nature methods

ANALYSIS

https://doi.org/10.1038/s41592-019-0690-6

Benchmarking algorithms for gene regulatory network inference from single-cell transcriptomic data

Aditya Pratapa[1], Amogh P. Jalihal[2], Jeffrey N. Law[2], Aditya Bharadwaj[1] and T. M. Murali[1*]

We present a systematic evaluation of state-of-the-art algorithms for inferring gene regulatory networks from single-cell transcriptional data. As the ground truth for assessing accuracy, we use synthetic networks with predictable trajectories, literature-curated Boolean models and diverse transcriptional regulatory networks. We develop a strategy to simulate single-cell transcriptional data from synthetic and Boolean networks that avoids pitfalls of previously used methods. Furthermore, we collect networks from multiple experimental single-cell RNA-seq datasets. We develop an evaluation framework called BEELINE. We find that the area under the precision-recall curve and early precision of the algorithms are moderate. The methods are better in recovering interactions in synthetic networks than Boolean models. The algorithms with the best early precision values for Boolean models also perform well on experimental datasets. Techniques that do not require pseudotime-ordered cells are generally more accurate. Based on these results, we present recommendations to end users. BEELINE will aid the development of gene regulatory network inference algorithms.

Single-cell RNA-sequencing technology has made it possible to trace cellular lineages during differentiation and to identify new cell types[1,2]. A central question that arises now is whether we can discover the gene regulatory networks (GRNs) that control cellular differentiation and drive transitions from one cell type to another. In such a GRN, each edge connects a transcription factor (TF) to a gene it regulates. Ideally, the edge is directed from the TF to the target gene, represents direct rather than indirect regulation and corresponds to activation or inhibition.

Single-cell expression data are especially promising for computing GRNs because, unlike bulk transcriptomic data, they do not obscure biological signals by averaging over all the cells in a sample. However, these data have features that pose significant difficulties; for example, substantial cellular heterogeneity[3], cell-to-cell variation in sequencing depth, the high sparsity caused by dropouts[4] and cell-cycle-related effects[5]. Despite these challenges, over a dozen methods have been developed or used to infer GRNs from single-cell data[6–19]. An experimentalist seeking to analyze a new dataset faces a daunting task in selecting an appropriate inference method since there are no widely accepted ground-truth datasets for assessing algorithm accuracy and the criteria for evaluation and comparison of methods are varied.

We have developed BEELINE, a comprehensive evaluation framework to assess the accuracy, robustness and efficiency of GRN inference techniques for single-cell gene expression data based on well-defined benchmark datasets (Fig. 1). BEELINE incorporates 12 diverse GRN inference algorithms. It provides an easy-to-use and uniform interface to each method in the form of a Docker image. BEELINE implements several measures for estimating and comparing the accuracy, stability and efficiency of these algorithms. Thus, BEELINE facilitates reproducible, rigorous and extensible evaluations of GRN inference methods for single-cell gene expression data.

Results

Overview of algorithms. We surveyed the literature and bioRxiv preprints for papers that either published a new GRN inference algorithm or used an existing approach. We ignored methods that did not assign weights or ranks to the interactions, required additional datasets or supervision, or sought to discover cell-type-specific networks. We selected 12 algorithms using these criteria (Methods).

We used BEELINE to evaluate these approaches on over 400 simulated datasets (across six synthetic networks and four curated Boolean models) and five experimental human or mouse single-cell RNA-seq datasets. Since eight algorithms require pseudotime-ordered cells, we used datasets (both simulated and real) that focus on cell differentiation and development, processes in which there is a meaningful temporal progression of cell states. We did not study GRNs relevant to other biological processes; for example, changes in disease states or differences among cell types.

Datasets from synthetic networks. Our motivations for using synthetic networks were two-fold. First, we wanted to use a known GRN that could serve as the ground truth. Second, we desired to create in silico single-cell gene expression datasets that were isolated from any limitations of pseudotime inference algorithms. Therefore, we started with six synthetic networks (Supplementary Fig. 1a and Supplementary Table 1). Simulating these networks should produce a variety of different trajectories seen in differentiating and developing cells[20]. Several recent studies on GRN inference[14,16,17,21,22] have used GeneNetWeaver[23] to create in silico single-cell gene expression datasets. However, when we simulated the six synthetic networks using GeneNetWeaver (Methods), we observed no discernible trajectories in the two-dimensional projections of these data (Supplementary Fig. 1d).

We therefore used our BoolODE approach (Methods) to simulate these networks. For each gene in a GRN, BoolODE requires a Boolean function that specifies how that gene's regulators combine to

[1]Department of Computer Science, Virginia Tech, Blacksburg, VA, USA. [2]Genetics, Bioinformatics, and Computational Biology Ph.D. Program, Virginia Tech, Blacksburg, VA, USA. *e-mail: murali@cs.vt.edu

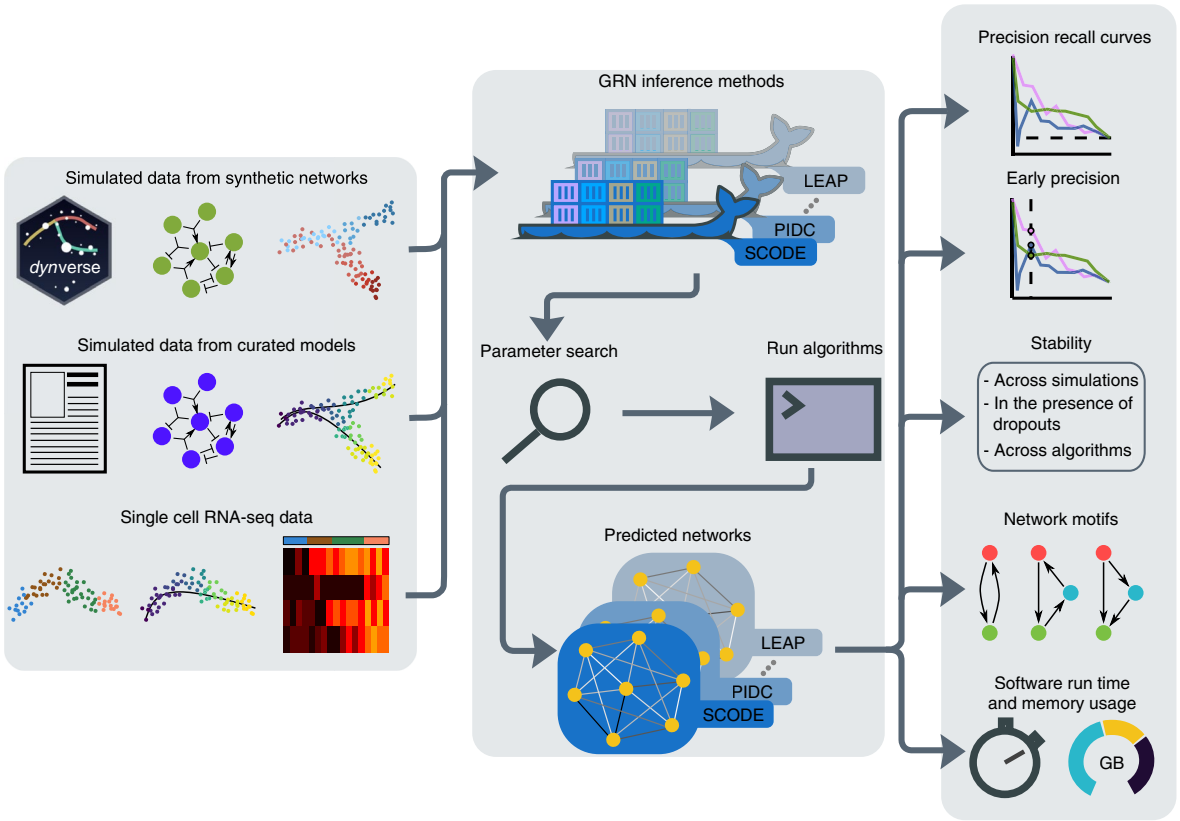

Fig. 1 | An overview of the BEELINE evaluation framework. We apply GRN inference algorithms to three types of data: datasets from synthetic networks, datasets from curated Boolean models from the literature and experimental single-cell transcriptional measurements. We process each dataset through a uniform pipeline: preprocessing, Docker containers for 12 GRN inference algorithms, parameter estimation, postprocessing and evaluation. We compare algorithms based on accuracy (AUPRC and early precision), stability of results (across simulations, in the presence of dropouts and across algorithms), analysis of network motifs and scalability.

control its state. We represent each Boolean function as a truth table, which we convert into a nonlinear ordinary differential equation (ODE). This approach provides a reliable method to capture the logical relationships among the regulators precisely in the components of the ODE. We add noise terms to make the equation stochastic[20,24].

For each network, we applied BoolODE by sampling ODE parameters ten times and generating 5,000 simulations per parameter set (Methods). We created five datasets per parameter set, one each with 100, 200, 500, 2,000 and 5,000 cells by sampling one cell per simulation, to obtain 50 different expression datasets. Analyzing the two-dimensional projections of these simulations reassured us that BoolODE was successful in correctly simulating the network models (Supplementary Fig. 1b,c and Supplementary Note 1.1).

Setting each network as the ground truth, we executed the 12 algorithms on every one of the 50 simulated datasets. For those GRN inference methods that required time information, we provided the simulation time at which we sampled each cell. For the bifurcating, bifurcating converging and trifurcating networks, we ran the algorithms that need time information on each trajectory individually and combined the outputs (Methods). Six algorithms required one or more parameters to be specified. We performed a parameter sweep to determine the values that gave the highest median area under the precision-recall curve (AUPRC) (Supplementary Note 1.2).

For each network–algorithm pair, Fig. 2 displays the median AUPRC ratio (the AUPRC divided by that of a random predictor). Supplementary Figs. 2 and 3 show the box plots of AUPRC and area under the receiver operating characteristic curve (AUROC) values. The methods performed best for the linear network: 10 out of 12 algorithms had a median AUPRC ratio greater than 2.0. Seven methods had a median AUPRC ratio greater than 5.0 for the linear long network. The cycle, bifurcating converging, bifurcating and trifurcating networks were progressively harder to infer, with no algorithm achieving an AUPRC ratio of two or more on the last network. Single-cell regularized inference using time-stamped expression profiles (SINCERITIES) obtained the highest median AUPRC ratio for four out of the six networks. Single-cell inference of networks using Granger ensembles (SINGE) had the highest median AUPRC ratio for cycle and partial information decomposition and context (PIDC) for trifurcating.

We examined the effect of the number of cells on AUPRC by comparing the values for 100, 200, 500 and 2,000 cells to those for 5,000 cells (Supplementary Note 1.3). As the number of cells increased from 100 to 500, the number of algorithms with significantly lower AUPRC values in comparison to 5,000 cells decreased from seven to four. The number of cells had no significant effect on five algorithms: gene network inference with ensemble of trees (GENIE3), GRN variational Bayesian expectation-maximization (GRNVBEM), lag-based expression association for pseudotime-series (LEAP), single-cell network synthesis (SCNS) and SCODE.

To examine the stability of the results, we considered the GRNs formed by the k edges with the highest ranks, with k set to the number of edges in each synthetic network, computed the Jaccard indices of all pairs of GRNs, and recorded the medians of these values (Fig. 2 and Methods). While SINCERITIES, SINGE and SCRIBE had the three highest median-of-median AUPRC ratios, the networks they predicted were relatively less stable (median Jaccard index between 0.28 and 0.35). PPCOR and PIDC, the other two methods in the top five, had higher median Jaccard indices of 0.62 each.

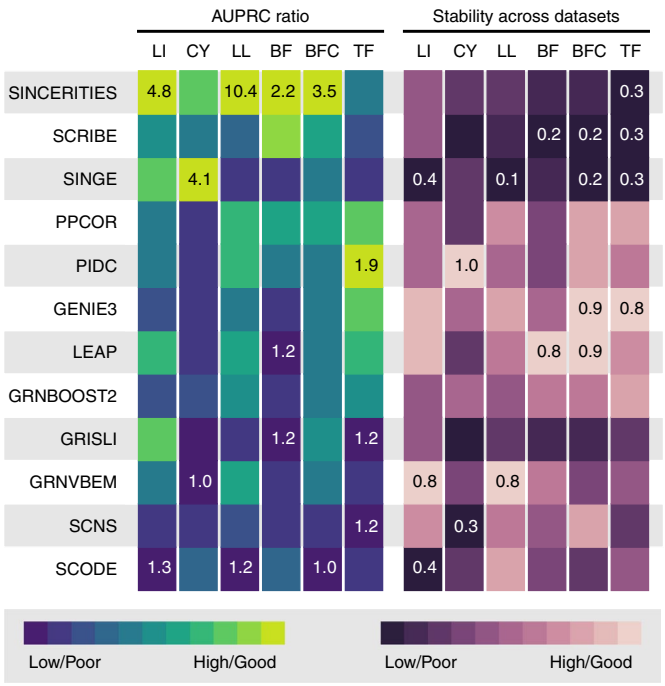

Fig. 2 | Summary of results for datasets from synthetic networks. The first six columns display the median AUPRC ratios for the 20 datasets with 2,000 and 5,000 cells, with algorithms (rows) ordered in decreasing order of the median of the per-network median AUPRC ratios. The next set of six columns displays the median stability scores across multiple datasets (Methods). For each network, the color in each cell is proportional to the corresponding value (scaled between 0 and 1). We display the highest and lowest values for each network inside the corresponding cells. Abbreviations: LI, linear; CY, cycle; LL, linear long; BF, bifurcating; BFC, bifurcating converging and TF, trifurcating.

We simulated the inferred GRNs to see if they had the same number of steady states as the ground-truth networks. Apart from the linear network, we found that more than 60% of the GRNs yielded more steady states than in the ground truth (Supplementary Note 1.4). For networks with multiple steady states, there was no clear benefit to computing a single GRN after combining all trajectories over merging the GRNs inferred for each trajectory (Supplementary Note 1.5).

Datasets from curated models. Dense subnetworks of large-scale GRNs that have been used to generate simulated datasets of single-cell gene expression[14,16,17,21,22] may not capture the complex regulation in any specific developmental process. To avoid this pitfall, we selected four published Boolean models: mammalian cortical area development (mCAD)[25], ventral spinal cord (VSC) development[26], hematopoietic stem cell (HSC) differentiation[27] and gonadal sex determination (GSD)[28] (Fig. 3 and Supplementary Table 2).

We confirmed that the BoolODE-simulated datasets for each Boolean model (1) captured the same number of steady states as in the model (Fig. 3b) and (2) matched the unique gene expression pattern that characterized each steady state of that model, as reported in the corresponding publication (Supplementary Note 2.1). Encouraged by these results, we used BoolODE to create ten different datasets with 2,000 cells for each model. For each dataset, we generated one version with a dropout rate of $q = 50$ and another with a rate of $q = 70$ (Methods)[14]. We computed pseudotimes using Slingshot[29] for each dataset and provided these values to the algorithms, to mimic a real analysis pipeline. We performed a parameter sweep and selected the values that gave the highest median AUPRC for each model (Supplementary Note 2.2). We then ran each of the 12 algorithms on each of the 120 datasets (30 per model).

Figure 4 summarizes our findings for the datasets without dropouts. Only four methods (gene regulation inference for single-cell with linear differential equations and velocity inference (GRISLI), SCODE, SINGE and SINCERITIES) had a median AUPRC ratio greater than one for the mCAD model. The reason may be the high density of the underlying network (Supplementary Table 2). For the VSC model, which only has inhibitory edges, three methods (PIDC, GRNBoost2 and GENIE3) had an AUPRC ratio greater than 2.5. These three methods also had an AUPRC ratio close to 2.0 for the HSC model. PPCOR, GRISLI and SCRIBE tied for the highest median AUPRC ratio of 1.4 for the GSD model. Overall, GENIE3, GRNBoost2 and PIDC had among the highest median AUPRC ratios for two out of the four models. SINCERITIES, SCRIBE and SINGE, which were the best algorithms according to the AUPRC ratios for the datasets from synthetic networks, had a close to random median AUPRC ratio for all four curated models. We address this trend in the Discussion.

Supplementary Figs. 4 and 5 show distributions of the AUPRC and AUROC values for all dropout rates. To study the effect of dropouts, for each algorithm, we compared the distributions of AUPRC scores across all Boolean models between $q = 0$ and $q = 50$ and between $q = 0$ and $q = 70$. Four and seven GRN inference methods had a statistically significant difference in AUPRC values for the 0–50 and the 0–70 comparison, respectively (Supplementary Note 2.3). The four algorithms that were unaffected by dropout rates (GRNVBEM, LEAP, SCRIBE and SINCERITIES) had worse-than-random AUPRC values on the mCAD and VSC datasets.

Next, we studied the early precision and early precision ratio (EPR) values of the top-k predictions (Methods) for each of the four models. In at least one of the four models, 11 algorithms had a median EPR less than or close to one; that is, similar to a random predictor (black squares in Fig. 4). In the 29 cases when the median EPR was at least one, it was 1.5 or larger only 16 times. The mCAD model had the smallest number of algorithms (two, GRISLI and SCODE) with median EPR larger than one. For datasets with dropouts, we did not see any clear trends in terms of smaller or larger early precision values compared to the dropout-free results (Supplementary Fig. 6).

We also investigated if the GRN inference methods were better at recovering activating edges or inhibitory edges. The mCAD model was again an outlier for EPR for activating and inhibitory edges, with only SCODE having slightly better-than-random scores for both. Overall, the GRN inference algorithms performed poorly when it comes to recovering the true edges within the top-k predictions.

We examined which algorithms produced similar reconstructions. For every model, the three best-performing methods (PIDC, GENIE3 and GRNBoost2) had similar outputs (Supplementary Note 2.4). In addition, LEAP and PPCOR were similar to the first three methods for the mCAD and GSD models. Pairwise similarities were poor for the other algorithms.

The GRNs formed by the top-k edges contained a higher than expected number of feedforward loops and lower than expected feedback loops and mutual interactions (Supplementary Note 2.5). Further, a very large fraction of false positives in the top-k edges corresponded to paths of length two in the ground-truth networks (Supplementary Note 2.6). This tendency to predict 'indirect' interactions could be the reason for the low EPR values.

Experimental single-cell RNA-seq datasets. We selected five experimental single-cell RNA-seq datasets, two in human and three in mouse cells, comprising seven cell types (Methods and Supplementary Table 3). We collected three different types of ground-truth network: cell-type-specific ChIP–seq, nonspecific

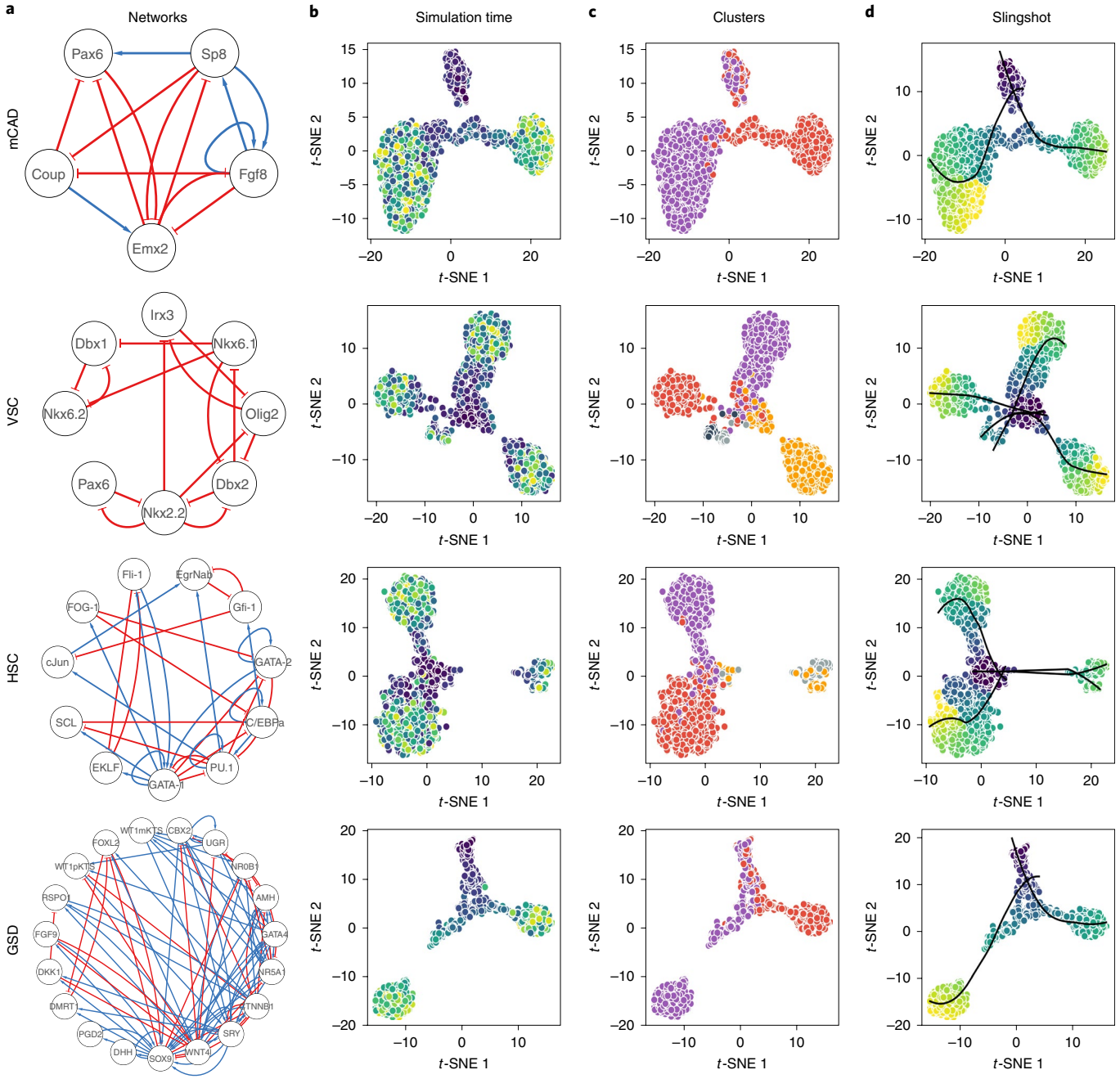

Fig. 3 | Visualization of *t*-SNE projections of simulations reveals trajectories leading to steady states that correspond to those of the curated models. Each row in the figure corresponds to a model, indicated on the left: mCAD, VSC, HSC differentiation and GSD determination. **a**, Network diagrams of the models. **b**, *t*-SNE visualizations of 2,000 cells sampled from the BoolODE output. The color of each point indicates the corresponding simulation time. **c**, Each color corresponds to a different subset of cells obtained by using *k*-means clustering of simulations, with *k* set to the number of steady states reported in the relevant publication (two for mCAD, five for VSC, four for HSC and two for GSD). **d**, Pseudotimes and principal curves (black) computed by Slingshot showing correspondence with simulation times in **b** and clusters in **c**, respectively. Colors of simulation time and pseudotime: blue for early, green for intermediate and yellow for later.

ChIP–seq and functional interaction networks (Methods and Supplementary Table 4).

To measure the running time of the algorithms, we selected three cell types, namely, human mature hepatocytes (hHEP), human embryonic stem cells (hESCs) and erythroid-lineage mouse hematopoietic stem cells (mHSC-E), which contained different numbers of cells. We executed the algorithms on multiple subsets of highly varying genes in each dataset. For 5,000 genes, the running times ranged from minutes (PPCOR, LEAP, SINCERITIES and SCODE) to hours (GRNBOOST2) to nearly a day (GENIE3 and PIDC). GRNVBEM, SCRIBE and SINGE were the slowest methods taking a few hours to nearly a day for 1,000 genes (Supplementary Fig. 7). There was little to no variation in the running times of any method with increasing number of cells. Since the implementations of GENIE3, GRNBoost2 and SINGE are multithreaded, their wall-clock times can be lowered by a factor proportional to the number of threads available. Most of the algorithms did not require more than 4 GB of RAM for up to 2,000 genes.

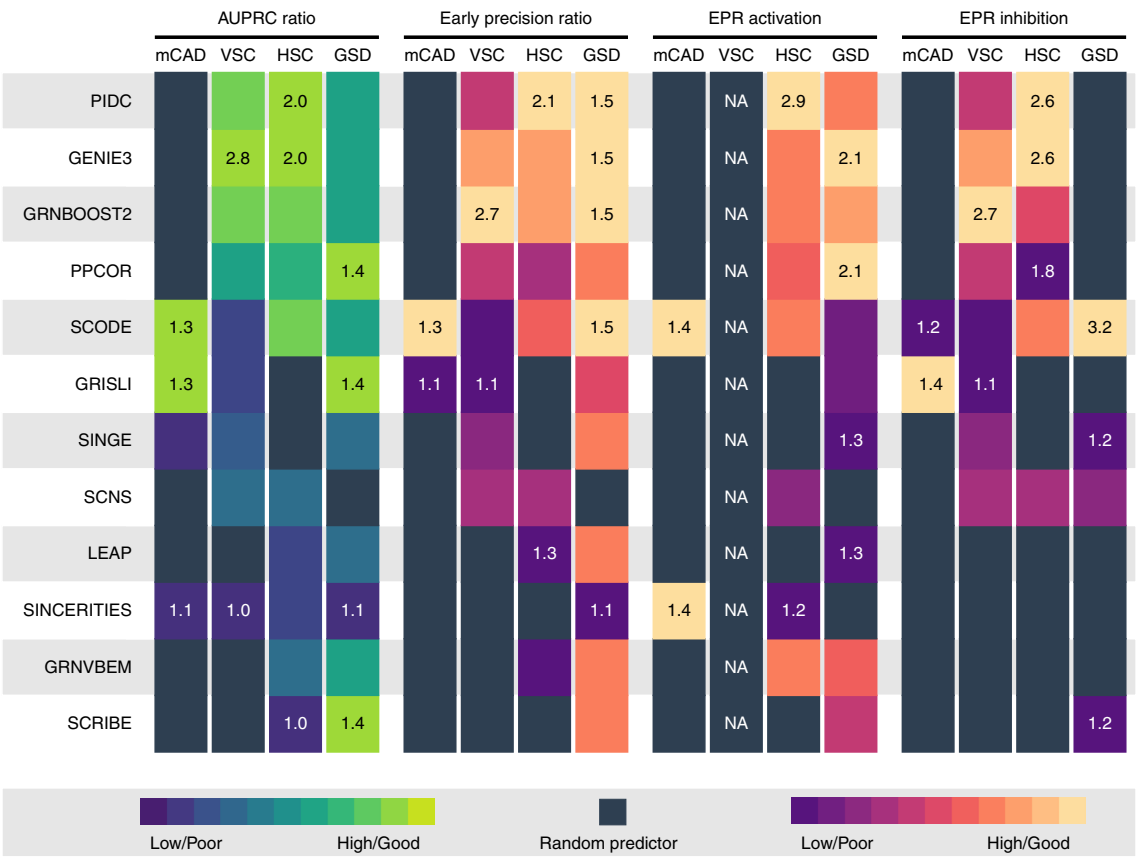

Fig. 4 | Summary of results for ten datasets without dropouts from curated models. Rows correspond to algorithms ordered by decreasing median of the per-model median AUPRC ratios. The four sets of four columns each display the median AUPRC ratios, median EPR, median EPR for activating edges and median EPR for inhibitory edges. For each model, the color in each cell is proportional to the corresponding value (scaled between 0 and 1, ignoring values that are less than that of a random predictor, shown as black squares). We display the highest and lowest values for each model inside the corresponding cells.

For further analysis, we selected the five algorithms with the highest median AUPRC in each of the earlier two datasets: SINCERITIES, SCRIBE, SINGE, PPCOR and PIDC for synthetic networks and PIDC, GENIE3, GRNBoost2, PPCOR and SCODE for Boolean models; PIDC and PPCOR were in both sets. We did not retain SINGE and SCRIBE because of the time taken for parameter search.

For each RNA-seq dataset, we created four subsets of genes containing all the significantly varying TFs and either (1) the 500 or (2) 1,000 most-varying genes and only the (3) 500 or (4) 1,000 most-varying genes. We intersected each ground-truth network with each set of genes.

We compared the algorithms based on the EPR, reasoning that predicted interactions of higher confidence will be more interesting to experimentalists. We used the top-k networks, setting k equal to the number of edges in the corresponding reduced ground-truth dataset. We performed parameter search to optimize the EPR (Supplementary Note 3.1).

In general, the algorithms achieved lower EPR values in networks with higher densities; that is, the cell-type-specific ChIP–seq networks (Fig. 5 and Supplementary Fig. 8). While the STRING and nonspecific ChIP–seq networks had similar densities, the GRN methods typically achieved a higher EPR for the former. STRING networks contain both physical and functional (indirect) interactions. The higher EPR ratios obtained for STRING networks compared to ChIP–seq networks of similar density suggested that a substantial fraction of the edges in the inferred GRNs were indirect, reinforcing our findings for Boolean models. The densities of the cell-type-specific ChIP–seq networks varied considerably (from 0.08 to 0.58). The EPR of most of the methods on these networks was close to 1, that is, similar to a random predictor, with no EPR value exceeding 1.5.

GENIE3, PIDC and GRNBoost2 were the three methods with the highest EPR values for experimental datasets (Fig. 5 and Supplementary Fig. 8), just as they were for curated models (Fig. 4). These methods also performed equally well for AUPRC ratios (Supplementary Figs. 9 and 10). When we computed modules in the GRNs output by these methods, we found that they had a good concordance with clusters determined directly from the gene expression data (Supplementary Note 3.2). PPCOR, which was consistently in the top-four methods for both synthetic networks and curated models, had only a slightly better-than-random EPR across all experimental datasets. PPCOR's AUPRC decreased moderately as we increased dropouts in datasets from curated models (Supplementary Fig. 4). We speculate that dropouts in experimental single-cell RNA-seq datasets had a severe effect on PPCOR.

To examine the effect of the number of genes, as well as including all significantly varying TFs, we evaluated the three top performing methods, PIDC, GENIE3 and GRNBoost2, on the nonspecific ChIP–seq and STRING networks. We found that the median EPR had a statistically significant improvement when we included all significantly varying TFs in the analysis (Supplementary Note 3.3). However, the addition of highly varying genes (500 versus 1,000) did not lead to a significant improvement in EPR values. The AUPRC ratio did not vary with the number of genes.

Almost every algorithm produced nearly identical outputs when we ran it on the same dataset multiple times (Supplementary Note 3.4). However, for the same dataset, the results varied

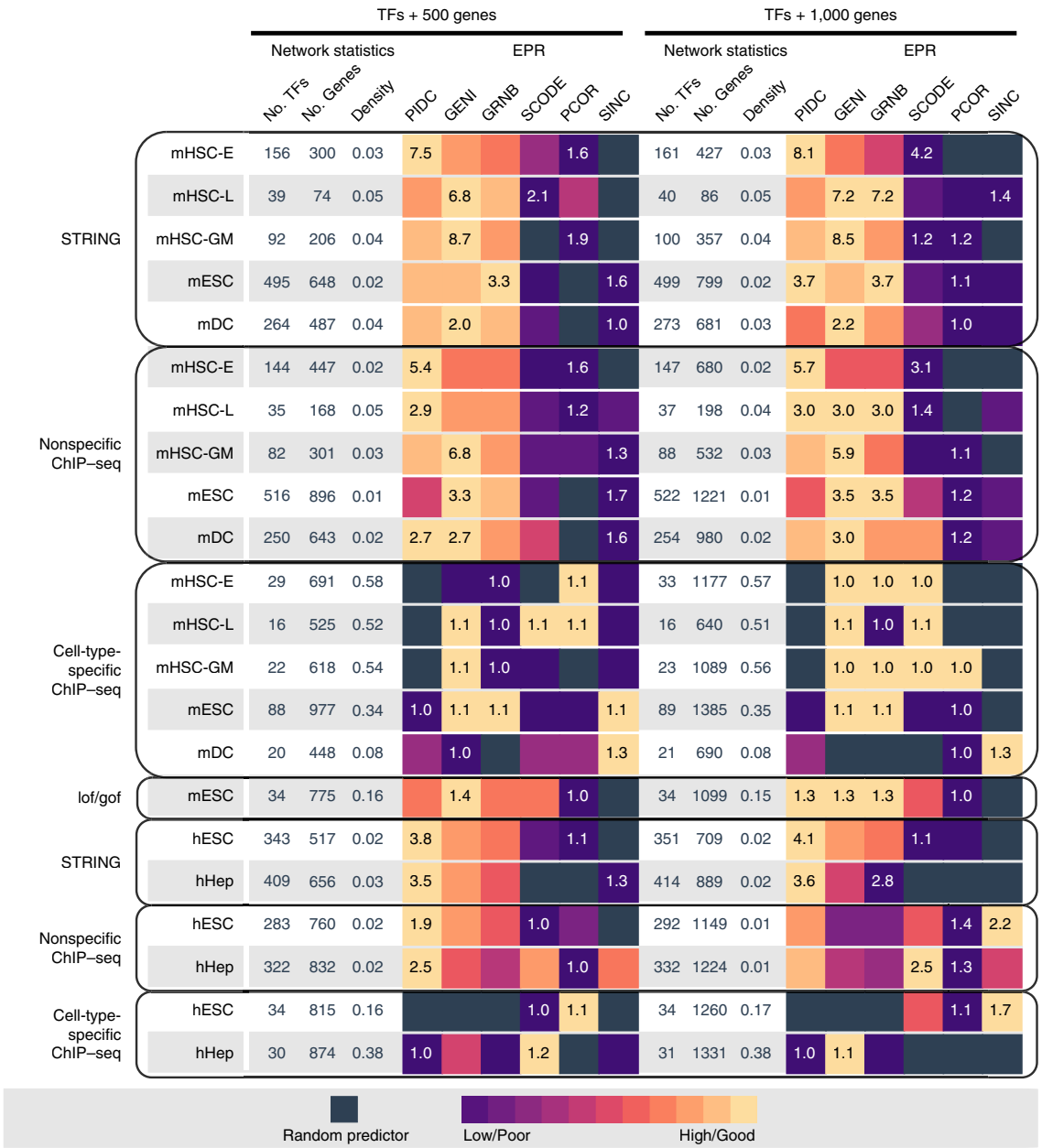

Fig. 5 | Summary of EPR results for experimental single-cell RNA-seq datasets. The left half of the figure (TFs + 500 genes) shows results for datasets composed of all significantly varying TFs and the 500 most-varying genes. Each row corresponds to one scRNA-seq dataset. The first three columns report network statistics. The next six columns report EPR values. The right half (TFs + 1,000 genes) shows results for all significantly varying TFs and the 1,000 most-varying genes. In both sections, algorithms are sorted by median EPR across the datasets (rows) for the TFs + 500 gene set. For each dataset, the color in each cell is proportional to the corresponding value scaled between 0 and 1 (ignoring values that are less than that of a random predictor, which are shown as black squares). We display the highest and lowest values for each dataset inside the corresponding cells. Abbreviations: GENI, GENIE3; GRNB, GRNBoost2; lof/gof, loss-of-function/gain-of-function; PCOR, PPCOR and SINC, SINCERITIES.

substantially from one algorithm to another (Supplementary Note 3.5), in contrast to curated models where the top performing methods yielded similar results (Supplementary Note 2.4). Moreover, ensembles of the algorithm outputs did not perform systematically better than the best method for each dataset (Supplementary Note 3.5). This result stands in contrast to the success of ensembles in inferring GRNs from bulk transcriptional data[30].

Discussion

We have presented BEELINE, a framework for benchmarking algorithms that infer GRNs from single-cell gene expression data. Figure 6 summarizes the properties of the algorithms and the insights from this study. Despite considerable variation in algorithm performance across the different types of data, we noted a few trends. The synthetic networks were easier to recover than the curated models. The reason may be that the synthetic networks have simple and well-defined trajectories. For curated models, each of which has multiple trajectories, we found that methods that do not require pseudotime information (GENIE3, GRNBoost2 and PIDC) performed the best. Methods that performed well for Boolean models also inferred GRNs of good accuracy for experimental datasets. Nevertheless, the overall performance of these approaches was less than ideal.

A surprising trend was that the best-performing algorithms for datasets from synthetic networks (SINCERITIES, SCRIBE and

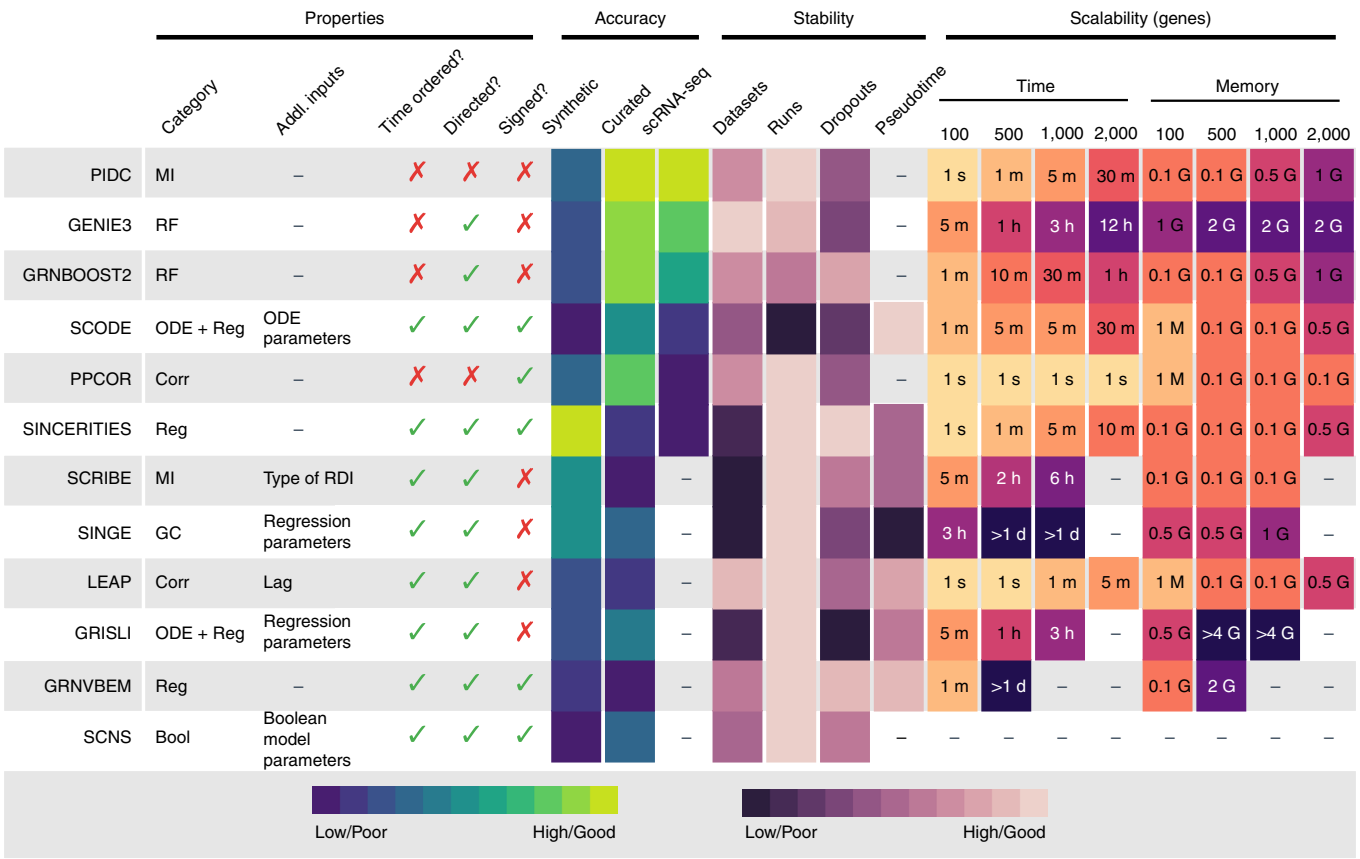

Fig. 6 | Summary of properties of GRN inference algorithms and results obtained from BEELINE. Each row corresponds to one of the algorithms included in our evaluation. The first six columns display algorithm methodology, required additional inputs, whether the method needs cells to be time-ordered, and whether the inferred edges are directed and signed. The next three columns summarize the results in Figs. 2, 4 and 5. The next four columns present results for different types of stability. The final set of columns contain the running time and memory usage. For the 'Pseudotime' column, we only considered the seven methods that required these values, ignoring SCNS due to its long execution time. See Methods for details on how we generated this figure. Abbreviations: MI, mutual information; RF, random forest; Corr, Correlation; Reg, regression; GC, Granger causality and Bool, Boolean model.

SINGE, Fig. 2) had poor results on datasets from curated models (Fig. 4); SINCERITIES had close to or worse-than-random EPRs on experimental datasets as well (Fig. 5). When we inferred GRNs for synthetic networks using shuffled pseudotimes (Methods), we observed a general decrease in performance with an increase in the size of the window over which we shuffled the pseudotime values, with the effect being most pronounced for SINCERITIES, SCRIBE and SINGE ('Pseudotime' in Fig. 6 and Supplementary Fig. 11). This analysis suggests that these algorithms may be sensitive to accurate pseudotime imputation.

Based on these observations, we make specific recommendations for users seeking to apply these methods.

i. PIDC, GENIE3 and GRNBoost2 are the methods of choice, since they were leading and consistent performers for curated models and experimental datasets in terms of accuracy.
ii. GENIE3 and PIDC also had better stability across multiple runs, whereas GRNBoost2 was less sensitive to the presence of dropouts. Since these methods do not require pseudotime-ordered cells, they are immune to any errors in pseudotime computation. As the quality of pseudotime inference improves, SINCERITIES may become a good choice, especially since it is stable across multiple runs and in the presence of dropouts.
iii. Since GRNBoost2 and GENIE3 have multithreaded implementations[8], they are as efficient as PIDC for 2,000 genes or fewer.
iv. Our results suggest that adding more highly varying genes (1,000 rather than 500) and/or considering all significantly varying TFs contribute to significant improvements in the EPR of the best-performing algorithms. However, there is no effect on AUPRC. A recent best-practice guide[31] has suggested using 1,000–5,000 highly variable genes for single-cell RNA-seq analyses such as clustering and differential expression. However, GRN algorithms may require significant computation time beyond 1,000 genes. Hence, the strategy for selection of genes merits further analysis.

Inference of GRNs has been an active area of research for more than 20 years. Our evaluation shows that GRN inference remains a challenging problem. One possible reason is that single-cell RNA-seq techniques may not still provide sufficient resolution and variation in expression for the reliable inference of GRNs despite rapid advances both in the number of cells that can be measured and the depth of coverage[32]. There may also be inherent shortcomings to the assumption that statistical relationships between expression patterns correspond to regulatory interactions. In this context, we observed that false positive edges form feedforward loops when added to ground-truth networks (Supplementary Note 2.5). To avoid such indirect interactions, it may be important to integrate additional types of data such as known TF binding sites or ChIP–seq measurements[15]. Finally, a target gene's expression level may change even if the regulating TF does not vary in abundance. Recent approaches that interrogate single cells along multiple modalities[33,34] may be important for the next generation of GRN inference algorithms.

BoolODE was a critical component of our analysis. We developed BoolODE after noting that reported AUROC or precision at early recall values for GRN algorithms were often close to that of a random predictor[6,10,11,13,15], as we also observed. Therefore, we reasoned that it would be valuable to the community to benchmark GRN algorithms by applying them to accurate simulations of Boolean models with predictable trajectories. BoolODE is successful at this task and promises to be useful as an independent tool.

As single-cell experiments become more complex, cellular trajectories will also be more intricate, perhaps involving multiple stages of bifurcation and/or cycling. A key challenge that lies ahead is accurately computing the underlying GRNs. We hope that scientists will use BEELINE in conjunction with BoolODE as they develop new approaches for GRN inference.

Online content

Any methods, additional references, Nature Research reporting summaries, source data, extended data, supplementary information, acknowledgements, peer review information; details of author contributions and competing interests; and statements of data and code availability are available at https://doi.org/10.1038/s41592-019-0690-6.

Received: 4 June 2019; Accepted: 22 November 2019;
Published online: 6 January 2020

References

1. Villani, A.-C. et al. Single-cell RNA-seq reveals new types of human blood dendritic cells, monocytes, and progenitors. *Science* **356**, eaah4573 (2017).
2. Rozenblatt-Rosen, O., Stubbington, M. J. T., Regev, A. & Teichmann, S. A. The human cell atlas: from vision to reality. *Nature* **550**, 451–453 (2017).
3. Wagner, A., Regev, A. & Yosef, N. Revealing the vectors of cellular identity with single-cell genomics. *Nat. Biotechnol.* **34**, 1145–1160 (2016).
4. Kharchenko, P. V., Silberstein, L. & Scadden, D. T. Bayesian approach to single-cell differential expression analysis. *Nat. Methods* **11**, 740–742 (2014).
5. Buettner, F. et al. Computational analysis of cell-to-cell heterogeneity in single-cell RNA-sequencing data reveals hidden subpopulations of cells. *Nat. Biotechnol.* **33**, 155–160 (2015).
6. Huynh-Thu, V. A., Irrthum, A., Wehenkel, L. & Geurts, P. Inferring regulatory networks from expression data using tree-based methods. *PLoS One* **5**, e12776 (2010).
7. Kim, S. ppcor: An R package for a fast calculation to semi-partial correlation coefficients. *Commun. Stat. Appl. Methods* **22**, 665–674 (2015).
8. Moerman, T. et al. GRNBoost2 and Arboreto: efficient and scalable inference of gene regulatory networks. *Bioinformatics* **35**, 2159–2161 (2018).
9. Aubin-Frankowski, P.-C. & Vert, J.-P. Gene regulation inference from single-cell RNA-seq data with linear differential equations and velocity inference. Preprint at *bioRxiv* https://doi.org/10.1101/464479 (2018).
10. Deshpande, A., Chu, L.-F., Stewart, R. & Gitter, A. Network inference with Granger causality ensembles on single-cell transcriptomic data. Preprint at *bioRxiv* https://doi.org/10.1101/534834 (2019).
11. Huynh-Thu, V. A. & Sanguinetti, G. Combining tree-based and dynamical systems for the inference of gene regulatory networks. *Bioinformatics* **31**, 1614–1622 (2015).
12. Specht, A. T. & Li, J. LEAP: constructing gene co-expression networks for single-cell RNA-sequencing data using pseudotime ordering. *Bioinformatics* **33**, 764–766 (2017).
13. Matsumoto, H. et al. SCODE: an efficient regulatory network inference algorithm from single-cell RNA-Seq during differentiation. *Bioinformatics* **33**, 2314–2321 (2017).
14. Chan, T. E., Stumpf, M. P. H. & Babtie, A. C. Gene regulatory network inference from single-cell data using multivariate information measures. *Cell Syst.* **5**, 251–267 (2017).
15. Aibar, S. et al. SCENIC: single-cell regulatory network inference and clustering. *Nat. Methods* **14**, 1083–1086 (2017).
16. Papili Gao, N., Ud-Dean, S. M. M., Gandrillon, O. & Gunawan, R. SINCERITIES: inferring gene regulatory networks from time-stamped single cell transcriptional expression profiles. *Bioinformatics* **34**, 258–266 (2018).
17. Sanchez-Castillo, M., Blanco, D., Tienda-Luna, I. M., Carrion, M. C. & Huang, Y. A Bayesian framework for the inference of gene regulatory networks from time and pseudo-time series data. *Bioinformatics* **34**, 964–970 (2018).
18. Woodhouse, S., Piterman, N., Wintersteiger, C. M., Göttgens, B. & Fisher, J. SCNS: a graphical tool for reconstructing executable regulatory networks from single-cell genomic data. *BMC Syst. Biol.* **12**, 59 (2018).
19. Qiu, X. et al. Towards inferring causal gene regulatory networks from single cell expression measurements. Preprint at *bioRxiv* https://doi.org/10.1101/426981 (2018).
20. Saelens, W., Cannoodt, R., Todorov, H. & Saeys, Y. A comparison of single-cell trajectory inference methods. *Nat. Biotechnol.* **37**, 547–554 (2019).
21. Lim, C. Y. et al. BTR: training asynchronous Boolean models using single-cell expression data. *BMC Bioinforma.* **17**, 355 (2016).
22. Chen, S. & Mar, J. C. Evaluating methods of inferring gene regulatory networks highlights their lack of performance for single cell gene expression data. *BMC Bioinformatics* **19**, 232 (2018).
23. Schaffter, T., Marbach, D. & Floreano, D. GeneNetWeaver: in silico benchmark generation and performance profiling of network inference methods. *Bioinformatics* **27**, 2263–2270 (2011).
24. Ocone, A., Haghverdi, L., Mueller, N. S. & Theis, F. J. Reconstructing gene regulatory dynamics from high-dimensional single-cell snapshot data. *Bioinformatics* **31**, 89–96 (2015).
25. Giacomantonio, C. E. & Goodhill, G. J. A Boolean model of the gene regulatory network underlying mammalian cortical area development. *PLoS Comput. Biol.* **6**, e1000936 (2010).
26. Lovrics, A. et al. Boolean modelling reveals new regulatory connections between transcription factors orchestrating the development of the ventral spinal cord. *PLoS One* **9**, e111430 (2014).
27. Krumsiek, J., Marr, C., Schroeder, T. & Theis, F. J. Hierarchical differentiation of myeloid progenitors is encoded in the transcription tactor network. *PLoS One* **6**, e22649 (2011).
28. Ríos, O. et al. A Boolean network model of human gonadal sex determination. *Theor. Biol. Med. Model.* **12**, 26 (2015).
29. Street, K. et al. Slingshot: cell lineage and pseudotime inference for single-cell transcriptomics. *BMC Genomics* **19**, 477 (2018).
30. Marbach, D. et al. Wisdom of crowds for robust gene network inference. *Nat. Methods* **9**, 796–804 (2012).
31. Luecken, M. D. & Theis, F. J. Current best practices in single-cell RNA-seq analysis: a tutorial. *Mol. Syst. Biol.* **15**, 8746 (2019).
32. Svensson, V., Vento-Tormo, R. & Teichmann, S. A. Exponential scaling of single-cell RNA-seq in the past decade. *Nat. Protoc.* **13**, 599–604 (2018).
33. Stuart, T. et al. Comprehensive integration of single-cell data. *Cell* **177**, 1888–1902.e21 (2019).
34. Mimitou, E. P. et al. Multiplexed detection of proteins, transcriptomes, clonotypes and CRISPR perturbations in single cells. *Nat. Methods* **16**, 409–412 (2019).

Methods

Regulatory network inference algorithms. We briefly describe each algorithm we have included in this evaluation. We have ordered the methods chronologically by year and month of publication. Every software package had an open source license, other than GRNVBEM and GRISLI, which did not have any license.

1. GENIE3 (ref. [6]). Developed originally for bulk transcriptional data, GENIE3 computes the regulatory network for each gene independently. It uses tree-based ensemble methods such as random forests to predict the expression profile of each target gene from profiles of all the other genes. The weight of an interaction comes from the importance of an input gene in the predictor for a target gene's expression pattern. Aggregating these weighted interactions over all the genes yields the regulatory network. This method was the top performer in the DREAM4 in silico network challenge (multifactorial subchallenge).
2. PPCOR[7]. This R package computes the partial and semi-partial correlation coefficients for every pair of variables (genes, in our case) with respect to all the other variables. It also computes a *P* value for each correlation. We use this package to compute the partial correlation coefficients. Since these values are symmetric, this method yields an undirected regulatory network. We use the sign of the correlation, which is bounded between −1 and 1, to signify whether an interaction is inhibitory (negative) or activating (positive).
3. LEAP[12]. Starting with pseudotime-ordered data, LEAP calculates the Pearson's correlation of normalized mapped-read counts over temporal windows of a fixed size with different lags. The score recorded for a pair of genes is the maximum Pearson's correlation over all the values of lag that the method considers. The software includes a permutation test to estimate false discovery rates. Since the correlation computed is not symmetric, this method can output directed networks.
4. SCODE[13]. This method uses linear ODEs to represent how a regulatory network results in observed gene expression dynamics. SCODE relies on a specific relational expression that can be estimated efficiently using linear regression. In combination with dimension reduction, this approach leads to a considerable reduction in the time complexity of the algorithm.
5. PIDC[14]. This method uses concepts from information theory. For every pair of genes *x* and *y*, given a third gene *z*, the method partitions the pairwise mutual information between *x* and *y* into a redundant and a unique component. It computes the ratio between the unique component and the mutual information. The sum of this ratio over all other genes *z* is the proportional unique contribution between *x* and *y*. The method then uses per-gene thresholds to identify the most important interactions for each gene. The resulting network is undirected since the proportional unique contribution is symmetric.
6. SINCERITIES[16]. Given time-stamped transcriptional data, this method computes temporal changes in each gene's expression through the distance of the marginal distributions between two consecutive time points using the Kolmogorov–Smirnov statistic. To infer regulatory connections between TFs and target genes, the approach uses Granger causality; that is, it uses the changes in the gene expression of TFs in one time window to predict how the expression distributions of target genes shift in the next time window. The authors formulate inference as a ridge regression problem. They infer the signs of the edges using partial correlation analyses.
7. SCNS[18]. This method takes single-cell gene expression data taken over a time course as input and computes logical rules (Boolean formulas) that drive the progression and transformation from initial cell states to later cell states. By design, the resulting logical model facilitates the prediction of the effect of gene perturbations (for example, knockout or overexpression) on specific lineages.
8. GRNVBEM[17]. This approach infers a Bayesian network representing the gene regulatory interactions. It uses a first-order autoregressive model to estimate the fold change of a gene at a specific time as a linear combination of the expression of the gene's regulators in the Bayesian network at the previous time point. It infers the GRN within a variational Bayesian framework. This method can associate signs with its directed edges.
9. SCRIBE[19]. Similar to PIDC, this approach uses ideas from information theory. The relevant concept here is conditioned restricted directed information, which measures the mutual information between the past state (expression values) of a regulator and the current state of a target gene conditioned on the state of the target at the previous time point. To obtain efficiency for large datasets, the authors use an unconditioned version called RDI, followed by the context likelihood of relatedness algorithm[35] to remove edges that correspond to indirect effects. We used this strategy for experimental single-cell RNA-seq datasets.
10. GRNBoost2 (ref. [8]). GRNBoost2 is a fast alternative for GENIE3, especially suited for datasets with tens of thousands of observations. Like GENIE3, GRNBoost2 trains a regression model to select the most important regulators for each gene in the dataset. GRNBoost2 achieves its efficiency by using stochastic gradient boosting machine regression with early stopping regularization to infer the network.
11. GRISLI[9]. Like SCODE, this approach uses a linear ODE-based formalism. GRISLI estimates the parameters of the model using different ideas. Taking either the experimental time of the cells or estimated pseudotime as input, it first estimates the velocity of each cell, that is, how each gene's expression value changes as each cell undergoes a dynamical process[36]. It then computes the structure of the underlying GRN by solving a sparse regression problem that relates the gene expression and velocity profiles of each cell.
12. SINGE[10]. The authors observe that while many gene inference algorithms start by computing a pseudotime value for each cell, the distribution of cells along the underlying dynamical process may not be uniform. To address this limitation, SINGE uses kernel-based Granger causality regression to alleviate irregularities in pseudotime values. SINGE performs multiple regressions, one for each set of input parameters, and aggregates the resulting predictions using a modified Borda method.

In summary, most algorithms developed explicitly for single-cell transcriptomic data required the cells to be ordered by pseudotime in the input, with PIDC[14] being an exception. These methods ideally require datasets corresponding to linear trajectories; some techniques recommend that data with branched trajectories be split into multiple linear ones before input[10,19]. In contrast, methods that had originally been developed for bulk transcriptional data did not impose this requirement[6,7]. Almost all the methods we included output directed networks with exceptions being PPCOR and PIDC[7,14]. Only five methods output signed networks, that is, they indicated whether each interaction was activating or inhibitory[7,13,16–18]. A number of methods inferred each pairwise interaction independently of the others, sometimes conditioned on the other genes[7,12,14,19]. Several other methods computed all the regulators of a gene simultaneously but solved the problem independently for each gene[6,8–10,13,16,18].

Other methods. We next discuss other papers on this topic and our rationale for not including them in the comparison. We did not consider a method if it was supervised[37] or used additional information; for example, a database of TFs and their targets[15] or a lineage tree[38]. We did not include methods that output a single GRN without any edge weights[21,24], since any such approach would yield just a single point on a precision-recall curve. Other than SCNS, we did not consider methods that output Boolean networks[21,38,39].

BoolODE: converting Boolean models to ODEs. GeneNetWeaver[23,40,41] is a widely used method to simulate bulk transcriptomic data from GRNs. GeneNetWeaver has also been applied in single-cell analysis[14,16,17,21,22] but has limitations, as we have demonstrated (Supplementary Fig. 1). To deal with this challenge, we develop a method called BoolODE that systematically and accurately converts a Boolean model into a system of stochastic differential equations (SDEs).

We start this section by giving an overview of GeneNetWeaver. Next, we describe our BoolODE framework that we have developed and highlight its differences with GeneNetWeaver. We end this section by summarizing BoolODE and the reasons we prefer it over GeneNetWeaver.

GeneNetWeaver. This method starts with a network of regulatory interactions among TFs and their targets. It computes a connected, dense subnetwork around a randomly selected seed node and converts this network into a system of differential equations. To express this network in the form of ODEs, it assigns each node *i* in the network a 'gene' variable (x_i) representing the level of messenger RNA expression and a 'protein' variable (p_i) representing the amount of TF produced by protein translation as follows:

$$\frac{\mathrm{d}[x_i]}{\mathrm{dt}} = mf(R_i) - l_x[x_i]$$

$$\frac{\mathrm{d}[p_i]}{\mathrm{dt}} = r[x_i] - l_p[p_i]$$

where m is the mRNA transcription rate, l_x is the mRNA degradation rate, r is the protein translation rate and l_p is the protein degradation rate. In the first equation, R_i denotes the set of regulators of node i. The nonlinear input function $f(R_i)$ captures all the regulatory interactions controlling the expression of node i (ref. [41]); we specify it below.

If there are N regulators for a given gene, there are 2^N possible configurations of how the regulators can bind to the gene's promoter. Considering cooperative effects of regulator binding, the probability (Pr) of each configuration $S \in 2^{R_i}$, the powerset of R_i, is given by the following equation[42]:

$$\Pr(S) = \frac{\prod_{q\in S}([q]/k)^n}{1 + \sum_T \prod_{q\in T}([q]/k)^n}$$

where k and n are the Hill threshold and Hill coefficient, respectively. Here, we use q to denote a single regulator in the configuration S and the product in the numerator ranges over all regulators that are present (bound) in S. In the summation in the denominator of this equation, the set T ranges over all members of the powerset 2^{R_i} other than the empty set. GeneNetWeaver further introduces a randomly sampled parameter $\alpha_s \in [0,1]$ to specify the efficiency of transcription

activation by a specific configuration S of bound regulators. Thus, the function $f(R_i)$ thus takes the following form:

$$f(R_i) = \sum_{S \in 2^{R_i}} \alpha_S \Pr(S)$$

Next, GeneNetWeaver adds a noise term to each equation to mimic stochastic effects in gene expression[23]. In addition, to create variations among individual experimental samples, GeneNetWeaver recommends adopting a multifactorial perturbation[23] that increases or decreases the basal activation of each gene in the GRN simultaneously by a small, randomly selected value. GeneNetWeaver removes this perturbation after the first half of the simulation. Simulating this system of SDEs generates the requisite gene expression data.

BoolODE uses Boolean models to create simulated datasets. To generate simulated time course data for our analysis, we used the GeneNetWeaver framework with one critical difference and one minor variation. The form of the equations used by BoolODE is identical to that of GeneNetWeaver. The critical difference is that we do not sample the α_S parameters in the above equation randomly; that is, we do not combine the regulators of each gene using a random logic function. Instead, we use the fact that in both the artificial networks and the literature-curated models, we know the Boolean function that specifies how the states of the regulators control the state of the target genes. Moreover, we can express any arbitrary Boolean function in the form of a truth table relating the input states (that is, activities of TFs) to the output state (the activity of target gene). For a gene with N regulators in its Boolean function, we explore all 2^N combinations of TF states and evaluate the transcriptional activity of each specific regulator configuration. Since the value of the Boolean function is the logical disjunction ('or') of all these values, we set the α value to one (respectively, zero) for every configuration that evaluates to 'on' (respectively, 'off'). The following example illustrates our approach. Consider a gene X with two activators (P and Q) and one inhibitor (R), represented by the following rule:

$$X = (P \vee Q) \wedge \neg(R)$$

The truth table corresponding to this rule along with the α parameters is shown in Supplementary Table 5. Therefore, the ODE governing the time dynamics of gene X is

$$\frac{d[X]}{dt} = m\left(\frac{\alpha_0 + \alpha_P[P] + \alpha_Q[Q] + \alpha_R[R] + \alpha_{PQ}[P][Q] + \alpha_{PR}[P][R] + \alpha_{QR}[Q][R] + \alpha_{PQR}[P][Q][R]}{1+[P]+[Q]+[R]+[P][Q]+[P][R]+[Q][R]+[P][Q][R]}\right)$$

$$-l_x[X] = m\left(\frac{[P]+[Q]+[P][Q]}{1+[P]+[Q]+[R]+[P][Q]+[P][R]+[Q][R]+[P][Q][R]}\right) - l_x[X]$$

since only α_P, α_Q and α_{PQ} have the value one and every other parameter has the value zero.

Next, we discuss the minor variation of BoolODE from GeneNetWeaver, which is in how we sample kinetic parameters. The GeneNetWeaver equations use four kinetic parameters: one each for mRNA transcription, protein translation and mRNA and protein degradation rates. Saelens et al.[20] sample them uniformly from parameter specific intervals. Independently for every dataset, we sample each parameter from a normal distribution using the value shown in Supplementary Table 6 as the mean and a standard deviation of up to 10% of this mean value. Within a single dataset and for all simulations for that dataset, we fix each parameter (for example, mRNA degradation rate) for all genes. We choose the values in Supplementary Table 6 so as to achieve the following characteristics: The maximal steady state achievable by the mRNAs is two (the value of m/l_x), of the proteins is ten (the value of r/l_p), and the time scale of protein production is ten times that of the mRNAs (since the characteristic time scale of production is inversely proportional to the degradation rate).

To create stochastic simulations, we use the formulation proposed by Saelens et al.[20] to modify the ODE expressions as follows:

$$\frac{d[x_i]}{dt} = mf(R_i) - l_x[x_i] + s\sqrt{[x_i]}\Delta W_t$$

$$\frac{d[p_i]}{dt} = r[x_i] - l_p[p_i] + s\sqrt{[p_i]}\Delta W_t$$

$$\Delta W_t = \mathcal{N}(0, h)$$

where s is the noise strength. We use $s = 10$ in our simulations. We use the Euler–Maruyama scheme for numerical integration of the SDEs with a time step of $h = 0.01$.

Defining a single cell. We define the vector of gene expression values corresponding to a particular time point in a model simulation as a single cell. For every analysis, we sample one time point; that is, one cell from a single simulation. Using this procedure, for a dataset generated from 5,000 simulations, we can obtain up to 5,000 cells.

Creating GeneNetWeaver simulations for comparison with BoolODE. To simulate a synthetic network using GeneNetWeaver, we used its edge list as the input network to GeneNetWeaver. To create the simulations, we used the default options of the noise parameter (0.05) and multifactorial perturbations. We only performed wildtype simulations and used the DREAM4 time series output format for comparison with the BoolODE output.

Summary. We developed the BoolODE approach to convert Boolean functions specifying a GRN directly to ODE equations. Our proposed BoolODE pipeline accepts a file describing a Boolean model as input, creates an equivalent ODE model, adds noise terms and numerically simulates a stochastic time course. Different model topologies can produce different numbers of steady states. Since we carry out stochastic simulations, we perform a large number of simulations in an attempt to ensure that we can reach every steady state. Our analysis of the trajectories computed by BoolODE on datasets from curated models demonstrates the success of our approach in this regard (Supplementary Note 2.1). We prefer BoolODE over a direct application of GeneNetWeaver to create datasets from synthetic networks and datasets from curated models for three reasons: (1) a dense regulatory subnetwork computed around a randomly selected node, as used by GeneNetWeaver, may not correspond to a real biological process; (2) GeneNetWeaver introduces a random, initial, multifactorial perturbation and removes it halfway to create variations in the expression profiles of genes across samples. This stimulation may not correspond to how single-cell gene expression data is collected and (3) GeneNetWeaver's SDEs do not appear to capture single-cell expression trajectories, as we have shown in Supplementary Fig. 1d).

Datasets. A major challenge that arises when we evaluate GRN inference algorithms for single-cell RNA-seq data is that the 'ground truth', that is, the network of regulatory interactions governing the dynamics of genes of interest, is usually unknown. Consequently, it is a common practice to create artificial graphs or extract subnetworks from large-scale transcriptional networks.

To address this challenge, we used three sets of networks that serve as the ground truth for GRN inference. The first group included six 'toy' networks with specific topologies that give rise to different cellular trajectories with predictable qualitative properties[20]. For the second set of networks, we curated four published Boolean models that explore gene regulatory interactions underlying various developmental and tissue differentiation processes. Mutual inhibition between a pair of genes is a key characteristic of each of these models; this type of relationship is important in creating branching gene expression trajectories. The regulatory networks underlying the Boolean models serve as the ground truth during evaluation. We used the third group of networks for experimental scRNA-seq datasets. We matched each scRNA-seq dataset with an appropriate ChIP–seq-derived network connecting TFs to their targets; we ensured that the ChIP–seq data was collected in the same or similar cell type as the scRNA-seq measurements. In addition, we used noncell-type-specific transcriptional regulatory networks[43–45] as well as the functional interactions in the STRING[46] database as the ground truth.

Creating datasets from synthetic networks. We now describe how we selected synthetic networks, converted these networks into systems of SDEs, simulated these systems, and preprocessed the resulting datasets for input to GRN inference algorithms.

Selecting networks. To create synthetic datasets exhibiting diverse temporal trajectories, we use six 'toy' networks created in Dynverse, a comparison of pseudotime inference algorithms[20] (Supplementary Fig. 1 and Supplementary Table 1). When simulated as SDEs, we expect the models produce to trajectories with the following qualitative properties:

1. Linear. A gene activation cascade that results in a single temporal trajectory with distinct final and initial states.
2. Linear long. Similar to linear but with a larger number of intermediate genes.
3. Cycle. An oscillatory circuit that produces a linear trajectory where the final state overlaps with the initial state.
4. Bifurcating. A network that contains a mutual inhibition motif between two genes resulting in two distinct branches starting from a common trajectory.
5. Trifurcating. Mutual inhibition motifs involving three genes in this network result in three distinct steady states.
6. Bifurcating converging. An initial bifurcation creates two branches, which ultimately converge to a single steady state.

We then used the following approach to simulate the above networks.

Converting networks into SDE models and simulating them. We manually convert each of these networks to a Boolean model: we set a node to be 'on' if and only if at least one activator is 'on' and every inhibitor is 'off'. We simulate these networks using BoolODE. We use the initial conditions specified in the Dynverse software[20]

(Supplementary Table 6). To sample cells from the simulations that capture various locations along a trajectory, we limit the duration of each simulation according to the characteristics of the model (Supplementary Table 7). For example, we simulated the linear long network for 15 time units but the linear network, which has fewer nodes, for five time units.

Comparing simulated datasets with the expected trajectories from synthetic networks. It is common to visualize simulated time courses from ODE/SDE models as time course plots or phase plane diagrams with two or three dimensions. The latter are useful to qualitatively explore the state-space of a model at hand. The recent popularity of *t*-distributed stochastic neighbor embedding (*t*-SNE) as a tool to visualize and cluster high dimensional scRNA-seq data motivated us to consider this technique to visualize simulated single-cell data. Supplementary Fig. 1b shows two-dimensional *t*-SNE visualizations of each of the toy models.

Preprocessing datasets from synthetic networks for GRN algorithms. Pseudotime inference methods perform well for linear and bifurcating trajectories[20]. However, even the best-performing pseudotime algorithm fails to accurately identify more complex trajectories such as cycle and bifurcating converging. Therefore, we sought to develop an approach for preprocessing synthetic datasets that mimicked a real single-cell gene expression pipeline while isolating the GRN inference algorithms from the limitations of pseudotime techniques. Accordingly, we used the following four-step approach to generate single-cell gene expression data from each of the six synthetic networks:

1. Using the values in Supplementary Table 6 as the means, and 10% of these values as the standard deviations, we sampled a parameter set. We then used BoolODE to perform 5,000 simulations using this parameter set.
2. We represented each simulation as a $|G|\times|C|$ matrix, where G is the set of genes in the model and C is the set of cells in the simulation. We converted each of the 5,000 matrices into a one-dimensional vector of length $|G|\times|C|$ and clustered the vectors using k-means clustering with k set to the number of expected trajectories for each network. For example, the bifurcating network in Supplementary Fig. 1b has two distinct trajectories, so we used $k=2$. We use the cluster information in step 4.
3. We then randomly sampled one set each of 100, 200, 500, 2,000 and 5,000 cells from the 5,000 simulations.
4. Finally, we set as input to each algorithm the $|G|\times|D|$ matrix, where G is the set of genes in the model and D is the set of randomly sampled cells. For those methods that require time information, we specified the simulation time at which the cell was sampled along with the trajectory (cluster) to which each cell belonged.

We repeated this procedure on ten different sampled parameter sets to obtain 50 datasets. We ran each algorithm on these 50 datasets.

Note that we used the same set of sampled parameters for all simulations in a dataset and that we varied parameters only across datasets. As an alternative, we considered sampling a different set of parameters per simulation since they may vary from cell to cell. However, this approach caused so much variation that we could not recapitulate the steady states of the Boolean models in the BoolODE-created data.

Note that we clustered simulations themselves (with each simulation represented by the complete time courses of all genes) before we sampled cells. The reason we adopted this ordering is that the goal of the clustering was to partition the cells such that each group would (1) correspond to a distinct steady state of the network and (2) contain cells sampled from the entire time course of the simulations. Clustering the simulations helped us to compute these types of cluster. Subsequently sampling one cell from each simulation permitted us to assign each cell to the cluster to which its simulation belonged and satisfy both properties we desired.

In contrast, if we had sampled cells and then clustered them, we were concerned that some cells would have belonged to a cluster corresponding only to early time points and some only to intermediate, resulting in the possibility that some steady states would not have any clusters. Alternatively, we would have had to increase the number of clusters we sought to compute, which may also have resulted in clusters corresponding only to early or only to intermediate time points.

Creating datasets from curated models. While the synthetic models presented above are useful for generating simulated data with a variety of specific trajectories, these networks do not correspond to any real cellular process. To create simulated datasets that better reflect the characteristics of single-cell transcriptomic datasets, we turned to published Boolean models of GRNs, as these models are reflective of the real 'ground-truth' control systems in biology.

Since tissue differentiation and development are active areas of investigation by single-cell methods, we examined the literature from the past ten years to look for published Boolean models of GRNs involved in these processes. We selected four published models for analysis. Supplementary Table 2 lists the size of the regulatory networks and the number of steady states. Below, we discuss the biological background, the interpretation of model steady states, and the expected type of trajectories for each of these Boolean models.

mCAD. Giacomantonio et al. explored mCAD as a consequence of the expression of regulatory TFs along an anterior-posterior gradient[25]. The model contains five TFs connected by 14 interactions, captures the expected gene expression patterns in the anterior and posterior compartments, respectively, and results in two steady states. Figure 3 displays the regulatory network underlying the model along with a *t*-SNE visualization of the trajectories simulated using BoolODE. In Supplementary Note 2.1, we show that the two clusters observed in the *t*-SNE visualization correspond to the two biological states captured by the Boolean model.

VSC development. Lovrics et al. investigated the regulatory basis of VSC development[26]. The model consisting of eight TFs involved in ventralization contains 15 interactions, all of which are inhibitory. It succeeds in accounting for five distinct neural progenitor cell types. We expect to see five steady states from this model. Figure 3 shows the regulatory network underlying the model along with the *t*-SNE visualization of the trajectories simulated using BoolODE. In Supplementary Note 2.1, we show that the five steady-state clusters observed in the *t*-SNE visualization correspond to the five biological states captured by the model.

HSC differentiation. Krumsiek et al. investigated the GRN underlying myeloid differentiation[27]. The proposed model has 11 TFs and captures the differentiation of multipotent myeloid progenitor (CMP cells) into erythrocytes, megakaryocytes, monocytes and granulocytes. The Boolean model exhibits four steady states, each corresponding to one of the four cell types mentioned above. Figure 3 shows the regulatory network of the HSC model along with the *t*-SNE visualization of the trajectories simulated using BoolODE. In Supplementary Note 2.1, we show that the four steady-state clusters observed in the *t*-SNE visualization correspond to the four biological states captured by the Boolean model.

GSD. Rios et al. modeled the gonadal differentiation circuit that regulates the maturation of the bipotential gonadal primordium into either male (testes) or female (ovary) gonads[28]. The model consists of 18 genes and a node representing the urogenital ridge, which serves as the input to the model. For the wildtype simulations, the Boolean model predominantly exhibits two steady states corresponding to the Sertoli cells (male gonad precursors) or the granulosa cells (female gonad precursors), and one rare state corresponding to a dysfunctional pathway. Figure 3 shows the regulatory network of the GSD model along with the *t*-SNE visualization of the trajectories simulated using BoolODE. In Supplementary Note 2.1, we show that the two steady-state clusters observed in the *t*-SNE visualization correspond to the two predominant biological states capture by the Boolean model.

Preprocessing datasets from curated models for GRN algorithms. As with the datasets from synthetic networks, we used the following approach to generate simulated datasets for each of the four curated models. Using the values in Supplementary Table 6 as means and 5% of these values as the standard deviations, we sampled a parameter set. We used BoolODE to perform 2,000 simulations and randomly sampled one cell from each simulation. We repeated this procedure on ten different sampled parameter sets to obtain ten datasets. To closely mimic the preprocessing steps performed for real single-cell gene expression data, we use the following steps for each dataset:

1. Pseudotime inference using Slingshot. In the case of datasets from synthetic networks, we used the simulation time directly for those GRN inference methods that required cells to be time-ordered. In contrast, for datasets from curated models, we ordered cells by pseudotime, which we computed using Slingshot[29]. We selected Slingshot for pseduotime inference due to its proven success in correctly identifying cellular trajectories in a recent comprehensive evaluation of this type of algorithm[20]. Slingshot needs a lower dimensional representation of the gene expression data as input. In addition, if the cells belong to multiple trajectories, Slingshot needs a vector of cluster labels for the cells, as well as the cluster labels for cells in the start and end states in the trajectories. To obtain these additional input data for Slingshot, we used the following procedure:
   (a) We use *t*-SNE on the $|G|\times|C|$ matrix representing the data to obtain a two-dimensional representation of the cells.
   (b) We performed k-means clustering on this lower dimensional representation of the cells with k set to one more than the expected number of trajectories. For example, since we knew that the GSD model had a bifurcating trajectory, we used $k=3$.
   (c) We computed the average simulation time of the cells belonging to each of the k clusters. We set the cluster label corresponding to the smallest average time as the starting state and the rest of clusters as ending states.
   (d) We then ran Slingshot with the *t*-SNE-projected data and starting and ending clusters as input. We obtained the trajectories to which each cell belonged and its pseudotime as output from Slingshot.

   Figure 3d displays the results for one dataset of 2,000 cells for each of the four models. We observed that Slingshot does a very good job of correctly identifying cells belonging to various trajectories. Further, the pseudotime computed for each cell by Slingshot is highly correlated with the

simulation time at which we sampled the cell (Supplementary Table 8). Note that the *k*-means clustering whose results we display in Fig. 3c is different from the *k*-means clustering described above. To obtain the results in Fig. 3c, we followed the procedure described for synthetic models: we clustered the simulations (complete time courses) themselves, with *k* set to the number of steady states. Our goal was to confirm visually that each cluster contained cells spanning the entire length of the simulation. In contrast, before applying Slingshot, we clustered the (lower dimensional representation of the samples) cells; we set *k* as described above so that we could input to Slingshot one starting cluster and as many ending clusters as the number of steady states.

2. Inducing dropouts in datasets from curated models. We used the same procedure as Chan et al.[14] to induce dropouts, which are commonly seen in single-cell RNA-seq datasets, especially for transcripts with low abundance[47]. We created a dataset with a dropout rate of q as follows: for every gene, we sorted the cells in increasing order of that gene's expression value. We set the expression of that gene in each of the lowest qth percentile of cells in this order to zero with a q% chance. For example, choosing $q = 50$ resulted in a 50% chance of setting a gene's expression values below the 50th percentile to 0, which affected about 25% of the dataset. Note that we used the same parameter twice simply for convenience; we do not expect the trends we find to deviate considerably if we had used two different parameters for percentile and for probability. We applied two different dropout rates: $q = 50$ and $q = 70$. We used Slingshot to recompute pseudotime for each dataset with dropouts. We did note a considerable decrease in this correlation when we added dropouts to the simulated datasets, for all but the GSD network (Supplementary Table 8).

At the end of this step, we had 30 datasets with 2,000 cells for each of the four models: ten datasets without dropouts, ten with a dropout rate of $q = 50$ and ten with a rate of $q = 70$.

Collecting experimental single-cell RNA-seq datasets and ground-truth networks. Experimental single-cell RNA-seq datasets. We obtained five different single-cell RNA-seq datasets, three in mouse and two in human (Supplementary Table 3). There were a total of seven cell types across these datasets. We preprocessed each dataset using the procedure described in the corresponding paper. In general, if the publication did not provide normalized expression values, we log-transformed the transcripts per kilobase million or fragments per kilobase million counts using a pseudocount of 1 and used the results as the expression values. We additionally filtered out any genes that were expressed in fewer than 10% of the cells. We describe the details of the datasets and the pseudotime computation below:

1. mHSCs[48]. We obtained the normalized expression data for 1,656 HSPCs across 4,773 genes from the supplementary data provided by the authors. We used the first three dimensions from DiffusionMap to compute pseudotime values. We used the E-SLAM population as the starting cell type and computed pseudotime using Slingshot along three lineages, namely erythroid, granulocyte-monocyte and lymphoid. We inferred GRNs for each lineage independently.
2. Mouse embryonic stem cells (mESC)[49]. This dataset contains scRNA-seq expression measurements for 421 primitive endoderm (PrE) cells differentiated from mESCs, collected at five different time points (0, 12, 24, 48 h up to 72 h). SCODE[13], SINGE[10] and GRISLI[9] used this dataset to evaluate their performance. We computed pseudotime using Slingshot with cells measured at 0 h as the starting cluster and the cells measured at 72 h as the ending cluster.
3. Mouse dendritic cells[50]. This dataset corresponds to over 1,700 bone-marrow derived dendritic cells under various conditions. Following SCRIBE[19], we used the lipopolysaccharide stimulated wildtype cells measured at 1, 2, 4 and 6 h. We then computed the pseudotime using Slingshot with cells measured at 1 h as the starting cluster and the cells measured at 6 h as the ending cluster.
4. hHEPs[51]. This dataset is from an scRNA-seq experiment on induced pluripotent stem cells (iPSCs) in two-dimensional culture differentiating to hepatocyte-like cells. The dataset contains 425 scRNA-seq measurements from multiple time points: days 0 (iPSCs), 6, 8, 14 and 21 (mature hepatocyte-like). We computed the pseudotime using Slingshot with cells measured on day 0 (iPSCs) as the starting cluster and the cells measured on 21 (mature hepatocytes) as the ending cluster.
5. hESCs[52]. This dataset is from a time course scRNA-seq experiment derived from 758 cells along the differentiation protocol to produce definitive endoderm cells from human embryonic stem cells, measured at 0, 12, 24, 36, 72 and 96 h. We computed the pseudotime with cells measured at 0 h as the starting cluster and the cells measured at 96 h as the ending cluster. SCODE[13] used this dataset to evaluate its performance.

Once we obtained the pseudotime values for the cells in each dataset, we computed which genes had varying expression values across pseudotime. We used the general additive model implemented in the 'gam' R package to compute the variance and the *P* value of this variance. We used the Bonferroni method to correct for testing multiple hypotheses. Supplementary Table 3 provides the statistics on the number of significantly varying genes and TFs in each dataset after using a corrected *P* value cutoff of 0.01. We selected genes for GRN inference in two different ways.

i. We considered all genes with a *P* value less than 0.01. We selected variance thresholds so that we obtained 500 and 1,000 highly varying genes. We recorded the number of TFs in these sets.
ii. We started by including all TFs whose variance had *P* value at most 0.01. Then, we added 500 and 1,000 additional genes as in the previous option. This approach enabled the GRN methods to consider TFs that may have a modest variation in gene expression but still regulate their targets.

After applying a GRN inference algorithm to a dataset, we only considered interactions outgoing from a TF in further evaluation.

Ground-truth networks collection and processing. In the GRN inference literature, a common practice is to evaluate the accuracy of a resulting network by comparing its edges to an appropriate database of TFs and their targets. For example, SCODE used the RikenTFdb and animalTFDB resources to define ground-truth networks for mouse and human gene expression data, respectively[13]. We used three types of ground-truth dataset (Supplementary Table 4).

1. Cell-type-specific. For each experimental scRNA-seq dataset, we searched the ENCODE, ChIP-Atlas and ESCAPE databases for ChIP–seq data from the same or similar cell type. We also included the loss-of-function/gain-of-function (lof/gof) dataset from the ESCAPE database.
2. Nonspecific. Here, we used the following resources:
   (a) DoRothEA[45] integrates ChIP–seq and transcriptional regulatory information from multiple sources. We considered two levels of evidence in this database: A (curated/high confidence) and B (likely confidence).
   (b) RegNetwork[43] incorporates genome-wide TF–TF, TF–gene, TF–micro RNA regulatory relationships in human and mouse collected from various sources. We used the TF–TF and TF–gene interactions for our analysis.
   (c) TRRUST[44] contains TF–target interactions collected based on text-mining followed by manual curation for human and mouse.
3. Functional. Finally, we used the human and mouse STRING[46] networks. An interaction here is functional and need not correspond to transcriptional regulation. We selected this type of ground-truth network due to our observation that many GRN methods predict indirect interactions for Boolean models.

Evaluation pipeline. One of the major challenges we faced was that the GRN inference methods we included in this evaluation were implemented in a variety of languages such as R, MATLAB, Python, Julia and F#. To obtain an efficient and reproducible pipeline, we Dockerized the implementation of each algorithm. Supplementary Table 9 contains details on the specific software or GitHub commit versions we downloaded and used in our pipeline. For methods implemented in MATLAB, we created MATLAB executable files (.mex files) that we could execute within a Docker container using the MATLAB Runtime. We further studied the publications and the documentation of the software (and the source code, on occasion) to determine how the authors recommended that their methods be used. We implemented these suggestions as well as we could. We provide more details in Supplementary Note 4.

Inputs. For datasets from synthetic networks and for datasets from curated models, we provided the gene expression values obtained from the simulations directly after optionally inducing dropouts in the second type of data. Eight out of the 12 methods also required some form of time information for every cell in the dataset (Fig. 6). Of these, two methods (GRNVBEM and LEAP) only required cells to be ordered according to their pseudotime and did not require the pseudotime values themselves.

Parameter estimation. Six of the methods also required one or more parameters to be specified. To this end, we performed parameter estimation for each of these methods separately for datasets from synthetic networks, datasets from curated models and real datasets, and provided them with the parameters that resulted in the best AUPRC values. See Supplementary Note 1.2 for details.

Output processing. Ten of the GRN inference methods output a confidence score for every possible edge in the network, either as an edge list or as an adjacency matrix, which we converted to a ranked edge list. We gave the same rank to edges with the same confidence scores.

Performance evaluation. We used a common evaluation pipeline across all the datasets considered in this paper. We evaluated the result of each algorithm using the following criteria:

1. AUPRC, AUROC. We computed areas under the precision-recall and receiver operating characteristic curves using the edges in the relevant network as ground truth and ranked edges from each method as the predictions. We ignored self-loops for this analysis since some methods such as PPCOR always assigned the highest rank to such edges and some other methods such as SINGE always ignored them. Most of the networks we considered have a

density of 0.3 or less; that is, the positive-to-negative ratio is worse than 1:3. Since the GRN inference problem for these networks is moderately imbalanced, we focus on AUPRC scores in the main text[53,54]. For readers who are interested in AUROC plots, we provide them as supplementary figures (only for the synthetic networks and curated models).

2. Stability across multiple runs. We executed each algorithm ten times on a dataset to ask if the inferred networks changed from one run to another. We represented every result by the corresponding ranked list of edges. For every pair of results, we computed the Spearman's correlation of the ranked lists. We performed this analysis for curated models and for experimental datasets.
3. Identifying top-k edges. We first identified top-k edges for each method, where k equaled the number of edges in the ground-truth network (excluding self-loops). In cases where multiple edges were tied for a rank of k, we considered all of them. If a method provided a confidence score for fewer than k edges, we used only those edges. For experimental single-cell RNA-seq datasets, this number varied from one ground-truth dataset to another.
4. Stability across multiple datasets. For each method, once we obtained the set of top-k edges for each dataset, we then computed the Jaccard index of every pair of these sets. We used the median of the values as an indication of the robustness of a method's output to variations in the simulated datasets from a given synthetic network or curated model.
5. Early precision (EP) and EPR. We defined early precision as the fraction of true positives in the top-k edges. We also computed the EPR, which represents the ratio of early precision value and the early precision for a random predictor for that model. A random predictor's precision is the edge density of the ground-truth network.
6. Early precision of signed edges. We desired to check whether there were any differences in how accurately a GRN inference algorithm identified activating edges in comparison to inhibitory edges. To this end, we computed the top-k_a edges from the ranked list of edges output by each method, where k_a is the number of activating edges in the ground-truth network. In this step, we ignored any inhibitory edges in the ground-truth network. We defined the early precision of activating edges as the fraction of true edges of this type in the top-k_a edges. We used an analogous approach to compute early precision of inhibitory edges. We also computed the EPR for these values. We performed this analysis only for curated models.

Datasets with multiple trajectories. Seven methods we evaluated require pseudotime-ordered cells (Fig. 6), but cannot directly handle data with branched trajectories. In these cases, as suggested by many of these methods, we separated the cells into multiple linear trajectories using Slingshot and applied the algorithm to the set of cells in each trajectory individually. To combine the GRNs, for each interaction, we recorded the largest score for it across all the networks and ranked the interactions by these values. In the case of GRISLI, which outputs ranked edges, for each interaction, we took the best (smallest) rank for it across all the networks.

As an alternative, we merged all the trajectories into one set of cells and executed each algorithm on this dataset. We performed this analysis for the three synthetic networks with multiple trajectories (bifurcating, bifurcating converging and trifurcating).

Shuffling simulation times. We investigated the effect of shuffling the pseudotime values on the performance of methods that require this information. We used three window sizes, namely, 15, 30 and 45%, which defined the range of indices to sample from as a fraction of the total number of cells in a dataset. Thus, for a dataset with 2,000 cells, using a window size of 30% resulted in a range of $2{,}000 \times 0.3 = 600$. Therefore, after sorting the cells in increasing order of pseudotime, for every index i, we sampled a new index from the interval $[\max(0, i-300), \min(i+300, 2{,}000)]$ and swapped the cells at these two indices. We performed this analysis for the three synthetic networks with a single trajectory (linear, cycle and Linear long). Here each original (unshuffled) pseudotime was equal to the simulation time.

Procedure for creating Fig. 6. We chose the following measures presented in earlier sections for summarize our key findings:

Accuracy. We computed the median of the per-network median AUPRC value obtained for datasets containing 2,000 and 5,000 cells for the six synthetic networks (Fig. 2). For datasets from curated models, we used the median of the per-model median AUPRC value obtained for ten datasets without dropouts for each the four datasets from curated models (Fig. 4). For experimental single-cell RNA-seq datasets, we used the median EPR obtained for each dataset-evaluation network combination (all significantly varying TFs and the 500 most-varying genes) (Fig. 5). We have ordered the algorithms by their median EPR score across experimental single-cell RNA-seq datasets followed by median AUPRCs in datasets from synthetic networks.

Stability. We present four such measures: (1) across datasets: the median score of the Jaccard index obtained for all six datasets from synthetic networks (Fig. 2); (2) across runs: median Spearman's correlation of outputs for each of the four datasets from curated models (one dataset each, Supplementary Note 3.4); (3) across dropouts: median percentage decrease in AUPRC for the $q = 50$ dropout rate compared to no dropouts (Supplementary Fig. 4) and (4) across pseudotime: median percentage decrease in AUPRC after shuffling time values within a 30% window compared to the original simulation time for datasets from synthetic networks (Supplementary Fig. 11).

Scalability. Median running times and memory consumption for three different scRNA-seq datasets, namely hHEP, hESC and mHSC-E, which contain 400, 750 and 1,000 cells, respectively (Supplementary Fig. 7). Missing values indicate that either the method did not complete even after running for over a day or it gave a runtime error.

Reporting Summary. Further information on research design is available in the Nature Research Reporting Summary linked to this article.

Data availability

The datasets simulated from the synthetic networks and curated models and the processed experimental single-cell gene expression datasets are available on Zenodo at https://doi.org/10.5281/zenodo.3378975. The gene experimental scRNA-seq datasets we downloaded from Gene Expression Omnibus had the accession numbers GSE81252 (hHEP), GSE75748 (hESC), GSE98664 (mESC), GSE48968 (mouse dendritic cell) and GSE81682 (mHSC). Source data for Figs. 2 and 4–6 are provided with the paper.

Code availability

A Python implementation of the BEELINE framework is available under the GNU General Public License v.3 at https://github.com/murali-group/BEELINE.

References

35. Faith, J. J. et al. Large-scale mapping and validation of *Escherichia coli* transcriptional regulation from a compendium of expression profiles. *PLoS Biol.* **5**, e8 (2007).
36. La Manno, G. et al. RNA velocity of single cells. *Nature* **560**, 494–498 (2018).
37. Yuan, Y. & Bar-Joseph, Z. Deep learning for inferring gene relationships from single-cell expression data. Preprint at *bioRxiv* https://doi.org/10.1101/365007 (2019).
38. Chen, H. et al. Single-cell transcriptional analysis to uncover regulatory circuits driving cell fate decisions in early mouse development. *Bioinformatics* **31**, 1060–1066 (2015).
39. Hamey, F. K. et al. Reconstructing blood stem cell regulatory network models from single-cell molecular profiles. *Proc. Natl Acad. Sci. USA* **114**, 5822–5829 (2017).
40. Marbach, D., Schaffter, T., Mattiussi, C. & Floreano, D. Generating realistic in silico gene networks for performance assessment of reverse engineering methods. *J. Comput. Biol.* **16**, 229–239 (2009).
41. Marbach, D. et al. Revealing strengths and weaknesses of methods for gene network inference. *Proc. Natl Acad. Sci. USA* **107**, 6286–6291 (2010).
42. Ackers, G. K., Johnson, A. D. & Shea, M. A. Quantitative model for gene regulation by λ phage repressor. *Proc. Natl Acad. Sci. USA* **79**, 1129–1133 (1982).
43. Liu, Z. P., Wu, C., Miao, H. & Wu, H. RegNetwork: an integrated database of transcriptional and post-transcriptional regulatory networks in human and mouse. *Database* **2015**, bav095 (2015).
44. Han, H. et al. TRRUST v2: an expanded reference database of human and mouse transcriptional regulatory interactions. *Nucleic Acids Res.* **46**, D380–D386 (2018).
45. Garcia-Alonso, L., Holland, C. H., Ibrahim, M. M., Turei, D. & Saez-Rodriguez, J. Benchmark and integration of resources for the estimation of human transcription factor activities. *Genome Res.* **29**, 1363–1375 (2019).
46. Szklarczyk, D. et al. STRING v11: protein–protein association networks with increased coverage, supporting functional discovery in genome-wide experimental datasets. *Nucleic Acids Res.* **47**, 607–613 (2018).
47. Brennecke, P. et al. Accounting for technical noise in single-cell RNA-seq experiments. *Nat. Methods* **10**, 1093–1095 (2013).
48. Nestorowa, S. et al. A single-cell resolution map of mouse hematopoietic stem and progenitor cell differentiation. *Blood* **128**, 20–31 (2016).
49. Hayashi, T. et al. Single-cell full-length total RNA sequencing uncovers dynamics of recursive splicing and enhancer RNAs. *Nat. Commun.* **9**, 619 (2018).
50. Shalek, A. K. et al. Single-cell RNA-seq reveals dynamic paracrine control of cellular variation. *Nature* **510**, 363–369 (2014).
51. Camp, J. G. et al. Multilineage communication regulates human liver bud development from pluripotency. *Nature* **546**, 533–538 (2017).
52. Chu, L. F. et al. Single-cell RNA-seq reveals novel regulators of human embryonic stem cell differentiation to definitive endoderm. *Genome Biol.* **17**, 173 (2016).

53. Saito, T. & Rehmsmeier, M. The precision-recall plot is more informative than the ROC plot when evaluating binary classifiers on imbalanced datasets. *PLoS One* **10**, e0118432 (2015).
54. Saito, T. & Rehmsmeier, M. Precrec: fast and accurate precision-recall and ROC curve calculations in R. *Bioinformatics* **33**, 145–147 (2017).

Acknowledgements

Grants from the National Science Foundation (nos. CCF-1617678 and DBI-1759858) and the National Cancer Institute (grant no. UH2CA203768) supported this work. The research is also based on work supported by the Office of the Director of National Intelligence (ODNI), Intelligence Advanced Research Projects Activity (IARPA), via the Army Research Office (ARO) under cooperative agreement no. W911NF-17-2-0105. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of the ODNI, IARPA, ARO or the US Government. The US Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright annotation thereon. The funding bodies played no role in the design of the study, the collection, analysis and interpretation of data or in writing the manuscript.

Author contributions

A.P. and T.M.M. conceived and designed the analysis and selected the GRN algorithms. A.P. implemented BEELINE and led the analysis. A.P.J., A.P. and T.M.M. developed BoolODE and A.P.J. implemented it. A.P. and A.P.J. created and processed datasets. A.P., A.P.J., J.N.L. and A.B. contributed evaluation strategies. All authors analyzed the results. A.P., A.P.J. and T.M.M. wrote the paper. T.M.M. supervised the project.

Competing interests

The authors declare no competing interests.

Additional information

Supplementary information is available for this paper at https://doi.org/10.1038/s41592-019-0690-6.

Correspondence and requests for materials should be addressed to T.M.M.

Peer review information Nicole Rusk and Lin Tang were the primary editors on this article and managed its editorial process and peer review in collaboration with the rest of the editorial team.

Reprints and permissions information is available at www.nature.com/reprints.

nature research

Corresponding author(s): T. M. Murali

Last updated by author(s): Oct 1, 2019

Reporting Summary

Nature Research wishes to improve the reproducibility of the work that we publish. This form provides structure for consistency and transparency in reporting. For further information on Nature Research policies, see Authors & Referees and the Editorial Policy Checklist.

Statistics

For all statistical analyses, confirm that the following items are present in the figure legend, table legend, main text, or Methods section.

n/a	Confirmed	
☐	☒	The exact sample size (*n*) for each experimental group/condition, given as a discrete number and unit of measurement
☐	☒	A statement on whether measurements were taken from distinct samples or whether the same sample was measured repeatedly
☐	☒	The statistical test(s) used AND whether they are one- or two-sided *Only common tests should be described solely by name; describe more complex techniques in the Methods section.*
☒	☐	A description of all covariates tested
☐	☒	A description of any assumptions or corrections, such as tests of normality and adjustment for multiple comparisons
☒	☐	A full description of the statistical parameters including central tendency (e.g. means) or other basic estimates (e.g. regression coefficient) AND variation (e.g. standard deviation) or associated estimates of uncertainty (e.g. confidence intervals)
☐	☒	For null hypothesis testing, the test statistic (e.g. *F*, *t*, *r*) with confidence intervals, effect sizes, degrees of freedom and *P* value noted *Give P values as exact values whenever suitable.*
☒	☐	For Bayesian analysis, information on the choice of priors and Markov chain Monte Carlo settings
☒	☐	For hierarchical and complex designs, identification of the appropriate level for tests and full reporting of outcomes
☒	☐	Estimates of effect sizes (e.g. Cohen's *d*, Pearson's *r*), indicating how they were calculated

Our web collection on statistics for biologists contains articles on many of the points above.

Software and code

Policy information about availability of computer code

Data collection	No software used.
Data analysis	The Python code used in this paper is available at https://github.com/Murali-group/Beeline/. The file provided at https://github.com/Murali-group/Beeline/blob/master/requirements.txt lists the primary software dependencies of BEELINE with version numbers. In addition, Supplementary File 1 contains details on the specific versions or GitHub commit numbers of the gene regulatory network inference algorithms we compared in this study.

For manuscripts utilizing custom algorithms or software that are central to the research but not yet described in published literature, software must be made available to editors/reviewers. We strongly encourage code deposition in a community repository (e.g. GitHub). See the Nature Research guidelines for submitting code & software for further information.

Data

Policy information about availability of data

All manuscripts must include a data availability statement. This statement should provide the following information, where applicable:

- Accession codes, unique identifiers, or web links for publicly available datasets
- A list of figures that have associated raw data
- A description of any restrictions on data availability

The datasets simulated from the synthetic networks and curated models and the processed experimental single-cell gene expression datasets are available on Zenodo at https://doi.org/10.5281/zenodo.3378975

Field-specific reporting

Please select the one below that is the best fit for your research. If you are not sure, read the appropriate sections before making your selection.

☒ Life sciences ☐ Behavioural & social sciences ☐ Ecological, evolutionary & environmental sciences

For a reference copy of the document with all sections, see nature.com/documents/nr-reporting-summary-flat.pdf

Life sciences study design

All studies must disclose on these points even when the disclosure is negative.

Sample size	Sample sizes for the publicly available experimental datasets were determined in the original publications.
Data exclusions	There were no data exclusions.
Replication	We did not perform any replicate analysis.
Randomization	Randomization is not relevant to our study as we reanalyzed publicly available data or generated our own synthetic data.
Blinding	Blinding is not applicable as we reanalyzed publicly available data or generated our own synthetic data.

Reporting for specific materials, systems and methods

We require information from authors about some types of materials, experimental systems and methods used in many studies. Here, indicate whether each material, system or method listed is relevant to your study. If you are not sure if a list item applies to your research, read the appropriate section before selecting a response.

Materials & experimental systems

n/a	Involved in the study
☒	☐ Antibodies
☒	☐ Eukaryotic cell lines
☒	☐ Palaeontology
☒	☐ Animals and other organisms
☒	☐ Human research participants
☒	☐ Clinical data

Methods

n/a	Involved in the study
☒	☐ ChIP-seq
☒	☐ Flow cytometry
☒	☐ MRI-based neuroimaging