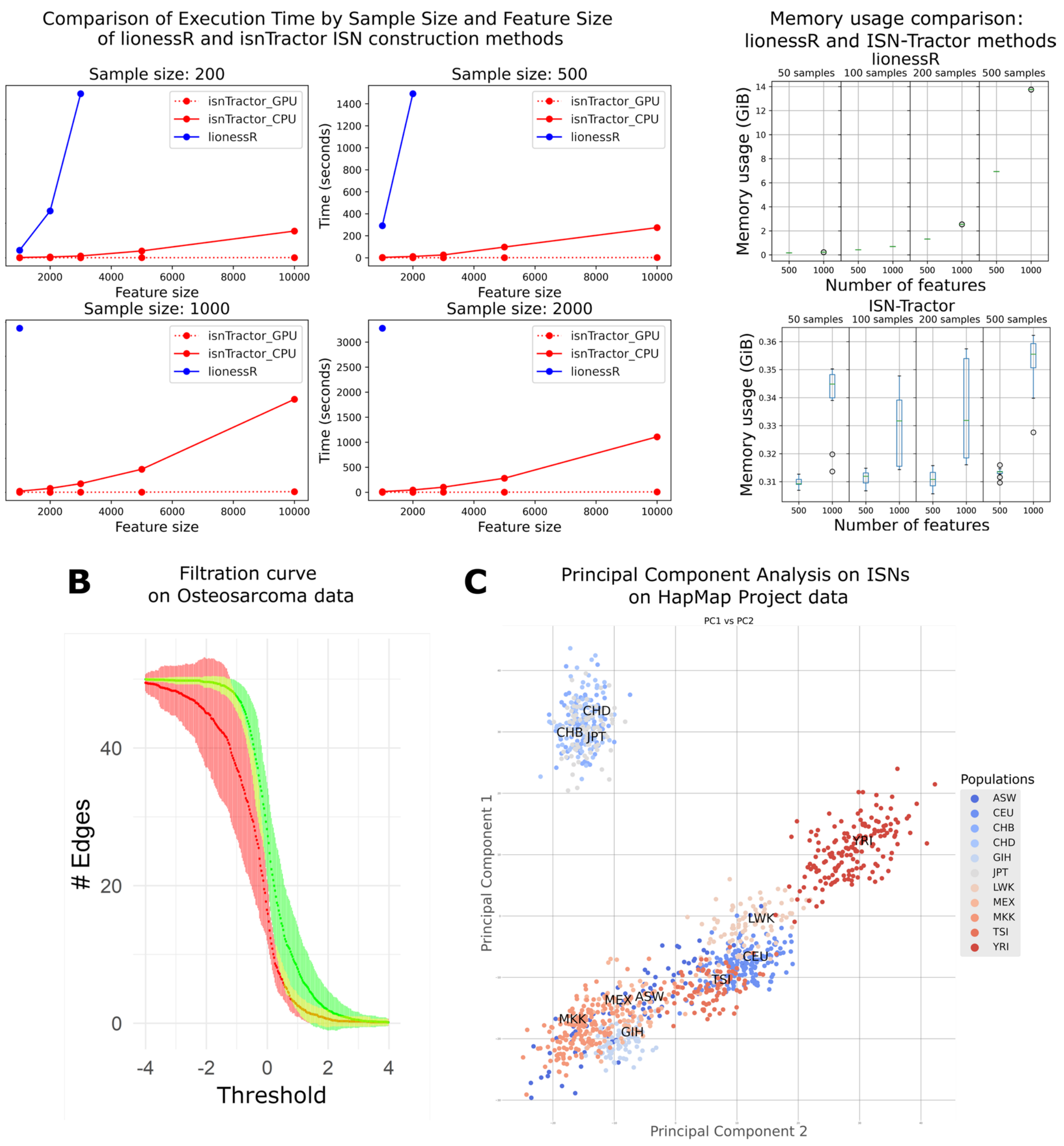

Figure 1. (**A**) shows the time and memory usage comparison between ISN-tractor and LionessR. ISN-tractor outperforms LionessR in both benchmarks. We computed dense ISNs using the Incremental Pearson algorithm in ISN-tractor for this comparison. This approach was chosen because LionessR can only compute dense ISNs with the Pearson correlation. (**B**) shows the Filtration curves computed by ISN-tractor on Osteosarcoma patients with good/poor MFS, respectively in green and red. The overlap is highlighted in yellow. (**C**) shows the PCA obtained from SNP-array based ISNs for individuals coming from different populations from the HapMap dataset.

we considered the subset of the 2000 most variably expressed genes (including 1931 protein-coding genes, 10 pseudogenes, 21 non-coding genes, and 38 immunoglobulin genes).

To construct cancer-specific ISNs we first trained a Random Forest model to select the 300 most relevant features (genes) for cancer classification in the 6 cancer types. We then built ISNs based on these most relevant 300 genes utilizing ISN-tractor and the interactions (edges) between genes were measured by Pearson correlation. To ensure clear visualization, we focused on the top 500 most variable gene interactions among the ISNs. The resulting plots effectively depicted representative and distinctive patterns of edge weights for each cancer type (Fig. S4 from Suppl. Mat., section 3.3). For instance, the most connected gene in the mean ISN of cancer type BRCA is *SCGB1D2*, whose association with breast cancer has been previously discovered[21]. Another insightful finding comes from the gene *S100P* as one of the most connected genes in the ISN of THCA. The *S100* proteins are known for regulating a variety of intracellular and extracellular functions relating to many cancer types including thyroid[22]. As one of the most recently reported gene in the *S100* family, the study on *S100P* is still an ongoing process. Our results may suggest a potential link between *S100P* and thyroid carcinoma. The highly connected gene *IGFBP6* in the ISN of HNSC has previously discovered role as an oncosuppressor for nasopharyngeal carcinoma, which is a type of head and neck squamous cell carcinoma[23].

Visualizing ISNs analysis with ISN-tractor filtration curves

Graph Filtration Curves (FC)[24] are a powerful analysis tool to understand the topology of graphs that we incorporated in ISN-tractor. FC define threshold values that are used to *filter* the edges in the graph, analysing how topology changes in function of varying thresholds by comparing measures such as the graph connectivity. Visualizing the change of these measures provides insights into the graphs' characteristics and hierarchical structure. ISN-tractor allows researchers to group multiple ISNs based on phenotypes of interest and then directly compare them for different filtration values.

To showcase the potential of FCs for the analysis of ISNs, we applied this concept to a dataset comprising gene expression data from the 53 Osteosarcoma patients analysed in Ref.[12]. The information was sourced from the Gene Expression Omnibus (GEO) under accession number GSE42352. The samples included in the dataset had a minimum of 70% tumor content and viability, and their RNA profiles were generated using Illumina human-6 v2.0 microarray beadchips. The patients were divided into two groups: one with 19 samples who did not experience metastasis within 5 years of primary tumour diagnosis (better Metastasis-Free Survival, also called MFS), and another with 34 samples who did develop metastasis within 5 years (poor MFS).

Using the ISN-tractor library, we built ISNs for all the samples. Following the approach of[12], we focus on the top 50 significant edges, identified through LIMMA analysis[25]. LIMMA analysis pinpoints edges with the most noteworthy statistical difference between the two groups.

Subsequently, we used ISN-tractor to construct FCs, depicted in Fig. 1B, to visually discern between the ISNs of individuals who experienced metastasis and those who did not. The number of edges was selected as the topological property for filtration. For each filtration value, FCs were averaged for different MFS outcomes, and the confidence interval was depicted using the standard deviation. This process generated a distinct FC for each MFS outcome. While there was some overlap between the FCs of poor and better MFS for low and high threshold values, the FC highlighted topological distinctions among individuals with different outcomes, particularly for moderate filtration values. Additionally, it provided insights into the optimal threshold for network binarization.

Clustering HapMap data with ISN-tractor

As a fourth and last use case example, here we illustrate the application of ISN-tractor in constructing ISNs from SNP array data. Specifically, we utilized data sourced from the HapMap Project[26], a publicly available dataset renowned for its comprehensive SNP coverage. The HapMap dataset comprises 498,314 SNPs obtained from 1,301 samples, stratified across 11 distinct populations. Initially, SNP positional mapping was conducted across the entirety of annotated genes using the Human Genome Assembly **GRCh38.105** as a reference. ISN-tractor effectively assigned SNPs to 6,751 genes through this mapping process. To streamline the ISN computation and mitigate computational complexity, the network was sparsified, restricting edge formation solely to interactions documented within the Human Reference Interactome (HuRI)[14]. Consequently, sparse ISNs were constructed, enhancing computational efficiency while retaining biologically relevant interactions. To underscore the versatility of the library, we used a specific similarity metric to compute the edge weights, instead of the conventional correlation measures natively supported by ISN-tractor. We thus implemented the Linkage Disequilibrium metric[27] via the scikit-allel Python library[28], showing that the user can extend ISN-tractor with their libraries of choice if needed. We then proceeded with the analysis, visually exploring the genetic relationships among diverse populations with a Principal Component Analysis (PCA) of the constructed ISNs, demonstrating the flexibility of ISN-tractor in accommodating different similarity metrics and external libraries for edge weight computation. The results, depicted in Fig. 1C, underscored the discernible clustering patterns among populations, thereby elucidating the underlying genetic structure.

Conclusions

In this paper, we present ISN-tractor, a novel python library for the analysis of -omics data with Individual Specific Networks (ISNs). ISN-tractor provides both implementation and conceptual enhancements that make it more efficient, both in time and memory usage, when compared to existing ISN methods. In addition, it is also able to build ISNs from various -omics data, such as expression, proteomics, and genotype arrays.

When applying ISN-tractor to genotype arrays, users should account for differences from proteomics or expression arrays, which are more conventionally used in ISNs construction. Proteomics and expression data are continuous and dynamic, capturing protein abundance or gene expression levels across conditions or time points, whereas genotype data is discrete, representing static genetic variations. For proteomics and expression arrays, ISN-tractor computes correlations between protein or gene expression levels to infer interaction networks, reflecting functional relationships and regulatory mechanisms. This allows the detection of subtle network topology changes associated with different phenotypes or disease states, enhancing understanding of biological processes.

For genotype arrays, ISN-tractor maps Single Nucleotide Polymorphisms (SNPs) to user-defined genes or genomic regions, constructing networks based on SNP correlations. These networks highlight genetic relationships and potential epistatic interactions, providing insights into the genetic architecture of complex traits and diseases. The sparsity and static nature of genotype data require additional data processing steps, such as sparsifying the network based on known biological interactions to reduce computational complexity while preserving relevant edges.

While ISN-tractor's approach is versatile across different -omics data types, users should consider each data type's unique characteristics and biological implications to optimize their analyses and derive meaningful insights.

ISN-tractor has the potential to expand the applicability of ISN-based data analysis on the larger datasets that are continuously nowadays, and its open source nature ensures that it can be further expanded following the needs of the community, including adding additional similarity measures (i.e., biweight midcorrelation, commonly used in gene co-expression analysis[29]), also allowing its integration in new precision medicine technological paradigms, such as Federated Learning[30].

In particular, the incremental calculation of Pearson correlation that we propose in this paper is promising for Federated Learning applications, since it shows how individual data can be added to a population without sharing the data itself but only the sufficient statistics calculated on them. This adaptation enable secure and privacy-preserving collaborative research efforts based on ISN data analysis.

In essence, ISN-tractor, not only overcomes the limitations of existing ISN-building methods but also presents a robust platform for advancing the exploration of Individual Specific molecular landscapes. The demonstrated capabilities of ISN-tractor hold the promise of deepening our understanding of complex biological systems on a genome- and proteome-wide scale.

Data availability

HapMap data: https://ftp.ncbi.nlm.nih.gov/hapmap/genotypes/; TCGA data: https://portal.gdc.cancer.gov/repository/; Human Genome Assembly: https://www.ensembl.org/Homo_sapiens/Info/Index; HuRI: http://www.interactome-atlas.org/; Osteosarcoma data: Gene Expression Omnibus (GEO) under accession number GSE42352 https://www.ncbi.nlm.nih.gov/geo/query/acc.cgi?acc=GSE42352.

Code availability

ISN-tractor is available from our git repository https://github.com/GiadaLalli/ISN-tractor and as Python package https://pypi.org/project/ISN-tractor/.

Received: 26 February 2024; Accepted: 31 July 2024
Published online: 06 August 2024

References

1. Lee, L.Y.-H. & Loscalzo, J. Network medicine in pathobiology. *Am. J. Pathol.* **189**(7), 1311–1326 (2019).
2. Kuijjer, M. L., Tung, M. G., Yuan, G., Quackenbush, J. & Glass, K. Estimating sample-specific regulatory networks. *Iscience* **14**, 226–240 (2019).
3. Yu, X. *et al.* Identification of long noncoding RNA biomarkers for hepatocellular carcinoma using single-sample networks. *BioMed Res. Int.* **2020**, 8579651 (2020).
4. Gosak, M. *et al.* Network science of biological systems at different scales: A review. *Phys. Life Rev.* **24**, 118–135 (2018).
5. Yan, J., Risacher, S. L., Shen, L. & Saykin, A. J. Network approaches to systems biology analysis of complex disease: integrative methods for multi-omics data. *Brief. Bioinform.* **19**(6), 1370–1381 (2018).
6. Zhu, K., Pian, C., Xiang, Q., Liu, X. & Chen, Y. Personalized analysis of breast cancer using sample-specific networks. *PeerJ* **8**, 9161 (2020).
7. Huang, Y., Chang, X., Zhang, Y., Chen, L. & Liu, X. Disease characterization using a partial correlation-based sample-specific network. *Brief. Bioinform.* **22**(3), 062 (2021).
8. Tanaka, Y. *et al.* Dynamic changes in gene-to-gene regulatory networks in response to sars-cov-2 infection. *Sci. Rep.* **11**(1), 1–13 (2021).
9. Liu, X., Wang, Y., Ji, H., Aihara, K. & Chen, L. Personalized characterization of diseases using sample-specific networks. *Nucleic Acids Res.* **44**(22), 164–164 (2016).
10. Li, L., Dai, H., Fang, Z. & Chen, L. c-csn: Single-cell rna sequencing data analysis by conditional cell-specific network. *Genom. Proteom. Bioinform.* **19**(2), 319–329 (2021).
11. Dai, H., Li, L., Zeng, T. & Chen, L. Cell-specific network constructed by single-cell RNA sequencing data. *Nucleic Acids Res.* **47**(11), 62–62 (2019).
12. Kuijjer, M. L., Hsieh, P.-H., Quackenbush, J. & Glass, K. lionessr: Single sample network inference in r. *BMC Cancer* **19**, 1–6 (2019).
13. Imambi, S., Prakash, K.B., & Kanagachidambaresan, G. Pytorch. Programming with TensorFlow: Solution for Edge Computing Applications, 87–104 (2021).
14. Luck, K. *et al.* A reference map of the human binary protein interactome. *Nature* **580**(7803), 402–408 (2020).
15. Mering, C. V. *et al.* String: A database of predicted functional associations between proteins. *Nucleic Acids Res.* **31**(1), 258–261 (2003).
16. Kanehisa, M. The kegg database. In *'In Silico' Simulation of Biological Processes: Novartis Foundation Symposium 247* Vol. 247 (ed. Kanehisa, M.) 91–103 (Wiley, 2002).
17. Kamburov, A., Wierling, C., Lehrach, H. & Herwig, R. Consensuspathdb—A database for integrating human functional interaction networks. *Nucleic Acids Res.* **37**(1), 623–628 (2009).
18. Verplaetse, N., Passemiers, A., Arany, A., Moreau, Y. & Raimondi, D. Large sample size and nonlinear sparse models outline epistatic effects in inflammatory bowel disease. *Genome Biol.* **24**(1), 224 (2023).
19. Weinstein, J. N. *et al.* The cancer genome atlas pan-cancer analysis project. *Nat. Genet.* **45**(10), 1113–1120 (2013).
20. Colaprico, A. *et al.* Tcgabiolinks: An r/bioconductor package for integrative analysis of tcga data. *Nucleic Acids Res.*https://doi.org/10.1093/nar/gkv1507 *(2015).*
21. Sjödin, A., Ljusliner, I., Henriksson, R. & Hedman, H. Mammaglobin and lipophilin B expression in breast tumors and their lack of effect on breast cancer cell proliferation. *Anticancer Res.* **28**(3A), 1493–1498 (2008).
22. Jiang, H. *et al.* Calcium-binding protein S100P and cancer: Mechanisms and clinical relevance. *J. Cancer Res. Clin. Oncol.* **138**(1), 1–9. https://doi.org/10.1007/s00432-011-1062-5 (2012).
23. Kuo, Y.-S., Tang, Y.-B., Lu, T.-Y., Wu, H.-C. & Lin, C.-T. IGFBP-6 plays a role as an oncosuppressor gene in NPC pathogenesis through regulating EGR-1 expression. *J. Pathol.* **222**(3), 299–309. https://doi.org/10.1002/path.2735 (2010).
24. O'bray, L., Rieck, B., & Borgwardt, K. Filtration Curves for Graph Representation; Filtration Curves for Graph Representation. https://doi.org/10.1145/3447548.3467442 (2021).
25. Smyth, G. K. Linear models and empirical bayes methods for assessing differential expression in microarray experiments. *Stat. Appl. Genet. Mol. Biol.*https://doi.org/10.2202/1544-6115.1027 *(2004).*
26. Gibbs, R.A., Belmont, J.W., Hardenbol, P., Willis, T.D., Yu, F., Yang, H., Ch'ang, L.-Y., Huang, W., Liu, B., Shen, Y., et al. The International Hapmap Project (2003).

27. Rogers, A. R. & Huff, C. Linkage disequilibrium between loci with unknown phase. *Genetics* **182**(3), 839–844 (2009).
28. Miles, A., & Harding, N. scikit-allel: A Python package for exploring and analysing genetic variation data (2016).
29. Langfelder, P. & Horvath, S. WGCNA: An r package for weighted correlation network analysis. *BMC Bioinform.* **9**, 559. https://doi.org/10.1186/1471-2105-9-559 (2008).
30. Raimondi, D. *et al.* Genome interpretation in a federated learning context allows the multi-center exome-based risk prediction of Crohn's disease patients. *Sci. Rep.* **13**(1), 19449 (2023).

Author contributions
GL: Conceptualization, Methodology, Software, Validation, Visualisation, Writing: Original Draft, Review & Editing; ZL: Methodology, Software, Visualisation, Writing: Original Draft, Review & Editing; FM: Methodology, Software, Formal Analysis, Visualisation, Writing: Original Draft, Review & Editing; JC: Software, Validation, Writing: Original Draft, Review & Editing; YM: Supervision; DR: Conceptualization, Writing: Original Draft, Review & Editing, Supervision, Project Administration; KVS: Resources, Funding Acquisition.

Funding
GL, ZL, FM, KVS have received funding from the European Union's Horizon 2020 research and innovation programme under the H2020 Marie Skłodowska-Curie grant agreement No 860895 (TranSYS); DR is funded by a FWO senior post-doctoral fellowship, grant number 12Y5623N.

Competing interests
The authors declare no competing interests.

Additional information
Supplementary Information The online version contains supplementary material available at https://doi.org/10.1038/s41598-024-69067-2.

Correspondence and requests for materials should be addressed to G.L.