SOFTWARE TOOL ARTICLE

SASCRiP: A Python workflow for preprocessing UMI count-based scRNA-seq data [version 1; peer review: 2 approved with reservations]

Darisia Moonsamy, Nikki Gentle

School of Molecular and Cell Biology, University of the Witwatersrand, Johannesburg, South Africa

V1 First published: 15 Feb 2022, 11:190
<https://doi.org/10.12688/f1000research.75243.1>
Latest published: 15 Feb 2022, 11:190
<https://doi.org/10.12688/f1000research.75243.1>

Abstract

In order to reduce the impact of technical variation inherent in single-cell RNA sequencing (scRNA-seq) technologies on biological interpretation of experiments, rigorous preprocessing and quality control is required to transform raw sequencing reads into high-quality, gene and transcript counts. While hundreds of tools have been developed for this purpose, the vast majority of the most widely used tools are built for the R software environment. With an increasing number of new tools now being developed using Python, it is necessary to develop integrative workflows that leverage tools from both platforms. We have therefore developed, SASCRiP (Sequencing Analysis of Single-Cell RNA in Python), a modular single-cell preprocessing workflow that integrates functionality from existing, widely used R and Python packages, and additional custom features and visualizations, to enable preprocessing of scRNA-seq data derived from technologies that use unique molecular identifier (UMI) sequences in a single Python analysis workflow. We describe the utility of SASCRiP using datasets derived from peripheral blood mononuclear cells sequenced using droplet-based, 3'-end sequencing technology. We highlight SASCRiP's diagnostic visualizations and fully customizable functions, and demonstrate how SASCRiP provides a highly flexible, integrative Python workflow for preparing unprocessed UMI count-based scRNA-seq data for subsequent downstream analyses. SASCRiP is freely available through [PyPi](#) or from the [GitHub page](#).

Keywords

single-cell RNA-seq, gene expression, workflow, Python, R

Open Peer Review

Approval Status ? ?

	1	2
version 1		
15 Feb 2022	view	view
1. Fabiola Curion , Technische Universität München, Munich, Germany		
Luke Zappia , Helmholtz Zentrum München, Munich, Germany		
2. Fangming Xie , University of California Los Angeles, California, USA		

Any reports and responses or comments on the article can be found at the end of the article.

This article is included in the **Cell & Molecular Biology** gateway.

This article is included in the **Bioinformatics** gateway.

This article is included in the **Python** collection.

Corresponding author: Nikki Gentle (nikki.gentle@wits.ac.za)

Author roles: **Moonsamy D:** Formal Analysis, Investigation, Methodology, Software, Writing – Original Draft Preparation; **Gentle N:** Conceptualization, Funding Acquisition, Methodology, Project Administration, Supervision, Writing – Review & Editing

Competing interests: No competing interests were disclosed.

Grant information: This work was funded by the National Research Foundation (NRF). NG is funded through the Thuthuka Research Grant (122000) and DM through a NRF Free Standing Masters Bursary (123456).

The funders had no role in study design, data collection and analysis, decision to publish, or preparation of the manuscript.

Copyright: © 2022 Moonsamy D and Gentle N. This is an open access article distributed under the terms of the **Creative Commons Attribution License**, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

How to cite this article: Moonsamy D and Gentle N. **SASCRiP: A Python workflow for preprocessing UMI count-based scRNA-seq data [version 1; peer review: 2 approved with reservations]** F1000Research 2022, 11:190 <https://doi.org/10.12688/f1000research.75243.1>

First published: 15 Feb 2022, 11:190 <https://doi.org/10.12688/f1000research.75243.1>

Introduction

Since a method for single-cell RNA sequencing (scRNA-seq) was first proposed (Tang *et al.*, 2009), a number of diverse technologies have emerged for studying gene expression at single-cell resolution. Among these, droplet-based, 3'-end sequencing technologies such as Drop-seq (Macosko *et al.*, 2015), inDrop (Klein *et al.*, 2015), and in particular, 10X Genomics Chromium (Zheng *et al.*, 2017) have become increasingly popular for studying gene expression across different cell types and cellular states due to their lower sequencing cost per cell, and higher throughput relative to other available technologies. These technologies rely on the use of short unique molecular identifier (UMI) sequences that are used to both quantify gene expression and reduce technical variation inherently present in scRNA-seq datasets (Islam *et al.*, 2014; Kivioja *et al.*, 2011). However, datasets derived using these UMI count-based technologies need to undergo several preprocessing and quality control steps before they can be used to address biological hypotheses.

As a result, hundreds of software tools have been developed to perform these preprocessing and quality control steps, with the vast majority of these software tools having historically been built for the R software environment (Zappia *et al.*, 2018). One of the most popular R packages is undoubtedly Seurat (Stuart *et al.*, 2019), which has become an integral component of many scRNA-seq workflows due to the wide range of analysis methods it offers and the utility of its custom data structures. However recently, an increasing number of new tools are being developed using the Python programming language (Zappia & Theis, 2021). One such tool is kb-python (Bray *et al.*, 2016; Melsted *et al.*, 2021), a wrapper for the kallisto | bustools scRNA-seq workflow (Melsted *et al.*, 2021). Therefore there is a growing need to develop tools that enable integration of widely used existing R-based tools like Seurat and newer Python-based tools and analysis workflows like kb-python.

We therefore introduce SASCRiP (Sequencing Analysis of Single-Cell RNA in Python), a flexible and modular Python package designed to integrate kb-python, selected Seurat features, and additional custom data processing and visualization functions in a way that simplifies and streamlines preprocessing and quality control of UMI count-based scRNA-seq data in preparation for downstream analyses like clustering, data integration, and differential gene expression analysis.

Methods

Implementation

SASCRiP (Moonsamy, 2021a) implements preprocessing of scRNA-seq datasets through four main functions, namely kallisto_bustools_count, seurat_matrix, run_cqc, and sctransform_normalize (Figure 1).

The kallisto_bustools_count function wraps functions from the kb-python package, which itself wraps functions from the kallisto | bustools workflow. This function takes as input unprocessed paired-end FASTQ files derived from UMI-based scRNA-seq experiments. These are then, by default, pseudoaligned to a user-specified indexed transcriptome. The user also has the option to generate a new index and transcript-to-genes mapping file if these are not available. All aligned sequences are stored within a barcode UMI set (BUS) file that contains the barcode and UMI sequences, as well as the aligned transcripts. The kallisto_bustools_count function uses BUStools (Melsted *et al.*, 2019) to remove polymerase chain reaction (PCR) duplicates and correct the barcode sequences that differ by 1 hamming distance from a barcode whitelist. The kallisto_bustools_count function includes barcode whitelists for 10x Genomics Chromium v1, v2 and v3, as well as for InDrops v3 data. For other UMI-based technologies, kallisto_bustools_count includes parameters that allow for the use of user-supplied barcode whitelists. By default, the BUS file is filtered to remove barcodes with no corresponding transcript information. Gene-level count matrices are then returned in the matrix market (mtx) format, where cells and genes are represented in rows and columns, respectively.

```
import sascrip
from sascrip import sascrip_functions

sascrip_functions.kallisto_bustools_count(
    list_of_fastq_files, # list of all the input FASTQ files in the required order
    single_cell_technology, # type of single-cell tech used to generate the data
    output_directory_path, # path where all output files will be saved
    species_index, # path to the kallisto index for the species of interest
    species_t2g # path to the transcripts-to-genes mapping file for the species of interest
)
```

Figure 1. Overview of the Sequencing Analysis of Single-Cell RNA in Python (SASCRiP) workflow. The SASCRiP workflow begins with the input of FASTQ files, and a Seurat object and mtx matrix containing gene expression values are produced as output. All SASCRiP functions, namely kallisto_bustools_count, seurat_matrix, run_cqc, and sctransform_normalize, produce intermediate files that are used as input into the following function.

SASCRiP also includes an additional, optional function, `edit_10xv1_fastq`, that transforms scRNA-seq data obtained from experiments using 10x Genomics Chromium v1 technology into a format compatible with the kallisto | bustools workflow. Although these types of files are already supported by the kallisto | bustools workflow, most FASTQ files stored in public sequencing data repositories/databases are not provided in the required format. In situations where only two FASTQ files (one containing the transcript and UMI sequences, and the other containing the barcode sequences) are provided, the `edit_10xv1_fastq` function separates the UMI and transcript sequences into two new FastQ files, so that the three files required (containing the transcript, UMI, and barcode sequences) are available as input for the kallisto | bustools workflow.

```
import sascrip
from sascrip import sascrip_functions

sascrip_functions.edit_10xv1_fastq(
    directory_with_fastqs, # path to directory containing 10xv1 FASTQ files
    output_directory # path where edited FASTQ files will be stored
)
```

Count matrices obtained using the kallisto | bustools workflow cannot be directly imported into Seurat. Therefore the SASCRiP function, `seurat_matrix`, converts the output obtained from BUSTools (where genes are

represented as rows and cells as columns) into a format compatible with Seurat, where cells are represented as rows and genes as columns. The `seurat_matrix` function also converts gene identifiers in the transcript-to-gene index file from the kallisto | bustools workflow from ENSEMBL IDs to their corresponding HGNC gene symbols, as required by Seurat.

```
import sascrip
from sascrip import sascrip_functions

sascrip_functions.seurat_matrix(
    mtx_bustools_matrix, # mtx matrix generated by kallisto_bustools_count
    bustools_gene_index, # genes index file generated with the mtx matrix
    bustools_barcode_index # barcode index file generated with the mtx matrix
    transcript_to_genes_file # path to transcript-to-genes files used/generated
                           # with kallisto_bustools_count
    output_directory # path where Seurat-compatible files will be saved
)
```

Gene-level count matrices obtained from either Seurat or SASCRiP's `seurat_matrix` function can then be used as input for SASCRiP's `run_cqc` function, which leverages Seurat's custom data structures to identify low quality cells. The `run_cqc` function classifies cells as being of low quality based on two per-cell metrics: the total number of genes detected, and the percentage of sequencing reads that map to mitochondrial genes. These metrics are calculated automatically when a Seurat object is created. By default, `run_cqc` first removes cells with fewer than 200 genes (Ilicic *et al.*, 2016), then cells where more than 10% of sequencing reads map to mitochondrial genes Osorio & Cai (2021). However, all parameters used to distinguish low quality cells from high quality cells can be defined by the user.

The `run_cqc` function also calculates the median absolute deviation (MAD) of the total number of genes per cell, allowing for the identification and removal of cell doublets. SASCRiP prioritises the removal of heterotypic doublets, such that, by default, cells with a total gene count higher than six MAD values from the median are classified as outliers and removed. The MAD values and outlier thresholds are calculated using the following equation:

$$\tilde{x} = \text{median}(x)$$

$$\text{MAD} = \text{median}(|x_i - \tilde{x}|) + 1.483$$

$$n\text{MAD} = n * (\text{MAD})$$

$$\text{flag as outlier if : total gene count} > \tilde{x} + n\text{MAD}$$

Here, x represents the total gene count for all cells, x_i is the gene count for a given cell, and n is the number of MAD values used to classify a cell as an outlier. The MAD value cut-off threshold can be manually defined by the user. Alternatively, other outlier detection methods, such as standard deviation, are also included within SASCRiP and can be selected as a substitute to MAD.

```
import sascrip
from sascrip import sascrip_functions

sascrip_functions.run_cqc(
    input_files, # directory containing Seurat-compatible files
    "sample", # user-given name of sample
    output_directory # path where all output files will be saved
)
```

Multiple output files are produced by the `run_cqc` function. Cell quality control metrics and UMI count data are returned as Seurat objects, in rds format, and a log file is created. The cell quality control metrics can also be returned in tsv format, to facilitate data visualization; and UMI counts can be returned as an mtx matrix, for use with alternative analysis tools. In this way, output from SASCRiP can be easily integrated into alternative workflows and analysis pipelines that require different file formats.

SASCRiP performs normalization and variance stabilization through the `sctransform_normalize` function, which serves as a wrapper for `sctransform` (Hafemeister & Satija, 2019). In this way, technical variation present in the dataset due to differences in sequencing depth between cells is reduced. All parameters incorporated within the original `sctransform::vst` function can be modified through the `sctransform_normalize` function. Corrected UMI counts are log-normalized in order to obtain gene expression values and the 2000 most variable genes in the dataset are identified and returned. Normalized gene expression values or UMI counts may be returned, either as a Seurat object or as an mtx file.

```
import sascrip
from sascrip import sascrip_functions

sascrip_functions.sctransform_normalize(
    seurat_object, # path to saved subset Seurat object
    "sample", # user-given name of sample
    output_directory # path where normalised data are saved
)
```

The modularity of SASCRiP allows all functions to be implemented independently, but the package also includes an all-in-one function, `sascrip_preprocess`, that implements the entire workflow from start to finish. In order to ensure that appropriate quality control measures are applied at each stage of the workflow, quality control metrics are provided at multiple checkpoints in the SASCRiP workflow when the `sascrip_preprocess` function is executed. These metrics can be printed to the screen while running the workflow and/or written to file for later use.

Operation

All SASCRiP functions are designed and implemented in Python 3 (v3.7 or higher), and the package is available on [PyPi](#). As SASCRiP also incorporates a number of R packages (including Seurat), R (v3.6 or higher) is also required. Any additional R packages required can then be installed through SASCRiP, if needed. SASCRiP was developed primarily for use on Unix-based operating systems, however the source code can be adapted for use on Windows platforms (as described in the SASCRiP documentation). SASCRiP can be used to process scRNA-seq datasets consisting of up to 10 000 cells on a standard laptop with 8 Gb RAM. At least 16 Gb of RAM is recommended for larger datasets.

Use cases

In order to demonstrate the utility of the SASCRiP workflow, we reanalyzed three scRNA-seq datasets derived from peripheral blood mononuclear cells (PBMCs) obtained from healthy individuals (Zheng *et al.*, 2017). Cell counts range from 2000 to 10 000 cells per donor. SASCRiP's `sascrip_preprocess` function was used to perform all preprocessing steps required to generate a high-quality gene expression dataset that could then be readily input into Seurat for cell clustering and annotation. All SASCRiP functions were implemented using the default parameters.

As these datasets were obtained using 10X Genomics Chromium v1 technology, SASCRiP's `edit_10xv1_fastq` function was first used to generate the three FASTQ files required by the `kallisto_bustools_count` function. Gene-level count matrices were then obtained using the `kallisto_bustools_count` function, and cell quality control metrics were calculated using the `run_cqc` function (Moonsamy, 2021b). Visualization of these metrics allowed low quality cells (Figure 2) and cell doublets (Figure 3) to be easily identified and removed. These metrics, together with the UMI count data, were then input as Seurat objects, into the `sctransform_normalize` function. This allowed for normalization and variance stabilization of these UMI counts (Figure 4), so that gene expression could be quantified. In addition, the 2000 most highly variable genes, based on standardized variance, were identified (Figure 5). The normalization data was then output as a Seurat object, in preparation for clustering.

Finally, to demonstrate the quality of the preprocessed data obtained from SASCRiP, Seurat v3.2.0 was used to cluster the two datasets identified as being of high quality (donors B and C) following processing with SASCRiP. The `RunPCA` function was used to perform dimension reduction (Stuart *et al.*, 2019), and graph-based clustering was performed using the `FindNeighbours` function (Xu & Su, 2015). In both datasets, the preprocessed data obtained from SASCRiP produced distinct clusters (Figure 6A), corresponding to major PBMC cell types (Figure 6B), suggesting that SASCRiP produces high quality, preprocessed data, suitable for downstream scRNA-seq analyses and applications.

Figure 2. Sequencing Analysis of Single-Cell RNA in Python allows for calculation and visualization of cell quality control metrics, including the total number of genes detected per cell, and the total number of sequencing reads that align to mitochondrial genes per cell. Cells that fall (A) below the given threshold for gene count (in this case, 200) and (B) above the maximum threshold for mitochondrial percentage (in this case, 10%) are shown in red.

Figure 3. Sequencing Analysis of Single-Cell RNA in Python allows for the detection and visualization of outliers using median absolute deviation (MAD). Potential cell doublets (black dots) are identified relative to the median total gene count per cell. Cells with a total gene count more than six MADs from the median are flagged for removal.

Figure 4. Sequencing Analysis of Single-Cell RNA in Python allows for normalization and variance stabilization of UMI-based count data. The total number of UMIs relative to the total number of unique genes detected is visualized before (blue) and after (pink) normalization and variance stabilization using regularized negative binomial regression, as implemented in `scran` (Hafemeister & Satija, 2019). Each point represents a single cell.

Figure 5. Sequencing Analysis of Single-Cell RNA in Python identifies the top 2000 most highly variable genes. The top 2000 most highly variable genes (purple) are identified based on standardized variance. Each dot represents a gene with its standardized variance shown on the y-axis and the average gene expression (calculated using the normalized gene counts) on the x-axis. The 10 genes with the greatest variance across all cells are labelled.

Figure 6. Data preprocessed using Sequencing Analysis of Single-Cell RNA in Python can be used to identify clusters corresponding to distinct cell types in peripheral blood mononuclear cells. Uniform manifold approximation and projection of ~8000 cells derived from a single donor is shown, coloured according to (A) the clusters identified in the dataset, and (B) the expression of known cell type markers in each cluster. *CD3D* indicates T cell clusters, *MS4A1* B cell clusters, *CD68* monocyte clusters, and *FCER1A* dendritic cell clusters.

Conclusions

SASCRiP is a Python package that provides a simple, streamlined workflow for preprocessing UMI count-based scRNA-seq data through a series of parameterized functions. To ensure flexibility, these functions allow users to either adopt a set of clearly defined default parameters or to modify any or all of these parameters as they see fit. Also, due to its modular design, SASCRiP's four major functions (`kallisto_bustools_count`, `seurat_matrix`,

cell_cqc, sctransform_normalize) can be executed either independently to produce custom visualizations and/or intermediary files that can be used as input for other scRNA-seq tools, or all-in-one to produce high-quality, normalized gene expression datasets that are ready to use for downstream analyses. Collectively, these functions seamlessly integrate the functionality of the widely used R packages, Seurat and sctransform, into a custom Python workflow built around kb-python's implementation of the kallisto | bustools workflow. In this way, pseudoalignment, quantification of gene expression, removal of low quality cells and cell doublets, and normalization and variance stabilization can be performed within a single scRNA-seq analysis pipeline.

Data availability

Underlying data

All data underlying the results are freely available through 10X Genomics:

Donor A:

https://cf.10xgenomics.com/samples/cell-exp/1.1.0/frozen_pbmc_donor_a/frozen_pbmc_donor_a_fastqs.tar

Donor B:

https://cf.10xgenomics.com/samples/cell-exp/1.1.0/frozen_pbmc_donor_b/frozen_pbmc_donor_b_fastqs.tar

Donor C:

https://cf.10xgenomics.com/samples/cell-exp/1.1.0/frozen_pbmc_donor_c/frozen_pbmc_donor_c_fastqs.tar

Extended data

Zenodo: SASCRiP Supporting Data <https://doi.org/10.5281/zenodo.5899870> (Moonsamy, 2021b)

Data are available under the terms of the [Creative Commons Attribution 4.0 International license](#) (CC-BY 4.0).

Software availability

Software available from: <https://pypi.org/project/SASCRiP>

Source code available from: <https://github.com/Darisia/SASCRiP>

Archived source code at time of publication: <https://doi.org/10.5281/zenodo.5554770> (Moonsamy, 2021a)

License: [GNU GPLv3](#)

References

- Bray NL, Pimentel H, Melsted P, et al.: **Near-optimal probabilistic RNA-seq quantification.** *Nat Biotechnol.* 2016; **34**(5): 525–527.
[PubMed Abstract](#) | [Publisher Full Text](#)
- Hafemeister C, Satija R: **Normalization and variance stabilization of single-cell RNA-seq data using regularized negative binomial regression.** *Genome Biol.* 2019; **20**(1): 296.
[PubMed Abstract](#) | [Publisher Full Text](#) | [Free Full Text](#)
- Ilicic T, Kim JK, Kolodziejczyk AA, et al.: **Classification of low quality cells from single-cell RNA-seq data.** *Genome Biol.* 2016; **17**: 29.
[PubMed Abstract](#) | [Publisher Full Text](#) | [Free Full Text](#)
- Islam S, Zeisel A, Joost S, et al.: **Quantitative single-cell RNA-seq with unique molecular identifiers.** *Nat Methods.* 2014; **11**(2): 163–166.
[PubMed Abstract](#) | [Publisher Full Text](#)
- Kivioja T, Vähäurautio A, Karlsson K, et al.: **Counting absolute numbers of molecules using unique molecular identifiers.** *Nat Methods.* 2011; **9**(1): 72–4.
[PubMed Abstract](#) | [Publisher Full Text](#)
- Klein AM, Mazutis L, Akartuna I, et al.: **Droplet barcoding for single-cell transcriptomics applied to embryonic stem cells.** *Cell.* 2015; **161**(5): 1187–1201.
[PubMed Abstract](#) | [Publisher Full Text](#) | [Free Full Text](#)
- Macosko EZ, Basu A, Satija R, et al.: **Highly parallel genome-wide expression profiling of individual cells using nanoliter droplets.** *Cell.* 2015; **161**(5): 1202–1214.
[PubMed Abstract](#) | [Publisher Full Text](#) | [Free Full Text](#)
- Melsted P, Boeshaghi AS, Liu L, et al.: **Modular, efficient and constant-memory single-cell RNA-seq preprocessing.** *Nat Biotechnol.* 2021; **39**(7): 813–818.
[PubMed Abstract](#) | [Publisher Full Text](#)
- Melsted P, Ntranos V, Pachter L: **The barcode, UMI, set format and BUSTools.** *Bioinformatics.* 2019; **35**(21): 4472–4473.
[PubMed Abstract](#) | [Publisher Full Text](#)
- Moonsamy D: **SASCRiP (0.1.2).** *Zenodo.* 2021a.
<http://www.doi.org/10.5281/zenodo.5554770>
- Moonsamy D: **SASCRiP Supporting Data (0.1.2) [Data set].** *Zenodo.* 2021b.
<http://www.doi.org/10.5281/zenodo.5899870>
- Osorio D, Cai JJ: **Systematic determination of the mitochondrial**

proportion in human and mice tissues for single-cell RNA-sequencing data quality control. *Bioinformatics*. 2021; **37**(7): 963–967.
[PubMed Abstract](#) | [Publisher Full Text](#) | [Free Full Text](#)
 Stuart T, Butler A, Hoffman P, *et al.*: **Comprehensive Integration of Single-Cell Data**. *Cell*. 2019; **177**(7): 1888–1902.e21.
[PubMed Abstract](#) | [Publisher Full Text](#) | [Free Full Text](#)
 Tang F, Barbacioru C, Wang Y, *et al.*: **mRNA-Seq whole-transcriptome analysis of a single cell**. *Nat Methods*. 2009; **6**(5): 377–382.
[PubMed Abstract](#) | [Publisher Full Text](#)
 Xu C, Su Z: **Identification of cell types from single-cell transcriptomes using a novel clustering method**.

Bioinformatics. 2015; **31**(12): 1974–1980.
[PubMed Abstract](#) | [Publisher Full Text](#) | [Free Full Text](#)
 Zappia L, Phipson B, Oshlack A: **Exploring the single-cell RNA-seq analysis landscape with the scRNA-tools database**. *PLoS Comput Biol*. 2018; **14**(6): e1006245.
[PubMed Abstract](#) | [Publisher Full Text](#) | [Free Full Text](#)
 Zappia L, Theis FJ: **Over 1000 tools reveal trends in the single-cell RNA-seq analysis landscape**. *Genome Biol*. 2021; **22**(1): 301.
[PubMed Abstract](#) | [Publisher Full Text](#) | [Free Full Text](#)
 Zheng GX, Terry JM, Belgrader P, *et al.*: **Massively parallel digital transcriptional profiling of single cells**. *Nat Commun*. 2017; **8**: 14049.
[PubMed Abstract](#) | [Publisher Full Text](#) | [Free Full Text](#)

Open Peer Review

Current Peer Review Status: ? ?

Version 1

Reviewer Report 14 March 2022

<https://doi.org/10.5256/f1000research.79090.r125623>

© 2022 Xie F. This is an open access peer review report distributed under the terms of the [Creative Commons Attribution License](#), which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

Fangming Xie

University of California Los Angeles, California, CA, USA

Moonsamy and Gentle presented Sequencing Analysis of Single-Cell RNA in Python (SASCRIp), a modular analysis pipeline that processes UMI-based scRNA-seq data from raw sequencing reads to transcript counts, normalization, and visualizations. The software integrated existing tools, including kallisto-bustools (kb python as its Python wrapper) and Seurat (an R package), into a single Python workflow. The authors showcased the usage of their pipeline using public PBMC scRNA-seq datasets.

Strengths

Overall, the manuscript was well written and has made its main message clear. The authors did a good job presenting their workflow, functionalities, and results. As a technical manuscript, I liked the code vignettes displayed throughout the main text. Its Github repo also contains a detailed README document.

Major limitations

- The software is mainly a Python wrapper around two existing tools: kb-python (in Python) and Seurat (in R). As kb-python is already a Python wrapper of Kallisto, it is not clear to me what this new wrapper has to offer more than the existing one.
- The authors should comment on native Python scRNA-seq analysis packages, such as the widely used Scanpy, and highlight the similarities and differences of their package.
- While the README file is useful, it would be more helpful if the authors can also include a tutorial or an example script that runs through the workflow with a small chunk of data, as Seurat did.

Minor comments

- In Page 5, the authors used MAD to flag potential doublets. The equations for MAD need a bit more explanation/justification. For example, why adding 1.483? What is the unit of x? Is there a citation for this method?
- In Figure 2, it would be helpful to include the number or the fraction of cells in each

color/category, in addition to the scatter plots. (In Figure 2A (left), it looks like most cells were low-quality, at least that's the first impression.)

- In Figure 3, donor A looks very different from the other donors. What was the reason?

Is the rationale for developing the new software tool clearly explained?

Partly

Is the description of the software tool technically sound?

Yes

Are sufficient details of the code, methods and analysis (if applicable) provided to allow replication of the software development and its use by others?

Partly

Is sufficient information provided to allow interpretation of the expected output datasets and any results generated using the tool?

Yes

Are the conclusions about the tool and its performance adequately supported by the findings presented in the article?

Yes

Competing Interests: No competing interests were disclosed.

Reviewer Expertise: Bioinformatics, Neuroscience, Computational analysis of single-cell transcriptomics and epigenomics data

I confirm that I have read this submission and believe that I have an appropriate level of expertise to confirm that it is of an acceptable scientific standard, however I have significant reservations, as outlined above.

Reviewer Report 22 February 2022

<https://doi.org/10.5256/f1000research.79090.r123750>

© 2022 Zappia L et al. This is an open access peer review report distributed under the terms of the [Creative Commons Attribution License](#), which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

Fabiola Curion

Helmholtz Munich, Technische Universität München, Munich, Germany

Luke Zappia

Helmholtz Munich, Helmholtz Zentrum München, Munich, Germany

The authors have presented a workflow/package to deal with common steps of the pre-processing of single-cell RNA-seq data from raw reads until the normalization step, producing data that can be readily used by downstream tools. Whilst we appreciate it is useful to have a Python interface for some of these steps, it is not clear what is the advantage of using a further wrapper around the existing Python implementation of Kallisto | Bustools. The second half of the package is a wrapper around the commonly used R tool Seurat but it is not clearly demonstrated how useful it is to wrap Seurat instead of using the equivalent functionality in existing Python packages such as Scanpy. We believe there are some useful additions, such as the conversion of FASTQ files from old chemistry versions to new ones that can be used by Bustools and the transposition of the matrix output from Kallisto | Bustools into the default Seurat structure (gene x cell), but these could be better emphasised in the text.

Specific comments

- The wrapper around kallisto is not well justified. If the selling point of this solution is to deal with v1 10x chemistry that could be better emphasised.
- The technical description of read sequence structure for 10x and similar droplet-based methods is somewhat lacking clarity. Please provide one or two sentences describing the bioinformatics processing needed to deal with read structures for different chemistries.
- The conversion of Bustools output to Seurat is a good idea. However, this is described twice in the manuscript which is confusing.
- We suggest highlighting that there is a further entry point into the workflow (count matrix as opposed to FASTQ files)
- You suggest a metric for detecting doublets based on outliers in the number of expressed genes but there is no analysis to show how this effectively discriminates known doublets from singlets. We would suggest applying it to one of the datasets used to demonstrate other doublet-detection methods.
- It is not clear if the workflow includes plots as default outputs at any given checkpoint. If this is the case please describe that in the text.
- We found Figure 2 to be quite unclear. Are the left and right columns two different datasets? Why are the scales in panel A different but the same scale is used in panel B? Better labeling of the plots and more detail in the caption would help clarify these issues.
- We could not find code for the analysis presented in the paper. Please include this in a publicly available repository.

Software

- We were able to install the software using the provided instructions
- We were not able to test the software because no example dataset was provided. We suggest writing a short vignette/tutorial showing how the different parts of the package can be used on a small test dataset.
- There is good function documentation in the GitHub README but we suggest moving this to

another format (such as a [Read The Docs](#) page) which would be more accessible for users.

Is the rationale for developing the new software tool clearly explained?

Partly

Is the description of the software tool technically sound?

Partly

Are sufficient details of the code, methods and analysis (if applicable) provided to allow replication of the software development and its use by others?

Partly

Is sufficient information provided to allow interpretation of the expected output datasets and any results generated using the tool?

No

Are the conclusions about the tool and its performance adequately supported by the findings presented in the article?

No

Competing Interests: No competing interests were disclosed.

Reviewer Expertise: Single cell genomics, bioinformatics

We confirm that we have read this submission and believe that we have an appropriate level of expertise to confirm that it is of an acceptable scientific standard, however we have significant reservations, as outlined above.

The benefits of publishing with F1000Research:

- Your article is published within days, with no editorial bias
- You can publish traditional articles, null/negative results, case reports, data notes and more
- The peer review process is transparent and collaborative
- Your article is indexed in PubMed after passing peer review
- Dedicated customer support at every stage

For pre-submission enquiries, contact research@f1000.com

F1000Research