FastCount: A Fast Gene Count Software for Single-cell RNA-seq Data

Jinpeng Liu
jinpeng.liu@uky.edu
University of Kentucky
Lexington, Kentucky, USA

Xinan Liu
hitcs1xa@gmail.com
University of Kentucky
Lexington, Kentucky, USA

Ye Yu
sdy1990@gmail.com
University of Kentucky
Lexington, Kentucky, USA

Chi Wang
chi.wang@uky.edu
University of Kentucky
Lexington, Kentucky, USA

Jinze Liu
jinze.liu@vcuhealth.org
Virginia Commonwealth University
Richmond, Virginia, USA

ABSTRACT

Motivation: The advent of single cell RNA-seq (scRNA-seq) enables scientists to characterize the transcriptomic response of cells under different conditions and understand expression heterogeneity at single cell level. One of the fundamental steps in scRNA-seq analysis is to summarize raw sequencing reads into a list of gene counts for each individual cell. However, this step remains to be most time-consuming and resource intensive in the analysis workflow due to the large amount of data produced in a scRNA-seq experiment. It is further complicated by the special handling of cell barcodes and unique molecular identifiers (UMIs) information in the read sequences. For example, the gene count summarization of 10X Chromium sequencing by standard Cell Ranger *count* often takes many hours to finish when running on a computing cluster. Although several alignment-free algorithms have been developed to improve efficiency, their derived gene count suffer from poor concordance with Cell Ranger *count* and algorithm-specific bias[1].

Results: In this work, we present a light-weight *k*-mer based gene count algorithm, FastCount, to support efficient UMI counts from single cell RNA-seq data. We demonstrate that FastCount is over an order of magnitude faster than Cell Ranger *count* while achieving competitive accuracy on 10X Genomics single cell RNA-seq data. FastCount is a stand-alone program implemented in C++. The source code is located at <https://bitbucket.org/merckey/fastcount/>.

CCS CONCEPTS

• Applied computing; • Life and medical sciences;

KEYWORDS

single cell RNA-seq, UMI quantification, alignment-free, *k*-mers

ACM Reference Format:

Jinpeng Liu, Xinan Liu, Ye Yu, Chi Wang, and Jinze Liu. 2021. FastCount: A Fast Gene Count Software for Single-cell RNA-seq Data. In *12th ACM*

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from [permissions@acm.org](https://permissions.acm.org).

BCB '21, August 1–4, 2021, Gainesville, FL, USA

© 2021 Association for Computing Machinery.

ACM ISBN 978-1-4503-8450-6/21/08...\$15.00

<https://doi.org/10.1145/3459930.3469544>

International Conference on Bioinformatics, Computational Biology and Health Informatics (BCB '21), August 1–4, 2021, Gainesville, FL, USA. ACM, New York, NY, USA, 8 pages. <https://doi.org/10.1145/3459930.3469544>

1 INTRODUCTION

The advent of single cell RNA-seq (scRNA-seq) [7, 11, 20] enables scientists to characterize the transcriptomic landscape of large number of cells at single cell level for the discovery of novel cell types and their biological functions. Similar to the bulk RNA-seq, the first step in analyzing scRNA-seq data is to map the sequencing reads to the reference genome/transcriptome to identify the genes of origin and quantify the gene abundance in each cell. This is followed by downstream analysis such as cell clustering, cell type identification and trajectory analysis.

Among all the scRNA-seq analysis steps, the most time-consuming step is the UMI-based gene count summarization. This involves the calculation of the total number of UMIs associated with each gene, for a more accurate single cell characterization [2, 3]. Several tools have been developed by reusing the bulk RNA-seq quantification methods and taking into account of the UMIs and cell barcodes information incorporated in the scRNA-seq protocols ([6, 13–16, 20]). For example, Cell Ranger, the standard software package for processing the scRNA-seq data generated by the 10X Genomics Chromium library, first aligns the reads to the reference genome using STAR [4] and then performs the downstream UMI correction, deduplication and counting using only reads confidently mapped to exons. The zUMIs pipeline [14] also uses STAR for the read alignment task. Reads are then assigned to the exon and intron features using featureCounts [9]. These alignment-based algorithms are resource intensive and time-consuming. Recently, two alignment-free tools Alevin [16] and Kallisto+Bustools [13] are developed based on their previous bulk transcript-level RNA-seq analysis modules. Comparing to the traditional alignment-based methods, they offer significant speed improvement. However, a recent study has shown that the alignment-free algorithms may report gene artifacts that are not expected to express in the tissues[1].

In this paper, we present FastCount, a new tool that is capable of alignment-free UMI count summarization for scRNA-seq data leveraging gene-specific *k*-mers. Different from the alignment-based algorithms that include a computationally intensive read mapping step, FastCount assigns a read directly to the potential gene features based on the *k*-mer origin information. FastCount uses a novel

Figure 1: A toy example illustrating FastCount algorithm. (a) The construction of GeneOthello index from the reference gene sequences. In this example, k -mers ($k=3$) in the reference set are extracted from each reference transcript. Their occurrences within genes are tabulated and are categorized into gene signatures and cliques (Top). A GeneOthello index is constructed to store the many-to-one mapping between the k -mers and the gene feature indices (Bottom). (b) Read assignment procedure. k -mers in a read are extracted and queried against GeneOthello to obtain their gene features. Continuous k -mers with the same gene feature are clustered into feature windows. The gene assignment of a read is determined by the gene that dominates the longest feature window. (c) Individual barcode identification and gene assignment for a paired-end read. FastCount first removes read pairs with invalid barcode or UMI sequences using read1. If read1 corresponds to valid barcode and UMI sequences, it proceeds to assign read2 to gene features using FastCount read assignment algorithm shown in (b). (d) UMI summarization in individual cells. Our method iterates through all the read pairs for gene-level assignment and keeps track of the number of reads assigned to each gene feature for individual cells using a barcode+UMI counts matrix. It then corrects UMIs based on the UMI counts and UMI sequence differences for each gene feature. The final feature-barcode matrix summarizes the number of unique UMIs for each gene feature in each cell.

data structure, GeneOthello, to classify reads to the corresponding gene identifiers sharing approximately the same set of k -mers. It conducts read count in a gene without necessitating detailed read alignment information. FastCount is implemented to handle

UMI and cell barcode information specific to scRNA-seq data, allowing a speedy assessment of scRNA-seq cell distribution using raw sequencing reads. We demonstrate through experiment that FastCount scRNA-seq application using 10X Genomics data and

compare the performance to 10X Genomics' toolkit, Cell Ranger. FastCount is over an order of magnitude faster than Cell Ranger *count* with very competitive accuracy. When comparing with two other alignment-free algorithms, Kallisto and Alevin, FastCount shows much higher concordance with Cell Ranger *count*. In conclusion, FastCount strikes a great balance between the speed and accuracy and can be a great alternative for cell gene count summarization in large scale scRNA-seq analysis.

Figure 2: The percentage of k -mers as a function of maximum k -mer occurrences in Human reference genome GRCh38 ($k = 21$).

2 METHODS

The term "gene count" typically refers to the number of reads sequenced in each gene within a given RNA-seq sample. Calculating gene count requires assigning each read back to the gene where it is sequenced from. Unlike read alignment, read assignment does not require high resolution base-by-base continuous matching between the read sequence and genomic reference. Instead, read assignment can be simplified to the identification of the gene that best matches the set of k -mers present in the read. We show that when $k = 21$, the majority of k -mers (93.9% in GRCh38) are unique to the genes carrying them (Figure 2). Therefore, the gene-specific k -mers may serve as signatures for the read classification problem. The rest of the k -mers (6.1% in GRCh38) appear in more than one gene. The proposed method FastCount leverages this property and turns gene count into a read assignment problem.

FastCount is implemented with the functionalities to perform quality filtering using barcode and UMI sequences; assign read to gene feature using the gene identity sequence; group reads based on the barcode, gene feature and UMI information to correct UMI sequences, collapse UMI duplicates and finally summarize the feature-barcode UMI count matrix. In this section, we illustrate methods behind each of these steps.

2.1 Construction of GeneOthello Index

The first step of FastCount method is to establish a mapping between k -mers and genes. The k -mers in RNA-seq are expected to be from mRNA transcriptome although these can be expanded to

include other RNA species as well. Therefore, we extract all the k -mers present in the reference transcript sequences of each gene. These k -mers S are divided into two categories: gene-specific k -mers and gene-clique k -mers. Each of the gene-specific k -mers can uniquely identify one gene. Each of k -mers in a gene clique set is not unique to a gene and instead, is shared by a set of genes, which we refer to as a gene clique. Those k -mers are often a result of repetitive sequences but provide useful information on handling reads that map ambiguously to multiple genes. In total, we would like to establish a mapping between the set of transcriptome k -mers and their associated gene features as shown in Figure 1a. Formally, let G be genes corresponding to gene-specific k -mers and C be the set of gene cliques. Let F be a feature set containing genes in G and gene cliques in C , where $F = G \cup C$ and $G \cap C = \emptyset$. Thus, there exists a many-to-one mapping between $S \rightarrow F$, such that $m(s) = F$.

The efficiency of gene count procedure relies on how fast one can quickly mapping a read to a gene. In our case, it is translated to how fast we can map a k -mer to a gene. To this end, we have adopted a hashing classifier, called othello, to facilitate the mapping between k -mers and the features F [18]. Othello is a minimal perfect hashing algorithm (MPH) that provides key-to-value query in constant time. The othello algorithm has demonstrated great scalability in both memory and query speed in several Bioinformatics applications [10, 19].

We build an index, named GeneOthello, to store the many-to-one mapping between the aforementioned transcriptome k -mers and gene features. We encode gene features F as a set of l -bit integers $V = \{0, 1, 2, \dots, |G|, |G|+1, |G|+2, \dots, |G|+|C|\}$, where 0 is specially allocated for k -mers with occurrence $> n$ and $l = \lceil \log_2(|F| + 1) \rceil$. GeneOthello $O(S, V)$ maps the predefined set of k -mers S to the gene features in V . Let $T : S \rightarrow V$ be the function that maps k -mers to the gene features in V , where $T(k)$ indicates the feature of a k -mer $s \in S$. GeneOthello maintains a query function $\tau : U \rightarrow I$ that maps the universal set of k -mers, U (i.e. $U = \theta^k, \theta = \{A, G, C, T\}$) to the set of all l -bit integers, $I = \{0, 1, \dots, 2^l - 1\}$. Therefore, $S \subset U$ and $V \subset I$. GeneOthello has the following properties: 1) given a k -mer $s \in S$, a GeneOthello querying $\tau(s)$ guarantees returning the correct gene feature id v ; 2) for any alien k -mers, $s' \notin S$, GeneOthello has a higher probability to assign s' to a dummy feature $\tau(s') \in I - V$ than a false positive feature $\tau(s') \in V$ [19].

The GeneOthello structure includes a pair of hash function $\langle h_a, h_b \rangle$ and two arrays of l -bit integers A and B . The hash functions and the content of the integer arrays are precomputed during GeneOthello construction. The hash functions provide the mapping between the universal k -mer set and the corresponding index location in A and B , that is $h_a : U \rightarrow \{0, 1, \dots, m_a - 1\}$ and $h_b : U \rightarrow \{0, 1, \dots, m_b - 1\}$. A query of a k -mer s on GeneOthello will first yield indices $h_a(s)$ and $h_b(s)$. The feature $\tau(s)$ of k -mer s is computed by the integer values at $A[h_a(s)]$ and $B[h_b(s)]$ as $\tau(s) = A[h_a(s)] \oplus B[h_b(s)]$. The procedure only accesses two memory locations in A and B and a bitwise XOR operation, making the query execution extremely fast.

2.2 Read Assignment to Gene Features

To assign a read to the correct gene feature, Fastcount first decomposes a read sequence into consecutive k -mers (Figure 1b) and each

Table 1: Runtime comparison on 10X Genomics single cell gene expression datasets in different tools

Dataset	Chemistry	Species	Total Reads	Runtime (Minutes)			
				FastCount	Cell Ranger	Alevin	Kallisto
1K PBMCs	V3	Human	66,601,887	4.1	154.2	11.9	3.0
10K PBMCs	V3	Human	638,901,019	38.9	1563.5	121.7	26.4
4K PBMCs	V2	Human	379,462,522	22.2	914.0	66.7	16.9
8K PBMCs	V2	Human	784,064,148	46.5	1865.3	132.6	34.4
1K Heart	V3	Mouse	84,512,390	5.1	186.2	7.4	3.6
10K Heart	V3	Mouse	290,439,571	21.6	674.5	26.4	12.1

k -mer is queried against the GeneOthello index to retrieve its gene features. Ideally, the read assignment is straightforward when all k -mers pointing to the same gene. However, this process is often complicated by the presence of alien k -mers that are absent from the reference transcriptome. Alien k -mers can be a result of system artifacts during the RNA-seq library preparation and sequencing, contamination or novel transcript isoforms [8, 17]. When querying on alien k -mers, majority of them can be easily detected by GeneOthello. But in the cases that an alien k -mer may be falsely allocated to a reference gene feature, failure of detecting it may lead to a false positive read assignment.

FastCount tackles the challenge of alien k -mers using the following strategies: 1) when constructing GeneOthello index, we intentionally expand the size of I such that many alien k -mers are categorized into a group that is not overlapping with any valid transcriptome features $I - V$. Given the predefined V , we can increase the probability of an alien k -mer falls in $I - V$ by increasing I , that is $\frac{2^I - |V|}{2^I}$ [10]. 2) in the case that an alien k -mer is falsely assigned to the existing gene features, we require the presence of at least two consecutive k -mers returning the same gene feature to be considered as a feature segment for gene assignment. As the indices queried by alien k -mers are random, the chance of seeing the same gene assignment from the two different alien k -mers is very low [10].

FastCount performs read assignment with the following procedure: 1) it iterates through each k -mer along the read to retrieve its gene feature by querying against the GeneOthello index. With the presence of alien k -mers, gene features along the reads may be divided into section. We merge continuous gene features into feature segments and reject the dummy features from alien k -mers. Each feature segment is weighted by the $w_j = d_j^2$, where d_j is the number of k -mers in the feature segment j , to take into account of the k -mer spatial information on the read. 2) We filter feature segments with length < 2 from feature assignment to remove the possible alien k -mers query falsely return to the known features. 3) Gene segments are then ranked by the weights to identify the most dominant feature. 4) The rest of the feature segments are compared to the dominant feature. If there are no consensus gene feature among the segments, the read is removed from the assignment; if they all indicate a single consensus gene, the read is assigned to the gene feature; if they indicate to a set of genes, the case of multi-mapping read, we will not include this read for feature assignment.

2.3 Handling of scRNA-seq specific information

Another key step in scRNA-seq analysis is to appropriately handle the single-cell specific information in the cell barcode and UMI sequence. FastCount accepts sequencing FASTQ files that are expected to contain barcode and UMI information in one of the read file and the gene expression read file in another. For example, a typical read from 10X Genomics Chromium technique is in the form of read pairs, read1 and read2 (Figure 1c). Read1 contains the cell barcode sequence that is used to tag the cell origin of the cDNA followed by the UMI sequence which labels the cDNA molecule. Read2 is the sequenced nucleotide bases from the cDNA fragment used in to determine the gene identity.

FastCount first filters the low quality barcode and UMI sequences. Due to potential sequencing and amplification errors in the barcode and UMI sequences, the number of detected cells and UMI counts are usually inflated. FastCount parses the barcode and UMI from the read1 and looking for any "N" bases. Read pairs with 'N's in either barcode or UMI will be skipped for the read assignment step. Reads pass the barcode and UMI filtering will be further compared against the barcode whitelist which is the full list of all known barcode sequences that are available for cell tagging during the library preparation. If a barcode is 1-hamming distance apart from a barcode in the whitelist, it will be replaced to the whitelist barcode. Barcodes more than 1-hamming distance away from the whitelist are considered invalid. Only reads with valid barcode+UMI sequences are assigned to its potential gene features based on its k -mers information using the FastCount read assignment algorithm. FastCount keeps tracking the association between the UMIs and gene features for each individual cell and stores the UMI counts in a barcode+UMI counts matrix. After iterating through all the reads for the feature assignment, FastCount will try to further mitigate errors in UMIs using the barcode+UMI counts information (Figure 1d). The UMI correction is performed per cell per gene feature. FastCount compares one UMI against the rest of UMIs under the same gene feature within each cell looking for the ones that are 1-hamming distance apart. An UMI with lower count is corrected towards the higher count UMI. The UMI corrected counts matrix then undergoes UMI deduplication to group duplicate UMIs and collapse them into a single consensus one. Finally, FastCount counts the number of unique UMIs for each gene-level feature in each individual cell and reports the feature-barcode matrix in the Market Exchange Format for downstream analysis.

Table 2: UMI counts correlation per gene between Cell Ranger for FastCount, Alevin and Kallisto

Datasets	FastCount			Alevin			Kallisto		
	<i>Pearson</i>	<i>Spearman</i>	<i>MRD</i>	<i>Pearson</i>	<i>Spearman</i>	<i>MRD</i>	<i>Pearson</i>	<i>Spearman</i>	<i>MRD</i>
pbmc_1k_v3	0.999	0.975	0.029	0.996	0.951	0.057	0.998	0.924	0.096
pbmc_10k_v3	0.999	0.973	0.033	0.996	0.948	0.060	0.998	0.920	0.100
heart_1k_v3	0.998	0.980	0.023	0.989	0.952	0.066	0.992	0.936	0.080
heart_10k_v3	0.996	0.977	0.031	0.988	0.954	0.062	0.992	0.943	0.067
pbmc4k	0.999	0.989	0.022	0.993	0.972	0.050	0.993	0.935	0.113
pbmc8k	0.999	0.987	0.024	0.991	0.970	0.051	0.994	0.940	0.100

Figure 3: The scatter plots of the total number of UMI counts per cell between CR and FastCount, Alevin and Kallisto in the six 10X Genomics scRNA-seq datasets.

3 RESULTS

3.1 Evaluation datasets

Due to the lack of ground truth scRNA-seq dataset, our evaluation focuses on the scRNA-seq data generated using the widely accepted 10X Genomics Chromium platform. The filtered feature-barcode matrix quantified by *count* function in the Cell Ranger pipeline is used as the reference dataset. We compare how well each tested method correlates with Cell Ranger’s UMI counting results. We include six datasets published by 10X Genomics (Table 1) to exam the performance of the different tools in terms of different species, tissue types, sequencing depth and library chemistry. Datasets pbmc_1k_v3 and pbmc_10k_v3 were derived from human peripheral blood mononuclear cells (PBMCs) and were prepared using the 10X Chromium v3 library. We also include heart_1k_v3

and heart_10k_v3 which are cells from whole heart of an E18 mouse. Additionally, pbmc4k and pbmc8k are chosen to evaluate the performance on data generated by 10X Chromium v2 chemistry. The number of cells in these datasets ranges from 1,000 to 8,000 per sample. They were processed by 10X Genomics using the standard Cell Ranger pipeline. The raw sequencing FASTQ files are analyzed by each tool using the recommended settings. The resulting feature-barcode matrices are compared with the reference datasets quantified by Cell Ranger *count* to evaluate the concordance in gene- and cell-level expression as well as in the downstream biological inference.

3.2 Accuracy analysis

We assess the accuracy of FastCount in scRNA-seq quantification and compare its performance to Cell Ranger *count*, Alevin [16] and

Figure 4: Pearson correlation of UMI counts within each cell as a function of total UMI counts per cell .

Kallisto [13]. To make the evaluation comparable to Cell Ranger, we download the reference packages provided by 10X Genomics for human and mouse samples. We then subset the GENCODE [5] transcript sequence files for either species (human GRCh38.p12 release 30, mouse GRCm38.p6 release M21) with the same set of transcripts used by Cell Ranger. The filtered transcripts are used to construct the reference indices for FastCount, Kallisto and Alevin.

Each of the six datasets is processed using default settings by the tested algorithms to generate the raw feature-barcode matrix. We then subset this matrix to keep the same set of features and cells reported by the Cell Ranger pipeline. The filtered matrices are compared against the reference Cell Ranger’s results.

We first calculate the gene-level correlation of the UMI counts for each gene across all cells between the tested algorithms and Cell Ranger *count* using the pearson’s correlation, spearman’s correlation and median relative difference (MRD) statistics per cell (Table 2). The results indicate that FastCount shows the highest degree of agreement to the reference results in all datasets (median pearson’s correlation = 0.998, spearman’s correlation = 0.973 and MRD = 0.033). In comparison, Alevin and Kallisto show a less degree of concordance with cell ranger especially with scRNA-seq from heart tissues.

We next exam the correlation on cell-level expression in total UMI counts between the tested algorithms and Cell Ranger. We calculate the total mRNA abundance within each cell by the sum of the UMI counts for each expressed gene in a cell. The correspondence in the total UMI counts per cell between the tested methods and Cell Ranger is presented in Figure 3. We see high agreements

between the three alignment-free methods and Cell Ranger with points centered around the diagonal line in the scatter plots. FastCount has the highest degree of concordance in all the six tested datasets, while the total UMI counts by Alevin and Kallisto show larger variations from the Cell Ranger reference results. Kallisto consistently overestimate the total gene expression level in the cells. We also observe that many low-expressed cells in the two mouse heart tissue datasets (heart_1k_v3 and heart_10k_v3) are missing in Alevin while they are all identified by FastCount, Kallisto and Cell Ranger *count*. These findings agree with the benchmark paper reported by [1].

We further assess the cell-level concordance by calculating the pearson’s correlation in the UMI count of the same cell between the tested methods and Cell Ranger. We plot the density of the correlation coefficient as a function of total UMI counts per cell in Figure 4. Each point represents a cell and the points are colored based on the number of neighboring points indicating the overall distribution of the correlation. The pearson’s correlation in the cell-level UMI counts between FastCount and Cell Ranger clustered into a much narrower range and most of the cells have correlation coefficient close to 1. In contrast, many cells show relatively lower correlation in Alevin and Kallisto indicated by the much more spread out points on the graphs.

Lastly, to evaluate the the impact of the different cell gene count tools on downstream cell clustering analysis. We perform dimensionality reduction using t-Distributed Stochastic Neighbor Embedding (t-SNE) for each dataset using scater package [12]. Figure 5 shows that the t-SNE plots from Cell Ranger and FastCount are

Figure 5: Compatible t-SNE plots using the feature-barcode matrices generated by Cell Ranger and FastCount for six different scRNA-seq datasets.

very similar across all six data sets, retaining all the clustering structures and embedding shapes. In contrast, the distribution of clusters varies significantly using Alevins and Kallisto cell count results. The difference becomes more apparent when the composition of clusters is complex.

3.3 Runtime comparison

We benchmark the runtime used for each algorithm using the six testing datasets. All methods are evaluated on a 16-core Intel Xeon E5-2670 @ 2.60 GHZ Linux server with 64 GB of RAM using 10 threads. The result is shown in Table 1. In general, FastCount is at least 1.5 orders of magnitude faster than Cell Ranger *count*. For example, FastCount used only 4.1 minutes to process 1K PBMCs data set which contains around 1000 cells and 66 millions of reads, while Cell Ranger took 154.2 minutes. For 8K PBMCs which contains around 8000 cells and 784 millions of reads, FastCount used only 46.5 minutes, while Cell Ranger needed 31 hours. FastCount is around 3 times faster than Alevin except for the 2 mouse heart samples. This is because Alevin pre-filtered many (25.6% in 1K Heart, 26.8% in 10K Heart) cells before the read assignment and UMI quantification steps which largely reduced the number of reads required for processing. This pre-filtering step resulted in many cells missing from the final expression matrix. Kallisto is the fastest algorithm but the runtime between FastCount and Kallisto is quite close.

4 CONCLUSION

In this paper, we present FastCount, a novel alignment-free approach to quantify gene count in individual cells with scRNA-seq

data. Comparing to the other alignment-free algorithms, FastCount shows a higher degree of concordance with Cell Ranger’s *cellranger count* in both cell-level and gene-level expression, in the meantime provides 1.5 orders of magnitude speed improvement over Cell Ranger.

FastCount assigns sequencing reads directly to genes using gene-specific and gene-clique *k*-mers. It fully utilizes the cell barcode and UMI information in the sequencing reads to computationally allocate reads to the cells of origin while identifying technical artifacts from PCR amplification. We evaluate the accuracy of FastCount on scRNA-seq quantification using the results reported by Cell Ranger as the reference data sets. We observe that different quantification pipeline implementation influences the single cell experiment qualities in terms of genes/cells expression level and the downstream analysis.

The read processing step of scRNA-seq data is the first and most fundamental step during the scRNA-seq analysis pipeline. However, the commonly used Cell Ranger *count* often requires many hours of CPU times and extensive resources that are only available to computational servers. Being capable of processing 1,000 cells under 5 minutes on a regular laptop, Fastcount provides a lightweight but fast alternative, making it possible to combine cell count together with downstream analysis such as Seurat or Scanpy in one pipeline.

ACKNOWLEDGMENTS

REFERENCES

- [1] Ralf Schulze Brüning, Lukas Tombor, Marcel H. Schulz, Stefanie Dimmeler, and David John. 2021. Comparative Analysis of common alignment tools for single cell RNA sequencing. *bioRxiv* (2021), 2021.02.15.430948. <https://doi.org/10.1101/2021.02.15.430948>
- [2] Andrew Butler, Paul Hoffman, Peter Smibert, Efthymia Papalexi, and Rahul Satija. 2018. Integrating single-cell transcriptomic data across different conditions, technologies, and species. *Nature Biotechnology* 36, 5 (2018), 411–420. <https://doi.org/10.1038/nbt.4096>
- [3] W. Chen, Y. Li, J. Easton, D. Finkelstein, G. Wu, and X. Chen. 2018. UMI-count modeling and differential expression analysis for single-cell RNA sequencing. *Genome Biol* 19, 1 (2018), 70. <https://doi.org/10.1186/s13059-018-1438-9>
- [4] A. Dobin, C. A. Davis, F. Schlesinger, J. Drenkow, C. Zaleski, S. Jha, P. Batut, M. Chaisson, and T. R. Gingeras. 2013. STAR: ultrafast universal RNA-seq aligner. *Bioinformatics* 29, 1 (2013), 15–21. <https://doi.org/10.1093/bioinformatics/bts635>
- [5] A. Frankish, M. Diekhans, A. M. Ferreira, R. Johnson, I. Jungreis, J. Loveland, J. M. Mudge, C. Sisu, J. Wright, J. Armstrong, I. Barnes, A. Berry, A. Bignell, S. Carbonell Sala, J. Chrast, F. Cunningham, T. Di Domenico, S. Donaldson, I. T. Fiddes, C. García Giron, J. M. Gonzalez, T. Grego, M. Hardy, T. Hourlier, T. Hunt, O. G. Izuogu, J. Lagarde, F. J. Martin, L. Martinez, S. Mohanan, P. Muir, F. C. P. Navarro, A. Parker, B. Pei, F. Pozo, M. Ruffier, B. M. Schmitt, E. Stapleton, M. M. Suner, I. Sycheva, B. Uszczyńska-Ratajczak, J. Xu, A. Yates, D. Zerbino, Y. Zhang, B. Aken, J. S. Choudhary, M. Gerstein, R. Guigó, T. J. P. Hubbard, M. Kellis, B. Paten, A. Reymond, M. L. Tress, and P. Flicek. 2019. GENCODE reference annotation for the human and mouse genomes. *Nucleic Acids Res* 47, D1 (2019), D766–d773. <https://doi.org/10.1093/nar/gky955>
- [6] T. Hashimshony, N. Senderovich, G. Avital, A. Klochendler, Y. de Leeuw, L. Anavy, D. Gennert, S. Li, K. J. Livak, O. Rozenblatt-Rosen, Y. Dor, A. Regev, and I. Yanai. 2016. CEL-Seq2: sensitive highly-multiplexed single-cell RNA-Seq. *Genome Biol* 17 (2016), 77. <https://doi.org/10.1186/s13059-016-0938-8>
- [7] A. M. Klein, I. Mazutis, I. Akartuna, N. Tallapragada, A. Veres, V. Li, L. Peshkin, D. A. Weitz, and M. W. Kirschner. 2015. Droplet barcoding for single-cell transcriptomics applied to embryonic stem cells. *Cell* 161, 5 (2015), 1187–1201. <https://doi.org/10.1016/j.cell.2015.04.044>
- [8] Samuel Levy, Granger Sutton, Pauline C Ng, Lars Feuk, Aaron L Halpern, Brian P Walenz, Nelson Axelrod, Jiaqi Huang, Ewen F Kirkness, Gennady Denisov, Yuan Lin, Jeffrey R MacDonald, Andy Wing Chun Pang, Mary Shago, Timothy B Stockwell, Alexia Tsimouri, Vineet Bafna, Vikas Bansal, Saul A Kravitz, Dana A Busam, Karen Y Beeson, Tina C McIntosh, Karin A Remington, Josep F Abril, John Gill, Jon Borman, Yu-Hui Rogers, Marvin E Frazier, Stephen W Scherer, Robert L Strausberg, and J. Craig Venter. 2007. The Diploid Genome Sequence of an Individual Human. *PLOS Biology* 5, 10 (09 2007), 1–32. <https://doi.org/10.1371/journal.pbio.1000163>

- 1371/journal.pbio.0050254
- [9] Y. Liao, G. K. Smyth, and W. Shi. 2014. featureCounts: an efficient general purpose program for assigning sequence reads to genomic features. *Bioinformatics* 30, 7 (2014), 923–30. <https://doi.org/10.1093/bioinformatics/btt656>
 - [10] X. Liu, Y. Yu, J. Liu, C. F. Elliott, C. Qian, and J. Liu. 2018. A novel data structure to support ultra-fast taxonomic classification of metagenomic sequences with k-mer signatures. *Bioinformatics* 34, 1 (2018), 171–178. <https://doi.org/10.1093/bioinformatics/btx432>
 - [11] E. Z. Macosko, A. Basu, R. Satija, J. Nemesh, K. Shekhar, M. Goldman, I. Tirosh, A. R. Bialas, N. Kamitaki, E. M. Martersteck, J. J. Trombetta, D. A. Weitz, J. R. Sanes, A. K. Shalek, A. Regev, and S. A. McCarroll. 2015. Highly Parallel Genome-wide Expression Profiling of Individual Cells Using Nanoliter Droplets. *Cell* 161, 5 (2015), 1202–1214. <https://doi.org/10.1016/j.cell.2015.05.002>
 - [12] D. J. McCarthy, K. R. Campbell, A. T. Lun, and Q. F. Wills. 2017. Scater: pre-processing, quality control, normalization and visualization of single-cell RNA-seq data in R. *Bioinformatics* 33, 8 (2017), 1179–1186. <https://doi.org/10.1093/bioinformatics/btw777>
 - [13] Páll Melsted, A. Sina Boeshaghi, Fan Gao, Eduardo Beltrame, Lambda Lu, Kristján Eldjárn Hjorleifsson, Jase Gehring, and Lior Pachter. 2019. Modular and efficient pre-processing of single-cell RNA-seq. *bioRxiv* (2019), 673285. <https://doi.org/10.1101/673285>
 - [14] S. Parekh, C. Ziegenhain, B. Vieth, W. Enard, and I. Hellmann. 2018. zUMIs - A fast and flexible pipeline to process RNA sequencing data with UMIs. *Gigascience* 7, 6 (2018). <https://doi.org/10.1093/gigascience/giy059>
 - [15] T. Smith, A. Heger, and I. Sudbery. 2017. UMI-tools: modeling sequencing errors in Unique Molecular Identifiers to improve quantification accuracy. *Genome Res* 27, 3 (2017), 491–499. <https://doi.org/10.1101/gr.209601.116>
 - [16] Avi Srivastava, Laraib Malik, Tom Smith, Ian Sudbery, and Rob Patro. 2019. Alevin efficiently estimates accurate gene abundances from dscRNA-seq data. *Genome Biology* 20, 1 (2019), 65. <https://doi.org/10.1186/s13059-019-1670-y>
 - [17] Margaret A. Taub, Hector Corrada Bravo, and Rafael A. Irizarry. 2010. Overcoming bias and systematic errors in next generation sequencing data. *Genome Medicine* 2, 12 (2010), 87. <https://doi.org/10.1186/gm208>
 - [18] Y. Yu, D. Belazzougui, C. Qian, and Q. Zhang. 2018. Memory-Efficient and Ultra-Fast Network Lookup and Forwarding Using Othello Hashing. *IEEE/ACM Transactions on Networking* 26, 3 (2018), 1151–1164.
 - [19] Y. Yu, J. Liu, X. Liu, Y. Zhang, E. Magner, E. Lehnert, C. Qian, and J. Liu. 2018. SeqOthello: querying RNA-seq experiments at scale. *Genome Biol* 19, 1 (2018), 167. <https://doi.org/10.1186/s13059-018-1535-9>
 - [20] G. X. Zheng, J. M. Terry, P. Belgrader, P. Ryvkin, Z. W. Bent, R. Wilson, S. B. Ziraldo, T. D. Wheeler, G. P. McDermott, J. Zhu, M. T. Gregory, J. Shuga, L. Montesclaros, J. G. Underwood, D. A. Masquelier, S. Y. Nishimura, M. Schnall-Levin, P. W. Wyatt, C. M. Hindson, R. Bharadwaj, A. Wong, K. D. Ness, L. W. Beppu, H. J. Deeg, C. McFarland, K. R. Loeb, W. J. Valente, N. G. Ericson, E. A. Stevens, J. P. Radich, T. S. Mikkelsen, B. J. Hindson, and J. H. Bielas. 2017. Massively parallel digital transcriptional profiling of single cells. *Nat Commun* 8 (2017), 14049. <https://doi.org/10.1038/ncomms14049>