Table of Contents

Introduction 1.1

Seurat 1.2

Introduction

This gitbook is used to introduce the detailed algorithm of some important bioinformatics paper, which only include a very brief introduction of the algorithm in the methods.

Normalization

$$X = \begin{bmatrix} x_{11} & \cdots & x_{1m} \\ \vdots & \ddots & \vdots \\ x_{n1} & \cdots & x_{nm} \end{bmatrix}$$

Each row represents a gene, each column represents a single cell. X is the raw read count matrix.

To normalize X, we mean get the count per ten thousands read, and the +1, and log transform it. This is to adjust the influence of sequence depth.

$$x_{ij} = log(\frac{x_{ij}}{\sum_{k=1}^{n} x_{kj}} \cdot 10000 + 1)$$

In the equation above, x_ij on RHS are raw read count.

Standardization

To make each row has mean 0, and standard deviation 1.

$$x_{ij} = \frac{x_{ij} - \bar{x_{i\cdot}}}{\sigma_{i\cdot}}$$

$$\bar{x_{i\cdot}} = \frac{\sum_{k=1}^{m} x_{ik}}{m}$$

$$\sigma_{i\cdot} = \frac{\sum_{k=1}^{m} (x_{ik} - \bar{x_{i\cdot}})^2}{m-1}$$

In the equation above, x_{ij} on RHS are normalized according to step 1.

Feature Selection

Individual Dataset

Based on unnormalized raw read counts.

For each gene i, we can calculate its mean $\bar{x_{i\cdot}}$ and standard deviation $\sigma_{i\cdot}$ for raw read counts, log transform them. We then fit a curve for these two variables across all cells, by calculating a local fitting of polynomials of degree 2 (R function loess, span = 0.3). For a given gene i, we can know from the fitted curve about its expected standard deviation: σ_i

Denote: $z_{ij} = \frac{x_{ij} - \bar{x_{i\cdot}}}{\sigma_i}$.

We can calculate $\sum_{k=1}^{m} max(z_{ik}^2, \sqrt{m}) \cdot 1_{xik \neq 0} + 1_{xik=0} z_{ik}^2)$, which is nearly the ratio of real standard deviation and the expected standard deviation, namely, $\frac{\sigma_{i\cdot}}{\sigma_i}$. By choosing genes with the highest ratio, we can eliminate the influence of mean on the variability of the gene.

Multiple Datasets

1. Features must exist in every dataset (not necessarily HVG) that are going to be integrated.
2. Features must be highly variable in at least one individual dataset.
3. Select features that most frequently exist in HVG of each individual dataset (Let's assume there are 50 datasets, a gene is an HVG for 48 of them, then its frequency is 48.).
4. For features with the same frequency in all datasets, calculate the median of their rank in the HVG list (if a gene does not have a rank for a specific list, then just ignore this list and calculate the median rank in all other list that include this gene), select the lower ones. Identification of anchor correspondence bewteen two datasets

Identification of anchor correspondence bewteen two datasets

Before doing CCA, we need to check features again.

In previous selection of HVGs, we cannot exclude that some features are HVG in one dataset, but are not HVG or even not expressed in other datasets. Therefore, in this step, we need to exclude selected features with zero variance in any of the datasets to be integrated.

Then, let's do CCA.

Let us define gene expression matrix of two single cell samples as follows:

$$X = \begin{bmatrix} x_{11} & \cdots & x_{1m} \\ \vdots & \ddots & \vdots \\ x_{n1} & \cdots & x_{nm} \end{bmatrix} \quad Y = \begin{bmatrix} y_{11} & \cdots & y_{1p} \\ \vdots & \ddots & \vdots \\ y_{n1} & \cdots & y_{np} \end{bmatrix}$$

To be noted, here X and Y are normalised and scaled as mentioned before.

The goal of canonical correlation analysis (CCA) is to find projection vectors u and v such that the correlation between Xu and Yv is maximized.

$$\max_{u,v} \frac{Cov(Xu, Yv)}{\sqrt{Var(Xu)}\sqrt{Var(Yv)}}$$

In CCA, we consider each column of X as a variable, each row of X as a realization of the variables.

If we denote X as random variables, namely

$$X = [X_1, X_2, \cdots, X_m]$$

Then,

$$Xu = \sum_{j=1}^{m} X_j u_j$$

is a combination of random variables. Xu has n realizations, corresponding to n genes.

If we standardise each column of X with mean 0 and standard deviation 1, this means

$$EX_j = 0, VarX_j = 1, for\ j = 1, 2, \cdots, m$$

Then,

$$EXu = \sum_{j=1}^{m} u_j EX_j = 0$$

$$VarXu = \sum_{i=1}^{m}\sum_{j=1}^{m} u_i u_j Cov(X_i, X_j) = \sum_{i=1}^{m}\sum_{j=1}^{m} u_i u_j EX_i X_j$$

For samples (consider X as realization instead of random variables again), this means,

$$VarXu = \sum_{i=1}^{m}\sum_{j=1}^{m} u_i u_j EX_i X_j = \sum_{i=1}^{m}\sum_{j=1}^{m} u_i u_j \frac{1}{n}\sum_{k=1}^{n} x_{ki} x_{kj} = \frac{1}{n} u^T X^T X u$$

Similarly, we can have:

$$VarYv = \sum_{i=1}^{m}\sum_{j=1}^{m} v_i v_j EY_i Y_j = \sum_{i=1}^{m}\sum_{j=1}^{m} v_i v_j \frac{1}{n}\sum_{k=1}^{n} y_{ki} y_{kj} = \frac{1}{n} v^T Y^T Y v$$

$$Cov(Xu, Yv) = \sum_{i=1}^{m}\sum_{j=1}^{p} u_i v_j EX_i Y_j = \sum_{i=1}^{m}\sum_{j=1}^{p} u_i v_j \frac{1}{n}\sum_{k=1}^{n} x_{ki} y_{kj} = \frac{1}{n} u^T X^T Y v$$

Therefore, after standarization of each column of X, the optimization goal can be simplified as:

$$\max_{u,v} \frac{u^T X^T Y v}{\sqrt{u^T X^T X u}\sqrt{v^T Y^T Y v}}$$

In many scRNA-seq experiments, the number of genes of interest that are shared between the two data sets is often much smaller than the total number of cells that were measured (*n* << *m* + *p*). Consequently, the vectors *u* and *v* that are returned from CCA as described in equation above will not be unique.

One potential solution to this is to regularize or penalize the CCA procedure to promote sparsity. However, this would assign many cells zero loadings in the resulting projections and result in a complete loss of information for a significant proportion of cells. Therefore, we treat the covariance matrix within each data set as diagonal, a solution that has demonstrated promising results in other high-dimensional problems. We substitute the identity matrix for $X^T X$ and Y^T to arrive at equation below.

$$\max_{u,v} u^T X^T Y v \; subject\; to\; u^T u = 1, v^T v = 1$$

Find the first k projection vectors (here u_i is a vector in R^n, instead of a real number shown above):

$$u_1, u_2, \cdots, u_k;\; v_1, v_2, \cdots, v_k;$$

Denote:

$$U = \begin{bmatrix} u_1 & u_2 & \cdots & u_k \end{bmatrix} = \begin{bmatrix} u_{11} & u_{12} & \cdots & u_{1k} \\ u_{21} & u_{22} & \cdots & u_{2k} \\ \vdots & \vdots & \ddots & \vdots \\ u_{m1} & u_{m2} & \cdots & u_{mk} \end{bmatrix}$$

$$V = \begin{bmatrix} v_1 & v_2 & \cdots & v_k \end{bmatrix} = \begin{bmatrix} v_{11} & v_{12} & \cdots & v_{1k} \\ v_{21} & v_{22} & \cdots & v_{2k} \\ \vdots & \vdots & \ddots & \vdots \\ v_{p1} & v_{p2} & \cdots & v_{pk} \end{bmatrix}$$

For each cell i in X,

$$\begin{bmatrix} u_{i1} & u_{i2} & \cdots & u_{ik} \end{bmatrix}$$

would be its new coordinates after CCA. The same is true for each cell i in Y.

Especially, we have

$$||u_j||^2 = 1 \; for \; j = 1, 2, \cdots, k; \; ||v_j||^2 = 1 \; for \; j = 1, 2, \cdots, k$$

Canonical correlation vectors (CCV) project the two datasets into a correlated low-dimensional space, but global differences in scale (for example, differences in normalization between datasets) can still preclude comparing CCV across datasets. To address this, we perform L2-normalization of the cell embeddings.

$$\hat{u_i} = \frac{u_i}{|u_i|}, |u_i| = \sqrt{\sum_{j=1}^{k} u_{ij}^2}$$

Following dimensional reduction, we identified the K-nearest neighbors (KNNs) for each cell within its paired dataset, based on the L2-normalized CCV. Finally, we identify mutual nearest neighbors (MNN; pairs of cells, with one from each dataset, that are contained within each other's neighborhoods). We refer to these pairwise correspondences as "anchors". The size of this neighborhood (k.anchor parameter in FindTransferAnchors and FindIntegrationAnchors) was set to 5 for all analyses in this manuscript.

Anchor scoring

The robust identification of anchor correspondences is key for effective downstream integration. Incorrect anchor pairs representing cells from distinct biological states can lead to incorrect downstream conclusions. In particular, cells that represent a biological state unique to one dataset should theoretically not participate in anchor pairs, yet in practice, they will do so with low frequency. We therefore implement two steps (filtering and scoring anchors) to mitigate the effects of any incorrectly identified anchors.

First, we ensure that the anchors we identify in low-dimensional space also are supported by the underlying high-dimensional measurements. To do this, we return to the original data and examine the nearest neighbors of each anchor query cell in the reference dataset. We perform the search using the max.features (200) genes with the strongest association with previously identified CCV, using the TopDimFeatures function in Seurat, and search in L2-normalized expression space.

The meaning of gene loadings?

The search for the 200 genes is as follows:

1. Initialise n = 2
2. For i in 1, 2, ..., k (the number of dimensions we choose when we did CCA), calculate Xu_i(X is normalised and scaled, but not standardised), which are gene loadings, select n genes with highest loadings (n/2 positive and n/2 negative) for each dimension.
3. Increase n by 2, repeat 2 until the total number is 200 (the highest number that does not exceed 200, for example, if n = 20, #Genes = 196, n = 22, #Genes=226, then choose genes when n = 20).

L2-normalized expression is normalised count data with L2 normalised per cell. To be noted, L2 normalisation is done with just the 200 selceted genes, so the so called L2-normalized expression space only contains the 200 genes.

Query dataset: input data 1.

Reference dataset: input data 2.

If the anchor reference cell is found within the first k.filter (200) neighbors *of the query cell it is anchored*, then we retain this anchor. Otherwise, we remove this anchor from further analyses. We do not include a mutual neighborhood requirement for this step, as it is primarily intended as a check to ensure that we do not identify correspondences between reference and query cells with very divergent expression profiles. This procedure is uniformly applied with default parameters (max.features = 200, k.filter = 200), for all analyses in this manuscript.

Additionally, to further minimize the influence of incorrectly identified anchors, we implemented a method for scoring anchors that is similar to the use of shared nearest neighbor (SNN) graphs in graph-based clustering algorithms. By examining the consistency of edges between cells in the same local neighborhood, SNN metrics add an additional level of robustness to edge identification [Levine et al., 2015]. For each reference anchor cell, we determine its k.score (30) nearest within-dataset neighbors and its k.score nearest neighbors in the query dataset. This gives us four neighbor matrices that we combine to form an overall neighborhood graph. For each anchor correspondence, we compute the shared neighbor overlap between the anchor and query cells, and assign this value as the anchor score. To dampen the potential effect of outlier scores, we use the 0.01 and 0.90 quantiles to rescale anchor scores to a range of 0 to 1.

For example, for anchor pair 1, which is composed of cell 1 and cell 2. Cell 1 have 30 NNs in query datasets and 30 in reference datasets, cell 2 also have 60 NNs together. If there is 30 of them overlap, then the score is 30. Then, for all of the scores, we can scale it using 0.01 and 0.9 quantile. Assume *s* is the vector of scores for all anchors,

$$s_i' = \frac{s_i - quantile_{0.01}(s)}{quantile_{0.9}(s) - quantile_{0.01}(s)}$$

$$s_i'' = \begin{matrix} s_i & \text{if } 0 < s_i' < 1 \\ 0 & \text{if } s_i' \leqslant 0 \\ 1 & \text{if } s_i' \geqslant 1 \end{matrix}$$

s'' is then the new scores for all anchors.

Anchor weighting

We construct a weight matrix *W* that defines the strength of association between each query cell *c*, and each anchor *i*.

For each cell *c* in the query dataset, we identify the nearest *k.weight* anchors cells in the query dataset in PCA space (Features used are HVGs selected in Feature Selection of Multiple Datasets). Nearest anchors are then weighted based on their distance to the cell *c* over the distance to the *k.weight*-th anchor cell and multiplied by the anchor score (S_{ai}). For each cell *c* and anchor a_i (both are from query datasets, for anchors, we only include anchor cells in the query datasets), we first compute the weighted distances as:

$$D_{c,i} = 1 - \frac{dist(c, a_i)}{dist(c, a_{k.weight})} S_{a_i}$$

We then apply a Gaussian kernel:

$$\tilde{D_{c,i}} = 1 - e^{\frac{-D_{c,i}}{(2/sd)^2}}$$

where *sd* is the Gaussian kernel bandwidth, set to 1 by default. Finally, we normalize across all *k*:*weight* anchors:

$$W_{c,i} = \frac{\tilde{D_{c,i}}}{\sum_{j=1}^{k.weight} \tilde{D_{c,j}}}$$

To be noted, for each cell *c* in the query dataset, we can find *k.weight* nearest anchor cells in the query dataset, but for the *k.weight* anchor cells, we have more than *k.weight* anchor pairs. For example, we selected 100 nearest anchor cells for cell 1, there are 152 related anchor pairs for these 100 anchor cells. Then we just order these 152 anchor pairs according to the distances between cell 1 and their corresponding anchor cells, we just select the top 100 anchor pairs and calculate the distance based on scores of these 100 pairs. This means that we did not use all 100 nearest anchor cells in the query datasets, but just part of them. For anchors in the query dataset which are not in the neareast 100 anchor cells of cell *c*, the weight is set as 0.

By this way, we constructed a weight matrix *W* that defines the strength of association between each query cell *c*, and each anchor *i*. These weights are based on two components: the distance between the query cell and the anchor, and the previously computed anchor score. In this way, query cells in distinct biological states (for example alpha cells and gamma cells) will be influenced by distinct sets of anchors, enabling context-specific batch correction. Additionally, robust anchors (with high scores) will gain influence across the query dataset, while inconsistent anchors will be downweighted.

Data integration for reference assembly

Once we have identified anchors and constructed the weights matrix, we follow the strategy outlined previously for batch correction. We first calculate the matrix *B*, where each column represents the difference between the two expression vectors for every pair of anchor cells, *a*:

$$B = Y - X$$

Each column of B represent an anchor pairs identified before. Columns of Y are anchor cells of the query dataset, columns of X are anchor cells of the reference dataset. Rows only include HVGs selected in the feature selection procedures.

We then calculate a transformation matrix, *C*, using the previously computed weights matrix and the integration matrix as:

$$C = BW^T$$

Each column of W^T can evaluate how much different anchor cells in the query dataset have the influence on the corresponding query cell.

We then subtract the transformation matrix, *C*, from the original expression matrix, *Y*, to produce the integrated expression matrix:

$$\hat{Y} = Y - C$$

This step is implemented in the IntegrateData function in Seurat. The corrected expression matrix can be treated as a single normalized scRNA-seq matrix, and can be processed downstream using any single-cell analytical toolkit. Notably, in Seurat, we continue to store the original uncorrected expression data, to facilitate downstream comparisons across datasets.