Automated Cell Type Annotation with Reference Cluster Mapping

Valerio Galanti[1,2], Lingting Shi[2,3], Elham Azizi[1,2,3,4], Yining Liu[1,2*], Andrew J. Blumberg[1,2,5*]

[1]Department of Computer Science, Columbia University, 500 W 120th St., New York, 10027, New York, USA.
[2]Irving Institute for Cancer Dynamics, Columbia University, Room 601 Schermerhorn Hall, New York, 10027, New York, USA.
[3]Department of Biomedical Engineering, Columbia University, 1210 Amsterdam Ave, New York, 10027, New York, USA.
[4]Data Science Institute, Columbia University, Mudd Building, New York, 10027, New York, USA.
[5]Department of Mathematics, Columbia University, 2990 Broadway Room 509, New York, 10027, New York, USA.

*Corresponding author(s). E-mail(s): yl4536@columbia.edu; ab4808@columbia.edu ;
Contributing authors: vg2551@columbia.edu; ls3456@columbia.edu; ea2690@columbia.edu;

Abstract

Single-cell RNA sequencing has transformed the field of cellular biology by providing unprecedented insights into cellular heterogeneity. However, characterizing scRNA-seq datasets remains a significant challenge. We introduce RefCM, a novel computational method that combines optimal transport and integer programming to enhance the annotation of scRNA clusters using established reference datasets. Our method produces highly accurate cross-technology, cross-tissue, and cross-species mappings while remaining tractable at atlas scale, outperforming existing methods across all these tasks. By providing precise annotations, RefCM can enable the discovery of new cell types, states, and relationships in single-cell transcriptomic data.

Keywords: scRNA-seq, optimal transport, cell type annotation

1 Introduction

Single-cell RNA sequencing (scRNA-seq) has opened new frontiers in biological research by capturing gene expression profiles at individual cell resolution [1–5]. This technology reveals the remarkable diversity within tissues, where populations previously thought to be homogeneous are now known to contain distinct cellular subtypes and states [6, 7]. A fundamental challenge in scRNA-seq analysis is cell type annotation: assigning biologically meaningful identities to cells based on their transcriptional signatures. Robust cell type identification underpins key biological investigations, from decoding developmental programs to elucidating disease mechanisms [8–10].

Traditional approaches to cell type annotation rely heavily on expert analysis of gene expression patterns. The standard pipeline involves clustering cells and identifying marker genes that distinguish each cluster [11]. These markers are detected through statistical tests for differential expression and interpreted using gene set enrichment (GSEA) analysis to match clusters with known cell type signatures [12]. While this approach has enabled numerous biological discoveries, it faces significant scalability challenges as datasets grow to hundreds of thousands of cells [13]. Manual annotation becomes increasingly impractical, requiring substantial time and expertise to resolve ambiguous cases and maintain consistency across studies[14–20]. The variability in marker gene selection and interpretation further complicates reliable annotation at scale [21].

The rapid growth of single-cell reference atlases [22, 23] has enabled a new paradigm for cell type annotation: *reference mapping*. This approach promises to automate and standardize annotation by leveraging existing expert-annotated datasets [24]. Reference mapping algorithms align query and reference data for supervised label transfer, often via a shared embedding space [25, 26]. A number of methods have been developed to tackle this problem: Seurat [27] uses dataset alignment through "anchors" followed by label transfer via weighted voting, scANVI [28] employs probabilistic modeling to learn cell embeddings and identities jointly, and CellTypist [29] applies logistic regression to harmonized data. Other widely used approaches include batch-integration methods such as SCALEX [30] and similarity-based annotation methods such as SingleR [31] and scmap [32]. However, these methods often struggle with biological and technical variations between reference and query data, particularly when annotating cells across different experimental conditions, tissues, or species [21, 33].

While cell-to-cell mapping approaches offer high resolution, they can be computationally intensive and sensitive to technical noise. This motivates a different strategy based on two key biological principles: cell types represent stable transcriptional states [34–37], and relative gene expression patterns are largely preserved across technologies [38–40]. Under these principles, we can transfer reference labels at the cluster level rather than at the level of individual cells, an approach we term *reference cluster mapping*. This coarser-grained strategy promises to be more robust and computationally efficient, yet has received comparatively less attention. Current methods like CIPR [41] and ClustifyR [42] rely primarily on correlation between averaged expression profiles, discarding valuable information about expression heterogeneity within clusters. This oversimplification particularly impacts performance in complex scenarios such as partial matching or hierarchical cell type relationships.

In this work, we present RefCM, a novel algorithm that addresses these challenges through a distinct approach to reference cluster mapping. By employing the Wasserstein metric from optimal transport (OT) theory, RefCM quantifies transcriptomic similarity between cell populations while preserving their internal heterogeneity. We frame annotation transfer as a graph matching problem solved via integer programming, enabling flexible matching constraints that can handle partial correspondences and hierarchical relationships. Our method demonstrates strong performance across diverse experimental scenarios, achieving consistently high accuracy in challenging cross-species comparisons where prior approaches often degrade. Moreover, the algorithm maintains this performance across different technologies and phenotypes while supporting scalable, practical runtimes on large atlases and variations in label resolution. RefCM improves upon existing approaches, providing more reliable automated annotation of single-cell data, particularly for complex comparative studies between species.

2 Results

2.1 The RefCM model

RefCM is an automated cell type annotation algorithm that leverages optimal transport theory to map query clusters to reference cell types. At its core, the algorithm compares transcriptional profiles between unannotated query cells and annotated reference cells to transfer cell type labels based on expression similarities. Formally, RefCM takes as input two gene expression matrices: a query dataset $\mathcal{Q}$ and a reference dataset $\mathcal{R}$. Additionally, it utilizes two mapping functions: φ, which assigns query cells to their cluster labels $\mathcal{C}$, and γ_r, which assigns reference cells to their known cell type annotations $\mathcal{A}_r$ — effectively defining a clustering of $\mathcal{R}$. The goal is to produce a mapping γ_q that assigns each query cell to either a reference cell type in $\mathcal{A}_r$ or marks it as a novel cell population, denoted by θ.

The RefCM workflow consists of three main steps. First, both datasets are projected into a shared embedding space to enable direct comparison. Second, RefCM computes a cost matrix $W \in \mathbb{R}^{|\mathcal{C}|\times|\mathcal{A}_r|}$ between query clusters and reference cell types using the Wasserstein metric. This metric captures the full distribution of gene expression within clusters, with lower values in W indicating lower transport costs, equivalently higher similarity, between populations. The cost matrix naturally represents a bipartite graph matching problem between query clusters and reference cell types, which RefCM solves through integer programming to handle differences in annotation resolution between datasets. The resulting matches provide labels for query clusters, with unmatched clusters designated as novel cell types θ. For a complete mathematical description of the algorithm, see Section 4.1.

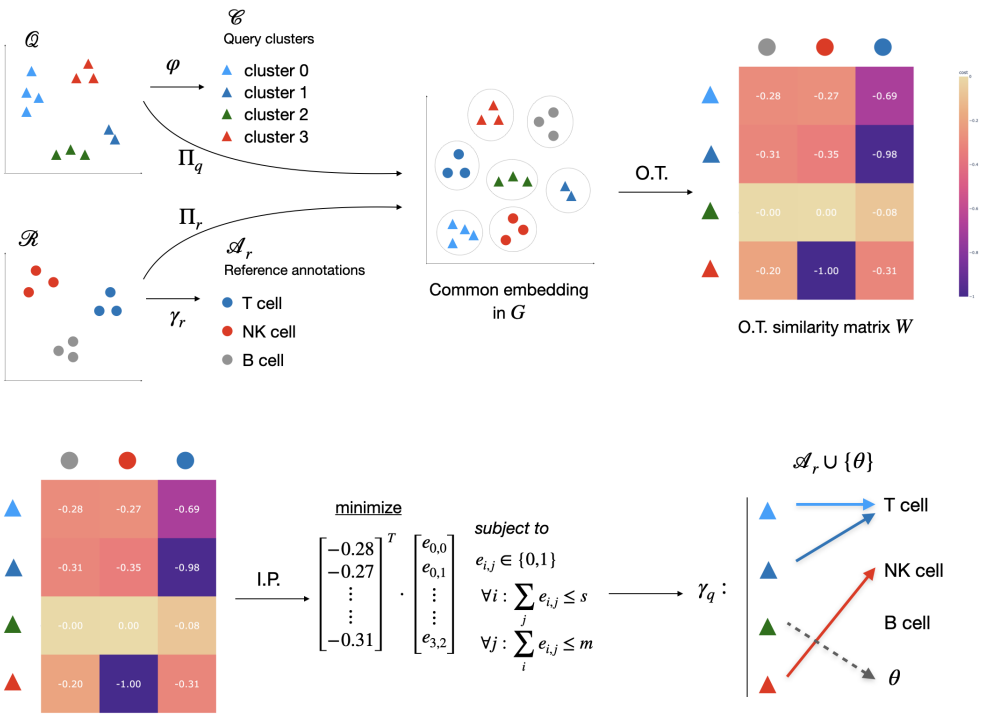

Fig. 1 Overview of RefCM. Query and reference datasets are embedded in a common space (top-middle), where pairwise Wasserstein costs between clusters are computed (top-right). An integer program then determines the best cluster matches under given constraints (bottom-middle), which are used to annotate query clusters with reference labels. Unmatched clusters are labeled as novel (bottom-right).

2.2 Model evaluation

We conducted comprehensive experiments to evaluate RefCM against current state-of-the-art methods: Seurat [27], scANVI [28], CellTypist [29], CIPR [41], SingleR [31], scmap [32], clustifyr [42], SCALEX [30], SATURN [43], and a linear SVM classifier [21]. Our evaluation framework addresses three key types of annotation challenges: biological variation spanning both phenotypic differences and evolutionary distance between species, technical variation across sequencing technologies, and differences in annotation granularity between reference atlases including cell type hierarchies and resolution mismatches. We first evaluate performance on cross-technology, cross-phenotype, and cross-species annotation tasks. We then examine annotation granularity across different reference atlases, and finally assess runtime scaling across increasing dataset sizes.

To assess cross-technology annotation, we leveraged the scIB pancreas benchmark [44], which comprises 9 pancreas datasets collected using different sequencing technologies. We additionally evaluated cross-technology transfer on the PBMC Bench1

dataset [21] and on CellBench [45], a controlled cross-platform benchmark. For cross-phenotype comparison, we examined the impact of aging on cell type profiles using monkey adrenal gland datasets from [46]. We also evaluated cross-individual generalization using the Tabula Muris Senis dataset [47]. Finally, we evaluated cross-species annotation at different evolutionary distances. We began with the Allen Brain Atlas datasets [48, 49] to compare annotations between functionally distinct brain regions: the anterior lateral motor cortex (ALM) and primary visual cortex (VISp) from mouse brain samples, and the middle temporal gyrus (MTG) from human brain tissue. We then extended our analysis to more distant evolutionary relationships using frog and zebrafish embryogenesis datasets [50], where reduced gene homology presents additional challenges. For detailed descriptions of all datasets, see Section 4.6.

To enable fair comparison between methods, we adapted both RefCM and competing approaches to work across different annotation frameworks. Since many competing methods are *reference mapping* methods that operate on individual cells rather than clusters, we augmented them with majority voting (prefixed with `MV-`) to effectively convert them to *reference cluster mapping* methods. For cluster-level evaluations, we used the benchmark-provided oracle partitioning of query cells as an upper bound on reference cluster mapping performance [51], consistent with common annotation workflows in which cell types are first defined at the cluster level before assigning labels. Cell annotation accuracy was computed as closed-set accuracy: the percentage of query cells whose true type is present in the reference that were assigned the correct label, to avoid confounding comparisons by method-specific rejection/novelty detection procedures and sensitive threshold hyperparameters that are not uniformly available across approaches. See Section 4.7 for detailed benchmarking procedures.

2.3 RefCM performs robustly across technical and biological variation

Comprehensive benchmark evaluations show that RefCM consistently outperforms existing methods across diverse annotation challenges (Fig. 2). RefCM attains near-perfect accuracy across cross-technology tasks (scIB pancreas and PBMC Bench1), phenotypically varied monkey adrenal gland samples, and evolutionarily distant brain region comparisons, achieving perfect accuracy in all settings except Tabula Muris Senis and frog–zebrafish embryogenesis (where it remains the top-performing method). RefCM also achieves the best overall performance on Tabula Muris Senis group-k-fold cross-individual prediction, highlighting robust transfer across individuals. RefCM maintains high accuracy even for rare cell types on Tabula Muris Senis, correctly labeling clusters with fewer than 100 cells in the query or reference (Supplementary Fig. 1). Gains are most pronounced in cross-species brain annotations, where competing methods frequently fall below 65% accuracy even when aided by clustering information. These results highlight RefCM's unique capability to overcome the technical and biological barriers that typically hinder accurate cell type annotation across diverse datasets.

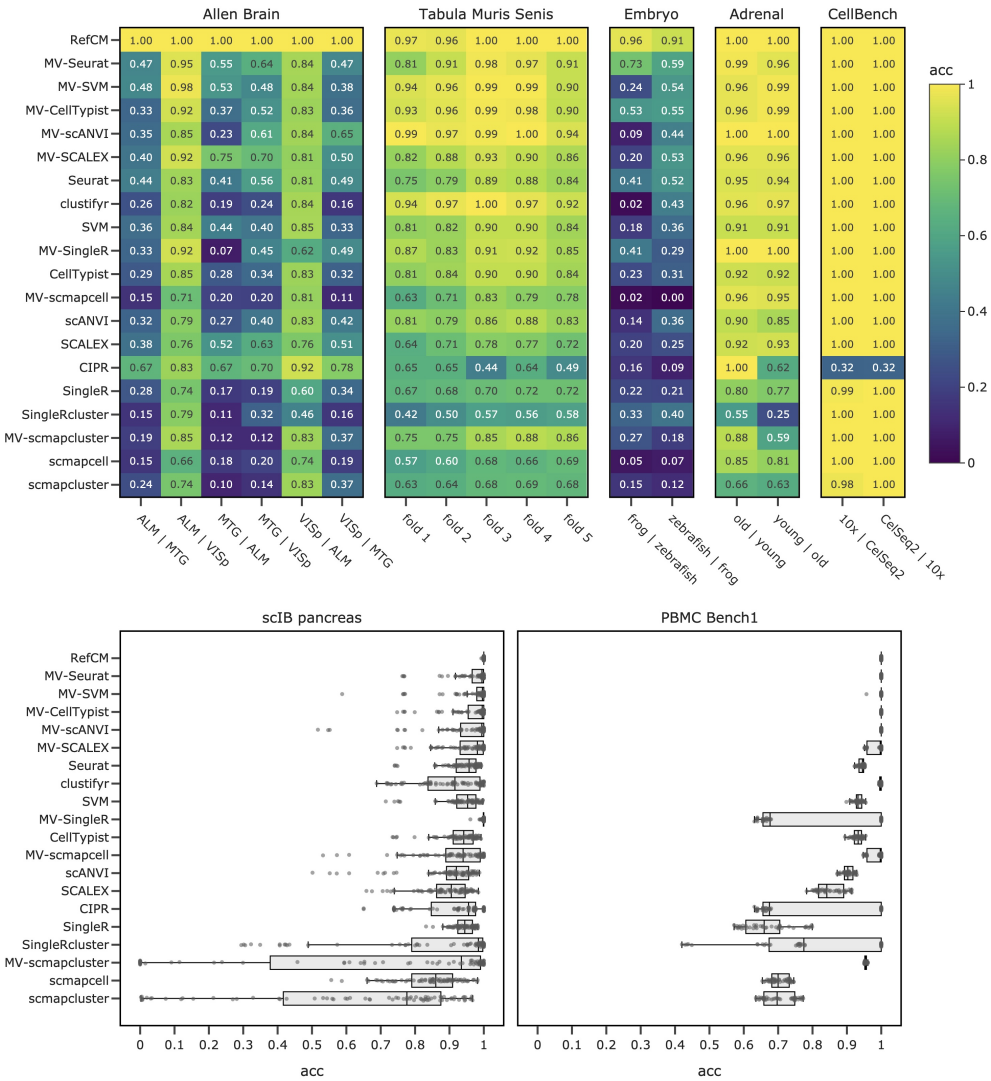

Fig. 2 Benchmark performance of RefCM and baseline methods across diverse annotation scenarios. Methods are ordered by median accuracy across all tasks. Top row: accuracy heatmaps for Allen Brain cross-tissue and cross-species annotations, Tabula Muris Senis (droplet) group-k-fold cross-individual prediction, frog–zebrafish embryogenesis cross-species transfer, monkey adrenal gland cross-age transfer, and CellBench cross-technology evaluation. Bottom row: accuracy distributions for pairwise cross-technology annotations on scIB pancreas and PBMC Bench1 datasets. Throughout, the notation $X \mid Y$ denotes annotation of query dataset X given Y as reference.

2.4 Accurate mapping across distant evolutionary relationships

To demonstrate RefCM's effectiveness in cross-species annotation, we focused on challenging cases with significant evolutionary distance. Fig. 3 visualizes performance differences between methods when annotating human MTG using mouse ALM as

reference. While competing methods show considerable misclassification (red cells), RefCM maintains high accuracy across the diverse cell populations present in these evolutionarily distinct brain regions.

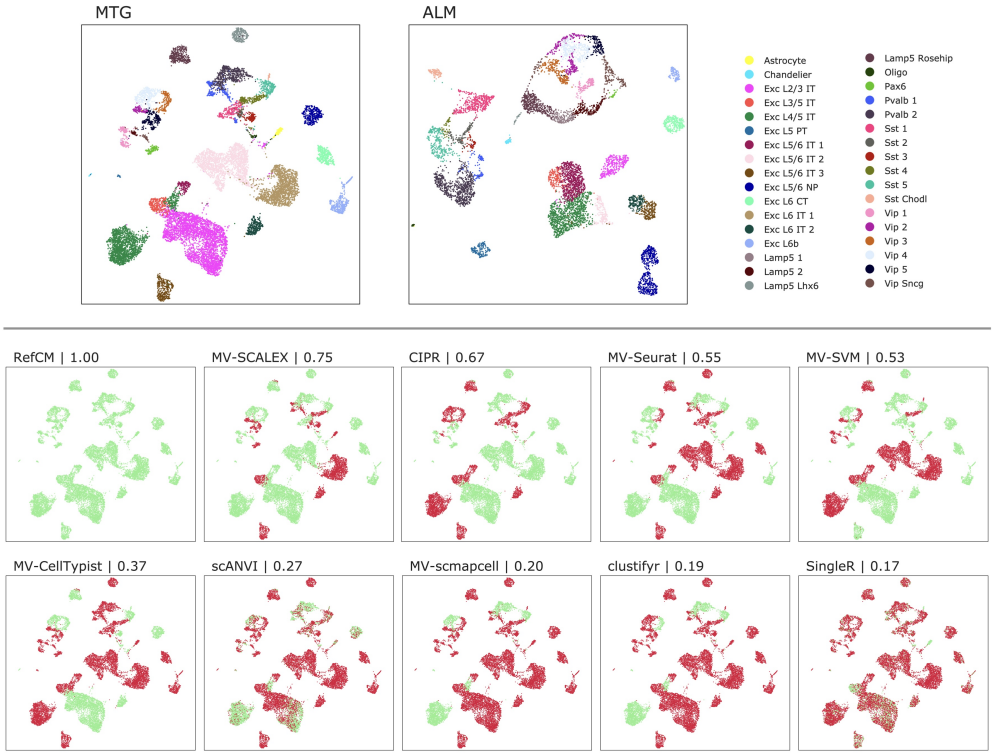

Fig. 3 Cross-species annotation of MTG (human) given ALM (mouse) as reference. UMAP plots (top). Correct (green) and incorrect (red) cell annotations with accuracies for each method (bottom). For each method, we show the best-performing variant on this task (e.g., `SingleR`, `SingleRcluster`, or `MV-SingleR`).

On the evolutionarily distant frog–zebrafish embryogenesis transfer task, RefCM performs on par with SATURN [43] when the latter is augmented with majority voting to leverage clustering information; both methods correctly map 25 out of 28 common cell types. Notably, RefCM's explicit handling of novel cell types enables it to identify 5 out of 14 non-common cell types, providing an explicit treatment of novel populations rather than forcing a best-match assignment. In contrast, several cell-level annotation baselines are more sensitive to distribution shift between reference and query data in this setting, which we observe as increased misclassification and reduced robustness under cross-species transfer.

2.5 Resolving resolution differences and annotation hierarchies

To evaluate RefCM's ability to handle differences in annotation resolution and hierarchical relationships, we leveraged the Allen Brain dataset's multi-level annotation structure. These datasets are particularly well-suited for this analysis as they combine cross-species variation with two distinct levels of annotation granularity: a coarse level with 3 super-types and a fine-grained level with 34 cell types (Fig. 4). By allowing many-to-one and one-to-many mappings (i.e., permitting cluster merging and splitting), we tested RefCM's performance on all six possible dataset combinations, considering both coarse-to-fine and fine-to-coarse annotation scenarios. RefCM successfully recovered the hierarchical relationships in all cases, accurately mapping super-types to their constituent subtypes and vice versa. This ability to transfer labels across mismatched resolutions while maintaining accurate cross-species mapping is not supported by many existing approaches, which typically assume comparable annotation granularity between reference and query datasets.

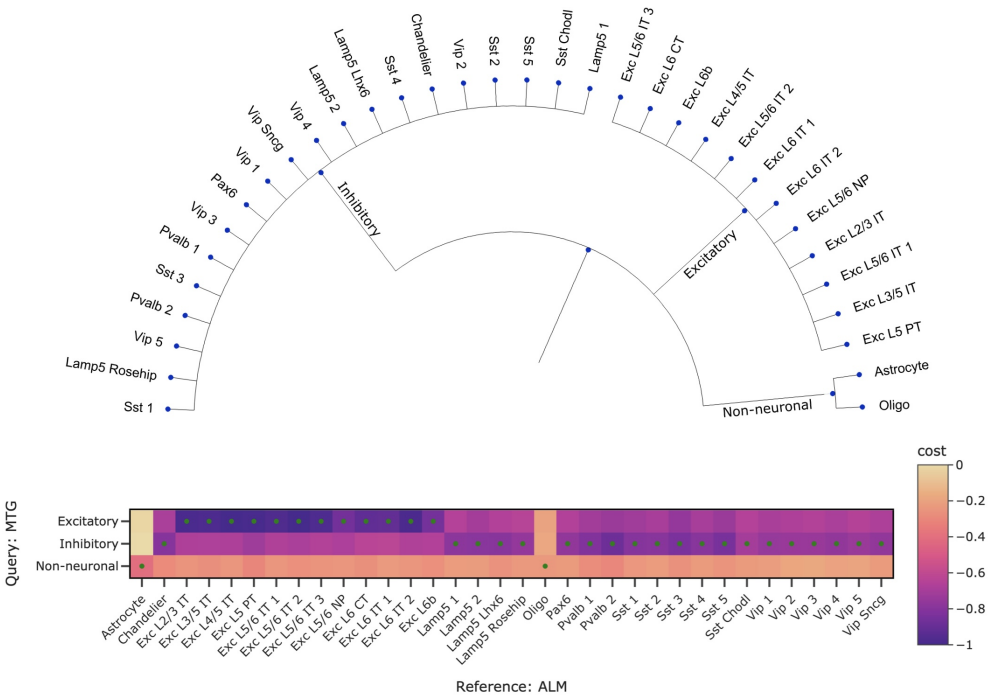

Fig. 4 Top: Allen Brain Atlas cell type hierarchy. Bottom: Example of mapping coarse type-level labels from VISp to the granular level on the ALM dataset. The color gradient illustrates the Wasserstein distances computed between clusters in the OT-step, and dots indicate the final mappings returned by RefCM.

2.6 Runtime scaling and atlas-scale deployment

Beyond accuracy, practical atlas-scale annotation requires methods that remain tractable as the number of cells grows and that support repeated query annotation. Single-cell annotation methods differ substantially in both their compute profile and how costs split between reference setup and query-time prediction. Many approaches are primarily CPU-bound and benefit from multicore parallelism, whereas neural-network methods typically achieve their best throughput with GPU acceleration. Separately, several methods explicitly separate a costly reference training step from lightweight inference (e.g., CellTypist and SVM), enabling a model trained once on a large atlas to be reused across many downstream queries.

We benchmarked wall-time runtimes on Tabula Muris subsamples across $n =$ 5,000–200,000 cells (80/20 reference/query split; 10–100 cell types), reporting mean runtime over 5 runs on an NVIDIA GH200 system (Fig. 5). RefCM scales favorably across all tested sizes and completes end-to-end annotation in 151 seconds at $n = 200{,}000$, comparable to Seurat (146 seconds) and substantially faster than GPU-accelerated baselines even when run with dedicated GPU resources in our benchmark (SCALEX: 3407 seconds; scANVI: 4485 seconds), as well as several widely used methods (CIPR: 1428 seconds; scmapcell: 1173 seconds; SingleR: 2957 seconds). These results indicate that RefCM achieves strong accuracy while remaining practical at atlas scale without relying on GPU acceleration.

The runtime decomposition at $n = 200{,}000$ further clarifies deployment trade-offs. CellTypist and SVM devote nearly all runtime to reference training (904 and 2015 seconds, respectively) but then achieve sub-second query inference (0.57 and 0.34 seconds), which is advantageous when a fixed pretrained classifier can be reused. In contrast, methods dominated by query-time computation (e.g., CIPR with 1417 seconds spent in annotation) incur comparable cost whenever the query changes. RefCM is CPU-bound and parallelizes naturally across query–reference cluster pairs in the OT step; the final integer-program step is negligible at the cell-type counts considered here. While RefCM does not provide a standalone pretrained classifier, it offers fast end-to-end runtimes together with strong performance, making it well suited to iterative analysis settings where references and queries vary across runs.

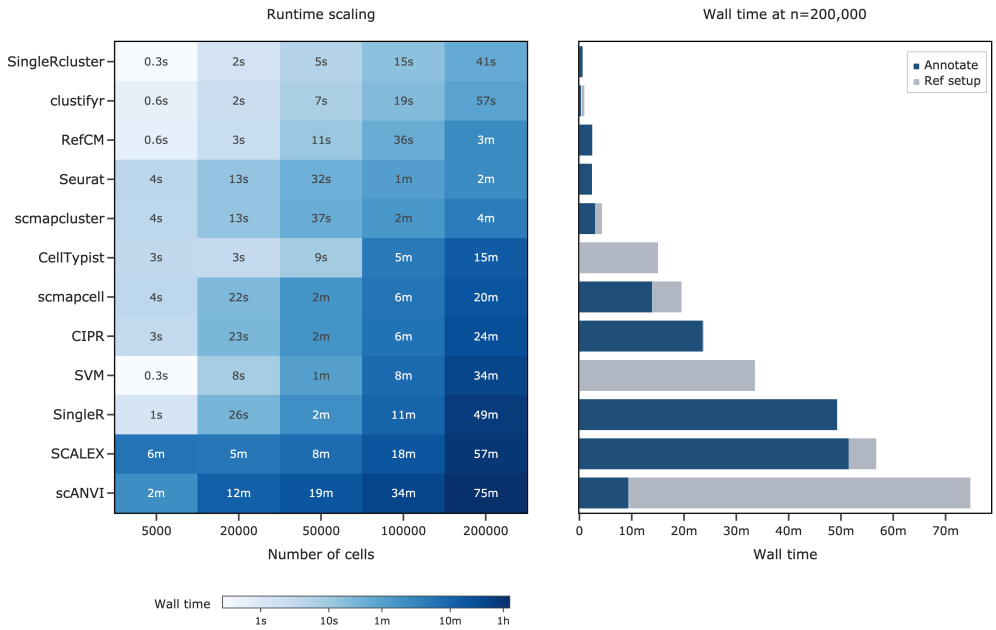

Fig. 5 Wall-time runtimes (mean over 5 runs) on Tabula Muris subsamples with fixed sizes $n \in \{5\,000, 20\,000, 50\,000, 100\,000, 200\,000\}$ (80/20 reference/query split) and corresponding increasing numbers of cell types (10–100). Left: runtime heatmap across methods and dataset sizes (log scale); methods are sorted by mean wall time across sizes. Right: runtime decomposition at $n = 200{,}000$ into reference setup (training where applicable) and query annotation. Experiments were run on an NVIDIA GH200 system.

3 Discussion

RefCM introduces a novel approach to automated cell type annotation by combining optimal transport theory with integer programming. At its core, the method quantifies transcriptomic similarity between reference and query clusters through optimal transport, translating the annotation problem into a flexible graph matching framework. This formulation naturally supports partial correspondences, including novel populations, and transfer across mismatched annotation resolutions, addressing key challenges in single-cell annotation.

We demonstrated RefCM's strong performance across diverse experimental settings, from technical variation across sequencing technologies to biological differences in age and species. The method particularly excels in cross-species annotation, where distribution shifts pose significant challenges for existing approaches. The frog–zebrafish task is especially illustrative: despite limited gene overlap, and the fact that homologous genes can diverge in function across species, RefCM performs remarkably well. SATURN addresses this setting by leveraging protein embeddings, yet RefCM achieves comparable (and slightly better) results using only the intersection of highly variable genes as detailed in the methods section. This suggests that optimal-transport

distances between expression distributions can extract substantial cross-species signal even under imperfect homology, positioning RefCM as a competitive alternative in settings where embedding-based approaches are typically used. These results also suggest the potential for future work to integrate protein embeddings with RefCM for further gains.

In addition to accuracy, our runtime benchmarks indicate that RefCM remains practical at atlas scale. RefCM is CPU-bound and benefits directly from multicore parallelism, since query–reference cluster pairs can be evaluated independently in the OT step, while the final integer-program mapping is negligible compared to the OT computation. In our benchmark at $n = 200{,}000$, RefCM achieves competitive end-to-end runtimes without requiring GPU acceleration, whereas many neural-network baselines effectively require GPU acceleration to remain computationally tractable at this scale. This makes RefCM a practical choice for day-to-day single-cell workflows, where analyses are refined iteratively and runtime and compute costs can be limiting.

There are a number of future directions to explore. The success of RefCM suggests that the notion of cell type is largely captured by the empirical distribution of expression vectors. Nonetheless, a natural question is whether improved representations can provide more refined cell type information, for example by defining the OT cost in learned embedding spaces (e.g., scVI- or SATURN-style embeddings) in addition to gene space. More generally, we might search for a refined notion of cell type that captures subtler biological phenomena than are accessible from expression alone. In another direction, since the effectiveness of RefCM hinges on the accurate clustering of the query dataset, an important next step is to develop clustering procedures that are explicitly informed by the reference. Current approaches typically fall into two categories: unsupervised clustering or supervised classification. While unsupervised clustering can uncover novel populations, its outcomes are sensitive to hyperparameter choices (e.g., the number of clusters or resolution) that are difficult to set in advance. Supervised classification methods can leverage reference information, but are not designed to discover new clusters or populations, and most do not explicitly incorporate the local geometry and structure of the query object they perform inference on. A promising middle ground is a supervised (or joint) clustering approach that can transfer structure from one or more references while retaining the ability to identify novel populations. Finally, RefCM currently relies primarily on cross-dataset distances between clusters. It would be interesting to incorporate within-dataset geometry, for example by using OT to define graphs of cluster–cluster relationships within the query and reference and regularizing the integer program to preserve these structures, in a manner related in spirit to Gromov–Wasserstein formulations.

4 Methods

4.1 Algorithm Overview

1. **Input:** RefCM takes two primary data inputs.
   - A query dataset $\mathcal{Q} = \{q_1, \dots q_{n_q}\} \subset \mathbb{R}^{|G_q|}$ containing gene expression vectors for n_q cells, usually count and log-normalized expression values, where each cell's expression is measured across a set of genes G_q.
   - A reference dataset $\mathcal{R} = \{r_1, \dots r_{n_r}\} \subset \mathbb{R}^{|G_r|}$ containing gene expression vectors for n_r cells, measured across a set of genes G_r.

   Note: The gene labels G_q and G_r may differ, which is particularly common when comparing across species. RefCM can also work with data that has already been embedded in a common space $\mathcal{E}$.

2. **Input:** The reference dataset $\mathcal{R}$ includes cell type labels:
   - A set of cell types $\mathcal{A}_r = \{a_1, \dots a_{m_r}\}$
   - An annotation map $\gamma_r \colon \mathcal{R} \mapsto \mathcal{A}_r$ that assigns each reference cell to one of these types, i.e. $\forall r \in \mathcal{R}, \gamma_r(r) \in \mathcal{A}_r$.

   These annotations effectively partition the reference cells into clusters $\gamma_r^{-1}(a_i)$, where each group contains all cells of a particular type.

3. **Problem:** Our task is now finding the set of cell types $\mathcal{A}_q \subseteq \mathcal{A}_r \cup \{\theta\}$ that our query dataset presents, where θ denotes a novel cell-type — or rather one that is not present in our reference — and a corresponding assignment $\gamma_q \colon \mathcal{Q} \mapsto \mathcal{A}_q$ such that $\gamma_q(q)$ is the correct type annotation for every cell $q \in \mathcal{Q}$.

4. **Problem:** We make the further assumption that the query $\mathcal{Q}$ is equipped with a clustering (i.e., partition) $\mathcal{Q} = \cup_i \mathcal{C}_i$, and we make the simplifying assumption that:

   $$\forall i \ \forall q_u, q_v \in \mathcal{C}_i, \quad \gamma_q(q_u) = \gamma_q(q_v)$$

   That is, γ_q can be regarded as an assignment $\{\mathcal{C}_i\} \to \mathcal{A}_q$. Note that we do not require this assignment to be injective; in some cases multiple clusters map to the same type, which we will call "merging". In the "splitting case", a single cluster may map to multiple reference types; then the cluster-level annotation becomes set-valued, e.g.,

   $$\Gamma_q : \{\mathcal{C}_i\} \to \mathcal{P}(\mathcal{A}_q), \qquad \Gamma_q(\mathcal{C}_i) \subseteq \mathcal{A}_q.$$

5. **Algorithm:** We embed the query and reference data set in a common space $\mathbb{R}^{|\mathcal{G}|}$. For example, a particularly simple embedding uses only the set of genes common to both reference and query, $G_q \cap G_r$.

6. **Algorithm:** Next, we compute the Wasserstein optimal transport cost from mapping the clusters of the query dataset to the clusters from the annotated reference dataset. Specifically, we regard a cluster as specifying the empirical distribution on its constituent cells and compute the Wasserstein loss between these distributions

to compare clusters. This results in a cost matrix W between query and reference clusters in the embedding space that captures the similarity of the underlying population distributions [52].

7. **Algorithm:** The resulting matrix defines a bipartite graph matching problem, which we solve using integer programming with additional biological constraints. The parameters and constraints are described in more detail in Section 4.4.

8. **Output:** Based on the optimal matching, we assign reference cell type labels to query clusters. Query clusters without matches are labeled as novel cell populations (θ).

4.2 Joint Embeddings

To accurately compare genetic expression variability between query and reference cell types, it is essential to first map both datasets to a joint embedding space. Here, we begin by log-normalizing both query and reference datasets and select highly variable genes (HVG) from each to reduce the original gene expression spaces to subspaces that preserve relevant biological information [5, 27, 53]. By performing HVG selection independently on both query and reference datasets, we identify the most informative genes in each context. We then take the union of these high-variable gene sets while ensuring that only common genes between the query and reference datasets are retained. This unified set of high-variable genes forms the basis of the joint embedding space, enabling cellular comparisons in the following OT step. More formally, if $\mathcal{Q}$ and $\mathcal{R}$ are our query and reference datasets, respectively, and G_q and G_r represent their gene sets, our embedding space is then $\mathcal{G} = (\texttt{hvg}(\mathcal{Q}) \cap G_r) \bigcup (\texttt{hvg}(\mathcal{R}) \cap G_q)$. Our embedding maps $\pi_{\mathcal{Q}}$ and $\pi_{\mathcal{R}}$ are then simply the canonical projections of $\mathcal{Q}$ and $\mathcal{R}$ into this common subspace.

4.3 Optimal Transport & Wasserstein distance

Optimal transport (OT) is a mathematical framework for comparing probability distributions by reallocating mass (or probability) from one distribution to another in the most cost-effective manner, where the cost is defined in terms of a distance metric between points in the distributions. Unlike simpler metrics that compare summary statistics (e.g., means or variances), OT considers the full shape of distributions, making it particularly valuable for comparing heterogeneous cell populations.

Consider two discrete distributions: a source distribution μ with points $\{x_i\}$ and a target distribution ν with points $\{y_j\}$. The goal of optimal transport is to find a transport plan π that reallocates the mass from μ to ν while minimizing the total transportation cost:

$$\min_{\pi} \sum_{i,j} c(x_i, y_j)\pi_{i,j}$$

subject to the constraints:

$$\sum_j \pi_{i,j} = \mu_i \text{ for all } i$$
$$\sum_i \pi_{i,j} = \nu_j \text{ for all } j$$

where $\pi_{i,j}$ represents the amount of mass transported from x_i to y_j, μ_i and ν_j are the masses at points x_i and y_j, and $c(x_i, y_j)$ is the ground-cost (transport cost) of moving a unit of mass between these points.

The minimal transportation cost is known as the Wasserstein distance (or Earth Mover's distance), which quantifies the similarity between distributions. Intuitively, this can be understood as the minimum effort required to reshape one distribution into another.

In the RefCM algorithm, we apply this concept to compare cell type distributions between query and reference datasets (Figure 6). We treat each cluster as a discrete uniform distribution over its constituent cells, with the transportation cost between cells defined by a ground cost in the shared gene expression space (typically Euclidean distance or a cost derived from the inner product between embeddings). The Wasserstein distance quantifies the cost of transforming the distribution of gene expressions in the query cluster into the distribution in the reference dataset's cluster, capturing the structural differences between them (with lower cost for more similar populations). This cost metric forms the basis for further computation to optimally map query and reference cell types between datasets.

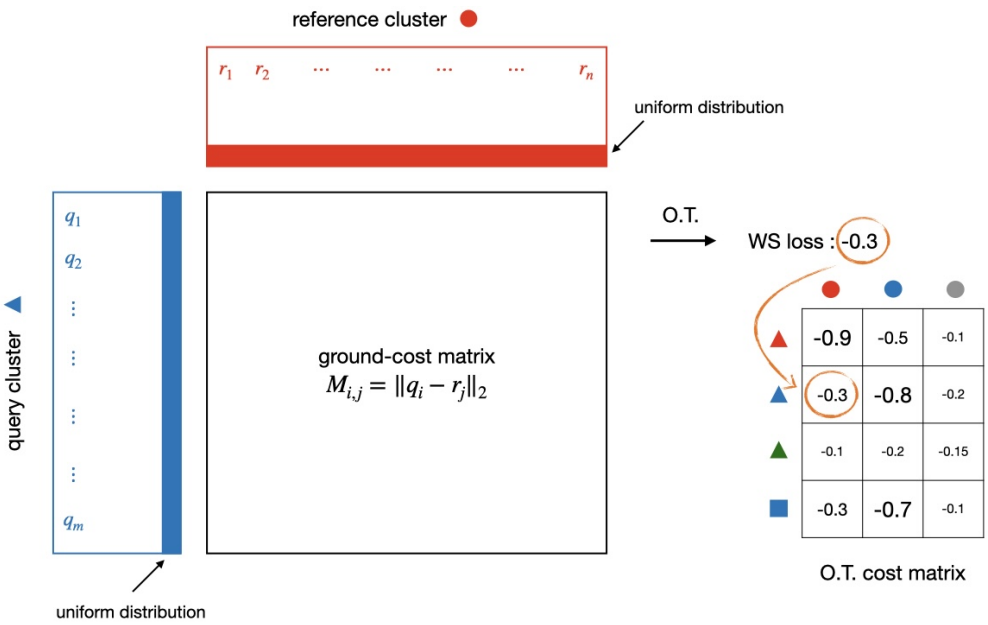

Fig. 6 Optimal Transport between query and reference clusters.

To construct the cost matrix used in the next step, we compute the pairwise Wasserstein distance between each query cluster $\mathcal{C}_i$ and reference cell type $\gamma_r^{-1}(a_j)$. Let $c(\cdot,\cdot)$ denote the ground-cost function in the shared embedding space (e.g. Euclidean distance). We define the unnormalized Wasserstein cost matrix

$$\texttt{WS}^*(i,j) = W\big(\mathcal{C}_i, \gamma_r^{-1}(a_j); c\big),$$

where $W(\cdot,\cdot;c)$ denotes the Wasserstein (Earth Mover's) distance between the two clusters under the ground cost c, with uniform weights over cells in each cluster. We then rescale these values to the interval $[-1, 0]$ via

$$\texttt{WS}(i,j) = \frac{\texttt{WS}^*(i,j) - \max S}{\max S - \min S},$$

where $S = \{\texttt{WS}^*(i,j) \mid 1 \le i \le m_q,\ 1 \le j \le m_r\}$. The resulting matrix WS is therefore a normalized Wasserstein cost matrix, with lower values indicating lower transport cost (i.e. greater similarity) between populations.

4.4 Integer programming for relaxed graph matching

To determine the optimal cluster-to-cluster mappings between query and reference datasets, we devised an integer program based on the Wasserstein cost (WS) matrix obtained in the previous step. This cost matrix represents a complete bipartite graph,

where clusters (or cell types) from the query and reference datasets form the graph's nodes, and the edge weights correspond to the WS cost values. The integer programming framework enables us to identify the set of edges that minimizes the total mapping cost while adhering to a set of prescribed constraints. This approach effectively disentangles the optimal set of cluster mappings, ensuring accurate and efficient alignment of cell types between the query and reference datasets. We now describe the matching constraints, which can be adjusted depending on the application.

Formally, we define $G(V, E)$ as a fully connected bipartite graph where edges $e_{i,j}$ connect query cluster i to reference cell type j, with edge weight:

$$w_{i,j} = \texttt{WS}(i, j)$$

For each edge in the graph, we introduce a binary decision variable $x_{i,j}$ indicating whether that edge is included in the optimal mapping. Our objective is to minimize the sum of costs over the chosen edges

$$\min \sum_{i,j} w_{i,j} x_{i,j}$$

subject to the constraints:

$$\sum_{j} x_{i,j} \leq s, \quad \sum_{i} x_{i,j} \leq m, \quad x_{i,j} \in \{0, 1\}$$

Due to differences in clustering scales, granularities, or cell distributions, it is often necessary to match multiple clusters from the query set to a single reference cluster, a phenomenon we term "cluster merging." Conversely, a single cluster from the query set may correspond to multiple reference clusters, which we refer to as "cluster splitting." This consideration is further motivated by the hierarchical structure of cell types, where we may need to match finer-grained clusters to a broader parent type or, alternatively, more coarse clusters to specific reference subtypes.

The first constraint ensures that each cluster in the query set can be connected to up to $s \in \mathbb{N}$ clusters in the reference dataset, allowing for cluster splitting into as many as s reference clusters. Conversely, the second constraint guarantees that each cluster in the reference set can be mapped to up to $m \in \mathbb{N}$ clusters in the query set, permitting the merging of up to m query clusters into a single reference cluster. The third constraint restricts the linear program decision variables to binary values, ensuring the optimal decision variable outcomes align with the intended matching framework.

Finally, because the reference dataset may not encompass all cell types present in the query dataset, it is essential to account for non-common cell types and facilitate cell type discovery. To address this, we establish a threshold for identifying novel cell types by discarding high-cost query–reference correspondences. Specifically, we set all costs $w_{i,j}$ whose normalized value lies in the worst p fraction of the global cost distribution, i.e., above the $(1 - p)$-quantile τ_p of the normalized cost matrix, to infinity. We then

threshold edges as

$$w_{i,j} = \begin{cases} \texttt{WS}(i,j) & \text{if } \texttt{WS}(i,j) < \tau_p \\ +\infty & \text{otherwise.} \end{cases}$$

This ensures that the integer programming solver will never select these edges. By constraining all remaining costs to be between -1 and 0, the solver favors returning a set of edges that minimizes the overall cost while adhering to the constraints listed above. This approach effectively detects and handles non-common cell types, enabling accurate cell type discovery.

4.5 Implementation

RefCM is implemented in Python. We use POT [54] to compute Wasserstein distances and PuLP [55] to formulate the integer programs, solved with the GNU Linear Programming Kit (GLPK) [56]. Through POT, RefCM supports alternative OT formulations (e.g., Sinkhorn-regularized, unbalanced, and Gromov–Wasserstein OT), with EMD used by default. The optimal transport step dominates computation and is parallelized across query–reference cluster pairs using a `ThreadPoolExecutor`. The default EMD solve runs in POT's compiled backend and releases the Python GIL during the core computation, enabling multicore parallelism without duplicating large in-memory objects. The subsequent integer-program step is negligible compared to OT. We report end-to-end runtime scaling and a breakdown of reference setup versus query annotation in Fig. 5. The RefCM package is available on PyPI, with source code and documentation on GitHub.

4.6 Datasets

Across all experiments, we use the cell-type annotations distributed with each dataset/benchmark release as ground truth, without re-clustering or otherwise re-annotating cells. When a dataset provides a predefined cross-dataset label space, we evaluate in that label space (e.g., Allen Brain at the 3- and 34-population levels [48] [49]; frog–zebrafish using the SATURN type mapping [43]). For Tabula Muris Senis, we evaluate cross-individual generalization using group-k-fold splits by mouse identity to prevent individual-level leakage and mitigate instability from highly variable pairwise overlaps in cell-type composition across mice.

Collection	Dataset	# cells	# genes	# cell types
Tabula Muris Senis [47]	Droplet	245,389	20,138	123
	FACS	110,824	22,966	120
scIB Pancreas [44]	celseq	1,004	19,093	13
	celseq2	2,285	19,093	13
	fluidigmc1	638	19,093	13
	indrop1	1,937	19,093	14
	indrop2	1,724	19,093	14
	indrop3	3,605	19,093	14
	indrop4	1,303	19,093	14
	smarter	1,492	19,093	4
	smartseq2	2,394	19,093	13
PbmcBench pbmc1 [38] Downloaded from [21]	10Xv2	23,154	22,280	9
	10Xv3	19,690	21,905	8
	CEL-Seq2	19,754	20,041	7
	Drop-Seq	23,154	19,922	9
	inDrop	21,832	17,159	7
	Seq-Well	18,966	21,059	7
	Smart-Seq2	18,886	22,617	6
CellBench [57] Downloaded from [21]	10X	3,803	10,217	5
	CEL-Seq2	570	10,217	5
Allen-Brain [48] [49] Downloaded from [21]	ALM	8,128	16,024	3 & 34
	MTG	14,055	16,024	3 & 34
	VISp	12,552	16,024	3 & 34
Monkey Adrenal Gland [46]	young	25,461	19,093	12
	old	33,869	19,093	12
Frog-Zebrafish [50] Pre-processed as in [43]	frog	96,935	24,956	36
	zebrafish	63,371	30,032	42

Table 1 Summary of datasets used.

4.7 Performance benchmarking

We compared RefCM to widely used cell type annotation baselines and followed the default or recommended workflows in the corresponding official implementations. Unless otherwise noted, we report closed-set accuracy and disable method-specific rejection/novelty detection options to ensure comparability across approaches. We provided each method with its expected input representation: log-normalized expression

for CellTypist [29], Seurat [27], CIPR [41], SingleR [31], scmap [32], and clustifyr [42], and raw counts for scANVI [28] and SCALEX [30].

All runtime experiments were run on a Lambda NVIDIA GH200 (Grace Hopper) instance, representative of the heterogeneous CPU+GPU nodes increasingly common in modern academic and institutional HPC environments. The system pairs a 72-core NVIDIA Grace CPU with 480 GB of LPDDR5X system memory (ECC) and an NVIDIA H100 (Hopper) Tensor Core GPU with 96 GB HBM3, connected by a cache-coherent NVLink-C2C interconnect (up to 900 GB/s). For R-based baselines, we evaluated common parallel execution configurations (including `future()`-based and `BiocParallel`-based backends where supported) and report runtimes for the fastest supported configuration for each method. For Python-based baselines, we likewise enabled method-supported parallelism (e.g., `n_jobs`/`n_workers` where exposed) and used the fastest supported configuration where applicable. Finally, we do not report memory usage: observed memory footprints depend strongly on the parallelization strategy (e.g., process-based workers vs. threads, object copying vs. shared memory, and garbage collection behavior), and for GPU-accelerated methods additionally on host–device staging and caching, making fair, comparable cross-method accounting difficult.

4.7.1 RefCM

Across all benchmarks, we ran RefCM with fixed default parameters: Euclidean distance as the ground cost, discovery threshold 0.0, the Earth Mover's Distance solver (`emd`) from POT, and the HVG embedding (Section 4.2). We set $s = m = 1$ for standard benchmarks to enforce one-to-one mappings and keep accuracy well-defined under splitting/merging, and set $s = m = -1$ only for hierarchy recovery experiments (Section 2.5). For HVG selection, we used the union of the top-1200 HVGs selected independently in the query and reference, yielding $\sim$2000 joint features in practice depending on overlap, comparable to the 2000-HVG convention used by baseline methods; our ablation (Supplementary Fig. 2) confirms that performance is stable across the 500–3000 range.

4.7.2 CIPR

Following the recommended workflow and documentation, we used Seurat preprocessing to obtain `log1p`-normalized expression and computed query cluster marker statistics with `FindAllMarkers`. We provided CIPR with this differential-expression table and used the `logfc_dot_product` mode. For the reference, we constructed a custom reference profile by averaging expression within each reference label (`AverageExpression`) and supplied this as `custom_reference`. Each query cluster was assigned to the reference label with the highest CIPR identity score.

4.7.3 clustifyr

We ran clustifyr in its standard correlation-based cluster annotation mode. We constructed a reference profile matrix by averaging log-normalized expression within each reference label (`average_clusters`) and computed query cluster profiles analogously

from the provided query clustering. We then used `clustify` to assign each query cluster to the reference label with the highest correlation (Spearman correlation by default).

4.7.4 SingleR

We used SingleR following the Bioconductor vignette. We constructed `SingleCellExperiment` objects for reference and query, computed log-normalized expression values, and ran SingleR using the benchmark-provided reference labels. We evaluated both the default per-cell mode and the cluster-aware mode via the `clusters=` option. In figures, we refer to the per-cell method as `SingleR` and the cluster-aware variant as `SingleRcluster`. Since `SingleRcluster` is not equivalent to majority-voting per-cell predictions, we also report a majority-vote aggregation of `SingleR` cell-level predictions.

4.7.5 scmap

We ran scmap following the Bioconductor workflow. We used scmap's feature selection (`selectFeatures`; 500 genes) on the reference and built either a cell-level index (`indexCell`) or a cluster-centroid index over reference labels (`indexCluster`). We then annotated each query cell with `scmapCell` or `scmapCluster`, respectively. For cluster-level evaluation, we aggregated both per-cell prediction methods by majority vote within each query cluster. We set scmap's rejection threshold to 0 to avoid conflating unassigned predictions with misclassifications under closed-set accuracy. In figures, we refer to these as `scmapCell` and `scmapCluster`, respectively.

4.7.6 Seurat

We ran Seurat label transfer using the standard anchor-based workflow. We normalized and `log1p`-transformed both datasets, selected variable features, computed anchors with `FindTransferAnchors` (`dims=1:30`), and transferred labels to the query using `TransferData` with default settings.

4.7.7 CellTypist

Following the official documentation, we provided CellTypist with library-size normalized expression (scaled to 10,000 counts per cell) followed by `log1p`. We trained CellTypist on the reference labels using the reference HVGs (2000 genes). At prediction time, we restricted to the subset of these HVGs present in the query; if this gene set differed from the one used for training, we retrained CellTypist on the reference using the restricted gene set before annotating the query.

4.7.8 scANVI

We ran scANVI (scvi-tools) [28] on raw UMI counts, restricting both reference and query to the top-2000 HVGs. We first trained a two-layer scVI model [58] on the reference (dropout 0.2) and then initialized scANVI from this model. We fine-tuned scANVI

on the reference and subsequently adapted it using the query as in the standard scArches workflow, and used the resulting model to predict query cell types.

4.7.9 SCALEX

We ran SCALEX following the official documentation. We provided raw count matrices and used SCALEX's built-in preprocessing, which performs library-size normalization, `log1p` transformation, HVG selection (2000 genes by default), and scaling prior to model fitting. We trained SCALEX on the reference and projected the query into the learned latent space using the provided projection workflow. To avoid dropping cells used for evaluation, we disabled SCALEX's default query-side QC filtering (cell filtering by `min_features`). We then applied SCALEX's label-transfer utility on the latent representations to obtain query predictions.

4.7.10 SVM

We trained a linear SVM classifier (scikit-learn `LinearSVC`) [59] on the reference using the benchmark-provided labels and restricting to the reference HVGs (2000 genes). At prediction time, we restricted to the intersection of genes between reference and query; if the intersection differed from the training gene set, we retrained `LinearSVC` on the reference using the intersected genes before predicting query labels.

Declarations

Some journals require declarations to be submitted in a standardized format. Please check the Instructions for Authors of the journal to which you are submitting to see if you need to complete this section. If yes, your manuscript must contain the following sections under the heading 'Declarations':

- Funding: N/A
- Conflict of interest/Competing interests (check journal-specific guidelines for which heading to use): N/A
- Ethics approval and consent to participate: N/A
- Consent for publication: N/A
- Data availability: All data is available on our GitHub repository
- Materials availability: N/A
- Code availability: All code is available on our GitHub repository
- Author contribution: N/A

If any of the sections are not relevant to your manuscript, please include the heading and write 'Not applicable' for that section.

References

[1] Tang, F. *et al.* mRNA-Seq whole-transcriptome analysis of a single cell. *Nature Methods* **6**, 377–382 (2009).

[2] Wagner, D. E. *et al.* Single-cell mapping of gene expression landscapes and lineage in the zebrafish embryo. *Science* **360**, 981–987 (2018).

[3] Kolodziejczyk, A. A., Kim, J. K., Svensson, V., Marioni, J. C. & Teichmann, S. A. The technology and biology of single-cell RNA sequencing. *Molecular cell* **58**, 610–620 (2015).

[4] Cao, J. *et al.* Comprehensive single-cell transcriptional profiling of a multicellular organism. *Science* **357**, 661–667 (2017).

[5] Zheng, G. X. *et al.* Massively parallel digital transcriptional profiling of single cells. *Nature communications* **8**, 14049 (2017).

[6] Villani, A.-C. *et al.* Single-cell RNA-seq reveals new types of human blood dendritic cells, monocytes, and progenitors. *Science* **356**, eaah4573 (2017).

[7] La Manno, G. *et al.* Molecular architecture of the developing mouse brain. *Nature* **596**, 92–96 (2021).

[8] Aizarani, N. *et al.* A human liver cell atlas reveals heterogeneity and epithelial progenitors. *Nature* **572**, 199–204 (2019).

[9] Luecken, M. D. & Theis, F. J. Current best practices in single-cell rna-seq analysis: a tutorial. *Molecular systems biology* **15**, e8746 (2019).

[10] Rosenberg, A. B. *et al.* Single-cell profiling of the developing mouse brain and spinal cord with split-pool barcoding. *Science* **360**, 176–182 (2018).

[11] Amezquita, R. A. *et al.* Orchestrating single-cell analysis with Bioconductor. *Nature methods* **17**, 137–145 (2020).

[12] Subramanian, A. *et al.* Gene set enrichment analysis: a knowledge-based approach for interpreting genome-wide expression profiles. *Proceedings of the National Academy of Sciences* **102**, 15545–15550 (2005).

[13] Svensson, V., Vento-Tormo, R. & Teichmann, S. A. Exponential scaling of single-cell RNA-seq in the past decade. *Nature Protocols* **13**, 599–604 (2018).

[14] Lähnemann, D. *et al.* Eleven grand challenges in single-cell data science. *Genome Biology* **21**, 31 (2020).

[15] Wu, D. & Smyth, G. K. Camera: a competitive gene set test accounting for inter-gene correlation. *Nucleic acids research* **40**, e133–e133 (2012).

[16] Kim, S.-Y. & Volsky, D. J. PAGE: parametric analysis of gene set enrichment. *BMC bioinformatics* **6**, 1–12 (2005).

[17] Alhamdoosh, M. *et al.* Combining multiple tools outperforms individual methods in gene set enrichment analyses. *Bioinformatics* **33**, 414–424 (2017).

[18] Ma, Y. *et al.* Integrative differential expression and gene set enrichment analysis using summary statistics for scRNA-seq studies. *Nature communications* **11**, 1585 (2020).

[19] Geistlinger, L., Csaba, G. & Zimmer, R. Bioconductor's EnrichmentBrowser: seamless navigation through combined results of set-& network-based enrichment analysis. *BMC bioinformatics* **17**, 1–11 (2016).

[20] Huang, D. W. *et al.* DAVID Bioinformatics Resources: expanded annotation database and novel algorithms to better extract biology from large gene lists. *Nucleic acids research* **35**, W169–W175 (2007).

[21] Abdelaal, T. *et al.* A comparison of automatic cell identification methods for single-cell RNA sequencing data. *Genome biology* **20**, 1–19 (2019).

[22] Regev, A. *et al.* The human cell atlas. *elife* **6**, e27041 (2017).

[23] 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).

[24] Pasquini, G., Arias, J. E. R., Schäfer, P. & Busskamp, V. Automated methods for cell type annotation on scRNA-seq data. *Computational and Structural Biotechnology Journal* **19**, 961–969 (2021).

[25] Lotfollahi, M., Hao, Y., Theis, F. J. & Satija, R. The future of rapid and automated single-cell data analysis using reference mapping. *Cell* **187**, 2343–2358 (2024).

[26] Lotfollahi, M. *et al.* Mapping single-cell data to reference atlases by transfer learning. *Nature biotechnology* **40**, 121–130 (2022).

[27] Stuart, T. *et al.* Comprehensive integration of single-cell data. *cell* **177**, 1888–1902 (2019).

[28] Xu, C. *et al.* Probabilistic harmonization and annotation of single-cell transcriptomics data with deep generative models. *Molecular systems biology* **17**, e9620 (2021).

[29] Domínguez Conde, C. *et al.* Cross-tissue immune cell analysis reveals tissue-specific features in humans. *Science* **376**, eabl5197 (2022).

[30] Xiong, L. *et al.* Online single-cell data integration through projecting heterogeneous datasets into a common cell-embedding space. *Nature Communications* **13**, 6118 (2022). URL https://doi.org/10.1038/s41467-022-33758-z.

[31] Aran, D. *et al.* Reference-based analysis of lung single-cell sequencing reveals a transitional profibrotic macrophage. *Nature Immunology* **20**, 163–172 (2019). URL https://doi.org/10.1038/s41590-018-0276-y.

[32] Kiselev, V. Y., Yiu, A. & Hemberg, M. scmap: projection of single-cell rna-seq data across data sets. *Nature Methods* **15**, 359–362 (2018). URL https://doi.org/10.1038/nmeth.4644.

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

[34] Zeng, H. What is a cell type and how to define it? *Cell* **185**, 2739–2755 (2022).

[35] Arendt, D. *et al.* The origin and evolution of cell types. *Nature Reviews Genetics* **17**, 744–757 (2016).

[36] Regier, J. C. *et al.* Toward reconstructing the evolution of advanced moths and butterflies (lepidoptera: Ditrysia): an initial molecular study. *BMC Evolutionary Biology* **9**, 280 (2009).

[37] Morris, S. A. *et al.* Dissecting engineered cell types and enhancing cell fate conversion via cellnet. *Cell* **158**, 889–902 (2014).

[38] Ding, J. *et al.* Systematic comparison of single-cell and single-nucleus RNA-sequencing methods. *Nature Biotechnology* **38**, 737–746 (2020).

[39] Mereu, E. *et al.* Benchmarking single-cell RNA-sequencing protocols for cell atlas projects. *Nature Biotechnology* **38**, 747–755 (2020).

[40] Chen, H. *et al.* Assessment of computational methods for the analysis of single-cell ATAC-seq data. *Genome Biology* **20**, 241 (2019).

[41] Ekiz, H. A., Conley, C. J., Stephens, W. Z. & O'Connell, R. M. CIPR: a web-based r/shiny app and r package to annotate cell clusters in single cell RNA sequencing experiments. *BMC bioinformatics* **21**, 1–15 (2020).

[42] Fu, R. *et al.* clustifyr: an R package for automated single-cell RNA sequencing cluster classification. *F1000Research* **9** (2020).

[43] Rosen, Y. *et al.* Toward universal cell embeddings: integrating single-cell RNA-seq datasets across species with SATURN. *Nature Methods* (2024). URL https://doi.org/10.1038/s41592-024-02191-z.

[44] Luecken, M. D. *et al.* Benchmarking atlas-level data integration in single-cell genomics. *Nature methods* **19**, 41–50 (2022).

[45] Su, S. *et al.* Cellbench: R/bioconductor software for comparing single-cell rna-seq analysis methods. *Bioinformatics* **36**, 2288–2290 (2020).

[46] Wang, Q. *et al.* Aging induces region-specific dysregulation of hormone synthesis in the primate adrenal gland. *Nature Aging* **4**, 396–413 (2024).

[47] Almanzar, N. *et al.* A single-cell transcriptomic atlas characterizes ageing tissues in the mouse. *Nature* **583**, 590–595 (2020). URL https://doi.org/10.1038/s41586-020-2496-1.

[48] Tasic, B. *et al.* Shared and distinct transcriptomic cell types across neocortical areas. *Nature* **563**, 72–78 (2018).

[49] Hodge, R. D. *et al.* Conserved cell types with divergent features in human versus mouse cortex. *Nature* **573**, 61–68 (2019).

[50] Briggs, J. A. *et al.* The dynamics of gene expression in vertebrate embryogenesis at single-cell resolution. *Science* **360**, eaar5780 (2018).

[51] Diaz-Mejia, J. J. *et al.* Evaluation of methods to assign cell type labels to cell clusters from single-cell RNA-sequencing data. *F1000Res.* **8**, 296 (2019).

[52] Weed, J. & Bach, F. Sharp asymptotic and finite-sample rates of convergence of empirical measures in Wasserstein distance (2019).

[53] Satija, R., Farrell, J. A., Gennert, D., Schier, A. F. & Regev, A. Spatial reconstruction of single-cell gene expression data. *Nature Biotechnology* **33**, 495–502 (2015). URL https://doi.org/10.1038/nbt.3192.

[54] Flamary, R. *et al.* Pot: Python optimal transport. *Journal of Machine Learning Research* **22**, 1–8 (2021). URL http://jmlr.org/papers/v22/20-451.html.

[55] Iain Dunning, M. O., Stuart Mitchell. PuLP: A Linear Programming Toolkit for Python. *Optimization Online* (2011). URL https://optimization-online.org/?p=11731.

[56] GLPK (GNU Linear Programming Kit). URL https://www.gnu.org/software/glpk/glpk.html.

[57] Tian, L. *et al.* Benchmarking single cell rna-sequencing analysis pipelines using mixture control experiments. *Nature Methods* **16**, 479–487 (2019).

[58] Gayoso, A. *et al.* A Python library for probabilistic analysis of single-cell omics data. *Nature biotechnology* **40**, 163–166 (2022).

[59] Pedregosa, F. *et al.* Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research* **12**, 2825–2830 (2011).