Published online 12 July 2023

NAR Genomics and Bioinformatics, 2023, Vol. 5, No. 3 **1**
https://doi.org/10.1093/nargab/lqad069

Transmorph: a unifying computational framework for modular single-cell RNA-seq data integration

Aziz Fouché [1,2,3,4,*], Loïc Chadoutaud[1,2,3], Olivier Delattre[5] and Andrei Zinovyev [1,2,3,6,*]

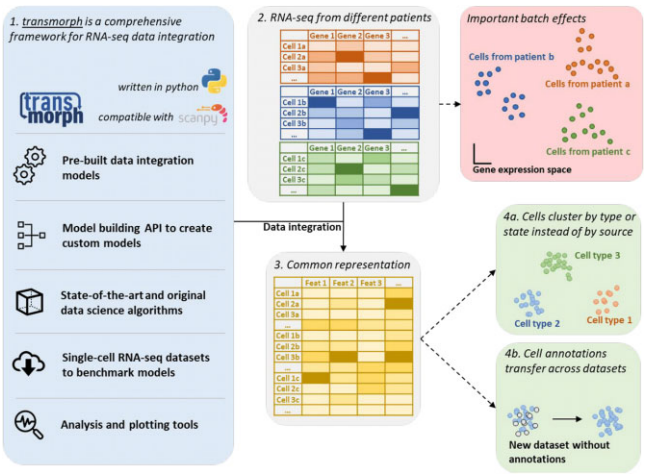
[1]Institut Curie, PSL Research University, 75005 Paris, France, [2]INSERM, 75005 Paris, France, [3]MINES ParisTech, PSL Research University, CBIO-Centre for Computational Biology, 75005 Paris, France, [4]Ecole Normale Supérieure Paris-Saclay, 91190 Gif-sur-Yvette, France, [5]INSERM U830, Equipe Labellisée LNCC, SIREDO Oncology Centre, Institut Curie, 75005 Paris, France and [6]In silico R&D, Evotec, 31400 Toulouse, France

Received January 18, 2023; Revised June 02, 2023; Editorial Decision June 20, 2023; Accepted July 10, 2023

ABSTRACT

Data integration of single-cell RNA-seq (scRNA-seq) data describes the task of embedding datasets gathered from different sources or experiments into a common representation so that cells with similar types or states are embedded close to one another independently from their dataset of origin. Data integration is a crucial step in most scRNA-seq data analysis pipelines involving multiple batches. It improves data visualization, batch effect reduction, clustering, label transfer, and cell type inference. Many data integration tools have been proposed during the last decade, but a surge in the number of these methods has made it difficult to pick one for a given use case. Furthermore, these tools are provided as rigid pieces of software, making it hard to adapt them to various specific scenarios. In order to address both of these issues at once, we introduce the *transmorph* framework. It allows the user to engineer powerful data integration pipelines and is supported by a rich software ecosystem. We demonstrate *transmorph* usefulness by solving a variety of practical challenges on scRNA-seq datasets including joint datasets embedding, gene space integration, and transfer of cycle phase annotations. *transmorph* is provided as an open source *python* package.

GRAPHICAL ABSTRACT



INTRODUCTION

Batch effects occur in most applications involving datasets gathered across multiple sources or experiments, and describe strong dataset-specific signals which are often irrelevant to the studied biological questions. Data integration is a computational paradigm aiming to learn a joint embedding of datasets in which batch effects are regressed out (Figure 1A), meaning only dataset-independent factors are expressed. The idea is to combine information contained in several datasets, each of those being supposedly biased by its own specific batch effects. We focus here on the so-called *horizontal data integration* (1) which seeks to integrate datasets obtained within the same domain with overlapping feature spaces. This is different from *vertical* and *diagonal data integration* where cells are measured in different domains, also known as multi-level or multi-omics data integration. This scenario involves specific strategies and algorithms which are beyond the scope of this project (see for instance (2)).

*To whom correspondence should be addressed. Tel: +33 156246989; Email: aziz.fouche@curie.fr
Correspondence may also be addressed to Andrei Zinovyev. Tel: +33 156246989; Email: andrei.zinovyev@curie.fr

Data integration is an important preprocessing step for applications involving several datasets. In some cases, approaches as simple as centering and normalization/scaling of features may suffice, but more complex batch effects often require more subtle, dedicated algorithms to be satisfactorily removed. Data integration can serve various purposes. The most common usage is to embed items from all datasets into a joint low dimensional space like in Harmony (3), which can then be used to carry out various techniques such as clustering, label transfer, or visualization. Another use case is to directly perform integration in gene space like in the mutual nearest neighbors (MNN) method (4) so that algorithms needing interpretable features such as matrix factorization methods can be used. Finally, integration can be carried out without embedding data points into an explicit feature space, for instance by outputting a joint graph of cells across datasets like in BBKNN (batch balanced *k* nearest neighbours)(5).

Data integration finds particularly important applications in single-cell biology. Starting with a biological tissue, a single-cell dataset is generated and contains individual molecular measurements (for instance gene expression, SNPs, or chromatin accessibility) about single cells of the tissue. The strength of single-cell analysis is its ability to both provide an insight into intrinsic cell state, while also giving access to population-level information that can for instance be used to estimate cell types distribution within a tissue, which makes this technology extremely relevant for analyzing patient samples in medicine. Due to genetic and environmental differences between individuals, batch effects are very prone to appear when dealing with single-cell datasets coming from different patients. The intertwining of batch-dependent and batch-independent factors is therefore an obstacle to the analysis of large comprehensive datasets built by aggregating data from different individuals, notably when building cell atlases (see for instance (6)). Data integration is consequently a necessary technology to develop in order to mitigate dataset-specific signals while preserving relevant biological signals proper to the system of interest.

Many approaches have been proposed over the last decade to tackle data integration in single-cell, resorting to a variety of algorithmic strategies. The most widely used software today is probably Harmony (3), which iteratively clusters and corrects cell measurements in a low dimensional space to achieve integration. Other methods such as mutual nearest neighbors (4), CONOS (7), Seurat (8) and BBKNN (5) work under the hypothesis that batch effects are almost orthogonal to biological effects. Under this assumption, they use nearest neighbors-like algorithms across datasets as a measure of cell–cell similarity. Other data integration methods like SCOT (9) and Pamona (10) use optimal transport-based algorithms to estimate cell-cell similarity, assuming the existence of similar manifolds supporting datasets containing cells of the same type. Finally, variational autoencoders (VAEs) and other neural network-based tools like scvi (11) have been proposed, which use VAEs to directly learn dataset embeddings inside a joint latent space.

As aforementioned, integration of single-cell RNA-seq data has been a prolific topic for the last decade, and dozens of methods still appear each year in the literature (1). We believe this surge of methods comes with an urgent need for organization, classification, and large-scale benchmarks (see for instance (12)) of both (a) end-to-end data integration pipelines, in order to guide the computational biologists that need to apply data integration to their research projects, and (b) algorithms, to guide methodologists who conceive new data integration methods. For this reason we introduce *transmorph*, an intuitive computational framework that aims to decompose data integration methods into basic algorithmic units that can be freely rewired to make emerge new data integration pipelines. These units can either be extracted from the literature, such as a nearest neighbors search or a PCA projection, or can be customized at will by the user. This flexibility allows scientists to harness integration methods adapted to the specificities of their data, for instance by choosing an output algorithm that produces an integration directly in gene space, or by selecting a matching algorithm that takes data topology into account, such as optimal transport (Figure 1B).

In addition, *transmorph* is endowed with other features that facilitate its integration within real-life workflows of scRNA-seq data analysis (Figure 1C). It is fully compatible with the standard scRNA-seq library *scanpy* (13) as they handle the same type of objects, is interfaced with other first-class data integration tools such as *Harmony* (3) and *scvi* (11), and contains several annotated scRNA-seq datasets as well as standard quality assessment metrics and plotting functions to validate data integration algorithms. Finally, a comprehensive *model API* is available to allow the user to implement their own algorithmic units, using an object-oriented specification (Figure 1D). For these reasons, we think *transmorph* is both an original and a powerful asset to design, apply, and benchmark data integration methods.

MATERIALS AND METHODS

transmorph implementation and algorithms

The *transmorph* framework is provided as an open-source Python package and can be downloaded from the PyPi package repository. The following paragraphs describe algorithms and resources that are either original or key to understanding all the results. Additional methods can be found in the supplementary note.

Local inverse Simpson's index

Local inverse Simpson's index (LISI) is an objective integration metric introduced in Harmony (3), which assesses neighborhood heterogeneity of a data point in terms of a given label. Simpson's diversity index is a diversity metric notably used in ecology to measure class diversity in a set of objects by computing the probability for two randomly selected items to share the same class. For any set of objects $S = \{x_i\}_{i \leq n}$ endowed with labels $y_i \in \mathcal{L}$, we denote by n_l for $l \in \mathcal{L}$ the number of samples in S with label l. Then, Simpson's index of set S is given by

$$D_{\mathcal{L}}(S) = \sum_{l \in \mathcal{L}} \left(\frac{n_l}{n} \right)^2 . \quad (1)$$

For $k > 0$ a *perplexity* parameter (we use $k = 90$) and x an embedded point, we compute its k-nearest neighbors k-nn(x) which is used as set S. $\text{LISI}_{\mathcal{L}}(x, k) = D_{\mathcal{L}}(k\text{-nn}(x))^{-1}$ is defined as the inverse of Simpson's index and estimates, for a given embedded point x, label diversity in its k-nearest neighborhood. As suggested in Harmony, we can use LISI in two modes:

- Batch-LISI, where points are labeled by their initial batch. This metric measures local batch diversity in embedding, higher diversity meaning higher batch mixing.
- Class-LISI, where points are labeled by their class. This metric measures local class diversity in the embedding, lower diversity meaning lower mixing between cell types.

Monitoring these two values allows objective comparison of different integration pipelines, ideally increasing batch-LISI while avoiding increasing class-LISI.

Graph embedding algorithm

Joint graph embedding is an algorithm able to build a joint weighted graph of cells from all batches, where two cells are linked together if they appear to be similar. This graph is weighted according to a UMAP-like methodology, meaning it can be embedded in a low dimensional space using UMAP (14) or MDE (15) optimizers. The joint embedding algorithm consists of four major steps. More details can be found in the Supplementary note.

(1) For each dataset, compute the k-nn graph of its cells weighted according to UMAP membership methodology.
(2) For each pair of batches, weight matching edges according to UMAP membership methodology.
(3) Build a joint graph combining edges of steps 1 and 2, and filter edges so that only the most heavily weighted ones are kept.
(4) Embed the joint graph into an abstract feature space using a graph embedding optimizer such as UMAP or MDE.

Discrete optimal transport

Discrete optimal transport (OT) problem can be naturally pictured as follows (16). Assuming a set of n warehouses containing goods to deliver to m factories, the optimal transport problem consists in finding the cheapest way to transport all goods to factories knowing the cost of transporting goods is proportional to both mass carried and distance traveled. In the single-cell case, it provides a natural way to match cells between two datasets embedded in a common expression space. Originally brought into the field as a way to predict cell fate (17), it has more recently been shown to be an interesting asset for matching cells across datasets in integration tools like SCOT (9) and Pamona (10). Details about optimal transport computation and its variants can be found in the Supplementary note.

The main practical issue of optimal transport application for data integration is its mass conservation constraint: OT will always match *all* mass from the source dataset $\mathbf{X}_a$ to the target dataset $\mathbf{X}_b$, regardless of the possible batch-specific samples. Consequently, all cells in $\mathbf{X}_a$ will be mapped to at least one cell in $\mathbf{X}_b$, even though some cells from $\mathbf{X}_a$ may belong to a cell type missing in $\mathbf{X}_b$. Even worse, if there is a class imbalance between datasets (e.g. 50% of cell type A in dataset $\mathbf{X}_a$, and 25% of cell type A in dataset $\mathbf{X}_b$), there will necessarily be wrong assignments using this method. Exact computation of optimal transport is furthermore computationally expensive, of the order of $\mathcal{O}((n+m)^3)$ which makes it inefficient for large-scale problems (typically above 10^4 points). The Supplementary note contains an alternate approximate and unbalanced formulation which provides a good approximation of the solution at a more reasonable cost, while also dealing with the class imbalance issue.

Independent component analysis

Independent component analysis (ICA) is a matrix factorization (MF) approach where the signals captured by each individual matrix factor are optimized to become as mutually independent as possible. ICA was shown to be a useful tool for unraveling the complexity of cancer biology from the analysis of different types of omics data. Such works highlight the use of ICA in dimensionality reduction, deconvolution, data pre-processing, meta-analysis, and others applied to different data types (transcriptome, methylome, proteome, single-cell data) (18).

In ICA we search for an approximation of the observed probability density function $P(x_1, x_2, .., x_n)$ by $\hat{P}(s_1, s_2, ..., s_n)$, where new s_i variables are some linear combinations of the initial variables x_i. We search for such linear transformation that $\hat{P}(s_1, s_2, ..., s_n)$ deviates as little as possible from the product of its marginal distributions $P(s_1) \times P(s_2), .., P(s_n)$ where the deviation is usually defined in terms of information geometry (e.g., as Kullback–Leibler divergence). It is shown that ICA is efficient in detecting and correcting the batch effects in omics datasets (18).

ICA is not a data dimensionality reduction technique *per se*: therefore, it is usually applied on top of reduced (e.g. by standard PCA) and whitened representation of the initial dataset. Therefore, the choice of the number of independent components is an important hyperparameter (19). In the simplest approach, the ICA solution represents a rotation of the whitened data point cloud such that each normalized coordinate deviates as much as possible from the standard Gaussian distribution (20).

In our experiments, we used the stabilized version of ICA (21) which is shown to be the optimal MF approach for reproducible analysis of transcriptomic data (22). We applied it to cells labeled as T-cells from all datasets to prevent dataset-specific cell type imbalance to bias the components.

Barycentric embedding and label transfer

Barycentric merging is the simplest merging to set up. It works under three assumptions, (i) one batch $\mathbf{X}_r$ is defined as *reference* and all batches will be corrected towards it; (ii) reference batch $\mathbf{X}_r$ must be expressed in a feature space; (iii) for every matching $\mathbf{M}_{sr} \in \mathbb{R}^{n_s \times n_r}$, every row must have at least one nonzero element ($\left\| \mathbf{M}_{sr} \mathbf{1}_{n_r} \right\|_0 = n_s$). Assumption (i) is fulfilled by user choice and necessitates choosing a

good quality batch with representative items within every sample type. Reference choice always introduces a bias in the integration, which should not be overlooked in the interpretation of the results. Assumption (ii) is easy to verify in practice, as datasets are often vectorized and represented as $n \times d$ real-valued matrices. Assumption (iii) necessitates the choice of a *semicomplete* matching, which maps every sample from batch X_i to at least one sample from batch X_r. Transportation-based matchings usually verify this assumption, while nearest neighbors-based matchings usually do not. Failing to verify assumption (iii) will cause non-matched points to be projected to the $\mathbf{0}$ of $\mathbf{X}_r$ feature space.

Let X_s be a batch to correct with respect to the reference batch $\mathbf{X}_r$ given a semicomplete, row-normalized matching matrix $\mathbf{M}_{sr}$. For every sample $x_k \in X_s$, the kth row $\boldsymbol{\alpha}_k = \mathbf{M}_{sr,\,k}$ provides a weighting vector which assesses the likelihood of x_k corresponding to any sample of $\mathbf{X}_r$. Barycentric merging F^{Bary} will then project x_k into $\mathbf{X}_r$ feature space $\mathcal{X}_r$ so that

$$F^{\text{Bary}}_{\mathbf{X}_r,\alpha}(\mathbf{x}_k) = \arg\min_{\mathbf{x}\in\mathcal{X}_r} \sum_{i\leq n_r} \alpha_i \left\|\mathbf{X}_{r,i} - \mathbf{x}\right\|_2^2. \tag{2}$$

It is easy to show $\mathbf{x}_k = \sum_{i\leq n_r} \alpha_i \mathbf{X}_{r,i}$ is the solution to this problem. Therefore, F^{Bary} can be easily rewritten to project the whole $\mathbf{X}_s$ dataset onto $\mathbf{X}_r$ given $\mathbf{M}_{sr}$ via

$$F^{\text{Bary}}_{\mathbf{X}_r,\mathbf{M}_{sr}}(X_s) = \mathbf{M}_{sr}\mathbf{X}_r. \tag{3}$$

Barycentric merging has been used in several data integration pipelines such as Seurat (8), SCOT (9) and Pamona (10), and generally yields good results, though there are a few downsides to consider. First, choosing a reference introduces a high bias in the integration, and in some applications, there may be no good option for reference; for instance, every batch could miss at least one sample class. The barycenter problem also intrinsically relies on a metric. This is an issue for high dimensional problems, for instance in scRNA-seq datasets where the curse of dimensionality is a real concern; in this case, barycenter has little to no interpretable sense. A common solution is to first reduce the dimensionality of $\mathbf{X}_r$ using principal component analysis or non-linear methods such as UMAP (14) or MDE (15). One of the other uses of this method is to use a matching computed in a different space than the final embedding. Typically, one computes a matching in a lower dimensional representation (e.g. PC space) but uses total feature space for the embedding. This notably allows obtaining corrected feature counts for all batches with respect to a reference. Combined with a high-quality matching and reference batch, barycenter merging can nonetheless provide an efficient, high-quality integration without necessitating batches to be originally in the same space.

Label transfer was carried out in the integrated space using a simple nearest-neighbors approach. We used the *scikit-learn* implementation of a k-nearest neighbors classifier using $k = 10$ and Euclidean distance, and the majority rule.

Single-cell RNA-seq datasets

We used public datasets to benchmark our framework and compare its capabilities with other state-of-the-art integration pipelines. They were chosen to mimic various real-life scenarios, with total dataset sizes in the tens of thousands. All datasets contain RNA-seq data, acquired using 10× technology.

- The Zhou databank was collected from (23) through the Curated Cancer Cell Atlas (3CA) website and contains osteosarcoma data from 11 different patients, ranging from 866 to 14 322 cells for a total of 64 557 cells. Each cell was annotated by the authors with a cell type among chondrocyte, endothelial, fibroblast, mesenchymal stem cell (MSC), myeloid, myoblast, osteoblast, osteoclast, pericyte, T cell.
- The Chen databank was collected from (24) using the 3CA website and contains 61 870 nasopharyngeal cancer single-cell RNA-seq data from 14 different patients, ranging from 1087 to 11 210 cells. Each cell was annotated by the authors with a cell type among B cell, endothelial, epithelial, macrophage, malignant, NK cell, plasma and T cell.

Raw counts have been preprocessed following standard guidelines using the *scanpy* python package (13). First, cells with low gene counts or high mitochondrial gene expression were filtered. Raw counts were then normalized to 10 000 per cell, followed by neighborhood pooling using 5 nearest neighbors. Counts were then $\log(1 + x)$ transformed, and for each dataset, the top 10 000 most variable genes were kept. All these preprocessed annotated databanks can be automatically downloaded through our framework, in order to serve for benchmarking integration methods.

Retrieving cell cycle signal from scRNA-seq datasets

Raw counts for Ewing sarcoma cell lines datasets CHLA9, CHLA10 and TC71 were obtained from (25). Raw counts and annotations for the osteosarcoma U2OS dataset were obtained from (26). They were preprocessed according to state-of-the-art guidelines. Raw counts per cell were normalized to 10 000 to account for differences in global expression and were then $\log(1 + x)$ transformed; the top 10 000 variable genes were kept in each dataset. Data points were eventually pooled to reduce noise, by setting every cell counts vector to the average of its 5 nearest neighbors (neighbors were determined using Euclidean distance in a 30-PC space). We used cell cycle genes identified in (27) to characterize G1/S and G2/M signals. For fast cell cycle datasets TC71 and U2OS, we used only a subset of informative G1/S genes which helped to retrieve a proper circular signal (CDK1, UBE2C, TOP2A, TMPO, HJURP, RRM1, RAD51AP1, RRM2, CDC45, BLM, BRIP1, E2F8 and HIST2H2AC). Integration was carried out using full gene space.

Benchmarking methods and parameters

All benchmarks have been run on a laptop equipped with 32GB of RAM, an Intel CPU i7-10750H (12 cores)

processor at 5 GHz and an NVIDIA GPU GeForce GTX 1650 Ti Mobile.

- EmbedMNN was used on preprocessed counts with *transmorph* v0.2.0, using default parameters: 'bknn' matching, 10 matching neighbors, 10 embedding neighbors, UMAP optimizer and 2 dimensions.
- BKNNCorrection was used on preprocessed counts with *transmorph* v0.2.0, using default parameters: 'bknn' matching, 30 matching neighbors, 10 linear correction neighbors.
- TransportCorrection was used on preprocessed counts with *transmorph* v0.2.0, using solver= 'unbalanced', entropy_epsilon=0.02, unbalanced_reg=5.
- Harmony was used with default parameters directly on preprocessed counts using the *rpy2* python interface. We also tried the *harmonypy* python implementation, interfaced *viascanpy*. It successfully converged in under 10 iterations in both cases and produced comparable results.
- scvi was used on raw counts following the authors' guidelines with n_layers=2 and n_latent=30, and was optimized during 124 epochs (automatically chosen by the software).
- We used the *scanpy* implementation of BBKNN on preprocessed counts. We carried out BBKNN with default parameters on a 50-PC representation of datasets using default parameters, using neighbors_within_batch=3 and 10 annoy trees.
- Seurat was used in RStudio after converting AnnData datasets to h5seurat using the SeuratDisk package. We carried out the integration using SelectIntegrationFeatures, FindIntegrationAnchors and IntegrateData with default parameters. We were not able to complete the last integration step despite our efforts due to memory usage issues.

RESULTS

transmorph allows conceiving end-to-end data integration models

Despite potentially achieving very good integration results in specific use cases, we believe that existing data integration algorithms are flawed by their intrinsic rigidity. By constraining the user to a fixed algorithm, they tend to excel in some use cases while struggling in others, as we show in the next sections. Also, the lack of access to their internal algorithms can make results difficult to interpret. Furthermore, these internal algorithms cannot be easily modified when needed, notably when the user needs a particular output type that the algorithm is not able to provide. For instance, despite the fact is usually yields high-quality embeddings, the Harmony algorithm cannot perform data integration in gene expression space, which can be a downside for the subsequent application of deconvolution methods such as independent component analysis. Another issue we can mention is the fact some matching paradigms are not suited for certain datasets topologies, as we show in the last subsection where nearest neighbors-based algorithms have trouble matching cycling cells. Finally, some algorithms do not scale as well as others to large datasets, which can disqualify certain tools from being applied in these situations, such as optimal transport-based methods.

To address these limitations we present *transmorph*, a novel and ambitious data integration framework. It features a modular way to create data integration algorithms using basic algorithmic and structural blocks, as well as analysis tools including embedding quality assessment and plotting functions. The framework also provides annotated, high quality and ready-to-use datasets to benchmark algorithms (Figure 1C). Finally, it is meant to be easily expansible by allowing the user to define new algorithmic modules if necessary. In this framework, data integration models can be assembled by combining four classes of algorithms: transformations, matchings, embeddings, and evaluators (Figure 1D, E, Table 1).

- **Transformation** algorithms take as input a set of datasets and return a new representation for each of them, embedded in some feature space (there can be one separate feature space per dataset or one common feature space). Transformations are generally used during preprocessing: classic examples are PCA, neighborhood-based data pooling, or common highly variable genes selection.
- **Matching** algorithms estimate a similarity measure between cells across datasets. They are the core component of our integration framework, as their quality directly influences cell-cell proximity in the final embedding. *transmorph* uses three main categories of matching: (a) label-based matchings which require datasets to be labeled beforehand and match items of a similar label; (b) neighbor-based matchings which match items close items with respect to some metric; (c) transport-based matchings which leverage a distance metric between items within or across datasets to compute a similarity between items relying on topological correspondence.
- **Embedding** algorithms are a particular class of transformations that take as additional input similarity relationships between samples that were estimated via a matching. They return an integrated view of all datasets jointly embedded in a common feature space so that matched items tend to be close to one another in the final representation. The embedding step is in general the last step in an integration model and is chosen depending on the required output type. For instance, a joint embedding of datasets in an abstract space suits applications like visualization or clustering. At the same time, matrix factorization algorithms often require the embedding to be performed in an expressive feature space.
- **Evaluation** algorithms are quality control points that can be added to a pipeline to test a condition. They are used to either set a branching point that leads to different outcomes or create an iterative structure within a model ('*repeat until the integrated representation satisfies this property*'). This strategy is notably used within the Harmony algorithm, where an iterative clustering and correction procedure is applied until an integration metric (Local Inverse Simpson's Index in this case) is considered satisfactory.

This expressive framework allows the building of complex data integration models suited for many applications

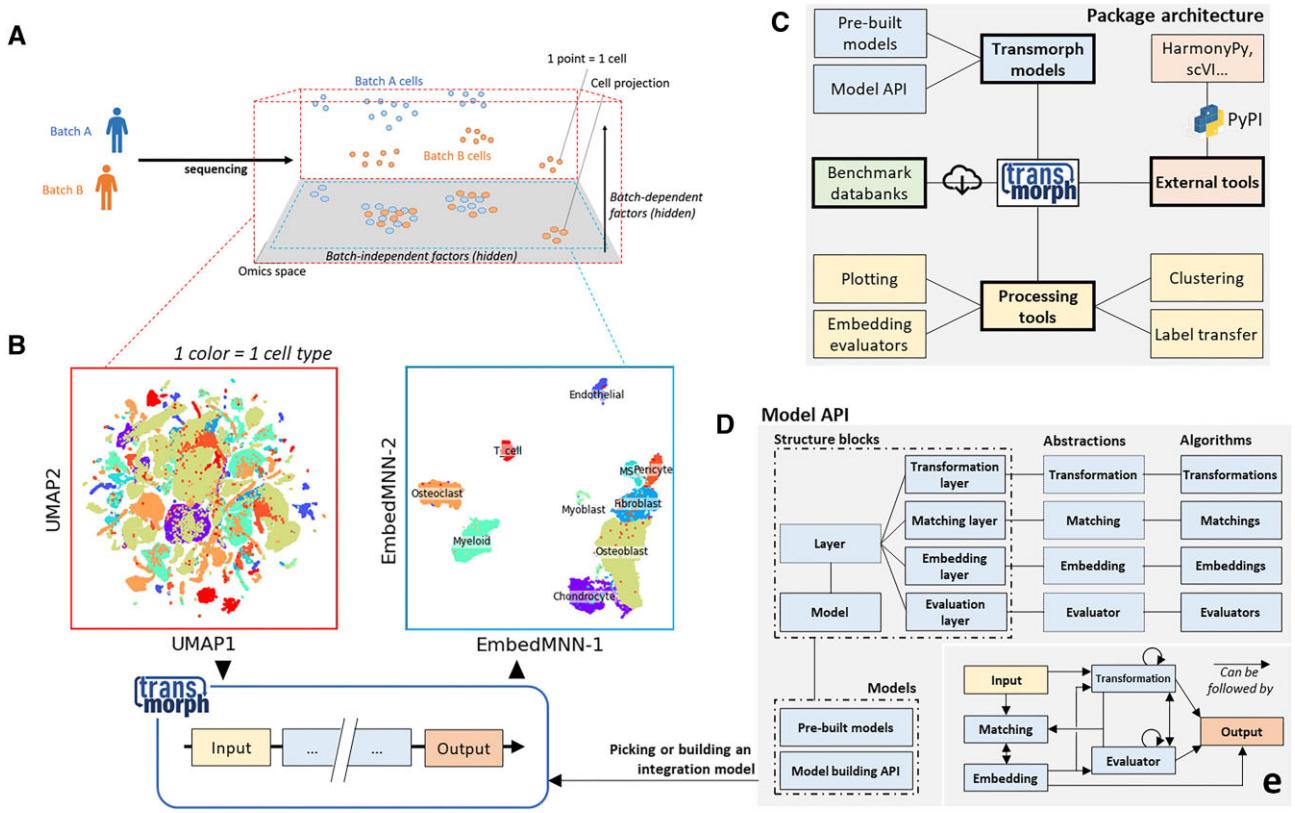


Figure 1. Transmorph is a framework for scRNA-seq data integration. (**A**) Schematic representation of the data integration problem. (**B**) Transmorph integration models conduct data integration of scRNA-seq datasets. Once the integration has been performed, cells cluster by type or state instead of origin. (**C**) Transmorph global package architecture, featuring internal and external models, benchmarking scRNA-seq databanks, and analysis tools. (**D**) Architecture of the model API, which allows engineering new data integration models using basic building blocks. (**E**) Directed compatibility chart of the model API modules, with arrows indicating how algorithms can be articulated within *transmorph* pipelines.

with high computational efficiency and integration quality because each algorithmic module can be optimized independently. It also provides an objective comparison between algorithmic modules for a given application. Finally, it is supported by a sound software ecosystem with benchmarking databanks, pre-built models, and post-analysis tools, which allows one to carry out data integration within a scRNA-seq analysis workflow efficiently. Our framework is provided as an open-source Python package, and the following results showcase its capabilities to solve various challenging real-life problems of single-cell RNA-seq data integration while being on par with existing tools in terms of performance. It has been developed to be easily used in notebook environments, with a strong focus on computational efficiency so that models can be run on small machines in a few minutes, even in applications involving tens of thousands of cells and more than ten different datasets and cell types.

Transmorph models perform on par with other state-of-the-art tools

We will first present how the *transmorph* framework can be used to create data integration models able to compute a low dimensional joint embedding of two or more datasets so that similar cells end up close to one another independently from their source. This type of task is typically used for visual data exploration or as a preprocessing step before carrying out a clustering algorithm, allowing clusters to only depend on cell type rather than on the original batch (Figure 2A). A good joint dataset embedding algorithm should be able to function in a fully unsupervised fashion while being improved by additional labeling information, and should not require choosing a reference dataset as this induces an important bias. Ideally, it should also be able to tackle the joint embedding of more than two datasets simultaneously, with reasonable computational efficiency. We built a *transmorph* model for this application, called *EmbedMNN*, described in (Figure 2B). EmbedMNN is conceptually inspired by CONOS (7), and starts with a few preprocessing steps (normalizations and dimensionality reduction). It then combines a nearest neighbors-based joint graph construction step with a low dimensional graph construction, followed by an embedding step using either UMAP (14) or minimum distortion embedding (MDE) (15). This allows EmbedMNN to work without requiring a reference and with datasets of various topologies, and to output an embedding in a latent space that will be exploitable for clustering and visualization. Furthermore, EmbedMNN can work either in a fully unsupervised fashion or can take into account label information to prune matching edges between samples of different labels; we test both variants in this application.

Table 1. Presentation of the main algorithmic modules available in the *transmorph* framework that can be used to build data integration pipelines. A brief explanation is given for each of them, additional information as well as algorithm parameters are available in the *Materials and Methods* section and in the *transmorph* documentation

Transformations	Matchings	Embeddings	Models
Common Features: Selects and orders common genes between either all datasets or pairs of datasets.	**KNN:** Matches nearest neighbors of each cell across batches.	**Barycenter:** Projects each cell in a *query dataset* to the average value of its matches in a *reference batch* that must be specified by the user. This embedding can produce a result in gene space, that can then be treated as scRNA-seq data. It necessitates that all cells in the *query* dataset have a match.	**TransportCorrection** Takes as input two or more scRNA-seq datasets, with one chosen as an alignment reference. It computes optimal transport between each dataset and the reference. It then uses the barycentric embedding to align each dataset to the reference and can output the result either in PC space or in gene expression space.
Standardization: Normalize expression values per gene or per cell in order to improve the quality of geometric methods.	**MNN:** Matches cells that mutually belong to the nearest neighbors of the other across batches.	**Graph Embedding:** Links cells from all datasets into a single common weighted graph. This weighted graph is then embedded in a space whose dimensionality is chosen by the user, a space that is used as this module's output. Due to the nonlinearity of this approach, the final representation can be used for clustering or other topological analyses.	**EmbedMNN:** Takes as input two or more scRNA-seq datasets, without requiring a reference to be specified. Cells are matched using a nearest neighbor scheme chosen by the user (KNN or MNN), and are organized within a joint weighted graph whose specification is described in *Material and Methods*. This graph is finally embedded using UMAP or MDE.
Pooling: Pools each cell vector towards an average of its neighbors to reduce the effect of outliers.	**Optimal Transport:** Matches cells across datasets using an optimal transport approach, with each dataset viewed as a mixture of Dirac distributions. This algorithm performs best when datasets topologies are similar, and penalizes translations, scaling and rotations.	**LinearCorrection:** Computes correction vectors from cells in the *query dataset* and their match in the *reference dataset*. Unmatched cells are then attributed to a mixture of vectors of the nearest matched cells. All cells are eventually translated according to the correction vector that has been computed.	**MNNCorrection:** Takes as input two or more scRNA-seq datasets, with one chosen as an alignment reference. Cells are matched using the nearest neighbor scheme chosen by the user (KNN or MNN). It then uses a linear correction module to align cells from each query dataset onto the reference one.
PCA: Linearly projects cell vectors into a variance-preserving, low dimensional basis to reduce the curse of dimensionality effect.	**Gromov-Wasserstein:** Matches cells across datasets using a Gromov-Wasserstein algorithm which only accounts for data topology, without penalizing isometric transformations.		
ICA: Linarly projects cell vectors into a low dimensional basis of statistically independent vectors to reduce the curse of dimensionality effect.	**Fused Gromov-Wasserstein:** Matches cells across datasets using a linear mixture of optimal transport and Gromov–Wasserstein in order to balance the penalty between geometry and topology.		
	Combined: Combines several matchings into a single one.		

Even though *transmorph* is not a data integration algorithm *per se*, but rather a framework to conceive data integration methods, we decided to benchmark the EmbedMNN model against other state-of-the-art horizontal integration algorithms. For this benchmark, we selected three algorithms designed to solve the joint embedding problem: Harmony (3), which uses a clustering-driven, iterative strategy to optimize the embedded representation. scvi (11), a deep learning framework that uses variational autoencoders to compute a latent integrated representation of datasets. BBKNN (5), which builds a weighted joint graph of datasets together using a batch-balanced variant of k-nearest neighbors. We embedded Harmony, scvi latent representation, and BBKNN results into a 2D space using UMAP (14) so that all methods' output space is comparable.

The benchmarking databank consists of 11 single-cell osteosarcoma datasets gathered from (23), containing approximately 65 000 cells in total, annotated by the authors with ten different cell types (Figure 2A, Supplementary Figure S1A). We will use this author's annotation as the 'ground truth' for this application and measure how the different methods deviate from it. This use case is quite challenging due to dataset size, number of batches, and number of classes, but it illustrates a reasonable real-life use case of data integration. Integration performance can be

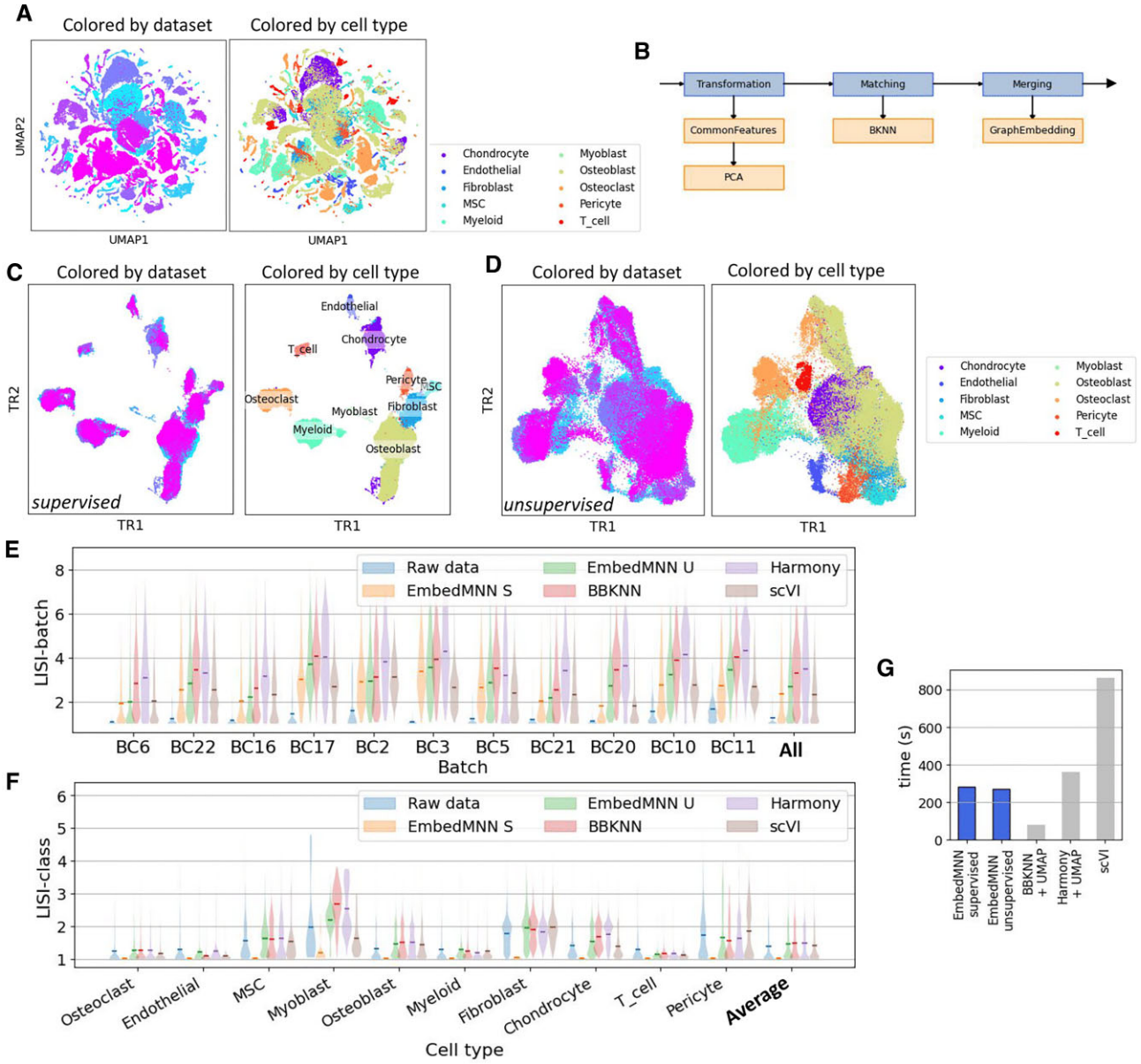


Figure 2. Integration of 11 osteosarcoma scRNA-seq datasets ($n = 64\,557$) from different patients. (**A**) Initial UMAP representation of the osteosarcoma datasets in their common genes space. (**B**) Architecture of the pre-built EmbedMNN integration model, computational modules are executed from left to right and from top to bottom. (**C**) Integration results with the supervised version of EmbedMNN. (**D**) Integration results with the unsupervised version of EmbedMNN. (**E**) LISI-batch score of various integration algorithms (higher is better), mean is marked. (**F**) LISI-class score of various integration algorithms (lower is better), mean is marked. (**G**) Execution time of various integration algorithms.

objectively measured through four integration metrics: set and class mixing using a lightened version of local inverse Simpson's index (LISI) introduced in Harmony, clustering specificity using Louvain or Leiden community detection algorithm (28,29), and computation time. It is to note that Harmony directly uses batch-LISI as a stopping criterion during its optimization procedure, so we have to expect it to have superior batch-LISI scores.

All methods could compute the integrated embedding in a reasonable amount of time given the number of data points (Figure 2G, Supplementary Figure S1), with the best performer being BBKNN + UMAP with 1min10s, taking advantage of the highly optimized *C++* nearest neighbors approximation library *annoy*. Both supervised and unsupervised versions of EmbedMNN algorithms could finish in under 5 minutes. At the same time, Harmony took 5min30s plus an extra 30s of UMAP computation to obtain a 2D embedding. scvi was the longest to complete, with around 10 minutes in total, but in all fairness, the minimum loss seemed to be reached between the 2 and 3 min mark.

Computed joint representations were reasonable overall for all methods, with effective batch mixing and cell type clustering (Figure 2C, D, Supplementary Figure S2A, B). Nonetheless, no method achieved both excellent batch mixing and cell type separation, which is to be expected on such complex datasets (a large number of cells, patients, and

cell types). Unsurprisingly, the supervised version of EmbedMNN outperformed all other methods by a large margin both in terms of local cell types homogeneity and clustering purity (Figure 2F, Supplementary Figure S2C, D), with a very low LISI-class score for all cell types and a near-100% cluster purity, as it leveraged complete label information. This allowed it to prune edges between cells of different types during the matching step, which resulted in a very clean cells graph to embed. On the other hand, supervised EmbedMNN is associated with inferior batch mixing (Figure 2E), and more explicit cluster delimitation after integration which can be an obstacle for some trajectory inference algorithms. The unsupervised version of EmbedMNN appears to be on par with the other methods, with good LISI-class and LISI-batch scores (Figure 2E, F) and good clustering purity (Supplementary Figure S2C, D).

Overall, this shows that *transmorph* provides a framework capable of creating data integration models of sufficient quality to tackle joint dataset integration of challenging scRNA-seq datasets in terms of computational efficiency and integration quality. In the next section, we will show that its modularity allows the user to modify a *transmorph* model to change its output space (from an abstract space to a gene expression space), which is not possible to our knowledge with the other tools presented in this first scenario.

Performing integration in gene space by using an appropriate embedding

In some applications, providing a joint embedding of datasets into an abstract space is not suited, as original features (i.e. genes) do carry important information for output interpretability. This is for instance the case when performing matrix factorization algorithms such as independent component analysis (ICA) or non-negative matrix factorization (NMF), or when annotating cells with appropriate cell types. In this case, it is necessary to perform the integration directly within gene space, which brings some technical difficulties. Notably, gene spaces are often very large which is detrimental to the scalability of distance-based algorithms due to the curse of dimensionality. In this scenario, EmbedMNN, Harmony or BBKNN are not adapted, as they are unable to return their output in full gene space. This would normally imply we need to find another integration tool to carry out the integration in gene space, which would come with important time costs (package installation, data processing, workflow adaptation...). In this example, we demonstrate how the modular nature of the *transmorph* library can instead provide a way to adapt an existing model to suit a new application easily. We first identify that the embedding step of EmbedMNN is by design not adapted to a full gene space application. To tackle this limitation, we can swap this module for something more adapted like a linear correction step in gene space (Figure 3A), which instead leverages correction vectors in a similar fashion to what is used within the MNN (4) and Seurat (8) tools, and can handle the property of neighbor-based matchings that do not provide a match to every cell from the query dataset. Given a reference dataset, the linear correction approach consists in first, finding some matchings between query and reference items, then computing correction vectors from these queries to their references, to finally propagating these correction vectors along the query dataset to end up with corrected profiles. This last step allows for the alignment of query cells that have no match in the reference dataset. Furthermore, contrarily to graph embedding, linear correction step can be carried out in gene space to obtain a gene expression matrix as output. This makes it a natural choice for this application.

We use 14 nasopharyngeal carcinoma datasets gathered from (30) to benchmark the strategy (Figure 3B, Supplementary Figure S1B). The goal is to embed these datasets in the space defined as the intersection of their common most variable genes so that cells sharing the same annotation end up in close proximity after integration. This is once again a challenging task as the datasets are quite large (>60 000 cells to embed), there are eight different cell annotations, some datasets do not contain cells from all types, and the embedding space is large for a geometrical approach (>900 genes). To measure integration quality from another angle, we carry out ICA on T-cells from all datasets, which allows us to observe dataset-specific gene expression signals without the bias of cell type imbalance between datasets. As we can see, before integration the dataset-specific signal appears to be strongly correlated with several independent components (ICs) computed by ICA (Figure 3F, top).

BKNNCorrection completes in a very reasonable time of 1 minute and 33 seconds and provides a convincing correction (Figure 3B) by being associated with great improvements in LISI-batch (Figure 3C, top) while maintaining low levels of LISI-class [Figure 3C, bottom]. We were not able to successfully carry out Seurat integration on these datasets in a reasonable time and memory usage on this dataset using our machine. Overall, this showcases how *transmorph* provides a new way to easily tweak models, allowing them to tackle different scenarios with good efficiency and integration quality. We also eventually ensure most of the dataset-specific signal has disappeared after integration (Figure 3F, bottom), resulting in a weak correlation with any of the ICs recomputed by ICA on the integrated dataset. This is a desired property for subsequent accurate interpretation of the independent components through, for example, functional enrichment analysis.

Gene space integration can be leveraged to annotate cell types reliably

Gene space integration can be leveraged in a very natural way to perform cell type annotation. As integration outputs new gene counts for each cell, these new molecular profiles can be used within the integration space to perform clustering and cell type annotation via differential gene expression analysis. These newly found annotations can be expected to be more precise than annotations performed on each dataset individually and can allow rarer cell types to be identified with high statistical confidence. In particular, most cell type annotation strategies rely on prior cell clustering to label each cluster with a cell type according to marker genes. Frequently, rare cell types do not form a separate cluster in the original datasets due to their limited population size, while they should constitute a larger

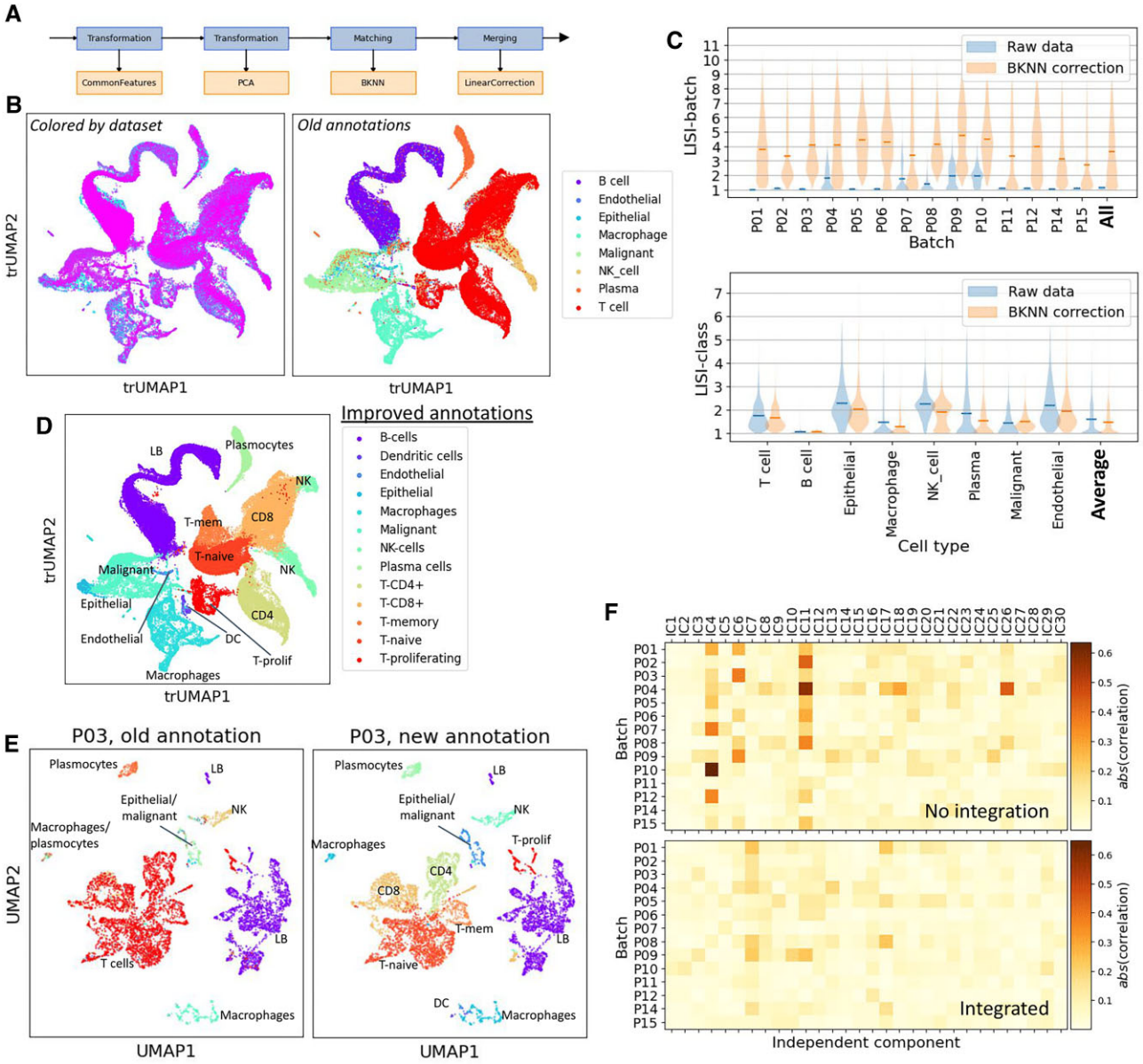


Figure 3. Gene space integration of 14 nasopharyngeal carcinomas scRNA-seq datasets from different patients (n = 61 870). (**A**) Architecture of the BKNNCorrection pre-built integration model performing integration in gene space. Computational modules are executed from left to right and from top to bottom.. (**B**) UMAP visualization of the integration result, colored by dataset (left) and by original cell type annotations (right). (**C**) LISI-batch (top, higher is better) and LISI-class (bottom, lower is better) before and after integration, mean is marked. (**D**) UMAP representation of the integration result, endowed with new cell type annotations determined within the integrated gene space. (**E**) UMAP representation of the dataset P03, with old (left) and improved (right) cell type annotations. Comparative plots for other datasets can be found as supplementary figure. (**F**) Absolute value of the correlation between each independent component and each batch among T-cells, before (top) and after (bottom) integration.

cluster once datasets have been integrated together. Newly found annotations can eventually be mapped back to the individual datasets. We will use this methodology to improve annotations found in the previously used nasopharyngeal carcinoma scRNA-seq datasets.

We performed a clustering of datasets integrated into the space of their common genes and performed a differential gene expression on these clusters (Supplementary Figure S3A, B). We then determined cell types by combining initial annotations, well-known marker genes as well as PanglaoDB (31). Doing so allowed us to confidently annotate 13 different cell types, greatly refining initial annotations (Figure 3B, D). Comparing old and new annotations for each cell shows most annotations have been made more precise rather than corrected (Supplementary Figure 3C), notably splitting the 'T cell' label into the various lymphoid lineage-associated labels 'T-naive', 'T-CD4+', 'T-CD8+', 'T-memory' and 'T-proliferating', and the 'macrophage' label into the myeloid lineage-associated labels 'macrophages' and 'dendritic cells'. The only different annotations were among 'epithelial', 'endothelial' and 'malignant', which is to be expected as nasopharyngeal

carcinomas are endothelial tumors, making these types hard to strictly separate. All the annotations were eventually be mapped back into the original datasets (Figure 3E, Supplementary Figure S5A, B), and convincingly annotated clusters that can be seen in exploratory data analysis. This notably allowed the identification of a very small subpopulation of dendritic cells notably characterized by the expression of CCR7 and CCLE9A genes as well as proliferating T lymphocytes, expressing high levels of proliferation markers like MKI67 and PCNA. These subpopulations were too rare in each dataset to form a distinct cluster, which explains why they could not be annotated initially. It is to note that the CD4 gene was not highly variable within all datasets and therefore it was missing in the integrated gene space. We validated the LT-CD4+ cluster by checking the CD4 expression in datasets in which the gene is present (Supplementary Figure S4). This application shows how the output of *transmorph* gene space models can be used to improve cell type annotations by integrating several datasets directly in gene space.

Transferring cell cycle phase annotations across osteosarcoma and Ewing sarcoma datasets

Cell cycle is one of the most fundamental biological processes through which biological cells grow and divide, but is yet to be fully understood. Single-cell transcriptomics offers great insight into its properties and dynamics, as gene expression regulation is a key factor for cell cycle progression. Gene expression modulation during the cell cycle can be visualized and interpreted by looking at the so-called cell cycle plots. In these plots, each cell is reduced to a small set of coordinates (typically between 2 and 4 (32)), each of those corresponding to the average transcription activity of genes associated with a specific cell cycle signal (e.g. G1/S phase, G2/M phase, histones) (Figure 4A). In this configuration, cells revolve along a one-dimensional cyclic trajectory throughout their progression in the cell cycle. Studying the geometry of these trajectories and cell distribution along them can provide exquisite insight into cell cycle speed, cell growth, or even eventual cell cycle arrest.

A challenging question when studying the cell cycle at the single-cell level is the automatic annotation of cells with cell cycle phases. Some phases like mitosis can be accurately identified by looking at markers such as the total number of raw counts which drops by a factor of two after cell division, but other phases are fuzzier, especially for lower-quality datasets, or fast-cycling cell types. Annotation of scRNA-seq data with cell cycle phases was studied experimentally in (26), where the authors used genetic constructs to follow the abundance of key cell cycle proteins which they can then relate to cell cycle phases, but doing so comes with important costs and experimenter time; a natural idea would be to transfer labels from datasets annotated using this methodology to other unlabeled ones. Unfortunately, this is not as easy as it seems: differences in preprocessing, cell types, and cell cycle properties can quite drastically affect a dataset topology and geometry, making many proximity-based methods irrelevant. A natural label transfer strategy can be pictured as follows (Figure 4B). First, we carry out data integration of all datasets into a common embedding space. Then, we predict cell cycle labels of unlabeled datasets in this common space using a supervised learning approach. Finally, the learned labels can be transferred back to the original representations to be interpreted.

In this experiment, we seek to automatically annotate three single-cell RNA-seq Ewing sarcoma datasets (CHLA9, CHLA10, and TC71) gathered from (25) (Figure 4A, Supplementary Figure S1C). To do so, we transfer the cell cycle phase using the author-provided annotations contained in an osteosarcoma dataset (U2OS) gathered from (26), onto the three Ewing sarcoma datasets. Preprocessing differences, geometrical specificities and apparent S/G2M label mixing within the U2OS reference dataset are tough difficulties to overcome both for integration and label transfer methods. We first perform the integration using BKNNCorrection, setting CHLA10 as the reference dataset considering its good quality and representativity (cells are scattered uniformly around the trajectory, and the central 'hole' is well resolved). Unfortunately, predicted cell cycle labels are not satisfying (Supplementary Figure S6): post-mitotic cells are associated with the G2/M label, S-phase is labeled too late on the trajectory, and some early G1 cells are labeled as S. This disappointing performance may be caused by a lack of orthogonality between cell cycle factors and batch effects. This is a crucial hypothesis for all neighbors-based dataset integration, not satisfying it results in a poor matching quality making integration unreliable.

This motivates the need to seek a more appropriate matching algorithm for this situation. We choose here a transportation-based matching, which is robust for applications where information is contained in data topology. It relies on discrete optimal transport that has been brought into the scRNA-seq field a few years ago in (17), which can be pictured as looking for the most economical way to move mass in a metric space from a point cloud onto another. This class of problems yields a natural and harmonious way to match cells across batches, by operating at the dataset level instead of operating at the cell level like in MNN. We can use the *transmorph* pre-built model TransportCorrection inspired from SCOT (9) and Pamona (10), which consists of a few preprocessing steps followed by a transport-based matching, used to project every query item onto the barycenter of its matches (Figure 4C). In this case, we had to use the unbalanced formulation of optimal transport (16,33) to account for cell cycle phase imbalance between 'standard' and 'fast' cell cycle datasets; this variant is also implemented in our framework. Label transfer using this model instead of BKNNCorrection yields much better labeling, entirely interpretable and in line with the patterns we expect for the 'standard' and 'fast' cell cycle (Figure 4D, Supplementary Figure S7A, B). We see mitosis point is now well identified by the automatic annotation, and S-phase labels are better located. Differential gene expression between the different identified labels yields well-known cell cycle genes specific to each phase, showing accurate annotation (Figure 4E, Supplementary Figure S7C). Among these genes we notably see a few well-known ones appear in all profiles such as the TOP2A gene which is associated with the G2/M phases, PCNA with the S phase, and CDC20 with the G1 phase. Therefore in this scenario, the transportation-based matching was clearly better suited

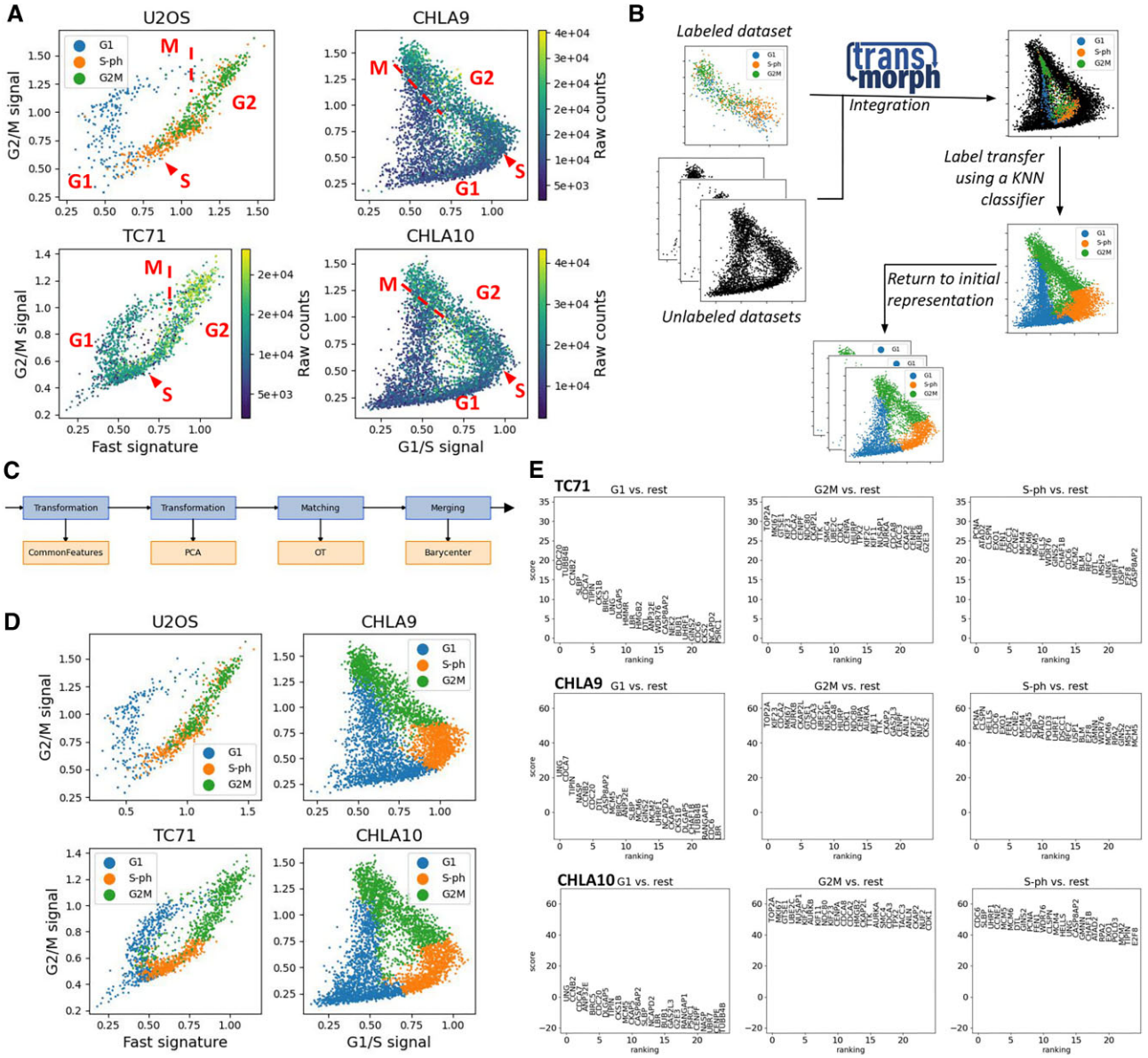


Figure 4. Transferring cell cycle phase annotations between osteosarcoma (U2OS, TC71) and Ewing sarcoma (CHLA9, CHLA10) scRNA-seq datasets. (**A**) Visualizing the cell cycle loop of each dataset, approximate positions of cell cycle phases are annotated. U2OS annotations are provided by the authors, other datasets are colored according to the number of read counts. (**B**) Schematic strategy for the data integration-based label transfer. (**C**) Architecture of the TransportCorrection pre-built integration model performing integration in gene space. Computational modules are executed from left to right and from top to bottom. (**D**) Automatically transferred annotations using the TransportCorrection model. (**E**) Differential gene expression was performed using a Wilcoxon rank-sum test, showing the most specific genes associated with cells of each label.

than the nearest neighbors-based one and allowed an accurate cell cycle label transfer. This shows how important choosing the right matching can be, and how *transmorph* addresses it.

DISCUSSION

Horizontal data integration and batch effect correction are key computational challenges, especially in computational biology to be able to properly analyze single-cell data from different batches or patients (1). We identified the need for modular methods to tackle this problem, and demonstrated the necessity to carefully combine trustworthy cell-cell similarity algorithms with relevant embedding algorithms. We also clearly showed how deceiving data integration can be when carried out improperly, which can be extremely detrimental to subsequent analyses. This alone motivates the need for more modular tools, where every algorithmic step can be controlled if necessary. To address this need and instead of introducing yet another data integration technique we present *transmorph*, a novel modular computational framework for data integration, implemented as an open-source *python* library. We provided a robust implementation for it and demonstrated its value through various real-life applications both in terms of efficiency, quality and versatility. We would like to highlight that EmbedMNN and

TransportCorrection models represent original and previously not proposed combinations of base algorithms that were connected into complete data integration methods, using *transmorph* as a toolbox for fast building and testing of data integration models. Furthermore, these pre-built models can easily be transformed into to a combinatorial number of alternative models by changing their constructor parameters (preprocessing steps, matching type, optimal transport flavor, supervised or unsupervised behavior, gene space output, or linear subspace output).

If *transmorph* is an expressive data integration framework that provides a way to articulate multiple algorithmic modules together in order to shape data integration pipelines, there still exists some expressiveness limitations to overcome. In particular, if trained deep learning models such as deep autoencoders (DAE) can be used as custom *transformation* modules, *transmorph* does not provide a way to either train or fine-tune them without relying on external libraries. For this reason, we think it is useful to mention the development of some recent DAE-based data integration algorithms, that use different approaches to couple several algorithmic paradigms such as Uniport (34) and MATHCLOT (35) that combine DAE and optimal transport, or SMILE (36) that replaces the decoder part by an information-based evaluator. Even if these different tools do not provide as much modularity as *transmorph* to deal with very different biological applications of horizontal data integration, they are certainly better suited for cases necessitating higher levels of abstraction such as cross-modality (vertical, diagonal, and mosaic) data integration.

We provide via *transmorph* several pre-built integration models ready to be used in daily workflows, with high efficiency and integration quality. For more advanced and specific applications, our framework also allows building integration models from scratch by combining a variety of algorithmic modules, all of which are implemented and optimized inside our library. We eventually provide complete interfaces which allow users to implement their own computational modules if they need to. All this is endowed with a rich software ecosystem including benchmarking datasets, integration metrics, monitoring, and plotting tools as well as interfaces with other state-of-the-art data integration tools like Harmony (3) and scvi (37).

We plan to continue maintaining *transmorph* in the future, in order to keep it up to speed with the ever-growing field of data integration methods. We will continue expanding it with new algorithms, either already existing or to come. We also would also like to add more support for vertical and diagonal integration, as for now the only diagonal matching is based on Gromov-Wasserstein which has an impractical computational time scaling to the size of current data integration problems. For instance, we plan to use gene space transformation to deal with specific vertical integration cases such as integration between RNA-seq and ATAC-seq data. We would eventually like to add domain adaptation methods to our framework (for instance by including supervised PCA (38) or domain adaptation PCA (39) to our preprocessing steps), in order to tighten the bridge towards this growing research field which presents many similarities with data integration.

There are still crucial questions to be answered in order to provide trustable data integration methods, especially in single-cell biology. Among these questions are the definition of relevant metrics to measure dissimilarity between cells (even more importantly across different domains), the research of sound and unbiased ways to measure integration quality, and the necessity to continue to carry out exhaustive benchmarks to identify the most appropriate data integration methods and algorithms for a given use case.

DATA AVAILABILITY

transmorph framework is available at https://github.com/Risitop/transmorph (permanent doi:10.5281/zenodo.8081763), and can also be downloaded from the PyPi repository (version 0.2.6 at the time of writing). Datasets can be directly downloaded from the package, and scripts to generate figures can be found on the package's GitHub, in the 'reproducibility' folder.

SUPPLEMENTARY DATA

Supplementary Data are available at NARGAB Online.

ACKNOWLEDGEMENTS

We thank Alexander Chervov (Institut Curie, U900) for his suggestions on how to process TC71 and U2OS fast cell cycle datasets, Marianyela Petrizzelli (former Institut Curie, U900) for her help in testing *transmorph* on non-Linux systems, Jane Merlevede (Institut Curie, U900) for testing *transmorph* at an early stage, Vicent Noël (Institut Curie, U900) for his suggestions on how to properly package and test *transmorph* and in alphabetical order Jonathan Bac (Institut Curie, U900), Nicolas Captier (Institut Curie, U900), Marco Ruscone (Institut Curie, U900) and Julien Vibert (former Institut Curie, U830) for the many insightful discussions and suggestions around this project.
Author contributions: A.F., L.C. and A.Z. conceptualized the package design and the computational study. A.F. and L.C. developed the package, A.F. carried out the applications, A.F, L.C., O.D. and A.Z wrote the manuscript, A.Z. and O.D. directed the project.

FUNDING

French government under the management of Agence Nationale de la Recherche as part of the 'Investissements d'avenir' program [ANR-19-P3IA-0001] (PRAIRIE 3IA Institute); European Union's Horizon 2020 program [826121, iPC project].
Conflict of interest statement. A.Z. is currently an employee of Evotec (SA) France.

REFERENCES

1. Argelaguet,R., Cuomo,A.S.E., Stegle,O. and Marioni,J.C. (2021) In: *Computational Principles and Challenges in Single-cell Data Integration*. Nature Publishing Group.
2. Hao,Y., Hao,S., Andersen-Nissen,E., Mauck,W.M. III, Zheng,S., Butler,A., Lee,M.J., Wilk,A.J., Darby,C., Zager,M. *et al.* (2021) Integrated analysis of multimodal single-cell data. *Cell*, **184**, 3573–3587.

3. Korsunsky,I., Millard,N., Fan,J., Slowikowski,K., Zhang,F., Wei,K., Baglaenko,Y., Brenner,M., Loh,P.-R. and Raychaudhuri,S. (2019) Fast, sensitive and accurate integration of single-cell data with Harmony. *Nat. Methods*, **16**, 1289–1296.
4. Haghverdi,L., Lun,A.T., Morgan,M.D. and Marioni,J.C. (2018) Batch effects in single-cell RNA-sequencing data are corrected by matching mutual nearest neighbors. *Nat. Biotech.*, **36**, 421–427.
5. Polański,K., Young,M.D., Miao,Z., Meyer,K.B., Teichmann,S.A. and Park,J.-E. (2020) BBKNN: fast batch alignment of single cell transcriptomes. *Bioinformatics*, **36**, 964–965.
6. Angelidis,I., Simon,L.M., Fernandez,I.E., Strunz,M., Mayr,C.H., Greiffo,F.R., Tsitsiridis,G., Ansari,M., Graf,E., Strom,T.-M. *et al.* (2019) An atlas of the aging lung mapped by single cell transcriptomics and deep tissue proteomics. *Nat. Commun.*, **10**, 963.
7. Barkas,N., Petukhov,V., Nikolaeva,D., Lozinsky,Y., Demharter,S., Khodosevich,K. and Kharchenko,P.V. (2019) Joint analysis of heterogeneous single-cell RNA-seq dataset collections. *Nat. Methods*, **16**, 695–698.
8. Butler,A., Hoffman,P., Smibert,P., Papalexi,E. and Satija,R. (2018) Integrating single-cell transcriptomic data across different conditions, technologies and species. *Nat. Biotech.*, **36**, 411–420.
9. Demetci,P., Santorella,R., Sandstede,B., Noble,W.S. and Singh,R. (2022) Scot: single-cell multi-omics alignment with optimal transport. *J. Comput. Biol.*, **29**, 3–18.
10. Cao,K., Hong,Y. and Wan,L. (2022) Manifold alignment for heterogeneous single-cell multi-omics data integration using Pamona. *Bioinformatics*, **38**, 211–219.
11. Lopez,R., Regier,J., Cole,M.B., Jordan,M.I. and Yosef,N. (2018) Deep generative modeling for single-cell transcriptomics. *Nat. Methods*, **15**, 1053–1058.
12. Luecken,M.D., Büttner,M., Chaichoompu,K., Danese,A., Interlandi,M., Müller,M.F., Strobl,D.C., Zappia,L., Dugas,M., Colomé-Tatché,M. *et al.* (2022) Benchmarking atlas-level data integration in single-cell genomics. *Nat. Methods*, **19**, 41–50.
13. Wolf,F.A., Angerer,P. and Theis,F.J. (2018) SCANPY: large-scale single-cell gene expression data analysis. *Genome Biol.*, **19**, 15.
14. Becht,E., McInnes,L., Healy,J., Dutertre,C.-A., Kwok,I.W., Ng,L.G., Ginhoux,F. and Newell,E.W. (2019) Dimensionality reduction for visualizing single-cell data using UMAP. *Nat. Biotech.*, **37**, 38–44.
15. Agrawal,A., Ali,A., Boyd,S. and othersothers (2021) Minimum-distortion embedding. *Found. Trends Mach. Learn.*, **14**, 211–378.
16. Peyré,G., Cuturi,M. and othersothers (2019) Computational optimal transport with applications to data science. *Found. Trends Mach. Learn.*, **11**, 355–607.
17. Schiebinger,G., Shu,J., Tabaka,M., Cleary,B., Subramanian,V., Solomon,A., Gould,J., Liu,S., Lin,S., Berube,P. *et al.* (2019) Optimal-transport analysis of single-cell gene expression identifies developmental trajectories in reprogramming. *Cell*, **176**, 928–943.
18. Sompairac,N., Nazarov,P.V., Czerwinska,U., Cantini,L., Biton,A., Molkenov,A., Zhumadilov,Z., Barillot,E., Radvanyi,F., Gorban,A. *et al.* (2019) Independent component analysis for unraveling the complexity of cancer omics datasets. *Int. J. Mol. Sci.*, **20**, 4414.
19. Kairov,U., Cantini,L., Greco,A., Molkenov,A., Czerwinska,U., Barillot,E. and Zinovyev,A. (2017) Determining the optimal number of independent components for reproducible transcriptomic data analysis. *BMC Genomics*, **18**, 712.
20. Hyvarinen,A. (1999) Fast and robust fixed-point algorithms for independent component analysis. *IEEE T. Neur. Networ.*, **10**, 626–634.
21. Captier,N., Merlevede,J., Molkenov,A., Seisenova,A., Zhubanchaliyev,A., Nazarov,P.V., Barillot,E., Kairov,U. and Zinovyev,A. (2022) BIODICA: a computational environment for Independent Component Analysis of omics data. *Bioinformatics*, **38**, 2963–2964.
22. Cantini,L., Kairov,U., de Reyniès,A., Barillot,E., Radvanyi,F. and Zinovyev,A. (2019) Assessing reproducibility of matrix factorization methods in independent transcriptomes. *Bioinformatics*, **35**, 4307–4313.
23. Zhou,Y., Yang,D., Yang,Q., Lv,X., Huang,W., Zhou,Z., Wang,Y., Zhang,Z., Yuan,T., Ding,X. *et al.* (2020) Single-cell RNA landscape of intratumoral heterogeneity and immunosuppressive microenvironment in advanced osteosarcoma. *Nat. Commun.*, **11**, 6322.
24. Chen,S., Lake,B.B. and Zhang,K. (2019) High-throughput sequencing of the transcriptome and chromatin accessibility in the same cell. *Nat. Biotech.*, **37**, 1452–1457.
25. Miller,H.E., Gorthi,A., Bassani,N., Lawrence,L.A., Iskra,B.S. and Bishop,A.J. (2020) Reconstruction of Ewing sarcoma developmental context from mass-scale transcriptomics reveals characteristics of EWSR1-FLI1 permissibility. *Cancers*, **12**, 948.
26. Mahdessian,D., Cesnik,A.J., Gnann,C., Danielsson,F., Stenström,L., Arif,M., Zhang,C., Le,T., Johansson,F., Shutten,R. *et al.* (2021) Spatiotemporal dissection of the cell cycle with single-cell proteogenomics. *Nature*, **590**, 649–654.
27. Tirosh,I., Izar,B., Prakadan,S.M., Wadsworth,M.H., Treacy,D., Trombetta,J.J., Rotem,A., Rodman,C., Lian,C., Murphy,G. *et al.* (2016) Dissecting the multicellular ecosystem of metastatic melanoma by single-cell RNA-seq. *Science*, **352**, 189–196.
28. Blondel,V.D., Guillaume,J.-L., Lambiotte,R. and Lefebvre,E. (2008) Fast unfolding of communities in large networks. *J. Stat. Mech. Theor. Exp.*, **2008**, P10008.
29. Traag,V.A., Waltman,L. and Van Eck,N.J. (2019) From Louvain to Leiden: guaranteeing well-connected communities. *Sci. Rep.*, **9**, 5233.
30. Chen,Y.-P., Yin,J.-H., Li,W.-F., Li,H.-J., Chen,D.-P., Zhang,C.-J., Lv,J.-W., Wang,Y.-Q., Li,X.-M., Li,J.-Y. *et al.* (2020) Single-cell transcriptomics reveals regulators underlying immune cell diversity and immune subtypes associated with prognosis in nasopharyngeal carcinoma. *Cell Res.*, **30**, 1024–1042.
31. Franzén,O., Gan,L.-M. and Björkegren,J.L. (2019) PanglaoDB: a web server for exploration of mouse and human single-cell RNA sequencing data. *Database*, **2019**, baz046.
32. Zinovyev,A., Sadovsky,M., Calzone,L., Fouché,A., Groeneveld,C.S., Chervov,A., Barillot,E. and Gorban,A.N. (2021) Modeling progression of single cell populations through the cell cycle as a sequence of switches. *Front. Mol. Biosci.*, **8**, 793912.
33. Liero,M., Mielke,A. and Savaré,G. (2018) Optimal entropy-transport problems and a new Hellinger–Kantorovich distance between positive measures. *Invent. Math.*, **211**, 969–1117.
34. Cao,K., Gong,Q., Hong,Y. and Wan,L. (2022) A unified computational framework for single-cell data integration with optimal transport. *Nat. Commun.*, **13**, 7419.
35. Gossi,F., Pati,P., Chouvardas,P., Martinelli,A.L., Kruithof-de Julio,M. and Rapsomaniki,M.A. (2023) Matching single cells across modalities with contrastive learning and optimal transport. *Brief. Bioinform.*, **24**, bbad130.
36. Xu,Y., Das,P. and McCord,R.P. (2022) SMILE: mutual information learning for integration of single-cell omics data. *Bioinformatics*, **38**, 476–486.
37. Gayoso,A., Lopez,R., Xing,G., Boyeau,P., Valiollah Pour Amiri,V., Hong,J., Wu,K., Jayasuriya,M., Mehlman,E., Langevin,M. *et al.* (2022) A Python library for probabilistic analysis of single-cell omics data. *Nat. Biotech.*, **40**, 163–166.
38. Barshan,E., Ghodsi,A., Azimifar,Z. and Jahromi,M.Z. (2011) Supervised principal component analysis: visualization, classification and regression on subspaces and submanifolds. *Patt. Recog.*, **44**, 1357–1371.
39. Mirkes,E.M., Bac,J., Fouché,A., Stasenko,S.V., Zinovyev,A. and Gorban,A.N. (2022) Domain adaptation principal component analysis: base linear method for learning with out-of-distribution data. *Entropy*, **25**, 33.