Fast and robust non-negative matrix factorization for single-cell experiments

Zachary J. DeBruine[1,*], Karsten Melcher[2] and Timothy J. Triche Jr[2]
[1]Department of Structural Biology, Van Andel Institute, Grand Rapids, Michigan, USA
[2]Department of Epigenetics, Van Andel Institute, Grand Rapids, Michigan, USA
[*]Corresponding author: zach.debruine@vai.org

See "RcppML" R package on CRAN for NMF implementation

RcppML development version: github.com/zdebruine/RcppML/issues

Abstract

Non-negative matrix factorization (NMF) is an intuitively appealing method to extract additive combinations of measurements from noisy or complex data. NMF is applied broadly to text and image processing, time-series analysis, and genomics, where recent technological advances permit sequencing experiments to measure the representation of tens of thousands of features in millions of single cells. In these experiments, a count of zero for a given feature in a given cell may indicate either the absence of that feature or an insufficient read coverage to detect that feature ("dropout"). In contrast to spectral decompositions such as the Singular Value Decomposition (SVD), the strictly positive imputation of signal by NMF is an ideal fit for single-cell data with ambiguous zeroes. Nevertheless, most single-cell analysis pipelines apply SVD or Principal Component Analysis (PCA) on transformed counts because these implementations are fast while current NMF implementations are slow. To address this need, we present an accessible NMF implementation that is much faster than PCA and rivals the runtimes of state-of-the-art SVD. NMF models learned with our implementation from raw count matrices yield intuitive summaries of complex biological processes, capturing coordinated gene activity and enrichment of sample metadata. Our NMF implementation, available in the RcppML (Rcpp Machine Learning library) R package, improves upon current NMF implementations by introducing a scaling diagonal to enable convex L1 regularization for feature engineering, reproducible factor scalings, and symmetric factorizations. RcppML NMF easily handles sparse datasets with millions of samples, making NMF an attractive replacement for PCA in the analysis of single-cell experiments.

Introduction

In 1999, Lee and Seung popularized Non-Negative Matrix Factorization (NMF) as a method to decompose data into additive "parts", for example, to extract facial features from sets of images (Lee & Seung, 1999). Non-negativity constraints on the input data give a colinearly optimized and additive collection of factors, contrasting with spectral decompositions such as PCA that describe sequentially dependent axes of variance.

Just as a matrix of pixels encoding a face might be decomposed into distinct facial features, a matrix of mRNA transcript counts in single cells may be decomposed into constituent biological processes (Brunet et al., 2004; Clark et al., 2019), orchestrated by patterns of coordinated gene expression. NMF exposes biology in a manner that is robust across experiments, enabling transfer learning (Stein-O'Brien et al., 2019). SVD and PCA can also be used to capture lower-dimensional representations, but the sequential rotations of coordinate axes to explain orthogonal variation in data are far less intuitive (as illustrated in Figure 1) and are relevant only to the input.

Single-cell count data is strictly non-negative and thus is well-suited to a non-negative dimensional reduction. PCA makes assumptions that requires transforming, scaling, and centering the data, is unable to impute signal, and can introduce perverse behavior. By contrast, missing signal in NMF is imputed up to a point that maximizes error of reconstruction (Lin & Boutros, 2020). Despite the theoretical attractiveness of NMF, many methods in single-cell analysis rely on variance-based decompositions (Qiu et al., 2017; Stuart et al., 2019) because SVD and PCA are relatively fast and yield reasonable input for graph-based visualization and clustering.

NMF has not been widely adopted by the single-cell community because current implementations are too slow. Perhaps more demonstrations of the usefulness of NMF are also needed. For instance, NMF can use raw counts instead of normalized, centered, and scaled data. NMF can also serve as both a dimensional reduction in place of PCA (Elyanow et al., 2020) and a resource for learning novel biology (Clark et al., 2019). With sufficient meta-analysis and dataset integration, it is possible that latent factors learned by NMF may replace manually curated toolkits such as GO terms in many gene set enrichment analyses (Stein-O'Brien et al., 2018). We make the case for replacing PCA with NMF in many single-cell analyses, present computational improvements enabling our fast implementation, and discuss remaining challenges for future work.

Results

Dimensional reduction of scRNA-seq data with NMF is suitable as a replacement for PCA

To make the case for NMF as a sensitive dimensional reduction, we analyzed 103,000 single blood cells from a human embryonic cell atlas (Cao et al., 2020). First, the optimal rank for a factorization was determined by plotting the mean squared error of an NMF model versus factorization rank for models of rank 1 to 100. An inflection point in the resulting curve was established around a rank of 35 (Figure 1a). A high-quality NMF model was learned within 1 minute on an average desktop computer with $k = 35$ using unnormalized counts and all genes. A k-Nearest Neighbor (k-NN) graph was constructed from sample loadings in this model and a UMAP embedding was learned to assess the sensitivity of the projection (Figure 1b). We observed generally distinct separation of cell types as annotated in the Human Embryonic Cell Atlas using PCA.

Principal component analysis (PCA) was run using the standard Seurat workflow. First, data was log-normalized (a step not needed in NMF), then scaled and centered (effectively converting a large sparse matrix to dense), then PCA was run for only the top 3000 highly variable genes, where 3000 genes were selected because much more than that is intractable. The inflection point in the "elbow plot" of standard deviation vs. rank coincided with approximately $k = 35$ (Figure 1d). The resulting UMAP embedding clearly delineated atlas-annotated cell types (Figure 1e), perhaps better than NMF. The major caveat, however, is that cell types were originally classified from a PCA embedding by the atlas authors and thus are inherently biased towards PCA as a ground truth embedding. A graph-based clustering on NMF coordinates using parameters comparable to the PCA clustering gave interpretable results that corresponded roughly to the PCA-designated clusters (Figure S2).

NMF factors recover relevant biological metadata in an interpretable manner

To examine whether NMF and PCA embeddings encode biological information, we examined enrichment for annotated cell types in latent factors. Some NMF factors mapped almost entirely to certain cell types while others were distributed across many cell types (Figure 1c). PCA factors, however, were randomly represented across all cell types with both positive and negative weightings (Figure 1f), which is confusing.

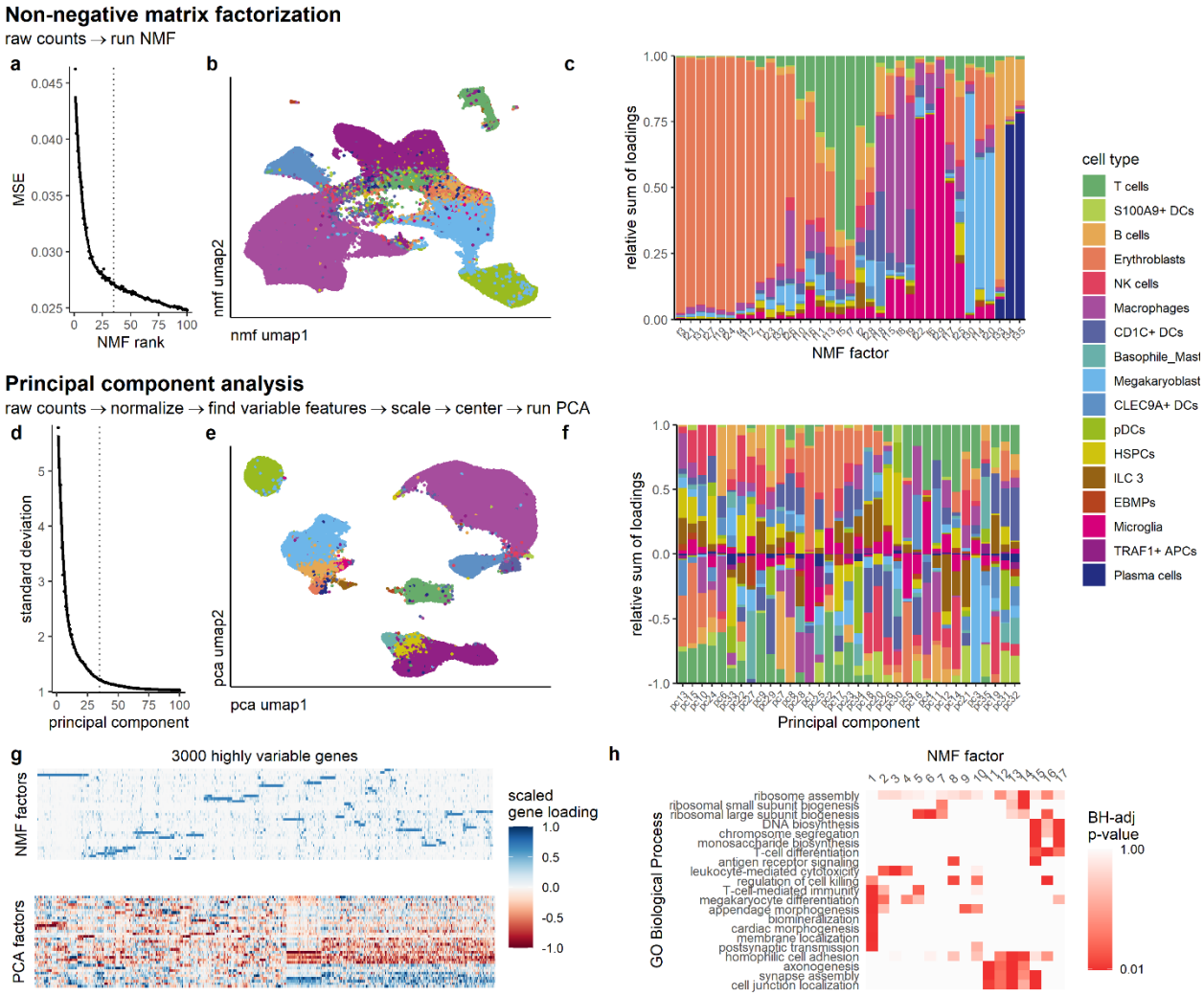

Figure 1. NMF, unlike PCA, captures biologically informative factors from 100,000 single blood cell transcriptomes.

a) Mean squared error (MSE) of NMF vs. factorization rank. Model at $k = 35$ selected for analysis (dotted line).

b) 100,000 blood cells on UMAP coordinates computed from a k-nearest neighbor graph on sample loadings in an NMF model. Cells are colored by cell type as annotated in the original dataset publication.

c) Relative sum of loadings for all NMF factors across all annotated cell types.

d) Additional standard deviation explained by each principal component (1 to 100 PCs) from a PCA using the top 3000 highly variable genes. Model at $k = 35$ selected for analysis (dotted line).

e) As in (b) for PCA.

f) As in (c) for PCA.

g) Gene loadings in PCA and NMF for 3000 highly variable genes as identified by Seurat. Maximum absolute value of loadings for each gene is scaled to 1.

h) Significantly enriched GO biological process terms in selected NMF factors. Terms were automatically selected which were variably enriched across factors with high significance.

NMF factors capture additive signals and thus unique sets of genes are described by each factor, while principal components capture axes of variation and thus genes receive negative or positive weights and groups of expressed genes are not identifiable (Figure 1g).

The unique sets of genes captured by NMF factors describe biological processes. To demonstrate this, we ran gene set enrichment analysis using Gene Ontology biological process terms for all NMF factors. A few differentially enriched terms compiled in an unsupervised manner showed that NMF factors captured immune cell-specific processes, cell cycle and proliferation processes, and microenvironment-specific processes in separate factors (Figure 1h). More systematic GO term enrichment is presented in Figure 2a.

NMF factors may explain any type of annotatable metadata. For instance, some NMF factors were differentially regulated across developmental days spanning an 8-week window of development (Figure S1a), while PCA factors gave confusing results (Figure S1b). Some NMF factors were strongly enriched in cells from male or female embryos (Figure S1c) or in cells obtained from certain source organs (Figure S1d).

NMF factors recover known biological processes in a structured manner

In the case of single-cell matrices of RNA transcript counts within cells, NMF factors represent biological processes (Clark et al., 2019; Stein-O'Brien et al., 2018). We systematically examined GO term enrichment in NMF and SVD rank-35 models. SVD was used in place of PCA because PCA of 100,000 cells and 25,000 genes on a dense matrix is nearly intractable, and no centering or scaling of a sparse matrix (and thus conversion to dense format) is necessary for SVD. Just over 3600 GO terms were significantly enriched in at least one of the SVD and NMF factors. The NMF factors manifested distinct patterns of GO term enrichment, indicating that factors describe largely unique biological processes (Figure 2a). However, SVD patterns manifested no such unique patterns, and factors that were enriched appeared to be non-specific (Figure 2b). Indeed, each NMF pattern had at least 100 enriched GO terms while most SVD patterns had none to very few (Figure 2c). These results show that SVD (and by extension, PCA) cannot capture biological processes, but rather solves for sequential normalizations towards axes of associated variation in the data, while NMF captures known genetic processes that coactivate to constitute transcriptional state.

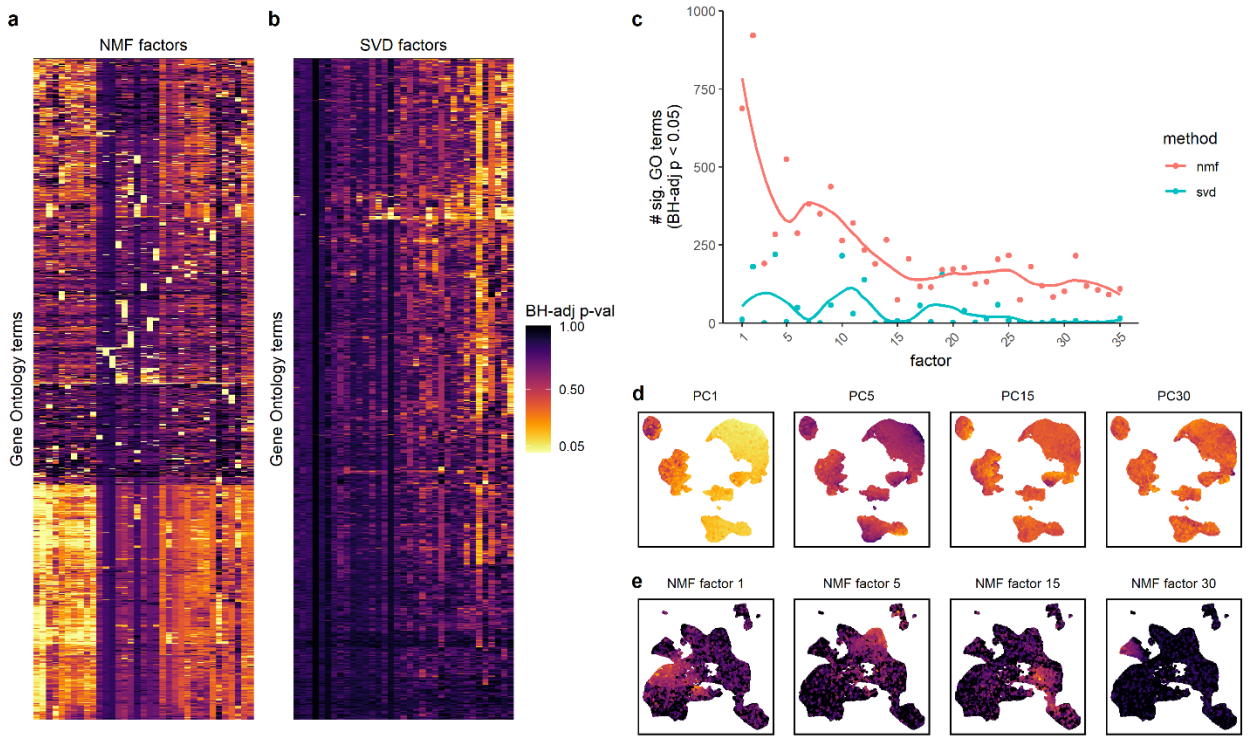

Figure 2. NMF factors, unlike principal components, represent biological processes that are differentially regulated across cell types.

a) Gene Ontology term enrichment across all 35 NMF factors, BH-adjusted *p*-values are shown. All terms with at least one significant value in either the NMF or SVD gene set enrichment analysis are shown.
b) as in (a) but for the first 35 SVD factors. Factors and term ordering not the same as in (a).
c) Number of significant GO terms (defined by BH-adjusted *p*-value < 0.05) in each NMF and SVD factor. Factors 1 to 35 are ranked by singular value (SVD) or proportion of total signal explained (NMF).
d) PCA sample loadings mapped to a UMAP embedding learned from the PCA (as in Figure 1E)
e) NMF sample loadings mapped to a UMAP embedding learned from the NMF model (as in Figure 1b). Color scale is a double square-root transformation for clarity.

PCA embeddings and NMF loadings are dispersed very differently throughout their respective UMAP embeddings. Principal component cell embeddings are broad gradients passing through all cells, explaining a small fraction of the variation in each cell (Figure 2d). NMF loadings are specific, localizing to distinct populations of cells explaining additive and distinct signals (Figure 2e). This again shows that NMF factors describe basic biology in addition to serving as a sensitive dimensional reduction, thus doing what PCA is commonly used to do while providing additional information.

RcppML NMF is nearly as fast as state-of-the-art SVD

Widespread adoption of NMF as a method in single-cell analysis requires reasonable runtime, which is not available in current NMF implementations. We benchmarked our NMF implementation against the Seurat PCA workflow and implicitly restarted Lanczos bidiagonalization SVD (irlba), a sparse matrix SVD known for its class-leading speed (Baglama & Reichel, 2005). For datasets of both 2,700 PBMCs (Figure 3a) and 40,000 bone marrow cells from the Human Cell Atlas (Figure 3b), RcppML NMF was nearly as fast as irlba SVD. RcppML NMF was significantly faster than Seurat PCA, especially in the larger dataset, demonstrating that NMF is certainly fast enough for any single-cell analyses that can make use of PCA. Of note, only 3000 highly variable genes were used to run Seurat PCA while NMF used all genes, since PCA is unfortunately intractable with all genes on larger datasets. RcppML NMF also performed impressively on benchmarking tests in random sparse matrices regardless of factorization rank, matrix sparsity, number of samples in the matrix, or rectangularity of the matrix (Figure S5). These benchmarks were performed on an average desktop computer.

Diagonalized NMF enables symmetric factorization and robust factor scalings

NMF algorithms require an initialization to kickstart iterative updates. Like most other NMF implementations, RcppML NMF uses random initialization. Different random initializations give different solution paths and different relative distributions of values in w and h. This means that factors across independent restarts, while highly correlated, will be scaled differently. Thus, downstream analyses must normalize the model in some way to account for these differences.

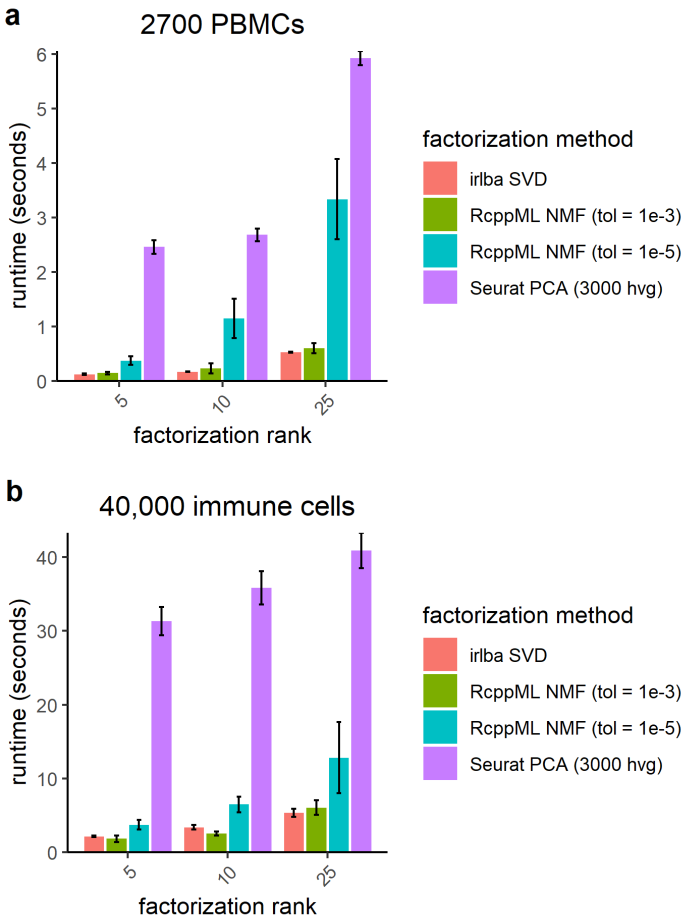

Figure 3. RcppML NMF is faster than Seurat PCA and nearly as fast as state-of-the-art SVD.

a) Runtime for factorization of 2,800 cells from the PBMC3k dataset using irlba SVD, RcppML NMF at two tolerances ($1e^{-3}$ corresponds to rank-checking quality, $1e^{-5}$ corresponds to publication quality), and Seurat PCA (with normalization, scaling, and centering, and using only 3000 highly variable features as opposed to all features used in NMF).

b) Runtime for factorization of 40,000 cells from the Human Cell Atlas Bone Marrow dataset (available through SeuratData). NMF settings as in (a).

To address this issue of factor scaling, we introduce a scaling diagonal in RcppML NMF which scales each factor in *w* and *h* to sum to 1 after each update. This means that the distribution of factors and the relative contributions of features and samples to each factor is always comparable. Such model "diagonalization" does not change the information content or loss of the model, it only linearly scales the model. To show this, we devise a measure of model "robustness" which measures the cosine similarity of the best possible factor pairing between two models (Figure S3a). Robustness is not affected by model diagonalization (compare Figure S3b and S3c), although it does depend on model tolerance. Notice that low-rank models give exactly robust solutions, while higher-rank models are hindered from discovering these exactly robust models by machine precision during alternating updates.

Diagonalization enables symmetric factorization of symmetric matrices. In factorization of a symmetric matrix, it is expected that *w* and *h* will converge from different random initial states towards a shared common state. Without diagonalization, the distributions of *w* and *h* are not the same, and thus factor scalings are linear transformations of one another (Figure 4c). With diagonalization, *w* and *h* converge towards one another, and symmetry of the model is guaranteed (Figure 4d).

The scaling diagonal of NMF is different in nature to that of an SVD: NMF factors are optimized simultaneously while SVD factors (for example in rank-truncated SVD) are optimized sequentially. Thus, while the diagonal indicates the relative contribution of each factor to the model loss, it is never a method for quick-and-dirty rank determination using the "elbow method" as in PCA or SVD.

L1 Regularization improves NMF model feature sparsity and interpretation

L1 regularization (i.e. LASSO) is a method for increasing the sparsity of a solution. Increased sparsity allows for easier interpretation of significantly associated variables. L1 regularization has previously been proposed for NMF (Lin & Boutros, 2020), but unfortunately the penalty is not predictable without knowing the distribution of the model to be learned beforehand. In most NMF algorithms, the distribution of the model is at the mercy of the initialization. Diagonal scaling ensures that the distribution of loadings within each factor is always constant with each update,

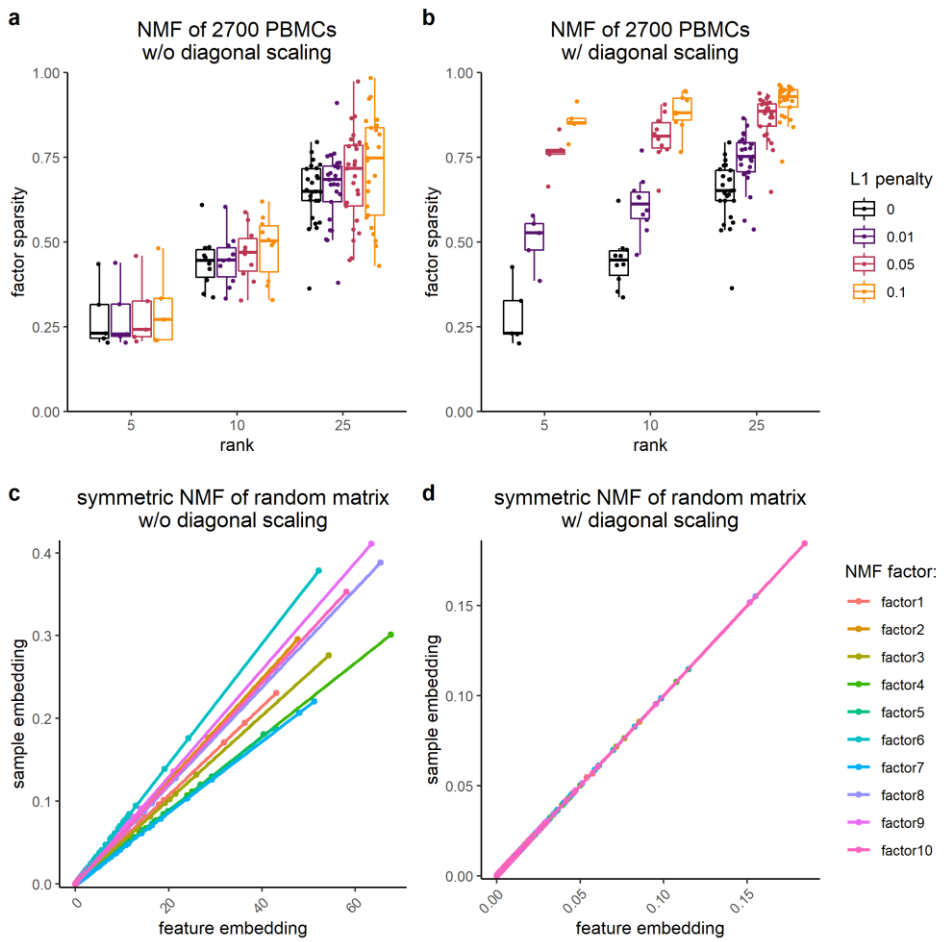

Figure 4. Diagonalized NMF gives robust factor scalings that enable L1 regularization and symmetric factorization.

a) Sparsity of factors (factors are points) given by an undiagonalized NMF of 2700 PBMCs at several L1 penalties and ranks.
b) As in (b), but with diagonal scaling.
c) Rank-10 undiagonalized NMF of a random symmetric 1000 x 1000 90% sparse matrix. Loadings in "*w*" are plotted against corresponding loadings in "*h*", colored by to which of the 10 factors they belong.
d) As in (c), but with diagonal scaling.

and thus regularizations are consistent, and act as expected, with a penalty of 1 guaranteeing complete sparsity (Figure S4a).

Without diagonalization, numerical instability can result when less important factors are penalized unfairly and driven to total sparsity while more significant factors remain very dense (Figure 4a). In other words, the model seeks to reduce the loss objective based on the distribution of unscaled factors. However, with diagonalization, L1 regularization gives factors that are equally sparse regardless of how much each factor contributes to reduction of the loss objective, and no factors are driven towards zero (Figure 4b). Factors in L1-regularized diagonalized NMF correlate strongly with factors in unregularized diagonalized NMF because L1 regularization does not change the content of the learned model (Figure S4b).

To determine how L1 regularization affects factor sparsity, we factorized the Mouse Organogenesis Cell Atlas at several ranks and L1 penalties. This dataset consists of 1.3 million cells, is approximately 95% sparse, and was factorized in its' entirety at many combinations of rank and L1 penalty in a few hours on a high-performance computing node with 40 processors. We found that L1 regularization did not significantly affect the mean squared error of factorization regardless of rank (Figure S4c), although it had a profound effect on the sparsity of factors (Figure S4d). Since NMF factors describe biological processes, we ran gene set enrichment analysis on Gene Ontology terms and found that L1 penalties did not significantly affect the number of enriched GO terms per factor regardless of rank (Figure S4e). In summary, L1 regularization in diagonalized NMF only sets near-zero coefficients to zero (Figure S4f), thus improving the interpretability of latent factors. While this behavior should always be expected of L1 regularization in NMF, all other implementations of NMF that do not scale factors before each update cannot give a convexly fitted L1-regularized model.

A novel Non-Negative Least Squares algorithm increases factorization speed

A major bottleneck in NMF by alternating least squares is finding solutions to non-negative least squares systems of equations. The NNLM R package has shown great promise for sequential coordinate descent least squares initialized with an approximate solution given by the model from the previous iteration (Lin & Boutros, 2020). We considered initializing coordinate descent non-negative least squares with either a zero-filled vector (see an example solution path in Figure S7a),

a randomly-filled vector, an unconstrained least squares solution, or an approximate solution that we refer to as "Fast Active Set Tuning" (FAST). In FAST, an unconstrained least squares solution is computed, all negative values set to zero (an "active set"), and all other values are added to a "feasible set". An unconstrained least squares solution is then solved for the "feasible set", any negative values in the resulting solution are set to zero, and the process is repeated until the feasible set solution is strictly positive. The algorithm has a definite convergence guarantee because the feasible set will either converge or become smaller with each iteration. The result is generally exact or nearly exact for small well-conditioned systems (< 50 variables) within 2 iterations and thus quickly sets up coordinate descent very well (see example solution path in Figure S7b). The FAST method is similar to the first phase of the previously described so-called "TNT-NN" algorithm (Myre et al., 2017), but the latter half of that method relies heavily on heuristics to find the true active set, which we avoid by using coordinate descent instead.

A fortran77 implementation of NNLS by Lawson-Hanson has been the gold standard NNLS algorithm for many decades, yet FAST-CD NNLS convincingly outperforms this standard for any random system greater than 30 variables in size (Figure S7c).

For solving random ill-conditioned systems, the FAST subroutine alone performs well indeed, but solutions deteriorate in quality as system size increases (Figure S7d). However, FAST-CD NNLS still outperforms coordinate descent NNLS initialized randomly, with zeros, or with unconstrained least squares.

In the context of matrix factorization, extensive experimentation and benchmarking across many diverse datasets has led us to initialize solutions with unconstrained least squares and refine by coordinate descent. We note that while FAST-CD performed well on ill-conditioned random systems compared to FAST or CD subroutines alone (Figure S7e), this advantage was lost when using real data with non-random signal structure (Figure S7f-g).

Discussion

NMF is well-suited to dimensional reduction of single-cell data for visualization, use as an embedding for clustering, and learning patterns of coordinated gene activities. Our algorithm brings compute time down to nearly that of the fastest SVD implementations and adds

diagonalization to aid interpretability and convex regularization. Taken together, this work makes a case for adoption of NMF into mainstream single-cell analysis pipelines.

However, NMF shares some challenges with its signed dimensional reduction counterparts. For example, rank determination remains a difficult problem, although in Figure 1a we show that rank determination can be as easy as that for PCA using an inflection point in a curve of rank vs. an objective (loss for NMF and standard deviation for PCA). Unfortunately, this "elbow plot" method is highly subjective. Future work must explore more quantitative approaches such as cross-validation against robustness objectives, or iterative rank updating with a stopping criterion based on relative change in mean squared error, for example. Cross-validation with missing value imputation has previously been proposed (Lin & Boutros, 2020), yet this is computationally intensive and sensitive to all hyperparameters that are required. However, rigorous rank-determination methods for PCA are at least equally consuming as those for NMF, such as jackstraw or *k*-fold cross-validation.

Reproducibility is yet another challenge for NMF. Alternating least squares requires an initialization, such as a random or non-negative double SVD model (NNDSVD) (Esposito, 2021). While NNDSVD is "robust", it differs fundamentally in nature from NMF, and any non-random initialization can trap updates into a local minimum even if random noise is added to the model and zeros are filled. Different random initializations can lead to different solution paths toward local minima, as finding the global minima is NP-hard (Vavasis, 2009). Setting a random seed to guarantee reproducibility of a factorization model is far from ideal when the learned factors are intended to describe fundamental biology. However, for most applications and including scRNA-seq, replicate models give similar errors of reconstruction, many factors are robust, and those that are not as robust share the remaining information among themselves differently between replicates. Some heuristics have been proposed to find "consensus" NMF factors across multiple models (Kotliar et al., 2019; Stein-O'Brien et al., 2019), but these methods require significant compute time, no longer treat NMF as a dimensional reduction, and may be biased towards factors explaining dominant signals. While robustness may be perceived as a weakness of NMF, it is important to realize that SVD is very sensitive to minor adversarial attacks on the input data, and is only robust for individual datasets, not across datasets. For example, different normalizations of the data or batch effects can lead to fundamentally different SVD results across most factors. On the

other hand, because NMF factors are collectively updated, distinct technical issues are usually explained by a single factor while other factors are left unaffected, making it robust across datasets. This same reasoning explains why no normalization of input data is required for NMF – the method naturally applies any necessary scalings or corrections.

NMF is an established method for extracting additive signals from non-negative data. For single-cell analysis, NMF describes coordinated gene activities and serves as a sensitive dimensional reduction for visualization and clustering. It seems more reasonable to cluster cells on factors describing biological processes than on principal components explaining variance. Further, it seems more reasonable to visualize cells based on underlying additive signals rather than embeddings learned from sequential normalizations of the data. Finally, while PCA and SVD are imputation-incompetent signed decompositions, NMF imputes missing values, denoises false positives, and respects non-negative inputs with non-negative outputs. Non-negative matrix factorization has the potential to systematically capture the complexities of genetic diversity, and it can do so simply, interpretably, and quickly.

Materials and Methods

NMF Problem Definition

Non-negative matrix factorization seeks to decompose a matrix A into orthogonal lower-rank non-negative matrices *w* and *h*:

$$A \approx wh \tag{1}$$

In the above equation, considering scRNA-seq datasets, *A* is a sparse matrix of unnormalized raw gene counts as rows and cells as columns, *w* is a dense matrix of "genes-by-factors", and *h* is a dense matrix of "factors-by-cells".

NMF algorithms generally require some initialization of *w* and/or *h* and use alternating updates of *w* and *h* to refine the model until convergence as determined by some stopping criteria. At convergence, the cross-product *wh* will approximate the input *A*.

Alternatively, inspired by SVD or PCA, one might consider the equation:

$$A \approx wdh \tag{2}$$

NMF may be diagonalized by scaling factors in *h* and *w* to sum to 1 after each alternating update. For example, after *h* is updated, *d* is set equal to the sum of factors in *h* and then each factor in *h* is scaled to sum to 1. Next, *w* is updated, after which *d* is set to the sum of factors in *w* and each factor in *w* is scaled to sum to 1.

NMF Algorithm

RcppML NMF makes use of the traditional algorithm for NMF by alternating least squares (ALS) (Kim & Park, 2007). This algorithm is fast and flexible, especially compared to multiplicative updates (Lin & Boutros, 2020). In ALS-NMF, w and h are updated alternatively with each iteration. For example, to update a column of *h* (h_i) given a column of *A* (A_i) and *w*, a system of equations is determined as follows:

$$a = w^T w \tag{3}$$

$$b = A_i w \tag{4}$$

$$a h_i = b \tag{5}$$

A similar approach may be used for updating rows of *w*, only *A* must first be transposed, and *h* used in place of *w* in the above equations. Equations #3-5 describe projection of a linear model which is a fundamental method in transfer learning. Projections of dense models onto sparse matrices may be performed using the R function "*RcppML::project*".

NMF Stopping Criteria

A well-established measure of convergence for matrix factorizations is given by the relative change in mean squared error (MSE) between consecutive iterations (Lin & Boutros, 2020). For a given iteration *i*, this measure is given by:

$$\Delta MSE = 2\frac{\sum(A - w_{i-1}h_{i-1})^2 - \sum(A - w_i h_i)^2}{\sum(A - w_{i-1}h_{i-1})^2 + \sum(A - w_i h_i)^2} \tag{6}$$

Calculating change in MSE requires computing the cross-product of *w* and *h* for each iteration for a large dense approximation of the model in *wh*, then subtracting dense *wh* from sparse *A*. Such a dense-sparse operation, even with parallelization and proper use of sparse matrix iterators (as in "*RcppML::mse*"), is extremely intensive, often more so than the factorization itself.

Because MSE depends on *w* and *h*, the relative change in *w* and *h* across consecutive iterations may serve as a more efficient stopping criteria. Consider a stopping criteria (R) using Pearson correlation (R^2) between models in consecutive iterations. For a given iteration *i*:

$$R = cor(w_{i-1}, w_i) \tag{7}$$

For a symmetric factorization, the stopping criteria could be correlation between *w* and *h*, since at convergence $w \approx h$:

$$R = cor(w_i, h_i) \tag{8}$$

Indeed, for a symmetric factorization, the change in MSE between consecutive iterations, the correlation between w across consecutive iterations, and the correlation between *w* and *h* for a given iteration, follow extremely similar trends (Figure S6a). Since calculating correlation between models is at least an order of magnitude faster than calculating MSE (Figure S6b), RcppML NMF uses eqn. #7 as a stopping criterion in place of MSE.

NMF Program Implementation

RcppML is a library of R functions that are lightweight wrappers around C++ functions coded with Rcpp and RcppEigen (Bates & Eddelbuettel, 2013; Eddelbuettel & François, 2011). The Eigen C++ library is arguably the fastest non-commercial BLAS and linear algebra library. We began by working in Armadillo, achieved comparable performance in base Rcpp, but found Eigen to be at least twice as fast for all use cases in addition to 3-5x faster at computing Cholesky decompositions. All RcppML Eigen code is provided in a single C++ file so that functions can be readily incorporated into other C++ programs and software without using Rcpp wrappers, provided the Eigen header library is loaded.

An Rcpp sparse matrix class was written to handle zero-copy by-reference access to R objects in C++. This contrasts with Eigen and Armadillo sparse matrices, for example, which form deep copies. Constant sparse matrix forward column iterators are used for contiguous access to non-zero values. Such optimization for sparsity provides performance gains for matrices >80% sparse, below which the overhead of sparse iterators begins to underperform dense operations. scRNA-seq count data is rarely <80% sparse and usually between 92-98% sparse.

Block-pivoting in alternating updates of *w* and *h* require transposition of sparse matrix *A*. Transposition of very sparse column-major matrices is an inefficient operation. We profiled in-place updating of *w* and found that the computational cost of in-place updating surpassed the benefit of avoiding transposition after about 3 iterations. RcppML NMF therefore does not provide support for in-place updates of *w* and uses block-pivoting instead.

Parallelization with OpenMP is implemented across rows in *w* and columns in *h* for each alternating least squares update. By default, all available CPU threads are used. Calculation of the right-hand side of systems of equations is included in the parallelization loop alongside the least squares solver to minimize overhead and maximize the contiguousness of memory address access within threads.

The FAST least squares algorithm relies heavily on the Eigen Cholesky module for extremely fast LLT decompositions and solutions by forward-backward substitution. For projecting a linear model, the LLT decomposition for the left-hand side of the system is computed only once (i.e., “preconditioned”). Sequential coordinate descent least squares is adapted from the NNLM RcppArmadillo package (Franc et al., 2005; Lin & Boutros, 2020), with modest optimizations.

Support for matrix factorization without non-negativity constraints is also provided. In this case, a pre-conditioned LLT decomposition is used to quickly project models during alternating updates, requiring only forward-backward substitution for solutions across all rows or columns.

We explored the use of “float” precision for matrix factorization and found that tolerances of even $1e^{-5}$ to $1e^{-6}$ were difficult to achieve (“double” precision permits $1e^{-10}$ to $1e^{-12}$), and performance gains were underwhelming (~10-15% at best over “double”). Despite additional numerical

stability gained by model diagonalization, in our opinion NMF should not sacrifice precision to this extent for marginal runtime gains and thus we support only "double" types.

Extensive code profiling and experimentation ensured we took full advantage of compile-time optimization and Eigen-facilitated vectorization. We were mindful of passing by reference, in-place updating, and contiguous memory access patterns for optimal use of the cache.

Further marginal speed gains may be possible with GPU acceleration and Eigen BLAS optimization, but these are beyond the scope of our work. We are confident that we have made the most of any high-level programmatic optimization possible. Our attention now turns to providing support for dense factorizations (i.e., symmetric matrices in single-cell analysis pipelines tend to be dense), zero-masked and weighted factorization (i.e., for methods with significant dropout and structured sparsity such as ATAC-seq), and supporting the method with cross-validation utilities, and accelerated rank-2 exact factorization solutions.

Benchmarks

All runtime benchmarks and nearly all figures were generated on an average desktop computer to show that any user can make good use of NMF for single-cell datasets. Benchmarks were run using a 64-bit OS running Windows 10 on an Intel Core i5-8400 CPU processor at 2.80GHz with 8GB of RAM with the R "microbenchmark" package. Runtimes were faster on a computing cluster, but relative performance between methods was similar. Random sparse matrices were generated using the "rsparsematrix" function from the R "Matrix" package.

Dimensional Reduction of Embryonic Blood Cells

NMF was run on raw unnormalized RNA transcript count data from 100,000 blood cells from the Human Embryonic Cell Atlas to 100 iterations at $k = 35$. PCA was run on log-normalized, centered, and scaled count data using Seurat default parameters. k-NN graphs were constructed with $k = 20$ on dimensions 1 to 35, UMAP plots were constructed from k-NN graphs using dimensions 1 to 35, 50 neighbors, and "Seurat::RunUMAP" defaults. Cellular metadata packaged with the Blood Cells dataset was used for enrichment analyses for cell type, organ type, embryo sex, and developmental day. GO enrichment analysis used msigdbr human C5 pathways, fast gene set enrichment

analysis (fgsea), and BH-adjusted p-values. Many terms in the NMF factors were highly significant (p-adj < 1e-8), but the significance cutoff was defined at p-adj < 0.05 for convention.

Data Availability

Figures 1 and 2 make use of the Human Embryonic Cell Atlas dataset, "Blood Cells" processed dataset available at https://descartes.brotmanbaty.org/bbi/human-gene-expression-during-development/, along with associated cell and gene metadata (Cao et al., 2020). Datasets used for benchmarking are available via SeuratData, specifically the "pbmc3k" dataset of 2,800 PBMC cells supplied by 10x genomics, and the "hcabm40k" dataset, containing 40000 bone marrow cells from the Human Cell Atlas (Han et al., 2020). The Mouse Organogenesis Cell Atlas was used in Figure S7 (Cao et al., 2019).

Code Availability

The latest stable release of the RcppML R package is publicly available on CRAN. The development version is available at github.com/zdebruine/RcppML. Rmarkdown files to reproduce all figures are appended to this document as supplementary information.

Funding

This work has been supported by the Van Andel Institute Graduate School (ZJD) the Van Andel Institute (KM and TJT), and the Michelle Lunn Hope Fund and Grand Rapids Community Foundation (TJT).

Competing Interests

None declared.

References

Baglama, J., & Reichel, L. (2005). Augmented implicitly restarted lanczos bidiagonalization methods. *SIAM Journal on Scientific Computing*, *27*(1). https://doi.org/10.1137/04060593X

Bates, D., & Eddelbuettel, D. (2013). Fast and elegant numerical linear algebra using the rcppeigen package. *Journal of Statistical Software*, *52*(5). https://doi.org/10.18637/jss.v052.i05

Brunet, J. P., Tamayo, P., Golub, T. R., & Mesirov, J. P. (2004). Metagenes and molecular pattern discovery using matrix factorization. *Proceedings of the National Academy of Sciences of the United States of America*, *101*(12). https://doi.org/10.1073/pnas.0308531101

Cao, J., O'Day, D. R., Pliner, H. A., Kingsley, P. D., Deng, M., Daza, R. M., Zager, M. A., Aldinger, K. A., Blecher-Gonen, R., Zhang, F., Spielmann, M., Palis, J., Doherty, D., Steemers, F. J., Glass, I. A., Trapnell, C., & Shendure, J. (2020). A human cell atlas of fetal gene expression. *Science*, *370*(6518). https://doi.org/10.1126/SCIENCE.ABA7721

Cao, J., Spielmann, M., Qiu, X., Huang, X., Ibrahim, D. M., Hill, A. J., Zhang, F., Mundlos, S., Christiansen, L., Steemers, F. J., Trapnell, C., & Shendure, J. (2019). The single-cell transcriptional landscape of mammalian organogenesis. *Nature*, *566*(7745). https://doi.org/10.1038/s41586-019-0969-x

Clark, B. S., Stein-O'Brien, G. L., Shiau, F., Cannon, G. H., Davis-Marcisak, E., Sherman, T., Santiago, C. P., Hoang, T. V., Rajaii, F., James-Esposito, R. E., Gronostajski, R. M., Fertig, E. J., Goff, L. A., & Blackshaw, S. (2019). Single-Cell RNA-Seq Analysis of Retinal Development Identifies NFI Factors as Regulating Mitotic Exit and Late-Born Cell Specification. *Neuron*, *102*(6). https://doi.org/10.1016/j.neuron.2019.04.010

Eddelbuettel, D., & François, R. (2011). Rcpp: Seamless R and C++ integration. *Journal of Statistical Software*, *40*(8). https://doi.org/10.18637/jss.v040.i08

Elyanow, R., Dumitrascu, B., Engelhardt, B. E., & Raphael, B. J. (2020). NetNMF-SC: Leveraging gene-gene interactions for imputation and dimensionality reduction in single-cell expression analysis. *Genome Research*, *30*(2). https://doi.org/10.1101/gr.251603.119

Esposito, F. (2021). A review on initialization methods for nonnegative matrix factorization: Towards omics data experiments. *Mathematics*, *9*(9). https://doi.org/10.3390/math9091006

Franc, V., Hlaváč, V., & Navara, M. (2005). Sequential Coordinate-Wise Algorithm for the Non-negative Least Squares Problem. *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, *3691 LNCS*, 407–414. https://doi.org/10.1007/11556121_50

Han, X., Zhou, Z., Fei, L., Sun, H., Wang, R., Chen, Y., Chen, H., Wang, J., Tang, H., Ge, W., Zhou, Y., Ye, F., Jiang, M., Wu, J., Xiao, Y., Jia, X., Zhang, T., Ma, X., Zhang, Q., … Guo, G. (2020). Construction of a human cell landscape at single-cell level. *Nature*, *581*(7808). https://doi.org/10.1038/s41586-020-2157-4

Kim, H., & Park, H. (2007). Sparse non-negative matrix factorizations via alternating non-negativity-constrained least squares for microarray data analysis. *Bioinformatics*, *23*(12). https://doi.org/10.1093/bioinformatics/btm134

Kotliar, D., Veres, A., Nagy, M. A., Tabrizi, S., Hodis, E., Melton, D. A., & Sabeti, P. C. (2019). Identifying gene expression programs of cell-type identity and cellular activity with single-cell RNA-Seq. *ELife*, *8*. https://doi.org/10.7554/eLife.43803

Lee, D. D., & Seung, H. S. (1999). Learning the parts of objects by non-negative matrix factorization. *Nature*, *401*(6755). https://doi.org/10.1038/44565

Lin, X., & Boutros, P. C. (2020). Optimization and expansion of non-negative matrix factorization. *BMC Bioinformatics*, *21*(1). https://doi.org/10.1186/s12859-019-3312-5

Myre, J. M., Frahm, E., Lilja, D. J., & Saar, M. O. (2017). TNT-NN: A Fast Active Set Method for Solving Large Non-Negative Least Squares Problems. *Procedia Computer Science*, *108*. https://doi.org/10.1016/j.procs.2017.05.194

Qiu, X., Mao, Q., Tang, Y., Wang, L., Chawla, R., Pliner, H. A., & Trapnell, C. (2017). Reversed graph embedding resolves complex single-cell trajectories. *Nature Methods*, *14*(10). https://doi.org/10.1038/nmeth.4402

Stein-O'Brien, G. L., Arora, R., Culhane, A. C., Favorov, A. V., Garmire, L. X., Greene, C. S., Goff, L. A., Li, Y., Ngom, A., Ochs, M. F., Xu, Y., & Fertig, E. J. (2018). Enter the Matrix: Factorization Uncovers Knowledge from Omics. In *Trends in Genetics* (Vol. 34, Issue 10). https://doi.org/10.1016/j.tig.2018.07.003

Stein-O'Brien, G. L., Clark, B. S., Sherman, T., Zibetti, C., Hu, Q., Sealfon, R., Liu, S., Qian, J., Colantuoni, C., Blackshaw, S., Goff, L. A., & Fertig, E. J. (2019). Decomposing Cell Identity for Transfer Learning across Cellular Measurements, Platforms, Tissues, and Species. *Cell Systems*, *8*(5). https://doi.org/10.1016/j.cels.2019.04.004

Stuart, T., Butler, A., Hoffman, P., Hafemeister, C., Papalexi, E., Mauck, W. M., Hao, Y., Stoeckius, M., Smibert, P., & Satija, R. (2019). Comprehensive Integration of Single-Cell Data. *Cell*, *177*(7). https://doi.org/10.1016/j.cell.2019.05.031

Vavasis, S. A. (2009). On the Complexity of Nonnegative Matrix Factorization. *Http://Dx.Doi.Org/10.1137/070709967*, *20*(3), 1364–1377. https://doi.org/10.1137/070709967