DINCTA: DATA INTEGRATION AND CELL TYPE ANNOTATION OF SINGLE CELL TRANSCRIPTOMES

Songting Shi

Department of Scientific and Engineering Computing
School of Mathematical Sciences
Peking University
Beijing 300071, P. R. China
songtingstone@gmail.com

September 29, 2020

ABSTRACT

We proposed a method for data integration and cell type annotation (Dincta) of single cell transcriptomes in a unify framework. The Dincta can handle three cases. In the first case, the data has been annotated the cell type for all cells, Dincta can integrate the the data into a common low dimension embedding space such that cells with different cell types separate while cells from the different batches but in the same cell type cluster together. In the second case, the data was only annotated for part of cells, such as one sample, Dincta can integrate the data into a common low dimension embedding space such that cells with different cell types separate while cells from the different batches but in the same cell type cluster together. Moreover, it can infer the known or novel cell type of the cells with unknown cell type initially. In the third case, there are no cell type information of cells, we can run Dincta in an unsupervised way. It can infer the number of new cell types and annotate the cells into its correspond cell type, and do data integration keeping cells from different cell type separate while removing the batch effects to mix cells in the same cell type. Dincta is simple, accurate and efficient to integrate data, which keeps the cell type information preserved while removes the batch effects, and infers the known or novel cell types of cells.

Keywords BatchEffect · GeneExpression · RNASeq · SingleCell · Transcriptomics · Cell Annotation

1 Introduction

As the sequencing technology updates rapidly, we can sequence thousands of the cells in the gene expression levels in a short time. Now, there are huge mounts of single cell RNA-seq data generated and need to be analyzed. However, there are obstacles which hinder to analyze the scRNA-seq directly. The batch effects in the data which come from different samples, technologies or platforms hide the true biological interest variables. The researchers endeavor to solve this problem, and there emerges a bunch of tools for this task, such as MNN Correct (Lun et al., 2016), Seurat multiCCA (Butler et al., 2018), Scanorama (Hie et al., 2019), BBKNN (Polanski et al., 2019), Harmony (Korsunsky et al., 2018) and so on.

Harmony is one of the impressive tools to remove batch effects. It first projects the scRNA-seq raw counts data into a low dimension space (e.g. PCA). Secondly, it uses batch membership and cluster membership independent penalty to modify the soft k -means objective, which produces the soft cluster assignments. Third, it uses the the soft cluster assignments to do the linear correction of embedding, such that each cluster contains cells from different batch match to the batch frequency of the whole data. It iterates until the cluster assignments get stable.

There is a problem in Harmony. It uses a KL divergence loss to make that the batch membership is independent of the cluster membership. In the case that some cell type only occurs in part of samples (batches), this will case a problem. Since Harmony encourages that each cluster (subtype) consists of all batches. But in this case, the cluster should only contain cells in the batch which it contains the cell type. Harmony has a trend over correcting the batch effects.

DINCTA: DATA INTEGRATION AND CELL TYPE ANNOTATION OF SINGLE CELL TRANSCRIPTOMES

If we know the cell type information of cells, this problem can be solved. We can constraint the penalty which only mixes cells in one cell type from different batches which contain the cell type, and do not mix cells from different cell types. That is the essential idea of Dincta. Here, we must to determine the cell type of cells. Note that in the k -means, each cluster is the surrogate of the cell types. As we reveal latter, the cell type assignment and cluster assignment can connect to each other via a probability relationship. The number of cell types can be determined by the rank of the cluster assignment matrix and we can construct the cell type assignment from the cluster assignment. Once getting the cell type information of the cells, we can move the low dimensional embedding of the cells such the embedding of cells belong to the same cell type will close to each other even they come from different batches, and the embedding of cells belong to different cell types separate a lot. As the embeddings of cells move toward the correct position, we will get more accurate cluster assignments and cell type assignments. And with the enhanced cluster assignments and cell type assignments, it will result in a more successful data integration. This build a positive enhance loop to make the Dincta converge to the final stable solution, i.e. correct cell type assignments and the data integration which mixes cells in the same cell type and keeps cells from different cell types separate.

2 Results

2.1 Dincta Mechanism

Dincta starts from the inputs of low dimensional embeddings (usually PCA embeddings) and the cell type information of the cells which have been annotated. It first run the constrained soft k -means with the batch mixing penalty and cell type preserved penalty, to get the cluster assignment probability for each cluster. In the initial state, the cluster may be separated by the batches. With the batch mixing penalty, it will adjust the cluster assignment probability for each cell such that the batch frequency of cells in this cluster approximates the batch frequency of cells in the cell type which the cluster belongs to. With the cell type penalty, Dincta will adjust the cluster assignments to match the cell type assignments. Once we get the cluster assignment matrix, Dincta will infer the known or novel cell type of cells, which is coded in the cell type assignment matrix. After we get the batch mixing and cell type preserved adjusted cluster assignment matrix, Dincta use it to correct the embedding, such that the embedding of cells belong to the same cell type move to each other with a step, but the cells with different cell types keep separate. With the moved embeddings, it rerun the next time soft k -means clustering with the batch mixing and cell type preserved penalty. And do the novel cell type inferring if need, cell type assignment updating, embedding correcting, until finally the cluster assignments and cell type assignments get stable. After that, there is an option whether to refine the cell type assignment or not. If yes, Dincta will compute the fitness between the cluster assignments and cell type assignments for each cell with unknown cell type initially. If the cluster assignments and cell type assignments of a cell are not match to each other, then Dincta will mark this cell with unknown cell type; if they match to each other, Dincta will mark that cell with known cell type. After that, Dincta re-iterate the clustering, inferring and embedding correcting procedures described above until getting the stable and matchable cluster assignments and cell type assignments.

2.2 Performance Metric

We use three metrics to assess the results of Dincta. The first is the LISI (local inverse Simpson's Index) metric (Korsunsky et al., 2018). It consists of cLSI (cell-type LISI) and iLSI (integration LISI). The cLSI is defined for each cell, it roughly measures the number of cell types in the local neighborhood of this cell. In the ideal case, we expect that cLSI approaches 1. The iLSI is also defined for each cell, it roughly measures the number of batches (samples, datasets, technologies) in a neighborhood of this cell. If there are two batches, and each batch consists of the same cell types, we expect the iLSI approaches 2.

The second metric is the match score (see in Methods **match score**) between the cluster assignments and the cell type assignments of cells. It was defined by the KL divergence of the cell type assignments and the cell type assignments which are transformed from the cluster assignments. The lower the value of match score, the fitness is better between the cluster assignments and cell type assignments.

The third metric is the inferring accuracy (see in Methods Algorithm 8). And there are two types of inferring accuracy. One type is that the cell type assignments come from the cluster assignments directly (see Methods Algorithm 7) and comparing this cell type assignments with the golden cell type assignments of cells. The accuracy is determined by that the number of cells whose cell type were correctly inferred divided by the total number of cells (see Methods Algorithm 8). In the following tables, we denote this kind of inferring accuracy by the method name with suffix (with R). The another one is that cell type assignments come from the Dincta infer function (Algorithm 4) and comparing this cell type assignments with the golden cell type assignments of cells. In the ideal case, the inferring accuracy should approach 1.

DINCTA: DATA INTEGRATION AND CELL TYPE ANNOTATION OF SINGLE CELL TRANSCRIPTOMES

Table 1: LISI Scores of Cell-Line Data

Name	cLISI mean (std)	iLISI mean (std)
PCA	1.004 (0.049)	1.152 (0.262)
Harmony	1.006 (0.062)	1.694 (0.271)
Dincta (Whole Known)	1.001 (0.025)	1.694 (0.265)
Dincta (Partial Unknown)	1.003 (0.046)	1.694 (0.265)
Dincta (Whole Unknown)	1.003 (0.046)	1.693 (0.267)

Table 2: Match Score between Cell Type Assignment and Cluster Assignment of Cell-Line Data

Name	Match Score
Harmony	131.8
Dincta (Whole Known)	35.5
Dincta (Partial Unknown)	30.6
Dincta (Whole Unknown)	31.0

In the all experiments below, we use the cell type annotated in Korsunsky et al. (2018) as golden labels of cells.

Besides the three metrics above, we also plot the density of the cLISI and iLISI to visually sense the behaviors of the algorithms. In order to visually check the results of cell type preserving and batch-diverse mixing, we plot the UMAP of the PCA embedding, Harmony embedding and Dincta embedding.

2.3 Quantifying Performance in Cell-Line Data.

We first run Dincta on the cell-line data to check that it can do the correct jobs. The cell-line data consists of three samples. The first two samples: Jurkat and T293 consist of cells with the pure Jurkat cell type and pure T293 cell type, respectively. The third sample 'half' consists of cells with 50/50 mixing of two kinds of cell types: Jurkat and T293. And the cell type of cells can be labeled Jurkat or T293 unambiguously.

After pooling all the cells together, we performed a joint analysis. Before integration, cells group primarily by dataset (see Figure 1 (row at PCA), Table 1 (row at PCA) and Figure 2)

We perform three experiments on the cell-line data. In the first experiment, we use the whole cell annotation of cells annotated in the Korsunsky et al. (2018). We feed these cell type information and the PCA embedding to the Dincta. After Dincta, we compare the integration results coming from the Harmony (Figure 3) with Dincta (Figure 4). The UMAP figures of Dincta and Harmony show that both algorithms get the similar results. Once zooming in the cell type UMAP figure of Harmony, we can find several misclassified cells in the center of cells with Jurkat cell type. Dincta also misclassified few cells but the number of misclassified cells is less than Harmony's. The inferring accuracy table (Table 3 (rows at Dincta (Whole Known with R) and Harmony (with R)) also verifies this observation. We see that Dincta perform better than Harmony in the match score as show in table 2 (rows at Harmony and Dincta (Whole Known)). From the LISI results (Table 1 (rows at Dincta (Whole Known with R) and Harmony) and Figure 1 (row as dincta_w_k)), Dincta has lower mean value of cLISI which show that it preserves the cell type separate better than Harmony while keeps the similar iLISI values. This indicates that the Harmony over corrects the batch effects a little and mixes a few cells with different cell types.

Table 3: Inferring Accuracy of Cell-Line Data

Name	accuracy	number of infer correct cells	total number of cells
Harmony (with R)	0.9942	9423	9478
Dincta (Whole Known with R)	0.9994	9472	9478
Dincta (Partial Unknown)	0.9979	9458	9478
Dincta (Partial Unknown with R)	0.9979	9458	9478
Dincta (Whole Unknown)	0.9979	9458	9478
Dincta (Whole Unknown with R)	0.9978	9457	9478

DINCTA: DATA INTEGRATION AND CELL TYPE ANNOTATION OF SINGLE CELL TRANSCRIPTOMES

Figure 1: LISI density of cell-line data

Figure 2: PCA results of cell-line data.

DINCTA: DATA INTEGRATION AND CELL TYPE ANNOTATION OF SINGLE CELL TRANSCRIPTOMES

Figure 3: Harmony results of cell-line data.

Figure 4: Dincta (with whole cell type information as input) results of cell-line data.

DINCTA: DATA INTEGRATION AND CELL TYPE ANNOTATION OF SINGLE CELL TRANSCRIPTOMES

Figure 5: Dincta (with partial cell type information) results of cell-line data. The top left and top middle figures are the UMAP of the PCA embedding, colored by the partial cell types (`fake_cell_type`) of cells which is cell type information input to Dincta and colored by the inferred cell type information (`cell_type_predict`), respectively. The rest four figures are the UMAP of the Dincta embedding colored by dataset, `cell_type`, `fake_cell_type` and `cell_type_predict`, respectively.

In the second experiment, we mask the cell type information of cells in half and t293 dataset to the "unknown" cell type, only provide the cell type information of cells in jerkat dataset. We feed the partial cell type information (`fake_cell_type`) and the PCA embedding to the Dincta. After Dincta, we plot the UMAP of Dincta embedding as showed in Figure 5). We see that Dincta performs similarly as we feed all the cell type information. Dincta infers the masked "T293" cell type as the "new_cell_type_1". And it assigns the cell type of cells with the masked "unknown" cell type initially almost correctly, which can be checked from the Table 3 (rows at Dincta (Partial Unknown) and Dincta (Partial Unknown with R)). From Table 1 (row at Dincta (Partial Unknown)) and Figure 1 (row at Dincta_p_u), we find Dincta also performs a little better than results of the Harmony. This experiment reveals a road to do the cell type annotation in practice. First, we run a cluster algorithm (e.g. DinctaClassification) on one sample, and use the marker genes to determine the cell type of cells. Then when a new sample comes, we can run the Dincta to infer the known or novel cell type of cells in the new sample.

In the third experiment, we mask the cell type of all cells to the "unknown" cell type. After Dincta, it infers that there are 2 cell types in the data. From the figure 6, we can easily find that "new_cell_type_1" is the "T293" cell type, and "new_cell_type_2" is the "Jurkat" cell type. Dincta achieves almost the same cell type assignments as the cell type of cells which were annotated separately in each dataset by the marker genes. Comparing the inferring accuracy, we see that Dincta achieves very high Inferring Accuracy with only a little performance drop (Table 3(rows at Dincta (Whole Unknown) and Dincta (Whole Unknown with R))). From Table 2 (row at Dincta (Whole Unknown)), we see that the cluster assignments and cell type assignments also fit. This experiment shows that Dincta is robust even under all the cell type information is masked out.

2.4 Quantifying Performance in PBMC data.

To assess how Dincta might perform under more challenging scenarios, we gathered three dataset of human PBMCs, each assayed with the Chromium 10X platform but prepared with different protocols: 3' and v1 (3pV1), 3' end v2 (3pV2) and 5' end (5p) chemistries. To make the task harder, we remove the cells with "mono" cell type in protocol 5p, remove the cells with "tcells" cell type in protocol 3pV1 and remove the cells with "bcells" cell type in protocol 3pV2. The composition of cell types for the three datasets tabulates in Table 4

DINCTA: DATA INTEGRATION AND CELL TYPE ANNOTATION OF SINGLE CELL TRANSCRIPTOMES

Figure 6: Dincta (with no cell type information) results of cell-line data. The top left and top middle figures are the UMAP of the PCA embedding, colored by the no cell type information (unknown) (fake_cell_type) which is cell type information input to the Dincta and colored by the inferred cell type information (cell_type_predict), respectively. The rest four figures is the UMAP of the Dincta embedding colored by dataset, cell_type, fake_cell_type and cell_type_predict, respectively.

Table 4: Cell Type Composition of PBMCs

Protocol	bcells	dc	mk	mono	nk	tcells
fivePrime (5p)	76	1911	107	36	0	310
threefresh (3pv2)	86	0	43	43	2042	315
threepv1 (3pv1)	33	586	69	20	1200	297

After pooling all the cells together, we performed a joint analysis. Before integration, cells group primarily by dataset (see Figure 7 (row at PCA), Table 5 (row at PCA) and Figure 8).

We perform three experiments on the PBMCs data. In the first experiment, we use the whole cell annotation of cells annotated in the Korsunsky et al. (2018). We feed these cell type information and the PCA embedding to the Dincta. After Dincta, we compare the integration results coming from the Harmony (Figure 9) with Dincta (Figure 10). Once zooming in Figure 9, we find that in the cluster of "bcells", there are some cells from protocol 3pv2. This should not happen, since the protocol 3pv2 don't own cells with "bcells" cell type. The similar phenomena occur in the clusters of cell type "mono" and "tcells". It means that Harmony over corrects the batch effects and mixes a small part of cells with different cell types. We zoom in Figure 10, Dincta does a better job. In the cluster of "bcells", it consists of cells from

Table 5: LISI Scores of PBMC data

Name	cLISI mean (std)	iLISI mean (std)
PCA	1.017 (0.112)	1.015 (0.110)
Harmony	1.042 (0.161)	1.682 (0.345)
Dincta (Whole Known)	1.014 (0.103)	1.640 (0.389)
Dincta (Partial Unknown)	1.017 (0.107)	1.662 (0.378)
Dincta (Whole Unknown)	1.034 (0.141)	1.646 (0.355)

DINCTA: DATA INTEGRATION AND CELL TYPE ANNOTATION OF SINGLE CELL TRANSCRIPTOMES

Table 6: Match Score between Cell Type Assignment and Cluster Assignment of PBMCs

Name	Match Score
Harmony	2188.6
Dincta (Whole Known)	84.4
Dincta (Partial Unknown)	650.3
Dincta (Whole Unknown)	1889.5

Table 7: Inferring Accuracy of PBMCs

Name	accuracy	number of infer correct cells	total number of cells
Harmony (with R)	0.8964	12962	14460
Dincta (Whole Known with R)	0.9982	14434	14460
Dincta (Partial Unknown)	0.9539	13794	14460
Dincta (Partial Unknown with R)	0.9049	13086	14460
Dincta (Whole Unknown)	0.9128	13200	14460
Dincta (Whole Unknown with R)	0.9019	13042	14460

Figure 7: LISI of PBMC data.

DINCTA: DATA INTEGRATION AND CELL TYPE ANNOTATION OF SINGLE CELL TRANSCRIPTOMES

Figure 8: PCA results of PBMC data.

Figure 9: Harmony results of PBMC data.

DINCTA: DATA INTEGRATION AND CELL TYPE ANNOTATION OF SINGLE CELL TRANSCRIPTOMES

Figure 10: Dincta (with whole cell type information as input) results of PBMC data.

Figure 11: Dincta (with partial cell type information) results of PBMC data. The top left and top middle figures are the UMAP of the PCA embedding, colored by the partial cell types (fake_cell_type) of cells which is cell type information input to Dincta and colored by the inferred cell type information (cell_type_predict), respectively. The rest four figures are the UMAP of the Dincta embedding colored by donor, cell_type, fake_cell_type and cell_type_predict, respectively.

DINCTA: DATA INTEGRATION AND CELL TYPE ANNOTATION OF SINGLE CELL TRANSCRIPTOMES

Figure 12: Dincta (with no cell type information) results of PBMC data. The top left and top middle figures are the UMAP of the PCA embedding, colored by the no cell type information (unknown) (fake_cell_type) which is cell type information input to the Dincta and colored by the inferred cell type information (cell_type_predict), respectively. The rest four figures is the UMAP of the Dincta embedding colored by donor, cell_type, fake_cell_type and cell_type_predict, respectively.

protocols 5p and 3pv1 majorly, and only few cells comes from protocol 3pv2. The similar phenomena also occur in the cluster of cells with "mono" and "tcells" cell type. This shows that Dincta mixes the cells with the same cell type from different batches (protocols), while keeps the cells with different cell types separate. This can be also verified from Table 5 (rows at Harmony and Dincta (Whole Known)), Figure 7 (rows at dincta_w_k and harmony)), Table 2 (rows at Harmony and Dincta (Whole Known)) and Table 7 (rows at Dincta (Whole Known with R) and Harmony (with R)).

In the second experiment, we mask the the cell type information of cells measured with 5' end (5p) chemistries protocol to the "unknown" cell type. And we also mask the cell type "mono" for all cells as "unknown" cell type. We feed the partial cell type information (fake_cell_type) and the PCA embedding to the Dincta. From the Figure 11), we see that Dincta performs similarly as we feed all the cell type information. Dincta infers the the masked "mono" cell type as the two subtypes "new_cell_type_1" and "new_cell_type_2". And it assigns the cell type of cells with the masked "unknown" cell type initially with high accuracy, which can be checked from the Table 7 (rows at Dincta (Partial Unknown) and Dincta (Partial Unknown with R)). From Table 5 (row at Dincta (Partial Unknown)) and Figure 7 (row at Dincta_p_u), we find Dincta also performs better than Harmony.

In the third experiment, we mask the cell type of all cells to the "unknown" cell type. After Dincta, it infers that there are 9 cell types in the data. As we see in the Figure 12, Dincta achieves similar cell type assignment results as the cell type of cells which were annotated separately in each dataset by the marker genes. There is some mismatch between the golden labels and the predicted cell types. Dincta annotates the the cells with "dc" cell type and part of cells with "mono" cell type to the single "new_cell_type_9". And part of cells with "mono" cell type was annotated to the "new_cell_type_3". Also the cells with the "tcells" cell type are annotated with subtypes "new_cell_type_4", "new_cell_type_5" and "new_cell_type_7". So there still has the improve space of Dincta. Although it has these mistakes, Dincta almost keeps the cells with different inferred new cell types separate. From the cluster of cells with "bcells" cell type, there was almost no cells from the protocol 3pv2, which is better than the results of Harmony. The similar phenomena occur in the clusters of cells with "tcells" and "mono" cell type. From Table 7 (rows at Dincta (Whole Unknown) and Dincta (Whole Unknown with R)), Table 6 (row at Dincta (Whole Unnown)), Dincta achieves a little better results than Harmony's.

DINCTA: DATA INTEGRATION AND CELL TYPE ANNOTATION OF SINGLE CELL TRANSCRIPTOMES

Table 8: LISI Scores of Pancreas Data

Name	cLISI mean (std)	iLISI mean (std) (dataset)	iLISI mean (std) (donor)
PCA	1.061 (0.213)	1.028 (0.147)	1.904 (1.216)
Harmony	1.067 (0.233)	1.832 (0.701)	4.298 (2.082)
Dincta (Whole Known)	1.059 (0.184)	1.781 (0.676)	4.158 (2.043)
Dincta (Partial Unknown)	1.057 (0.174)	1.794 (0.682)	4.203 (2.068)
Dincta (Whole Unknown)	1.068 (0.227)	1.807 (0.692)	4.218 (2.051)

Table 9: Match Score between Cell Type Assignment and Cluster Assignment of Pancreas Data

Name	Match Score
Harmony	2653.5
Dincta (Whole Known)	1264.9
Dincta (Partial Unknown)	1997.8
Dincta (Whole Unknown)	2640.8

2.5 Quantifying Performance in Pancreas Islet Data.

To assess Dincta on a more complex experimental design, we gathered human pancreatic islet cells from three independent studies as described in Korsunsky et al. (2018), each generated with a different technological platform. The batch effects result from both different technologies and the 36 donors used in these studies. A successful integration of these studies must be performed simultaneously over the variations from the different technologies and donors.

After pooling all the cells together, we performed a joint analysis. Before integration, cells group primarily by dataset (see Figure 13 (row at PCA), Table 8 (row at PCA) and Figure 14).

We perform three experiments on the pancreas islet data.

In the first experiment, we use the whole cell type information of cells. We feed the whole cell type information and the PCA embedding to the Dincta. After Dincta, we compare the integration results coming from the Harmony (Figure 15) with Dincta (Figure 16). The cluster of cells with "neural" cell type consists of cells with almost the pure "neural" cell type in the UMAP of Dincta's embedding, while the cluster of cells with "neural" cell type contains several cells with "endothelial" cell type in the UMAP of Harmony's embedding. The similar phenomena also occur in the clusters of cells with other cell types. This shows that Dincta mixes the cells with the same cell type from different batches, while tries to keep the cells with different cell types separate. This can be also verified from Table 8 (rows at Harmony and Dincta (Whole Known)), Figure 13 (rows at dincta_w_k and harmony)), Table 9 (rows at Harmony and Dincta (Whole Known)) and Table 10 (rows at Dincta (Whole Known with R) and Harmony (with R)). In this experiment, Dincta does a better job than Harmony.

In the second experiment, we mask the the cell type information of cells measured with celseq2 to the "unknown" cell type. And we also mask the cell type "alpha" for all cells as "unknown" cell type. After Dincta, we plot the UMAP of Dincta embedding as showed in Figure 17). We see that Dincta performs similarly as we feed all the cell type information. Dincta infers the the masked "alpha" cell type as "new_cell_type_1". And it assigns the cell type of cells with the masked "unknown" cell type initially with high accuracy, which can be checked from the Table 10 (rows at Dincta (Partial Unknown) and Dincta (Partial Unknown with R)). From Table 8 (row at Dincta (Partial Unknown)) and Figure 13 (row at Dincta_p_u), we find Dincta also performs a little better than results of the Harmony.

Table 10: Inferring Accuracy of Pancreas Data

Name	accuracy	number of infer correct cells	total number of cells
Harmony (with R)	0.9616	14180	14746
Dincta (Whole Known with R)	0.9775	14415	14746
Dincta (Partial Unknown)	0.9710	14319	14746
Dincta (Partial Unknown with R)	0.9697	14300	14746
Dincta (Whole Unknown)	0.9622	14189	14746
Dincta (Whole Unknown with R)	0.9622	14189	14746

DINCA: DATA INTEGRATION AND CELL TYPE ANNOTATION OF SINGLE CELL TRANSCRIPTOMES

Figure 13: LISI of pancreas data

Figure 14: PCA results of pancreas data.

Figure 15: Harmony results of pancreas data.

DINCTA: DATA INTEGRATION AND CELL TYPE ANNOTATION OF SINGLE CELL TRANSCRIPTOMES

Figure 16: Dincta (with whole cell type information as input) results of pancreas data.

Figure 17: Dincta (with partial cell type information) results of pancreas data. The top left and top middle figures are the UMAP of the PCA embedding, colored by the partial cell types (fake_cell_type) of cells which is cell type information input to Dincta and colored by the inferred cell type information (cell_type_predict), respectively. The rest four figures are the UMAP of the Dincta embedding colored by dataset, cell_type, fake_cell_type and cell_type_predict, respectively.

In the third experiment, to make the task more harder, we mask the cell type of all cells to the "unknown" cell type. After Dincta, it infers that there are 10 cell types in the data. As we see in the Figure 18, Dincta achieves similar cell type assignment results as the cell type of cells which were annotated separately in each dataset by the marker genes. Dincta almost keeps the cells with different inferred new cell types separate. From Table 9 (row at Dincta (Whole Unknown)), we see that the cluster assignments and cell type assignments also fit to each other better than Harmony's results. From Table 10 (rows at Dincta (Whole Unknown) and Dincta (Whole Unknown with R)), Dincta achieves a little better results than Harmony's. This experiment shows that Dincta is robust even under all the cell type information is masked out.

3 Discussion

Dincta builds a unify framework to do the data integration and cell type annotation simultaneously. We can input all or partial or none of the cell type information of cells which are annotated ahead of running Dincta. Dincta can use these cell type information to infer known or novel cell type of cells, cluster cells in the same cell type together while keeping cells from different cell types separate.

DINCTA: DATA INTEGRATION AND CELL TYPE ANNOTATION OF SINGLE CELL TRANSCRIPTOMES

Figure 18: Dincta (with no cell type information) results of pancreas data. The top left and top middle figures are the UMAP of the PCA embedding, colored by the no cell type information (unknown) (fake_cell_type) which is cell type information input to the Dincta and colored by the inferred cell type information (cell_type_predict), respectively. The rest four figures is the UMAP of the Dincta embedding colored by dataset, cell_type, fake_cell_type and cell_type_predict, respectively.

When there is only one batch, this problem reduce to the cell type inference and assignment problem with no need to do data integration. We implemented this simple case in the function "DinctaClassification" of Dincta Package.

Currently, Dincta is based on the frequency clustering problem essentially. In the future, it is worth to generalize the algorithm to case which use the count values of mRNA.

4 Methods

We now begin to derive our method.

Overview. The Dincta algorithm inputs a PCA embedding (Z) of cells, along with their batch assignments (Φ_B), the all, partial or none cell type assignments (Φ_C), the unknown cell type indicator (U) of cells and the scaling factor α for the cells with unknown cell type. This algorithm was summarized as Algorithm 1 below. The outer loop updates the unknown cell type indicator (Algorithm 6) until it converges by checking the fitness between the cell type assignments and cluster assignments; the inner loop iterates between two complementary stages: maximum diversity clustering and inferring (Algorithm 2), and a mixture model based linear batch correction (Algorithm 5). The clustering and inferring step use the batch corrected embedding $\hat{Z}$ to compute a soft assignment of cells to the clusters, encoded in the matrix R , and then use the cluster assignment matrix R to update the cell type assignment matrix Φ_C . The correction step uses these soft clusters to compute a new corrected embedding from the original one. Efficient implementations of Dincta, including the clustering, inferring and correction subroutines, are available as part of the R package at <https://github.com/songtingstone/dincta>.

Glossary. For reference, we define all data structures used in all Dincta functions. For each one, we define its dimensions and possible values, as well as an intuitive description of what it means in context. The dimensions are stated in terms of d , the dimensionality of the embedding (for example, number of PCs); B , the number of batches; C , the number of cell types (containing the "unknown" cell type and new cell types); N_b , the number of cells in batch b ; and K , the number of clusters.

$Z \in \mathbb{R}^{d \times N}$ The input embedding, to be corrected in Dincta, This is often PCA embeddings of cells.

$\hat{Z} \in \mathbb{R}^{d \times N}$ The integrated embedding, output by Dincta.

$R \in [0, 1]^{K \times N}$ The soft cluster assignment matrix of cells (columns) to clusters (rows).

DINCTA: DATA INTEGRATION AND CELL TYPE ANNOTATION OF SINGLE CELL TRANSCRIPTOMES

Algorithm 1 Dincta

```

1: function DINCTATE( $Z, \Phi_B, \Phi_C, u, \alpha$ )
2:    $\hat{Z} = Z$ 
3:   repeat
4:     repeat
5:        $R, \Phi_C \leftarrow \text{Cluster\_Infer}(\hat{Z}, \Phi_B, \Phi_C, u, \alpha)$ 
6:        $\hat{Z} \leftarrow \text{Correct}(Z, R, \Phi_B)$ 
7:     until convergence
8:      $u \leftarrow \text{Refine}(R, \Phi_C)$ 
9:   until convergence
10:  return  $\hat{Z}, \Phi_C$ 

```

$O \in \mathbb{R}^{K \times C \times B}$ The observed co-occurrence array of cells in clusters, cell types and batches. For the cells with "unknown" cell type, we scale its contribution by a scaling factor α .

$E \in \mathbb{R}^{K \times C \times B}$ The expected co-occurrence array of cells in clusters, cell types and batches. For the cells with "unknown" cell type, we scale its contribution by a scaling factor α .

$O_B \in \mathbb{R}^{K \times B}$ The observed co-occurrence matrix of cells in clusters and batches. For the cells with "unknown" cell type, we scale its contribution by a scaling factor α .

$E_B \in \mathbb{R}^{K \times B}$ The expected co-occurrence matrix of cells in clusters and batches. For the cells with "unknown" cell type, we scale its contribution by a scaling factor α .

$O_C \in \mathbb{R}^{K \times C}$ The observed co-occurrence matrix of cells in clusters and cell types. For the cells with "unknown" cell type, we scale its contribution by a scaling factor α .

$Y \in [0, 1]^{d \times K}$ Cluster centroid locations in the k -means clustering algorithm.

$\Phi_B \in [0, 1]^{B \times N}$ The batch assignment membership of cells.

$\Phi_C \in [0, 1]^{C \times N}$ The cell type assignment membership of cells. If there are cells with "unknown" cell type, then $C = C_{\text{original}} + 1 + C_{\text{new}}$ where C_{original} is the number of cell type known in the dataset. 1 accounts for the special "unknown" cell type, and C_{new} is the number of new cell types in the data. For the initial input Φ_C of Dincta, $C_{\text{new}} = 0$. And if the cell type of i th cell is unknown, then $\Phi_C(\cdot, i) = [0, \dots, 0, 1] \in \mathbb{R}^{C_{\text{original}}+1}$, i.e the last row of Φ_C is for the "unknown" cell type. As the progress of the Dincta, it will infer the the number of new cell types C_{new} , soft assign each cell with known or novel cell types, and also reduce the "unknown" cell type row $\Phi_C(C_{\text{original}} + 1, \cdot)$ to $\mathbf{0}$. If the cell types of all cells are known initially, then $C = C_{\text{original}}$ and there is no inference step.

$u \in \{0, 1\}^N$ The indicator vector of "unknown" cell type. $u_i = 1$ if the i -th cell's cell type is unknown. $u_i = 0$ if the i -th cell's cell type is known. When we have the whole cell type information, then $u = \mathbf{0}$.

$\alpha \in [0, 1]$ The scaling factor for the cells which have unknown cell type.

$u_\alpha \in \{\alpha, 1\}^N := \alpha u + (1 - u)$ The scaling vector. $u_\alpha(i) = 1$ if the cell type of the i -th cell is known. $u_\alpha(i) = \alpha$ if the cell type of the i -th cell is unknown. It was used in the loss scaling and E , O and M computation, reduce the weight of information coming from the cells which have "unknown" cell type.

$\Phi \in [0, 1]^{C \times B \times N}$ The scaled cell type and batch assignment membership of cells, $\Phi(c, b, i) := \Phi_C(c, i)\Phi_B(b, i)u_\alpha(i)$.

$M \in [0, 1]^{K \times C}$ The condition probability $Pr(k|c)$ matrix of cells in clusters (columns), cell types (rows).

$H \in [0, 1]^{C \times K}$ The condition probability $Pr(c|k)$ matrix of cells in clusters (rows), cell types (columns).

$N_\alpha := u_\alpha^T \mathbf{1}$ The effective count of cells.

$N_C := \Phi_C u_\alpha \in \mathbb{R}^C$ The effective count vector of cells in cell types.

$N_{CB} := \Phi_C \text{diag}(u_\alpha) \Phi_B^T \in \mathbb{R}^{C \times B}$ The effective observed co-occurrence matrix of cells in cell types and batches.

$Pr_{B|C} \in \mathbb{R}^{C \times B}$ The probability matrix of the $Pr(b|c)$.

DINCTA: DATA INTEGRATION AND CELL TYPE ANNOTATION OF SINGLE CELL TRANSCRIPTOMES

$\mathcal{U}$ The index of cells with unknown cell type.

$\mathcal{U}_{original}$ The index of cells with unknown cell type in the initial state.

$\mathcal{I}$ The index of cells with known cell type.

$\mathcal{I}_{original}$ The index of cells with known cell type in the initial state.

Assumptions about the input data. In this article, we always use Dincta on a low-dimension embedding of the cells. To be clear about the properties of low-dimensional embedding space, we explicitly state three assumptions as described in Korsunsky et al. (2018):

1. Cells are embedded into a low-dimensional space as the result of PCA. The PCA embedding captures the variation of gene expression in a compact orthonormal space. For this reason, the default input to Dincta is a matrix of gene expression data, normalized for library size. We then perform PCA on this high-dimensional matrix and use the eigenvalue-scaled eigen vectors as the low-dimensional embedding input to Dincta.
2. Gene expression has been normalized for library size. In RNA-seq, each cell will be sequenced to a different depth, which results in different library sizes for each cell. It is best practice to account for this source of technical variation before performing PCA. In this article, we use the standard log counts-per10,000 (CP10K) transformation. As a result of the depth transformation, expression values are turned into relative frequencies inside each cell. Thus, it is impossible for every gene to be up-regulated in one group of cells.
3. The low-dimensional nearest-neighbor structure induced by Euclidean distance should be preserved with common similarity metrics such as cosine similarity and correlation. This can be easily checked by computing sparse nearest-neighbor graphs and comparing the adjacency matrices. Cells with less than 20% overlapping neighbors can be removed as outliers.

Maximum Diversity Clustering and Inferring. We developed a clustering algorithm to maximize the diversity among batches within clusters and do not mix cells with different cell types. After we get the soft cluster assignment matrix, we use it to update cell type assignment matrix (inferring). We present this method as follows.

First, we review a previously published objective function for soft k -means clustering. Second, we add a diversity maximizing regularization term to this objective function, and derive this regularization term as the penalty on the statistical conditional dependence between two random variables: batch membership and cluster assignment, when given the cell type membership. Third, we add a penalty that constraint the the cluster membership in accordance with the cell type membership. We then derive and present pseudocode for an algorithm to optimize the objective function.

If there are cells with unknown cell type, we derive and present pseudocode to infer the known or novel cell type of cells with the cluster assignments.

Background: entropy regularization for soft k -means. The basic objective function for classical k -means clustering, in which each cell belongs to exactly one cluster, is defined by the distance from cells to their assigned centroids.

$$\begin{aligned} \min_{R,Y} \sum_{i,k} R_{ki} \|Z_i - Y_k\|^2 \\ \text{s.t. } \forall_i \forall_k R_{ki} \in \{0, 1\} \end{aligned} \quad (1)$$

Above, Z is some feature space of the data and Y is centroids matrix. R_{ki} denotes the membership of cell i in cluster k and take values in 0 and 1. To transform this into a soft clustering objective, we follow the derivation of Korsunsky et al. (2018) and add an entropy regularization term over R , weighted by a hyperparameter σ_e . Now, R_{ki} can take values between 0 and 1 with the constraint that R_i must be a proper probability distribution with support $[1, K]$.

$$\begin{aligned} \min_{R,Y} \sum_{i,k} R_{ki} \|Z_i - Y_k\|^2 + \sigma_e R_{ki} \log R_{ki} \\ \text{s.t. } \forall_i \forall_k R_{ki} > 0, \forall_i \sum_{k=1}^K R_{ki} = 1 \end{aligned} \quad (2)$$

As σ_e approaches 0, this penalty approaches hard clustering. As σ_e approaches infinity, the entropy of R over-weights the data-centroid distances and each data point is assigned equality to all clusters.

DINCTA: DATA INTEGRATION AND CELL TYPE ANNOTATION OF SINGLE CELL TRANSCRIPTOMES

In addition to soft assignment regularization, the function below penalizes cluster with low batch-diversity in each cell type. This penalty, derived in the following section, depends on the cluster, cell type and batch identities $\Omega(R, \Phi_B, \Phi_C, u_\alpha) = \sum_{i,k} R_{ki} \log(O_{ki}/E_{ki}) \Phi_i$.

$$\begin{aligned} \min_{R,Y} \sum_{i,k} R_{ki} \|Z_i - Y_k\|^2 + \sigma_e R_{ki} \log R_{ki} + \sigma_e \theta R_{ki} \log \left(\frac{O_{ki}}{E_{ki}} \right) \Phi_i \\ \text{s.t. } \forall_i \forall_k R_{ki} > 0, \forall_i \sum_{k=1}^K R_{ki} = 1 \end{aligned} \quad (3)$$

For each batch variable, we add a new parameter θ . The θ decides the degree of penalty for dependence between batch membership and cluster assignment in each cell type. When $\theta = 0$, the problem reverts back to 2, with no penalty on the dependence. As θ increases, the objective function favors more independence between batch b and cluster assignment in each cell type.

Objective Function for Maximum Diversity Clustering. The full objective function for Dincta's clustering builds on the previous section. In addition to soft assignment regularization and encourage batch-diversity in each type, we constrained the R satisfied the cell type probability relationship $Pr(k) = \sum_c Pr(k|c)Pr(c)$ by penalizing the Kullback Leibler divergence between R and $M\Phi_C$.

$$\begin{aligned} \min_{R,Y} \sum_{i,k} R_{ki} \|Z_i - Y_k\|^2 + \sigma_e R_{ki} \log R_{ki} + \sigma_e \theta R_{ki} \log \left(\frac{O_{ki}}{E_{ki}} \right) \Phi_i \\ + \sigma_c u_\alpha(i) R_{ki} \log \left(\frac{R_{ki}}{(M\Phi_C)_{ki}} \right) \\ \text{s.t. } \forall_i \forall_k R_{ki} > 0, \forall_i \sum_{k=1}^K R_{ki} = 1 \end{aligned} \quad (4)$$

We add a new parameter σ_c . Here the σ_c decides the degree of penalty for the consistence between batch membership R and batch membership inferred from the probability relationship $M\Phi_C$. When $\sigma_c = 0$, the problem reverts back to 3, with no penalty on the discrepancy between batch membership R and batch membership inferred from the probability relationship $M\Phi_C$. As σ_c increases, the objective function favors the cluster assignments to be close to the cluster assignments inferred from the cell type assignments for each cell. As σ_c approaches infinity, it will yield a hard constraint $R = M\Phi_C$.

For saving the computation power, we use the cosine distance instead of Euclidean distance, between Z and Y , which will constrain the clustering problem to the sphere clustering problem. Note that the squared Euclidean distance is equivalent to cosine distance when the vectors are L_2 normalized. Therefore, assuming that all Z_i and Y_k have a unity L_2 norm, the squared Euclidean distance above can be re-written as a dot product.

$$\begin{aligned} \min_{R,Y} \sum_{i,k} R_{ki} 2(1 - Y_k^T Z_i) + \sigma_e R_{ki} \log R_{ki} + \sigma_e \theta R_{ki} \log \left(\frac{O_{ki}}{E_{ki}} \right) \Phi_i \\ + \sigma_c u_\alpha(i) R_{ki} \log \left(\frac{R_{ki}}{(M\Phi_C)_{ki}} \right) \\ \text{s.t. } \forall_i \forall_k R_{ki} > 0, \forall_i \sum_{k=1}^K R_{ki} = 1 \end{aligned} \quad (5)$$

Cluster Diversity Score. Here, we discuss and derive the diversity penalty term $\Omega(\cdot)$, defined in the previous section. For simplicity, we discuss diversity with respect to a single batch variable, as the multiple batch penalty terms are additive in the objective function. The goal of $\Omega(\cdot)$ is to penalize the statistical dependence between batch identity and cluster assignment in each cell type. The observed co-occurrence counts (O) are compared to the counts expected under independence (E) for each cell type. We use the Kullback Leibler divergence (D_{KL}), an information theoretic distance between two distributions to define the penalty. In this section, we define the O and E distributions, as we as the D_{KL} penalty, in the context of the probabilistic cluster assignment matrix R .

$$\begin{aligned} O_{kcb} &= N_\alpha \Pr(k, c, b) \\ O_{kcb} &= N_\alpha \Pr(c) \Pr(k, b|c) \\ O_{kcb} &= N_\alpha \frac{N_C(c)}{N_\alpha} \frac{\sum_{i=1}^N \Phi_C(c, i) \mathbf{1}_{i \in b} R_{ki} u_\alpha(i)}{N_C(c)} \\ O_{kcb} &= \sum_{i=1}^N \Phi_C(c, i) \Phi_B(b, i) R_{ki} u_\alpha(i) \\ E_{kcb} &= N_\alpha \Pr(k, c, b) \\ E_{kcb} &= N_\alpha \Pr(c) \Pr(b|c) \Pr(k|c) \\ E_{kcb} &= N_\alpha \frac{N_C(c)}{N_\alpha} \frac{\sum_{i=1}^N \Phi_C(c, i) \mathbf{1}_{i \in b} u_\alpha(i)}{N_C(c)} \frac{\sum_{i=1}^N \Phi_C(c, i) R_{ki} u_\alpha(i)}{N_C(c)} \\ E_{kcb} &= \frac{N_{CB}(C, b)}{N_C(c)} \sum_{i=1}^N \Phi_C(c, i) R_{ki} u_\alpha(i) \end{aligned} \quad (6)$$

DINCTA: DATA INTEGRATION AND CELL TYPE ANNOTATION OF SINGLE CELL TRANSCRIPTOMES

Next, we define the KL divergence in terms of R . Note that both O and E depend on R . However, in the derivation below, we expand one of the O terms. This serves a functional purpose in the optimization procedure, described later. Intuitively, in the update step of R for a single cell, we compute O and E on all the other cells. In this way, we decide how to assign the single cell to clusters given the current distribution of batches, clusters and cell types.

$$\begin{aligned} D_{KL}(O||E) &= \sum_{k=1}^K \sum_{c=1}^C \sum_{b=1}^B O_{kcb} \log \frac{O_{kcb}}{E_{kcb}} \\ D_{KL}(O||E) &= \sum_{k=1}^K \sum_{c=1}^C \sum_{b=1}^B \sum_{i=1}^N \Phi_C(c, i) \Phi_B(b, i) R_{ki} u_\alpha(i) \log \frac{O_{kcb}}{E_{kcb}} \\ D_{KL}(O||E) &= \sum_{i=1}^N \sum_{k=1}^K \sum_{c=1}^C \sum_{b=1}^B \Phi_C(c, i) \Phi_B(b, i) R_{ki} u_\alpha(i) \log \frac{O_{kcb}}{E_{kcb}} \\ D_{KL}(O||E) &= \sum_{i=1}^N \sum_{k=1}^K R_{ki} \left[\sum_{c=1}^C \sum_{b=1}^B \log \frac{O_{kcb}}{E_{kcb}} \Phi(c, b, i) \right] \end{aligned} \quad (7)$$

Define

$$\log \frac{O_{ki}}{E_{ki}} \Phi_i := \sum_{c=1}^C \sum_{b=1}^B \log \frac{O_{kcb}}{E_{kcb}} \Phi(c, b, i) \quad (8)$$

, we can simplify the above equation to the following equation.

$$D_{KL}(O||E) = \sum_{i=1}^N \sum_{k=1}^K R_{ki} \log \frac{O_{ki}}{E_{ki}} \Phi_i \quad (9)$$

Optimization. Optimization of equation (5) admits an expectation-maximization framework, iterating between cluster assignment (R) and centroid (Y) estimation.

We solve for the optimal assignment R , for each cell i . First, we set up the Lagrangian with dual parameter λ and solve for the partial derivative with respect to each cluster k .

$$\begin{aligned} L_{R,Y} &= \sum_{i,k} [R_{ki} 2(1 - Y_k^T Z_i) + \sigma_e R_{ki} \log R_{ki} + \sigma_e \theta R_{ki} \log \left(\frac{O_{ki}}{E_{ki}} \right) \Phi_i + \sigma_c u_\alpha(i) R_{ki} \log \left(\frac{R_{ki}}{(M\Phi_C)_{ki}} \right)] \\ &\quad + \lambda \sum_i \left[\left(\sum_{k=1}^K R_{ki} \right) - 1 \right] \\ \frac{\partial L(R_i, \lambda)}{\partial R_{ki}} &= 0 \\ &= 2(1 - Y_k^T Z_i) + \sigma_e (1 + \log R_{ki}) + \sigma_e \theta \log \left(\frac{O_{ki}}{E_{ki}} \right) \Phi_i + \sigma_c (1 + \log R_{ki} - \log((M\Phi_C)_{ki})) + \lambda \\ \log R_{ki} &= -\frac{2(1 - Y_k^T Z_i)}{\sigma_e + u_\alpha(i)\sigma_c} - 1 - \frac{\sigma_e}{\sigma_e + u_\alpha(i)\sigma_c} \theta \log \left(\frac{O_{ki}}{E_{ki}} \right) + \frac{u_\alpha(i)\sigma_c}{\sigma_e + u_\alpha(i)\sigma_c} \log((M\Phi_C)_{ki}) - \frac{\lambda}{\sigma_e + u_\alpha(i)\sigma_c} \\ R_{ki} &= \exp \left(-\frac{2(1 - Y_k^T Z_i)}{\sigma_e + u_\alpha(i)\sigma_c} - \frac{\sigma_e}{\sigma_e + u_\alpha(i)\sigma_c} \theta \log \left(\frac{O_{ki}}{E_{ki}} \right) + \frac{u_\alpha(i)\sigma_c}{\sigma_e + u_\alpha(i)\sigma_c} \log((M\Phi_C)_{ki}) \right) \exp \left(-\frac{\lambda}{\sigma_e + u_\alpha(i)\sigma_c} - 1 \right) \end{aligned}$$

Next, we use the probability constraint $\sum_{k=1}^K R_{ki} = 1$ to solve for $\exp \left(-\frac{\lambda}{\sigma_e + u_\alpha(i)\sigma_c} - 1 \right)$

$$\begin{aligned} 1 &= \sum_{k=1}^K R_{ki} \\ 1 &= \sum_{k=1}^K \exp \left(-\frac{2(1 - Y_k^T Z_i)}{\sigma_e + u_\alpha(i)\sigma_c} - \frac{\sigma_e}{\sigma_e + u_\alpha(i)\sigma_c} \theta \log \left(\frac{O_{ki}}{E_{ki}} \right) + \frac{u_\alpha(i)\sigma_c}{\sigma_e + u_\alpha(i)\sigma_c} \log((M\Phi_C)_{ki}) \right) \\ &\quad \exp \left(-\frac{\lambda}{\sigma_e + u_\alpha(i)\sigma_c} - 1 \right) \\ \exp \left(-\frac{\lambda}{\sigma_e + u_\alpha(i)\sigma_c} - 1 \right) &= \frac{1}{\sum_{k=1}^K \exp \left(-\frac{2(1 - Y_k^T Z_i)}{\sigma_e + u_\alpha(i)\sigma_c} - \frac{\sigma_e}{\sigma_e + u_\alpha(i)\sigma_c} \theta \log \left(\frac{O_{ki}}{E_{ki}} \right) + \frac{u_\alpha(i)\sigma_c}{\sigma_e + u_\alpha(i)\sigma_c} \log((M\Phi_C)_{ki}) \right)} \end{aligned}$$

Finally, we substitute $\exp \left(-\frac{\lambda}{\sigma_e + u_\alpha(i)\sigma_c} - 1 \right)$ to remove the dependency of R_{ki} on the dual parameter λ .

$$R_{ki} = \frac{\exp \left(-\frac{2(1 - Y_k^T Z_i)}{\sigma_e + u_\alpha(i)\sigma_c} - \frac{\sigma_e}{\sigma_e + u_\alpha(i)\sigma_c} \theta \log \left(\frac{O_{ki}}{E_{ki}} \right) + \frac{u_\alpha(i)\sigma_c}{\sigma_e + u_\alpha(i)\sigma_c} \log((M\Phi_C)_{ki}) \right)}{\sum_{k=1}^K \exp \left(-\frac{2(1 - Y_k^T Z_i)}{\sigma_e + u_\alpha(i)\sigma_c} - \frac{\sigma_e}{\sigma_e + u_\alpha(i)\sigma_c} \theta \log \left(\frac{O_{ki}}{E_{ki}} \right) + \frac{u_\alpha(i)\sigma_c}{\sigma_e + u_\alpha(i)\sigma_c} \log((M\Phi_C)_{ki}) \right)} \quad (10)$$

The denominator term above makes sure that R_{ki} sums to one. In practice (Algorithm 2), we compute the numerator and divide by the sum.

Centroid estimation Y . Our clustering algorithm use cosine distance instead Euclidean distance. In the context of k -means clustering, this approach was pioneered by Dhillon and Modha (2001). We adopt their centroid estimation

DINCTA: DATA INTEGRATION AND CELL TYPE ANNOTATION OF SINGLE CELL TRANSCRIPTOMES

procedure for our algorithm. Instead of just computing the mean position of all cells that belong in cluster k , this approach then L_2 normalizes each centroid vector to make it unit length.

The unnormalized expectation of centroid position for cluster k would be $Y_k = \mathbb{E}_{R_k} Z = \sum_i R_{ki} Z_i$. In vector form, for all centroids, this is $Y = R^T Z$. The final position of the cluster centroids is given by this summation followed by L_2 normalization of each centroid. This procedure is implemented in Algorithm 2 in the section { *Compute Cluster Centroids* }

Our algorithm can be run in two modes. In the first mode, we have the full cell type information of all cells. And we only need to find the cluster assignments R and correct the embedding Z by R . In the second mode, there are some or all cells with unknown cell type, and our algorithm can infer the known or novel cell type of cells with "unknown" cell type initially from cluster assignments R . In both kind of modes, the procedure of updating R is the same. Since at this time, we feed the Φ_C to the procedure as it as the true cell type assignments matrix. Now we give the maximum diversity clustering and inferring algorithm in Algorithm 2 and left the deduction of function update Φ_C later.

Implementation details. The update steps of R and Y derived above form the core of maximum diversity clustering and inferring, outlined as Algorithm 1. This section explains the other implementation details of this pseudocode. Again, for simplicity, we discuss details related to diversity penalty terms θ , Φ_B , O and E for each batch variable independently.

Block updates of R . Unlike in regular k -means, the optimization procedure above for R cannot be faithfully parallelized, as the values of O , E and M changed with R . The exact solution therefore depends an online procedure. For speed, we can coarse grain this procedure and update R in small blocks (for example, 5% of the data). Meanwhile, O , E and M are computed on the held out data. In practice, this approach succeeds in minimizing the objective for sufficiently small block size. In the algorithm, these blocks are included as the Update Blocks in the for loop.

Centroid initialization. We initialize cluster centroids by using regular k -means clustering, implemented in the base R k -means function. We use ten random restarts and keep the best one. We then L_2 normalize the centroids to prepare them for spherical k -means clustering.

Regularization for smoother penalty. The diversity penalty term O_{kcb}/E_{kcb} can tend toward infinity if there are no cells from batch b , cell type c and cluster k . This extreme penalty can erroneously force cells into an inappropriate cluster. To protect against this, we add 1 to O and E to ensure that the fraction is stable $(1 + O_{kcb})/(1 + E_{kcb})$. In this way, we can reduce the variance in the sample estimation of the counts matrix.

If some cell type has too small number of cells, then the $(1 + O_{kcb})/(1 + E_{kcb})$ may fluctuate a lot. We can use the $(1 + O_C(k.b))/(1 + E_C(k.b))$ to replace it, where $O_C(k.b) := \sum_c O_{kcb}$, $E_C(k.b) := \sum_c E_{kcb}$. This can reduce the variance a lot. And it is implemented in the Dincta by set the the parameter "R.cross.entropy.type='b'".

K , the number of clusters. The number of clusters K use in Dincta soft clustering should be set to a value that reflects the size and complexity of the dataset. Too few clusters will not capture the number of biologically distinct cell types and states. Too many clusters will give too much weight to batch-specific outliers and prevent effective integration. As a heuristic, we assume that the dataset have at most 100 distinct cell types and that each cluster should have at least 30 cells. We set the default number of clusters, K , to lie between these two extremes for N cells.

$$K = \min\left(100, \frac{N}{30}\right)$$

Cell Type Inference

Cell type assignment and cluster assignment relation. Here, we discuss the probability relationship between cell type assignment and cluster assignment. And then we use their relations to infer the cell type of the cells. Let $\Pr(k, c)$ be the probability that cell type c and cluster k occur together, $\Pr(k)$ be the probability that cluster k occurs, and $\Pr(c)$ be the probability that cell type c occurs. Then we have the follow probability relationships

$$\begin{aligned} \Pr(k) &= \sum_{c=1}^C \Pr(k, c) \\ &= \sum_{c=1}^C \Pr(k|c)\Pr(c) \end{aligned} \quad (11)$$

$$\begin{aligned} \Pr(c) &= \sum_{k=1}^K \Pr(k, c) \\ &= \sum_{k=1}^K \Pr(c|k)\Pr(k) \end{aligned} \quad (12)$$

We can apply the above probability relation equation to each cell to get the probability relationship for each cell, which reveals how to transform between the cluster assignments and cell type assignments.

DINCTA: DATA INTEGRATION AND CELL TYPE ANNOTATION OF SINGLE CELL TRANSCRIPTOMES

Algorithm 2 Maximum Diversity Clustering And Inferring

```

1: function CLUSTER_INFER( $\hat{Z}, \Phi_C, \Phi_B, u, \alpha$ )
2:   { Initialize Cluster Centroids }
3:    $Y \leftarrow \text{kmeans}(\hat{Z}, K)$ 
4:   for  $i \leftarrow 1 \dots N$  do                                     ▶  $L_2$  Normalization
5:      $\hat{Z}(\cdot, i) \leftarrow \hat{Z}(\cdot, i) / \|\hat{Z}(\cdot, i)\|_2$ 
6:   for  $k \leftarrow 1 \dots K$  do                                     ▶  $L_2$  Normalization
7:      $Y(\cdot, k) \leftarrow Y(\cdot, k) / \|Y(\cdot, k)\|_2$ 
8:   repeat
9:      $u_\alpha = \alpha * u + (1 - u)$ 
10:     $N_C = \Phi_C u_\alpha$ 
11:     $N_{CB} = \Phi_C \text{diag}(u_\alpha) \Phi_B^T$ 
12:    for  $c \leftarrow 1 \dots C$  do
13:       $Pr_{B|C}(c, \cdot) = N_{CB}(c, \cdot) / N_C(c)$ 
14:    for  $k \leftarrow 1 \dots K$  do
15:      for  $c \leftarrow 1 \dots C$  do
16:        for  $b \leftarrow 1 \dots B$  do
17:           $E(k, c, b) = \sum_{i=1}^N u_\alpha(i) R(k, i) \Phi_C(c, i) Pr_{B|C}(c, b)$ 
18:           $O(k, c, b) = \sum_{i=1}^N u_\alpha(i) R(k, i) \Phi_C(c, i) \Phi_B(b, i)$ 
19:    for all Update Blocks do
20:       $in \leftarrow$  cells to update in block
21:      { Compute  $O$ ,  $E$  and  $M$  on left out data }
22:      for  $k \leftarrow 1 \dots K$  do
23:        for  $c \leftarrow 1 \dots C$  do
24:          for  $b \leftarrow 1 \dots B$  do
25:             $E_{in}(k, c, b) = \sum_{i \in in} u_\alpha(i) R(k, i) \Phi_C(c, i) Pr_{B|C}(c, b)$ 
26:             $O_{in}(k, c, b) = \sum_{i \in in} u_\alpha(i) R(k, i) \Phi_C(c, i) \Phi_B(b, i)$ 
27:       $E \leftarrow E - E_{in}$ 
28:       $O \leftarrow O - O_{in}$ 
29:      for  $k \leftarrow 1 \dots K$  do
30:        for  $c \leftarrow 1 \dots C$  do
31:           $O_C(k, c) = \sum_{b=1}^B O(k, c, b)$ 
32:       $M = O_C \text{diag}(O_C^T \mathbf{1})^{-1}$ 
33:      { Update and Normalize  $R$  }
34:      for  $i \in in$  do
35:         $R_i \leftarrow \exp \left( -\frac{2(1-Y^T Z_i)}{\sigma_e + u_\alpha(i) \sigma_c} - \frac{\sigma_e}{\sigma_e + u_\alpha(i) \sigma_c} \theta \log \left( \frac{O_i + 1}{E_i + 1} \right) \Phi_i + \frac{u_\alpha(i) \sigma_c}{\sigma_e + u_\alpha(i) \sigma_c} \log((M \Phi_C)_i) \right)$ 
36:         $R_i \leftarrow \frac{R_i}{\mathbf{1}^T R_i}$                                      ▶  $i$ -th column of  $R$ , see equation (10)
37:      { Compute  $O$  and  $E$  with full data }
38:      for  $k \leftarrow 1 \dots K$  do
39:        for  $c \leftarrow 1 \dots C$  do
40:          for  $b \leftarrow 1 \dots B$  do
41:             $E_{in}(k, c, b) = \sum_{i \in in} R(k, i) \Phi_C(c, i) Pr_{B|C}(c, b)$ 
42:             $O_{in}(k, c, b) = \sum_{i \in in} R(k, i) \Phi_C(c, i) \Phi_B(b, i)$ 
43:       $E \leftarrow E + E_{in}$ 
44:       $O \leftarrow O + O_{in}$ 
45:      { Compute Cluster Centroids }
46:       $Y \leftarrow Z R^T$ 
47:      for  $i \leftarrow 1 \dots N$  do                                     ▶  $L_2$  Normalization
48:         $Y(\cdot, i) \leftarrow Y(\cdot, i) / \|Y(\cdot, i)\|_2$ 
49:      if  $\sum_i u_i > 0$  then
50:        { infer new cell type and update cell type assignment matrix }
51:        Update_Phi_C( $R, \Phi_C, u, \alpha, C_{known}$ )                                     ▶ Can update once every  $t$  rounds, e.g  $t = 5$ .
52:      until convergence
53:      return  $R$ 

```

DINCTA: DATA INTEGRATION AND CELL TYPE ANNOTATION OF SINGLE CELL TRANSCRIPTOMES

Before doing that, we prepare some necessary notations. Note that R_{ki} is the probability that cell i assigned to the cluster k , $\Phi_B(b, i)$ is the probability that cell i belong to batch b , and $\Phi_C(c, i)$ is the probability that cell i assigned to the cell type c . With the three quantities, we can estimate the the transform probability matrix element $\Pr(c|k)$ by H_{ck} .

$$\begin{aligned} H_{ck} &= \Pr(c|k) \\ H_{ck} &= \frac{\Pr(k, c)}{\Pr(k)} \\ H_{ck} &= \frac{O_C(k, c)}{\sum_{c=1}^C O_C(k, c)} \\ O_C(k, c) &= \sum_{i=1}^N u_\alpha(i) R_{ki} \Phi_C(c, i) \end{aligned} \quad (13)$$

Similarly, we can estimate the the transform probability matrix element $\Pr(k|c)$ by M_{kc} .

$$\begin{aligned} M_{kc} &= \Pr(k|c) \\ M_{kc} &= \frac{\Pr(k, c)}{\Pr(c)} \\ M_{kc} &= \frac{O_C(k, c)}{\sum_{k=1}^K O_C(k, c)} \\ O_C(k, c) &= \sum_{i=1}^N u_\alpha(i) R_{ki} \Phi_C(c, i) \end{aligned} \quad (14)$$

With these preparations, we apply the probability relation equation (11) to each cell. We get the following equation.

$$R_{ki} = \sum_{c=1}^C M_{kc} \Phi_C(c, i) \quad (15)$$

Similarly, we apply the probability relation equation (12) to each cell. We get the following equation.

$$\Phi_C(c, i) = \sum_{k=1}^K H_{ck} R_{ki} \quad (16)$$

The above equations can be represented in the matrix form. Note that $M = O_C \text{diag}(O_C^T \mathbf{1})^{-1}$, $O_C = R \text{diag}(u_\alpha) \Phi_C^T$, we can represent M by R and Φ_C with the following equation.

$$\begin{aligned} M &= O_C \text{diag}(O_C^T \mathbf{1})^{-1} \\ &= R \text{diag}(u_\alpha) \Phi_C^T \text{diag}(\Phi_C \text{diag}(u_\alpha) R^T \mathbf{1})^{-1} \\ &= R \text{diag}(u_\alpha) \Phi_C^T \text{diag}(\Phi_C u_\alpha)^{-1} \end{aligned} \quad (17)$$

, where the third equality comes from the R is a probability matrix which means the sum of each column of R is $\mathbf{1}$.

Similarly, note that $H = O_C^T \text{diag}(O_C \mathbf{1})^{-1}$, $O_C = R \text{diag}(u_\alpha) \Phi_C^T$, we can represent H by R and Φ_C with the following equation.

$$\begin{aligned} H &= O_C^T \text{diag}(O_C \mathbf{1})^{-1} \\ &= \Phi_C \text{diag}(u_\alpha) R^T \text{diag}(R \text{diag}(u_\alpha) \Phi_C^T \mathbf{1})^{-1} \\ &= \Phi_C \text{diag}(u_\alpha) R^T \text{diag}(R u_\alpha)^{-1} \end{aligned} \quad (18)$$

, where the third equality comes from the Φ_C is a probability matrix which means the sum of each column of Φ_C is $\mathbf{1}$.

Combining the equation (15) and (17), we get the following equation.

$$\begin{aligned} R &= M \Phi_C \\ &= R \text{diag}(u_\alpha) \Phi_C^T \text{diag}(\Phi_C u_\alpha)^{-1} \Phi_C \end{aligned} \quad (19)$$

Similarly, combining the equation (16) and (18), we get the following equation.

$$\begin{aligned} \Phi_C &= H R \\ &= \Phi_C \text{diag}(u_\alpha) R^T \text{diag}(R u_\alpha)^{-1} R \end{aligned} \quad (20)$$

From equation (19) and (16), we see that R and Φ_C have very close relationship. To make the relation more clearly, we multiply $\text{diag}^{1/2}(u_\alpha)$ in the left and right side of equation (19) and (16). This give the following equations.

$$R \text{diag}^{1/2}(u_\alpha) = R \text{diag}^{1/2}(u_\alpha) [\Phi_C \text{diag}^{1/2}(u_\alpha)]^T \text{diag}(\Phi_C u_\alpha)^{-1} [\Phi_C \text{diag}^{1/2}(u_\alpha)] \quad (21)$$

DINCTA: DATA INTEGRATION AND CELL TYPE ANNOTATION OF SINGLE CELL TRANSCRIPTOMES

$$\Phi_C \text{diag}^{1/2}(u_\alpha) = \Phi_C \text{diag}^{1/2}(u_\alpha) [R \text{diag}^{1/2}(u_\alpha)]^T \text{diag}(Ru_\alpha)^{-1} [R \text{diag}^{1/2}(u_\alpha)] \quad (22)$$

We list some of the property about $R \text{diag}^{1/2}(u_\alpha)$, $\Phi_C \text{diag}^{1/2}(u_\alpha)$ and their invariant projection matrix $T_C := [\Phi_C \text{diag}^{1/2}(u_\alpha)]^T \text{diag}(\Phi_C u_\alpha)^{-1} [\Phi_C \text{diag}^{1/2}(u_\alpha)]$ and $T_R := [R \text{diag}^{1/2}(u_\alpha)]^T \text{diag}(Ru_\alpha)^{-1} [R \text{diag}^{1/2}(u_\alpha)]$.

1. The invariant projection matrix T_C of $R \text{diag}^{1/2}(u_\alpha)$ is symmetric semi-positive definite matrix. Its largest eigenvalue is 1 of algebraic multiplicity at least of rank of $R \text{diag}^{1/2}$.
2. Each row of $R \text{diag}^{1/2}$ is the right eigenvector of the matrix T_C about the eigenvalue 1. $u_\alpha^{1/2}$ is the left eigenvector of T_C about the eigenvalue 1.
3. The invariant projection matrix T_R of $\Phi_C \text{diag}^{1/2}(u_\alpha)$ is symmetric semi-positive definite matrix. Its largest eigenvalue is 1 of algebraic multiplicity at least of rank of $\Phi_C \text{diag}^{1/2}(u_\alpha)$.
4. Each row of $\Phi_C \text{diag}^{1/2}(u_\alpha)$ is the right eigenvector of the matrix T_R about the eigenvalue 1. $u_\alpha^{1/2}$ is the left eigenvector of T_R about the eigenvalue 1.

Let we look at the relationship between R and Φ_C . R is the soft cluster assignment matrix, where the number of clusters is pre-specified and it usually larger than the number of the cell types. Φ_C is the cell type assignment matrix. And many clusters may be mapped to the same cell type. R is fine grained version of Φ_C , may be viewed as the subtypes assignment matrix. So we can combine the information from R to infer the cell type of cells with unknown cell type initially by the formula $\Phi_C(\cdot, i) = M^T R(\cdot, i)$ if M is accurate. If we find the correct cluster assignment R , we also get the correct cell type assignment Φ_C .

In the practice, the M is not an accurate estimation of $\Pr(k|C)$, since it depends on Φ_C which is not accurate. However, note that if we know the cell type assignments $\Phi_C(\cdot, \mathcal{I})$ for the cells with known cell type initially and we have the correct cluster assignment matrix R , we can infer the cell type assignments $\Phi_C(\cdot, \mathcal{U})$ for the cells with unknown cell type by solving the linear equation (22). In details, we fetch out the columns indexed by $\mathcal{U}$ in the left and right side of (22), we get the following equation.

$$\begin{aligned} \Phi_C(\cdot, \mathcal{U}) \text{diag}^{1/2}(u_\alpha(\mathcal{U})) &= \Phi_C \text{diag}(u_\alpha) R^T \text{diag}(Ru_\alpha)^{-1} R(\cdot, \mathcal{U}) \text{diag}^{1/2}(u_\alpha(\mathcal{U})) \\ &= [\Phi_C(\cdot, \mathcal{U}), \Phi_C(\cdot, \mathcal{I})] \text{diag}([u_\alpha(\mathcal{U})^T, u_\alpha(\mathcal{I})^T]^T) [R(\cdot, \mathcal{U}), R(\cdot, \mathcal{I})]^T \\ &\quad \text{diag}(Ru_\alpha)^{-1} R(\cdot, \mathcal{U}) \text{diag}^{1/2}(u_\alpha(\mathcal{U})) \\ &= (\Phi_C(\cdot, \mathcal{U}) \text{diag}(u_\alpha(\mathcal{U})) R(\cdot, \mathcal{U})^T + \Phi_C(\cdot, \mathcal{I}) \text{diag}(u_\alpha(\mathcal{I})) \\ &\quad R(\cdot, \mathcal{I})^T) \text{diag}(Ru_\alpha)^{-1} R(\cdot, \mathcal{U}) \text{diag}^{1/2}(u_\alpha(\mathcal{U})) \\ &= \Phi_C(\cdot, \mathcal{U}) \text{diag}(u_\alpha(\mathcal{U})) R(\cdot, \mathcal{U})^T \text{diag}(Ru_\alpha)^{-1} R(\cdot, \mathcal{U}) \text{diag}^{1/2}(u_\alpha(\mathcal{U})) \\ &\quad + \Phi_C(\cdot, \mathcal{I}) \text{diag}(u_\alpha(\mathcal{I})) R(\cdot, \mathcal{I})^T \text{diag}(Ru_\alpha)^{-1} R(\cdot, \mathcal{U}) \text{diag}^{1/2}(u_\alpha(\mathcal{U})) \end{aligned} \quad (23)$$

From above equation, we see that cell type assignment matrix $\Phi_C(\cdot, \mathcal{U})$ for the cells with unknown cell type satisfies the following linear system

$$\begin{aligned} &\Phi_C(\cdot, \mathcal{U}) \text{diag}^{1/2}(u_\alpha(\mathcal{U})) (I - [R(\cdot, \mathcal{U}) \text{diag}^{1/2}(u_\alpha(\mathcal{U}))]^T \text{diag}(Ru_\alpha)^{-1} [R(\cdot, \mathcal{U})] \text{diag}^{1/2}(u_\alpha(\mathcal{U}))]) \\ &= \Phi_C(\cdot, \mathcal{I}) \text{diag}(u_\alpha(\mathcal{I})) R(\cdot, \mathcal{I})^T \text{diag}(Ru_\alpha)^{-1} R(\cdot, \mathcal{U}) \text{diag}^{1/2}(u_\alpha(\mathcal{U})) \end{aligned} \quad (24)$$

The matrix $I - [R(\cdot, \mathcal{U}) \text{diag}^{1/2}(u_\alpha(\mathcal{U}))]^T \text{diag}(Ru_\alpha)^{-1} [R(\cdot, \mathcal{U})] \text{diag}^{1/2}(u_\alpha(\mathcal{U}))$ is symmetric semi-positive definite, and $\Phi_C(\cdot, \mathcal{U})$ should be a probability matrix, i.e. $\mathbf{1}^T \Phi_C(\cdot, \mathcal{U}) = \mathbf{1}^T$. In generally, the solution is given by the following constrained least square problem.

$$\begin{aligned} \min_{\Phi_C(\cdot, \mathcal{U})} & \|\Phi_C(\cdot, \mathcal{U}) \text{diag}^{1/2}(u_\alpha(\mathcal{U})) (I - [R(\cdot, \mathcal{U}) \text{diag}^{1/2}(u_\alpha(\mathcal{U}))]^T \text{diag}(Ru_\alpha)^{-1} [R(\cdot, \mathcal{U})] \text{diag}^{1/2}(u_\alpha(\mathcal{U}))]) \\ & \quad - \Phi_C(\cdot, \mathcal{I}) \text{diag}(u_\alpha(\mathcal{I})) R(\cdot, \mathcal{I})^T \text{diag}(Ru_\alpha)^{-1} R(\cdot, \mathcal{U}) \text{diag}^{1/2}(u_\alpha(\mathcal{U}))\|_F \\ \text{s.t.} & \mathbf{1}^T \Phi_C(\cdot, \mathcal{U}) = \mathbf{1}^T \end{aligned} \quad (25)$$

, where $\|A\|_F := \sqrt{\sum_{ij} A_{ij}^2}$ is the Frobenius norm of matrix A .

DINCTA: DATA INTEGRATION AND CELL TYPE ANNOTATION OF SINGLE CELL TRANSCRIPTOMES

Note that the above problem is not easy to solve. Luckily, as in the following theorem, if R satisfies a mild condition, the least square problem without constraint has an explicit unique solution, and we can normalize the solution on each column to make it to the probability matrix as an approximate solution of the constrained least square problem. We summarize it in the following theorem, and proof is in the supplementary.

Theorem 1. *If $R(\cdot, \mathcal{I})u_\alpha(\mathcal{I}) > 0$ elementwisely, then $I - [R(\cdot, \mathcal{U})\text{diag}^{1/2}(u_\alpha(\mathcal{U}))]^T \text{diag}(Ru_\alpha)^{-1} [R(\cdot, \mathcal{U})\text{diag}^{1/2}(u_\alpha(\mathcal{U}))]$ is symmetric positive definite hence invertible and the least square problem*

$$\min_{\Phi_C(\cdot, \mathcal{U})} \|\Phi_C(\cdot, \mathcal{U})\text{diag}^{1/2}(u_\alpha(\mathcal{U}))(I - [R(\cdot, \mathcal{U})\text{diag}^{1/2}(u_\alpha(\mathcal{U}))]^T \text{diag}(Ru_\alpha)^{-1} [R(\cdot, \mathcal{U})\text{diag}^{1/2}(u_\alpha(\mathcal{U}))]) - \Phi_C(\cdot, \mathcal{I})\text{diag}(u_\alpha(\mathcal{I}))R(\cdot, \mathcal{I})^T \text{diag}(Ru_\alpha)^{-1} R(\cdot, \mathcal{U})\text{diag}^{1/2}(u_\alpha(\mathcal{U}))\|_F \quad (26)$$

has an unique solution

$$\begin{aligned} \Phi_C(\cdot, \mathcal{U}) &= \Phi_C(\cdot, \mathcal{I})\text{diag}(u_\alpha(\mathcal{I}))R(\cdot, \mathcal{I})^T \text{diag}(Ru_\alpha)^{-1} R(\cdot, \mathcal{U})\text{diag}^{1/2}(u_\alpha(\mathcal{U})) \\ &\quad [I - [R(\cdot, \mathcal{U})\text{diag}^{1/2}(u_\alpha(\mathcal{U}))]^T \text{diag}(Ru_\alpha)^{-1} [R(\cdot, \mathcal{U})\text{diag}^{1/2}(u_\alpha(\mathcal{U}))]]^{-1} \\ &\quad \text{diag}^{-1/2}(u_\alpha(\mathcal{U})) \\ &= \Phi_C(\cdot, \mathcal{I})\text{diag}(u_\alpha(\mathcal{I}))R(\cdot, \mathcal{I})^T \text{diag}(Ru_\alpha)^{-1} R(\cdot, \mathcal{U})\text{diag}^{1/2}(u_\alpha(\mathcal{U}))[I - A^T A]^{-1} \\ &\quad \text{diag}^{-1/2}(u_\alpha(\mathcal{U})) \end{aligned} \quad (27)$$

and can be computed efficiently in the following representation

$$\Phi_C(\cdot, \mathcal{U}) = \Phi_C(\cdot, \mathcal{I})\text{diag}(u_\alpha(\mathcal{I}))R(\cdot, \mathcal{I})^T \text{diag}(Ru_\alpha)^{-1} R(\cdot, \mathcal{U})\text{diag}^{1/2}(u_\alpha(\mathcal{U})) [I + A^T (I - AA^T)^{-1} A] \text{diag}^{-1/2}(u_\alpha(\mathcal{U})) \quad (28)$$

where $A := \text{diag}(Ru_\alpha)^{-1/2} R(\cdot, \mathcal{U})\text{diag}^{1/2}(u_\alpha(\mathcal{U}))$. And the solution of problem (26) can be approximated by normalize each column of $\Phi_C(\cdot, \mathcal{U})$ to have mass 1 in equation (28). i.e.

$$\Phi_C(\cdot, \mathcal{U}) = \Phi_C(\cdot, \mathcal{U})\text{diag}^{-1}(\mathbf{1}^T \Phi_C(\cdot, \mathcal{U})) \quad (29)$$

In general, the matrix AA^T is semi-positive and the its largest eigenvalue $\lambda_{\max}(AA^T)$ is bound by $\max_k \beta_k$, where $\beta := [R(\cdot, \mathcal{U})u_\alpha(\mathcal{U})]/[Ru_\alpha]$ and the operator $/$ is element-wise division.

If we have enough cells whose cell types are known, the cell types of all cells include in the cells with known cell type, and R finally converged to the correct cluster assignments, we can faithfully infer the unknown cell type's cell type from the known cell type assignments matrix $\Phi_C(\cdot, \mathcal{I})$ through solving the problem (26).

On the other side, if a new cell type in the data set occurs but we do not know initially, then the R_{ki} will catch these new cell type with some new cluster k , and $[R(k, \mathcal{I}_{original})\mathbf{1}]_k = 0$. Then the assumption of Theorem 1 will be violated.

In this time, we can infer the new cell type assignment from $R(k, \mathcal{U}_{original})$ for the core cells of the new cell types. We can gain the intuition from a simple example. If there are two cell types, and cells with the two cell types are separate very well. In the first cell type $c = 1$, there are 2 clusters $k = 1, 2$, in the second cell type $c = 2$, there are 3 clusters $k = 3, 4, 5$. And suppose that the first cell type has N_1 cells, and there are N_2 cells are the second cell type. Then in the ideal case, R should be in the following form

$$R = \begin{bmatrix} 1/2 & \dots & 1/2 & 0 & \dots & 0 \\ 1/2 & \dots & 1/2 & 0 & \dots & 0 \\ 0 & \dots & 0 & 1/3 & \dots & 1/3 \\ 0 & \dots & 0 & 1/3 & \dots & 1/3 \\ 0 & \dots & 0 & 1/3 & \dots & 1/3 \end{bmatrix} \quad (30)$$

Now suppose that we don't know the cell types information, i.e. $\mathcal{I}_{original}$ is empty, can we infer the number of cell types from the R matrix? the answer is yes! The number of cell types is the rank of matrix R , here is 2. We can get Φ_C by combining the rows of R , i.e., $\Phi_C(1, \cdot) = \sum_{k=1,2} R(k, \cdot)$, $\Phi_C(2, \cdot) = \sum_{k=3,4,5} R(k, \cdot)$. Indeed, this gives us an unsupervised method to classify cells with the help of the soft kmeans, which is implemented in DinctaClassification function of Dincta package.

We generalize the above results in the following theorem as the foundation of Dincta to infer new cell types.

DINCTA: DATA INTEGRATION AND CELL TYPE ANNOTATION OF SINGLE CELL TRANSCRIPTOMES

Theorem 2. Suppose that there are N cells belong to C cell types, and each cell type c has N_c cells. Moreover, each cell type separates very well, which means that $\Phi_C \in \{0, 1\}^{C \times N}$. Suppose that Dincta algorithm output an ideal matrix R such that each cluster k only belong to one cell types c . Denote $K_c := \{k | \Pr(c|k) = 1, k = 1, \dots, K\}$, and $K_c := |K_c|$. And suppose that we can reorder the columns of R such that R has the following form

$$R = \begin{bmatrix} B_1 & \mathbf{0} & \cdots & \mathbf{0} \\ \mathbf{0} & B_2 & \cdots & \mathbf{0} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{0} & \mathbf{0} & \cdots & B_C \end{bmatrix} \quad (31)$$

where $B_c \in \mathbb{R}^{K_c \times N_c}$ is

$$B_c = \begin{bmatrix} 1/K_c & \cdots & 1/K_c \\ \vdots & \ddots & \vdots \\ 1/K_c & \cdots & 1/K_c \end{bmatrix} \quad (32)$$

and the corresponding Φ_C has the following form

$$\Phi_C = \begin{bmatrix} \mathbf{1}_{N_1}^T & \mathbf{0} & \cdots & \mathbf{0} \\ \mathbf{0} & \mathbf{1}_{N_2}^T & \cdots & \mathbf{0} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{0} & \mathbf{0} & \cdots & \mathbf{1}_{N_C}^T \end{bmatrix} \quad (33)$$

,where $\mathbf{1}_{N_c}$ is the column vector whose element is 1 and its length is N_c . Let $A := \text{diag}^{-1/2}\{R\mathbf{1}\}R R^T \text{diag}^{-1/2}\{R\mathbf{1}\}$.

Then A has the following form

$$A = \begin{bmatrix} F_1 & \mathbf{0} & \cdots & \mathbf{0} \\ \mathbf{0} & F_2 & \cdots & \mathbf{0} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{0} & \mathbf{0} & \cdots & F_C \end{bmatrix} \quad (34)$$

where $F_c \in \mathbb{R}^{K_c \times K_c}$ is

$$F_c = \begin{bmatrix} 1/K_c & \cdots & 1/K_c \\ \vdots & \ddots & \vdots \\ 1/K_c & \cdots & 1/K_c \end{bmatrix} \quad (35)$$

The rank of Φ_C , R , A are the same, i.e. $\text{rank}(\Phi_C) = \text{rank}(R) = \text{rank}(A) = C$. A is a symmetric semi-positive definite real matrix with eigenvalue 1 of algebraic multiplicity C and eigenvalue 0 of algebraic multiplicity $K - C$. Formally, the eigendecomposition of A has a special form $\hat{U}I\hat{U}^T$, where $I \in \{0, 1\}^{C \times C}$ is the identity matrix and $\hat{U} := U \text{diag}[\frac{1}{\sqrt{K_1}}, \frac{1}{\sqrt{K_2}}, \dots, \frac{1}{\sqrt{K_C}}] \in \mathbb{R}^{K \times C}$ is the columns orthogonal matrix whose columns are eigenvectors of A . And U is represented by

$$U = \begin{bmatrix} \mathbf{1}_{K_1} & \mathbf{0} & \cdots & \mathbf{0} \\ \mathbf{0} & \mathbf{1}_{K_2} & \cdots & \mathbf{0} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{0} & \mathbf{0} & \cdots & \mathbf{1}_{K_C} \end{bmatrix} \quad (36)$$

Moreover, $\Phi_C = U^T R$, i.e. U is also the gather matrix that combines the rows of R to form the cell type assignment matrix Φ_C .

The proof of this theorem is simple and left to the reader.

In the above theorem, it gives us the intuition that the number of cell types is the algebraic multiplicity of eigenvalue 1 of the symmetric semi-positive definite real matrix A . And if one can find the special orthogonal eigenvectors matrix U of A , we can recover the cluster assignment matrix R to the cell type assignment matrix Φ_C . As the assumption in the theorem, this phenomenon only occurs when that the cell type in the dataset is separate very well. In the reality, the dataset may be noisy, and also there are intermediate cell states which restrict the cluster assignment matrix do not satisfy the well separation assumption. Although there are intermediate states and cells in the boundary between two

DINCTA: DATA INTEGRATION AND CELL TYPE ANNOTATION OF SINGLE CELL TRANSCRIPTOMES

cell types, we can always find the core cells in the center of each cell type, and the core cells of each cell type satisfy the well separation assumption. We can select out those cells out and use the cluster assignments of the core cells to form the matrix A . Then A can determine the number of cell types and the gather matrix U . And the cell type assignments for the core cells can be obtained from U and R . Once we know the cell type assignments for the core cells with the new cell types, we regard these cell type assignments as known information. Combining with the already known cell type information, we can use the formula (28) and (29) to infer the cell types of the other cells on the boundary of cell types or the intermediate state cells.

In order to make the ideas to become computable, we must find an algorithm to compute the gather matrix U . Below, we elucidate the intuition to make this happen. We first use the eigenvalue algorithm to find the eigenvalues of A and determine the C as the rank of A . Note that the number of 1s in U is K , so we can compute the U by find a path that join (add) the columns of the identity matrix $I \in \mathbb{R}^{K \times K}$ sequentially to get the U . To do this, we need an objective to determine which two columns to join in each time.

We now first give our objective function, and then give a simple example to illustrate the key ideas.

Let the $U_K = I$ be the start point of U , and $U_{K-t} \in \mathbb{R}^{K \times (K-t)}$ is the result after t times sequentially combining the columns of U_K . And after $t = K - C$ combination, we get that the gather matrix $U_C = U$. Suppose that we are in the step $t \geq 1$, and currently we have the $U_{K-t+1} \in \mathbb{R}^{K \times (K-t+1)}$, we want to find the index which two columns of U_{K-t+1} should be combined to get the $U_{K-t} \in \mathbb{R}^{K \times (K-t)}$. The key idea is that we find which is the smallest element of the off-diagonal elements of the objective matrix $D_{K-t+1} := [AU_{K-t+1} - U_{K-t+1}]^T [AU_{K-t+1} - U_{K-t+1}]$. If there are multiple smallest elements, we select the element whose index is (i, j) such that $D_{K-t+1}(i, i) + D_{K-t+1}(j, j)$ is the maximum among all these multiple smallest elements. And suppose that is element index is (c_1, c_2) , then we should combine the c_1, c_2 of U_{K-t+1} by adding to get the U_{K-t} .

To get some insights, let us think a simple case. Suppose that there is only one cell type, and this cell type is represented by K clusters. Now $A := \frac{1}{K} \mathbf{1}\mathbf{1}^T \in \mathbb{R}^{K \times K}$, and we have $U = \mathbf{1} \in \mathbb{R}^{K \times 1}$. One can easily imagine that we can get U by sequentially combining the columns of initially identity matrix $U_K = I \in \mathbb{R}^{K \times K}$ to get the final $U := \mathbf{1}$. At step 1, $D_K = (AI - I)^T (AI - I) = (AI - I)^T (A - I)$, one can easily verify that

$$D_K = \begin{bmatrix} 1 - \frac{1}{K} & -\frac{1}{K} & -\frac{1}{K} & \cdots & -\frac{1}{K} \\ -\frac{1}{K} & 1 - \frac{1}{K} & -\frac{1}{K} & \cdots & -\frac{1}{K} \\ -\frac{1}{K} & -\frac{1}{K} & 1 - \frac{1}{K} & \cdots & -\frac{1}{K} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ -\frac{1}{K} & -\frac{1}{K} & -\frac{1}{K} & \cdots & 1 - \frac{1}{K} \end{bmatrix}$$

Note that each row sum, column sum and the sum of all elements of D_K is zero, i.e. $\sum_i D_K(i, j) = 0$, $\sum_j D_K(i, j) = 0$, $\sum_{ij} D_K(i, j) = 0$. On this time we can random select two columns of U_K to combine. For simplicity, we select the first two columns of U_K to combine, this will give us the following U_{K-1} :

$$U_{K-1} = \begin{bmatrix} 1 & 0 & \cdots & 0 \\ 1 & 0 & \cdots & 0 \\ 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 1 \end{bmatrix}$$

At step 2, $D_{K-1} = (AU_{K-1} - U_{K-1})^T (AU_{K-1} - U_{K-1})$, one can easily verify that

$$D_{K-1} = \begin{bmatrix} 2 - \frac{4}{K} & -\frac{2}{K} & -\frac{2}{K} & \cdots & -\frac{2}{K} \\ -\frac{2}{K} & 1 - \frac{1}{K} & -\frac{1}{K} & \cdots & -\frac{1}{K} \\ -\frac{2}{K} & -\frac{1}{K} & 1 - \frac{1}{K} & \cdots & -\frac{1}{K} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ -\frac{2}{K} & -\frac{1}{K} & -\frac{1}{K} & \cdots & 1 - \frac{1}{K} \end{bmatrix}$$

One can verify that the D_{K-1} can be obtained by combine the first two rows of D_K by adding to get $\hat{D}_K$ and then combine the first two columns of $\hat{D}_K$ by adding to get D_{K-1} . And in general, it is true, which can save our computation resources. Repeat above steps, we will finally get the $U = \mathbf{1}$ and $D_1 = 0$.

DINCTA: DATA INTEGRATION AND CELL TYPE ANNOTATION OF SINGLE CELL TRANSCRIPTOMES

We summarize the above results in the following more general form of A , and the proof is simple but tedious and omits.

Theorem 3. Suppose that A has the following form

$$A = \begin{bmatrix} F_1 & \mathbf{0} & \cdots & \mathbf{0} \\ \mathbf{0} & F_2 & \cdots & \mathbf{0} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{0} & \mathbf{0} & \cdots & F_C \end{bmatrix} \quad (37)$$

where $F_C \in \mathbb{R}^{K_C \times K_C}$ is

$$F_C = \begin{bmatrix} 1/K_C & \cdots & 1/K_C \\ \vdots & \ddots & \vdots \\ 1/K_C & \cdots & 1/K_C \end{bmatrix} \quad (38)$$

Then A is a symmetric semi-positive definite real matrix with the eigendecomposition of $A := \hat{U}I\hat{U}^T$, where $I \in \{0, 1\}^{C \times C}$ is the identity matrix and $\hat{U} := U \text{diag}[\frac{1}{\sqrt{K_1}}, \frac{1}{\sqrt{K_2}}, \dots, \frac{1}{\sqrt{K_C}}] \in \mathbb{R}^{K \times C}$ is the columns orthogonal matrix whose columns are eigenvectors of A . And U is represented by

$$U = \begin{bmatrix} \mathbf{1}_{K_1} & \mathbf{0} & \cdots & \mathbf{0} \\ \mathbf{0} & \mathbf{1}_{K_2} & \cdots & \mathbf{0} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{0} & \mathbf{0} & \cdots & \mathbf{1}_{K_C} \end{bmatrix} \quad (39)$$

We can get U by the following algorithm. Let the $U_K = I$ be the start point of U , and $U_{K-t} \in \mathbb{R}^{K \times (K-t)}$ is the result after t times sequentially combining the columns of U_K . In the step $t \geq 1$, we have $U_{K-t+1} \in \mathbb{R}^{K \times (K-t+1)}$. Define the objective matrix $D_{K-t+1} := [AU_{K-t+1} - U_{K-t+1}]^T [AU_{K-t+1} - U_{K-t+1}]$. And let m be the minimum of the off-diagonal elements of D_{K-t+1} , i.e. $m := \min_{i \neq j} D_{K-t+1}(i, j)$, and let $\mathcal{M} := \{(i, j) | D_{K-t+1}(i, j) = m, i \neq j\}$ be the index set achieved the minimum. We find the matrix index (c_1, c_2) such that $D_{K-t+1}(c_1, c_1) + D_{K-t+1}(c_2, c_2) = \max\{D_{K-t+1}(i, i) + D_{K-t+1}(j, j) | (i, j) \in \mathcal{M}, (c_1, c_2) \in \mathcal{M}, c_1 < c_2\}$. U_{K-t} can be obtained by combining the c_1, c_2 columns of U_{K-t+1} by adding. And $D_{K-t} := [AU_{K-t} - U_{K-t}]^T [AU_{K-t} - U_{K-t}]$ can be obtained by combining c_1, c_2 columns of D_{K-t+1} to get $\hat{D}_{K-t+1}$, and then combine the c_1, c_2 rows of $\hat{D}_{K-t+1}$ by adding. And after $t = K - C$ combinations, we get that the gather matrix $U_C = U$.

We summarize above theorem in the Algorithm 3.

Now we can use this basic idea to infer the new cell types. Note that once there are new cell types in the cells with index $\mathcal{U}_{original}$, the soft kmeans clustering will catch the new cell types with some clusters. The indicator of this situations occuring is that $[R(k, \mathcal{I}_{original})\mathbf{1}]_k$ is near 0. In this case, we gather all new clusters $\mathcal{K}_{new} = \{k | [R(k, \mathcal{I}_{original})\mathbf{1}]_k < \delta\}$, with a threshold δ . Usually $\delta = 6$ is enough. We gather the index of the cells which belong to the new cell types to $\mathcal{U}_{candidate} := \{i | \sum_{k \in \mathcal{K}_{new}} R_{ki} > p_{new}, i \in \mathcal{U}_{original}\}$ with the probability threshold p_{new} , e.g. $p_{new} = 0.8$.

In the progress of Dincta, the information in R is noisy, R is biased to the ideal cluster assignment matrix. To reduce the impact of the noise, we select cells from $\mathcal{U}_{candidate}$ such that cells are in the center of the cluster it belongs to. We called these cells as the main cells, and they are gathered together to form the main set which is the index set of these main cells. We now explain the strategy used to gather the main set. The idea is very simple. We first estimate the effective number cells in cluster $k \in \mathcal{K}_{new}$ by $N_k := \sum_{i \in \mathcal{U}_{candidate}} R_{ki}$. Then we set the select fraction γ_s of N_k for the each k set, respectively. For each $k \in \mathcal{K}_{new}$, we sort the $R(k, \mathcal{U}_{candidate})$ about the cell i with value R_{ki} in descend order. And we select the top $\gamma_m N_k$ cells ids to the main set, and finally join all ids to get the main set $\mathcal{U}_m$. As we shows later, we will fill the cell type assignment matrix $\Phi_C(\cdot, \mathcal{U}_m)$.

To finish the inference of new cell type task, we must determine the number of new cell types and find the gather matrix U to transform the cluster assignment to the cell type assignment for the cells in the main set. And we can use the results of Theorem 3 to finish this.

Note that in the above simple example, we see that the number of the cell types is determined by the rank of the cluster assignment for those core cells which has a clear cell type, i.e. not on the boundary of two cell types. We now find a strategy to select the core cells from the whole data points, then we gather all these cells index into the sample set S .

Algorithm 3 Find_Gather_U

```

1: function FIND_GATHER_U( $A, C$ )
2:    $U_K = I \in \mathbb{R}^{K \times K}$ 
3:    $D_K = [AU_K - U_K]^T[AU_K - U_K]$ 
4:   for  $t \leftarrow 1, K - C$  do
5:     Find the minimum of the off-diagonal of matrix  $D_{K-t+1}$ ,  $m = \min_{i \neq j} D_K(i, j)$ .
6:     Find the index of  $D_{K-t+1}$  on which it achieves the minimum  $m$ :  $\mathcal{M} := \{(i, j) | D_K(i, j) = m\}$ .
7:     Find the index  $(c_1, c_2) \in \mathcal{M}$  such that  $D_{K-t+1}(c_1, c_1) + D_{K-t+1}(c_2, c_2) = \max\{D_{K-t+1}(i, i) + D_{K-t+1}(j, j) | (i, j) \in \mathcal{M}, i \neq j\}$  and  $c_1 < c_2$ .
8:     Reorder the columns of  $U_{K-t+1}$ , such that the  $c_1, c_2$  columns to be the last two columns, and keep the relative order of the other columns and move them to the first  $K - t - 1$  columns. Also reorder the rows and columns of  $D_{K-t+1}$ , such that  $c_1, c_2$  rows and  $c_1, c_2$  columns to be the last two rows and columns, respectively, and keep the relative order of the other rows and columns and move them to the first  $K - t - 1$  rows and  $K - t - 1$  columns, respectively. Denote the reordered matrix still as the  $U_{K-t+1}$  and  $D_{K-t+1}$ .
9:     Combine the last two columns of  $U_{K-t+1}$  by adding to get  $U_{K-t}$ , i.e.
           
$$U_{K-t}(\cdot, 1 : (K - t - 1)) := U_{K-t+1}(\cdot, 1 : (K - t - 1))$$

           
$$U_{K-t}(\cdot, K - t) := U_{K-t+1}(\cdot, K - t) + U_{K-t+1}(\cdot, K - t + 1)$$

10:    Combine the last two rows and columns of  $D_{K-t+1}$  to get  $D_{K-t}$ , i.e.
           
$$D_{K-t}(1 : (K - t - 1), 1 : (K - t - 1)) := U_{K-t+1}(1 : (K - t - 1), 1 : (K - t - 1))$$

           
$$U_{K-t}(1 : (K - t - 1), K - t) := U_{K-t+1}(1 : (K - t - 1), K - t)$$

           
$$U_{K-t}(K - t, 1 : (K - t - 1)) := U_{K-t+1}(K - t, 1 : (K - t - 1))$$

           
$$U_{K-t}(K - t, K - t) := U_{K-t+1}(K - t, K - t)$$

           
$$U_{K-t}(K - t, K - t + 1) := U_{K-t+1}(K - t, K - t + 1)$$

           
$$U_{K-t}(K - t + 1, 1 : (K - t - 1)) := U_{K-t+1}(K - t + 1, 1 : (K - t - 1))$$

           
$$U_{K-t}(K - t + 1, K - t) := U_{K-t+1}(K - t + 1, K - t)$$

           
$$U_{K-t}(K - t + 1, K - t + 1) := U_{K-t+1}(K - t + 1, K - t + 1)$$

           
$$U_{K-t}(K - t + 1, K - t) := U_{K-t+1}(K - t + 1, K - t)$$

11:   return  $U_C$ 

```

To get the sample set, we use the strategy similar as we do to form the main set. We first estimate the effective number of cells in each cluster $k \in \{1, 2, \dots, K\}$ by $N_k := \sum_{i=1}^N R_{ki}$. Then we set the select fraction γ_S of N_k for the each k set, respectively. For each $k \in \{1, 2, \dots, K\}$, we sort the $R(k, \cdot)$ about the cell i in descend order. Then we select the top $\gamma_S N_k$ cells to the sample set, and finally join all ids to get the sample set $\mathcal{S}$.

After obtaining the sample set $\mathcal{S}$, we compute the eigen decomposition of $A_S := \text{diag}^{-1/2}(R(\cdot, \mathcal{S})\mathbf{1})R(\cdot, \mathcal{S})R(\cdot, \mathcal{S})^T \text{diag}^{-1/2}(R(\cdot, \mathcal{S})\mathbf{1})$. From the Theorem 3, the number of cell types is determined by algebraic multiplicity of eigenvalue 1 of A_S . Note that there are noise in estimate R , we use the $C_{all} = |\{k | \lambda_{A_S}(k) > \delta_\lambda\}|$ to determine of number of the cell types of the whole data, where $\delta_\lambda \in (0, 1]$ is a threshold, e.g. $\delta_\lambda = 0.9$. Then we get the gather matrix $U_S \in \mathbb{R}^{C_{all} \times K}$ for A_S by the Algorithm 3.

To determine the number of new cell types C_{new} and the corresponding gather matrix $U_{new} \in \mathbb{R}^{C_{new} \times K}$ for the new cell types, we need to determine which columns of U_S are the gather vectors for the new cell types. The indicator that one column of U_S is the gather vector of new cell types is that $\frac{\mathbf{1}^T U_S(K_{new}, c)}{\mathbf{1}^T U_S(\cdot, c)}$ close to 1. And we denote the index of new cell type as $C_{new} := \{c | \frac{\mathbf{1}^T U_S(K_{new}, c)}{\mathbf{1}^T U_S(\cdot, c)} > \delta_C\}$ where $\delta_C \in (0, 1]$ is threshold which determines the closeness to 1, usually $\delta_C = 0.5$ is enough. And the number of new cell types is given by $C_{new} := |C_{new}|$. Finally, the gather matrix $U_{new} := U_S(\cdot, C_{new})$, and $\Phi_C(C_{new}, \mathcal{U}_{main}) = U_{new}^T R(\cdot, \mathcal{U}_{main})$ and $\Phi_C(\{1, \dots, C\} - C_{new}, \mathcal{U}_{main}) = \mathbf{0}$.

Get all the efforts together, if the data contains new cell types, we can use the method described to find the main set of cells which belong to the new cell type. Then we get the cell assignment matrix for these main cells. After that, we combine the cell type information of the original cells $\mathcal{I}_{original}$ with the main cells with the new cell type $\mathcal{U}_{main}$ together, i.e. $\mathcal{I} = \mathcal{I}_{original} \cup \mathcal{U}_{new}$. And the index of cells with unknown cell type reduced to $\mathcal{U} = \mathcal{U}_{original} - \mathcal{U}_{main}$. We can infer the cell type these cells by the equation (28) and (29).

DINCTA: DATA INTEGRATION AND CELL TYPE ANNOTATION OF SINGLE CELL TRANSCRIPTOMES

We summarize the method which infers the cell type information matrix Φ_C from the cluster assignment matrix R in the Algorithm 4.

Algorithm 4 Infer Cell Type Information

```

1: function UPDATE_PHI_C(  $R, \Phi_C, u, \alpha, C_{known}$  )
2:    $u_\alpha = \alpha u + (1 - u)$ 
3:   The index of cells with cell type known currently.  $\mathcal{I}_{original} = \{i | u_i = 0\}$ .
4:   The index of cells with cell type unknown currently.  $\mathcal{U}_{original} := \{i | u_i = 1\}$ .
5:   Initialize the index of cells with the new cell types  $\mathcal{U}_{main} = \emptyset$ . And the index of the core cells  $S = \emptyset$ 
6:    $\mathcal{K}_{new} = \{k | [R(\cdot, \mathcal{I})\mathbf{1}]_k < \delta\}$ 
7:   {check whether occurs the new clusters}
8:   if  $|\mathcal{K}_{new}| > 0$  then
9:     {gather the index of candidate cells with new cell types}
10:     $\mathcal{U}_{candidate} := \{i | \sum_{k \in \mathcal{K}_{new}} R_{ki} > p_{new}, i \in \mathcal{U}\}$ 
11:    {gather the index of main cells with new cell types}
12:    for  $k \in \mathcal{K}_{new}$  do
13:       $N_k := \sum_{i \in \mathcal{U}_{candidate}} R_{ki}$ 
14:      Let  $\mathcal{T}$  be the sort index of  $R_{k, \mathcal{U}_{candidate}}$  about  $i$  with the value  $R_{ki}$  in the descend order.
15:       $\mathcal{U}_{main} = \mathcal{U}_{main} \cup \{\text{The cell index of top } \gamma_m N_k \text{ of } \mathcal{T}\}$ 
16:    {gather the index of core cells}
17:    for  $k \leftarrow 1, \dots, K$  do
18:       $N_k := \sum_{i=1}^N R_{ki}$ 
19:      Let  $\mathcal{T}$  be the sort index of  $R(k, \cdot)$  about  $i$  with the value  $R_{ki}$  in the descend order.
20:       $S = S \cup \{\text{The cell index of top } \gamma_s N_k \text{ of } \mathcal{T}\}$ 
21:     $A := \text{diag}^{-1/2} \{R(\cdot, S)\mathbf{1}\} R(\cdot, S) R(\cdot, S)^T \text{diag}^{-1/2} \{R(\cdot, S)\mathbf{1}\}$ 
22:    Compute the eigenvalues of  $A$ , denote it as  $\lambda_A \in \mathbb{R}^K$ 
23:     $C_{all} := |\{k | \lambda_A(k) > \delta_\lambda\}|$ 
24:     $\mathcal{U} \leftarrow \text{Find\_Gather\_U}(A, C_{all})$ 
25:    {find the index of new cell types}
26:     $C_{new} := \{c | \frac{\mathbf{1}^T U_S(\mathcal{K}_{new}, c)}{\mathbf{1}^T U_S(\cdot, c)} > \delta_c\}$ 
27:     $C_{new} := |C_{new}|$  and  $C := C_{known} + 1 + C_{new}$ 
28:     $\mathcal{U}_{new} := U(C_{new}, \cdot)$ 
29:    {infer the cell type assignments for the main cells}
30:     $\Phi_C(\cdot, \mathcal{U}_{main}) = \mathbf{0}$ 
31:     $\Phi_C((C_{known} + 1) : C, \mathcal{U}_{main}) = U_{new}^T R(\cdot, \mathcal{U}_{main})$ 
32:    normalize each columns of  $\Phi_C(\cdot, \mathcal{U}_{main})$  such that the sum of each column is 1.
33:   $\mathcal{U} = \mathcal{U}_{original} \cup \mathcal{U}_{main}$ 
34:   $\mathcal{I} = \mathcal{I}_{original} \cup \mathcal{U}_{main}$ 
35:  Update  $\Phi_C(\cdot, \mathcal{U})$  with the equation (28) and (29).
36:  return  $\Phi_C$ 

```

Linear mixture model correction. We use the same correction model as in Korsunsky et al. (2018). To make the paper easy to read, we repeat the idea here.

In this section, we refer to all effects to be integrated out of the original embedding as batch effects. This dose not indicate that these effects are just technical. The terminology is only meant for convenient.

Mixture of experts model. Once we get the batch-diverse clusters assignments, we would like to remove batch-specific variation from each cluster. We achieve this with a variation on the original mixture of experts model by Jordan and Jacobs Jordan and Jacobs (1994). In the context of Dincta, each cell is probabilistically assigned to a small set of experts. This assignment was computed previously in the clustering and inferring step. Conditioned on a cluster/expert, mixture of experts assumes a linear relationship between the response and independent variables. Thus, we condition on cluster/expert k and define a Gaussian probability distribution for the response variables.

$$Z | \{\Phi_i^*, R_i = k\} = \mathcal{N}(Z_i | W_k^T \Phi_{B_i}^*, \sigma_k^2 \mathbf{I}) \quad (40)$$

DINCTA: DATA INTEGRATION AND CELL TYPE ANNOTATION OF SINGLE CELL TRANSCRIPTOMES

Here, the mean is $\mu_k = W_k^T \Phi_B^*$, and the covariance is $\sigma_k^2 \mathbf{I}$ for simplicity. Note that the design matrix above (Φ_B^*) is not the same as the one used in the clustering step (Φ_B). We augment the original design matrix Φ_B to include an intercept term: $\Phi_B^* = [\mathbf{1}^T; \Phi_B]$. These intercept terms capture batch-independent (that is, cell type) variation in each cluster or expert. We can also achieve more complex behavior, such as reference mapping (Reference mapping) by modifying Φ_B^* .

$$\Phi_B^* = [\mathbf{1}^T; \Phi_B] = \begin{bmatrix} 1 & \cdots & 1 \\ \Phi_B(1, 1) & \cdots & \Phi_B(1, N) \\ \vdots & \ddots & \vdots \\ \Phi_B(B, 1) & \cdots & \Phi_B(B, N) \end{bmatrix} \quad (41)$$

With this generative formulation, we can solve for the parameters (W_k) of the linear model for each cluster or expert independently.

$$W_k = (\Phi_B^* \text{diag}(R_k) \Phi_B^{*T})^{-1} \Phi_B^* \text{diag}(R_k) Z^T \quad (42)$$

Above, $\text{diag}(R_k)$ is the diagonal matrix of cluster membership terms for cluster k . Z is the matrix of original PCA embeddings.

$$\text{diag}(R_k) = \begin{bmatrix} R_{k1} & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & R_{kN} \end{bmatrix}$$

Each column in W_k corresponds to a PC dimension, from PC_1 to PC_d . The first row of W_k corresponds to batch-independent intercept terms. The subsequent rows (1 to B) correspond to one-hot encoded batch assignments from the original design matrix Φ_B , i.e. the batch effects vectors.

To get the cell-specific correction values, we take the expectation of W_k with respect to the cluster assignment probability distribution R . In particular, each cell is modeled by a batch-independent intercept, represented in the first row of W_k ($W_{k[0, \cdot]}$) and its batch-dependent terms represented by the remaining rows $W_{k[1:B, \cdot]}$.

$$Z_i = \sum_k R_{ki} [W_{k[0, \cdot]} + W_{k[1:B, \cdot]}^T \Phi_B^*(1 : B, i)] + \epsilon_i$$

We split W_k into two parts because we want to retain the intercept terms and remove the batch-dependent terms. To get the corrected embeddings ($\hat{Z}_i$), we subtract the batch-specific term ($\sum_k W_{k[1:B, \cdot]}^T \Phi_B^*(1 : B, i)^*$) from the original embedding (Z).

$$\hat{Z}_i = Z_i - \sum_k R_{ki} W_{k[1:B, \cdot]}^T \Phi_B^*(1 : B, i) \quad (43)$$

What remains is the cell-type specific intercept $W_{k[0, \cdot]}$ and the cell-specific residual ϵ_i .

$$\hat{Z}_i = \sum_k R_{ki} W_{k[0, \cdot]}^T + \epsilon_i \quad (44)$$

Unfortunately, for the design matrix Φ_B^* , the formulation in equation (44) does not have a solution. This is because Φ_B^* is not full rank and thus $\Phi_B^* \text{diag}(R_k) \Phi_B^{*T}$ is not invertible. This singularity arise from the fact that the sum of a one-hot encoded categorical variable is equal to the intercept. To address this colinearity, we penalize nonintercept term in W_k with an L_2 norm, akin to ridge regression. This shrinks the W_k terms to 0. Just as in ridge regression, instead of inverting $\Phi_B^* \text{diag}(R_k) \Phi_B^{*T}$, we invert $\Phi_B^* \text{diag}(R_k) \Phi_B^{*T} + \lambda \mathbf{I}$, which is not singular, where $\mathbf{I} := \text{diag}([0, 1, \dots, 1])$. The solution for W_k now becomes

$$W_k = (\Phi_B^* \text{diag}(R_k) \Phi_B^{*T} + \lambda \mathbf{I})^{-1} \Phi_B^* \text{diag}(R_k) Z^T \quad (45)$$

Algorithm 5 Mixture of Experts Correct

```

1: function CORRECT( $Z, R, \Phi_B$ )
2:    $\hat{Z} \leftarrow Z$ 
3:    $\Phi_B^* = [\mathbf{1}^T; \Phi_B]$ 
4:   for  $k \leftarrow 1, \dots, K$  do
5:      $W_k = (\Phi_B^* \text{diag}(R_k) \Phi_B^{*T} + \lambda \mathbf{I})^{-1} \Phi_B^* \text{diag}(R_k) Z^T$ 
6:      $W_{k[0,:]} \leftarrow \mathbf{0}$ 
7:      $\hat{Z} = Z - W_k^T \Phi_B^* \text{diag}(R_k)$ 
8:   return  $\hat{Z}$ 

```

λ is the ridge penalty hyperparameter. Larger values of λ will shrink W_k more toward 0. In practice, we set $\lambda = 1$. These correction steps are summarized in Algorithm 5.

To understand that how this correction will move the embedding of the cells with the same cell type clustering together, we look at a very simple example. Suppose that there are only two batch, and each batch contains only one common cell type. And suppose that Φ_B^* be in the following form:

$$\Phi_B^* = \begin{bmatrix} 1 & \dots & 1 & 1 & \dots & 1 \\ 1 & \dots & 1 & 0 & \dots & 0 \\ 0 & \dots & 0 & 1 & \dots & 1 \end{bmatrix}$$

And $R \in \mathbb{R}^{2 \times N}$ in the following ideal form.

$$R = \begin{bmatrix} 1/2 & \dots & 1/2 & 1/2 & \dots & 1/2 \\ 1/2 & \dots & 1/2 & 1/2 & \dots & 1/2 \end{bmatrix}$$

Now,

$$W_1 = W_2 = \begin{bmatrix} N/2 & N_1/2 & N_2/2 \\ N_1/2 & N_1/2 + \lambda & 0 \\ N_2/2 & 0 & N_2/2 + \lambda \end{bmatrix}^{-1} \begin{bmatrix} 1/2 \sum_{i=1}^N Z_i^T \\ 1/2 \sum_{i=1}^{N_1} Z_i^T \\ 1/2 \sum_{i=N_1+1}^N Z_i^T \end{bmatrix}$$

. Let $\lambda = 1$, with some mathematical calculations, we have

$$W_1 = W_2 = \frac{1}{2N(1 + \frac{N_1 N_2}{N})} \begin{bmatrix} N_1 N_2 (\bar{Z}_{B_1}^T + \bar{Z}_{B_2}^T) + 2N \bar{Z}^T \\ N_1 N_2 (\bar{Z}_{B_1}^T - \bar{Z}_{B_2}^T) \\ N_1 N_2 (\bar{Z}_{B_2}^T - \bar{Z}_{B_1}^T) \end{bmatrix}$$

. where $\bar{Z} := 1/N \sum_{i=1}^N Z_i$, $\bar{Z}_{B_1} := 1/N_1 \sum_{i=1}^{N_1} Z_i$ and $\bar{Z}_{B_2} := 1/N_2 \sum_{i=N_1+1}^N Z_i$. Suppose that $N_1 = N_2 = N/2$, then

$$W_1 = W_2 \approx \begin{bmatrix} \frac{1}{2}(\bar{Z}_{B_1}^T + \bar{Z}_{B_2}^T) \\ \frac{1}{2}(\bar{Z}_{B_1}^T - \bar{Z}_{B_2}^T) \\ \frac{1}{2}(\bar{Z}_{B_2}^T - \bar{Z}_{B_1}^T) \end{bmatrix}$$

.

With the equation (43), the embedding of cells in the batch 1 B_1 will be corrected by

$$\hat{Z}_i = Z_i - 1/2(\bar{Z}_{B_1} - \bar{Z}_{B_2})$$

. i.e. it will correct the embedding by move Z_i toward the middle point $\bar{Z}$ between the centroid $\bar{Z}_{B_1}$ of batch 1 and centroid $\bar{Z}_{B_2}$ of batch 2. Similarly, the embedding of cells in the batch 2 B_2 will be corrected by

$$\hat{Z}_i = Z_i - 1/2(\bar{Z}_{B_2} - \bar{Z}_{B_1})$$

. i.e. it will correct the embedding by move Z_i toward the middle point $\bar{Z}$ between the centroid $\bar{Z}_{B_1}$ of batch 1 and centroid $\bar{Z}_{B_2}$ of batch 2. After correcting, the embedding of cells in the two batches will mix together, and move toward the center point $\bar{Z} := 1/N \sum_{i=1}^N \hat{Z}_i$.

DINCTA: DATA INTEGRATION AND CELL TYPE ANNOTATION OF SINGLE CELL TRANSCRIPTOMES

Reference mapping. We can map query datasets onto a reference dataset, which is proposed in Korsunsky et al. (2018). We achieved this by modifying Φ_B^* so that batch terms for reference cells do not get models or corrected. For every cell i in a reference dataset, set $\Phi_B^*(i) = [1, 0, \dots, 0]$. This makes Dincta explain cell i in terms of an intercept and nothing else. Since intercept terms do not get removed, cell i never changes its embedding (that is, $Z_i = \hat{Z}_i$).

Caveat. This section assumes the modeled data are orthogonal and each normally distributed. This is not true for the L_2 normalized data used in the spherical clustering. Regression in this space requires the estimation and interpolation of rotation matrices, a difficult problem. We instead perform batch correction in the unnormalized space. The corrected data $\hat{Z}$ are the L_2 -normalized for the next iteration of clustering.

Refine the cell type information. Once the inner loop converges in the Algorithm 1, we can update the unknown cell type information indicator vector u of cells. Here, we should find a criteria to identify which cell's cell type is correct assigned. We do this by comparing KL divergence of Φ_C and $\Psi_C := \Phi_C R^T \text{diag}^{-1}(R\mathbf{1})R$ for each cell. In detail, the KL divergence distance of $\Phi_C(\cdot, i)$ and $\Psi_C(\cdot, i)$ is $d(i) := \sum_k \Phi_C(k, i) \log \frac{\Phi_C(k, i)}{\Psi_C(k, i)}$. And we select the cells with the index $\mathcal{U} = \{i | d(i) > \delta_d\}$ as the cells with unknown cell type, where $\delta_d \in [0, \infty)$ is a threshold, usually $\delta_d = 1e-2$ is enough. Then we update u with $u(\mathcal{U}) = \mathbf{1}$ and $u(\mathcal{I}) = \mathbf{0}$ where $\mathcal{I} = \{1, 2, \dots, N\} - \mathcal{U}$. We summarize these steps in the Algorithm 6

Algorithm 6 Refine cell type assignment

```

1: function REFINE( $R, \Phi_C$ )
2:    $\Psi_C := \Phi_C R^T \text{diag}^{-1}(R\mathbf{1})R$ 
3:   for  $i \leftarrow 1, \dots, N$  do
4:      $d(i) := \sum_k \Phi_C(k, i) \log \frac{\Phi_C(k, i)}{\Psi_C(k, i)}$ 
5:      $\mathcal{U} = \{i | d(i) > \delta_d\}$  and  $\mathcal{I} = \{1, 2, \dots, N\} - \mathcal{U}$ 
6:      $u(\mathcal{U}) = \mathbf{1}$  and  $u(\mathcal{I}) = \mathbf{0}$ 
7:   return  $u$ 
```

Dincta Classification Note that if there exists only one sample, and there is no need to do the batch correcting. We can shrink this model just by removing the batch variable and correcting step. This is implemented in Dincta by the function DinctaClassification. We omit the description of the algorithm and the numerical results for it to save space and make the main idea more clear.

Infer Cell Type from the Cluster Assignment Matrix To compare inferring accuracy, we can infer the number of cell types from directly from the cluster assignment matrix R . And this method also suit for any algorithm output a cluster assignment matrix.

The basic idea is that we define $A := \text{diag}^{-1/2}(R\mathbf{1})RR^T \text{diag}^{-1/2}(R\mathbf{1})$, and the number of eigenvalues of A which are close to 1 as the number of cell types, i.e. $C := |\{k | \lambda_A(k) > \delta_A\}|$. Moreover, if we have already know there are C_{known} cell types, we set $C := \max(C_{known}, |\{k | \lambda_A(k) > \delta_A\}|)$. Then we use the find gather matrix algorithm 3 to find the gather matrix U . Finally, the cell type assignment matrix can be obtained by $\Phi_C := U^T R$. We summarize it in the following algorithm 7

Algorithm 7 Infer Cell Type Information From Cluster Assignment Matrix

```

1: function INFER_PHI_C( $R, C_{known}$ )
2:    $A := \text{diag}^{-1/2}(R\mathbf{1})RR^T \text{diag}^{-1/2}(R\mathbf{1})$ 
3:    $C := \max(C_{known}, |\{k | \lambda_A(k) > \delta_A\}|)$ 
4:    $U = \text{Find\_Gather\_U}(A, C)$ 
5:    $\Phi_C = U^T R$ 
6:   return  $\Phi_C$ 
```

Performance and benchmarking. Assessing the degree of mixing during batch correction and dataset integration is an open problem. Several group have proposed methods to quantify the diversity of batches within local neighborhoods, defined by kNN graphs, of the embedded space. Butler et al. (2018) provide a statistical test to evaluate the degree of mixing, while Azizi et al. (2018) report the entropy of these distributions. Korsunsky et al. (2018) provide a interpretable metric LISI which is sensitive to the local diversity to quantify the diversity of mixing. In this paper, we use the LISI metric.

DINCTA: DATA INTEGRATION AND CELL TYPE ANNOTATION OF SINGLE CELL TRANSCRIPTOMES

Match Score We use the $m_s := \sum_i d(i)$ as the match score between the cell type assignment matrix and the cluster assignments matrix, where $d(i) := \sum_k \Phi_C(k, i) \log \frac{\Phi_C(k, i)}{\Psi_C(k, i)}$ and $\Psi_C := \Phi_C R^T \text{diag}^{-1}(R\mathbf{1})R$.

Comparison Between Cell Type Assignment Matrix Now, suppose that we have two cell type assignment matrix, one is the golden cell type assignment matrix Φ_C by the experts and the other is the inferred cell type assignment matrix Ψ_C such as from Dincta or output by the Algorithm 7, we propose the following algorithm 8 to compute the inferring accuracy of the Ψ_C .

Algorithm 8 Inferring Accuracy

```

1: function INFER_ACCURACY( $\Phi_C, \Psi_C$ )
2:    $O = \Psi_C \Phi_C^T$ 
3:    $P = \text{diag}^{-1}(O\mathbf{1})O \in \mathbb{R}^{C_p \times C_r}$ 
4:   Find the index vector  $m \in \mathbb{R}^{C_p}$ , such that  $m(c_p) := \{c_r | P(c_p, c_r) = \max P(c_p, \cdot)\}$ , i.e.  $m(c_p)$  is the index of position on which  $P(c_p, \cdot)$  achieves the maximum on row  $c_p$ . When there is a tie on the maximum, random chose one of the index on which achieve the maximum.
5:    $n_{correct} = 0$ 
6:   for  $i \leftarrow 1 : N$  do
7:     Find  $t_i$  such that  $\Psi_C(t_i, i) = \max_c \Psi_C(c, i)$ 
8:     Find  $s_i$  such that  $\Phi_C(s_i, i) = \max_c \Phi_C(c, i)$ 
9:     if  $m(t_i) = s_i$  then
10:       $n_{correct} = n_{correct} + 1$ 
11:   Accuracy is  $a := n_{correct}/N$ 
12:   return  $a$ 

```

Analysis details. *Preprocessing scRNAseq data.* Since we use the same data in Korsunsky et al. (2018), see the preprocessing steps in it.

Visualization. We used the UMAP algorithm to visualize cells in a two-dimensional space. For all analyses, UMAP was run with the following parameters: $k = 30$ nearest neighbors, correlation based distance and $\text{min_dist} = 0.3$.

Comparison to other algorithms. Note that the three datasets is analyzed in the Korsunsky et al. (2018) with the Harmony package and do the comprehensive comparison with other algorithm, such as MNN Correct (Lun et al., 2016), Seurat multiCCA (Butler et al., 2018), Scanorama (Hie et al., 2019), BBKNN (Polanski et al., 2019), it shows that Harmony do a better work on these datasets. For simplicity, we only compare the results of Dincta with the Harmony outputs.

Dincta parameters. By default, we set the following parameters for Dincta: $\theta = 2$, $K = 100$, $\tau = 0$, $\sigma_e = \sigma_c = 0.1$, $\text{block_size} = 0.05$, $\epsilon_{\text{cluster}} = 10^{-5}$, $\epsilon_{\text{Dincta}} = 10^{-4}$, $\text{max_iter}_{\text{cluster}} = 200$, $\text{max_iter}_{\text{Dincta}} = 20$, $\text{max_iter}_{\text{refine}} = 3$, $\lambda = 1$, $\alpha = 0.5$, $\delta = 6$, $p_{\text{new}} = 0.8$, $\gamma_m = 0.6$, $\gamma_s = 0.9$, $\delta_c = 0.5$, $\delta_\lambda = 0.9$, $\delta_d = 1e-2$. For the pancreas analysis, we set donors to be the primary covariate ($\theta = 1$) and technology secondary ($\theta = 2$).

Harmony parameters. By default, we set the following parameters for Harmony: $\theta = 2$, $K = 100$, $\tau = 0$, $\sigma = 0.1$, $\text{block_size} = 0.05$, $\epsilon_{\text{cluster}} = 10^{-5}$, $\epsilon_{\text{Harmony}} = 10^{-4}$, $\text{max_iter}_{\text{cluster}} = 200$, $\text{max_iter}_{\text{Harmony}} = 20$, $\lambda = 1$. For the pancreas analysis, we set donors to be the primary covariate ($\theta = 1$) and technology secondary ($\theta = 2$).

5 Data availability

All data analyzed in this article are publicly available in the <https://github.com/immunogenomics/harmony2019>. And the original links of data can be found in Supplementary Table 8 in Korsunsky et al. (2018).

6 Code availability

Dincta are available as R package on <https://github.com/songtingstone/dincta>. Scripts to reproduce results of the primary analyses will be made available on <https://github.com/songtingstone/dincta2020>.

References

- Azizi, E., Carr, A., Plitas, G., Cornish, A. E., Konopacki, C., Prabhakaran, S., Nainys, J., Wu, K., Kiseliovas, V., Setty, M., et al. (2018). Single-cell map of diverse immune phenotypes in the breast tumor microenvironment. *Cell*, 174(5):1293–1308.
- Butler, A., Hoffman, P. J., Smibert, P., Papalexi, E., and Satija, R. (2018). Integrating single-cell transcriptomic data across different conditions, technologies, and species. *Nature Biotechnology*, 36(5):411–420.
- Dhillon, I. S. and Modha, D. S. (2001). Concept decompositions for large sparse text data using clustering. *Machine Learning*, 42(1):143–175.
- Hie, B., Bryson, B. D., and Berger, B. (2019). Efficient integration of heterogeneous single-cell transcriptomes using scanorama. *Nature Biotechnology*, 37(6):685–691.
- Jordan, M. I. and Jacobs, R. A. (1994). Hierarchical mixtures of experts and the em algorithm. *Neural Computation*, 6(2):181–214.
- Korsunsky, I., Fan, J., Slowikowski, K., Zhang, F., Wei, K., Baglaenko, Y., Brenner, M. B., Loh, P., and Raychaudhuri, S. (2018). Fast, sensitive, and accurate integration of single cell data with harmony. *bioRxiv*, page 461954.
- Lun, A. T. L., McCarthy, D. J., and Marioni, J. C. (2016). A step-by-step workflow for low-level analysis of single-cell rna-seq data with bioconductor. *F1000Research*, 5:2122–2122.
- Polanski, K., Young, M. D., Miao, Z., Meyer, K. B., Teichmann, S. A., and Park, J. (2019). Bbknn: Fast batch alignment of single cell transcriptomes. *Bioinformatics*, 36(3):964–965.

7 Supplementary

Theorem 4. If $R(\cdot, \mathcal{I})u_\alpha(\mathcal{I}) > 0$ elementwisely, then

$I - [R(\cdot, \mathcal{U})\text{diag}^{1/2}(u_\alpha(\mathcal{U}))]^T \text{diag}(Ru_\alpha)^{-1} [R(\cdot, \mathcal{U})\text{diag}^{1/2}(u_\alpha(\mathcal{U}))]$ is symmetric positive definite hence invertible and the least square problem

$$\min_{\Phi_C(\cdot, \mathcal{U})} \|\Phi_C(\cdot, \mathcal{U})\text{diag}^{1/2}(u_\alpha(\mathcal{U}))(I - [R(\cdot, \mathcal{U})\text{diag}^{1/2}(u_\alpha(\mathcal{U}))]^T \text{diag}(Ru_\alpha)^{-1} [R(\cdot, \mathcal{U})\text{diag}^{1/2}(u_\alpha(\mathcal{U}))]) - \Phi_C(\cdot, \mathcal{I})\text{diag}(u_\alpha(\mathcal{I}))R(\cdot, \mathcal{I})^T \text{diag}(Ru_\alpha)^{-1} R(\cdot, \mathcal{U})\text{diag}^{1/2}(u_\alpha(\mathcal{U}))\|_F$$

has an unique solution

$$\begin{aligned} \Phi_C(\cdot, \mathcal{U}) &= \Phi_C(\cdot, \mathcal{I})\text{diag}(u_\alpha(\mathcal{I}))R(\cdot, \mathcal{I})^T \text{diag}(Ru_\alpha)^{-1} R(\cdot, \mathcal{U})\text{diag}^{1/2}(u_\alpha(\mathcal{U})) \\ &\quad [I - [R(\cdot, \mathcal{U})\text{diag}^{1/2}(u_\alpha(\mathcal{U}))]^T \text{diag}(Ru_\alpha)^{-1} [R(\cdot, \mathcal{U})\text{diag}^{1/2}(u_\alpha(\mathcal{U}))]]^{-1} \\ &\quad \text{diag}^{-1/2}(u_\alpha(\mathcal{U})) \\ &= \Phi_C(\cdot, \mathcal{I})\text{diag}(u_\alpha(\mathcal{I}))R(\cdot, \mathcal{I})^T \text{diag}(Ru_\alpha)^{-1} R(\cdot, \mathcal{U})\text{diag}^{1/2}(u_\alpha(\mathcal{U}))[I - A^T A]^{-1} \\ &\quad \text{diag}^{-1/2}(u_\alpha(\mathcal{U})) \end{aligned}$$

and can be compute efficiently in the following representation

$$\Phi_C(\cdot, \mathcal{U}) = \Phi_C(\cdot, \mathcal{I})\text{diag}(u_\alpha(\mathcal{I}))R(\cdot, \mathcal{I})^T \text{diag}(Ru_\alpha)^{-1} R(\cdot, \mathcal{U})\text{diag}^{1/2}(u_\alpha(\mathcal{U})) [I + A^T (I - AA^T)^{-1} A] \text{diag}^{-1/2}(u_\alpha(\mathcal{U}))$$

where $A := \text{diag}(Ru_\alpha)^{-1/2} R(\cdot, \mathcal{U})\text{diag}^{1/2}(u_\alpha(\mathcal{U}))$. And the solution of problem (26) can be approximated by normalize each column of $\Phi_C(\cdot, \mathcal{U})$ in equation (28). i.e.

$$\tilde{\Phi}_C(\cdot, \mathcal{U}) = \Phi_C(\cdot, \mathcal{U})\text{diag}^{-1}(\mathbf{1}^T \Phi_C(\cdot, \mathcal{U}))$$

In general, the matrix AA^T is semi-positive and the its largest eigenvalue $\lambda_{\max}(AA^T)$ is bound by $\max_k \beta_k$, where $\beta := [R(\cdot, \mathcal{U})u_\alpha(\mathcal{U})]/[Ru_\alpha]$ and the operator $/$ is element-wise division.

Proof. We first show that $\lambda_{\max}(AA^T) \leq \max_k \beta_k$, where $\beta := [R(\cdot, \mathcal{U})u_\alpha(\mathcal{U})]/[Ru_\alpha]$. Let $x \in \mathbb{R}^K$ with the unit length, i.e. $\sum_{k=1}^K x_k^2 = 1$. Note that $\lambda_{\max}(AA^T) = \max_{\|x\|_2=1} x^T AA^T x$.

$$\begin{aligned} x^T AA^T x &= \sum_{i \in \mathcal{U}} (\sum_{k=1}^K A_{ki} x_k)^2 \\ &= \sum_{i \in \mathcal{U}} (\sum_{k=1}^K R_{ki} \sqrt{u_\alpha(i)} / \sqrt{[Ru_\alpha]_k} x_k)^2 \\ &= \sum_{i \in \mathcal{U}} (\sum_{k=1}^K \sqrt{R_{ki} u_\alpha(i)} / \sqrt{[Ru_\alpha]_k} x_k \sqrt{R_{ki}})^2 \\ &\leq \sum_{i \in \mathcal{U}} \sum_{k=1}^K R_{ki} u_\alpha(i) / [Ru_\alpha]_k x_k^2 \\ &= \sum_{k=1}^K [R(\cdot, \mathcal{U})\text{diag}(u_\alpha(\mathcal{U}))]_k / [Ru_\alpha]_k x_k^2 \\ &= \sum_{k=1}^K \beta_k x_k^2 \\ &\leq \max_k \beta_k \end{aligned}$$

, where the first inequality comes from the Cauchy inequality and $\sum_k R_{ki} = 1$.

Next, we show that if $R(\cdot, \mathcal{I})u_\alpha(\mathcal{I}) > 0$, then $I - [R(\cdot, \mathcal{U})\text{diag}^{1/2}(u_\alpha(\mathcal{U}))]^T \text{diag}(Ru_\alpha)^{-1} [R(\cdot, \mathcal{U})\text{diag}^{1/2}(u_\alpha(\mathcal{U}))] = I - A^T A$ is positive hence invertible. Note that if $R(\cdot, \mathcal{I})u_\alpha(\mathcal{I}) > 0$, then $\beta_k = \frac{[R(\cdot, \mathcal{U})u_\alpha(\mathcal{U})]_k}{[R(\cdot, \mathcal{U})u_\alpha(\mathcal{U})]_k + [R(\cdot, \mathcal{I})u_\alpha(\mathcal{I})]_k} < 1$. So we have $\lambda_{\max}(A^T A) = \lambda_{\max}(AA^T) \leq \max_k \beta_k < 1$. Then $\lambda_{\max}(I - A^T A) > 0$, i.e. $I - [R(\cdot, \mathcal{U})\text{diag}^{1/2}(u_\alpha(\mathcal{U}))]^T \text{diag}(Ru_\alpha)^{-1} [R(\cdot, \mathcal{U})\text{diag}^{1/2}(u_\alpha(\mathcal{U}))]$ is positive hence invertible and the least square problem has the unique solution.

Finally, the equality $(I - AA^T)^{-1} = I + A^T (I - AA^T)^{-1} A$ can be proved by check that $[I - AA^T][I + A^T (I - AA^T)^{-1} A] = I$, which is very easy to verify. And the proof is finished. $\square$