Interpreting batch correction of single-cell variational inference at scale

Katherine Wu

Electrical Engineering and Computer Sciences
University of California, Berkeley

Technical Report No. UCB/EECS-2022-91
http://www2.eecs.berkeley.edu/Pubs/TechRpts/2022/EECS-2022-91.html

May 13, 2022

Acknowledgement

I would like thank my advisor Nir Yosef for his advice and feedback throughout the project, as well as everyone on the scvi-tools team for their support over the past two years.

Interpreting batch correction of single-cell variational inference at scale

by

Katherine Wu

A thesis submitted in partial satisfaction of the

requirements for the degree of

Masters of Science

in

Electrical Engineering and Computer Science

in the

Graduate Division

of the

University of California, Berkeley

Committee in charge:

Professor Nir Yosef, Chair
Professor Nilah Ioannidis

Spring 2022

The thesis of Katherine Wu, titled Interpreting batch correction of single-cell variational inference at scale, is approved:

Chair	Nir Yosef	Date	5/11/22
	Nilah Ioannidis	Date	5/13/22
		Date	

University of California, Berkeley

Interpreting batch correction of single-cell variational inference at scale

Abstract

Interpreting batch correction of single-cell variational inference at scale

by

Katherine Wu

Masters of Science in Electrical Engineering and Computer Science

University of California, Berkeley

Professor Nir Yosef, Chair

Single-cell RNA sequence datasets often contain unwanted technical variation from differences in sample collection, protocol, sequencing depth, experimental labs, and biological factors. These nuisance factors, known as batch effects, are especially common in newer datasets that span multiple conditions and hundreds of donors. To correct for such batch effects, integration methods like single-cell variational inference (scVI) combine samples of data and produce a self-consistent version for downstream analysis. In this thesis, we benchmark scVI's current performance on complex integration tasks of 100+ donor datasets, evaluating its ability to both remove batch effects and retain important biological information. We further propose the addition of a donor embedding to the model architecture, and demonstrate that the embedding is effective at interpreting batch correction for confounding covariates. Finally, we assess scVI integration in relation to gene expression through a scoring protocol that measures the batch sensitivity of each gene.

To my friends and family.

Contents

List of Figures

List of Tables

Acknowledgments

I would like thank my advisor Nir Yosef for his advice and feedback throughout the project, as well as everyone on the scvi-tools team for their support over the past two years.

Chapter 1

Introduction

1.1 Background

scRNA-seq technology and advances

Single-cell RNA sequencing (scRNA-seq) is a method that enables gene expression measurement at the single-cell resolution. Prior to single-cell sequencing, it was impossible to decouple inter-sample variation from intra-sample variation— all cell types were mixed together, so it was not possible to understand relationships between individual genes and specific biological phenomena. With scRNA-seq, it is now possible to distinguish cell type clusters, arrange populations of cells according to novel hierarchies, and identify cells transitioning between states. This can lead to a clearer view of the dynamics of tissue and organism development, as well as structures within cell populations that were previously seen as homogeneous [8].

To sequence cells in this manner, the scRNA-seq procedure involves taking cells from a tissue and adding each cell to an individual droplet of water. A droplet-specific barcode is added to the mRNA in order to trace back which molecule came from which cell. Finally, the mRNA is translated into cDNA and sequenced [3]. The result is a gene expression matrix X_{ng} that contains the counts for the expression of every gene g in cell n.

This data allows researchers to understand the many facets of a cell's unique molecular identity. Cells can be categorized by cell type, which can further be classified into finer subtypes through a hierarchical taxonomy. They can also be categorized by more transient properties, referred to as cell states. For example, we can look at the temporal progression of a cell during differentiation or the temporal vascillation of a cell during the cell cycle. We can also look at spatial context, such as its physical position in the tissue and the identity of neighboring cells [17]. Together these factors span the space of possible cell states and can be likened to a superposition of 'basis vectors,' each determining a different aspect of cellular organization and function.

Beyond cellular identity, another goal of single cell sequencing is to construct a comprehensive atlas of all human cell types and subtypes, including activity states, physical loca-

tions, and lineage relationships through development [11]. The Human Cell Atlas project, for example, aims to transform our understanding of the organization and function of tissues in health and disease. Researchers have also used scRNA-seq to explore cellular interactions in immunology [15], tumor evolution [5], and other subfields of biology [1].

Technical challenges in single-cell data

The analyses that biologists wish to perform on scRNA-seq data can be separated into several main categories [19]:

- Filtering: Checking if a cell is in a droplet or if it is just noise, and filtering genes to know which ones have biological signal
- Clustering: Finding what cell types are present in the experiment
- Differential Expression: Finding what genes are expressed in those cells
- Disentanglement: Separating technical variance from biological signal
- Imputation: Establishing whether a zero in a matrix is a technical or biological zero
- Multiple donor scenarios: Understanding heterogeneity of samples with multiple human donors (especially with different clinical phenotypes)

This thesis will focus on the challenge of separating technical variance from biological signal in multiple donor scenarios. In single-cell datasets with a large number of donors, data can be confounded by technical variables such as site and method of sampling. Samples are also affected by variables of biological interest, such as a donor's age, sex, and clinical status.

scVI

Single-cell variational inference, or scVI, is a fully probabilistic approach for the normalization and analysis of scRNA-seq data. Unlike other models that assume a generalized linear model, scVI is based on a hierarchical Bayesian model with conditional distributions specified by deep neural networks. The transcriptome of each cell is encoded through a nonlinear transformation into a low-dimensional latent vector of normal random variables. This latent space is then decoded by another nonlinear transformation to generate a posterior estimate of the distributional parameters of each gene in each cell [6]. scVI stands out from other methods by modelling two key noise factors in scRNA-seq data— variation in library size and batch effects. It also is able to ensure consistency and interpretability by performing a range of analysis tasks using the same generalized model, whereas other methods require different models for different tasks. Finally, while other methods can only be applied to tens of thousands of cells, scVI can be applied to hundreds of thousands of cells.

1.2 Problem Formulation

As mentioned earlier, current datasets often include samples generated from multiple labs and across multiple conditions. Because cells come from different sources, unwanted technical variation arises from differences in sequencing depth, sequencing lanes, read length, plates or flow cells, protocol, experimental labs, sample acquisition and handling, sample composition, reagents or media, and sampling time [7]. This increased complexity results in nonlinear and nested batch effects in single-cell data.

Data integration methods such as scVI have been created to mitigate these batch effects; they combine datasets or samples of high-throughput sequencing data to produce a self-consistent version of the data for downstream analysis. In this thesis, we have four goals:

- Evaluate scVI's performance on integration tasks of 100+ donor datasets: we will choose and compute metrics for the chosen datasets, as well as qualitatively evaluate performance through latent space visualizations.
- Interpret scVI batch effects: how well does scVI capture batch effects from different donors/sites/other covariates? The addition of the batch embedding will help us understand the donor latent space.
- Interpret scVI batch correction for individual genes: what genes are most sensitive to batch? Are there biological or technical properties that make a gene more prone to batch effects?
- Improve scVI performance: by switching from a one-hot encoding to a batch embedding, we hope to reduce the time and memory constraints required to train the scVI model.

1.3 Related Work

In addition to scVI, several other data integration methods exist for biologists to integrate samples of data and remove batch effects. In particular, it is found that BBKNN, Scanorama, and scVI perform well on complex integration tasks, while Seurat v3 performs well on simpler tasks with distinct biological signals [2, 6, 7, 9]. Harmony and scVI are partially effective for scATAQ-seq data integration [7]. Below is a summary of the popular data integration methods used in scRNA-seq.

Scanorama

Scanorama uses panorama stitching of scRNA-seq data to correct for batch effects through similar cells identified across batches [2]. Approximate SVD is used to transform gene expression data into a lower dimensional subspace. Then, an approximate nearest neighbor search is performed to identify mutually linked cells across batches. It searches across all

batches and determines the priority of dataset merging based on percentage of matching cells in the batch. Finally, batches are merged into panoramas using a weighted average of vectors between local matching cells.

BBKNN

Batch balanced k-nearest neighbors (BBKNN) first computes the k-nearest neighbors in a lower dimensional principal component space. The nearest neighbors are identified in a batch-balanced manner using Euclidean distances. It transforms the neighbor information into connectivities to construct a graph that links cells together across batches [9]. The resulting graph is used for clustering and other standard workflows.

Harmony

Harmony is an unsupervised joint embedding method that uses iterative clustering to align cells from different batches. The algorithm combines the batches and projects the data into a lower dimensional space using PCA. It uses an iterative procedure consisting of 4 steps: first the cells are grouped into multiple-dataset clusters using a variant of soft k-means clustering. Then Harmony computes a global centroid for each cluster and a centroid for each dataset. A correction factor is then computed for each dataset. Finally, the corection factor is used to correct each cell with a cell-specific factor. This procedure is applied iteratively until convergence. The resulting normalized Harmony vectors can be used as input for analysis workflows [4].

Seurat 3

Seurat 3 uses canonical correlation analysis to compute the linear combinations of genes with maximum correlation across batches. It then identifies mutual nearest neighbors (MNNs) of similar cell states across batches in the normalized CCA subspace. The shared nearest neighbor graphs are used to assess cell type similarity. A correction vector is computed using the difference in expression profiles between cells to perform the data transformation [16].

1.4 Contributions

In this thesis, we benchmark scVI on large single-cell donor studies and evaluate its ability to account for batch effects in complex integration tasks. We further propose the addition of a batch embedding to scVI's existing model architecture. Finally, we explore how the batch embedding can be used to understand batch correction with multiple confounding covariates. We structure the thesis as follows:

- In section 2, we provide a description of the metrics used to benchmark SCVI on large donor datasets. We further describe scVI's existing architecture and present the addition of the batch embedding. Finally, we describe the experiment setup for evaluating scVI's performance.

- In section 3, we describe the datasets used to evaluate scVI. We then present an evaluation of batch effects and runtime. We interpret the batch embedding space using UMAP visualizations of the batch embedding. Finally, we discuss genes that are "sensitive to batch."

Chapter 2

Methodology

2.1 Metrics

The accuracy of our single cell generative model is determined by both its ability to remove batch effects and its ability to conserve biological variance. We implement the series of metrics listed below to evaluate scVI's performance on both axes. For removal of batch effects, batch silhouette width (ASW) measures batch effect removal per cell identity label and batch local inverse Simpson's Index (iLISI) measures batch effect removal independent of cell identity. For conservation of biological variance, we have the following label conservation metrics: cell type local inverse Simpson's Index (cLISI), Adjusted Rand Index (ARI), Normalized Mutual Information (NMI), and cell type silhouette width. Finally, we evaluate the model on scalability and usability through measurement of CPU time and memory use.

Silhouette Score

The silhouette score, or average silhouette width (ASW), is calculated using the mean within-cluster distance and mean nearest-cluster distance for each sample [13]. We define the silhouette coefficient for a sample as

$$s(i) = \frac{b(i) - a(i)}{\max\left(a(i), b(i)\right)}$$

where $a(i)$ is the average distance from point i to all points in the same cluster and $b(i)$ is the lowest average distance from i to all points in the same cluster c among all clusters C. Thus, the silhouette width ranges between -1 and 1, where 1 represents dense and well-separated clusters and 0 or -1 correspond to overlapping clusters or strong classification.

ASW is used for determining the degree of separation of the clusters, and the distances are defined based on scVI embedding output. In this case, "clusters" refer to the groupings of cells of the same cell type or batch label. We calculate silhouette width for both cell type labels and batch labels. Within cell type ASW, the silhouette score is linearly scaled to

range $[0, 1]$ using the following equation:

$$\text{ASW}_c = \frac{\text{ASW} + 1}{2}$$

where larger values indicate that cells of different types make well-separated clusters. For batch ASW, we again scale to range $[0, 1]$ using

$$\text{ASW}_b = 1 - |\text{ASW}|$$

In this case, a score of 1 would indicate that the batches are ideally mixed, while a score of 0 would indicate strongly separated, not mixed batches.

LISI

LISI can also be used to assess both batch mixing (iLISI) and cell type separation (cLISI). This metric is measured by looking at the neighborhood of each cell to see what labels they come from. More specifically, we count the number of cells that can be drawn from a neighbor list before one label is observed twice [7]. This is a score from 1 to N, where N is the total number of labels in the data set. In the case of iLISI, we use batches as the labels. Ideally they should come from batches representative of the overall population, so a higher score represents better batch mixing. For cLISI, we use cell type labels, so a lower score represents better cell type separation. Both cLISI and iLISI are rescaled by min/max observed median scores across tasks.

As an example, suppose that one of the columns in the metadata is a categorical variable with three categories. If LISI is approximately equal to 3 for an item in the matrix, that means that the item is surrounded by neighbors from all 3 categories. On the other hand, if LISI is approximately equal to 1, then the item is surrounded by neighbors from 1 category.

Adjusted Rand Index

The rand index measures similarity between two clusters by considering all pairs of samples and counting the number of pairs that are assigned to the same cluster when given the predicted and true clusterings [10]. To compute the Adjusted Rand Index, we adjust the raw RI score to account for chance: $ARI = \frac{RI - E[RI]}{max(RI) - E[RI]}$. The index ranges from 0 to 1, with 0 indicating random labelling and 1 indicating that the clusterings are identical. In our case, we compare SCVI's labels with the labels given by the NMI-optimized Louvain clustering computed on the integrated dataset.

$$\text{ARI} = \frac{\sum_{i,j} \binom{n_{ij}}{2} - [\sum_i \binom{a_i}{2} \sum_j \binom{b_j}{2}] \binom{n}{2}}{\frac{1}{2}[\sum_i \binom{a_i}{2} + \sum_j \binom{b_j}{2}] - [\sum_i \binom{a_i}{2} \sum_j \binom{b_j}{2}] \binom{n}{2}}$$

Normalized Mutual Information

Normalized mutual information also compares the overlap of the SCVI cell type labels with the Louvain clusters computed on the integrated dataset [6]. It ranges from 0 to 1, where 0 indicates an uncorrelated clustering and 1 indicates a perfect match. It is scaled using the mean of entropy terms for cell type and cluster labels.

$$\text{NMI} = \frac{I(P;T)}{\sqrt{H(P)H(T)}}$$

where H is Shannon entropy, I is mutual entropy, and P,T are the empirical categorical distributions for predicted and true clusterings.

2.2 scVI Model Architecture

scVI models the observed expression x_{ng} of each gene g in cell n as a sample drawn from a zero-inflated negative binomial (ZINB) distribution $p(x_{ng}|z_n, s_n, l_n)$ conditioned on the batch annotation s_n of each cell and two latent random variables. One latent variable, l_n, is a one-dimensional Gaussian that represents nuisance variation due to differences in capture efficiency and sequencing depth. In the most recent version of scVI, l_n is treated as an observed variable instead, equal to the total RNA UMI (Unique Molecular Identifier) count of the cell. UMIs are "molecular barcodes" added to sequencing libraries before PCR amplification, so UMI count represents the absolute number of observed transcripts per cell. The second latent variable, z_n, is a low-dimensional vector of Gaussians representing the remaining variation, and is intended to reflect the biological differences between cells. It represents each cell as a point in a low-dimensional latent space that can be used for clustering and visualization [6].

We use a neural network to map the latent variables to the parameters of the ZINB distribution.

Generative Process

$$\begin{aligned}
z_n &\sim \text{Normal}(0, I) \\
l_n &\sim \text{LogNormal}(l_\mu^T s_n, l_{\sigma^2}^T s_n) \\
\rho_n &= f_w(z_n, s_n) \\
\pi_{ng} &= f_h^g(z_n, s_n) \\
x_{ng} &\sim \text{ZINB}(l_n \rho_n, \theta_g, \pi_{ng})
\end{aligned}$$

$$\begin{aligned}
f_w(z_n, s_n) &: R^d \times 0, 1^K \to \Delta^{G-1} \\
f_h(z_n, s_n) &: R^d \times 0, 1^K \to (0, 1)^T
\end{aligned}$$

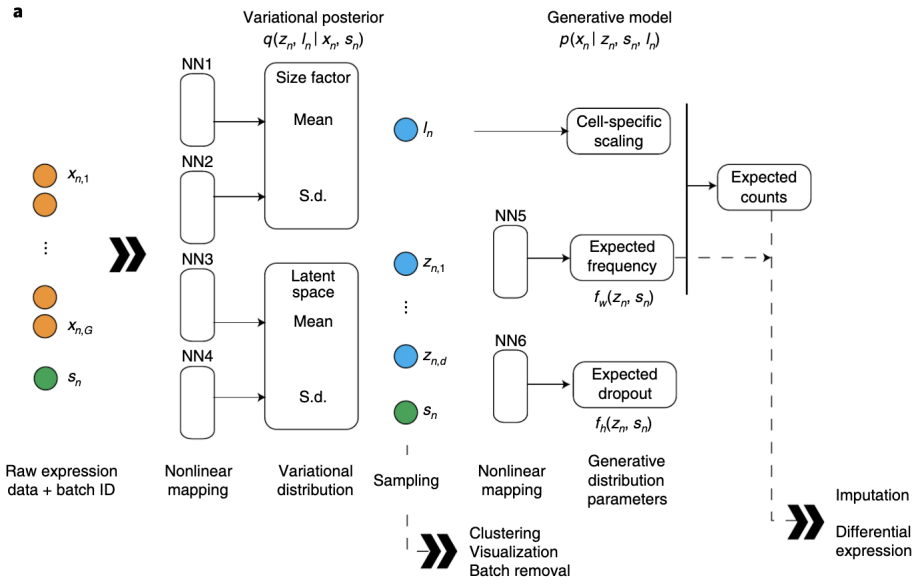


Figure 2.1: scVI model architecture [Image taken from Lopez et al., 2018]

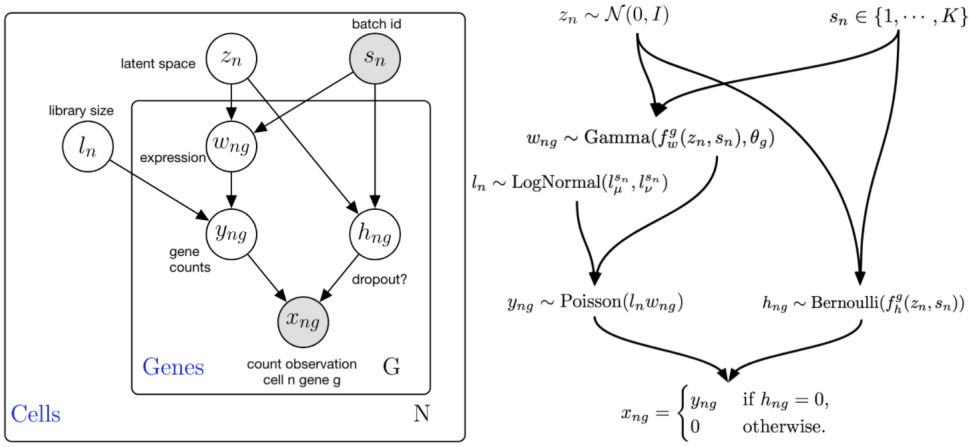


Figure 2.2: scVI generative process [Image taken from scvi-tools.org]

As described above, gene expression depends on cell specific latent variable z_n and batch id s_n. The library size variable l_n depends on the empirical mean and variance of the log library size over cells, l_μ and l_{σ^2}. It should be noted that l_n is not the log library size itself, but a scaling factor that correlates strongly with library size. The generative process uses two neural networks f_w and f_h. The f_w network decodes the denoised/normalized gene expression, which by default is a vector that sums to 1 within a cell. It is constrained during inference to encode the mean proportion of transcripts expressed across all genes through a softmax activation in the last layer. The f_h network decodes whether or not a particular entry has dropped out due to technical effects, otherwise known as the the non-zero inflation probability. Both f_w and f_h can be interpreted as expected frequencies. The expression data x_{ng} are generated from a zero inflated negative binomial distribution parameterized by its mean, inverse dispersion, and non-zero-inflation probability[1].

The two neural networks allow us to go beyond a linear model framework to better encode gene expression. Each network has one, two, or three fully connected layers. The activation functions are ReLU functions. Weights for some layers are shared between f_w and f_h.

Inference

We wish to determine the posterior distribution $p(x_{ng}|z_n, l_n, s_n)$, which combines prior knowledge from the latent space with information acquired from data matrix X. However, we cannot directly apply Bayes rule because the denominator $p(x_n|s_n)$ in intractable. We can instead apply variational inference to learn the model parameters and approximate posterior distribution. First we integrate out the latent variables w_{ng}, h_{ng} and y_{ng} because $p(x_{ng}|z_n, l_n, s_n)$ has closed-form density. We can now approximate the posterior $p(z_n, l_n|x_n, s_n)$ using our variational distribution $q(z_n, l_n|x_n, s_n)$, assuming that:

$$q(z_n, l_n|x_n, s_n) = q(z_n|x_n, s_n)q(l_n|x_n, s_n)$$

The variational distribution $q(z_n|x_n, s_n)$ is a Gaussian with a diagonal covariance matrix, mean, and covariance given by an encoder network applied to (x_n, s_n). The two priors in this case are x_n, the expression data of cell n, and s_n, the batch id of cell n. The variational distribution $q(l_n|x_n, s_n)$ is a log normal distribution with a scalar mean and variance also given by an encoder network applied to (x_n, s_n).

The variational lower bound is as follows:

$$\begin{aligned} \log p(x|s) \geq\ & E_{q(z,l|x,s)} \log p(x|z, l, s) \\ & - D_{KL}(q(z|x, s)||p(z)) \\ & - D_{KL}(q(l|x, s)||p(l)) \end{aligned}$$

To optimize the lower bound, we use the analytic expression for $p(x|z, l, s)$ and for the Kullback-Leibler divergences. Coupled with neural network approximation, this allows us

[1] Adopting scVI's current best practices, my work uses a negative binomial distribution instead of the original ZINB distribution.

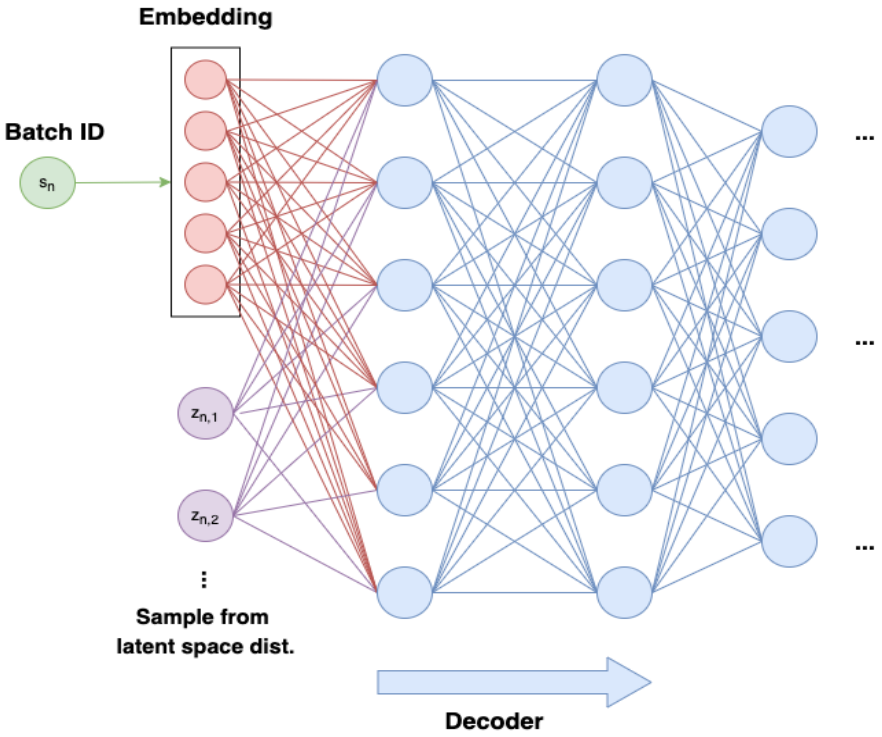


Figure 2.3: scVI generative model architecture with batch embedding

to efficiently carry out inference with arbitrary models, including those with conditional distributions specified by neural networks. We also subsample from the training set to increase stochasticity. At each iteration, we focus on a randomly sampled subset of the data and do not need to go through the whole dataset. Since the number of genes is limited to a few thousands, these mini-batches of cells can be handled by a GPU.

We use an Adam optimizer with $\epsilon = 0.01$ and batch normalization during learning. We optimize the objective function until convergence.

2.3 Batch Embedding

In the current architecture, we represent each donor as a one-hot encoding. For each s_n in S we represent the donor by a vector of size $1 \times |S|$, and all elements of the vector are 0 except for the element at index n. In this representation, we feed in a one-hot encoding matrix of size $n \times |S|$ into the decoder, where n is the number of samples and $|S|$ is the number of donors.

One drawback to this representation is its large size— as the dataset size increases, the size of the one-hot encoding matrix scales linearly to the number of donors sampled. While

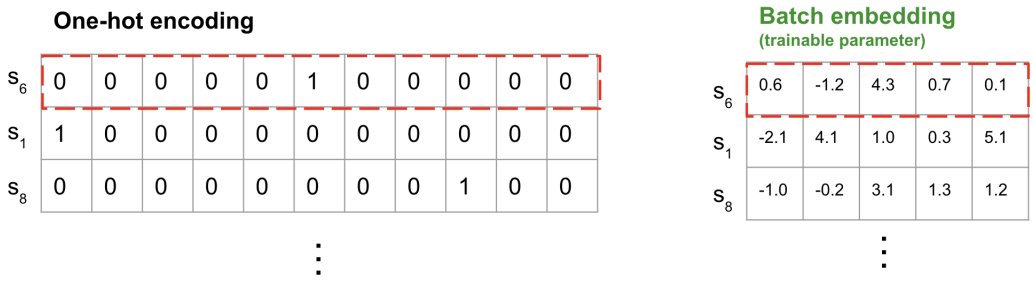


Figure 2.4: Batch embedding visualization

this would not be a issue for single-cell datasets with a few donors, there is a decrease in performance when we scale up to the 100+ donor datasets that are becoming more common in single-cell studies. Furthermore, it treats each donor as an independent entity with no relation to other donors, when in reality we know that donors are interconnected to other donors through covariates such as age and health status. Indeed, what we really desire is some notion of similarity between donors that would allow us to encode each donor into the matrix more efficiently. For example, we can encode differing levels of similarity based on their age, gender, health conditions, and geographic location, among other factors. If we are able to encode similarities between donors into our matrix representation before we pass it into the decoder, we can reduce the size of the matrix as well as discover the greatest sources of variation and similarity between donors. This will further allow us to investigate the extent to which non-biological variation, i.e. technical variation from different sampling methods or sites, plays a role in the generation of the final latent space.

To achieve this goal, we propose a batch embedding that condenses this input matrix from size $n \times |S|$ into a matrix of size $n \times d$, where d is the dimension of the embedding (we use a default of $d = 5$ in our model). Instead of using a one-hot encoding for every donor, we represent each donor by a vector of size $1 \times d$. We can think of each dimension as a donor attribute that the model learns, i.e. an attribute that explains the variation in donors well. Thus, we treat the batch embedding as parameters in our model that get updated during training.

2.4 Experiment Setup

To evaluate batch effects, we run the original scVI model on subsets of the donor population ranging from 10 donors to 100 donors, going in increments of 10 donors. We repeat this procedure for 3 trials per subset. In doing so, we can see how the number of donors in the dataset affects the batch effects in the resulting data after training. For the Stephenson dataset (see description of all datasets below), we use `initial_clustering` as the label key and `patient_id` as the batch key. For the Ren dataset, we use `cell_type` as the label

key and `patient` as the batch key. In both cases, the batch key is chosen to reflect the assumption that each patient comes from a different batch. While other covariates such as smoking status, age, and site could have been used, we chose to use the patient variable in this case because it the most obvious "batch" in the data. For label key, we chose the variable corresponding to the sample's cell type labelling.

First we subset the AnnData to only contain observations from the randomly chosen set of d donors. We then subset the AnnData to the top 1200 highly variable genes, and filter genes to only include genes with a count greater than three using the scanPy library [18]. We then train the SCVI model on the subsetted AnnData using the default parameters: 128 nodes per hidden layer, 10 dimensional latent space, 1 hidden layer for encoder and decoder NNs, and 0.1 dropout rate. We use a constant negative binomial dispersion parameter per gene across cells and a Normal latent distribution. We use a negative binomial to represent gene likelihood. During training, we record runtime and memory usage of the model.

After the model is trained, we compute a k nearest neighbor graph on the latent space and use UMAP to estimate connectivities of the data points. To visualize the generated latent space, we plot the UMAP and use colors corresponding to the covariates we wish to investigate. We also run PCA on the original data with 50 principle components and compute the kNN graph for the PCA-generated latent space. We then plot the PCA UMAP as a measure of comparison for the SCVI generated latent space.

To evaluate the SCVI model, we compute metrics on the resulting AnnData. Removal of batch effects is evaluated through the Batch ASW and iLISI metrics. Conservation of biological variance is measured through cLISI, ARI, NMI, and cell type ASW.

Chapter 3

Results

3.1 Datasets

COVID-19 Immune Features (Ren 2021)

This dataset contains 284 single-cell RNA sequence samples from 196 COVID-19 patients and controls, totalling 1.46 million cells [12]. Data was gathered from 39 institutes or hospitals from different regions in China. It includes 171 COVID-19 patients, with 22 patients with mild or moderate symptoms, 54 hospitalized patients with severe symptoms, 95 recovered patients, and 25 healthy controls. The cohort ranges from age 6 to 92 years old, with aged patients enriched in severe groups. No significant different between sex of patients was noted between moderate and severe groups. From the 284 samples, 249 were obtained from peripheral blood mononuclear cells (PBMCs) with or without further sorting for B or T cells, and 35 were from the respiratory system, with 12 bronchoalveolar lavage fluid (BALF) samples, 22 sputum samples, and 1 sample of pleural fluid mononuclear cells (PFMCs).

Most samples were subjected to scRNA-seq based on the 10x Genomics 5' sequencing platform to generate both gene expression and T cell receptor or B cell receptor data. In total, 1,462,702 single cells were obtained, with an average of 4,835 unique molecular identifiers (UMIs) representing 1,587 genes. 64 cell types were derived. It was noticed that CD8+, CD4+ T, and plasma B cells were more enriched in BALF than PBMCs. Additionally, proliferative and activated B and T cells and macrophages were more enriched in severe COVID-19 patients in the disease progression stage.

Single-cell multiomics analysis of the immune response in COVID-19 (Stephenson 2021)

Single-cell transcriptome, surface proteome and T and B lymphocyte antigen receptor analyses were performed for over 780,000 peripheral blood mononuclear cells from a cohort of 130 patients with varying severities of COVID-19 [14]. Data was gathered from 3 UK centers in Newcastle, Cambridge, and London, and controls included healthy volunteers, individu-

als with non-COVID-19 severe respiratory illness, and healthy volunteers administered with IV-LPS as a surrogate for inflammatory response. In total, 781,123 cells from 143 samples were included in the dataset.

Cells were manually annotated based on RNA expression of known marker genes supported by surface protein expression of markers employed in flow cytometry. 18 cell subsets were defined, with an additional 27 cell states identified after subclustering. A relative expansion of proliferating lymphocytes, proliferating monocytes, platelets, and mobilized hematopoietic stem and progenitory cells was observed with increasing COVID-19 severity. Plasmablasts and B cells were also expanded in severe and critical disease.

3.2 Batch Effect Evaluation

Visualization of Latent Space

Comparing the UMAPs of the PCA-generated latent space and the scvi-generated latent space, we see that scVI successfully clusters cell types for both datasets, across all numbers of donors (3.1, 3.7). In addition, we can observe how covariates contribute to batch effects by looking at the scVI latent space through the lens of the donors in which the cells came from. We first examine the features qualitatively to understand the major confounding factors in the data.

In the Stephenson dataset, we plot the UMAPs for patient ID, COVID-19 status, sex, site, and age. From the patient ID UMAP (3.10), we see some stratification by patient in the upper right cluster as well as the bottom right cluster. In particular, the patches of red and yellow along the left side of the CD4 and CD8 cell clusters indicate that there is technical or biological variation from those donors. To investigate this further, we can observe the sites that the data was gathered from (3.6). This UMAP plots the scVI latent space and colors the cells by the site of the donors, which come from Cambridge, Newcastle, and Sanger. From this plot, we see that the variation from earlier is explained by technical variation. The Sanger donor cells are all stratified to the left side of the CD cell clusters, while the Ncl donor cells are all stratified to the right. For the B cell cluster, the left side contains Ncl cells while the right contains Cambridge and Sanger cells. The Cambridge cells tend to be in the middle of the cells of the other two sites, and more mixed with the cells of the Sanger site. There also appears to be fewer Cambridge cells overall; the CD14 and CD18 cluster on the bottom right has very few blue patches. To round out the analysis, we explore other covariates such as the clinical status, sex, and age of the donors. From Figure 3.5, we see that the large patch of 'nan' values stratified in the clinical status UMAP is similar to the stratification in the site UMAP: the Sanger site produces many 'nan' values for clinical status. Looking at sex and age, both covariates appear to be well mixed within their cell type clusters.

We run similar analyses for the Ren dataset, which also contains gene expression data for a large sample of COVID-19 donors. In Figure 3.10, we observe that the patient ID UMAP

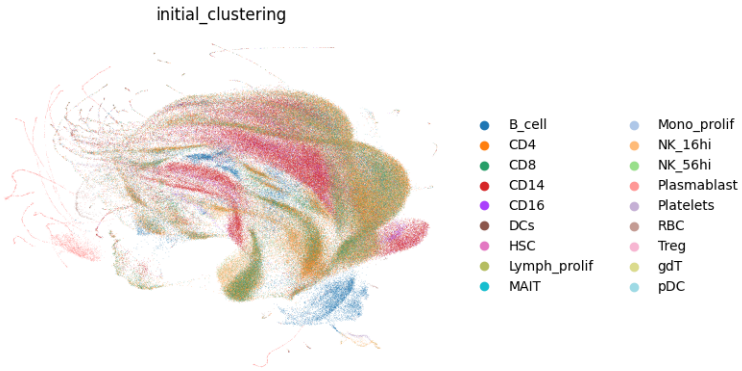


Figure 3.1: Stephenson PCA cell type UMAP for 50 donors

also exhibits standalone patches of color indicating clusters that are not well mixed. For example, there are large yellow, magenta, and green patches in the monocyte cluster and smaller unique patches of color along the edges of the CD4 and CD8 clusters. These indicate that certain donors exhibit variation, either biologically or due to technical error, that stratify them from the other donors in the study. Looking at the UMAPs for batch and city (3.11), we see that some of the variation can be explained by the different sample sites that the data was gathered from. In particular, the Beijing and Harbin sites are stratified from the rest and from their own patches on the edges of the cell clusters. Looking at the sample type and COVID-19 severity of the donors, we can get a clearer picture of what contributes to this city-to-city variation. More specifically, we see that severe/critical COVID patients are stratified away from mild/moderate ones in all of the major clusters. This corresponds to the variation we saw earlier with Beijing and Harbin sites. We also see the exact same stratification with sample type, in that it appears that severe/critical patients had frozen PBMC samples while mild/moderate patients had fresh PBMC samples taken.

Metrics

We now observe the overall benchmarking metrics across different numbers of donors. The goal of this experiment is to discover if batch effects increase or decrease as a result of scaling up to larger donor datasets. In particular, the hypothesis is that having more donors in the experiment would lead to more sources of technical variation (i.e. donors across various cities, ages, severities, etc.), and thus create greater batch effects in the data that could be captured through the benchmarking metrics.

Table 3.1 contains the cell type ASW, batch ASW, iLISI, and cLISI metrics for the batch-embedding scVI latent space computed on the Stephenson dataset, with metrics computed using 'site' as the batch. Metrics were calculated using 3 trials per subset, and averaged for every subset. We note that the cell type ASW remains relatively neutral as the number of

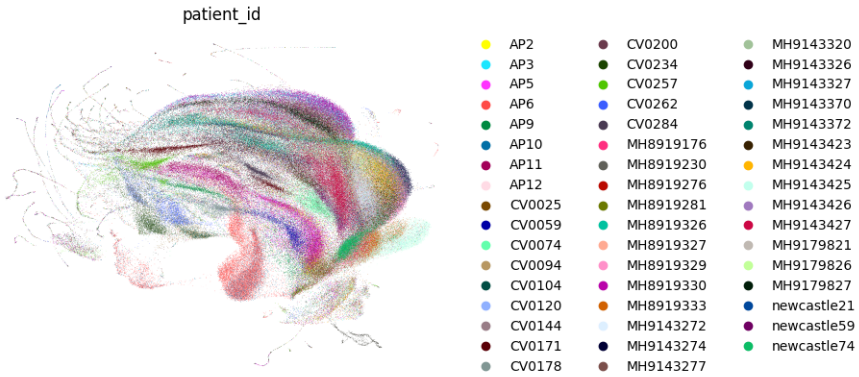


Figure 3.2: Stephenson PCA patient ID UMAP for 50 donors

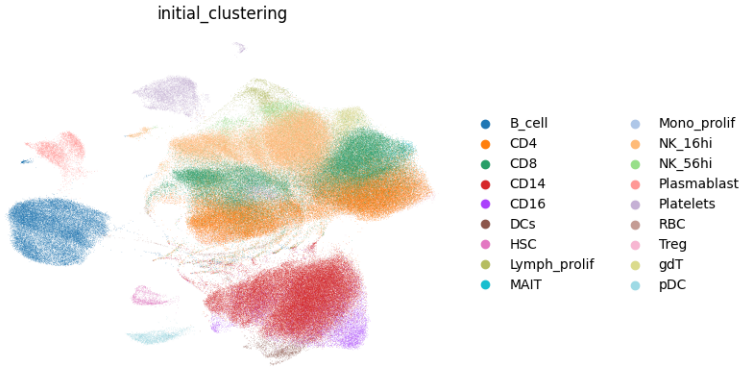


Figure 3.3: Stephenson scVI cell type UMAP for 50 donors

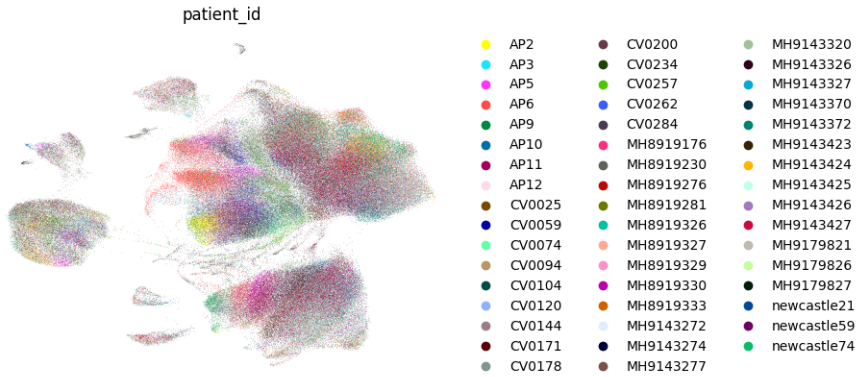


Figure 3.4: Stephenson scVI patient ID UMAP for 50 donors

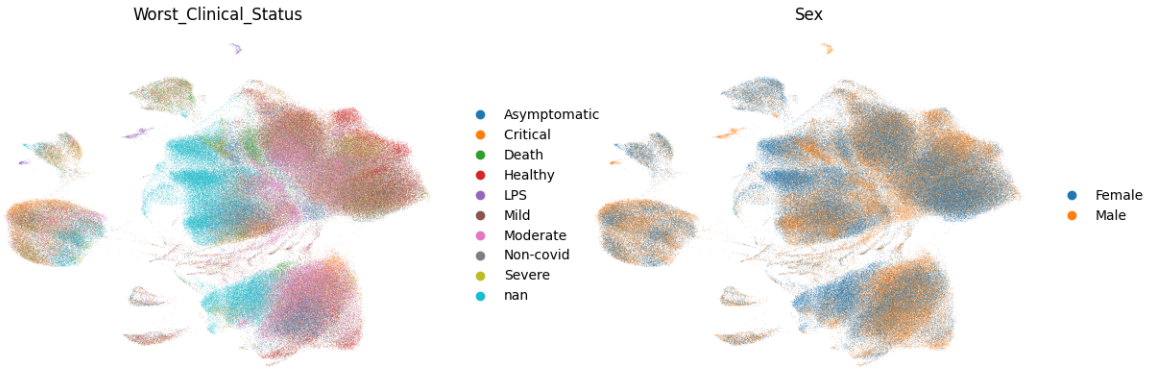


Figure 3.5: Stephenson scVI COVID status and sex UMAPs for 50 donors

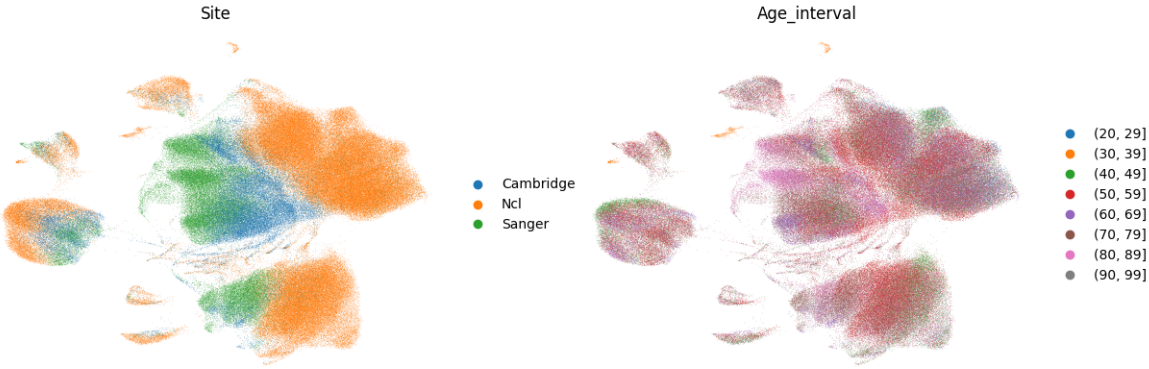


Figure 3.6: Stephenson scVI site and age UMAPs for 50 donors

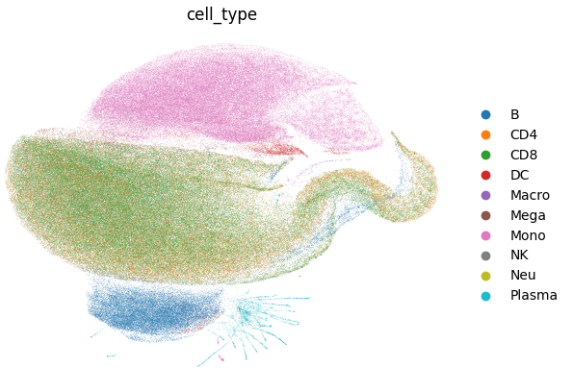


Figure 3.7: Ren PCA cell type UMAP for 50 donors

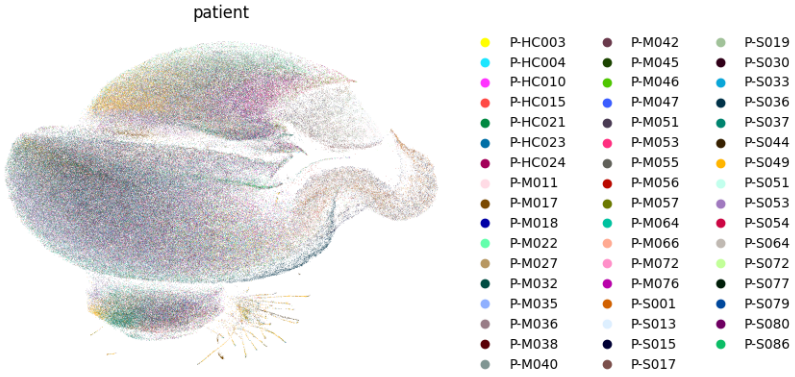


Figure 3.8: Ren PCA patient ID UMAP for 50 donors

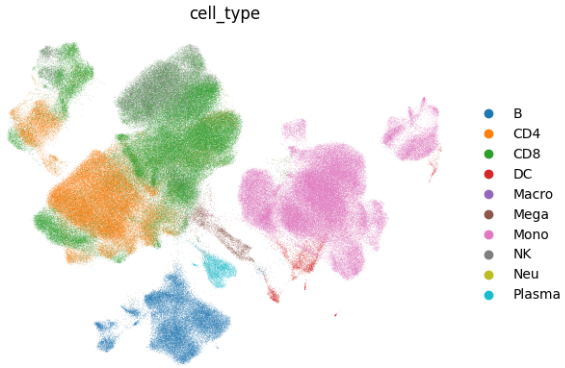


Figure 3.9: Ren scVI cell type UMAP for 50 donors

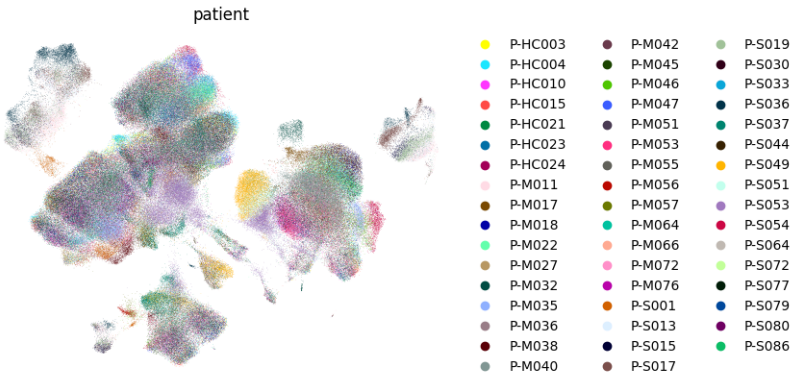


Figure 3.10: Ren scVI patient ID UMAP for 50 donors

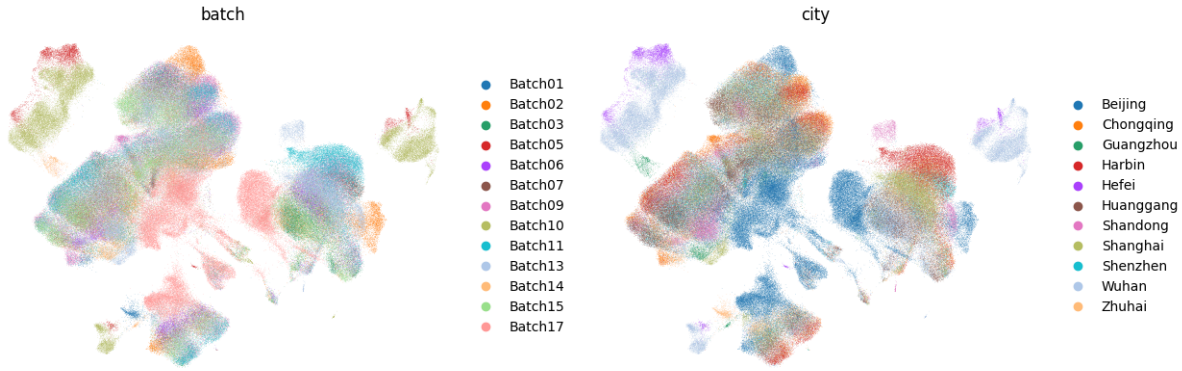


Figure 3.11: Ren scVI batch and city UMAPs for 50 donors

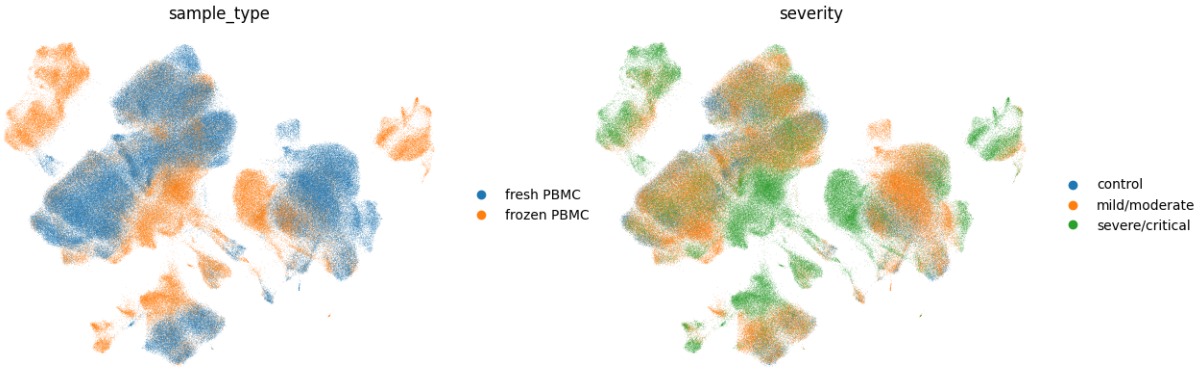


Figure 3.12: Ren scVI sample and severity UMAPs for 50 donors

donors increases: there is an increase from 10 to 20 donors, but after that it fluctuates in the range between .53 and .54 and does not exhibit a noticeable pattern. We can interpret this to mean that scVI's overall ability to cluster cell types remains the same, and does not get significantly worse as we scale to 100+ donor datasets. Looking at batch ASW, there is a similarly large drop from 10 to 20 donors, and then the rest of the values fluctuate around .40 to .44. We notice a slight downward trend in that the values are generally higher in the 10-40 donor range, and decrease as we scale up to 50-100 donors. More specifically, we see the 10-40 donor range has values ranging from .43 to .46, while the 50-100 donor range drops into the .40-.42 values. While the decrease in batch ASW score is not extreme, there is some small drop in the scores as we scale to more donors. This is a small indication that batch effects increase as a result of the increasing numbers of donors. We can also observe the iLISI and cLISI metrics: similar to batch ASW, iLISI begins relatively high at 1.53 and decreases as we go into the 50-100 donor range. cLISI fluctuates from 1.36 to 1.43, but does not exhibit a pattern as the number of donors increases.

We can similarly observe the metrics for the Ren dataset in Table 3.2. We see that cell type ASW fluctuates from .53-.55 while batch ASW fluctuates from .45 to .50. There is a drop from 10 to 20 donors, but the rest of the values are small fluctuation. Unlike the Stephenson metrics, batch silhouette width for this dataset does not exhibit a noticeable pattern; the values are as high around 20-30 donors as they are around 80-90 donors and tend to jump up and down regardless of number of donors. iLISI values paint a similar story: the values jump from 1.31 to 1.41 without much pattern. cLISI values fluctuate without pattern as well. The main takeaway from these metrics is that there is not a noticeable increase in batch effects as a result of scaling to one hundred donors, which is what we initially expected. Instead, scVI's batch correction ability remains the same even with many more batches.

donors	cell type ASW	batch ASW	iLISI	cLISI
10	0.518	0.464	1.530	1.432
20	0.539	0.436	1.310	1.379
30	0.540	0.437	1.280	1.375
40	0.535	0.434	1.259	1.393
50	0.534	0.414	1.142	1.372
60	0.536	0.421	1.165	1.379
70	0.535	0.410	1.114	1.374
80	0.534	0.421	1.124	1.365
90	0.534	0.406	1.129	1.373
100	0.530	0.399	1.094	1.381

Table 3.1: Metrics for Stephenson 21 scVI with batch embedding, measured using batch key = 'site'

3.3 Run Time Evaluation

From our runs of the batch embedding model and vanilla scVI model for the Stephenson dataset, we find that the batch embedding version always trains around 620-640 seconds while the vanilla model varied from 600-900 seconds. Since computational power of the cluster played a role in training time, it is hard to tell whether differences in time are due to the usage of the cluster or to changes in the model. We would need to run further benchmarking on a machine to confirm if the batch embedding model improves the training time. However, we note that in both cases the train time is not significantly long— a variation in train time from 10 to 15 minutes is reasonable for the typical scVI user.

donors	cell type ASW	batch ASW	iLISI	cLISI
10	0.546	0.501	1.307	1.225
20	0.549	0.465	1.324	1.190
30	0.547	0.473	1.302	1.203
40	0.544	0.469	1.409	1.221
50	0.545	0.452	1.272	1.226
60	0.539	0.457	1.304	1.189
70	0.534	0.446	1.322	1.204
80	0.549	0.475	1.392	1.205
90	0.542	0.463	1.366	1.203
100	0.553	0.470	1.308	1.199

Table 3.2: Metrics for Ren 21 scVI with batch embedding, measured using batch key = 'sample_type'

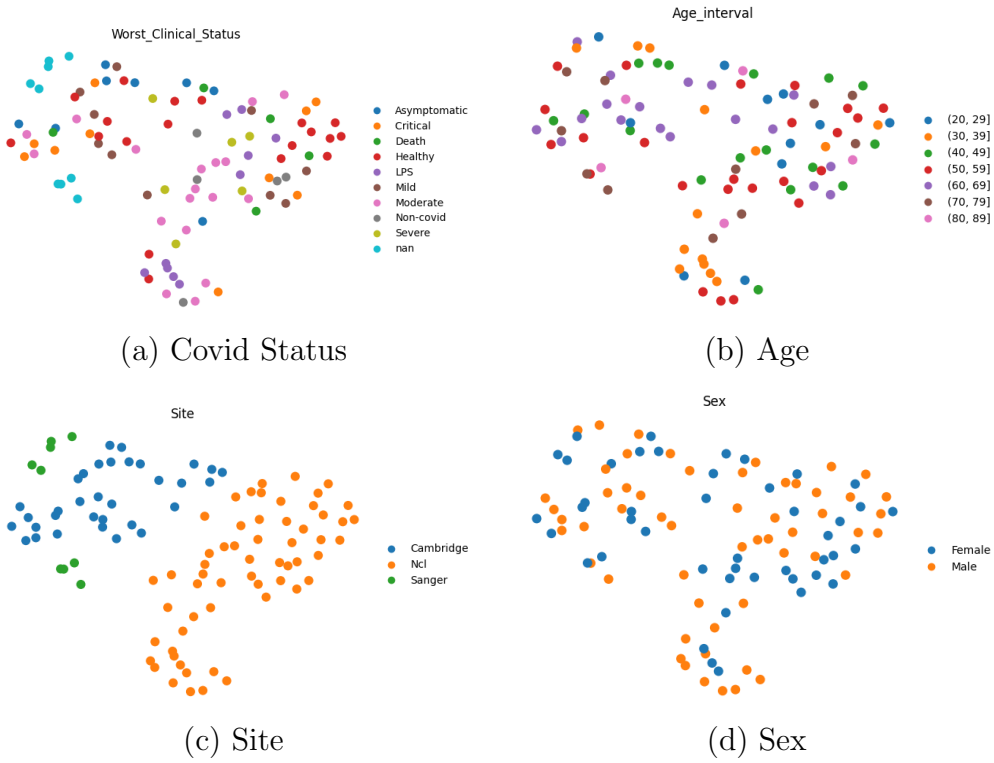
3.4 Interpretation of the Batch Embedding Space

Latent Space Visualization

Beyond memory improvements, the batch embedding gives us the opportunity to interpret the latent space formed by the donors and look for associations in covariate variables. As an example, the batch embedding UMAP for the Stephenson dataset with 100 donors is shown in figure 3.13. To plot this graph, we took the $n \times 5$ matrix outputted by the inference step during training and projected the values onto a UMAP, coloring each donor by its COVID status, age, site, and sex. We see that the UMAPs for age and sex are well mixed, as expected, while site is the covariate that has the largest stratification. In particular, we notice that the Ncl and Cambridge sites are in completely separate clusters, and the Sanger site is also not a part of these clusters. Looking at COVID status, the variety in possible statuses makes it difficult to notice specific clusters for any one status. However, we can again note that 'nan' values are separated from the rest, and correspond to the same donors as the Sanger site.

Analyzing the Ren dataset batch embedding in figure 3.14, we see that donors are clustered by batch and city. Similar to our earlier analysis of the overall latent space, we see that sample type is a major covariate contributing to technical variation: the fresh and frozen PBMC samples are not well mixed in the batch embedding. Severity appears more well mixed in this case.

Overall, we note that this qualitative batch embedding analysis is useful for understanding donor stratification in large single-cell studies. It provides a visualization of how the scVI model groups batches of cells and accounts for batch effects during inference. From this analysis, we find that scVI batch correction is generally successful— the batch embedding captures nuisance variation such as batch and site, while ignoring biological variation such



(a) Covid Status (b) Age

(c) Site (d) Sex

Figure 3.13: Batch embedding for Stephenson with 100 donors

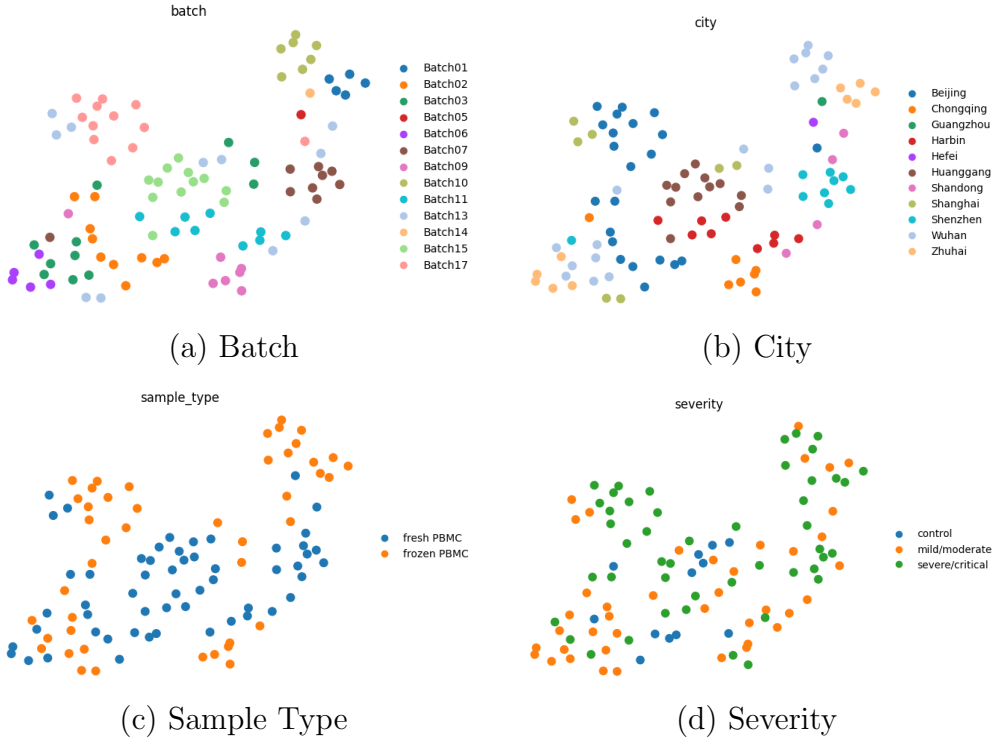
as COVID-19 status.

Metrics

To understand how well the batch embedding captures variation in the donors, we calculate the batch silhouette metric for each batch embedding from 10 donors to 100 donors. From table 3.3, we see that ASW initially increases from 10-30 donors, but then fluctuates for the remaining portion of the experiment. The fluctuation in ASW scores does not point to a clear pattern as number of donors increases. We note that this may be because the clustering of the donors is already well separated in the batch embedding even in the smaller donor subsamples.

3.5 Sensitivity to Batch in Gene Expression Data

In this section, we want to see how gene expression is actually being corrected, and which genes tend to be more corrected than others. We will use the procedure described in algorithm 1 to obtain a relative "sensitivity to batch" ranking of the genes. We then use the



(a) Batch (b) City

(c) Sample Type (d) Severity

Figure 3.14: Batch embedding for Ren with 100 donors

donors	ASW
10	0.478
20	0.484
30	0.531
40	0.526
50	0.539
60	0.539
70	0.534
80	0.524
90	0.528
100	0.548

Table 3.3: Stephenson batch embedding silhouette width

consensus ranking algorithm described in algorithm 2 to compile the overall rank of each gene from ten trial runs of the S2B algorithm. By finding the mean and standard deviation of each rank, we can see how stable the S2B scores are for different groups of donors. This

will tell us if S2B is really a property of the gene or whether what we're seeing is just random fluctuation. Once we have these aggregate S2B values, we will look at what characterizes these genes, and see if any patterns appear for "batchy" genes. More specifically, we can look at mean gene expression values, expression variance, gene sequence length, and biological interpretation of individual genes.

To summarize, the overall goal of this section is to expose how batch correction works in the scVI model in the context of gene expression. By computing this "sensitivity to batch" value, we can find which genes are more batch corrected than others in the scVI model and understand reasons underlying the need for batch correction. For scVI users, this could be a useful metric alongside benchmarking metrics to understand the gene expression outputs of the scVI model.

Algorithm 1 Sensitivity to Batch Algorithm

procedure S2B(S) ▷ S is a subset of donors
  for $g \in G$ **do**
    for $c \in C$ **do**
      $s \leftarrow$`batch of cell c`
      p[c] = $\mathbf{E}[P(g_c|s)]$ ▷ Generate using scVI decoder
    end for
    $R_g \leftarrow$`std(p)/mean(p)`
  end for
  return R ▷ R contains the relative s2b rank of every gene g
end procedure

Algorithm 2 Consensus Ranking Algorithm

Input: n, number of trials, and d, number of donors in each subset S

procedure CONSENSUS(n, d)
  for $i \in [1, n]$ **do**
    $S \leftarrow$ RANDOM(d)
    $R_i \leftarrow$S2B(S)
  end for
  `sort R by mean(R)`
  return `top 10 genes in R`
end procedure

S2B genes after scVI batch correction

Table 3.4 displays the genes with the highest sensitivity to batch (e.g., the genes that were most batch corrected in the Stephenson dataset). Next to the name of each gene, we have

Gene	Mean Rank	Std of Rank
MTRNR2L8	1190.9	8.70
MYOM2	1185.6	10.73
GP1BB	1183.5	18.64
NAPSA	1173.0	29.66
IL1R2	1154.6	18.24
AL133415.1	1146.7	52.48
AC007952.4	1143.2	56.09
TRAV30	1140.7	70.91
IGHV1OR15-1	1137.9	31.87
CH25H	1127.6	49.94

Table 3.4: Relative rank of highest "sensitive to batch" genes

the mean rank and standard deviation of rank. In figure 3.15, we can see a visualization of the consensus ranking genes: the genes with the highest mean rank, or the most batchy genes, also had the lowest standard deviation in rank. This indicates the highest S2B genes remained consistent across different samples of donors. In this case, the highest S2B genes are the points on the rightmost bottom corner of the plot. We can also notice that for genes without a very high or low rank, the standard deviation is quite high; there is little agreement across different samples because all genes in the middle had similar S2B values.

We can now look into the characteristics of the genes that are sensitive to batch. From table 3.5, we note that the expression mean and variance for gene MTRNR2L8 are high, but other genes do not exhibit particularly high or low levels of gene expression (mean expression is 0.015, mean variance is 0.076).

We can further interpret gene expression through stratification by site and clinical status, two covariates that we determined were large sources of technical and biological variance. From figure 3.16, we can see four of the genes that exhibited the biggest difference across sites. MTRNR2L8, AL133415.1, and AC007952.4 all had much greater expression in samples from the Cambridge site than the other two sites. IL1R2 had much greater expression in the Ncl site than the other sites. From figure 3.17, we see genes stratified by clinical COVID-19 status.

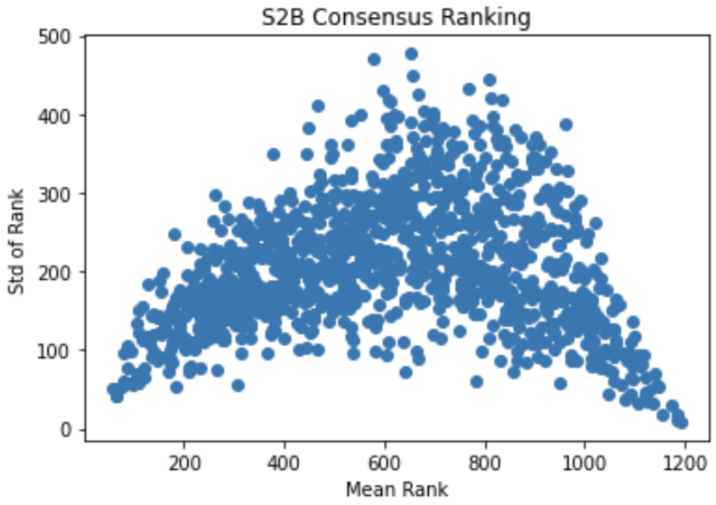
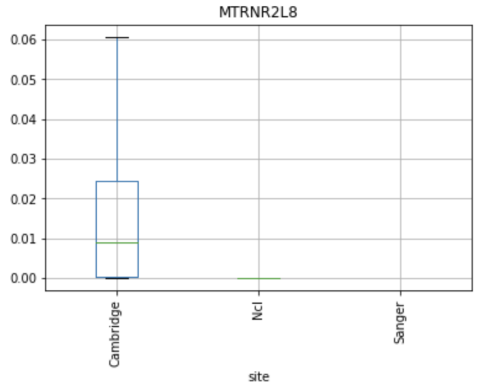


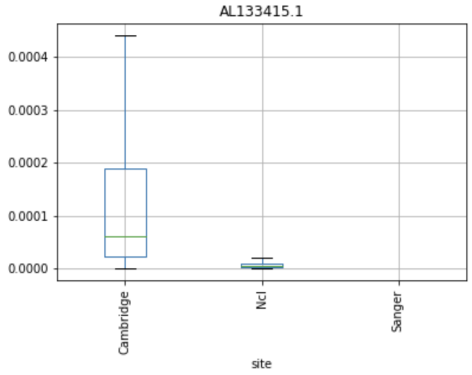
Figure 3.15: Consensus ranking of genes

Gene	Expression mean	Expression variance
MTRNR2L8	1.860	157.5
MYOM2	0.074	0.263
GP1BB	0.011	0.072
NAPSA	0.041	0.264
IL1R2	0.060	0.325
AL133415.1	0.028	0.073
AC007952.4	0.114	0.573
TRAV30	0.017	0.072
IGHV1OR15-1	0.015	6.101
CH25H	0.006	0.015

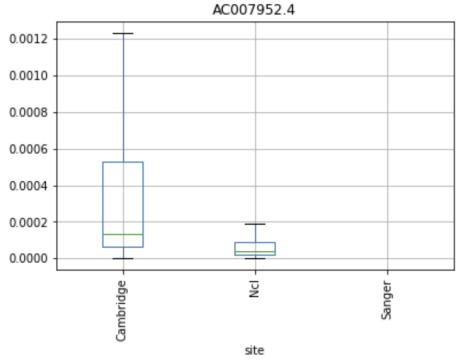
Table 3.5: Gene expression of highest "sensitive to batch" genes



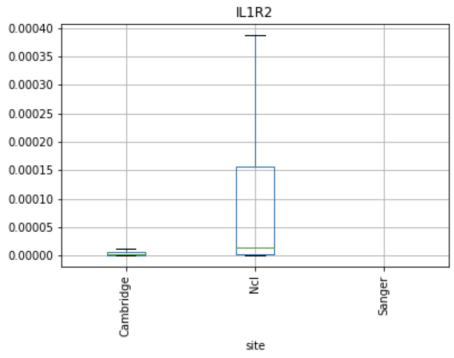
(a) MTRNR2L8



(b) AL133415.1

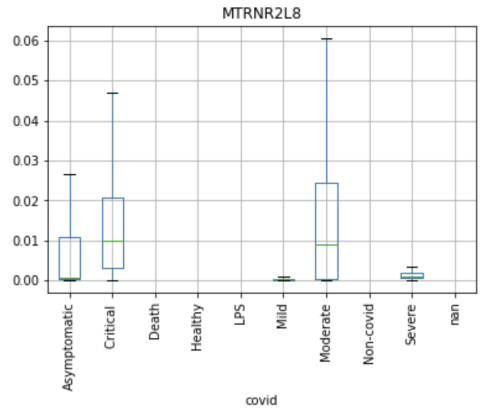


(c) AC007952.4

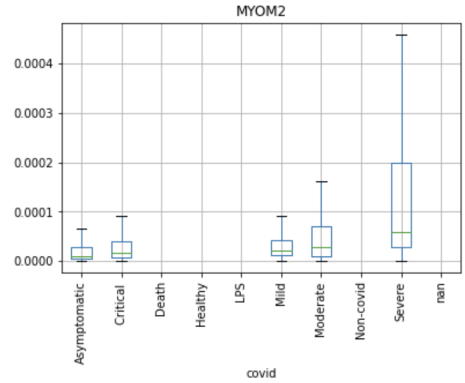


(d) IL1R2

Figure 3.16: S2B gene expression grouped by site



(a) MTRNR2L8



(b) MYOM2

Figure 3.17: S2B gene expression grouped by clinical status

Chapter 4

Conclusion

4.1 Summary

In this thesis, we demonstrate that scVI does not degrade in performance when scaling up to datasets with 100+ donors. Barring a small decrease in ability to remove batch effects, scVI continues to capture biological variation and cluster the data well. By adding the batch embedding to the model architecture, we can identify donor-level phenotypes and find out which covariates contribute most to batch effects in the data. This allows us to understand how the scVI model works under the hood by revealing how scVI encodes batches (either donors or other covariates) and corrects for batch-related variation. Finally, with the addition of the "sensitivity to batch" score for genes, we can identify the genes that are most batch-corrected and investigate patterns that may cause a gene to become more batch sensitive. In addition, this provides useful functionality for scVI users who may want to take note of or remove the most "batchy" genes from their gene expression analysis.

4.2 Code Contributions

- Batch embedding: The batch embedding option is implemented as part of the scvi-tools codebase and can be enabled during training with the parameter `use_batch_embedding` set to `True`. In addition, functions were added for the user to retrieve the batch embedding and individual donor encodings from the model.
- Metrics: All metrics used during benchmarking are implemented as util functions within scvi-tools and can be used for any trained model here. We also created an experiment pipeline for large donor studies.
- Sensitivity to batch: We implemented the function `get_batchy_genes()`, which returns genes that are most sensitive to batch along with their sensitivity score.

- Large donor analysis tutorial: Finally, here is a jupyter notebook tutorial demonstrating usage of the above features— batch embedding, metrics, "sensitivity to batch" gene analysis— on one example dataset.

Bibliography

[1] Dominic Grun, Anna Lyubimova, Lennart Kester, Kay Wiebrands, et al. Single-cell messenger rna sequencing reveals rare intestinal cell types. *Nature*, 525(7568):251–255, 2015.

[2] Brian Hie, Bryan Bryson, and Bonnie Berger. Efficient integration of heterogeneous single-cell transcriptomes using scanorama. *Nature biotechnology*, 37(6):685–691, 2019.

[3] Allon M Klein, David A Weitz, and Marc W Kirschner. Droplet barcoding for single-cell transcriptomics applied to embryonic stem cells. *Cell*, 161:1187–1201, 2015.

[4] Ilya Korsunsky, Nghia Millard, Jean Fan, Kamil Slowikowski, Fan Zhang, Kevin Wei, Yuriy Baglaenko, Michael Brenner, Po-ru Loh, and Soumya Raychaudhuri. Fast, sensitive and accurate integration of single-cell data with harmony. *Nature methods*, 16(12):1289–1296, 2019.

[5] D.A. Lawson, K. Kessenbrock, R.T. Davis, N. Pervolarakis, and Z. Werb. Tumour heterogeneity and metastasis at single-cell resolution. *Nature Cell Biology*, 20(12):1349, 2018.

[6] R. Lopez, Cole Regier, J., M.B., et al. Deep generative modeling for single-cell transcriptomics. *Nature Methods*, 15:1053–1058, 2018.

[7] M.D. Luecken, M. Büttner, K. Chaichoompu, et al. Benchmarking atlas-level data integration in single-cell genomics. *Nature Methods*, 19:41–50, 2022.

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

[9] Krzysztof Polański, Matthew D Young, Zhichao Miao, Kerstin B Meyer, Sarah A Teichmann, and Jong-Eun Park. Bbknn: fast batch alignment of single cell transcriptomes. *Bioinformatics*, 36(3):964–965, 2020.

[10] William M Rand. Objective criteria for the evaluation of clustering methods. *Journal of the American Statistical association*, 66(336):846–850, 1971.

[11] Aviv Regev, Sarah A. Teichmann, Eric S. Lander, Ido Amit, Christophe Benoist, et al. The human cell atlas. *bioRxiv*, 2017.

[12] X. Ren, W. Wen, X. Fan, et al. Covid-19 immune features revealed by a large-scale single-cell transcriptome atlas. *Cell*, 184(7):1895–1913, 2021.

[13] Peter J Rousseeuw. Silhouettes: a graphical aid to the interpretation and validation of cluster analysis. *Journal of computational and applied mathematics*, 20:53–65, 1987.

[14] E. Stephenson, G. Reynolds, R.A. Botting, et al. Single-cell multi-omics analysis of the immune response in covid-19. *Nature Medicine*, 27:904–916, 2021.

[15] Michael J. T. Stubbington, Orit Rozenblatt-Rosen, Aviv Regev, and Sarah A. Teichmann. Single-cell transcriptomics to explore the immune system in health and disease. *Science*, 358(6359):58–63, 2017.

[16] H.T.N. Tran, K.S. Ang, M. Chevrier, et al. A benchmark of batch-effect correction methods for single-cell rna sequencing data. *Genome Biology*, 21(12), 2020.

[17] A. Wagner, A. Regev, and N. Yosef. Revealing the vectors of cellular identity with single-cell genomics. *Nature Biotechnology*, 34:1145–1160, 2016.

[18] F.A. Wolf, P. Angerer, and F.J. Theis. Scanpy: large-scale single-cell gene expression data analysis. *Genome Biology*, 19(15), 2018.

[19] Nir Yosef, Michael Jordan, and Romain Lopez. A deep generative model for gene expression profiles from single-cell rna sequencing. Master's thesis, EECS Department, University of California, Berkeley, May 2018.