JOURNAL OF COMPUTATIONAL BIOLOGY
Volume 29, Number 11, 2022

Pp. 1233–1236
DOI: 10.1089/cmb.2022.0244

The scINSIGHT Package for Integrating Single-Cell RNA-Seq Data from Different Biological Conditions

KUN QIAN,[1] SHIWEI FU,[2,3] HONGWEI LI,[1] and WEI VIVIAN LI[2,3]

ABSTRACT

Data integration is a critical step in the analysis of multiple single-cell RNA sequencing samples to account for heterogeneity due to both biological and technical variability. scINSIGHT is a new integration method for single-cell gene expression data, and can effectively use the information of biological condition to improve the integration of multiple single-cell samples. scINSIGHT is based on a novel non-negative matrix factorization model that learns common and condition-specific gene modules in samples from different biological or experimental conditions. Using these gene modules, scINSIGHT can further identify cellular identities and active biological processes in different cell types or conditions. Here we introduce the installation and main functionality of the scINSIGHT R package, including how to preprocess the data, apply the scINSIGHT algorithm, and analyze the output.

Keywords: clustering, data integration, non-negative matrix factorization, scRNA-seq.

1. INTRODUCTION

With the increasing popularity of single-cell RNA sequencing (scRNA-seq) technologies, integration is often a necessary step in data analysis to account for data heterogeneity due to both biological and technical variability. Although multiple methods have been developed for batch-effect removal or integrative analysis of multiple scRNA-seq samples (Tran et al., 2020), they do not effectively account for the information of biological conditions when single-cell samples come from multiple biological conditions, such as different experimental conditions or disease phases. This negligence might lead to compromised results of data integration and cell population identification.

To address this challenge, we have proposed a new method named scINSIGHT (Qian et al., 2022), which uses a joint non-negative matrix factorization model to learn common and condition-specific gene modules, and further identifies cell populations based on the common gene modules. In particular, when there are multiple samples from the same biological condition, scINSIGHT can effectively use the condition information to help distinguish between biological and technical variability, which is challenging to achieve

[1]School of Mathematics and Physics, China University of Geosciences, Wuhan, China.
[2]Department of Biostatistics and Epidemiology, Rutgers School of Public Health, Rutgers, The State University of New Jersey, Piscataway, New Jersey, USA.
[3]Department of Statistics, University of California, Riverside, California, USA.

using other matrix factorization tools for genomic data (Yang and Michailidis, 2016; Stein-O'Brien et al., 2019; Welch et al., 2019; Zhang and Zhang, 2019). In this article, we describe the functionality and usage of the scINSIGHT R package, which is released under the GPL-3 license and available at (https://github.com/Vivianstats/scINSIGHT).

2. INSTALLATION

The scINSIGHT package is written in R and is available on the Comprehensive R Archive Network (CRAN) or GitHub.

```
# To install scINSIGHT from CRAN:
install.packages("scINSIGHT")
# To install scINSIGHT from GitHub:
if(!require(devtools)){install.packages("devtools")}
library(devtools)
devtools::install_github("Vivianstats/scINSIGHT")
```

3. CREATING AN SCINSIGHT OBJECT

To use the scINSIGHT method, users need to create an `scINSIGHT` object, which will be used to execute the algorithm and store the outputs. To create this object, scINSIGHT takes a list of normalized gene expression matrices and corresponding biological conditions as input. Each matrix represents one single-cell sample, where rows correspond to genes and columns correspond to cells. Besides, gene names (e.g., row names) should match and be in the same order in all expression matrices.

When data are already normalized and gene features are selected, users can directly use the `create_scINSIGHT` function to initialize an `scINSIGHT` object (Sheng and Li, 2021; Qian et al., 2022). Here we take six normalized matrices as an example, which are stored in a list object (`sim.counts`) and can be downloaded from https://doi.org/10.5281/zenodo.6551832. These six samples are simulated data from three time points (T1, T2, and T3), each containing 500 cells and 4977 genes (Qian et al., 2022).

```
sim.sample=c("S1","S2","S3","S4","S5","S6")
# Name of list should represent sample names
names(sim.counts)=sim.sample
sim.condition=c("T1","T1","T2","T2","T3","T3")
# Name of condition vector should match sample names
names(sim.condition)=sim.sample

# Create an scINSIGHT object
sim.scobj=create_scINSIGHT(norm.data=sim.counts,
condition=sim.condition)
```

If only raw count data are available, we suggest to first use the Seurat package (Stuart et al., 2019) to perform normalization and gene feature selection, then follow the aforementioned example to create the scINSIGHT object.

4. MODEL FITTING AND INTERPRETING RESULTS

After creating the `scINSIGHT` object, users can fit the model by running the `run_scINSIGHT` function, which takes the `scINSIGHT` object and other customizable parameters as input. Within the function, we have implemented a block coordinate descent algorithm to solve the matrix factorization problem, and a clustering algorithm to identify cell populations based on the identified common gene modules (Qian et al., 2022). The output will also be stored in an `scINSIGHT` object.

Several key parameters that can be customized include the following: (1) `K:` an integer or an integer vector specifying the candidate number(s) of common gene modules (defaults to `seq(5,15,2)`). If given a vector, an optimal number will be selected based on a stability criterion (Qian et al., 2022). (2) `num.cores:` an integer specifying the number of cores used for parallel computation. (3) `B:` an integer specifying the number of matrix initializations used for each value of `K` (defaults to 5).

In this example, we apply scINSIGHT to `sim.sobj` with the default parameters:

```
sim.scobj=run_scINSIGHT(sim.scobj, K=seq(5,15,2),
out.dir='./', B=5, num.cores=5)
```

After running the algorithm, the results will be stored in the `scINSIGHT` object `sim.scobj.` (1) The selected parameters are in `sim.scobj@parameters`. In this example, the selected `K` (number of common gene modules) is 13. (2) The clustering results are stored in `sim.scobj@clusters`, which is list of integer vectors. Each vector provides inferred cluster labels in one sample. (3) The expression levels of the common gene modules are stored in `sim.scobj@W_2`, which is a list of matrices. Each matrix represents one sample, and rows correspond to cells and columns correspond to the common gene modules. The normalized expression is stored in `sim.scobj@norm.W_2`, and can be used for visualization and downstream analysis. (4) The memberships of common gene modules are stored in `sim.scobj@V`, where rows correspond to common gene modules and columns correspond to genes. The values represent the coefficients of genes in corresponding modules. (5) The expression levels of the condition-specific gene modules are stored in `sim.scobj@W_1`, and the memberships are stored in `sim.scobj@H`.

Using the learned `sim.scobj` object, we can visualize the integrated data colored by the inferred clusters or true cell types. The code hereunder provides an example for the visualization task.

```
library(Rtsne)
library(ggplot2)
library(stringr)

# True cell types of single cells
celltype=toupper (str_replace_all (colnames (Reduce
(cbind, sim.counts)),"[.].*",""))
# Cluster labels of single cells
clusters=as.character (unlist (sim.scobj@clusters))

# Using normalized expression levels of common gene
modules to run tSNE
W2=Reduce(rbind, sim.scobj@norm.W_2)
set.seed(1)
res_tsne=Rtsne(W2, dims=2, check_duplicates=FALSE)

da_tsne=data.frame(tSNE1=res_tsne$Y[,1], tSNE2=
res_tsne$Y[,2], celltype=celltype, clusters=clusters)

# tSNE plots colored by cluster or cell type
p1=ggplot (da_tsne, aes (x=tSNE1, y=tSNE2, color=
celltype)) + geom_point (cex=.3, stroke=.3)

p2=ggplot (da_tsne, aes (x=tSNE1, y=tSNE2, color=
clusters)) + geom_point (cex=.3, stroke=.3)

cowplot:: plot_grid (p1, p2, nrow=1, ncol=2)
```

ACKNOWLEDGMENTS

We thank members of the Wei Vivian Li Lab for the helpful comments on our article.

AUTHOR DISCLOSURE STATEMENT

The authors declare they have no conflicting financial interests.

FUNDING INFORMATION

This study was supported by NIH R35GM142702 and Rutgers Busch Biomedical Grant (to W.V.L.).

REFERENCES

Qian, K., Fu, S., Li, H., et al. 2022. scINSIGHT for interpreting single-cell gene expression from biologically heterogeneous data. *Genome Biol*. 23, 82.

Sheng, J., and Li, W.V. 2021. Selecting gene features for unsupervised analysis of single-cell gene expression data. *Brief Bioinform*. 22, bbab295.

Stein-O'Brien, G. L., Clark, B. S., Sherman, T., et al. 2019. Decomposing cell identity for transfer learning across cellular measurements, platforms, tissues, and species. *Cell Syst*. 8, 395–411.

Stuart, T., Butler, A., Hoffman, P., et al. 2019. Comprehensive integration of single-cell data. *Cell* 177, 1888–1902.

Tran, H.T.N., Ang, K.S., Chevrier, M., et al. 2020. A benchmark of batch-effect correction methods for single-cell RNA sequencing data. *Genome Biol.* 21, 1–32.

Welch, J.D., Kozareva, V., Ferreira, A., et al. 2019. Single-cell multi-omic integration compares and contrasts features of brain cell identity. *Cell* 177, 1873–1887.

Yang, Z., and Michailidis, G. 2016. A non-negative matrix factorization method for detecting modules in heterogeneous omics multi-modal data. *Bioinformatics* 32, 1–8.

Zhang, L., and Zhang, S. 2019. Learning common and specific patterns from data of multiple interrelated biological scenarios with matrix factorization. *Nucleic Acids Res.* 47, 6606–6617.

Address correspondence to:
Dr. Wei Vivian Li
Department of Statistics
University of California, Riverside
900 University Avenue, Olmsted Hall 1337
Riverside, CA 92591
USA

E-mail: vivian.li@rutgers.edu