*Bioinformatics*, 36(6), 2020, 1946–1947
doi: 10.1093/bioinformatics/btz823
Advance Access Publication Date: 6 November 2019
Applications Note



Gene expression

# Improved *dropClust* R package with integrative analysis support for scRNA-seq data

**Debajyoti Sinha[1,2], Pradyumn Sinha[3,†], Ritwik Saha[3,†], Sanghamitra Bandyopadhyay[1,*] and Debarka Sengupta[4,5,6,*]**

[1]SyMeC Data Center, Indian Statistical Institute, Kolkata 700108, India, [2]Department of Computer Science & Engineering, University of Calcutta, Kolkata 700098, India, [3]Department of Computer Science & Engineering, Delhi Technological University, New Delhi 110042, India, [4]Department of Computer Science & Engineering, [5]Department of Computational Biology, Center for Artificial Intelligence, Indraprastha Institute of Information Technology, New Delhi 110020, India and [6]Center for Artificial Intelligence, Indraprastha Institute of Information Technology, New Delhi 110020, India

*To whom correspondence should be addressed.
†The authors wish it to be known that these authors contributed equally.
Associate Editor: Bonnie Berger

Received on April 5, 2019; revised on September 16, 2019; editorial decision on October 25, 2019; accepted on November 2, 2019

## Abstract

**Summary:** DropClust leverages Locality Sensitive Hashing (LSH) to speed up clustering of large scale single cell expression data. Here we present the improved dropClust, a complete R package that is, fast, interoperable and minimally resource intensive. The new dropClust features a novel batch effect removal algorithm that allows integrative analysis of single cell RNA-seq (scRNA-seq) datasets.

**Availability and implementation:** dropClust is freely available at https://github.com/debsin/dropClust as an R package. A lightweight online version of the dropClust is available at https://debsinha.shinyapps.io/dropClust/.

**Contact:** sanghami@isical.ac.in or debarka@iiitd.ac.in

**Supplementary information:** Supplementary data are available at *Bioinformatics* online.

## 1 Introduction

With the advent of single cell transcriptomics, it is now possible to discern phenotypic diversity among seemingly similar cells in complex tissues (Tanay and Regev, 2017). A large pool of cells presents the opportunity of identifying cell types which are previously unseen due to their limited presence or rarity (Jindal *et al.*, 2018). The observed exponentiation of single cell data dimensions demands high performance data analytics capabilities. DropClust R package is one of the speediest clustering algorithms for large scale scRNA-seq data (Sinha *et al.*, 2018). However, the existing implementation requires interfacing with multiple programming languages *viz.* R, python and C. This hinders seamless installation experience on the user end. Also, it's unfit for personal computer use, due to its memory requirements.

Here, we present an upgraded version of the dropClust pipeline (Sinha *et al.*, 2018) with substantially improved execution time and memory efficiency. The current dropClust R package also enables integrative analysis (Haghverdi *et al.*, 2018) of scRNA-seq datasets obtained from independent experiments. The current implementation adopts the widely used *SingleCellExperiment* container R class (Lun *et al.*, 2018) thereby enhancing its interoperability with other tools.

## 2 Improvements and new features

***Selection of Principal Components:*** In the existing version of dropClust, principal components are rank-ordered based on the number of Gaussian mixtures detected from projection of single cells on individual components. We replaced the Gaussian Mixture Model (GMM) by the Gaussian Kernel Density estimator to identify the number of modes in a distribution (more details in Supplementary Information). This improvement led to a 3.5× speed-up over the GMM based approach.

***Integrative analysis:*** A simple, rank based method has been introduced to mitigate batch-effect during clustering. The algorithm takes a merged expression matrix as input. As features, it selects, the union of cluster-specific differentially up-regulated genes obtained from batch-wise dropClust analyses. As part of an additional gene filtering step, some more genes are dropped to ensure that the remaining genes do not display similar expression levels in a large fraction of the cells under study. Cell-wise expression ranks are then used for the final clustering (Supplementary Information).

***Disambiguation of post-hoc cluster assignment:*** When presented with large scale scRNA-seq data, dropClust clusters only a subset of the cells. Each remaining cell is assigned to one of its closest clusters based on the most frequently observed cluster identities, among its

nearest neighbours. Clustering outcomes improved significantly upon the introduction of an acceptance threshold to discard ambiguous cluster assignments. This is illustrated in the Supplementary Material.

***Resolving software dependencies:*** Existing dropClust uses python module `LSHForest` and C module `Louvain` for nearest neighbour search and community detection respectively. We replaced `LSHForest` with `RcppAnnoy`, which implements a significantly faster version of LSH (Andoni *et al.*, 2017). C implementation of Louvain was replaced by its R equivalent, as part of the popular `igraph` R package. DropClust pipeline performs dispersion based gene selection at an early stage. This step uses R function `var` through `apply`, which creates multiple copies of the expression matrix, thereby incurring significant memory footprint. We implemented a custom function called `ColDispersion` in Rcpp that operates on a single copy of the matrix, which reduces both memory usage and execution time.

***Addressing software adaptability:*** To increase modularity and cross package interaction, the latest implementation of dropClust adopts the widely used `SingleCellExperiment` container class (Lun *et al.*, 2018). The container class allows the user to store raw and the normalized single-cell expression profiles along with multiple layers of information (annotations) regarding individual cells and genes, accessible via the `rowData`/`colData` modules. The use of SingleCellExperiment enables seamless interaction with other best practice software utilities.

## 3 Key steps involved in dropClust analysis

***Pre-processing:*** The modules `FilterCells()` and `FilterGenes()` return filtered raw counts that can be accessed using `count()`. Total count normalization is performed using `CountNormalize()`. The normalized count matrix can be accessed using `normcounts()` of the `SingleCellExperiment` class.

***Selecting highly variable genes:*** The normalized expression count values are used to rank the genes by their dispersion scores. The dispersion score associated with each gene can be accessed using `rowData(.)$dispersion_norm`. The top `ngenes_keep` genes are flagged as TRUE in the `rowData(.)$HVG` vector.

***Sampling:*** To expedite clustering dropClust allows the user to perform cell sub-sampling using `Sampling()`. The user can choose SPS or random sampling by setting the value of `use.method` as 'sps' or 'random' respectively. The sampled single cells are marked `TRUE` in the `colData(.)$Sampling` vector.

***Clustering*** The previous dropClust version had hierarchical clustering as the only option for single cell clustering. In the current implementation, `Cluster()` allows the users to choose from three alternative approaches: topological clustering using Louvian (Blondel *et al.*, 2008), agglomerative hierarchical clustering and k-means. For this, user argument `method` can be assigned an appropriate value from 'louvian', 'hclust' and 'kmeans'. `Cluster()` also allows the passing of method-specific arguments. Cluster identities are accessible via `colData(.)$ClusterIDs`. In case of prior cell sub-sampling, the same can be retrieved using `colData(.)$Sample_ClusterIDs`.

***Marker Identification:*** `FindMarkers()` returns cluster specific deferentially up-regulated genes using one-vs-rest strategy. Output of the differential expression (DE) analysis is stored in a list consisting of `DE_up`, a vector of unique DE gene names and `DE_res`, a list of data frames for respective cluster specific DE genes, supplemented with corrected and raw *P*-values along with the log fold change (LFC) information.

***Batch effect removal*** The batch effect removal is carried out in two steps. In the first step, individual expression matrices are merged using the `Merge` module. The union of DE genes retrieved independently on each data are used as feature genes subsequently, for integrative clustering and low dimensional embedding of single cells. `Correction()` function of the dropClust package can be used for this step. `Correction()` facilitates integrative analysis using our newly proposed rank-based integrative analysis strategy, as discussed above. As an alternative method, the user can also use `mnnCorrect` (Haghverdi *et al.*, 2018). `Correction()` can be instructed the same by setting `method` as `fastmnn`. Post-integration low dimensional embeddings and rank-transformed expression matrix are stored in the '`CComponents`' and '`RankMat`' elements respectively in the `reducedDim` list.

## 4 Comparative results

We used a large scRNA-seq dataset containing about 68 000 peripheral blood mononuclear cell (PBMC) transcriptomes (Zheng *et al.*, 2017) to benchmark various methods. Unsupervised clusters obtained using Seurat (Stuart *et al.*, 2019), Scanpy (Wolf *et al.*, 2018) and dropClust offered ARI of 0.33, 0.34 and 0.40, respectively. The current dropClust attained a 4.5× speedup as compared to the original implementation of the software, whereas 1.3× and 26× speedup over Scanpy and Seurat respectively. The latest implementation of the software consumed 8.3 times less memory compared to the original version (Supplementary Table S1 and Fig. S7). We also compared dropClust integrative analysis pipeline with three recently published batch-correction methods on two gold-standard datasets. Visual inspection clearly reveals the superior performance of dropClust (Supplementary Figs S3 and S4).

## Funding

This work was supported by INSPIRE Faculty grant (DST/INSPIRE/04/2015/003068) awarded to Debarka Sengupta, J. C. Bose Fellowship (SB/S1/JCB-033/2016) awarded to Sanghamitra Bandyopadhyay by the Dept. of Sci. and Tech., Govt. of India and the SyMeC Project grant (BT/Med-II/NIBMG/SyMeC/2014/Vol. II), Dept. of BioTech., Govt. of India.

*Conflict of Interest*: none declared.

## References

Andoni,A. *et al.* (2017) LSH forest: practical algorithms made theoretical. In: *Proceedings of the Twenty-Eighth Annual ACM-SIAM Symposium on Discrete Algorithms*. SIAM, pp. 67–78.

Blondel,V.D. *et al.* (2008) Fast unfolding of communities in large networks. *J. Stat. Mech. Theory Exp.*, **2008**, P10008.

Haghverdi,L. *et al.* (2018) Batch effects in single-cell RNA-sequencing data are corrected by matching mutual nearest neighbors. *Nat. Biotechnol.*, **36**, 421.

Jindal,A. *et al.* (2018) Discovery of rare cells from voluminous single cell expression data. *Nat. Commun.*, **9**, 4719.

Lun,A. *et al.* (2018) *SingleCellExperiment: S4 Classes for Single Cell Data. R Package Version*, **1.6.0**.

Sinha,D. *et al.* (2018) dropClust: efficient clustering of ultra-large scRNA-seq data. *Nucleic Acids Res.*, **46**, e36.

Stuart,T. *et al.* (2019) Comprehensive integration of single-cell data. *Cell*, **177**, 1888.

Tanay,A. and Regev,A. (2017) Scaling single-cell genomics from phenomenology to mechanism. *Nature*, **541**, 331.

Wolf,F.A. *et al.* (2018) Scanpy: large-scale single-cell gene expression data analysis. *Genome Biol.*, **19**, 15.

Zheng,G.X. *et al.* (2017) Massively parallel digital transcriptional profiling of single cells. *Nat. Commun.*, **8**, 14049.