Molecular BioSystems

PAPER

View Article Online
View Journal | View Issue

Cite this: *Mol. BioSyst.*, 2017, **13**, 1304

GeneSPIDER – gene regulatory network inference benchmarking with controlled network and data properties†

Andreas Tjärnberg, ‡[abc] Daniel C. Morgan, ‡[ab] Matthew Studham,[ab] Torbjörn E. M. Nordling §*[ad] and Erik L. L. Sonnhammer§*[ab]

Received 25th January 2017,
Accepted 20th April 2017

DOI: 10.1039/c7mb00058h

rsc.li/molecular-biosystems

A key question in network inference, that has not been properly answered, is what accuracy can be expected for a given biological dataset and inference method. We present GeneSPIDER – a Matlab package for tuning, running, and evaluating inference algorithms that allows independent control of network and data properties to enable data-driven benchmarking. GeneSPIDER is uniquely suited to address this question by first extracting salient properties from the experimental data and then generating simulated networks and data that closely match these properties. It enables data-driven algorithm selection, estimation of inference accuracy from biological data, and a more multifaceted benchmarking. Included are generic pipelines for the design of perturbation experiments, bootstrapping, analysis of linear dependence, sample selection, scaling of SNR, and performance evaluation. With GeneSPIDER we aim to move the goal of network inference benchmarks from simple performance measurement to a deeper understanding of how the accuracy of an algorithm is determined by different combinations of network and data properties.

1 Introduction

The goal of gene regulatory network (GRN) inference is to understand how genes influence each other in terms of their expression, *i.e.* to unravel the transcriptional regulatory influences.[12] A primary objective in network inference is to obtain a network where each link corresponds to a true regulatory interaction in the biological system, *i.e.* to avoid false positives. A secondary objective is that each real link becomes inferred, *i.e.* to avoid false negatives. The quality of the inferred network model, that is, the number of false positives and negatives, depends both on the inference method and the data. Here we focus on dynamic models of regulatory networks and algorithms that are designed to take these effects into account.

Benchmarks are a common method to objectively compare performance of different network inference algorithms. Famous challenges, such as the DREAM challenge,[14] are designed around data sets from multiple "gold standard" biological systems where the known links or the network's main properties are assumed true.

In practice, artificial networks and data from simulated experiments are commonly used due in part to the current lack of a "true" network of gene regulation for any biological system, as well as the temporal and monetary ease of generating data *in silico*. In particular, the properties of the networks and data are easier to adapt to the specific needs of a project. Simulations often meet biological experiments half way by incorporating knowledge derived from *in vivo* data to create realistic networks and *in silico* datasets.[2,11,16,19] For example, a substantial number of transcriptional regulatory interactions are known for *Escherichia coli* and *Saccharomyces cerevisiae*, and can be used when deciding the structure of the artificial network model.[23,25] However, it is important to distinguish between (i) networks and data that are intended to mimic real biology as closely as possible, and (ii) networks and data that are intended to be approximations, only capturing certain properties of their counterparts.

The network construction process usually involves modelling networks as a system of ordinary differential equations (ODEs), with *e.g.* linear,[2] or nonlinear[24,29] models. The perturbation, *i.e.* system input, used in the experimental design is typically a single gene knockdown/knockout, or a system wide perturbation, *i.e.* a perturbation that randomly alters multiple genes at a time.

[a] *Stockholm Bioinformatics Center, Science for Life Laboratory, Sweden. E-mail: torbjorn.nordling@nordlinglab.org, erik.sonnhammer@scilifelab.se*
[b] *Department of Biochemistry and Biophysics, Stockholm University, Sweden*
[c] *Department of Physics, Chemistry and Biology, Linköping University, Sweden*
[d] *Department of Mechanical Engineering, National Cheng Kung University, No. 1 University Road, 70101, Tainan, Taiwan*

† Electronic supplementary information (ESI) available: Source code freely available for download at https://bitbucket.org/sonnhammergrni/genespider, implemented in Matlab. See DOI: 10.1039/c7mb00058h

‡ These authors contributed equally to this work.

§ These authors contributed equally to this work.

To streamline the creation of new benchmarks of network inference methods several packages have been published to facilitate the creation a benchmark data sets and networks.[5,10,24,29] A common feature of these benchmarking packages is the focus on network structure as the main feature controllable by the user. *SynTReN*[29] utilizes selected dynamic structures and sub-networks from biological systems, and *E. coli* and *S. cerevisiae. netsim*[5] takes a different approach by generating dynamical models, incorporating structurally random network motifs, which they dub modular topology models. *GeNGe*[10] uses a nonlinear model similar to *SynTReN* and allows for the generation of networks with specific network motifs. All the above methods utilize some form of non-linear dynamics to produce gene expressions from the network structure.

The GRN inference community has organized around challenges, such as the DREAM challenge, to benchmark network inference methods by applying them to the same data and comparing their performance.[14] The package *GeneNetWeaver*[24] expands upon ideas similar to those of the *SynTReN* and *GeNGe* packages and is used to create datasets for these challenges. *GeneNetWeaver* enables the user to choose among a variety of *in silico* standard perturbation designs when generating networks and time-series data, as well as to define the number of nodes and in-degree. Its nonlinear dynamical model and simulated data are based on several types of omics data in order to mimic a real biological system.

Publicly available gene expression datasets typically suffer from few data points compared to the high number of genes and possible interactions, large measurement uncertainty both in the perturbations and responses, *i.e.* poor signal to noise ratio (SNR), and redundant nearly collinear variables, *i.e.* ill-conditioned data matrices.[17] Considering this, previously published benchmark packages support surprisingly few perturbation design alternatives and data properties. Noise is typically added to the input and/or output as a percentage of the magnitude of the applied perturbation or measured signal, rather than the SNR. Due to the ill-conditioning, the former does not typically allow for control of the later. In a similar ambition as Kurtz *et al.*,[21] we seek to control certain aspects of data generation. However, no previously published package makes it clear to what extent the specific model, or the perturbation design, effects the data generated or the inference being made. None of the previously published packages facilitate this analysis. Nor do they facilitate control of observable data properties, such as the SNR and ill-conditioning, which as we have demonstrated should be used to guide the algorithm selection.[28]

The DREAM challenge, for example, does not define observable properties that could guide a choice of inference method or give an estimate of how informative the dataset is. Instead it focuses on network properties and motifs when trying to evaluate to what extent a specific inference approach is useful. While this might reveal hard to infer motifs and network structures, it becomes meaningless when faced with a dataset since the motifs can not be observed *a priori*. Furthermore, the perturbation design is not connected to the network properties, thus equal perturbation design could give arbitrarily "good" or "bad" data given a specific network model. While an algorithm may generally perform well, it might act to the contrary faced with a specific set of data properties.

A more recent benchmarking package, NetBenchmark,[4] aims to evaluate inference methods by bundling datasets from *Rogers simulator*,[22] *SynTReN* and *GeneNetWeaver*, and benchmarking across several existing inference algorithms in a unified workbench. However, the fundamental issues of the methods aggregated are neither solved nor addressed, simply evaluated in the same way. Benchmarks should provide guidance on both (i) which inference method to use for a particular dataset, and (ii) which errors to expect in the inferred network. Benchmarking should be a data driven procedure.

Here we present GeneSPIDER, a benchmarking suite designed specifically to deal with the issues presented above. GeneSPIDER focuses on key properties influencing inference performance,[28] allowing the user to "tune" certain parameters in the context of the core functionality expected from a benchmarking suite. In the generation of the dynamical network model, GeneSPIDER focuses on stability[32] and interampatteness.[18] By interampatteness we mean the ratio between the system's ability to amplify and attenuate different signals, which for a linear system corresponds to the condition number of the network matrix. This is strongly correlated with the difficulty of inferring the correct network, as attenuated signals typically are hidden in the noise and essentially determines the accuracy of the inferred network, since network inference is an inverse problem.[17]

The observed signal is determined by the applied perturbation and the attenuation of it by the system. The strength of the perturbations could and should be scaled to counteract the intrinsic attenuation of signals. The condition number of the dataset has been shown to serve as a proxy for the interampatteness in the context of known perturbations.[17,28] The benefit here is that the condition number of the dataset can be observed and hence provide information on how difficult it is to infer a specific network from the available data.

The noise estimate in the direction of the data's smallest variation is expected to have the greatest impact on the data quality, as in any inverse problem. Therefore we provide variation estimates and apply noise scaled towards the principle directions of variance in the data (by default the smallest). Since the variation in the data can be distributed very differently among the directions from one dataset to the next, this is necessary to correctly see what effect a certain level of noise can have on the data. This makes it possible to have two observable properties, signal to noise ratio and the condition number of the data, as tune-able parameters not previously incorporated into benchmarking pipelines, yet informative for property-dependent error analysis and confidence estimation. This informs the user how informative the data set is and what performance should be expected from different methods, as shown previously.[27,28]

GeneSPIDER improves upon existing packages by providing a data driven approach to benchmarking focused on independently controlling properties of network and data. This is essential for establishing the relation between properties and inference accuracy. One of the most important questions in network inference is what accuracy can be expected for a given

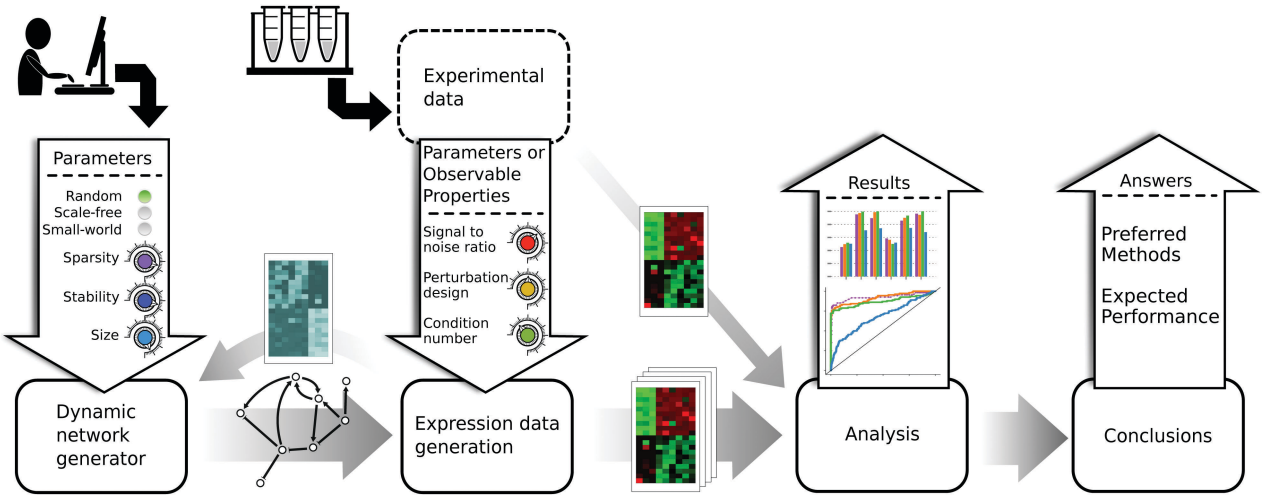

Fig. 1 Schematic workflow of network and data generation followed by analysis in GeneSPIDER. (i) Import or generate networks with a range of properties; stabilize and modify the interampatteness degree. (ii) Generate expression data with specific SNR, perturbation design, and condition number. (iii) Analyze GRN inference benchmark results in relation to the network and data properties. (iv) Draw conclusions about which methods to use under what conditions. The figure also shows an alternative path where GeneSPIDER extracts properties from experimental data in order to generate simulated networks and data. Such a benchmark provides information about the expected accuracy for a given real dataset.

real dataset. GeneSPIDER is able to address this question by extracting salient properties from experimental data and then generate simulated networks and data that closely match these properties see Fig. 1. In this article we demonstrate the workflow of Fig. 1, with code examples given in the ESI,† in order to lower the initial usage threshold, focusing on generating approximations with tuned properties and avoiding unnecessary complexity in order to gain insight.

2 Methods and GeneSPIDER capabilities

The functions in the following sections cover the five typical phases in benchmarking of inference methods based on *in silico* experiments: generation of network models (phase A), design of perturbation experiments (phase B), simulations of the designed experiments on the generated systems (phase C), inference of network models from the simulated datasets using the methods of interest (phase D), and analysis of the performance of the methods (phase E).

2.1 Model formalism and notation

A linear dynamical system described by ordinary differential equations (ODEs),

$$\frac{\mathrm{d}}{\mathrm{d}t}\check{x}_i(t) = \sum_{j=1}^{N} \check{a}_{ij}\check{x}_j(t) + p_i(t) - f_i(t)$$
$$y_i(t) = \check{x}_i(t) + e_i(t), \quad (1)$$

is used to approximate the local behaviour of a biological system in the current version of GeneSPIDER. This approximation is commonly used to describe GRNs, see *e.g.* Gardner *et al.*,[8] Yuan *et al.*[31] In GRNs the state vector $\check{x}(t) = [\check{x}_1(t), \check{x}_2(t),\ldots,\check{x}_N(t)]^T$ represents actual mRNA expression changes relative to the initial state of the system in N genes. The vector $p(t) = [p_1(t), p_2(t),\ldots,p_N(t)]^T$ represents the applied perturbation, which, in general, is corrupted by the noise $f(t)$. The perturbations could be *e.g.* gene siRNA knock-downs or gene over-expressions using a plasmid with an extra copy of the gene. The response vector $y(t) = [y_1(t), y_2(t),\ldots,y_N(t)]^T$ represents the measured expression changes, which is the "true" expression corrupted by the noise $e(t)$. Our network is the interaction matrix with parameters $\check{a}_{ij}$, which represent the influence of an expression change of gene j on gene i. A positive value of $\check{a}_{ij}$ represents an activation, while a negative value represents an inhibition. The value of the $\check{a}_{ij}$ parameter gives the relative strength of the interaction. Here $\vee$ is used to mark "true" variables that are not corrupted by noise or measurement errors.

2.2 Network generation

Network generation (phase A) is done in four steps: (i) generation of a network graph with desired topology, (ii) assignment of random weights to the links, (iii) stabilization of the system, and (iv) tuning of the IAA degree of the system corresponding to the network.

GeneSPIDER includes three algorithms for directed and undirected network topology generation (step i): random, scale-free, and small-world; auto-regulation/degradation self-loops are supported for directed networks. The desired number of links is set by providing a sparsity parameter to the algorithm. The sparsity is defined as $s = L/N^2$, where L is the number of links in the network and N^2 the number of possible links in a directed graph. The algorithm generating random graphs, where each link has the same probability p_1 to exist, is based upon the assumption that every graph, $G(N,L)$, is equally probable.[6] The algorithm generating scale-free graphs,[3] where each node added to the network is attached to previous nodes with a probability p_1

proportional to the number of edges each of the previous nodes has, is based upon the preferential attachment principle.[1]

This algorithm produces a network with bidirectional links only. However, directed biological networks rarely have bidirectional links, so we at random eliminate one of the links in each symmetric link pair with probability p_{remove}, implying that we keep both with probability $1 - p_{\text{remove}}$. When eliminating a link we eliminated the outgoing link a_{ji} with probability p_{outgoing}.

The algorithm generating small-world graphs,[30] where nodes are serially connected with the l closest neighbours in a ring lattice, is based on the implementation suggested by Prettejohn *et al.*[20] Each node is then considered to randomly be disconnected from a neighbour and reconnected to another node in the ring lattice. Graphs that are partly scale-free can be generated by seeding the preferential attachment algorithm with *e.g.* a small-world network so that a small-world-scale-free (SW-SF) network is generated.

After the topological layout, a weight is assigned to each link present by assigning weights drawn from a iid normal distribution with mean zero and standard deviation one (step ii). This is done in order to convert the graph into a dynamical system. The weight $\check{a}_{ij}$ of each link that is present determines the effect that node j has on node i.

Since a stable system guarantees that all quantities remain finite, a stable network model is typically desired. The weighted network may therefore need to be stabilized (step iii) and have its condition number tuned, *i.e.* tuning of the interampatteness (IAA) degree.[18] IAA is a generic property of biochemical networks, akin to the data property collinearity. When assigning random weights to each link there is no guarantee that the resulting system is stable. Stability is guaranteed if all eigenvalues of the interaction matrix are negative. Data generation for benchmarking of inference algorithms should be restricted to stable systems, because, while the local linearisation of a globally stable nonlinear system can be unstable,[13] it is not possible to infer an unstable linear system in practice; even the smallest amount of process noise will move the system away from the unstable steady-state. Note that every isolated system must be globally stable, or risk some state variable (mRNA abundance) becoming infinite and violating mass-conservation laws. We assure stability by employing a method that forces all eigenvalues of our interaction matrix to be negative,[32] enabling the creation of a stable linear dynamic system from most unstable network graphs. However, this does not work for all graphs because the algorithm is based on making the diagonal dominating and negative, whereupon addition of a few links typically solves the problem. The algorithm forces the eigenvalues to be negative by incorporation of a convex optimization protocol.[9] While convexity guarantees that the algorithm for each initial network returns only one final network, two different initial networks may yield the same final network. Desired network properties can be arrived upon by iterating over the substantially reduced space of stable linear dynamical systems.

The IAA degree tuning (step iv) is enabled by the user defining limits on the eigenvalues of the network in the convex optimization. This sets an upper bound on the inverse condition number, which can be infeasible or arbitrarily conservative. In practice, one therefore needs to generate a network ensemble with different random topological weights and select those of condition number (IAA degree) within the desired interval. Note that links are removed if their strength falls below a user defined threshold. This ensures that we keep link strengths within a reasonable range to avoid trivial links as well as possible numerical issues when working with the network in subsequent steps. However, this can result in a slight deviation from the desired sparsity.

2.3 Data generation

The purpose of the experiment design (phase B) is to generate a perturbation matrix P, containing either a sequence of perturbations in a time-series experiment or a sequence of steady-state experiments. Only a small number of genes should be perturbed in each perturbation, so that it may be practically implemented *in vivo* or *in vitro* given current technical limitations. Publicly available real gene expression datasets have either been generated by perturbing one gene at a time (single gene perturbation), two genes at a time (double gene perturbations), or by a system perturbation, such as a change in environmental factors or introduction of a drug, which is thought to affect several genes at once but typically is quantitatively unknown. These perturbations are trivial to generate in MATLAB using *e.g.* the function for generation of diagonal random matrices diag(randn(N,M)) or sparse random matrices sprandn(N,M).

We also provide a method to generate a pseudo-optimal perturbation design that counteracts intrinsic signal attenuation based on an inversion of the "true" system. Small elements in the designed perturbation matrix P are removed to make it sparse, while keeping the condition number of the response matrix $\kappa(Y)$ close to 1. This implies that the response matrix will be close to an identity matrix, with equal signal strength in N linearly independent directions. Note that while this method cannot be used for biological experiments (because the "true" system is unknown), it can be used to generate informative datasets for benchmarking.

The experiments are simulated (phase C) either by calculating the steady-state response, as in eqn (S2) (ESI†), or the time-series response, as in eqn (1). The system dynamics are simulated with a time step $t = \tau_N/10$, where τ_N is the smallest time constant of the system given by the eigenvalues of the network matrix. We typically first simulate noise-free measurements and then add noise drawn from the desired distribution, typically a standardized normal distribution with zero mean and variance one. This enables us to tune the SNR by scaling the noise matrix, which for normally distributed noise corresponds to scaling of the standard deviation, such that the desired SNR is obtained.

2.4 GRN inference methods

A core piece of any inference pipeline is the inference method itself, which takes data and algorithm parameters as input and yields one or several network estimates as output (phase D). In GeneSPIDER we have created wrappers with a common entry format for a number of published and in-house inference methods to simplify their use, see Table S13 (ESI†). Wrappers make it easy

to incorporate future inference algorithms and provide the same benchmark capabilities in our standardised environment.

We here introduce a modified version of the Robust Network Inference algorithm (RNI)[17] by applying a range of cut-offs to the confidence scores, and call this version RNI cut-off (RNICO). This way different network estimates ranging from empty to full are obtained by gradually lowering the confidence cut-off independent of how informative the data is, while native RNI for uninformative data produce an empty network. The significance level $\alpha = 0.01$ was used in RNICO to scale Nordling's confidence score such that every link with a value above one exists in the true network for 99 out of every 100 links fulfilling this criterion.

2.5 Evaluation of inferred networks

Measures of network similarity are needed to assess the performance of any given inference method (phase E), and we have decided to implement many such measures to accommodate researchers' preference. GeneSPIDER currently provides a function for comparison of networks that includes 18 system measures, 3 signed topological measures, 1 correlation measure, 12 graph measures, and 9 directed graph measures, see Section S1.4.2.3 (ESI†). The GeneSPIDER toolbox includes many standard measures of performance, including MCC and AUROC, with which to compare available inference pipelines and tune such for specific dataset properties.

2.6 Quantification of network and system properties

To investigate if a relation exists between a network property and the performance of an inference algorithm, the network property first needs to be objectively quantified and a measure of it implemented. GeneSPIDER contains functionality for the calculation of basic graph properties, such as number of nodes and links, basic topology measures reported in Prettejohn *et al.*[20] such as the clustering coefficient, the degree distribution, the average path length, and advanced measures such as the number of strong components. We calculate these measures for each network structure based on the network type, either directed or undirected. We have also implemented functions for calculation of system properties, such as the time constant and IAA degree.

2.7 Quantification of data properties

For the expression data we have tried to quantify the informativeness and difficulty to infer the correct network in terms of signal to noise ratio, similarity between experiments, and the condition number. GeneSPIDER contains several different SNR calculations; see definitions in Section S1.5.1 (ESI†). The condition number is calculated with the function cond.

3 A case study using the benchmark suite

To demonstrate the application of GeneSPIDER we generated a benchmark suite of 200 networks and 600 expression datasets that includes networks of four different topologies with varied IAA degrees and sizes. Based on these we simulated data each with three SNR variants, and analyzed the performance of four GRN inference methods in relation to the properties of the networks and expression data. The four methods tested in this benchmark are ARACNe,[15] least squares cut-off (LSCO),[27] RNICO, and the Glmnet implementation of LASSO.[7,26] Fig. 1, gives an overview of the GeneSPIDER workflow and some parameters of networks and data that are possible to vary.

3.1 Network generation

We generated 10 networks of sizes $N \in \{10, 50, 100\}$ for each of the four classes: random, small-world, scale-free and small-world-scale-free, and for each of two IAA levels, $\kappa \in \{\text{low}, \text{high}\}$. In total 200 networks were generated. Scale-free and small-world-scale-free are missing for $N = 10$ because this is to small for scale-free like properties to be relevant. The small-world networks only have high IAA degree because of limitations in the IAA adjustment method compounded by the fact that cascades, which are present in small-world networks due to the initial ring lattice, in general increases the IAA degree.[18] The two IAA degree levels were: IAA degree of $9N$ to $11N$ (high) and IAA degree of $0.5N$ to N (low). We scaled the IAA degree with the network size because the condition number of a random matrix with normally distributed entries tends to grow with the matrix size. The IAA degree of each network is shown in Fig. S1 (ESI†).

IAA can be difficult to tune for certain network structures and sparsity levels, so we allowed some flexibility relative to the specified sparsity. In particular, networks with a low IAA degree are difficult to find for very sparse networks, since they tend to contain cascade and feedback loops that in general increase the IAA degree. We therefore implemented a larger sparsity coefficient for these sparse networks, *i.e.* the addition of one or two links to the network, with secondary link addition if the algorithm had trouble stabilizing the network at the desired IAA degree. Performance analysis of the network inference methods was carried out on multiple 10-gene random networks.

The sparsity coefficient was set for each network generation method, scaled by the size of the network to maintain a relatively low mean degree per node, so all networks can be considered truly sparse. The sparsity ranges of the different network topology classes and sizes in this benchmark are shown in Table 1. The degree distributions for each network topology class and size are shown in Fig. S2 (ESI†).

3.2 Data generation

Data was generated at three signal-to-noise levels for each network in Section 3.1. The common gene-by-gene steady-state perturbation paradigm was repeated three times, *i.e.* $P = [I, I, I]$, giving a total of $M = 3 \cdot N$ samples of simulated expression changes in each gene in the response matrix Y. Independent, normally distributed noise with zero mean and variance λ was added to each expression change. Only one noise realization was generated per network size to avoid consideration of differences in noise when comparing the inference result within each group. The variance of the noise was scaled according to equation (S9) (ESI†)

Table 1 Properties of networks in this benchmark. Self-loops are included. SW-SF stands for small-world and scale-free

Network class	N	Sparsity	Links	Avg. degree
Random	10	0.25	25	2.5
	50	0.062–0.064	155–160	3.1–3.2
	100	0.038–0.04	380–400	3.8–4.0
Small-world	10	0.19–0.3	19–30	1.9–3.0
	50	0.096–0.1	240–250	4.8–5.0
	100	0.048–0.068	480–680	4.8–6.8
Scale-free	50	0.092–0.096	230–240	4.6–4.8
	100	0.079–0.084	790–840	7.9–8.4
SW-SF	50	0.086–0.13	215–325	4.3–6.5
	100	0.075–0.092	753–920	7.53–9.2

to give exactly SNR $\in \{0.01, 1, 100\}$, with significance level $\alpha = 0.01$ in all cases.

The dependence of the condition number, $\kappa(Y)$, on SNR is shown in Fig. 2. Compared to Fig. S1 (ESI†), it is clear that for SNR ≥ 1, $\kappa(Y)$ closely follows IAA degree as it should since $P = [I, I, I]$. For SNR < 1 the noise in some cases has a strong effect on $\kappa(Y)$ as expected.

3.3 Inference method performance analysis

A measure of similarity between the inferred network and the "true" network used to generate the data is needed to evaluate the performance of inference methods. We here used area under the receiver operating characteristic (AUROC). The receiver operating characteristic (ROC) depicts the true-positive rate (TPR), *i.e.* sensitivity or recall, against the false-positive rate (FPR), *i.e.* fall-out, as a function of the regularisation parameter for LSCO and Glmnet or the confidence score for RNICO and ARACNe, see the example in Fig. 3.

By inferring networks for 100 different values of the regularisation parameter or each confidence score that adds a link we brought the sparsity of the inferred network from an empty network to a full network. An AUROC of one corresponds to inference of the true network for some regularisation parameter or confidence score with inclusion of all existing links first and then all non-existing ones, while zero corresponds to inclusion of all non-existing links first and then all existing ones such that all networks that can be inferred have the least resemblance to the true network. To compare the four inference methods across the networks and data sets in the benchmark suite, we summarize their AUROC values as bars plots in Fig. 4. As ARACNe is unable to predict self-loops, we also made a separate benchmark that ignores self-loops entirely (Fig. S3, ESI†).

Some major trends can be observed:

- LSCO and RNICO always give the most accurate networks in cases of SNR 100. The AUROC is almost one and the correct network can in most cases be inferred. RNICO is the preferred method because it can be used to prove the existence of links and provides reliable confidence scores under mild assumptions. To obtain an accurate network using LSCO the correct ζ value must be selected, while RNICO for a large range of ζ values will provide accurate networks (ESI,† figures, MCC panels). However, the correct cut-off to use in RNICO is to set Nordling's confidence score to one, since all links with confidence above one can be proven to exist.
- Glmnet and ARACNe always fail to infer the correct network for SNR 100, even when LSCO and RNICO succeed, and provide network estimates that are inferior to LSCO and RNICO. This surprising property of Glmnet was previously observed in ref. 28.
- ARACNe cannot infer self-loops by design, *i.e.* diagonal elements of the interaction matrix. It should therefore be compared to its own null model. Even though it can perform better than

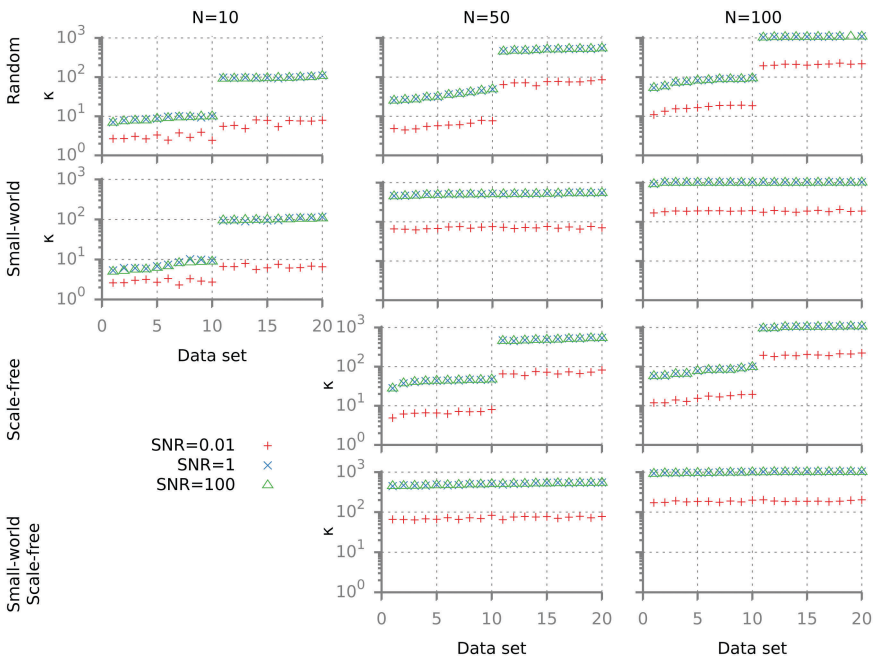

Fig. 2 Condition numbers of the response matrices in this benchmark. Three different SNRs were used: 100, 1, and 0.01. The y-axis shows the condition number $\kappa(Y)$, and the x-axis shows the number in order of increasing IAA.

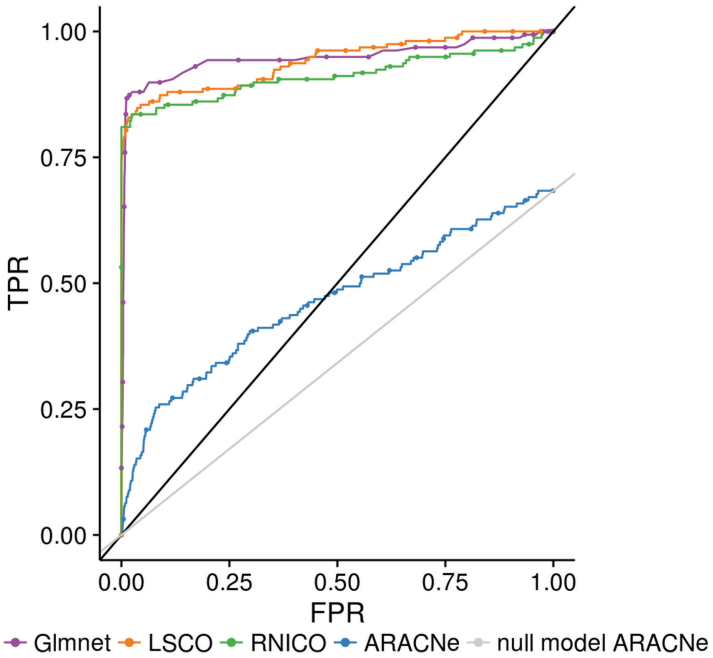

Fig. 3 The ROC curve of four inference methods-Glmnet (LASSO), LSCO, RNICO, and ARACNe-on the Random 50 gene, high IAA, SNR 1 dataset generated by GeneSPIDER. Note that ARACNe cannot infer self-loops and therefore has a different null model.

random, *i.e.* its null model, in most cases its performance is very poor. ARACNe is clearly outperformed by the other methods in all cases with SNR = 1 and 100. For SNR = 0.01 it performs similarly to the other methods when not counting self-loops (Fig. S3, ESI†), but generally close to 0.5 which is the accuracy of random inferences.

• Glmnet, LSCO, and RNICO have similar AUROC values when SNR = 1, but a good network estimate is given for a narrow range of ζ values for Glmnet and LSCO, while RNICO yields good estimates for a broad range of values (ESI,† figures, MCC panels). It is hence easier to obtain a good estimate using RNICO. None of the links can typically be proven to exist using RNICO in this case and the threshold for the confidence score must be selected far below one to get good network estimates.

• Glmnet is the best method for SNR = 0.01 and N = 100. In this setting the AUROC values are always below 0.8 however, thus the inferred networks are quite unreliable.

• In this dataset, changing only the interampatteness gave no clear trend across all methods and property settings.

• Comparing Fig. 3 with Fig. S3 (ESI†), *i.e.* benchmarking with and without considering self-loops, shows that for Glmnet, LSCO, and RNICO this mainly has an effect for SNR = 0.01, where the self-loops improve the performance. For N = 50 and SNR = 0.01, the self-loops alone explain the improvement above random assignment (AUROC = 0.5). As expected, ARACNe always scores better when not considering self-loops.

3.4 Benchmark conclusions

The goal of GeneSPIDER is to provide a common environment for network inference testing and evaluation. We have demonstrated some of its capabilities by generating a benchmark suite of networks and datasets, and used it to analyze the performance of four inference methods.

It is clear from the benchmarking that mutual information based methods such as ARACNe are generally performing substantially worse than system-optimising methods. This highlights the importance of understanding the system as a whole, which is not achievable by unconnected calculations on the links. ARACNe is further hampered by the fact that it cannot predict self-loops, which are important to bring the system to a stable state, but even if self-loops are ignored its performance is far below the other methods on the two high-SNR settings. In general, we observed a larger difference in performance in the jump from SNR 0.01 to 1 than from 1 to 100, suggesting a threshold level of noise from which knowledge is recoverable, after which point the noise decrease is less relevant.

Once an algorithm is decided upon, having been vetted against methods of varying strengths and weaknesses, the capability of that choice must be evaluated. Again, our benchmark allowed us to draw the following conclusions: all methods perform poorly when the SNR is below one because the data simply is not informative enough for network inference. For high SNRs RNI yields networks that are identical or almost identical to the "true" network.

4 Discussion

GeneSPIDER is a package for GRN inference analysis, for method choice, evaluation and optimization. It allows the user to control both network and data properties, and contains methods for exploring and analyzing these properties, as well as those of the inference method and its performance. This opens up new possibilities to understand how these properties affect the performance of inference algorithms.

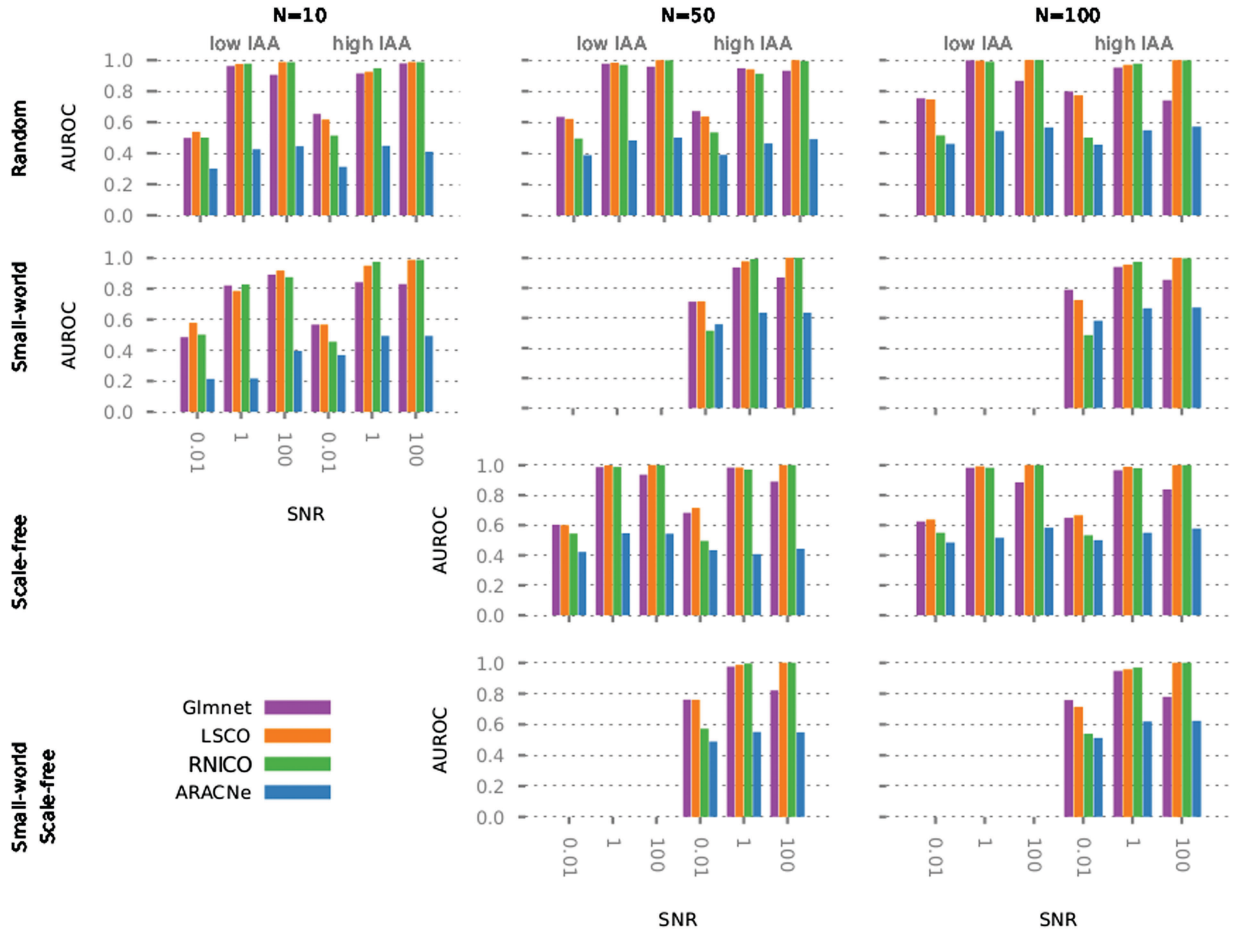

Fig. 4 Performance of four inference methods – ARACNe, Glmnet (LASSO), LSCO, and RNICO – on the benchmark suite generated by GeneSPIDER when considering self-loops. The AUROC of an inference method is shown by coloured bars, which are grouped according to IAA degree of the network and SNR of the data (*x*-axis). The highest and lowest IAA networks were selected among the 20 networks in each group. Individual ROC curves, MCC plots per sparsity, and density plots per sparsity are available for each condition set in ESI† figures including and excluding self-loops.

GeneSPIDER can create networks with different topological properties that are typical for biological networks. Given a network, GeneSPIDER can then simulate expression data that is as informative as typical biological datasets. However, its unique strength is varying these properties across a wide range of values to gain insights about the limitations and expected performance of a given inference method. After all, biological data from different sources varies in terms of network and data properties, and the performance of inference methods depend on these properties. Therefore, when a network inference method is applied to a biological dataset, it is important to make sure that the method has been benchmarked on simulated data with the same properties. This is not standard practice today, even though it would provide information about the reliability of the inferred network. GeneSPIDER can also be used to investigate how network properties such as topology and IAA affect data properties. The nature of the generated data also depends on the experimental design employed for perturbation. Therefore, GeneSPIDER supports a range of different experimental design schemes. For instance, one can control the number of experiments and how many genes are perturbed in each experiment. This can be of great value when designing wet lab experiments – if a particular design gives optimal results in a benchmark, then it should be the preferred choice when performing real experiments.

It has previously been observed that L_1 methods such as Glmnet perform considerably better at low than at high IAA.[28] Such a trend was not clearly observable in this benchmark, probably because the low IAA setting here was not as low as in the previous study.

The GeneSPIDER package is implemented in MATLAB and provides native storage functionality, as well as export options to formats more easily acceptable by other programming languages, including XML and JSON. GeneSPIDER fills a gap between theoretical analysis and experimental setup and could be incorporated in many current GRN inference and benchmarking pipelines.

Acknowledgements

This work was partly supported by the Swedish strategic research program eSSENCE, a startup grant by the National Cheng Kung University, and a grant from the Ministry of science and technology in Taiwan (105-2218-E-006-016-MY2).

References

1 R. Albert and A.-L. Barabási, Statistical mechanics of complex networks, *Rev. Mod. Phys.*, 2002, **74**, 47–97.
2 M. Bansal, V. Belcastro, A. Ambesi-Impiombato and D. Di Bernardo, How to infer gene networks from expression profiles, *Mol. Syst. Biol.*, 2007, **3**(78), 78.
3 A.-L. Barabási and R. Albert, Emergence of scaling in random networks, *Science*, 1999, **286**(5439), 509–512.
4 P. Bellot, C. Olsen, P. Salembier, A. Oliveras-Verges and P. Meyer, Netbenchmark: a bioconductor package for reproducible benchmarks of gene regulatory network inference, *BMC Bioinf.*, 2015, **16**(1), 312.
5 B. Di Camillo, G. Toffolo and C. Cobelli, A gene network simulator to assess reverse engineering algorithms, *Ann. N. Y. Acad. Sci.*, 2009, **1158**(1), 125–142.
6 P. Erdos and A. Rényi, On the evolution of random graphs, *Bull. Inst. Internat. Statist*, 1961, **38**(4), 343–347.
7 J. Friedman, T. Hastie and R. Tibshirani, Regularization paths for generalized linear models *via* coordinate descent, *J. Stat. Softw.*, 2010, **33**(1), 1–22.
8 T. S. Gardner, D. Bernardo, D. Lorenz and J. J. Collins, Inferring genetic networks and identifying compound mode of action *via* expression profiling, *Science*, 2003, **301**(7), 102–105.
9 M. Grant, S. Boyd and Y. Ye, CVX: Matlab software for disciplined convex programming, 2008.
10 H. Hache, C. Wierling, H. Lehrach and R. Herwig, Genge: systematic generation of gene regulatory networks, *Bioinformatics*, 2009, **25**(9), 1205–1207.
11 H. Hache, H. Lehrach and R. Herwig, Reverse engineering of gene regulatory networks: A comparative study, *EURASIP J. Bioinf. Syst. Biol.*, 2009, **2009**(1), 617281.
12 M. Hecker, S. Lambeck, S. Toepfer, E. van Someren and R. Guthke, Gene regulatory network inference: data integration in dynamic models – a review, *Bio. Systems*, 2009, **96**(1), 86–103.
13 H. K. Khalil and J. Grizzle, *Nonlinear systems*, Prentice hall, New Jersey, 1996, vol. 3.
14 D. Marbach, J. C. Costello, R. Küffner, N. M. Vega, R. J. Prill, D. M. Camacho, K. R. Allison, M. Kellis, J. J. Collins and G. Stolovitzky, Wisdom of crowds for robust gene network inference, *Nat. Methods*, 2012, **9**(8), 796–804.
15 A. A. Margolin, I. Nemenman, K. Basso, C. Wiggins, G. Stolovitzky, R. D. Favera and A. Califano, ARACNE: an algorithm for the reconstruction of gene regulatory networks in a mammalian cellular context, *BMC Bioinf.*, 2006, **7**(suppl 1), S7.
16 V. Narendra, N. Lytkin, C. Aliferis and A. Statnikov (2011). *A comprehensive assessment of methods for de-novo reverse-engineering of genome-scale regulatory networks*. *Genomics*, **97**(1), 7–18.
17 T. E. M. Nordling, Robust inference of gene regulatory networks, PhD thesis, KTH School of Electrical Engineering, Automatic Control Lab, 2013.
18 T. E. M. Nordling and E. W. Jacobsen, Interampatteness – a generic property of biochemical networks, *IET systems biology*, 2009, **3**(5), 388–403.
19 C. a. Penfold and D. L. Wild, How to infer gene networks from expression profiles, revisited, *Interface Focus*, 2011, **1**(6), 857–870.
20 B. J. Prettejohn, M. J. Berryman and M. D. McDonnell, Methods for generating complex networks with selected structural properties for simulations: A review and tutorial for neuroscientists, *Front. Comput. Neurosci.*, 2011, **5**, 11.
21 Z. D. Kurtz, C. L. Müller, E. R. Miraldi, D. R. Littman, M. J. Blaser and R. A. Bonneau, Sparse and Compositionally Robust Inference of Microbial Ecological Networks, *PLoS Comput. Biol.*, 2015, **11**(5), e1004226.
22 S. Rogers and M. Girolami, A Bayesian regression approach to the inference of regulatory networks from gene expression data, *Bioinformatics*, 2005, **21**(14), 3131–3137.
23 H. Salgado, M. Peralta-Gil, S. Gama-Castro, A. Santos-Zavaleta, L. Muñiz Rascado, J. S. García-Sotelo, V. Weiss, H. Solano-Lira, I. Martínez-Flores, A. Medina-Rivera, G. Salgado-Osorio, S. Alquicira-Hernández, K. Alquicira-Hernández, A. López-Fuentes, L. Porrón-Sotelo, A. M. Huerta, C. Bonavides-Martínez, Y. I. Balderas-Martínez, L. Pannier, M. Olvera, A. Labastida, V. Jiménez-Jacinto, L. Vega-Alvarado, V. del Moral-Chávez, A. Hernández-Alvarez, E. Morett and J. Collado-Vides, RegulonDB v8.0: omics data sets, evolutionary conservation, regulatory phrases, cross-validated gold standards and more, *Nucleic Acids Res.*, 2013, **41**(D1), D203–D213.
24 T. Schaffter, D. Marbach and D. Floreano, GeneNetWeaver: *in silico* benchmark generation and performance profiling of network inference methods, *Bioinformatics*, 2011, **27**(16), 2263–2270.
25 M. C. Teixeira, P. T. Monteiro, J. F. Guerreiro, J. P. Goncalves, N. P. Mira, S. C. dos Santos, T. R. Cabrito, M. Palma, C. Costa, A. P. Francisco, S. C. Madeira, A. L. Oliveira, A. T. Freitas and I. Sá-Correia, The yeastract database: an upgraded information system for the analysis of gene and genomic transcription regulation in saccharomyces cerevisiae, *Nucleic Acids Res.*, 2013, D161–D166.
26 R. Tibshirani, Regression shrinkage and selection *via* the lasso, *J. R. Stat. Soc. Series B: Stat. Methodol.*, 1996, **58**(1), 267–288.
27 A. Tjärnberg, T. E. M. Nordling, M. Studham and E. L. L. Sonnhammer, Optimal sparsity criteria for network inference, *J. Comput. Biol.*, 2013, **20**(5), 398–408.
28 A. Tjärnberg, T. Nordling, M. Studham, S. Nelander and E. Sonnhammer, Avoiding pitfalls in l1-regularised inference of gene networks, *Mol. BioSyst.*, 2015, **11**(1), 287–296.
29 T. Van den Bulcke, K. Van Leemput, B. Naudts, P. van Remortel, H. Ma, A. Verschoren, B. De Moor and K. Marchal, Syntren: a generator of synthetic gene expression data for design and analysis of structure learning algorithms, *BMC Bioinf.*, 2006, **7**(1), 1–12.
30 D. J. Watts and S. H. Strogatz, Collective dynamics of 'small-world' networks, *Nature*, 1998, **393**(6684), 440–442.
31 Y. Yuan, G.-B. Stan, S. Warnick and J. Goncalves, Robust dynamical network structure reconstruction, *Automatica*, 2011, **47**(6), 1230–1235.
32 M. M. Zavlanos, a. A. Julius, S. P. Boyd and G. J. Pappas, Inferring stable genetic networks from steady-state data, *Automatica*, 2011, **47**(6), 1113–1122.

