A brief introduction to preprocessing and batch correction for single-cell sequencing data

Johnathan Lo

8/8/20

Contents

1 Introduction and motivation

1.1 What is single cell sequencing?

Single-cell sequencing is a technology that allows the sequencing of genetic information from individual cells. The targeted genetic information is usually mRNA transcripts, although DNA can be sequenced using single-cell technology as well (Luecken u. Theis, 2019). Single-cell mRNA sequencing helps isolate the gene expression of individual cells, which paints a more nuanced and finer-grained picture of gene expression across a given collection of cells.

This is as opposed to bulk mRNA sequencing, which can only inform us of average levels of expression across an entire sample (Wolf u. a., 2018).The additional information garnered from sampling gene expression from individual cells allows us to deduce additional parameters relevant to gene expression, such as variance, median, and empirical distribution of expression levels, and define groups of cells with different expression distributions. While traditional bulk sequencing data may have expression signals obscured by extrema or outliers, single-cell sequencing allows patterns to become more defined and significant.

1.2 What does single cell data look like?

Single-cell sequencing data is usually organized as a matrix where the rows are genes and columns are cells. The values populating the matrix are measures of gene expression levels, or counts, usually as a raw number of transcripts (Luecken u. Theis, 2019). In practice, the units are not important so long as the scale given is universally applied across cells. From the standpoint of statistical analysis, we consider the cells as observations and the genes as features.

1.3 How and why do we apply statistics to single cell sequencing data?

After sequencing, statistical analysis is applied to single-cell sequencing data to extract useful information. Analysis of single-cell sequencing data usually takes place in three general stages: preprocessing, network construction, and further downstream analysis. Single-cell sequencing studies are useful for informing us about how certain treatments affect gene expression, or identifying aberrant expression patterns in clinical cases. Applications for single-cell sequencing and its statistical analysis range from developmental

biology, where we can identify gene regulatory networks at each stage of development, to cancer genomics and infectious disease, where we can pinpoint our body's immune response to systemic stress and identify weaknesses (Luecken u. Theis, 2019).

2 Preprocessing

2.1 Motivation

Raw sequencing data must be processed before any sort of analysis can be conducted. Raw data from the sequencing machine is not only noisy, but difficult to manipulate. Most sequencing platforms come with proprietary software for processing raw data, such as Cell Ranger, indrops, SEQC, and zUMIs.

However, although this initial processing removes some noise and organizes data into an easily manipulated format, additional processing must still occur to enable GRN construction and other downstream analysis (Wolf u. a., 2018). This step is known as preprocessing. In this step, the aim is to remove all sources of variation in the data that are not relevant to the targeted analysis, i.e. variation due to technological covariates, environmental covariates, or irrelevant biological covariates.

2.2 Overview

Packages for preprocessing are available on a variety of platforms, including R and Python. For our analyses, we will be using the scanPy package in Python.

Preprocessing typically takes the data through 3 phases: measured, corrected, and reduced. Across these phases, there are overall 7 distinct steps: loading, exploration, QC, normalization, correction, feature selection, and dimensionality reduction (Luecken u. Theis, 2019).

Loading data is the first step to analyzing sequencing data, and it is not always as easy as it sounds. Different data formats will require different techniques, packages, or software to load it properly into a workspace. We will be loading MEX-formatted data into our pipeline using methods built into scanPy; however, other data formats may or may not be so amenable. In scanPy, data is loaded as AnnData objects, which are observations x variables matrices with annotations built into the margins. Users are advised to keep in mind objected oriented programming principles when manipulating AnnData objects (Wolf u. a., 2018).

Once data is loaded, preliminary data exploration can take place to get a sense

of the scale and variability of the data. Users can visualize the count depth by genes, by cells, or both. From this visualization, users can obtain a heuristic strategy for the proceeding steps.

The next step, QC, uses thresholding to eliminate observations that may be contaminated, distorted, or otherwise faulty. These types of faulty observations include dead-/lysed cells or doublets. Dead/lysed cells are detected by low overall count depth with a high proportion of mitochondrial transcripts, while doublets are detected by an excess of count depth (Luecken u. Theis, 2019).

After eliminating faulty observations, we assume the remaining observations are valid and try to correct them to remove unwanted noise. Normalization is undertaken to help balance out gene expression across cells. Various technological and sampling covariates can lead to discrepancies in count depths between cells when there is none in reality, and this can lead to distortions in downstream analyses where cells are clustered by specific gene expression. The assumption behind normalization is that there is some fundamental distribution of total count depth across cells that should generally be adhered to. The most common and simplest form of normalization is CPM, or counts per million, where all cells are normalized such that they have the same total count depth. This is, of course, an assumption that may or may not be biologically valid. Other methods use parametric modelling of total count depth that allows for variation in total count depth between cells and scales counts for individual cells by an inference algorithm (Luecken u. Theis, 2019).

Correction takes the principle behind normalization and extends it further to adjust or scale gene count values for each cell independently (as opposed to scaling across the entire observation as in normalization). While normalization deals with removing the effects of technological covariates across cells, correction takes a finer-grained approach and removes technological effects across genes as well. For example, it is easy to imagine that the sampling of certain genes may be biased by the length or content of their transcripts. Technical dropouts caused by instrument error should also be accounted for. Correction also aims to remove unwanted biological effects, such as cell cycle effects (if these are not relevant to the treatment being studied) (Luecken u. Theis, 2019). Both technological and biological correction are conducted by regressing against the unwanted covariates and retaining the residuals, and unknown values are imputed based on parameterized distributions for count data. Finally, and most importantly for this article, correction aims to remove batch effects and data integration effects, which is variation resulting from combining data that were obtained under non-identical conditions.

The final two steps of preprocessing are feature selection and dimensionality reduction. Both of these steps are aimed at enhancing computational efficiency for downstream analyses. In feature selection, we remove uninteresting genes and retain interesting ones. The metric for determining what is and is not interesting can vary and depend on the analysis at hand but the most common method is to select only the most highly variable genes (HVGs) using a metric like mean-variance ratio. After these HVGs are selected, dimensionality reduction takes place by taking linear or nonlinear combinations of the remaining features to produce components that explain the greatest variance (Luecken u. Theis, 2019).

3 Batch correction/data integration

3.1 Motivation

With the vast influx of sequencing data in recent years, it has become increasingly important to devise methods for leveraging such quantity of data. The main way of doing so is by combining data from different sources into a single dataset. These data may be from separate runs within the same experiment or entirely separate experiments. In either case, such merged datasets must tackle a new set of challenges in addition to the issues already dealt with in other areas of preprocessing (Luecken u. Theis, 2019). By merging data from disparate sources, these datasets introduce a new source of variation, called 'batch effects', which must be eliminated to avoid hindering subsequent downstream analyses. Such batch effects may cause data to cluster by batches, which is uninformative, rather than by cell type or gene expression, which is typically the target of such studies. For example, in Figure 1 (Korsunsky u. a., 2019), we observe a combined dataset composed of 3 batches - one derived from pure t293 cells, one derived from pure jurkat cells, and one derived from a a 50/50 combination of both. Because of technological and environmental differences in how these data were collected, there is a noticeable batch effect that causes PCA to cluster cells into 3 groups rather than the 2 groups (t293 and jurkat) that are actually present.

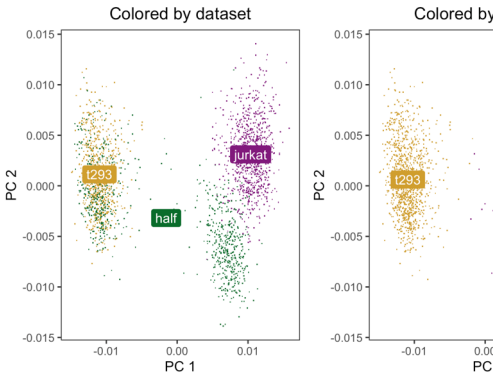

Figure 1: Fig 1 - 3 datasets composed of two cell lines - t293 and jurkat. Note how clustering with PCA reveals batch effects.

3.2 Methods

There are many available methods for applying batch correction to a dataset. In this article, we will focus on understanding, applying, and evaluating three of them - Harmony, UnionCom, and MMD-ResNet.

Harmony is a linear batch correction method that consists of 2 iterated steps. First, in PCA-space, a centroid is defined for each batch within each cluster. For example, in a dataset with 3 batches, each cluster (as defined by a neighborhood of size k) will have 3 centroids unless one of the batches is missing from the cluster altogether. Then, these centroids are incrementally moved closer to one another by scaling the values of cluster members by batch (Korsunsky u. a., 2019). This method is fast due to using purely linear methods; however, it relies on the assumption that there is independence between cluster and batch assignment, which is not always viable. Also, in the rare case that clusters are largely homogenous for a single batch (for example, where the batches have little or no correspondence in their features and so cluster by batch) the neighborhood size for clusters will have to become so large as to become meaningless, as every cluster less than the full dataset will be completely homogenous for a batch.

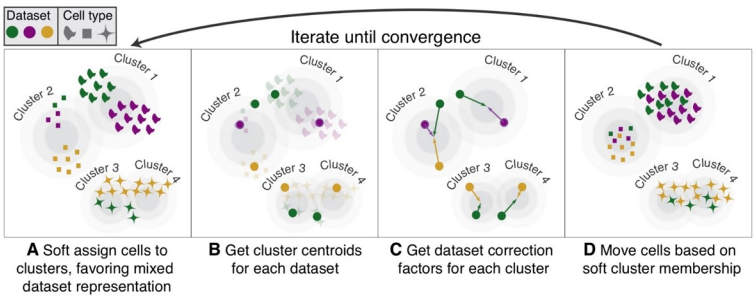

Figure 2: Fig 2 - illustrating Harmony process

Unioncom is a nonlinear method for batch correction that is expressly designed for data integration (i.e. merging data from different experiments rather than simply different batches of the same experiment). Importantly, UnionCom allows for integration of data where the features do not correspond. The UnionCom approach begins by constructing a distance matrix for each dataset and using an MNN-derived algorithm to align these distance matrices to each other, essentially mapping the feature space of each dataset to a common feature space (Cao u. a., 2020). UnionCom is not quite as fast as Harmony, but makes up for it with the ability to integrate data without correspondence. However, UnionCom rests on the assumption that the datasets being integrated are fundamentally

from the same biological data generating process and contain the same cell types in the same global structure. If one dataset contains cell types that the others do not, spurious alignment can occur, where the resulting common feature space does not in fact preserve the real cell types of the dataset with more cell types.

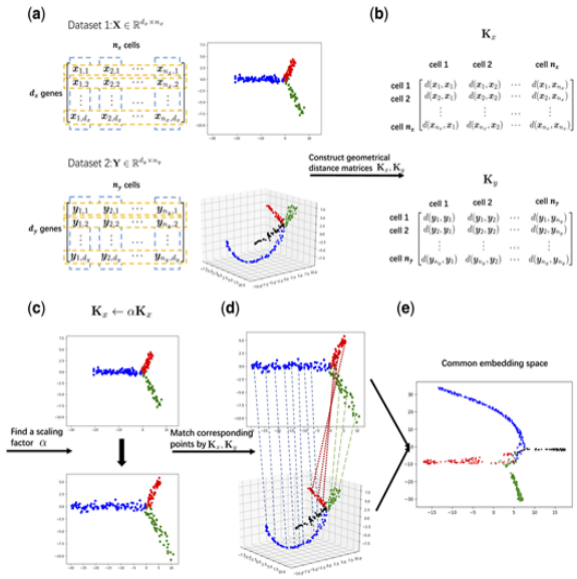

Figure 3: Fig 3 - the UnionCom process

The final method to be tested is MMD-ResNet, which is a deep learning approach that uses residual neural networks to minimize the maximum mean discrepancy of the batches with respect to PCA space. The maximum mean discrepancy (MMD) is a measure of the dissimilarity between two probability distributions, and the goal of MMD-ResNet is to integrate datasets by attempting to match the distribution of each primary component between batches. Only two batches can be integrated at once, and the matching is done by setting the distributions of one of the batches as a target, and letting the other batch be subject to scaling and adjustment to bring its distribution closer to the first. Ultimately, the resulting integration will have

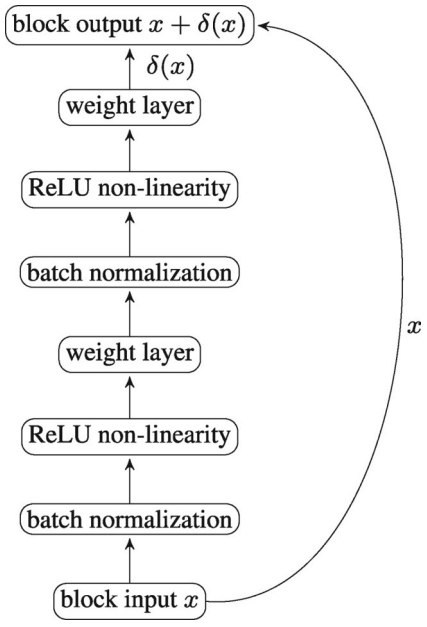

Figure 4: Fig 4 - the MMD-ResNet process

maximized the likelihood that the data from the second dataset comes from the first U. Shaham (2017). As a deep-learning method, MMD-ResNet has the potential to be the most computationally costly, and relies on similar assumptions as UnionCom, but instead of making assumptions about global structure under clustering algorithms, it makes assumptions about the underlying true distribution of data in PCA space.

All three methods vary in their application, speed, and accuracy, and when choosing a method to implement, it is vital to take into consideration the characteristics of the dataset. Most importantly, the three methods differ fundamentally in their assumptions, and so care should be taken when deciding which assumption best fits the circumstances.

3.3 Evaluating batch correction

To compare the effectiveness of each of these three methods, evaluation methods have to be considered. Evaluation methods can be visual or quantitative. In this section, we will consider 3 visualization methods, PCA, t-SNE, and UMAP, and 4 quantitative metrics, ASW, ARI, LISI, and kBET.

PCA, or primary component analysis, is perhaps the most well known of all the methods here. It iteratively finds linear combinations of the features that are orthogonal to one another and account for the most variance. These are termed primary components and are typically ordered by variance explained (Pearson, 1901). Thus, by visualizing data in the first 2 or 3 PCs, we can hopefully observe enough of the variance in the data to make a judgment on whether or not batch correction occurred successfully. If clusters in PCA space tend to segregate by batches, we can conclude that batch correction was poor.

t-SNE, or t-distributed stochastic neighbor embedding, is a graph-based method for dimensionality reduction. It first constructs a similarity matrix in high dimensions for the dataset where the pairwise similarity for each observation is recorded. Then, as data is reduced to lower and lower dimensions, it tries to maintain keep the similarity matrix as close to the original as possible. In this way, the local structure of clusters remains intact when plotting data in low dimensional t-SNE space (van der Maaten u. Hinton, 2008). Neighbors in high dimensions will remain neighbors in low dimensions. However, the global structure of clusters (i.e. the distance between clusters themselves) is not maintained in this method. It uses a cost function based on KL-divergence, which is a measure

of the difference between distributions.

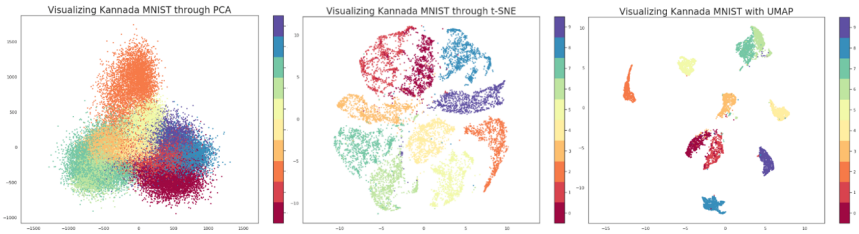

Figure 5: Fig 5 - illustrating the differences between dimn reduction methods - credit: Parul Pandey

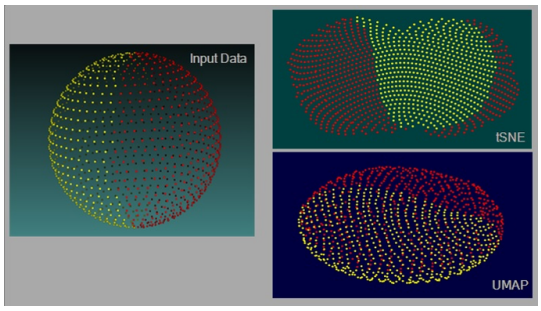

Figure 6: Fig 6 - Local vs global structure in t-SNE and UMAP - credit: James X. Li

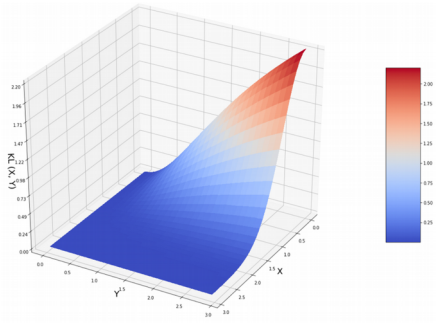

Figure 7: Fig 7 - the KL divergence function, with X axis as distance in high dim'n, Y-axis as distance in low dim'n - credit: N. Oskolkov

UMAP, or uniform manifold approximation and projection, is similar to t-SNE in that it is also a graph-based method. However, rather than deduce an embedding based on similarity between observations, UMAP takes the observations as-is and constructs a variable distance metric to stitch together a complete manifold (wherein the variable distance metric essentially helps impute the embedding over space where observations do

not exist). Using this to bring the data into lower dimensions, UMAP is able to preserve both global and local structure. The cost function for UMAP is based on cross entropy (McInnes u. a., 2018).

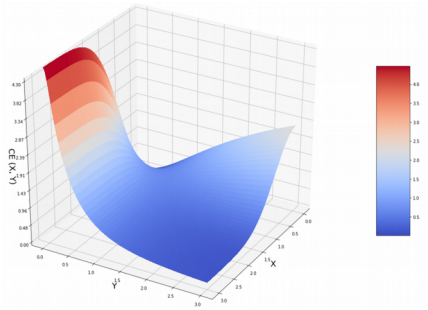

Figure 8: Fig 8 - the CE function, as before - credit: N. Oskolkov

As far as quantitative metrics go, a basic and essential metric is kBET, which stands for k-nearest neighbor batch effect test. This measures batch mixing on a local level by taking the k-nearest neighbors of a random data point (a random neighborhood) and counting the number of neighbors from each batch. It assumes that the batches are from repetitions of the same experiment and that major differences are only in technical covariates, or in other words, that the global proportion of batches should be reflected in the local proportion of batches in each cluster. It performs a chi-squared-based test on each random neighborhood to determine if difference from overall batch distribution is significant at some level, and low rejection rates indicate good mixing (Buttner u. a., 2019). Of course, however, it quite limiting to assume that each cluster should have equal proportions of each batch and furthermore, it is not able to distinguish if each cluster has local asymmetry with respect to batches.

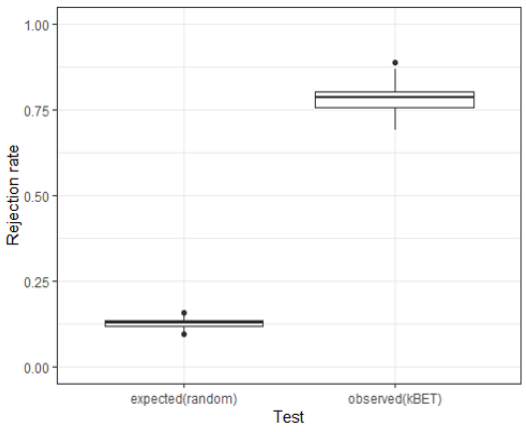

Figure 9: Fig 9 -kBET example

LISI, or local inverse Simpson's index, attempts to address the issues with kBET by taking random neighborhoods and describing how many observations are expected to

be sampled from the neighborhood before two are drawn from the same batch. Rather than reject or fail to reject neighborhoods based on some null distribution of batches for each cluster, LISI instead returns a landscape of batch mixing across PCA space, which allows the user to interpret whether or not adequate integration has occurred (Korsunsky u. a., 2019).

ASW, or average silhouette width, tries to determine how well mixed observations are within each cluster by characterizing the local structure of each cluster. It constructs "silhouettes" of each cluster that differentiate observations within the cluster based on how close they are to each other. The average width of a silhouette shows how closely clustered different categories of data are. By performing ASW on batches, we can see whether batches tend to cluster together or apart. ASW ranges from -1 to 1 inclusive, where a high value indicates poor mixing (or good separation), a value close to 0 indicates good mixing, and low (close to -1) values indicate incorrect labeling of clusters (i.e. the observations are closer to another cluster than their own) (Rousseeuw, 1987).

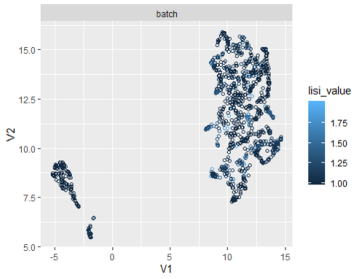

Figure 10: Fig 10 -LISI example

Finally, we have ARI, which stands for adjusted rand index. This is similar to ASW in that it tries to determine whether clusters segregate by batches or other covariates. Rather than use silhouettes, however, ARI simply assumes that clusters should have a distribution of batches similar to the global distribution (like kBET). Data is "partitioned" in two ways, and the partitions are compared to each other to see if membership in one partition predicts membership in the other. For the purposes of evaluating batch correction, these partitions are simply cluster membership and batch membership. Like ASW, it ranges from -1 to 1 and a value close to 0 indicates good mixing, or in this case, random "partitioning" (Hubert u. Arabie, 1985).

4 Case Studies

In the following section, we test the efficacy of the above batch correction methods on some edge cases. These cases are not representative of the majority of situations faced by the typical gene expression researcher, but do represent potential hurdles that must be overcome in future methods.

4.1 Case Study 1

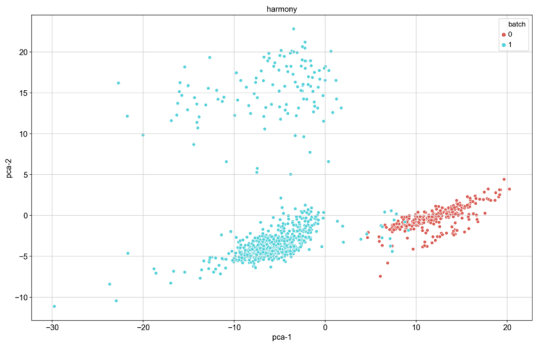

Figure 11: Fig 11 - UMAP with results from Harmony on nc-dataset - predictably bad

In this case study, we have 2 batches of PBMC data that we want to combine into a single dataset. As these samples were obtained on different occasions under different circumstances, they represent a typical case of data integration, rather than mere batch correction. Fortunately, the datasets have complete correlation in their feature spaces, so no guesswork on that part is required. However, to take it up another notch, the data was also combined in a fashion where the correspondence between features was artificially removed. Both the dataset with corresponding features (control) and dataset with non corresponding features (nc) were subjected to the same preprocessing pipeline followed by batch correction with Harmony, UnionCom, or MMD-ResNet. The success of the integration was evaluated using each of the 7

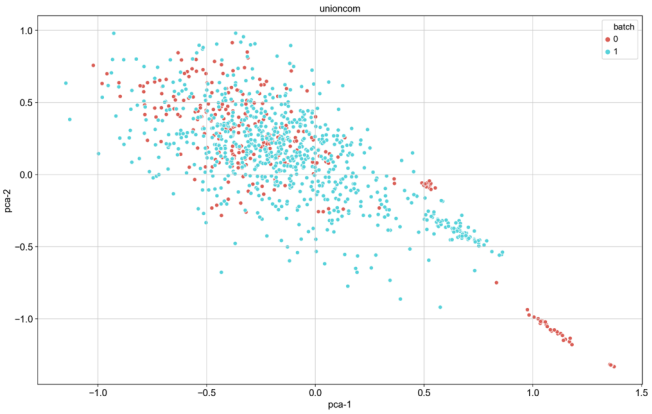

Figure 12: Fig 12 - UMAP with results from UnionCom on nc-dataset - quite good as expected

methods outlined above.

Results largely conformed to what we expected, given the assumptions and algorithms behind each of the methods. Naturally, all three methods had a relatively easy time integrating the control dataset. Harmony, for example, had an extremely difficult time aligning the data without correspondence; this is because Harmony depends on a shared feature space where it can choose between clusters that have differential mixing of batches - without correspondence, there is hardly any mixing at all, no matter which PC is chosen! With UnionCom, it is a different story, and integration performs quite well. The contrast between the two can be seen in figures 11 and 12, as well as in the results from quantitative metrics. It was harder to predict how well MMD-ResNet would do with non-corresponding data. As it turns out, MMD-ResNet performed nearly as poorly as Harmony by all metrics. The reason for this I am not sure of, but it may be related to adjusting tuning parameters.

4.2 Case Study 2

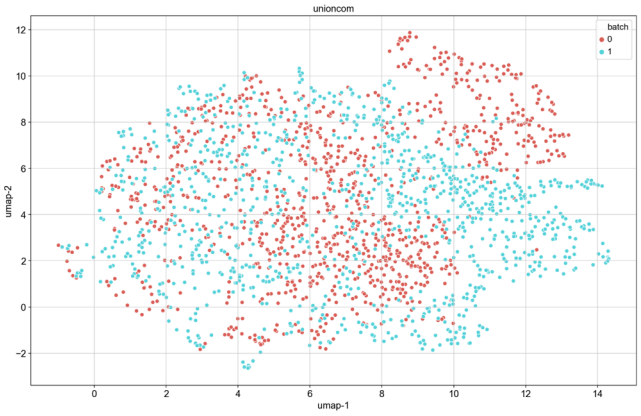

Figure 13: Fig 13 - UMAP with results from UnionCom

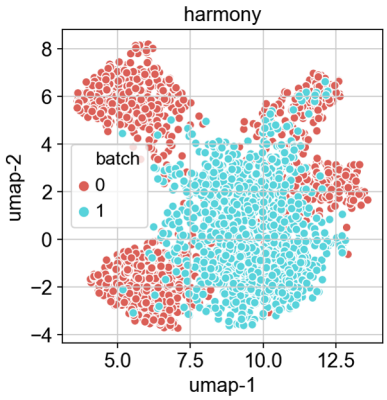

Figure 14: Fig 14 - UMAP with results from Harmony

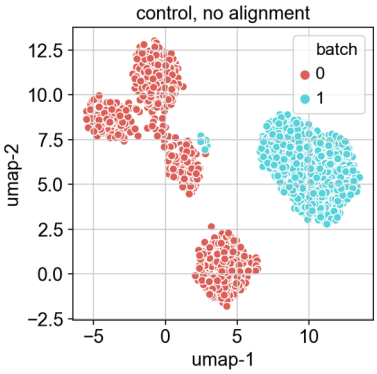

Figure 15: Fig 15 - UMAP of original, unintegrated data

In this second case study, we examine how well these integration methods preserve the identity of individual observations, i.e. their position in local and global structure. This is an important aspect of integration to analyze, as we want to know if batch correction methods end up distorting the data so much so that the original information is completely lost or irretrievably distorted. To accomplish this, we take two datasets with identical observations that have been measured using different technologies (RNA-seq vs ATAC-seq). The observations across these two datasets have perfect correspondence. To align these datasets properly, we want to not only mix batches well, but minimize the distance between identical observations, i.e. an observation from the RNA-seq dataset should wind up as close as possible to its twin in the ATAC-seq dataset. The difficulty of this is compounded by the fundamental technological difference between ATAC-seq data and RNA-seq; however, we hypothesize that given identical cell types, the data should cluster similarly across both datasets.

In practice, however, we find that the two datasets do not even remotely cluster

together, as can be observed in Figure 15. Without at least some batch mixing, Harmony does a very poor job of integrating the datasets, as does MMD-ResNet. Only UnionCom does an adequate job of batch mixing. However, when we look at the distance between points, as shown in figure 16, we see that there does not seem to be evidence, at least from a heuristic perspective, that UnionCom is able to preserve the identity of cells. Analysis of the quantitative metrics affirms that indeed, UnionCom performs best out of all three methods when it comes to integrating data that clusters by batches, with Harmony in second and MMD-ResNet in third.

The poor results of this second case study could be for a number of reasons. Most likely, the ATAC-seq data is fundamentally different from RNA-seq data; that is, observations truly cluster into different types based on the sequencing used, and this clustering is not an artefact of batch effects. Figure 15 lends credence to this idea, as we can see that without any interference or correction whatsoever, the two types of data still cluster distinctly from each other. This is also reasonable from a biological standpoint, as ATAC-seq is meant to access chromatin accessibility, which is not necessarily correlated with gene expression (mRNA-seq) in all circumstances. Chromatin accessibility is necessary, but not sufficient for expression, and this could be a fundamental reason that interfered with this analysis.

For this reason, the circumstances outlined by this case study remain an interesting situation to interrogate with these integration methods. In future work, it may be more suitable to use a different dataset where the batches are both ATAC-seq or mRNA-seq but sequenced with different platforms or technology. This may keep the batches different enough so that alignment is significant, but similar enough so that alignment can occur appropriately without completely distorting the data.

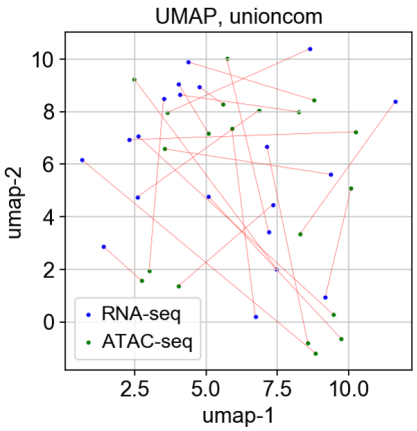

Figure 16: Fig 16 - graph of UnionCom-integrated data with twins connected by lines

References

[3 2009] *Manifold alignment without correspondence.* IJCAI, 2009 (1273-1278)

[Buttner u. a. 2019] Buttner, M. ; Miao, Z. ; Wolf, F. ; Teichmann, S. ; Theis, F.: A test metric for assessing single-cell RNA-seq batch correction. In: *Nat Methods* 16 (2019), Nr. 43-9

[Cao u. a. 2020] Cao, K. ; Bai, X. ; Hong, Y. ; Wan, L.: Unsupervised topological alignment for single-cell multi-omics integration. In: *Bioinformatics* 36 (2020), Nr. i48-i56. – doi:10.1093/bioinformatics/btaa443

[Hubert u. Arabie 1985] Hubert, L. ; Arabie, P.: Comparing partitions. In: *J Classif* 2 (1985), Nr. 193-218

[Korsunsky u. a. 2019] Korsunsky, I. ; Millard, N. ; J. Fan, et a.: Fast, sensitive, and accurate integration of single-cell data with Harmony. In: *Nat Methods* 16 (2019), Nr. 1289-1296. – doi:10.1038/s41592-019-0619-0

[Luecken u. Theis 2019] Luecken, M. ; Theis, F.: Current best practices in single-cell RNA-seq analysis: a tutorial. In: *Mol Syst Biol* 15 (2019), Nr. e8746. – doi:10.15252/msb.20188746

[van der Maaten u. Hinton 2008] Maaten, L. van d. ; Hinton, G.: Visualizing data using t-SNE. In: *J Mac Lrn Res* 9 (2008), Nr. 2579-2605

[McInnes u. a. 2018] McInnes, L. ; Healy, J. ; Melville, J.: UMAP: Uniform Manifold Approximation and Projection for dimension reduction. In: *arXiv* 1802 (2018)

[Pearson 1901] Pearson, K.: On Lines and Planes of Closest Fit to Systems of Points in Space. In: *Philosophical Magazine* 2 (1901), Nr. 559-572. – doi:10.1080/14786440109462720

[Rousseeuw 1987] Rousseeuw, P.: Silhouettes: a graphical aid to the interpretation and validation of cluster analysis. In: *J Comput Appl Math* 20 (1987), Nr. 53-65

[Tran u. a. 2020] Tran, H. ; Ang, K. ; M. Chevrier, et a.: A benchmark of batch-effect correction methods for single-cell RNA sequencing data. In: *Genome Biol* 21 (2020), Nr. 12. – doi:10.1186/s13059-019-1850-9

[U. Shaham 2017] U. Shaham, et a.: Removal of batch effects using distribution-matching residual networks. In: *Bioinformatics* 33 (2017), Nr. 2539-2546. – doi:10.1093/bioinformatics/btx196

[Wolf u. a. 2018] Wolf, F. ; Angerer, P. ; Theis, F.: SCANPY: large-scale single-cell gene expression data analysis. In: *Genome Biol* 19 (2018), Nr. 15. – doi:10.1186/s13059-017-1382-0