UMIche: A platform for robust UMI-centric simulation and analysis in bulk and single-cell sequencing

Jianfeng Sun[1,2,#], Shuang Li[3], Stefan Canzar[4] & Adam P. Cribbs[1,2]

[1] Botnar Research Centre, Nuffield Department of Orthopaedics, Rheumatology and Musculoskeletal Sciences, National Institute of Health Research Oxford Biomedical Research Unit (BRU), University of Oxford, Oxford, UK
[2] Oxford Centre for Translational Myeloma Research, University of Oxford, Oxford OX3 7LD, UK
[3] Gene Center, Ludwig-Maximilians University of Munich, Munich, Germany
[4] Faculty of Informatics and Data Science, University of Regensburg, 93053 Regensburg, Germany

[#] Correspondence: jianfeng.sun@ndorms.ox.ac.uk

Abstract

Unique molecular identifiers (UMIs) have actively been utilised by various RNA sequencing (RNA-seq) protocols and technologies to remove polymerase chain reaction (PCR) duplicates, thus enhancing counting accuracy. However, errors during sequencing processes often compromise the precision of UMI-assisted quantification. To overcome this, various computational methods have been proposed for UMI error correction. Despite these advancements, the absence of a unified benchmarking and validation framework for UMI deduplication methods hinders the systematic evaluation and optimisation of these methods. Here, we present UMIche, an open-source, UMI-centric computational platform designed to improve molecular quantification by providing a systematic, integrative, and extensible framework for UMI analysis. Additionally, it supports the development of more effective UMI deduplication strategies. We show that through its integration of a broad spectrum of UMI deduplication methods and computational workflows, UMIche significantly advances the accuracy of molecular quantification and facilitates the generation of high-fidelity gene expression profiles.

Key words: UMI, UMI deduplication, PCR duplicates, Molecular quantification, Genome analysis

Introduction

RNA sequencing (RNA-seq) technologies have been advanced through optimised sequencing protocols that enable the generation of high-quality transcriptional expression profiles [1–3]. Central to these optimisations are the accurate sequencing of nucleotides [4] and the effective quantification of reads [5], which support both sequence-based [6,7] and count matrix-based studies [8,9]. Accurate molecular quantification has garnered increasing attention due to its critical role in elucidating gene regulation mechanisms across diverse

cellular functions [10,11]. To achieve precise gene expression quantification, the attachment of Unique Molecular Identifiers (UMIs) to the sequencing molecule at the 3' or 5' end has been widely adopted [12]removes PCR duplicates in the majority of sequencing protocols [13–15]. The use of UMIs maximises counting accuracy while minimising complexity, particularly in cDNA libraries [13]. However, the error-prone nature of PCR amplification and sequencing processes can reduce the accuracy of UMI-based quantification [16,17]. Therefore, numerous computational methods and tools have been developed for UMI error correction [18–21].

Current strategies for UMI collapsing or deduplication predominantly rely on edit distance-dependent methods [22–26]. Notable examples, including UMI-tools [27] and UMICollapse [28], which create UMI graphs with an edit distance threshold to merge compositionally similar UMIs, assuming them to originate from the same molecule. Conversely, a few methods, such as DAUMI [29] and TRUmiCount [30], employ alternative approaches that model and evaluate the distributions or abundances of potential PCR duplicates to effectively collapse UMIs.

Despite these advancements, there remains a need for an integrated platform that supports systematic analysis, evaluates established methods, and fosters the development of new ones. The recent introduction of homoblock UMI correction approaches has allowed the leveraging of recurrence patterns in homoblocks for error correction [31,32]. Consequently, new error correction strategies, such as majority voting and set cover optimisation, have been developed [33,34]. These strategies enable the collapsing of nucleotides within dimer and trimer blocks (termed intra-molecular collapsing) before collapsing between UMIs (termed inter-molecular collapsing). However, there is a lack of systematic benchmarking for these studies.

Here, we introduce a **UMI** tool ni**che** (UMIche), a comprehensive platform that integrates multiple UMI collapsing methods and streamlines computational pipelines to facilitate the evaluation and development of strategies for removing PCR duplicates. UMIche operates at the platform level, harmonising UMI approaches from several different sequencing protocols, calculating deduplication counts, exploring features of different methods and visualising analysis results. To better support the UMI-centric analysis, we further provide a deep learning-based framework for on-demand simulation of UMI count matrices at the single-cell level. These data can then be used to generate sequencing reads to aid in benchmarking UMI collapsing tools. Overall, UMIche aims to enhance understanding of UMI collapsing by optimising the combination and use of existing methods and data resources. We also

demonstrate several use cases to illustrate how UMIche can be leveraged to achieve optimal performance in UMI deduplication.

Results

Overview of UMIche

UMIche is a UMI-centric platform designed to support the development and evaluation of UMI collapsing methods, using both external and internal simulated or experimental data across multiple sequencing contexts, such as bulk and single-cell levels. It is highly modular, featuring Python interfaces to facilitate seamless integration into other computational programs and analysis pipelines. UMIche comprises main four modules: UMI collapsing method implementation, UMI collapsing pipeline implementation, UMI count matrix implementation, and visualisation of various UMI metrics (**Figure 1**). The core of UMIche offers multifaceted functionalities for UMI deduplication at the single locus, bulk, single-cell levels.

1. **UMI collapsing modules**. We include modules to handle the deduplication of PCR artefacts tagged with two types of UMIs inclusive of monomer and trimer UMIs (**Figure 1a**). Recently, the range of methods for UMI collapsing has broadened from traditional edit distance-dependent approaches (*e.g.*, constructing UMI graphs with edit distance to assist molecular quantification) to methods that do not rely on edit distance, especially following the introduction of homoblock UMIs [31,32] (*e.g.*, the majority vote and set cover optimisation methods). To comprehensively support UMI collapsing, UMIche includes executables for both approaches. The investigation into edit distance-free methods has enhanced our understanding of how closely the number of UMI-tagged reads can approximate the original transcript count under extreme conditions. UMIche thus facilitates the exploration of the limits of UMI collapsing methodologies.
2. **UMI collapsing pipeline**. To enable systematic and rapid analysis, UMIche provides workflows running at the pipeline level (**Figure 1b**). To harmonise experimental and simulated sequencing protocols, UMIche standardises input data formats by accepting read alignments in BAM file format [35] or converting FastQ reads to BAM files. To minimise the workload within Python scripts, we implemented a YMAL-dependent [36] analytical workflow, streamlining the initialisation of parameters required for both upstream and downstream analyses.
3. **Simulated ground-truth count matrix** . For single-cell studies, generating count matrices with ground-truth labels is essential for simulating reads, which are

subsequently used for in silico testing of UMI collapsing methods or sequencing protocols (Figure 1c). To facilitate this, we implemented a deep learning-based framework, DeepConvCVAE, to provide users with count matrices on-demand. With a GPU support, this framework operates at high speed, efficiently handling the substantial data load of single-cell datasets. Users can retrain the deep learning model using their own data or generate a count matrix based on a built-in model for general testing purpose. Detailed descriptions of this framework are provided in the subsequent sections.

4. **Visualisation**. This module is designed to generate bar and line plots to facilitate the interpretation of analysis results (Figure 1d). It visualises deduplicated molecule counts and gene expression levels measured by different UMI collapsing methods. Additionally, we offer a graph plotting service to visually represent different UMI clusters. This feature aids in understanding how various UMIs are depicted in a UMI graph and partitioned into distinct clusters. Such visualisations can helps study the trajectories of tagging molecules (e.g., UMIs) across multiple PCR cycles and provide insights into how errors in UMIs evolve under varying conditions, such as increasing sequencing depths, higher down-sampling, and extending PCR cycle.

Streamlining analysis with pipelines in UMIche

In UMIche, we have organised functionalities into specific pipelines to streamline the analysis process. There are two main pipelines: one dedicated to deduplicating UMIs (Step 3 in Figure 2a) and the other focused on UMIs trajectories (Step 4 in Figure 2a). The deduplication pipeline provides standard deduplication results, whereas the trajectory pipeline offers insights that enhance methodologies and improve the performance of UMI deduplication methods. Both pipelines are designed to be managed uniformly, with consistent parameter configuration and file reformatting, to minimise user effort.

We propose strategies for both inter-molecular collapsing (collapsing between different UMIs) and also intra-molecular collapsing (collapsing of nucleotides within UMIs). Inter-molecular collapsing is used to eliminate PCR artefacts by collapsing UMIs, while homotrimer UMIs facilitate intra-molecular collapsing when redundant signals are detected between nucleotides in trimer blocks. Interestingly, intra-molecular collapsing can be incorporated at multiple stages of the inter-molecular collapsing process, maximising the performance gains in UMI deduplication results and providing new insight into UMI collapsing strategies.

The pipelines in UMIche automate these processes, facilitating the discovery of new knowledge using both simulated and experimental short-read and long-read data in bulk

RNA-seq and scRNA-seq conditions. In the following sections, we present several use cases to illustrate how UMIche pipelines advance UMI deduplication performance.

Case 1: Improving molecular deduplication computational strategies using homotrimer UMIs

Our efforts to accurately measure transcriptional expression profiles have led to the development of homoblock UMIs, which surpass the limitations of current computational demultiplexing approaches. This novel UMI deduplication method combines experimentally identifiable errors within the UMI sequence with computation-based sample demultiplexing, thereby enhancing UMI duplication. The introduction of homotrimer UMIs has enabled the use of a "majority vote" strategy and the implementation of a set cover optimisation technique, thanks to recurring nucleotides patterns. Despite their potential, a thorough comparative analysis of these approaches is needed to identify the most efficient method for specific sequencing contexts and to inform the development of future improvements.

To address this, we first review the process of removing PCR duplicates using homotrimer UMIs with majority vote and the set cover optimisation (**Figure 3**). The majority vote demultiplexing process involves two steps: (*i*) obtaining monomer UMIs by collapsing blocks of homotrimer UMIs (which we refer to as intra-molecular collapsing) and (*ii*) removing homotrimer UMIs according to repeated monomer UMIs (which we refer to as inter-molecular collapsing) (**Figure 3a and 3c**). The set cover demultiplexing process involves three steps: (*i*) splitting homotrimer UMIs into monomer UMIs, (*ii*) covering highly similar homotrimer UMI with their split monomer UMIs, and (*iii*) removing homotrimer UMIs according to repeated monomer UMIs (**Figure 3a and 3b**). It remains unclear whether further improvements in UMI deduplication performance can be achieved by deduplicating those retained homotrimer UMIs. As shown in **Figure 3d**, we are interested in utilising UMIche to perform the second round of intra-molecular collapsing of homotrimer UMIs and apply external graph-based UMI collapsing methods, built on top of the retained homotrimer UMIs.

1. Boosting UMI deduplication effects by altering input materials to set cover optimisation

We first investigated whether alterations in the content of monomer UMIs input to set cover optimisation could lead to different molecular counting results. During the process of splitting homotrimer UMIs into monomer UMIs, we compared UMI deduplication performance using two strategies. The first strategy, *spMV*, involves splitting monomer UMIs from homotrimer UMIs by resolving nucleotide heterogeneity obviated in each trimer block using majority vote (**Figure 3a**). The second strategy, *spALL*, splits monomer UMIs without addressing

nucleotide heterogeneity conflicts in trimer blocks. Our findings indicate that using set cover optimisation, the UMI deduplication performance with the *spALL* strategy outperforms that with the *spMV* strategy across a range of error rates (**Figure 3c**). Notably, the *spALL* strategy closely approximates the actual number of molecules, especially under conditions of high error rates. This makes the *spALL* strategy particularly useful for error-prone sequencing technologies or challenging sequencing conditions.

2. Comparison between majority voting and set cover optimisation

Next, we investigated whether the homotrimer UMIs resulting from the first-stage deduplication process could be further deduplicated in the second-stage process using external computational tools (e.g., UMI-tools). to select the most common nucleotide in each trimer block, with a random nucleotide chosen. To achieve this, we applied intra-molecular collapsing, combining three nucleotides in trimer blocks into monomer UMIs, as input for the well-established graph-based methods in UMI-tools (**Figure 3a and 3b**). The majority vote strategy was used if all three were different. This intra-collapsing strategy, termed *cbMV,* was used for the second round of UMI deduplication. We also developed another strategy, *cbRAN*, where a nucleotide is randomly selected from each trimer block without majority voting. Our results demonstrate that incorporating external computational tools significantly improves counting accuracy (**Figure 3c and 3d**), with the *cbMV* strategy yielding better results than *cbRAN*. This improvement is attributed to the inability of random nucleotide selection to correct errors effectively, particularly under high sequencing error rates. Our data suggests that majority voting can be progressively applied to enhance UMI deduplication across multiple stages. A comparison of methods within UMI-tools indicates that the Directional method remains the most accurate for molecular counting (**Figure 3e**), consistent with the main finding from previous studies [27].

3. Understanding the influence of monomer UMI splitting strategies on UMI deduplication effects

We then investigated how different UMI splitting strategies influence the performance of UMI deduplication. We calculated the accumulated number of homotrimer or monomer UMIs resolved or not resolved by the set cover optimisation technique. A group of homotrimer UMIs is considered resolved during the deduplication process if a monomer UMI, identified by the greedy algorithm as having the maximum count (see section *Methods* for greater details), is one of the UMIs collapsed from these homotrimer UMIs (**Figure 5a**). Otherwise, this UMI is counted as not resolved.

Our findings indicate that using UMIs split by the *spMV* strategy results in a similar deduplicated count to that obtained by the *spALL* strategy at low and moderate error rates.

However, at high error rates, the *spMV* strategy leads to a significant increase in the deduplicated count (**Figure 5b**), accompanied by a sharp decrease in the number of homotrimer UMIs detected by the set cover optimisation for removal (**Figure 5c**). The final deduplicated count (or the accumulated number of removed homotrimer UMIs) is proportionally correlated with the number of the input monomer UMIs split by *spMV* and *spALL* (**Figure 5d**). Specifically, a higher number of input monomer UMIs correspond to more homotrimer UMIs, which are likely a result of PCR amplification (**Figure 5e and 5f**). One possible explanation is that the monomer UMIs split by *spMV* constitute a subset of all possible monomer UMIs split by *spALL*, making them less representative of homotrimer UMIs. This subset nature of monomer UMIs split by *spMV* reduces the likelihood of covering homotrimer UMIs effectively, thereby impacting the overall deduplication process.

Case 2: DeepConvCVAE for simulating cell type-specific gene expression matrix for modelling UMI counts

The recent surge of interest in scRNA-seq stems from its ability to elucidate the correlation between genetics and diseases at the level of individual cell types with greater granularity [37–39]. To discern the cell-specific gene expression landscapes, numerous scRNA-seq analysis tools have emerged. The reliability of these tools in practical applications hinges on their evaluation against realistically labelled data [40,41]. Consequently, many simulators have been developed to ensure the availability of ground truth data [42–44]. Most of these simulations are based on statistical inference techniques, using well-estimated parameters of probability distributions to characterise given scRNA-seq data [45,46]. However, statistical inference methods have two limitations: long training time and a lack of denoising strategies.

Firstly, methods in this category typically lack technical support from computing acceleration technologies, such as graphics processing units (GPUs), particularly when compared to deep learning computing libraries. However, the practical utility of simulated data varies across cell, tissue, and disease types, potentially necessitating model retraining to accommodate a range of application conditions. On the other hand, the inherent variability of complex biological systems often eludes capture by rigid statistical inference methods that rely solely on fixed probability distribution assumptions [47,48]. This limitation may constrain their ability to effectively denoise data exhibiting multimodal distributions.

To address these limitations, we utilised a variant of the variational autoencoder (VAE) [49], the conditional VAE (CVAE) [50,51], to simulate transcriptomics data conditioned on cell types. The CVAE allows for training simulators on highly dimensional, yet sparse single-cell gene expression data at a significantly accelerated speed. Additionally, it integrates various

types of neural networks into the parameterized statistical inference process, offering greater flexibility than purely statistical inference methods for denoising multimodally distributed data.

To train a CVAE-based single-cell gene expression simulator, we construct a log-likelihood term $\ell(\theta, \varphi, \mathbf{x}, \mathbf{y})$ as the optimisation objective

$$\ell(\theta, \varphi, \mathbf{x}, \mathbf{y}) \geq \sum_{i=1}^{n} \mathbb{E}_{\mathbf{z}\sim q_\varphi}\left[\ln p_\theta(\mathbf{x}^{(i)}|\mathbf{z}, \mathbf{y}^{(i)})\right] - KL(q_\varphi\left(\mathbf{z}|\mathbf{x}^{(i)}, \mathbf{y}^{(i)}\right)||p_\theta(\mathbf{z}|\mathbf{y}^{(i)})))$$

where q is the encoder (the inference model) paramaterised φ to construct the statistical terms of a latent variable $\mathbf{z}$ using the gene expression values of a cell $\mathbf{x}^{(i)}$ corresponding to type $\mathbf{y}^{(i)}$. The decoder (the generative model) parameterized by θ, generates new single-cell data samples using the constructed $\mathbf{z}$. KL represents the Kullback–Leibler (KL) divergence [52] quantifying the difference between the two distributions q and p from their respective generative and inference processes. The full description of the modelling process can be found in section *Methods*.

The use of the latent variable $\mathbf{z}$ is popular in single-cell studies for addressing data denoising, perturbation, and batch effects. To better fit distributions q and p, we employed convolutional neural networks (CNNs) [53] in both the encoder and the decoder due to their reduced parameters and fast training speed compared to fully connected neural networks [54] (**Figure 6**). However, CNNs can be replaced by any other types of neural networks, providing flexibility for users.

The trained model simulated gene expression data of 2000 cells for each of 11 cell clusters (**Figure 7**). After 10 training epochs, the simulated data has been shown discriminative between different cell clusters observed in PCA [55], TSNE [56], and UMAP [57] plots. We also integrated the overall framework of DeepConvCVAE into UMIche, for data simulation using trained models, data retraining, and framework modifications for other single-cell objective-specific studies.

Conclusions

In this work, we introduced UMIche, a comprehensive platform specifically designed for UMI-centric analysis, supporting the development and validation of UMI collapsing methods using both simulated and experimental bulk RNA-seq or scRNA-seq data. It offers four main modules: UMI collapsing method implementation, UMI collapsing pipeline implementation, UMI count matrix implementation, and visualization of UMI metrics. These modules are implemented to allow easy integration into existing computational programs and analysis pipelines. We present the architecture and functionalities of UMIche, along with case studies

demonstrating its utility in advancing UMI deduplication performance in single-cell sequencing studies. UMIche has important implications for generating novel methodological insights by integrating recent advances in utilizing homotrimer UMIs for deduplicating PCR artefacts. This integration leverages error-tolerant recurrence patterns, offering a promising approach to enhance UMI collapsing methods and safeguard the accessibility to highly accurate gene expression profiles. We expect the use of UMIche to bridge measuring genomic observations and understanding biological mechanisms.

Method

UMI collapsing

UMI collapsing plays an important role in error correction and molecular quantification after sequencing [27]. The development of homotrimer UMIs makes it possible to collapse within UMIs before/after collapsing between homotrimer UMIs [31]. However, it remains murky as to how much we can gain in improving UMI collapsing effects from their combination. In UMIche, we constructed a homotrimer UMI-based collapsing scheme by combining both intra-UMI collapsing and inter-UMI collapsing methods, which is formulated below.

Step 1: Splitting homotrimer UMIs into monomer UMIs.

Let $S^T = \{s_i^T \mid i = 1,2,\dots,n \ and \ n \in \mathbb{N}\}$ be the set of n homotrimer UMIs of length l observed at a single genomic locus. Let $W = \{A, T, C, G\}$ be the set of four types of nucleotides. For any homotrimer UMI $s_i^T \in S^T$, its jth trimer block is represented by set $b_{i,j}^T = \{x_p | x_p \in W\}$ for which $p \in P = \{1,2,3\}$ is a set of positional coordinates of elements in $b_{i,j}^T$. The set of all of its trimer blocks B_i^T is denoted as

$$\bigcup_{j \in \{1,2,\dots,\frac{l}{3}\}} b_{i,j}^T$$

Consider function $f: P \to b_{i,j}^T$ for projecting a positional coordinate into a nucleotide contained in $b_{i,j}^T$. We aim to replace a trimer block $b_{i,j}^T$ with a new set $\bar{b}_{i,j}^T$, in which heterogeneity among nucleotides is obviated by the majority vote strategy. We retain $b_{i,j}^T$ for $\bar{b}_{i,j}^T$ if given $\forall x \in b_{i,j}^T$, $\nexists p_1 \neq p_2$, $p_1 \in P$ and $p_2 \in P$, such that $f(p_1) = f(p_2) = x$. Otherwise, we construct $\bar{b}_{i,j}^T = \{x\}$. Accordingly, B_i^T turns to $\bar{B}_i^T = \bigcup_{j \in \{1,2,\dots,\frac{l}{3}\}} \bar{b}_{i,j}^T$.

Let F be a function for splitting a homotrimer UMI into monomer UMIs. Specifically, it uses as input the set of trimer blocks of a homotrimer UMI, takes turn selecting a nucleotide from each homotrimer block, and finally merges those selected nucleotides into monomer UMIs.

According to whether or not to apply f to each trimer block, utilising F to obtain split monomer UMIs results in two respective sets of sequences.

spALL. If f is not applied, we obtained a set containing all possible monomer UMIs from a homotrimer UMI. We term this strategy *split-to-all* (*spALL*). The resulting set of its split n_i monomer UMIs is phased to be

$$F\left(B_i^T\right) = \left\{s_{i,n_i}^M \mid 1,2,\ldots,n_i \; and \; n_i \in \mathbb{N}\right\}$$

spMV. Let $\mathcal{B}_i^T = \left\{f(b_{i,j}^T) \mid j = 1,2,\ldots,\frac{n}{3} \; and \; n \in \mathbb{N}\right\}$ denote the set of heterogeneity-obviated trimer blocks of a homotrimer UMI $\forall s_i^T \in S^T$. Thus, its split m_i monomer UMIs comprise the following set

$$F\left(\mathcal{B}_i^T\right) = \left\{s_{i,m_i}^M \mid 1,2,\ldots,m_i \; and \; m_i \in \mathbb{N}\right\}$$

We term this strategy *split-by-majority-vote* (*spMV*).

Step 2: Homotrimer UMI deduplication (deduplication stage 1)

1. Deduplication using set cover optimisation. Let g be the set cover optimisation function [58,59] for removing PCR artefacts using a set of homotrimer UMIs at a given genomic locus. The deduplicated count c at the locus is written as

$$c = g(S^T)$$

To solve function g, we use the greedy algorithm [33], which iteratively finds subsets of homotrimer UMIs from set S^T through their overlapped monomer UMIs with the maximum counts. At each iteration, specifically, the greedy algorithm attempts to identify a monomer UMI s^M that can account for a subset of the largest number of homotrimer UMIs. The homotrimer UMIs are thought to derive from the same original homotrimer UMI at the locus, which are then popped out of S^T in the end of each iteration. The remaining homotrimer UMIs in S^T are iteratively deduplicated likewise, until none of monomer UMIs is found to account for more than one homotrimer UMI. Suppose that there are k iterations to identify these k monomer UMIs, each accounting for at least two homotrimer UMIs. There are r monomer UMIs left, each corresponding to only one homotrimer UMI. Hence, the final deduplicated count c is the sum of the both, like that,

$$c = k + r$$

This implies that the problem of identifying PCR duplicates at the level of homotrimer UMIs is converted to the problem of settling the optimised coverage between sets at the level of monomer UMIs.

Let $S^K = \{n_i | n_i \in \mathbb{N}\ for\ i = 0,1,2,\dots,k\}$ be the set of the numbers of homotrimer UMIs in k subsets $(S_1^T, S_2^T, \dots, S_k^T)$ for which monomer UMIs account after k iterations. Note that $n_0 = 0$. In each iteration, the identified homotrimer UMIs are collected into $S_k^T = \{s_i^T | s_i^T \in S^T\ for\ i = n_{k-1}+1, n_{k-1}+2, \dots, n_k\}$. The union of the k subsets is a collection of all homotrimer UMIs in which at least two can be accounted for by one monomer UMI, denoted as

$$S_K^T = \bigcup_{i \in \{1,2,\dots,k\}} S_i^T$$

Let $S_K^M = \{s_i^M | i = 1,2,\dots,k\}$ be the set of these monomer UMIs after k iterations. The remaining homotrimer UMIs, whose split monomer UMIs are unable to be overlapped with other homotrimer UMIs, are stored in set $S_R^T = S^T - S_K^T$. Let $S_R^M = \{s_i^M | i = 1,2,\dots,r\}$ be the set of these monomer UMIs after k iterations.

Definition 1.1. Let $u: S_K^T \rightarrow S_K^M$ be a **many-to-one** mapping from homotrimer UMIs to monomer UMIs. Original homotrimer UMIs are considered to have duplicates if given $\forall s^M \in S_K^M$, $\exists s^T \in S_K^T$, such that

$$u(s^T) = s^M$$

Definition 1.2. Let $v: S_R^T \rightarrow S_R^M$ be a **one-to-one** mapping from homotrimer UMIs to monomer UMIs. Original homotrimer UMIs are not considered to have duplicates if given $\forall s^M \in S^M$, $\nexists s_1^T \neq s_2^T$, $s_1^T \in S^T$ and $s_2^T \in S^T$, such that

$$v(s_1^T) = v(s_2^T) = s^M$$

2. Deduplication using majority voting. Let h be the majority voting function for removing PCR artefacts using a set of homotrimer UMIs at a given genomic locus. The deduplicated count c at the locus is written as

$$c = h(S^T)$$

To solve function h, we begin by obtaining a sole monomer UMI that corresponds to a homotrimer UMI. This is achieved by collapsing a trimer block $b_{i,j}^T$ into a single nucleotide by the majority vote strategy and then concatenating all single nucleotides into a monomer UMI. S. We collapse $b_{i,j}^T$ into the recurring nucleotide x if given $\forall x \in b_{i,j}^T$, $\exists p_1 \neq p_2$, $p_1 \in P$ and $p_2 \in P$, such that $f(p_1) = f(p_2) = x$. Otherwise, a random sampling operation is applied to obtain a nucleotide $b_{i,j}^T \overset{R}{\leftarrow} x$. Suppose that $S^M = \{s_i^M | i = 1,2,\dots,c\}$ is the set of unrepeated monomer UMIs collapsed from homotrimer UMIs in S^T. In this case, the number of the unrepeated monomer UMIs is said to be the deduplicated count c.

Let $S^C = \{n_i | n_i \in \mathbb{N}\ for\ i = 0,1,2, \dots, c\}$ be the set of the numbers of homotrimer UMIs in c subsets ($S_1^T, S_2^T, \dots, S_c^T$). Note that $n_0 = 0$. Each subset $S_c^T = \{s_i^T | s_i^T \in S^T\ for\ i = n_{c-1} + 1, n_{c-1} + 2, \dots, n_c\}$ contains homotrimer UMIs that are all collapsed into a common monomer UMI. The union of the c subsets is set S^T, denoted as

$$S^T = \bigcup_{i \in \{1,2,\dots,c\}} S_i^T$$

Definition 2.1. Let $w: S^T \rightarrow S^M$ be a **many-to-one** mapping from homotrimer UMIs to monomer UMIs. Original homotrimer UMIs are considered to have duplicates if given $\forall s^M \in S^M$, $\exists s^T \in S^T$, such that

$$w(s^T) = s^M$$

Step 3: Retaining homotrimer UMIs.

1. Set cover optimisation. We randomly sample homotrimer UMIs from the k subsets ($S_1^T, S_2^T, \dots, S_k^T$), respectively. This forms a new set D_K^T, given by

$$D_K^T = \{s_i^T | S_i^T \overset{R}{\leftarrow} s_i^T\ for\ i = 1,2, \dots, k\}$$

Then, together with set S_R^T, $D^T = D_K^T + S_R^T$ is taken as the set that comprises homotrimer UMIs that are one-to-one mapped to all identified monomer UMIs.

2. Majority voting. We randomly sample homotrimer UMIs from the c subsets ($S_1^T, S_2^T, \dots, S_c^T$), respectively, leading to

$$D^T = \{s_i^T | S_i^T \overset{R}{\leftarrow} s_i^T\ for\ i = 1,2, \dots, c\}$$

D^T is said to be the set that comprises homotrimer UMIs that are one-to-one mapped to all monomer UMIs collapsed by the majority vote strategy.

Step 4: Collapsing homotrimer UMI

1. Collapsing blocks by the majority vote strategy (*cbMV*). We define function q to concatenate a set of nucleotides into a sequence. Let $S_i^E = \{e_j | e_j \in b_{i,j}^T\ for\ j = 1,2, \dots, \frac{l}{3}\}$ be a set of nucleotides, each added by collapsing a trimer block $b_{i,j}^T$ of any homotrimer UMI $s_i^T \in D^T$ into a single nucleotide by the majority vote strategy. e_j is nucleotide x if given $\forall x \in b_{i,j}^T$, $\exists p_1 \neq p_2$, $p_1 \in P$ and $p_2 \in P$, such that $f(p_1) = f(p_2) = x$. Otherwise, a random sampling operation is applied, such that $e_j = b_{i,j}^T \overset{R}{\leftarrow} x$. Subsequently, we build set $D^M =$

$\{q(S_i^E)|i = 1,2,\dots,c\}$, which adds each concatenated monomer UMI for each homotrimer UMI $s_i^T \in D^T$.

2. Collapsing blocks randomly (*cbRAN*). We define function q to concatenate a set of nucleotides into a sequence. Let $S_i^E = \{e_j | e_j \in b_{i,j}^T \; for \; j = 1,2,\dots,\frac{l}{3}\}$ be a set of nucleotides, each added by collapsing a trimer block $b_{i,j}^T$ of any homotrimer UMI $s_i^T \in D^T$ into a single nucleotide through the random sampling strategy, such that $e_j = b_{i,j}^T \overset{R}{\leftarrow} x$. Then, we build set $D^M = \{q(S_i^E)|i = 1,2,\dots,c\}$, which adds each concatenated monomer UMI for each homotrimer UMI $s_i^T \in D^T$.

Step 5: Homotrimer UMI deduplication (deduplication stage 2)

Through steps 1-4, a collection of homotrimer UMIs in S^T are deduplicated into a bundle of monomer UMIs in D^T. Finally, an external UMI deduplication method φ, which relies typically on the use of graph modelling techniques, is applied on D^T to further remove UMI deduplicates. The calculation result $\varphi(D^T)$ is deemed as the deduplication count c for this step.

UMI count matrix simulation

Let $\mathbf{x}$ be a count matrix with n cells and m genes in the following form.

$$\begin{pmatrix} x_1^{(1)} & \cdots & x_m^{(1)} \\ \vdots & \ddots & \vdots \\ x_1^{(n)} & \cdots & x_m^{(n)} \end{pmatrix}_{n \times m}$$

Cell types of the n cells are known and stored in vector $\mathbf{y} = [y_1, y_2, \dots, y_n]$ in which there are ℓ unique cell types in total. In $\mathbf{x}$, $\mathbf{x}^{(i)} = \left[x_1^{(i)}, x_2^{(i)}, \dots, x_m^{(i)}\right]$ represents expression values of m genes for cell i, whose cell type corresponds to y_i in $\mathbf{y}$. We might as well consider fitting $\mathbf{x}$ to a probability distribution function p first and then drawing data from this distribution to approximate $\mathbf{x}$, as the use of probabilistic models is usually thought of to be more generalizable and preferable to capture and discover the underlying patterns of biological systems and gene regulation mechanisms behind the variable, yet incompletely characterized biological data, such as single-cell count matrices. The goal of the probabilistic modeling process is set to estimate parameters θ from p so as to approximate $\mathbf{x}$, which is described as

$$\mathbf{x} \sim p_\theta(\mathbf{x})$$

As the effect of approximation of θ may be affected by unknown patterns that are unobservable from $\mathbf{x}$, a latent variable $\mathbf{z}$ needs to be introduced to settle this problem [60], which turns $p_\theta(\mathbf{x})$ to a conditional distribution $p_\theta(\mathbf{x}|\mathbf{z})$ from which $\mathbf{x}$ is about to be drawn. Consider the following probabilistic model [61],

$$p_\theta(\mathbf{x}) \sim \int p_\theta(\mathbf{x}, \mathbf{z})\, \mathbf{dz}$$

eq.1

According to Bayes' theorem $p_\theta(\mathbf{x}, \mathbf{z}) = p_\theta(\mathbf{x}|\mathbf{z})p_\theta(\mathbf{z}) = p_\theta(\mathbf{z}|\mathbf{x})p_\theta(\mathbf{x})$, it turns to

$$p_\theta(\mathbf{x}) = \int p_\theta(\mathbf{x}|\mathbf{z})p(\mathbf{z})\, \mathbf{dz}$$

Due to the intractability to solve the above integral [49,62], we are unable to obtain the suitable estimates of $p_\theta(\mathbf{x}|\mathbf{z})$. As for the prior distribution of $p(\mathbf{z})$, it is more practical to consider its posterior distribution $p(\mathbf{z}|\mathbf{x})$ as it is conditioned on the observations $\mathbf{x}$, i.e., the realistic posterior distribution after the model in ***eq.1*** is fed with $\mathbf{x}$. However, after the formula transformation below, it is also intractable to solve $p_\theta(\mathbf{z}|\mathbf{x})$ according to [63].

$$p_\theta(\mathbf{z}|\mathbf{x}) = \frac{p_\theta(\mathbf{x}|\mathbf{z})p(\mathbf{z})}{\int p_\theta(\mathbf{x}|\mathbf{z})p(\mathbf{z})\, \mathbf{dz}}$$

Hence, we sought to find another distribution $q_\varphi(\mathbf{z}|\mathbf{x})$ parameterized by φ to substitute $p_\theta(\mathbf{x}|\mathbf{z})$, such that,

$$q_\varphi(\mathbf{z}|\mathbf{x}) \approx p_\theta(\mathbf{z}|\mathbf{x})$$

eq.2

Reconsider the probabilistic model present in ***eq.1*** before solving $q_\varphi(\mathbf{z}|\mathbf{x})$. To estimate θ from the model, it is common to apply a log-likelihood function as follows

$$\ell(\theta, \mathbf{x}) = \sum_{i=1}^{n} \ln p_\theta(\mathbf{x}^{(i)})$$

For a given single cell is written as

$$\begin{aligned}
\ell\left(\theta, \mathbf{x}^{(i)}\right) &= \ln p_\theta\left(\mathbf{x}^{(i)}\right) \\
&= \ln \int p_\theta\left(\mathbf{x}^{(i)}, \mathbf{z}\right) \mathbf{dz} \\
&= \ln \int q_\varphi\left(\mathbf{z}|\mathbf{x}^{(i)}\right) \frac{p_\theta\left(\mathbf{x}^{(i)}, \mathbf{z}\right)}{q_\varphi\left(\mathbf{z}|\mathbf{x}^{(i)}\right)} \mathbf{dz}
\end{aligned}$$

$$
\begin{aligned}
&= \ln \mathbb{E}_{\mathbf{z}\sim q_\varphi}\left[\frac{p_\theta(\mathbf{x}^{(i)},\mathbf{z})}{q_\varphi(\mathbf{z}|\mathbf{x}^{(i)})}\right] \\
&\geq \mathbb{E}_{\mathbf{z}\sim q_\varphi}\left[\ln\frac{p_\theta(\mathbf{x}^{(i)},\mathbf{z})}{q_\varphi(\mathbf{z}|\mathbf{x}^{(i)})}\right] \\
&= \mathcal{L}(\theta,\varphi,\mathbf{x}^{(i)})
\end{aligned}
$$

$\mathcal{L}(\theta,\varphi,\mathbf{x}^{(i)})$ is known as the evidence lower bound (ELBO) function of $\ell\left(\theta,\mathbf{x}^{(i)}\right)$ [64]. After applied with Bayes' theorem, $\mathcal{L}$ is further written to be the following form

$$
\begin{aligned}
\mathcal{L}(\theta,\varphi,\mathbf{x}^{(i)}) &= \mathbb{E}_{\mathbf{z}\sim q_\varphi}\left[\ln p_\theta\left(\mathbf{x}^{(i)},\mathbf{z}\right) - \ln q_\varphi\left(\mathbf{z}|\mathbf{x}^{(i)}\right)\right] \\
&= \mathbb{E}_{\mathbf{z}\sim q_\varphi}\left[\ln p_\theta(\mathbf{x}^{(i)},\mathbf{z})\right] - \mathbb{E}_{\mathbf{z}\sim q_\varphi}\left[\ln q_\varphi\left(\mathbf{z}|\mathbf{x}^{(i)}\right)\right] \\
&= \mathbb{E}_{\mathbf{z}\sim q_\varphi}\left[\ln p_\theta(\mathbf{x}^{(i)}|\mathbf{z})p_\theta(\mathbf{z})\right] - \mathbb{E}_{\mathbf{z}\sim q_\varphi}\left[\ln q_\varphi\left(\mathbf{z}|\mathbf{x}^{(i)}\right)\right] \\
&= \mathbb{E}_{\mathbf{z}\sim q_\varphi}\left[\ln p_\theta(\mathbf{x}^{(i)}|\mathbf{z}) + \ln p_\theta(\mathbf{z})\right] - \mathbb{E}_{\mathbf{z}\sim q_\varphi}\left[\ln q_\varphi\left(\mathbf{z}|\mathbf{x}^{(i)}\right)\right] \\
&= \mathbb{E}_{\mathbf{z}\sim q_\varphi}\left[\ln p_\theta(\mathbf{x}^{(i)}|\mathbf{z})\right] + \mathbb{E}_{\mathbf{z}\sim q_\varphi}\left[\ln p_\theta(\mathbf{z})\right] - \mathbb{E}_{\mathbf{z}\sim q_\varphi}\left[\ln q_\varphi\left(\mathbf{z}|\mathbf{x}^{(i)}\right)\right] \\
&= \mathbb{E}_{\mathbf{z}\sim q_\varphi}\left[\ln p_\theta(\mathbf{x}^{(i)}|\mathbf{z})\right] - \left(\mathbb{E}_{\mathbf{z}\sim q_\varphi}\left[\ln q_\varphi\left(\mathbf{z}|\mathbf{x}^{(i)}\right)\right] - \mathbb{E}_{\mathbf{z}\sim q_\varphi}\left[\ln p_\theta(\mathbf{z})\right]\right)
\end{aligned}
$$

eq.3

Thus, altogether, the log-likelihoods of all observed gene expression values across n single cells with respect to its ELBO (the right side of the equation) are expressed as

$$
\ell(\theta,\varphi,\mathbf{x}) \geq \sum_{i=1}^{n} \mathbb{E}_{\mathbf{z}\sim q_\varphi}\left[\ln p_\theta(\mathbf{x}^{(i)}|\mathbf{z})\right] - \left(\mathbb{E}_{\mathbf{z}\sim q_\varphi}\left[\ln q_\varphi\left(\mathbf{z}|\mathbf{x}^{(i)}\right)\right] - \mathbb{E}_{\mathbf{z}\sim q_\varphi}\left[\ln p_\theta(\mathbf{z})\right]\right)
$$

eq.4

Let $\mathcal{L}_1$ be $\mathbb{E}_{\mathbf{z}\sim q_\varphi}\left[\ln p_\theta(\mathbf{x}^{(i)}|\mathbf{z})\right]$, which is a term that we expect to use to reconstruct data based on θ-parameterized distribution p conditioned on $\mathbf{z}$. Let $\mathcal{L}_2$ be $\mathbb{E}_{\mathbf{z}\sim q_\varphi}\left[\ln q_\varphi\left(\mathbf{z}|\mathbf{x}^{(i)}\right)\right] - \mathbb{E}_{\mathbf{z}\sim q_\varphi}\left[\ln p_\theta(\mathbf{z})\right]$. In fact, $\mathcal{L}_2$ is the Kullback–Leibler (KL) divergence that measures the discrepancy between two distributions [52]. Recall ***eq.2***, compared with the estimation of prior distributions of $\mathbf{z}$, it is likely more rational to estimate its posterior distributions if given the observed data $\mathbf{x}^{(i)}$. Hence, $\mathcal{L}_2$ is reformatted as

$$
\mathcal{L}_2 = KL\left(q_\varphi\left(\mathbf{z}|\mathbf{x}^{(i)}\right)||p_\theta(\mathbf{z})\right)
$$

eq.5

According to ***eq.5***, it becomes clear that a perfect estimation of the true posterior distribution $p_\theta(\mathbf{z}|\mathbf{x}^{(i)})$ can be achieved if $\mathcal{L}_2$ is 0, suggesting that a substituted distribution of $q_\varphi(\mathbf{z}|\mathbf{x}^{(i)})$ bears a very close resemblance to $p_\theta(\mathbf{z})$.

$\mathcal{L}_1$ is often referred to as the generative model (**the decoder**) to reconstruct $\mathbf{x}$, while $\mathcal{L}_2$ is often referred to as the inference model (**the encoder**) to build a Gaussian distribution of input $\mathbf{x}$. An autoencoder with such a setting, which learns a distribution from input data and use the learned distribution to simulate new data, is referred to as the variational autoencoder (VAE) [49].

Let $p_\theta(\mathbf{z})$ be a multivariate Gaussian distribution $\mathcal{N}(0, \mathrm{I})$ where I represents a standard diagonal covariance matrix. Let $q_\varphi(\mathbf{z}|\mathbf{x}^{(i)})$ be a multivariate Gaussian distribution $\mathcal{N}(\mu_\mathbf{z}^{(i)}, \Sigma_\mathbf{z}^{(i)})$ with respect to the gene expression values of cell i. $\mu_\mathbf{z}^{(i)} = \left[\mu_{\mathbf{z},1}^{(i)}, \mu_{\mathbf{z},2}^{(i)}, \dots, \mu_{\mathbf{z},\mathcal{U}}^{(i)}\right]$ is a vector that shares the same dimension of $\mathcal{U}$ to the latent variable $\mathbf{z}$. $\Sigma_\mathbf{z}^{(i)} = \mathrm{I} \odot \sigma_\mathbf{z}^{2(i)}$ ($\odot$ for element-wise multiplication) is a diagonal covariance matrix whose diagonal elements are $\sigma_{\mathbf{z},1}^{(i)}, \sigma_{\mathbf{z},2}^{(i)}, \dots, \sigma_{\mathbf{z},\mathcal{U}}^{(i)}$. Then, combined with the multivariate Gaussian function, ***eq.5*** is finally reformatted as

$$\begin{aligned}\mathcal{L}_2 &= KL(\mathcal{N}(\mu_\mathbf{z}, \sigma_\mathbf{z}) || \mathcal{N}(0, \mathrm{I})) \\ &= \frac{1}{2}\sum_{u=1}^{\mathcal{U}} (\mu_{\mathbf{z},u}^{(i)\,2} + \sigma_{\mathbf{z},u}^{(i)\,2} - 1 - \ln \sigma_{\mathbf{z},u}^{(i)\,2}) \\ &= \frac{1}{2}(\mu_\mathbf{z}^{(i)\mathrm{T}} \mu_\mathbf{z}^{(i)} + tr(\Sigma_\mathbf{z}^{(i)}) - \mathcal{U} - \ln tr(\Sigma_\mathbf{z}^{(i)}))\end{aligned}$$

eq.6

According to ***eq.6***, the inference model in the encoder is set up for learning two Gaussian distribution parameters, $\mu_\mathbf{z}$ and $\Sigma_\mathbf{z}$, from the single-cell count matrix $\mathbf{x}$. In our case, to perform mapping $q_\varphi(\mathbf{z}|\mathbf{x}^{(i)})$**:** $\mathbf{x} \rightarrow \mu_\mathbf{z}, \Sigma_\mathbf{z}$, we employed a convolutional neural network [54] and a multiple perceptron (with 1 dense layer with 16 neurons) [65] to transform input $\mathbf{x}$ into values ranging from 0 to 1 to approximate $\mu_\mathbf{z}$ and $\Sigma_\mathbf{z}$. The convolutional neural network comprises two layers, with the first placed with 16 kernels with a size of 3×3 and the second placed with 32 kernels with a size of 3×3 (**Figure 6**).

Before performing the reconstruction of single-cell gene expression data approximate to $\mathbf{x}$ in the decoder, we made use of the output of the encoder, $\mu_\mathbf{z}$ and $\Sigma_\mathbf{z}$, to build $\mathbf{z}$ taken as input to the decoder. Following the common practice, we employed the reparameterization trick, calculated by

$$\mathbf{z}^{(i)} = \mu_{\mathbf{z}}^{(i)} + \sigma_{\mathbf{z}}^{(i)} \odot \epsilon$$

where ϵ is a $\mathcal{U}$-dimension vector whose elements are drawn from another Gaussian distribution $\mathcal{N}(0, \mathrm{I})$. In our case, to avoid very small values during training, we artificially added a value $\sqrt[2]{e}$ to $\sigma_{\mathbf{z}}^{(i)}$. Then, $\mathbf{z}^{(i)}$ is transformed to

$$\mathbf{z}^{(i)} = \mu_{\mathbf{z}}^{(i)} + (\sqrt[2]{e} + \sigma_{\mathbf{z}}^{(i)}) \odot \epsilon$$

In the decoder, we placed the same neural network architecture with the encoder (**Figure 6**). As the decoder requires the reconstructed $\mathbf{x}$ to be as more similar as possible, the loss of the decoder is set to minimise the cross entropy (*CE*) [66] between the ground-truth and simulated gene expression values. The loss of the encoder is $\mathcal{L}_2$ itself (***eq.6***) as the KL divergence aims to contract the difference between a posterior distribution of $\mathbf{z}$ under q_φ, and its prior distribution under p_θ. Thus, the total loss of the model is the joint loss of the encoder and the decoder, given by

$$\boldsymbol{loss} = CE(\mathbf{x}, \mathbf{x}_{\boldsymbol{decoder}}) + \mathcal{L}_2$$

eq.7

We used the root mean square propagation technique (RMSProp) [67] to optimize the model and set the batch size to 32 cells. Using convolutional neural networks as both the encoder and the decoder is the significant reduction of parameters required to train and simulate the single-cell gene expression data. Although the above model is capable of generating gene expression data, it is still unable to produce cell type-specific gene expression data. Similar to introducing the latent variable that observable data $\mathbf{x}$ is conditioned on, $\mathbf{y}$ can be seen as another condition further introduced to the model in ***eq.4***, termed CVAE [50,51] and rewritten as

$$\ell(\theta, \varphi, \mathbf{x}, \mathbf{y}) \geq \sum_{i=1}^{n} \mathbb{E}_{\mathbf{z}\sim q_\varphi}\left[\ln p_\theta(\mathbf{x}^{(i)}|\mathbf{z}, \mathbf{y}^{(i)})\right] - (\mathbb{E}_{\mathbf{z}\sim q_\varphi}\left[\ln q_\varphi\left(\mathbf{z}|\mathbf{x}^{(i)}, \mathbf{y}^{(i)}\right)\right] - \mathbb{E}_{\mathbf{z}\sim q_\varphi}\left[\ln p_\theta(\mathbf{z}|\mathbf{y}^{(i)})\right])$$

To this end, a simple operation further required in coding is the concatenation of $\mathbf{x}$ and $\mathbf{y}$ as the input to both the encoder and the decoder. As a result, the cell types specific to their relative gene expression data are together trained and learned by the VAE model.

Training the VAE model on the 68k PBMC dataset

We pulled from 10xGenomics the *68k PBMCs Donor A* dataset with 32,738 genes expressed in 68,579 cells [68]. Each cell has at least one expressed gene. We left with 20,387 genes after removing genes with zero expressions in all cells. Next, genes that are

expressed in less than 20 cells are also removed. The cell clusters were detected from barcodes using the CellRanger 1.1 pipeline and obtained under tab "*Zheng et al.*" from https://github.com/10XGenomics/single-cell-3prime-paper/tree/master/pbmc68k_analysis. Then, the processed dataset was split into 5 folds using the stratified shuffle split method to ensure that cells from different cell clusters (CD8+ Cytotoxic T, CD8+/CD45RA+ Naive Cytotoxic, CD4+/CD45RO+ Memory, CD19+ B, CD4+/CD25 T Reg, CD56+ NK, CD4+ T Helper2, CD4+/CD45RA+/CD25- Naive T, CD34+, Dendritic, and CD14+ Monocyte) were evenly allocated in each fold (**Supplementary Figs. 6-8**). Data were normalized into 0-1 prior to training. The parameters of the final model were optimized on each training fold. All operations above were automatically performed with Tensorflow [69] and Keras [70], the trained model DeepConvCVAE is made available in UMIche.

Data and code availability

The standalone package of UMIche is made available at https://github.com/cribbslab/umiche. The data used in this paper can be accessed through https://cribbslab.github.io/umiche.

The *68k PBMCs Donor A* dataset can be downloaded through tab "*Zheng et al.*" at https://www.10xgenomics.com/resources/datasets and the information about their cell types can be found at https://github.com/10XGenomics/single-cell-3prime-paper/tree/master/pbmc68k_analysis.

Declaration of competing interest

A.P.C is listed as an inventor on several patents filed by Oxford University Innovations concerning single-cell sequencing technologies. The other authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Author contribution

J.S. conceived this research. J.S. performed the data analysis, plotted the data, developed the software, and wrote the manuscript. J.S., S.L., and S.C. contributed to the development of the set coverage optimisation method and computational analyses. J.S. and A.P.C. revised the manuscript and acquired funding. All authors have given approval to the final version of the manuscript.

Acknowledgement

This work was financially supported by the Medical Research Council (MRC) career development fellowship (MR/V010182/1).

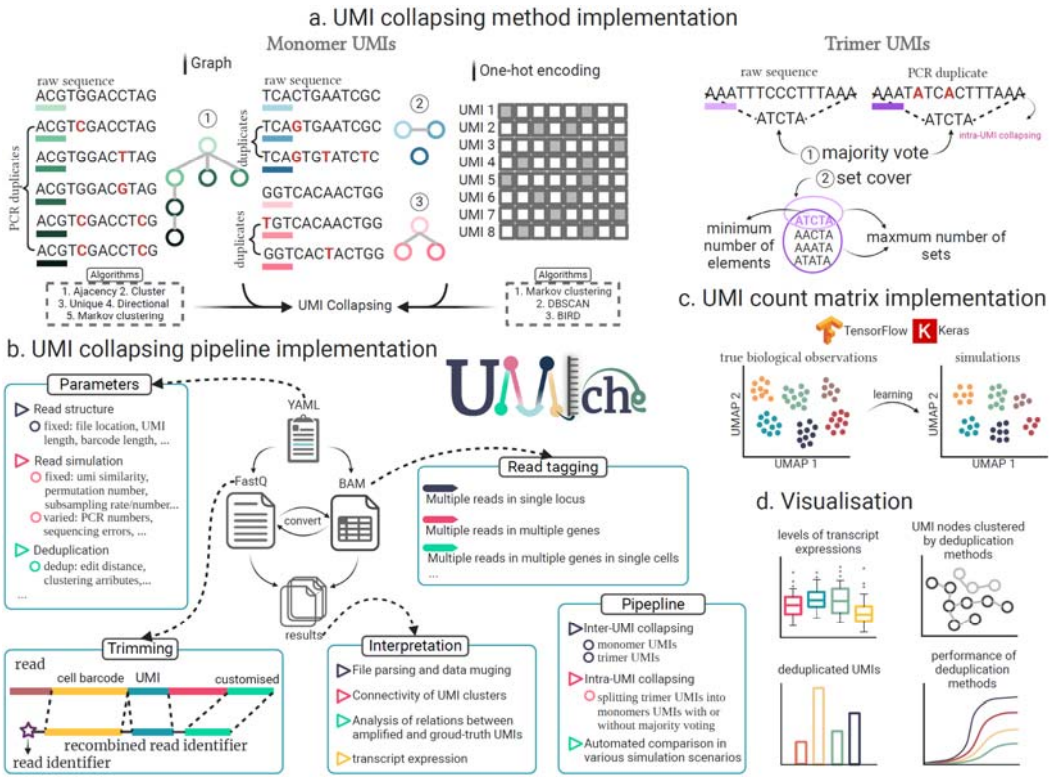


Figure 1. Overview of UMIche. It contains UMI collapsing method implementation (**a**), UMI collapsing pipeline implementation (**b**), UMI count matrix implementation (**c**), and visualisation of UMI deduplication metrics (**d**). All methods in mclUMI and UMI-tools have been re-implemented for deduplicating monomer UMIs. Majority voting and set coverage optimisation specialised for deduplicating UMIs of homoblocks have been made available. Together with 3 Euclidean distance-based clustering methods that process the one-hot encodings of UMIs sequences, there are a total of 12 UMI deduplication methods available in UMIche. We provide a module to simulate single-cell UMI count matrix on-demand with ground-truth labels, facilitating the examination of the performance of UMI deduplication methods. We built a few computational pipelines following the same protocol to pre- and post-process sequencing data, including read tagging, trimming UMIs from reads, collapsing UMIs, etc. This feature allows a rapid benchmark of performance of UMI deduplication tools on large-scale data. A plotting module is provided to readily visualise the performance of multiple UMI deduplication tools.

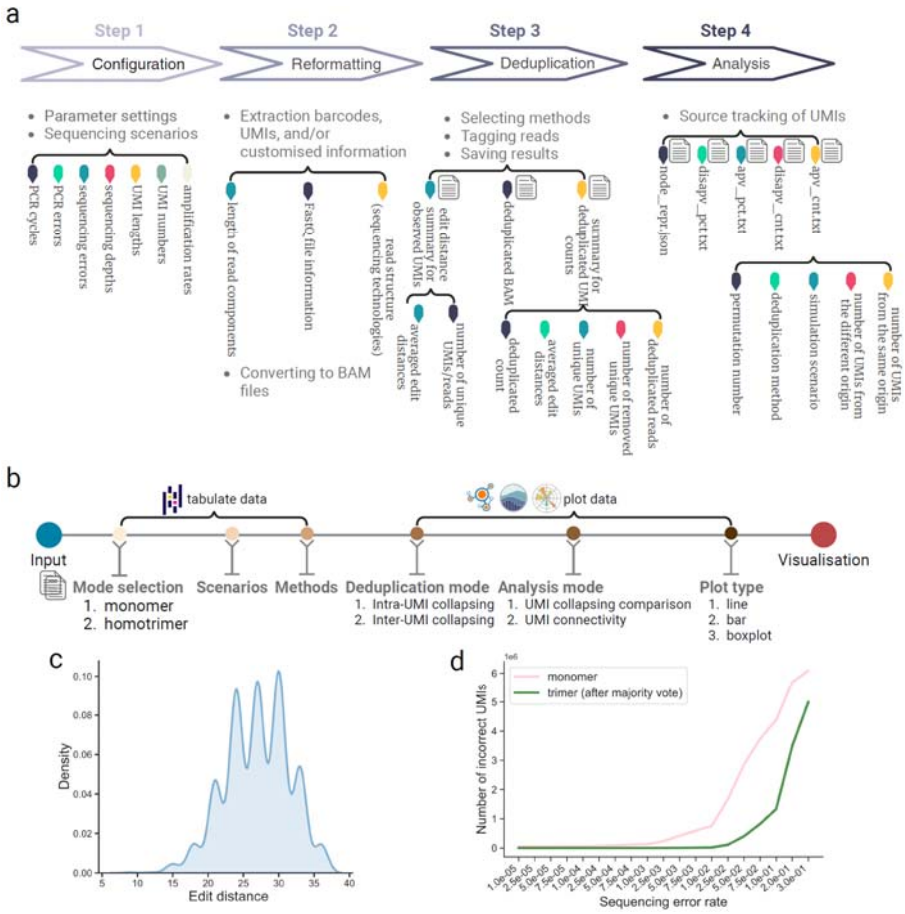


Figure 2. Workflow of the pipeline module in UMIche. **a**, typical modules in a UMIche pipeline. **b**, processing and visualisation of deduplication results. **c**, an example of visualising edit distances between simulated UMIs using a UMIche pipeline. **d**, an example of counting numbers of error monomer and trimer UMIs using a UMIche pipeline.

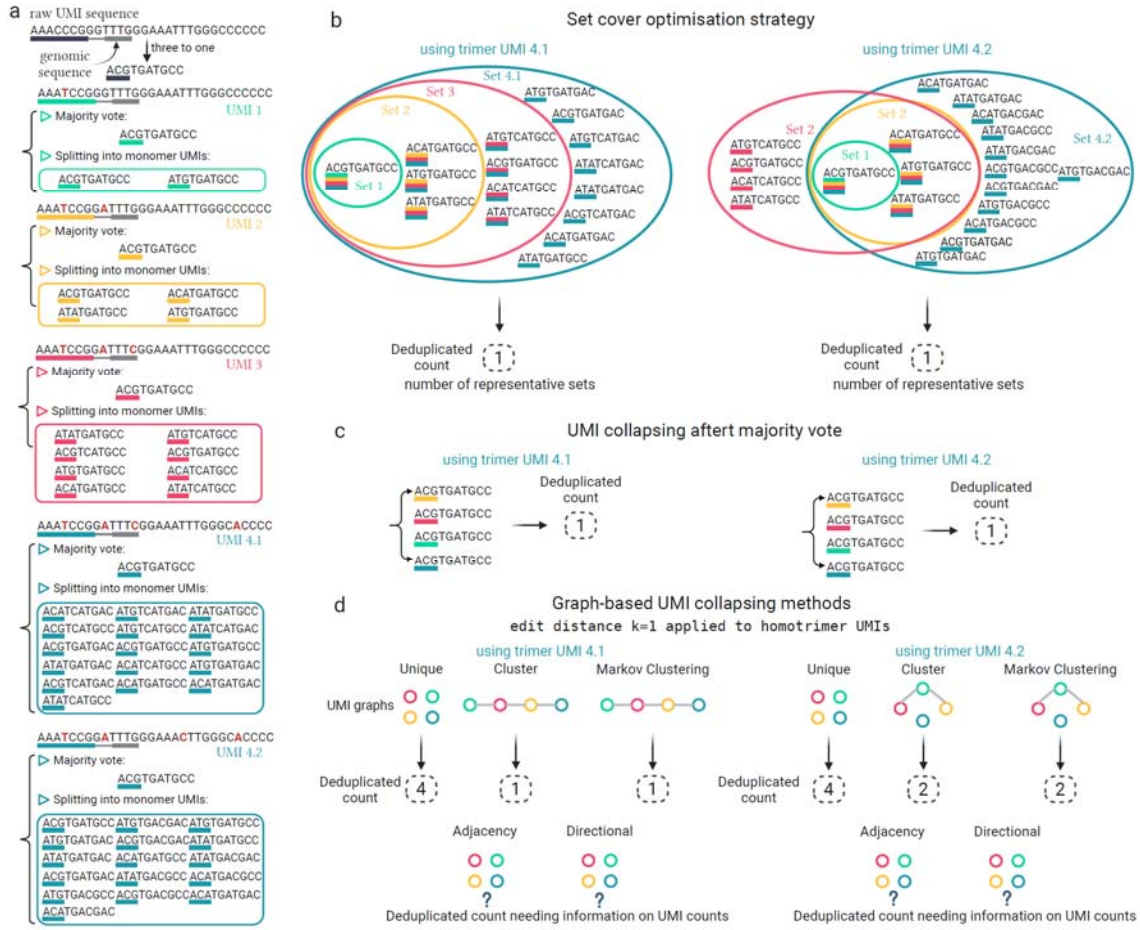


Figure 3. Working principle of the set cover optimisation strategy. **a**, an original homotrimer UMI and 5 its PCR duplicates with errors. **b**, covering monomer UMIs split from the homotrimer UMIs using the set cover optimisation technique. **c**, illustration of deduplicating the homotrimer UMIs from **a** using the majority vote strategy. **d**, illustration of deduplicating the homotrimer UMIs from **a** using graph-based UMI collapsing methods.

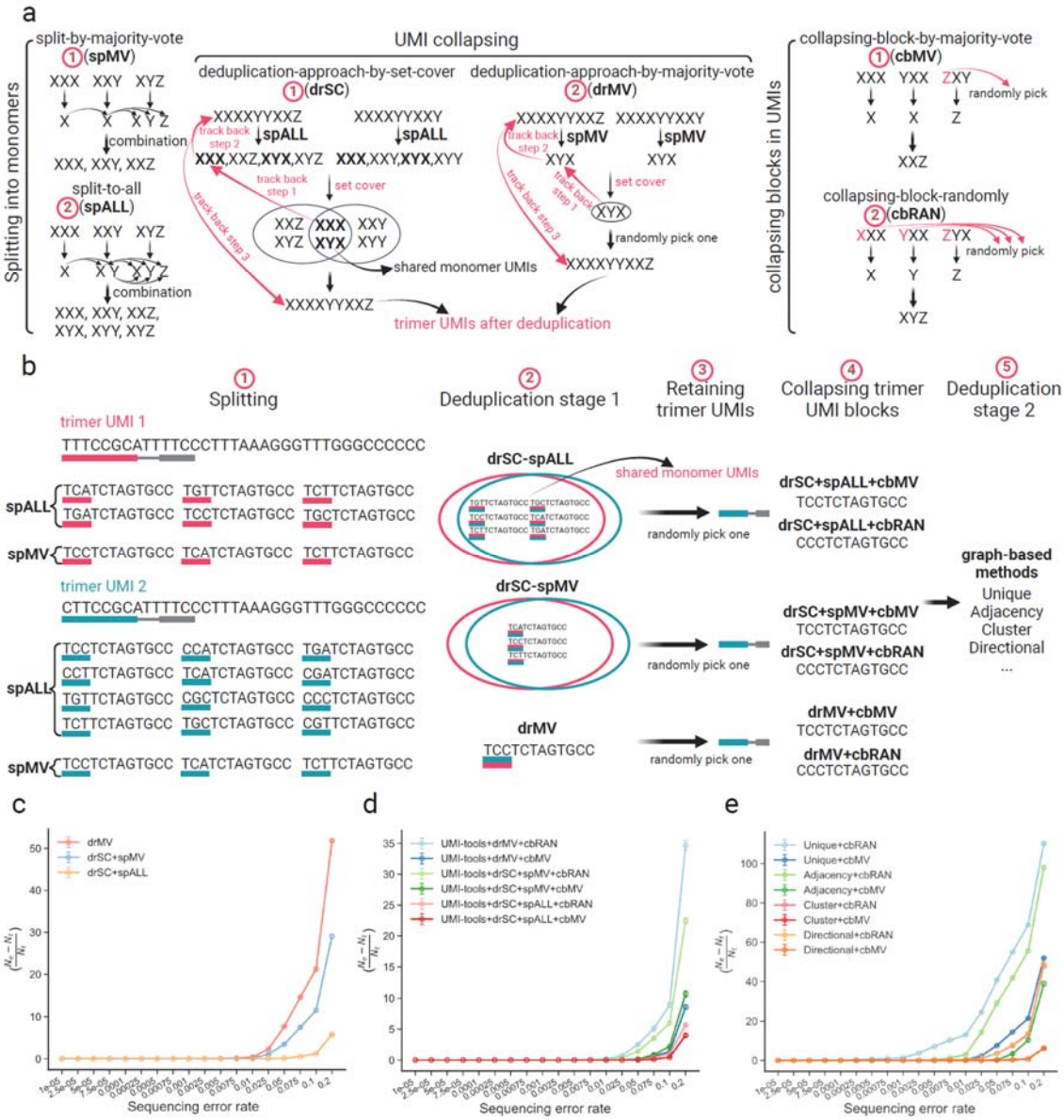


Figure 4. Comprehensive comparison between UMI collapsing methods. **a**, block-wise splitting homotrimer UMIs into monomer UMIs using the split-by-majority-vote (*spMV*) and split-to-all (*spALL*) strategies. Illustration of UMI collapsing by the set cover optimisation approach (*drSC*) and the majority vote approach (*drMV*). Illustration of collapsing blocks in UMIs by the majority vote strategy (*cbMV*) and the random selection (*cbRAN*). **b**, UMI collapsing by 1) splitting homotrimer UMIs into monomer UMIs, the first round of UMI collapsing, retaining homotrimer UMIs, collapsing UMI blocks, and the second round of UMI collapsing. **c**, Comparison of deduplication results between the *drMV* and *drSC* methods. N_e represents the estimated deduplicated count and N_t represents the true count. **d**, Comparison of UMI-tools deduplication results between the *drMV* and *drSC* methods. **e**, Comparison of deduplication results between different UMI-tools methods (*Unique*, *Adjacency*, *Cluster*, and *Directional*).

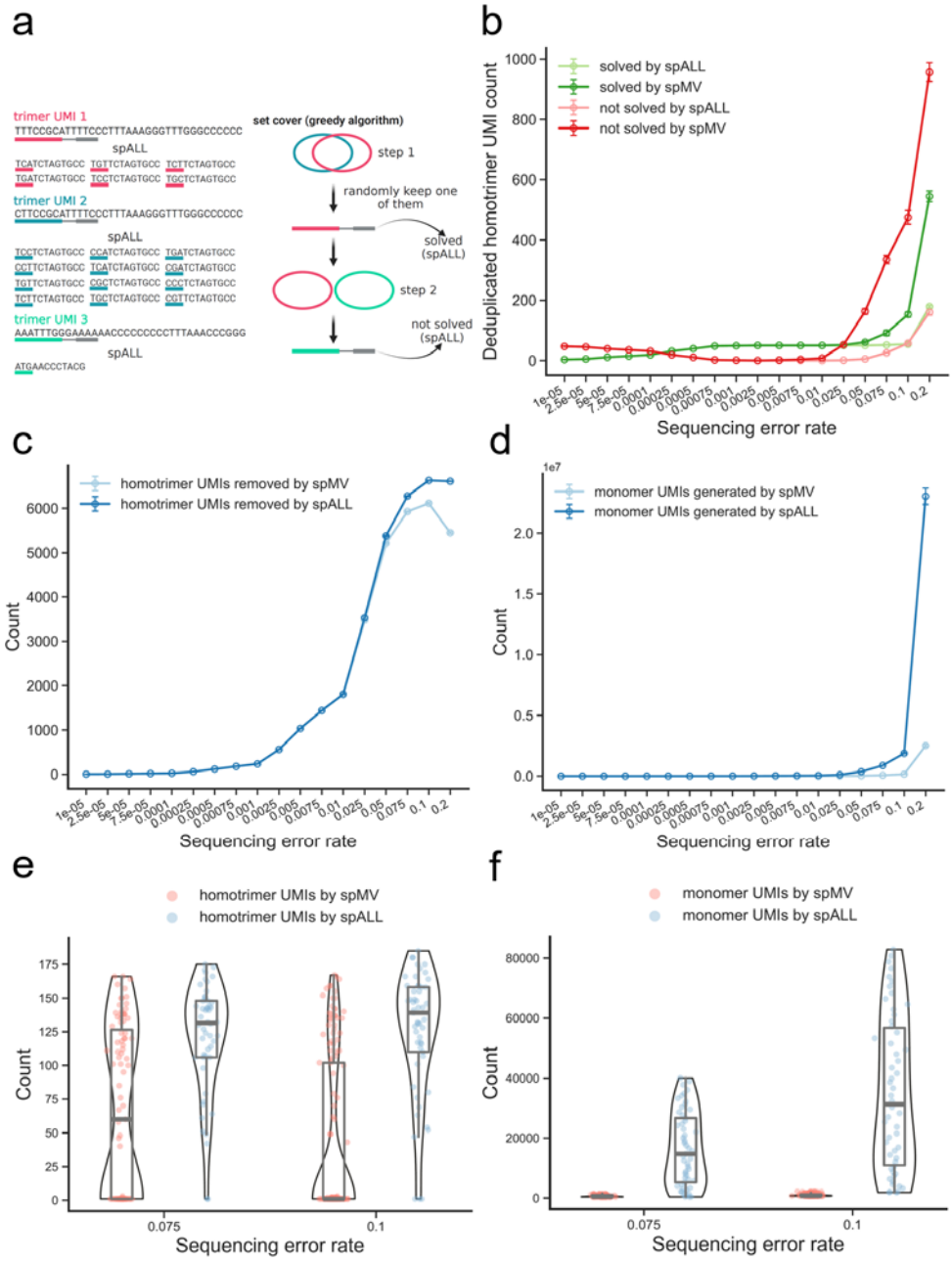


Figure 5. Analysis of split monomer UMIs and deduplicated homotrimer UMIs during set cover optimisation. **a**, illustration of solved UMIs by the set cover optimisation technique. **b**, Comparison of deduplicated counts of homotrimer UMIs solved by the set cover optimisation technique if the UMIs are split by *spMV* and *spALL*. **c**, Counts of homotrimer UMIs removed by set cover optimisation when the UMIs are split by *spMV* and *spALL*. **d**, Counts of monomer UMIs generated by *spMV* and *spALL* **e**, Counts of homotrimer UMIs removed per step using set cover optimisation when the UMIs are split by *spMV* and *spALL*. Each dot represents the count of removed homotrimer UMIs in each step of using the greedy algorithm for solving the set cover problem. **f**, Counts of monomer UMIs generated by *spMV* and *spALL* per step using set cover optimisation.

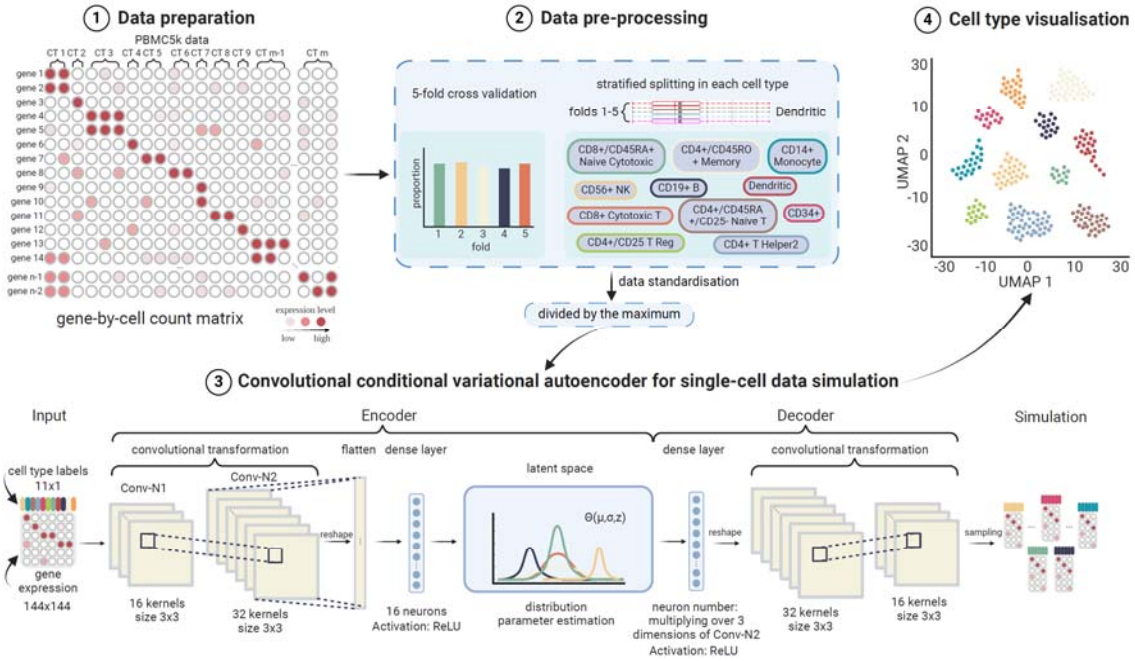


Figure 6. DeepConvCVAE for UMI count matrix simulation. It includes data preparation, data pre-processing, learning process, and cell type visualisation. 68k PBMCs scRNA-seq data was used and downloaded from https://github.com/10XGenomics/single-cell-3prime-paper/tree/master/pbmc68k_analysis. Single-cell data are filtered without a cell expressing below three genes. We used a stratified 5-fold cross validation to split the datasets for training. This ensures the cells from each cell type are mostly even across the 5 folds. Single-cell data were transformed through convolutional neural networks (CNNs) in the encoder and decoder of the conditional variational autoencoder. To suit the input size to the CNN, the single-cell count matrix is reshaped as a 144×144 square matrix. The same settings were used in CNNs in both the encoder and the decoder.

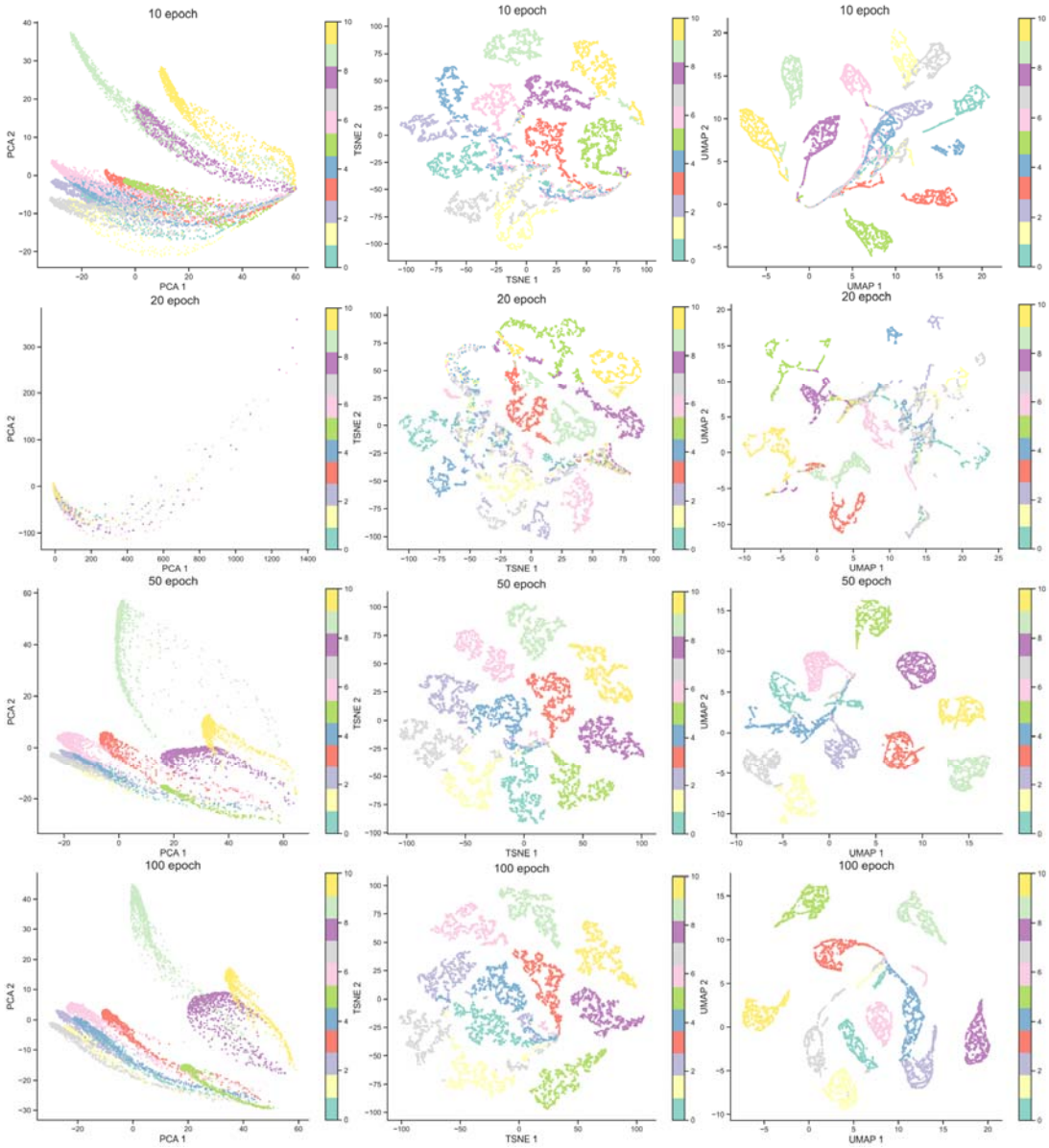


Figure 7. UMAP plots of simulated UMI counts by DeepConvCVAE at 10, 20, 50, and 100 epochs.

References

1. Ozsolak F, Milos PM. RNA sequencing: advances, challenges and opportunities. Nat Rev Genet [Internet]. 2011;12:87–98. Available from: https://doi.org/10.1038/nrg2934

2. Conesa A, Madrigal P, Tarazona S, Gomez-Cabrero D, Cervera A, McPherson A, et al. A survey of best practices for RNA-seq data analysis. Genome Biol [Internet]. 2016;17:13. Available from: https://doi.org/10.1186/s13059-016-0881-8

3. Marioni JC, Mason CE, Mane SM, Stephens M, Gilad Y. RNA-seq: An assessment of technical reproducibility and comparison with gene expression arrays. Genome Res [Internet]. 2008;18:1509–17. Available from: http://genome.cshlp.org/content/18/9/1509.abstract

4. Sereika M, Kirkegaard RH, Karst SM, Michaelsen TY, Sørensen EA, Wollenberg RD, et al. Oxford Nanopore R10.4 long-read sequencing enables the generation of near-finished bacterial genomes from pure cultures and metagenomes without short-read or reference polishing. Nat Methods [Internet]. 2022;19:823–6. Available from: https://doi.org/10.1038/s41592-022-01539-7

5. Kukurba KR, Montgomery SB. RNA Sequencing and Analysis. Cold Spring Harb Protoc [Internet]. 2015;2015:pdb.top084970. Available from: http://cshprotocols.cshlp.org/content/2015/11/pdb.top084970.abstract

6. Wagner N, Çelik MH, Hölzlwimmer FR, Mertes C, Prokisch H, Yépez VA, et al. Aberrant splicing prediction across human tissues. Nat Genet [Internet]. 2023;55:861–70. Available from: https://doi.org/10.1038/s41588-023-01373-3

7. Eggertsson HP, Kristmundsdottir S, Beyter D, Jonsson H, Skuladottir A, Hardarson MT, et al. GraphTyper2 enables population-scale genotyping of structural variation using pangenome graphs. Nat Commun [Internet]. 2019;10:5402. Available from: https://doi.org/10.1038/s41467-019-13341-9

8. Luecken MD, Theis FJ. Current best practices in single-cell RNA-seq analysis: a tutorial. Mol Syst Biol [Internet]. 2019;15:e8746. Available from: https://www.embopress.org/doi/abs/10.15252/msb.20188746

9. Heumos L, Schaar AC, Lance C, Litinetskaya A, Drost F, Zappia L, et al. Best practices for single-cell analysis across modalities. Nat Rev Genet [Internet]. 2023;24:550–72. Available from: https://doi.org/10.1038/s41576-023-00586-w

10. Shiroguchi K, Jia TZ, Sims PA, Xie XS. Digital RNA sequencing minimizes sequence-dependent bias and amplification noise with optimized single-molecule barcodes. Proceedings of the National Academy of Sciences [Internet]. 2012;109:1347–52. Available from: https://www.pnas.org/doi/abs/10.1073/pnas.1118018109

11. Kim TH, Zhou X, Chen M. Demystifying “drop-outs” in single-cell UMI data. Genome Biol [Internet]. 2020;21:196. Available from: https://doi.org/10.1186/s13059-020-02096-y

12. Chen W, Li Y, Easton J, Finkelstein D, Wu G, Chen X. UMI-count modeling and differential expression analysis for single-cell RNA sequencing. Genome Biol [Internet]. 2018;19:70. Available from: https://doi.org/10.1186/s13059-018-1438-9

13. Svensson V, Natarajan KN, Ly L-H, Miragaia RJ, Labalette C, Macaulay IC, et al. Power analysis of single-cell RNA-sequencing experiments. Nat Methods [Internet]. 2017;14:381–7. Available from: https://doi.org/10.1038/nmeth.4220

14. Salomon R, Kaczorowski D, Valdes-Mora F, Nordon RE, Neild A, Farbehi N, et al. Droplet-based single cell RNAseq tools: a practical guide. Lab Chip [Internet]. 2019;19:1706–27. Available from: http://dx.doi.org/10.1039/C8LC01239C

15. Hwang B, Lee JH, Bang D. Single-cell RNA sequencing technologies and bioinformaticspipelines. Exp Mol Med [Internet]. 2018;50:1–14. Available from: https://doi.org/10.1038/s12276-018-0071-8

16. Marx V. How to deduplicate PCR. Nat Methods [Internet]. 2017;14:473–6. Available from: https://doi.org/10.1038/nmeth.4268

17. Islam S, Zeisel A, Joost S, La Manno G, Zajac P, Kasper M, et al. Quantitative single-cell RNA-seq with unique molecular identifiers. Nat Methods [Internet]. 2014;11:163–6. Available from: https://doi.org/10.1038/nmeth.2772

18. Ziegenhain C, Hendriks G-J, Hagemann-Jensen M, Sandberg R. Molecular spikes: a gold standard for single-cell RNA counting. Nat Methods [Internet]. 2022;19:560–6. Available from: https://doi.org/10.1038/s41592-022-01446-x

19. Orabi B, Erhan E, McConeghy B, Volik S V, Le Bihan S, Bell R, et al. Alignment-free clustering of UMI tagged DNA molecules. Bioinformatics [Internet]. 2018;35:1829–36. Available from: https://doi.org/10.1093/bioinformatics/bty888

20. Österlund T, Filges S, Johansson G, Ståhlberg A. UMIErrorCorrect and UMIAnalyzer: Software for Consensus Read Generation, Error Correction, and Visualization Using Unique Molecular Identifiers. Clin Chem [Internet]. 2022;68:1425–35. Available from: https://doi.org/10.1093/clinchem/hvac136

21. Camara PG. Methods and challenges in the analysis of single-cell RNA-sequencing data. Curr Opin Syst Biol [Internet]. 2018;7:47–53. Available from: https://www.sciencedirect.com/science/article/pii/S2452310017301944

22. Fu Y, Wu P-H, Beane T, Zamore PD, Weng Z. Elimination of PCR duplicates in RNA-seq and small RNA-seq using unique molecular identifiers. BMC Genomics [Internet]. 2018;19:531. Available from: https://doi.org/10.1186/s12864-018-4933-1

23. Arguel M-J, LeBrigand K, Paquet A, Ruiz García S, Zaragosi L-E, Barbry P, et al. A cost effective 5' selective single cell transcriptome profiling approach with improved UMI design. Nucleic Acids Res [Internet]. 2016;45:e48–e48. Available from: https://doi.org/10.1093/nar/gkw1242

24. You Y, Tian L, Su S, Dong X, Jabbari JS, Hickey PF, et al. Benchmarking UMI-based single-cell RNA-seq preprocessing workflows. Genome Biol [Internet]. 2021;22:339. Available from: https://doi.org/10.1186/s13059-021-02552-3

25. Petukhov V, Guo J, Baryawno N, Severe N, Scadden DT, Samsonova MG, et al. dropEst: pipeline for accurate estimation of molecular counts in droplet-based single-cell RNA-seq experiments. Genome Biol [Internet]. 2018;19:78. Available from: https://doi.org/10.1186/s13059-018-1449-6

26. Choi YH, Kim JK. Dissecting Cellular Heterogeneity Using Single-Cell RNA Sequencing. Mol Cells [Internet]. 2019;42:189–99. Available from: https://www.sciencedirect.com/science/article/pii/S1016847823003886

27. Smith T, Heger A, Sudbery I. UMI-tools: modeling sequencing errors in Unique Molecular Identifiers to improve quantification accuracy. Genome Res [Internet]. 2017;27:491–9. Available from: http://genome.cshlp.org/content/27/3/491.abstract

28. Liu D. Algorithms for efficiently collapsing reads with Unique Molecular Identifiers. PeerJ [Internet]. 2019;7:e8275. Available from: https://doi.org/10.7717/peerj.8275

29. Peng X, Dorman KS. Accurate estimation of molecular counts from amplicon sequence data with unique molecular identifiers. Bioinformatics [Internet]. 2023;39:btad002. Available from: https://doi.org/10.1093/bioinformatics/btad002

30. Pflug FG, von Haeseler A. TRUmiCount: correctly counting absolute numbers of molecules using unique molecular identifiers. Bioinformatics [Internet]. 2018;34:3137–44. Available from: https://doi.org/10.1093/bioinformatics/bty283

31. Sun J, Philpott M, Loi D, Li S, Monteagudo-Mesas P, Hoffman G, et al. Correcting PCR amplification errors in unique molecular identifiers to generate accurate numbers of sequencing molecules. Nat Methods [Internet]. 2024;21:401–5. Available from: https://doi.org/10.1038/s41592-024-02168-y

32. Philpott M, Watson J, Thakurta A, Brown T, Brown T, Oppermann U, et al. Nanopore sequencing of single-cell transcriptomes with scCOLOR-seq. Nat Biotechnol [Internet]. 2021;39:1517–20. Available from: https://doi.org/10.1038/s41587-021-00965-w

33. Young NE. Greedy Set-Cover Algorithms. In: Kao M-Y, editor. Encyclopedia of Algorithms [Internet]. New York, NY: Springer New York; 2016. p. 886–9. Available from: https://doi.org/10.1007/978-1-4939-2864-4_175

34. Homotrimer barcodes enable accurate counting of RNA molecules during high-throughput RNA sequencing. Nat Methods [Internet]. 2024;21:379–80. Available from: https://doi.org/10.1038/s41592-024-02169-x

35. Tischler G, Leonard S. biobambam: tools for read pair collation based algorithms on BAM files. Source Code Biol Med [Internet]. 2014;9:13. Available from: https://doi.org/10.1186/1751-0473-9-13

36. Cribbs AP, Luna-Valero S, George C, Sudbery IM, Berlanga-Taylor AJ, Sansom SN, et al. CGAT-core: a python framework for building scalable, reproducible computational biology workflows. bioRxiv [Internet]. 2019; Available from: https://www.biorxiv.org/content/early/2019/03/18/581009

37. Vandereyken K, Sifrim A, Thienpont B, Voet T. Methods and applications for single-cell and spatial multi-omics. Nat Rev Genet [Internet]. 2023;24:494–515. Available from: https://doi.org/10.1038/s41576-023-00580-2

38. Li X, Wang C-Y. From bulk, single-cell to spatial RNA sequencing. Int J Oral Sci [Internet]. 2021;13:36. Available from: https://doi.org/10.1038/s41368-021-00146-0

39. Andrews TS, Kiselev VY, McCarthy D, Hemberg M. Tutorial: guidelines for the computational analysis of single-cell RNA sequencing data. Nat Protoc [Internet]. 2021;16:1–9. Available from: https://doi.org/10.1038/s41596-020-00409-w

40. Li WV, Li JJ. A statistical simulator scDesign for rational scRNA-seq experimental design. Bioinformatics [Internet]. 2019;35:i41–50. Available from: https://doi.org/10.1093/bioinformatics/btz321

41. Korthauer KD, Chu L-F, Newton MA, Li Y, Thomson J, Stewart R, et al. A statistical approach for identifying differential distributions in single-cell RNA-seq experiments. Genome Biol [Internet]. 2016;17:222. Available from: https://doi.org/10.1186/s13059-016-1077-y

42. Zappia L, Phipson B, Oshlack A. Splatter: simulation of single-cell RNA sequencing data. Genome Biol [Internet]. 2017;18:174. Available from: https://doi.org/10.1186/s13059-017-1305-0

43. Song D, Wang Q, Yan G, Liu T, Sun T, Li JJ. scDesign3 generates realistic in silico data for multimodal single-cell and spatial omics. Nat Biotechnol [Internet]. 2024;42:247–52. Available from: https://doi.org/10.1038/s41587-023-01772-1

44. Sun T, Song D, Li WV, Li JJ. scDesign2: a transparent simulator that generates high-fidelity single-cell gene expression count data with gene correlations captured. Genome Biol [Internet]. 2021;22:163. Available from: https://doi.org/10.1186/s13059-021-02367-2

45. Baruzzo G, Patuzzi I, Di Camillo B. SPARSim single cell: a count data simulator for scRNA-seq data. Bioinformatics [Internet]. 2019;36:1468–75. Available from: https://doi.org/10.1093/bioinformatics/btz752

46. Zhang X, Xu C, Yosef N. Simulating multiple faceted variability in single cell RNA sequencing. Nat Commun [Internet]. 2019;10:2611. Available from: https://doi.org/10.1038/s41467-019-10500-w

47. Assefa AT, Vandesompele J, Thas O. SPsimSeq: semi-parametric simulation of bulk and single-cell RNA-sequencing data. Bioinformatics [Internet]. 2020;36:3276–8. Available from: https://doi.org/10.1093/bioinformatics/btaa105

48. Bacher R, Kendziorski C. Design and computational analysis of single-cell RNA-sequencing experiments. Genome Biol [Internet]. 2016;17:63. Available from: https://doi.org/10.1186/s13059-016-0927-y

49. Kingma DP, Welling M. Auto-Encoding Variational Bayes. 2022.

50. Lukošiute K, Raaijmakers G, Doctor Z, Soares-Santos M, Nord B. KilonovaNet: Surrogate models of kilonova spectra with conditional variational autoencoders. Mon Not R Astron Soc [Internet]. 2022;516:1137–48. Available from: https://doi.org/10.1093/mnras/stac2342

51. Sohn K, Lee H, Yan X. Learning Structured Output Representation using Deep Conditional Generative Models. In: Cortes C, Lawrence N, Lee D, Sugiyama M, Garnett R, editors. Adv Neural Inf Process Syst [Internet]. Curran Associates, Inc.; 2015. Available from: https://proceedings.neurips.cc/paper_files/paper/2015/file/8d55a249e6baa5c06772297520da2051-Paper.pdf

52. van Erven T, Harremos P. Rényi Divergence and Kullback-Leibler Divergence. IEEE Trans Inf Theory. 2014;60:3797–820.

53. LeCun Y, Kavukcuoglu K, Farabet C. Convolutional networks and applications in vision. Proceedings of 2010 IEEE International Symposium on Circuits and Systems. 2010. p. 253–6.

54. Krizhevsky A, Sutskever I, Hinton GE. ImageNet Classification with Deep Convolutional Neural Networks. Commun ACM [Internet]. 2017;60:84–90. Available from: https://doi.org/10.1145/3065386

55. Jolliffe IT, Cadima J. Principal component analysis: a review and recent developments. Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences [Internet]. 2016;374:20150202. Available from: https://royalsocietypublishing.org/doi/abs/10.1098/rsta.2015.0202

56. van der Maaten L, Hinton G. Visualizing Data using t-SNE. Journal of Machine Learning Research [Internet]. 2008;9:2579–605. Available from: http://jmlr.org/papers/v9/vandermaaten08a.html

57. Becht E, McInnes L, Healy J, Dutertre C-A, Kwok IWH, Ng LG, et al. Dimensionality reduction for visualizing single-cell data using UMAP. Nat Biotechnol [Internet]. 2019;37:38–44. Available from: https://doi.org/10.1038/nbt.4314

58. Alon N, Awerbuch B, Azar Y, Buchbinder N, Naor J (Seffi). The Online Set Cover Problem. SIAM Journal on Computing [Internet]. 2009;39:361–70. Available from: https://doi.org/10.1137/060661946

59. Cormode G, Karloff H, Wirth A. Set cover algorithms for very large datasets. Proceedings of the 19th ACM International Conference on Information and Knowledge Management [Internet]. New York, NY, USA: Association for Computing Machinery; 2010. p. 479–88. Available from: https://doi.org/10.1145/1871437.1871501

60. Cemgil T, Ghaisas S, Dvijotham K, Gowal S, Kohli P. The Autoencoding Variational Autoencoder. In: Larochelle H, Ranzato M, Hadsell R, Balcan MF, Lin H, editors. Adv Neural Inf Process Syst [Internet]. Curran Associates, Inc.; 2020. p. 15077–87. Available from: https://proceedings.neurips.cc/paper_files/paper/2020/file/ac10ff1941c540cd87c107330996f4f6-Paper.pdf

61. Chen X, Kingma DP, Salimans T, Duan Y, Dhariwal P, Schulman J, et al. Variational Lossy Autoencoder. International Conference on Learning Representations [Internet]. 2017. Available from: https://openreview.net/forum?id=BysvGP5ee

62. Kingma DP, Salimans T, Jozefowicz R, Chen X, Sutskever I, Welling M. Improved Variational Inference with Inverse Autoregressive Flow. In: Lee D, Sugiyama M, Luxburg U, Guyon I, Garnett R, editors. Adv Neural Inf Process Syst [Internet]. Curran Associates, Inc.; 2016. Available from: https://proceedings.neurips.cc/paper_files/paper/2016/file/ddeebdeefdb7e7e7a697e1c3e3d8ef54-Paper.pdf

63. Salimans T, Kingma D, Welling M. Markov Chain Monte Carlo and Variational Inference: Bridging the Gap. In: Bach F, Blei D, editors. Proceedings of the 32nd International Conference on Machine Learning [Internet]. Lille, France: PMLR; 2015. p. 1218–26. Available from: https://proceedings.mlr.press/v37/salimans15.html

64. Kingma DP, Welling M. An Introduction to Variational Autoencoders. Foundations and Trends® in Machine Learning [Internet]. 2019;12:307–92. Available from: http://dx.doi.org/10.1561/2200000056

65. Hinton G, Deng L, Yu D, Dahl GE, Mohamed A, Jaitly N, et al. Deep Neural Networks for Acoustic Modeling in Speech Recognition: The Shared Views of Four Research Groups. IEEE Signal Process Mag. 2012;29:82–97.

66. de Boer P-T, Kroese DP, Mannor S, Rubinstein RY. A Tutorial on the Cross-Entropy Method. Ann Oper Res [Internet]. 2005;134:19–67. Available from: https://doi.org/10.1007/s10479-005-5724-z

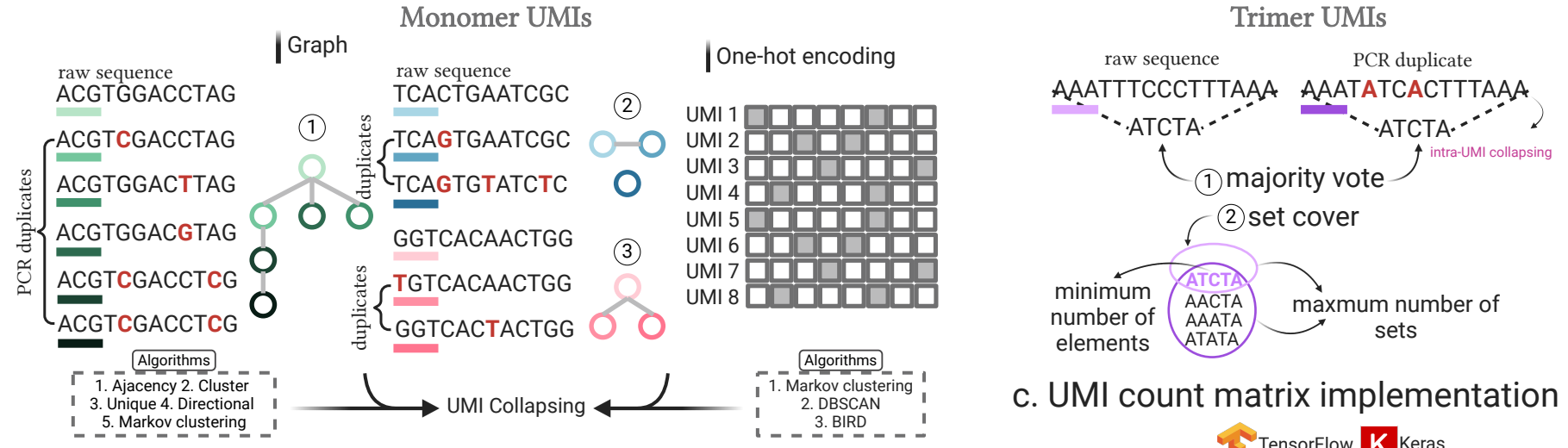
67. Zou F, Shen L, Jie Z, Zhang W, Liu W. A sufficient condition for convergences of adam and rmsprop. Proceedings of the IEEE/CVF Conference on computer vision and pattern recognition. 2019. p. 11127–35.

68. Zheng GXY, Terry JM, Belgrader P, Ryvkin P, Bent ZW, Wilson R, et al. Massively parallel digital transcriptional profiling of single cells. Nat Commun [Internet]. 2017;8:14049. Available from: https://doi.org/10.1038/ncomms14049

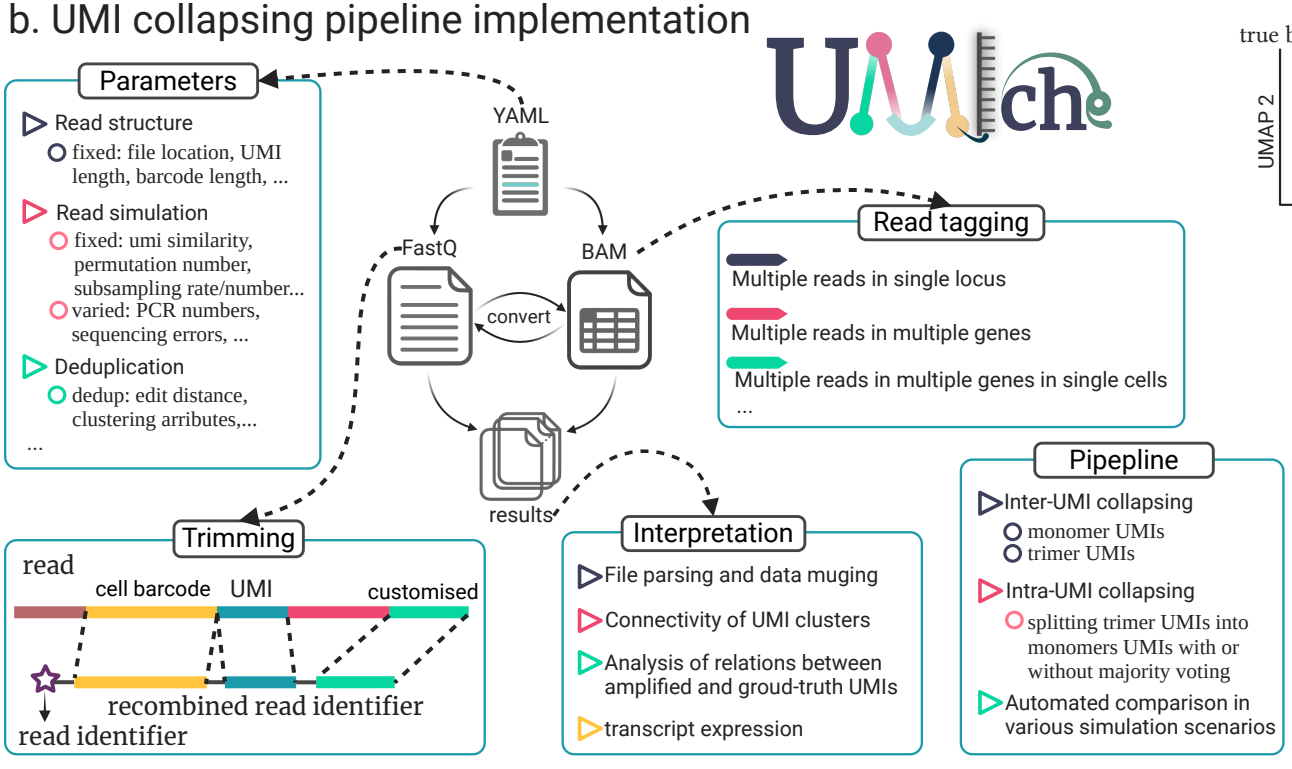
69. Martín~Abadi, Ashish~Agarwal, Paul~Barham, Eugene~Brevdo, Zhifeng~Chen, Craig~Citro, et al. TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems [Internet]. 2015. Available from: https://www.tensorflow.org/

70. Chollet F, others. Keras [Internet]. GitHub; 2015. Available from: https://github.com/fchollet/keras

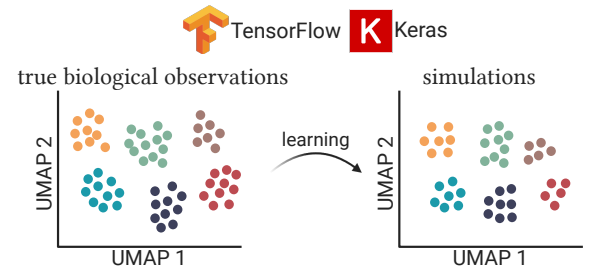
a. UMI collapsing method implementation



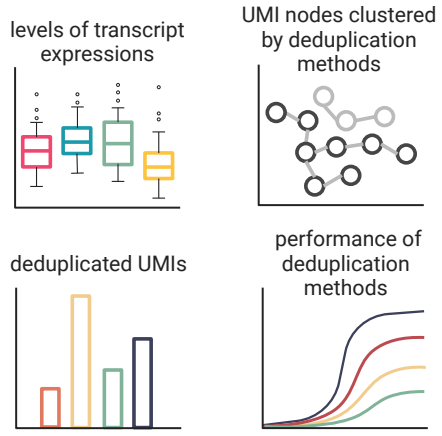
b. UMI collapsing pipeline implementation

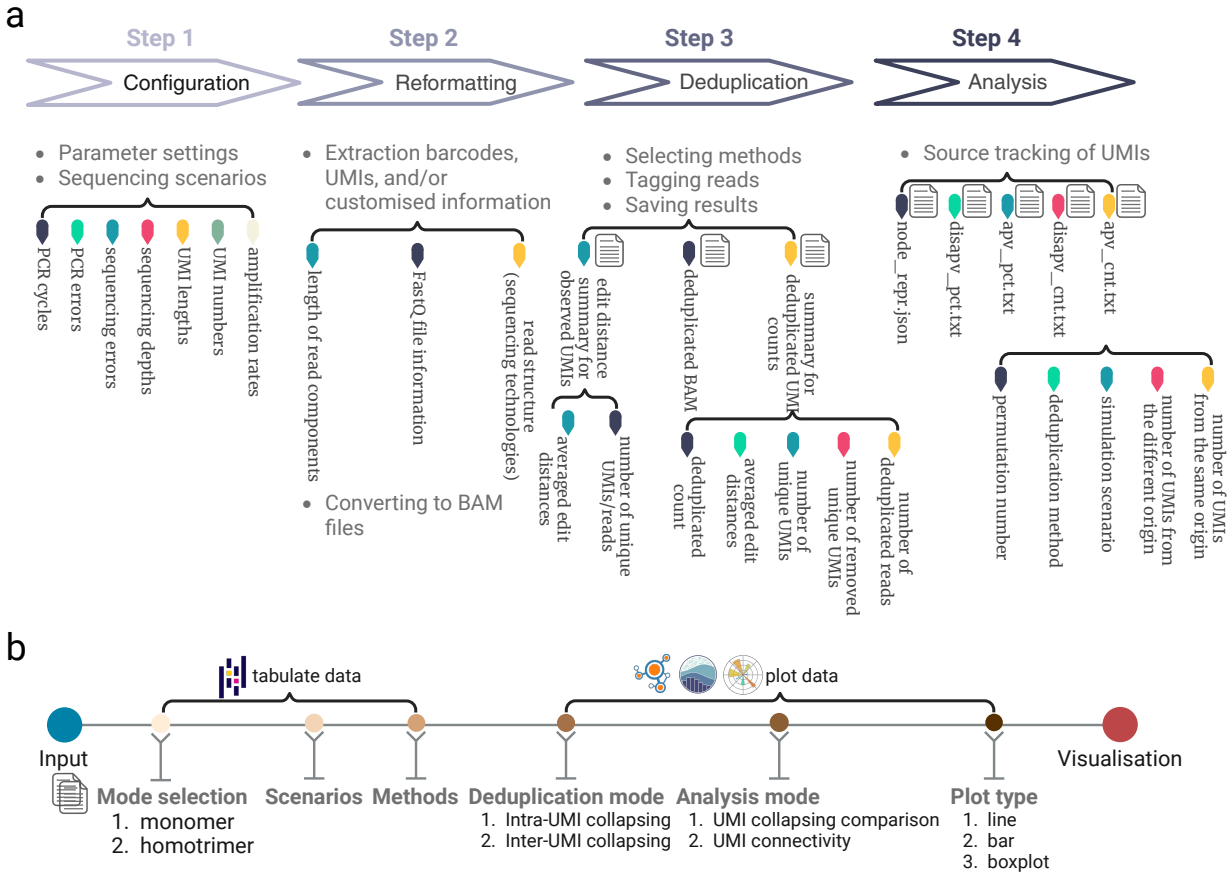


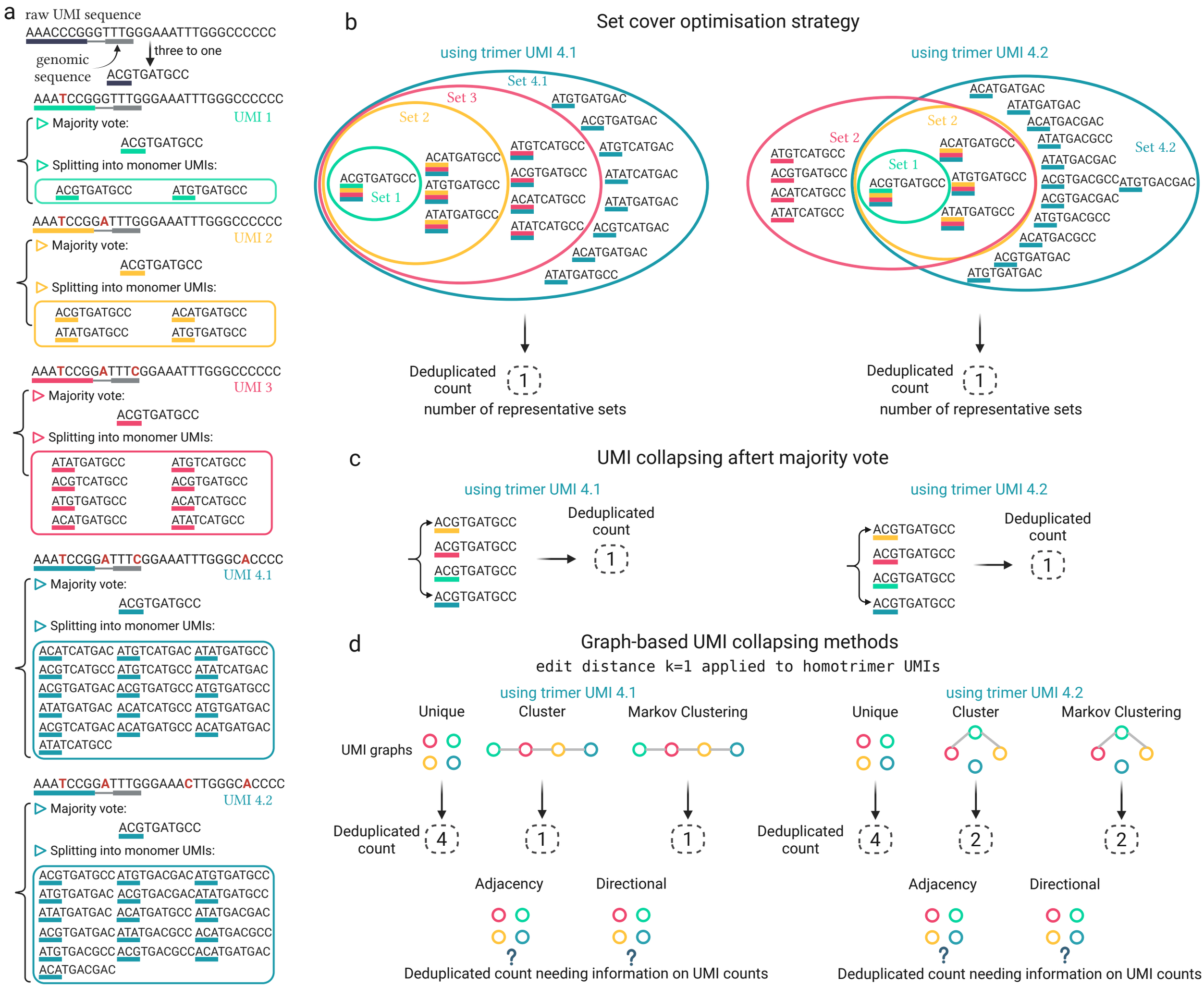
c. UMI count matrix implementation

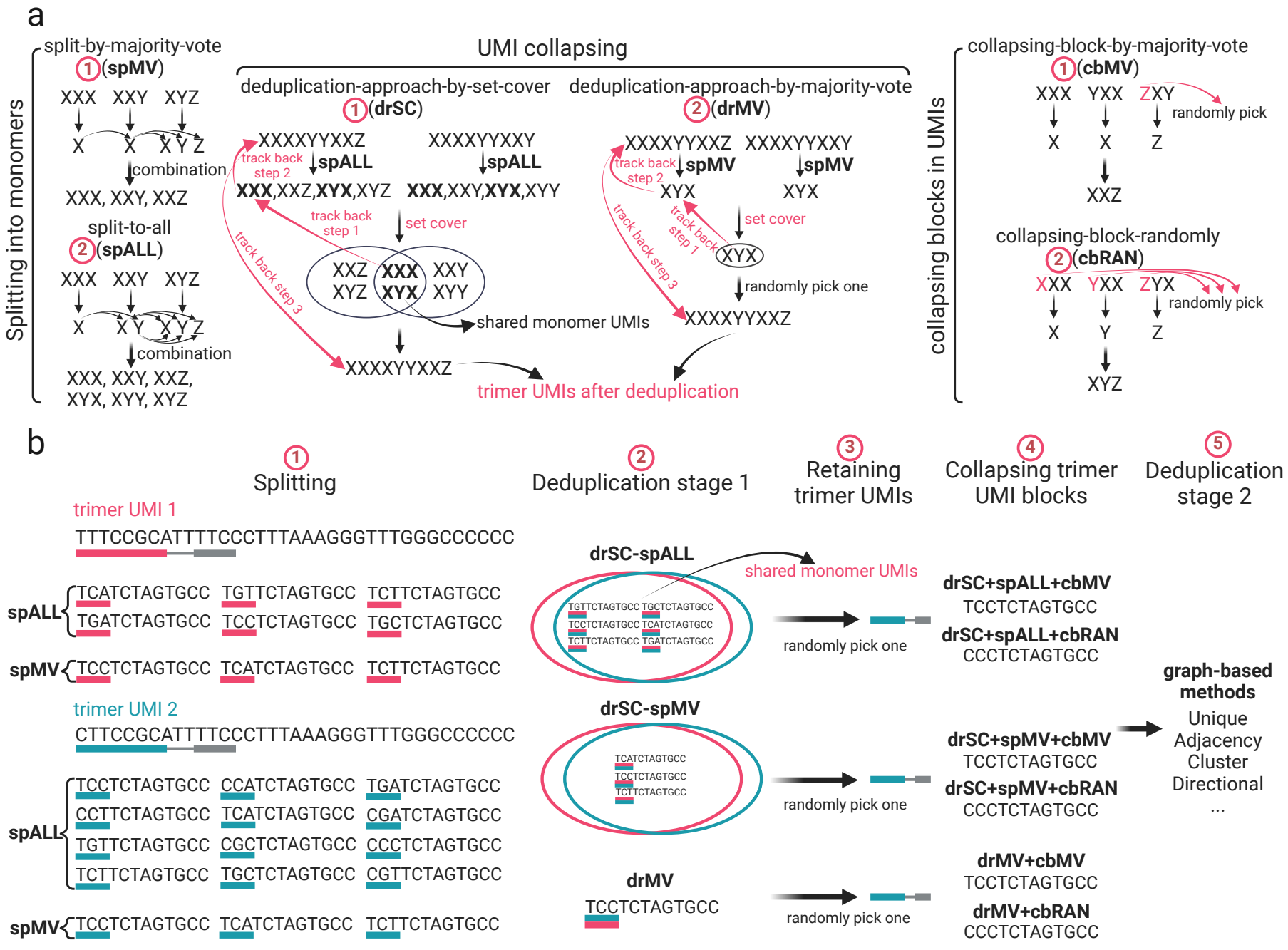


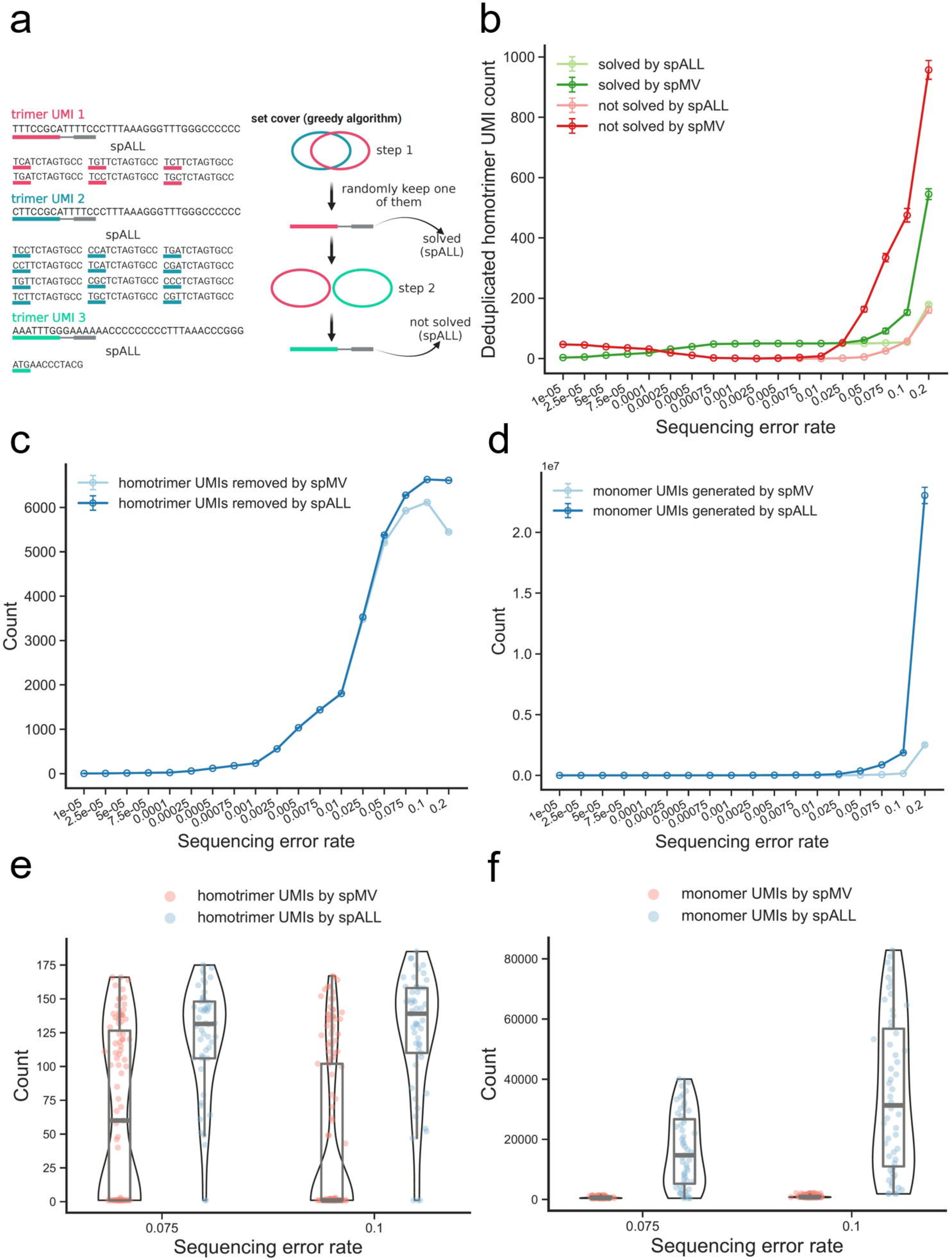
d. Visualisation

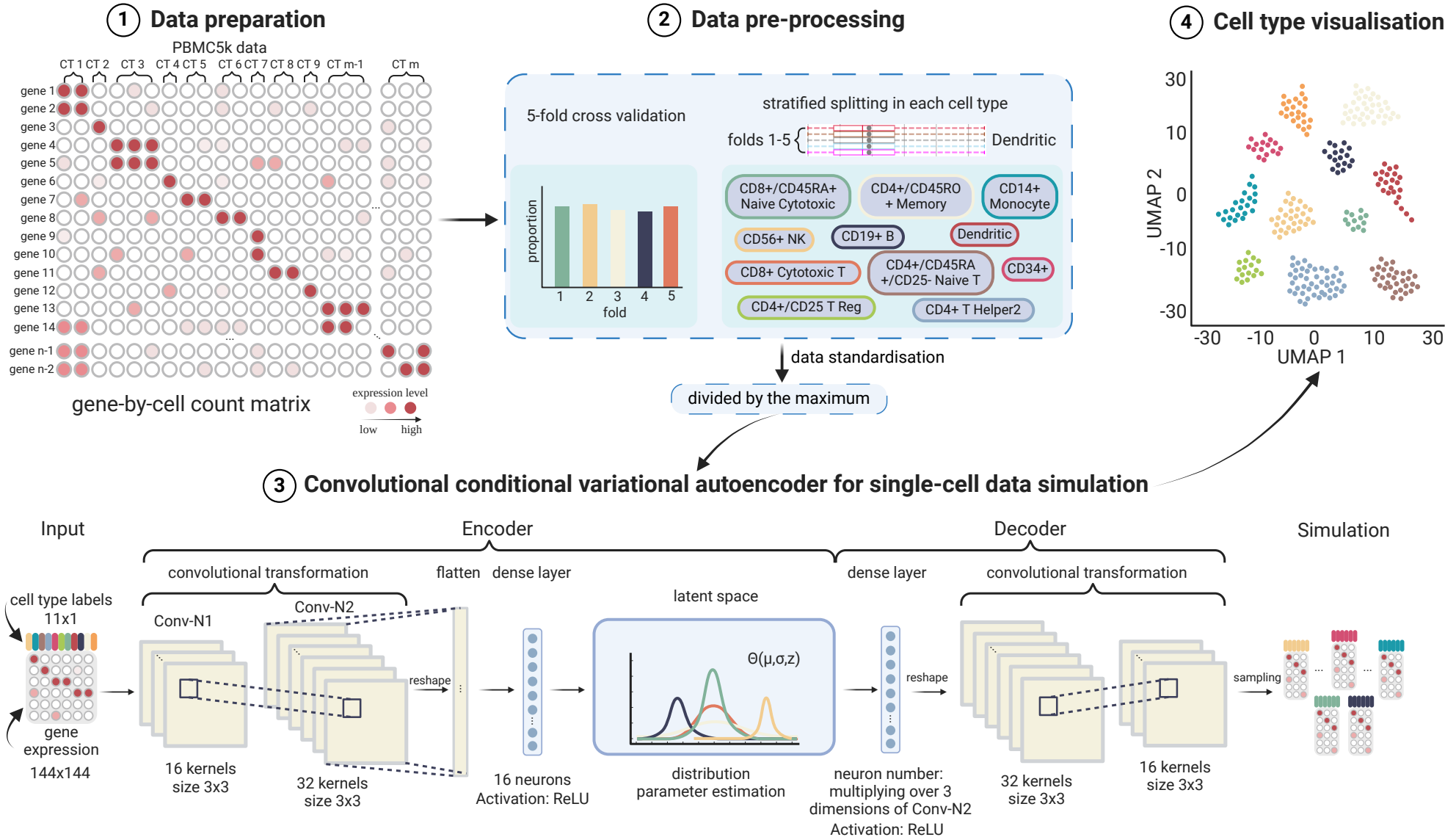


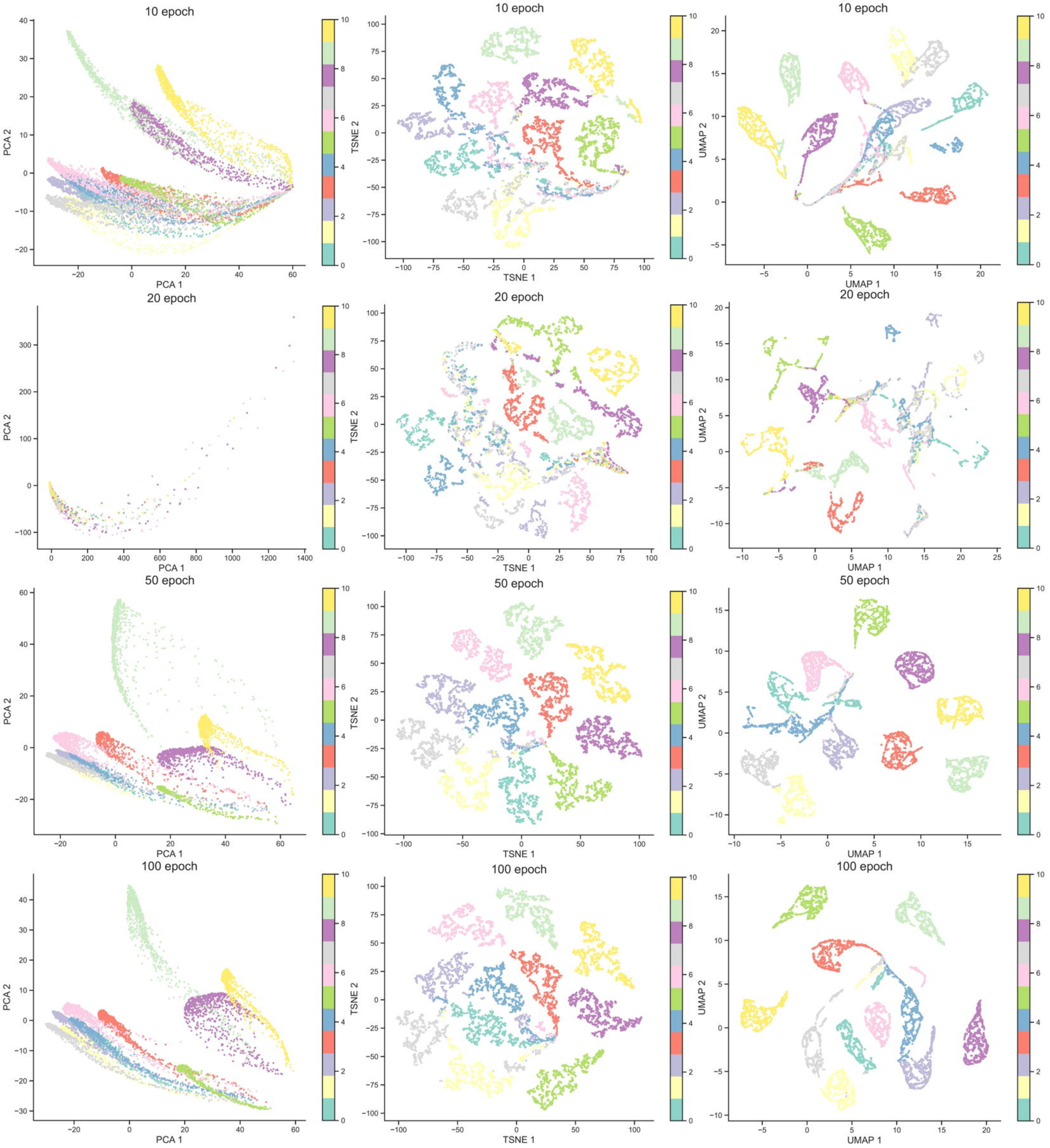

a
Step 1
Configuration
Step 2
Reformatting
Step 3
Deduplication
Step 4
Analysis
Parameter settings
Sequencing scenarios
PCR cycles
PCR errors
sequencing errors
sequencing depths
UMI lengths
UMI numbers
amplification rates
Extraction barcodes, UMIs, and/or customised information
length of read components
FastQ file information
read structure (sequencing technologies)
Converting to BAM files
Selecting methods
Tagging reads
Saving results
edit distance summary for observed UMIs
averaged edit distances
number of unique UMIs/reads
deduplicated BAM
deduplicated count
averaged edit distances
number of unique UMIs
number of removed unique UMIs
number of deduplicated reads
summary for deduplicated UMI counts
Source tracking of UMIs
node_repr.json
disapv_pct.txt
apv_pct.txt
disapv_cnt.txt
apv_cnt.txt
permutation number
deduplication method
simulation scenario
number of UMIs from the different origin
number of UMIs from the same origin
b
tabulate data
plot data
Input
Mode selection
1. monomer
2. homotrimer
Scenarios
Methods
Deduplication mode
1. Intra-UMI collapsing
2. Inter-UMI collapsing
Analysis mode
1. UMI collapsing comparison
2. UMI connectivity
Plot type
1. line
2. bar
3. boxplot
Visualisation
c
Density
Edit distance
d
1e6
monomer
trimer (after majority vote)
Number of incorrect UMIs
Sequencing error rate


a
raw UMI sequence
AAACCCGGGTTTGGGAAATTTGGGCCCCCC
genomic sequence
three to one
ACGTGATGCC
AAATCCGGGTTTGGGAAATTTGGGCCCCCC
UMI 1
Majority vote:
ACGTGATGCC
Splitting into monomer UMIs:
ACGTGATGCC ATGTGATGCC
AAATCCGGATTTGGGAAATTTGGGCCCCCC
UMI 2
Majority vote:
ACGTGATGCC
Splitting into monomer UMIs:
ACGTGATGCC ACATGATGCC
ATATGATGCC ATGTGATGCC
AAATCCGGATTTCGGAAATTTGGGCCCCCC
UMI 3
Majority vote:
ACGTGATGCC
Splitting into monomer UMIs:
ATATGATGCC ATGTCATGCC
ACGTCATGCC ACGTGATGCC
ATGTGATGCC ACATCATGCC
ACATGATGCC ATATCATGCC
AAATCCGGATTTCGGAAATTTGGGCACCCC
UMI 4.1
Majority vote:
ACGTGATGCC
Splitting into monomer UMIs:
ACATCATGAC ATGTCATGAC ATATGATGCC
ACGTCATGCC ATGTCATGCC ATATCATGAC
ACGTGATGAC ACGTGATGCC ATGTGATGCC
ATATGATGAC ACATCATGCC ATGTGATGAC
ACGTCATGAC ACATGATGCC ACATGATGAC
ATATCATGCC
AAATCCGGATTTGGGAAACTTGGGCACCCC
UMI 4.2
Majority vote:
ACGTGATGCC
Splitting into monomer UMIs:
ACGTGATGCC ATGTGACGAC ATGTGATGCC
ATGTGATGAC ACGTGACGAC ATATGATGCC
ATATGATGAC ACATGATGCC ATATGACGAC
ACGTGATGAC ATATGACGCC ACATGACGCC
ATGTGACGCC ACGTGACGCC ACATGATGAC
ACATGACGAC
b
Set cover optimisation strategy
using trimer UMI 4.1
Set 4.1
Set 3
Set 2
Set 1
ACGTGATGCC
ACATGATGCC
ATGTGATGCC
ATATGATGCC
ATGTCATGCC
ACGTGATGCC
ACATCATGCC
ATATCATGCC
ATGTGATGAC
ACGTGATGAC
ATGTCATGAC
ATATCATGAC
ATATGATGAC
ACGTCATGAC
ACATGATGAC
ATATGATGCC
Deduplicated count
1
number of representative sets
using trimer UMI 4.2
Set 2
Set 1
Set 4.2
ACATGATGAC
ATATGATGAC
ACATGACGAC
ATATGACGCC
ATATGACGAC
ACGTGACGCC
ACGTGACGAC
ATGTGACGCC
ACATGACGCC
ACGTGATGAC
ATGTGATGAC
ATGTCATGCC
ACGTGATGCC
ACATCATGCC
ATATCATGCC
ACATGATGCC
ATGTGATGCC
ATATGATGCC
Deduplicated count
1
number of representative sets
c
UMI collapsing aftert majority vote
using trimer UMI 4.1
ACGTGATGCC
ACGTGATGCC
ACGTGATGCC
ACGTGATGCC
Deduplicated count
1
using trimer UMI 4.2
ACGTGATGCC
ACGTGATGCC
ACGTGATGCC
ACGTGATGCC
Deduplicated count
1
d
Graph-based UMI collapsing methods
edit distance k=1 applied to homotrimer UMIs
using trimer UMI 4.1
Unique
Cluster
Markov Clustering
UMI graphs
Deduplicated count
4
1
1
using trimer UMI 4.2
Unique
Cluster
Markov Clustering
Deduplicated count
4
2
2
Adjacency
Directional
?
?
Deduplicated count needing information on UMI counts
Adjacency
Directional
?
?
Deduplicated count needing information on UMI counts


a
Splitting into monomers
split-by-majority-vote
1 (spMV)
XXX XXY XYZ
X X XYZ
combination
XXX, XXY, XXZ
split-to-all
2 (spALL)
XXX XXY XYZ
X XY XYZ
combination
XXX, XXY, XXZ,
XYX, XYY, XYZ
UMI collapsing
deduplication-approach-by-set-cover
1 (drSC)
XXXXYYXXZ
spALL
XXXXYYXXY
spALL
XXX,XXZ,XYX,XYZ
XXX,XXY,XYX,XYY
track back step 2
track back step 1
track back step 3
set cover
XXZ XXX XXY
XYZ XYX XYY
shared monomer UMIs
XXXXYYXXZ
deduplication-approach-by-majority-vote
2 (drMV)
XXXXYYXXZ XXXXYYXXY
spMV
spMV
XYX
XYX
track back step 2
track back step 1
track back step 3
set cover
XYX
randomly pick one
XXXXYYXXZ
trimer UMIs after deduplication
collapsing blocks in UMIs
collapsing-block-by-majority-vote
1 (cbMV)
XXX YXX ZXY
randomly pick
X X Z
XXZ
collapsing-block-randomly
2 (cbRAN)
XXX YXX ZYX
randomly pick
X Y Z
XYZ
b
1 Splitting
2 Deduplication stage 1
3 Retaining trimer UMIs
4 Collapsing trimer UMI blocks
5 Deduplication stage 2
trimer UMI 1
TTTCCGCATTTTCCCTTTAAAGGGTTTGGGCCCCCC
spALL
TCATCTAGTGCC TGTTCTAGTGCC TCTTCTAGTGCC
TGATCTAGTGCC TCCTCTAGTGCC TGCTCTAGTGCC
spMV
TCCTCTAGTGCC TCATCTAGTGCC TCTTCTAGTGCC
trimer UMI 2
CTTCCGCATTTTCCCTTTAAAGGGTTTGGGCCCCCC
spALL
TCCTCTAGTGCC CCATCTAGTGCC TGATCTAGTGCC
CCTTCTAGTGCC TCATCTAGTGCC CGATCTAGTGCC
TGTTCTAGTGCC CGCTCTAGTGCC CCCTCTAGTGCC
TCTTCTAGTGCC TGCTCTAGTGCC CGTTCTAGTGCC
spMV
TCCTCTAGTGCC TCATCTAGTGCC TCTTCTAGTGCC
drSC-spALL
shared monomer UMIs
TGTTCTAGTGCC TGCTCTAGTGCC
TCCTCTAGTGCC TCATCTAGTGCC
TCTTCTAGTGCC TGATCTAGTGCC
randomly pick one
drSC+spALL+cbMV
TCCTCTAGTGCC
drSC+spALL+cbRAN
CCCTCTAGTGCC
drSC-spMV
TCATCTAGTGCC
TCCTCTAGTGCC
TCTTCTAGTGCC
randomly pick one
drSC+spMV+cbMV
TCCTCTAGTGCC
drSC+spMV+cbRAN
CCCTCTAGTGCC
graph-based methods
Unique
Adjacency
Cluster
Directional
...
drMV
TCCTCTAGTGCC
randomly pick one
drMV+cbMV
TCCTCTAGTGCC
drMV+cbRAN
CCCTCTAGTGCC
c
UMI-tools+drMV+cbRAN
UMI-tools+drMV+cbMV
UMI-tools+drSC+spMV+cbRAN
UMI-tools+drSC+spMV+cbMV
UMI-tools+drSC+spALL+cbRAN
UMI-tools+drSC+spALL+cbMV
$\left(\frac{N_e - N_t}{N_t}\right)$
Sequencing error rate
d
Unique+cbRAN
Unique+cbMV
Adjacency+cbRAN
Adjacency+cbMV
Cluster+cbRAN
Cluster+cbMV
Directional+cbRAN
Directional+cbMV
$\left(\frac{N_e - N_t}{N_t}\right)$
Sequencing errorrate
e
drMV
drSC+spMV
drSC+spALL
$\left(\frac{N_e - N_t}{N_t}\right)$
Sequencing error rate


a
trimer UMI 1
TTTCCGCATTTTCCCTTTAAAGGGTTTGGGCCCCCC
spALL
TCATCTAGTGCC TGTTCTAGTGCC TCTTCTAGTGCC
TGATCTAGTGCC TCCTCTAGTGCC TGCTCTAGTGCC
trimer UMI 2
CTTCCGCATTTTCCCTTTAAAGGGTTTGGGCCCCCC
spALL
TCCTCTAGTGCC CCATCTAGTGCC TGATCTAGTGCC
CCTTCTAGTGCC TCATCTAGTGCC CGATCTAGTGCC
TGTTCTAGTGCC CGCTCTAGTGCC CCCTCTAGTGCC
TCTTCTAGTGCC TGCTCTAGTGCC CGTTCTAGTGCC
trimer UMI 3
AAATTTGGGAAAAAACCCCCCCCCTTTAAACCCGGG
spALL
ATGAACCCTACG
set cover (greedy algorithm)
step 1
randomly keep one of them
solved (spALL)
step 2
not solved (spALL)
b
solved by spALL
solved by spMV
not solved by spALL
not solved by spMV
Deduplicated homotrimer UMI count
Sequencing error rate
c
homotrimer UMIs removed by spMV
homotrimer UMIs removed by spALL
Count
Sequencing error rate
d
monomer UMIs generated by spMV
monomer UMIs generated by spALL
Count
Sequencing error rate
e
homotrimer UMIs by spMV
homotrimer UMIs by spALL
Count
Sequencing error rate
f
monomer UMIs by spMV
monomer UMIs by spALL
Count
Sequencing error rate


1 Data preparation
PBMC5k data
CT 1 CT 2 CT 3 CT 4 CT 5 CT 6 CT 7 CT 8 CT 9 CT m-1 CT m
gene 1
gene 2
gene 3
gene 4
gene 5
gene 6
gene 7
gene 8
gene 9
gene 10
gene 11
gene 12
gene 13
gene 14
gene n-1
gene n-2
gene-by-cell count matrix
expression level
low
high
2 Data pre-processing
5-fold cross validation
stratified splitting in each cell type
folds 1-5
Dendritic
proportion
1 2 3 4 5
fold
CD8+/CD45RA+ Naive Cytotoxic
CD4+/CD45RO + Memory
CD14+ Monocyte
CD56+ NK
CD19+ B
Dendritic
CD8+ Cytotoxic T
CD4+/CD45RA +/CD25- Naive T
CD34+
CD4+/CD25 T Reg
CD4+ T Helper2
data standardisation
divided by the maximum
4 Cell type visualisation
UMAP 2
UMAP 1
30
10
0
-10
-30
3 Convolutional conditional variational autoencoder for single-cell data simulation
Input
Encoder
Decoder
Simulation
convolutional transformation
flatten
dense layer
dense layer
convolutional transformation
latent space
cell type labels
11x1
gene expression
144x144
Conv-N1
Conv-N2
reshape
16 kernels
size 3x3
32 kernels
size 3x3
16 neurons
Activation: ReLU
Θ(μ,σ,z)
distribution
parameter estimation
neuron number:
multiplying over 3
dimensions of Conv-N2
Activation: ReLU
reshape
32 kernels
size 3x3
16 kernels
size 3x3
sampling


10 epoch
PCA 1
PCA 2
TSNE 1
TSNE 2
UMAP 1
UMAP 2
20 epoch
50 epoch
100 epoch