Cytomulate: Accurate and Efficient Simulation of CyTOF data

Yuqiu Yang[1,2], Kaiwen Wang[1], Zeyu Lu[1], Tao Wang[2,3,*], Xinlei Wang[1,*]

1 Department of Statistical Science, Southern Methodist University, Dallas, TX, USA, 75275.

2 Quantitative Biomedical Research Center, Department of Population and Data Sciences, University of Texas Southwestern Medical Center, Dallas, TX, USA, 75390.

3 Center for the Genetics of Host Defense, University of Texas Southwestern Medical Center, Dallas, TX, USA, 75390.

* Corresponding authors

Short title: Cytomulate: Simulation of CyTOF data

Key words: CyTOF, simulation, proteomics

Abstract:

Recently, many analysis tools have been devised to offer insights into data generated via Cytometry by time-of-flight (CyTOF). However, objective evaluations of these methods remain absent as most evaluations are conducted against real data where the ground truth is generally unknown. In this paper, we develop Cytomulate, a reproducible and accurate simulation algorithm of CyTOF data, which could serve as a foundation for future method development and evaluation. We demonstrate that Cytomulate can capture various characteristics of CyTOF data and is superior in learning overall data distributions than single-cell RNA-seq-oriented methods such as Splatter and generative models like LAMBDA.

Background:

Recent years have seen exponential growth in applications of mass cytometry (CyTOF) which characterizes proteomics profiles of tens of thousands or millions of heterogeneous single cells on 30-40 protein channels with minimal spill-over effects (1–3). As the technique of CyTOF matures, many data analysis methods have been proposed to offer insights into various aspects of

single-cell experiments (4). For example, in (5), the authors have observed that the signal intensity of a CyTOF experiment would vary over time due to changes in instrument performance. Their proposed method is currently the dominant tool to alleviate the temporal effect using polystyrene beads embedded with metal lanthanides. On the other hand, methods such as CytofRUV (6), and Cytonorm (7), etc. aim at getting rid of batch effects to facilitate comparisons across different CyTOF experiments. Deep learning algorithms such as residual networks (ResNet) (8) and generative adversarial networks (GAN) (9) were also adopted to resolve the same problem. While some software such as FlowSOM (10), LAMBDA (11), and Bayesian trees (12) focus on automatic cell type identification, others like CytoTree (13) attempt to explore the trajectories along which cells differentiate into other cells. Various data visualization and dimension reduction algorithms have also been applied to CyTOF (14). Typical examples include: t-SNE (15), UMAP (16), MDS (17), SAUCIE (18), scvis (19), etc. As the output of a CyTOF acquisition software is in Flow Cytometry Standard (FCS) format, packages such as flowCore (20) and Seurat (21) have incorporated standardized and efficient pipelines to read, write, and preprocess CyTOF data.

Despite the vast array of methods applicable to CyTOF, most of the CyTOF analysis methods developed by researchers are often evaluated against real data. Without a gold standard and known truth, such as labels and actual distributions, performance assessment becomes somewhat subjective while also blurring the line between method validation and result discovery. The few studies with simulated datasets are highly specialized and workflow-dependent. For example, CytoGLMM (22) focuses on a single cell type for the purpose of differential expression analysis. In (23), the authors further used the simulated datasets in (22) to evaluate performance on various differential expression detection methods. (24) proposed a Bayesian hierarchical model based on Gaussian distributions to study the difference in the protein functional network of a single cell type under two experimental conditions. In (25) the simulation was done by manually splitting a real dataset into two overlapping subsets. These simulation procedures are not specifically designed for general-purpose simulations of CyTOF samples with many cell types, trajectories, batches, etc.

Simulated datasets, albeit being synthetic, contain ground truth against which methods and algorithms can be reliably tested. Instead of *ad hoc* simulation in each study mentioned above, a formal simulation tool naturally provides a framework for a systematic evaluation for new methodological research in the field. By varying parameters and settings of key factors, researchers can easily generate an unbiased coverage of the system under study in a straightforward manner. On the other hand, real experiments require detailed prior planning, precise executions, and potentially abundant resources. The flexibility along with cost and time benefits make simulation a vital tool for the field. However, the current lack of a rigorous, reliable simulation algorithm in CyTOF calls for the necessity of a foundation for developing and validating all other CyTOF data analysis tools.

By contrast, various methods have been developed for simulating single cell RNA-sequencing (scRNA-seq) data, such as Simple (26), Lun (27), BASICS (28), and Splatter (29), among which Splatter is dominant. Since both scRNA-seq and CyTOF are single-cell sequencing techniques, it might be tempting to tweak scRNA-seq simulation tools such as Splatter and apply them to CyTOF. However, such a process can be highly nontrivial and the end results may be unsatisfactory if only small adjustments are sought after. First of all, these two techniques are of different biological nature. scRNA-seq examines the expression of thousands of RNAs of several thousand cells, whereas CyTOF captures information on typically 30-40 proteins for tens of thousands or even millions of cells. Therefore, compared with scRNA-seq, CyTOF is more closely related to clinical phenotypes and is more likely to identify rare cell populations. Furthermore, while drop-out is a common issue in scRNA-seq data, it is a much less encountered problem in CyTOF experiments. What's more, the expressions in scRNA-seq are integer-valued "counts", in contrast to real-valued data or even negative values in CyTOF expression matrices due to noise and normalization by the inverse hyperbolic sine function (30), which, for the sake of clarity, will be referred to as asinh-transformation herein and after. Consequently, a sequence of nonlinear transformations is almost inevitable to render the expression matrices acceptable inputs to simulation tools like Splatter, which will yield biased results. Finally, the expression levels of various protein channels in CyTOF are naturally dependent. Extending the discrete random variable model such as the Poisson-Gamma mixture model adopted by Splatter to capture the dependency structures present in CyTOF might not be a trivial task. The same logic applies to other generic simulation algorithms that are not designed with CyTOF in mind.

To bridge this gap in literature, we developed Cytomulate, the first formal simulation tool that is capable of generating simulated datasets that capture various characteristics of CyTOF data. Our framework combines the flexibility of generating complex, realistic data with efficiency and usability, allowing researchers to acquire data at scale and practitioners to easily simulate datasets for their workflows.

Results:

Characteristics of CyTOF data

To motivate the design choices of Cytomulate and highlight the differentiating features between CyTOF and scRNA-seq, we explored prominent characteristics of CyTOF using various public and in-house datasets (e.g., Finck (5), Levine_13dim (31), Levine_32dim (31), CytoNorm (7), CyAnno (32), Covid (14); Table 1). In this section, we mainly use the protein expressions from the Levine_13dim dataset as an example for demonstration purposes. Similar phenomena have been observed across many CyTOF datasets. The Levine_13dim dataset is publicly available via the R package HDCytoData (33). As the protein expressions have already been pre-processed to correct unwanted artifacts, such as temporal effect, only asinh-transformation was applied.

High volume, high dimension, and heterogeneity: The expression matrix associated with a typical CyTOF experiment usually contains tens of thousands or even millions of cell events (rows), 30-40 protein channels (columns), from a pool of cells of 10-30 cell types. For example, the data of the first patient in Levine_13dim has 167,044 cell events, 13 protein channels, and 14 cell types manually gated by the authors. Compared to a typical scRNA-seq dataset, the number of the cells in a CyTOF like this is far larger albeit with fewer channels. Therefore, efficiency in both extracting major patterns from numerous cell events and generating a large dataset should be crucial for any potential simulation tools.

Zero-inflation: Like scRNA-seq data, the protein expressions observed in a CyTOF experiment often contain features that are abundant with zeros or values close to zero. Biologically, these columns correspond to surface proteins that are often too scarce to be detected by the machine. Take, for example the CD4 T cells in the dataset, we see in Fig. 1a and Fig. 1b that the majority of the columns contain a high proportion of values close to zero. Yet, it is not necessarily correct to assume the same amount of zero-inflation as in the case of scRNA-seq as the latter is typically much more sparse due to drop-out.

Dependent features: In Fig. 1c, we calculated the correlation matrix among all the protein channels using the data of Naive CD4+ T cells. Clearly, markers such as CD44 and CD45 are correlated, implying that the independence assumption between columns might be too restrictive. This suggests modeling the data jointly instead of marginally.

Normality: To explore the normality of protein expressions after standard asinh transformation and preprocessing, we performed dimension reduction using Principal Component Analysis (PCA). All cell populations resemble roughly elliptic shapes (Fig. 1d), suggesting that the data is Gaussian or near Gaussian because of the assumptions of PCA. Thus, the usage of Gaussian distributions as basic building blocks of a simulation model is a sensible approach. This key characteristic also differentiates CyTOF from scRNA-seq, which is typically modeled with counts. As explored in the comparison study later in this paper, adapting a model for discrete counts for CyTOF is not necessarily a good idea.

Temporal effect: As illustrated in (5), the signals of a single CyTOF experiment would typically vary over time due to changes in instrument performance. Since Cytomulate aims at modeling asinh-transformed protein expressions, unlike in (5), we chose to demonstrate the signal drift on transformed data (Supp. Fig. 1a). As a first attempt to model the signal drift, we centered each signal, fitted a smoothing spline against one of them, and subtracted the fitted value from each signal. In Supp. Fig. 1b, we see that the adjusted signals overlap and are roughly homoscedastic, suggesting that the overall shape of the signal drift (temporal effect) may be shared by all cell types within a single batch and that we could model the effect using an additive model.

Batch effect: Apart from the temporal effect, many researchers also reported the presence of batch effects across multiple CyTOF experiments due to sample collection, stimulation, staining,

etc (34). In (7) they reported that not only are the batch effects cell-type specific, they also vary depending on protein channels. In Supp. Fig. 1c, we illustrated this phenomenon by an interaction plot using the mean expressions of the protein channels of CD4 T cells and CD8 T cells measured on two patients in the Levine_32dim dataset (33). We see that not only do expression levels differ between two batches, the differences also vary by cell types and protein channels. This implies that in addition to the main batch effect, we should also take into consideration the interaction effects.

Cellular trajectory: Similar to scRNA-seq, a common task is to study cellular trajectory where the expressions of an individual cell are assumed to vary according to some continuous path that connects one cell type to another. In the context of scRNA-seq, lots of methods have been developed to align cells on continuous paths with typical examples Slingshot and Monocle (35–37). To our knowledge, however, CytoTree (13) is the only available method of trajectory inference designed for CyTOF. It is questionable whether trajectory inference methods are interchangeable between these two technologies. For the purpose of demonstration, we simulated an example of the cellular trajectories following CytoTree's online tutorial in Supp. Fig. 1d.

Cytomulate Framework

In light of the previous exploratory data analysis, we propose Cytomulate, an accurate and efficient simulation algorithm of CyTOF data based on Gaussian Mixture Models (GMM) (38), with add-on features to mimic various patterns/effects existing in real data. Combining the ability to accurately approximate the densities of real CyTOF datasets and the class-leading efficiency, Cytomulate is able to synthesize realistic datasets in seconds for any research needs. More specifically, Cytomulate achieves the task of accurate simulation via two simulation modes selectable by the user: creation mode and emulation mode. The former caters to those who need granular controls over setting, whereas the latter aims to mimic existing datasets. The following sections elaborate on each mode with greater details as the foundation to the comparison study that follows.

Creation mode aims at quickly generating synthetic datasets that capture some aforementioned characteristics of CyTOF data such as zero-inflation, column dependencies, and near normality. As this mode does not intend to extract the data distribution of some specific dataset as a reference, it is particularly useful for testing methods on a large scale. For example, in (14), Wang et al. used this mode of Cytomulate to evaluate the performance of 20 dimension reduction methods from various aspects such as local and global structure preservation on 425 simulated datasets with different attributes (e.g., different numbers of cell types and numbers of differentiation paths). This mode allows practitioners to precisely define each simulated dataset with known ground truth, which is not often or at all possible with real data.

Specifically, the creation mode randomly generates average cell expressions and then utilizes GMMs from which to sample cell expressions. Users also have the options to enable complex

simulations by simply specifying the necessary parameters, such as the number of trees for cell differentiation trajectories. The details of complex simulation will be discussed later. With users having full control of their simulated dataset, the creation mode is untethered by any existing—yet oftentimes implicit—assumptions of real datasets, such as the exact cell types, circumstances of the patient, the physical machine, etc. By eliminating these factors, Cytomulate allows researchers to develop and test methods based on unbiased results that are generalizable to wide ranging real datasets. Further, it can also serve as a pedagogical tool for researchers to explore the characteristics of CyTOF datasets easily.

Emulation mode, on the other hand, is designed to learn the distribution of some real CyTOF data. Instead of approximating the entire dataset as a whole, which is nontrivial, Cytomulate takes advantage of cell types as prior knowledge. Hence, it requires as inputs the expression matrix of one particular CyTOF dataset to mimic and the associated cell type labels. Although Cytomulate can potentially perform cell type identification or clustering using common algorithms of the likes of K-means, we leave the choice of such algorithm for cell typing to our users as a design choice for the following two reasons: 1. According to (39) and (40), the k-means algorithm and GMM-based model (SWIFT) are often not the optimal choice for clustering CyTOF data; 2. There are already plenty of well-established cell identification algorithms and automatic gating techniques such as FlowSOM (10), Bayesian Tree (12), ACDC (41), SCINA (42), etc, all of which have similar goals but slightly different design considerations. For example, using FlowSOM requires minimal prior knowledge on the cell population other than some crude idea on the number of cell types. On the other hand, Bayesian Tree and ACDC use a template as an input to facilitate the automatic gating process. Since we expect the users of the emulation mode to be somewhat experienced in the field, we think it would be better to let the users choose and apply their own cell identification algorithms that suit their situations the best. The resulting expression matrices of this mode are thus expected to be a closer approximation to the real data, as opposed to those generated by the creation mode.

To dive into the details, Cytomulate, at its core, uses a GMM with truncation below 0 to approximate and generate the probability density of a given cell type. The Bayesian Information Criterion (BIC) (43) is adopted for model selection. Although the exploratory data analysis in the previous section shows that a multivariate Gaussian might be a good candidate for modeling a specific cell type in a CyTOF experiment, this should be taken with a grain of salt since the Levine_32dim dataset provides labels with not only major cell types but also granular subtypes. However, we should not expect all users to be experts in identifying granular cell types. For example, there are 4 subtypes of B cells in Levine_32dim: mature B cells, plasma B cells, pre-B cells, and pro-B cells. If the users are only interested in B cells, a single multivariate Gaussian would be inadequate in describing the data generating mechanism (see the comparison study in the next section for details). Therefore, we chose GMM as the base for its flexibility and its mathematical tractability.

To cater to complex needs of methodological developments for CyTOF data, in both simulation modes, users are allowed to vary cell abundances, noise levels, batch effects, temporal effects, as well as cellular trajectories. The simulated batch effects, temporal effects, cellular trajectories, as well as the noise are then added once we sample from the GMMs. In Fig. 2a, we outlined the overall framework of Cytomulate with input and output examples, model specification, and possible workflows for downstream analyses. Further details on the two simulation modes and the probabilistic model can be found in the Methods section.

We also modularized all the aforementioned functionalities so that each of them can be easily fine-tuned or turned off with a simple interface, according to an user's specific need. Cytomulate returns the final expression matrices, the cell type labels associated with the expression matrices, as well as cellular trajectory information for each cell event. Optionally, users can also choose PyCytoData as the output option for downstream analyses compatibility (Fig. 2b). In the following sections, we conduct some comparison studies and demonstrate the usage of Cytomulate through examples.

Simulation Comparison Study

Although Cytomulate is the first-of-its-kind simulation tool designed specifically for CyTOF experiments, we will be remiss not to include benchmarks since it's a common practice in the field to adopt similar technologies from other fields to CyTOF. To showcase the superior statistical and computational properties of Cytomulate, we performed a series of empirical experiments using the emulation mode. In this simulation setting, the real benchmark datasets provide a gold standard for assessing the accuracy of each simulation method and the overall validity of the underlying model. More specifically, we focused on whether each model is capable of capturing the features of CyTOF datasets and the overall distributions and relationships between all protein channels.

For the comparison study, we included the following models from within Cytomulate and other fields:

1. Splatter (29)
2. Latent Allocation Model with Bayesian Data Analysis (LAMBDA) (11)
3. Cytomulate constrained to one Gaussian component with diagonal covariance per cell type (Independent Gaussian or simply IG)
4. Cytomulate constrained to one Gaussian component with full covariance per cell type (Dependent Gaussian or simply DG)
5. Cytomulate with full covariance and model selection via Bayesian Information Criterion (BIC) from GMMs with components varying from 1 to 9 (Cytomulate)

Given that Splatter and LAMBDA are not designed for CyTOF simulation, below we give a brief description of each followed by the rationale for their inclusion in our comparison study.

Splatter, or specifically Splat, is originally designed for simulating scRNA-seq data. Its core model uses a Gamma-Poisson mixture to simulate counts observed in scRNA-seq experiments. The authors also take into consideration outliers, library size, mean-variance trend, and technical dropout. As Splatter expects integers as inputs, to simulate CyTOF data which usually have been transformed into a numeric scale using the arcsinh transformation, we can carry out its inverse transformation, round up the results to the nearest integer (30), apply Splatter to simulate counts, and transform the simulated expressions back into the numeric scale. We included Splatter to demonstrate the consequences of adapting scRNA-seq simulation methods to CyTOF, which outweigh the tempting benefit of convenience for any rigorous CyTOF studies.

LAMBDA is a Bayesian hierarchical model designed for automatic cell identification in CyTOF data. The model assumes that given a cell type, the cell expressions follow a zero-inflated multivariate Gaussian distribution. The cell types, on the other hand, are generated from a Categorical distribution whose parameters are controlled by some clinical information. Parameter estimation is accomplished by a stochastic EM algorithm. It is worth pointing out that LAMBDA is not a simulation tool and its simulation functionality is only a byproduct of LAMBDA. More critically, LAMBDA cannot generate new datasets without using reference datasets. Thus, for those who are interested in the Creation Mode or those who do not have access to any previous samples (*e.g.* CyTOF for some rare diseases), this is not a suitable method at all. We included the method in our study because of its rigorous parameter estimation procedure, which would provide us with a yardstick against which we can evaluate the quality of the simulated datasets in terms of the first and second moments.

We also included multiple versions of Cytomulate with progressively more complex model structures. By taking into consideration the linear dependencies among protein channels, the DG model should show the value of joint modeling when compared with the IG model. Compared with the DG model, our core model, Cytomulate, should demonstrate the merit of modeling non-normality. The inclusion of these three models will not only demonstrate the flexibility of the Cytomulate framework, but it will also validate the characteristics of CyTOF, as shown in the exploratory data analysis, using actual benchmarks.

To demonstrate the versatility of Cytomulate, six publicly available datasets from different species and diverse anatomic sites are collected: Levine_32dim (31), Levine_13dim (31), Samusik (44), CyAnno (32), Covid (14), and LG (45). If multiple samples exist for one dataset, the first sample is used. See Table 1 for details. All of these datasets have been bead-normalized and transformed by using the arcsinh transformation. Cell type information is also publicly available for the first four datasets. Cell types in Covid and LG were identified via FlowSOM and manual gating (14).

Cytomulate well approximates real data

In this comparison study, we make use of all six datasets and include the aforementioned five models as candidates. Three metrics are adopted to quantify the quality of the simulated datasets: 1. L2 distance between the simulated mean and the observed mean; 2. Frobenius distance between the simulated covariance and observed covariance; 3. Kullback-Leibler (KL) divergence between the simulated data and observed data. See the Methods section for further details. For each dataset, the process is repeated 50 times for 50 simulated datasets.

Using the CyAnno dataset, detailed comparisons show that Cytomulate outperforms all other methods benchmarked in this study with the smallest KL divergence (Fig. 3). As the KL divergence is a measure of how similar two distributions are, Cytomulate's resounding lead is a testament to its ability to yield a better approximation to the real data compared with other methods. For mean estimation, all methods except Splatter work well, with LAMBDA achieving the smallest Mean Squared Error (MSE) as expected, closely followed by Cytomulate. Even though LAMBDA's estimation procedure should theoretically yield decent results for the first two moments, Cytomulate still outperformed LAMBDA in estimating the covariance structure. This result also empirically demonstrated that potential improvements can be made if only the first two moments are considered, but this improvement—if having any practical significance at all—may come at the cost of worse KL Divergence and efficiency. Splatter, due to the usage of nonlinear transformations and marginal modeling, performs the worst in all benchmarks by an overwhelmingly wide margin, demonstrating that Splatter is suboptimal when simulating CyTOF data.

To visually validate the rankings of each method, we applied UMAP to illustrate the resulting expression matrices of each method (Fig. 4a - Fig. 4d, Supp. Fig. 2). Due to the granularity of the cell type labels included in the dataset, at the first glance, results returned by Cytomulate and LAMBDA resemble the real data. However, if we examine the clusters at the bottom right of the central cluster more closely, we see that Cytomulate depicts the data in far more detail than does LAMBDA. The preservation of small clusters provides an advantage in preserving rare cell types and their unique phenotypes, whereas LAMBDA's resolution limitations tend to group these cells into adjoining clusters.

In balance, considering results from all six datasets and all three metrics, Cytomulate performs the best in terms of averaged ranks. By using GMMs at the cell type level, Cytomulate always gives the best approximation to the real data as measured by the KL divergence (Fig. 4e) . As expected, by formally deriving a rigorous estimation procedure, LAMBDA is able to achieve the highest ranking in terms of mean and sometimes covariance. However, as we have seen previously, the sufficiency of only considering the first two moments highly depends on the data and the coarseness of the cell type labels. It is also evident that Splatter always gives the worst performance due to the bias introduced by nonlinear transformations. Even working with IG would yield a performance boost.

Cytomulate is efficient

As a single CyTOF experiment could contain tens of thousands or millions of cell events along with 30 - 40 protein channels, computational efficiency including estimation and simulation is a significant issue. Specifically, we measured the estimation time (time required to construct the model and estimate the parameters) and simulation time (time needed to generate data from the estimated model) as our metrics for algorithm efficiency as all methods benchmarked involve these two steps. We conducted an analysis on the running time of each method by varying the number of cells and the number of markers using the CD8+ Naive T cells in the CyAnno dataset, which contains 8,414 cell events and 39 protein channels.To be more specific, while evaluating the impact of sample size , we used all 39 protein channels and applied bootstrap sampling to generate expression matrices containing various numbers of cell events. On the other hand, we fixed the number of cell events to 8,414 and sampled the columns to be included when investigating the effect of the number of markers on processing time. Besides including the five models previously described and benchmarked for accuracy, we also included Cytomulate with 5-component GMMs (Restricted Cytomulate) to gauge the impact of model selection on efficiency.

Overall both estimation time and simulation time for all methods under consideration appear to be polynomial with respect to the number of cells and the number of protein channels. Cytomulate is consistently faster than LAMBDA in all situations (Supp. Fig3a-d). For a moderate-size dataset containing 100,000 cells, even Cytomulate with model selection from 9 models takes around only 2 minutes (Supp. Fig. 3a). If restrictions are imposed, the estimation can be done within seconds. On the other hand, LAMBDA, due to the usage of a stochastic EM algorithm for the model estimation, is twice as slow. When sample size reaches one million cells, LAMBDA's estimation time exceeds one hour, which makes it not viable for large scale simulations. Surprisingly, LAMBDA uniquely suffers from poor scalability with regard to the number of markers as compared with other methods (Supp. Fig. 3b)

In terms of simulation time (Supp. Fig. 3c-d), all methods are relatively efficient as compared to estimation time. Despite the efficiency of all methods here, Cytomulate is always best with or without model selection, while Splatter is the least efficient. This characteristic allows Cytomulate to quickly generate many replicates using a single dataset as a reference, which is a common task. As a unique advantage to the Emulation mode, Cytomulate can perform parameter estimation using a small sample and then scale up by simulating a much larger sample. This further improves the scalability of Cytomulate given its efficiency in terms of simulation time. With the increasing capability of CyTOF to produce larger datasets and more channels, Cytomulate is fully capable of handling large datasets.

Cytomulate is robust against cell-type resolution

Although an array of cell type labels is a required input for the emulation mode, we do not expect high granularity of the labels. Namely, it is not necessary to identify all the subtypes of a given cell type as a prerequisite for Cytomulate. In fact, different datasets and analyses call for different cell typing approaches: a large dataset with only B cells may warrant more detailed cell types to reveal their differentiation stages, whereas small whole blood samples may not afford the same resolution with reasonable cluster sizes. In this experiment, we demonstrate that even if the cell identification is not of high resolution, Cytomulate is still able to yield a satisfactory approximation of the data distribution. We included IG, DG, Cytomulate, LAMBDA, and Splatter for comparison.

As we mentioned previously, there are 4 subtypes of B cells in Levine_32dim dataset: mature B cells, plasma B cells, pre-B cells, and pro-B cells. In the first setting, we provided every method with these detailed subtype labels. In the second setting, however, we amalgamated all the labels into “B cells” and fed this information to each method.

We used UMAP to visually compare the results under two settings (Fig. 5). Although all methods except for Splatter seem to give acceptable approximations to the real data when subtypes are given (Fig. 5 and Supp. Fig. 4), only Cytomulate is able to model the 4 major clusters and 1 small cluster present in the real data in the absence of detailed subtype information (Fig. 5b). As an example of an egregiously bad loss of detail, when detailed information is not given, LAMBDA approximates the entire dataset with one Gaussian distribution which puts its mode around the middle of the two separated major clusters where the actual density is low (Fig. 5d).

To quantify the comparisons, we repeated the process 50 times and computed the KL divergence (46) of the simulated datasets of each method under two settings (Fig. 5f). Although more detailed cell subtypes would consistently improve the overall performance, Cytomulate appears to be the most robust among all methods. In fact, both settings of Cytomulate outperform all other methods regardless of cell-typing details. The robustness of Cytomulate makes it an appealing choice for practitioners, especially those who work with public datasets with existing cell types, since Cytomulate offers the greatest flexibility in terms of input. With reasonable cell type input, Cytomulate can accurately capture the prominent characteristics of the data.

Complex simulations:

The core model of Cytomulate accounts for the expressions of each cell type as an essential part for simulation. Beyond the core model and unlike LAMBDA, Cytomulate can easily be extended to incorporate more complex effects commonly encountered in CyTOF experiments such as batch effects, temporal effects, as well as cell differentiations. While some of these effects are oftentimes considered nuisances with corrections desired, the ability to approximate them is crucial for methodological developments in the field of normalization or other pre-processing

techniques. Also, the rising potential of trajectory inference with CyTOF datasets instead of scRNA-seq calls for a more thorough consideration of this topic. This section details the overall design and implementation of these complex simulation scenarios.

Temporal effects:

As discussed in (5), due to cell debris, etc. the sensitivity of a CyTOF machine may vary over time during data generation, causing signals for cells from one batch to fluctuate. In Cytomulate, we assume that the temporal effect for each batch is generated by a function defined on the unit interval such that the value at time zero is always zero, representing that there is no temporal effect at the beginning. We provide users with 3 options to simulate a temporal effect, each with a different level of control over the overall shape of the temporal effect:

1. Brownian Bridge (47): The shape of the entire trajectory is randomly generated from a Brownian Bridge where the ending value is randomly sampled from a normal distribution. To evaluate the function on the unit interval, the Akima spline (48) is used for interpolation. This option gives the users minimal control over the evolution of the temporal effects.

2. Polynomial: This option allows the user to specify the rough shape of the trajectory *via* the coefficients of a polynomial. A linear transformation will be used to make sure the resulting temporal effect starts at 0 and ends at a value randomly generated from a normal distribution.

3. Spline: During the preprocessing steps, it is possible that the users can extract the temporal effect component. The spline option permits the users to add the original temporal effect back to the simulated dataset via a smoothing spline. Since the size of the simulated dataset might differ from that of the original dataset, Cytomulate will automatically rescale the time interval so that the entire trajectory could be retained in the simulated dataset (Fig. 6a).

Batch effects:

Previous research in the field (7), has indicated that the batch effects in CyTOF experiments vary not only by batches but also by cell types and protein channels. To simulate batch effects, we decompose the effects into global batch effects which assume different values for different batches and local batch effects which vary depending on the cell types and protein channels in a similar fashion to the way we decompose effects into main effects and interactions in Analysis of Variance (ANOVA). Further details can be found in the Methods section. As a demonstration, we simulated two batches with 3 cell types using the creation mode of Cytomulate (Fig. 6b).

Cellular trajectories:

To simulate cellular trajectories, Cytomulate adopts a two-step procedure. In the first step, Cytomulate simulates the underlying graph of the cellular trajectories. Given the simulated graph, the actual protein expressions are then sampled in the second step.

Specifically, in the first step, a polytree (a directed acyclic graph whose underlying undirected graph is a tree) or a polyforest (a collection of nonoverlapping polytrees) is generated (49). A tree represents a single trajectory, where edges connect different cell types, allowing cells to differentiate along defined paths. In the Creation Mode, this is done by first randomly grouping the cell types and applying Kruskal's algorithm (50) with random weights to sketch the underlying tree or forest. In the Emulation Mode, this is accomplished by the Clauset-Newman-Moore greedy modularity maximization algorithm (51) (52) followed by the Kruskal's algorithm with the L2 distances among observed average cell means as weights. In both circumstances the root node represents the starting point of differentiation, where its child nodes inherit certain channels that define the entire lineage. To find the root node and convert the graph to a directed acyclic graph (DAG), a depth-first search with a randomly selected node as the root on each graph component is performed.

In the second step, Cytomulate starts the differentiation process. For select channels that change between each pair of nodes, a Brownian Bridge combined with interpolating splines is used to generate the actual protein expressions along the polytree or the polyforest in a fashion similar to how Splatter (29) simulates paths. The time parameter of differentiation is determined by a Beta distribution. This scheme allows smooth transitions between cell types while also ensuring incremental changes along the tree (*i.e.* each tree contains a family of cell types with gradual changes). Further details can be found in the Methods section. For illustration, we used the creation mode of Cytomulate to generate 5 cell types aligned on one polytree (Fig. 6c).

Applications of Cytomulate within a Unified Framework:

Framework Overview:

To ensure the interoperability of Cytomulate with downstream analyses and related workflows, we designed our software package using open standards in the field. Namely, our framework is expression matrix-driven with the optional support for PyCytoData—an easy-to-use interface for CyTOF and downstream analyses (Fig. 2b). Aside from two simulation modes, we support two main usage modes of Cytomulate. For users who prefer batch processing and working with servers, we recommend our command-line interface (CLI). Using a single command with the appropriate arguments and flags, users can generate their desired dataset and save it to disk in the format of their choice. Further, the CLI ensures scalability and interoperability by allowing users to utilize shell scripts and seamlessly integrate their results with other up- or downstream software. Alternatively, our interactive Python mode is designed for those who primarily work within Python and would benefit from the most flexibility. This interface supports fine-tuning of details while allowing users to explore the datasets interactively: from simulation to dimension reduction (DR) and plotting, a few lines of readable code are all they need.

With the wide adoption of CyTOF and the subsequent proliferation of analysis methods, a unified framework is increasingly important. While few methods, except ubiquitous tools such as PCA, have become virtually platform- and language-agnostic, many otherwise good software suffers from poor usability and extensibility. For example, in the process of evaluating Cytomulate, we found that LAMBDA falls far short of having a user-friendly interface: in fact, we spent a significant amount of time adapting the algorithm to our workflow for it to be viably assessed. To address these issues, we first developed the two usage modes as previously mentioned. The CLI uses the operating system's shell as a common denominator to work with other parts of the workflow as necessary, whereas the interactive mode supports outputting a PyCytoData object. The latter brings the benefit of a suite of integrated downstream analysis tools and utility functions, including builtin normalization and CytofDR for DR evaluation (14). Beyond the software itself, our continuous integration and development workflows along with detailed documentation (https://cytomulate.readthedocs.io) ensure that Cytomulate is well maintained and easy for any user to adopt with minimal to no learning curve.

Applications of Cytomulate in DR Benchmarks and Beyond:

As a practical application of Cytomulate, a previous DR benchmark study (14) included extensive usage of the creation mode to evaluate DR methods. Specifically, Wang et al. simulated 425 CyTOF datasets by specifying and systematically varying key features of CyTOF expressions. To further demonstrate the capabilities of Cytomulate for downstream analyses, we accessed and analyzed a subset of the results published on Database of Actionable Immunology's (DBAI) (53–55) CytofDR Playground (https://dbai.biohpc.swmed.edu/cytof-dr-playground/). In particular, we chose samples with 100,000 cells and the following parameters: the number of markers (30, 35, 40), the number of cell differentiation paths (2, 3, 5), and the number of cell types (5, 10, 15, 20, 30). These criteria yield 225 datasets in total with 25 replicates for each configuration. As for DR methods, we focused on the top-performing and most popular ones in the field according to the benchmarks: MDS (56), tSNE (15), UMAP (16), PCA (57), and SAUCIE (18).

Using two major categories (Global Structure Preservation and Local Structure Preservation) and the two subcategories of the Downstream Category, we validated the previous results that MDS is superior for structure preservation whereas UMAP and SAUCIE are better for clustering-based metrics (Supp. Fig. 5). These results are indeed concordant with results obtained from real datasets. A crucial advantage of employing Cytomulate instead of relying solely on real datasets is that users can specify characteristics of each sample, a task hardly achievable in real datasets without artificially subsampling and manipulations. Indeed, a proper benchmark requires the detailed controls necessary to distinguish between various effects, such as the number of cell types and the number of differentiation paths. Cytomulate not only supports all these configurations with both of our interfaces but also produces sensible results for practitioners.

While we defer to the original DR paper for those interested in the details of DR methods and their performances with regard to simulation settings, we would like to emphasize again the unified framework that enables this type of analysis. Users of Cytomulate can easily generate datasets and replicate the aforementioned DR results by using Cytomulate in conjunction with CytofDR through the PyCytoData framework. In fact, the potential goes much beyond these DR methods and benchmarks. With the flexibility of our framework and the open source initiative, other researchers can effortlessly extend the functionalities by simply using the PyCytoData object with their own workflows. For example, a batch normalization study can generate multiple batches using Cytomulate and then analyze them by simply accessing the information from the object. With the rising popularity of CyTOF, we foresee that Cytomulate will become an increasingly more valuable tool as the first step in the workflows of researchers and developers.

Discussion and conclusions

Simulation is a powerful, flexible, and cost-effective tool for performance evaluation, which plays a vital role in methodological development. A common way to test analysis methods is to conduct simulation studies, where generating multiple datasets, with different parameters or assumptions, can be quickly accomplished with minimal cost. The lack of a formal method for accurately synthesizing CyTOF data warrants a dedicated statistical tool with careful considerations of CyTOF's unique characteristics, such as its throughput and the dependency between protein channel expressions.

In this paper, we filled in the aforementioned void by introducing Cytomulate, a tool for simulating CyTOF data accurately and efficiently. It builds on Gaussian Mixture Models (GMM) with a fast and robust estimation procedure and appealing mathematical properties to approximate the real densities of CyTOF channels. Furthermore, Cytomulate features a well-defined probabilistic model to capture prominent aspects of CyTOF data while also incorporating two simulation modes: creation mode and emulation mode to cater to the needs of various researchers. By thoroughly benchmarking Cytomulate's estimation and simulation efficiency, resemblance to real data, and robustness again prior cell-type resolution on six publicly available datasets, we demonstrated that Cytomulate is superior in approximating the overall data distributions regardless of the granularity of the input. Our algorithm further has the versatility of simulating more complicated phenomena commonly encountered in a CyTOF experiment such as signal drifts (temporal effects), batch effects, and cell differentiations, which are partially or totally unsupported by other tools. Finally, we showcased Cytomulate's integration with PyCytoData as an open standard for CyTOF data processing in Python while also highlighting Cytomutate's usage in a DR benchmark study (14). By integrating the philosophy of user-centric software and the superior performance of our probabilistic model, Cytomulate offers a practical step towards addressing the usability problem in the field (14,37).

One potential limitation of this study involves the estimation procedure we chose for the emulation mode. As a trade-off for efficiency, various aspects of CyTOF data such as means, covariances, and cell differentiations are captured in a sequential manner instead of being modeled jointly. Consequently, although the GMMs provide a reasonable approximation to the data distribution of CyTOF due to its universal approximation to densities (58), the estimated parameters are likely biased. Due to the sheer number of cells typically encountered in CyTOF and the complexity of the model, traditional inference methods such as Markov Chain Monte Carlo (MCMC) might not ameliorate this situation. Fortunately, tremendous progress has been made in the field of deep generative models, which has been successfully applied to tasks such as estimating densities and synthesizing realistic-looking images on large datasets (59,60). Some typical examples include normalizing flow (61), variational auto-encoder (62), and generative adversarial networks (63). Exploiting said methods to gain deeper insights into CyTOF datasets thus could be a possible future direction. We are optimistic that our simulation framework as a first in the field will spark further interests in this exciting direction, and our model will serve as a reference for any future work.

Materials and methods

Probabilistic model of Cytomulate:

The probabilistic model of Cytomulate is intended to model the $arcsinh(./5)$-transformed CyTOF data. In this section, we will gradually build up towards the final model aided by what we observed in the section on characteristics of CyTOF data.

Basic model

We will start with the simplest scenario where we only have one batch, $n = 1, \cdots, N$ cell events on only one cell type with measurements on $m = 1, \cdots, M$ protein channels.

We start with a continuous latent variable for the protein expressions:

$$X_n \sim f(\mu, \Sigma),$$

where $f(\mu, \Sigma)$ denotes some distribution with an M-by-1 matrix of mean μ, and an M-by-M matrix of variance-covariance matrix Σ. Given the near-Gaussianity nature of CyTOF data, we use a Gaussian Mixture Model to approximate this distribution in Cytomulate.

Since protein expressions should be non-negative, we further truncate the variable below zero. Mathematically, this can be accomplished by

$$I(X_n \succ 0) \circ X_n,$$

where $\circ$ is the Hadamard product, I is an indicator function, and $\succ$ is a component-wise inequality.

Finally, we incorporate the machine noise into the final protein expressions resulting in the basic model of Cytomulate:

$$Y_n = I(X_n \succ 0) \circ X_n + E_n,$$

where $E_n \sim L$ and L is some error distribution.

Core model

The core model of Cytomulate is only a slight generalization upon the basic model by considering $p = 1, \cdots, P$ cell types in one batch.

We denote the probability of observing the pth cell type by π_p. Then, for the nth cell event, we assume that the cell type is sampled from

$$\Pi_n \sim Categorical(\pi),$$

where $\pi = [\pi_1, \cdots, \pi_P]$ is a P-by-1 probability vector.

Suppose the cell type for the nth cell event is p.We then update the notation for the latent variables accordingly:

$$X_n \sim f^p(\mu^p, \Sigma^p),$$

where $f^p(\mu^p, \Sigma^p)$ denotes some distribution with mean μ^p, and variance-covariance matrix Σ^p.

The computation of the final protein expressions remain unchanged:

$$Y_n = I(X_n \succ 0) \circ X_n + E_n,$$

where $E_n \sim L$ and L is some error distribution.

Complex simulations

Batch effects

To add batch effects, suppose we have $b = 1, \cdots, B$ batches, $m = 1, \cdots, M$ protein channels, and $p = 1, \cdots, P$ cell types. For the bth batch, we have $n = 1, \cdots, N^b$ cell events.

Similar to the setup in the core model, we assume that the cell type of the nth cell event in the bth batch is sampled from

$$\Pi_n^b \sim Categorical(\pi^b),$$

where $\pi = [\pi_1^b, \cdots, \pi_P^b]$ is a P-by-1 probability vector. The corresponding latent variable is then denoted by $X_n^b \sim f^p(\mu^p, \Sigma^p)$. Note that while we allow the cell abundances to vary among batches, we only use one distribution for one cell type.

With this setup, we introduce a set of random vectors, each representing the batch effect on the pth cell type in the bth batch:

$$\Psi^{b,p} \sim N(\psi^b 1_{M\times 1}, \sigma^2 I_{M\times M}),$$

where

$$\psi^b \sim N(0, \sigma_\psi^2)$$

represents the center around which the batch effects on the cell types and protein channels in the bth batch vary. We further add two sets of constraints to ensure the identifiability of the parameters:

$$\sum_{b=1}^{B} \psi^b = 0,$$

$$\sum_{p=1}^{P} \Psi_m^{b,p} = \sum_{m=1}^{M} \Psi_m^{b,p} = 0.$$

Temporal effects

Suppose we have $b = 1, \cdots, B$ batches, and $m = 1, \cdots, M$ protein channels. For the bth batch, we have $n = 1, \cdots, N^b$ cell events. To add temporal effects to the nth cell event in the bth batch, we assume that there is a stochastic process:

$$H^b: [0, 1]^M \mapsto R^M,$$

such that $H^b(0) = 0$. This reflects our belief that there should be no temporal effects at the beginning of an experiment. Denote the realization of H^b by Φ^b, then the temporal effect on the nth cell event is calculated as:

$$\Phi^b(\frac{n}{N^b}).$$

Cellular trajectories

The setup of simulating cellular trajectories is a little more involved. Again, we assume that we have $b = 1, \cdots, B$ batches, $m = 1, \cdots, M$ protein channels, and $p = 1, \cdots, P$ cell types. For the b th batch, we have $n = 1, \cdots, N^b$ cell events.

To get started, we first fix cell type p. We suppose that it differentiates into z^p "children" denoted by

$$p^{c_1}, \cdots, p^{c_{z^p}}.$$

In Cytomulate, we ensure that the collection of "parent-children" relationships form a polytree (a directed acyclic graph whose underlying undirected graph is a tree) or a polyforest (a collection of nonoverlapping polytrees) (49).

To accomplish this, a weighted complete undirected graph with cell types as nodes are constructed. In the Creation Mode, the weights on the edges are randomly assigned. In the Emulation Mode, the L2 distances among observed average protein expressions of all cell types are used as weights.

Next, we carve out a spanning tree or a spanning forest via the Clauset-Newman-Moore greedy modularity maximization algorithm (51) (52) followed by the Kruskal's algorithm (50). Finally, to convert the graph to a directed acyclic graph (DAG), a depth-first search with a randomly selected node as the root on each graph component is performed.

Now fix the "child" cell type p^c. We construct the corresponding cellular trajectory from cell type p to cell type p^c in a similar fashion to how we construct temporal effects. We start with a stochastic process

$$G^{p \to p^c}: [0, 1]^M \mapsto R^M,$$

such that the mth component satisfies

$$G_m^{p\to p^c}(0) = 0, \text{ and } G_m^{p\to p^c}(1) = \mu_m^{p^c} - \mu_m^p,$$

where μ_m^p and $\mu_m^{p^c}$ are the mean expression levels of the mth protein channel of cell type p and cell type p^c, respectively. Inspired by Splatter (29), we adopt Brownian Bridges to generate the actual realizations of cellular trajectories denoted by $g^{p\to p^c}$.

When generating the final protein expressions for the nth cell event in the bth batch whose cell type is p, we first sample the "child" cell type p^c from

$$\Omega_n^{b,p} \sim Categorical(\omega^p),$$

where $\omega^p = [\omega_1^p, \cdots, \omega_{z^p}^p]$ is a z^p-by-1 probability vector. Notice that the parameters of the distribution only vary with cell types.

We then sample its mth component of the actual position t_n^b on the cellular trajectory $g^{p\to p^c}$

$$T_{n,m}^b \sim Beta(\alpha^b, \beta^b).$$

Final model

With these setups, the final protein expressions for the nth cell event in the bth batch whose cell type is p and who is differentiating into cell type p^c can be simulated via the following two steps:

$$Z_n^b = X_n^b + g^{p\to p^c}(T_n^p) + \Psi^{b,p} + \Phi^b,$$

$$Y_n^b = I(Z_n^b > 0) \circ Z_n^b + E_n^b,$$

where $E_n^b \sim L$ and L is some error distribution.

Evaluation metrics

In this section, we elaborate on the metrics we used in the comparison studies.

Suppose in the simulated batch we have $p = 1, \cdots, P$ cell types. Then, for the pth cell type, we calculate the sample mean m_p and the sample covariance as c_p. We carry out the same procedure on the observed dataset and find the corresponding m_p^r, and c_p^r. The L2 distance between the

simulated mean and the observed mean is found via $dm_p = ||m_p - m_p^r||_2$. The Frobenius distance (64) between the simulated covariance and the observed covariance is calculated as $dc_p = ||c_p - c_p^r||_F$. And, the KL divergence dk_p is estimated following the method proposed in (46). Finally, the discrepancy measures for all the cell types are aggregated by a weighted average where the weights are decided by the cell abundances.

Software implementation details

All analyses, plotting, and benchmarks were performed using the R (4.0.2) and Python (3.8.8) programming environments in MacOS 12.4. All default settings were used for Cytomulate and Splatter. For the lack of defaults in LAMBDA, we set the number of iterations to 100 and skipped the multinomial sampling process given that all cell types were known.

Cytomulate is implemented and supported in Python (v3.5 or later), and the CLI is available on all platforms and systems where a compatible python interpreter is present with the proper package installations. The detailed documentation and tutorials are freely available and hosted on the cloud using ReadTheDocs (https://cytomulate.readthedocs.io). All software features have been thoroughly tested with the *pytest* framework.

For the efficiency benchmarks, we carried out the experiment on a MacBook Pro with a 2.9 GHz Quad-Core Intel Core i7 processor running MacOS 12.4. Since Cytomulate is implemented in Python while Splatter and LAMBDA are R packages, we used Python's time module to measure the performance of Cytomulate-related methods and Sys.time function in R to time Splatter and LAMBDA. All times are reported in seconds.

Statistical analyses

We employed Pearson correlation coefficients for all correlation matrices and plots. For all rank-based metrics, higher ranks represent better performance. In cases where tied ranks occurred, we used the default behavior of averaging tied ranks. For all boxplots appearing in this study, box boundaries represent interquartile ranges, whiskers extend to the most extreme data point which is no more than 1.5 times the interquartile range, and the line in the middle of the box represents the median. All barplots showing discrepancies among all methods under comparison on various evaluation metrics were plotted on the logarithm scale. To increase visual clarity, we added one to all the data points before plotting so that the results would all be positive. Error bars with width being two standard deviations were also added to further facilitate comparison.

Data availability

The Levine_32dim, Levine_13dim, and Samusik datasets can be accessed from the HDCytoData package in R (https://github.com/lmweber/HDCytoData). The CyAnno (32) and LG (45) can be accessed from their respective publications. The Covid cohort is hosted on DBAI (https://dbai.biohpc.swmed.edu/) with the following accession codes: CyTOF_00000000000001 to CyTOF_00000000000012.

Code availability

The Cytomulate algorithm is available at cytomulate.readthedocs.io, and also shared through the Database of Actionable Immunology (dbai.biohpc.swmed.edu) (53–55).

Figures and Table Legends

Fig. 1 Characteristics of a typical CyTOF dataset. All four panels are from the Levine_13dim as an example. (a) Heatmap of channel expression for all naive CD4+ T cells. (b) Density plot of all channel expressions from naive CD4+ T cells. (c) Correlation matrix of all channels as computed with Pearson correlation coefficients. (d) Scatter plot of all cells using the first two principal components and colored with provided cell types.

Fig. 2 Overview of cytomulate model and software pipeline. (a) The model structure and input and output pipelines of Cytomulate. The input stage demonstrates the vectors or parameters needed for the algorithm and Cytomulate uses the provided information accordingly for estimation and simulation. The outputs are in the forms of expression matrices for additional downstream analyses. (b) The software pipeline of Cytomulate. The flowchart showcases Cytomulate's role and integration with industry standard tools in every step of the research and development process.

Fig. 3 Detailed results on simulation performance benchmark with the CyAnno dataset. (a) MSE of mean expression of all channels for each cell type between the original and simulated datasets. (b) MSE of the covariance matrix of all channels for each cell type between the original and simulated datasets. (c) KL divergence between the original and simulated datasets.

Fig. 4 Detailed results on embedding and overall performance. (a) The UMAP embedding of the original CyAnno dataset. In (b), (c), and (d), the UMAP embeddings of simulated datasets

generated by LAMBDA, Cytomulate, and Splatter are plotted respectively. (e) The overall performance of all methods across all datasets tested. Larger circles and brighter colors indicate higher ranks with superior performance.

Fig. 5 Detailed results on performance with different cell subtypes. For all analyses and visualizations in this plot, Levine_32dim dataset with B cells only is used**.** (a) UMAP embedding of Cytomulate simulation with B cell subtype labels. (b) UMAP embedding of Cytomulate simulation without B cell subtype labels. (c) UMAP embedding of LAMBDA simulation with B cell subtype labels. (d) UMAP embedding of LAMBDA simulation without B cell subtype labels. (e) UMAP embedding of the original B cell expressions. (f) KL divergence between the original and simulated expressions. A lower value represents more accurate overall simulation.

Fig. 6 Examples of effects addressed in complex simulations. (a) Example temporal effects generated using the three methods: Brownian Bridge, Polynomial, and Spline. (b) Example of a simulated sample batch effect. Color difference represents batches, and point shape indicates cell types. (c) Example of the cell differentiation trajectory generated by Cytomulate. The trajectories are noted with light blue arrows.

Table 1 Datasets used in the study. A table that provides the dataset name, species, the anatomic site, and its source for accession.

Supp. Fig. 1 Examples of complex characteristics observed in a real dataset, Levine_13dim. (a) The temporal effect present in the dataset, as measured by the signal drift of five bead channels. (b) The corrected time effects after bead normalization. (c) The batch effect of the two patients in Levine_13dim visualized with an interaction plot in which the mean expression of protein channels change from sample to sample.

Supp. Fig. 2 Additional UMAP embeddings of simulated datasets in emulation mode based on the CyAnno dataset. (a) The embedding of the simulated dataset using the IG method. (b) The embedding of the simulated dataset using the DG method.

Supp. Fig. 3 Runtime benchmarks of all simulation methods in the study. In (a) and (b), we measured the estimation time of each method with various numbers of cells and markers

respectively. The estimation time includes construction of the model object and estimation of parameters used in each model based on the dataset given. In (c) and (d), we benchmarked the simulation time of each method against different numbers of cells and markers. Simulation time is defined as the time necessary to generate data from the estimated parameters.

Supp. Fig. 4 Additional results on performance with different cell subtypes. All parts of the figure utilizes the Levine_32dim dataset with B cells only. (a) UMAP embedding of IG simulation with B cell subtype labels. (b) UMAP embedding of IG simulation without B cell subtype labels. (c) UMAP embedding of DG simulation with B cell subtype labels. (d) UMAP embedding of DG simulation without B cell subtype labels. (e) UMAP embedding of Splatter simulation with B cell subtype labels. (f) UMAP embedding of IG simulation with B cell subtype labels.

Supp. Fig. 5 Performance of DR methods benchmarked with Cytomulate's creation mode. Five methods are included, and all rankings are accessed from CytofDR Playground. "*" indicates 10% downsampling of the datasets used for DR and evaluation. (a) Rankings of DR methods using global structure preservation metrics. (b) Rankings of DR methods using local structure preservation metrics. (c) Rankings of DR methods using cluster reconstruction metrics. (d) Rankings of DR methods using cluster concordance metrics.

Funding Sources

This study was supported by the National Institutes of Health (NIH) [CCSG 5P30CA142543/TW, NIH 1R01CA258584/TW, XW, U01AI156189/TW], Cancer Prevention Research Institute of Texas [CPRIT RP190208/TW, XW], and Southern Methodist University [Dean's Research Council Award 2022/XW, TW].

Author Contributions

Y.Y. performed all analyses. Y.Y. and K.W. created the *Cytomulate* Package. Z.L provided analysis pipelines for LAMBDA. X.W. conceived the study. T.W. and X.W. designed and supervised the whole study. All authors wrote the manuscript.

Declaration of Interests

The authors declare no competing interests related to this work.

Bibliography

1. Lou X, Zhang G, Herrera I, Kinach R, Ornatsky O, Baranov V, et al. Polymer-based elemental tags for sensitive bioassays. Angew Chem Int Ed. 2007;46(32):6111–4.
2. Spitzer MH, Nolan GP. Mass cytometry: single cells, many features. Cell. 2016 May 5;165(4):780–91.
3. Bandura DR, Baranov VI, Ornatsky OI, Antonov A, Kinach R, Lou X, et al. Mass cytometry: technique for real time single cell multitarget immunoassay based on inductively coupled plasma time-of-flight mass spectrometry. Anal Chem. 2009 Aug 15;81(16):6813–22.
4. Chester C, Maecker HT. Algorithmic Tools for Mining High-Dimensional Cytometry Data. J Immunol. 2015 Aug 1;195(3):773–9.
5. Finck R, Simonds EF, Jager A, Krishnaswamy S, Sachs K, Fantl W, et al. Normalization of mass cytometry data with bead standards. Cytometry A. 2013 May;83(5):483–94.
6. Trussart M, Teh CE, Tan T, Leong L, Gray DH, Speed TP. Removing unwanted variation with CytofRUV to integrate multiple CyTOF datasets. eLife. 2020 Sep 7;9.
7. Van Gassen S, Gaudilliere B, Angst MS, Saeys Y, Aghaeepour N. Cytonorm: A normalization algorithm for cytometry data. Cytometry A. 2020 Mar;97(3):268–78.
8. Shaham U, Stanton KP, Zhao J, Li H, Raddassi K, Montgomery R, et al. Removal of batch effects using distribution-matching residual networks. Bioinformatics. 2017 Aug 15;33(16):2539–46.
9. Upadhyay U, Jain A. Removal of Batch Effects using Generative Adversarial Networks. arXiv. 2019;
10. Van Gassen S, Callebaut B, Van Helden MJ, Lambrecht BN, Demeester P, Dhaene T, et al. FlowSOM: Using self-organizing maps for visualization and interpretation of cytometry data. Cytometry A. 2015 Jul;87(7):636–45.
11. Abe K, Minoura K, Maeda Y, Nishikawa H, Shimamura T. Model-based clustering for flow and mass cytometry data with clinical information. BMC Bioinformatics. 2020 Sep 17;21(Suppl 13):393.
12. Ji D, Nalisnick E, Qian Y, Scheuermann RH, Smyth P. Bayesian trees for automated cytometry data analysis. BioRxiv. 2018 Sep 19;
13. Dai Y, Xu A, Li J, Wu L, Yu S, Chen J, et al. CytoTree: an R/Bioconductor package for analysis and visualization of flow and mass cytometry data. BMC Bioinformatics. 2021 Mar 22;22(1):138.

14. Wang K, Yang Y, Wu F, Song B, Wang X, Wang T. A Comparison of Dimension Reduction Methods for Cytometry by Time-of-Flight Data. BioRxiv. 2022 Apr 27;

15. Laurens van der M, Geoffrey H. Visualizing Data using t-SNE. J Mach Learn Res. 2008;

16. McInnes L, Healy J, Melville J. UMAP: Uniform Manifold Approximation and Projection for Dimension Reduction. arXiv. 2018;

17. Hastie T, Tibshirani R, Friedman J. The Elements of Statistical Learning: Data Mining, Inference, and Prediction, Second Edition. Springer; 2008.

18. Amodio M, Srinivasan K, van Dijk D, Mohsen H, Yim K, Muhle R, et al. Exploring Single-Cell Data with Multitasking Deep Neural Networks. BioRxiv. 2017 Dec 19;

19. Ding J, Condon A, Shah SP. Interpretable dimensionality reduction of single cell transcriptome data with deep generative models. Nat Commun. 2018 May 21;9(1):2002.

20. Hahne F, LeMeur N, Brinkman RR, Ellis B, Haaland P, Sarkar D, et al. flowCore: a Bioconductor package for high throughput flow cytometry. BMC Bioinformatics. 2009 Apr 9;10:106.

21. Hao Y, Hao S, Andersen-Nissen E, Mauck WM, Zheng S, Butler A, et al. Integrated analysis of multimodal single-cell data. Cell. 2021 Jun 24;184(13):3573-3587.e29.

22. Seiler C, Ferreira A-M, Kronstad LM, Simpson LJ, Le Gars M, Vendrame E, et al. CytoGLMM: conditional differential analysis for flow and mass cytometry experiments. BMC Bioinformatics. 2021 Mar 22;22(1):137.

23. Arend L, Bernett J, Manz Q, Klug M, Lazareva O, Baumbach J, et al. A systematic comparison of novel and existing differential analysis methods for CyTOF data. Brief Bioinformatics. 2022 Jan 17;23(1).

24. Mitra R, Müller P, Qiu P, Ji Y. Bayesian hierarchical models for protein networks in single-cell mass cytometry. Cancer Inform. 2014 Dec 10;13(Suppl 4):79–89.

25. Abdelaal T, Höllt T, van Unen V, Lelieveldt BPF, Koning F, Reinders MJT, et al. CyTOFmerge: integrating mass cytometry data across multiple panels. Bioinformatics. 2019 Oct 15;35(20):4063–71.

26. Robinson MD, McCarthy DJ, Smyth GK. edgeR: a Bioconductor package for differential expression analysis of digital gene expression data. Bioinformatics. 2010 Jan 1;26(1):139–40.

27. Lun ATL, Bach K, Marioni JC. Pooling across cells to normalize single-cell RNA sequencing data with many zero counts. Genome Biol. 2016 Apr 27;17:75.

28. Vallejos CA, Marioni JC, Richardson S. BASiCS: Bayesian Analysis of Single-Cell Sequencing Data. PLoS Comput Biol. 2015 Jun 24;11(6):e1004333.

29. Zappia L, Phipson B, Oshlack A. Splatter: simulation of single-cell RNA sequencing data.

Genome Biol. 2017 Sep 12;18(1):174.

30. Data scientist's primer to analysis of mass cytometry data [Internet]. [cited 2022 May 26]. Available from: https://biosurf.org/cytof_data_scientist.html

31. Levine JH, Simonds EF, Bendall SC, Davis KL, Amir ED, Tadmor MD, et al. Data-Driven Phenotypic Dissection of AML Reveals Progenitor-like Cells that Correlate with Prognosis. Cell. 2015 Jul 2;162(1):184–97.

32. Kaushik A, Dunham D, He Z, Manohar M, Desai M, Nadeau KC, et al. CyAnno: A semi-automated approach for cell type annotation of mass cytometry datasets. Bioinformatics. 2021 May 25;

33. Lukas M. CS. HDCytoData. Bioconductor. 2018;

34. Leipold MD, Obermoser G, Fenwick C, Kleinstuber K, Rashidi N, McNevin JP, et al. Comparison of CyTOF assays across sites: Results of a six-center pilot study. J Immunol Methods. 2018 Feb;453:37–43.

35. Street K, Risso D, Fletcher RB, Das D, Ngai J, Yosef N, et al. Slingshot: cell lineage and pseudotime inference for single-cell transcriptomics. BMC Genomics. 2018 Jun 19;19(1):477.

36. Trapnell C, Cacchiarelli D, Grimsby J, Pokharel P, Li S, Morse M, et al. The dynamics and regulators of cell fate decisions are revealed by pseudotemporal ordering of single cells. Nat Biotechnol. 2014 Apr;32(4):381–6.

37. Saelens W, Cannoodt R, Todorov H, Saeys Y. A comparison of single-cell trajectory inference methods. Nat Biotechnol. 2019 May;37(5):547–54.

38. Bishop CM. Pattern Recognition and Machine Learning (Information Science and Statistics). Softcover reprint of the original 1st ed. 2006. Springer; 2016.

39. Liu X, Song W, Wong BY, Zhang T, Yu S, Lin GN, et al. A comparison framework and guideline of clustering methods for mass cytometry data. Genome Biol. 2019 Dec 23;20(1):297.

40. Weber LM, Robinson MD. Comparison of clustering methods for high-dimensional single-cell flow and mass cytometry data. Cytometry A. 2016 Dec 19;89(12):1084–96.

41. Lee H-C, Kosoy R, Becker CE, Dudley JT, Kidd BA. Automated cell type discovery and classification through knowledge transfer. Bioinformatics. 2017 Jun 1;33(11):1689–95.

42. Zhang Z, Luo D, Zhong X, Choi JH, Ma Y, Wang S, et al. SCINA: A Semi-Supervised Subtyping Algorithm of Single Cells and Bulk Samples. Genes (Basel). 2019 Jul 12;10(7).

43. Schwarz G. Estimating the dimension of a model. Ann Statist. 1978 Mar;6(2):461–4.

44. Samusik N, Good Z, Spitzer MH, Davis KL, Nolan GP. Automated mapping of phenotype space with single-cell data. Nat Methods. 2016 Jun;13(6):493–6.

45. Hawley D, Ding J, Thotakura S, Haskett S, Aluri H, Kublin C, et al. RNA-Seq and CyTOF immuno-profiling of regenerating lacrimal glands identifies a novel subset of cells expressing muscle-related proteins. PLoS ONE. 2017 Jun 29;12(6):e0179385.

46. Perez-Cruz F. Kullback-Leibler divergence estimation of continuous distributions. 2008 IEEE International Symposium on Information Theory. IEEE; 2008. p. 1666–70.

47. Grimmett GR, Stirzaker DR. Probability and Random Processes. 3rd ed. Oxford: Oxford University Press; 2001.

48. Akima H. A new method of interpolation and smooth curve fitting based on local procedures. J ACM. 1970 Oct 1;17(4):589–602.

49. Dasgupta S. Learning Polytrees. arXiv. 2013;

50. Cormen TH, Leiserson CE, Rivest RL, Stein C. Introduction to Algorithms, 3rd Edition (The MIT Press). 3rd ed. Cambridge, Mass: The MIT Press; 2009.

51. Clauset A, Newman MEJ, Moore C. Finding community structure in very large networks. Phys Rev E. 2004 Dec 6;70(6).

52. Hagberg AA, Schult DA, Swart PJ. Exploring Network Structure, Dynamics, and Function using NetworkX. In: Varoquaux G, Vaught T, Millman J, editors. Proceedings of the 7th Python in Science conference. SciPy conference proceedings; 2008. p. 11–5.

53. Zhu J, Gouru A, Wu F, Berzofsky JA, Xie Y, Wang T. BepiTBR: T-B reciprocity enhances B cell epitope prediction. iScience. 2022 Feb 18;25(2):103764.

54. Zhang Z, Xiong D, Wang X, Liu H, Wang T. Mapping the functional landscape of T cell receptor repertoires by single-T cell transcriptomics. Nat Methods. 2021 Jan 6;18(1):92–9.

55. Lu T, Zhang Z, Zhu J, Wang Y, Jiang P, Xiao X, et al. Deep learning-based prediction of the T cell receptor–antigen binding specificity. Nat Mach Intell. 2021 Sep 23;

56. Kruskal JB. Multidimensional scaling by optimizing goodness of fit to a nonmetric hypothesis. Psychometrika. 1964 Mar;29(1):1–27.

57. Ringnér M. What is principal component analysis? Nat Biotechnol. 2008 Mar;26(3):303–4.

58. Goodfellow I, Bengio Y, Courville A. Deep Learning (Adaptive Computation and Machine Learning series). Illustrated. Cambridge, Massachusetts: The MIT Press; 2016.

59. Ruthotto L, Haber E. An Introduction to Deep Generative Modeling. arXiv. 2021;

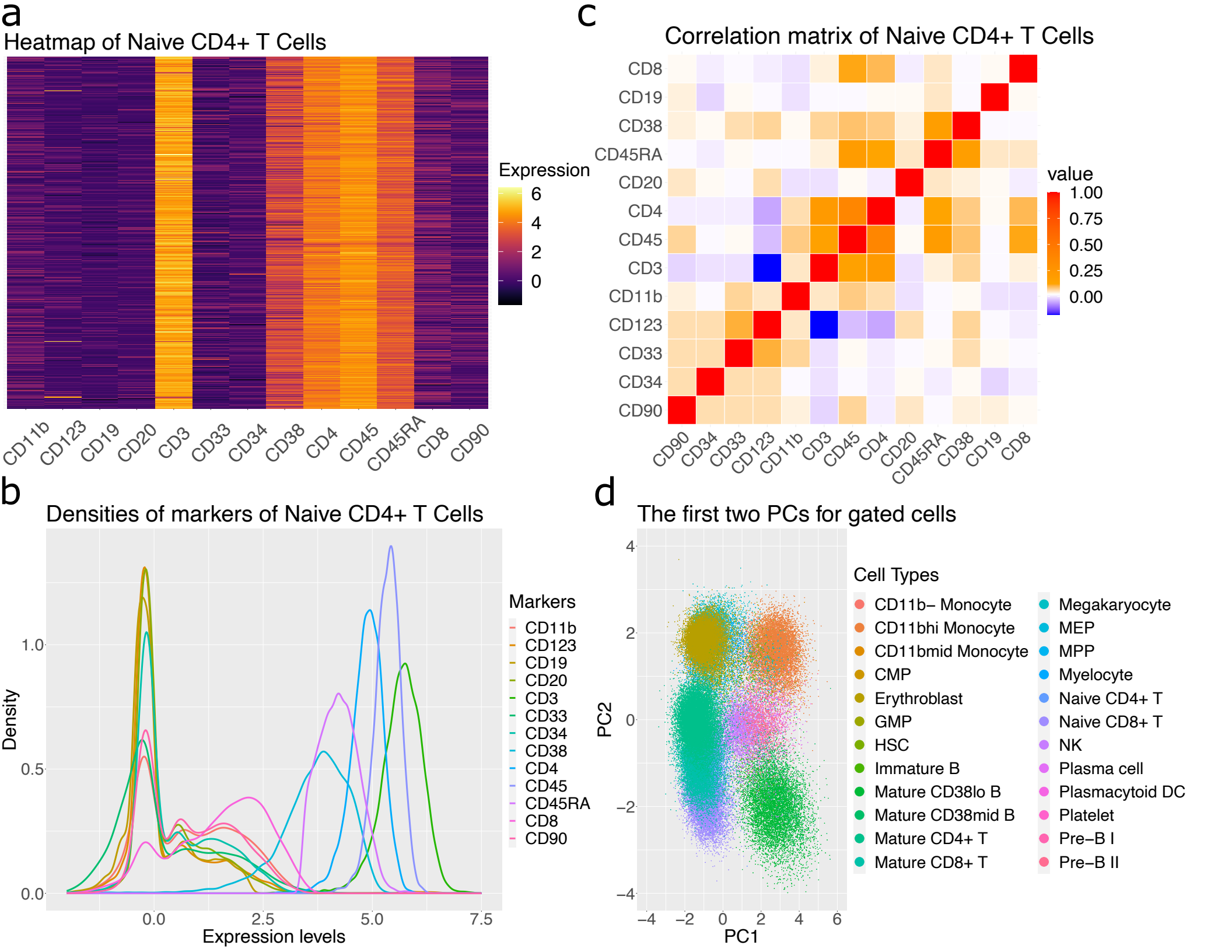
60. Bond-Taylor S, Leach A, Long Y, Willcocks CG. Deep Generative Modelling: A Comparative Review of VAEs, GANs, Normalizing Flows, Energy-Based and Autoregressive Models. arXiv. 2021;

61. Rezende DJ, Mohamed S. Variational Inference with Normalizing Flows. arXiv. 2015;

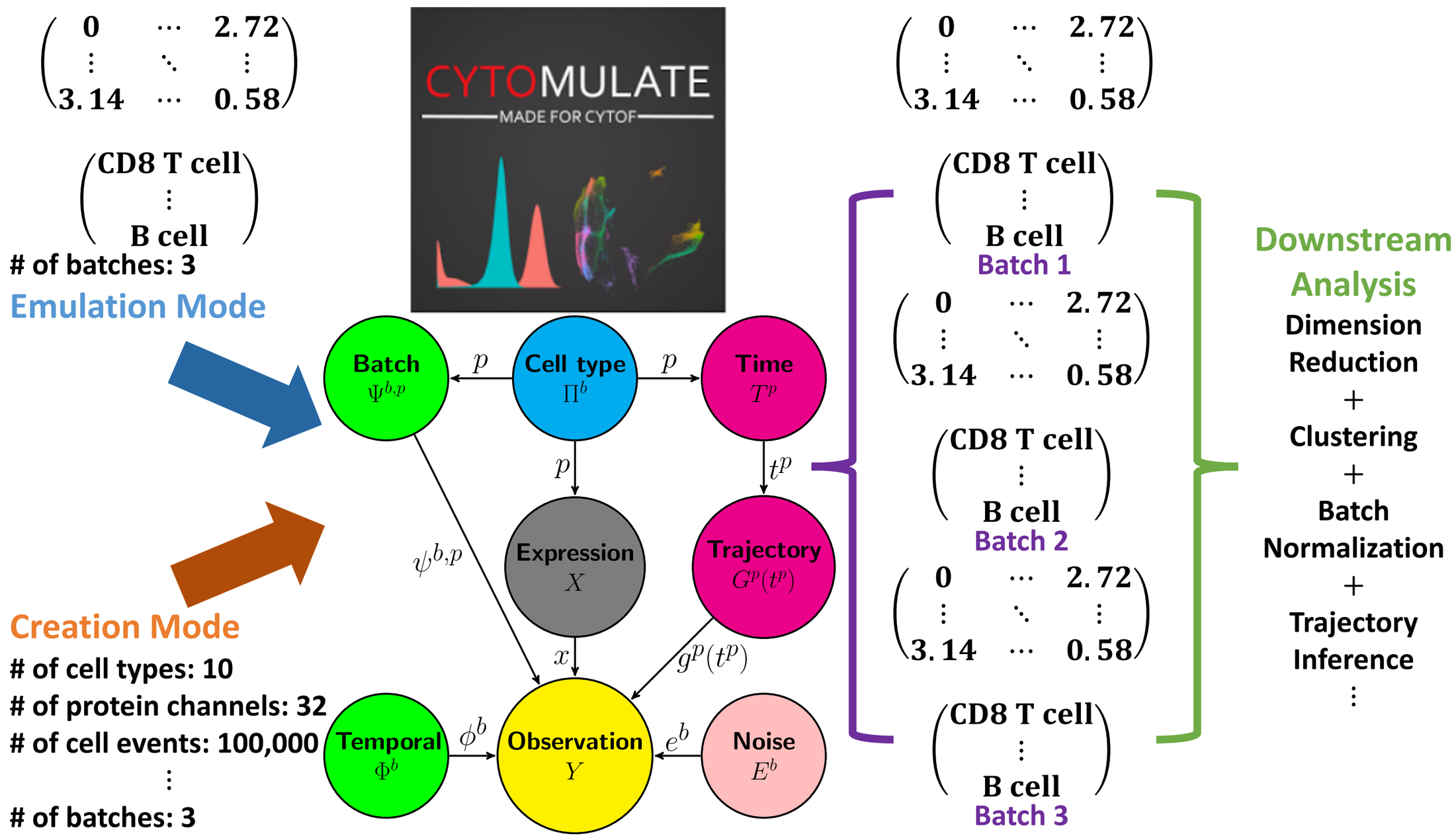
62. Kingma DP, Welling M. Auto-Encoding Variational Bayes. arXiv. 2013;

63. Goodfellow IJ, Pouget-Abadie J, Mirza M, Xu B, Warde-Farley D, Ozair S, et al. Generative Adversarial Networks. arXiv. 2014;

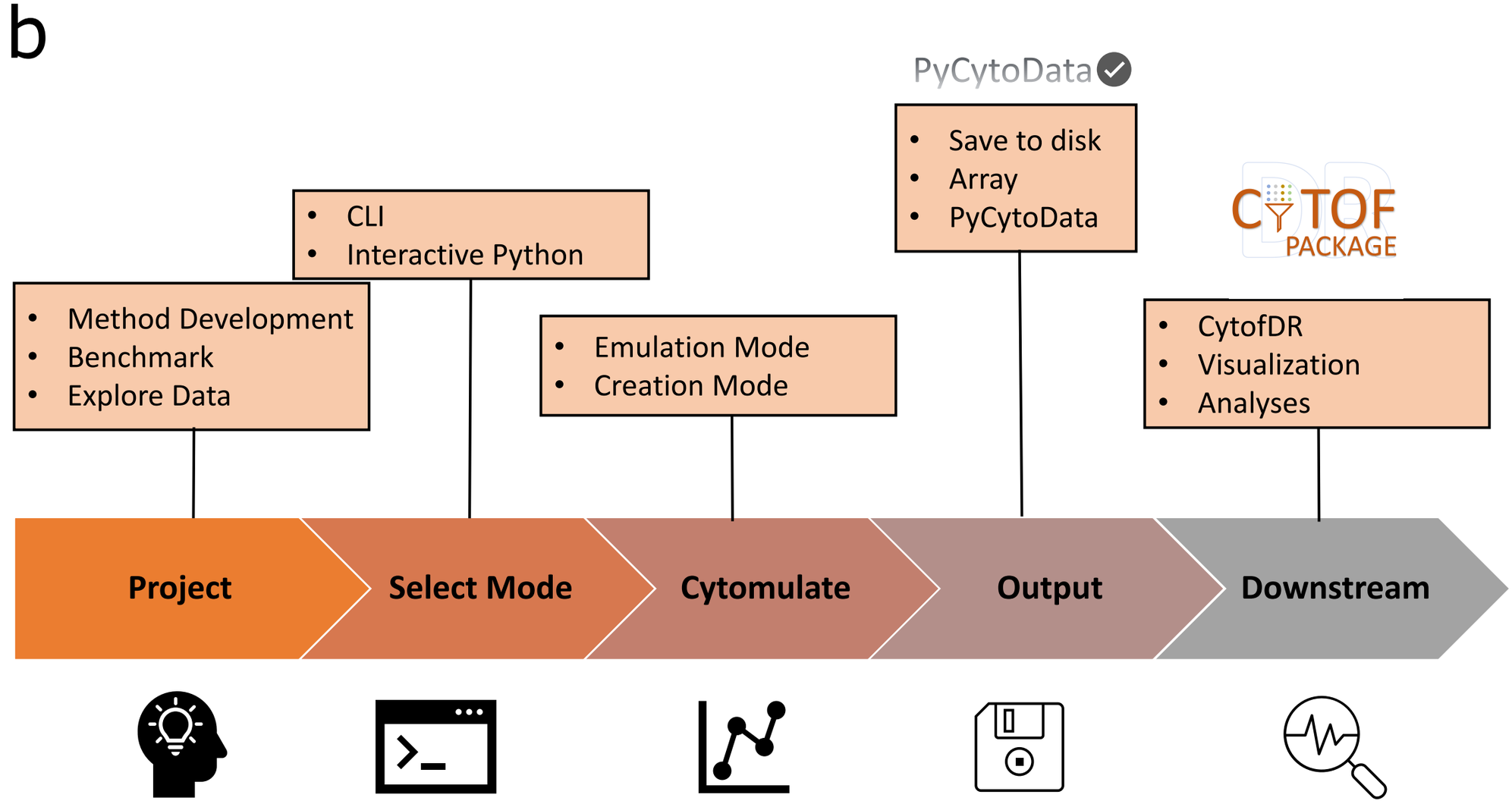
64. Trefethen LN, David Bau III. Numerical Linear Algebra. 1st ed. Philadelphia: SIAM: Society for Industrial and Applied Mathematics; 1997.

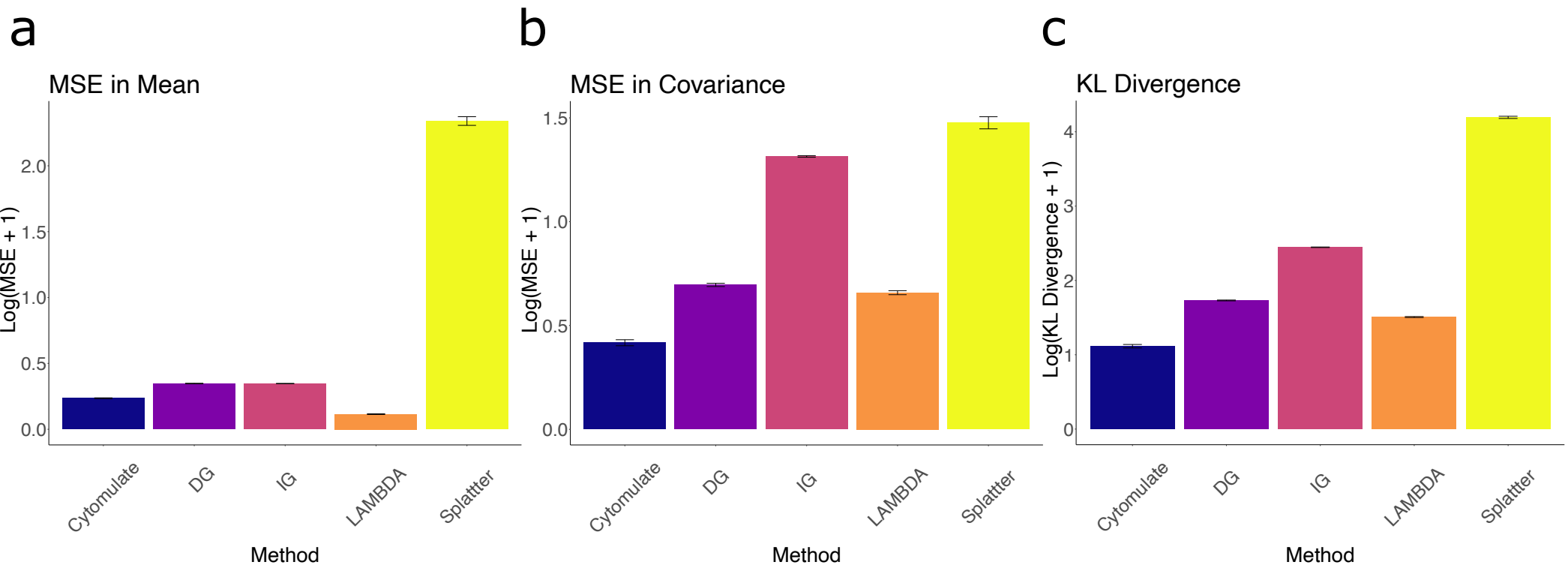

a
Heatmap of Naive CD4+ T Cells
Expression
6
4
2
0
CD11b
CD123
CD19
CD20
CD3
CD33
CD34
CD38
CD4
CD45
CD45RA
CD8
CD90
c
Correlation matrix of Naive CD4+ T Cells
value
1.00
0.75
0.50
0.25
0.00
CD8
CD19
CD38
CD45RA
CD20
CD4
CD45
CD3
CD11b
CD123
CD33
CD34
CD90
b
Densities of markers of Naive CD4+ T Cells
Density
1.0
0.5
0.0
0.0
2.5
5.0
7.5
Expression levels
Markers
CD11b
CD123
CD19
CD20
CD3
CD33
CD34
CD38
CD4
CD45
CD45RA
CD8
CD90
d
The first two PCs for gated cells
PC2
4
2
0
−2
−4
−4
−2
0
2
4
6
PC1
Cell Types
CD11b− Monocyte
CD11bhi Monocyte
CD11bmid Monocyte
CMP
Erythroblast
GMP
HSC
Immature B
Mature CD38lo B
Mature CD38mid B
Mature CD4+ T
Mature CD8+ T
Megakaryocyte
MEP
MPP
Myelocyte
Naive CD4+ T
Naive CD8+ T
NK
Plasma cell
Plasmacytoid DC
Platelet
Pre−B I
Pre−B II

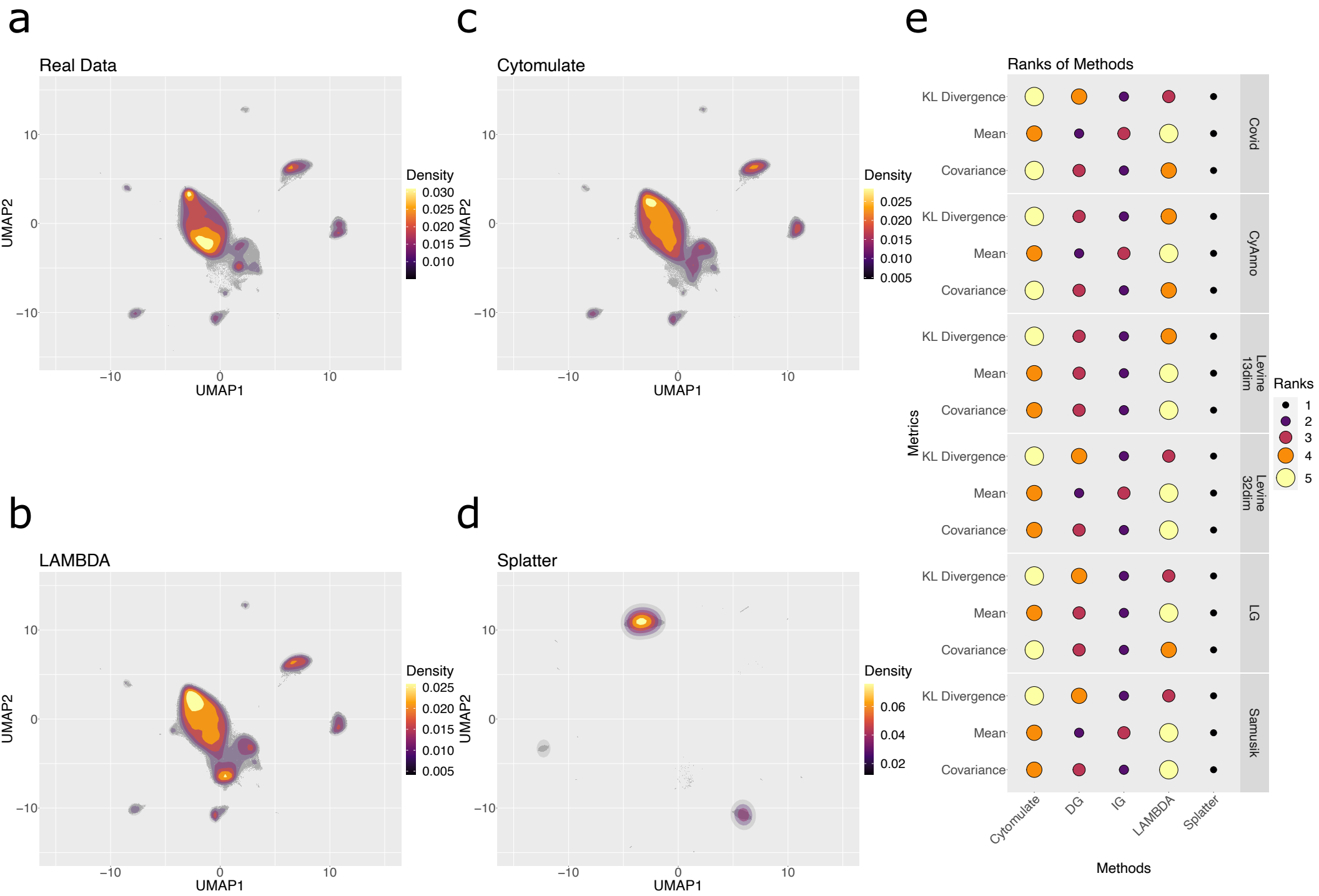
a

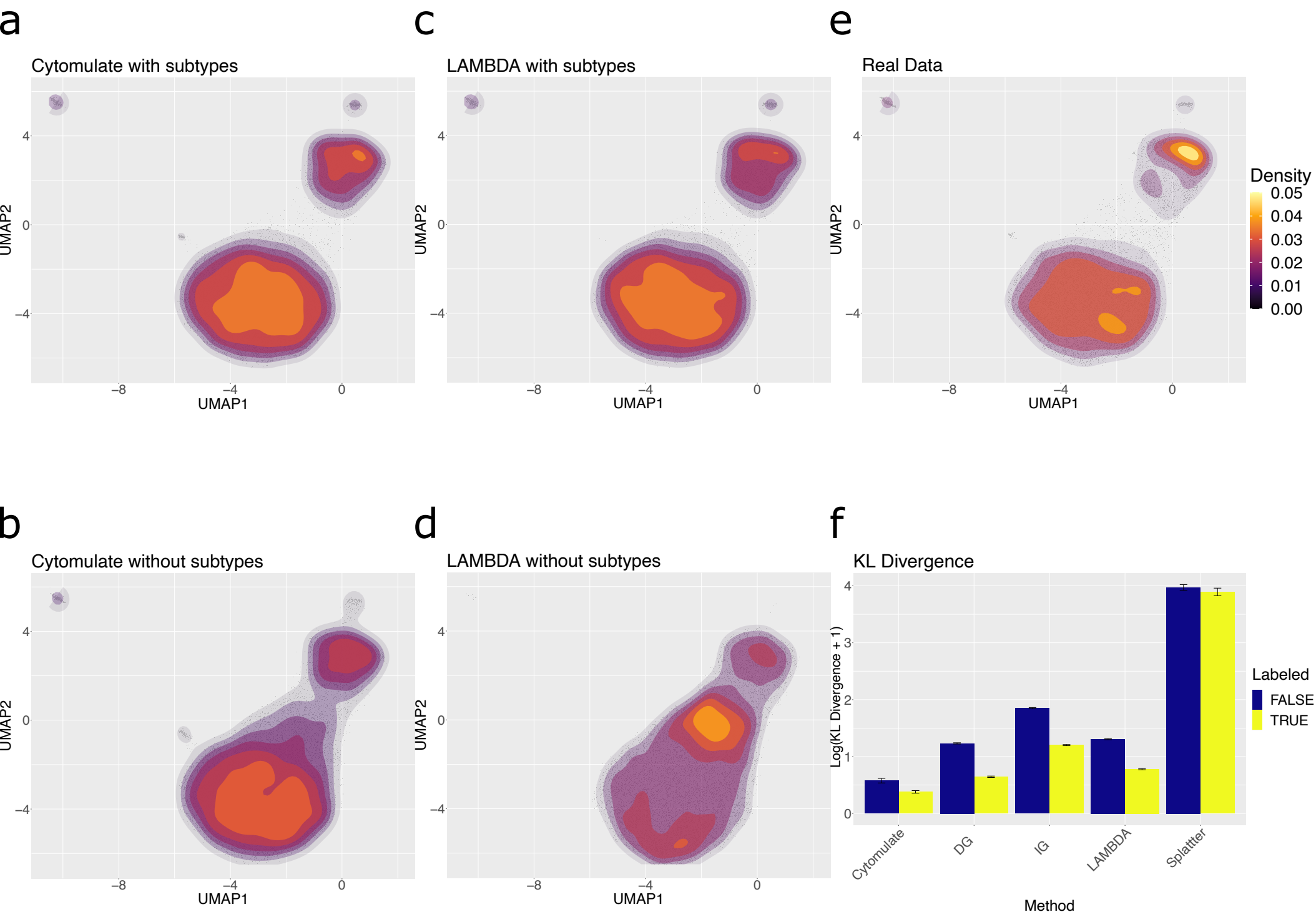

CYTOMULATE
MADE FOR CYTOF
# of batches: 3
Emulation Mode
Creation Mode
# of cell types: 10
# of protein channels: 32
# of cell events: 100,000
# of batches: 3
Batch
Cell type
Time
Expression
Trajectory
Temporal
Observation
Noise
Batch 1
Batch 2
Batch 3
Downstream Analysis
Dimension Reduction
+
Clustering
+
Batch Normalization
+
Trajectory Inference

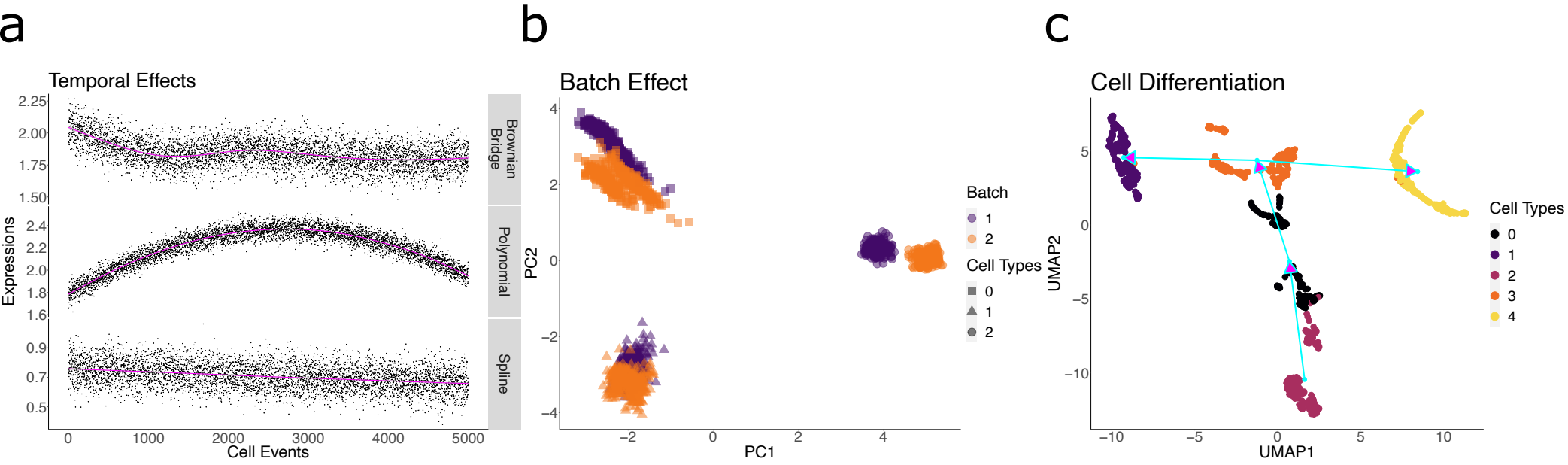
b


PyCytoData
CYTOF PACKAGE
Method Development
Benchmark
Explore Data
CLI
Interactive Python
Emulation Mode
Creation Mode
Save to disk
Array
PyCytoData
CytofDR
Visualization
Analyses
Project
Select Mode
Cytomulate
Output
Downstream


a
MSE in Mean
Log(MSE + 1)
0.0
0.5
1.0
1.5
2.0
Cytomulate
DG
IG
LAMBDA
Splattter
Method
b
MSE in Covariance
Log(MSE + 1)
0.0
0.5
1.0
1.5
Cytomulate
DG
IG
LAMBDA
Splattter
Method
c
KL Divergence
Log(KL Divergence + 1)
0
1
2
3
4
Cytomulate
DG
IG
LAMBDA
Splattter
Method


a
Real Data
UMAP2
UMAP1
Density
0.030
0.025
0.020
0.015
0.010
c
Cytomulate
UMAP2
UMAP1
Density
0.025
0.020
0.015
0.010
0.005
e
Ranks of Methods
Metrics
KL Divergence
Mean
Covariance
Covid
CyAnno
Levine 13dim
Levine 32dim
LG
Samusik
Ranks
1
2
3
4
5
Cytomulate
DG
IG
LAMBDA
Splatter
Methods
b
LAMBDA
UMAP2
UMAP1
Density
0.025
0.020
0.015
0.010
0.005
d
Splatter
UMAP2
UMAP1
Density
0.06
0.04
0.02


a
Cytomulate with subtypes
c
LAMBDA with subtypes
e
Real Data
b
Cytomulate without subtypes
d
LAMBDA without subtypes
f
KL Divergence
UMAP1
UMAP2
Density
0.05
0.04
0.03
0.02
0.01
0.00
Log(KL Divergence + 1)
Method
Cytomulate
DG
IG
LAMBDA
Splatter
Labeled
FALSE
TRUE


a
Temporal Effects
Expressions
Cell Events
Brownian Bridge
Polynomial
Spline
b
Batch Effect
PC1
PC2
Batch
1
2
Cell Types
0
1
2
c
Cell Differentiation
UMAP1
UMAP2
Cell Types
0
1
2
3
4

Dataset	Species	Anatomic Site
Levine_32dim	Human	Bone Marrow
Levine_13dim	Human	Bone Marrow
Samusik	Mouse	Bone Marrow
CyAnno	Human	Peripheral Blood
Covid	Human	Peripheral Blood
LG	Mouse	Lacrimal Gland

Source	Number of Markers
HDCytoData	32
HDCytoData	13
HDCytoData	39
https://flowrepository.org/id/FR-FCM-Z2V9	39
https://dbai.biohpc.swmed.edu/cytof-database.php	43
https://flowrepository.org/experiments/1177/download_ziped_files	31

Number of Cells Events	Number of Cell Types
72,463	32
167,044	24
53,173	24
123,033	39
208,712	21
284,055	7