Interpretable phenotype decoding from multi-condition sequencing data with ALPINE

Wei-Hao Lee[1,†], Lechuan Li[2,†], Ruth Dannenfelser[2], & Vicky Yao[2,3,4*]

[1] Systems, Synthetic, and Physical Biology, Rice University
[2] Department of Computer Science, Rice University
[3] Ken Kennedy Institute, Rice University
[4] Rice Synthetic Biology Institute, Rice University
[†]These authors contributed equally to this work.
[*]Correspondence to: vy@rice.edu

ABSTRACT

As sequencing techniques advance in precision, affordability, and diversity, an abundance of heterogeneous sequencing data has become available, encompassing a wide range of phenotypic features and biological perturbations. Unfortunately, increased resolution comes with a cost of increased complexity of the biological search space, even at the individual study level, as perturbations are now often examined across many dimensions simultaneously, including different: donor phenotypes, anatomical regions and cell types, and time points. Furthermore, broad integration across studies promise unique opportunity to explore the molecular underpinnings of distinct healthy and disease states, larger than the original scope of the individual study. To fully realize the promise of both individual higher resolution studies and large cross-study integrations we need a robust methodology that can disentangle the influence of technical and non-relevant phenotypic factors, isolating relevant condition-specific signals from shared biological information while also providing interpretable insights into the genetic effects of these conditions. Current methods typically excel in only one of these areas. To address this gap, we developed ALPINE, a supervised non-negative matrix factorization (NMF) framework that effectively separates both technical and non-technical factors while simultaneously offering direct interpretability of condition-associated genes. Through simulations across 4 different scenarios, we demonstrate that ALPINE outperforms existing methods in both isolating the effect of different phenotypic conditions and prioritizing condition-associated genes. Furthermore, ALPINE has favorable performance in batch effect removal compared with state-of-the-art integration methods. When applied to real-world case studies, we showcase how ALPINE can be used to extract insights into the biological mechanisms that underlie differences between phenotypic conditions.

1 Introduction

Biological complexity is incredibly multidimensional—complex diseases impact different tissues and cell types in unique ways [1, 2, 3], and biological factors such as patient sex can also influence disease response [4, 5, 6]. The rise of single cell technologies has fueled excitement for atlas-scale initiatives that capture cell-level diversity across numerous variables, but has also highlighted the challenges of data harmonization and interpretability [7, 8].

The immediate recognition of a need for harmonization methods focused on removing unwanted technical variation (e.g., single cell platform, experimental laboratory) has driven substantial method development efforts [9, 10], which typically aim to project cells from different datasets or samples into an integrated space [11, 12]. However, during this integration process, the focus on aligning cell types can come at the cost of treating inter-individual variation as a batch effect that is removed [10]. Thus, while these approaches can be effective at aligning cell types to capture consensus signals across populations, they risk obscuring important, nested condition signals, such as sex- or tissue-specific differences with respect to disease state.

Recent realization that data harmonization can potentially come at the sacrifice of some interpretability has led to method development efforts to build disentangled representations of scRNA-seq data by explicitly modeling the batch effects and biological conditions during the integration process [13, 14, 15, 16]. This

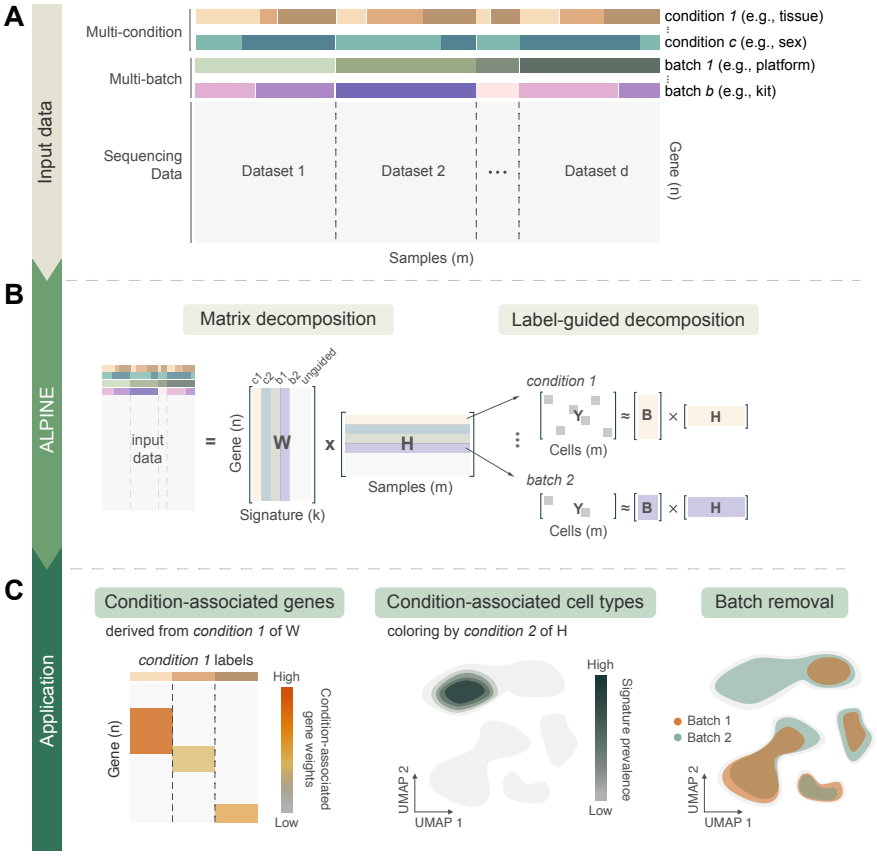

Figure 1. Overview of ALPINE for disentangling the effects of conditions and batches. (A) Example of a dataset with multiple condition and batch effects, creating challenges in cross-study analyses. (B) ALPINE's workflow incorporates an NMF-based approach with two main components: classic matrix decomposition and label-guided decomposition. The label-guided decomposition enables identification of condition- or batch-associated components. (C) ALPINE extends beyond standard single-cell analyses (e.g., clustering and cell embedding) by uncovering condition-associated genes and cells and addressing batch effects in a principled way, facilitating deeper insights across complex datasets.

modeling approach aims to enable the identification of condition-associated genes as well as more principled batch effect removal, though this task becomes more challenging when there are multiple groups of batches and conditions (as opposed to multiple levels within a single condition). For example, methods like scINSIGHT [13] and scParser [16] use matrix factorization-based approaches to enable interpretable representation decomposition, but inherently, they can only handle different variables within one condition. When there are multiple biological conditions, they need to be represented as a concatenation of all combinations of variables (e.g., tissue+sex), which makes it challenging to isolate and interpret individual conditions, let alone comparisons between conditions. Meanwhile, though methods such as scDisInFact [14] and scDisco [15] can explicitly model each biological condition separately, they use variational autoencoders to identify cell embeddings, which limits the ability to link genes with specific conditions, only providing a notion of which genes are generally associated with a broad category.

Here, we introduce ALPINE (Adaptive Layering of Phenotypic and Integrative Noise Extraction), a novel, flexible approach designed to address the complexities of multi-condition and multi-batch scenarios with improved interpretability (Figure 1). ALPINE builds on the typically unsupervised non-negative matrix factorization (NMF) framework to incorporate supervised, label-guided decomposition of biological conditions and/or technical batches. The use of supervision sets it apart from previous NMF-based integration methods such as LIGER [17], and the novel joint supervised-unsupervised representation enables natural support for modeling multiple conditions in a way that methods like scParser [16] cannot. This process enables users to directly extract meaningful condition-associated genes, remove batch-associated signatures, and use the unguided components to build a low-dimensional embedding of any remaining variation. In single cell datasets, the unguided components will typically capture the condition- and batch-agnostic cell type variation. The decomposed condition-associated signatures can be simultaneously analyzed for gene associations as well as cell type associations, including between conditions, which can enable efficient exploration of otherwise unwieldy datasets.

2 Results

2.1 Overview of the ALPINE framework

ALPINE extends traditional NMF by decomposing scRNA-seq expression data (Figure 1A) into interpretable components that reflect both known covariates and shared biological variation. Specifically, the method separates the data into "guided" components (directly linked to provided labels such as batch or phenotypic condition), as well as an "unguided" component that captures shared residual variation (Figure 1B, Methods). This dual strategy enables ALPINE to not only reconstruct the original data accurately, but also to generate condition-specific gene signatures that are readily interpretable and remove batch effects present in the data (Figure 1C). To tackle the computational challenges associated with large-scale single-cell datasets, ALPINE employs a mini-batch training strategy to improve computational efficiency and enhance model generalizability (Methods). This approach leverages subsets of cells to update the cell embedding matrix, reducing memory usage and mitigating overfitting. This strategy is particularly valuable when processing large-scale scRNA-seq datasets. Overall, ALPINE's design provides a powerful framework for disentangling complex biological and technical signals, delivering both high-quality integration and direct interpretability of condition-specific effects.

2.2 ALPINE can disentangle condition information from shared cell information

To evaluate ALPINE's ability to handle multi-condition and multi-batch datasets, we simulate one batch effect and two condition effects (stimulation and severity), each with four distinct perturbation scenarios (Figure 2A, Methods). In addition to benchmarking against the two disentanglement methods we could successfully run, scDisInFact and scParser, we also calculate two baseline comparisons (using the raw counts directly or principal components as input).

In the training data, we find that in the naive, overlap, and two-patterns scenarios, where we introduce fold changes to all of the cell types for a given set of genes, ALPINE embedding shows better performance than scDisInFact (Figure 2B, naive: p=4.13e-3; overlap: p=1.26e-2; two-patterns: p=2.5e-3, Wilcoxon rank-sum test) and comparable performance with scParser (naive: p=0.29; overlap: p=0.45; two-patterns: p=0.10, Wilcoxon rank-sum test). In the cell-specific scenario, where only a subset of cell types are subjected to the condition effect (while all cell types are affected by the batch effect), the ALPINE embedding significantly outperforms scParser and scDisInFact (scParser: p=4.94e-2; scDisInFact: p=8.15e-3, Wilcoxon rank-sum test). This simulation is a better reflection of real data, as it many diseases or treatments affect only a subset of cell types. For holdout set comparisons, only scDisInFact and ALPINE can generalize existing trained models to new datasets. ALPINE shows comparable performance compared to scDisInFact across all scenarios (naive: p=0.59; overlap: p=0.36; two-patterns: p=0.10; cell-specific: p=0.88, Wilcoxon rank-sum test).

In addition to using ALPINE's cell embeddings, we can also reconstitute cell counts using only the unguided portions of ALPINE's W and H matrices, which we demonstrate also performs well across all scenarios, often even better than using the embeddings directly. Thus, ALPINE's reconstructed counts can be used as a batch-removed, integrated dataset for downstream single cell analyses.

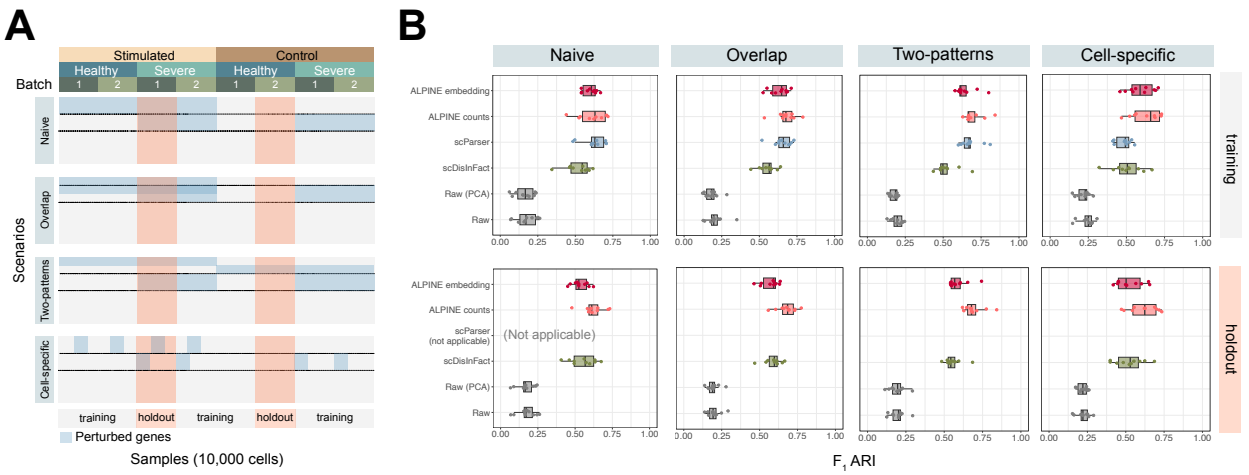

Figure 2. Systematic benchmarking for multi-batch and multi-condition disentanglement using different simulated scenarios. (A) Perturbation arrangement of count matrices in the simulated datasets. For each scenario, 8 count matrices are generated, corresponding to batch and 2 condition types (simulation and severity). Four scenarios include: adding signals to one label of each condition with independent perturbations (Naive); shared gene perturbations between conditions (Overlap); signals added to both labels of each condition (two-patterns); and cell-specific signals (Cell-specific). **(B)** Box plots compare the F1 ARI performance of ALPINE (embedding and counts) with two existing methods (scParser and scDisInFact) and two baselines (PCA and Raw) across training and holdout datasets. ALPINE (both embedding and counts) shows consistently strong performance, especially in more complex scenarios.

2.3 ALPINE efficiently identifies condition-associated genes and cells

Using the same four scenarios, we can further assess to what degree the actual condition-associated genes are prioritized by each method. Both ALPINE and scParser successfully identify the true perturbed genes in simpler tasks, such as in the naive and overlap scenarios (Figure 3A). However, as the complexity of the task approaching more realistic conditions, such as in the two-patterns and cell-specific scenarios, the AUPRC for scParser begins to decline. This decrease is likely due to scParser's strong implicit integration of all labels during the matrix factorization stage. In scenarios where not all cells are subject to the condition effect, this approach hampers scParser's effectiveness. While scDisInFact is capable of detecting some associated genes, it does not capture perturbed genes as effectively as scParser and ALPINE. We also note that ALPINE and scDisInFact have similar overall runtimes (Figure 3B), while scParser requires substantial time for its optimization procedure. Once trained, ALPINE has very efficient execution times. We do note that part of

the runtime limitations of scParser could be due to the fact that it does not have a GPU implementation, but even directly comparing ALPINE's CPU implementation, we see that scParser is still many orders of magnitude slower (Figure S1).

ALPINE's decomposition framework enables it to not only identify condition-associated genes, but simultaneously identify cell types associated with specific conditions. We first calculate $H_{\text{stimulation}}Y_{\text{cell}}^{\top}$ to ascertain the associations between signatures and labels, enabling us to determine which signatures are most relevant to each label. Subsequently, we can directly visualize the relevant component of $H_{\text{stimulation}}$ to reveal which cell types are predominantly affected by the condition (Figure 3C). The low-dimensional representation visualizes the cell embeddings, with colors indicating prevalence scores derived from the condition-associated signatures. This color-coding reflects the cell types subject to which conditions. Such functionality, to our knowledge, the first of its kind for methods that learn disentangled representations.

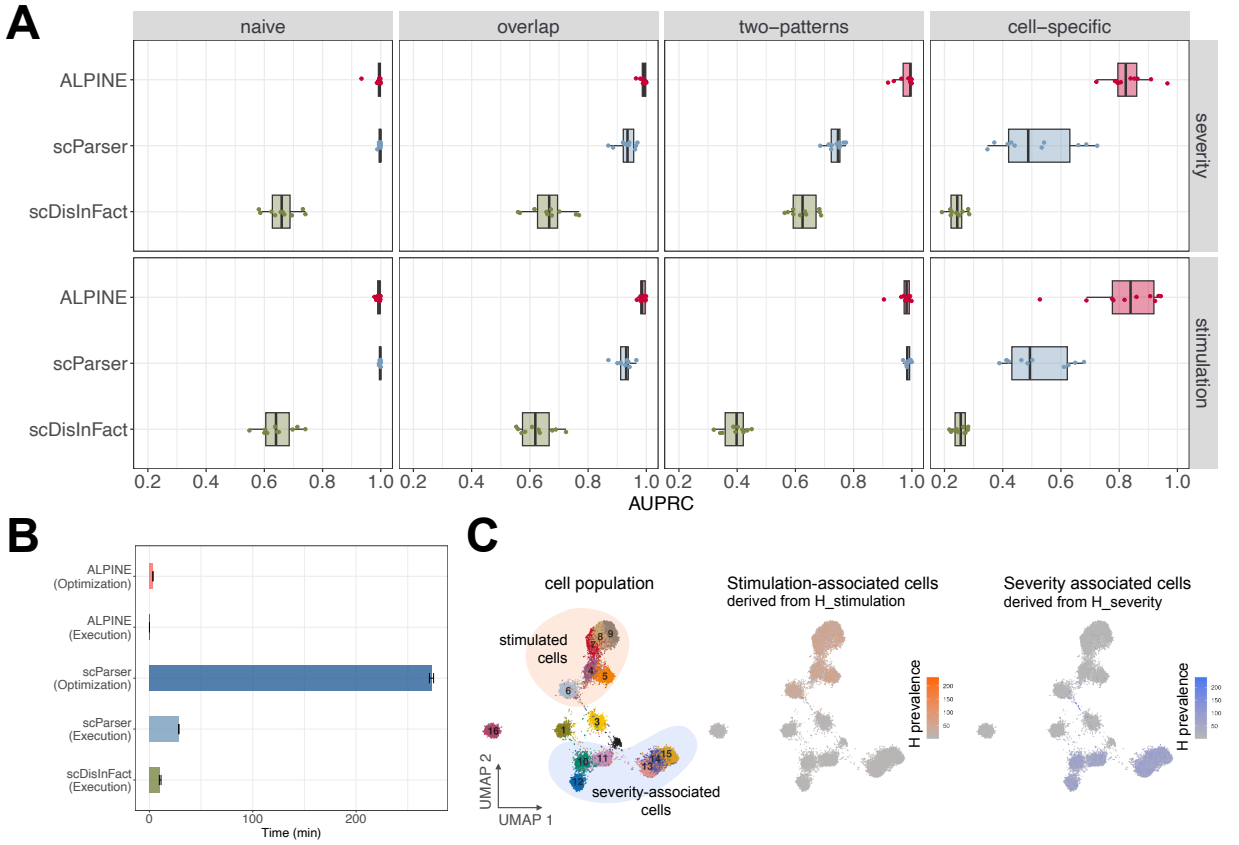

Figure 3. ALPINE can accurately extract condition-associated genes. (A) Boxplot of the AUPRC of ALPINE, scDisInfact, and scParser on condition-associated gene detection in four scenarios. ALPINE has the best performance in overlap, two-patterns, and cell-specific, as well as comparable performance to scParser in the naive case, demonstrating ALPINE's ability to capture conditon-specific signals and enable better interpretability. **(B)** Algorithm run time (execution) for ALPINE, scParser, scDisInfact and hyperparameter searching (optimization) for ALPINE and scParser. scDisInfact does not provide an optimization function. **(C)** UMAP plot of cell embeddings with batch and condition effects removed, colored by cell type. Highlighted are cells under two conditions with distinct gene perturbations. ALPINE's condition embeddings capture condition-associated signatures, accurately identifying stimulation-associated (types 4-9) and severity-associated cell types (types 10-15).

We also use these simulated scenarios to compare training ALPINE using mini-batch versus full-batch updates (Figure S2). On the training set, both approaches yield similar F1 ARI scores. However, on unseen holdout sets, mini-batch training typically provides results in better performance, confirming its advantage in generalizing to new data (Figure S2A). The only exception occurs in the cell-specific scenario, likely due to the fact that only a tiny proportion of cells are actually affected by the unique conditions, and there may

be variable proportions of perturbed cells in each mini-batch. In addition, relative to full-batch training, the mini-batch training strategy typically results in more rapid convergence (Figure S2B). Thus, we see that mini-batch training tends to enhance model robustness while reducing computational resource demands.

2.4 ALPINE achieves state-of-the-art performance in batch effect removal and cell type annotation in real datasets

After demonstrating ALPINE's power and interpretability in simulation datasets, we further verify ALPINE's effectiveness using real data. Following the setup in [11], we use three datasets: (1) aligned cell types from two batches. The data is from 15,476 human peripheral blood mononuclear cells, and they are from two different sequencing platforms (10x 3' and 10x 5') [18]; (2) aligned cell types with multiple batches. The data consists of five studies of human pancreatic cells from four different technologies with a total of 14,767 cells [19, 20, 21, 22, 23]; (3) non-identical cell types, with mouse retina data from two studies, including 62,851 cells [24, 25]. Besides scParser and scDisInFact, we include seven methods (Seurat, Harmony, Liger, and scVI, Scanoramam MNN, ComBat) that are designed to perform batch removal and cell type clustering, even though they do not extract interpretable condition features. As a further baseline, we also include using the PCA using the top 20 PCs to assess how severe the original batch effects are in the input dataset.

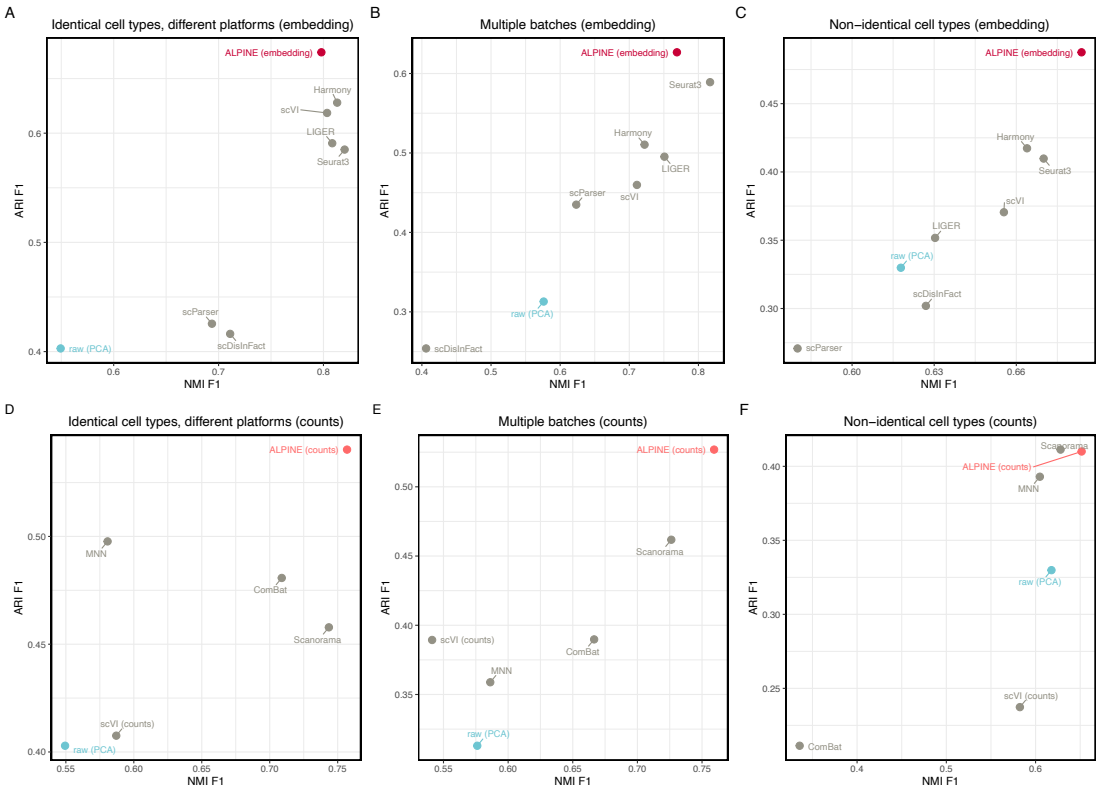

Figure 4. Comparative study of ALPINE against other methods for batch effect removal and cell type annotation. Adjusted Rand Index (ARI) and Normalized Mutual Information (NMI), of ALPINE versus scDisInFact [14], scParser [16], Seurat3 [26], Harmony [27], LIGER [28], scVI [29], Scanorama [30], MNN [31], and ComBat [32], in batch effect removal using three real datasets. Methods producing low-dimensional embeddings (scDisInFact, scParser, Seurat3, Harmony, LIGER, scVI) are compared with ALPINE embeddings in **(A)-(C)**, while methods reconstructing counts (Scanorama MNN, ComBat, and scVI (counts)) are compared with ALPINE-reconstructed counts in **(D)-(F)**. **(A)&(D)**. Human peripheral blood monouclear cell datasets with two batches and matched cell types. **(B)&(E)**. Pancreatic cells dataset with five batches and matched cell types. **(C)&(F)**. Mouse retina data with two batches and non-identical cell types. Both ALPINE embedding and counts show the best or comparable ARI F1 score and NMI F1 score in all three datasets.

In all three scenarios, the ALPINE embedding shows better or comparable performance in batch removal and cell type identifying compared to state-of-the-art batch removal methods (Figure 4, Figure S3). The other two condition-disentanglement methods, scDisInFact and scParser, generally perform poorly. We suspect that the poor performance of scParser stems from its assumption that all cells in the same batch undergo identical transformations and its poor scalability, which required us to limit its optimization time to 12 hours. For scDisInFact, it is possible that the lack of clear stopping criteria and hyperparameter optimization, as well as the requirement that a condition label be given even when there is no additional condition information, may be related to its suboptimal performance. The ALPINE embedding consistently has the best ARI scores and comparable NMI scores (scenario 1: ALPINE embedding 0.798 vs best: 0.820; scenario 2: ALPINE embedding 0.769 vs best: 0.816; scenario 3: ALPINE embedding is the best) to the next best method. Using ALPINE reconstructed counts, we also see the highest-performing ARI and NMI in all three scenarios, except ARI for scenario 3 (ALPINE counts 0.410 vs best: 0.411). Comparing cell type ASW and batch ASW scores, both ALPINE embedding and reconstructed counts also demonstrate strong performance compared to existing batch correction methods (Figure S3). In summary, ALPINE can effectively model the expression pattern in real datasets, and its label-guided decomposition can effectively isolate batch effect signals while preserving complex cell type information.

2.5 Cross-species integration and condition analysis with ALPINE

We further demonstrate ALPINE's versatility through an additional case study by applying it to a white adipose tissue dataset containing diverse batches (assays: Drop-seq, 10x 3′ v3) and conditions (organisms: Homo sapiens, Mus musculus; sex: male, female; depot tissues: subcutaneous adipose tissue, periovarian fat pad, omental fat pad, inguinal fat pad, and epididymal fat pad). The original study was only able to analyze each species separately, but here, we demonstrate that ALPINE is capable of identifying unifying cell embeddings across species, and the interpretability of its components further enables a more comprehensive comparison of cellular characteristics across species. Before further downstream analysis, we find that applying ALPINE directly with associated conditions effectively removes batch effects (Figure 5A), yielding well-mixed cell populations across species, with the notable exception of adipocytes, which remain distinct.

Interestingly, we find that ALPINE's cell embeddings accurately reflect a low abundance of adipocytes in Drop-seq versus a higher representation in sNuc-Seq, recapitulating the established platform-specific difficulty of capturing adipocytes (Figure 5B). We also find distinct cell composition variations across fat depots, notably a higher macrophage proportion in the male mouse epididymal fat pad, which prompted further exploration of sex-related differences. Within the sex-specific embeddings, ALPINE correctly captured increased macrophages in male mice, as well as mammary gland epithelial cells unique to female mice (Figure 5C). In human samples, we see a more even distribution of immune cell types compared to mice, highlighting ALPINE's ability to capture detailed, biologically relevant signals across species.

An important unanswered question is what biological factors drive the distinct differences between mouse and human adipocytes. In the original study, this comparison was infeasible because the mouse and human cells were not embedded in a shared space. We calculated the difference between human and mouse adipocyte gene signatures and applied GSEA to reveal functional distinctions (Figure 5D), identifying several adipose-related signals, including melanogenesis [33], which has been reported in human adipose tissue, where various melanogenesis-related genes contribute to the presence of melanin. Additionally, we find the Hedgehog signaling pathway [34], known for its dysregulation during adipocyte differentiation. Other enriched terms, such as the sphingolipid signaling pathway [35], further demonstrate the biological changes associated with adipose tissue. These findings underscore the utility of our methods in uncovering the underlying biological functions linked to specific conditions.

We also compare ALPINE's findings with those of scDisInFact on the same dataset (Figure S4). Unfortunately, due perhaps to the larger size of this dataset, we were unable to obtain results from scParser. Initially, we analyze the integration measures for cells, covariates, and batch, observing comparable outcomes except

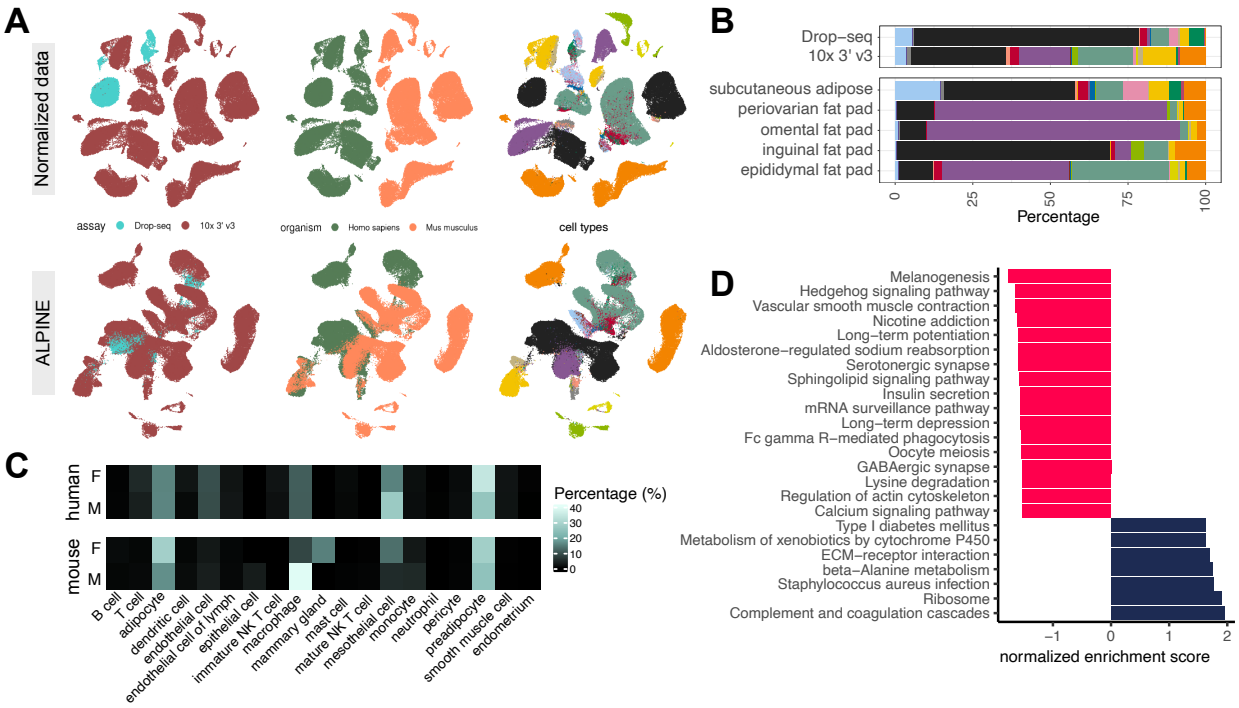

Figure 5. Cross-species analysis in adipose tissue with multi-condition analysis (A) The original dataset displays multiple batch and condition effects, with colors indicating various categories. Post-ALPINE application, batch effects from sequencing technologies are removed, resulting in aligned clusters of similar cell types across species, although adipocytes remain distinct due to biological differences. **(B)** Condition embeddings reveal associations between cell types and assay/tissue labels, accurately capturing expected cell types. Notably, Drop-seq fails to identify fragile adipocytes, which are detectable only via sNuc-Seq. Tissue embeddings further represent accurate cell-type abundances across conditions. The stacked bar plot is color-coded by cell types, consistent with (A). **(C)** Analysis of sex embeddings, separated by species, indicates that human cell compositions are more similar between males and females, while mouse samples exhibit significant differences, underscoring the identified sex-related variations. **(D)** The bar plots display normalized enrichment scores for KEGG terms from the GSEA analysis, comparing human and mouse adipocytes using gene data from W_{organism}. Positive values indicate higher enrichment in humans, while red bars denote higher scores in mice.

for ARI_{cell} (Supplemental Tables 1 and 2). scDisInFact's conditional variational autoencoder-based model limits its ability to derive covariate-associated weights for each sample, so we are unable to construct a covariate-associated cell embedding space to enable the composition analysis described above.

We find that a further limitation of scDisInFact for interpretability is an inability to associate gene scores within individual labels. For the cross-species analysis, this means that we can only obtain a single organism-associated gene list, as opposed to distinct human- and mouse-associated genes. Comparing the individual genes identified (Methods), ALPINE identifies 383 and 226 significantly associated genes for mouse and human, respectively, while scDisInFact finds 112 general organism-associated genes. There are only 5 genes found in common between scDisInFact's genes with ALPINE (all with the human-associated signatures). Interestingly, among these are four mitochondrial genes (MT-CO2, MT-CO3, MT-ATP6, MT-ND3) that are the top-ranked genes by both methods. Although no cross-species comparison has been conducted on these genes, one study indicates that these mitochondrial genes play a crucial role in cold-induced mitochondrial biogenesis in white adipose tissue [36]. Another study also identified MT-CO2, MT-ND3, and MT-ATP6 as being correlated with BMI in their meta-analysis, potentially linking them to participants in the study with high BMI [37]. While this shared identification is exciting, unfortunately, the organism-associated gene scores from scDisInFact do not yield significant GO enrichment using GSEA (FDR ≤ 0.1) for further functional comparisons with ALPINE. In general, these results illustrate that ALPINE not only achieves effective cross-

species integration, but also delivers enhanced interpretability by explicitly delineating condition-specific signals.

3 Discussion

In this study, we demonstrate that ALPINE is capable of simultaneously addressing multiple challenges in single cell data analysis, including disentangling complex batch and condition effects and finding condition-associated effects on genes, cells, and inter-condition interactions. Compared to existing disentanglement methods, ALPINE provides significant advantages in integration performance, interpretability, scalability. In particular, ALPINE not only captures general conditional signals, but also cell-specific effects, which are essential for obtaining nuanced biological insights. ALPINE is also the only disentanglement method that exhibits comparable batch effect removal performance on real datasets to state-of-the-art integration methods.

While ALPINE demonstrates strengths in disentangling batch and condition effects and capturing both general and cell-specific signals, it may face limitations in scenarios with complex, non-additive, or non-linear interactions among conditions, as these would not be fully captured by the underlying NMF framework. Additionally, handling highly variable cell populations or very subtle condition effects may reduce interpretability. Future work could focus on extending ALPINE's framework to capture non-linear interactions. Furthermore, it is also possible for biological variables of interest to exhibit more complex, interesting structure, such as continuous values or nested hierarchical structure. ALPINE can be naturally extended by considering different loss terms for different types of guided variables to handle that additional complexity.

In summary, ALPINE's flexible and interpretable framework enables it to meet a wide range of analytical needs, providing robust performance across datasets with varying batch and condition complexities. As a scalable and versatile tool, ALPINE has significant potential for advancing condition-specific studies in diverse single-cell applications, offering researchers a powerful method to uncover subtle biological patterns and interrelationships within complex, heterogeneous data. Our implementation of ALPINE package and code for the analysis described herein is available on Github at `https://github.com/ylaboratory/ALPINE`.

4 Methods

4.1 Overview

ALPINE aims to decompose a heterogeneous single cell transcriptomics dataset into interpretable low dimensional components. Specifically, a given single cell sample is often associated with different technical batch effects and phenotypic conditions, which we term 'guided variables.' ALPINE jointly optimizes for the decomposition of an input dataset into guided and unguided components and includes a built-in hyperparameter tuning framework to efficiently select hyperparameters.

4.2 Label-guided matrix decomposition

Given a log-transformed single-cell RNA expression matrix $X \in \mathbb{R}^{m \times n}$, where m is the number of genes and n is the number of cells, ALPINE decomposes X into a non-negative gene feature matrix $W \in \mathbb{R}^{m \times k}$ and a cell embedding matrix $H \in \mathbb{R}^{k \times n}$, where k represents the total number of latent components.

For a given guided variable, we define the corresponding submatrices as $W^{(i)}_{\text{guided}} \in \mathbb{R}^{m \times k^{(i)}_{\text{guided}}}$ and $H^{(i)}_{\text{guided}} \in \mathbb{R}^{k^{(i)}_{\text{guided}} \times n}$, ensuring that the total number of components satisfies: $\sum_{i=1}^{c} k^{(i)}_{\text{guided}} + k_{\text{unguided}} = k$, where c denotes the number of user-specified guided variables.

Each component is uniquely associated with either a guided variable or an unguided shared variation. In scenarios where certain guided variables are entirely confounded with each other (e.g., each condition

is exclusively sequenced in a distinct batch), the corresponding guided variables become mathematically collinear (analogous to including perfectly correlated predictors in a regression model). Under such conditions, the model cannot uniquely partition the variation between the different guided variables, leading to an identifiability problem. To mitigate this, we recommend combining the confounded guided variables into a single variable that captures their joint effect. This approach allows ALPINE to focus on extracting shared biological signals, as the experimental design would limit the ability to disentangle the individual contributions of the confounded guided variables.

Overall, the matrices W and H are formed by concatenating their respective submatrices:

$$W = \begin{bmatrix} | & | & & | & | \\ W^{(1)}_{\text{guided}} & W^{(2)}_{\text{guided}} & \cdots & W^{(c)}_{\text{guided}} & W_{\text{unguided}} \\ | & | & & | & | \end{bmatrix} \quad \text{and} \quad H = \begin{bmatrix} — & H^{(1)}_{\text{guided}} & — \\ — & H^{(2)}_{\text{guided}} & — \\ & \vdots & \\ — & H^{(c)}_{\text{guided}} & — \\ — & H_{\text{unguided}} & — \end{bmatrix}.$$

To effectively disentangle the signals of different guided variables and shared cell information, ALPINE uses a binary guided indicator matrix, $Y^{(i)} \in \mathbb{R}^{d_i \times n}$ for each guided variable i, where d_i is the number of classes for that variable. The primary goal is to ensure that each cell embedding $H^{(i)}_{\text{guided}}$ encapsulates all relevant information for $Y^{(i)}_{\text{guided}}$. To achieve this, $H^{(i)}_{\text{guided}}$ is used to reconstruct $Y^{(i)}$ through multiplication with a learned transformation matrix $B^{(i)} \in \mathbb{R}^{d_i \times k^{(i)}_{\text{guided}}}$. The training objective function is thus a combination between the unsupervised reconstruction loss and supervised prediction loss:

$$L = \underset{X,Y,W,H,B \geq 0}{\arg\min} \left[\underbrace{||X - WH||^2_F}_{\text{reconstruction loss}} + \underbrace{\sum_{i=1}^{c} \lambda_i \text{KL}(Y^{(i)} | B^{(i)} H^{(i)}_{\text{guided}})}_{\text{prediction loss}} + J(W) \right] \tag{1}$$

Here, $|| \cdot ||_F$ represents the Frobenius norm that quantifies the reconstruction error between the matrix X and its approximation WH, and λ_i is a hyperparameter that balances reconstruction and prediction loss. KL represents the Kullback-Leibler divergence, which is particularly suitable for fitting the model to the binary condition matrix $Y^{(i))}$, as it quantifies the discrepancy between the predicted probabilities and the actual binary outcomes. We also include a regularization term, $J(W)$, to encourage the model to learn more unique and generalizable signatures by incorporating elastic net regularization and orthogonality in the W matrix [38].

$$J(W) = \underbrace{\alpha(l1_{\text{ratio}}||W||_1 + (1 - l1_{\text{ratio}})||W||_2)}_{\text{Elastic net regularization}} + \underbrace{\beta \sum_{i<j} X^{\top}_{\cdot i} X_{\cdot j}}_{\text{orthogonal regularization}} \tag{2}$$

where hyperparameter α represents the weight for the Elastic net regularization and $l1_{\text{ratio}}$ is the weight balancing LASSO and Ridge regularization. The final term, weighted by hyperparameter β, computes the sum of the product of two signatures, where lower values indicate lower similarity, promoting orthogonality.

The reconstruction loss optimizes for preservation of key biological information while the supervised prediction loss helps disentangle signals from different guided variables into the designated components. For a given guided variable, examining the sub-matrix $W^{(i)}_{\text{guided}}$ shows the contribution of individual genes

to a low-dimension representation of the variable, while $H^{(i)}_{\text{guided}}$ highlights the relative prevalence of the corresponding gene signatures in each cell. The remaining components W_{unguided} and H_{unguided} capture clean biological information that is consistent across different batches or conditions, which can be used for clustering cells or assigning cell types.

We derive the multiplicative updates for W, B, and H based on Equation (1):

$$W \leftarrow W \odot \frac{XH^\top}{W^\top(HH^\top + \alpha * (1 - l1_{\text{ratio}}) * I + \beta(\mathbf{1}_{k\times k} - I_{k\times k})) + \alpha * l1_{\text{ratio}} * \mathbf{1}_{m\times k}} \tag{3}$$

$$B^{(i)} \leftarrow B^{(i)} \odot \frac{(\frac{Y^{(i)}}{B^{(i)}H^{(i)}_{\text{guided}}})H^{(i)}_{\text{guided}}{}^\top}{\mathbf{1}_{Y^{(i)}}H^{(i)}_{\text{guided}}{}^\top} \tag{4}$$

$$P = \begin{bmatrix} P_1 & P_2 & \cdots & P_g & \mathbf{0}_{H_{\text{unguided}}} \end{bmatrix} \tag{5}$$

$$Q = \begin{bmatrix} Q_1 & Q_2 & \cdots & Q_g & \mathbf{0}_{H_{\text{unguided}}} \end{bmatrix} \tag{6}$$

$$H \leftarrow H \odot \frac{2W^\top X + P}{2W^\top WH + Q} \tag{7}$$

In the multiplicative update of W, $\mathbf{1}$ and I denote the ones matrix and the diagonal matrix, respectively. The detailed derivation of the regularization term can be found in Lin et al. [38]. Here, $\mathbf{1}_{Y_i}$ represents a matrix of ones with the same shape as $Y^{(i)}$ and $\mathbf{0}_{H_{\text{unguided}}}$ represents a matrix of zeros with the same shape as H_{unguided}. The supervised learning updates P_i and Q_i are formulated as $P_i = \lambda_i B^{(i)}{}^\top(\frac{Y^{(i)}}{B^{(i)}H^{(i)}_{\text{guided}}})$ and $Q_i = \lambda_i B^{(i)}{}^\top \mathbf{1}_{Y^{(i)}}$, where P and Q represent combinations of guiding adjustments derived from the prediction loss associated with the guided variables. Because there are no labels for the unguided components, we use zeros to fill the P and Q submatrices. During the updating process, P and Q help regularize H with guided label information, pushing the components to be specific to each guided variable. The final output of ALPINE are the final lower-dimensional representations W, H, and B that compartmentalize shared biological information and different phenotypic conditions.

4.3 Generalization to unseen datasets

Given an unseen dataset, ALPINE can use pre-trained W and B matrices from a previous integration to generalize to new single-cell data. For the new dataset, we use the new counts (X_{new}) and iteratively update the new dataset cell embedding H_{new} based on Equation 7 (with P and Q being matrices of 0s). Note that while a corresponding indicator matrix (Y_{new}) can optionally be used to update $B^{(i)}$, P, and Q using Equations 4-6, we do not recommend doing so to avoid potential overfitting.

4.4 Mini-batch training

A common limitation of existing matrix factorization approaches for single-cell analysis is the extensive computational resources required. As single-cell datasets grow larger, processing the entire expression matrix becomes increasingly time-consuming and memory-intensive. To address this challenge, ALPINE uses a mini-batch training approach that updates only a subset of the samples at a time, thereby conserving training resources. We choose to deploy the mini-batch strategy only on the sample space and not the feature space, since by the nature of scRNA-seq data, the number of features (i.e., genes ~20,000) is fixed and capturing gene-gene relationships is critical, whereas the number of samples (i.e., cells) can be extremely large (ranging from tens of thousands to millions). For reference, using full-batch training on a dataset with 1 million cells would use more than 40 GB of memory, making it infeasible for many GPUs, while mini-batch training can be easily adapted for available hardware.

For each sample, ALPINE calculates a weight based on the overall class representation (concatenating all guided variables), specifically weight$(c) = \frac{m}{z \times m_c}$, where m is the total number of samples in the dataset, z is the total number of class groups, and m_c is the number of samples annotated to class c. Within each mini-batch, samples are chosen based on these weights. W, H, and B are updated following Equations (3-7), but for H, in each step, only the subset of cells that correspond to the selected batch is updated. By updating the cell embedding matrix H using only a subset of cells in each mini-batch, we reduce the computational burden and memory footprint, also mitigating overfitting and reducing the potential bias introduced by label imbalance in the guided variables. This sampling strategy is analogous to stochastic gradient descent, where updating parameters using subsets of data helps avoid local minima and improves convergence speed.

4.5 Hyperparameter selection

The main hyperparameters of ALPINE include the number of components for both guided ($k^{(1)}_{\text{guided}}, \ldots, k^{(c)}_{\text{guided}}$) and unguided ($k_{\text{unguided}}$) variables, along with λ_i, α, β, and $l1_{\text{ratio}}$ from Equations (1-2). To enhance user experience and reduce manual tuning, users only need to specify a range for the total number of components, k, and the optimization process automatically allocates components between the guided and unguided parts.

Internally, given c representing the number of user-specified guided variables, we consider the total number of components, k, as being distributed among $c+1$ "parts," i.e., the c guided variables and 1 unguided part. To tune for the specific number of components in each part, every component, including the unguided one, is assigned a proportion: $\gamma^{(i)}_{\text{guided}}$ for guided and γ_{unguided} for unguided, with the constraint that $\sum_{i=1}^{c} \gamma^{(i)}_{\text{guided}} + \gamma_{\text{unguided}} = 1$. Furthermore, to ensure a sufficient number of unguided components even as the number of user-provided guided variables increase, we enforce $k_{\text{unguided}} \geq \sum_{i=1}^{c} k^{(i)}_{\text{guided}}$, or equivalently, $\gamma_{\text{unguided}} \geq \frac{1}{2}$. The optimization algorithm dynamically tunes the component allocation following these constraints, making the process more efficient and user-friendly.

The prediction loss typically converges faster than the reconstruction loss, so the number of epochs is automatically selected based on the elbow of the reconstruction loss curve to avoid overfitting. The rest of the hyperparameters are tuned with Bayesian optimization using Tree of Parzen Estimators [39] from the search space with 50 calls. The unguided component should capture shared biological signals across conditions, independent of the guided variables. To optimize the model's hyperparameters, we assess whether H_{unguided} has successfully excluded residual guided information. Specifically, clusters (C_{unguided}) are derived from the H_{unguided} matrix using the Leiden algorithm. We then compute both the adjusted rand index (ARI) and homogenenity score (HS) between these clusters and the guided variable labels, averaging the scores across all guided variables. Since the goal is to minimize the influence of guided labels on H_{unguided}, we seek to minimize both ARI and HS. Minimizing ARI ensures that the clustering of H_{unguided} does not reflect the guided labels, while minimizing the HS ensures that clusters are well-mixed with respect to these labels. Together, the loss function being optimized is:

$$L_{tune} = \frac{1}{g}\sum_{i}^{g} \text{ARI}(C_{\text{unguided}}, Y^{(i)}_{\text{guided}}) + \frac{1}{g}\sum_{i}^{g} \text{HS}(C_{\text{unguided}}, Y^{(i)}_{\text{guided}}). \tag{8}$$

The Bayesian optimization selects the set of hyperparameters that minimizes L_{tune}. We use [50, 100] as our search space for k, the total number of components. λ_i can depend on the complexity of the corresponding guiding variable i, so we use a large range [10^1, 10^6]. The ranges for α, β, and $l1_{\text{ratio}}$ are [0, 0.5], [0, 0.5], and [0, 1], respectively.

To gain a better understanding of ALPINE's sensitivity to different hyperparameters, we explore performance variability across each of the major hyperparameters using three real datasets. Using the optimized hyperparameter set from our optimizer, we vary one hyperparameter at a time while keeping the others fixed.

We find that performance the optimizer consistently finds strong sets of hyperparameters, and the hyperparameters that have the largest effect on performance are the total number of components, the guided (batch) component ratio, as well as the contribution of prediction loss (λ) (Figure S5). The optimal hyperparameter values does vary across different datasets, highlighting the benefit of our adaptive optimizer.

To further enhance the generalizability of ALPINE, we added a cross-validation option to the optimization process. The training data is randomly split into k folds. For each fold, we calculate L_{tune} as described previously, selecting the hyperparameter set with the smallest average L_{tune}. This cross-validation step helps ALPINE generalize better to unseen data. We used 4-fold cross-validation for the simulation dataset. For the benchmark and adipose tissue datasets, we did not apply cross-validation, as those tasks do not require applying trained ALPINE models to unseen data.

4.6 Data simulation for conditional gene detection and batch removal

To demonstrate that ALPINE effectively separates diverse covariate influences, including both phenotypic conditions and batch effects, we use the Symsim tool [40] to simulate 10,000 cells with 500 genes generated from a phylogenetic tree with 16 distinct cell types as ground truth, including a simulated "rare" cell type with 200 cells and all other cell types having equal sizes. To introduce multiple conditions, we define two key categories: "stimulation" (comprising control and stimulation) and "severity" (encompassing healthy and severe states) and all four resultant condition combinations (Figure 2A). We apply these conditions across all cell types by replicating the original 10,000 cells four times, assigning a distinct condition to each group. In total, we simulate 40,000 cells and introduced two batch effects to simulate batch variation.

We simulate 4 scenarios of gene expression perturbation: naive, overlap, two-patterns, and cell-specific. In the naive scenario, perturbations to gene expression are applied uniformly. The overlap scenario models cases where both conditions influence a subset of the same genes. In the two-patterns scenario, the stimulated and control conditions perturb different genes, while the severe condition perturbs the same genes. Finally, the cell-specific scenario is the most complex, as it involves selective perturbing of cell types, thus simulating condition effects within only specific cell populations.

4.7 Identifying genes associated with specific conditions and affected cell types

Recomposing the various decomposed matrices allows for interpretable analysis of resulting associations. For example, gene-disease associations, can be calculated $W_{\text{disease}}H_{\text{disease}}Y_{\text{disease}}^{\top}$, resulting in a matrix of genes by disease labels. Additionally, the $W_{\text{guided}}^{(i)}H_{\text{guided}}^{(i)}$ matrix offers insights into the relationships between genes and corresponding cells, enabling users to explore the effects of various conditions on cells. Alternatively, users may investigate the $H_{\text{guided}}^{(i)}Y_{\text{unguided}}^{\top}$ matrix to uncover associations between guided variables and labels, providing a comprehensive understanding of condition-associated signatures and their impact across cell populations.

4.8 Benchmarking existing condition disentangling methods

We attempted to compare ALPINE to the 4 existing condition disentangling methods: scDisInFact, scDisco, scParser, and scINSIGHT. During model training, scDisco frequently encountered NaN issues so we were unable to obtain comparable results, and scINSIGHT required extensive training time (>24 hours) even on small datasets, so we also excluded it from comparisons. scParser requires a single guided variable input, so we concatenated multiple condition labels for compatibility. For a fair evaluation, we adhered to the recommended hyperparameter settings for scDisInFact and utilized the optimization function in scParser to identify optimal parameters. For larger, real-world datasets, we limited the optimization time for scParser to 12 hours for computational tractability.

4.9 Performance evaluation metrics for model assessment and identifying condition-associated genes

To provide objective comparisons of the performance of different tools, we use the adjusted rand index (ARI), normalized mutual information (NMI), and Average Silhouette Width (ASW) to evaluate how well cells are blended across different conditions. Ideally, for cell clustering, ARI and NMI scores close to 1 indicate that each cluster contains only one type of cell, reflecting pure clusters. However, for assessing guided variables, the goal is to have cells from different conditions and batches be well-blended, so to calculate a single performance metric, we report $1 - \text{ARI}_{\text{guided}}$, where a value closer to 1 represents better mixing of conditions or batches. We use the transformation from scIB [10] to process the ASW score, mapping both batch and cell type ASW to a 0-1 range, with higher values indicating better performance. To prevent majority cell types from dominating the results, the final batch ASW score is weighted by number of cells in a cell type.

For easier representation when there are situations with multiple batches and conditions, we adopted the F1 score equation from [9] to evaluate the performance for removing batch effects while retaining cell type information:

$$\text{F1}_{ARI} = \frac{g \times \text{ARI}_{\text{unguided}} \times \Pi_{i=1}^{g} \text{ARI}_{\text{guided}}^{(i)}}{\text{ARI}_{\text{unguided}} + \sum_{i=1}^{g} \text{ARI}_{\text{guided}}^{(i)}} \tag{9}$$

where g is the number of guided variables, which includes both batch and condition variables. The F1 score for the normalized mutual information (NMI) follows the same formula as the ARI, with ARI values simply replaced by their corresponding NMI values.

For each benchmarking dataset, the Leiden clustering resolution was automatically selected based on a 50-50 train-test split on the dataset, conducting a grid search for the optimal resolution, ranging from 0.2 to 5. All reported scores are derived from the test set results in our simulations.

To evaluate each tool's ability to identify true condition-associated genes, we calculated the Area Under the Precision-Recall Curve (AUPRC), comparing each tool's top-weighted genes against ground truth perturbations. For scDisInFact, we directly used the weights provided by the tool. Since scParser requires conditions to be combined into a single integrated label (e.g., control + severe), we assessed gene weights from each associated signature individually against the ground truth condition-specific genes, selecting the highest AUPRC achieved for each condition.

4.10 Benchmarking existing batch removal tools

We compare ALPINE to 7 existing single cell batch removal methods, Seurat [26], Harmony [27], LIGER [28], scVI [29], Scanorama [30], MNN [31], ComBat [32] and 2 condition disentangling methods, scParser [16] and scDisInfact [14], for batch effect removal tasks. Using Scanpy [41], we filter out genes present in fewer than 5 cells and cells expressing fewer than 300 genes in each dataset. For models assuming a zero-inflated negative binomial distribution, such as scDisInFact and scVI, we provide raw counts as input. For Seurat, Harmony, LIGER, and scParser, we provide log-transformed data as required. For scDisInfact, which requires an additional condition beyond batch for execution, we create a dummy condition by assigning the same label to all cells.

4.11 Data preprocessing for the adipose tissue dataset

The adipose tissue dataset [42] includes both human and mouse data, sequenced on two platforms, single-cell RNA-seq and single-nucleus RNA-seq, with 166,149 human cells and 197,721 mouse cells initially collected. For analysis, we subsampled 50,000 cells from each species, totaling 100,000 cells (27.48% of the dataset). To combine the two species' single-cell data, we used orthologous gene mapping, which yielded 15,417

shared genes. The combined data was then normalized and log-transformed to unify library sizes across species. We use the provided batch (i.e., organism) and biological condition variables (i.e., assay, sex, depot tissues) as guided variables to ALPINE for signal decomposition. In comparing our results with those of scDisInFact, we used the same guided labels and designated the assay as the primary batch, while treating the remaining variables as covariates of interest for training the scDisInFact model. To extract important genes associated with labels from scDisInFact, we first transformed the gene scores into z-scores, then calculated the corresponding p-values, followed by False Discovery Rate (FDR) correction. We applied the same procedure to the gene signatures identified by ALPINE and used an FDR threshold of ≤ 0.05 to select significant genes.

References

[1] Johannes C Melms et al. "A molecular single-cell lung atlas of lethal COVID-19". In: *Nature* 595.7865 (2021), pp. 114–119.

[2] Jingyao Zeng et al. "CancerSCEM: a database of single-cell expression map across various human cancers". In: *Nucleic acids research* 50.D1 (2022), pp. D1147–D1155.

[3] Tushar Kamath et al. "Single-cell genomic profiling of human dopamine neurons identifies a population that selectively degenerates in Parkinson's disease". In: *Nature neuroscience* 25.5 (2022), pp. 588–595.

[4] Carole Ober, Dagan A Loisel, and Yoav Gilad. "Sex-specific genetic architecture of human disease". In: *Nature Reviews Genetics* 9.12 (2008), pp. 911–922.

[5] Stella A Belonwu et al. "Sex-stratified single-cell RNA-Seq analysis identifies sex-specific and cell type-specific transcriptional responses in Alzheimer's disease across two brain regions". In: *Molecular neurobiology* (2022), pp. 1–18.

[6] Zhaohao Huang et al. "Effects of sex and aging on the immune cell landscape as assessed by single-cell transcriptomic analysis". In: *Proceedings of the National Academy of Sciences* 118.33 (2021), e2023216118.

[7] Elisabetta Mereu et al. "Benchmarking single-cell RNA-sequencing protocols for cell atlas projects". In: *Nature biotechnology* 38.6 (2020), pp. 747–755.

[8] Orit Rozenblatt-Rosen et al. "Building a high-quality human cell atlas". In: *Nature Biotechnology* 39.2 (2021), pp. 149–153.

[9] Hoa Thi Nhu Tran et al. "A benchmark of batch-effect correction methods for single-cell RNA sequencing data". In: *Genome biology* 21 (2020), pp. 1–32.

[10] Malte D Luecken et al. "Benchmarking atlas-level data integration in single-cell genomics". In: *Nature methods* 19.1 (2022), pp. 41–50.

[11] Michael Eisenstein. "Single-cell RNA-seq analysis software providers scramble to offer solutions." In: *Nature Biotechnology* 38.3 (2020), pp. 254–257.

[12] Ricard Argelaguet et al. "Computational principles and challenges in single-cell data integration". In: *Nature biotechnology* 39.10 (2021), pp. 1202–1215.

[13] Kun Qian et al. "scINSIGHT for interpreting single-cell gene expression from biologically heterogeneous data". In: *Genome biology* 23.1 (2022), p. 82.

[14] Ziqi Zhang et al. "scDisInFact: disentangled learning for integration and prediction of multi-batch multi-condition single-cell RNA-sequencing data". In: *Nature Communications* 15.1 (2024), p. 912.

[15] Renjing Liu et al. "Integration of scRNA-seq data by disentangled representation learning with condition domain adaptation". In: *BMC bioinformatics* 25.1 (2024), p. 116.

[16] Kai Zhao, Hon-Cheong So, and Zhixiang Lin. "scParser: sparse representation learning for scalable single-cell RNA sequencing data analysis". In: *Genome biology* 25.1 (2024), p. 223.

[17] Joshua D Welch et al. "Single-cell multi-omic integration compares and contrasts features of brain cell identity". In: *Cell* 177.7 (2019), pp. 1873–1887.

[18] Grace XY Zheng et al. "Massively parallel digital transcriptional profiling of single cells". In: *Nature communications* 8.1 (2017), p. 14049.

[19] Maayan Baron et al. "A single-cell transcriptomic map of the human and mouse pancreas reveals inter-and intra-cell population structure". In: *Cell systems* 3.4 (2016), pp. 346–360.

[20] Åsa Segerstolpe et al. "Single-cell transcriptome profiling of human pancreatic islets in health and type 2 diabetes". In: *Cell metabolism* 24.4 (2016), pp. 593–607.

[21] Mauro J Muraro et al. "A single-cell transcriptome atlas of the human pancreas". In: *Cell systems* 3.4 (2016), pp. 385–394.

[22] Yue J Wang et al. "Single-cell transcriptomics of the human endocrine pancreas". In: *Diabetes* 65.10 (2016), pp. 3028–3038.

[23] Yurong Xin et al. "RNA sequencing of single human islet cells reveals type 2 diabetes genes". In: *Cell metabolism* 24.4 (2016), pp. 608–615.

[24] Karthik Shekhar et al. "Comprehensive classification of retinal bipolar neurons by single-cell transcriptomics". In: *Cell* 166.5 (2016), pp. 1308–1323.

[25] Evan Z Macosko et al. "Highly parallel genome-wide expression profiling of individual cells using nanoliter droplets". In: *Cell* 161.5 (2015), pp. 1202–1214.

[26] Tim Stuart et al. "Comprehensive integration of single-cell data". In: *cell* 177.7 (2019), pp. 1888–1902.

[27] Ilya Korsunsky et al. "Fast, sensitive and accurate integration of single-cell data with Harmony". In: *Nature methods* 16.12 (2019), pp. 1289–1296.

[28] Jialin Liu et al. "Jointly defining cell types from multiple single-cell datasets using LIGER". In: *Nature protocols* 15.11 (2020), pp. 3632–3662.

[29] Romain Lopez et al. "Deep generative modeling for single-cell transcriptomics". In: *Nature methods* 15.12 (2018), pp. 1053–1058.

[30] Brian Hie, Bryan Bryson, and Bonnie Berger. "Efficient integration of heterogeneous single-cell transcriptomes using Scanorama". In: *Nature biotechnology* 37.6 (2019), pp. 685–691.

[31] Laleh Haghverdi et al. "Batch effects in single-cell RNA-sequencing data are corrected by matching mutual nearest neighbors". In: *Nature biotechnology* 36.5 (2018), pp. 421–427.

[32] W Evan Johnson, Cheng Li, and Ariel Rabinovic. "Adjusting batch effects in microarray expression data using empirical Bayes methods". In: *Biostatistics* 8.1 (2007), pp. 118–127.

[33] Manpreet Randhawa et al. "Evidence for the ectopic synthesis of melanin in human adipose tissue". In: *The FASEB Journal* 23.3 (2009), p. 835.

[34] Coralie Fontaine et al. "Hedgehog signaling alters adipocyte maturation of human mesenchymal stem cells". In: *Stem cells* 26.4 (2008), pp. 1037–1046.

[35] Bartłomiej Łukaszuk et al. "Adipose tissue place of origin and obesity influence sphingolipid signaling pathway in the adipocytes differentiated from ADMSCs isolated from morbidly obese women". In: *Biochemical Pharmacology* 223 (2024), p. 116158.

[36] Ryo Ito et al. "Mitochondrial biogenesis in white adipose tissue mediated by JMJD1A-PGC-1 axis limits age-related metabolic disease". In: *Iscience* 27.4 (2024).

[37] Aldi T Kraja et al. "Associations of mitochondrial and nuclear mitochondrial variants and genes with seven metabolic traits". In: *The American Journal of Human Genetics* 104.1 (2019), pp. 112–138.

[38] Xihui Lin and Paul C Boutros. "Optimization and expansion of non-negative matrix factorization". In: *BMC bioinformatics* 21.1 (2020), p. 7.

[39] James Bergstra, Daniel Yamins, and David Cox. "Making a science of model search: Hyperparameter optimization in hundreds of dimensions for vision architectures". In: *International conference on machine learning*. PMLR. 2013, pp. 115–123.

[40] Xiuwei Zhang, Chenling Xu, and Nir Yosef. "Simulating multiple faceted variability in single cell RNA sequencing". In: *Nature communications* 10.1 (2019), p. 2611.

[41] F Alexander Wolf, Philipp Angerer, and Fabian J Theis. "SCANPY: large-scale single-cell gene expression data analysis". In: *Genome biology* 19 (2018), pp. 1–5.

[42] Margo P Emont et al. "A single-cell atlas of human and mouse white adipose tissue". In: *Nature* 603.7903 (2022), pp. 926–933.