NAR Genomics and Bioinformatics, 2023, **5**, 1–7
https://doi.org/10.1093/nargab/lqad105
Advance access publication date: 30 November 2023
Application Notes

scPipe: an extended preprocessing pipeline for comprehensive single-cell ATAC-Seq data integration in R/Bioconductor

Shanika L. Amarasinghe[1,2], Phil Yang[1], Oliver Voogd[1], Haoyu Yang[1], Mei R.M. Du[1], Shian Su[1], Daniel V. Brown[1,3], Jafar S. Jabbari[1,3], Rory Bowden[1,3] and Matthew E. Ritchie[1,3,*]

[1]The Walter and Eliza Hall Institute of Medical Research, Parkville, Victoria, 3052, Australia
[2]The Australian Regenerative Medicine Institute, Monash University, Clayton, Victoria, 3800, Australia
[3]Department of Medical Biology, The University of Melbourne, Parkville, Victoria, 3010, Australia
[*]To whom correspondence should be addressed. Tel: +61 3 9345 2856; Fax: +61 3 9347 0852; Email: mritchie@wehi.edu.au

Abstract

`scPipe` is a flexible R/Bioconductor package originally developed to analyse platform-independent single-cell RNA-Seq data. To expand its preprocessing capability to accommodate new single-cell technologies, we further developed `scPipe` to handle single-cell ATAC-Seq and multimodal (RNA-Seq and ATAC-Seq) data. After executing multiple data cleaning steps to remove duplicated reads, low abundance features and cells of poor quality, a *SingleCellExperiment* object is created that contains a sparse count matrix with features of interest in the rows and cells in the columns. Quality control information (e.g. counts per cell, features per cell, total number of fragments, fraction of fragments per peak) and any relevant feature annotations are stored as metadata. We demonstrate that `scPipe` can efficiently identify 'true' cells and provides flexibility for the user to fine-tune the quality control thresholds using various feature and cell-based metrics collected during data preprocessing. Researchers can then take advantage of various downstream single-cell tools available in Bioconductor for further analysis of scATAC-Seq data such as dimensionality reduction, clustering, motif enrichment, differential accessibility and cis-regulatory network analysis. The `scPipe` package enables a complete beginning-to-end pipeline for single-cell ATAC-Seq and RNA-Seq data analysis in R.

Introduction

Single-cell sequencing technology has undergone rapid development in the past decade to allow researchers to study cellular heterogeneity across multiple omic modalities, including the transcriptome, epigenome and proteome. Single Cell Assay for Transposase Accessible Chromatin Sequencing (scATAC-Seq), is a relatively recent approach for profiling chromatin accessibility at single-cell resolution (1) that has been widely used to define chromatin state across cell types, discover cis- and trans- regulatory regions, identify master regulators, and characterise gene regulatory networks (2).

The four main protocols for scATAC-Seq include the combinatorial indexing approach (sci-ATAC-Seq) (3), microfluidics-based methods (scATAC-Seq) (1), nano-well based protocols (μscATAC-Seq) (4) and droplet-based (10X scATAC-Seq, dscATAC-Seq and dsciATAC-seq) approaches (5,6).

The general workflow for scATAC-Seq data analysis comprises of (i) preprocessing: demultiplexing, adaptor trimming, read mapping, quality control, cell calling and multiplet removal *(optional)*; (ii) feature matrix construction: defining regions via peak calling or genome binning, counting defined features, transformation and dimensionality reduction and (iii) downstream analysis: cell clustering, secondary peak calling, visualisation, differential accessibility analysis and *cis*-regulatory network analysis (7,8).

The growing popularity of scATAC-Seq technology warrants the development of analysis tools that can preprocess this type of data efficiently and effectively. There are currently 14 tools that can preprocess scATAC-Seq data that are summarised in Table 1. Some are written in R (i.e. `ArchR`, `BROCKMAN`, `ChromSCape`, `chromVAR`, `Destin`, `scABC`, `SCRAT`, `Snaptools`) while others are Python and Shell based. Furthermore, the feature matrix construction step varies amongst these tools where some of them use bulk peak calls, TSS regions, *k*-mers, or a genome binning approach. It has been shown previously that out of these, a genome binning approach is more sensitive to the detection of rare open chromatin regions present in a sub-population of cells (9).

A current limitation of these tools is that most cannot handle data from multiple technologies (i.e. droplet-based and plate-based) or they require the use of specific data structures that have limited compatibility with current well-known, light-weight, single-cell data structures such as *SingleCellExperiment* objects (*SCEs*) in R/Bioconductor.

In this study, we extend `scPipe` (10) to enable handling of scATAC-Seq data and generation of *SCE* objects that can be manipulated using various *SCE*-friendly downstream analysis tools available in R/Bioconductor (11). An *SCE* container, most popular for scRNA-Seq data storage, can easily be used to store scATAC-Seq data due to its flexibility (i.e., rows representing features (genomic regions) and columns representing cells). `scPipe` is able to take scATAC-Seq reads (*FASTQ* format) as input, which are demultiplexed (if needed) and filtered based on quality and Ns (i.e. non-called bases), aligned to the reference genome and filtered based on various quality

Received: September 18, 2023. **Editorial Decision:** November 15, 2023. **Accepted:** November 23, 2023

Table 1. Summary of current packages available for scATAC-Seq data preprocessing

Method / tool name	Year	Language	Input	Multiple Technologies?	Feature matrix Construction method
APEC	2019	Python	*FASTQ*, matrix + peaks	✓	Peak
ArchR	2020	R	*BAM*, fragments	✓	Bin
BROCKMAN	2018	R	*FASTQ*	×	*k*-mer
Cell Ranger ATAC	2018	Command line	*FASTQ*	×	Peak
ChromSCape	2019	R Shiny	Various	×	Bin, Peak, TSS
chromVAR	2017	R	*BAM* + peaks	×	Motif, *k*-mer
Cusanovich *et al*. 2018	2015	Scripts	*BAM*	×	Peak
Destin	2019	R	*FASTQ*, *BAM* + peaks	✓	Peak
Gene scoring	2019	Scripts	*FASTQ*	×	TSS
scABC	2018	R	*FASTQ*	×	Peak
scasat	2018	JupyterNotebook	*BAM*s	×	Peak
scATAC-pro	2019	Command line	*FASTQ*, various	✓	Peak
SCRAT	2017	R,WEB	*BAM*s	×	Various
Snaptool	2019	Python	*FASTQ*	✓	Bin, Peak

metrics such as mapping rate, fraction of reads mapping to the mitochondrial genes and the number of duplicate or high-quality reads. Cleaned up data will then be used to conduct cell calling and optionally peak calling, to generate the final feature by cell sparse matrix which is stored as a *SCE* object. These improvements also allow scPipe to be used for multiome projects that collect both scRNA-Seq and scATAC-Seq on the same cells.

Materials and methods

Architecture of the scPipe scATAC-Seq module

scPipe was developed using the R (12) / Bioconductor (13) platform, with the underlying code written in C++, R and Python with the Rcpp (14) and reticulate (15) packages used to wrap the C++ and Python code for R, respectively. Data from both UMI (Unique Molecular Identifiers) and non-UMI protocols can be handled by scPipe. The pipeline for scATAC-Seq data preprocessing is initiated with *FASTQ* files and outputs include a feature count matrix and a variety of quality control (QC) statistics and a standalone HTML report generated using rmarkdown (16) that contains a summary of the QC statistics collected during data preprocessing.

Demultiplexing and read alignment

Demultiplexing is performed by the *sc_atac_trim_barcode()* function (Figure 1A) using a similar approach to that used in scPipe for scRNA-Seq analysis. The scATAC-Seq module can accommodate data in *.fastq* format as well as *.csv* format, with the user defining which format the data is in and the relevant demultiplexing strategy will be executed. In brief, the barcode is either extracted from the reads themselves based on the entries in a *csv* file where the second column of the file contains the barcodes or from the sequences of a complementary *FASTQ* file and appended to the read names in the *FASTQ* files containing the ATAC sequences.

A read correction step is incorporated to ensure sequencing errors that appear in barcodes are identified and removed to avoid unwanted data loss. In summary, after the barcodes are determined, error correction is carried out with a hamming distance of 1 to correct for sequencing errors. Each barcode can be optionally validated against a known list of valid barcodes (either as is or the reverse complement) and the frequency of each valid barcode is then counted. scPipe corrects barcodes that are not matched to the valid barcode list by allowing for one mismatched base in comparison with the valid barcode list, and counting the number of corrected barcodes in the complete dataset. In the case that multiple sequences are found matching the valid barcode list with one mismatched base, each mismatched base's quality (Q) score is compared and the sequence with the lowest Q score at the mismatch position is considered valid.

Demultiplexed data is stored in four data files; i.e. complete-match: no error correction was necessary for the data in this file; partial-match: error-corrected reads; no-match: data that were unable to be demultiplexed even after error correction; and full-data: a concatenated version of the data across the 3 aforementioned categories. The user is able to select either one of the more stringent files or the complete dataset for downstream analysis (by default, the complete dataset is used).

Read alignment is run with the function *sc_aligning()* which uses *Rsubread::align()* (17) internally (Figure 1A). The user has to define the technology (i.e. RNA or ATAC) to execute the most appropriate alignment approach (i.e. single-end alignment for RNA and paired-end for ATAC). The resulting *BAM* file contains the read name in the first column, and unmapped reads are denoted by an asterisk (*).

Demultiplexing aligned reads, removing duplicates, creating fragments and peak calling

Demultiplexing aligned reads is performed by the *sc_atac_bam_tagging()* function which extracts the barcode information from the read names of the aligned *BAM* file and generates a new column with the tag CB:z: to record the cell barcode (Figure 1A). As some reads may not have an assigned barcode, this column in the resulting demultiplexed *BAM* file may be empty in such positions.

The main difference in the scATAC-Seq module compared to original scRNA-Seq module is the need to extract biological information from both reads in the former. Therefore, it was important to retain both reads (i.e. forward and reverse) for scATAC-Seq data as compared to retaining only the forward of the pair of reads for scRNA-Seq module where reverse will be the barcode followed by the polyA tail, which can be discarded after demultiplexing of the read takes place.

Removal of duplicate reads uses samtools, where the *BAM* file is processed with the *removeduplicates* function (18).

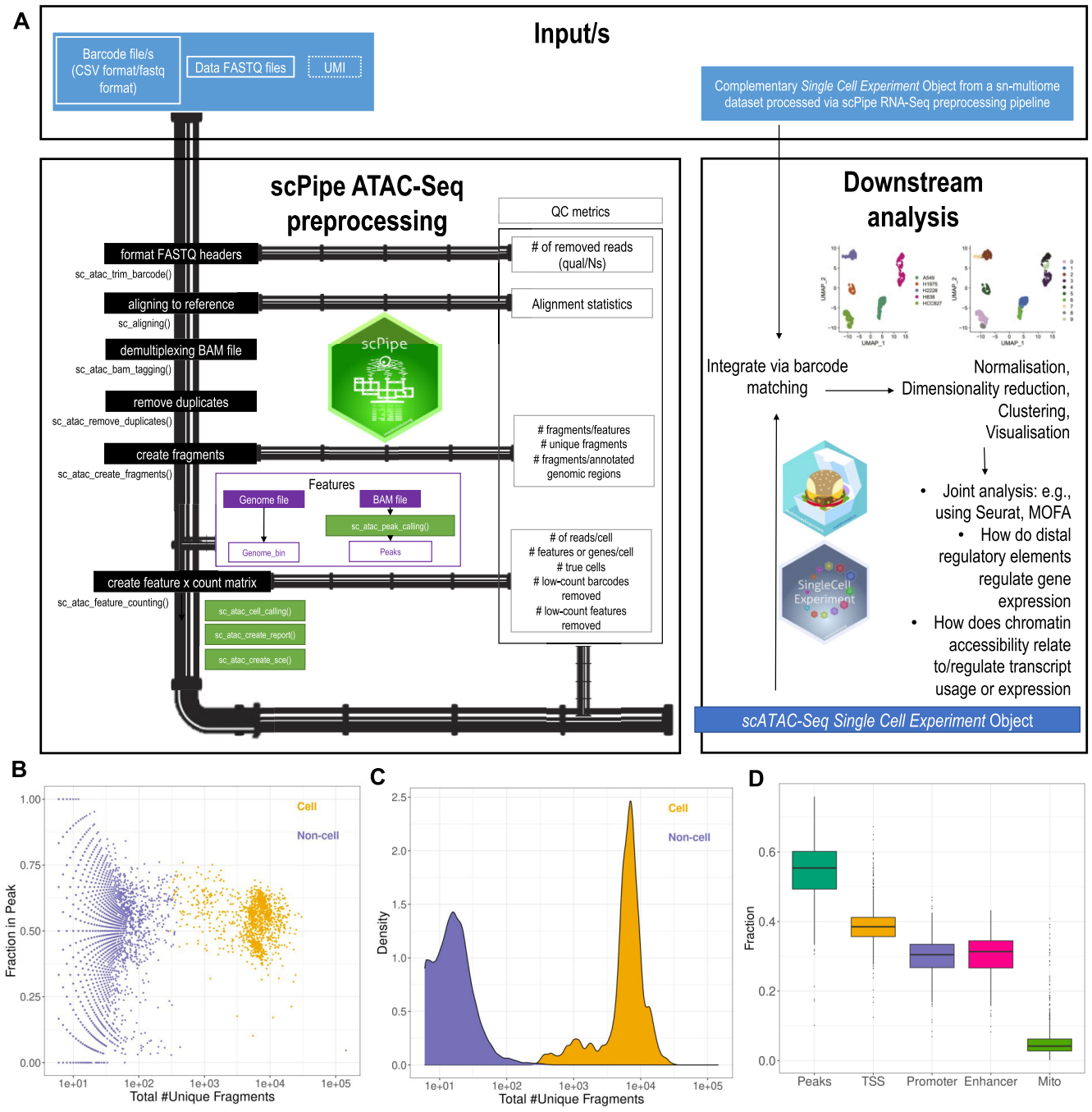

Figure 1. Overview of the `scPipe` scATAC-Seq module and its QC outputs. **(A)** The pipeline is shown on the left and the QC metrics gathered during preprocessing are shown on the right. Purple coloured boxes denote the inputs and light colour purple define the inputs that are optionally accepted as they are not incorporated into current scATAC-Seq library preparations yet. Blue colour box depicts the final output. Black boxes denote the main pipeline steps that should be followed. Green boxes denote the steps that are running within these the main pipeline without having to call them specifically, still can also be called separately if needed. **(B)** QC plot showing the separation of 'cell' and 'non-cell' based on the fraction of fragments overlappping peaks (y-axis) vs total number of fragments (x-axis) after cell calling step of scPipe. **(C)** QC plot showing the separation of cell and 'non-cell' based on read density (y-axis) versus total number of fragments (x-axis) after cell calling step of scPipe. **(D)** QC plot showing the fraction of features overlapping different functional regions (i.e., Peaks, TSS, Promoter, Enhancer, Mitochondrial genes).

This step can be run externally to scPipe and the resulting *BAM* file can be re-entered to the scPipe pipeline if samtools is not installed locally.

Fragment file generation was adopted from a python-based tool named Sinto (version 0.8) (19). Briefly, the fragment file is in *.BED* format created from the aligned read file (i.e. *BAM*) with the position of each Tn5 integration site, barcode of the cell that the fragment belongs to and the number of times the fragment was sequenced while collapsing the PCR duplicates. This is achieved by first extracting the cell barcode sequence associated with each read and adjusting the alignment positions for the 9 bp Tn5 shift by applying $+4/-5$ to the start and end position of the paired reads. Next, fragments below a certain quality threshold and a size larger than a maximum provided by the user are removed before collapsing any duplicates (if present). This python-based capability was integrated into the scPipe R package using basilisk (20). Feature matrix construction can be via a *.bed* file provided to the workflow (e.g. bulk peak file from MACS3) or using the reference genome (i.e. 'genome_bin' approach). In the 'genome_bin' approach, the genome is cut in to chunks of user-defined size and the overlap is calculated between the *bed* file and the fragment file. Using such an approach is resource intensive, yet more sensitive to rare open regions than using a bulk approach. Furthermore, being able to converge on the same features via this approach will make downstream integration of the multiple scATAC-Seq data sets more convenient. An optional peak calling step can also be carried out using the R version of MACS3 (MACSr (21)) which has been implemented to execute bulk-level peak calling across all the data. This is a faster method than the in-built *genome−bin* approach described above to identify features to generate the feature × cell matrix.

Generating the feature × cell matrix, SCE objects and QC statistics

The externally created (e.g. via Cell Ranger) or scPipe generated feature matrix and fragment file is used as the main input to generate the feature × cell count matrix. This step involves multiple filtering steps including cell calling and row-wise (i.e. feature-level), and column-wise (i.e. cell-level) filtering as well as read correction for the Tn5 cut site. A *GenomicAlignments* object (22) is generated for the fragments with cell barcode information and the feature file (peaks/bins) with the feature coordinate information which is then overlapped to generate the unfiltered counts matrix. This matrix then enters the cell calling step to distinguish 'true' cells from ambient background DNA. It has been previously been shown that filtering cells based on multiple QC metrics is the most effective approach to do cell calling (7), and this approach has been adopted in scPipe. This can be called with the parameter *cell_calling=filter* within the *sc_atac_feature_counting()* function. The resulting matrix is then converted to a binary matrix as well as a *SCE* object (11). An HTML report is generated using rmarkdown as an optional output of this step. This report summarises the QC metrics gathered during preprocessing to allow data quality to be visually assessed (Figure 1B–D), and is easy to share with collaborators. ScPipe's ATAC-Seq module also logs all details in the workflow so that a user can track the progress if a step fails or wants to further explore the pipeline parameters used.

Integration of SCE objects

Another important aspect of scPipe is the ability to integrate datasets where a common barcode file is available to match the cells between *SCE* objects to create a combined *SCE* object, as occurs for example in 10X Genomics multiome experiments. The R/Bioconductor package MultiAssayExperiment is used for this (23) to output combined *SCE* objects with colData() from the same barcodes bound together (Figure 1A). If there are barcodes with missing data from one or more *SCE* objects, they will still be included with the addition of 'NAs' to columns corresponding to the cells that do not share common barcodes. Storing data in this way allows the user to further leverage Bioconductor software and design principles to make biologically relevant inferences (e.g. to correct for batch effects, perform integration and clustering).

Human lung adenocarcinoma cell line dataset

The cell culture and sample preparation of mixtures of cells from different cell lines was performed as previously described (24). Briefly, five human lung adenocarcinoma cell lines (A549, H1975, H2228, H838 and HCC827) were obtained from ATCC (https://www.atcc.org/) and cultured separately in Roswell Park Memorial Institute (RPMI) 1640 medium with 10% fetal calf serum and 1% penicillin-streptomycin at 37°C with 5% carbon dioxide until near 100% confluency. Cells were counted manually using a hemocytometer and mixed with equal number to form single-cell suspension with around 2 000 000 cells. Cells were permeabilised with nuclei EZ Lysis Buffer from Sigma-Aldrich supplemented with Protector RNase Inhibitor and filtered to isolate nuclei. Nuclei underwent fluorescence-activated nuclei sorting (FANS) and 1 600 000 nuclei were output from the sorter. After concentrating the nuclei suspension to achieve 10 000 nuclei recovery during Gel Bead-In EMulsions (GEM) generation, 11 960 nuclei were input to form GEMs that were used to generate 10× Multiome (GEX + ATAC) libraries. The cDNA library was sequenced using Illumina NextSeq 500 with recommended cycles and the ATAC library was sequenced using Illumina NextSeq 500 with the custom recipe. FASTQ files were then generated with Cell Ranger ARC 2.0.0 *mkfastq*. The raw data are available from GEO under accession number GSE224045.

We have processed the scATAC-Seq data in two ways; (i) via the Cell Ranger ATAC pipeline, and (ii) via the scPipe pipeline. The scRNA-Seq data was processed through the standard Cell Ranger pipeline. The paired read were aligned to the hg38 reference. The tool demuxlet (25) was used to assign the cell line identity (i.e. ground truth) to the data using variant information as previously described (24). Next we developed custom scripts (available via GitHub) using mainly ggplot2(26), Seurat (27) and NMI R packages to generate plots for visualisation and comparison.

Results

Resource requirements of scPipe on scATAC-Seq data

We ran the scPipe scATAC-Seq pipeline on both the 20% and 80% GEMs to profile time and memory usage. For the 20% sample, which contains 31,749,732 reads, the pipeline ran for approximately 140 minutes, and required approximately 85 GB of RAM allocated. The longest processing step

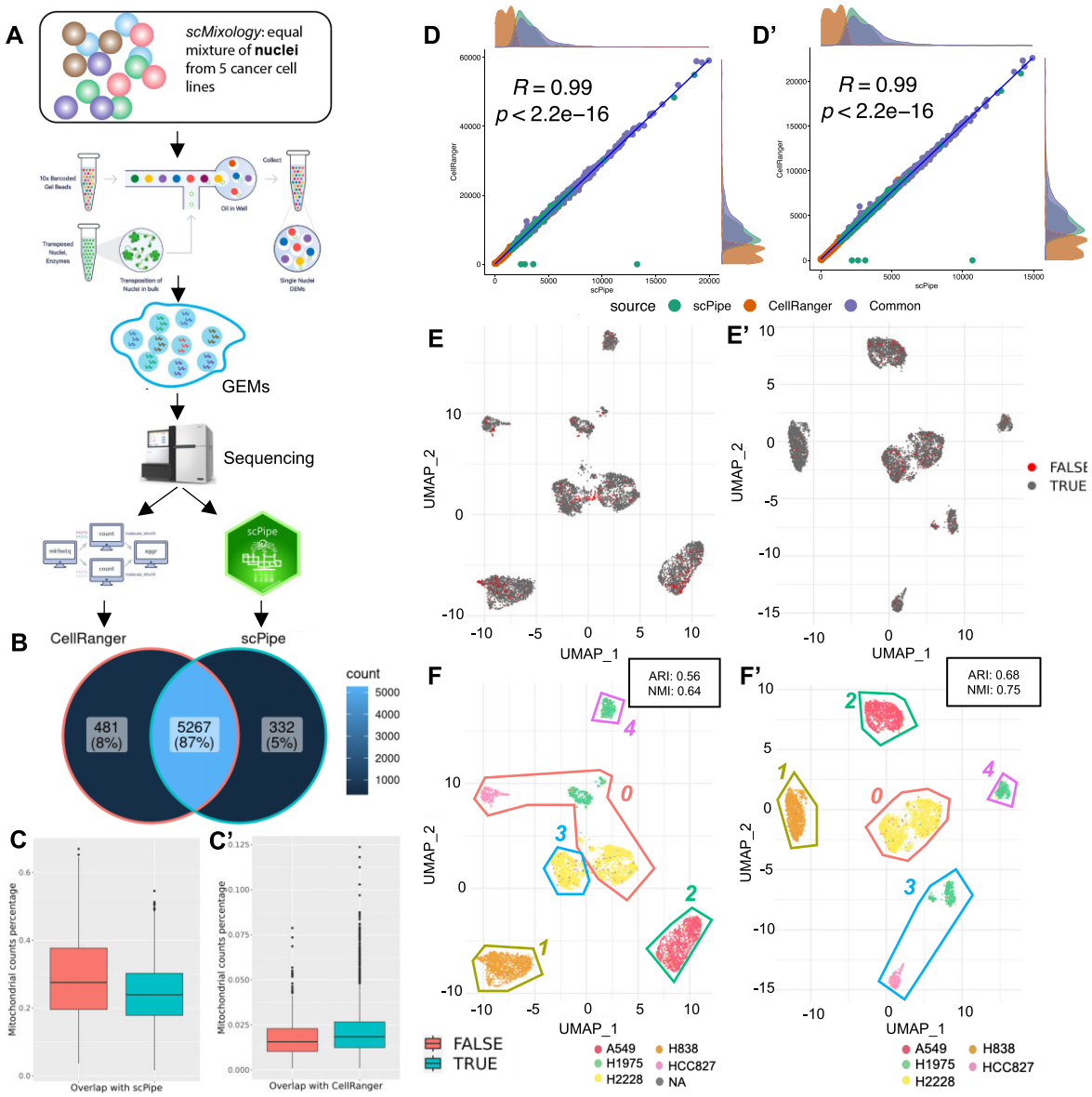

Figure 2. Comparing `scPipe` and `Cell Ranger` on a 10× dataset. **(A)** Summary of experimental design, which used cells from five distinct lung adenocarcinoma cell lines. An equal mixture of cells and nuclei were captured by the 10X protocol, and sequenced on the Illumina platform (see Materials and Methods). *FASTQ* files were generated by `Cell Ranger ARC` 2.0.0 and these reads were processed by `scPipe` and `Cell Ranger`. The panels of this figure pertain to the 80% results. **(B)** Venn diagram showing the overlap of cell barcodes detected by `Cell Ranger` and `scPipe`. **(C)** Box plot showing the percentage of mitochondrial gene counts in cells that are called by `Cell Ranger` and are common with/unique in comparison to `scPipe` or (C') `scPipe` and are common with/unique in comparison to `Cell Ranger` output. **(D)** Scatter plot of the per cell total counts and **(D')** number of features per cell obtained from `Cell Ranger` and `scPipe` in cells called in common between the two and cells that were unique to each of the two pipelines. Marginal density plots show the count distributions for each category. **(E)** UMAP plot generated for `Cell Ranger` and **(E')**`scPipe` output. Cell barcodes that only exist in (E) `Cell Ranger` or (E') `scPipe` are highlighted in red. **(F)** UMAP coloured by the ground truth for `Cell Ranger` and **(F')**`scPipe`. `Seurat` identified clusters are demarcated by coloured lines and numbered. The ARI and NMI values calculated per dataset are shown in the top right of the panels.

was *sc_aligning*, which internally uses `Rsubread`, and required 63 minutes to complete. For the 80% sample, containing 112,463,635 reads, the pipeline required approximately 604 minutes (10 hours) to complete and 540 GB of RAM allocated.

Downstream comparison between `scPipe` and `Cell Ranger ATAC`

We compared the clusters identified from `scPipe` preprocessed data to those obtained from the `Cell Ranger ATAC` pipeline, a popular tool for 10X scATAC-Seq preprocessing. A dataset that contains around 5,800 cells with ground truth available in the form of cell line identity labels obtained using variant information was used to compare the results from the two approaches (Figure 2A). `Cell Ranger` returned 5728 cells and `scPipe` 5599 cells after QC, with 5267 in common (Figure 2B). Inspection of features overlapping the mitochondrial genes as a QC metric shows that `Cell Ranger` has called more dead/poor quality cells that should be excluded from downstream analysis (Figure 2C). On the other hand, the mitochondrial contamination is lower and similar

for scPipe-only called cells and those identified by scPipe in common with Cell Ranger (Figure 2C'). Overall, there was high concordance between the called cell (Figure 2D) and feature counts (Figure 2D') within for the cells in common between the two pipelines. Moreover, 'scPipe-only' called cells contained counts and features that were more similar those found in common than the counts and features from 'Cell Ranger-only' called cells, whose distributions were concentrated towards the lower ranges for both quantities. The UMAP (28) generated from the Cell Ranger output shows that the 129 cells that appear in the Cell Ranger results but not in scPipe tend to cluster together away from the large clusters or in the margins of them (Figure 2E). In contrast, the UMAP generated from the scPipe output shows that the cells that only appear in scPipe results but not in Cell Ranger tend to be located within the main clusters (Figure 2E').

Comparison of the cell line labels assigned to these cells using demuxlet (25) to the clustering results obtained using the scATAC-Seq data alone in a Seurat analysis, was made by calculating the adjusted rand index (ARI) and normalised mutual information (NMI) scores which should be closer to 1 when the clustering is highly concordant between the two approaches. For the Cell Ranger data, an ARI of 0.56 and NMI of 0.64 (Figure 2F) were obtained. The scPipe generated output resulted in a higher ARI of 0.68 and NMI of 0.75 (Figure 2F'), indicating better concordance between the Seurat clustering and demuxlet results. Taken together, scPipe seemed to identify higher quality cells that were better separated according to the biological signal present in the data relative to the Cell Ranger processed data.

Discussion

To ensure flexibility, scPipe can preprocess data from a variety of different scATAC-Seq protocols that includes droplet-based and plate-based methods. scPipe can also handle combinatorial barcoding options when demultiplexing the reads, and can handle Unique Molecular Identifier (UMI) based scATAC-Seq approaches if they arise in the future. The resulting preprocessed data output by scPipe is stored in a versatile *SingleCellExperiment* (*SCE*) format, which means that the data can be further analysed downstream using *SCE*-object friendly tools available in R/Bioconductor. We demonstrate that data preprocessing with scPipe's scATAC-Seq preprocessing workflow produces comparable results to those from 10×'s Cell Ranger ATAC pipeline. As the integration of numerous large-scale data modalities becomes more routine, scPipe provides convenient and scalable preprocessing options for both scRNA-Seq and scATAC-Seq data, allowing data integration and joint analyses.

Data availability

scPipe's scATAC-Seq preprocessing module is available from Bioconductor, DOI: https://doi.org/doi:10.18129/B9.bioc.scPipe. The code used for the timing calculations and generating Figure 2 (panels B–F) are available from Zenodo, DOI: https://zenodo.org/doi/10.5281/zenodo.10185302.

Acknowledgements

This work was supported by funding from the Chan Zuckerberg Initiative DAF, an advised fund of Silicon Valley Community Foundation (Grant No. 2019-002443 to M.E.R.), Australian National Health and Medical Research Council (NHMRC) Investigator Grant (2017257 to M.E.R.), the Australian Research Council (Discovery Project No. 200102903 to M.E.R.), the Genomics Innovation Hub, Victorian State Government Operational Infrastructure Support, Australian Government NHMRC IRIISS and support from the Australian Cancer Research Foundation. The authors are grateful to Dr Saskia Freytag and Mr Reza Ghamsari for providing feedback on this manuscript.

Funding

Australian Research Council [200102903]; National Health and Medical Research Council [2017257]; Chan Zuckerberg Initiative [2019-002443].

Conflict of interest statement

None declared.

References

1. Buenrostro,J.D., Wu,B., Litzenburger,U.M., Ruff,D., Gonzales,M.L., Snyder,M.P., Chang,H.Y. and Greenleaf,W.J. (2015) Single-cell chromatin accessibility reveals principles of regulatory variation. *Nature*, **523**, 486–490.
2. Baek,S. and Lee,I. (2020) Single-cell ATAC sequencing analysis: from data preprocessing to hypothesis generation. *Comput. Struct. Biotechnol. J.*, **18**, 1429–1439.
3. Cusanovich,D.A., Hill,A.J., Aghamirzaie,D., Daza,R.M., Pliner,H.A., Berletch,J.B., Filippova,G.N., Huang,X., Christiansen,L., DeWitt,W.S., *et al.* (2018) A Single-cell atlas of in vivo mammalian chromatin accessibility. *Cell*, **174**, 1309–1324.
4. Mezger,A., Klemm,S., Mann,I., Brower,K., Mir,A., Bostick,M., Farmer,A., Fordyce,P., Linnarsson,S. and Greenleaf,W. (2018) High-throughput chromatin accessibility profiling at single-cell resolution. *Nat. Commun.*, **9**, 6–11.
5. Satpathy,A.T., Granja,J.M., Yost,K.E., Qi,Y., Meschi,F., McDermott,G.P., Olsen,B.N., Mumbach,M.R., Pierce,S.E., Corces,M.R., *et al.* (2019) Massively parallel single-cell chromatin landscapes of human immune cell development and intratumoral T cell exhaustion. *Nat. Biotechnol.*, **37**, 925–936.
6. Lareau,C.A., Duarte,F.M., Chew,J.G., Kartha,V.K., Burkett,Z.D., Kohlway,A.S., Pokholok,D., Aryee,M.J., Steemers,F.J., Lebofsky,R., *et al.* (2019) Droplet-based combinatorial indexing for massive-scale single-cell chromatin accessibility. *Nat. Biotechnol.*, **37**, 916–924.
7. Yu,W., Uzun,Y., Zhu,Q., Chen,C. and Tan,K. (2020) scATAC-pro: a comprehensive workbench for single-cell chromatin accessibility sequencing data. *Genome Biol.*, **21**, 94.
8. Chen,H., Lareau,C., Andreani,T., Vinyard,M.E., Garcia,S.P., Clement,K., Andrade-Navarro,M.A., Buenrostro,J.D. and Pinello,L. (2020) Assessment of computational methods for the analysis of single-cell ATAC-seq data. *Genome Biol.*, **20**, 241.
9. Fang,R., Preissl,S., Li,Y., Hou,X., Lucero,J., Wang,X., Motamedi,A., Shiau,A.K., Zhou,X., Xie,F., *et al.* (2021) Comprehensive analysis of single cell ATAC-seq data with SnapATAC. *Nat. Commun.*, **12**, 1337.
10. Tian,L., Su,S., Dong,X., Amann-Zalcenstein,D., Biben,C., Seidi,A., Hilton,D.J., Naik,S.H. and Ritchie,M.E. (2018) scPipe: a flexible R/Bioconductor preprocessing pipeline for single-cell RNA-sequencing data. *PLoS Comput. Biol.*, **14**, e1006361.
11. Amezquita,R.A., Lun,A.T., Becht,E., Carey,V.J., Carpp,L.N., Geistlinger,L., Marini,F., Rue-Albrecht,K., Risso,D., Soneson,C., *et al.* (2019) Orchestrating single-cell analysis with Bioconductor. *Nat. Methods*, **17**, 137–145.

12. R Core Team (2020) In: *R: a Language and Environment for Statistical Computing*. Vienna, Austria.
13. Huber,W., Carey,V.J., Gentleman,R., Anders,S., Carlson,M., Carvalho,B.S., Bravo,H.C., Davis,S., Gatto,L., Girke,T., *et al.* (2015) Orchestrating high-throughput genomic analysis with Bioconductor. *Nat. Methods*, **12**, 115–121.
14. Eddelbuettel,D. (2013) *Seamless R and C++ Integration with Rcpp*. Springer, NY.
15. Allaire,J., Ushey,K., Tang,Y. and Eddelbuettel,D. (2017) reticulate: R Interface to Python. https://github.com/rstudio/reticulate.
16. Allaire,J., Xie,Y., McPherson,J., Luraschi,J., Ushey,K., Atkins,A., Wickham,H., Cheng,J., Chang,W. and Iannone,R. (2021) rmarkdown: Dynamic Documents for R , R package version 2.7. https://github.com/rstudio/rmarkdown.
17. Liao,Y., Smyth,G.K. and Shi,W. (2019) The R package Rsubread is easier, faster, cheaper and better for alignment and quantification of RNA sequencing reads.. *Nucleic Acids Res.*, **47**, e47.
18. Li,H., Handsaker,B., Wysoker,A., Fennell,T., Ruan,J., Homer,N., Marth,G., Abecasis,G. and Durbin,R. (2009) The Sequence Alignment/Map format and SAMtools. *Bioinformatics*, **25**, 2078–2079.
19. Stuart,T. (2022) Sinto: single-cell analysis tools. https://github.com/timoast/sinto.
20. Lun,A.T.L. (2022) basilisk: a Bioconductor package for managing Python environments. *J. Open Source Softw.*, **7**, 4742.
21. Hu,Q. (2022) MACSr: MACS: model-based analysis for ChIP-Seq. https://bioconductor.org/packages/MACSr.
22. Lawrence,M., Huber,W., Pagès,H., Aboyoun,P., Carlson,M., Gentleman,R., Morgan,M.T. and Carey,V.J. (2013) Software for Computing and annotating Genomic ranges. *PLoS Comput. Biol.*, **9**, e1003118.
23. Ramos,M., Schiffer,L., Re,A., Azhar,R., Basunia,A., Rodriguez,C., Chan,T., Chapman,P., Davis,S.R., Gomez-Cabrero,D., *et al.* (2017) Software for the integration of multiomics experiments in bioconductor. *Cancer Res.*, **77**, e39–e42.
24. Tian,L., Dong,X., Freytag,S., Cao,K.A.L., Su,S., JalalAbadi,A., Amann-Zalcenstein,D., Weber,T.S., Seidi,A., Jabbari,J.S., *et al.* (2019) Benchmarking single cell RNA-sequencing analysis pipelines using mixture control experiments. *Nat. Methods*, **16**, 479–487.
25. Kang,H.M., Subramaniam,M., Targ,S., Nguyen,M., Maliskova,L., McCarthy,E., Wan,E., Wong,S., Byrnes,L., Lanata,C.M., *et al.* (2017) Multiplexed droplet single-cell RNA-sequencing using natural genetic variation. *Nat. Biotechnol.*, **36**, 89–94.
26. Wickham,H. (2016) *ggplot2: Elegant Graphics for Data Analysis*. Springer-Verlag, NY.
27. 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.
28. McInnes,L., Healy,J., Saul,N. and Großberger,L. (2018) UMAP: Uniform Manifold Approximation and Projection. *J. Open Source Softw.*, **3**, 861.

Received: September 18, 2023. **Editorial Decision:** November 15, 2023. **Accepted:** November 23, 2023