GRouNdGAN: GRN-guided simulation of single-cell RNA-seq data using causal generative adversarial networks

Yazdan Zinati[1], Abdulrahman Takiddeen[1], and Amin Emad[1,2,3,*]

[1] Department of Electrical and Computer Engineering, McGill University, Montreal, QC, Canada

[2] Mila, Quebec AI Institute, Montreal, QC, Canada

[3] The Rosalind and Morris Goodman Cancer Institute, Montreal, QC, Canada

* Corresponding Author:

Amin Emad

755 McConnell Engineering Building

3480 University Street, Montreal, Quebec, Canada, H3A 0E9

Email: amin.emad@mcgill.ca

Abstract

We introduce GRouNdGAN, a gene regulatory network (GRN)-guided causal implicit generative model for simulating single-cell RNA-seq data, *in-silico* perturbation experiments, and benchmarking GRN inference methods. Through the imposition of a user-defined GRN in its architecture, GRouNdGAN simulates steady-state and transient-state single-cell datasets where genes are causally expressed under the control of their regulating transcription factors (TFs). Training on three experimental datasets, we show that our model captures non-linear TF-gene dependences and preserves gene identities, cell trajectories, pseudo-time ordering, and technical and biological noise, with no user manipulation and only implicit parameterization. Despite imposing rigid causality constraints, it outperforms state-of-the-art simulators in generating realistic cells. GRouNdGAN learns meaningful causal regulatory dynamics, allowing sampling from both observational and interventional distributions. This enables it to synthesize cells under conditions that do not occur in the dataset at inference time, allowing to perform *in-silico* TF knockout experiments. Our results show that *in-silico* knockout of cell type-specific TFs significantly reduces cells of that type being generated, even though GRouNdGAN was not trained on any perturbation datasets. Interactions imposed through the GRN are emphasized in the simulated datasets, resulting in GRN inference algorithms assigning them much higher scores than interactions not imposed but of equal importance in the experimental training dataset. Benchmarking various GRN inference algorithms reveals that GRouNdGAN effectively bridges the existing gap between simulated and biological data benchmarks of GRN inference algorithms, providing gold standard ground truth GRNs and realistic cells corresponding to the biological

system of interest. Our results show that GRouNdGAN is a stable, realistic, and effective simulator with various applications in single-cell RNA-seq analysis.

Introduction

Unraveling gene regulatory interactions plays a crucial role in studying biological processes under different conditions[1-3], simulating knockdown and knockout experiments[4,5], and identifying therapeutic drug targets[6,7]. Gene regulatory interactions can be depicted using gene regulatory networks (GRNs), often modeled as directed acyclic graphs (DAGs) with genes as nodes and regulatory interactions as directed edges. Many algorithms have been proposed in the literature for gene regulatory network reconstruction using bulk or single-cell gene expression data; either alone[8-14] or in combination with other data modalities[15-18]. These algorithmic developments in GRN inference have been facilitated by technological advances in high-throughput single-cell RNA sequencing (scRNA-seq), which enabled the measurement of transcriptomic profiles in individual cells, leading to a wealth of large, high-resolution single-cell gene expression datasets. The prevalence of such datasets has provided great insights at the cellular level in areas such as gene regulation[19,20], perturbation experiments[21,22], cell identification[23], and trajectory inference[24,25], and have opened the door for the use of data-intensive modeling techniques such as deep learning[26,27].

While these recent advances have provided great biological insights, evaluating the performance of GRN Inference algorithms remains challenging[28,29]. This is due to the lack of reliable ground truth for the biological processes under study. Existing evaluation approaches often resort to

curated databases, such as GTRD[30], ESCAPE[31], and KEGG[32], which aggregate known regulatory interactions from the literature utilizing sources such as ChIP-seq datasets, loss-of-function/gain-of-function experiments, motif enrichment analysis and text mining. While this approach sheds some light on known interactions, the regulatory interactions found in these databases are typically aggregated from a wide range of datasets and may not be specific to the biological system being studied, given the cell-type and context specificity of gene regulation. An alternative strategy is to verify regulatory interactions by conducting perturbation experiments on the system under study or a system that closely replicates its conditions[29]. This approach, however, is tedious, lengthy, and expensive. Another approach is to employ scRNA-seq simulation tools. However, most simulators lack the essential properties for this task, such as the preservation of gene identities and generation based on a user-provided ground truth GRN. For example, simulators, such as scGAN[26], cscGAN[26], scDESIGN2[33], and SPARSIM[34], can capture inter-gene dependencies and correlations in their datasets. As such, they are capable of *implicitly* modeling regulatory relations of their reference datasets. However, since these models do not explicitly impose a known GRN (that can act as the ground truth) in generating their datasets, they are not suitable for benchmarking GRN inference methods.

A small subset of existing simulators, such as BoolODE[28], SERGIO[4], and GeneNetWeaver (GNW)[35], explicitly incorporate GRNs that underlie causal TF-gene dynamics. GNW was used to generate the gold standard networks for Dialogue on Reverse Engineering Assessment and Methods (DREAM) network inference challenges[35-37]. However, being a bulk RNA-seq tool, its simulated datasets do not replicate the distribution of experimental scRNA-seq datasets nor exhibit their

statistical properties despite attempts to adapt the method for single-cell generation through externally inducing dropout events[10,38]. SERGIO and BoolODE, on the other hand, were specifically designed to simulate scRNA-seq data based on a ground truth GRN using stochastic differential equations (SDEs)[4,28]. Both tools have been used to benchmark a variety of GRN inference algorithms. However, often a mismatch between the GRN benchmarking results obtained based on experimental and simulated data have been reported. For example, in BEELINE's benchmarking results (using BoolODE), some of the top ranking GRN inference tools on simulated datasets reported near random performance on experimental datasets[28]. The observed mismatch in GRN inference results can be attributed to the poor resemblance between simulated and real datasets. Both SERGIO and BoolODE enable the user to generate more realistic simulated datasets by carefully selecting the values of the SDE model parameters. In addition, using a reference dataset, SERGIO allows the user to fine-tune the added technical noise in an iterative procedure until the generated and reference datasets are statistically matched[4]. While these steps can help improve the resemblance to real datasets, they can often be suboptimal and pose an undue burden on the end-user: for example, in SERGIO, the user must evaluate the similarity based on five different statistics and change three parameters iteratively until desired resemblance is achieved (which itself can be subjective). In addition, since the causal GRN is imposed in the "clean" dataset, but technical noise is added afterwards, this step may affect and change the encoded causal relationships in a non-trivial way (which may explain why the authors observed that the performance of GRN inference methods dropped close to random when applied to the noisy dataset[4]). Furthermore, SERGIO and BoolODE do not readily preserve gene identities and make simplifying assumptions regarding cooperative regulation. In SERGIO, the

88 combinatorial effect of multiple regulatory TFs is merely the sum of their individual effects.
89 BoolODE allows for more complex coregulatory patterns by requiring the user to define Boolean
90 functions relating a gene to its regulators as a truth table. This results in only 2^R TF state
91 combinations for a gene with R regulators. These shortcomings demonstrate the need for
92 simulation tools that are capable of generating scRNA-seq data that better resemble real
93 datasets, while retaining the underlying gene regulatory dynamics specified by a user-define
94 GRN.
95
96 We present GRouNdGAN (GRN-guided in silico simulation of single-cell RNA-seq data using Causal
97 generative adversarial networks), a causal implicit generative model for the GRN-guided
98 simulation of single-cell RNA-seq data inspired by CausalGAN[39]. Given an input user-specified
99 GRN and a reference scRNA-seq dataset, GRouNdGAN can be trained to generate simulated
100 scRNA-seq data that is both indistinguishable from real scRNA-seq datasets and faithful to the
101 causal regulatory interactions defined in the input GRN. Unlike model-based simulators that rely
102 on simplifying assumptions regarding co-regulatory patterns, in GRouNdGAN these patterns are
103 *learned* through complex functions. This allows it not to compromise on the underlying
104 complexity of the system and to model elaborate regulatory dynamics. Using three experimental
105 scRNA-seq datasets, we show that GRouNdGAN provides state-of-the-art performance in scRNA-
106 seq realistic data generation, while preserving gene identities, causal gene regulatory
107 interactions, and cellular dynamics (e.g., lineage trajectory and pseudo-time ordering). This is
108 achieved through implicit parameterization without the need for manual fine-tuning. Since causal
109 TF-gene regulatory relationships defined by the input GRN are enforced in GRouNdGAN's

generated data, GRouNdGAN can be used for benchmarking GRN inference algorithms with the input GRN as a ground truth. Using GRouNdGAN, we benchmark seven GRN inference methods on our simulated datasets and find the results to be aligned with BEELINE's experimental results. Furthermore, the causal structure of GRouNdGAN enables it to be used for sampling from both interventional and observational data distributions. Using this property, we use GRouNdGAN to simulate knockout experiments, creating cells under conditions that were not originally observed in the data. Our results show the potential for GRouNdGAN to bridge the gap between existing simulated and experimental datasets.

Results

GRouNdGAN generates simulated single-cell RNA-seq data using causal generative adversarial networks

GRouNdGAN is a deep learning model that generates simulated scRNA-seq data while imposing a user-defined causal GRN to describe the regulatory relationships of the genes and transcription factors (TFs). The architecture of GRouNdGAN builds on the causal generative adversarial network (CausalGAN)[39] and includes a causal controller, several target generators, a critic, a labeler and an anti-labeler (Figure 1), all implemented as separately parameterized neural networks.

Training is composed of two steps. First, the causal controller that generates TF expression values is pre-trained (Figure 1A). This is achieved by incorporating the causal controller as the generator of a Wasserstein GAN (WGAN)[40] that generates simulated samples resembling samples of a reference dataset (Methods). In the second step (Figure 1B), TF expression values (generated by the pre-trained causal controller) along with randomly generated noise values are provided as input to the target generators. Target generators produce

expression values of target genes of the TFs, while incorporating the dependency of TF-gene relationships in the causal GRN. To achieve this, each target generator only accepts as input a noise value along with the generated expression values of TFs that causally regulate it (Figure 1B, Methods). The generated expression values of genes and TFs are then fed to a library-size normalization (LSN) layer[26].

The GRouNdGAN's critic's role is to quantify the Wasserstein distance between the reference and simulated data distributions. The target generators are trained in an adversarial manner to generate such realistic datapoints that are indistinguishable from the experimental datapoints by the critic. The labeler and the anti-labeler estimate the causal controller's TF expression values *only* from the target genes' expressions and ensure that the generated target gene expressions are encoded with their regulators' information. The labeler is trained using both real and generated values, while the anti-labeler is trained only from the values generated by the target generators. The details of this model and training procedure are provided in Methods, the architectural choices are provided in Supplementary Tables S1 and S2, and an ablation study is provided in Supplementary File S1 and Table S3.

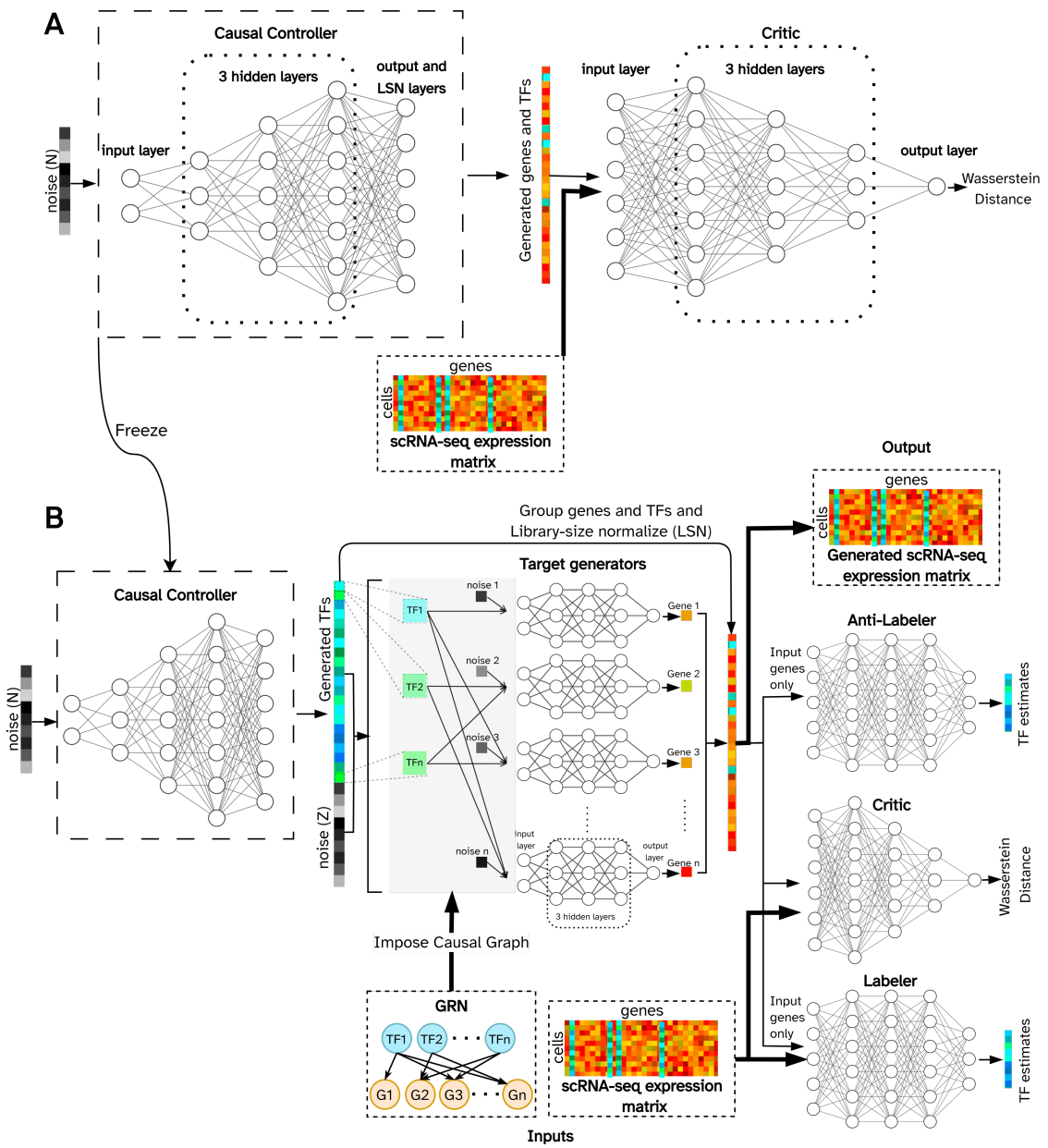

Figure 1: Architecture and training procedure of GRouNdGAN. A) A WGAN is pre-trained to generate realistic simulated cells. B) The LSN layer of the generator of the trained WGAN (panel A) is removed, its weights are frozen, and is used as the causal controller to generate unnormalized TF expression values. These TF expression values along with a noise vector are provided as input to the target generators, following the provided causal GRN. The generated gene and TF expression values are reorganized and passed through the LSN layer. The normalized simulated expression vectors and experimental reference data are then passed to the critic to estimate Wasserstein distance between the reference and the generated data distributions. The anti-labeler estimates TF values based on generated target gene expressions. The labeler performs a similar task, but in addition to receiving generated values, it also utilizes target gene expression values from the reference data. Labeler and anti-labeler ensure that the causal GRN is incorporated by the target generators. Details of the model are provided in Methods.

161

162 **GRouNdGAN generates realistic scRNA-seq data**

163 We used GRouNdGAN to generate data based on three reference datasets. The first dataset
164 contained scRAN-seq profiles of 68579 human peripheral blood mononuclear cells (PBMCs)
165 represented by UMI counts from 10x Genomics[41] corresponding to eleven cell types (henceforth
166 "PBMC-All"). We formed a dataset corresponding to the most common cell type in PBMC-All
167 dataset (CD8+ Cytotoxic T-cell) containing scRAN-seq profile of 20773 cells (henceforth "PBMC-
168 CTL"). Additionally, we obtained the scRNA-seq profile of 2730 cells corresponding to
169 differentiation of hematopoietic stem cells to different lineages from mouse bone marrow
170 b(measured using MARS-seq)[20] (henceforth "BoneMarrow"). For each dataset, we trained
171 GRouNdGAN on a randomly selected training set and evaluated on a held-out test set (Methods).
172 Figures 2A-2B and Supplementary Figures S1-S3 show the t-distributed stochastic neighbor
173 embedding (t-SNE) plots of real (experimental) and simulated cells, qualitatively revealing their
174 similarity.

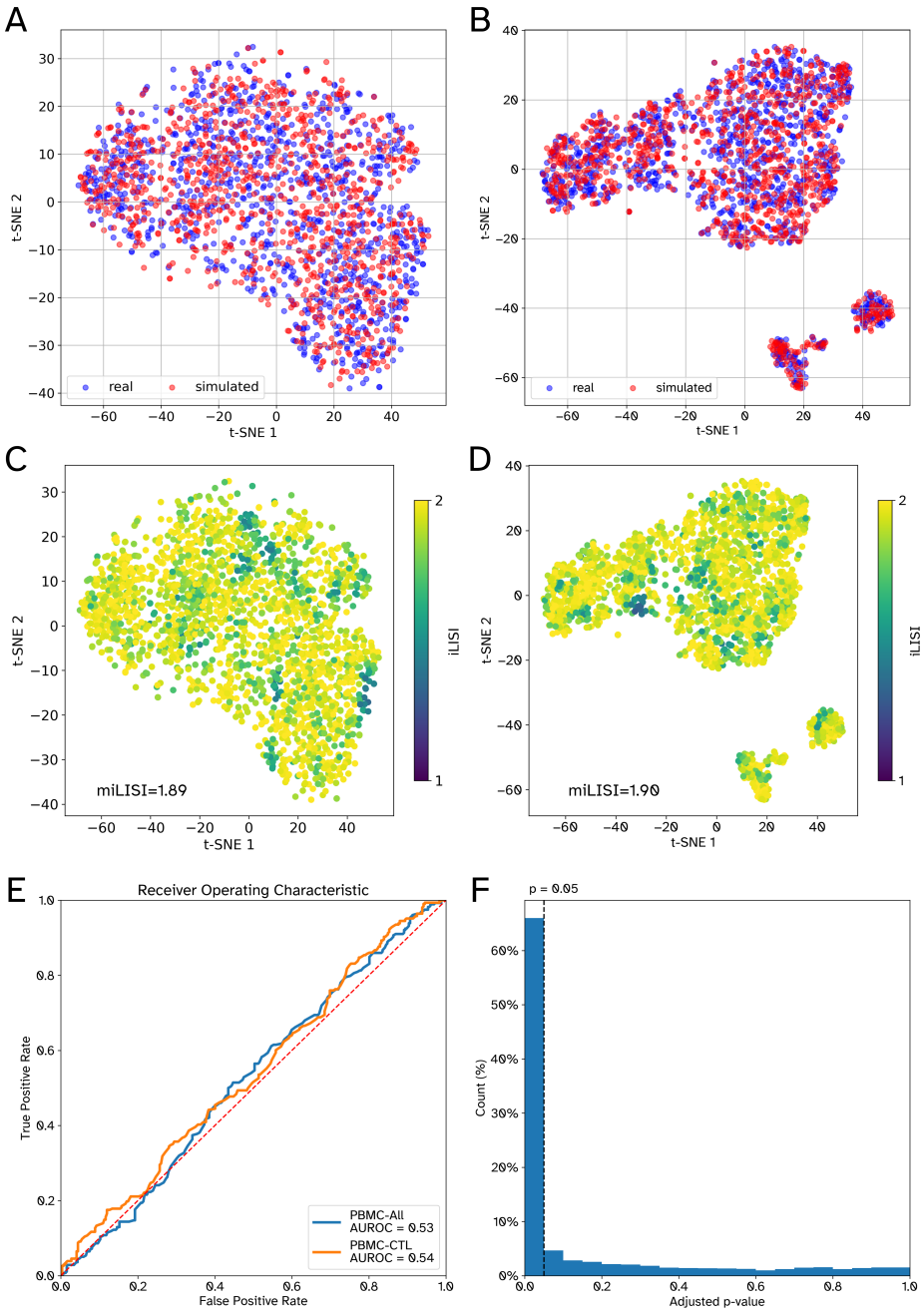

Figure 2: Performance of GRouNdGAN in generating realistic scRNA-seq data. All plots correspond to 1000 real (experimental) held-out test cells and 1000 simulated cells. Each gene in the imposed GRN of GRouNdGAN is regulated by 15 TFs (constructed using GRNBoost2 from the experimental training set). Panels A and B depict the t-SNE plots of real (red) and simulated (blue) cells for PBMC-CTL and PBMC-All, respectively, while panels C and D also show their iLISI distributions. E) The ROC curve of a Random Forest classifier in distinguishing between simulated and real cells. F) The distribution of adjusted p-values corresponding to the TF perturbation study on the PBMC-CTL dataset. The p-values are obtained using two-sided Wilcoxon signed ranked tests and are adjusted for multiple hypotheses following the Benjamini-Hochberg procedure.

To quantitatively assess the resemblance of simulated and experimental datasets, we used five metrics including Euclidean distance, Cosine distance, maximum mean discrepancy (MMD)[42], mean integration local inverse Simpson's index (miLISI)[43], and the area under the receiver operating characteristic curve (AUROC) of a random forests (RF) classifier trying to distinguish simulated and experimental cells (Methods). As a reference ("control"), we also calculated these metrics using two halves of the experimental test set. Table 1 shows the performance of GRouNdGAN, control, and three state-of-the-art simulators as baselines for the PBMC-CTL dataset (see Supplementary Table S4 for training set performance). We should note that to assess data quality systematically and fairly, we only included methods that automatically generate scRNA-seq data resembling experimental data, since the quality of data generated by methods that rely on the input of the user for distribution matching (e.g., SERGIO or BoolODE) is user dependent and may be subjective.

Table 1: Performance of different simulators in generating realistic scRNA-seq data using the PBMC-CTL dataset. The metrics are calculated between a simulated dataset of 1000 cells and the held-out test set of 1000 real cells (see Supplementary Table S4 for training set performance). Each gene in the imposed GRN of GRouNdGAN is regulated by 15 TFs (constructed using GRNBoost2 from the experimental training set). For the first three metrics, a value closer to zero is preferred, for RF AUROC a value closer to 0.5 is preferred, and for miLISI a value closer to 2 is preferred. Best performance values (excluding control) are in bold-face. The control shows the value of the metrics comparing the two halves of the real dataset. For the first two metrics, the values correspond to the distance of the mean centroids of the real and simulated cells.

Simulator	Cosine distance	Euclidean distance	MMD	RF AUROC	miLISI
GRouNdGAN	**0.00057**	**182**	**0.026**	**0.54**	**1.891**
scGAN[26]	0.00095	222	0.031	0.59	1.888
scDESIGN2[33]	0.00100	229	0.065	0.76	1.736
SPARsim[34]	0.00104	235	0.309	0.95	1.625
Control	0.00019	99	0.012	0.50	1.909

209 Next, we sought to determine whether GRouNdGAN can also generate samples corresponding
210 to different cell types. If cell type information of the experimental dataset is available a priori,
211 one can easily generate realistic samples of each cell type separately using GRouNdGAN (similar
212 to the PBMC-CTL analysis). However, we asked whether our model could generate realistic
213 samples *without* such prior knowledge, a much more challenging task. This is particularly useful
214 for scenarios in which such information is unavailable. In addition, not requiring cell type
215 information reduces the amount of user involvement (to annotate cell types) and improves the
216 usability of the model. We repeated the analysis above using the PBMC-All dataset, containing
217 eleven cell types. Although GRouNdGAN did not receive cell type/cluster information, we did not
218 observe any sign of mode collapse (Supplementary Figure S2). In fact, it was able to generate
219 cells from distinct clusters of this dataset (Figure 2B) with a test set miLISI score of 1.90.
220 GRouNdGAN outperformed all simulators that did not use any extra cell cluster labels or ratios
221 (Table 2). When compared to four simulators that utilized this side information, it was still the
222 top performing simulator based on MMD and RF AUROC and was the second best performing
223 model according to the Euclidean distance (only 6% higher than the best performing model) and
224 miLISI (only 0.5% lower than the best performing model) (Table 2).

225

226 **Table 2:** Performance of different simulators in generating realistic scRNA-seq data using the
227 PBMC-All dataset. Baseline models that enable utilization of cell cluster labels or ratios are run
228 with and without this information. The metrics are calculated between a simulated dataset of
229 1000 cells and the held-out test set of 1000 real cells. Each gene in the imposed GRN of
230 GRouNdGAN is regulated by 15 TFs (constructed using GRNBoost2 from the experimental training
231 set). For the first three metrics, a value closer to zero is preferred, for RF AUROC a value closer
232 to 0.5 is preferred, and for miLISI a value closer to 2 is preferred. Best performance values
233 (excluding control) are in bold-face. The control shows the value of the metrics comparing the
234 two halves of the real dataset. For the first two metrics, the values correspond to the distance of
235 the mean centroids of the real and simulated cells.

236

Simulator	Cell cluster labels/ratios provided as input	Cosine distance	Euclidean distance	MMD	RF AUROC	miLISI
GRouNdGAN	No	0.00028	168	**0.017**	**0.53**	1.90
cscGAN[26]	Yes	0.00025	242	0.030	0.56	**1.91**
cWGAN	Yes	0.00053	239	0.027	0.57	1.89
scGAN[26]	No	0.00081	300	0.035	0.58	1.90
scDESIGN2[33]	Yes	0.00032	181	0.046	0.86	1.81
scDESIGN2[33]	No	0.00057	256	0.124	0.91	1.53
SPARsim[34]	Yes	**0.00019**	**158**	0.287	0.96	1.66
SPARsim[34]	No	0.00035	245	0.307	0.96	1.39
Positive control	NA	0.00022	116	0.012	0.50	1.91

237

238 We repeated the analysis above using the BoneMarrow dataset, which includes continuous
239 transitional cell states and has a much smaller number of cells compared to the other two
240 datasets. GRouNdGAN outperformed all other simulators based on all five metrics
241 (Supplementary Table S4). These results show that GRouNdGAN is a stable simulator of scRNA-
242 seq data and even though it does not utilize extra information regarding the cell types present in
243 these heterogenous datasets, it automatically generates realistic simulated samples reflecting
244 different cell types without facing mode collapse. (See Supplementary File S1 and Table S5 for
245 further stability analysis and the effect of different GRN properties on the performance). Most
246 importantly, unlike the aforementioned simulators it incorporates of a causal GRN in the data.

247

248 **GRouNdGAN imposes a causal GRN in the simulated data**

249 Since one of the main motivations of this work was to impose causal GRNs in the simulated data,
250 we set out to systematically assess this ability. For this purpose, we performed *in-silico* TF

perturbation (knockout) experiments on the simulated cells using the PBMC-CTL dataset. Since our goal was to assess whether the expression of the target genes of the TFs (in the imposed GRN) change as a result of these perturbations, we ensured that the perturbations were performed on the same batch of simulated cells, forming matched case/control experiments. We achieved this by performing a forward pass after setting the expression of each TF to zero (one at a time) in the causal controller output, while keeping all other parameters and TF expressions unchanged. There was no change in the expression of genes that did not have an incoming edge from the knocked-out TF (as expected), while the expression of the genes regulated by the knocked-out TF changed. We performed two-sided Wilcoxon signed-rank tests comparing the expression of a gene across all cells before and after the knockout of one of its regulating TFs. Figure 2F shows the distribution of the adjusted p-values (Benjamini-Hochberg) for each TF-gene edge in the GRN. Based on these analyses, in the majority (66.5%) of cases, the knockout of the regulating TF of a gene significantly (adjusted p-value < 0.05) altered the expression of the gene, showcasing that the GRN is indeed imposed in the expression profiles of simulated cells. Note that since each gene is regulated by multiple TFs, we cannot expect that knockout of a single TF in all cases result in a significant change in the expression of its target genes.

The imposed GRN can be reconstructed from the simulated data

To test whether the imposed GRN can be reconstructed from the generated data, we first applied GRNBoost2[8] to the reference PBMC-CTL dataset and recorded the top ten TFs for each gene. Then for each gene, we created two disjoint sets of five TFs based on the even/odd parity of TF indices in the ranked list: TFs ranked as 1st, 3rd, 5th, 7th, and 9th were connected to the gene and

273 were imposed by GRouNdGAN ("positive control GRN"), while the remaining five TFs were not
274 imposed ("negative control GRN"), resulting in two GRNs with identical densities and the same
275 number of TFs and target genes. When considering the positive and negative control GRNs as
276 ground truths (separately), the GRN reconstructed using GRNBoost2 from the experimental
277 scRNA-seq dataset resulted in comparable AUPRC (0.28 and 0.24, respectively, Supplementary
278 Table S6), reflecting that the two GRNs contain edges of comparable aggregated importance.
279
280 Next, we simulated data using GRouNdGAN by imposing only the positive control GRN, and used
281 GRNBoost2, PIDC[10] GENIE3[9], and PPCOR[44] to reconstruct the underlying GRN (Figure 3,
282 Supplementary Figure S4, and Supplementary Table S6). All GRN inference methods could
283 reconstruct the imposed edges with a much higher AUPRC compared to that of the original
284 reference dataset, while they did not assign high scores to the unimposed edges, resulting in
285 close to random AUPRC. Figure 3 shows the performance of GRNBoost2 when applied to the
286 experimental data, to data simulated by GRouNdGAN and to data simulated by scGAN. The
287 performance for scGAN (for both GRNs) is lower than the reference (experimental) dataset,
288 showcasing that it has not preserved the TF-gene relationships of the experimental dataset well.
289 The performance is relatively consistent between positive (Figures 3A and 3C) and negative
290 control (Figures 3B and 3D) GRNs using the reference and the scGAN-generated data; however,
291 the performance is much higher for the positive control GRN (compared to negative control GRN)
292 when using GRouNdGAN-simulated data. Similar patterns were observed when we repeated
293 these analyses using the BoneMarrow dataset (Supplementary Table S6 and Figure S5). These

results show that the imposed edges were accentuated by GRouNdGAN, while the unimposed edges were disrupted and could not be found by GRN inference methods.

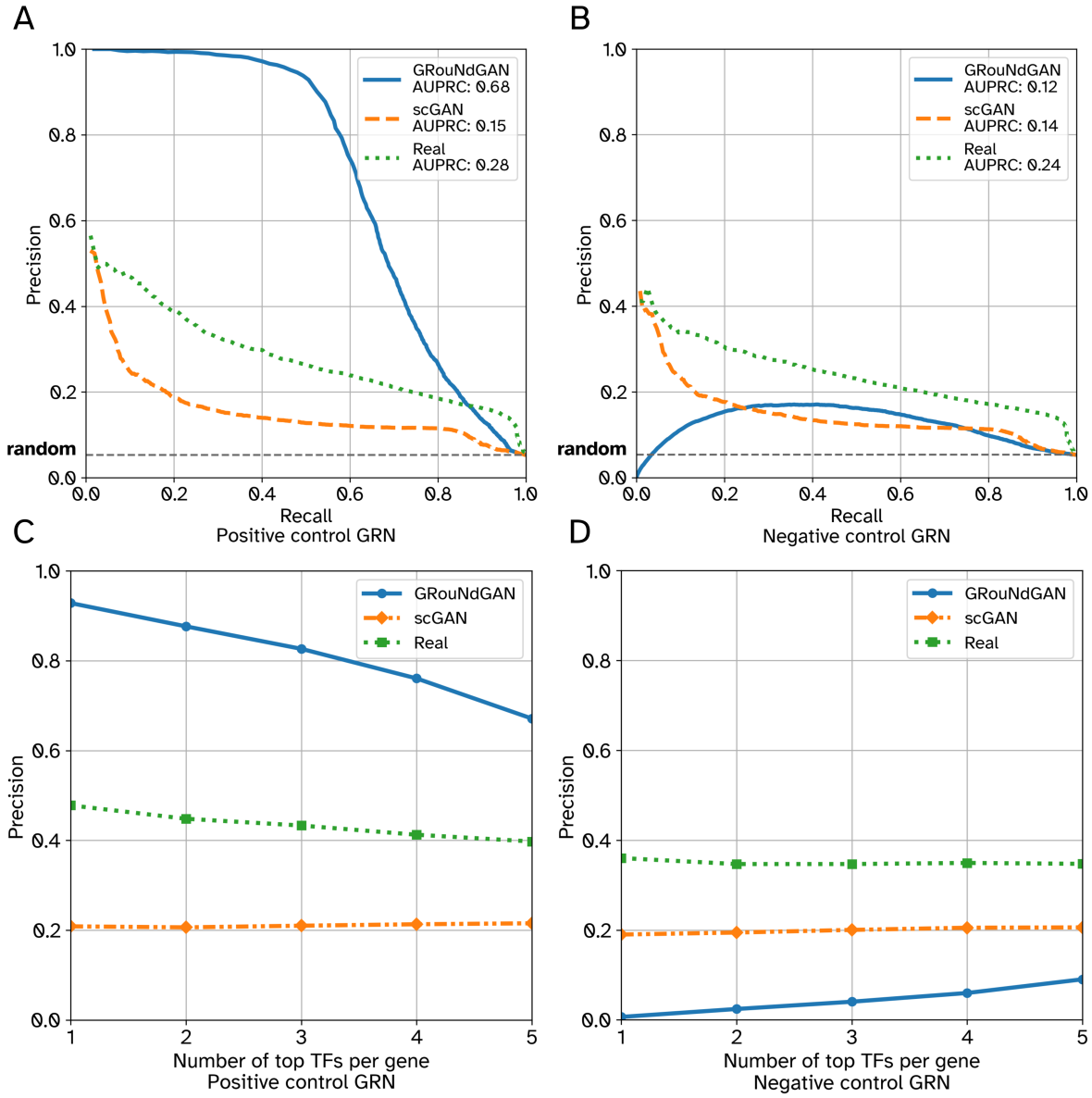

Figure 3: Performance of GRNBoost2 in recovering the imposed edges versus unimposed edges (PBMC-CTL). Left column shows the AUPRC (A) and Precision at k (per gene) (C) when the imposed edges (positive control GRN) were considered the ground truth. Right column shows the AUPRC (B) and Precision at k (per gene) (D) when the unimposed edges (negative control GRN) were considered the ground truth. Precision at k (per gene) refers to the precision when top k TFs for each gene is used to form the reconstructed GRN.

304 Next, we asked whether GRNBoost2 can reconstruct the imposed GRN, if GRNs of different
305 densities are used to simulate data. Using simulated data in which each gene was regulated by
306 15, 10, 5, and 3 TFs, we observed that in all cases the GRN imposed by GRouNdGAN could be
307 inferred from the simulated data (Supplementary Table S6). Finally, we repeated the controlled
308 analysis above, swapping the role of positive and negative control GRNs. Similar to the results
309 observed in Figure 3, the imposed edges were inferred by GRNBoost2, while the unimposed
310 edges could not be accurately discovered (Supplementary Table S6).
311
312 These results show that 1) GRouNdGAN imposes the causal GRN in the simulated data, 2) the
313 GRN inference methods can identify the imposed edges, and 3) the TF-gene relationships that
314 originally existed in the real experimental data but were not imposed by GRouNdGAN are
315 disrupted during simulation. The last property is particularly desirable for benchmarking GRN
316 inference methods to ensure that the regulatory relationships present in the input reference
317 dataset (but not imposed) do not bias the simulated data, which could result in an inflated false
318 positive rate.
319
320 **GRouNdGAN-simulated data preserves the trajectory of single cells and can be used for**
321 **pseudo-time inference**
322 In addition to deciphering discrete cell types, scRNA-seq data is often used to determine
323 continuous cell transitions during biological processes such as differentiation, using trajectory
324 inference and pseudo-time ordering of cells[24,25,45-49]. We sought to determine whether the
325 simulated data generated by GRouNdGAN conforms to the transitional states and pseudo-time

326 ordering of the experimental scRNA-seq data. This is particularly important, since one of our goals
327 was to benchmark GRN inference methods, some of which utilize pseudo-time information[12-14,50-
328 54]. For this purpose, we used the BoneMarrow dataset corresponding to differentiation of
329 hematopoietic stem cells to different lineages.
330
331 We used Partition-based graph abstraction (PAGA)[24] for trajectory inference and analysis of the
332 experimental and GRouNdGAN-generated data. PAGA computes graph-like maps of data
333 manifolds faithful to the global topology of single-cell data, retaining both continuous and
334 discrete structures of the data. Supplementary Figure S6 shows the PAGA graphs, in which nodes
335 capture discrete states in the differentiation process and the edges signify transitions between
336 them. To compare the trajectories identified using the original BoneMarrow dataset with those
337 identified using simulated data, we used the gene expression of markers of cell types present in
338 this dataset (Supplementary Table S7)[24]. Figure 4 shows the gene expression patterns of two
339 marker genes for erythroid cells (*Gata1, Klf1*), neutrophils (*Mpo, Ctsg*), and basophils (*Mcpt8,*
340 *Prss34*) for PAGA trajectory graphs of both datasets (See Supplementary Figures S7-S13 for
341 marker genes of all cell types). The comparison reveals similar activation patterns between
342 corresponding regions (nodes) of PAGA graphs of the experimental and GRouNdGAN data,
343 revealing several trajectories (e.g., erythroid and neutrophil branches). Strikingly, even the
344 number of nodes activated in each figure scales similarly between the two datasets: for example,
345 in Figures 4E and 4F the two markers of basophils are consistently highly expressed in only one
346 node in each dataset.
347

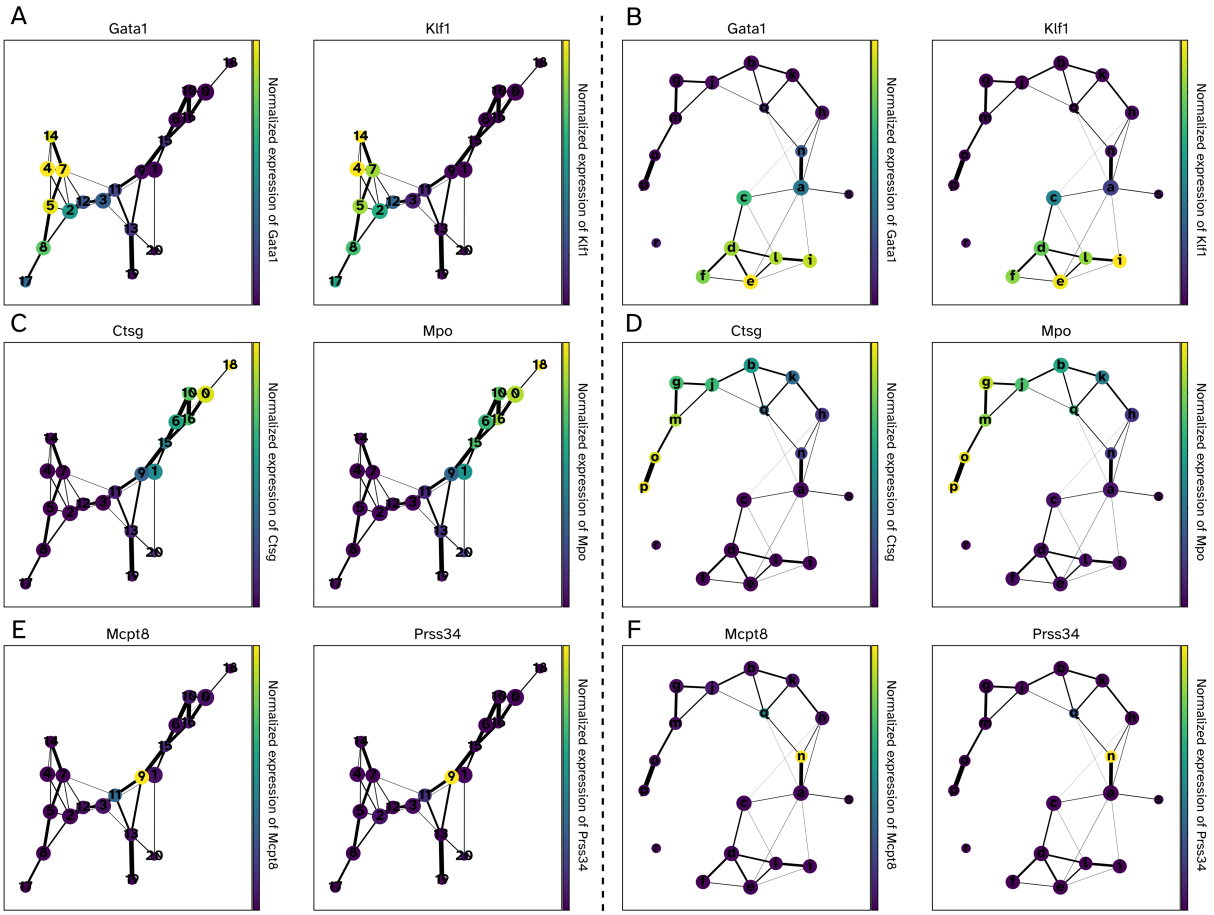

Figure 4: PAGA graphs and cell type marker gene activation patterns in experimental and simulated BoneMarrow dataset. In each figure, nodes represent Louvain clusters (capturing discrete states), edges represent transitions between states, and edge weights indicate confidence in the presence of connections. Low-connectivity edges below a threshold of 0.01 were discarded. Left panels (A, C, E) show the PAGA graph computed on GRouNdGAN-generated cells, and right panels (B, D, F) show the PAGA graph for the original BoneMarrow dataset. Node colors represent normalized expression of marker genes for erythroid cells (A, B), neutrophils (C, D), and basophils (E, F).

Next, we annotated different cell types present in both datasets using these marker genes (following PAGA's official tutorial) (Figures 5A and 5B). The GRouNdGAN's annotated graph shows similar topological features to that of the experimental data and captures known biological properties of hematopoiesis, which were also noted by Wolf, et al.[24] using the experimental dataset (e.g., proximity between monocytes and neutrophils or association between

megakaryocyte and erythroid progenitors). These observations show that the key characteristics of the experimental data are retained in GRouNdGAN's generated data.

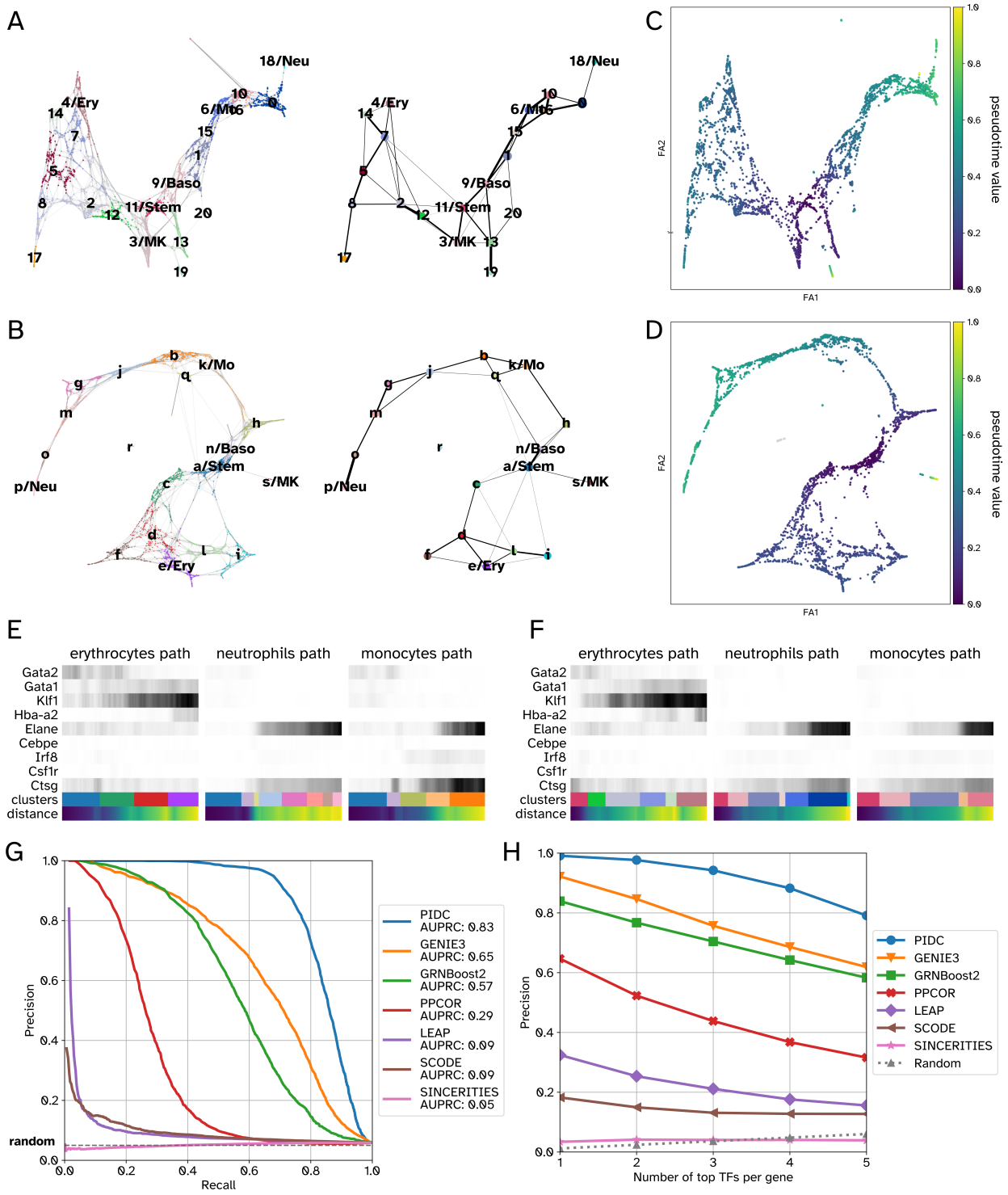

Figure 5: Consistency in trajectories and pseudo-temporal orderings between data generated by GRouNdGAN and the experimental BoneMarrow hematopoietic dataset. Panels A and B show

368 the cell type annotations of simulated and experimental data, respectively. The figures on the
369 left show the force-directed graphs (obtained using ForceAtlas2[55]) and the figures on the right
370 show the PAGA graphs. Panels C and D show the PAGA-initialized force-directed graphs with cells
371 colored by their inferred pseudo-time value. The following abbreviations are used: Stem for stem
372 cells, Ery for erythroid cells, Neu for neutrophils, Mo for monocytes, MK for Megakaryocytes, and
373 Baso for Basophils. Panels E and F display expression changes in the marker genes of erythrocyte
374 (*Gata2*, *Gata1*, *Klf1*, and *Hba-a2*), neutrophil (*Elane* and *Cebpe*), and monocyte (*Irf8*, *Csf1r*, and
375 *Ctsg*) branches for experimental (E) and simulated data (F). The distance refers to the geodesic
376 distance from a root cell (here a stem cell). A darker shade of grey shows higher expression.
377 Panels (G, H) show the performance of different GRN inference methods on data generated by
378 GRouNdGAN using the BoneMarrow dataset.
379

380 We used the diffusion pseudotime algorithm[45] (in scanpy[56]) to infer the progression of cells
381 through geodesic distance (Figures 5C and 5D). We noticed strong concordance when examining
382 pseudo-time values assigned to cell subgroups in the experimental and GRouNdGAN-generated
383 data. To track these patterns, we focused on marker gene expression changes along erythroid,
384 neutrophil, and monocyte trajectories (Supplementary Figures S7-S9), similar to the analysis
385 performed by Wolf, et al.[24]. In both datasets, the activation of neutrophils' marker, *Elane,* and
386 monocytes' marker, *Irf8,* were predominantly observed towards the later stages of their
387 trajectories (corresponding to larger pseudo-time values) (Figures 5E and 5F). Additionally, we
388 observed the activation of erythroid maturity marker genes *Gata2*, *Gata1*, *Klf1*, and *Hba-a2*
389 roughly in sequential order along the erythroid trajectory in both datasets (Figures 5E and 5F),
390 concordant with previous findings[24]. The striking similarity between the activation patterns of
391 these marker genes in both datasets and the preservation of their ordering highlights
392 GRouNdGAN's ability to capture dynamic transcriptional properties of single cell data, leading to
393 correct trajectory inference and pseudo-time ordering.
394

395 **Benchmarking GRN inference methods using GRouNdGAN confirms prior insights obtained**
396 **from curated experimental datasets**

397 A recent GRN inference benchmarking study, BEELINE[28], found conflicting performance results
398 when testing methods on curated biological datasets and synthetically generated ones: top two
399 performing models on synthetic data where among the bottom three performing ones on
400 curated data. We sought to re-investigate this pattern using data generated from GRouNdGAN
401 with the BoneMarrow dataset and the positive and negative control GRNs explained earlier. We
402 used seven GRN inference methods used by BEELINE including GRNBoost2[8], GENIE3[9], PIDC[10],
403 PPCOR[44], LEAP[12], SCODE[13], and SINCERITIES[14], capturing a wide range of methods with a broad
404 spread of performances reported on synthetic and curated data in the BEELINE study. We
405 observed a striking resemblance between the performance pattern (order) of the methods on
406 GRouNdGAN-simulated data and curated biological data from BEELINE (Figure 5G-5H and
407 Supplementary Table S6 versus Figure 4 of BEELINE[28]): the only difference was that the order of
408 LEAP[12] and SCODE[13] were swapped in our analysis. We also repeated this analysis using the
409 PBMC-CTL dataset (Supplementary Table S6), obtaining similar patterns.

410

411 Overall, PIDC[10] outperformed all methods using both datasets, followed by GENIE3[9] and
412 GRNBoost2[8]. LEAP[12], SCODE[13] and SINCERITIES[14] (which use pseudo-time and could only be
413 applied to the BoneMarrow dataset) performed worse than other methods, a behavior that was
414 also observed in the BEELINE study on curated biological data (but not on synthetic data). All
415 methods performed close to random on unimposed negative control GRN edges. Overall, these
416 results not only show that GRouNdGAN can be used for benchmarking GRN inference methods,

417 but also shows that the insights obtained from it matches those obtained from curated biological
418 datasets, the formation of which requires extensive amount of work and effort.
419
420 ***In-silico* perturbation experiments using GRouNdGAN**
421 In many studies, one is interested to characterize the relationship between TFs' expression and
422 phenotypic labels (e.g., cell types). Differential expression (DE) analysis is a common approach to
423 identify TFs (or other genes) most associated with a specific cell type. However, to directly test
424 whether perturbing the expression of such candidate TFs change the expression of genes
425 representative of a cell type, perturbation experiments (e.g., TF knockout) are desired. Since
426 GRouNdGAN is a causal implicit generative model, it provides the capability to sample from
427 interventional distributions, thereby enabling *in-silico* perturbation experiments. GRouNdGAN
428 provides this functionality at inference time by enabling manipulation of TFs' expressions as the
429 outputs of the causal controller. This change is propagated along the GRN, impacting the
430 expression of genes regulated by the perturbed TF(s). By enabling a deterministic mode of
431 operation, GRouNdGAN allows for exact comparison of gene expressions before and after
432 perturbation while maintaining the invariance of other critical parameters such as input noises,
433 LSN layer per-cell scaling factor, and unperturbed TFs (Methods).
434
435 To test this functionality, we asked whether the knockout of top 3 TFs most differentially
436 expressed (Mann Whitney U test) between each cell type and the rest would result in less
437 simulated cells being generated in the vicinity of the cells of that type. If the answer is positive,
438 this implies that the perturbation in the TFs' expression has modified the gene expression profile

439 of generated cells so that they no longer resemble that particular cell type (confirming our
440 expectation). Figure 6A shows the UMAP embedding of the experimental dataset (miLISI = 1.94
441 when compared with simulated data) and Figure 6B shows the density plots of the experimental
442 and simulated data, revealing a high degree of resemblance. We focused on the changes in the
443 iLISI values of experimentally profiled cells of a specific type, when jointly mapped to the same
444 embedding space with simulated data, before and after knockout. In order for the changes in the
445 iLISI values to be observable, it was necessary to focus on cell types that occupy relatively distinct
446 regions of the embedding space; otherwise, the presence of other cell types in that region would
447 confound the results. We identified four cell types (CD14+ monocytes, CD19+ B cells, Dendritic
448 cells, CD56+ natural killer cells) that occupied relatively distinct regions of the embedding space
449 (Supplementary Figure S14). Figure 6C shows the distribution of iLISI values for each cell type
450 (only for real cells), calculated along with unperturbed simulated cells (blue) and with perturbed
451 simulated cells (orange). Figure 6D-6E show the iLISI values and the distribution of generated
452 cells after the knockout experiment for CD19+ B cells as an example (see Supplementary Figure
453 S15-S18 for other cell types). In all cases, the iLISI values significantly reduced after the knockout
454 experiment (Figure 6C, Supplementary Table S8), showcasing that after the knockout of top TFs
455 associated with a cell type, GrouNdGAN generated much fewer cells resembling that cell type
456 (even though it did not have the knowledge of the cell type annotations). Instead, the simulated
457 cells were dispersed into other cell types, yet they retained meaningful positions within the
458 overall dataset embedding space. In addition, the miLISI of the other cell types remained
459 relatively unchanged (Supplementary Table S8). This analysis shows that *in-silico* TF perturbation

experiments using GrouNdGAN produces results concordant with results directly obtained from real biological data.

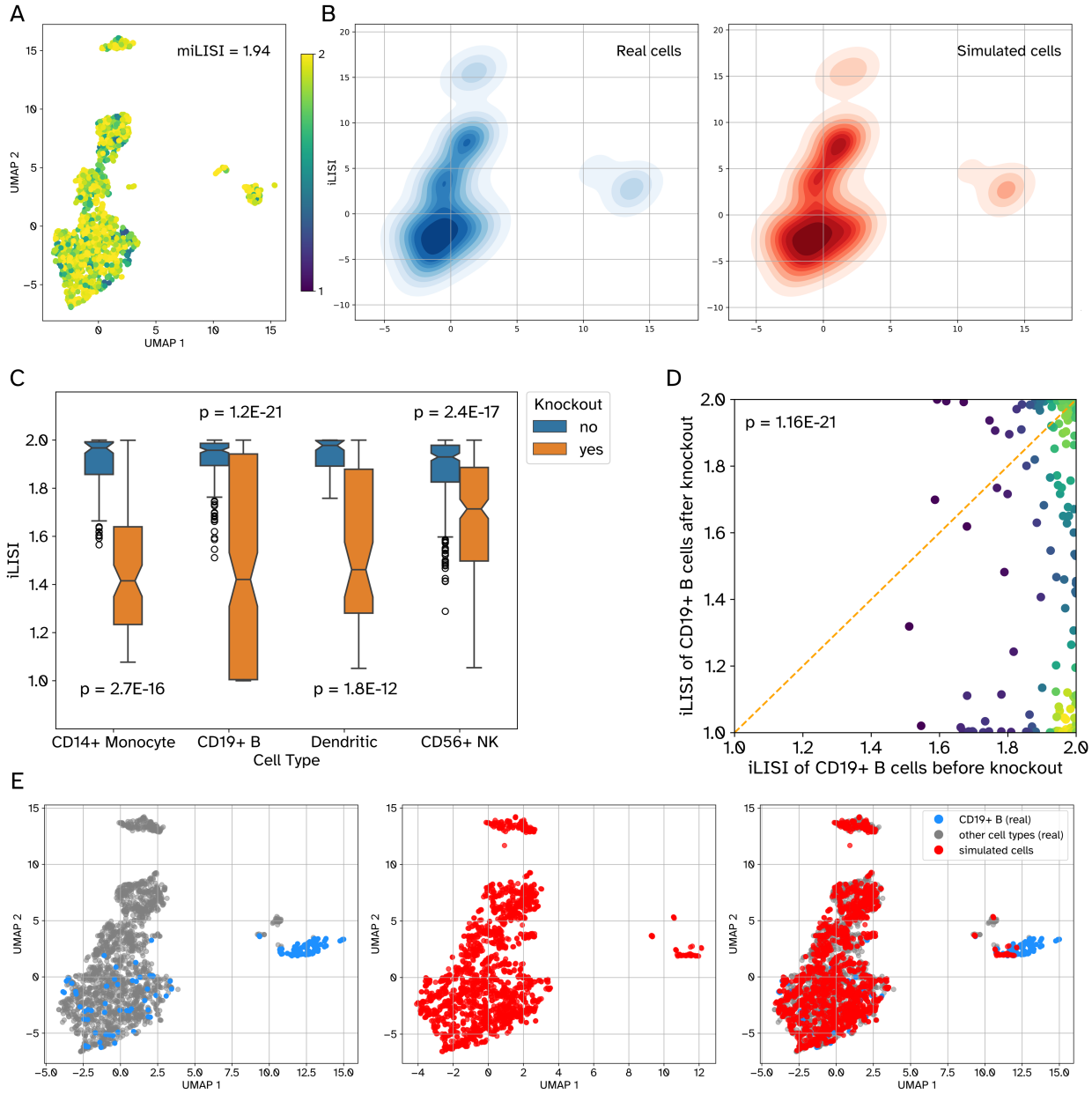

Figure 6: TF knockout experiments and their effect on cell type generation. A) The UMAP shows the distribution of 2000 randomly selected cells from the experimental PBMC-All dataset. The color shows the iLISI value of each cell (miLISI = 1.94) calculated from a UMAP embedding, jointly obtained from the experimental cells and the same number of GRouNdGAN-generated cells. TFs were omitted as features when generating UMAP plots. In the simulated data, each gene was regulated by 15 TFs, identified using GRNBoost2. B) Each plot shows the density of cells in the

embedding space: the left plot (blue) corresponds to the real cells and the right plot (red) shows the simulated cells. C) The boxplots show the distribution of iLISI values of real cells, calculated along with unperturbed simulated cells (blue) and with perturbed simulated cells (orange). For each cell type, top three most differentially expressed TFs were knocked out. The p-values reported in this figure were calculated using one-sided Wilcoxon signed rank tests. D) The scatter plot shows the iLISI values of CD19+ B cells calculated along with unperturbed simulated cells (x-axis) and along with perturbed simulated cells (y-axis). The circles correspond to real cells and their colors reflect the density of datapoints in that region. E) The UMAP embedding of real cells (left), simulated cells after knockout of top TFs of CD19+ B cells (middle), and all together (right).

Discussion

GRouNdGAN is a causal implicit generative model designed to simulate realistic scRNA-seq datasets based on a user-defined GRN and a reference dataset. By incorporating GRN connections in its architecture and including auxiliary tasks, it imposes causal TF-gene relationships in the gene expression profile of simulated cells. These causal relationships are verifiable by *in-silico* knockout experiments and identifiable by GRN inference methods. Using three experimental datasets, we demonstrated that GRouNdGAN achieves state-of-the-art performance on various tasks including realistic scRNA-seq data generation, GRN inference benchmarking, and performing *in-silico* knockout experiments. Even when applied to heterogenous datasets, it generated realistic datasets (with no sign of mode collapse) without utilizing information regarding cell types/clusters, achieving best or second-best performance compared to models that used this information. We should note that it is trivially possible to use cell type/cluster information with GRouNdGAN: one can simply provide subsets of the experimental dataset corresponding to a cell type/cluster as a reference, one at a time. However, the fact that this model does not require this information makes it easier to use and broadens its applicability. We also showed that GRouNdGAN-generated data preserves complex cellular

495 dynamics and patterns present in the reference dataset such as lineage trajectories, pseudo-time
496 orderings, and patterns of cell type markers' activation.
497
498 GRouNdGAN imposes user-defined causal TF-gene regulatory interactions, while excluding
499 regulatory interactions that are present in the reference dataset, but not in the input GRN. This
500 property makes GRouNdGAN-simulated datasets ideal for benchmarking GRN inference
501 algorithms with the user-specified GRN serving as the ground truth. Building on this, we
502 benchmarked seven popular GRN inference algorithms and showed that GRouNdGAN
503 benchmarking results highly coincided with results reported by BEELINE[28] on experimental
504 datasets. This highlights GRouNdGAN's ability to bridge the gap between GRN benchmarking
505 results obtained on simulated and real datasets without requiring excessive user manipulation.
506 Since GRouNdGAN generates data based on a structural causal model, it can be used to sample
507 synthetic cells from both observational and interventional distributions. Using this property, we
508 conducted *in-silico* perturbation experiments by knocking out key regulating TFs in various cell
509 types and observed that the targeted cell-states were suppressed in the generated dataset.
510
511 GRouNdGAN maintains key advantages over existing tools that incorporate GRNs in scRNA-seq
512 data simulation such as SERGIO and BoolODE. First, during training, GRouNdGAN's model
513 parameters are optimized to minimize any mismatch between real and simulated data
514 distributions. This enables it to replicate scRNA-seq data characteristics, such as technical and
515 biological noise, without the need for any user-intervention or further post-processing. In
516 contrast, SERGIO simulates data by first generating a "clean" dataset based on SDEs. Then, the

517 user would iteratively fine-tune technical noise parameters, such as dropout rate, library size,
518 and outlier genes, until the simulated and reference datasets are statistically matched. This data-
519 matching procedure puts extra burden on the end-user and might lead the user to overly rely on
520 fine-tuning technical noise parameters, instead of SDE parameters. Over relying on added
521 technical noise to achieve data matching can potentially corrupt the data, rendering the
522 underlying regulatory patterns unobservable in the final simulated dataset. This might explain
523 the near-random GRN inference performance reported by SERGIO on their noisy datasets[4].
524 Another advantage of GRouNdGAN is gene identity preservation. GRouNdGAN is trained such
525 that each of its generated genes or TFs mimic the expression patterns of a corresponding gene
526 or TF in the reference dataset. In contrast, BoolODE and SERGIO simulate pseudo-genes whose
527 identities cannot be readily mapped to their counterparts in the reference dataset. Gene identity
528 preservation can be attained in SERGIO or BoolODE by fine-tuning the SDE model parameters
529 until the expression patterns of a given generated gene aligns with a corresponding gene in the
530 reference dataset. This, however, adds further burden to the end-user. Finally, GRouNdGAN
531 learns complex non-linear co-regulatory patterns from the data, instead of relying on simplifying
532 assumptions or user input. In contrast, SERGIO assumes that the combined effect of multiple
533 regulatory TFs is simply the sum of their individual effects. Similarly, BoolODE requires the user
534 to provide a truth table specifying combinatory TF co-regulation rules for each target gene. In
535 contrast, GRouNdGAN's neural network-based generator enables it to implicitly learn complex
536 non-linear co-regulatory patterns that may represent co-expression, co-inhibition, or co-
537 activation of individual genes. This is achieved during the training of GRouNdGAN without the
538 need for user intervention.

Several key considerations need to be taken into account when using GRouNdGAN. First, our analyses showed that the architectural and hyperparameter choices of the presented model result in good performance across different datasets. We performed hyperparameter fine-tuning on one dataset and used the same set of values for all three datasets and for different choices of GRNs, obtaining state-of-the-art performance. However, when trained on a completely new dataset, it is still advisable to test different choices of the hyperparameters using a validation set to obtain the best performance. Second, the default architecture of the model includes LSN layers and, as such, the model generates library-size normalized gene expression values. Our ablation study (Supplementary File S1) showed that without the LSN layer, GRouNdGAN can still generate realistic profiles; however, including this layer improves how well the GRN is imposed and stabilizes the training procedure. Third, an important consideration for any causal inference problem is that given an observational dataset, its underlying causal graph is not unique. When benchmarking GRN inference methods, this problem exists regardless of how the ground truth GRN is obtained (even using knockout/knockdown experiments), and a GRN inference algorithm may find a plausible causal GRN, yet not the one that actually generated the data. Despite this, our analysis using the positive and negative control GRNs showed that GRN inference methods could detect the imposed edges while ignoring others that were unimposed but whose patterns originally existed in the reference dataset. Based on this observation, we posit that the imposed GRN by GRouNdGAN, while not the only causal GRN capable of generating the simulated dataset, is the most probable one. However, verifying this requires theoretical analyses that are well beyond the scope of this study. Finally, although GRouNdGAN can generate data with any GRN,

561 the quality of the data deteriorates if the imposed TF-gene relationships are significantly different
562 from those observable in the reference dataset. The reason is that in such a scenario, generating
563 realistic simulated datapoints and imposing the GRN act as contradictory requirements. This is a
564 fundamental problem that is not unique to GRouNdGAN and is applicable to any simulator with
565 these two requirements. On the other hand, simulators that do not try to impose a causal graph
566 (e.g., scGAN) can focus on generating realistic scRNA-seq samples, even at the expense of
567 disrupting some of the interactions present in the reference dataset (as we observed in Figure
568 3 with scGAN). As a result, for GRN inference method benchmarking, we recommend first finding
569 a set of regulatory edges from the experimental dataset (e.g., using GRNBoost2) and imposing
570 those in the simulated data. This ensures that the imposed edges are encoded in the simulated
571 data, the unimposed (or undetected) edges are disrupted (as shown in our analyses), and the
572 simulated dataset resembles the reference dataset.
573
574 The existing GRouNdGAN model only allows imposing a GRN in the form of a bipartite graph and
575 does not support regulation of TFs by other TFs, an assumption that we plan to relax with the
576 future versions of the model. Moreover, we intend to augment the model's architecture in the
577 future to enable imposing a causal graph with three layers of nodes capturing TF-gene-phenotype
578 relationships. This is particularly useful for development and evaluation of phenotype-relevant
579 GRNs, a concept that we have introduced in the past[11]. Additionally, a conditional version of
580 GRouNdGAN can be developed, allowing the user to generate cells of specific cell-type or on a
581 specific stage along a lineage trajectory (i.e., by conditioning on pseudo-time). Finally, by

582 incorporating DAG structural learning[57], we intend to enable GRouNdGAN to infer a GRN while
583 simultaneously learning to mimic the reference scRNA-seq dataset.
584
585 **Conclusion**
586 Overall, GRouNdGAN is a powerful and user-friendly tool that enables researchers to simulate
587 realistic scRNA-seq datasets, benchmark GRN inference algorithms, and perform *in-silico*
588 perturbation experiments, with many new features that will be added in the future.
589
590 **Methods**
591 **Datasets and preprocessing**
592 We downloaded the human peripheral blood mononuclear cell (PBMC Donor A) dataset
593 containing the single-cell gene expression profiles of 68579 PBMCs represented by UMI counts
594 from the 10x Genomics (https://support.10xgenomics.com/single-cell-gene-
595 expression/datasets/1.1.0/fresh_68k_pbmc_donor_a). This dataset contains a large number of
596 well-annotated cells and has been used by other models to generate synthetic scRNA-seq
597 samples[26]. We also downloaded the scRNA-seq profile of 2730 cells corresponding to
598 differentiation of hematopoietic stem cells to different lineages from mouse bone marrow[20]
599 ("BoneMarrow" dataset) from Gene Expression Omnibus (accession number: GSE72857)
600
601 We followed the pre-processing steps of scGAN and cscGAN[26] using scanpy version 1.8.2[56]. In
602 each dataset, cells with nonzero counts in less than ten genes were removed. Similarly, genes
603 that only had nonzero counts in less than three cells were discarded. Next, library-size

604 normalization was performed on the counts per cell with a library size equal to 20,000 in order
605 to be consistent with previous studies [26]. Next, top 1000 highly variable genes were selected using
606 the dispersion-based method described by Satija et al.[58].
607
608 **GRouNdGAN's model architecture**
609 GRouNdGAN consists of 5 components, each implemented using separately parameterized
610 neural networks: a causal controller, target generators, a critic, a labeler, and an anti-labeler
611 (Figure 1B).
612
613 *Causal controller*: The role of causal controller is to generate the expression of TFs that causally
614 control the expression of their target genes based on a user-defined gene regulatory network
615 (Figure 1B). To achieve this, it is first pre-trained (Figure 1A) as the generator of a Wasserstein
616 GAN (WGAN). GANs are a class of deep learning models that can learn to simulate non-parametric
617 distributions[59]. They typically involve simultaneously training a generative model (called the
618 "generator") that produces new samples from noise, and its adversary, a discriminative model
619 that tries to distinguish between real and generated samples (called the "discriminator"). The
620 generator's goal is to generate samples so realistic that the discriminator cannot determine
621 whether it is real or simulated (an accuracy value close to 0.5). Through adversarial training, the
622 generator and discriminator receive feedback, allowing them to co-evolve in a symbiotic manner.
623
624 The main difference between a WGAN and a traditional GAN is that in the former, a Wasserstein
625 distance is used to quantify the similarity between the probability distribution of the real data

and the generator's produced data (instead of Kullback–Leibler or Jensen–Shannon divergences). Wasserstein distance has been shown to stabilize the training of WGAN without causing mode collapse[40]. The detailed formulation of the Wasserstein distance used as the loss function in this study is provided in Supplementary File S1. In addition, instead of a discriminator, WGAN uses a "critic" that estimates the Wasserstein distance between real and generated data distributions. In our model, we added a gradient penalty term for the critic (proposed by Gulrajani et al.[60] as an alternative to weight clipping used in the original WGAN) in order to overcome vanishing/exploding gradients and capacity underuse issues.

In the pretraining step, we trained a WGAN with a generator (containing an input layer, three hidden layers, and an output layer), a library-size normalization (LSN) layer[26], and a critic (containing an input layer, three hidden layers, and an output node). A noise vector of length 128, with independent and identically distributed elements following a standard Gaussian distribution, was used as the input to the generator. The output of the generator was then fed into the LSN layer to generate the gene and TF expression values. The details of hyperparameters and architectural choices of this WGAN are provided in Supplementary Table S1. Although we were only interested in generating expression of TFs using the generator of this WGAN (in the second step of pipeline), the model was trained using all genes and TFs to properly enforce the library-size normalization. Once trained, we discarded the critic and the LSN layer, froze the weights of the generator and used it as the "causal controller"[39] to generate expression of TFs (Figure 1B).

648 *Target generators*: The role of target generators is to generate the expression of genes causally
649 regulated by TFs based on the topology of a GRN. Consider a target gene Gj regulated by a set of
650 TFs: $\{TF_1, TF_2, \dots, TF_n\}$. Under the causal sufficiency assumption and as a result of the manner
651 by which TFs' expressions are generated from independent noise variables, we can write $E_{Gj} =$
652 $f_{Gj}(E_{TF1}, E_{TF2}, \dots, E_{TFn}, N_{Gj})$ and $E_{TFi} = f_{TFi}(N_{TFi})$ for $i = 1, 2, \dots, n$, where E represents
653 expression, N represents a noise variable, and f represents a function (to be approximated using
654 neural networks). All noise variables are jointly independent. Following the theoretical and
655 empirical results of CausalGAN[39], we can use feedforward neural networks to represent functions
656 f by making the generator inherit its neural network connections from the causal GRN. To
657 achieve this, we generate each gene in the GRN by a separate generator such that target gene
658 generators do not share any neural connections (Figure 1B).
659
660 As input, the generator of each gene accepts a vector formed by concatenating a noise variable
661 and a vector of non-library size normalized TF expressions from the causal controller,
662 corresponding to the TFs that regulate the gene in the imposed GRN (i.e., its parents in the graph).
663 The expression values of TFs and the generated values of target genes are arranged into a vector,
664 which is passed to an LSN layer for normalization. We used target generators with three hidden
665 layers of equal width. The width of the hidden layers of a generator is dynamically set as twice
666 the number of its regulators (the noise variable and the set of regulating TFs). If the imposed GRN
667 is relatively dense and contains more than 5000 edges, we set the depth to 2 and the width
668 multiplier to 1 to be able to train on a single GPU. The details of hyperparameters and
669 architectural choices are provided in Supplementary Table S2.

In practice, generating each target gene's expression using separate neural networks introduces excess overhead. This is because in every forward pass, all target genes' expressions must be first generated before collectively being sent into the LSN layer. As a result, instead of parallelizing the generation of each target gene's expression (which due to the bottleneck above does not provide a significant computational benefit), we implemented target generators using a single large sparse network. This allows us to reduce the overhead and the training time to train the model on a single GPU, and to benefit from GPU's large matrix multiplication. We mask weights and gradients to follow the causal graph, while keeping the generation of genes independent from each other. From a logical standpoint, our implementation has the same architecture described earlier, but is significantly more computationally efficient.

Critic: Similar to a traditional WGAN, the objective of GRouNdGAN's critic (Figure 1B) is to estimate the Wasserstein distance between real and generated data distributions. We used the same critic architecture as the WGAN trained in the first stage.

Labeler and anti-labeler: Although the main role of the target generators is to produce realistic cells to confuse the critic, it is crucial that they rely on the TFs' expression (in addition to noise) in doing so. One potential risk is that the target generators disregard the expression of TFs and solely rely on the noise variables. This is particularly probable when the imposed GRN does not conform to the underlying gene expression programs of real cells in the reference dataset; in

691 such a scenario, and to make realistically looking simulated cells, it is more convenient for a
692 WGAN to simply ignore the strict constraints of the GRN and solely rely on noise.
693
694 To overcome this issue, we used auxiliary tasks and neural networks known as "labeler" and
695 "anti-labeler"[39]. The task of these two networks is to estimate the causal controller's TF
696 expressions (here called labels) from the target genes' expressions alone (by minimizing the
697 squared L2 norm between each element's TF estimates and their true value). This resembles the
698 idea behind an autoencoder and ensures that the model will not disregard the expression of TFs
699 in generating the expression of their target genes. The anti-labeler is trained solely based on the
700 outputs of the target generators, while the labeler utilizes both the outputs of target generators
701 and the expression of real cells (Figure 1B). They are both implemented as fully connected
702 networks with a width of 2000 and a depth of 3 and optimized using the AMSGrad[61] algorithm.
703 Each layer, except the last one, utilizes a ReLU activation function and batch normalization. In
704 addition to WGAN losses, we add labeler and anti-labeler losses to the generator to minimize
705 both. This is different from the approach used in CausalGAN, where the anti-labeler's loss is
706 maximized in the early training stages to ensure that the generator doesn't fall into label-
707 conditioned mode collapse. In GRouNdGAN, the causal controller is pretrained and generates
708 continuous labels (TFs expression) and does not face a similar issue. As a result, we instead
709 minimized the loss of the anti-labeler from the beginning and as such the labeler and anti-labeler
710 both act as auxiliary tasks to ensure the generated gene expression values take advantage of TFs
711 expression.
712

713 **Training procedure and hyperparameter tuning**

714 We follow a two-step training procedure comprising of training two separate WGANs (Figure 1)
715 to train GRouNdGAN. The generators and critics in both GANs are implemented as fully
716 connected neural networks with rectified linear unit (ReLU) activation functions in each layer,
717 except for the last layer of the critic. The weights were initialized using He initialization[62] for layers
718 containing ReLU activation and Xavier initialization[63] for other layers (containing linear
719 activations). We used Batch Normalization[64] to normalize layer inputs for each training
720 minibatch, except for the critic. This is since using it in the critic invalidates the gradient penalty's
721 objective, as it penalizes the norm of the critic's gradient with respect to the entire batch rather
722 than to inputs independently. An LSN layer[26] was used in both WGANs (Figure 1A and 1B) to scale
723 counts in each simulated cell to make it consistent with the library size of input reference dataset.
724 This normalization results in a dramatic decrease in convergence time and smooths training by
725 mitigating the inherent heterogeneity of scRNA-seq data.

726

727 GRouNdGAN solves a min-max game between the generator (f_g) and the critic (f_c), with the
728 following objective function:

729 $$\min_{f_g} \max_{\|f_c\|_L \leq 1} \mathbb{E}_{x \sim \mathbb{P}_r} f_c(x) - \mathbb{E}_{x \sim f_g(\mathbb{P}_{noise})} f_c(x). \quad (1)$$

730 We alternated between minimizing the generator loss for one iteration and maximizing the
731 generator loss for five iterations. We employed the AMSGrad[61] optimizer with the weight decay
732 parameters $\beta_1 = 0.5$, $\beta_2 = 0.9$ and employed an exponentially decaying learning rate for the
733 optimizer of both the critic and generator.

734

735 The hyperparameters were tuned using a validation set consisting of 1000 cells from the PBMC-
736 CTL dataset (Supplementary Tables S1-S2) based on the Euclidean distance and the RF AUROC
737 score, which were consistently in accord. The same hyperparameters were used for all other
738 analyses and the other two datasets (PBMC-All and BoneMarrow).
739
740 **Causal GRN preparation**
741 This section describes the creation of the causal graph inputted to GRouNdGAN used to impose
742 a causal structure on the model. GRouNdGAN accepts a GRN in the form of a bipartite directed
743 acyclic graph (DAG) as input, representing the relationship between TFs and their target genes.
744 In this study, we created the causal graph using the 1000 most highly variable genes identified in
745 the preprocessing step. First, the set of TFs among the highly variable genes were identified based
746 on the AnimalTFDB3.0 database[65] and a GRN was inferred using GRNBoost2[8] (with the list of TFs
747 provided). Supplementary Table S5 shows the characteristics of the different GRNs used in this
748 study.
749
750 **Evaluation of the resemblance of real and simulated cells**
751 We evaluated all models using held-out test sets containing randomly selected cells from each
752 reference dataset (1000 cells from PBMC-CTL and PBMC-All and 500 cells from BoneMarrow). To
753 quantify the similarity between real and generated cells, we employed various metrics. For each
754 cell represented as a datapoint in a low dimensional embedding (e.g., t-SNE or UMAP), the
755 integration local inverse Simpson's Index (iLISI)[43] captures the effective number of datatypes (real
756 or simulated) to which datapoints of its local neighborhood belong based on weights from a

Gaussian kernel-based distributions of neighborhoods. The miLISI is the mean of all these scores and in our study ranges between 1 (poor mixing of real and simulated cells) and 2 (perfect mixing of real and simulated cells). Additionally, we calculated the cosine and Euclidean distances of the centroids of real cells and simulated cells, where the centroid was obtained by calculating the mean along the gene axis (across all simulated or real cells).

To estimate the proximity of high-dimensional distributions of real and simulated cells without creating centroids, we used the maximum mean discrepancy (MMD)[42]. Given two probability distributions p and q and a set of independently and identically distributed (i.i.d.) samples from them, denoted by X and Y, MMD with respect to a function class $\mathcal{F}$ is defined as

$$\mathrm{MMD}[\mathcal{F}, X, Y] \coloneqq \sup_{f \in \mathcal{F}} \left(\mathbb{E}_x[f(x)] - \mathbb{E}_y[f(y)] \right), \tag{2}$$

where sup refers to supremum and $\mathbb{E}$ denotes expectation. When the MMD function class $\mathcal{F}$ is a unit ball in a reproducing kernel Hilbert space (RKHS) $\mathcal{H}$ with kernel k, the population MMD takes a zero value if and only if $p = q$ and a positive unique value if $p \neq q$. The squared MMD can be written as the distance of mean embeddings μ_p, μ_q of distributions p and q, which can be expressed in terms of kernel functions:

$$MMD^2[\mathcal{F}, p, q] = \left\| \mu_p - \mu_q \right\|_{\mathcal{H}}^2 \tag{3}$$

$$= \mathbb{E}_{x,x'}[k(x, x')] - 2\mathbb{E}_{x,y}[k(x, y)] + \mathbb{E}_{y,y'}[k(y, y')].$$

Following existing implementations of MMD in the single-cell domain[26,66], we chose a kernel that is the sum of three Gaussian kernels to increase sensitivity of the kernel to a wider range:

$$k(x, y) = \sum_i \left(\frac{\|x - y\|^2}{\sigma_i^2} \right), i = 1,2,3, \tag{4}$$

778 where σ_i denote standard deviations and were chosen to be the median of the average distance
779 between a point to its 25 nearest neighbors divided by factors of 0.5, 1, and 2 in the three kernels,
780 respectively.
781
782 We also used a random forests (RF) classifier and used its area under the receiver operating
783 characteristic (AUROC) curve to determine whether the real and simulated cells can be
784 distinguished from each other. Consistent with previous studies[26,67], we first performed a
785 dimensionality reduction using principal component analysis (PCA) and used the top 50 PCs of
786 each cell as the input features to the RF model, which improves the computational efficiency of
787 this analysis. The RF model was composed of 1000 trees and the Gini impurity was used to
788 measure the quality of a split.
789
790 **Baseline simulator models**
791 We compared the performance of GRouNdGAN to scDESIGN2[33], SPARsim[34], and three GAN-
792 based methods: scGAN[26], cscGAN with projection-based conditioning[26], and a conditional WGAN
793 (cWGAN). The cWGAN method conditions by concatenation following the cGAN framework[68].
794 More specifically, it concatenates a one-hot encoded vector (representing the cluster number or
795 cell type) to the noise vector input to the generator and cells forwarded to the discriminator. We
796 did not train the cWGAN or cscGAN on the PBMC-CTL dataset, since it contains only one cell type.
797 For the PBMC-All and the BoneMarrow dataset, we trained all models above. Additionally, we
798 simulated data using scDESIGN2 and SPARsim with and without cell cluster information, as they
799 allow providing such side information in their training.

To train models that utilized cell cluster information, we performed Louvain clustering and provided the cluster information and ratio of cells per cluster during training. Clustering was done by following the cell ranger pipeline[41], based on the raw unprocessed dataset (and independent of the pre-processing steps described earlier for training simulators). First, genes with no UMI count in any of the cells were removed. Then the gene expression profile of each cell was normalized by the total UMI of all (remaining) genes, and highly variable genes were identified. The gene expression profile of each cell was then re-normalized by the total UMI of retained highly variable genes, and each gene vector (representing its expression across different cells) was z-score normalized. Given the normalized gene expression matrix above, we found top 50 principal components (PCs) using PCA analysis. These PCs were then used to compute a neighborhood graph with a local neighborhood size of 15, which was used in Louvain clustering. We ran the Louvain algorithm with a resolution of 0.15.

For SPARSim, we set all sample library sizes to 20000 and estimated gene expression level intensities and simulation parameters by providing it with both raw and normalized count matrices. When cell cluster information was provided, distinct SPARSim simulation parameters were estimated per cell for each cluster. scDESIGN2 accepts input matrices where entries are integer count values; we thus performed rounding on the expression matrix before fitting scDESIGN2. With cluster information provided, a scDESIGN2 model was fit separately for cells of each cluster, and similar to conditional GANs, the ratio of cells per cluster was provided to the method.

822

823 ***In-silico* perturbation experiments using GroundGAN**

824 To perform perturbation experiments using GroundGAN, we put the trained model in a
825 deterministic mode of operation. This is necessary to ensure that the perturbation experiments
826 are performed on the same batch (i.e., replicate) of generated cells to form matched case/control
827 experiments. To do this, we performed a forward pass through the generator and then saved the
828 input noise to the causal controller, the input noise to the target generators, the TF expression
829 values generated by the causal controller, and the per-cell scaling factor of the LSN layer.
830 Subsequent passes through the generators used the saved parameters so that ensuing runs
831 always output the same batch of cells (instead of generating new unmatched cells).

832

833 **Trajectory inference and pseudo-time analysis**

834 Following official PAGA tutorial for the BoneMarrow dataset
835 (https://github.com/scverse/scanpy-tutorials/blob/master/paga-paul15.ipynb), we used
836 (Partition-based graph abstraction) PAGA[24] for trajectory inference and analysis. We built force-
837 directed graphs[69] (with ForceAtlas2[55]) using the top 20 principal components of the data (using
838 principal component analysis or PCA) and a neighborhood graph of observations computed using
839 UMAP (to estimate connectivities). We next denoised the graph by representing it in the diffusion
840 map space and computed distances and neighbors as before using this new representation. After
841 denoising, we then ran the Louvain clustering algorithm with a resolution of 0.6. Finally, we ran
842 the PAGA algorithm on the identified clusters and used the obtained graph to initialize and
843 rebuild the force-directed graph.

844

845 **GRN inference methods**

846 In our GRN benchmarking analysis, we focused on seven GRN inference algorithms: GENIE3[9],
847 GRNBoost2[8], PPCOR[44], PIDC[10], LEAP[12], SCODE[13] and SINCERITIES[14], which were used in the
848 BEELINE study[28]. Of these methods, LEAP requires pseudo-time ordering of cells, while SCODE
849 and SINCERITIES require both pseudo-time ordering and pseudo-time values. Since not all
850 algorithms inferred the edge directionality or its sign (activatory or inhibitory nature), we did not
851 consider these factors in our analysis to be consistent among different models. We ran these
852 models as docker containers using the docker images provided by BEELINE's GitHub
853 (https://github.com/Murali-group/Beeline)[28] with the default parameters used in BEELINE. To
854 benchmark algorithms requiring pseudo-time ordering of cells, we computed the pseudo-times
855 of GRouNdGAN-simulated data (based on the BoneMarrow dataset) using PAGA[24] and diffusion
856 pseudotime[45], following the methodology described earlier. In the GRN inference benchmark
857 analysis, we did not provide the list of TFs to GRNBoost2 to make it consistent with other GRN
858 inference methods.

859

860 **Competing interests**

861 The authors have no conflict of interest to declare.

862

863 **Funding**

864 This work was supported by grants from Natural Sciences and Engineering Research Council of
865 Canada (NSERC) [RGPIN-2019-04460] (AE), Government of Canada's New Frontiers in Research

Fund (NFRF) [NFRFE-2019-01290] (AE), Canada Foundation for Innovation (CFI) JELF [project 40781], and McGill Initiative in Computational Medicine (AE). This research was enabled in part by support provided by the Digital Research Alliance of Canada (https://alliancecan.ca) and Compute Canada (www.computecanada.ca).

Data availability

The PMBC dataset is available from the 10x genomics repository (corresponding to healthy donor A) from the link https://support.10xgenomics.com/single-cell-gene-expression/datasets/1.1.0/fresh_68k_pbmc_donor_a?. The BoneMarrow dataset is available in the Gene Expression Omnibus repository under accession number GSE72857.

Code availability

Our implementation and evaluation of GRouNdGAN in Python 3.9.6 using the PyTorch framework[70] along with instructions is freely available under the GNU Affero General Public License v3.0 on GitHub (https://github.com/Emad-COMBINE-lab/GRouNdGAN). The repository also contains our PyTorch implementation of scGAN, projection-based conditioning cscGAN[26] and a variation of cscGAN that uses conditioning by concatenation (cWGAN).

References

1 Lee, T. I. *et al.* Transcriptional regulatory networks in Saccharomyces cerevisiae. *science* **298**, 799-804 (2002).

2 Che, D. *et al.* Dynamic and modular gene regulatory networks drive the development of gametogenesis. *Briefings in bioinformatics* **18**, 712-721 (2017).

3 Olson, E. N. Gene regulatory networks in the evolution and development of the heart. *Science* **313**, 1922-1927 (2006).

4 Dibaeinia, P. & Sinha, S. SERGIO: a single-cell expression simulator guided by gene regulatory networks. *Cell systems* **11**, 252-271. e211 (2020).
5 Yang, Y. *et al.* Gene knockout inference with variational graph autoencoder learning single-cell gene regulatory networks. *Nucleic Acids Research*, gkad450 (2023).
6 Madhamshettiwar, P. B., Maetschke, S. R., Davis, M. J., Reverter, A. & Ragan, M. A. Gene regulatory network inference: evaluation and application to ovarian cancer allows the prioritization of drug targets. *Genome medicine* **4**, 1-16 (2012).
7 Manuel, A. M., Dai, Y., Jia, P., Freeman, L. A. & Zhao, Z. A gene regulatory network approach harmonizes genetic and epigenetic signals and reveals repurposable drug candidates for multiple sclerosis. *Human Molecular Genetics* **32**, 998-1009 (2023).
8 Moerman, T. *et al.* GRNBoost2 and Arboreto: efficient and scalable inference of gene regulatory networks. *Bioinformatics* **35**, 2159-2161 (2019).
9 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).
10 Chan, T. E., Stumpf, M. P. & Babtie, A. C. Gene regulatory network inference from single-cell data using multivariate information measures. *Cell systems* **5**, 251-267. e253 (2017).
11 Emad, A. & Sinha, S. Inference of phenotype-relevant transcriptional regulatory networks elucidates cancer type-specific regulatory mechanisms in a pan-cancer study. *NPJ systems biology and applications* **7**, 9 (2021).
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 Papili Gao, N., Ud-Dean, S. M., Gandrillon, O. & Gunawan, R. SINCERITIES: inferring gene regulatory networks from time-stamped single cell transcriptional expression profiles. *Bioinformatics* **34**, 258-266 (2018).
15 Aibar, S. *et al.* SCENIC: single-cell regulatory network inference and clustering. *Nature methods* **14**, 1083-1086 (2017).
16 Bravo González-Blas, C. *et al.* SCENIC+: single-cell multiomic inference of enhancers and gene regulatory networks. *Nature Methods*, 1-13 (2023).
17 Kartha, V. K. *et al.* Functional inference of gene regulation using single-cell multi-omics. *Cell genomics* **2** (2022).
18 Badia-i-Mompel, P. *et al.* Gene regulatory network inference in the era of single-cell multi-omics. *Nature Reviews Genetics*, 1-16 (2023).
19 Shalek, A. K. *et al.* Single-cell RNA-seq reveals dynamic paracrine control of cellular variation. *Nature* **510**, 363-369 (2014).
20 Paul, F. *et al.* Transcriptional heterogeneity and lineage commitment in myeloid progenitors. *Cell* **163**, 1663-1677 (2015).
21 Avey, D. *et al.* Single-cell RNA-seq uncovers a robust transcriptional response to morphine by glia. *Cell reports* **24**, 3619-3629. e3614 (2018).
22 Skinnider, M. A. *et al.* Cell type prioritization in single-cell data. *Nature biotechnology* **39**, 30-34 (2021).
23 Abdelaal, T. *et al.* A comparison of automatic cell identification methods for single-cell RNA sequencing data. *Genome biology* **20**, 1-19 (2019).
24 Wolf, F. A. *et al.* PAGA: graph abstraction reconciles clustering with trajectory inference through a topology preserving map of single cells. *Genome biology* **20**, 1-9 (2019).
25 Saelens, W., Cannoodt, R., Todorov, H. & Saeys, Y. A comparison of single-cell trajectory inference methods. *Nature biotechnology* **37**, 547-554 (2019).

26 Marouf, M. *et al.* Realistic in silico generation and augmentation of single-cell RNA-seq data using generative adversarial networks. *Nature communications* **11**, 1-12 (2020).
27 Ma, Q. & Xu, D. Deep learning shapes single-cell data analysis. *Nature Reviews Molecular Cell Biology* **23**, 303-304 (2022).
28 Pratapa, A., Jalihal, A. P., Law, J. N., Bharadwaj, A. & Murali, T. Benchmarking algorithms for gene regulatory network inference from single-cell transcriptomic data. *Nature methods* **17**, 147-154 (2020).
29 Emmert-Streib, F., Dehmer, M. & Haibe-Kains, B. Gene regulatory networks and their applications: understanding biological and medical problems in terms of networks. *Frontiers in cell and developmental biology* **2**, 38 (2014).
30 Kolmykov, S. *et al.* GTRD: an integrated view of transcription regulation. *Nucleic acids research* **49**, D104-D111 (2021).
31 Xu, H. *et al.* ESCAPE: database for integrating high-content published data collected from human and mouse embryonic stem cells. *Database* **2013**, bat045 (2013).
32 Kanehisa, M. & Goto, S. KEGG: kyoto encyclopedia of genes and genomes. *Nucleic acids research* **28**, 27-30 (2000).
33 Sun, T., Song, D., Li, W. V. & Li, J. J. scDesign2: a transparent simulator that generates high-fidelity single-cell gene expression count data with gene correlations captured. *Genome biology* **22**, 1-37 (2021).
34 Baruzzo, G., Patuzzi, I. & Di Camillo, B. SPARSim single cell: a count data simulator for scRNA-seq data. *Bioinformatics* **36**, 1468-1475 (2020).
35 Schaffter, T., Marbach, D. & Floreano, D. GeneNetWeaver: in silico benchmark generation and performance profiling of network inference methods. *Bioinformatics* **27**, 2263-2270 (2011).
36 Marbach, D. *et al.* Revealing strengths and weaknesses of methods for gene network inference. *Proceedings of the national academy of sciences* **107**, 6286-6291 (2010).
37 Marbach, D. *et al.* Wisdom of crowds for robust gene network inference. *Nature methods* **9**, 796-804 (2012).
38 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**, 1-21 (2018).
39 Kocaoglu, M., Snyder, C., Dimakis, A. G. & Vishwanath, S. Causalgan: Learning causal implicit generative models with adversarial training. *arXiv preprint arXiv:1709.02023* (2017).
40 Arjovsky, M., Chintala, S. & Bottou, L. in *International conference on machine learning.* 214-223 (PMLR).
41 Zheng, G. X. *et al.* Massively parallel digital transcriptional profiling of single cells. *Nature communications* **8**, 1-12 (2017).
42 Gretton, A., Borgwardt, K. M., Rasch, M. J., Schölkopf, B. & Smola, A. A kernel two-sample test. *The Journal of Machine Learning Research* **13**, 723-773 (2012).
43 Korsunsky, I. *et al.* Fast, sensitive and accurate integration of single-cell data with Harmony. *Nature methods* **16**, 1289-1296 (2019).
44 Kim, S. ppcor: an R package for a fast calculation to semi-partial correlation coefficients. *Communications for statistical applications and methods* **22**, 665 (2015).
45 Haghverdi, L., Büttner, M., Wolf, F. A., Buettner, F. & Theis, F. J. Diffusion pseudotime robustly reconstructs lineage branching. *Nature methods* **13**, 845-848 (2016).
46 Trapnell, C. *et al.* Pseudo-temporal ordering of individual cells reveals dynamics and regulators of cell fate decisions. *Nature biotechnology* **32**, 381 (2014).
47 Kumar, P., Tan, Y. & Cahan, P. Understanding development and stem cells using single cell-based analyses of gene expression. *Development* **144**, 17-32 (2017).

48 Street, K. *et al.* Slingshot: cell lineage and pseudotime inference for single-cell transcriptomics. *BMC genomics* **19**, 1-16 (2018).
49 Hwang, B., Lee, J. H. & Bang, D. Single-cell RNA sequencing technologies and bioinformatics pipelines. *Experimental & molecular medicine* **50**, 1-14 (2018).
50 Aubin-Frankowski, P.-C. & Vert, J.-P. Gene regulation inference from single-cell RNA-seq data with linear differential equations and velocity inference. *Bioinformatics* **36**, 4774-4780 (2020).
51 Qiu, X. *et al.* Inferring causal gene regulatory networks from coupled single-cell expression dynamics using scribe. *Cell systems* **10**, 265-274. e211 (2020).
52 Sanchez-Castillo, M., Blanco, D., Tienda-Luna, I. M., Carrion, M. & Huang, Y. A Bayesian framework for the inference of gene regulatory networks from time and pseudo-time series data. *Bioinformatics* **34**, 964-970 (2018).
53 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 systems biology* **12**, 1-7 (2018).
54 Deshpande, A., Chu, L.-F., Stewart, R. & Gitter, A. Network inference with Granger causality ensembles on single-cell transcriptomics. *Cell reports* **38**, 110333 (2022).
55 Jacomy, M., Venturini, T., Heymann, S. & Bastian, M. ForceAtlas2, a continuous graph layout algorithm for handy network visualization designed for the Gephi software. *PloS one* **9**, e98679 (2014).
56 Wolf, F. A., Angerer, P. & Theis, F. J. SCANPY: large-scale single-cell gene expression data analysis. *Genome biology* **19**, 1-5 (2018).
57 Gao, Y., Shen, L. & Xia, S.-T. in *ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP).* 3320-3324 (IEEE).
58 Satija, R., Farrell, J. A., Gennert, D., Schier, A. F. & Regev, A. Spatial reconstruction of single-cell gene expression data. *Nature biotechnology* **33**, 495-502 (2015).
59 Goodfellow, I. *et al.* Generative adversarial networks. *Communications of the ACM* **63**, 139-144 (2020).
60 Gulrajani, I., Ahmed, F., Arjovsky, M., Dumoulin, V. & Courville, A. C. Improved training of wasserstein gans. *Advances in neural information processing systems* **30** (2017).
61 Reddi, S. J., Kale, S. & Kumar, S. On the convergence of adam and beyond. *arXiv preprint arXiv:1904.09237* (2019).
62 He, K., Zhang, X., Ren, S. & Sun, J. in *Proceedings of the IEEE international conference on computer vision.* 1026-1034.
63 Glorot, X. & Bengio, Y. in *Proceedings of the thirteenth international conference on artificial intelligence and statistics.* 249-256 (JMLR Workshop and Conference Proceedings).
64 Ioffe, S. & Szegedy, C. in *International conference on machine learning.* 448-456 (PMLR).
65 Hu, H. *et al.* AnimalTFDB 3.0: a comprehensive resource for annotation and prediction of animal transcription factors. *Nucleic acids research* **47**, D33-D38 (2019).
66 Shaham, U. *et al.* Removal of batch effects using distribution-matching residual networks. *Bioinformatics* **33**, 2539-2546 (2017).
67 Shu, H. *et al.* Modeling gene regulatory networks using neural network architectures. *Nature Computational Science* **1**, 491-501 (2021).
68 Mirza, M. & Osindero, S. Conditional generative adversarial nets. *arXiv preprint arXiv:1411.1784* (2014).
69 Islam, S. *et al.* Characterization of the single-cell transcriptional landscape by highly multiplex RNA-seq. *Genome research* **21**, 1160-1167 (2011).
70 Paszke, A. *et al.* Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems* **32** (2019).