Efficient and precise single-cell reference atlas mapping with Symphony

Joyce B. Kang[1-5], Aparna Nathan[1-5], Nghia Millard[1-5], Laurie Rumker[1-5], D. Branch Moody[3], Ilya Korsunsky[1-5]**, Soumya Raychaudhuri[1-6]**

[1] Center for Data Sciences, Brigham and Women's Hospital, Boston, MA, USA

[2] Division of Genetics, Department of Medicine, Brigham and Women's Hospital and Harvard Medical School, Boston, MA, USA

[3] Division of Rheumatology, Inflammation, and Immunity, Department of Medicine, Brigham and Women's Hospital and Harvard Medical School, Boston, MA, USA

[4] Department of Biomedical Informatics, Harvard Medical School, Boston, MA, USA

[5] Program in Medical and Population Genetics, Broad Institute of MIT and Harvard, Cambridge, MA, USA

[6] Versus Arthritis Centre for Genetics and Genomics, Centre for Musculoskeletal Research, Manchester Academic Health Science Centre, The University of Manchester, Manchester, UK

** These authors jointly supervised this work.

Correspondence to:
Ilya Korsunsky
Harvard New Research Building
77 Avenue Louis Pasteur
Boston, MA 02115
ikorsunskiy@bwh.harvard.edu

Soumya Raychaudhuri
Harvard New Research Building
77 Avenue Louis Pasteur, Suite 250
Boston, MA 02115
soumya@broadinstitute.org
Ph: 617-525-4484 Fax: 617-525-4488

Abstract

Recent advances in single-cell technologies and integration algorithms make it possible to construct large, comprehensive reference atlases from multiple datasets encompassing many donors, studies, disease states, and sequencing platforms. Much like mapping sequencing reads to a reference genome, it is essential to be able to map new query cells onto complex, multimillion-cell reference atlases to rapidly identify relevant cell states and phenotypes. We present Symphony, a novel algorithm for building compressed, integrated reference atlases of $\geq 10^6$ cells and enabling efficient query mapping within seconds. Based on a linear mixture model framework, Symphony precisely localizes query cells within a low-dimensional reference embedding without the need to reintegrate the reference cells, facilitating the downstream transfer of many types of reference-defined annotations to the query cells. We demonstrate the power of Symphony by (1) mapping a query containing multiple levels of experimental design to predict pancreatic cell types in human and mouse, (2) localizing query cells along a smooth developmental trajectory of human fetal liver hematopoiesis, and (3) harnessing a multimodal CITE-seq reference atlas to infer query surface protein expression in memory T cells. Symphony will enable the sharing of comprehensive integrated reference atlases in a convenient, portable format that powers fast, reproducible querying and downstream analyses.

Introduction

Advancements in single-cell RNA-sequencing (scRNA-seq) have launched an era in which individual studies can routinely profile 10^4-10^6 cells[1–3], and multimillion-cell datasets are already emerging[4,5]. Single-cell resolution enables the discovery and refinement of cell states across diverse clinical and biological contexts[6–11]. To date, most studies redefine cell states from scratch, making it difficult to compare results across studies and thus hampering reproducibility. Coordinated large-scale efforts, exemplified by the Human Cell Atlas (HCA)[12], aim to establish comprehensive and well-annotated reference datasets comprising millions of cells that capture the broad spectrum of cell states. Building these reference datasets requires integrating multiple datasets which may have been collected under different technical and biological conditions. Hence, reference construction requires application of one of many recently developed single-cell integration algorithms[13–19]. Our group previously developed Harmony[15], a fast, accurate, and well-reviewed method[20] that is able to explicitly model complex study design, a property that makes it suitable for integrating complex datasets into reference atlases[21–24]. Once such atlases are constructed, powerful mapping algorithms will make it possible to rapidly and reproducibly map new single-cell datasets onto the reference and automatically annotate them by transferring information from nearby reference cells.

Fast mapping of query cells against a large, stable reference is a well-recognized open problem[25] and active area of research[18,26,27]. One inefficient but accurate approach to project reference and query cells into a joint embedding is to integrate both sets of cells together *de novo*, resulting in what might be considered a “gold standard” embedding. While this is a reasonable approach for relatively small reference datasets, the strategy is intractable for atlas-sized references with millions of cells. It requires users to “rebuild” the reference for each analysis, and requires potentially cumbersome and administratively challenging exchanges of large-scale datasets. Furthermore, it may corrupt the reference embedding once a reference is carefully constructed and annotated. It is instead preferable to freeze the reference when mapping new query cells onto it.

High-quality reference mapping requires both a fast and accurate mapping algorithm and a framework to efficiently store a reference dataset. An ideal reference mapping algorithm must meet four key requirements: handle complex study design in both the reference and query, scale to large datasets, map with high accuracy, and enable inference of diverse query cell annotations based on reference cells. Here, we present Symphony, a novel algorithm to compress a large, integrated reference and map query cells to a precise location in the reference embedding within seconds. Through multiple real-world dataset analyses, we show that Symphony can enable accurate downstream inference of cell type, developmental trajectory position, and protein expression, even when the query itself contains complex confounding technical and biological effects.

Results

Symphony compresses an integrated reference for efficient query mapping

Symphony comprises two main algorithms: reference compression and mapping (**Methods, Fig. S1a**). Symphony *reference compression* captures and structures information from multiple reference datasets into an integrated and concise format that can subsequently be used to map query cells (**Fig. 1a-b**). Symphony builds upon the same linear mixture model framework as Harmony[17]. Briefly, in a low-dimensional embedding, such as principal component analysis (PCA), the model represents cell states as soft clusters, in which a cell's identity is defined by probabilistic assignments across one or more clusters. For *de novo* integration of the reference, cells are iteratively assigned soft cluster memberships, which are used as weights in a linear mixture model to remove unwanted covariate-dependent effects. To store the reference efficiently without saving information on individual reference cells, Symphony computes summary statistics learned in the low-dimensional space (**Fig. 1b, Methods**), returning computationally efficient data structures containing the "minimal reference elements" needed to map new cells. These include the means and standard deviations used to scale the genes, the gene loadings from PCA (or another low dimensional projection, e.g. canonical correlation analysis [CCA]) on the reference cells, soft-cluster centroids from the integrated reference,

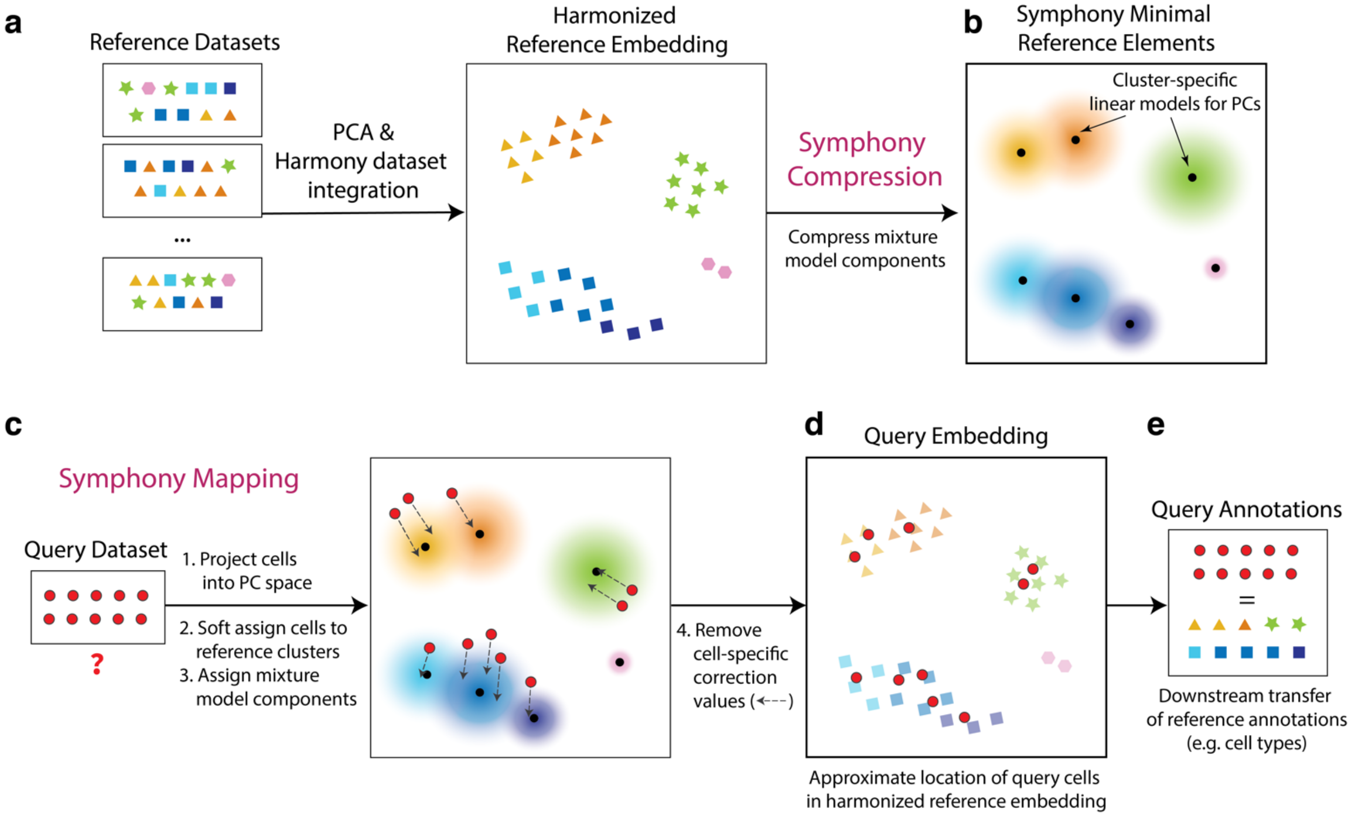

Figure 1 | Overview of Symphony framework and algorithm. Symphony comprises two algorithms: Symphony compression **(a-b)** and Symphony mapping **(c-d)**. **(a)** To construct a reference, cells from multiple datasets are embedded in a lower-dimensional space (e.g. PCA), in which dataset integration (Harmony) is performed to remove dataset-specific effects. Shape indicates distinct cell types, and color indicates finer-grained cell subtypes or states. **(b)** Symphony compression represents the information captured within the harmonized reference in a concise, portable format based on computing summary statistics for the reference-dependent components of the linear mixture model. Symphony returns the minimal reference elements needed to efficiently map new query cells to the reference. **(c)** Given an unseen query dataset and compressed reference, Symphony mapping localizes the query cells to their appropriate locations within the integrated reference embedding **(d)**. The reference cell locations do not change during mapping. **(e)** The shared Symphony feature embedding can be used for downstream transfer of reference-defined annotations to the query cells.

and two “compression terms” (a *k* x 1 vector and *k* x *d* matrix, where *k* is the number of clusters and *d* is the dimensionality of the embedding) (**Methods, Supplementary Equations, Fig. S1b**).

To map new query cells to the compressed reference, we apply Symphony *mapping*. The algorithm approximates integration of reference and query cells *de novo* (**Methods**), but uses only the minimal reference elements to compute the mapping (**Fig. S1c**). First, Symphony projects query gene expression profiles into the same uncorrected low-dimensional space as the reference cells (e.g. PCs), using the saved scaling parameters and reference gene loadings (**Fig. 1c**). Second, Symphony computes soft cluster assignments for the query cells based on proximity to the reference cluster centroids. Finally, to correct unwanted user-specified technical and biological effects in the query data, Symphony assumes the soft cluster assignments from the previous step and uses stored mixture model components to regress out the query batch effects (**Fig. 1d**). Importantly, the reference cell embedding remains stable during mapping. Embedding the query within the reference coordinates enables downstream transfer of annotations from reference cells to query cells, including discrete cell type classifications, quantitative cell states (e.g. position along a trajectory), or expression of missing genes or proteins (**Fig. 1e**).

Symphony approximates *de novo* integration without reintegration of the reference cells

As we demonstrate in the **Methods**, Symphony is equivalent to running *de novo* Harmony integration if three conditions are met: (I) all cell states represented in the query data set are captured by the reference dataset, (II) the number of query cells is much smaller than the number of reference cells, and (III) the query dataset has a design matrix that is independent of reference datasets (i.e. non-overlapping batches in reference and query). As the scope of available single-cell atlases continues to grow, it is reasonable to assume that reference datasets are large and all-inclusive, making conditions (I) and (II) well-supported. Condition (III) is also typically met if the query data was generated in separate experiments from the reference.

To demonstrate that Symphony mapping closely approximates running *de novo* integration on all cells, we applied Symphony to 20,792 peripheral blood mononuclear cells (PBMCs) assayed with three different 10x technologies: 3'v1, 3'v2, and 5'. We performed three mapping experiments. For each, we built an integrated Symphony reference from two technologies, then mapped the third technology as a query. The resulting Symphony embeddings were compared to a gold standard embedding obtained by running Harmony on all three datasets together. Visually, we found that the Symphony embedding for each mapping experiment (**Fig. 2a**) closely reproduced the overall structure and cell type information of the gold standard embedding (**Fig. 2b**). To quantitatively assess the degrees of dataset mixing we use the Local Inverse Simpson's Index (LISI)[26–30] metric; higher LISI scores correspond to better mixing of cells across batches. LISI scores in Symphony embeddings (mean LISI 2.16, 95% CI [2.16, 2.17]) and *de novo* integration embeddings (mean LISI 2.14, 95% CI [2.13, 2.15]) were nearly identical (**Fig. 2c**, **Methods**).

To directly assess similarity of the local neighborhood structures, we computed the correlation between the local neighborhood adjacency graphs generated by Symphony and *de novo* integration. We define a new metric called k-nearest-neighbor correlation (k-NN-corr), which quantifies how well the local neighborhood structure in a given embedding is preserved in an alternative embedding by looking at the correlation of neighbor cells sorted by distance (**Fig. S2a-e**). Anchoring on each query cell, we calculate (1) the pairwise similarities to its *k* nearest reference neighbors in the gold standard embedding and (2) the similarities between the same query-reference neighbor pairs in the alternate embedding (**Methods**), then calculate the Spearman correlation between (1) and (2). k-NN-corr ranges from -1 to +1, where +1 indicates a perfectly preserved sorted ordering of neighbors. We find that for *k*=500, the Symphony embeddings produce a k-NN-corr >0.4 for 77.3% of cells (and positive k-NN-corr for 99.9% of cells), demonstrating that Symphony not only maps query cells to the correct broad cluster but also preserves the distance relationships between nearby cells in the same local region (**Fig. 2d)**. As a comparison, we calculated k-NN-corr for a simple PC projection of the query cells (with no correction step) using the original reference gene loadings prior to integration and observed significantly lower correlations (Wilcoxon signed-rank p<2.2e-16), with k-NN-corr >0.4 for 39.9% of cells (**Fig. S2f**).

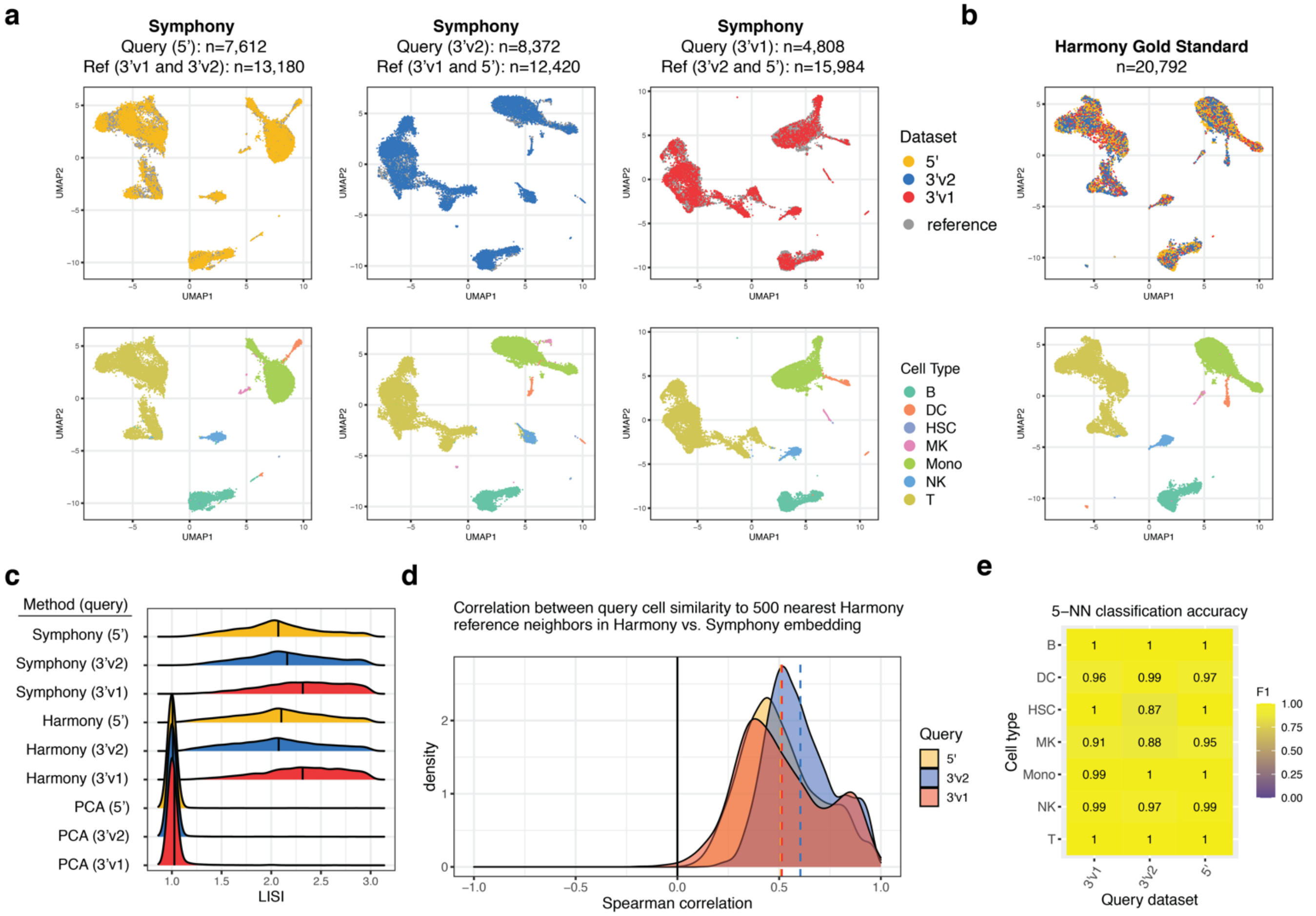

Figure 2 | Symphony approximates *de novo* integration without reintegration of the reference. Three PBMC datasets were sequenced with different 10x protocols: 5' (yellow, n=7,697 cells), 3'v2 (blue, n=8,380 cells), and 3'v1 (red, n=4,809 cells). We ran Symphony three times, each time mapping one dataset onto the other two. **(a)** Symphony embeddings generated across the three mapping experiments (columns). Top row: cells colored by query (yellow, blue, or red) or reference (gray), with query cells plotted in front. Bottom row: cells colored by cell type: B cell (B), dendritic cell (DC), hematopoietic stem cell (HSC), megakaryocyte (MK), monocyte (Mono), natural killer cell (NK), or T cell (T), with query cells plotted in front. **(b)** For comparison, gold standard *de novo* Harmony embedding colored by dataset (top) and cell type (bottom). **(c)** Distribution of LISI scores for query cells in the Symphony embeddings, gold standard, and a standard PCA pipeline on all cells. **(d)** Distribution of k-NN-corr (Spearman correlation between the similarities between the neighbor-pairs in the Harmony embedding and the similarities between the same neighbor-pairs in the Symphony embedding) across query cells for k=500, colored by query dataset. **(e)** Classification accuracy as measured by cell type F1 scores for downstream query cell type annotation using 5-NN on the Symphony embedding.

Symphony enables accurate cell type classification

If Symphony is effective, then cells should be mapped close to cells of the same cell type, enabling accurate cell type classification. To test this, we performed post-mapping query cell type classification in the 10x PBMCs example from above. We used a 5-NN classifier to annotate query cells across 7 cell types based on the nearest reference cells in the harmonized embedding and compared the predictions to the ground truth labels assigned *a priori* with lineage-specific marker genes (**Methods, Table S2**). Across all three experiments, predictions using the Symphony embeddings achieved 99.5% accuracy overall, with a median cell type F1-score (harmonic mean of precision and recall, ranging from 0 to 1) of 0.99 (**Fig. 2e, Table S3**). This indicates that Symphony appropriately localizes query cells in harmonized space to enable the accurate transfer of cell type labels.

Automatic cell type classification represents an open area of research[28–32]. Existing supervised classifiers assign a limited set of labels to new cells based on training data and/or marker genes. To benchmark Symphony-powered downstream inference against existing classifiers, we followed the same procedure as a benchmarking analysis in Abdelaal et al. (2019)[28]. The benchmark compared 22 cell type classifiers on the PbmcBench dataset consisting of two PBMC samples sequenced using 7 different protocols[33]. For each protocol train-test pair (42 experiments) and donor train-test pair (additional 6 experiments) (**Methods**), we built a Symphony reference from the training dataset then mapped the test dataset. We used the resulting harmonized feature embedding to predict query cell types using three downstream models: 5-NN, SVM with radial kernel, and multinomial logistic regression. The Symphony-based classifiers achieve consistently high cell type F1-scores (average median F1 of 0.79-0.83) comparable to the top three supervised classifiers for this benchmark (scmapcell, singleCellNet, and SCINA, average median F1 of 0.77-0.83) (**Fig. S3a**). Notably, as discussed in Abdelaal et al., the median F1-score alone can be misleading given that some classifiers (including SCINA) leave low-confidence cells as “unclassified”, whereas we used Symphony to assign a label to every cell. This benchmark is also arguably suboptimal in that the reference in each experiment is comprised of a single dataset (no reference integration involved).

Symphony maps against a large reference within seconds

To demonstrate scalability to large reference atlases, we evaluated Symphony's computational speed. We downsampled a large memory T cell dataset[34] to create benchmark reference datasets with 20,000, 50,000, 100,000, 250,000, and 500,000 cells (from 12, 30, 58, 156, and 259 donors, respectively). Against each reference, we mapped three different-sized queries: 1,000, 10,000, and 100,000 cells (from 1, 6, and 64 donors) and measured total elapsed runtime (**Fig. S4, Table S4**). The speed of the reference building process is comparable to that of running *de novo* integration since they both start with expression data and require a full pipeline of scaling, PCA, and Harmony integration. However, a reference need only be built and saved once in order to map all subsequent query datasets onto it. For instance, initially building a 500,000-cell reference with Symphony took 5,163 seconds (86.1 min) and mapping a subsequent 10,000-cell query onto it took only 0.99 secs, compared to 4,806 secs (80.1 mins) for *de novo* integration on all cells. Symphony offers a 5000x speedup in this application. These results show that Symphony scales efficiently to map against multimillion-cell references, enabling it to power potential web-based queries within seconds.

Importantly, Symphony mapping time does not depend on the number of cells or batches in the reference since the reference cells are modeled post-batch correction (**Methods**); however, it does depend on the reference complexity (number of centroids *k* and dimensions *d*) and number of query cells and batches (**Table S4**) since the query mapping algorithm solves for the query batch coefficients for each of the reference-defined clusters.

Mapping a query dataset with multi-level experimental design in human and mouse pancreas

Symphony is designed to handle query datasets with multiple batches, technologies, individuals, or other structures. To demonstrate, we used Symphony in a scenario in which both the reference and the query have complex experimental designs (**Fig 3a**). The reference contained 6,177 pancreatic islet cells from 32 human donors across four independent studies[35–38], each employing a different plate-based scRNA-seq technology (CEL-seq, CEL-seq2, Smart-seq2, and Fluidigm C1; **Fig 3b**). We

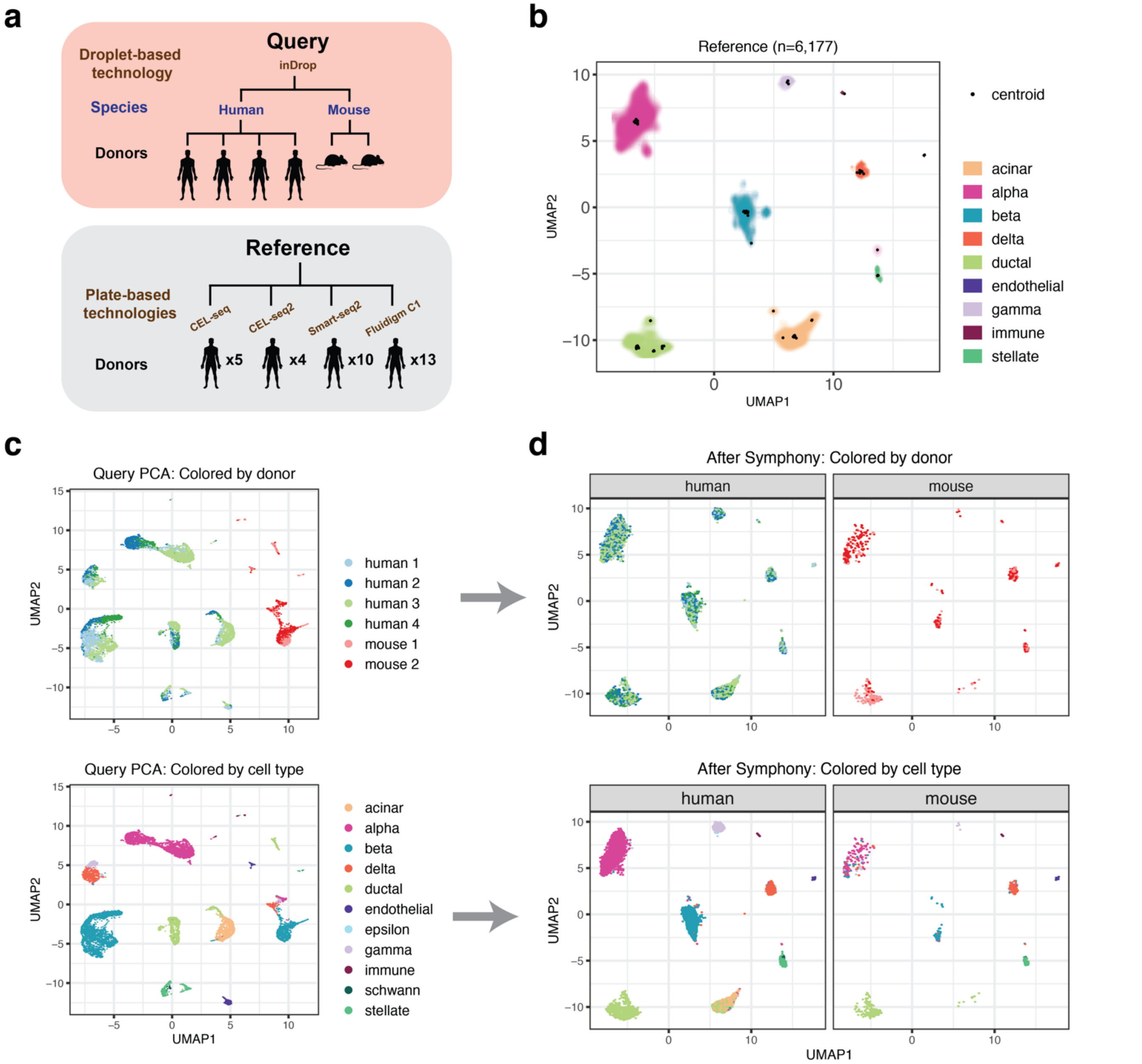

Figure 3 | Symphony maps pancreas cells with multi-level query. (a) Symphony mapping can model and remove multiple potentially nested sources of batch variation in the query, such as technology, species, and donor. In this example, the query dataset (n=10,455 cells, from 4 human donors and 2 mouse donors) was sequenced on a new technology (inDrop) previously unseen in the reference. **(b)** A Symphony reference (n=6,177 cells, 32 donors) was built from four human pancreas datasets, each assayed using different technologies, and broad cell types were annotated using canonical marker genes. Contour fill represents density of reference cells. Black points represent soft-cluster centroids in the Symphony mixture model. **(c)** Without mapping, a standard PCA pipeline shows that query cells exhibit strong species and donor effects. **(d)** Query cells are mapped against the reference by simultaneously removing the effect of technology, species, and donor in the query such that the cells group by cell type with mixing between species and among donors. Top row colored by donor; bottom row colored by cell type as previously defined by Baron et al. (2016).

integrated across donors and technologies, defined clusters, and manually annotated cell types using cluster-specific marker genes (**Methods, Table S5**). The query contained 8,569 pancreatic islet cells from 4 human donors and 1,866 cells from 2 mice, all profiled with inDrop, a droplet-based scRNA-seq technology absent in the reference[39].

PCA of the query dataset alone revealed large sources of variation from both species and donor identity (**Fig. 3c**). Symphony mapped the multi-level droplet-based query onto the plate-based reference by simultaneously modeling and removing the effects of technology, species, and donor within the query (**Fig. 3d**). By removing all three nested sources of variation, we accurately predicted query cell types with a 5-NN classifier in the harmonized embedding: median cell type F1-scores of 0.97 (overall accuracy 96.4%) for human and median cell type F1 of 0.94 (overall accuracy 86.4%) for mouse cells, with ground truth labels defined by the original publication[39] (**Table S6**). By mapping against a reference, Symphony is able to overcome strong species effects and simultaneously map analogous cell types between mouse and human.

Localizing query cells along a reference-defined trajectory of human fetal liver hematopoiesis

A successful mapping method should position cells not only within cell type clusters but also along smooth transcriptional gradients, commonly used to model differentiation and activation processes over time (**Fig. 4a**). To test Symphony in a gradient mapping context, we built and mapped to a reference atlas profiling human fetal liver hematopoiesis, containing 113,063 liver cells from 14 donors spanning 7-17 post-conceptional weeks of age and 27 author-defined cell types, sequenced with 10x 3' chemistry (**Fig. 4b, Fig. S5a**)[40]. Trajectory analysis of immune populations with the force directed graph (FDG) algorithm[40] highlights relationships among progenitor and differentiated cell types (**Fig. 4c**). Notably, the hematopoietic stem cell and multipotent progenitor population branches into three major trajectories, representing the lymphoid, myeloid, and megakaryocyte-erythroid-mast (MEM) lineages. This reference contains two forms of annotation for downstream query inference: discrete cell types and positions along differentiation gradients.

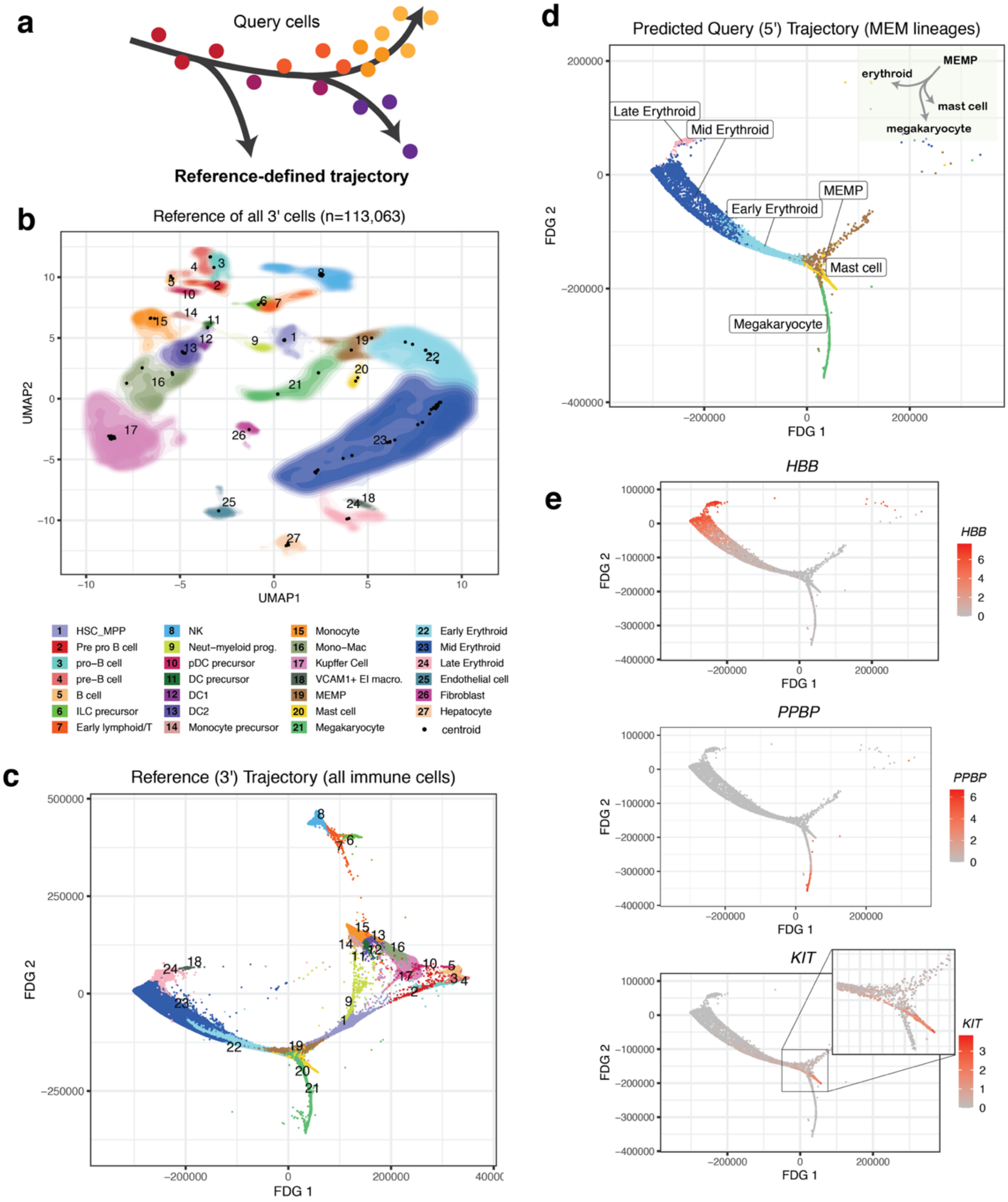

Figure 4 | Localizing query cells along a trajectory of fetal liver hematopoiesis. (a) Symphony can precisely place query cells along a reference-defined trajectory. The reference (n=113,063 cells, 14 donors) was sequenced using 10x 3' chemistry, and the query (n=25,367 cells, 5 donors) was sequenced with 10x 5' chemistry. **(b)** Symphony reference colored by cell types as defined by Popescu et al. (2019). Contour fill represents density of cells. Black points represent soft-cluster centroids in the Symphony mixture model. **(c)** Reference developmental trajectory of 3'-sequenced immune cells (FDG coordinates obtained from original authors). Query cells in the MEM lineages (n=5,141 cells) were mapped against the reference and query coordinates along the trajectory were predicted with 10-NN **(d)**. The inferred query trajectory preserves branching within the MEM lineages, placing terminally differentiated states on the ends. **(e)** Expression of lineage marker genes (*PPBP* for megakaryocytes, *HBB* for erythroid cells, and *KIT* for mast cells). Cells colored by log-normalized expression of gene.

We mapped a query consisting of 21,414 new cells from 5 of the original 14 donors, sequenced with 10x 5' chemistry. We first inferred query cell types with k-NN classification (**Methods**) and confirmed accurate cell type assignment based on the authors' independent query annotations[40] (median cell type F1=0.92 across 14 held-out donor experiments within 3' dataset only, median cell type F1=0.83 for the 5'-to-3' experiment; **Fig. S6**). To evaluate query trajectory inference, we used the Symphony joint embedding to position query cells from the MEM lineage (n=5,141) in the reference-defined trajectory by averaging the 10 nearest reference cell FDG coordinates. The inferred query trajectory (**Fig. 4d**) recapitulated known branching from MEM progenitors (MEMPs, brown) into distinct megakaryocyte (green), erythroid (blue, pink), and mast cell (yellow) lineages. Moreover, transitions from MEMPs to differentiated types were marked by gradual changes in canonical marker genes (**Fig. 4e**): *PPBP* for megakaryocytes, *HBB* for erythrocytes, and *KIT* for mast cells. These gradual expression patterns are consistent with correct placement of query cells along differentiation gradients.

Inferring query surface protein marker expression by mapping to a reference assayed with CITE-seq

Recent technological advances in multimodal single-cell technologies (e.g., CITE-seq) make it possible to simultaneously measure mRNA and surface protein expression from the same cells using oligonucleotide-tagged antibodies[41,42]. With Symphony, we can construct a reference from these data, map query cells from experiments that measure only mRNA expression, and infer surface protein expression for the query cells to expand possible analyses and interpretations (**Fig. 5a**).

To demonstrate this, we used a CITE-seq dataset that measures the expression of whole-transcriptome mRNA and 30 surface proteins on 500,089 peripheral blood memory T cells from 271 samples[43]. We leveraged both mRNA and protein features to build a multimodal reference from 80% of samples (n=217) and map the remaining 20% of samples (n=54). Instead of using PCA, which is best for one modality[44], we used canonical correlation analysis (CCA) to embed reference cells into a space that leverages both. Specifically, CCA constructs a pair of correlated low-dimensional embeddings, one for mRNA and one for protein features, each with a linear projection function akin to gene loadings in PCA.

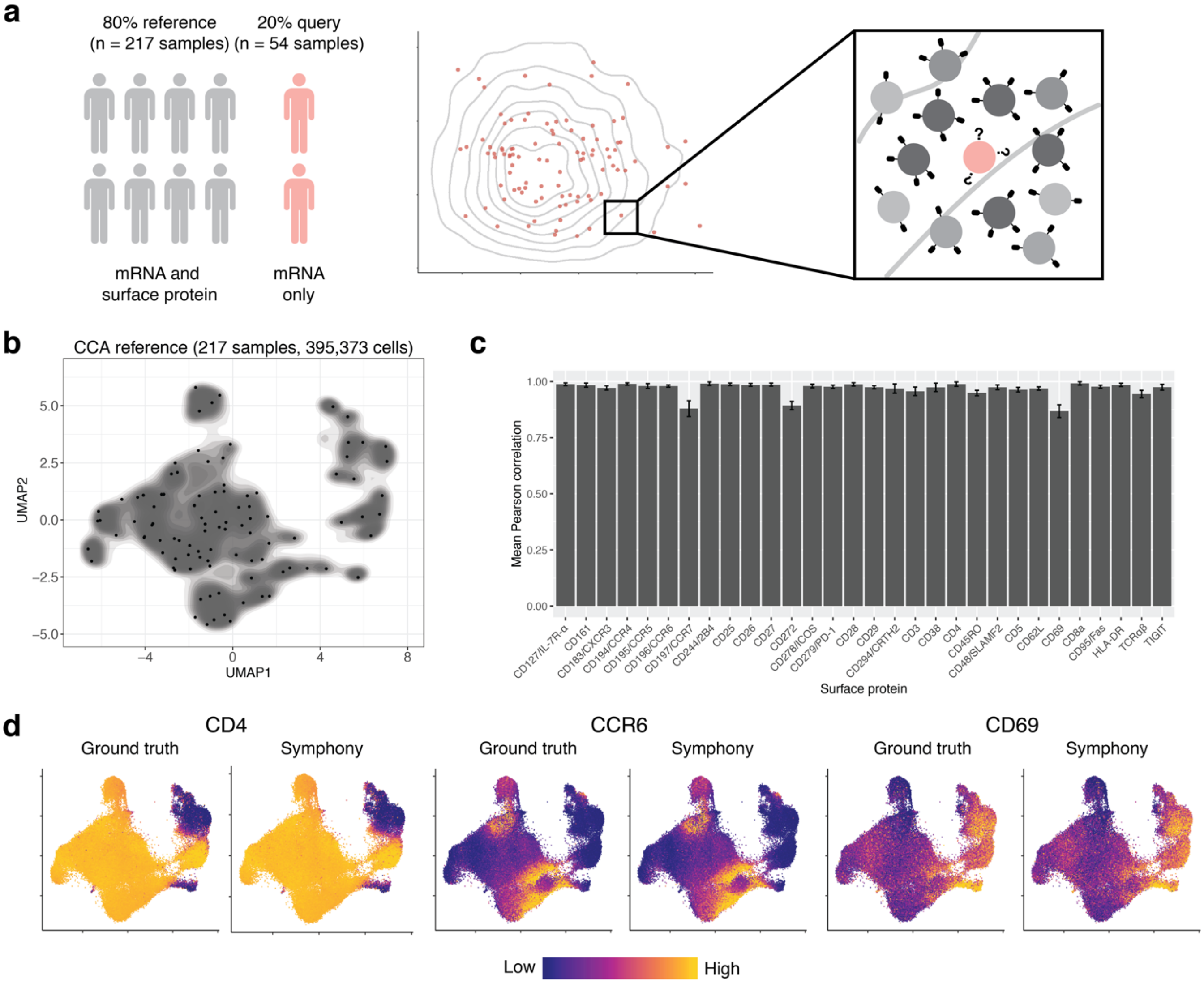

Figure 5 | Mapping onto a multimodal reference to infer query surface protein expression in memory T cells. (a) Schematic of multimodal mapping experiment. The dataset was divided into training and test sets (80% and 20% of samples, respectively). The training set was used to build a Symphony reference, and the test set was mapped onto the reference to predict surface protein expression in query cells (pink) based on 50-NN reference cells (gray). **(b)** Symphony reference built from mRNA/protein CCA embedding. Contour fill represents density of reference cells. Black points represent soft-cluster centroids in the Symphony mixture model. **(c)** We measured the accuracy of protein expression prediction with the Pearson correlation between predicted and ground truth expression for each surface protein across query cells in each donor. Bar height represents the average per-donor correlation for each protein, and error bars represent standard deviation. **(d)** Ground truth and predicted expression of CD4, CCR6, and CD69 based on CCA reference. Ground truth is the 50-NN-smoothed expression measured in the CITE-seq experiment. Colors are scaled independently for each marker from minimum (blue) to maximum (yellow) expression.

We corrected reference batch effects in CCA space with Harmony and built a Symphony reference (**Fig. 5b**), saving the gene loadings for the CCA embedding from mRNA features. Then, we mapped the held-out query using only mRNA expression to mimic a unimodal scRNA-seq experiment, reserving the measured query protein expression as a ground truth for validation. We accurately predicted the surface protein expression of each query cell using the 50-NN average from the reference cells in the harmonized embedding. For all proteins, we found strong concordance between predicted and (50-NN smoothed) measured expression (Pearson r: 0.88-0.99, **Fig. 5c,d**). For all but three proteins, we achieved comparable results with as few as 5 or 10 nearest neighbors (**Fig. S7a**).

We note that it is also possible to conduct the same analysis with a unimodal PCA-based reference built from the cells' mRNA expression only. This approach has slightly worse performance for some proteins (Pearson r: 0.65-0.97, **Fig. S7b-d**), demonstrating that a reference built jointly on both mRNA and protein permits better inference of protein expression than an mRNA-only reference, which is consistent with previous observations that mRNA expression is not fully representative of protein expression[41,42]. This analysis highlights how users can start with a low-dimensional embedding other than PCA, such as CCA, to better capture rich multimodal information in the reference.

Discussion

We frame reference mapping as a specialized case of integration, between one dataset and a second larger, more comprehensive, and previously integrated dataset. Because the reference is already integrated, it is natural to use the same mathematical framework from the integration to perform mapping. For instance, the scArches[26] algorithm uses an autoencoder-based framework to map to references built with autoencoder-based integration algorithms trVAE[46] and CVAE. Similarly, Symphony uses the mixture modeling framework to map to references built with Harmony mixture modeling integration. Symphony compresses the reference by extracting relevant reference-derived parameters from the mixture model to map query cells in seconds. With this compression, references can be distributed without the need to share raw expression data or donor-level metadata, which enables data privacy[28–32]. Symphony compression greatly reduces the size of a reference dataset: for the memory T cell dataset of 500,089 cells, the raw expression matrix is 8.9 GB, whereas the Symphony minimal reference elements are 1.3 MB.

Useful reference atlases contain annotations not present in the query, such as cell type labels (**Fig. 3**), trajectory coordinates (**Fig. 4**), or multimodal measurements (**Fig. 5**). Transfer of these annotations from reference to query is an open area of research that includes algorithms for automated cell type classification[47]. We approach annotation transfer in two steps. We first learn a predictive model in the reference embedding, then map query cells and use their reference coordinates to predict query annotations. In this two-step approach, Symphony mapping provides a feature space but is otherwise independent from the choice of downstream inference model. In PBMC type prediction (**Fig. S3**), we used Symphony embeddings to train multiple competitive classifiers: k-NN, SVM, and logistic regression. In our analyses, we were encouraged to find that a simple k-NN classifier can achieve high performance with only 5-10 neighbors. In practice, users can choose more complex inference models if it is warranted for certain annotation types. Moreover, we expect prediction results to improve with more accurate and reproducible annotation methods, such as consistent cell type taxonomies provided by the Cell Ontology[48] project and better modeling of multimodal expression data[17].

We defined three conditions under which Symphony and *de novo* integration with Harmony yield equivalent results. In subsequent examples, we showed that Symphony still performs well when the last two conditions are relaxed. The pancreas query contains more cells than its reference (**condition II**), while the liver hematopoiesis reference and query overlap in donors (**condition III**). Condition I, which requires comprehensive cell type coverage in the reference, is less flexible. When the query contains a brand new cell type, it will be aligned to its most transcriptionally similar reference cluster. Note that condition I only pertains to cell types and not clinical and biological contexts. For instance, we successfully mapped mouse pancreas query to an entirely human pancreas reference (**Fig. 3**), because the same pancreatic cell types are shared in both species. Mapping novel cell types is a current limitation and important direction for future work. For now, we advise users interested in novel cell type discovery to supplement a Symphony analysis with *de novo* analyses of the query alone.

Instead of one monolithic reference, we expect the proliferation of multiple, well-annotated specialized references. For instance, the memory T cell reference (**Fig. 5**) will be useful to annotate fine-grained T cell states, while an unsorted PBMC reference (**Fig. 2**) would better suit annotation of more diverse immune populations. Similarly, a reference with only healthy individuals is useful for annotation of cell types, while a reference with both healthy and diseased individuals is useful for annotation of cell types and pathological cell states.

As large-scale tissue and whole-organism single-cell reference atlases become available in the near future, Symphony will enable investigators to leverage the rich information in these references to perform integrative analyses and rapidly transfer reference coordinates and diverse annotations to new datasets.

Methods

1. Symphony

1.1 Symphony overview

The goal of single-cell reference mapping is to embed newly assayed query cells into an existing comprehensive reference atlas, facilitating the automated transfer of annotations from the reference to the query. The optimal mapping method needs to be able to operate at various levels of resolution, capture continuous intermediate cell states, and scale to multimillion cells[17]. Consider a scenario in which we wish to map a query of *m* cells against reference datasets with *n* cells, where *m*<<*n*. Unsupervised integration of measurements across donors, studies, and technological platforms is the standard way to compare single cell datasets and identify cell types. Hence, a “gold standard” reference mapping strategy might be to run Harmony integration on all *m*+*n* cells *de novo*. However, this approach is impractical because it is cumbersome and time-intensive to process all the cell-level data for the reference datasets every time a user wishes to reharmonize it with a query. Instead, we envision a pipeline where a reference atlas need only be carefully constructed and integrated once, and all subsequent queries can be rapidly mapped into the same stable reference embedding.

Symphony is a reference mapping method that efficiently places query cells in their precise location within an integrated low-dimensional embedding of reference cells, approximating *de novo* harmonization without the need to reintegrate the reference cells. Symphony is comprised of two algorithms: reference compression and mapping. Expanding upon the linear mixture model framework introduced in Harmony[18], Symphony compression takes in an integrated reference and faithfully compresses it by capturing the components of the model into efficient data structures. The output of reference compression is the minimal set of elements needed for mapping **(Fig. S1b)**. The Symphony mapping algorithm takes as input a new query dataset as well as minimal reference elements and returns the appropriate locations of the query cells within the integrated embedding **(Fig. S1c)**.

Once a harmonized reference is constructed and compressed using Symphony, subsequent mapping of query cells executes within seconds (**Fig. S4**). Efficient implementations of Symphony are available as part of an R package at https://github.com/immunogenomics/symphony, along with several precomputed references constructed from public scRNA-seq datasets. The following sections introduce the Symphony model, then describes Symphony compression and mapping in terms of the underlying data structures and algorithms. We also provide **Supplementary Equations** containing more detailed derivations for reference compression terms.

Glossary

We define all symbols for data structures used in the discussion of Symphony below, including their dimensions and possible values. Dimensions are in terms of the following parameters:

- *n:* the number of reference cells
- *m:* the number of query cells
- *N:* the total number of cells (*n* + *m*)
- *g:* the number of genes in the reference after any gene selection
- *d:* the dimensionality of the embedding (e.g. PCs). *d* applies to both reference and query.
- *b:* the number of batches in the reference
- *c:* the number of batches in the query
- *k:* the number of clusters in the mixture model for reference integration (representing latent cell states)

Reference-related symbols:

$G_r \in \mathbb{R}^{g \times n}$	Input reference gene expression matrix, prior to scaling.
$G_{rs} \in \mathbb{R}^{g \times n}$	Scaled reference gene expression matrix.
$X_r \in \{0,1\}^{b \times n}$	One-hot design matrix assigning reference cells (columns) to batches (rows).
$X_r' \in \{0\}^{c \times n}$	Zero matrix assigning reference cells (columns) to *query* batches (rows). All values are 0 because reference cells do not belong to query batches. This term is used in the derivation for the reference compression terms.

$\mu \in \mathbb{R}^{g \times 1}$ Reference gene means used to center each gene for PCA.

$\sigma \in \mathbb{R}^{g \times 1}$ Reference gene standard deviations used to scale each gene for PCA.

$U \in \mathbb{R}^{g \times d}$ Gene loadings from the original PCA (before Harmony integration).

$Z_r \in \mathbb{R}^{d \times n}$ Original (non-harmonized) PC embedding for reference cells.

$\hat{Z}_r \in \mathbb{R}^{d \times n}$ Integrated embedding for reference cells in harmonized PC (hPC) space, as output by Harmony.

$R_r \in [0,1]^{k \times n}$ Soft cluster assignment of reference cells (columns) to clusters (rows), as output by Harmony. Each column is a probability distribution that sums to 1.

$Y_{cos} \in \mathbb{R}^{d \times k}$ Cluster centroid locations in the harmonized embedding, L2 normalized.

$B_r \in \mathbb{R}^{k \times (1+b) \times d}$ 3D tensor of the estimated parameters (betas and intercepts) of the linear mixture model for each of *k* clusters for the reference cells.

$N_r \in \mathbb{R}^{k \times 1}$ First reference compression term. Vector containing the size of each of the k clusters, effectively the number of reference cells contained within them.

$C \in \mathbb{R}^{k \times d}$ Second reference compression term.

$Ref = \{\mu, \sigma, U, Y_{cos}, N_r, C\}$ Symphony minimal reference elements comprising $\mu, \sigma, U, Y_{cos}, N_r, C$.

Query-related symbols:

$G_q \in \mathbb{R}^{g \times m}$ Input query gene expression matrix, prior to scaling.

$G_{qs} \in \mathbb{R}^{g \times m}$ Query gene expression matrix, scaled by *reference* gene means μ and standard deviations σ.

$X_q \in \{0,1\}^{c \times m}$ Design matrix assigning query cells (columns) to query batches (rows).

$Z_q \in \mathbb{R}^{d \times m}$ Query cell locations in original (non-harmonized) PC embedding.

$\hat{Z}_q \in \mathbb{R}^{d \times m}$ Approximate query cell locations in integrated embedding (hPC space). Output of Symphony reference mapping.

$R_q \in [0,1]^{k \times m}$ Soft cluster assignment of query cells (columns) to clusters (rows). Each column is a probability distribution that sums to 1.

$B_q \in \mathbb{R}^{k \times (1+c) \times d}$ 3D tensor of the estimated parameters (betas and intercepts) of the linear mixture model for each of *k* clusters.

1.2 Symphony model and conditions for equivalence to Harmony integration

Symphony and Harmony both use a linear mixture model framework, but the two methods perform different tasks: Harmony integrates a reference, whereas Symphony compresses the reference and enables efficient query mapping. To motivate the Symphony model, it is helpful to first briefly review the mixture model, which serves as the basis. Harmony integrates scRNA-seq datasets across batches (e.g. multiple donors, technologies, studies) and projects the cells into a harmonized embedding where cells cluster by cell type rather than batch-specific effects. Harmony takes as input a low-dimensional embedding of cells (Z) and design matrix with assignments to batches (X) and outputs a harmonized embedding $(\hat{Z})$ with batch effects removed. Briefly, Harmony works by iterating between two subroutines—maximum diversity clustering and linear mixture model correction—until convergence. In the clustering step, cells are probabilistically assigned to soft clusters with a variant of soft *k*-means with a diversity penalty favoring clusters represented by multiple datasets rather than single datasets. In the correction step, each cluster learns a cluster-specific linear model that explains cell locations in PC space as a function of a cluster-specific intercept and batch membership. Then, cells are corrected by cell-specific linear factors weighted by cluster membership to remove batch-dependent effects. The full algorithm and implementation are detailed in Korsunsky et al. (2019)[47].

In the scenario of mapping *m* query cells against *n* reference cells, the *de novo* integration strategy would model all cells as in (1), where the *H* subscript denotes the Harmony solution, in contrast to the Symphony model which is presented in (2). Let $X_H \in \{0,1\}^{(c+b)\times(m+n)}$ represent the one-hot encoded design matrix assigning all cells across batches. X_H^* denotes X_H augmented with a row of 1s for the batch-independent intercept term: $X_H^* = 1||X_H$. The intercept terms represent cluster centroids (location of "experts" in the mixture of experts model). Z_H represents the low-dimensional PCA embedding of all cells. R_H represents the probabilistic assignment of cells across *k* clusters, and $diag(R_{Hk}) \in \mathbb{R}^{N\times N}$ denotes the diagonalized kth row of R_H. For each cluster *k*, the parameters of the linear mixture model $B_k \in \mathbb{R}^{(1+c+b)\times d}$ can therefore be solved for as in (1), using ridge regression with ridge penalty hyperparameter λ. Note that we do not penalize the batch-independent intercept term: $\lambda_0 = 0$, $\forall_{a\in[1:(c+b)]}\lambda_a = 1$.

De novo Harmony model:

$$B_k = (X_H^* \, diag(R_{Hk}) X_H^{*T} + \lambda I)^{-1} X_H^* diag(R_{Hk}) \, Z_H^T \qquad (1)$$

The goal of Symphony mapping is to add new query cells to the model in order to estimate and remove the query batch effects. Symphony mapping approximates *de novo* Harmony integration on all cells, except the reference cell positions in the harmonized embedding do not change. In order for Symphony mapping to be equivalent to *de novo* Harmony, several conditions must be met:

I. All cell states represented in the query dataset are captured by the reference datasets—i.e. there are no completely novel cell types in the query.

II. The number of reference cells is much larger than the query (*m*<<*n*).

III. The query dataset is obtained independent of the reference datasets—i.e. the reference batch design matrix (X_r) has no interaction with the query batch design matrix (X_q).

We consider these to be fair assumptions for large-scale reference atlases, allowing Symphony to make three key approximations:

(1) With a large reference, the reference-only PCs approximate the PCs for the combined reference and query datasets. This allows us to project the query cells into the pre-harmonized reference PCA space using the reference gene loadings (U).

(2) The cluster centroids (Y) for the integrated reference cells approximate the cluster centroids from harmonizing all cells.

(3) The reference cell cluster assignments (R_r) remains approximately stable with the addition of query cells.

Given these approximations, we can thereby harmonize the reference cells *a priori* and save the reference-dependent portions of the Harmony mixture model (**Supplementary Equations**). In Symphony, we model the reference cells as already harmonized with batch effects removed, so we can thereafter ignore the reference design matrix structure. The Symphony design matrix $X \in [0,1]^{c \times N}$ assigns all cells (reference and query) to *query* batches only. X^* denotes X augmented with a row of 1s

($X^*_{[0,\cdot]}$) corresponding to the batch-independent intercepts (we model the intercepts for all cells). The remaining *c* rows ($X^*_{[1:c,\cdot]}$) represent the one-hot batch assignment of the cells among the *c* query batches. Note that for the reference cell columns, these values are all 0 since the reference cells do not belong to any *query* batches. The parameters ($B_{qk} \in \mathbb{R}^{(1+c)\,\times\,d}$) of the model for each cluster *k* can then be solved for as in (2). Similar to Harmony, we use ridge regression penalizing the non-intercept terms, where $\lambda_0 = 0$, $\forall_{a\in[1:c]}\lambda_a = 1$.

Symphony model:

$$B_{qk} \approx (X^*\,diag(R_k)\,X^{*T} + \lambda I)^{-1}X^*\,diag(R_k)\,Z^T \qquad (2)$$

The matrix $R \in \mathbb{R}^{k\times N}$ denotes the assignment of query and reference cells (columns) across the reference clusters (rows). $Z \in \mathbb{R}^{d\times N}$ denotes the horizontal matrix concatenation of the uncorrected query cells in original PC space (Z_q) and corrected reference cells in harmonized space ($\hat{Z}_r$). For each cluster *k*, let matrix $B_{qk} \in \mathbb{R}^{(1+c)\,\times\,d}$ represent the query parameters to be estimated. The first row of B_{qk} represents the batch-independent intercept terms, and the remaining *c* rows of B_{qk} represent the query batch-dependent coefficients, which can be regressed out to harmonize the query cells with the reference. Note that the intercept terms from Symphony mapping should equal the cluster centroid locations from the integrated reference since the harmonized reference cells are modeled only by a weighted average of the centroid locations for the clusters over which it belongs (and a cell-specific residual). Hence, the reference cell positions should not change when removing query batch effects.

The matrices X^*, R_k, and Z in (2) can be partitioned into query and reference-dependent portions. In the **Supplementary Equations**, we show in detail how the reference-dependent portions can be further simplified into a *k* x 1 vector and *k* x *d* matrix (N_r and C), which we call “reference compression terms.” Intuitively, the vector N_r contains the size (in cells) of each reference cluster. The matrix $C = R_r\hat{Z}_r^T$ does not have as intuitive an explanation but follows from the derivation (**Supplementary Equations**). These terms can be computed at the time of reference building and saved as part of the minimal reference elements to reduce the necessary computations during mapping.

1.3 Reference building and compression

Reference compression is the key idea that allows for the efficient mapping of new query cells onto the harmonized reference embedding without the need to reintegrate all cells. To construct a Symphony reference with minimal elements needed for mapping, reference cells are first harmonized in a low-dimensional space (e.g. PCs) to remove batch-dependent effects. Symphony then compresses the Harmony mixture model components to be saved for subsequent query mapping.

Data structures

Symphony takes as input a gene expression matrix for reference cells (G_r) and corresponding one-hot-encoded design matrix (X_r) containing metadata about assignment of cells to batches. It outputs a set of data structures, referred to as the Symphony minimal reference elements, that captures key information about the reference embedding that can be subsequently used to efficiently map previously unseen query cells (**Algorithm 1**). These components include the gene mean (μ) and standard deviation (σ) used to scale the genes, the PCA gene loadings (U), the final L2-normalized cluster centroid locations (Y_{cos}), and precomputed values which we call the “reference compression terms” (N_r and C) that expedite the correction step of query mapping (**Supplementary Equations**). These elements are a subset of the components available once Harmony integration is applied to the reference cells. Note that other input embeddings, such as canonical correlation analysis (CCA), may be used in place of PCA as long as the gene loadings to perform query projection into those coordinates are saved.

Table 1 lists the Symphony minimal reference elements required to perform mapping. **Table 2** shows additional components of a “full” Harmony reference that are not included in the Symphony reference elements. Importantly, the dimensions of the Symphony data structures do not require information on the *n* individual reference cells and hence do not scale with the raw number of reference cells. Rather the components scale with the biological complexity captured (i.e. number of clusters *k* and dimensionality of embedding *d*). Conversely, the Harmony data structures store information on a per-cell basis (*n*). Note that in practice the integrated embedding of reference cells ($\hat{Z}_r$) listed in **Table 2** is needed to perform downstream transfer of annotations from reference to query cells (e.g. k-NN), but it is not required during any computations of the mapping step.

Table 1: Symphony minimal reference elements

$\mu \in \mathbb{R}^{g\times 1}$	Reference gene means used to center each gene for PCA.
$\sigma \in \mathbb{R}^{g\times 1}$	Reference gene standard deviations used to scale each gene for PCA.
$U \in \mathbb{R}^{g\times d}$	Gene loadings to project from expression to PCA (or CCA) space
$Y_{cos} \in \mathbb{R}^{d\times k}$	Cluster centroid locations in harmonized PC space, L2 normalized.
$N_r \in \mathbb{R}^{k\times 1}$	First reference compression term. Vector containing the size of each of the k clusters, effectively the number of reference cells contained within them.
$C \in \mathbb{R}^{k\times d}$	Second reference compression term.

Table 2: Additional components of Harmony reference

$G_r \in \mathbb{R}^{g\times n}$	Input reference gene expression matrix, prior to scaling.
$X_r \in \{0,1\}^{b\times n}$	Design matrix assigning reference cells (columns) to reference batches (rows).
$B_r \in \mathbb{R}^{k\times (1+b)\times d}$	3D tensor of the estimated parameters (betas and intercepts) of the linear mixture model for each of *k* clusters for the reference cells.
$\hat{Z}_r \in \mathbb{R}^{d\times n}$	Integrated embedding for reference cells in harmonized PC (“hPC”) space, as output by Harmony.
$R_r \in [0,1]^{k\times n}$	Soft cluster assignment of reference cells (columns) to clusters (rows), as output by Harmony. Each column is a probability distribution that sums to 1.

Algorithm

Starting from reference cell gene expression, we first perform within-cell library size normalization (if not already done) and variable gene selection to obtain G_r, scaling of the genes to have mean 0 and variance 1 (saving μ and σ for each gene), and PCA to embed the reference cells in a low-dimensional space, saving the gene loadings (U) (**Implementation Details**). Then, the PCA embedding (Z_r) and batch design matrix (X_r) are used as input to Harmony integration to harmonize over batch-dependent sources of variation. Given the resulting harmonized embedding ($\hat{Z}_r$) and final soft assignment of reference cells to clusters (R_r), the locations of the final reference cluster centroids $Y \in \mathbb{R}^{d \times k}$ can be calculated as in (3) and saved.

$$Y = \hat{Z}_r R_r^T \tag{3}$$

Symphony then computes the reference compression terms N_r (intuitively, the number of cells per cluster) and C, which does not have an intuitive explanation but can be directly computed as $C = R_r \hat{Z}_r^T$. Refer to the **Supplementary Equations** for a complete mathematical derivation of the compression terms. Symphony reference building ultimately returns the minimal reference elements: $\mu, \sigma, U, Y_{cos}, N_r$, and C (**Fig. S1a**).

Algorithm 1 Build Symphony reference

```
function BUILDREFERENCE($(G_r, X_r)$)
    $\mu, \sigma, G_{rs} \leftarrow$ SCALE$(G_r)$
    $U, Z_r \leftarrow$ PCA$(G_{rs})$
    $\hat{Z}_r, R_r \leftarrow$ HARMONIZE$(Z_r, X_r)$
    $Y \leftarrow \hat{Z}_r R_r^T$
    $Y_{cos} \leftarrow Y_{[\cdot,i]} / \|Y_{[\cdot,i]}\|_2$        ▷ $L_2$ normalize cluster centroids
    $N_r \leftarrow rowSums(R_r)$        ▷ First compression term
    $C \leftarrow R_r \hat{Z}_r^T$        ▷ Second compression term
    $Ref \leftarrow (\mu, \sigma, U, Y_{cos}, N_r, C)$
    return $Ref$        ▷ Return minimal reference elements
```

1.4 Symphony mapping

The Symphony mapping algorithm localizes new query cells to their appropriate locations in the harmonized embedding without the need to run integration on the reference and query cells altogether. The joint embedding of reference and query cells can be used for downstream analyses, such as transferring cell type annotations from the reference cells to the query cells.

Data structures

Symphony mapping takes as input the gene expression matrix for query cells (G_q), query design matrix assigning query cells to batches (X_q), and the precomputed minimal elements for a reference (Ref). It outputs a query object containing the locations of query cells in the integrated reference embedding ($\hat{Z}_q$; **Algorithm 2**). **Table 3** lists the components of the query object that is returned by Symphony.

Table 3: Components of Symphony query

$G_q \in \mathbb{R}^{g \times m}$	Input query gene expression matrix, prior to scaling.
$X_q \in \{0,1\}^{c \times m}$	Design matrix assigning query cells (columns) to query batches (rows).
$Z_q \in \mathbb{R}^{d \times m}$	Query cell locations in original (non-harmonized) PC embedding.
$\hat{Z}_q \in \mathbb{R}^{d \times m}$	Approximate query cell locations in integrated embedding (hPC space).
$R_q \in [0,1]^{k \times m}$	Soft cluster assignment of query cells (columns) to clusters (rows). Each column is a probability distribution that sums to 1.
$B_q \in \mathbb{R}^{k \times (1+c) \times d}$	3D tensor of the estimated parameters (betas and intercepts) of the linear mixture model for each of *k* clusters.

Algorithm

The input to the query mapping procedure is a gene expression matrix (G_q) and design matrix (X_q) for query cells, and the output is the locations of the cells in the harmonized embedding ($\hat{Z}_q$). At a high level, the mapping algorithm first projects the query cells into the original, non-harmonized PC space as the reference cells using the reference gene loadings (U) and assigns probabilistic cluster membership across the reference cluster centroid locations. Then, the query cells are modeled using the Symphony mixture model and corrected to their approximate locations in the integrated embedding by regressing out the query batch-dependent effects (**Algorithm 2**).

Projection of query cells into pre-harmonized PC Space

Symphony projects the query cells into the same original PCs (Z_r) as the reference. Symphony assumes that, given a much smaller query compared to the reference (*m*<<*n*), the PCs will remain approximately stable with the addition of query cells. To project the query cells, we first subset the query expression data by the same variable genes used in reference building and scale the normalized expression of each gene by the same mean and standard deviations used to scale the reference cells (μ, σ). Let G_{qs} denote the query gene expression matrix scaled by the reference gene means and standard deviations. We can then use the reference gene loadings (U) to project G_{qs} into reference PC space. In (4), $Z_q \in \mathbb{R}^{d \times m}$ denotes the PC embedding for the query cells. Note that if an alternate starting embedding (e.g. CCA) is used instead of PCA, the gene loadings must be saved to enable this query projection step.

$$Z_q = U^T G_{qs} = \Sigma_q V_q^T \tag{4}$$

Soft assignment across reference clusters

Once the query cells are projected into PC space, we soft assign the cells to the reference clusters using the saved reference centroid locations (Y_{cos}). Symphony assumes that the reference cluster centroid locations remain approximately stable with the addition of a much smaller query dataset since the query contains no novel cell types. Under these conditions, we use a previously published objective function for soft *k*-means clustering (5), which includes a distance term and an entropy regularization term over R weighted by hyperparameter σ. This is the same objective function as the clustering step of Harmony, except it does not include the diversity penalty term. In Harmony, the purpose of the diversity term is to penalize clusters that are only represented by one or a few datasets (suggesting they do not represent true cell types). In contrast, Symphony does not require the use of a diversity penalty because the reference centroids have already been established. Furthermore, the query cell types can comprise a subset of a larger set of reference cell types, and therefore not all clusters are necessarily expected to be represented in the query. We can solve for R_q, the optimal probabilistic assignment for query cells across each of the k reference clusters (**Implementation Details**).

$$\min_{R,Y} \sum_{i,k} R_{ki} \|Z_i - Y_k\|^2 + \sigma R_{ki} \log R_{ki} \tag{5}$$

$$\text{s.t. } \forall_i \forall_k R_{ki} > 0, \forall_i \sum_{k=1}^{K} R_{ki} = 1$$

Mixture of experts correction

The final step in Symphony mapping is to model then remove the query batch effects to obtain $\hat{Z}_q$, the approximate location of query cells in the harmonized reference embedding. In equation (2), we modeled the reference and query cells together and wish to solve for the query parameters $B_{qk} \in \mathbb{R}^{(1+c) \times d}$ for each cluster *k*. The reference-dependent terms in (2) were previously computed and saved in compressed form (N_r and C). With R_q and Z_q calculated from query cell projection and clustering, we can finally solve for B_{qk}. Similar to the correction step of Harmony, we obtain cell-specific correction values for the query cells by removing the batch-dependent terms captured in $B_{qk[1:c,\cdot]}$. Note that the reference batch terms are neither modeled nor corrected during reference mapping, so the harmonized reference cells do not move.

The final locations of the query cells in the harmonized embedding are estimated by iterating over all *k* clusters and subtracting out the non-intercept batch terms for each cell weighted by cluster membership (6). Intuitively, the query centroids are moved so that they overlap perfectly with the reference centroids in the harmonized embedding. $\hat{Z}_{q[i]}$ denotes the approximate location in harmonized PC space for query cell *i*.

$$Z_{q[i]} = \sum_k R_{q[k,i]} \left[B^T_{qk[0,\cdot]} + B^T_{qk[1:c,i]} X_q \right] + \varepsilon$$

$$\hat{Z}_{q[i]} = Z_{q[i]} - \sum_k R_{q[k,i]} \, B^T_{qk[1:c,\cdot]} X_q \qquad (6)$$

$$\hat{Z}_{q[i]} = \sum_k R_{q[k,i]} \, B^T_{qk[0,\cdot]} + \varepsilon$$

Algorithm 2 Map query cells onto reference

```
  function QUERYMAPPING($G_q, X_q, Ref$)
      $G_{qs} \leftarrow$ SCALE($G_q, Ref\$\mu, Ref\$\sigma$)        ▷ $\$$ denotes accessing a component of $Ref$
```

```
        $Z_q \leftarrow$ PCAPROJECTION$(G_{qs}, Ref\$U)$
        $R_q \leftarrow$ CLUSTER$(Z_q, Ref\$Y_{cos})$
        $\hat{Z}_q \leftarrow Z_q$
        for $k \leftarrow 1 \dots k$ do
            $E \leftarrow X_q^* R_q^{(k)} X_q^{*T}$                ⊳ $X_q^*$: query design matrix augmented with row of 1s
            $E_{[0,0]} \leftarrow E_{[0,0]} + Ref\$N_{r(k)}$
            $F \leftarrow X_q^* R_q^{(k)} Z_q^T$
            $F_{[0,\cdot]} \leftarrow F_{[0,\cdot]} + Ref\$C_{[k,\cdot]}$
            $B_{qk} \leftarrow (E + \lambda I)^{-1}(F)$
            $B_{qk[0,\cdot]} \leftarrow 0$                          ⊳ Do not correct the intercept terms
            $\hat{Z}_q \leftarrow \hat{Z}_q - B_{qk}^T X_q^* R_q^{(k)}$
    return $\hat{Z}_q$                                  ⊳ Return query locations
```

1.5 Implementation details

Reference building and compression

Variable gene selection and scaling

Starting with the gene expression matrix for reference cells, we perform log(CP10K) library size normalization of the cells (if not already done), subset by the top g variable genes by the vst method (as provided in Seurat[49]), which fits a line to the log(variance) and log(mean) relationship using local polynomial regression, then standardizes the features by observed mean and expected variance, calculating gene variance on the standardized values, which is re-implemented as a standalone function at https://github.com/immunogenomics/singlecellmethods. The data is scaled such that the expression of each gene has a mean expression of 0 and variance of 1 across all cells.

PCA

We perform dimensionality reduction on the scaled gene expression G_{rs} using principal component analysis (PCA). PCA projects the data a low-dimensional, orthonormal embedding that retains most of the variation of gene expression in the dataset. Singular value decomposition (SVD) is a matrix factorization method that can calculate the PCs for a dataset. Here, we use SVD (irlba package in R[48]) to perform PCA. SVD states that matrix G_{rs} with dimensions $g \times n$ can be factorized as:

$$G_{rs} = U\Sigma V^T \quad (7)$$

In (7), $\Sigma V^T = Z_r$ (dimensions $d \times n$) represents the embedding of reference cells in PC space, after truncating the matrix on the first d (by default, $d = 20$) PCs. The gene loadings ($U \in \mathbb{R}^{g \times d}$) are saved. Note that an alternative embedding, such as canonical correlation analysis (CCA) may be used in place of PCA, as long as the gene loadings are saved.

Harmony integration

The PCA embedding (Z_r) is then input to Harmony for dataset integration. By default, Symphony uses the default parameters for the cluster diversity enforcement ($\theta = 2$), the entropy regularization hyperparameter for soft *k*-means ($\sigma = 0.1$), and the number of clusters $k = \min\left(100, \frac{n}{30}\right)$. We save the L2-normalized cluster centroid locations Y_{cos} to the reference object since query mapping employs a cosine distance metric. If the reference has a single-level batch structure, no integration is performed, and the clusters are defined using soft k-means.

Query mapping

Normalization and scaling

The gene expression for query cells are assumed to be library size normalized in the same manner that was used to normalize the reference cells (e.g. log(CP10K)). During scaling, the query data is subset by the same variable genes from the reference datasets, and query gene expression is scaled by the *reference* gene means and standard deviations. Any genes present in the query but not the reference are ignored, and any genes present in the reference but not the query have scaled expression set to 0.

Clustering step uses cosine distance

As in Harmony, in practice we use cosine distance rather than Euclidean distance in the clustering step. For the computation of the distance term, we L2-normalize the columns (cells) of Z and columns (centroids) of Y_k such that the squared values sum to 1 across each column. Let the terms $Z_{q\_\cos\,[\cdot,i]}$ and $Y_{\cos\,[\cdot,\mathrm{k}]}$ represent the L2-normalized locations of query cell i and the reference centroid for cluster k in PC space, respectively. We compute the cosine distance between the cells and centroids. Since all $Z_{q\_\cos\,[\cdot,i]}$ and $Y_{\cos\,[\cdot,\mathrm{k}]}$ each have unity norm, the squared Euclidean distance $\left\|Z_{q\_\cos\,[\cdot,i]} - Y_{\cos\,[\cdot,\mathrm{k}]}\right\|^2$ is equivalent to the cosine distance $2\left(1 - \cos(Y_{\cos\,[\cdot,\mathrm{k}]}, Z_{q\_\cos\,[\cdot,i]})\right) = 2(1 - Y^T_{\cos\,[k,\cdot]} Z_{q\_\cos\,[\cdot,i]})$. Therefore, the objective function for query assignment to centroids becomes:

$$\min_{R,Y} \sum_{i,k} 2R_{q[k,i]}(1 - Y^T_{\cos\,[k,\cdot]} Z_{q\_\cos\,[\cdot,i]}) + \sigma R_{q[k,i]} \log R_{q[k,i]} \qquad (8)$$

$$\text{s.t. } \forall_i \forall_k R_{q[k,i]} > 0, \forall_i \sum_{k=1}^{K} R_{q[k,i]} = 1$$

We can solve the optimization problem using an expectation-maximization framework. Following the same strategy as Korsunsky et al. (2019), we calculate R_i, the optimal probabilistic assignment for each query cell i across each of the k reference clusters. In (9), we can interpret $R_{q[k,i]}$ as the probability that query cell i belongs to cluster k. The denominator term simply ensures that for any given cell i, the probabilities across all k clusters sum to one.

$$R_{q(k,i)} = \frac{\exp\left(-\frac{2}{\sigma}(1 - Y^T_{\cos\,[k,\cdot]} Z_{q\_\cos\,[\cdot,i]})\right)}{\sum_{k=1}^{K} \exp\left(-\frac{2}{\sigma}(1 - Y^T_{\cos\,[k,\cdot]} Z_{q\_\cos\,[\cdot,i]})\right)} \qquad (9)$$

2. Analysis details

2.1 10x PBMCs and pancreas examples

Preprocessing scRNA-seq data

As the three 10x PBMCs and reference pancreas datasets were previously preprocessed by our group as part of the Harmony publication, we used the same log(CP10K) normalized expression data, filtered as described in Korsunsky et al. (2019)[50]. The PBMCs consist of cells from three technologies: 3'v1 (n=4,808 cells), 3'v2 (8,372 cells), and 5' (7,612 cells). The pancreas reference datasets were each sequenced with a different technology: Fluidigm C1 (n=638 cells), CEL-seq (946 cells), CEL-seq2 (2,238 cells), Smart-seq2 (2,355 cells). The pancreas query dataset (inDrop, n=8,569 human and 1,886 mouse cells) along with author-defined cell type labels were downloaded from https://hemberg-lab.github.io/scRNA.seq.datasets/human/pancreas/.

Constructing the pancreas query with mouse and human

For the pancreas query (Baron et al., 2016), we downloaded both the human and mouse expression matrices. In order to combine the two matrices into a single aggregated query, we "humanized" the mouse expression matrix by mapping mouse genes to their orthologous human genes. This mapping was computed using the biomaRt R package[47], mapping `mgi_symbol` from the `mmusculus_gene_ensembl` database to `hgnc_symbol` from the `hsapien_gene_ensembl` database. We represented this map as a matrix, with mouse genes as rows, human genes as columns, and values in {0,1} assigned to denote whether a mouse gene maps to a human gene. We then normalized the matrix to have each column sum to one, effectively creating a count-preserving probabilistic map from d mouse to D human genes $M \in R^{D \times d}$. Mapping from mouse to human genes is then performed with matrix multiplication: $U_{human} = MU_{mouse}$. Note that while the mouse gene expression matrix U_{mouse} contains only integers ($U_{mouse} \in Z^{d \times N}$), the many-to-many mapping means that the mapped human gene expression matrix U_{human} may contain non-integers ($U_{human} \in R^{D \times N}$). For any human orthologs that were missing in the mouse expression data, we filled in the expression with zeroes. The mouse mapping was based on 2,140 overlapping genes (of the 3,000 variable genes used for reference building) for which the mouse genes had human orthologs in the reference.

Symphony mapping experiments

To construct each reference for query mapping, we aggregated all reference datasets into a single normalized expression matrix and identified the top *g* variable genes across all cells (*g*=2000 for PBMCs, *g*=3000 for pancreas) using the variance stabilizing transformation (vst) procedure[49]. We scaled the genes (mean = 0, variance = 1), performed PCA[18], and ran Harmony on the top 20 PCs and default 100 clusters. For PBMCs, we harmonized over 'technology' with default parameters. For pancreas, we harmonized over 'donor' (θ = 2) and 'technology' (θ = 4), with τ = 5. During Symphony mapping, we specified query 'technology' covariate for PBMCs and query 'donor', 'species', and 'technology' covariates for pancreas.

Constructing gold standard embedding

To construct the gold standard *de novo* Harmony embedding, we concatenated the reference and query datasets together into a single expression matrix, subsetted by the top *g* variable genes over all (both reference and query) cells (*g*=2000 for PBMCs, *g*=3000 for pancreas) and ran Harmony integration on the top 20 PCs[49]. For PBMCs, we harmonized over 'technology' with default parameters. For pancreas, we harmonized over 'donor' (θ = 2) and 'technology' (θ = 4), with τ = 5.

Assigning ground truth cell types

We clustered the cells in the gold standard embedding using the Louvain algorithm as implemented in the Seurat functions *BuildSNN* and *RunModularityClustering*[51]. For PBMCs, we used nn_k = 5 (to capture rare HSCs), nn_eps = 0.5, and resolution = 0.8. For pancreas, we used the same parameters except nn_k = 30. We labeled clusters with ground truth cell types according to expression of canonical lineage marker genes (**Table S2,5**). PBMCs were assigned across 7 types: T (*CD3D*), NK (*GNLY*), B (*MS4A1*), Monocytes (*CD14*, *FCGR3A*), DCs (*FCER1A*), Megakaryocytes (*PPBP*), and HSCs (*CD34*). Pancreas cells were assigned across 9 types: alpha (*GCG*), beta (*MAFA*), gamma (*PPY*), delta (*SST*), acinar (*PRSS1*), ductal (*KRT19*), endothelial (*CDH5*), stellate (*COL1A2*), and immune (*PTPRC*). Clusters were labeled if the AUC (calculated using presto[28]) for the corresponding lineage marker was >0.62. For clusters that did not express a specific lineage marker, we manually assigned a cell type based on the top differentially expressed genes (**Table S2,5**). In the PBMCs, cluster 20 was identified as low-quality cells (high in mitochondrial genes; **Table S2**). We removed all cells in this cluster (n=94) from further analyses. The final ground truth labels were used in downstream analyses and cell type classification accuracy evaluation.

Evaluation of cell type classification accuracy

We predicted query cell types by transferring reference cell type annotations using the *knn* function in the 'class' R package (k=5). Note for the pancreas human cell type classification, we excluded query epsilon and Schwann cells from the accuracy metrics because those cell types are not present in the reference. We calculated overall accuracy across all query cells and cell type F1 scores (the harmonic mean of precision and recall, ranging from 0 to 1). Precision = TP/(TP+FP), recall = TP/(TP+FN), F1 = (2 * precision * recall) / (precision + recall). Cell type F1 was the metric Abdelaal et al. recently used to benchmark automated cell type classifiers.[40] We used their *evaluate.R* script to calculate confusion matrices and F1 by cell type.

Quantifying local similarity between two embeddings

k-NN-correlation (k-NN-corr) is a new metric that quantifies how well a given alternative embedding preserves the local neighborhood structure with respect to a gold standard embedding. Anchoring on each query cell, we calculate (1) the pairwise similarities to its *k* nearest reference neighbors in the gold standard embedding and (2) the similarities between the same query-reference neighbor pairs in an alternate embedding (**Methods**), then calculate the Spearman (rank-based) correlation between (1) and (2). For similarity, we use the radial basis function kernel: *similarity*(x,y) = $\exp(-\|x-y\|^2/(2\sigma^2))$. For each query cell, we obtain a single k-NN-corr value capturing how well the relative similarities to its *k* nearest reference neighbors are preserved. Note that k-NN-corr is asymmetric with respect to which embedding is selected as the gold standard and which is selected as the alternative because the nearest neighbor pairs are fixed based on how they were defined in the gold standard. The distribution of k-NN-corr scores for all query cells can measure the embedding quality, where higher k-NN-corr indicates greater recapitulation of the gold standard. Lower values for *k* assess more local neighborhoods, whereas higher *k* assesses more global structure.

We calculated k-NN-corr between the gold standard Harmony embedding and two alternative embeddings: (1) the full Symphony mapping algorithm (projection, clustering, and correction) and (2) PCA-projection only as a comparison to a batch-naïve mapping. PCA-projection refers to the first step of Symphony mapping, where query cells are projected from gene expression to pre-harmonized PC space: $Z_q = U^T G_q$.

2.2 Fetal liver hematopoiesis trajectory inference example

We obtained post-filtered, post-doublet removal data directly from the authors[52] along with author-defined cell type annotations for 113,063 cells sequenced with 10x 3' end bias and a separate 25,367 cells sequenced with 10x 5' end bias. For building the harmonized reference from all 3' cells, we followed the same variable gene selection procedures as the original authors, using the Seurat variance/mean ratio (VMR) method with parameters min_expr = .0125, max_expr = 3, and min_dispersion = 0.625 (resulting in 1,917 variable genes). For each of 14 held-out donor experiments within the 3' dataset, we integrated the reference with Harmony on 13 donors (θ = 3). During Symphony mapping, we specified query 'donor' covariate. For mapping 5' cells against a 3' reference, we removed two donors (F2 and F5, n=3,953) from the 5' query based on low library complexity (**Fig. S5b**), leaving n=21,414 cells from 5 donors. We integrated the reference (all 14 donors sequenced with 3' end bias) with Harmony over 'donor' (θ = 3). During Symphony mapping, we specified both 'donor' and 'technology' as covariates. We predicted query cell types by transferring reference cell type annotations using the *knn* function in the 'class' R package (k=30). We visualized the aggregated confusion matrix across all 14 held-out donor experiments as well as the confusion matrix for the single 5'-to-3' experiment using ComplexHeatmap R package[40].

For the trajectory inference analysis, we obtained trajectory coordinates from the force directed graph (FDG) embedding of all 3'-sequenced cells from the original authors[53], forming a reference trajectory. We restricted the trajectory to immune cell types only (excluding hepatocytes, fibroblasts, and endothelial). We then mapped a subset of the query cells belonging to the MEM lineage (MEMPs, megakaryocytes, mast cells, early-late erythroid; n=5,141) to the reference-defined trajectory by averaging the FDG coordinates of the 10 reference immune cell neighbors in the Symphony embedding. Note: in addition to the author-provided FDG trajectory, we explored building a trajectory on the reference immune cells with DDRTree[43], but we found that the inferred trajectory was not as clean as FDG.

2.3 Memory T cell surface protein inference example

We used a memory T cell CITE-seq dataset collected from a tuberculosis disease progression cohort of 259 individuals of admixed Peruvian ancestry[54]. The dataset includes expression of the whole transcriptome (33,538 genes) and 30 surface protein markers from 500,089 memory T cells isolated from PBMCs. Including technical replicates, 271 samples were processed across 46 batches.

To assess protein prediction accuracy using Symphony embeddings, we randomly selected 217 samples (411,004 cells), normalized the expression of each gene (log2(CP10K)) and built a Symphony reference based on mRNA expression, correcting for donor and batch. The held-out 54 samples comprised the query that we mapped onto the reference. We predicted the expression of each of the 30 surface proteins in each of the query cells by averaging the protein's expression across the cell's 50 nearest reference neighbors. Nearest neighbors were defined based on Euclidean distance in the batch-corrected low-dimensional embedding. As a ground truth for each protein in each query cell, we computed a smoothed estimate of the cells' measured protein expression by averaging the protein's expression across the cell's 50 nearest neighbors in the batch-corrected complete PCA embedding of all 259 donors. We did not use the cells' raw measured protein expression due to dropout. We computed the Pearson correlation coefficient between our predicted expression and the ground truth expression across all cells per donor for each marker.

To assess protein prediction accuracy based on mapping to a joint mRNA and protein-based Symphony reference, we first built an integrated reference by using canonical correlation analysis (CCA) to project cells into a low-dimensional embedding maximizing correlation between mRNA and protein features. We randomly selected 217 samples (395,373 cells) to comprise this reference, and normalized the expression of each gene (log2(CP10K)), selected the top 2,865 most variable genes, and scaled (mean = 0, variance = 1) all mRNA and protein features. We computed 20 canonical variates (CVs) with the *cc* function in the CCA R package[55] and corrected the mRNA CVs for donor and batch effects with Harmony. Then, we used Symphony to construct a reference based on the batch-corrected CVs, gene loadings on each CV, and mean and standard deviation used to scale each gene prior to CCA. The held-out 54 samples comprised the query that we mapped onto the reference. As described above, we predicted the expression of each of the 30 surface proteins in each of the query cells based on the cell's 5, 10, or 50 nearest neighbors in the reference, estimated the smoothed ground truth expression of each protein in each query cell (now based on the batch-corrected CCA embedding of all 259 donors) and computed the Pearson correlation coefficient for each marker.

2.4 Visualization

For visualizing the embeddings using UMAP[28,33], we used the 'uwot' R package with the following parameters: n_neighbors=30, learning_rate=0.5, init = 'laplacian', metric = 'cosine', min_dist=0.1 (except min_dist=0.3 for fetal liver example). For each Symphony reference, we saved the uwot model at the time of UMAP using the *uwot::save_uwot* function and saved the path to the model file as part of the Symphony reference object. Saving the reference UMAP model allows for the fast projection of new query cells into reference UMAP space from the query embedding from Symphony mapping using the function *uwot::transform*.

To distinguish the reference plots from query plots, we visually present the reference embedding as a contour density instead of individual cells. The density plots were generated using ggplot2 function *stat_density_2d* with geom = 'polygon' and contour_var = 'ndensity'. We provide custom functions to generate these plots as part of the Symphony package.

2.5 Benchmarking against automatic cell type classifiers

We downloaded the PbmcBench benchmarking dataset used by a recent comparison of automatic cell type identification methods[28]. For each of 48 train-test experiments previously described[51], we used the same evaluation metrics (median cell type F1 score) to evaluate Symphony in comparison to the 22 other classifiers. We obtained the numerical F1-score results for all other classifiers for each of the 48 experiments directly from the authors in order to determine Symphony's place within the rank ordering of classifier performance.

During reference building, we tried two different gene selection methods: (1) unsupervised (top 2000 variable genes) and (2) supervised based on identifying the top 20 differentially expressed (DE) genes per cell type. Option (2) was included to give Symphony the same information as prior-knowledge classifiers (e.g. SCINA with 20 marker genes per cell type). We used the 'presto' package[56] for DE analysis. No integration was performed because the reference had a single-level batch structure (clusters were simply assigned using soft k-means). Onto each of 7 references (each representing 1 protocol for donor pbmc1), we mapped either a second protocol for donor pbmc1 (6 experiments) or the same protocol for donor pbmc2 (1 experiment). Given the resulting Symphony joint feature embeddings, we used three downstream classifiers to predict query cell types: 5-NN, SVM with a radial kernel, and glm_net with ridge[34]. A total of 6 Symphony-based classifiers were tested (2 gene selection methods * 3 downstream classifiers).

2.6 Runtime analysis

We downsampled a large memory T cell dataset[34] to create benchmark reference datasets with 20,000, 50,000, 100,000, 250,000, and 500,000 cells. For each, we built a reference (20 PCs, 100 centroids) integrating over 'donor' and mapped three different-sized queries: 1,000, 10,000, and 100,000 cells. To isolate the separate effects of number of query cells and number of query batches on mapping time, we mapped against the 50,000-cell reference: (1) varying the number of query cells (from 1,000 to 10,000 cells) while keeping the number of donors constant and (2) varying the number of query donors (6 to 120 donors) while keeping the number of cells constant (randomly sampling 10,000 cells). We also performed separate experiments varying the number of reference centroids (25 to 400) and number of dimensions (10 to 320 PCs) while keeping all other parameters constant. We ran all jobs on Linux servers allotted 4 cores and 64 GB of memory (Intel Xeon E5-2690 v.3 processors) and used the *proc.time* R function to measure elapsed time.

Data availability

Datasets for all analyses were obtained from the links in **Table S1.** All datasets are publicly available except the memory T cell CITE-seq data, which will be available at GEO accession GSE158769.

Code availability

We provide an implementation of Symphony along with prebuilt references from all examples at https://github.com/immunogenomics/symphony. Scripts reproducing results of this paper will be made available at https://gihub.com/immunogenomics/referencemapping.

Acknowledgements

We thank members of the Raychaudhuri Lab for helpful feedback and comments. We thank members of the Tuberculosis Research Unit (TBRU) LIMAA and Socios En Salud, in particular Megan Murray, Jessica Beynor, Yuriy Baglaenko, Sara Suliman, Ildiko van Rhijn, and Leonid Lecca, for their contributions to generating the memory T cell dataset. We would also like to thank Issac Goh, Muzlifah Haniffa, and other members of the Haniffa Lab for graciously providing preprocessed datasets from their fetal liver hematopoiesis study. This work is supported in part by funding from the National Institutes of Health (UH2AR067677, 1U01HG009088, U01 HG009379, and 1R01AR063759). The project described was supported by award Number T32GM007753 from the National Institute of General Medical Sciences (JBK). The content is solely the responsibility of the authors and does not necessarily represent the official views of the National Institute of General Medical Sciences or the National Institutes of Health.

Author contributions

I.K., J.B.K., and S.R. conceived the project. J.B.K. and I.K. developed the method and performed the analyses under the guidance of S.R. S.R., A.N., and D.B.M. contributed to generating the memory T cell dataset. A.N. performed analysis of the memory T cell dataset. All authors participated in interpretation and writing the manuscript.

Competing interests

SR receives research support from Biogen.

References

1. Klein, A. M. & Treutlein, B. Single cell analyses of development in the modern era. *Development* **146**, (2019).
2. Svensson, V., da Veiga Beltrame, E. & Pachter, L. A curated database reveals trends in single-cell transcriptomics. *biorxv* (2019) doi:10.1101/742304.
3. Han, X. *et al.* Construction of a human cell landscape at single-cell level. *Nature* (2020) doi:10.1038/s41586-020-2157-4.
4. Cao, J. *et al.* The single-cell transcriptional landscape of mammalian organogenesis. *Nature* **566**, 496–502 (2019).
5. Jerber, J. *et al.* Population-scale single-cell RNA-seq profiling across dopaminergic neuron differentiation. *bioRxiv* 2020.05.21.103820 (2020) doi:10.1101/2020.05.21.103820.
6. Zhang, F. *et al.* Defining inflammatory cell states in rheumatoid arthritis joint synovial tissues by integrating single-cell transcriptomics and mass cytometry. *Nat. Immunol.* **20**, 928–942 (2019).
7. Reyes, M. *et al.* An immune-cell signature of bacterial sepsis. *Nat. Med.* **26**, 333–340 (2020).
8. Kotliarov, Y. *et al.* Broad immune activation underlies shared set point signatures for vaccine responsiveness in healthy individuals and disease activity in patients with lupus. *Nat. Med.* **26**, 618–629 (2020).
9. Schafflick, D. *et al.* Integrated single cell analysis of blood and cerebrospinal fluid leukocytes in multiple sclerosis. *Nat. Commun.* **11**, 247 (2020).

10. Smillie, C. S. *et al.* Intra- and Inter-cellular Rewiring of the Human Colon during Ulcerative Colitis. *Cell* **178**, 714-730.e22 (2019).

11. Litviňuková, M. *et al.* Cells of the adult human heart. *Nature* (2020) doi:10.1038/s41586-020-2797-4.

12. Rozenblatt-Rosen, O., Stubbington, M. J. T., Regev, A. & Teichmann, S. A. The Human Cell Atlas: from vision to reality. *Nature* **550**, 451–453 (2017).

13. Haghverdi, L., Lun, A. T. L., Morgan, M. D. & Marioni, J. C. Batch effects in single-cell RNA-sequencing data are corrected by matching mutual nearest neighbors. *Nat. Biotechnol.* **36**, 421–427 (2018).

14. Hie, B., Bryson, B. & Berger, B. Efficient integration of heterogeneous single-cell transcriptomes using Scanorama. *Nat. Biotechnol.* **37**, 685–691 (2019).

15. Welch, J. D. *et al.* Single-Cell Multi-omic Integration Compares and Contrasts Features of Brain Cell Identity. *Cell* **177**, 1873-1887.e17 (2019).

16. Lopez, R., Regier, J., Cole, M. B., Jordan, M. I. & Yosef, N. Deep generative modeling for single-cell transcriptomics. *Nat. Methods* **15**, 1053–1058 (2018).

17. Korsunsky, I. *et al.* Fast, sensitive and accurate integration of single-cell data with Harmony. *Nat. Methods* **16**, 1289–1296 (2019).

18. Stuart, T. *et al.* Comprehensive Integration of Single-Cell Data. *Cell* **177**, 1888-1902.e21 (2019).

19. He, Z., Brazovskaja, A., Ebert, S., Camp, J. G. & Treutlein, B. CSS: cluster similarity spectrum integration of single-cell genomics data. *Genome Biol.* **21**, 224 (2020).

20. Tran, H. T. N. *et al.* A benchmark of batch-effect correction methods for single-cell RNA sequencing data. *Genome Biol.* **21**, 12 (2020).

21. Zhang, Q. *et al.* Landscape and Dynamics of Single Immune Cells in Hepatocellular Carcinoma. *Cell* **179**, 829-845.e20 (2019).

22. Wei, K. *et al.* Notch signalling drives synovial fibroblast identity and arthritis pathology. *Nature* **582**, 259–264 (2020).

23. Kirita, Y., Wu, H., Uchimura, K., Wilson, P. C. & Humphreys, B. D. Cell profiling of mouse acute kidney injury reveals conserved cellular responses to injury. *Proc. Natl. Acad. Sci. U. S. A.* **117**, 15874–15883 (2020).
24. Sandu, I. *et al.* Landscape of Exhausted Virus-Specific CD8 T Cells in Chronic LCMV Infection. *Cell Rep.* **32**, 108078 (2020).
25. Lähnemann, D. *et al.* Eleven grand challenges in single-cell data science. *Genome Biol.* **21**, 31 (2020).
26. Lotfollahi, M. *et al.* Query to reference single-cell integration with transfer learning. *bioRxiv* (2020).
27. Cao, Z.-J., Wei, L., Lu, S., Yang, D.-C. & Gao, G. Searching large-scale scRNA-seq databases via unbiased cell embedding with Cell BLAST. *Nat. Commun.* **11**, 3458 (2020).
28. Abdelaal, T. *et al.* A comparison of automatic cell identification methods for single-cell RNA sequencing data. *Genome Biol.* **20**, 194 (2019).
29. Zhang, Z. *et al.* SCINA: A Semi-Supervised Subtyping Algorithm of Single Cells and Bulk Samples. *Genes* **10**, (2019).
30. Kiselev, V. Y., Yiu, A. & Hemberg, M. scmap: projection of single-cell RNA-seq data across data sets. *Nat. Methods* **15**, 359–362 (2018).
31. Alquicira-Hernandez, J., Sathe, A., Ji, H. P., Nguyen, Q. & Powell, J. E. scPred: accurate supervised method for cell-type classification from single-cell RNA-seq data. *Genome Biol.* **20**, 264 (2019).
32. Tan, Y. & Cahan, P. SingleCellNet: A Computational Tool to Classify Single Cell RNA-Seq Data Across Platforms and Across Species. *Cell Syst* **9**, 207-213.e2 (2019).
33. Ding, J. *et al.* Systematic comparative analysis of single cell RNA-sequencing methods. *bioRxiv* 632216 (2019) doi:10.1101/632216.
34. Nathan, A. *et al.* Multimodal Profiling of 500,000 Memory T Cells from a Tuberculosis Cohort Identifies Cell State Associations with Demographics, Environment, and Disease. (2020) doi:10.2139/ssrn.3652337.

35. Segerstolpe, Å. *et al.* Single-Cell Transcriptome Profiling of Human Pancreatic Islets in Health and Type 2 Diabetes. *Cell Metab.* **24**, 593–607 (2016).

36. Lawlor, N. *et al.* Single-cell transcriptomes identify human islet cell signatures and reveal cell-type-specific expression changes in type 2 diabetes. *Genome Res.* **27**, 208–222 (2017).

37. Grün, D. *et al.* De Novo Prediction of Stem Cell Identity using Single-Cell Transcriptome Data. *Cell Stem Cell* **19**, 266–277 (2016).

38. Muraro, M. J. *et al.* A Single-Cell Transcriptome Atlas of the Human Pancreas. *Cell Syst* **3**, 385-394.e3 (2016).

39. Baron, M. *et al.* A Single-Cell Transcriptomic Map of the Human and Mouse Pancreas Reveals Inter- and Intra-cell Population Structure. *Cell Syst* **3**, 346-360.e4 (2016).

40. Popescu, D.-M. *et al.* Decoding human fetal liver haematopoiesis. *Nature* **574**, 365–371 (2019).

41. Stoeckius, M. *et al.* Simultaneous epitope and transcriptome measurement in single cells. *Nat. Methods* **14**, 865–868 (2017).

42. Peterson, V. M. *et al.* Multiplexed quantification of proteins and transcripts in single cells. *Nat. Biotechnol.* **35**, 936–939 (2017).

43. Nathan, A. *et al.* Multimodal memory T cell profiling identifies a reduction in a polyfunctional Th17 state associated with tuberculosis progression. *bioRxiv* 2020.04.23.057828 (2020) doi:10.1101/2020.04.23.057828.

44. Luecken, M. D. & Theis, F. J. Current best practices in single-cell RNA-seq analysis: a tutorial. *Mol. Syst. Biol.* **15**, e8746 (2019).

45. Lotfollahi, M., Naghipourfar, M., Theis, F. J. & Alexander Wolf, F. Conditional out-of-sample generation for unpaired data using trVAE. *arXiv [cs.LG]* (2019).

46. Berger, B. & Cho, H. Emerging technologies towards enhancing privacy in genomic data sharing. *Genome Biol.* **20**, 128 (2019).

47. Wang, S., Pisco, A. O., Karkanias, J. & Altman, R. B. Unifying single-cell annotations based on the Cell Ontology. *bioRxiv* 810234 (2019) doi:10.1101/810234.

48. Gayoso, A. *et al.* A Joint Model of RNA Expression and Surface Protein Abundance in Single Cells. *biorxiv* 791947 (2019) doi:10.1101/791947.

49. Baglama, J. & Reichel, L. Augmented Implicitly Restarted Lanczos Bidiagonalization Methods. *SIAM Journal on Scientific Computing* vol. 27 19–42 (2005).

50. Durinck, S., Spellman, P. T., Birney, E. & Huber, W. Mapping identifiers for the integration of genomic datasets with the R/Bioconductor package biomaRt. *Nat. Protoc.* **4**, 1184–1191 (2009).

51. Korsunsky, I., Nathan, A., Millard, N. & Raychaudhuri, S. Presto scales Wilcoxon and auROC analyses to millions of observations. *bioRxiv* 653253 (2019) doi:10.1101/653253.

52. Gu, Z., Eils, R. & Schlesner, M. Complex heatmaps reveal patterns and correlations in multidimensional genomic data. *Bioinformatics* **32**, 2847–2849 (2016).

53. Qi Mao, Li Wang, Tsang, I. W. & Yijun Sun. Principal Graph and Structure Learning Based on Reversed Graph Embedding. *IEEE Trans. Pattern Anal. Mach. Intell.* **39**, 2227–2241 (2017).

54. Leurgans, S. E., Moyeed, R. A. & Silverman, B. W. Canonical Correlation Analysis When the Data are Curves. *J. R. Stat. Soc. Series B Stat. Methodol.* **55**, 725–740 (1993).

55. McInnes, L., Healy, J. & Melville, J. UMAP: Uniform Manifold Approximation and Projection for Dimension Reduction. *arXiv [stat.ML]* (2018).

56. Friedman, J., Hastie, T. & Tibshirani, R. Regularization Paths for Generalized Linear Models via Coordinate Descent. *J. Stat. Softw.* **33**, 1–22 (2010).