Chapter 10

Inference of Gene Regulatory Network from Single-Cell Transcriptomic Data Using pySCENIC

Nilesh Kumar, Bharat Mishra, Mohammad Athar, and Shahid Mukhtar

Abstract

With the advent of recent next-generation sequencing (NGS) technologies in genomics, transcriptomics, and epigenomics, profiling single-cell sequencing became possible. The single-cell RNA sequencing (scRNA-seq) is widely used to characterize diverse cell populations and ascertain cell type-specific regulatory mechanisms. The gene regulatory network (GRN) mainly consists of genes and their regulators—transcription factors (TF). Here, we describe the lightning-fast Python implementation of the SCENIC (Single-Cell reEgulatory Network Inference and Clustering) pipeline called pySCENIC. Using single-cell RNA-seq data, it maps TFs onto gene regulatory networks and integrates various cell types to infer cell-specific GRNs. There are two fast and efficient GRN inference algorithms, GRNBoost2 and GENIE3, optionally available with pySCENIC. The pipeline has three steps: (1) identification of potential TF targets based on co-expression; (2) TF-motif enrichment analysis to identify the direct targets (regulons); and (3) scoring the activity of regulons (or other gene sets) on single cell types.

Key words Gene co-expression network, Gene regulatory network, scRNA-seq, RNA-Seq count data

1 Introduction

The RNA-seq technologies have been widely used for the discovery and relative quantification of transcripts in a diverse organisms including animals and plants [1–6]. Such transcriptomics data provides an opportunity for co-expression network construction to understand the correlation among gene expression patterns [7]. Furthermore, the regulatory nodes within co-expressed genes can be highlighted using a wide variety of indicators of centralities for the co-expression network [2, 8, 9]. The most prominent indicators of centralities applicable to biological networks, including co-expression networks, are degree, closeness centrality, betweenness centrality, eigenvector centrality, PageRank centrality, etc. [10, 11]. The main purpose of using these indicators of centralities is to rank the nodes that are often overlooked such as genes not currently known as TFs, but playing significant regulatory

Shahid Mukhtar (ed.), *Modeling Transcriptional Regulation: Methods and Protocols*, Methods in Molecular Biology, vol. 2328, https://doi.org/10.1007/978-1-0716-1534-8_10,

roles. It has been shown that most of the genes or proteins that play crucial roles in diverse biological processes possess high centrality [7, 10–14]. Recently, a similar approach has been applied to bracket regulatory genes involved in host modifying processes such as protein translation, ubiquitination, interferon signaling, and the cytokine storm in SARS-CoV-2 pathogenesis [15].

In recent years, single-cell RNA sequencing (scRNA-seq) [16] has beaertslabe the preferred choice. The key difference between conventional RNA-seq and scRNA-seq technologies is their resolution and precision at the cellular level rather than tissue level [16–19]. Each cell has a unique transcriptome and thus requires an individualized snapshot of its transcription profiles that can be provided by the scRNA-seq sequencing technique. While RNA-seq is only able to highlight expression patterns of large numbers of cells together, scRNA-seq provides a typical window of opportunity to obtain an expression profile at the single-cell level [19]. Additionally, the bulk analysis techniques are prone to incorrect assumptions of homogeneity of the pooled cells and usually underrate the relevant expression patterns in rare cell populations [20]. To be able to establish an understanding of the regulatory network of cells in different conditions, a wide variety of gene regulatory networks (GRNs) mapping algorithms have been published [21]. Some of the algorithms were developed for bulk sequencing data such as RNA-seq GRN but they can also be utilized for scRNA-seq GRN inference.

1.1 GENIE3

GENIE3 was the best performer algorithm in the DREAM4; *In Silico Multifactorial challenge*, for the inference of GRNs of bulk transcriptional data [22]. GENIE3 can recover regulatory networks from gene multivariate expression data. The resulting regulatory networks are directed graphs with "V" genes connected through unsigned directed edges. GRN is reconstructed using "V" regression task to obtain a subset of genes whose expression profiles are the most predictive of a target gene's expression profile.

1.2 PPCOR

PPCOR is an R package that can be used to aertslabpute the pairwise partial correlations and *p*-value for each pair of genes with respect to all the other genes [23]. After partial correlation aertslabputation, an undirected regulatory network is constructed where a sign of the correlation is used to infer negative and positive regulation.

1.3 GRNBoost2

GRNBoost2 is an algorithm faster than GENIE3, and a possible alternative when the datasets have a high volume of observations. The key difference between both methods is the use of stochastic gradient boosting machine regression with early stopping regularization to infer the network in GRNBoost2, while GENIE3 aertslabputes regression for each gene [24].

1.4 MICRAT

MICRAT (Maximal Information coefficient with Conditional Relative Average entropy and Time-series mutual information) is used to infer GRNs from time-series expression data. It uses the maximal information coefficient to construct an undirected network of genes. Subsequently, the conditional relative average entropies of each pair of genes are used to interpret the potential regulators and their targets in the network [25].

1.5 PIDC

The partial information decomposition (PIDC) method is based upon information theory to calculate statistical dependencies between triplets of genes in expression datasets. It measures the average ratio of unique information between two genes across all of the third genes in the remainder of the network [26].

2 Materials

1. The SCENIC workflow can be set up on a standard UNIX (Linux or MacOS reaertslabmended) or Windows 10 workstation. For efficient analysis, High Performance Computing (HPC) cluster is recommended. For Windows operation system, it is recommended to install Windows Subsystem for Linux.
2. The workflow is available in both R (version 4.0.3) and Python (version 3) environments. The Python implementation (*pySCENIC*) is recommended over R implementation because of its efficient multiprocessing capability since it uses Dask parallel processing.
3. Conda or Miniconda virtual environments for Python and BiocManager package manager for Bioconductor in R, respectively [27].
4. HPC cluster access, for faster processing 24 core and at least 64 gigabytes (Gb).
5. Create a virtual environment and install conda and pip packages needed for the analysis (*see* **Note 1**).
6. Create working directory and subdirectories (Bash Shell).

```
$ mkdir pySCENIC
$ $wd = pySCENIC
$ cd $wd
$ mkdir pySCENIC_data
$ mkdir pySCENIC_data/auxilliaries
$ mkdir pySCENIC_data/figures
$ mkdir pySCENIC_data/resources
$ mkdir pySCENIC_data/results
```

7. Download meta and supporting files (Bash Shell).

```
$ cd pySCENIC_data/auxilliaries
$ wget
https://raw.githubusercontent.aertslab/aertslab/pySCENIC/ma
ster/resources/lambert2018.txt
$ wget
https://resources.aertslab.org/cistarget/databases/hom
o_sapiens/hg19/refseq_r45/mc9nr/gene_based/hg19-500bp-
upstream-10species.mc9nr.feather
$ wget
https://resources.aertslab.org/cistarget/databases/hom
o_sapiens/hg19/refseq_r45/mc9nr/gene_based/hg19-tss-
centered-5kb-10species.mc9nr.feather
$ wget
https://resources.aertslab.org/cistarget/databases/hom
o_sapiens/hg19/refseq_r45/mc9nr/gene_based/hg19-tss-
centered-10kb-10species.mc9nr.feather
$ wget
https://resources.aertslab.org/cistarget/motif2tf/
motifs-v9-nr.hgnc-m0.001-o0.0.tbl
```

Keep the cell annotations, cell count, Transcripts per Kilobase Million (TMP) and metadata files in the "resources" directory (*see* **Note 2**).

3 Methods

For efficient processing, some of the analysis steps should be done in command-line mode, and the rest of the steps can be done using Python scripts. The whole workflow has four major steps: (1) preprocessing, (2) GRN analysis using expression data, (3) TF-motif enrichment and regulons-cell enrichment, and (4) scoring.

3.1 Preprocessing and Cleaning of the Data (Python Shell)

3.1.1 Import Python Libraries

```
import os, glob, re, pickle
from functools import partial
from collections import OrderedDict
import operator as op
from cytoolz import compose
import pandas as pd
import seaborn as sns
import numpy as np
import scanpy as sc
import anndata as ad
import matplotlib as mpl
import matplotlib.pyplot as plt
from pyscenic.export import export2loom,
```

```
add_scenic_metadata
from pyscenic.utils import load_motifs
from pyscenic.transform import df2regulons
from pyscenic.aucell import aucell
```

3.1.2 Setup the File and Folder Variables (Python Shell)

```
wd = "pySCENIC_data/" #working directory
RESOURCES_FOLDERNAME = wd + "resources/"
AUXILLIARIES_FOLDERNAME = wd + "auxilliaries/"
RESULTS_FOLDERNAME = wd + "results/"
FIGURES_FOLDERNAME = wd + "figures/"
BASE_URL = "http://motifcollections.aertslab.
org/v9/logos/"
COLUMN_NAME_LOGO = "MotifLogo"
COLUMN_NAME_MOTIF_ID = "MotifID"
COLUMN_NAME_TARGETS = "TargetGenes"
HUMAN_TFS_FNAME =
os.path.join(AUXILLIARIES_FOLDERNAME,'lambert2018.txt')
          RANKING_DBS_FNAMES = list(map(lambda fn:
          os.path.join(AUXILLIARIES_FOLDERNAME, fn),

['hg19-500bp-upstream-
          10species.mc9nr.feather',

'hg19-tss-centered-5kb-
          10species.mc9nr.feather',

'hg19-tss-centered-10kb-
          10species.mc9nr.feather']))
          MOTIF_ANNOTATIONS_FNAME =
        os.path.join(AUXILLIARIES_FOLDERNAME, 'motifs-
v9-
          nr.hgnc-m0.001-o0.0.tbl')
          DATASET_ID = "Experiment_name" # for example
GEO ID
          TCGA_CODE = 'CANCERTYPE' # TCGA cancer code.
          CELL_ANNOTATIONS_FNAME =
os.path.join(RESOURCES_FOLDERNAME," cell.annotations.csv")
          SAMPLE_METADATA_FNAME =
        os.path.join(RESOURCES_FOLDERNAME, "meta_data.
xlsx")
          EXP_MTX_TPM_FNAME =
          os.path.join(RESOURCES_FOLDERNAME,
          'Expression_tpm.csv')
          EXP_MTX_COUNTS_FNAME =
          os.path.join(RESOURCES_FOLDERNAME,
```

```
                    'Expression_counts.csv')
                    # Output files and folders.
                     METADATA_FNAME = os.path.join(RESULTS_FOLDE
RNAME,
                    '{}.metadata.csv'.format(DATASET_ID))
                       EXP_MTX_QC_FNAME = os.path.join(RESULTS_
FOLDERNAME,
                    '{}.qc.tpm.csv'.format(DATASET_ID))
                      ADJACENCIES_FNAME = os.path.join(RESULTS_
FOLDERNAME,
                    '{}.adjacencies.tsv'.format(DATASET_ID))
                  MOTIFS_FNAME = os.path.join(RESULTS_FOLDERNAME,
                    '{}.motifs.csv'.format(DATASET_ID))
                     REGULONS_DAT_FNAME = os.path.join(RESULTS_
FOLDERNAME,
                    '{}.regulons.dat'.format(DATASET_ID))
                       AUCELL_MTX_FNAME = os.path.join(RESULTS_
FOLDERNAME,
                    '{}.auc.csv'.format(DATASET_ID))
                  BIN_MTX_FNAME = os.path.join(RESULTS_FOLDERNAME
,
                    '{}.bin.csv'.format(DATASET_ID))
                   THR_FNAME = os.path.join(RESULTS_FOLDERNAME,
                    '{}.thresholds.csv'.format(DATASET_ID))
                  ANNDATA_FNAME = os.path.join(RESULTS_FOLDERNAME
,
                    '{}.h5ad'.format(DATASET_ID))
                  LOOM_FNAME = os.path.join(RESULTS_FOLDERNAME,
                    '{}_{}.loom'.format(TCGA_CODE, DATASET_ID))
```

3.1.3 Cleaning the Data (Python Shell)

```
df_annotations = pd.read_csv(CELL_ANNOTATIONS_FNAME)
df_annotations['samples'] =
df_annotations['samples'].str.upper()
df_annotations.rename(columns={'cell.types':
'cell_type', 'cells': 'cell_id', 'samples':
'sample_id',
                                    'treatment.group':
'treatment_group', 'Cohort': 'cohort'}, inplace=True)
          df_annotations['cell_type'] =
          df_annotations.cell_type.replace({
                 'Mal': 'Malignant Cell',
                 'Endo.': 'Endothelial Cell',
                 'T.CD4': 'Thelper Cell',
                 'CAF': 'Fibroblast',
                 'T.CD8': 'Tcytotoxic Cell',
                 'T.cell': 'T Cell',
```

```
                                'NK': 'NK Cell',
                                'B.cell': 'B Cell'})
            df_samples = pd.read_excel(SAMPLE_METADA TA_FNAME,
            header=2)
            df_samples = df_samples.drop(['Cohort'], axis=1)
            df_samples['Sample'] = df_samples.Sample.str.upper()
            df_metadata = pd.merge(df_annotations, df_samples,
            left_on='sample_id', right_on='Sample')
            df_metadata = df_metadata.drop(['Sample',
            'treatment_group'], axis=1)
                            df_metadata.rename(columns={'Patient':
'patient_id',
```

3.1.4 Quality Control of TPM Data (Python Shell)

```
df_tpm = pd.read_csv(EXP_MTX_TPM_FNAME, index_col=0)
df_tpm.shape
adata = sc.AnnData(X=df_tpm.T.sort_index())
df_obs = df_metadata[['cell_id', 'sample_id',
'cell_type', 'cohort', 'patient_id', 'age', 'sex',
treatment',
'treatment_group', 'lesion_type',
'site']].set_index('cell_id').sort_index()
adata.obs = df_obs
adata.var_names_make_unique()
sc.pp.filter_cells(adata, min_genes=200)
sc.pp.filter_genes(adata, min_cells=3)
adata.raw = adata
sc.pp.log1p(adata)
adata.write_h5ad(ANNDATA_FNAME)
adata.to_df().to_csv(EXP_MTX_QC_FNAME) # Write csv
file on the disk.
```

3.2 Gene Regulatory Network (GRN) Analysis Using grnboost2 (See Note 3)

For this step use command-line version of pySCENIC to use multiple processors (Bash Shell).

```
$ pyscenic grn
pySCENIC_data/results/GSE115978.qc.tpm.csv \
pySCENIC_data/auxilliaries/lambert2018.txt \
-o pySCENIC_data/results/GSE115978.adjacencies.tsv \
--num_workers 16
```

3.3 Create Regulons

Using RcisTarget then identify modules for which the regulator's binding motif is significantly enriched across the target genes and creates regulons with direct targets (*see* **Note 4**) (Python Shell).

```
$ pyscenic ctx
pySCENIC_data/results/GSE115978.adjacencies.tsv
```

```
pySCENIC_data/auxilliaries/hg19-500bp-upstream-
10species.mc9nr.feather
pySCENIC_data/auxilliaries/hg19-tss-centered-5kb-
10species.mc9nr.feather
pySCENIC_data/auxilliaries/hg19-tss-centered-10kb-
10species.mc9nr.feather \
--annotations_fname pySCENIC_data/auxilliaries/motifs-
v9-nr.hgnc-m0.001-o0.0.tbl \
--expression_mtx_fname
pySCENIC_data/results/GSE115978.qc.tpm.csv \
--output pySCENIC_data/results/GSE115978.motifs.csv \
--mask_dropouts \
--num_workers 10
```

3.4 Identify Cells with Active Gene Sets

Using AUCell method identify cells with active gene regulatory networks. For this part of the analysis use pyscenic.aucell Python library (Python Shell).

```
df_motifs = load_motifs(MOTIFS_FNAME)
regulons = derive_regulons(df_motifs)
auc_mtx = aucell(df_tpm.T, regulons, num_workers=26)
auc_mtx.to_csv(AUCELL_MTX_FNAME)
```

3.5 Save Loom File on Disk (Python Shell).

```
export2loom(adata.to_df(), regulons, LOOM_FNAME,
cell_annotations=adata.obs['cell_type'].to_dict(),
                    embeddings=OrderedDict([('AUCell + tSNE',
embedding_aucell_tsne), ('PCA + tSNE', embedding_pca_tsne)]),
                  auc_mtx = auc_mtx,
                  tree_structure=(),
                  title='{}_{}'.format(TCGA_CODE,
DATASET_ID),
                  nomenclature="HGNC",
auc_thresholds=thresholds,
                  compress=True)
```

4 Notes

1. pySCENIC Installation
   (a) Create and set up a new conda environment.

```
$ conda create -n scenic python=3.6
```

   (b) Activate the conda environment.

```
$ conda activate scenic
```

(c) Install conda packages

```
$ conda install numpy pandas matplotlib seaborn
$ conda install -c anaconda xlrd
$ conda install -c anaconda cytoolz
$ conda install seaborn scikit-learn statsmodels
numba pytables
$ conda install -c conda-forge python-igraph
louvain
$ conda install -c conda-forge multicore-tsne
$ conda install dask==2.9
```

(d) Install pip packages

```
$ pip install scanpy
$ pip install pyscenic
```

2. Resource directory file format.
   (a) Cell annotation csv file (comma delimited).

Cells	Samples	Cell. types	Treatment. group	Cohort	No.of. genes	No.of. reads
Cell 1	Mel78	Mal	post.treatment	Tirosh	8258	357,919
Cell 2	Mel79	Mal	treatment. naive	Tirosh	2047	5727
Cell 3	Mel88	Mal	post.treatment	Tirosh	5375	139,218
Cell 4	Mel79	Mal	treatment. naive	Tirosh	5648	73,996
Cell *n*	Mel78	Mal	post.treatment	Tirosh	7409	380,341

(b) The cell count file (CSV format). The first column should contain the gene names whereas the first rows should be named after the individual cell types. Keep the first cell of the file empty.

	Cell 1	Cell 2	Cell 3	Cell 4	Cell *n*
C9orf152	0	0	0	0	0
RPS11	370	1	75	15	345
ELMO2	43	0	2	18	43
CREB3L1	0	0	0	0	0
PNMA1	68	0	1	8	103

(c) The TPM normalized expression CSV file. The first column should contain the gene names whereas the first rows should be named after the individual cell types. Keep the first cell of the file empty.

	Cell 1	Cell 2	Cell 3	Cell 4	Cell *n*
C9orf152	0	0	0	0	0
RPS11	8.144184	5.915091	7.243164	6.019502	7.974753
ELMO2	2.639232	0	0.732052	3.687956	2.599318
CREB3L1	0	0	0	0	0
PNMA1	3.656496	0	0.536053	3.041418	4.132741

(d) The excel sheet of the metadata file.

Sample	Patient	Age	Sex	Treatment	Treatment group	Lesion type	Site
Mel53	Mel53	77	F	None	Untreated	Metastasis	Subcutaneous back lesion
Mel58	Mel58	83	M	Ipilimumab	Post-ICI (resistant)	Metastasis	Subcutaneous leg lesion
Mel60	Mel60	60	M	Trametinib, ipilimumab	Post-ICI (resistant)	Metastasis	Spleen
Mel71	Mel71	79	M	None	Untreated	Metastasis	Transverse colon
Mel72	Mel72	57	F	IL-2, nivolumab, ipilimumab + anti-KIR-Ab	Post-ICI (resistant)	Metastasis	External iliac lymph node

3. *Choose GRN algorithm*. In pySCENIC there are two available algorithms genie3 and grnboost2 out of these two method grnboost2 is the default. To choose genie3 use:

```
-m or --method [genie3]
```

4. Multiprocessing mode setup. Depending upon the Dask and pySCENIC version and HPC facility choose appropriate multiprocessing mode.

```
--mode "dask_multiprocessing" # Fast and default
--mode "custom_multiprocessing" or
```

Acknowledgments

This work was supported by the National Science Foundation (IOS-1557796 and IOS-2038872) to M.S.M., and U54 ES 030246 from NIH/NIEHS to M.A. and M.S.M. We also acknowledge Dr. Karolina Mukhtar for critical read of this book chapter.

References

1. Sengupta U, Ukil S, Dimitrova N et al (2009) Expression-based network biology identifies alteration in key regulatory pathways of type 2 diabetes and associated risk/complications. PLoS One 4(12):e8100. https://doi.org/10.1371/journal.pone.0008100
2. Garbutt CC, Bangalore PV, Kannar P et al (2014) Getting to the edge: protein dynamical networks as a new frontier in plant-microbe interactions. Front Plant Sci 5:312. https://doi.org/10.3389/fpls.2014.00312
3. Tully JP, Hill AE, Ahmed HM et al (2014) Expression-based network biology identifies immune-related functional modules involved in plant defense. BMC Genomics 15:421. https://doi.org/10.1186/1471-2164-15-421
4. Naqvi RZ, Zaidi SS, Akhtar KP et al (2017) Transcriptomics reveals multiple resistance mechanisms against cotton leaf curl disease in a naturally immune cotton species, Gossypium arboreum. Sci Rep 7(1):15880. https://doi.org/10.1038/s41598-017-15963-9
5. Naqvi RZ, Zaidi SS, Mukhtar MS et al (2019) Transcriptomic analysis of cultivated cotton Gossypium hirsutum provides insights into host responses upon whitefly-mediated transmission of cotton leaf curl disease. PLoS One 14(2):e0210011. https://doi.org/10.1371/journal.pone.0210011
6. Zaidi SS, Naqvi RZ, Asif M et al (2020) Molecular insight into cotton leaf curl geminivirus disease resistance in cultivated cotton (Gossypium hirsutum). Plant Biotechnol J 18(3):691–706. https://doi.org/10.1111/pbi.13236
7. Mishra B, Sun Y, Howton TC et al (2018) Dynamic modeling of transcriptional gene regulatory network uncovers distinct pathways during the onset of Arabidopsis leaf senescence. NPJ Syst Biol Appl 4:35. https://doi.org/10.1038/s41540-018-0071-2
8. Mishra B, Sun Y, Ahmed H et al (2017) Global temporal dynamic landscape of pathogen-mediated subversion of Arabidopsis innate immunity. Sci Rep 7(1):7849. https://doi.org/10.1038/s41598-017-08073-z
9. McCormack ME, Lopez JA, Crocker TH et al (2016) Making the right connections: network biology and plant immune system dynamics. Curr Plant Biol 5:2–12
10. Mishra B, Kumar N, Mukhtar MS (2019) Systems biology and machine learning in plant-pathogen interactions. Mol Plant-Microbe Interact 32(1):45–55. https://doi.org/10.1094/MPMI-08-18-0221-FI
11. Ahmed H, Howton TC, Sun Y et al (2018) Network biology discovers pathogen contact points in host protein-protein interactomes. Nat Commun 9(1):2312. https://doi.org/10.1038/s41467-018-04632-8
12. Jeong H, Mason SP, Barabasi AL et al (2001) Lethality and centrality in protein networks. Nature 411(6833):41–42. https://doi.org/10.1038/35075138
13. Li X, Li W, Zeng M et al (2020) Network-based methods for predicting essential genes or proteins: a survey. Brief Bioinform 21(2):566–583. https://doi.org/10.1093/bib/bbz017
14. van Leeuwen J, Pons C, Tan G et al (2020) Systematic analysis of bypass suppression of essential genes. Mol Syst Biol 16(9):e9828. https://doi.org/10.15252/msb.20209828
15. Kumar N, Mishra B, Mehmood A et al (2020) Integrative network biology framework elucidates molecular mechanisms of SARS-CoV-2 pathogenesis. iScience 23(9):101526. https://doi.org/10.1016/j.isci.2020.101526
16. Haque A, Engel J, Teichmann SA et al (2017) A practical guide to single-cell RNA-sequencing for biomedical research and clinical applications. Genome Med 9(1):75. https://doi.org/10.1186/s13073-017-0467-4
17. Olsen TK, Baryawno N (2018) Introduction to single-cell RNA sequencing. Curr Protoc Mol Biol 122(1):e57. https://doi.org/10.1002/cpmb.57
18. Hwang B, Lee JH, Bang D (2018) Single-cell RNA sequencing technologies and bioinformatics pipelines. Exp Mol Med 50(8):96. https://doi.org/10.1038/s12276-018-0071-8

19. Chen G, Ning B, Shi T (2019) Single-cell RNA-Seq technologies and related computational data analysis. Front Genet 10:317. https://doi.org/10.3389/fgene.2019.00317
20. Todorov H, Cannoodt R, Saelens W et al (2019) Network inference from single-cell transcriptomic data. Methods Mol Biol 1883:235–249. https://doi.org/10.1007/978-1-4939-8882-2_10
21. Chen S, Mar JC (2018) Evaluating methods of inferring gene regulatory networks highlights their lack of performance for single cell gene expression data. BMC Bioinformatics 19 (1):232. https://doi.org/10.1186/s12859-018-2217-z
22. Huynh-Thu VA, Irrthum A, Wehenkel L et al (2010) Inferring regulatory networks from expression data using tree-based methods. PLoS One 5(9). https://doi.org/10.1371/journal.pone.0012776
23. Kim S (2015) ppcor: an R package for a fast calculation to semi-partial correlation coefficients. Commun Stat Appl Methods 22 (6):665–674. https://doi.org/10.5351/CSAM.2015.22.6.665
24. Moerman T, Aibar Santos S, Bravo Gonzalez-Blas C et al (2019) GRNBoost2 and Arboreto: efficient and scalable inference of gene regulatory networks. Bioinformatics 35 (12):2159–2161. https://doi.org/10.1093/bioinformatics/bty916
25. Yang B, Xu Y, Maxwell A et al (2018) MICRAT: a novel algorithm for inferring gene regulatory networks using time series gene expression data. BMC Syst Biol 12 (Suppl 7):115. https://doi.org/10.1186/s12918-018-0635-1
26. Chan TE, Stumpf MPH, Babtie AC (2017) Gene regulatory network inference from single-cell data using multivariate information measures. Cell Syst 5(3):251–267.e253. https://doi.org/10.1016/j.cels.2017.08.014
27. Gentleman RC, Carey VJ, Bates DM et al (2004) Bioconductor: open software development for computational biology and bioinformatics. Genome Biol 5(10):R80. https://doi.org/10.1186/gb-2004-5-10-r80