nature protocols

PROTOCOL

https://doi.org/10.1038/s41596-020-0336-2

Check for updates

A scalable SCENIC workflow for single-cell gene regulatory network analysis

Bram Van de Sande[1,2,8], Christopher Flerin[1,2,8], Kristofer Davie[1], Maxime De Waegeneer[1,2], Gert Hulselmans[1,2], Sara Aibar[1,2], Ruth Seurinck[3,4], Wouter Saelens[3,4], Robrecht Cannoodt[3,4,5], Quentin Rouchon[3,4], Toni Verbeiren[6,7], Dries De Maeyer[6], Joke Reumers[6], Yvan Saeys[3,4] and Stein Aerts[1,2]✉

This protocol explains how to perform a fast SCENIC analysis alongside standard best practices steps on single-cell RNA-sequencing data using software containers and Nextflow pipelines. SCENIC reconstructs regulons (i.e., transcription factors and their target genes) assesses the activity of these discovered regulons in individual cells and uses these cellular activity patterns to find meaningful clusters of cells. Here we present an improved version of SCENIC with several advances. SCENIC has been refactored and reimplemented in Python (pySCENIC), resulting in a tenfold increase in speed, and has been packaged into containers for ease of use. It is now also possible to use epigenomic track databases, as well as motifs, to refine regulons. In this protocol, we explain the different steps of SCENIC: the workflow starts from the count matrix depicting the gene abundances for all cells and consists of three stages. First, coexpression modules are inferred using a regression per-target approach (GRNBoost2). Next, the indirect targets are pruned from these modules using cis-regulatory motif discovery (cisTarget). Lastly, the activity of these regulons is quantified via an enrichment score for the regulon's target genes (AUCell). Nonlinear projection methods can be used to display visual groupings of cells based on the cellular activity patterns of these regulons. The results can be exported as a loom file and visualized in the SCope web application. This protocol is illustrated on two use cases: a peripheral blood mononuclear cell data set and a panel of single-cell RNA-sequencing cancer experiments. For a data set of 10,000 genes and 50,000 cells, the pipeline runs in <2 h.

Introduction

Single-cell RNA-sequencing (scRNA-seq) has provided unprecedented resolution to the field of transcriptomics: cells can be interrogated in a comprehensive and unbiased way, enabling the characterization of complex biological processes at the cellular level. Bioinformatics methods have been specifically tailored to the analysis of scRNA-seq data and are under active development in multiple scripting languages, including several comprehensive toolkits such as Seurat[1] and Scanpy[2]. These packages include methods for quality control, pre-processing, visualization, expression-based profiling of cell clusters, and a suite of related analyses and currently serve as gold-standard examples for scRNA-seq analyses.

A key question at the heart of many single-cell experiments involves that of cellular identity and how that identity is developed and maintained. This cell identity is largely defined by an underlying gene regulatory network (GRN;cf. Box 1 for glossary and abbreviations), in which the coordinated expression of specific combinations of transcription factors (TFs) drives the expression of their respective target genes to establish a gene expression profile. As such, methods to interrogate which GRNs underlie transcriptional states and progression through state transitions are essential to revealing questions of cellular identity. Furthermore, as such methods aim to predict combinations of upstream regulators for specific cell types, they provide a handle for follow-up validation and ultimately manipulation of a cell type. We recently proposed SCENIC[3] as a method for network

[1]VIB Center for Brain & Disease Research, KU Leuven, Leuven, Belgium. [2]Department of Human Genetics, KU Leuven, Leuven, Belgium. [3]Data Mining and Modelling for Biomedicine, VIB Center for Inflammation Research, Ghent, Belgium. [4]Department of Applied Mathematics, Computer Science and Statistics, Ghent University, Ghent, Belgium. [5]Center for Medical Genetics, Ghent University Hospital, Ghent, Belgium. [6]Janssen Pharmaceutica, Beerse, Belgium. [7]Data Intuitive, Ghent, Belgium. [8]These authors contributed equally: Bram Van de Sande, Christopher Flerin.
✉e-mail: stein.aerts@kuleuven.vib.be

Box 1 | Glossary and abbreviations

Acronym	Term
API	application programming interface
AUC	area under the curve
bp	base pair
CLI	command line interface
CPM	counts per million
CRM or enhancer	cis-regulatory module or enhancer
CSV	comma separated value
DREMI	density resampled estimate of mutual information
EMT	epithelial-to-mesenchymal transition
FDR	false discovery rate
FET	Fisher's exact test
FPKM	fragment per kilobase per million reads
GENIE3	gene network inference with ensemble of trees
GEO	Gene Expression Omnibus
GRN	gene regulatory network
HDF5	hierarchical data format 5
HNSC	head and neck squamous cell carcinoma
HPC	high-performance computing
HVG	highly variable gene
KDE	kernel density estimation
kNN	*k*-nearest neighbor graph
LUAD	lung adenocarcinoma
LUSC	lung squamous cell carcinoma
MAGIC	Markov affinity-based graph imputation of cells
NES	normalized enrichment score
NHR	nuclear hormone receptor
NNLS	non-negative least squares
PBM	protein-binding microarray
PBMC	peripheral blood mononuclear cell
PCA	principal component analysis
PPI	protein–protein interaction
PTM	Pavlidis template matching
PTM	post-transcriptional modification
PWM or motif	position weight matrix or motif
QC	quality control
RSS	regulon specificity score
SCENIC	single-cell regulatory network inference and clustering
scRNA-seq	single-cell RNA sequencing
SKCM	skin cutaneous melanoma
TF	transcription factor
TFBS	transcription factor binding site
TI	trajectory inference
TME	tumor microenvironment
t-SNE	*t*-distributed stochastic neighbor embedding
TSS	transcription start site
UMAP	uniform manifold approximation and projection
UMI	unique molecular identifier
UQ	upper quartile (normalization)
WGCNA	weighted gene coexpression network analysis

Term	Description
Regulon	The small gene regulatory network consisting of a transcription factor connected to its direct target genes.
Targetome	The collection of all target genes of a transcription factor.

inference and motif discovery, allowing high-confidence prediction of key regulators and their direct target genes.

The continued advance of single-cell technology has pushed boundaries, with decreasing cost and technological improvements contributing to an increase in the size of data sets, on both the cell and gene levels. In response, and to anticipate the increasing computational demand of single-cell data sets, we set out to improve upon the original SCENIC implementation. Here we describe a refactored SCENIC workflow and provide a well-documented, fast, robust and easy-to-use computational toolbox. From the original R/Bioconductor implementation[3], we ported SCENIC to Python (resulting in pySCENIC), improving the run time through both parallelization and an improvement in computational efficiency. Additionally, we provide SCENIC in software containers, facilitating reproducibility and ease of deployment on consumer-grade computer hardware or in multi-node high performance or cloud-based computing environments. The entire workflow can be managed via Nextflow[4] or via an interactive Jupyter notebook (https://jupyter.org/), both of which are included in this protocol.

Workflow

Briefly, the SCENIC pipeline consists of three steps. First, candidate regulatory modules are inferred from coexpression patterns between genes (Steps 5 and 6). Next, coexpression modules are refined by the elimination of indirect targets using TF motif information (Step 6). Finally, the activity of these discovered regulons is measured in each individual cell and used for clustering (Steps 7 and 8; Fig. 1).

Pre-processing (Steps 1–4)

The SCENIC workflow starts with an expression matrix capturing the abundance of each gene's transcript in every cell interrogated in an scRNA-seq experiment. Typically, each cell is represented as a separate row in this matrix, whereas genes are depicted as columns. Although the matrix can be supplied in CSV format, the digital loom file format for storing single-cell gene expression profiles and its metadata is also supported (http://loompy.org/).

Many methods for mapping and quantifying gene transcripts from sequenced reads have been adapted to the requirements of single-cell protocols (e.g., STARsolo[5] and the pseudo-aligner Kallisto[6]). In addition, scRNA-seq-specific methods have been developed (e.g., Cell Ranger from 10x Genomics (https://www.10xgenomics.com/), zUMI[7], or the pseudo-aligner Alevin[8]). The SCENIC workflow works with the outputs of all these pipelines.

A minimal amount of filtering is needed before running a SCENIC analysis. On a cell level, we examine the number of expressed genes and remove cells that fell into the distribution extremes. In the peripheral blood mononuclear cell (PBMC) study case (Table 1), we discard cells with <200, and more than ~5,000, expressed genes; however, these thresholds must be determined empirically. We further filter out cells that have a large fraction of mitochondrial gene transcripts; these cells are thought to be of lower quality as this is indicative of cell membrane breach[9]. We again use the empirical distribution to select an upper threshold on mitochondrial genes expressed, and this is highly dependent on cell type but typically 5–15%. Finally, on the gene level, genes with low overall expression are removed; with our default settings, we remove genes expressed in fewer than three cells in the data set.

Many toolkits and guidelines are available to perform these pre-processing steps, with notable standouts in the R-based Seurat[1] (https://satijalab.org/seurat/) and the Python-based Scanpy[2] (https://scanpy.readthedocs.io/en/latest/index.html). For this protocol, we use Scanpy, which is efficient and scalable to large data sets, to maintain compatibility and interoperability with the Python-based version of SCENIC.

Network inference (Step 5)

In a first step, given a predefined list of TFs, regulatory interactions between these factors and putative target genes are inferred via regression-based network inference[10] from the expression or count matrix. More specifically, for every expressed gene, a regression model is built that predicts the expression of this gene across cells (the response) from the expression of these predefined TFs (the independent variables or regressors). A regulatory interaction between gene and TF is inferred by assessing the importance of these factors in the regression model.

Our workflow relies on a tree-based regression model and, by default, uses an efficient and distributed implementation based on gradient boosting machine regression (i.e., GRNBoost2 (ref. [11])). This algorithm is a reimplementation of the GENIE3 algorithm, which was based on a random forest[10]. These tree-based approaches are able to capture combinatorial transcriptional

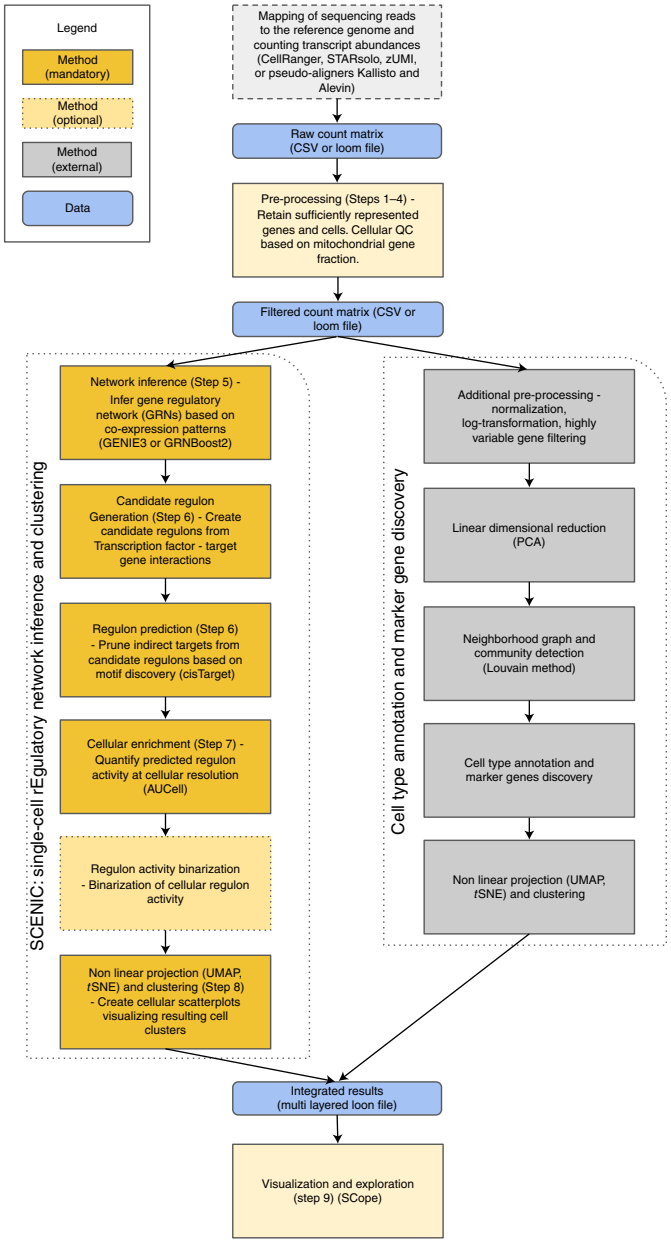


Fig. 1 | Schematic overview of the pipeline. Flowchart describing the steps of the SCENIC pipeline and the integration with general cell-type annotation and marker gene discovery pipelines.

control and complex interaction patterns with only a limited increase in the overall computational burden. In practice, the user of our pipeline can still opt for GENIE3, which is built into the pySCENIC framework. Other network inference algorithms can be substituted at this step with minor adaptations to the workflow.

The output of this step is a list of adjacencies connecting a TF with a target gene. A weight or importance is associated with these connections to distinguish strong from weak regulatory interactions.

Module generation (Step 6)

From these regulatory interactions, inferred from coexpression patterns, modules (i.e., a TF and its predicted target genes) are generated[12]. Multiple strategies are combined so that, for every factor, multiple modules of different sizes are created. Modules are created:

1 Based on the top N targets for each factor, with a default of 50 targets.

2 Based on the top N regulators for a target; the default settings select sets of 5, 10, and 50 regulators.

Table 1 | Overview of data sets used for case studies

Data set type	Data set source and identifier	# of samples/ patients	# of cells
Mouse somatosensory cortex S1 and hippocampal CA1 region[30]	https://www.ncbi.nlm.nih.gov/geo/—GSE60361	33 males, 34 females	3,005
PBMCs	https://support.10xgenomics.com/single-cell-gene-expression/datasets/3.0.0/pbmc_10k_v3	1 healthy donor	11,769
Skin cutaneous melanoma (SKCM)[40]	https://www.ncbi.nlm.nih.gov/geo/—GSE115978	31 patients	7,186
Head and neck squamous cell carcinoma (HNSC)[41]	https://www.ncbi.nlm.nih.gov/geo/—GSE103322	18 patients	5,902
Lung adenocarcinoma/ lung squamous cell carcinoma (LUAD/LUSC)[42]	https://www.ebi.ac.uk/arrayexpress/—E-MATB-6149 and E-MTAB-6653	5 patients	52,698

3 Based on a percentile score; the default is to select sets based on the 75th or 90th percentile of a factor's targets.

Subsequently, modules are differentiated between transcriptional activation and repression based on the correlation patterns between the expression of the regulator and its targets: targets that are positively correlated (Pearson product moment correlation; default $\rho \geq +0.03$) are assigned the activating module, whereas targets that are negatively correlated (default $\rho \leq -0.03$) are part of the repressing module. A low threshold of 0.03 is purposefully chosen so that linear, as well as more complex nonlinear, regulatory interactions identified by GRNBoost2/GENIE3 are not filtered out and pass through to the next step (cisTarget—Step 6). By default, only positively correlated modules are used in the subsequent steps because these negatively correlated modules are less numerous, their regulatory interaction weights are lower and the resulting regulons show overall less motif enrichment[3]. By default (as of version 0.9.18), pySCENIC calculates the correlation coefficient using the entire expression matrix, including dropouts. This avoids situations in which the correlation coefficient is determined by a small subset of cells with positive expression in the two genes being assessed. However, it is also possible to mask the cells that are involved in a dropout event for any of the two genes.

Finally, the resulting modules with <20 genes are not retained, as these modules tend to be less stable and more sensitive to target gene dropout. We end up with a final collection of modules. These modules are defined by the name of the regulator (TF), the list of target genes (which may be direct or indirect), the method of generation (top targets, top regulators, or top percentile targets) and the mode of transcriptional control (activation or repression). A factor can have multiple modules, but these modules are different in the method of generation and/or the transcriptional mode.

Motif enrichment and TF-regulon prediction (cisTarget step; Step 6)

Modules contain direct and indirect targets of a regulator because these regulatory interactions are only inferred based on coexpression patterns. Therefore, in this step, the putative regulatory regions of these target genes are searched for enriched motifs by comparing scores of cis-regulatory modules (CRMs) near the genes in the module, with the remaining genes in the genome. CRM scoring is done using hidden Markov models, with a large collection of position weight matrices (PWMs) following the procedures described in iRegulon[13] and i-cisTarget[14,15]. If the motif of the regulator TF is significantly enriched in one of its modules, this regulator and its predicted targets are retained for further analysis.

Technically, the cisTarget motif enrichment procedure relies on a ranking and recovery approach (Box 2): genome rankings are pre-computed for all motifs in our collection that are linked to a known TF. Multiple versions of whole-genome rankings are available as databases for three target species (*Homo sapiens*, *Mus Musculus*, or *Drosophila melanogaster*) (https://resources.aertslab.org/cistarget/). The databases vary slightly in two parameters: first, the boundaries of the putative regulatory region for genes (e.g., 500 bp upstream of the transcription start site (TSS) of a gene, 5-kb or 10-kb centered around TSS), and, second, the set of orthologous species searched for network-level conservation of CRMs (e.g., seven or ten species for *H. sapiens* and *M. musculus*, and 11 species for *D. melanogaster*).

Box 2 | Ranking and recovery framework

An efficient framework for cis-regulatory motif discovery lies at the basis of the regulon prediction or cisTarget step in the SCENIC workflow (Step 6). This framework was developed for iRegulon[13] and consists of two phases: (i) a ranking phase, which models transcriptional regulation by a TF as a genome wide ranking of genes, and (ii) a recovery phase, which quantifies the involvement of a factor in a set of genes (or gene signature) using a recovery-based statistic.

Whole-genome rankings as a universal model for co-regulation
For a large collection of PWMs or motifs, whole-genome rankings are pre-calculated by scoring putative regulatory regions for the presence of homotypic clusters of motif instances (via the hidden Markov model implementation of cluster-buster[45]). These PWMs model the sequence-specific binding of a TF. A different definition of a gene's regulatory region results in a different ranking database: 500 bp upstream of TSS captures predictions of TFBSs in the proximal promoter, and 5-kb centered around the TSS or TSS ± 10-kb capture more distal enhancers (i.e., clusters of TFBSs).
These motif-based gene rankings are further fine-tuned by scoring orthologous regulatory regions (mapped from *H. sapiens* or *M. musculus* to other species via LiftOver[46]) and integrating these rankings for orthologous species into a single ranking (using OrderStatistics[47]). Genes for which a homotypic cluster is conserved in other species will be ranked higher in the final ranking, reflecting the higher confidence in the prediction. This approach is also known as network-level conservation, and it takes into account functional conservation of TF–gene interactions in the GRN, improving the quality of prediction.

Generalization to epigenomic track discovery
In iRegulon[13] and i-cisTarget[14,15], which are both interactive tools for integrating and predicting regulatory features, gene sets can be analyzed for the enrichment of ChIP-seq peaks of publicly available ChIP-seq data sets, such as ENCODE[48]. We have used genome-wide rankings of all genes, based on ChIP-seq peak heights, in SCENIC to provide complementary results that can be viewed alongside the motif analysis.

Recovery-based approach to motif discovery
Given the pre-calculated whole-genome rankings for a comprehensive motif collection, motif discovery for a given set of genes as input (typically referred to as a gene signature) involves scanning the database for rankings in which the top-ranked fraction is enriched for this input set of genes. More specifically, the cumulative recovery of the foreground set in a whole-genome ranking is quantified using an AUC statistic. The AUC values are standardized (i.e., by mean subtraction and scaling by the standard deviation) and expressed as NESs. Motifs associated with an NES >3.0 are considered as enriched for the supplied signature. This corresponds to a FDR of 3–9% (ref. [13]).

From motif discovery to GRN reconstruction
To assign a TF to an enriched motif, SCENIC uses the component 'motif2TF'. This component maximizes the number of motifs in our collection for which a factor annotation is defined by integrating orthology between factors, similarity between motifs, and conversions between gene nomenclatures into a relational database. Orthologous relationships between TFs from Ensemble GeneTrees[49] are used to leverage annotations in different species, whereas motif similarity enables the recruitment of 'orphan' motifs (for example, computationally derived motifs without annotated factor). Pairwise similarity between motifs is calculated via TomTom[50]. Parameters for the motif2TF step include the feature AUC threshold, a threshold on rank to derive target genes and a threshold on the NES to select features. These parameters are user selectable; however, in practice, the defaults are used for a first pass on new data.
To define the subset of genes in the input set in control of the enriched factor, SCENIC defines a leading edge (i.e., an optimal threshold) on the factor's whole-genome ranking. Genes in the input set ranked higher are predicted direct targets of the enriched factor. This leading edge is defined by the maximum deviation between the recovery curve of the enriched motif associated with the factor and the pointwise average recovery across the motif collection plus two standard deviations. This approach is similar to the leading edge used by gene set enrichment analysis[51].

Regulon prediction (Step 6) and cellular enrichment (Step 7) reformulated as a ranking and recovery problem
Regulon prediction (Step 6) from a list of inferred direct and indirect regulatory interactions between a TF and its target genes can be reformulated as a ranking and recovery problem. Concretely, SCENIC scans for motifs in which the whole-genome ranking has steep initial recovery of a factor's direct and indirect targets. Only the genes that make up the leading edge of a motif annotated for the factor are retained in the final predicted regulon. If no factor-annotated motif is enriched, the input module is discarded. The union of all leading edges defines the final targetome if multiple motifs are enriched and annotated for the factor.
Similarly, to quantify the activity of these predicted regulons (Step 6) in the individual cells interrogated in a single-cell experiment, AUCell (Step 7) calculates the recovery of the predicted targetome of a factor on the expression abundance-based ranked list of genes in a cell. Hence, AUCell translates the quantification of a factor's cellular activity as a ranking and recovery problem.

An area under the curve (AUC) metric is used to assess significant recovery of a set of genes for a given whole-genome ranking.

For each enriched motif, cisTarget also leverages this ranking and recovery framework to determine a leading edge in the recovery curve and thereby prune the target genes of a module. Finally, the sets of predicted direct target genes across all modules sharing the same regulator are combined into a single resulting regulon.

As an optional addition to the motif enrichment analysis, SCENIC also incorporates the capability to perform enrichment using epigenomic track-based databases. These are generated from publicly available ChIP-seq experiments targeting TFs (such as those from ENCODE v3), and the resulting database provides a genome-wide ranking of all genes based on the average read density in the regulatory regions associated with each gene. Using these track databases, the module refinement steps are carried out using the same approach as the motif enrichment, whereby modules are selected and pruned when the upstream regulator (predicted by GRNBoost2) has enriched ChIP-seq peaks across its predicted target genes. Thus, with ChIP-seq track enrichment, the direct target genes in a regulon are based on actual TF binding sites (TFBSs). The use of track-based databases here is

optional but has the capability to provide an independent set of regulons that can provide complementary information alongside the motif-based regulons.

Cellular enrichment (AUCell step; Step 7)

A similar ranking and recovery framework is used to quantify the activity of the predicted regulons in the individual cells that make up the scRNA-seq experiment. In detail, each individual cell's transcriptome is modeled as a whole-genome ranking based on the expression of its genes. The enrichment of a regulon is subsequently assessed via recovery of its targetome on the cell's whole-genome ranking. The AUC metric measures the relative biological activity of a regulon in a given cell.

(Optional) Binarization of cellular regulon activity

The distribution of the cellular AUC values for a regulon has several shapes: the bimodal distribution reflects the presence of two distinct cell populations, whereas a skewed Gaussian distribution represents a more continuous activation pattern across the assayed cells.

To binarize the activity of the distribution and facilitate further statistical analysis, a two-component Gaussian mixture model is fitted to define an AUC threshold. This threshold labels cells as 'on' or 'off' for a given regulon. Skewed Gaussian AUC distributions are binarized by labeling cells part of the right tail as 'on'.

In some cases, the model does not provide a good fit to the AUC values distribution for a particular regulon, and we therefore recommend that the user manually verify the biological validity of the binarization. This can be accomplished via the web-based SCope visualization tool (see Step 7 below), which allows the distribution to be visualized and the threshold to be explored interactively.

Clustering of cells based on regulon activity (Step 8)

Quantification of regulon activity via AUCell is a biological dimensional reduction: the number of discovered regulons (k) is typically much lower than the number of genes (n), and therefore each cell can be represented by a k-dimensional vector instead of being represented by a point in n-dimensional space.

This eliminates the need to reduce the dimensionality via principal component analysis (PCA) before applying a nonlinear projection technique such as t-distributed stochastic neighbor embedding (t-SNE) or uniform approximation and projection (UMAP) for rendering visual groupings[16]. In addition, this regulon-based clustering also reveals the GRNs underlying the gene expression profiles.

The loom file format, which is a common and well-supported standard for storing and exchanging processed single-cell experiments, is the output generated by the SCENIC pipeline. It is based on the international standard for numeric data storage HDF5 (https://www.hdfgroup.org/solutions/hdf5/) and provides additional abstraction layers for storing expression matrices, embeddings, and cellular and gene-oriented metadata. CSV is also supported by SCENIC, but, for the loom output format, the stored expression matrix is augmented with row metadata capturing the regulons (i.e., the TF, the enriched motif and the target genes), whereas the AUCell values are represented as metadata associated with the samples and columns.

The loom file format can capture multiple embeddings and can therefore be used to store the output of other single-cell workflows. A typical workflow provided by Seurat or Scanpy is cell-type annotation and marker gene discovery (Box 3). These workflows provide nonlinear cellular projections (e.g., t-SNE on the dimensionally reduced gene space based on PCA) and cluster cells based on community detection algorithms run on nearest-neighbor graph abstractions of phenotypic similarities between cells. These clusters can be annotated for cell type based on differentially expressed genes between the discovered clusters. All output from these parallel workflows can be combined in the same loom file, which thereby provides a comprehensive and combined overview of a single-cell transcriptomics experiment. A template for this is provided in both the Jupyter notebook and Nextflow implementations of this workflow.

To discover regulators for the cell types defined by the aforementioned pipelines, several metrics are available to measure cell-type specificity of the predicted regulators by SCENIC (Box 4).

Interactive visualization and exploration via SCope (Step 9)

To facilitate the visualization and exploration of the single cells and their regulons, the results of this analysis pipeline can be exported and loaded into SCope[17]. SCope is an application that visualizes cellular scatter plots in an interactive way: the user can zoom in on the activity of genes, regulons and cell types (labeled via different color codes) for the same experiment. In addition to the cellular

Box 3 | Comparison and integration with standard scRNA-seq analysis

SCENIC is a specialized tool focused on GRN inference; however, it is essential to interpret the SCENIC results together with more general analyses of single-cell transcriptomics data. Our approach to this begins with basic filtering and then proceeds with pre-processing and further transformation that feeds into the general analysis steps, which include visualization, clustering, and cell annotation. For these procedures, we have generally used steps that have been recently published in a best practices guide outlined in ref. [52]. We outline these general steps in this section, but modifications are possible based on prior biological knowledge of the experimental design. In principle, there are several tools in different languages that could be used to accomplish these steps, but we use the Python-based Scanpy[2], which can be installed and used alongside pySCENIC.

Starting with the minimally filtered expression matrix (described in full in stages 1–4 of the Workflow (pre-processing) and in the PMBC case study), the pre-processing step applies several transformations to the data. We first normalize the expression matrix to have the same total counts per cell and then multiply by a scale factor (10,000 by default). The expression matrix is then log transformed, and a set of HVGs is selected based on the ratio of variance to mean. Technical effects are then regressed out of the data using both counts and percentage of mitochondrial reads as covariates. Finally, each gene is scaled to unit variance, with a maximum value set to 10 standard deviations above the mean.

Having obtained a reduced set of HVGs from the pre-processing step, we next reduce the dimensions of the expression data using PCA. The resulting eigenvalues indicate how much variance is explained by each component, and we select a top set of principal components (PCs) by looking for a leveling off in the curve of variance explained plotted against PC number. The top N PCs are then used to compute a neighborhood graph of the observations (cells). This graph can be visualized using *t*-SNE[53]. We also include another dimensionality reduction method, UMAP[16], which can provide a better picture of global cell connectivity.

We next cluster the cells using the Louvain method, which is widely used for community detection[54]. The Louvain method predicts clusters based directly on the neighborhood graph of cells previously computed and does not need prior knowledge of the number of clusters to be expected in the data. The number of clusters obtained can be controlled by adjusting the resolution parameter. These predicted clusters can be supplemented by other annotation methods, such as Garnett[55], an automated classifier based on expression of specific genes or manual curation.

The final step is to integrate the results of both this and the SCENIC pipelines together to produce a SCope-ready loom file. This file includes the full expression data, multiple dimensionality reductions from both SCENIC and the general analysis, the SCENIC regulons and cell-based annotations, such as clustering labels or quality control data.

activity of regulons, the enriched motifs and the predicted target genes of a regulon can be examined and exported. Moreover, analysis results from Scanpy can be compared with SCENIC results, or different scRNA-seq analyses can be compared. It is available as a web application (http://scope.aertslab.org) and can also be installed locally.

Applications of the method

This protocol applies to the reconstruction of GRNs from single-cell transcriptomics experiments. It can also be applied to large-scale bulk RNA-seq experiments that interrogate multiple conditions[18]. In this case, AUCell does not measure the biological activity of a regulon for a given cell but instead for the interrogated condition.

We currently provide databases for the following species: *H. sapiens*, *M. musculus*, or *D. melanogaster*. There is no limit to the number of cells in the experiment.

Advantages

User-friendliness

Straightforward installation of pySCENIC is achieved by making it available via Python's standard online package repository (PyPI). In addition, use on Linux-based systems is facilitated via the availability of containerized versions of the workflow (Box 5).

An easy-to-use command line interface (CLI) facilitates adoption of our method on Linux-based high-performance computing (HPC) infrastructure and enables integration with the R version of the method. Instructions for integration with R are provided in online vignettes (https://github.com/aertslab/SCENIC/blob/master/vignettes/importing_pySCENIC.Rmd). In addition, researchers favoring Python for their data analysis can make use of an API tailored to Jupyter notebooks.

Scalability and time efficiency

Almost all steps in the SCENIC pipeline are 'embarrassingly parallel', and this characteristic is exploited in this implementation (Steps 5–7): the workload can be parallelized by splitting the input data in multiple chunks that can be processed independently. The final results are easily reconstituted

Box 4 | Cell type specificity of discovered regulons

To better characterize the regulatory networks underlying the different cell types in an scRNA-seq experiment, the discovered regulons can be assessed for their cell-type specificity. Cell-type annotations for the cells in a scRNA-seq experiment can be generated using a general workflow as explained in Box 3 and are typically provided as metadata of publicly available data sets.

Similarly to the general approach for marker gene discovery for cell types, parametric (*t*-test) or nonparametric (Wilcoxon rank-sum test) methods can be used to find regulons with a statistically significant difference in AUCell score compared to all other cell types grouped together. In the same vein, the binarized activity of regulons can be relied on to find regulons in which 'on' state co-occurs with a given cell type using Fisher's exact test or the chi-squared test.

Recently, a novel method was developed and used to quantify the specificity of the regulons across different cell types[38]. For this method, the RSS does not require binarization of a regulon's enrichment score distribution and measures the distance between this distribution and the distribution of cell-type annotations using the Jensen–Shannon Divergence[38]. For a given cell type, the RSS for all predicted regulons is ranked from high to low, and highly cell-type-specific regulons are spotted as outliers. This RSS metric has recently been incorporated into the SCENIC codebase, and its use is demonstrated in the linked Jupyter notebooks.

Box 5 | Software containers and workflow system

An increase in new bioinformatic analysis methods for scRNA-seq has raised critical questions on reproducibility and usability of these methods. Reproducibility with computational tools means the ability for a researcher to obtain and use a publicly available tool, and have that code be identical to, and produce the same results as the original publication.

One way we address this code reproducibility issue here is to package software tools into virtualized packages called containers, which encapsulate the code into fully reproducible and portable computing environments. Two commonly used container systems are Docker (https://www.docker.com/) and Singularity (https://sylabs.io/singularity/)[21]. Although colloquially referred to as 'containers', an important distinction between images and containers is that an image is an immutable snapshot of a set of software packages and the parent environment, whereas a container is a running or active instance of an image. One or more containers can be started from the same image (e.g., to perform computations on different datasets or with varying parameters).

The use of a container system solves several issues a researcher might face when attempting to integrate a new tool into their workflow. Images are highly portable and can be easily transferred, allowing the same codebase to run on a wide spectrum of computing environments. The portable nature of images bypasses installation troubles: instead of the often tedious process of downloading and compiling the software and its dependencies, the image can be simply downloaded from a public repository or built from a recipe file provided alongside the codebase. In addition, the container provides isolation of the codebase and running processes from other processes within the host operating system. The static nature of the image means that any future researcher to use the image will receive the codebase exactly as it was packaged, ensuring interoperability of package dependencies.

Another convenience feature employed in this protocol is the use of the Nextflow workflow management system, which, in parallel with the containers, facilitates portability and reproducibility. SCENIC is by nature a pipeline, a series of computational tools (GRNBoost2, cisTarget and AUCell) applied in sequence to a data set. With this pipeline comes an issue of complexity: each tool has its own set of input and output files that need to be formatted properly, as well as specific parameters and reference files. Although every effort has been made to make these parameters well documented and transparent to the end-user, SCENIC remains a relatively complex pipeline to execute. Workflow management tools (e.g., Nextflow) address the complexity issues inherent in this protocol. The Nextflow framework facilitates automatic handling of the inputs and outputs between the steps. Using the Nextflow implementation provided (https://github.com/aertslab/SCENICprotocol), this entire protocol can be run automatically with a single shell command. However, this convenience comes with some potential drawbacks, primarily that no data set is properly captured with one set of predefined parameters. Manual intervention is necessary, especially in the filtering step (Step 3). In the Nextflow system, this could potentially be addressed running the workflow in two passes: the first pass to generate diagnostic plots of the empirical distribution of UMI counts, expression sparsity, etc., and the second pass to run the pipeline from the start using the user-defined parameters based on the initial diagnostics. Such a pipeline could be a valuable component to a semi-automated comprehensive system for analysis of scRNA-seq data that could scale to many samples.

from the processing of these individual chunks. A current trend in single-cell protocols is the exponential increase in the number of cells being interrogated per experiment[19]. Fortunately, the scalable nature of this reimplementation will assure the applicability of our method to data sets with hundreds of thousands of cells[19].

An efficient Python implementation ensures that the entire workflow can be executed in reasonable timeframes (Table 2).

The complexity of GENIE3 (Step 5) is log-linear (i.e., $n \times \log(n)$) in the number of cells, n, and, in the worst case, quadratic in the number of genes measured for these cells[20]. The time complexity of the subsequent cisTarget step (Step 6) is mainly driven by the number of modules to be pruned

Table 2 | Software run time for the selected case studies

Data set	Genes	Cells	Network inference—GRNBoost2 (Step 5)		Candidate regulon generation and regulon prediction (Steps 6A and 6B)		Cellular enrichment—AUCell (Step 7)	
			CPU time (min)	Memory (GB)	CPU time (min)	Memory (GB)	CPU time (min)	Memory (GB)
PBMCs	20,292	10,280	95	34	8	23	1.7	4.3
Mouse somatosensory cortex S1 and hippocampal CA1 region[30]	19,970	3,005	11	11	4	13	<1	4
Skin cutaneous melanoma (SKCM)[40]	22,527	7,186	32	21	4.4	27	<1	4.2
Head and neck squamous cell carcinoma (HNSC)[41]	21,519	5,902	26	18	5.4	24	<1	4.1
Lung adenocarcinoma/lung squamous cell carcinoma (LUAD/LUSC)[42]	9,919	51,628	95	52	4.5	23	1.3	23

These data sets were timed on machines equipped with an Intel Xeon Gold 6140 CPU (18 cores, 36 threads), with 192 GB of RAM, using 20 processes.

(i.e., the output of GENIE3/GRNBoost2) and the number of motif and track rankings used for pruning these modules. The latter is defined by the number of motifs available and the number of putative regulatory regions defined for each gene in the genome (Box 2). AUCell (Step 7) scales linearly with the number of cells in the experiment and the number of predicted regulons to be scored for each cell.

Reproducibility

Containerized SCENIC solutions are available via Docker (https://www.docker.com/) or Singularity[21], and guarantee ubiquitous deployment and reproducibility of the results. The SCENIC containers eliminate the possibility of idiosyncratic infrastructure incompatibilities and remove breaking updates in direct and indirect package dependencies (Box 5).

Robustness

The various steps of the pySCENIC pipeline are robust against the technical dropouts prevalent in single-cell protocols. In a recent benchmark study for GRN inference from single-cell transcriptomics data sets, which includes assessment of the effect of dropouts, GRNBoost2 (Step 5) was among the best-performing methods[22]. The subsequent cisTarget step (Step 6) is inherently unaffected by a few missing genes in a module because it relies on a ranking- and recovery-based approach to motif discovery (Box 2). AUCell (Step 7) quantifies the activity of the discovered regulons based on the enrichment of the entire predicted targetome of a factor. This renders the method less susceptible to a few missing target genes. As a consequence, the threshold that defines the binarization of a regulon's cellular activity is not affected by the absence of some of its targets.

Limitations

Stochastic nature of the method

It is important to note that SCENIC is a stochastic algorithm and should not be expected to produce precisely the same results (i.e., output regulons) for repeated applications to the same input expression matrix. The stochastic nature is due to the GRN inference step. Both GENIE3 and GRNBoost2 use regression tree approaches, and both use repeated random subsamplings of genes to effectively model the expression of a given target gene based on a set of TFs. As a result, the weights of the gene–regulator interactions in the final network vary slightly, and genes with similar weights might be ranked differently in repeated runs in the network adjacency matrix produced at the end of Step 5.

As a downstream result of these ranking differences, the final set of regulons identified at the end of the SCENIC procedure, as well as the precise target genes, might differ slightly. From benchmarking tests, this variation presents itself in two ways: first, by dropout of whole regulons and, second, in variation in the set of genes identified as targets of each TF. In whole-regulon dropout, much of the variability comes from regulons in which the TF targets a small set of genes (20–50), whereas regulons with many genes (200 or more) are more consistently identified. Among these more

consistent regulons, the set of target genes also varies, although this does not appear to depend on the regulon size. Despite this variation in regulon identification, benchmarking tests have shown SCENIC to produce results that are robust across multiple sample types, and we expect the consequences of this variation to be relatively minor in the context of this workflow.

There are two possible workarounds for this issue. One is to set the random seed used in the GRNBoost2 step to a fixed value, which stabilizes the network output but does not address the underlying uncertainty. The second workaround is to run the full SCENIC pipeline multiple times and compare the regulon endpoints. Regulons driven by TFs that appear less than a certain number of times can be identified as low confidence and discarded entirely, and target genes are able to be refined in the same way. In our tests, we have run the full SCENIC pipeline up to 100 times on the same data set and used the results to filter out inconsistent regulons. However, we recognize that this is a computationally challenging procedure and might not be possible in all cases. As a further possibility, the repeat runs could include only a few TFs of high interest, based on biological insight or analysis of the first-pass run.

GENIE3/GRNBoost2 (Step 5) cannot infer regulatory interactions controlled via post-transcriptionally modified regulators

Network inference methods rely on the assumption that a TF's regulatory function is correlated with its expression level. Although this premise is reasonable for most factors, for some families of TFs the gene regulatory activity is controlled via post-translational modification (PTM) of the factor itself or via binding of cofactors or ligands to the factor. For example, the class of nuclear hormone receptors (NHRs) is known to migrate to the cellular nucleus and activate the expression of their targets only after binding a steroid hormone.

CisTarget (Step 6) cannot predict regulons for a factor with unknown motif

According to a recent survey of human TFs, there are currently 1,200 TFs with a known motif out of the 1,600 estimated total[23]. Although the cisTarget step enables the SCENIC pipeline to better control the false-negative rate of predicted targets of a regulator compared to most other GRN reconstruction methods, this step renders the pipeline blind for the effect of regulation via factors with unknown motif.

The risk of not discovering an essential regulator is reduced via motif2TF (Box 2), which expands the number of TFs for which a motif is available by incorporating orthologuous relationships between factors. Moreover, this limitation can be mitigated by using track-based databases. These databases are generated from whole-genome tracks derived from ChIP-seq experiments against a TF and rank all genes in the genome based on the average read density in the putative regulatory regions associated with each gene.

Software and complexity

SCENIC by nature is a complex pipeline of several distinct analysis steps, and it could prove difficult to install or run or to understand the method and results. This protocol attempts to address each of these. Install issues are addressed with detailed instructions (for a Conda-based install), and the container images are available for immediate download and use, provided Docker or Singularity is available. However, container systems bring their own potential issues, including HPC compatibility and volume mounts, even though their advantages are increasingly clear in modern research settings. The detailed steps provided in this protocol are aimed at providing additional details on the procedure, algorithmic workings, and interpretation of a SCENIC analysis. Finally, this method is written in Python, which might present challenges for an experimental researcher used to the R language. To decrease this friction, we have made the key steps in pySCENIC accessible for use through the CLI, enabling an analysis to be completed without the use of Python.

AUCell (Step 7) provides minimal comparability

The metric used by AUCell to quantify the activity of a predicted regulon in individual cells of an experiment is an unnormalized enrichment score. Therefore, it can only be used to compare the activity of a regulon across cells of the same experiment. Because the enrichment score is not normalized for the size of the predicted targetome, the scores of different regulons cannot be compared.

Comparison with other tools

Many other tools that help identify cellular identity by unraveling the underlying regulatory networks are available[24]. To our knowledge, almost all of these methods mainly rely on coexpression patterns to predict regulatory interactions between a regulator and its target genes. SCENIC combines this approach with an additional pruning step for indirect targets of a regulator based on motif or track discovery. This additional computational step reduces the false-positive rate of predicted targets and makes SCENIC an integrative method[25]. Moreover, some of the computational methods require prior knowledge of the targetomes of the TFs, further limiting their predictive power and their applicability. Notable methods include SINCERA[26], ACTION[27], MAGIC[28], and SCINGE[29], and these methods could be used in place of SCENIC's default, GRNBoost2, with modifications to the procedure outlined here.

A recent benchmarking study attempted to formally benchmark 12 GRN methods in common use[22]. The methods tested covered those requiring pseudotemporally ordered cells and those that infer regulatory networks using no prior information aside from the expression matrix. The comparison included the same implementations of both GENIE3 and GRNBoost2 that are used in pySCENIC (from the Arboreto package), as well as other methods that could be potentially substituted into the GRN step of our SCENIC pipeline. The performance of these methods was largely disappointing, showing inconsistent results on simulated data of different network types, suggesting that each method is tailored toward identifying a specific subtype of regulatory interaction within the network. This benchmarking study illustrates the challenge of inferring accurate GRNs from single-cell data, and that combining these methods to generate a more complete network with many interaction types would be advantageous although difficult to achieve. This further highlights the usefulness of our integrative method in SCENIC, which augments initial findings provided by GRN inference with further refinement via the downstream ascertainment of cis-regulatory evidence (cisTarget step) to produce a more robust picture of TF–target interactions in single cells.

Experimental design

Gene expression matrix

As input, the SCENIC pipeline needs a gene expression matrix. Because the first step (Step 5) relies on methods based on regression trees, random forests for GENIE3 or gradient-boosted trees for GRNBoost2, no prior scaling of the expression matrix is required, nor does the matrix need to provide the expression in certain units: CPM, FPKM, or raw counts can all be supplied to GENIE3/GRNBoost2.

Study cases

To illustrate this protocol, we applied SCENIC to several cases (Table 1):

1 The original version of SCENIC was developed in R/Bioconductor[3], and the method was evaluated on a data set collected by Zeisel et al.[30] consisting of 3,005 cells of somatosensory mouse cortex and the hippocampus. This protocol focuses on a faster and scalable reimplementation of the SCENIC workflow in Python. Therefore, we used the Zeisel et al. data set as a case study to benchmark and compare the performance of both available versions.

2 PBMC data set provided by 10x Genomics. This data set was chosen because the cell types are well characterized and the data are of high quality, and, as such, it is commonly used as a benchmark data set for other scRNA-seq tools and tutorials.

3 Single-cell transcriptomics has revolutionized cancer research and has elucidated the role of noncancer cells in tumor initiation and progression by providing a cellular perspective on the cellular architecture present in cancer tissue[31]. Pioneered by Tirosh and Suva, scRNA-seq on human cancer biopsies comes with its logistic and technical challenges[32]. Understanding the regulators underlying the aberrant transcriptomes of cancer cells is paramount to unravel cancer biology and to connect these altered phenotypes to causal somatic mutations, chromosomal rearrangements, and/or paracrine cellular interactions with stromal or immunological host cells present in the tumor microenvironment (TME). To demonstrate the ability of SCENIC to discover regulators involved in cancer, we compiled a panel of three data sets. Part of the criteria for the selection of these cases was that these were human cancer biopsies, that a variety of different cell types were represented in the TME and that a sufficient number of cells were present in each cell type (20 or more cells of each type).

Materials

Currently, three versions of the SCENIC pipeline are available. This protocol focuses on the Python-based reimplementation and its containerized versions. For the initial R-based version, we refer the reader to the vignettes available online (https://github.com/aertslab/SCENIC).

In this protocol, we provide two methods to run the SCENIC pipeline: first, with interactive Jupyter notebooks, which are a common way to encapsulate full research procedures and demonstrate their use; and second, the SCENIC pipeline has been implemented in the Nextflow workflow management system (Box 5), which provides a semi-automated way to run the pipeline steps. The Nextflow system is potentially useful for bioinformatics core facilities or as a simplified way to automate SCENIC analyses. These methods are described in https://github.com/aertslab/SCENICprotocol.

Equipment

- A computer or server running MacOS X or a recent Linux distribution for which Python version 3.6 or later is available. If opting for Docker- or Singularity-based deployment, a version of either of these container platforms should be available for your Linux system (e.g., RedHat 7 or later)
- The expression matrix as CSV or loom file capturing the gene expression abundances of the cells assayed in the single-cell transcriptomics experiment
- A list of known TFs for the species of interest supplied in gene symbols or identifiers from the same source used for the expression matrix (e.g., using HGNC (HUGO Gene Nomenclature Committee) identifiers for both the TF list and expression matrix)
- One or more ranking databases. These databases store pre-calculated whole-genome rankings for motifs modeling the DNA sequence-specific binding of a TF
- A motif annotation file for the version of the motif collection used in the ranking databases chosen for your analysis
- (Optional) To visualize the results of the analysis, the SCope tool can be used. This tool is available as a web application (http://scope.aertslab.org), can be deployed on a local server or can be installed as a desktop application on Mac OS X systems

Equipment setup

Software installation

SCENIC requires Python version 3.6 or later and can be installed using the standard Python package installer pip (https://pip.pypa.io/en/stable/installing/), which retrieves the latest version from PyPI, Python's package index (https://pypi.org/). To avoid issues with package dependencies, we recommend the use of an isolated Python environment for running SCENIC. This can be accomplished via miniconda (https://docs.conda.io/en/latest/miniconda.html) or pipenv (https://github.com/pypa/pipenv) combined with virtualenv.

SCENIC containers are also available for download and immediate use (Box 5). In this case, no compiling or installation is required, provided either Docker (https://www.docker.com/) or Singularity[21] software is installed on the user's system. Images are available from both Docker Hub (https://hub.docker.com/) and Singularity Hub (https://singularity-hub.org/).

The SCope visualization tool can be locally installed by following the instructions provided on the GitHub repository website (https://github.com/aertslab/SCope).

Expression matrices

The expression matrices were downloaded from the Gene Expression Omnibus (GEO) (https://www.ncbi.nlm.nih.gov/geo/), ArrayExpress (https://www.ebi.ac.uk/arrayexpress/) and 10x Genomics websites (https://support.10xgenomics.com/single-cell-gene-expression/datasets).

List of TFs

SCENIC requires a list of TFs as a text file where each line represents the gene symbol of a single factor. The list of human and mouse TFs is available as a resource from SCENIC's GitHub repository (https://github.com/aertslab/pySCENIC/tree/master/resources). The list of factors is provided as HGNC symbols for *H. sapiens*, MGI symbols for *M. musculus* and FlyBase symbols for *D. melanogaster*. These factors are derived from the annotations available for the most recent version of the motif collection used by SCENIC. For *H. sapiens*, a restricted list of factors based on a recent curation in Lambert et al.[23] is also provided. These factors are used in the network inference step (Step 5) to restrict the number of regressors used in the tree-based regression per target approach.

Ranking databases

The ranking and recovery framework, on which the cisTarget step (Step 6) of SCENIC depends, needs databases of pre-calculated whole-genome rankings as an auxiliary resource (Box 2). These databases can be downloaded from https://resources.aertslab.org/cistarget/ and are available for *H. sapiens*, *M. musculus*, and *D. melanogaster*.

For each species, multiple databases exist that differ in:

The putative regulatory regions or the genomic search space attributed to each gene, which is scanned for the presence of motifs: 500 bp upstream of the TSS of a gene, 5-kb centered around the TSS or 10-kb centered around TSS. The wider the search space, the higher the recall of target genes of a factor, because targets for which transcriptional initiation is controlled via distant-acting enhancers can also be predicted when using larger regulatory search spaces.

The number of orthologous species taken into account when corroborating the discovered motifs via conservation. The more species taken into account, the more the false discovery rate (FDR) of targets is reduced. The downside of relying on conservation to control specificity is the potential to miss out on species-specific targets. These target genes are controlled via enhancers under diversifying selection, which will have no corresponding match in orthologous species.

The version of the motif collection used for generating the databases. The latest version 9 of the collection contains >24,000 motifs annotated for factors from several species and combines several techniques for determining the sequence-specific binding patterns of TFs (e.g., literature curation, protein-binding microarrays, or TF-targeted ChIP-seq).

Motif annotations

The cisTarget step also needs annotation tables for the ranking databases selected for the analysis. These tables can be downloaded from https://resources.aertslab.org/cistarget/ and associate the motifs with TFs that bind the TFBSs predicted for these motifs. Multiple versions of these annotations exist, and the version matching the one used for generating the database should be used.

(Optional) Case study data sets

The list of data sets used to demonstrate the method and the source from which these were downloaded are listed in (Table 1). The official acronyms for cancer types set out by The Cancer Genome Atlas (TCGA) consortium are used (https://gdc.cancer.gov/resources-tcga-users/tcga-code-tables/tcga-study-abbreviations).

Installation of SCENIC and configuration of environment

▲ **CRITICAL** The following steps need to be run only once for each system install.

1 Set up a Python environment. The following example commands should be run in a Unix shell (i.e., bash or similar) to set up the software. Using a Conda environment, we recommend the usage:

```
conda create -n scenic_protocol python=3.6
conda activate scenic_protocol
```

2 Install necessary and optional Python packages and configure usage with Jupyter(Lab) server.

```
conda install numpy pandas matplotlib seaborn
conda install -c anaconda cytoolz
 # Install scanpy (https://scanpy.readthedocs.io/en/latest/installation.html)
conda install seaborn scikit-learn statsmodels numba pytables
conda install -c conda-forge python-igraph louvain
conda install -c conda-forge multicore-tsne
pip install scanpy
# Install pySCENIC
pip install pyscenic
# Install environment as kernel for Jupyter
pip install --user ipykernel
python -m ipykernel install --user --name=scenic_protocol
```

3 Alternatively, if using a container system, these packages are already fully installed in the images, and you can quickly obtain them via download from a container hub. The following commands illustrate how to obtain the pySCENIC images for Docker and Singularity, respectively, as well as a basic test of the downloaded image:

```
# Docker:
docker pull aertslab/pyscenic:0.9.18
# Singularity (pulls from Docker Hub):
singularity pull --name aertslab-pyscenic-0.9.18.sif \
  docker://aertslab/pyscenic:0.9.18
# we recommend using a recent version of Singularity (v3 or later)
# basic usage (Singularity and Docker), interface to the command-line
version of pySCENIC:
docker run -it aertslab/pyscenic pyscenic:0.9.18 -h
singularity exec aertslab-pyscenic-0.9.18.sif pyscenic -h
```

4 Download auxiliary files: list of TFs, ranking databases and motif annotations (for human, in this example):

```
# transcription factors:

 wget https://raw.githubusercontent.com/aertslab/pySCENIC/master/resources/hs_hgnc_tfs.txt

# motif to TF annotation database:

 wget https://resources.aertslab.org/cistarget/motif2tf/motifs-v9-nr.hgnc-m0.001-o0.0.tbl
# genome ranking database:

 wget  https://resources.aertslab.org/cistarget/databases/homo_sapiens/hg38/refseq_r80/mc9nr/
gene_based/hg38__refseq-r80__10kb_up_and_down_tss.mc9nr.feather
```

Procedure

Pre-processing ● Timing 15 min

1 Download the expression matrix for the case study (PBMC 10k data set from 10x Genomics). In the section below, we fetch the data set from the command line using the wget Linux utility:

```
wget  http://cf.10xgenomics.com/samples/cell-exp/3.0.0/pbmc_10k_v3/pbmc_10k_v3_filtered_fea
ture_bc_matrix.tar.gz

tar xvf pbmc_10k_v3_filtered_feature_bc_matrix.tar.gz
```

2 Perform cleaning and quality control on the downloaded expression data set. We use the Scanpy (https://scanpy.readthedocs.io/en/latest/) package in this example and run the following statements in a Python interpreter session:

```
import scanpy as sc
adata = sc.read_10x_mtx('filtered_feature_bc_matrix/',
var_names='gene_symbols')
adata.var_names_make_unique()
# compute the number of genes per cell (computes 'n_genes' column)
sc.pp.filter_cells(adata, min_genes=0)
# mito and genes/counts cuts
mito_genes = adata.var_names.str.startswith('MT-')
# for each cell compute fraction of counts in mito genes vs. all genes
adata.obs['percent_mito'] = np.ravel(np.sum(
adata[:, mito_genes].X, axis=1)) / np.ravel(np.sum(adata.X,
axis=1))
# add the total counts per cell as observations-annotation to adata
adata.obs['n_counts'] = np.ravel(adata.X.sum(axis=1))
```

Table 3 | Frequently encountered questions

Step	Problem/question	Solution
5	Can I use another network inference tool in the SCENIC pipeline?	This can be accomplished as long as the inference method predicts direct regulatory interactions between a factor and a target gene. The results should be supplied to SCENIC as a list of weighted adjacencies (tabulate as three columns: factor, target gene and importance). For example, GENIE3 can be replaced by PTM (ref. [43]) as demonstrated in Potier et al.[18]. However, the popular module inference method WGCNA[44] detects modules of coexpressed genes and is therefore not compatible with SCENIC
6	Can SCENIC infer target genes for which transcriptional initiation is repressed by a factor?	By default, only regulons of targets transcriptionally activated by a factor are predicted by SCENIC. However, regulons that capture target genes for which the expression is negatively correlated with the expression of its TF can be inferred via the CLI parameter `'--all_modules`. As mentioned by Aibar et al.[3], negatively correlated modules are less numerous, their regulatory interaction weights (from GENIE3/GRNBoost2) are lower and the resulting regulons show overall less motif enrichment. We also assessed the biological relevance of these 'repressor regulons' on the Zeisel et al. data set[30] on the known REST repressor, and the results did not make biological sense (i.e., REST should be active in most cell types of non-neuronal origins—this was not the case). Although SCENIC is capable of identifying repressor regulons, their biological relevance must be interpreted with care
	Can I create my own ranking databases used for pruning indirect targets in the regulon prediction step?	Yes, this can be accomplished. See the FAQ for the code snippet to be used (https://pyscenic.readthedocs.io/en/latest/faq.html)
7	Can I compare the AUCell scores between regulons?	AUCell calculates an 'enrichment score' for the predicted targetome of a factor in the ranked expression profiles of cells. It is therefore important to focus on the distribution of these enrichment scores across cells for a regulon when trying to interpret these scores. Not only does this put the range of scores for the regulon into context, the shape of the distribution is also informative: e.g., a bimodal distribution clearly indicates the presence of two populations of cells in the experiment

3 Carry out the filtering steps with basic thresholds for genes and cells:

```
sc.pp.filter_cells(adata, min_genes=200)
sc.pp.filter_genes(adata, min_cells=3)
adata = adata[adata.obs['n_genes'] < 4000, :]
adata = adata[adata.obs['percent_mito'] < 0.15, :]
```

4 Create a loom file with the filtered data, to be used in downstream analyses:

```
import loompy as lp
row_attrs = {
"Gene": np.array(adata.var_names),
}
col_attrs = {
"CellID": np.array(adata.obs_names),
"nGene": np.array(np.sum(adata.X.transpose()>0, axis=0)).flatten(),
"nUMI": np.array(np.sum(adata.X.transpose(),axis=0)).flatten(),
}
lp.create("PBMC10k_filtered.loom",adata.X.transpose(),row_attrs,
col_attrs)
```

Network inference ● Timing 95 min

5 Run the GRNBoost2 network inference algorithm from the CLI:

```
pyscenic grn \
--num_workers 20 \
--output adj.tsv \
--method grnboost2 \
PBMC10k_filtered.loom \
hs_hgnc_tfs.txt
```

When using Docker or Singularity images, simply prepend the pySCENIC command with the appropriate container run options, including the mounting of the appropriate data directories (here, we mount the current working directory ($PWD), assuming all data and reference files are located within):

```
## Docker
docker run -it -v $PWD:$PWD aertslab/pyscenic:0.9.18 \
pyscenic grn \
{... }
## Singularity
singularity exec -B $PWD:$PWD aertslab-pyscenic-0.9.18.sif \
pyscenic grn \
{... }
```

For additional information and frequently encountered questions related to this step, see Table 3.
? TROUBLESHOOTING

Candidate regulon generation and regulon prediction ● Timing 8 min

6 Execute the following statement from the command line:

```
pyscenic ctx \
adj.tsv \
hg38__refseq-r80__10kb_up_and_down_tss.mc9nr.feather \
--annotations_fname motifs-v9-nr.hgnc-m0.001-o0.0.tbl \
--expression_mtx_fname PBMC10k_filtered.loom \
--mode "dask_multiprocessing" \
--output reg.csv \
--num_workers 20 \
--mask_dropouts
```

For additional information and frequently encountered questions related to this step, see Table 3.

Cellular enrichment ● Timing 1.7 min

7 Invoke the following shell statement to perform the AUCell step:

```
pyscenic aucell \
PBMC10k_filtered.loom \
reg.csv \
--output PBMC10k_SCENIC.loom \
--num_workers 20
```

For additional information and frequently encountered questions related to this step, see Table 3.

(Optional) Regulon activity binarization

The cellular activity pattern of a predicted regulon can be binarized as being in an 'on' or 'off' state based on the bimodal distribution of a regulon's AUCell values. When using the CLI to pySCENIC and using a loom file for input and output, this step occurs during the AUCell command above. Binarization of the AUC matrix can also be performed in an interactive Python session using the 'binarize' function in pySCENIC. This binarization can also be guided by an interactive widget in SCope that allows visual inspection of the AUCell scores distribution per regulon and fine-tuning of the binarization threshold.

Nonlinear projection and clustering ● Timing variable

8 Here, we illustrate the basic steps involved in generating a *t*-SNE and UMAP dimensionality reduction from the regulon AUC matrix. A full working example can be found in the associated Jupyter notebooks.

Execute the following statements in a Python interpreter:

```
import loompy as lp
import umap
from MulticoreTSNE import MulticoreTSNE as TSNE
lf = lp.connect("PBMC10k_SCENIC.loom", mode='r+', validate=False)
auc_mtx = pd.DataFrame(lf.ca.RegulonsAUC, index=lf.ca.CellID)
lf.close()
# UMAP
runUmap = umap.UMAP(n_neighbors=10, min_dist=0.4,
metric='correlation').fit_transform
dr_umap = runUmap(auc_mtx)
# tSNE
tsne = TSNE(n_jobs=20)
dr_tsne = tsne.fit_transform(auc_mtx)
```

Visualization in SCope ● Timing variable

9 Collection of the full set of analysis steps into a SCope-ready loom file is an extensive procedure that has been well documented in the associated Jupyter notebooks. Visit our accompanying website (https://github.com/aertslab/SCENICprotocol) for further instructions. The overall strategy is to collect the various results that have been generated over the course of this protocol. This includes dimensionality reductions (from both Scanpy and pySCENIC) that are to be embedded as well as any sample annotations, along with the pySCENIC regulon information. These results are then placed into the appropriate attributes of the loom file. This process can be easily adapted to include other sample annotations not present in our example data sets (e.g., timepoint, sample, or genotype information).

The resulting loom file can be uploaded into the SCope tool (http://scope.aertslab.org). The full features of SCope are demonstrated in a tutorial (http://scope.aertslab.org/#/Protocol_Cases/*/tutorial) as well as in the 'Anticipated results' section.

Troubleshooting

Troubleshooting advice can be found in Table 4. This table summarizes the issues reported by users via GitHub's tracking system (https://github.com/aertslab/pySCENIC/issues).

Table 4 | Troubleshooting table

Step	Problem	Possible reason	Solution
5	Executing the network inference step of SCENIC results in the following error message: `'ValueError: Intersection of gene_names and tf_names is empty.'`	The expression matrix needs to be provided (cell × gene identifiers), i.e., cell codes as rows and gene identifiers as columns	Either resort to the loom file format or transpose the expression matrix before supplying it the SCENIC. The latter can be achieved by using the keyword argument `'-t'` or `'--transpose'` in the CLI command or using the transpose method of the pandas DataFrame data structure (https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.DataFrame.transpose.html)
	Executing the network inference step of SCENIC results in the following error message: `'ValueError: Intersection of gene_names and tf_names is empty.'`	The list of TFs is supplied in a different nomenclature than the symbols/identifiers/names used to specify the genes in the expression matrix	Convert the list of TFs to the appropriate nomenclature using a web application such as ENSEMBL BioMart (https://m.ensembl.org/biomart/martview)
	When running the network inference step, the SCENIC process is unable to spawn workers and terminates prematurely	This step of the pipeline has a large memory requirement, and the memory consumption depends on the number of worker processes being created. If SCENIC is used on a system setup without swap space, processes allocating memory exceeding the current limits are killed by the operating system	Reduce the number of workers assigned to the network inference task. This can be accomplished via the CLI parameter `'--num_workers NUM_WORKERS'`

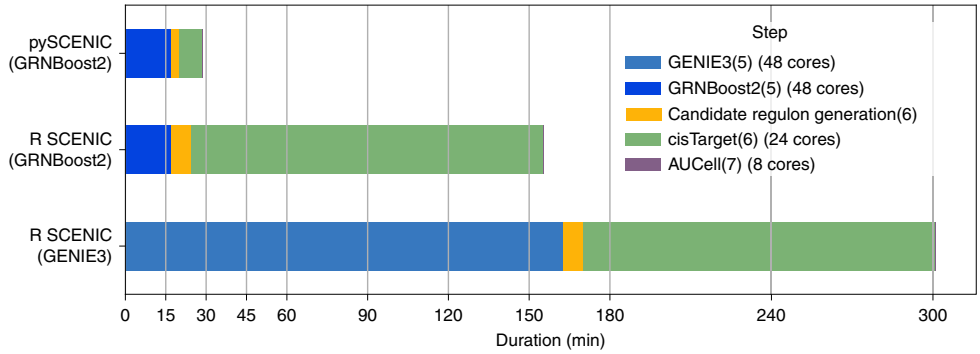


Fig. 2 | Speed comparison of complete SCENIC workflow. Comparison between implementations of wall time for running all steps of the SCENIC pipeline on a 3,005-mouse-brain-cell data set by Zeisel et al.[30]. The step numbers are shown in square brackets next to the method labels in the legend.

Timing

A typical run of a SCENIC pipeline, starting with the first stage of network inference (Step 5) and including the final stage of scoring the individual cells for the activity of the individual cells (i.e., the AUCell stage or Step 7), lasts a few hours. The execution times for the selected case studies is depicted in Table 2. For all these cases, three ranking databases for the motif search spaces were used (500 bp upstream of TSS, TSS ± 5 kb, and TSS ± 10 kb), each containing models for 24,453 motifs (version 9 of the motif collection). In contrast, the best practices steps we outline in Fig. 1, which are performed in parallel, take only a fraction of this time, typically <10 min on an HPC system.

We assessed the improvement in execution time for the reimplementation of the SCENIC workflow on the mouse somatosensory cortex and hippocampal regions data set from Zeisel et al.[30], and used the same parameters and databases as used in the original SCENIC publication[3]: two ranking databases (500 bp upstream of TSS and TSS ± 10 kb) of each 20,003 motif (version 8 of the motif collection). The wall time (i.e., the total elapsed real-world time) for running this experiment is ~28 min in the newly implemented Python version on a dual Intel Xeon E5-2680 v3 machine with 512 Gb of RAM. This is a tenfold reduction from the 5 h in the R version (Fig. 2). This wall-time reduction is due to the increase in computational efficiency of the GRNBoost2 algorithm compared to GENIE3 and the Python version of the cisTarget step compared to the R implementation. A further reduction in wall time is achievable by increased parallelization, leveraging multicore, and multinode HPC infrastructure.

Anticipated results

The primary results of this SCENIC pipeline are contained in a loom file that can be visualized and explored via SCope as well as programmatically in R or Python. In addition, custom analysis and publication-ready figures can be generated in a Jupyter notebook via integration with Scanpy. The results for all of the study cases presented here are viewable on SCope via the following link: http://scope.aertslab.org/#/Protocol_Cases/*/welcome. Jupyter notebooks showing the specific and complete steps taken to analyze these study cases are available from GitHub (https://github.com/aertslab/SCENICprotocol/tree/master/notebooks). Here, we cover the details of the PBMC and cancer data sets.

Study case 1: PBMC data

The PBMC data are provided by 10x Genomics as a technological demonstration of the data produced by their Chromium platform. This provides a high-quality public data set to test with this protocol. We first downloaded the data from the 10x Genomics website and used the filtered counts matrix ('Feature / cell matrix (filtered)') for further steps.

Filtering the raw count data

After loading the Cell Ranger–filtered data into Python using Scanpy's built-in functions to read 10x MTX files (`sc.read_mtx`), the first step is to apply basic filtering to the data. On a cell level, we focus on the percentage of mitochondrial reads, the number of expressed genes, and the total counts per cell. It is useful to look at the distribution of these features before applying any filters (Fig. 3).

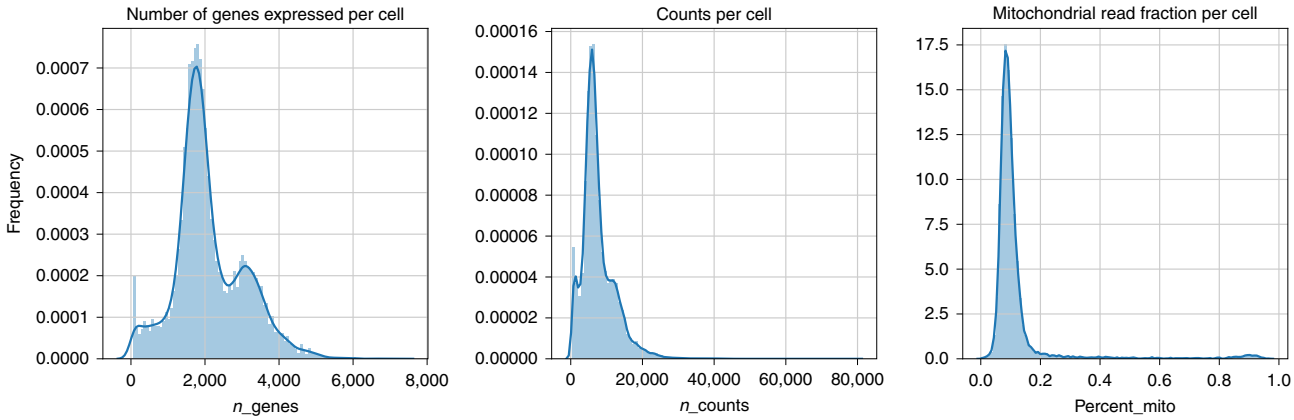


Fig. 3 | Summary statistics for the unfiltered counts matrix for the PBMC study case. From left to right, the panels show the number of genes expressed per cell (*n*_genes), the total number of counts per cell (*n*_counts), and the fraction of mitochondrial reads expressed per cell (percent_mito). The distribution of each variable is shown as a relative frequency histogram. The number of bins is set to 100 in all plots. A kernel density estimate is shown in a solid blue line.

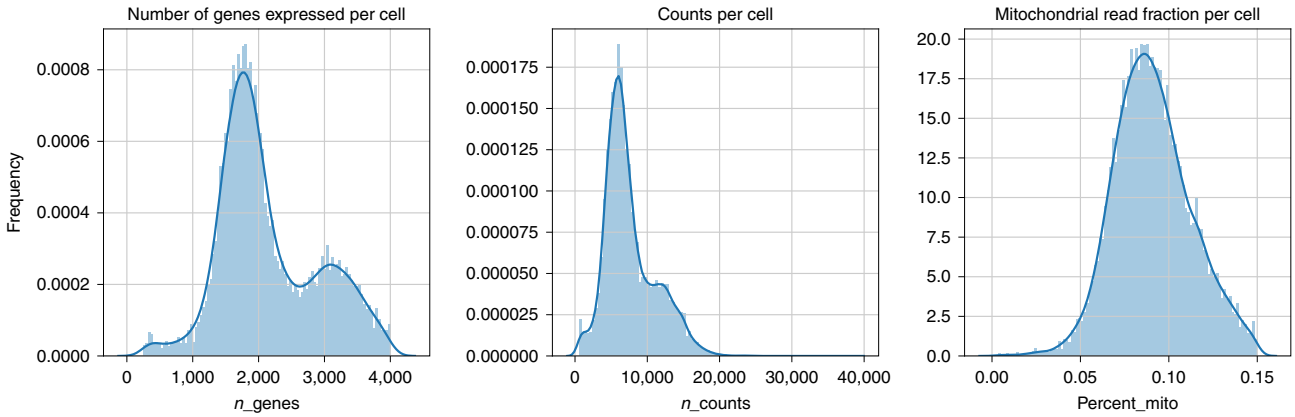


Fig. 4 | Summary statistics for the counts matrix after filtering for the PBMC study case. From left to right, the panels show the number of genes expressed per cell (*n*_genes), the total number of counts per cell (*n*_counts), and the fraction of mitochondrial reads expressed per cell (percent_mito). The distribution of each variable is shown as a relative frequency histogram. The number of bins is set to 100 in all plots. A kernel density estimate is shown in a solid blue line.

The Cell Ranger filtering on the number of genes expressed per cell has already removed many cells with zero gene expression. Based on the histogram in Fig. 3, we apply a further threshold to include only cells with >200 and <4,000 genes expressed. In addition, there are a number of cells with a high fraction of mitochondrial counts, and we apply a threshold to exclude cells above 15%. On a gene level, we filter out genes that are expressed in fewer than three cells in the data set. The diagnostic plot on the filtered data reflects the filtering performed (Fig. 4). The filtered expression matrix is then saved to disk using the loom file format.

Network inference (GRNBoost2)

The first step in the SCENIC pipeline is to perform GRN inference. The primary algorithm used for this is GRNBoost2. There are no major parameters to adjust in this step, except to provide the appropriate set of TFs, which is used to limit the number of regression steps that must be performed. Typically, the full list of known TFs for the appropriate organism is chosen (for human, this is a set of 1,797 TFs). The output of this step is a network graph representing TF–gene interactions along with the interaction weight.

Candidate regulon generation and regulon prediction using motifs (cisTarget)

Using the set of TF–gene interactions identified by GRNBoost, the module generation step creates unrefined modules that consist of the TF and all of the genes that it regulates, along with their interaction weights. For the PBMC data set, there are a total of 6,858 modules identified.

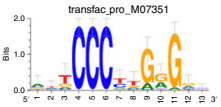
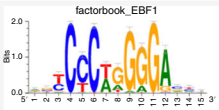
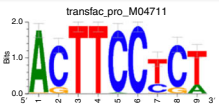
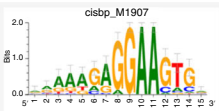
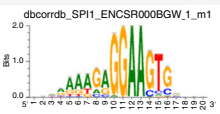
TF	MotifID	AUC	Annotation	Context	Motif similarity *Q*-value	NES	Rank at max	Target genes	Motif logo
EBF1	transfac_pro_M07351	0.1540	Gene is directly annotated	(hg38_refseq-r80_500bp_up_and_100bp_down_tss.mc9nr, top50, activating)	0.000000	6.271401	612	((FAM129C, 10.2553), (CD79A, 10.8223), (PAX5, 12.3934))	
	factorbook_EBFT	0.1381	Gene is directly annotated	(hg38_refseq-r80_500bp_up_and_100bp_down_tss.mc9nr, top50, activating)	0.000000	5.476125	756	((CD79A, 10.2553), (FAM129C, 10.8223), (P2RY10, 12.3934))	
PAX5	transfac_pro_M04711	0.1426	Gene is annotated for similar motif dbcorrdb_PAX5_ENCSR000BHJ_1_m2 ('PAX5 (ENCSR000BHJ-1, motif 2)'; *q*-value = 0.000317)	(hg38_refseq-r80_500bp_up_and_100bp_down_tss.mc9nr, top50, activating)	0.000317	5.273931	292	((CNR2, 25.2802), (KIAA0040, 27.7354), (STAP1, 32.354))	
	cisbp_M1907	0.1417	Gene is annotated for similar motif dbcorrdb_PAX5_ENCSR000BHJ_1_m2 ('PAX5 (ENCSR000BHJ-1, motif 2)'; *q*-value = 0.000547)	(hg38_refseq-r80_500bp_up_and_100bp_down_tss.mc9nr, top50, activating)	0.000547	5.230762	2,490	((CNR2, 23.633), (E2F5, 24.7007), (CD37, 25.2802))	
	dbcorrdb_SPI1_ENCSR000BGW_1_m1	0.1403	Gene is annotated for similar motif dbcorrdb_PAX5_ENCSR000BHJ_1_m2 ('PAX5 (ENCSR000BHJ-1, motif 2)'; *q*-value = 0.000405)	(hg38_refseq-r80_500bp_up_and_100bp_down_tss.mc9nr, top50, activating)	0.000405	5.161844	1,596	((ARHGAP24, 24.7007), (CD22, 25.2802), (BIRC3, 27.7354))	

Fig. 5 | Table of enriched motifs from cisTarget for a selected set of regulons related to B cells, generated within the SCENIC workflow. For each TF identified, the specific motif is shown, along with the motif logo. Additional information, including scores from the ranking and recovery procedure (AUC and NES), annotations, and an abbreviated list of the target genes with their network importance scores, is also shown in the table.

Tunable parameters here include the thresholds for the three methods used to generate the modules from the network graph. These include, first, (`"--top_n_targets"`), which selects 50 (by default) targets for each TF. Second, selection by the top regulators (`"--top_n_regulators"`), which selects multiple sets of regulators for each target (5, 10, and 50 by default). Third, selection based on a percentile score, (`"--thresholds"`), which takes multiple sets of targets for a TF, taking the top 75th and 90th percentiles by default. Finally, there is a lower bound on the number of target genes allowed in a module; the default value is to keep only modules with 20 or more genes.

After module creation, modules are then refined using the cisTarget framework. Parameters in this step include the feature AUC threshold (`"--auc_threshold"`; default of 0.05), the rank threshold to derive target genes (`"--rank_threshold"`; default of 5,000) and the normalized enrichment score (NES) threshold for selection of features (`"--nes_threshold"`; default of 3.0). We use the default values here for the PBMC data set for all parameters. The main result here is a set of enriched motifs for the modules (Fig. 5).

Cellular enrichment and binarization (AUCell)

After the creation of the refined regulons, cellular enrichment can be assayed with AUCell. Here, we quantify the activity of the regulons in each cell. For each regulon, the distribution is then binarized using a Gaussian mixture model. An example of three regulons is shown in Fig. 6, with the binarization thresholds shown as a red line.

Dimensionality reductions and visualization using SCope

Instead of the typical method using the top PCA components computed directly from expression counts as an input for the *t*-SNE or UMAP, the matrix of regulon AUC values for each cell can also be supplied to these nonlinear projection methods. Figure 7 demonstrates the results of both approaches, as applied to the PBMC data set. Here, the UMAP based on highly variable genes (HVGs) reflects cellular identity based on expression, and the different cell types form distinct clusters. In contrast, the SCENIC UMAP does not appear to show as much clustering; however, it is important to note that this reflects a different aspect of the underlying biology. The SCENIC UMAP is a transformation of the regulon AUC values, meaning that cells with similar regulons will cluster together. This can be

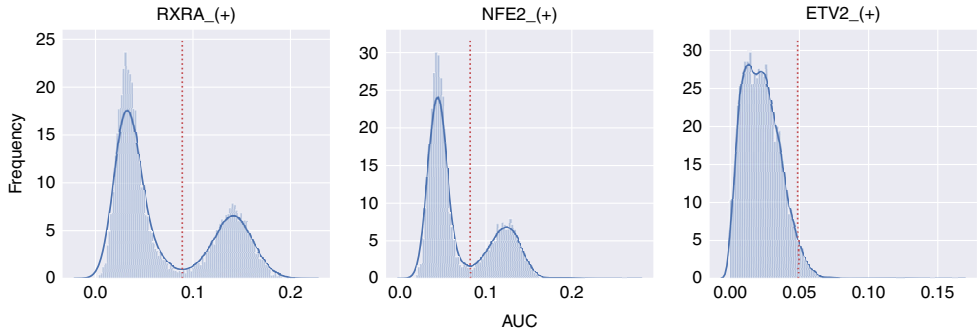


Fig. 6 | AUC distribution across cells for three sample PBMC regulons. The RXRA and NFE2 regulons demonstrate a clear bimodal distribution, reflecting biological differences between cell types, whereas the distribution of ETV2 across cells is more unimodal. The number of bins is set to 100 in all panels. The calculated binary threshold is shown as a red dotted line.

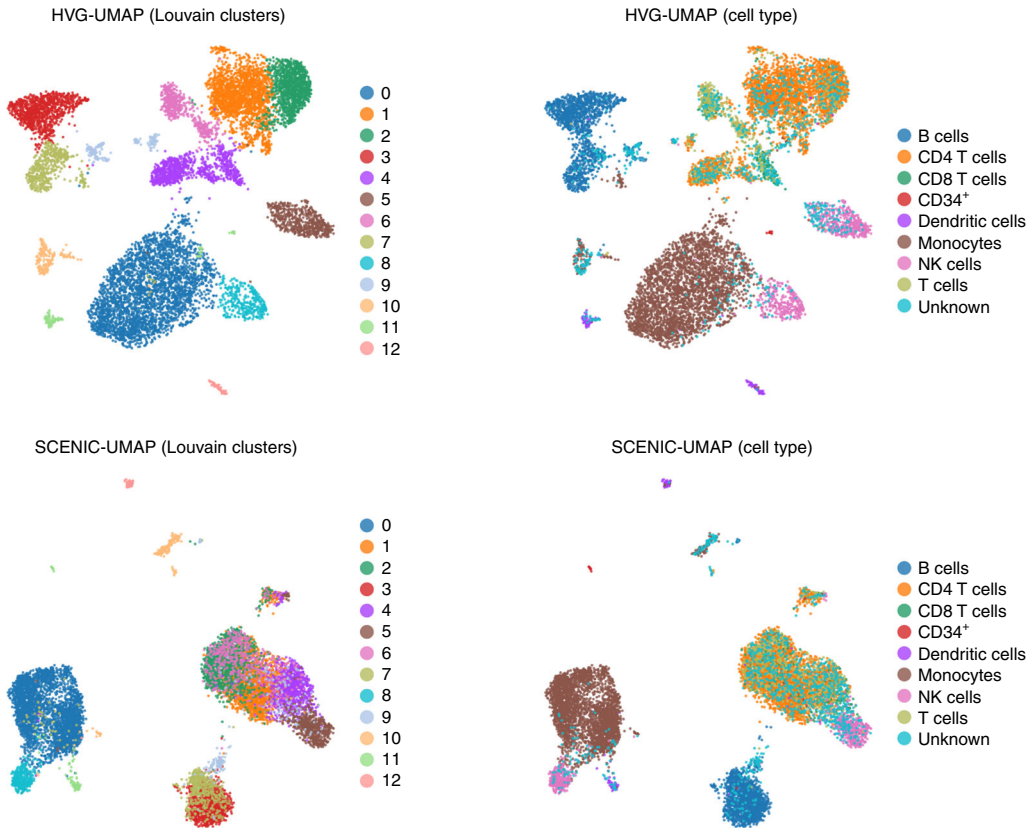


Fig. 7 | Dimensionality reduction plots for the PBMC study case. A UMAP projection using HVGs is shown in the top row, highlighting the Louvain clusters (left) and differences in cell types driven by individual gene expression (right). A UMAP generated from the SCENIC AUC matrix is shown in the bottom row and demonstrates how cells cluster together on the basis of shared regulon activity. The Louvain cluster colors highlight identical cells in each plot and are based on the HVG data representation. The cell types here are automatically classified using the Garnett tool (https://cole-trapnell-lab.github.io/garnett/).

seen in the T cell population, for example. Although the HVG-based UMAP shows clear clusters differentiating innate (natural killer (NK)) and adaptive T lymphoid cells ($CD4^+$ and $CD8^+$), in the SCENIC UMAP these are merged into one cluster, suggesting many shared regulons, although the long axis of the cluster still separates between innate versus adaptive lymphoid cells. Because HVG clustering is often more sensitive to detect cell types, due to a small number of highly expressed markers, we recommend running both analyses side by side and evaluating the regulon activity on the HVG-based UMAP as illustrated further below in Fig. 8.

Fig. 8 | The SCope tool enables interactive comparison of multiple visualizations for the PBMC study case. In this screenshot, the left pane shows the HVGs UMAP, and the right shows the SCENIC AUC UMAP. Three T cell-related regulons are highlighted: TCF7 (red), E2F2 (green), and EOMES (blue). The color intensity is based on the regulon AUC values. Here, each regulon forms essentially distinct clusters within the T cell population in the HVG pane (left), whereas these are grouped together in a larger regulatory cluster in the SCENIC analysis (right).

This concept can be further explored using the SCope visualization tool (http://scope.aertslab.org). Here, we use the final integrated loom file from this protocol, which contains multiple dimensionality reduction plots, along with cell type and clustering annotations.

We can interactively explore the T cell clustering example shown in Fig. 7 in SCope by using the 'Compare' tab and selecting the two (or more) plots of interest. With this, we can visualize the activity of regulons that are identified as generally active across T cells.

The merging of the T cell clusters in the SCENIC analysis is reflected by looking at the activity of multiple regulons: TCF7, which is active across CD4 and CD8 T cells[33], and E2F2, which regulates quiescence in T lymphocytes[34]. We also include eomesodermin (EOMES), a TF that is active in a subset of NK cells and CD8 T cells[35]. The AUC values for these three regulons are shown on the comparison plot (Fig. 8). On the HVG UMAP, we see that the TCF7 and E2F2 regulons show high activity in their specific cell types as well as more broad activity across the larger group of T cells, whereas the EOMES regulon is active in some CD8 T cells and NK cells, matching reports from the literature. On the SCENIC UMAP, these cell types are merged, reflecting the broader targets of these TFs, while showing a progression of the T cell subtypes across the long axis of the cluster. In the SCENIC results, the subset of NK cells active in EOMES has merged on the T cell cluster near the CD8 T cells, reflecting their shared activity.

These two visualizations therefore provide complementary ways of analyzing a single-cell data set and can be easily explored using a combination of these visualization methods.

A further exploration of specific regulons is possible using SCope. In B cells, a network of TFs regulates cell development and fate[36], and we are able to recover several of these regulons from a combination of the motif- and track-based analyses in the cisTarget step of SCENIC. Several TFs, including EBF1, E2A (encoded by TCF3), PU.1 (encoded by SPI1), and FOXO1, in conjunction with other factors, including BCL11A, act together to activate other regulators, such as PAX5 (ref. [37]).

For both the EBF1 and TCF3/E2A regulons, there is clear specificity for the B cell cluster (Fig. 9). Both regulons show high cellular activity in B cells, as measured by AUC. Furthermore, it is clear that the B cells specifically are 'on' based on the binary thresholds in the two regulons. When looking at the expression of the two TFs, EBF1 and TCF3 expression is present mainly in the B cell cluster, whereas TCF3 is also sparsely expressed at lower levels outside the B cells.

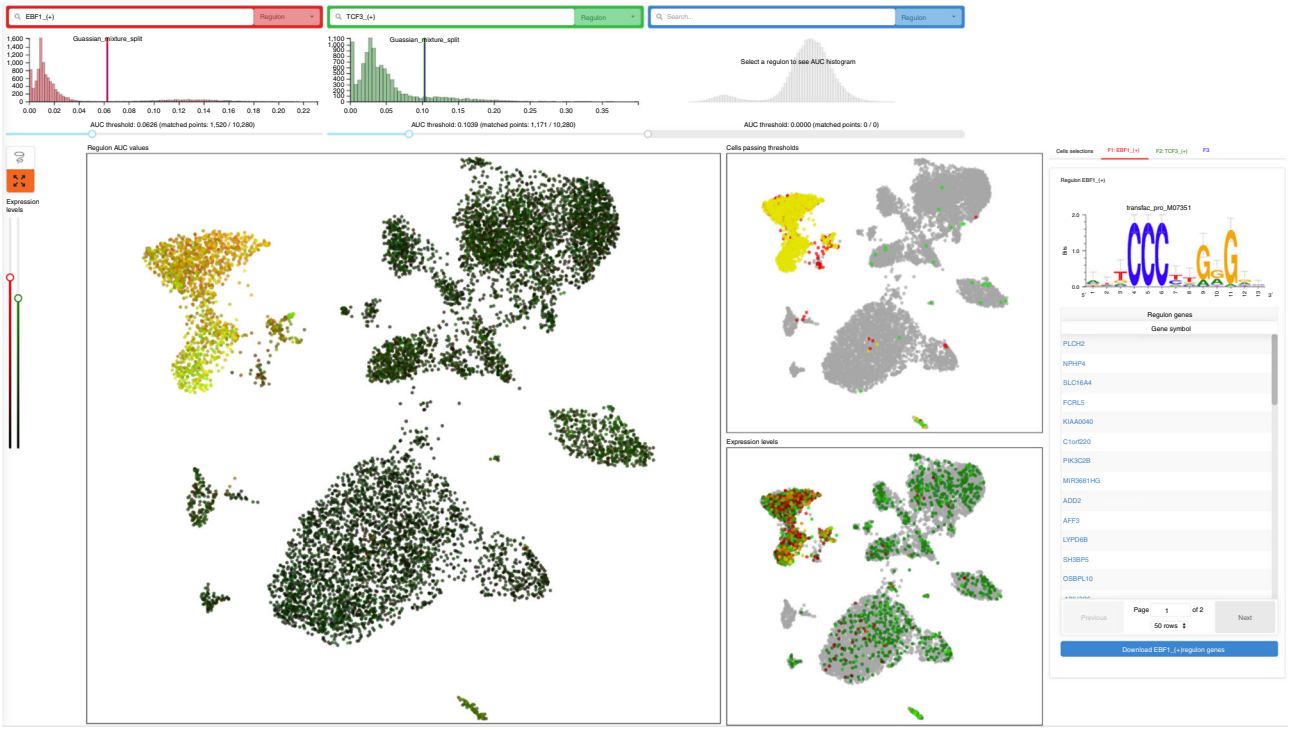


Fig. 9 | The SCope tool allows exploration of regulons. This illustrates the capabilities of the regulon tab in SCope for the PBMC study case. Here, two regulons with involvement in B cells are highlighted: EBF1 (red) and TCF3, which encodes E2A (green). Under each gene selection box is the distribution of the regulon AUC values for each cell, with the automatic (gray vertical lines) and interactively set binarization thresholds shown in the color of their respective box. The HVG UMAP is shown in all three plot windows, the color intensity representing different aspects of the data in each plot: regulon AUC values per cell (left plot), the cells that pass the binary threshold set in the AUC histogram (top right), and the expression values of the TF (bottom right). The motif logo for the EBF1 regulon, along with a list of target genes, is shown in the right panel.

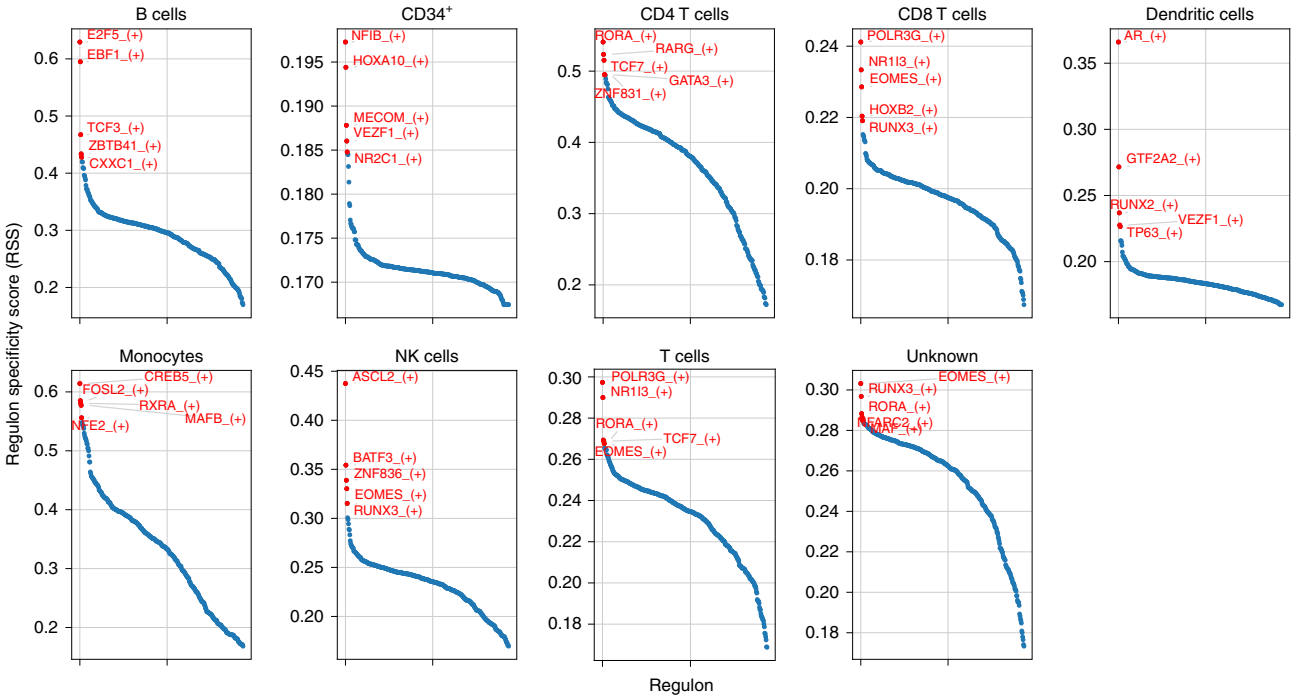


Fig. 10 | Regulon specificity score for each PBMC subtype. The top five regulons in each cell type are highlighted in red and labeled on the plot. The specificity score is shown on the *y* axis.

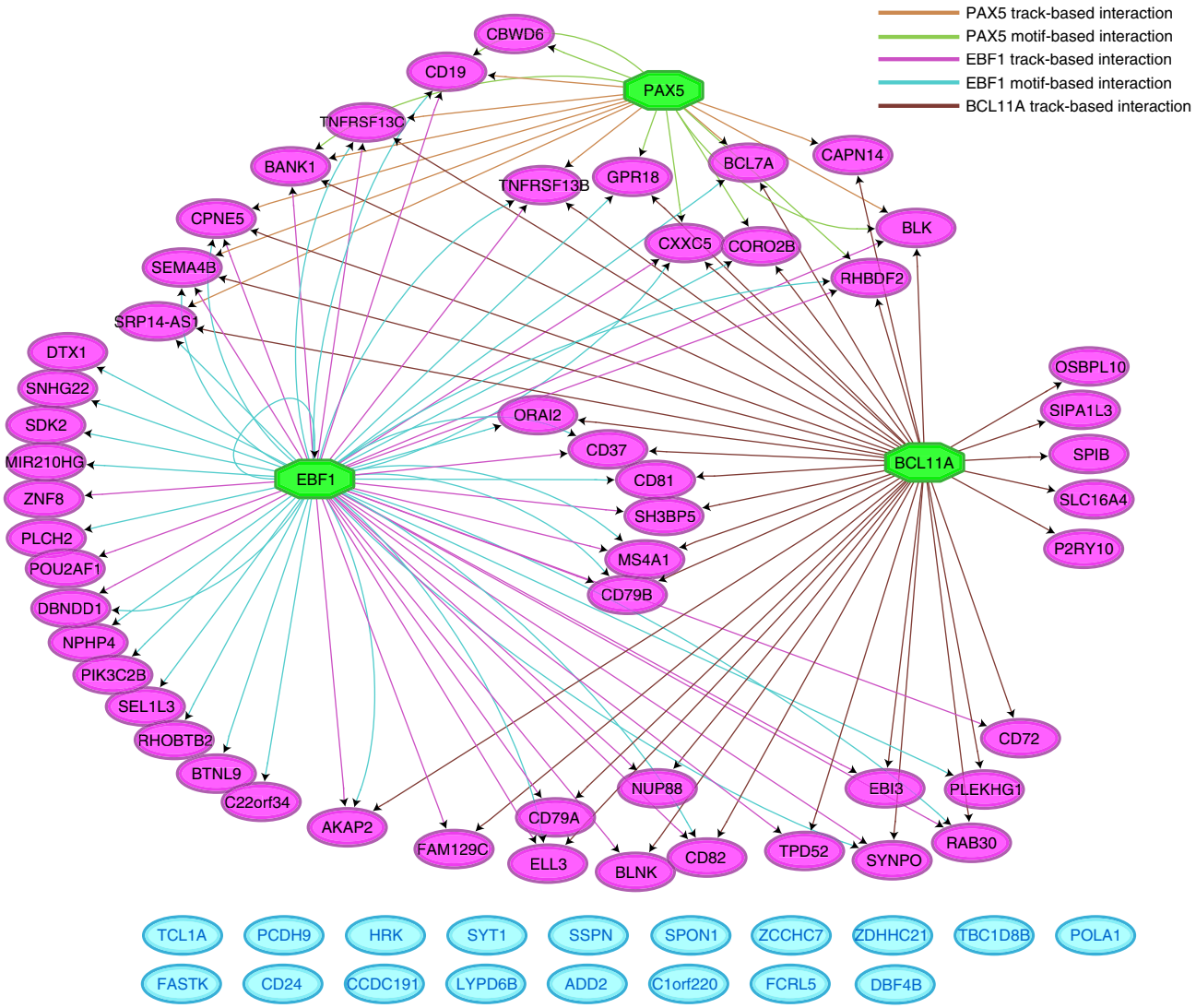


Fig. 11 | Extended analysis of the EBF1 regulon performed in iRegulon. This screenshot shows a network generated with iRegulon using the EBF1 target genes identified by SCENIC as an input. Using a combination of motif- and epigenomic track-based annotations, iRegulon is able to identify PAX5 and BCL11A (green nodes), which are additional interactors with EBF1 that are not present in the EBF1 regulon generated by SCENIC. The edges, representing the connections between each of the three TFs and their target genes, and shown as a line with an arrow directed to the target gene, are color coded based on the source of interaction data (either motif based or track based). The EBF1 regulator contains both motif- (blue lines) and track-based (magenta lines) interactions. Likewise, PAX5 contains a mix of motif- (green lines) and track-based (orange lines) edges. However, BCL11A was found solely in the track-based databases, and all edges are shown in brown. Input target genes found in the SCENIC EBF1 regulon, which are not identified as targets in this iRegulon analysis, are shown in cyan along the bottom of the figure.

Use of epigenomic track databases

We can further investigate regulons in detail by using a combination of motif- and track-based databases. This is achieved by using the track databases, in a separate run of the cisTarget step of the SCENIC pipeline, to generate sets of both motif- and track-based regulons. In the PBMC data, we

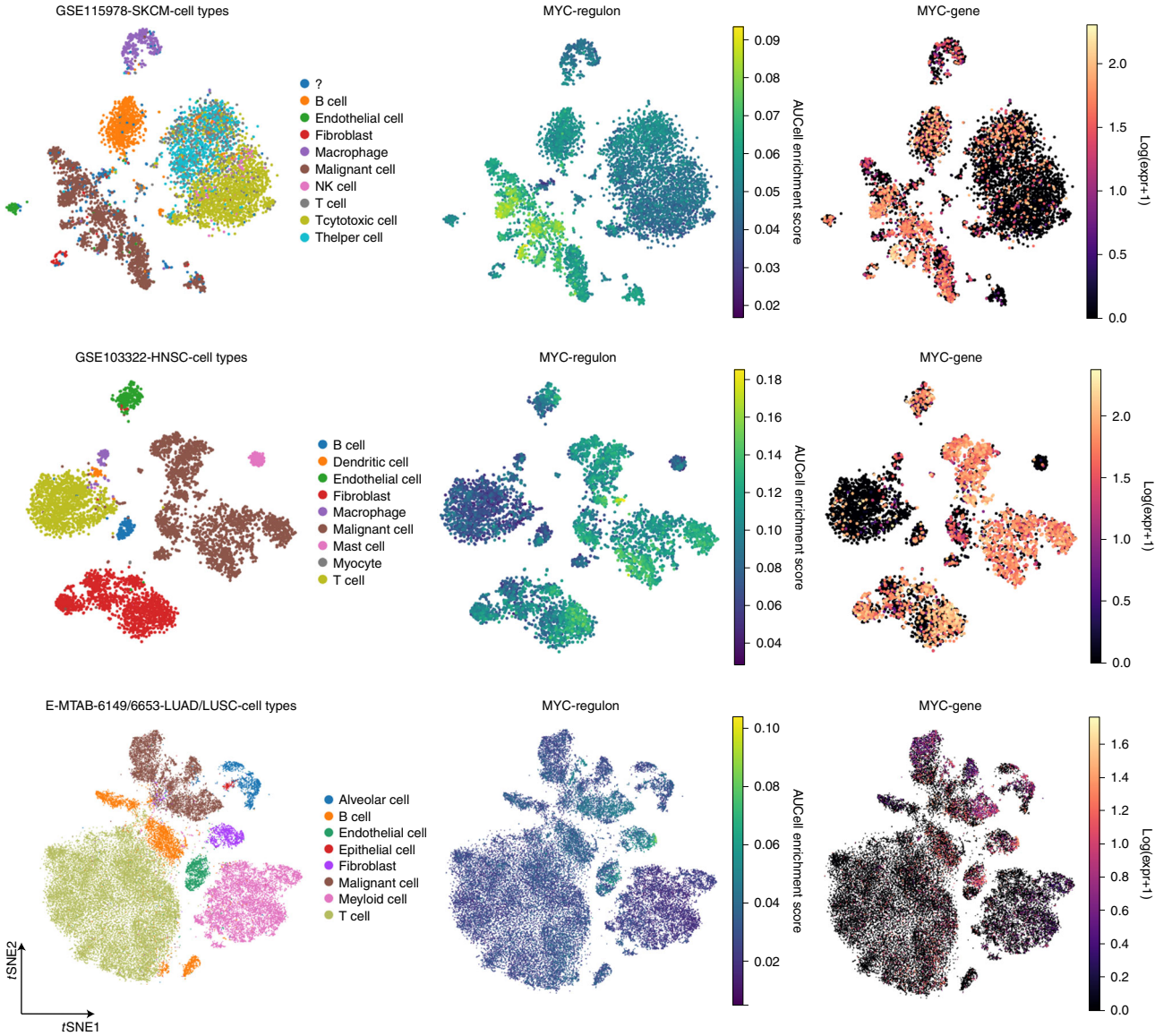


Fig. 12 | Overview of cancer single cell transcriptomics experiments. *t*-SNE nonlinear projections for the three scRNA-seq experiments on human cancer biopsies HNSC, skin cutaneous melanoma (SKCM), and lung adenocarcinoma/lung squamous cell carcinoma (LUAD/LUSC). Cellular scatter plots are generated using AUCell as a dimensional reduction step followed by *t*-SNE. Labels are provided by the authors of the respective publications. These plots clearly demonstrate that cancer cells cluster according to patient, whereas host cells constituting the TME are phenotypically similar and therefore form clusters spanning cells from different patients. For each cancer type, the expression of *MYC* and the AUCell scores for the predicted MYC regulon is given.

find a total of 375 motif-based and 41 track-based regulons, with 34 regulons overlapping. Although there are many fewer regulons found using the track-based databases, these are important to help lend support to the motif-based regulons, which are inherently limited to the set of motifs in the database. Using the motif database alone, we are able to find all of these major B cell factors described above. Of these, we are able to confirm the activity of both EBF1 and BCL11A in the B cell cluster using the independent track database.

Further analyses of SCENIC regulons are described below.

Regulon specificity score

In a typical expression-based analysis, a common goal is to identify marker genes that differentiate between the clusters. Here, we are able to perform a similar procedure that identifies regulons that are specific to clusters using the regulon specificity score (RSS)[38] (Box 4). The RSS is calculated for each

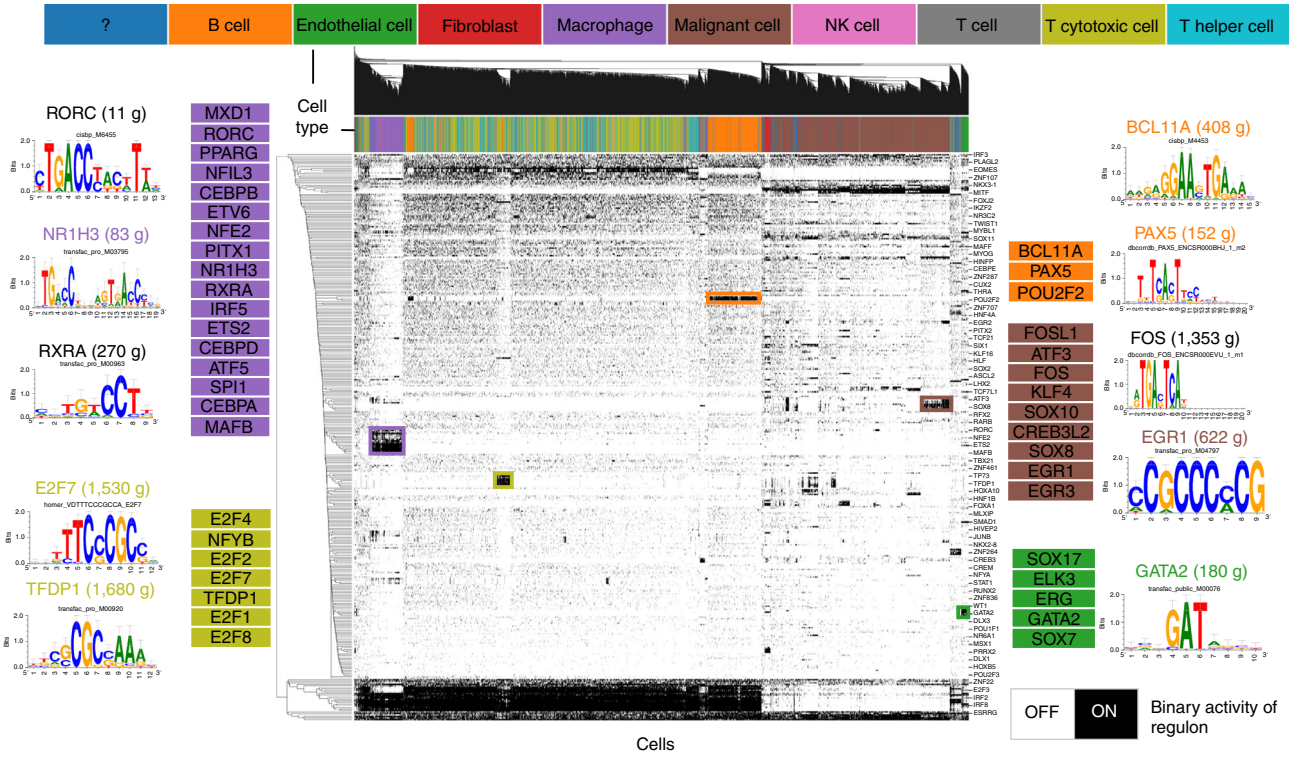


Fig. 13 | Binary heat map for the skin cutaneous melanoma (SKCM) data set. Binary activity for each cell is generated from the SCENIC AUC distribution and plotted as a heat map, with black blocks representing cells that are 'on'. For selected cell types, the figure depicts regulators and their associated motif. The number of predicted target genes is also given for each regulator. For example, the regulon based on the heterodimer motif E2F/TFDP1 is active in a small subset of cytotoxic T cells indicative of clonal expansion (yellow-green). The regulators BCL11A and PAX5 are defining the identity of the B cell compartment (orange). A collection of nuclear receptors (RORC, NR1H3, and RXRA) is active in macrophages residing in the TME (purple). FOS and EGR1 show high activity in a subcluster of malignant cells (brown). The cluster of endothelial cells (green) contain a set of regulators that include the well-known GATA2.

cell type separately, and the top five regulons are shown in Fig. 10. This is also possible to achieve for arbitrary clusters (i.e., Louvain clusters).

Network fine-tuning in Cytoscape

As described in the Limitations section, network inference is a stochastic method and, as a result, produces regulons with target genes that can change across multiple runs. In its default configuration, the module generation step produces six separate candidate modules from the GRN, which are subsequently pruned into a finalized regulon. For a particular regulon of interest, the unpruned modules, or the final regulon, can be explored using an external tool, iRegulon[13], which is available as a plugin for Cytoscape[39].

An online tutorial (http://iregulon.aertslab.org/tutorial.html) for iRegulon provides a guide on how to import a module and proceed through a detailed analysis and exploration of the gene networks within the module. iRegulon integrates multiple pieces of information, including the same TF and motif-, and track-based annotations as used in SCENIC. In iRegulon, a more detailed exploration of the regulatory network inferred for a module or regulon is possible, along with the ability to build sub-networks of key genes. It is also possible to investigate multiple candidate motifs in iRegulon, whereas SCENIC by default reports only the top selected. Thus, although SCENIC provides the ability to uncover regulons at a transcriptome level and provide clues for further investigation, a follow-up analysis with iRegulon has the potential to uncover more detailed regulatory interactions between genes of particular interest. An example of such an analysis is shown in Fig. 11.

Study case 2: scRNA-seq cancer compendium

For the second study case, we provide a documented Jupyter notebook describing the commands and analysis results of the SCENIC pipeline applied to three scRNA-seq data sets derived from different

cancers (Fig. 12). A similar protocol to the one outlined for the PBMC study case was followed closely here, with the addition of features specific to each cancer case (e.g., annotations for cell types or other metadata integration).

As described previously, these three data sets interrogate the cancer cell compartment and the TME from human biopsies taken from melanoma, head and neck squamous cell carcinoma (HNSC), and non-small cell lung carcinoma. The analysis provides insight into the regulators that define the identity of cell types constituting the TME (see companion Jupyter notebooks for RSS plots). The results can be visualized by plotting the AUCell enrichment scores of the predicted regulons on the cellular scatterplots (Fig. 12). Moreover, binarization of the AUCell scores for the predicted regulators and subsequent clustering of the cell–regulon matrix (Fig. 13) reveals clusters of regulators characteristic of cell types.

Reporting Summary

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

Data availability

All data analyzed within this protocol are publicly available. The PBMC 10k data set is directly available for download from the 10x Genomics company website: https://support.10xgenomics.com/single-cell-gene-expression/datasets/3.0.0/pbmc_10k_v3. The following data sets are available from the National Center for Biotechnology Information's GEO and are accessible through GEO Series accession numbers: GSE60361 (mouse brain data set), GSE115978 (human cutaneous melanoma), and GSE103322 (human HNSC). The non-small cell lung carcinoma data set can be downloaded from ArrayExpress (experiments E-MTAB-6149 and E-MTAB-6653). Additional metadata are available as the supplementary information files from the original publications that generated these data sets. The online version of the case studies used in this protocol is available on GitHub (https://github.com/aertslab/SCENICprotocol), including Jupyter notebooks, and the Nextflow project code, along with associated installation and usage instructions.

Code availability

SCENIC is available as a Python package at https://pypi.org/project/pyscenic/, and its source code is available on GitHub (https://github.com/aertslab/pySCENIC). The code in this manuscript has been peer reviewed.

References

1. Satija, R., Farrell, J. A., Gennert, D., Schier, A. F. & Regev, A. Spatial reconstruction of single-cell gene expression data. *Nat. Biotechnol.* **33**, 495–502 (2015).
2. Wolf, A. F., Angerer, P. & Theis, F. J. SCANPY: large-scale single-cell gene expression data analysis. *Genome Biol.* **19**, 15 (2018).
3. Aibar, S. et al. SCENIC: single-cell regulatory network inference and clustering. *Nat. Methods* **14**, 1083–1086 (2017).
4. Tommaso, P. et al. Nextflow enables reproducible computational workflows. *Nat. Biotechnol.* **35**, 316–319 (2017).
5. Dobin, A. et al. STAR: ultrafast universal RNA-seq aligner. *Bioinformatics* **29**, 15–21 (2013).
6. Bray, N. L., Pimentel, H., Melsted, P. & Pachter, L. Near-optimal probabilistic RNA-seq quantification. *Nat. Biotechnol.* **34**, 525–527 (2016).
7. Parekh, S., Ziegenhain, C., Vieth, B., Enard, W. & Hellmann, I. zUMIs—a fast and flexible pipeline to process RNA sequencing data with UMIs. *Gigascience* **7**, 1–9 (2018).
8. Srivastava, A., Malik, L., Smith, T., Sudbery, I. & Patro, R. Alevin efficiently estimates accurate gene abundances from dscRNA-seq data. *Genome Biol.* **20**, 65 (2019).
9. Ilicic, T. et al. Classification of low quality cells from single-cell RNA-seq data. *Genome Biol.* **17**, 29 (2016).
10. Huynh-Thu, V., Irrthum, A., Wehenkel, L. & Geurts, P. Inferring regulatory networks from expression data using tree-based methods. *PLoS ONE* **5**, e12776 (2010).
11. Moerman, T. et al. GRNBoost2 and Arboreto: efficient and scalable inference of gene regulatory networks. *Bioinformatics* **35**, 2159–2161 (2018).
12. Gaiteri, C., Ding, Y., French, B., Tseng, G. & Sibille, E. Beyond modules and hubs: the potential of gene coexpression networks for investigating molecular mechanisms of complex brain disorders. *Genes Brain Behav.* **13**, 13–24 (2014).
13. Janky, R. et al. iRegulon: from a gene list to a gene regulatory network using large motif and track collections. *PLoS Comput. Biol.* **10**, e1003731 (2014).

14. Herrmann, C., de Sande, B., Potier, D. & Aerts, S. i-cisTarget: an integrative genomics method for the prediction of regulatory features and cis-regulatory modules. *Nucleic Acids Res.* **40**, 1–44 (2012).
15. Imrichová, H., Hulselmans, G., Atak, Z., Potier, D. & Aerts, S. i-cisTarget 2015 update: generalized cis-regulatory enrichment analysis in human, mouse and fly. *Nucleic Acids Res.* **43**, W57–W64 (2015).
16. Becht, E. et al. Dimensionality reduction for visualizing single-cell data using UMAP. *Nat. Biotechnol.* **9**, 26 (2018).
17. Davie, K. et al. A single-cell transcriptome atlas of the aging *Drosophila* brain. *Cell* **174**, 1–38 (2018).
18. Potier, D. et al. Mapping gene regulatory networks in *Drosophila* eye development by large-scale transcriptome perturbations and motif inference. *Cell Rep.* **9**, 2290–2303 (2014).
19. Svensson, V., Vento-Tormo, R. & Teichmann, S. A. Exponential scaling of single-cell RNA-seq in the past decade. *Nat. Protoc.* **13**, 599–604 (2018).
20. Sanguinetti, G. & Huynh-Thu, V. A. *Gene Regulatory Networks: Methods and Protocols* (Springer, 2019).
21. Kurtzer, G. M., Sochat, V. & Bauer, M. W. Singularity: scientific containers for mobility of compute. *PLoS ONE* **12**, e0177459 (2017).
22. Pratapa, A., Jalihal, A. P., Law, J. N., Bharadwaj, A. & Murali, T. Benchmarking algorithms for gene regulatory network inference from single-cell transcriptomic data. *Nat. Methods* **17**, 147–154 (2020).
23. Lambert, S. A. et al. The human transcription factors. *Cell* **172**, 650–665 (2018).
24. Fiers, M. W. et al. Mapping gene regulatory networks from single-cell omics data. *Brief. Funct. Genomics* **17**, 246–254 (2018).
25. de Smet, R. & Marchal, K. Advantages and limitations of current network inference methods. *Nat. Rev. Microbiol.* **8**, 717–729 (2010).
26. Guo, M., Wang, H., Potter, S. S., Whitsett, J. A. & Xu, Y. SINCERA: a pipeline for single-cell RNA-seq profiling analysis. *PLoS Comput. Biol.* **11**, e1004575 (2015).
27. Mohammadi, S., Ravindra, V., Gleich, D. F. & Grama, A. A geometric approach to characterize the functional identity of single cells. *Nat. Commun.* **9**, 1516 (2018).
28. van Dijk, D. et al. Recovering gene interactions from single-cell data using data diffusion. *Cell* **174**, 716–729 (2018).
29. Deshpande, A., Chu, L.-F., Stewart, R. & Gitter, A. Network inference with Granger causality ensembles on single-cell transcriptomic data. Preprint at https://www.biorxiv.org/content/10.1101/534834v1 (2019).
30. Zeisel, A. et al. Cell types in the mouse cortex and hippocampus revealed by single-cell RNA-seq. *Science* **347**, 1138–1142 (2015).
31. Chen, X., Teichmann, S. A. & Meyer, K. B. From tissues to cell types and back: single-cell gene expression analysis of tissue architecture. *Annu. Rev. Biomed. Data Sci.* **1**, 1–23 (2018).
32. Tirosh, I. & Suvà, M. L. Deciphering human tumor biology by single-cell expression profiling. *Annu. Rev. Cancer Biol.* **3**, 1–16 (2018).
33. Obaldia, M. & Bhandoola, A. Transcriptional regulation of innate and adaptive lymphocyte lineages. *Annu. Rev. Immunol.* **33**, 1–36 (2014).
34. Laresgoiti, U. et al. E2F2 and CREB cooperatively regulate transcriptional activity of cell cycle genes. *Nucleic Acids Res.* **41**, 10185–10198 (2013).
35. Knox, J. J., Cosma, G. L., Betts, M. R. & McLane, L. M. Characterization of T-bet and eomes in peripheral human immune cells. *Front. Immunol.* **5**, 217 (2014).
36. Lin, Y. C. et al. A global network of transcription factors, involving E2A, EBF1 and Foxo1, that orchestrates B cell fate. *Nat. Immunol.* **11**, 635 (2010).
37. Boller, S. & Grosschedl, R. The regulatory network of B-cell differentiation: a focused view of early B-cell factor 1 function. *Immunol. Rev.* **261**, 102–115 (2014).
38. Suo, S. et al. Revealing the critical regulators of cell identity in the mouse cell atlas. *Cell Rep.* **25**, 1436–1445 (2018).
39. Shannon, P. et al. Cytoscape: a software environment for integrated models of biomolecular interaction networks. *Genome Res.* **13**, 2498–2504 (2003).
40. Jerby-Arnon, L. et al. A cancer cell program promotes T cell exclusion and resistance to checkpoint blockade. *Cell* **175**, 984–997 (2018).
41. Puram, S. V. et al. Single-cell transcriptomic analysis of primary and metastatic tumor ecosystems in head and neck cancer. *Cell* **171**, 1611–1624 (2017).
42. Lambrechts, D. et al. Phenotype molding of stromal cells in the lung tumor microenvironment. *Nat. Med.* **24**, 1277–1289 (2018).
43. Pavlidis, P. & Noble, W. S. Analysis of strain and regional variation in gene expression in mouse brain. *Genome Biol.* **2**, research0042.1 (2001).
44. Zhang, B. & Horvath, S. A general framework for weighted gene co-expression network analysis. *Stat. Appl. Genet. Mol.* **4**, Article17 (2005).
45. Frith, M. C., Li, M. C. & Weng, Z. Cluster-Buster: finding dense clusters of motifs in DNA sequences. *Nucleic Acids Res.* **31**, 3666–3668 (2003).
46. Zweig, A. S., Karolchik, D., Kuhn, R. M., Haussler, D. & Kent, J. W. UCSC genome browser tutorial. *Genomics* **92**, 75–84 (2008).
47. Aerts, S. et al. Gene prioritization through genomic data fusion. *Nat. Biotechnol.* **24**, 537–544 (2006).
48. Consortium, E. An integrated encyclopedia of DNA elements in the human genome. *Nature* **489**, 57–74 (2012).

49. Vilella, A. J. et al. EnsemblCompara GeneTrees: complete, duplication-aware phylogenetic trees in vertebrates. *Genome Res.* **19**, 327–335 (2009).
50. Gupta, S., Stamatoyannopoulos, J. A., Bailey, T. L. & Noble, W. Quantifying similarity between motifs. *Genome Biol.* **8**, R24 (2007).
51. Subramanian, A. et al. Gene set enrichment analysis: a knowledge-based approach for interpreting genome-wide expression profiles. *Proc. Natl Acad. Sci. USA* **102**, 15545–15550 (2005).
52. Luecken, M. D. & Theis, F. J. Current best practices in single-cell RNA-seq analysis: a tutorial. *Mol. Syst. Biol.* **15**, e8746 (2019).
53. van der Maaten, L. & Hinton, G. Visualizing data using t-SNE. *J. Mach. Learn. Res.* **9**, 2579–2605 (2008).
54. Blondel, V. D., Guillaume, J.-L., Lambiotte, R. & Lefebvre, E. Fast unfolding of communities in large networks. *J. Stat. Mech.* **2008**, P10008 (2008).
55. Pliner, H. A., Shendure, J. & Trapnell, C. Supervised classification enables rapid annotation of cell atlases. *Nat. Methods* **16**, 983–986 (2019).

Acknowledgements

This work was funded by VLAIO (no. HBC.2017.1003 to J.R., Y.S., and S. Aerts); by an ERC Consolidator Grant (no. 724226_cis-CONTROL to S. Aerts); and by the KU Leuven (grant no. C14/18/092 to S. Aerts). Computing was performed at the Vlaams Supercomputer Center. The funders had no role in study design, data collection and analysis, decision to publish, or preparation of the manuscript.

Author contributions

Conceptualization: B.V.d.S., C.F., J.R., Y.S., and S. Aerts; methodology: B.V.d.S., C.F., K.D., M.D.W., G.H., S. Aibar, R.S., W.S., R.C., Q.R., T.V., D.D.M., J.R., Y.S., and S. Aerts; software: B.V.d.S., C.F., K.D., M.D.W., G.H., S. Aibar, R.S., W.S., R.C., Q.R., T.V., and D.D.M.; validation, resources, and data curation: B.V.d.S. and C.F.; writing—original draft: B.V.d.S., C.F., and S. Aerts; writing—review and editing: B.V.d.S., C.F., and S. Aerts; visualization: B.V.d.S., C.F., and S. Aerts; supervision: S. Aerts., Y.S., and J.R.

Competing interests

The authors declare no competing interests.

Additional information

Supplementary information is available for this paper at https://doi.org/10.1038/s41596-020-0336-2.

Correspondence and requests for materials should be addressed to S.A.

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

Publisher's note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Received: 25 September 2019; Accepted: 17 April 2020;
Published online: 19 June 2020

Related links

Key references using this protocol

Davie, K. et al. *Cell* **174**, 982–998 (2018): https://doi.org/10.1016/j.cell.2018.05.057
Lambrechts, D. et al. *Nat. Med.* **24**, 1277–1289 (2018): https://doi.org/10.1038/s41591-018-0096-5
Wouters, J. et al. Preprint at *bioRxiv* (2019): https://www.biorxiv.org/content/10.1101/715995v2
Aibar, S. et al. *Nat. Methods* **14**, 1083–1086 (2017): https://doi.org/10.1038/nmeth.4463

nature research

Corresponding author(s): Stein Aerts

Last updated by author(s): Mar 20, 2020

Reporting Summary

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

Statistics

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

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

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

Software and code

Policy information about availability of computer code

Data collection	All data analyzed within this protocol are publicly available. No additional software was used for the data collection process.
Data analysis	Data analysis is described in-depth in this protocol, including version numbers. We mainly used two Python software packages: pySCENIC v0.9.18, Scanpy v1.4.4.post1, which are built upon Python 3.7+. These packages in turn have many dependencies, including anndata (v0.6.22), arboreto (v0.1.5), loompy (v2.0.17), numpy (v1.16.2), pandas (v0.23.4), scikit-learn (v0.21.2). In addition, we used Nextflow (v19.08.1-edge.5131).

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

Data

Policy information about availability of data

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

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

All data analyzed within this protocol are publicly available. The PBMC 10k data set is directly available for download from the 10x Genomics company website: https://support.10xgenomics.com/single-cell-gene-expression/datasets/3.0.0/pbmc_10k_v3. The following data sets are available from NCBI's Gene Expression Omnibus (GEO) and are accessible through GEO Series accession numbers: GSE60361 (mouse brain data set), GSE115978 (human cutaneous melanoma), and GSE103322 (human head and neck squamous cell carcinoma). The non-small cell lung carcinoma data set can be downloaded from ArrayExpress (experiments E-MTAB-6149 and E-MTAB-6653). Additional meta-data is available as the supplementary information files from the original publications that generated these data sets.

Field-specific reporting

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

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

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

Life sciences study design

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

Sample size	No sample size calculation was performed. All data used in this protocol were taken from public resources and used to demonstrate the use of the protocol.
Data exclusions	From the single cell RNA-seq public datasets, we performed basic filtering on cells and genes, to exclude those with potential technical issues. This is described in detail, including parameters used, in the Preprocessing section of the Workflow.
Replication	We verified that all analyses shown in this protocol were reproducible, and include steps to reproduce the analyses in Jupyter notebooks available at https://github.com/aertslab/SCENICprotocol/. For analysis steps that use stochastic algorithms, we provide the random seed that we used to generate the results included in the protocol.
Randomization	We used the sample classifications from the original publications, where applicable, for the public data analyzed in this protocol.
Blinding	The public data analyzed consisted of either single-sample experiments, or case/control studies, for which we used the sample groups from the original publications. We did not use any additional blinding of sample groups when analyzing these data.

Reporting for specific materials, systems and methods

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

Materials & experimental systems

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

Methods

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