Seidr: A toolkit for calculation of crowd networks

Bastian Schiffthaler[1], Alonso Serrano[1], Nicolas Delhomme[2], and Nathaniel R. Street[1]

[1]Umeå Plant Science Centre (UPSC), Department of Plant Physiology, Umeå University, Sweden
[2]Umeå Plant Science Centre (UPSC), Department of Plant Physiology, Swedish University of Agricultural Sciences, Sweden

Abstract

Gene network analysis offers a powerful tool to extract key players in complex RNA sequencing experiments which cannot be adequately examined via binary differential expression analysis. A number of gene network inference algorithms have been published, but any algorithm based on a single statistic tends to suffer from bias towards a particular type of interaction. The DREAM 5 challenge offered a potential solution via aggregation of several methods into a combined crowd network. Here we present *Seidr*, a unified tool to create gene networks and combine them into a unified meta-network. *Seidr* is implemented with a focus on speed and makes heavy use of parallelism where applicable.

Introduction

The continued decline in the cost of RNA sequencing experiments has led to an explosive increase in high throughput datasets in the last years. Many of these experiments are complex in nature and difficult to analyze in traditional binary differential expression comparisons (control vs. treatment).

Gene networks are a powerful tool to investigate complex transcriptomics data by creating a network of gene interactions and selecting key players via centrality metrics. These networks are often calculated by relying on a single scoring statistic, such as correlation, mutual information or regression statistics. The DREAM5 challenge benchmarked a large number of available inference algorithms and revealed that each method was biased towards certain interaction types (Marbach et al., 2012). Further, it was reported that aggregating all networks into a meta-network drastically improved the accuracy of the solution.

While meta networks perform well in accuracy, they are extremely diffcult to calculate for most researchers. The algorithms are written in varying programming languages (C, C++, R, MATLAB), which may require custom compilation by the user. Further, they require different input formats and/or are error prone. Another issue is the computational time required to calculate networks, which is further exacerbated by the use of inefficient program designs and by the choice of programming language.

In this article, we present *Seidr*, a comprehensive toolkit to calculate gene networks, aggregate them into a meta network and perform downstream analysis. *Seidr* is created with a heavy emphasis on execution speed and parallelism to allow for the calculation of all-vs-all calculations, while also offering a consistent front-end and user friendly command line interface. *Seidr* is available upon request to the authors.

Implementation

Seidr is implemented as a set of C++ executables which are intended to be executed in three core steps - network inference, ranking and aggregation (see Fig. 1). Afterwards, *Seidr* assists with threshold selection and calculation of various centrality metrics to prepare the data for input in graph partitioning or clustering

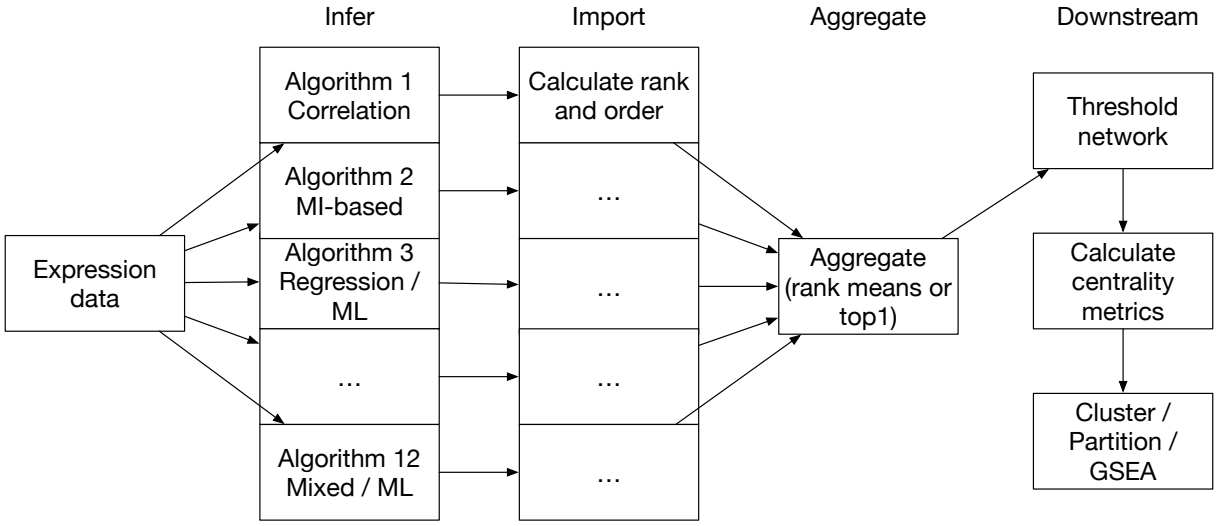


Figure 1: The typical workflow of *Seidr*. Several networks are calculated from expression data based on several published algorithms. These are then ranked and ordered in lower triangular order. All networks are aggregated to a consensus network using one of two strategies (mean or top). Finally, centrality statistics can be calculated and downstream analysis performed.

software or to carry out gene set enrichment analyses.

Base network inference

As a first step to the inference pipeline, base networks must be inferred using expression data from the set of experiments of interest. *Seidr* includes native C++ implementations of 12 published network inference algorithms based on fast open source libraries (see Tab. 1). These include the top performers of the DREAM5 study as well as some algorithms subsequently. Algorithms that pose a computational bottleneck have been implemented using OpenMPI at their core to allow parallel calculations at a scale of thousands of CPU cores spread across a network of hosts. *Seidr* includes a wrapper program which handles the majority of the work related to running the inference jobs on local machines or via job managers such as SLURM.

Ranking and ordering of network edges

In order to aggregate the edges of each individual network in the following step, the per network edge weights must first be made comparable. This is achieved by transforming edge weights into ranks. Edges with tied weights are assigned the mid-rank between the first and last tied weight. In case the input is a non-symmetrical matrix, only the better weight is chosen as the representative. Finally, the edges are ordered as the lower triangular of the symmetric node-node matrix.

Network aggregation

For each edge in the network a representative final edge is created by mean rank (borda) aggregation or by picking the best rank of all available methods (top1).

Network thresholding

The output of the aggregation will be a fully dense network, meaning that every node is connected to every other node with a certain edge weight. In order to proceed with a meaningful network, it is important to select an appropriate cutoff below which edges assumed to be noise or to lack biological meaning are removed. *Seidr* aids cutoff selection by calculating two network topology statistics - scale free fit and transitivity - at a user defined interval.

Calculation of centrality scores

Finally, *Seidr* makes use of the parallel NetworKit library (Staudt, Sazonovs, and Meyerhenke, 2014) to calculate centrality statistics

Table 1: List of algorithms implemented in *Seidr*. All algorithms were natively implemented in *Seidr*. Algorithms that pose a computational bottleneck were implemented in a parallel fashion.

Name	Published	Type	Orig. Lang.	Seidr Lang.	Orig. Parallel	Seidr Parallel
ANOVERENCE	Küffner et al., 2012	ANOVA	C++	C++	No	No
ARACNE	Margolin et al., 2006	MI + DPI	C++	C++	Yes	Yes
CLR	Faith et al., 2007; Daub et al., 2004	MI + CLR	MATLAB / C / C++	C++	No	Yes
Elastic Net ensemble	Ruyssinck et al., 2014	Elastic Net Regression	R (glmnet)	C++ (glmnet)	No	Yes
GENIE3	Huynh-Thu et al., 2010	Random Forest Regression	R (randomForest)	C++ (ranger)	No	Yes
NARROMI	Zhang et al., 2013	MI + Linear Programming	MATLAB	C++ (glpk)	No	Yes
Partial Correlation	Schäfer and Strimmer, 2005	Correlation	R	C++	No	No
Pearson Correlation	NA	Correlation	NA	C++	No	No
PLSNET	Guo et al., 2016	PLS	MATLAB	C++	No	Yes
Spearman Correlation	NA	Correlation	NA	C++	No	No
SVM ensemble	Ruyssinck et al., 2014	SVM regression	R (libsvm) / C	C++ (libsvm, liblinear)	No	Yes
TIGRESS	Haury et al., 2012	LASSO Regression	MATLAB / R	C++ (glmnet)	No	Yes

for further interpretation of the network. Currently, PageRank (Page et al., 1998), Closeness (Bavelas, 1950; Eppstein and Wang, 2000), Betweenness (Brandes, 2001; Geisberger, Sanders, and Schultes, 2008; Riondato and Kornaropoulos, 2016; Freeman, 1977), Strength, Eigenvector centrality and Katz (Katz, 1953) centrality are calculated for each node. Spanning Edge centrality (Teixeira et al., 2013) as well as Edge Betweenness centrality (Girvan and Newman, 2002) are calculated for each edge.

References

Bavelas, Alex (1950). "Communication Patterns in Task-Oriented Groups". In: *The Journal of the Acoustical Society of America* 22.6, pp. 725–730.

Brandes, Ulrik (2001). "A faster algorithm for betweenness centrality*". In: *The Journal of Mathematical Sociology* 25.2, pp. 163–177. arXiv: `arXiv:1506.05783v1`.

Daub, Carsten O et al. (2004). "Estimating mutual information using B-spline functions–an improved similarity measure for analysing gene expression data." In: *BMC bioinformatics* 5, p. 118.

Eppstein, David and Joseph Wang (2000). "Fast Approximation of Centrality". In: *Journal of Graph Algorithms and Applications* 8.1, p. 2. arXiv: `0009005 [cs.DS]`.

Faith, Jeremiah J et al. (2007). "Large-scale mapping and validation of Escherichia coli transcriptional regulation from a compendium of expression profiles." In: *PLoS biology* 5.1, e8.

Freeman, Linton C. (1977). "A Set of Measures of Centrality Based on Betweenness". In: *Sociometry* 40.1, p. 35.

Geisberger, Robert, Peter Sanders, and Dominik Schultes (2008). "Better Approximation of Betweenness Centrality". In: *Time*, pp. 90–100.

Girvan, M. and M. E. J. Newman (2002). "Community structure in social and biological networks". In: *Proceedings of the National Academy of Sciences* 99.12, pp. 7821–7826. arXiv: `0112110 [cond-mat]`.

Guo, Shun et al. (2016). "Gene regulatory network inference using PLS-based methods". In: *BMC Bioinformatics* 17.1, p. 545.

Haury, Anne-Claire et al. (2012). "TIGRESS: Trustful Inference of Gene REgulation using Stability Selection". In: *BMC Systems Biology* 6.1, p. 145. arXiv: `arXiv:1205.1181v1`.

Huynh-Thu, Vân Anh et al. (2010). "Inferring regulatory networks from expression data using tree-based methods". In: *PLoS ONE* 5.9, pp. 1–10.

Katz, Leo (1953). "A new status index derived from sociometric analysis". In: *Psychometrika* 18.1, pp. 39–43.

Küffner, Robert et al. (2012). "Inferring gene regulatory networks by ANOVA". In: *Bioinformatics* 28.10, pp. 1376–1382.

Marbach, Daniel et al. (2012). "Wisdom of crowds for robust gene network inference". In: *Nature Methods* 9.8, pp. 796–804.

Margolin, Adam A et al. (2006). "ARACNE: an algorithm for the reconstruction of gene regulatory networks in a mammalian cellular context." In: *BMC bioinformatics* 7 Suppl 1, S7. arXiv: `0410037 [q-bio]`.

Page, Lawrence et al. (1998). "The PageRank Citation Ranking: Bringing Order to the Web". In: *World Wide Web Internet And Web Information Systems* 54.1999-66, pp. 1–17. arXiv: `1111.4503v1`.

Riondato, Matteo and Evgenios M. Kornaropoulos (2016). "Fast approximation of betweenness centrality through sampling". In: *Data Mining and Knowledge Discovery* 30.2, pp. 438–475.

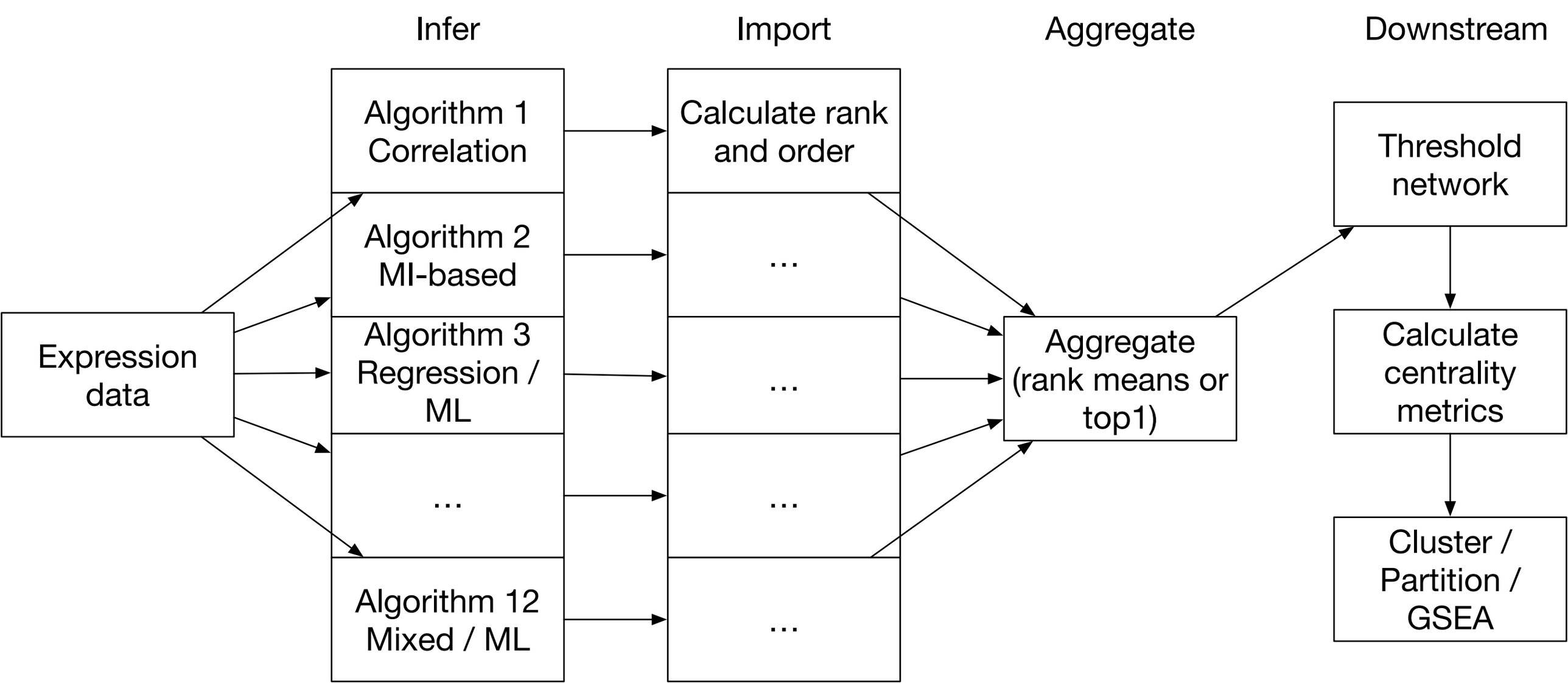
Ruyssinck, Joeri et al. (2014). "NIMEFI: Gene regulatory network inference using multiple ensemble feature importance algorithms". In: *PLoS ONE* 9.3.

Schäfer, Juliane and Korbinian Strimmer (2005). "A Shrinkage Approach to Large-Scale Covariance Matrix Estimation and Implications for Functional Genomics". In: *Statistical Applications in Genetics and Molecular Biology* 4.1.

Staudt, Christian, Aleksejs Sazonovs, and Henning Meyerhenke (2014). "NetworKit: An Interactive Tool Suite for High-Performance Network Analysis". In: *CoRR* abs/1403.3005. arXiv: `1403.3005`.

Teixeira, a S et al. (2013). "Spanning edge betweenness". In: *Workshop on Mining and Learning with Graphs*.

Zhang, Xiujun et al. (2013). "NARROMI: A noise and redundancy reduction technique improves accuracy of gene regulatory network inference". In: *Bioinformatics* 29.1, pp. 106–113.


Infer
Import
Aggregate
Downstream
Expression data
Algorithm 1 Correlation
Algorithm 2 MI-based
Algorithm 3 Regression / ML
...
Algorithm 12 Mixed / ML
Calculate rank and order
...
...
...
...
Aggregate (rank means or top1)
Threshold network
Calculate centrality metrics
Cluster / Partition / GSEA