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. 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, often represented as a gene regulatory network (GRN), 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]. Many algorithms have been proposed for GRN reconstruction using bulk or single-cell RNA sequencing data (scRNA-seq), alone[8-14] or with other modalities[15-18]. While these advances have provided great biological insights, evaluating the performance of GRN Inference algorithms remains challenging[19,20] due to the lack of reliable ground truth for the biological processes under study. Existing evaluation approaches often resort to curated databases[21-23]. However, the regulatory interactions in these databases are aggregated from a wide range of datasets and are not specific to a biological system, making them not ideal benchmarks due to context specificity of gene regulation. Another strategy is to verify regulatory interactions by conducting perturbation experiments on the system under study[20]. However, this approach is tedious, lengthy, and expensive. Another approach is to employ scRNA-seq simulators. Despite great progress in this domain, most simulators lack the essential properties for this task, such as the preservation of gene identities and simulation based on a user-provided ground truth GRN. For example, scGAN[24], cscGAN[24], scDESIGN2[25], and SPARSIM[26] capture inter-gene correlations in their datasets. However, since they do not explicitly impose a known GRN (that can act as the ground truth), they are not suitable for benchmarking GRN inference methods.

A small subset of simulators (e.g., BoolODE[19], SERGIO[4], and GeneNetWeaver (GNW)[27]) explicitly incorporate GRNs capturing transcription factor (TF)-gene dynamics. GNW was used to generate benchmarks for DREAM challenges[27-29]. 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 it for this purpose by externally inducing dropout events[10,30]. SERGIO and BoolODE were designed to simulate scRNA-seq data using stochastic differential equations (SDEs)[4,19] and have been used to benchmark a variety of GRN inference methods. However, often a mismatch between the benchmarking results based on experimental and their simulated data have been reported, which may be attributed to differences between the simulated and experimental datasets. For example, in the BEELINE study (using BoolODE), some of the top ranking methods on simulated datasets reported near random performance on experimental datasets[19]. Both simulators enable the user to simulate more realistic datasets by carefully selecting the values of the SDE parameters. Moreover, using a reference dataset, SERGIO allows fine tuning the added technical noise in an iterative procedure until the generated and reference datasets are matched[4]. While these steps can help improve the resemblance to real datasets, they are often suboptimal and pose an undue burden on the 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 GRN is imposed on the “clean” dataset, but technical noise is added afterwards, this step may change the encoded causal relationships in a non-trivial way (which may explain why 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 (see Discussion). These shortcomings demonstrate the need for simulators capable of generating realistic scRNA-seq data that retain the regulatory dynamics specified by a user-defined GRN. Importantly, with the growing interest in causal inference, simulators that impose *causal* GRNs are of great need.

GRouNdGAN (GRN-guided in silico simulation of single-cell RNA-seq data using Causal generative adversarial networks) is a causal implicit generative model for GRN-guided simulation of scRNA-seq data inspired by CausalGAN[31]. Given an input GRN and a reference dataset, it can be trained to generate simulated data that is both indistinguishable from the reference data and faithful to the causal regulatory interactions of the input GRN. Unlike model-based simulators that rely on simplifying assumptions regarding co-regulatory patterns, in GRouNdGAN these patterns are *learned* through complex functions. This allows it not to compromise on the underlying complexity of the system and to model elaborate regulatory dynamics. GRouNdGAN provides state-of-the-art performance in realistic scRNA-seq data generation, while preserving gene identities, user-defined causal gene regulatory interactions, and cellular dynamics (e.g., lineage trajectory and pseudo-time ordering). This is achieved through implicit parameterization without the need for manual fine-tuning. Using GRouNdGAN, we benchmark seven GRN inference methods and find the results to be aligned with BEELINE's experimental results[19]. Furthermore, the causal structure of GRouNdGAN enables it to be used for sampling from both interventional and observational data distributions, enabling *in silico* knockout experiments.

Results

88 **GRouNdGAN generates scRNA-seq data using causal generative adversarial networks**

89 GRouNdGAN is a deep learning model that generates scRNA-seq data while imposing a user-defined causal
90 GRN to describe the regulatory relationships of the genes and TFs. Its architecture builds on the causal
91 generative adversarial network[31] and includes a causal controller, target generators, a critic, a labeler and
92 an anti-labeler (Figure 1). Training includes two steps. First, the causal controller that generates TF
93 expression values is pre-trained (Figure 1A) as the generator of a Wasserstein GAN (WGAN)[32] (Methods). In
94 the second step (Figure 1B), TF expression values (generated by the pre-trained causal controller) and
95 randomly generated noise are provided as input to the target generators that produce target genes'
96 expressions, while incorporating the TF-gene relationships of the input GRN. To achieve this, as input, each
97 target generator only accepts a noise value and the generated expression values of TFs that causally regulate
98 it (Figure 1B, Methods). The generated expression of genes and TFs are then fed to a library-size
99 normalization (LSN) layer[24].

100

101 The critic's role is to quantify the Wasserstein distance between the reference and simulated
102 data. The target generators are trained in an adversarial manner to generate realistic datapoints
103 indistinguishable from the reference datapoints by the critic. The labeler/anti-labeler estimate
104 the causal controller's TF expression values *only* from the target genes' expressions to ensure
105 that the generated causal TF-gene dependencies are encoded. Details are provided in Methods,
106 the architectural choices in Tables S1-S2, and an ablation study in File S1 and Table S3.

107

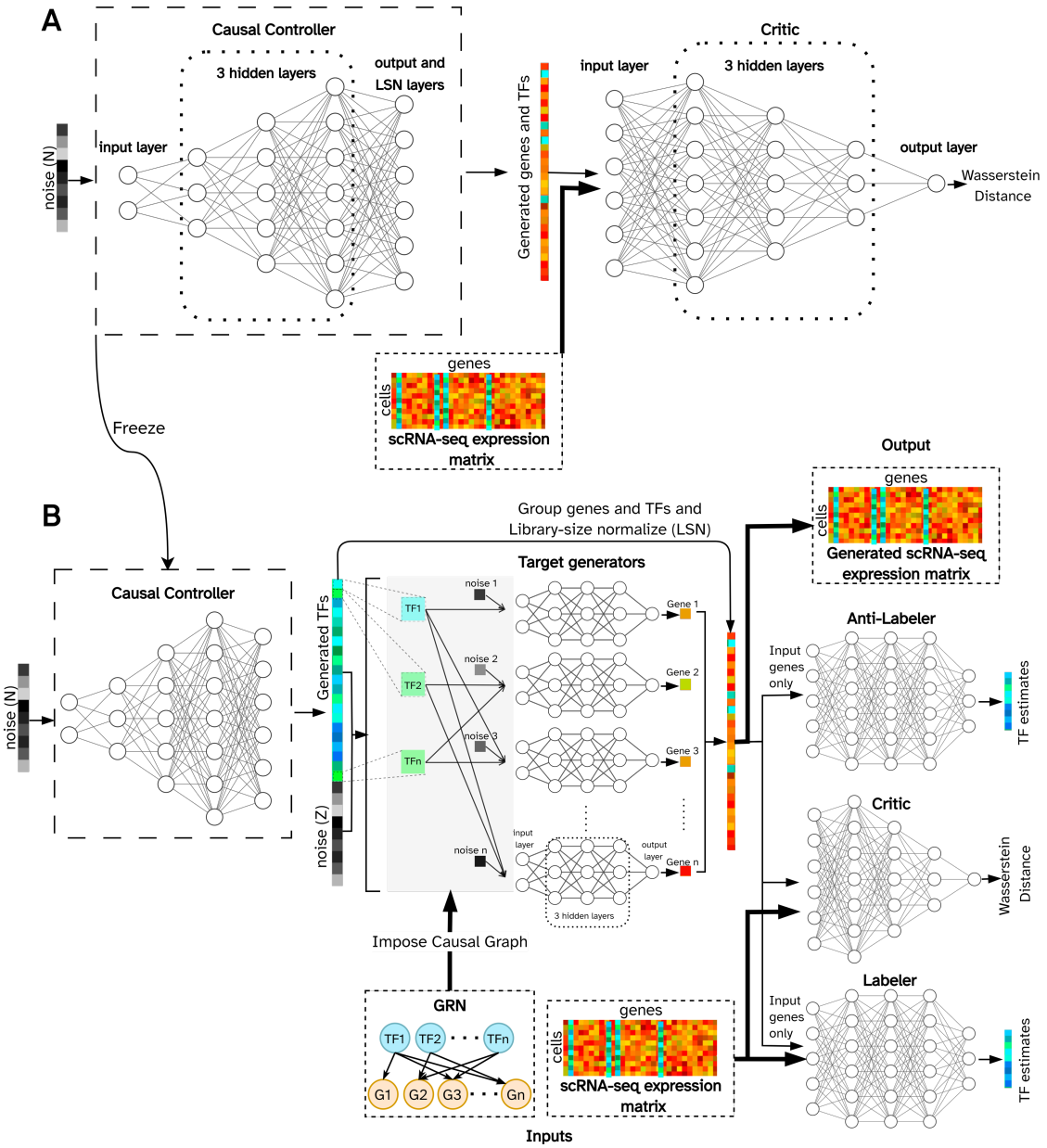

Figure 1: Architecture and training procedure of GRouNdGAN. A) A WGAN is pre-trained to generate realistic simulated cells. B) The LSN layer of the generator of the trained WGAN (panel A) is removed, its weights are frozen, and is used as the causal controller to generate unnormalized TF expression values. 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.

121

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

123 We trained GRouNdGAN on three datasets. The first dataset contained scRAN-seq profiles of
124 68579 human peripheral blood mononuclear cells (PBMCs) from 10x Genomics[33] corresponding
125 to eleven cell types ("PBMC-All"). We formed a dataset of the most common cell type in the
126 PBMC-All dataset containing 20773 CD8+ Cytotoxic T-cells ("PBMC-CTL"). Additionally, we
127 obtained the scRNA-seq (MARS-seq) profile of 2730 cells corresponding to differentiation of
128 hematopoietic stem cells to different lineages from mouse bone marrow[34] ("BoneMarrow"). We
129 used GRNBoost2[8] to identify fifteen TFs for each gene to form the input GRN. Note that these
130 identified interactions may contain both spurious correlations and causal relationships in the
131 reference dataset. However, when they are used as input to GRouNdGAN, they will correspond
132 to causal interactions in the *simulated* dataset (see Kocaoglu et al. for theoretical details of
133 causality in this architecture[31]). For each dataset, we trained GRouNdGAN on a randomly selected
134 training set and evaluated on a held-out test set (Methods, Table S4). Figures 2A-2B and S1-S3
135 show the t-SNE plots of reference and simulated cells, qualitatively revealing their similarity.

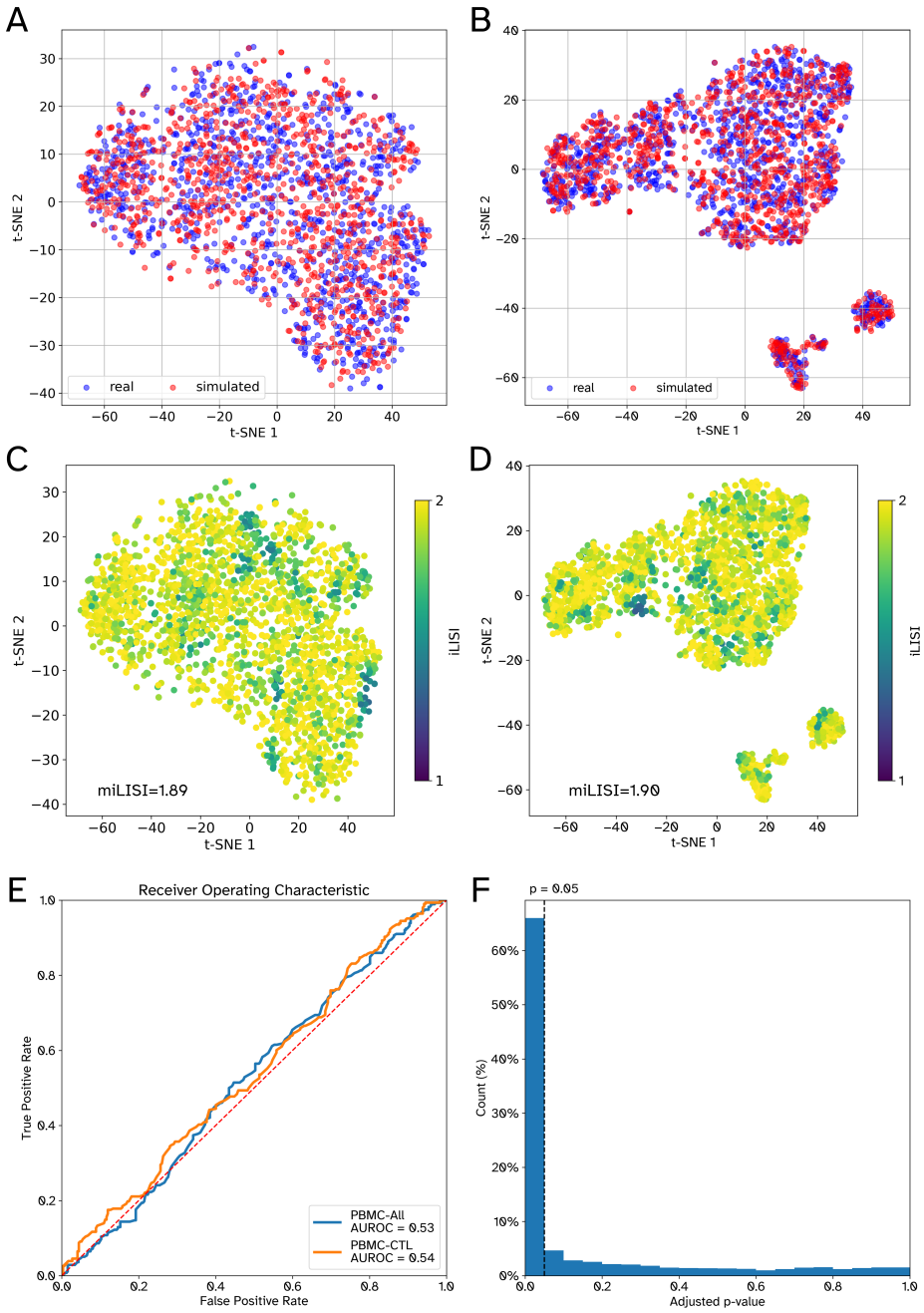

Figure 2: Performance of GRouNdGAN in generating realistic scRNA-seq data. All plots correspond to 1000 real (experimental) held-out test cells and 1000 simulated cells. Each gene in the imposed GRN of GRouNdGAN is regulated by 15 TFs (identified using GRNBoost2 from the experimental training set). Panels 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.

We quantitatively assessed their similarity using Euclidean distance, Cosine distance, maximum mean discrepancy (MMD)[35], mean integration local inverse Simpson's index (miLISI)[36], and the area under the receiver operating characteristic curve (AUROC) of a random forests (RF) classifier distinguishing simulated and experimental cells (Methods). As a "control", we calculated these metrics using two halves of the reference test set. Table 1 shows the performance of GRouNdGAN, control, and three state-of-the-art simulators for the PBMC-CTL dataset (see Table S4 for training set performance). 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 user's judgement for distribution matching (e.g., SERGIO and BoolODE) is user dependent and 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 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[24]	0.00095	222	0.031	0.59	1.888
scDESIGN2[25]	0.00100	229	0.065	0.76	1.736
SPARsim[26]	0.00104	235	0.309	0.95	1.625
Control	0.00019	99	0.012	0.50	1.909

We sought to determine whether GRouNdGAN can generate samples from different cell types. If cell type information of the reference dataset is available a priori, one can easily generate realistic

170 samples of each cell type separately (similar to the PBMC-CTL analysis) and use either cell type-
171 specific GRNs, a shared GRN, or a combination of both. However, we asked whether our model
172 could generate realistic samples *without* this knowledge, a more challenging task and useful for
173 when such information is unavailable. Also, not requiring cell type information reduces the
174 amount of user involvement (to annotate cell types) and improves model's usability. We
175 repeated the analysis using the PBMC-All dataset, containing eleven cell types. Although
176 GRouNdGAN did not receive cell type/cluster information, we did not observe mode collapse
177 (Figure S2), and it was able to generate cells from distinct clusters (Figure 2B, test set miLISI =
178 1.90). GRouNdGAN outperformed all simulators that did not use cell cluster information (Table
179 2) and compared to those that utilized this information, it was still the top performing based on
180 MMD and RF AUROC and the second best according to the Euclidean distance (6% higher than
181 the best) and miLISI (0.5% lower than the best) (Table 2). Finally, GRouNdGAN outperformed all
182 other simulators based on all metrics when we repeated the analysis using the BoneMarrow
183 dataset, containing continuous cell states and much fewer cells (Table S4). These results show
184 that GRouNdGAN is a stable simulator of scRNA-seq data and even without extra cell type
185 information in heterogenous datasets, it automatically generates realistic samples reflecting
186 different cell types (while imposing a GRN). (See File S1 and Table S5 for stability analysis and the
187 effect of different GRN properties on the performance).

188

189 **Table 2:** Performance of different simulators in generating realistic scRNA-seq data using the
190 PBMC-All dataset. Baseline models that enable utilization of cell cluster labels or ratios are run
191 with and without this information. The metrics are calculated between a simulated dataset of
192 1000 cells and the held-out test set of 1000 real cells. Each gene in the imposed GRN of
193 GRouNdGAN is regulated by 15 TFs (constructed using GRNBoost2 from the experimental training
194 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	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[24]	Yes	0.00025	242	0.030	0.56	**1.91**
cWGAN	Yes	0.00053	239	0.027	0.57	1.89
scGAN[24]	No	0.00081	300	0.035	0.58	1.90
scDESIGN2[25]	Yes	0.00032	181	0.046	0.86	1.81
scDESIGN2[25]	No	0.00057	256	0.124	0.91	1.53
SPARsim[26]	Yes	**0.00019**	**158**	0.287	0.96	1.66
SPARsim[26]	No	0.00035	245	0.307	0.96	1.39
Positive control	NA	0.00022	116	0.012	0.50	1.91

GRouNdGAN imposes a causal GRN in the simulated data

To assess the ability of GRouNdGAN in imposing the input causal GRN, we performed *in-silico* TF knockout experiments (one of GRouNdGAN's capabilities) on the simulated cells from the PBMC-CTL dataset. We performed 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, ensuring that the perturbations were performed on the same batch of cells forming matched case/control experiments. There was no change in the expression of genes that were not regulated by the knocked-out TF (as expected), while the expression of the regulated genes changed. Figure 2F shows the distribution of the adjusted p-values (Benjamini-Hochberg) for each TF-gene edge in the GRN, comparing the expression of a gene across all cells before and after the knockout of one of its regulating TFs (two-sided Wilcoxon signed-rank tests). In the

212 majority (66.5%) of cases, the knockout of a gene's regulating TF significantly (adjusted p-value <
213 0.05) altered its expression, showcasing that the GRN is indeed imposed in the expression profiles
214 of simulated cells. Note that since each gene is regulated by multiple TFs, we cannot expect that
215 knockout of a single TF result in a significant change in the expression of its target genes in all
216 cases.
217
218 **The imposed GRN can be reconstructed from the simulated data**
219 To test whether the imposed GRN can be reconstructed from the generated data, we first applied
220 GRNBoost2[8] to the reference PBMC-CTL dataset and recorded the top ten TFs for each gene. For
221 each gene, we created two sets of five TFs based on their even/odd parity in the ranked list: TFs
222 ranked 1st, 3rd, 5th, 7th, and 9th were connected to the gene and were imposed ("positive control
223 GRN"), while the remaining five TFs were not ("negative control GRN"), resulting in two GRNs
224 with identical densities and the same number of TFs and target genes. When considering the
225 positive and negative control GRNs as ground truths (separately), the GRN reconstructed using
226 GRNBoost2 from the reference dataset resulted in comparable AUPRC (0.28 and 0.24,
227 respectively, Table S6), reflecting that these edges are of comparable importance.
228
229 We simulated data using GRouNdGAN by imposing only the positive control GRN, and used
230 GRNBoost2, PIDC[10] GENIE3[9], and PPCOR[37] to reconstruct the underlying GRN (Figure 3, Figure S4,
231 Table S6). All methods reconstructed the imposed edges with a much higher AUPRC compared
232 to the reference dataset, while they did not assign high scores to the unimposed edges, resulting
233 in close to random AUPRC. Figure 3 shows the performance of GRNBoost2 applied to the

reference, GRouNdGAN-simulated, and scGAN-simulated data. The performance for scGAN (both GRNs) was lower than the reference dataset, showcasing that it had not preserved the TF-gene relationships of the experimental dataset well. The performance was relatively consistent between positive (Figures 3A and 3C) and negative control (Figures 3B and 3D) GRNs using the reference and scGAN-simulated data; however, it was much higher for the positive control GRN (compared to negative control GRN) using GRouNdGAN-simulated data. Similar patterns were observed when we repeated these analyses using the BoneMarrow dataset (Table S6, 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.

Next, we asked whether GRNBoost2 can reconstruct the imposed GRN, if GRNs of different densities are used to simulate data. Using data for which each gene was regulated by 15, 10, 5, and 3 TFs, we observed that in all cases the GRN imposed by GRouNdGAN could be inferred from the simulated data (Table S6). Finally, we repeated the controlled analysis above, swapping the role of positive and negative control GRNs. Similar to the results of Figure 3, the imposed edges were inferred by GRNBoost2, while the unimposed edges were not accurately discovered (Table S6). These results show that 1) GRouNdGAN imposes the causal GRN, 2) the GRN inference methods can identify the imposed edges, and 3) the TF-gene relationships present in the reference dataset but unimposed by GRouNdGAN are disrupted during simulation. The last property is particularly desirable for benchmarking GRN inference methods to ensure that the regulatory relationships present in the reference dataset (but unimposed) do not bias the simulated data, which could result in an inflated false positive rate.

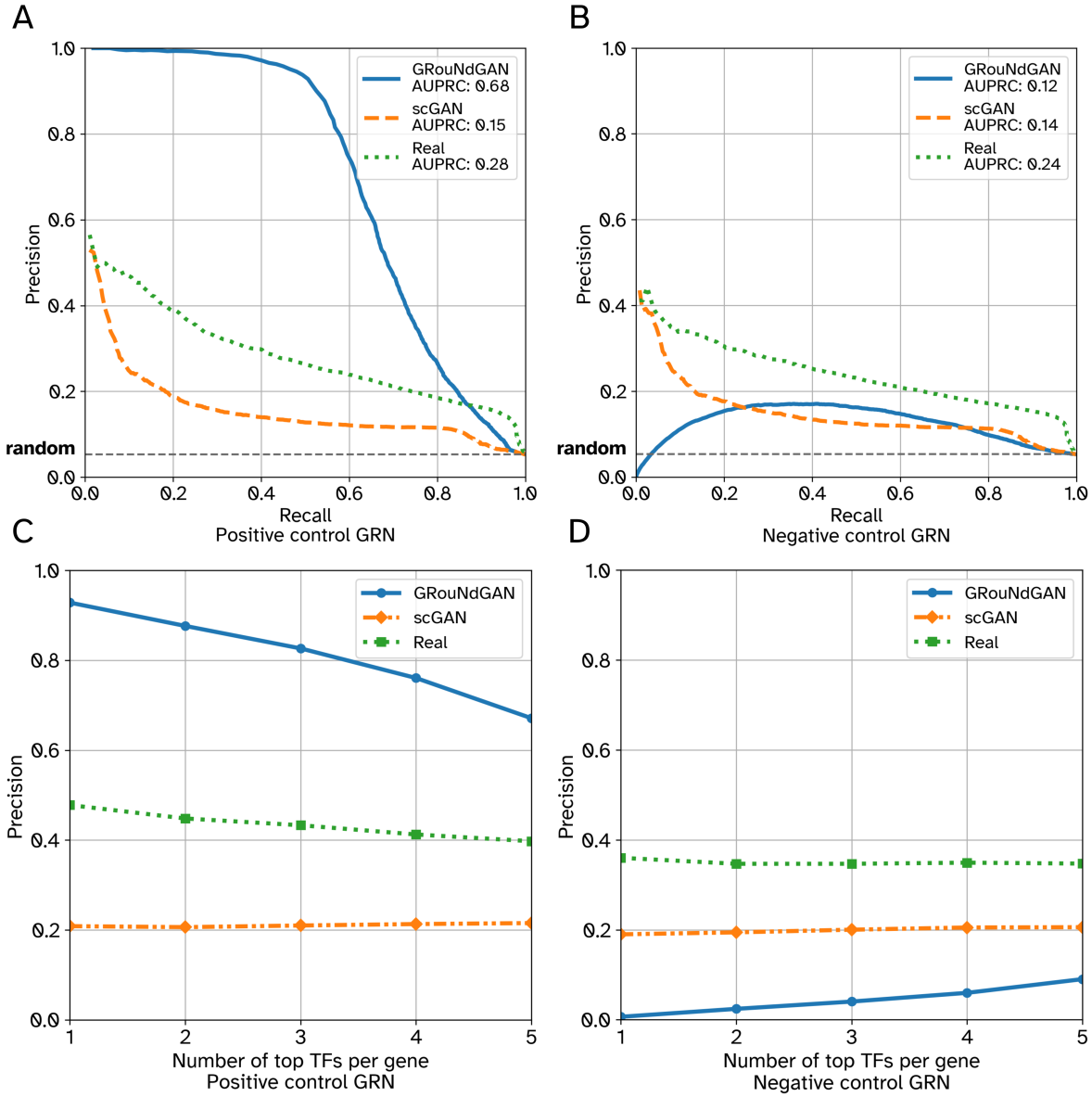

Figure 3: Performance of GRNBoost2 in recovering the imposed edges versus unimposed edges (PBMC-CTL). Left column shows the AUPRC (A) and Precision at k (per gene) (C) when the imposed edges (positive control GRN) were considered the ground truth. Right column shows the AUPRC (B) and Precision at k (per 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.

266 **GRouNdGAN-simulated data preserves trajectories and can be used for pseudo-time inference**

267 In addition to deciphering discrete states, scRNA-seq data is often used to determine continuous
268 cell transitions during biological processes such as differentiation, using trajectory and pseudo-
269 time inference[38-44]. We sought to determine whether GRouNdGAN-simulated data conforms to
270 the transitional states and pseudo-time of the reference data. This is particularly important for
271 benchmarking GRN inference methods that utilize pseudo-time information[12-14]. For this
272 purpose, we used the BoneMarrow dataset corresponding to differentiation of hematopoietic
273 stem cells to different lineages.

274

275 We used partition-based graph abstraction (PAGA)[38] for trajectory inference of the reference and
276 GRouNdGAN-simulated data. PAGA computes graph-like maps of data manifolds faithful to the
277 topology of data, retaining its continuous and discrete structures. Figure S6 shows the PAGA
278 graphs, where nodes capture discrete states, and the edges signify transitions between them. To
279 compare the identified trajectories, we used markers of the cell types present in this dataset
280 (Table S7)[38]. Figure 4 shows the expression patterns of two marker genes for erythroid cells
281 (*Gata1, Klf1*), neutrophils (*Mpo, Ctsg*), and basophils (*Mcpt8, Prss34*) for PAGA trajectory graphs
282 of both datasets (Figures S7-S13 shows marker genes of all cell types). The comparison showed
283 similar activation patterns between corresponding regions of PAGA graphs of the experimental
284 and GRouNdGAN data, revealing several trajectories (e.g., erythroid and neutrophil branches).
285 Strikingly, even the number of nodes activated in each figure scaled similarly between the two
286 datasets: e.g., the markers of basophils were consistently expressed in only one node in each
287 dataset (Figures 4E-4F).

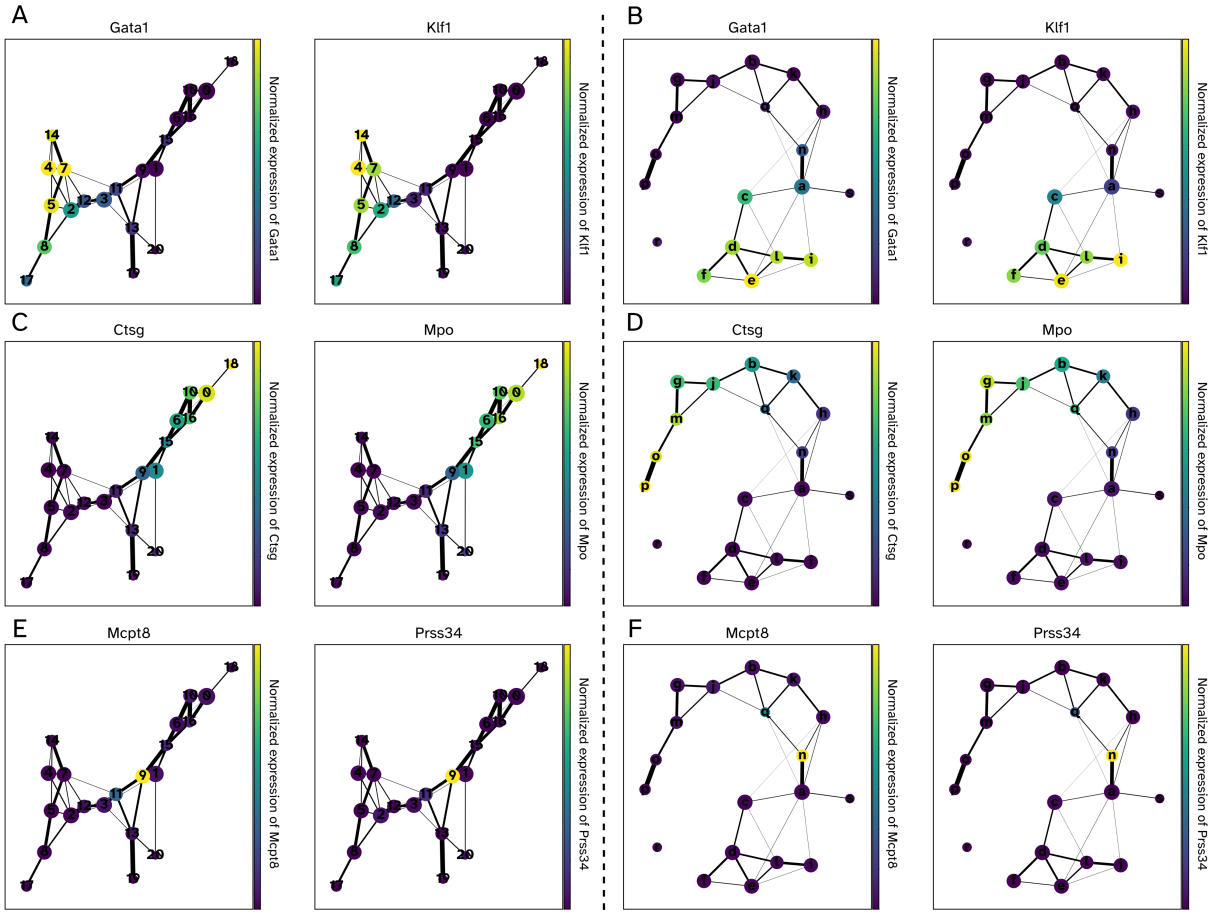

Figure 4: PAGA graphs and cell type marker gene activation patterns in experimental and simulated BoneMarrow dataset. In each figure, nodes represent Louvain clusters (capturing discrete states), edges represent transitions between states, and edge weights indicate confidence in the presence of connections. 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).

We annotated different cell types present in both datasets using these marker genes (following PAGA's official tutorial) (Figures 5A-5B). The GRouNdGAN's annotated graph showed similar topological features to that of the reference data and captured known biological properties of hematopoiesis, which were also noted by Wolf, et al.[38] using the reference dataset (e.g.,

proximity between monocytes and neutrophils or association between megakaryocyte and erythroid progenitors). These observations show that the key characteristics of the reference data are retained in GRouNdGAN-simulated data.

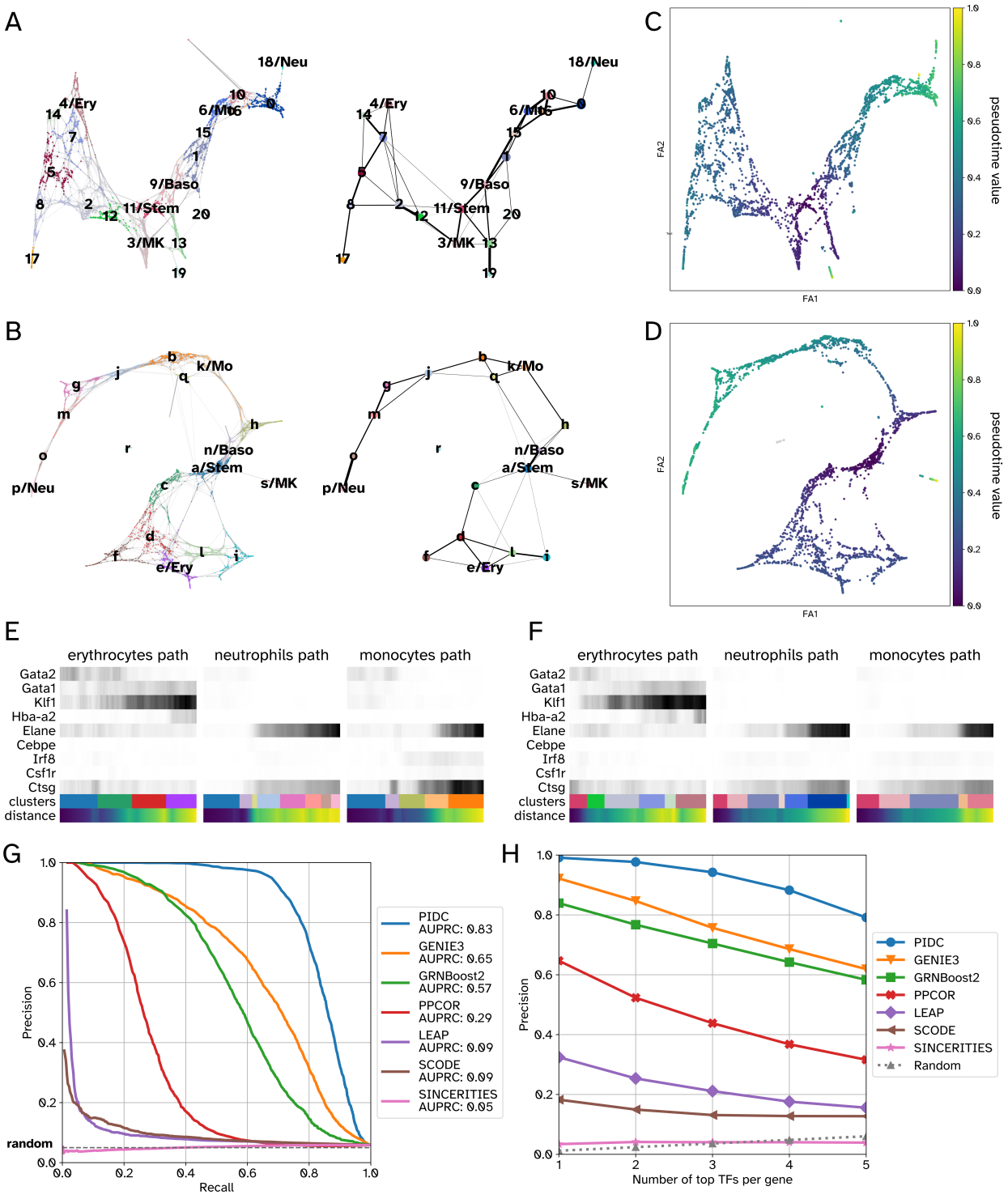

Figure 5: Consistency in trajectories and pseudo-temporal orderings between data generated by GRouNdGAN and the experimental BoneMarrow hematopoietic dataset. Panels A and B show the cell type annotations of simulated and experimental data, respectively. The figures on the left show the force-directed graphs (obtained using ForceAtlas2[45]) and the figures on the right show the PAGA graphs. Panels C and D show the PAGA-initialized force-directed graphs with cells colored by their inferred pseudo-time value. The following abbreviations are used: Stem for stem cells, Ery for erythroid cells, Neu for neutrophils, Mo for monocytes, MK for Megakaryocytes, and Baso for Basophils. Panels E and F display expression changes in the marker genes of erythrocyte (*Gata2*, *Gata1*, *Klf1*, and *Hba-a2*), neutrophil (*Elane* and *Cebpe*), and monocyte (*Irf8*, *Csf1r*, and *Ctsg*) branches for experimental (E) and simulated data (F). The distance refers to the geodesic distance from a root cell (here a stem cell). A darker shade of grey shows higher expression. Panels (G, H) show the performance of different GRN inference methods on data generated by GRouNdGAN using the BoneMarrow dataset.

We used the diffusion pseudotime algorithm[39] (in scanpy[46]) to infer the progression of cells through geodesic distance (Figures 5C-5D). We noticed strong concordance when examining pseudo-time values assigned to cell subgroups in the reference and GRouNdGAN-generated data. To track these patterns, we focused on marker expression changes along erythroid, neutrophil, and monocyte trajectories (Figures S7-S9), similar to the analysis performed by Wolf, et al.[38]. In both datasets, the activation of neutrophils' marker, *Elane,* and monocytes' marker, *Irf8,* were predominantly observed towards the later stages of their trajectories (Figures 5E-5F). Additionally, we observed the activation of erythroid maturity marker genes *Gata2*, *Gata1*, *Klf1*, and *Hba-a2* roughly in sequential order along the erythroid trajectory in both datasets (Figures 5E-5F), concordant with previous findings[38]. The striking similarity between the activation patterns of these markers in both datasets and the preservation of their ordering highlights GRouNdGAN's ability to capture dynamic transcriptional properties of scRNA-seq data, leading to correct trajectory inference and pseudo-time ordering.

335 **Benchmarking GRN inference methods using GRouNdGAN confirms prior insights from curated**
336 **experimental datasets**

337 The BEELINE study[19] found conflicting results when benchmarking GRN inference methods on
338 curated biological datasets and synthetic ones: top two performing models on synthetic data
339 where among the bottom three on biological data. We re-investigated this using GRouNdGAN-
340 simulated data with the BoneMarrow dataset and the positive and negative control GRNs
341 (described earlier). We used seven GRN inference methods used by BEELINE, GRNBoost2[8],
342 GENIE3[9], PIDC[10], PPCOR[37], LEAP[12], SCODE[13], and SINCERITIES[14], capturing a wide range of
343 methods with a broad spread of performances reported on synthetic and curated data in the
344 BEELINE study. We observed a striking resemblance between the performance pattern (order) of
345 the methods on GRouNdGAN-simulated data and curated biological data from BEELINE (Figure
346 5G-5H and Table S6 versus Figure 4 of BEELINE[19]): the only difference was the swapping of LEAP
347 and SCODE's order. We repeated this analysis using the PBMC-CTL dataset (Table S6), obtaining
348 similar patterns.

349

350 Overall, PIDC[10] outperformed all methods using both datasets, followed by GENIE3[9] and
351 GRNBoost2[8]. LEAP[12], SCODE[13] and SINCERITIES[14] (which use pseudo-time and could only be
352 applied to the BoneMarrow dataset) performed worse than others, matching their behavior in
353 the BEELINE study on curated biological data (but not on synthetic data). All methods performed
354 close to random on unimposed negative control GRN edges. These results not only show that
355 GRouNdGAN can be used for benchmarking GRN inference methods, but also shows that the

356 insights obtained from it matches those obtained from curated biological datasets, the formation
357 of which requires extensive amount of resources and effort.
358
359 ***In-silico* perturbation experiments using GRouNdGAN**
360 In many studies, one is interested to characterize the relationship between TFs' expression and
361 phenotypic labels (e.g., cell types). Differential expression (DE) analysis is a common approach to
362 identify TFs (or other genes) most associated with a specific cell type. However, to directly test
363 whether perturbing the expression of such candidate TFs change the expression of genes
364 representative of a cell type, perturbation experiments (e.g., TF knockout) are desired. As a causal
365 implicit generative model, GRouNdGAN provides the capability to sample from interventional
366 distributions, thereby enabling *in-silico* perturbation experiments. This functionality is provided
367 at inference time by enabling manipulation of TFs' expressions as the outputs of the causal
368 controller. This change is propagated along the GRN (through the model architecture), impacting
369 the expression of genes regulated by the perturbed TF(s). By enabling a deterministic mode of
370 operation, GRouNdGAN allows for exact comparison of gene expressions before and after
371 perturbation while maintaining the invariance of other parameters (Methods).
372
373 To test this functionality, we asked whether the knockout of top 3 TFs most differentially
374 expressed (Mann Whitney U test) between each cell type and the rest (identified from the
375 reference dataset) would result in less cells being generated in the vicinity of the cells of that
376 type. If positive, this implies that the perturbation in the TFs' expression has modified the gene
377 expression profile of generated cells so that they no longer resemble that particular cell type

378 (confirming our expectation). Figure 6A shows the UMAP embedding of the reference dataset
379 (miLISI = 1.94 when compared with simulated data) and Figure 6B shows the density plots of the
380 experimental and simulated data, revealing a high degree of resemblance. We focused on the
381 changes in the iLISI values of experimentally profiled cells of a specific type (annotated in the
382 original study), when jointly mapped to the same embedding space with simulated data, before
383 and after knockout. For the changes in the iLISI values to be observable, it was necessary to focus
384 on cell types that occupy relatively distinct regions of the embedding space: CD14+ monocytes,
385 CD19+ B cells, Dendritic cells, CD56+ natural killer cells (Figure S14); otherwise, the presence of
386 other cell types in that region would confound the results. Figure 6C shows the iLISI distributions
387 for each cell type (only for real cells), calculated along with unperturbed simulated cells (blue)
388 and perturbed simulated cells (orange). Figure 6D-6E show the iLISI values and the distribution
389 of generated cells after the knockout experiment for CD19+ B cells as an example (see Figures
390 S15-S18 for other cell types). In all cases, the iLISI values significantly reduced after the knockout
391 experiment (Figure 6C, Table S8), showcasing that after the knockout of top TFs associated with
392 a cell type, GrouNdGAN generated much fewer cells resembling that cell type (even though it did
393 not have the knowledge of the cell type annotations). Instead, the simulated cells were dispersed
394 into other cell types, yet they retained meaningful positions within the overall dataset
395 embedding space. Also, the miLISI of the other cell types remained relatively unchanged (Table
396 S8). This analysis shows that *in-silico* TF perturbation experiments using GrouNdGAN produces
397 results concordant with results directly obtained from real biological data, a direction that we
398 will explore further in the future studies.

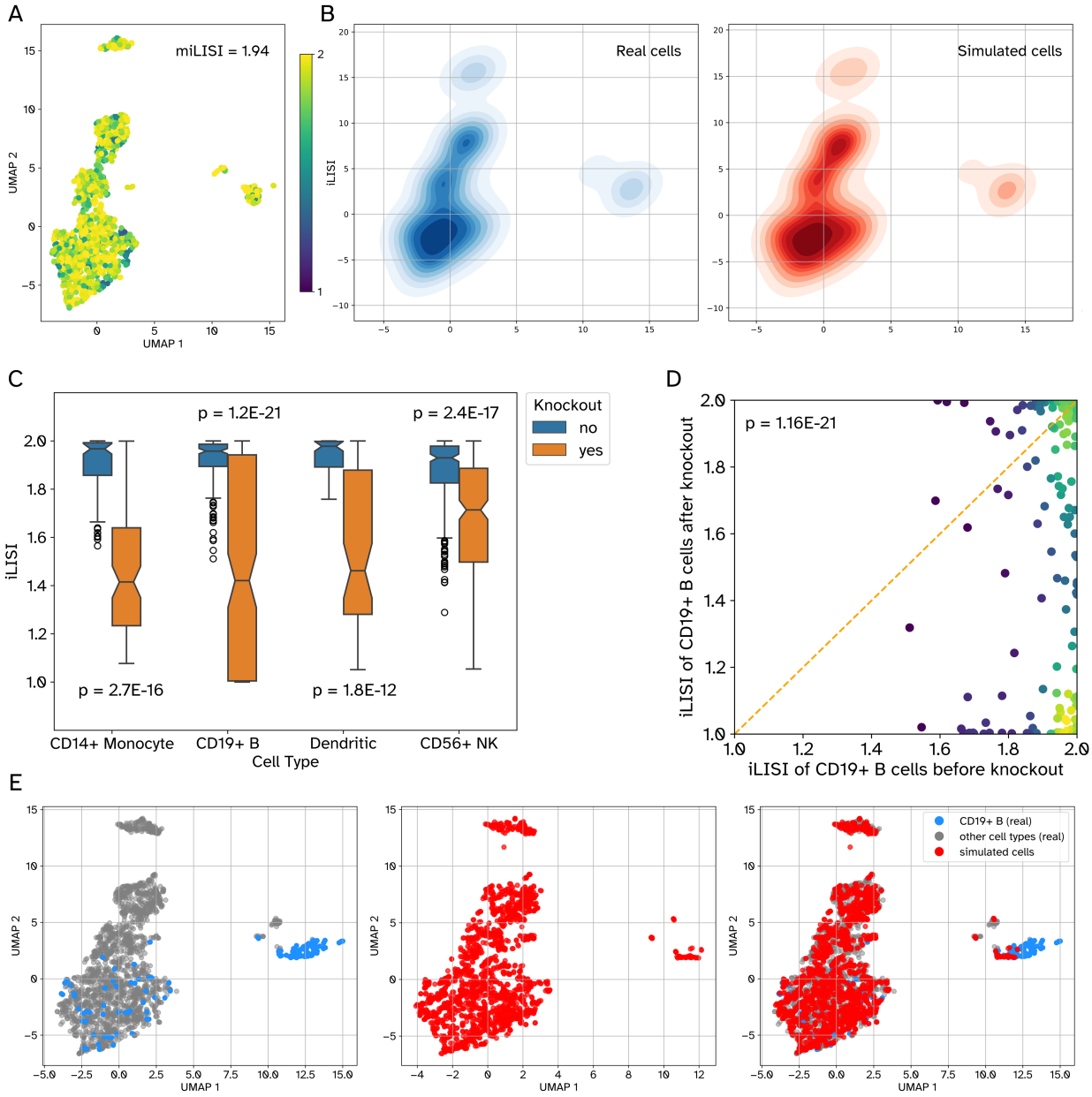

Figure 6: TF knockout experiments and their effect on cell type generation. A) The UMAP shows the distribution of 2000 randomly selected cells from the experimental PBMC-All dataset. The color shows the iLISI value of each cell (miLISI = 1.94) calculated from a UMAP embedding, jointly obtained from the 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

410 reported in this figure were calculated using one-sided Wilcoxon signed rank tests. D) The scatter
411 plot shows the iLISI values of CD19+ B cells calculated along with unperturbed simulated cells (x-
412 axis) and along with perturbed simulated cells (y-axis). The circles correspond to real cells and
413 their colors reflect the density of datapoints in that region. E) The UMAP embedding of real cells
414 (left), simulated cells after knockout of top TFs of CD19+ B cells (middle), and all together (right).
415

416 **Discussion**

417 GRouNdGAN is a causal implicit generative model designed to simulate realistic scRNA-seq
418 datasets based on a user-defined GRN and a reference dataset. By incorporating GRN
419 connections in its architecture and including auxiliary tasks, it imposes causal TF-gene
420 relationships in the gene expression profile of simulated cells. These causal relationships are
421 verifiable by *in-silico* knockout experiments and identifiable by GRN inference methods. (In
422 addition to the experimental analyses performed in this study, an interested reader should refer
423 to Kocaoglu et al.[31] for theoretical evidence of causality of data generated by this architecture.)
424

425 We demonstrated that GRouNdGAN achieves state-of-the-art performance on various tasks
426 including realistic scRNA-seq data generation, GRN inference benchmarking, and *in-silico*
427 knockout experiments. Even when applied to heterogenous datasets, it generated realistic
428 datasets without utilizing information regarding cell clusters, achieving best or second-best
429 performance compared to models that used this information. We should note that it is trivially
430 possible to use cell type/cluster information with GRouNdGAN: one can simply provide subsets
431 of the experimental dataset corresponding to a cell type/cluster as a reference, one at a time
432 with a cell-type specific (or shared) GRN. However, the fact that this model does not require this
433 information broadens its applicability. We showed that GRouNdGAN-generated data preserves

complex cellular dynamics and patterns of the reference dataset such as lineage trajectories, pseudo-time orderings, and patterns of cell type markers' activation.

GRouNdGAN imposes user-defined causal regulatory interactions, while excluding those present in the reference dataset, but not in the input GRN. This makes GRouNdGAN-simulated datasets ideal for benchmarking GRN inference algorithms with the input GRN as the ground truth. We benchmarked seven popular GRN inference algorithms and showed that our results coincide with results reported by BEELINE[19] on experimental datasets. This highlights GRouNdGAN's ability to bridge the gap between experimental and simulated datasets for GRN benchmarking without requiring excessive user manipulation. Additionally, since GRouNdGAN generates data based on a structural causal model, it can be used to sample synthetic cells from both observational and interventional distributions. Using this property, we conducted *in-silico* perturbation experiments by knocking out key regulating TFs in various cell types and observed that the targeted cell-states were suppressed in the generated dataset.

GRouNdGAN maintains key advantages over existing simulators that incorporate GRNs such as SERGIO and BoolODE. During training, GRouNdGAN's parameters are optimized to minimize any mismatch between real and simulated data distributions. This enables it to replicate scRNA-seq data characteristics (e.g., technical and biological noise) without any user intervention or post-processing. In contrast, SERGIO simulates data by first generating a "clean" dataset based on SDEs. Then, the user would iteratively fine-tune three noise parameters until the simulated and reference datasets are matched. This data-matching procedure puts extra burden on the user

and might lead the user to overly rely on fine-tuning technical noise parameters, instead of SDE parameters. Overreliance on added technical noise to achieve data-matching can potentially corrupt the data, rendering the underlying regulatory patterns unobservable in the final simulated dataset. This might explain the near-random GRN inference performance reported by SERGIO on their noisy datasets[4]. GRouNdGAN is trained such that each of its generated genes or TFs mimics the expression patterns of a corresponding gene or TF in the reference dataset, preserving gene/TF identities. In contrast, BoolODE and SERGIO simulate pseudo-genes whose identities cannot be readily mapped to their counterparts in the reference dataset. Gene identity preservation can be attained in these simulators by fine-tuning the SDE parameters until the expression patterns of a given generated gene aligns with a corresponding gene in the reference dataset. This, however, places further burden on the user. Finally, GRouNdGAN learns complex non-linear co-regulatory patterns from the data, instead of relying on simplifying assumptions or user input. In contrast, SERGIO assumes that the combined effect of multiple regulatory TFs is simply the sum of their individual effects. Similarly, BoolODE requires the user to provide a truth table specifying combinatory TF co-regulation rules for each target gene. In contrast, GRouNdGAN's neural network-based generator enables it to implicitly learn complex non-linear co-regulatory patterns that may represent co-expression, co-inhibition, or co-activation of individual genes. This is achieved during the training of GRouNdGAN without user intervention.

Several key points need to be considered 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 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. Second, the default architecture of the model includes LSN layers and, as such, the model generates library-size normalized expression values. Our ablation study (File S1) showed that without the LSN layer, GRouNdGAN can still generate realistic profiles; however, including it improves the GRN imposition and stabilizes the training. 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 generated the data. Despite this, our analysis using the positive/negative control GRNs showed that GRN inference methods could detect the imposed edges while ignoring others that were unimposed. 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, the quality of the data deteriorates if the imposed TF-gene relationships are significantly different from those consistent with the patterns in the reference dataset. The reason is that in such a scenario, generating realistic simulated datapoints and imposing the GRN act as contradictory requirements. This is a fundamental problem that is not unique to GRouNdGAN and is applicable to any simulator with

these two requirements. However, simulators that do not try to impose a causal graph (e.g., scGAN) can focus on generating realistic scRNA-seq samples, even at the expense of disrupting some of the interactions present in the reference dataset (as we observed in Figure 3 with scGAN). As a result, for GRN inference method benchmarking, we recommend first finding a set of regulatory edges from the reference dataset (e.g., using GRNBoost2) and imposing a selected subset of those in the simulated data. This ensures that the imposed edges are encoded in the simulated data, the unimposed (or undetected) edges are disrupted (as shown in our analyses), and the simulated dataset resembles the reference dataset. It is important to note that the regulatory edges inferred from the reference dataset do not need to be all causal; some of them could represent spurious correlations that are observable in the data. In fact, most GRN inference methods do not identify causal edges, but can be used to form the input GRN. When a subset of the (potentially) non-causal edges of the reference dataset are imposed by GRouNdGAN, they are imposed in a causal manner, representing the *causal* graph underlying the *simulated* data (and not the reference data).

The current version of GRouNdGAN only allows imposing a bipartite GRN and does not support regulation of TFs by other TFs, an assumption that we plan to relax with the future versions of the model. Moreover, we intend to augment the model's architecture in the future to enable imposing a causal graph with three layers of nodes capturing TF-gene-phenotype relationships. This is particularly useful for development and evaluation of phenotype-relevant GRNs, a concept that we have introduced in the past[11]. Additionally, a conditional version of GRouNdGAN can be developed, allowing the user to generate cells of specific cell type or on a specific stage along a

522 lineage trajectory. Finally, by incorporating DAG structural learning[47], we intend to enable
523 GRouNdGAN to infer a GRN while simultaneously learning to mimic the reference dataset.
524
525 **Methods**
526 **Datasets and preprocessing**
527 We downloaded the human peripheral blood mononuclear cell (PBMC Donor A) dataset
528 containing the single-cell gene expression profiles of 68579 PBMCs represented by UMI counts
529 from the 10x Genomics (https://support.10xgenomics.com/single-cell-gene-
530 expression/datasets/1.1.0/fresh_68k_pbmc_donor_a). This dataset contains a large number of
531 well-annotated cells and has been used by other models to generate synthetic scRNA-seq
532 samples[24]. We also downloaded the scRNA-seq profile of 2730 cells corresponding to
533 differentiation of hematopoietic stem cells to different lineages from mouse bone marrow[34]
534 ("BoneMarrow" dataset) from Gene Expression Omnibus (accession number: GSE72857)
535
536 We followed the pre-processing steps of scGAN and cscGAN[24] using scanpy version 1.8.2[46]. In
537 each dataset, cells with nonzero counts in less than ten genes were removed. Similarly, genes
538 that only had nonzero counts in less than three cells were discarded. Next, library-size
539 normalization was performed on the counts per cell with a library size equal to 20,000 in order
540 to be consistent with previous studies [24]. Next, top 1000 highly variable genes were selected using
541 the dispersion-based method described by Satija et al.[48].
542
543 **GRouNdGAN's model architecture**

544 GRouNdGAN consists of 5 components, each implemented using separately parameterized
545 neural networks: a causal controller, target generators, a critic, a labeler, and an anti-labeler
546 (Figure 1B).
547
548 *Causal controller*: The role of causal controller is to generate the expression of TFs that causally
549 control the expression of their target genes based on a user-defined gene regulatory network
550 (Figure 1B). To achieve this, it is first pre-trained (Figure 1A) as the generator of a Wasserstein
551 GAN (WGAN). GANs are a class of deep learning models that can learn to simulate non-parametric
552 distributions[49]. They typically involve simultaneously training a generative model (called the
553 "generator") that produces new samples from noise, and its adversary, a discriminative model
554 that tries to distinguish between real and generated samples (called the "discriminator"). The
555 generator's goal is to generate samples so realistic that the discriminator cannot determine
556 whether it is real or simulated (an accuracy value close to 0.5). Through adversarial training, the
557 generator and discriminator receive feedback, allowing them to co-evolve in a symbiotic manner.
558
559 The main difference between a WGAN and a traditional GAN is that in the former, a Wasserstein
560 distance is used to quantify the similarity between the probability distribution of the real data
561 and the generator's produced data (instead of Kullback–Leibler or Jensen–Shannon divergences).
562 Wasserstein distance has been shown to stabilize the training of WGAN without causing mode
563 collapse[32]. The detailed formulation of the Wasserstein distance used as the loss function in this
564 study is provided in File S1. In addition, instead of a discriminator, WGAN uses a "critic" that
565 estimates the Wasserstein distance between real and generated data distributions. In our model,

566 we added a gradient penalty term for the critic (proposed by Gulrajani et al.[50] as an alternative
567 to weight clipping used in the original WGAN) in order to overcome vanishing/exploding
568 gradients and capacity underuse issues.
569
570 In the pretraining step, we trained a WGAN with a generator (containing an input layer, three
571 hidden layers, and an output layer), a library-size normalization (LSN) layer[24], and a critic
572 (containing an input layer, three hidden layers, and an output node). A noise vector of length
573 128, with independent and identically distributed elements following a standard Gaussian
574 distribution, was used as the input to the generator. The output of the generator was then fed
575 into the LSN layer to generate the gene and TF expression values. The details of hyperparameters
576 and architectural choices of this WGAN are provided in Table S1 (in File S1). Although we were
577 only interested in generating expression of TFs using the generator of this WGAN (in the second
578 step of pipeline), the model was trained using all genes and TFs to properly enforce the library-
579 size normalization. Once trained, we discarded the critic and the LSN layer, froze the weights of
580 the generator and used it as the "causal controller"[31] to generate expression of TFs (Figure 1B).
581
582 *Target generators*: The role of target generators is to generate the expression of genes causally
583 regulated by TFs based on the topology of a GRN. Consider a target gene Gj regulated by a set of
584 TFs: $\{TF_1, TF_2, \dots, TF_n\}$. Under the causal sufficiency assumption and as a result of the manner
585 by which TFs' expressions are generated from independent noise variables, we can write $E_{Gj} =$
586 $f_{Gj}(E_{TF1}, E_{TF2}, \dots, E_{TFn}, N_{Gj})$ and $E_{TFi} = f_{TFi}(N_{TFi})$ for $i = 1, 2, \dots, n$, where E represents
587 expression, N represents a noise variable, and f represents a function (to be approximated using

588 neural networks). All noise variables are jointly independent. Following the theoretical and
589 empirical results of CausalGAN[31], we can use feedforward neural networks to represent functions
590 f by making the generator inherit its neural network connections from the causal GRN. To
591 achieve this, we generate each gene in the GRN by a separate generator such that target gene
592 generators do not share any neural connections (Figure 1B).
593
594 As input, the generator of each gene accepts a vector formed by concatenating a noise variable
595 and a vector of non-library size normalized TF expressions from the causal controller,
596 corresponding to the TFs that regulate the gene in the imposed GRN (i.e., its parents in the graph).
597 The expression values of TFs and the generated values of target genes are arranged into a vector,
598 which is passed to an LSN layer for normalization. We used target generators with three hidden
599 layers of equal width. The width of the hidden layers of a generator is dynamically set as twice
600 the number of its regulators (the noise variable and the set of regulating TFs). If the imposed GRN
601 is relatively dense and contains more than 5000 edges, we set the depth to 2 and the width
602 multiplier to 1 to be able to train on a single GPU. The details of hyperparameters and
603 architectural choices are provided in Table S2 (in File S1).
604
605 In practice, generating each target gene's expression using separate neural networks introduces
606 excess overhead. This is because in every forward pass, all target genes' expressions must be first
607 generated before collectively being sent into the LSN layer. As a result, instead of parallelizing
608 the generation of each target gene's expression (which due to the bottleneck above does not
609 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 such a scenario, and to make realistically looking simulated cells, it is more convenient for a WGAN to simply ignore the strict constraints of the GRN and solely rely on noise.

To overcome this issue, we used auxiliary tasks and neural networks known as "labeler" and "anti-labeler"[31]. The task of these two networks is to estimate the causal controller's TF expressions (here called labels) from the target genes' expressions alone (by minimizing the squared L2 norm between each element's TF estimates and their true value). This resembles the

632 idea behind an autoencoder and ensures that the model will not disregard the expression of TFs
633 in generating the expression of their target genes. The anti-labeler is trained solely based on the
634 outputs of the target generators, while the labeler utilizes both the outputs of target generators
635 and the expression of real cells (Figure 1B). They are both implemented as fully connected
636 networks with a width of 2000 and a depth of 3 and optimized using the AMSGrad[51] algorithm.
637 Each layer, except the last one, utilizes a ReLU activation function and batch normalization. In
638 addition to WGAN losses, we add labeler and anti-labeler losses to the generator to minimize
639 both. This is different from the approach used in CausalGAN, where the anti-labeler's loss is
640 maximized in the early training stages to ensure that the generator doesn't fall into label-
641 conditioned mode collapse. In GRouNdGAN, the causal controller is pretrained and generates
642 continuous labels (TFs expression) and does not face a similar issue. As a result, we instead
643 minimized the loss of the anti-labeler from the beginning and as such the labeler and anti-labeler
644 both act as auxiliary tasks to ensure the generated gene expression values take advantage of TFs
645 expression.
646
647 **Training procedure and hyperparameter tuning**
648 We follow a two-step training procedure comprising of training two separate WGANs (Figure 1)
649 to train GRouNdGAN. The generators and critics in both GANs are implemented as fully
650 connected neural networks with rectified linear unit (ReLU) activation functions in each layer,
651 except for the last layer of the critic. The weights were initialized using He initialization[52] for layers
652 containing ReLU activation and Xavier initialization[53] for other layers (containing linear
653 activations). We used Batch Normalization[54] to normalize layer inputs for each training

minibatch, except for the critic. This is since using it in the critic invalidates the gradient penalty's objective, as it penalizes the norm of the critic's gradient with respect to the entire batch rather than to inputs independently. An LSN layer[24] was used in both WGANs (Figure 1A and 1B) to scale counts in each simulated cell to make it consistent with the library size of input reference dataset. This normalization results in a dramatic decrease in convergence time and smooths training by mitigating the inherent heterogeneity of scRNA-seq data.

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

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

We alternated between minimizing the generator loss for one iteration and maximizing the generator loss for five iterations. We employed the AMSGrad[51] optimizer with the weight decay parameters $\beta_1 = 0.5$, $\beta_2 = 0.9$ and employed an exponentially decaying learning rate for the optimizer of both the critic and generator.

The hyperparameters were tuned using a validation set consisting of 1000 cells from the PBMC-CTL dataset (Tables S1-S2 in File S1) based on the Euclidean distance and the RF AUROC score, which were consistently in accord. The same hyperparameters were used for all other analyses and the other two datasets (PBMC-All and BoneMarrow).

673

674 **Causal GRN preparation**

675 This section describes the creation of the causal graph inputted to GRouNdGAN used to impose
676 a causal structure on the model. GRouNdGAN accepts a GRN in the form of a bipartite directed
677 acyclic graph (DAG) as input, representing the relationship between TFs and their target genes.
678 In this study, we created the causal graph using the 1000 most highly variable genes identified in
679 the preprocessing step. First, the set of TFs among the highly variable genes were identified based
680 on the AnimalTFDB3.0 database[55] and a GRN was inferred using GRNBoost2[8] (with the list of TFs
681 provided). Table S5 shows the characteristics of the different GRNs used in this study. It is
682 important to note that the regulatory edges identified from the reference dataset using
683 GRNBoost2 are not necessarily causal edges (and they do not need to be for the purpose of
684 forming the input GRN), but they are consistent with the patterns of the data. However, when
685 this (potentially non-causal) GRN is imposed by GRouNdGAN, it is imposed in a causal manner
686 and represents the causal data generating graph of the simulated data (and not the reference
687 data).

688

689 **Evaluation of the resemblance of real and simulated cells**

690 We evaluated all models using held-out test sets containing randomly selected cells from each
691 reference dataset (1000 cells from PBMC-CTL and PBMC-All and 500 cells from BoneMarrow). To
692 quantify the similarity between real and generated cells, we employed various metrics. For each
693 cell represented as a datapoint in a low dimensional embedding (e.g., t-SNE or UMAP), the
694 integration local inverse Simpson's Index (iLISI)[36] captures the effective number of datatypes (real

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)[35]. 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[24,56], we chose a kernel that is the sum of three Gaussian kernels to increase sensitivity of the kernel to a wider range:

716 $$k(x,y) = \sum_i \left(\frac{\|x-y\|^2}{\sigma_i^2} \right), i = 1,2,3, \quad (4)$$

717 where σ_i denote standard deviations and were chosen to be the median of the average distance
718 between a point to its 25 nearest neighbors divided by factors of 0.5, 1, and 2 in the three kernels,
719 respectively.

720

721 We also used a random forests (RF) classifier and used its area under the receiver operating
722 characteristic (AUROC) curve to determine whether the real and simulated cells can be
723 distinguished from each other. Consistent with previous studies[24,57], we first performed a
724 dimensionality reduction using principal component analysis (PCA) and used the top 50 PCs of
725 each cell as the input features to the RF model, which improves the computational efficiency of
726 this analysis. The RF model was composed of 1000 trees and the Gini impurity was used to
727 measure the quality of a split.

728

729 **Baseline simulator models**

730 We compared the performance of GRouNdGAN to scDESIGN2[25], SPARsim[26], and three GAN-
731 based methods: scGAN[24], cscGAN with projection-based conditioning[24], and a conditional WGAN
732 (cWGAN). The cWGAN method conditions by concatenation following the cGAN framework[58].
733 More specifically, it concatenates a one-hot encoded vector (representing the cluster number or
734 cell type) to the noise vector input to the generator and cells forwarded to the discriminator. We
735 did not train the cWGAN or cscGAN on the PBMC-CTL dataset, since it contains only one cell type.
736 For the PBMC-All and the BoneMarrow dataset, we trained all models above. Additionally, we

737 simulated data using scDESIGN2 and SPARsim with and without cell cluster information, as they
738 allow providing such side information in their training.
739
740 To train models that utilized cell cluster information, we performed Louvain clustering and
741 provided the cluster information and ratio of cells per cluster during training. Clustering was done
742 by following the cell ranger pipeline[33], based on the raw unprocessed dataset (and independent
743 of the pre-processing steps described earlier for training simulators). First, genes with no UMI
744 count in any of the cells were removed. Then the gene expression profile of each cell was
745 normalized by the total UMI of all (remaining) genes, and highly variable genes were identified.
746 The gene expression profile of each cell was then re-normalized by the total UMI of retained
747 highly variable genes, and each gene vector (representing its expression across different cells)
748 was z-score normalized. Given the normalized gene expression matrix above, we found top 50
749 principal components (PCs) using PCA analysis. These PCs were then used to compute a
750 neighborhood graph with a local neighborhood size of 15, which was used in Louvain clustering.
751 We ran the Louvain algorithm with a resolution of 0.15.
752
753 For SPARSim, we set all sample library sizes to 20000 and estimated gene expression level
754 intensities and simulation parameters by providing it with both raw and normalized count
755 matrices. When cell cluster information was provided, distinct SPARSim simulation parameters
756 were estimated per cell for each cluster. scDESIGN2 accepts input matrices where entries are
757 integer count values; we thus performed rounding on the expression matrix before fitting
758 scDESIGN2. With cluster information provided, a scDESIGN2 model was fit separately for cells of

759 each cluster, and similar to conditional GANs, the ratio of cells per cluster was provided to the
760 method.
761
762 ***In-silico* perturbation experiments using GroundGAN**
763 To perform perturbation experiments using GroundGAN, we put the trained model in a
764 deterministic mode of operation. This is necessary to ensure that the perturbation experiments
765 are performed on the same batch (i.e., replicate) of generated cells to form matched case/control
766 experiments. To do this, we performed a forward pass through the generator and then saved the
767 input noise to the causal controller, the input noise to the target generators, the TF expression
768 values generated by the causal controller, and the per-cell scaling factor of the LSN layer.
769 Subsequent passes through the generators used the saved parameters so that ensuing runs
770 always output the same batch of cells (instead of generating new unmatched cells).
771
772 **Trajectory inference and pseudo-time analysis**
773 Following official PAGA tutorial for the BoneMarrow dataset
774 (https://github.com/scverse/scanpy-tutorials/blob/master/paga-paul15.ipynb), we used
775 (Partition-based graph abstraction) PAGA[38] for trajectory inference and analysis. We built force-
776 directed graphs[59] (with ForceAtlas2[45]) using the top 20 principal components of the data (using
777 principal component analysis or PCA) and a neighborhood graph of observations computed using
778 UMAP (to estimate connectivities). We next denoised the graph by representing it in the diffusion
779 map space and computed distances and neighbors as before using this new representation. After
780 denoising, we then ran the Louvain clustering algorithm with a resolution of 0.6. Finally, we ran

781 the PAGA algorithm on the identified clusters and used the obtained graph to initialize and
782 rebuild the force-directed graph.
783
784 **GRN inference methods**
785 In our GRN benchmarking analysis, we focused on seven GRN inference algorithms: GENIE3[9],
786 GRNBoost2[8], PPCOR[37], PIDC[10], LEAP[12], SCODE[13] and SINCERITIES[14], which were used in the
787 BEELINE study[19]. Of these methods, LEAP requires pseudo-time ordering of cells, while SCODE
788 and SINCERITIES require both pseudo-time ordering and pseudo-time values. Since not all
789 algorithms inferred the edge directionality or its sign (activatory or inhibitory nature), we did not
790 consider these factors in our analysis to be consistent among different models. We ran these
791 models as docker containers using the docker images provided by BEELINE's GitHub
792 (https://github.com/Murali-group/Beeline)[19] with the default parameters used in BEELINE. To
793 benchmark algorithms requiring pseudo-time ordering of cells, we computed the pseudo-times
794 of GRouNdGAN-simulated data (based on the BoneMarrow dataset) using PAGA[38] and diffusion
795 pseudotime[39], following the methodology described earlier. In the GRN inference benchmark
796 analysis, we did not provide the list of TFs to GRNBoost2 to make it consistent with other GRN
797 inference methods.
798
799 **Competing interests**
800 The authors have no conflict of interest to declare.
801
802 **Authors' contributions:**

All authors contributed to the design of the study and the algorithm, the analyses, and the writing of the manuscript. AE supervised the study. YZ implemented the model and performed the statistical analyses. All authors read and approved the final manuscript.

Acknowledgements

This work was supported by grants from Natural Sciences and Engineering Research Council of 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). *This publication is part of the Human Cell Atlas – www.humancellatlas.org/publications.*

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. A collection of simulated datasets with known ground truth GRNs will be publicly provided upon publication to enable GRN inference benchmarking on various datasets.

824 **Code availability**

825 Our implementation and evaluation of GRouNdGAN in Python 3.9.6 using the PyTorch
826 framework[60] along with a tutorial is freely available under the GNU Affero General Public License
827 v3.0 on GitHub (https://github.com/Emad-COMBINE-lab/GRouNdGAN). The repository also
828 contains our PyTorch implementation of scGAN, projection-based conditioning cscGAN[24] and a
829 variation of cscGAN that uses conditioning by concatenation (cWGAN).

830

831 **Supplementary Material**

832 File S1: Contains supplementary notes, all supplementary figures, Tables S1-S3, and caption of
833 Tables S4-S6 (which are provided as separate files).

834

835 **References**

836 1 Lee, T. I. *et al.* Transcriptional regulatory networks in Saccharomyces cerevisiae. *science* **298**,
837 799-804 (2002).
838 2 Che, D. *et al.* Dynamic and modular gene regulatory networks drive the development of
839 gametogenesis. *Briefings in bioinformatics* **18**, 712-721 (2017).
840 3 Olson, E. N. Gene regulatory networks in the evolution and development of the heart. *Science*
841 **313**, 1922-1927 (2006).
842 4 Dibaeinia, P. & Sinha, S. SERGIO: a single-cell expression simulator guided by gene regulatory
843 networks. *Cell systems* **11**, 252-271. e211 (2020).
844 5 Yang, Y. *et al.* Gene knockout inference with variational graph autoencoder learning single-cell
845 gene regulatory networks. *Nucleic Acids Research*, gkad450 (2023).
846 6 Madhamshettiwar, P. B., Maetschke, S. R., Davis, M. J., Reverter, A. & Ragan, M. A. Gene
847 regulatory network inference: evaluation and application to ovarian cancer allows the
848 prioritization of drug targets. *Genome medicine* **4**, 1-16 (2012).
849 7 Manuel, A. M., Dai, Y., Jia, P., Freeman, L. A. & Zhao, Z. A gene regulatory network approach
850 harmonizes genetic and epigenetic signals and reveals repurposable drug candidates for
851 multiple sclerosis. *Human Molecular Genetics* **32**, 998-1009 (2023).
852 8 Moerman, T. *et al.* GRNBoost2 and Arboreto: efficient and scalable inference of gene regulatory
853 networks. *Bioinformatics* **35**, 2159-2161 (2019).
854 9 Huynh-Thu, V. A., Irrthum, A., Wehenkel, L. & Geurts, P. Inferring regulatory networks from
855 expression data using tree-based methods. *PloS one* **5**, e12776 (2010).
856 10 Chan, T. E., Stumpf, M. P. & Babtie, A. C. Gene regulatory network inference from single-cell
857 data using multivariate information measures. *Cell systems* **5**, 251-267. e253 (2017).

858 11 Emad, A. & Sinha, S. Inference of phenotype-relevant transcriptional regulatory networks
859 elucidates cancer type-specific regulatory mechanisms in a pan-cancer study. *NPJ systems*
860 *biology and applications* **7**, 9 (2021).
861 12 Specht, A. T. & Li, J. LEAP: constructing gene co-expression networks for single-cell RNA-
862 sequencing data using pseudotime ordering. *Bioinformatics* **33**, 764-766 (2017).
863 13 Matsumoto, H. *et al.* SCODE: an efficient regulatory network inference algorithm from single-cell
864 RNA-Seq during differentiation. *Bioinformatics* **33**, 2314-2321 (2017).
865 14 Papili Gao, N., Ud-Dean, S. M., Gandrillon, O. & Gunawan, R. SINCERITIES: inferring gene
866 regulatory networks from time-stamped single cell transcriptional expression profiles.
867 *Bioinformatics* **34**, 258-266 (2018).
868 15 Aibar, S. *et al.* SCENIC: single-cell regulatory network inference and clustering. *Nature methods*
869 **14**, 1083-1086 (2017).
870 16 Bravo González-Blas, C. *et al.* SCENIC+: single-cell multiomic inference of enhancers and gene
871 regulatory networks. *Nature Methods*, 1-13 (2023).
872 17 Kartha, V. K. *et al.* Functional inference of gene regulation using single-cell multi-omics. *Cell*
873 *genomics* **2** (2022).
874 18 Badia-i-Mompel, P. *et al.* Gene regulatory network inference in the era of single-cell multi-omics.
875 *Nature Reviews Genetics*, 1-16 (2023).
876 19 Pratapa, A., Jalihal, A. P., Law, J. N., Bharadwaj, A. & Murali, T. Benchmarking algorithms for
877 gene regulatory network inference from single-cell transcriptomic data. *Nature methods* **17**,
878 147-154 (2020).
879 20 Emmert-Streib, F., Dehmer, M. & Haibe-Kains, B. Gene regulatory networks and their
880 applications: understanding biological and medical problems in terms of networks. *Frontiers in*
881 *cell and developmental biology* **2**, 38 (2014).
882 21 Kolmykov, S. *et al.* GTRD: an integrated view of transcription regulation. *Nucleic acids research*
883 **49**, D104-D111 (2021).
884 22 Xu, H. *et al.* ESCAPE: database for integrating high-content published data collected from human
885 and mouse embryonic stem cells. *Database* **2013**, bat045 (2013).
886 23 Kanehisa, M. & Goto, S. KEGG: kyoto encyclopedia of genes and genomes. *Nucleic acids research*
887 **28**, 27-30 (2000).
888 24 Marouf, M. *et al.* Realistic in silico generation and augmentation of single-cell RNA-seq data
889 using generative adversarial networks. *Nature communications* **11**, 1-12 (2020).
890 25 Sun, T., Song, D., Li, W. V. & Li, J. J. scDesign2: a transparent simulator that generates high-
891 fidelity single-cell gene expression count data with gene correlations captured. *Genome biology*
892 **22**, 1-37 (2021).
893 26 Baruzzo, G., Patuzzi, I. & Di Camillo, B. SPARSim single cell: a count data simulator for scRNA-seq
894 data. *Bioinformatics* **36**, 1468-1475 (2020).
895 27 Schaffter, T., Marbach, D. & Floreano, D. GeneNetWeaver: in silico benchmark generation and
896 performance profiling of network inference methods. *Bioinformatics* **27**, 2263-2270 (2011).
897 28 Marbach, D. *et al.* Revealing strengths and weaknesses of methods for gene network inference.
898 *Proceedings of the national academy of sciences* **107**, 6286-6291 (2010).
899 29 Marbach, D. *et al.* Wisdom of crowds for robust gene network inference. *Nature methods* **9**,
900 796-804 (2012).
901 30 Chen, S. & Mar, J. C. Evaluating methods of inferring gene regulatory networks highlights their
902 lack of performance for single cell gene expression data. *BMC bioinformatics* **19**, 1-21 (2018).
903 31 Kocaoglu, M., Snyder, C., Dimakis, A. G. & Vishwanath, S. Causalgan: Learning causal implicit
904 generative models with adversarial training. *arXiv preprint arXiv:1709.02023* (2017).

32 Arjovsky, M., Chintala, S. & Bottou, L. in *International conference on machine learning.* 214-223 (PMLR).
33 Zheng, G. X. *et al.* Massively parallel digital transcriptional profiling of single cells. *Nature communications* **8**, 1-12 (2017).
34 Paul, F. *et al.* Transcriptional heterogeneity and lineage commitment in myeloid progenitors. *Cell* **163**, 1663-1677 (2015).
35 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).
36 Korsunsky, I. *et al.* Fast, sensitive and accurate integration of single-cell data with Harmony. *Nature methods* **16**, 1289-1296 (2019).
37 Kim, S. ppcor: an R package for a fast calculation to semi-partial correlation coefficients. *Communications for statistical applications and methods* **22**, 665 (2015).
38 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).
39 Haghverdi, L., Büttner, M., Wolf, F. A., Buettner, F. & Theis, F. J. Diffusion pseudotime robustly reconstructs lineage branching. *Nature methods* **13**, 845-848 (2016).
40 Trapnell, C. *et al.* Pseudo-temporal ordering of individual cells reveals dynamics and regulators of cell fate decisions. *Nature biotechnology* **32**, 381 (2014).
41 Kumar, P., Tan, Y. & Cahan, P. Understanding development and stem cells using single cell-based analyses of gene expression. *Development* **144**, 17-32 (2017).
42 Saelens, W., Cannoodt, R., Todorov, H. & Saeys, Y. A comparison of single-cell trajectory inference methods. *Nature biotechnology* **37**, 547-554 (2019).
43 Street, K. *et al.* Slingshot: cell lineage and pseudotime inference for single-cell transcriptomics. *BMC genomics* **19**, 1-16 (2018).
44 Hwang, B., Lee, J. H. & Bang, D. Single-cell RNA sequencing technologies and bioinformatics pipelines. *Experimental & molecular medicine* **50**, 1-14 (2018).
45 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).
46 Wolf, F. A., Angerer, P. & Theis, F. J. SCANPY: large-scale single-cell gene expression data analysis. *Genome biology* **19**, 1-5 (2018).
47 Gao, Y., Shen, L. & Xia, S.-T. in *ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP).* 3320-3324 (IEEE).
48 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).
49 Goodfellow, I. *et al.* Generative adversarial networks. *Communications of the ACM* **63**, 139-144 (2020).
50 Gulrajani, I., Ahmed, F., Arjovsky, M., Dumoulin, V. & Courville, A. C. Improved training of wasserstein gans. *Advances in neural information processing systems* **30** (2017).
51 Reddi, S. J., Kale, S. & Kumar, S. On the convergence of adam and beyond. *arXiv preprint arXiv:1904.09237* (2019).
52 He, K., Zhang, X., Ren, S. & Sun, J. in *Proceedings of the IEEE international conference on computer vision.* 1026-1034.
53 Glorot, X. & Bengio, Y. in *Proceedings of the thirteenth international conference on artificial intelligence and statistics.* 249-256 (JMLR Workshop and Conference Proceedings).
54 Ioffe, S. & Szegedy, C. in *International conference on machine learning.* 448-456 (PMLR).
55 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).

56 Shaham, U. *et al.* Removal of batch effects using distribution-matching residual networks. *Bioinformatics* **33**, 2539-2546 (2017).
57 Shu, H. *et al.* Modeling gene regulatory networks using neural network architectures. *Nature Computational Science* **1**, 491-501 (2021).
58 Mirza, M. & Osindero, S. Conditional generative adversarial nets. *arXiv preprint arXiv:1411.1784* (2014).
59 Islam, S. *et al.* Characterization of the single-cell transcriptional landscape by highly multiplex RNA-seq. *Genome research* **21**, 1160-1167 (2011).
60 Paszke, A. *et al.* Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems* **32** (2019).