CellDrift: Inferring Perturbation Responses in Temporally Sampled Single Cell Data

Supplementary Note

Performance of the generalized linear model on pseudo-time data
In addition to the real-time points, we also examined the performance of CellDrift on the pseudo-time data. We simulated single-cell datasets with bifurcating trajectories using the package dynverse [1] (Supplementary Methods). Datasets with small and large rates of differential expression (DE) genes were simulated (Figure S3). Then we applied the generalized linear model (GLM) of CellDrift as well as other differential expression methods, such as the t-test and Wilcoxon test, and evaluated their performance. Similar to the evaluation for real-time points, we used True Positive Rate (TPR) and False Discovery Rate (FDR) as the evaluation metrics. To predict DEGs, we generated pseudo-time bins based on the ordering of pseudo-time values of cells (Supplementary Methods). In simulation datasets with both small and large DE rates, CellDrift and the Wilcoxon test showed similar sensitivity, which is superior to the performance of other DE methods, such as the t-test (Figure S4A, B). In datasets with a small number of bins, CellDrift performed slightly better than the Wilcoxon test on the sensitivity in a small DE rate (Figure S4A), and the opposite was true for a large DE rate (Figure S4B). Similar to the results in real-time points, CellDrift showed a larger FDR in datasets with a small DE rate, while similar FDR values were observed between CellDrift and Wilcoxon test in datasets with a large DE rate (Figure S4A, B).

Identification of temporal patterns on small numbers of cells and time points
To systematically evaluate the performance of CellDrift in particular situations, such as datasets with a small number of cells or time points, we simulated a series of datasets with variable temporal patterns using the same framework mentioned the in the main text, and then evaluated the performance of CellDrift under such circumstances (Supplementary Methods). We first simulated datasets with various numbers of cells (20, 40, 60, 80, and 100) in each cell type and perturbation combination. Data with various noise levels and sequencing depths were simulated and the Adjusted Rand Index (ARI) was used to evaluate the temporal pattern recovery performance. The results showed that the ARI had a positive correlation with the numbers of cells across different noise levels and sequencing depths (Figure S8A, B). ARI values can be as low as 0.65-0.70 for datasets with 20 cells of each cell type and perturbation combination, while the value reached around 0.8 for datasets with 100 cells. We did not observe a clear decrease in prediction performance in datasets with higher noise levels or lower sequencing depths (Figure S8A, B), indicating these two technical covariates had little effect on prediction performance.

Additionally, we evaluated the performance of CellDrift in datasets with small numbers of time points (3, 4, and 5). We rarely saw ARI values lower than 0.8 in previous benchmark experiments with a greater number of time points (greater than 5). While in this study we observed that ARI values could be as low as around 0.7 under certain noise levels for simulation datasets with only 3 time points (Figure S8C). Prediction performance reached around 0.8 in datasets with 4 and 5 time points. Similar to the benchmark results of small
numbers of cells, we didn’t observe a significant influence of noise levels on the performance (Figure S8C).

**Recommended input data and expected performance of CellDrift**

As we showed in the benchmark experiments, the input data could have a great influence on the CellDrift performance (Figure 2, S6-S8). Below is a brief summary of recommended input data so that users can better understand the performance of CellDrift under different circumstances and take full advantage of the algorithm.

The performance of CellDrift can be influenced by the number of cells and time points. Although datasets with a small number of cells can still perform reasonably well, we recommend users provide at least 40 cells of each cell type and perturbation (Figure S8A, B) to achieve greater than 75% ARI. Datasets with at least 4 time points are recommended (Figure S8C). CellDrift is not very sensitive to noises (Figure 2C, S8A). Nevertheless, it is very sensitive to the size of the differential expression (Figure 2B), so input datasets with a very small number of differentially expressed genes between control and perturbation groups should not be used. Higher sequencing depth is recommended since the pattern identification performance is positively correlated with it (Figure S6A). CellDrift is good at distinguishing temporal patterns with minor differences if temporal patterns are mostly linear (Figure S6A). For non-linear or very complex temporal patterns, we recommend users provide high-quality data with distinct temporal patterns (Figure S6A).

**Functional Principal Component Analysis (FPCA) of COVID-19 Atlas**

To identify the main components of temporal patterns of perturbation effects in COVID-19, we did FPCA for the contrast coefficients of all genes in classical monocytes. We identified 3 eigenfunctions that explained over 99% temporal variance (Figure S9-S10, Supplementary Methods). The first eigenfunction can account for the majority of observed temporal patterns, as shown by the FPC 1 scores in Figure S9. 20 temporal perturbation patterns were mainly explained by FPC 1 (Figure S10A). Clusters 11 and 17 were distributed on two ends of the FPC1 scale (Figure S10B, C), and showcased distinct functional enrichment results (Figure S10D). Moreover, genes stratified by FPC1 scores represented positive-correlated, negatively-correlated, and insensitive temporal patterns (Figure S11).

**CellDrift identified a distinct temporal perturbation pattern of CD4+ T cell subpopulations in COVID-19**

Current single-cell data usually have multiple resolutions of cell annotations. To evaluate the CellDrift performance in different resolutions, we extended the CellDrift application to CD4+ T cell subpopulations in the COVID-19 example, including CD4+ Naive T cells, CD4+ Effector T cells, CD4+ Central Memory T cells, and CD4+ proliferative Effector T cells. The performance was compared with results in the previous analysis (Figure S12), where the general temporal patterns of CD4+ T cells were identified. In most CD4+ T cell subpopulations, there were no clear up- or down-regulated temporal patterns, but there were distinct patterns in proliferative Effector T cells (Figure S13). This suggested that the most dramatic changes occurred in the
proliferation of CD4+ T cells, which is consistent with the previous CellDrift findings that cell cycle and division underlined the temporal patterns of CD4+ T cells following COVID-19 infection (Figure S12B).

**CellDrift identified immune regulatory patterns across post-HIV-infection stages**

We also applied CellDrift to the single-cell data of HIV patients during the hyperacute HIV-1 infection [2] (Table S3, Supplementary Methods). Peripheral blood mononuclear cells (PBMC) were collected at multiple time points post-infection, and 7 cell types were annotated by the authors (Figure S16A). We identified two interesting temporal clusters of perturbation effects, including clusters 5 and 6, which were down-regulated and up-regulated patterns across time, respectively (Figure S16B-D). Genes such as IL1B and IL8 were highlighted in cluster 5, while genes such as CXCL10, ISG15 and IFIT1 were highlighted in cluster 6, which is consistent with the phenotypes of inflammatory monocytes with high activities of interferon-stimulated genes in the original paper. Unlike the down-regulation of metabolism genes in the COVID-19 responses (Figure 3B), we observed the down-regulation of immune regulatory genes in HIV-infected patients. Additionally, we did gene enrichment analysis and found that both enrichment results showcased strong associations with immune responses. Cluster 5 showed stronger enrichments in interleukin responses and B cell activities, while cluster 6 showed more significant enrichments in interferon responses (Figure S16E).

**CellDrift identified asynchronous gene programs for neurons with Autism gene defects**

We extended CellDrift to the analysis of single-cell pseudo-time data and demonstrated its ability to infer gene patterns in pseudo-time scales. We first extracted progenitor cells and projection neurons (PN) from day 35 control brain organoids and organoids with haploinsufficiency of autism risk gene SUV420H1 [3] (Table S3, Supplementary Methods). Pseudo-time of neural development on the projection neuron differentiation was inferred using Monocle3 (Figure S19A, Supplementary Methods). We first defined pseudo-time bins and then applied CellDrift to the data (Figure S19A). There were two interesting clusters of temporal perturbation patterns with almost opposite trends of perturbation effects among all generated patterns (Figure S19B). Cluster 2 was enriched with synapse activities and neuron projection development, whose temporal pattern was first upregulated and then downregulated throughout the projection neuron development trajectory. In contrast, cluster 4 was enriched with stem cell activities and precursor proliferation (Figure S19C), whose temporal trend was opposite to cluster 2. It suggests that in the late stage of the projection neuron trajectory, cells with defects of the autism risk gene SUV420H1 exhibit more stem cell activities and less mature neuron properties. In the original paper, the authors found the SUV420H1-mutant neurons with accelerated maturation but reduced spontaneous circuit activities. They concluded that the SUV420H1 defect can also induce long-term abnormalities in circuit activity. The findings were consistent with CellDrift temporal patterns where higher neuron projection development activities were observed in the early stages of the trajectory while reduced synapse activities were observed in the late stages.
Supplementary Methods

Comparison of reference and contrast models
As mentioned in the main text, we create a contrast model to determine whether or not perturbation effects vary across cell types by adding the interaction term $\beta_{gcp}$. To justify the performance improvement of models for some genes by adding the interaction term, we applied the likelihood ratio test (LRT) for the reference and contrast models with the null hypothesis ($H_0$) that the contrast model doesn’t significantly perform better than the reference model (Figure S1). Using a cutoff (FDR-adjusted $p$ values < 0.05) for LRT tests, we then identify genes with cell-type-specific perturbation effects.

Contrast coefficients using Estimated Marginal Means
We retrieve pairwise contrast coefficients $\Delta \beta_{gcp}$ based on predicted $\beta_{gp}$ and $\beta_{gcp}$, which are used to measure the difference between the perturbed state and baseline in specific cell types. We first get the mean value of contrast coefficient $\Delta \beta_{gcp}$ using the sum of perturbation coefficient $\beta_{gp}$ and cell-type-specific coefficient $\beta_{gcp}$. Then we get the pooled standard errors (SE) for pairwise tests of estimates from the generalized linear model, which are computed using the estimated covariance matrix of the parameter pairs. More details can be found in this tutorial: [https://glennwilliams.me/blog/2021/09/07/estimating-marginal-means-and-pairwise-tests-by-hand-in-python/](https://glennwilliams.me/blog/2021/09/07/estimating-marginal-means-and-pairwise-tests-by-hand-in-python/). $Z$-scores are derived by dividing means by the standard errors. Using these scores, we calculate $p$ values using normal approximation, which for a two-tailed test is $2 \times$the probability of $z$-scores on the negative scale. In CellDrift, $Z$-scores are used as the final values of contrast coefficients.

Functional clustering algorithms in Benchmark
We benchmarked three popular functional clustering algorithms in our study: KMeans (KM), Fuzzy C-means (FCM) and FPCA-based EMCluster. KMeans and Fuzzy C-means were implemented in scikit-fda [4], where the distance for pairwise observations $f$ and $g$ were measured by $l_p$-distance:

$$d(f, g) = d(g, f) = ||f - g||_p$$

where $|| \cdot ||_p$ denotes the $L_p$ norm. For each observation $f$ the $L_p$ norm is defined as:

$$||f|| = \left( \int_D ||f||^p dx \right)^{\frac{1}{p}}$$

where $D$ is the domain over which the functions are defined. Default $p$ is equal to 2. For KMeans functional clustering, let $X = \{x_1, x_2, \ldots, x_n\}$ be the functional data to be analyzed, and $V = \{v_1, v_2, \ldots, v_c\}$ be the set of center of clusters in $X$ dataset, where $n$ is the number of observations and $c$ is the number of clusters. KMeans iteratively computes cluster centroids to minimize the square error function:

$$J_{KM}(X; V) = \sum_{c=1}^{C} \sum_{j=1}^{n} D_{ij}^2$$

where $D_{ij}^2$ is the squared distance measured by $p$-norm:

$$D_{ij}^2 = ||x_{ij} - v_i|| = \int_I (|x_{ij} - v_i|^p dx)^{\frac{1}{p}}$$
Different from KMean clustering, Fuzzy C-means clustering aims to minimize the weighted square errors instead of square errors:

$$J_{FCM}(X; U, V) = \sum_{i=1}^{c} \sum_{j=1}^{n} u_{ij}^f D_{ij}^2$$

where $U$ is a fuzzy partition matrix that represent membership values of observations to each cluster:

$$u_{ij}^f = \left( \sum_{k=1}^{c} \frac{(D_{ij} / D_{kj})^2}{\hat{D}_{kj}} \right)^{-1}$$

Cluster centroids $V$ are updated by the following formula:

$$v_i = \frac{\sum_{j=1}^{n} u_{ij}^f x_j}{\sum_{j=1}^{n} u_{ij}^f}, 1 \leq i \leq c$$

Additionally, FPCA-based EMCluster was implemented using function $FClust$ from R package $fdapace$. In $fdapace$, functional principal component scores are first calculated using FPCA in $fdapace$, which is discussed below. Then clustering is performed using the EMCluster algorithm. EMCluster assumes a finite mixture Gaussian distribution with unstructured dispersion and applies an EM algorithm for model-based clustering. More details of it can be found in R package $EMCluster$.

**Decomposing temporal complexity with FPCA**

In real cases, it is easy to see thousands or even more samples (genes or cell type-perturbation combinations) in the input data. To decompose the complexity of the data on the time scale, we implemented Functional Principal Component Analysis (FPCA) and extracted top functional principal components (PCs) that explained most temporal variance. To resolve the sparsity issue of functional data due to the lack of time points, we used Principal Analysis by Conditional Estimation (PACE), which yields covariance and mean functions, eigenfunctions and principal component scores for both functional data and its derivatives with respect to time [5,6]. Notably, the covariance of curves in PACE is smoothed, resulting in smoothed temporal curves in the output. In conclusion, PACE can not only reduce temporal complexity but also generate smooth curves.

Functional principal component analysis was implemented using the R package $fdapace$, where PACE was performed to retrieve functional PCs [7]. Here is a brief introduction of PACE procedure in the package (revised based on tutorial of $fdapace$):

1. Smoothed mean $\hat{\mu}$ is calculated using local linear smoothing and all the available readings are aggregated together.
2. Raw covariance for each curve is calculated separately and then all raw covariances are aggregated to generate the sample raw covariance.
3. Smoothed covariance is estimated using the off-diagonal elements of the sample raw covariance.
4. Eigenanalysis is performed on the smoothed covariance to obtain the estimated eigenfunctions $\hat{\varphi}$ and eigenvalues $\hat{\lambda}$. The smoothed covariance is then projected on a positive semi-definite surface.
5. Conditional Expectation (PACE step) is used to estimate the corresponding scores $\hat{\xi}$:
\[ \hat{\xi}_{ik} = \hat{E}[\hat{\xi}_{ik} | Y_i] = \hat{\lambda}_k \hat{\Phi}_{ik} \sum_{Y_i}^{-1} (Y_i - \hat{\mu}_i) \]

**ANOVA Test for Differential Temporal Perturbation Effects**

We implemented the one-way functional ANOVA test to identify genes with differential patterns across perturbation groups [8]. Let \( P_1, P_2, \ldots, P_k \), be different perturbation groups and \( F_{CP_i} \) be functional data for the cell type \( C \) in perturbation group \( P \). We define \( E(F_{CP_1}) = m_1(t), E(F_{CP_2}) = m_2(t), \ldots \), where \( C \) represents the cell type. The null hypothesis is:

\[ H_0: m_1(t) = m_2(t) = \ldots = m_k(t) \]

In practice, one-way functional ANOVA was implemented using the `oneway_anova` function in the package scikit-fda. It is used to analyze whether there is a difference among means of the functional data.

Theoretically, let \( \{X_i\}_{i=1}^k \) be \( k \) independent samples and each one with \( n_i \) trajectories; let \( E(X_i) = m_i(t) \). The null hypothesis is:

\[ H_0: m_1(t) = m_2(t) = \ldots = m_k(t) \]

`oneway_anova` calculates a statistic that measures the variability between groups of samples:

\[ V_n = \sum_{i<j}^k w_i ||f_i - f_j||^2 \]

where \( \{f_i\}_{i=1}^k \) is a set of samples in the functional data; \( \{w_j\}_{j=1}^k \) is a set of weights, where \( w_i \) is related to the sample \( f_i \) for \( i = 1, \ldots, k \).

Under the null hypothesis this statistic \( V_n \) is asymptotically equivalent to `v_asymptotic_stat()`, where each sample is replaced by a gaussian process with zero mean and the same covariance as the original curve. More details of the procedure can be found in the original paper [8].

**Time Warping**

In some situations, the sampling time points can be variable for different perturbations. For example, single cell data of COVID-19 and sepsis patients were retrieved from different time scales in the same study [9]. To address this issue, we used the dynamic time warping (DTW) algorithm to align different time series. It aligns two time series with different lengths by comparing the similarity or calculating the distance between them [10]. It is necessary when samples in different perturbation groups are collected from unaligned time points. Application of DTW could make temporal curves comparable. We first choose a reference curve and then find the matching indices between query and reference curves. Values of time points on query curves are projected onto the matched time points in the reference curve, making temporal data from different curves comparable.

**Perturbation Data Simulation**

Simulation data with varying batch effects and differential expression sizes were generated from `Splatter` [11]. To mimic real single-cell data, we extracted initialization parameters from a CD14+ Monocytes of an interferon-stimulated PBMC dataset [12]. We define 3000 features and 1000 cells with 2 cell groups (Group 1, 2) and 3 batches (Batch 1, 2, 3). To investigate the influence of
batch effects, we selected varying batch effect sizes \((\text{batch}.\text{facLoc})\) from \([0.02, 0.1, 0.4, 0.7]\) for 3 batches. Uniform Manifold Approximation and Projection (UMAP) plots of the generated data with batch effect values of 0.02 and 0.1 show clear separation of the two conditions whereas a pronounced effect of batch appears for values of 0.4 and 0.7 (Figure S2). \(\text{batch}.\text{facScale}\) was set to 0.1. In this simulation experiment, the proportion of differential expression genes compared with baseline \((\text{de}.\text{prob})\) is controlled as 50\% for all three batches with shared DE genes, among which 50\% are downregulated genes. To simulate prominent batch effects for benchmark tasks, we intentionally introduced imbalance in the largest batch, with 600 cells in Batch 1 and 200 cells in Batch 2 and 3 (Figure 2C, D).

Additionally, to interrogate the influence of differential expression size, the differential expression parameter \(\text{de}.\text{prob}\) was defined with a series of values ranging in \([0.05, 0.2, 0.5, 0.8]\), representing the proportion of DE genes when compared to the baseline expression in the simulated data. To clarify, this parameter doesn’t denote the DE gene proportion between Group1 and Group2. In this benchmark experiment, simulated data contains 600 cells and only one batch.

To define the ground truth of DE genes between Group 1 and Group 2, we divided genes from simulated data into two categories: (i) unperturbed (negative) genes with differential expression factors \(\text{DEFac}\) in both Group1 and Group2 as 1; (ii) perturbed (positive) genes with absolute difference of \(\text{DEFac}\) between Group 1 and Group 2 in the top 75 percentile.

**Temporal Data Simulation**

Inspired by MEFISTO [13], simulation data were derived from a generative model by varying the number of time points per group in a \([0, 1]\) interval, the rate of missing time points, the noise levels and sequencing depths (Figure 3). The default base mean of simulated genes is 2. Default coefficient parameters for cell type, perturbation and the interaction effects are 0, 0.3 and 2, and corresponding scale parameters are 0.2, 0.2, 0.2, respectively. We defined 3 linear temporal patterns in our study and cell type-perturbation interaction coefficient is a time-dependent parameter \(h(t) = r_t t + r_0\). By default, \(r_t\) are -1, 0, 1 for negative-correlated, time-insensitive and positive-correlated temporal patterns. Corresponding intercepts \(r_0\) are 1, 1, 0.

We also defined 3 non-linear temporal patterns where \(h(t) = r_t(t − 0.5)^2 + r_0\) (Figure 3E), and default \(r_t\) are -4, 0, 4, representing various shapes of non-linear temporal patterns. We also defined \(r_t\) as \([-8, 0, 8]\) and \([-2, 0, 2]\) to represent different temporal curve shapes in the benchmark. In each parameter setting, 100 cells in each cell type and perturbation group per time point, and 20 genes were simulated for each temporal pattern. 12 replicates were generated for each parameter setting.

**Pseudo-time and Trajectory Data Simulation**

Apart from the aforementioned simulation of real temporal data, we also generated pseudo-time single cell data with trajectory differentiation information using the package dynverse [1]. We generated single-cell datasets with 500 cells and 2000 genes with bifurcating trajectories using the function generate_dataset of dyntoy. We simulated datasets with small (0.1) and large (0.5) ratios of differentially expressed genes of two trajectories (Figure S3).

**Generation of pseudo-time bins**
Pseudo-time trajectories were first generated by trajectory inference packages, such as Monocle3, PAGA [14] and [15]. In this manuscript, we demonstrated the performance of CellDrift with inferred trajectories and pseudo-time data from Monocle3 (Figure S19). In the CellDrift package, users need to provide both trajectory and pseudo-time information for cells of interest as the input covariates. In each lineage, maximal and minimal pseudo-time values were selected and the user-customized number of bins was generated evenly using the numpy \texttt{linspace} function. The median pseudo-time values were defined as the time points of bins and used as the input of functional data analysis in CellDrift.

**Benchmark Criteria**

In this study, we benchmarked two parts of the algorithm, including perturbed gene prediction using the generalized linear model and temporal pattern identification using functional data analysis.

As we mentioned previously, true perturbed genes and unperturbed genes were defined in the Splatter simulation data as positive and negative data. We defined predicted significantly perturbed genes in benchmarked methods as those with FDR-adjusted p values less than 0.05. We used the true positive rate (TPR or sensitivity) and the false discovery rate (FDR) as metrics to evaluate the sensitivity and false discoveries of GLM in identifying perturbed genes. These metrics have been widely used in established methods [16].

In the simulated pseudo-time data, true perturbed genes and unperturbed genes were defined by the package dyntoy from dynverse (Figure S3C, F). We defined predicted significantly perturbed genes as those with FDR-adjusted \( p \) values in at least one of the pseudo-time bins. Same metrics were used to evaluate the sensitivity and false discoveries.

In the benchmark experiment of temporal pattern identification, three temporal patterns were defined in the simulation data. Then we used the adjusted rand index (ARI) to measure the similarity between estimated functional clusters \( C_E \) with true temporal patterns \( C_T \):

\[
ARI (C_E, C_T) = \frac{\sum_{e,t} (n_{et}^2) - \left[ \sum_e (n_e^2) \sum_t (n_t^2) \right]}{\frac{1}{2} \left[ \sum_e (n_e^2) + \sum_t (n_t^2) \right] - \left[ \sum_e (n_e^2) \sum_t (n_t^2) \right] / (n^2)}
\]

where \( n \) is the total number of genes; \( n_e \) and \( n_t \) are the number of genes in each estimated cluster \( e \) and in the true cluster \( t \), respectively; and \( n_{et} \) is the number of genes shared by estimated cluster \( e \) and true cluster \( t \). ARI ranges from 0 to 1, where a larger value indicates more similarity between estimated clusters and true clusters.

Additionally, we utilized FPCA to retrieve smoothed curves of functional data, where we have imputed contrast coefficients for missing time points. To compare imputation performance with real simulated contrast coefficients, we used the Pearson correlation to measure the consistency between simulated and estimated contrast coefficients for missing time points in linear temporal patterns.

**Gene Set Enrichment Analysis**

We used ToppGene to do the gene set enrichment analysis (GSEA) and annotate CellDrift temporal clusters of genes. Gene ontologies, particularly biological processes, were used to
annotate important biological pathways of clusters of genes. FDR-adjusted $p$ values were used as the significance of enrichment results and $-\log_{10}[FDR \text{ adjusted } p \text{ values}]$ were used as enrichment scores.

**Gene Set Association Test with Hypergeometric Test**

We used reference gene sets from literature or public datasets to validate our generated gene lists from CellDrift. Such reference gene sets are usually compiled by experts with high credibility. We utilized the hypergeometric test which has been widely used in gene set enrichment analysis to evaluate the significance of associations between reference gene sets and CellDrift temporal patterns [17]. We first defined a reference gene set and a query gene list in a large pool of genes, where the query gene list is a CellDrift temporal pattern. We used the `stats.hypergeom` function from the `scipy` package and the derived $p$ values represent the intersection of these lists occurred by chance. We adjusted $p$ values using the FDR-correction procedure and results with adjusted $p$-values less than 0.05 were regarded as significant associations.

**Differential Expression Methods in Benchmark**

We benchmarked four differential expression methods in the study, including t-test, t-test (overdispersion), wilcoxon test and MAST, which are popular DE methods in single cell analysis. T-test, t-test (overdispersion) and wilcoxon test were implemented in Scanpy [18] using function `rank_genes_groups`. Compared with t-test, t-test (overdispersion) forces the variance of cells in both comparison groups to be the same to mitigate the statistical issues caused by imbalanced cell numbers. It was the default DE method in previous Scanpy versions. MAST is a generalized linear model that adjusts the cellular detection rate and the fraction of genes expressed in a cell in the differential expression analysis [19]. Compared with other methods, it has more flexibility to address the nuisance variation in single cell count data using the GLM framework.

**Dataset Description**

(1) **COVID-19 Atlas (COMBAT) [9]**: This is a large cohort of single-cell CITE-seq study of PBMC cells from control donors, COVID-19, influenza and sepsis patients. 116 hospitalized COVID-19 patients were recruited from a single site, including mild, mild (healthcare workers), severe and critical COVID-19 patients. Disease progression days (0–25 days since onset) were recorded for participants in the study. We extracted the six most common cell types in the study for the CellDrift analysis. More details can be found in Table S3.

(2) **COVID-19 Atlas (Ren et al.) [20]**: This is a large COVID-19 cohort of single-cell data with 284 samples from 196 COVID-19 patients. 1.46 million cells were collected from multiple tissues, including bronchoalveolar lavage fluid (BALF), sputum, peripheral blood mononuclear cells (PBMC) and pleural fluid mononuclear cells (PFMCs). Control donors, mild/moderate and severe/critical COVID-19 patients were recruited. The data was downloaded from GEO accession GSE158055 and used as validation data in our study. We extracted PBMC cell data from control donors and patients in progression (patients during convalescence were not
included in the analysis) for downstream analysis. B cells, CD4+ T cells, CD8+ T cells and Monocytes were included in CellDrift analysis.

(3) Fetal Gut Atlas [21]: This dataset contains single cell transcriptomics from human embryos with a post-conceptional age ranging from 6 to 10 weeks. Dissected tissues were divided into proximal small bowel (duodenum and jejunum, or duo-jejunum), distal small bowel (ileum), and large bowel (colon). Four lineages of cells, including epithelium, immune, mesenchymal and vasculature, were included in the data. We used data from epithelium and mesenchymal cells for downstream analysis and generated general temporal patterns for these two lineages of cells. The data was downloaded from Gut Cell Atlas: https://www.gutcellatlas.org.

(4) Hyperacute HIV1 infection [2]: This scRNA-seq data covers peripheral blood mononuclear cells from untreated individuals with HIV infections before and after acute infection. Longitudinal samples were collected from these patients from week 0 to year 1 post-infection. Multiple cell types involved in immune responses were annotated and analyzed. The data was downloaded from single cell portal SCP256.

(5) Brain organoid data of autism risk genes [3]: This is a large-scale single cell RNA-seq data for the investigation of haploinsufficiency of three autism risk genes in brain organoids, including SUV420H1, ARID1B and CHD8. It covers 745,000 brain organoid cells from early time points (such as 28 days) and late time points (such as 6 months). Data can be accessed from single cell portal SCP1129.

Reference

1. Saelens W, Cannoodt R, Todorov H, et al. A comparison of single-cell trajectory inference methods. Nat. Biotechnol. 2019; 37:547–554
2. Kazer SW, Aicher TP, Muema DM, et al. Integrated single-cell analysis of multicellular immune dynamics during hyperacute HIV-1 infection. Nat. Med. 2020; 26:511–518
3. Paulsen B, Velasco S, Kedaigle AJ, et al. Autism genes converge on asynchronous development of shared neuron classes. Nature 2022; 602:268–273
4. Ramos-Carreño C, Torrecilla JL, Suárez A. scikit-fda: A Python package for Functional Data Analysis. Different Varimax Rotation Approaches of Functional PCA for the evolution of COVID-19 pandemic in Spain. . . . . . . . . . 2019; 55
5. Chen K, Zhang X, Petersen A, et al. Quantifying Infinite-Dimensional Data: Functional Data Analysis in Action. Stat. Biosci. 2017; 9:582–604
6. Yao F, Müller H-G, Wang J-L. Functional Data Analysis for Sparse Longitudinal Data. J. Am. Stat. Assoc. 2005; 100:577–590
7. Dai X, Hadjipantelis PZ, Ji H, et al. fdapace: Functional data analysis and empirical dynamics. R package version 0. 3. 0 2017;
8. Cuevas A, Febrero M, Fraiman R. An anova test for functional data. Comput. Stat. Data Anal. 2004; 47:111–122
9. COvid-19 Multi-omics Blood ATlases (COMBAT) Consortium, Ahern DJ, Ai Z, et al. A blood atlas of COVID-19 defines hallmarks of disease severity and specificity. bioRxiv 2021;
10. . Dynamic Time Warping. Information Retrieval for Music and Motion 2007; 69–84
11. Zappia L, Phipson B, Oshlack A. Splatter: simulation of single-cell RNA sequencing data. Genome Biol. 2017; 18:174
12. Kang HM, Subramaniam M, Targ S, et al. Multiplexed droplet single-cell RNA-sequencing using natural genetic variation. Nat. Biotechnol. 2018; 36:89–94
13. Velten B, Braunger JM, Argelaguet R, et al. Identifying temporal and spatial patterns of variation from multimodal data using MEFISTO. Nat. Methods 2022; 19:179–186
14. Wolf FA, Hamey FK, Plass M, et al. PAGA: graph abstraction reconciles clustering with trajectory inference through a topology preserving map of single cells. Genome Biol. 2019; 20:59
15. Herman JS, Sagar, Grün D. FateID infers cell fate bias in multipotent progenitors from single-cell RNA-seq data. Nat. Methods 2018; 15:379–386
16. Dann E, Henderson NC, Teichmann SA, et al. Differential abundance testing on single-cell data using k-nearest neighbor graphs. Nat. Biotechnol. 2021;
17. Chen J, Bardes EE, Aronow BJ, et al. ToppGene Suite for gene list enrichment analysis and candidate gene prioritization. Nucleic Acids Res. 2009; 37:W305–11
18. Wolf FA, Angerer P, Theis FJ. SCANPY: large-scale single-cell gene expression data analysis. Genome Biol. 2018; 19:15
19. Finak G, McDavid A, Yajima M, et al. MAST: a flexible statistical framework for assessing transcriptional changes and characterizing heterogeneity in single-cell RNA sequencing data. Genome Biol. 2015; 16:278
20. Ren X, Wen W, Fan X, et al. Large-scale single-cell analysis reveals critical immune characteristics of COVID-19 patients. bioRxiv 2020;
21. Elmentaite R, Ross A, James KR, et al. Single-cell sequencing of developing human gut reveals transcriptional links to childhood Crohn’s disease.