Simulating single-cell gene expression count data with preserved gene correlations by scDesign2

Tianyi Sun [1], Dongyuan Song [2], Wei Vivian Li [3] and Jingyi Jessica Li [1,4,5,6,*]

ABSTRACT

scDesign2 is a transparent simulator that generates high-fidelity single-cell gene expression count data with gene correlations captured. This article shows how to download and install the scDesign2 R package; how to fit probabilistic models (one per cell type) to real data and simulate synthetic data from the fitted models; and how to use scDesign2 to guide experimental design and benchmark computational methods. Finally, a note is given about cell clustering as a preprocessing step before model fitting and data simulation.

Background

In the burgeoning field of single-cell transcriptomics, a pressing challenge is to benchmark various experimental protocols and numerous computational methods in an unbiased manner. Although dozens

[1] Department of Statistics, University of California, Los Angeles, CA 90095-1554

[2] Interdepartmental Program of Bioinformatics, University of California, Los Angeles, CA 90095-7246

[3] Department of Biostatistics and Epidemiology, Rutgers School of Public Health, Piscataway, NJ 08854

[4] Department of Human Genetics, University of California, Los Angeles, CA 90095-7088

[5] Department of Computational Medicine, University of California, Los Angeles, CA 90095-1766

[6] Department of Biostatistics, University of California, Los Angeles, CA 90095-1772

[*] To whom correspondence should be addressed. Email: jli@stat.ucla.edu

of simulators had been developed for single-cell RNA-seq (scRNA-seq) data, they lacked the capacity to simultaneously achieve the following three goals: preserving genes, capturing gene correlations, and generating any number of cells with varying sequencing depths. To fill in this gap, we developed a new simulator scDesign2 (Sun et al., 2021) to achieve all three goals. Notably, scDesign2 can generate high-fidelity synthetic data of multiple scRNA-seq protocols and other single-cell gene expression count-based technologies.

This article provides a brief guide to the scDesign2 R package. For help troubleshooting or to provide feedback, please submit an issue to the GitHub page, which contains more documentation.

Installation

The required R version is no earlier than version 3.6.3. To install the scDesign2 package, users can run the following code in R.

```
if(!require(devtools)) install.packages("devtools"); library(devtools);
devtools::install_github("JSB-UCLA/scDesign2");
```

To use the package after the installation, users can run

```
library(scDesign2);
```

Model Fitting and Data Simulation

The input of scDesign2 is a real single-cell gene expression count matrix, where each row represents a gene, each column a cell, and each entry the expression level of a gene in a cell. In addition, each column needs to be labelled with the cell type that the cell belongs to. Based on this count matrix, scDesign2 would first fit one parametric probabilistic model for each cell type and then use the fitted models to simulate data.

In the R package, we have included an example scRNA-seq dataset, which profiles the transcriptome of mouse small intestinal epithelial cells (Haber et al., 2017). The file `mouse_sie_10x.rds` is the full dataset, and the file `mouse_sie_10x_demo.rds` is a sub-dataset containing 1000 genes and 30% cells for demonstration. In the following example code, we will select four cell types from the sub-dataset and perform model fitting and data simulation for each cell type. In scDesign2, the function for model fitting is `fit_model_scDesign2()`, and the function for data simulation is `simulate_count_scDesign2()`.

- Load data

```
data_mat_demo <-
```

```
  readRDS(system.file("extdata", "mouse_sie_10x_demo.rds",
                      package = "scDesign2"));
```

- Select four cell types; obtain the total cell number and cell type proportions

```
cell_type_sel <- c("Goblet", "Tuft", "TA.Early", "Enterocyte.Progenitor");
data_mat_demo_sel <-
  data_mat_demo[, colnames(data_mat_demo) %in% cell_type_sel];
n_cell_old <- ncol(data_mat_demo_sel);
cell_type_prop <- prop.table(table(colnames(data_mat_demo_sel)));
```

- Fit models and simulate data for the four cell types (running time within 14 mins on 4 cores)

```
RNGkind("L'Ecuyer-CMRG"); set.seed(1);
copula_result <- fit_model_scDesign2(
  data_mat_demo, cell_type_sel, sim_method = "copula",
  ncores = length(cell_type_sel));
sim_count_copula <- simulate_count_scDesign2(
  copula_result, sim_method = "copula",
  n_cell_new = n_cell_old, cell_type_prop = cell_type_prop);
```

In this example, the selected cell types are in the `cell_type_sel` vector, the fitted models are in the `copula_result` object, and the synthetic dataset is the the `sim_count_copula` matrix. We set the synthetic dataset to have the same total cell number (`n_cell_old`) and expected cell type proportions (`cell_type_prop`) as those of the input data matrix `data_mat_demo`, but users may change the `n_cell_new` and `cell_type_prop` arguments in the `simulate_count_scDesign2()` function.

To evaluate the quality of the synthetic dataset, we will combine the synthetic cells with the real cells and examine whether they are indistinguishable in the t-SNE visualization.

```
if(!require(Rtsne)) install.packages("Rtsne"); library(Rtsne);
set.seed(1);
Rtsne_combined <-
  Rtsne(log(t(cbind(data_mat_demo_sel, sim_count_copula)) + 1));
Rtsne_combined_vis <-
  data.frame(x = Rtsne_combined$Y[, 1], y = Rtsne_combined$Y[, 2],
             group = factor(c(rep("real", ncol(data_mat_demo_sel)),
                              rep("synthetic", ncol(sim_count_copula)))),
```

```
            cell_type = factor(c(colnames(data_mat_demo_sel),
                                 colnames(sim_count_copula))));
attach(Rtsne_combined_vis);
plot(x = x, y = y, pch = c(16, 2)[group],
     col = c("red", "blue", "green", "black")[cell_type]);
legend("topleft", legend = levels(cell_type),
       col=c("red", "blue", "green", "black"), pch = 16, bty="n");
legend("bottomright", legend = c("real", "synthetic"), pch = c(21, 2));
detach(Rtsne_combined_vis);
```

The t-SNE visualization shows that the synthetic cells mix well with the real cells.

Applications to Experimental Design and Computational Benchmarking

Two important applications of scDesign2 are guiding experimental design and benchmarking computational methods. This requires generating synthetic data with varying cell numbers and sequencing depths.

Here we demonstrate how to generate synthetic datasets with a fixed total cell number and varying sequencing depths. We will use `cell_type_sel`, `n_cell_old`, `cell_type_prop`, and `copula_result` from the pervious code. The first step is to calculate the sequencing depth of the real data.

```
total_count_old <- sum(data_mat_demo_sel);
```

To vary the sequencing depth, we change `total_count_old` by a factor of 1/8, 1/4, 1/2, 2, 4, or 8. The vector `adj_factor` contains all the multiplicative factors considered.

```
adj_factor <- c(1/8, 1/4, 1/2, 1, 2, 4, 8);
```

Finally, we use the following code for data simulation. In the `simulate_count_scDesign2()` function, the key arguments include `total_count_old`, `n_cell_old`, `total_count_new`, and `n_cell_new`. The first two arguments are the sequencing depth and total cell number of the real data, and the last two arguments are the sequencing depth and total cell number of the synthetic data to be generated. To fix the total cell number, we set `n_cell_new` to `n_cell_old`; to vary the sequencing depth, we specify `total_count_new` as `total_count_old` multiplied by each factor in the `adj_factor` vector, up to rounding. The list `sim_count` contains the synthetic datasets, one for each new sequencing depth `total_count_new`.

```
set.seed(1);
```

```
sim_count <- lapply(1:length(adj_factor), function(iter){
  simulate_count_scDesign2(copula_result,
    total_count_old = total_count_old, n_cell_old = n_cell_old,
    total_count_new = round(adj_factor[iter] * total_count_old),
    n_cell_new = n_cell_old, cell_type_prop = cell_type_prop,
    reseq_method = 'mean_scale', cell_sample = TRUE)});
```

A Note on Cell Clustering

The model fitting and data simulation of scDesign2 is performed for each cell type separately. Hence, partitioning cells into cell types is an important preprocessing step of scDesign2. The partitioning can be done based on biological knowledge, e.g., cell type marker genes, or by a clustering algorithm, e.g., SC3 (Kiselev et al., 2017) or the Louvain algorithm (Blondel et al., 2008).

On the GitHub page, we provide a proof-of-concept demonstration of how to perform cell clustering using the Louvain algorithm in the Seurat package (Stuart et al., 2019) and how to evaluate the clustering result using the ROGUE score (Liu et al., 2020). For scDesign2 users who do not have pre-defined cell types, they may follow our demonstration to do cell clustering before using scDesign2 to simulate data.

Software Availability

The scDesign2 R package is released under the MIT Liscence and available at https://github.com/JSB-UCLA/scDesign2.

Competing Interests

The authors declare no competing interests.

Funding

This work was supported by the following grants: NSF DBI-1846216 and DMS-2113754, NIGMS R01GM120507 and R35GM140888, Johnson & Johnson WiSTEM2D Award, Sloan Research Fellowship, and UCLA David Geffen School of Medicine W.M. Keck Foundation Junior Faculty Award (to J.J.L); Rutgers School of Public Health Pilot Grant and NJ ACTS BERD Mini-Methods Grant (to W.V.L).

References

Blondel, V. D., Guillaume, J.-L., Lambiotte, R., and Lefebvre, E. (2008). Fast unfolding of communities in large networks. *Journal of Statistical Mechanics: Theory and Experiment*, 2008(10).

Haber, A. L., Biton, M., Rogel, N., Herbst, R. H., Shekhar, K., Smillie, C., Burgin, G., Delorey, T. M., Howitt, M. R., Katz, Y., et al. (2017). A single-cell survey of the small intestinal epithelium. *Nature*, 551(7680):333–339.

Kiselev, V. Y., Kirschner, K., Schaub, M. T., Andrews, T., Yiu, A., Chandra, T., Natarajan, K. N., Reik, W., Barahona, M., Green, A. R., and et al. (2017). Sc3: consensus clustering of single-cell rna-seq data. *Nature Methods*, 14(5):483–486.

Liu, B., Li, C., Li, Z., Wang, D., Ren, X., and Zhang, Z. (2020). An entropy-based metric for assessing the purity of single cell populations. *Nature communications*, 11(1):1–13.

Stuart, T., Butler, A., Hoffman, P., Hafemeister, C., Papalexi, E., III, W. M. M., Hao, Y., Stoeckius, M., Smibert, P., and Satija, R. (2019). Comprehensive integration of single-cell data. *Cell*, 177:1888–1902.

Sun, T., Song, D., Li, W. V., and Li, J. J. (2021). scDesign2: a transparent simulator that generates high-fidelity single-cell gene expression count data with gene correlations captured. *Genome biology*, 22(1):1–37.