USING REGULATORY NETWORKS TO ENHANCE SINGLE-CELL CLUSTERING

ALEX USELOFF

Thesis Submitted in Partial Fulfillment of the Requirements for the Degree of

Master of Science

Thesis Advisor: Mehmet Koyutürk

Department of Computer and Data Sciences

Case Western Reserve University

January, 2024

USING REGULATORY NETWORKS TO ENHANCE SINGLE-CELL CLUSTERING

Case Western Reserve University
Case School of Graduate Studies

We hereby approve the thesis[1] of

ALEX USELOFF

for the degree of

Master of Science

Mehmet Koyutürk	November 21, 2023
Committee Chair, Advisor Department of Computer and Data Sciences	Date
Jing Li	November 21, 2023
Committee Member Department of Computer and Data Sciences	Date
Yinghui Wu	November 21, 2023
Committee Member Department of Computer and Data Sciences	Date

[1]We certify that written approval has been obtained for any proprietary material contained therein.

Table of Contents

List of Tables

List of Figures

Acknowledgements

I would like to thank Dr. Mehmet Koyuturk from the bottom of my heart for being my research advisor and, along with that, for helping me and being with me throughout this entire process as it's been a very long journey. We essentially had to restart twice over the course of these 3 years when we hit dead ends, each time building off of what we had done well the previous time, so as you could imagine, he has given a lot of time and effort into coming up with ideas and working with me.

I would also like to thank Serhan Yilmaz for not only being a great role model as to what my work could potentially become, but for providing me with his knowledge and advice over the past 3 years. Furthermore, I would like to thank Thomas Varley for sharing his ideas with me at the beginning of this process, and for helping to introduce me to single-cell data. I would additionally like to thank Dr. Jing Li and Dr. Yinghui Wu for being a part of my thesis defense committee.

Lastly, I would like to thank the Computer and Data Sciences Department at Case Western Reserve University for the opportunity to take part in and execute this research that I have written this thesis about.

ABSTRACT

USING REGULATORY NETWORKS TO ENHANCE SINGLE-CELL CLUSTERING

ALEX USELOFF

The clustering of single-cell RNA-sequencing data has been established as an important first step in single-cell gene expression data analysis for scientists to identify cell type based on RNA level expression. This is important because once a cell type has been identified, the phenotype association, as well as the spatio-temporal dynamics of specific cell types, can be characterized, which could lead to identifying cells associated with cancers and other diseases. However, the high-dimensionality of the data poses computational challenges, while drop-outs (genes that are not identified despite being expressed) hamper the reliability of inference. Since established knowledge on transcriptional regulatory networks provide information on the regulatory relationships between genes, we hypothesize that regulatory networks can help remedy missing data, while also reducing dimensionality. To test this hypothesis, we use a previously existing regulatory network, modern clustering methods, and network propagation together to help enhance clustering performance, which enhances accurate identification of cell types.

1 Introduction

1.1 Background and Motivations

In recent years, technological advances have allowed for the analysis of individual cells using single-cell sequencing technologies[23]. Monitoring of molecular expression and activity at the single-cell level provides unique opportunities to break down the interactions between cells as well as the processes that cells go through[23]. Single-cell studies are also important in clinical settings because they contribute massively to our understanding of how an individual cell can influence the outcome of infections, drug or antibody resistance, and cancers[1,5,17,18,23].

Unsupervised learning, specifically clustering, is the main way that we are currently able to analyze single-cell RNA-sequencing data because it groups cells together without having to know each cell's type. This is important as the identification of cell type is a critical first step in data analysis; if we know a cell belongs to a certain group, then we can infer certain characteristics about it that we would not otherwise know based on the other cells in that grouping. The issue that most people face with clustering single-cell data, however, is that these datasets are very high-dimensional and sparse, which presents computational challenges[9].

There are five main ways that researchers currently are trying to get around this computational challenge, and we explain those methods with some examples in Chapter 2. The main two issues with them are that the vectors being clustered,

which represent cells, are very high in dimensions, which represent genes, and clustering algorithms are sensitive to sequencing platforms and dropouts, which can greatly reduce the quality and performance of clustering[4,13]. In addition to these problems, RNA level expression alone does not provide the full picture of activity in a cell; there are a lot of different types of regulation occurring, accounted for by mRNA-level expression. It is for this reason that we incorporate a known network of gene-to-gene interactions to enhance the clustering of single-cell RNA-sequencing data. Our hypothesis is that using a gene-to-gene interaction network in combination with the methods that currently exist to cluster single-cell data will result in better clustering performance, which will in turn lead to more cell types being identified properly. This is important because if more cell types are identified properly, then the correct courses of action can be taken as reactionary measures.

1.2 Contributions

We found that incorporating a known gene-to-gene interaction network in conjunction with network propagation, all of which will be described in much further detail in Chapter 3, enhances how clustering algorithms are able to perform on single-cell data. We specifically found clustering to be enhanced when we incorporate this regulatory network of interactions and perform propagation with genes that are known to be regulators. This makes sense because compared to genes that are solely targets, genes that are also regulators interact with many more genes in a number of different ways, leading there to be much more interaction information

available for regulators. To add on to this, thinking back to the dropout issue with most existing clustering algorithms for single-cell data mentioned in the previous paragraph, our network propagation step uses the regulatory network to enhance the information on the expression of each gene in each cell based on the way it interacts with other genes. This leaves us with many less dropouts as the regulatory network we use has information on most of the genes in the datasets we found, which is another reason as to why our methods lead to better clustering.

2 Related Works and State of the Art

There are five main strategies that existing algorithms developed to cluster single-cell gene expression data employ, and they all have advantages and drawbacks depending on what is trying to be accomplished[22]. The first of these is to reduce the dimensionality of the data before performing cluster analysis[22]. This category includes some well-known single-cell clustering algorithm methods such as SC3, CIDR, pcaReduce, SEURAT2, SIMLR, and SHARP[8,10,16,24,25,33]. Typically, in order to reduce the dimensions of the input data, these methods apply dimension reduction techniques, such as PCA, t-SNE, and UMAP to obtain a lower-dimensional representation of the data, and then partition the cells using established clustering algorithms like kmeans. The main drawback that needs to be kept in mind when using these methods is that they tend to be sensitive to sequencing platforms and dropouts, and the quality of clustering results can vary greatly.

The second category that these single-cell clustering algorithms can fall into is ones that iteratively search for hierarchical structures over both cells and genes[22]. This category includes well-known methods such as BackSPIN, SAIC, and Panoview[6,27,30]. These methods attempt to iteratively divide cells and genes into sub-groups, and then compare those subgroups to one another. The main drawback with these algorithms are that they require excessive computational power and tend to overestimate the number of cell types.

The third category is community detection algorithms[22]. These are some of the most commonly utilized single-cell clustering algorithms in general, and include ones like Louvain, Leiden, SEURAT3, SCANPY, and Monocle[2,3,19,21,26]. These methods embed community detection algorithms in their analysis pipeline by first converting single-cell RNA-sequencing data into networks, and then partitioning those networks using community detection algorithms. One thing to note about this particular category of algorithm in relation to what we are trying to do is that although these methods sound similar to ours since they create their own networks rather than using outside ones based off of other factors, the quality of these strongly depends on how their network is created, and although they can produce good results, they often overestimate the number of cell communities.

The next category of single-cell algorithms is cluster ensemble algorithms, which includes SAFE, SAME, and Sc-GPE[7,22,28,32]. These methods aggregate results from other algorithms, so they don't do anything new necessarily, but they try to combine complementary methods in order to limit drawbacks as much as possible. The main drawback is that these don't scale well for large datasets, which makes sense as they are having to go through the computational complexities of multiple algorithms. Another thing to note with these are that evaluating can be tough since each clustering method will have its own results, so combining them is not straight-forward at all.

The last category of single-cell clustering algorithms are ones that use functional information, which is usually in the form of regulatory networks. There are a number of algorithms that infer regulatory networks and other functional information from single-cell data, and most of these fall under the third category

menetioned above. None of these, however, use known interactions to improve how the clustering algorithms perform - they simply infer networks when given data, and then go through their dimensionality reduction and other processing techniques with those inferred networks. We, on the other hand, use a list of predefined, validated regulatory interactions in order to find trends in the genes. Those trends are then used to reduce dropouts and to propagate gene expression values, which in turn leads to enhanced clustering of single-cell data.

3 Methods

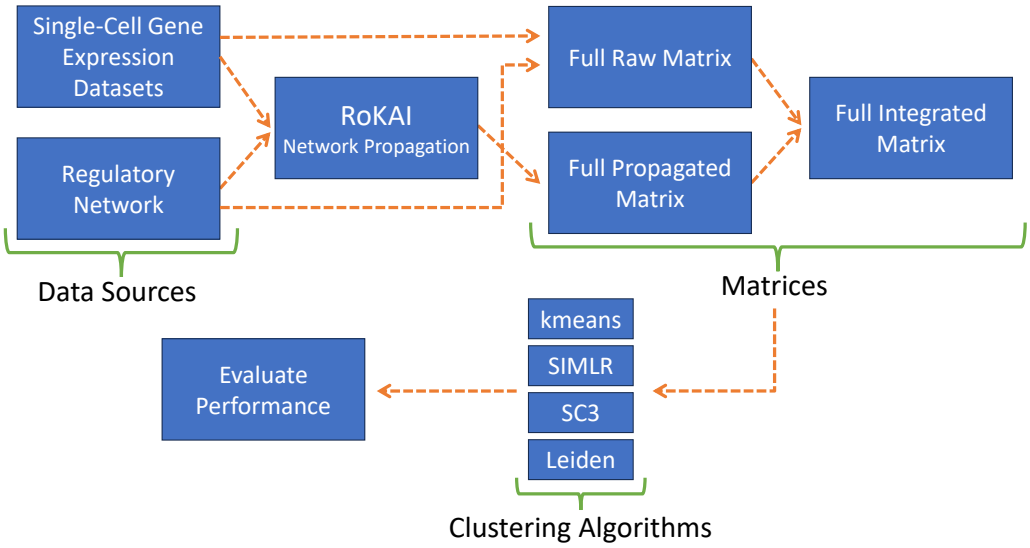


Figure 3.1. A Summary of the Overall Process Flow from Beginning to End

3.1 Gathering, Ingesting, and Cleaning the Data

The whole process the data goes through from beginning to end is outlined in Figure 3.1. As will be discussed in more detail in Section 4.1, the six single-cell gene expression datasets we use all come from one reliable source[20]. Tian et al., the creators of the datasets, store them all as .csv files in a GitHub repository, so that is where we downloaded them from[20]. For each of these .csv files, we load it into RStudio, then clean it up[15].

As can also be seen under "Data Sources" in Figure 3.1, completely separate from these six datasets, we also use a regulatory network so that we can know how the genes in the datasets interact with one another[11]. This regulatory network, which the creators call RegNetwork, is an integrated database of transcriptional and post-transcriptional regulatory networks in human and mice[11]. Essentially, they created this regulatory network by building a knowledge-based database of gene regulatory interactions for human and mice[11]. This knowledge base was built by collecting and integrating the documented regulatory interactions among transcription factors, microRNAs, and target genes from 25 carefully selected databases[11]. I explain more about specific numbers of genes in the regulatory network in Section 4.1, but what matters here is that once the regulatory network is loaded into RStudio along with the six datasets, we then only select genes in the datasets that are also in the regulatory network as well since otherwise, we would not know how a gene interacts with the others, and that is vital information we want to use to enhance clustering later on in the process.

3.2 Network Propagation with RoKAI

At this point in the process, each dataset has been cleaned, we have brought in and cleaned the regulatory network, and we have selected to only keep the genes that are also in the regulatory network, no matter what side of the network they are on (regulator, target, or both). Next, we perform some exploratory data analysis (EDA) on each cleaned and transformed dataset to make sure everything looks relatively normal, and then start to prepare for our next step: network propagation.

In our specific case, we incorporate a network propagation algorithm known as RoKAI[29]. RoKAI was created by Yilmaz et al. as a network-based framework to enhance the reliability of kinase activity inference[29]. To do this, RoKAI combines several sources of functional information so that it can capture coordinated changes in the ways proteins communicate with one another[29]. Since the signaling information that we have is in the form of the regulatory network mentioned in Section 3.1, that is the information we used in conjunction with the expression datasets from Tian et al. also mentioned in Section 3.1 as inputs into RoKAI's network propagation algorithm[20,29]. To elaborate on this some more, if we let n represent the total number of genes in the regulatory network, RoKAI's network propagation algorithm takes in two matrices: the first matrix, denoted as **b**, is an n x 1 matrix that represents an individual cell's expression, and the second matrix, denoted as **C**, is an n x n matrix that represents the gene-to-gene regulatory interaction network[29].

Before we can create **b** and **C**, we first need to create a dictionary for all the possible genes that we are working with. This was already mentioned in Section 3.1, but remember that we already selected only genes that are also in the regulatory network in each of the six datasets. This means that the union of the regulators and the targets in the regulator network are the gene names in this dictionary, so each name will be assigned to an index. Then, to create each matrix for an individual cell's expression, **b**, we parse through each gene that is in the dictionary, and if that gene is expressed in that cell, we copy its expression under the index respective to that gene name in the dictionary; if that gene is not expressed in that cell, we put a "0" under the gene's respective index and move on to the next gene[29]. Next, to create the matrix for the gene-to-gene regulatory interaction network, **C**, we first

create an n x n matrix that is all 0s, and then for each row in our regulatory network that represents a gene-to-gene interaction, we switch the "0" at the coordinates of the indices of the two gene names in the dictionary to a "1"[29]. We do this to denote that there is an interaction between the two genes that are represented by those coordinates, and we also switch the "0" to a "1" for the inverse coordinates as those also represent an interaction between the same genes[29]. As an example, let us say the first row of the regulatory network has gene A in the first column and gene B in the second column. In this example, gene A would match up to the index 1 in the dictionary and gene B would match to the index 2, and so we would switch the "0" to a "1" to represent this interaction at (1,2) and (2,1) in the gene-to-gene regulatory interaction network, **C**. Note that although the network is directed (with edges from the regulators to their targets), this matrix is symmetric since targets report on the expression/activity of their regulators, while regulators also report on the expression/activity of their targets.

Now that we have created the **b** and **C** matrices, the last matrix involved in this network propagation process is denoted as **v**, and it is a 1 x n vector that represents the cell's inferred activity[29]. These three matrices can then be represented by the linear system $\mathbf{Cv} = \mathbf{b}$, and using linear algebra, how **v** is computed when **C** and **v** are known can be seen in Equation 3.1:

$$\mathbf{v} = (\mathbf{C}^\top \mathbf{C})^{-1} \mathbf{C}^\top \mathbf{b} \tag{3.1}$$

This process is repeated for each cell in the dataset as **b** will be different for each cell. **C**, on the other hand, will stay the same each time since it represents the regulatory network and we are only using one. Once this process is repeated for

each cell in the dataset, we have as many **v** vectors as there are cells in the dataset. These **v** vectors, however, are each 1 x n in dimension, which is essentially a row, so to get the full Propagated data matrix, we simply row append all of the **v** vectors together. To get the full Raw data matrix that is comparable to this Propagated matrix, we do this same process with all the **b** matrices.

3.3 Clustering and Matrix Comparisons

We now have a full Raw matrix and a full Propagated matrix for each dataset. We keep each of these two full matrices, but for each one, we also create a copy where only the regulators are selected. This way, we can see the true effect of the propagation when we run the matrices through all the different clustering algorithms we decided to use, which is the next step. These four clustering algorithms are kmeans, SIMLR, SC3, and Leiden[8,15,21,25]. We started with kmeans because it is the simplest form of clustering, and wanted to compare it to the other methods that were created more specifically for clustering single-cell RNA-sequencing data[15]. SIMLR was second because although it was made for clustering single-cell data, it's much simpler than most of the other methods, so again we felt as though it was a good place to get basic metrics, and it is certainly a step up from kmeans[25]. Next is SC3, and this one we used because we had seen it more widely used than most others, so we wanted to be consistent and to again give a good basis for comparison[8]. Lastly is Leiden, and if you think back to Chapter 2, we had mentioned Leiden and that it falls into the "community detection" category of clustering algorithms[21]. This is different from SC3, which falls into the "reduce the dimensionality before

performing clustering" category, and so because they are in different categories and we wanted a couple of very reliable clustering algorithms that were meant for single-cell data, that is why we went with these[8]. Once we had chosen the four different clustering algorithms we wanted to put the four matrices through, we did so, and then recorded some performance metrics which will be described in Chapter 4.

4 Experimental Results and Discussion

4.1 Experimental Setup

We used six diverse datasets in our computational experiments to make sure that our results were not solely due to the specific dataset we used. Although all six datasets came from the same source, they were all designed in different ways, and have a varying number of ground truths, which we explain below[20]. According to Tian et al., these datasets were all created as "gold-standard benchmark datasets" that are designed to be used for the exact purpose we need them for: to compare the performance of different clusterings of single-cell RNA-sequencing data[20]. To create these datasets, Tian et al. designed a series of experiments using mixtures of cells in up to five cancer cell lines[20]. These cell lines are from human lung adenocarcinoma, and they were cultured separately[20]. They were made by mixing single cells from each cell line in equal proportions, with libraries generated using three different protocols: CEL-seq2, Drop-seq with Dolomite equipment, and 10x Chromium[12,20,31]. As mentioned earlier, we specifically used 6 of these datasets: one generated using CEL-seq2 from three cell lines (let us refer to this one as CEL-seq2-3gt), one generated using Drop-seq from three cell lines (let us refer to this one as Drop-seq), one generated using 10x Chromium from three cell lines (let us

refer to this one as 10x), and three more generated using Drop-seq, though these from five cell lines (let us refer to these as CEL-seq2-3gt-p1, CEL-seq2-3gt-p2, and CEL-seq2-3gt-p3) [12,20,31]. The details of these datasets can be seen in Table 4.1 below:

Dataset Name	**# of Cells**	**# of Genes**	**# of Clusters**
CEL-seq2	274	27,983	3
Drop-seq	225	15,096	3
10x	902	16,431	3
CEL-seq2-5gt-p1	297	15,475	5
CEL-seq2-5gt-p2	307	14,011	5
CEL-seq2-5gt-p3	305	13,371	5

Table 4.1. **Single-Cell Gene Expression Datasets.** Datasets used along with their respective number of cells, genes, and clusters.

As mentioned in Section 3.1, in addition to these six datasets, we also use a regulatory network so that we can know how the genes in the datasets interact with one another[11]. This regulatory network has 20,737 genes between both regulators and targets[11]. As also mentioned in Section 3.1, one of our uses of the regulatory network is to select only the genes in the expression datasets that are also in the regulatory network as otherwise, we would have no information on those genes, so we could not say confidently whether they act as regulators, targets, or both. Table 4.2 below shows details of how the six single-cell gene expression datasets intersect with the genes in the regulatory network.

Since we used 6 different datasets with a number of different matrices based off each one and then put all of those matrices through 4 different clustering algorithms, we needed a way to evaluate the performance each time we put a matrix through a clustering algorithm. This is where Adjusted Mutual Information (AMI)

Dataset Name	**Total # of Genes in RegNetwork**	**# of Regulators**	**# of Targets**
RegNetwork	20,737	1,902	20,714
CEL-seq2	11,939	1,090	11,937
Drop-seq	9,451	906	9,449
10x	10,292	974	10,290
CEL-seq2-5gt-p1	9,904	959	9,902
CEL-seq2-5gt-p2	9,439	918	9,437
CEL-seq2-5gt-p3	9,167	906	9,165

Table 4.2. **Regulatory Network Intersection Information.** RegNetwork intersection details with each of the 6 single-cell gene expression datasets regarding total number of genes, number of regulators, and number of targets.

and Adjusted Rand Index (ARI) come in, the two clustering performance metrics we use[14]. AMI is calculated using the following equation:

$$AMI(U,V) = \frac{MI(U,V) - E\{MI(U,V)\}}{max\{H(U),H(V)\} - E\{MI(U,V)\}} \tag{4.1}$$

where U and V are the two sets of labels being compared, *MI(U,V)* is the mutual information between U and V, *H(U)* and *H(V)* are the entropies of U and V, respectively, and *E{MI(U,V)}* is the expected mutual information between U and V under the assumption of independence.

In our case, we had the ground truth of each cell for each dataset, so please note that U and V are the clustering outcomes and their ground truths, respectively. AMI ranges from 0 to 1, where an AMI as close to 1 as possible is desired since an AMI of 0 indicates the ground truths are largely independent to the clustering results, and an AMI of exactly 1 indicates that the clustering results are equal to the ground truths[14]. AMI is an adjusted version of Mutual Information (MI), so not only does it measure the amount of information shared between two sets of data

like MI does, but AMI takes into consideration the expected amount of MI to occur by chance and adjusts itself accordingly[14].

Keeping in mind that Rand Index (RI) is derived using equation 4.2 directly below:

$$RI = \frac{TP + TN}{TP + FP + FN + TN} \tag{4.2}$$

where *TP* is the number of true positives, *TN* is the number of true negatives, *FP* is the number of false positives, and *FN* is the number of false negatives between *U* and *V*, ARI is determined using the following calculation:

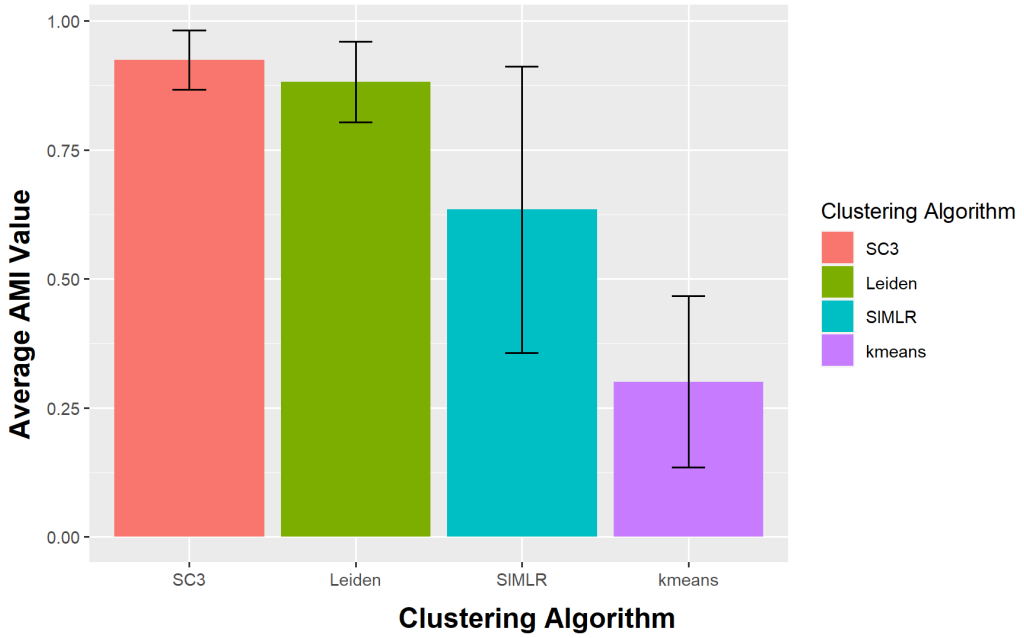
$$ARI = \frac{RI - E\{RI\}}{max\{RI\} - E\{RI\}} \tag{4.3}$$

where *E{RI}* is the expected RI of random clusterings. Contrary to AMI, ARI ranges from -1 to 1, but similar to AMI, a higher ARI is desired; an ARI of exactly 1 indicates perfect similarity between the clustering results and the ground truths, while an ARI of 0 suggests they are as similar as would be expected by random chance, and an ARI of less than 0 signifies that they are less similar than random chance[14]. And again, similar to how AMI differs from MI, ARI accounts for the expected similarities that could occur by random chance while Random Index (RI) alone does not[14].

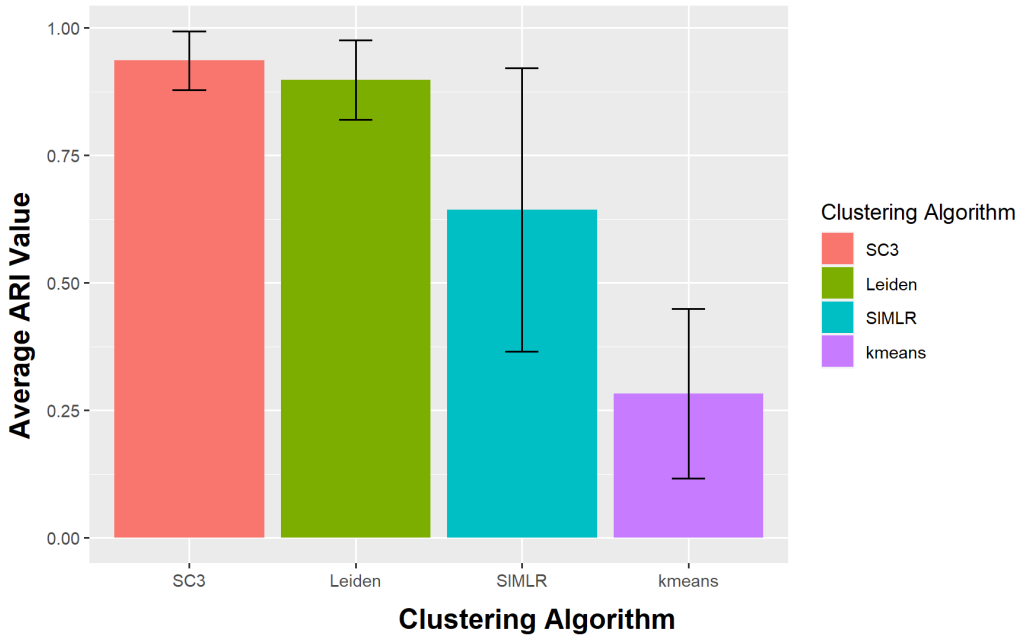
4.2 Benchmarking Clustering Algorithms

After our earlier processes of ingesting and cleaning the data and network propagation with RoKAI, we now have four matrices that we're working with: a full raw

matrix, a full propagated matrix, a raw matrix with regulators only, and a propagated matrix with regulators only. We also have four different clustering algorithms to compare results for each of the matrices: SC3, Leiden, SIMLR, and kmeans.

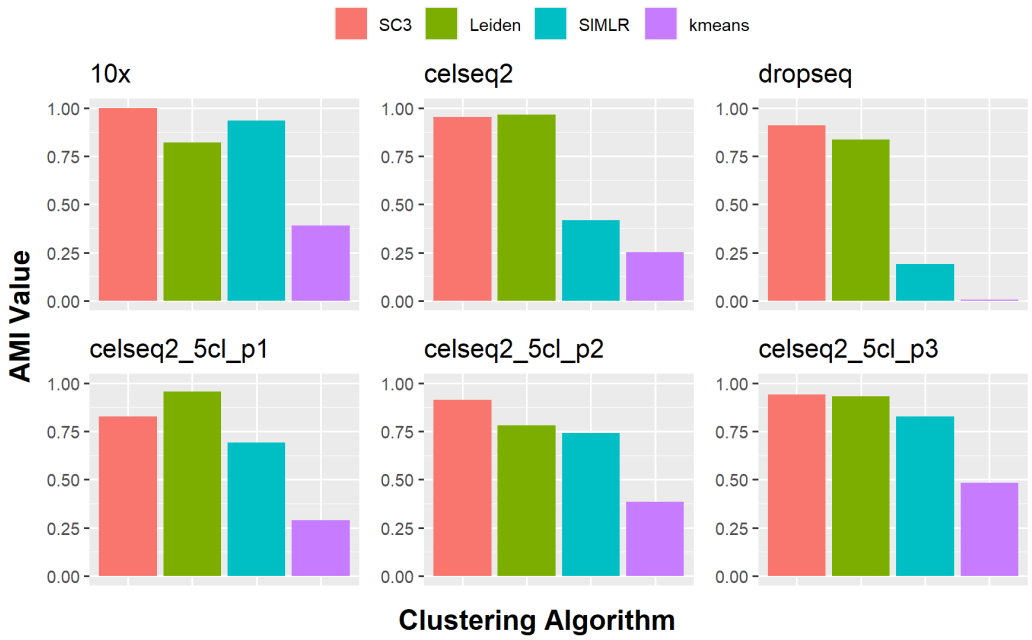


(a) Average AMI of SC3, Leiden, SIMLR, and kmeans Clustering Algorithms on all 6 Datasets for the Full Raw Data Matrix

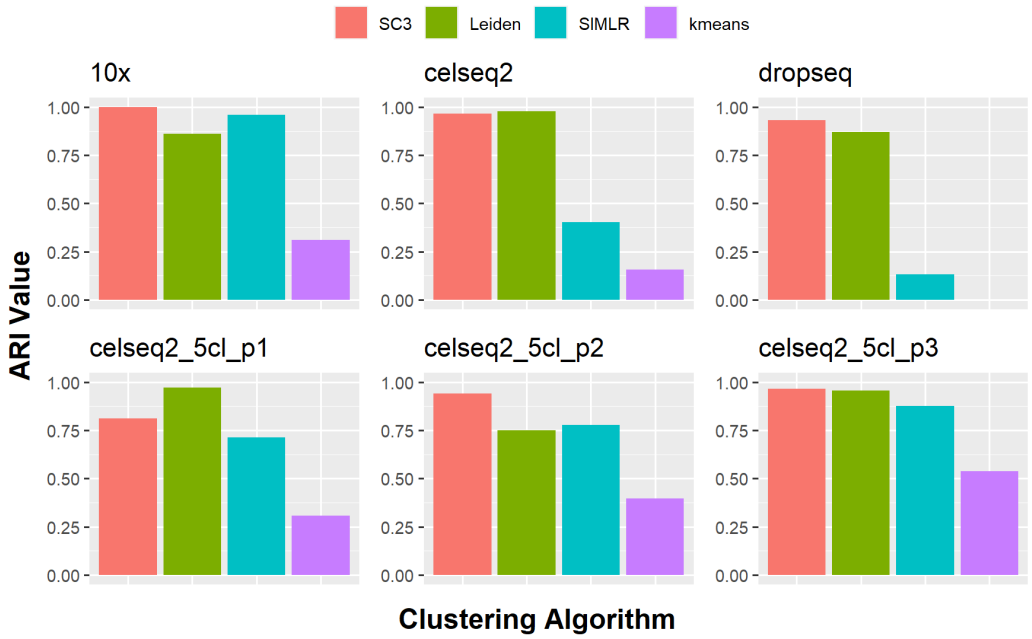


(b) Average ARI of SC3, Leiden, SIMLR, and kmeans Clustering Algorithms on all 6 Datasets for the Full Raw Data Matrix

Figure 4.1. The Performance of All Clustering Algorithms on Raw Data Matrices



(a) AMI of SC3, Leiden, SIMLR, and kmeans Clustering Algorithms on Each of the 6 Datasets for the Full Raw Data Matrix



(b) ARI of SC3, Leiden, SIMLR, and kmeans Clustering Algorithms on Each of the 6 Datasets for the Full Raw Data Matrix

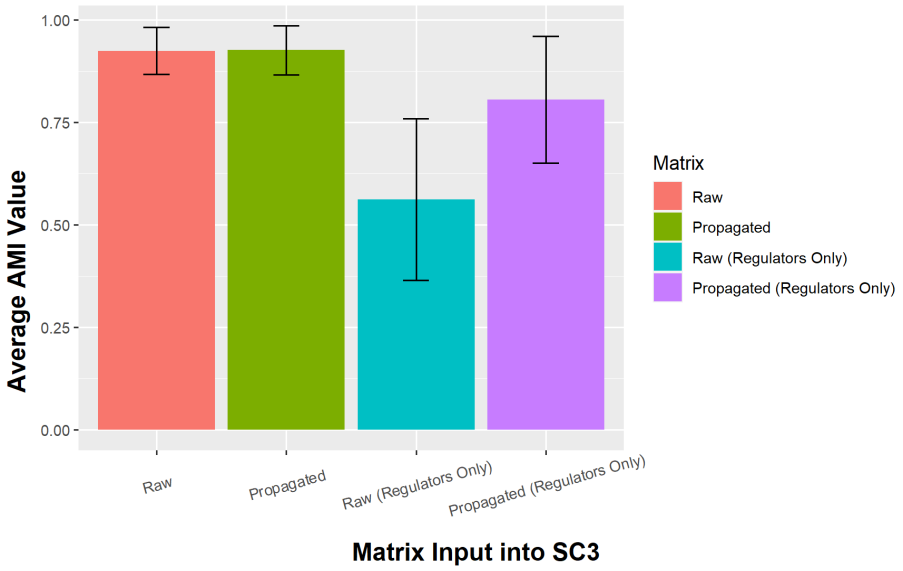
Figure 4.2. Comparing the Performance of the 4 Clustering Algorithms on Each Raw Data Matrix

The results of running the four matrices through the four clustering algorithms can be seen in Figures 4.1 and 4.2. Figure 4.1 clearly shows that based on average AMI and ARI, SC3 and Leiden are much more accurate and have much less uncertainty when compared to SIMLR and kmeans. Figure 4.2 goes on to further show this across each of the 6 individual datasets - for five out of the six datasets for both AMI and ARI, SC3 and Leiden far outperform SIMLR and kmeans. After looking at these figures, it is clear to see that SC3 and Leiden consistently perform the best based on Average AMI and Average ARI values. Because SC3 and Leiden are clearly the most reliable and consistent of these clustering algorithms, we decided to focus on just their metrics for the next couple of sections as we needed consistent results since we would be creating and comparing other as you will see shortly.

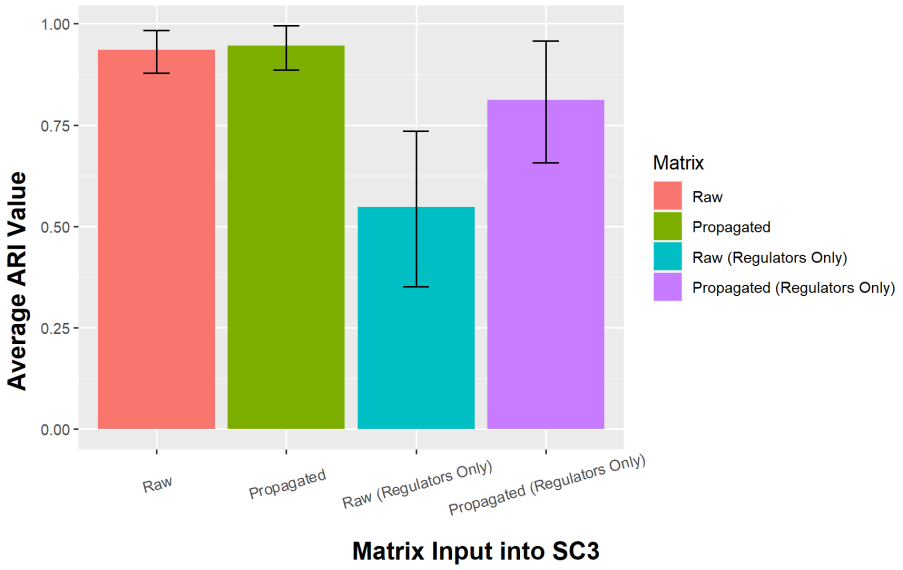
4.3 Using Regulators as Features

After deciding that SC3 and Leiden's metrics are clearly the best to use in terms of both accuracy and precision, we want to keep the clustering algorithms constant so we can see the variations in how each individual matrix is performing. To do this, we look at the metrics of SC3 and Leiden for the six different datasets by the four different matrix inputs.

4.3.1 SC3

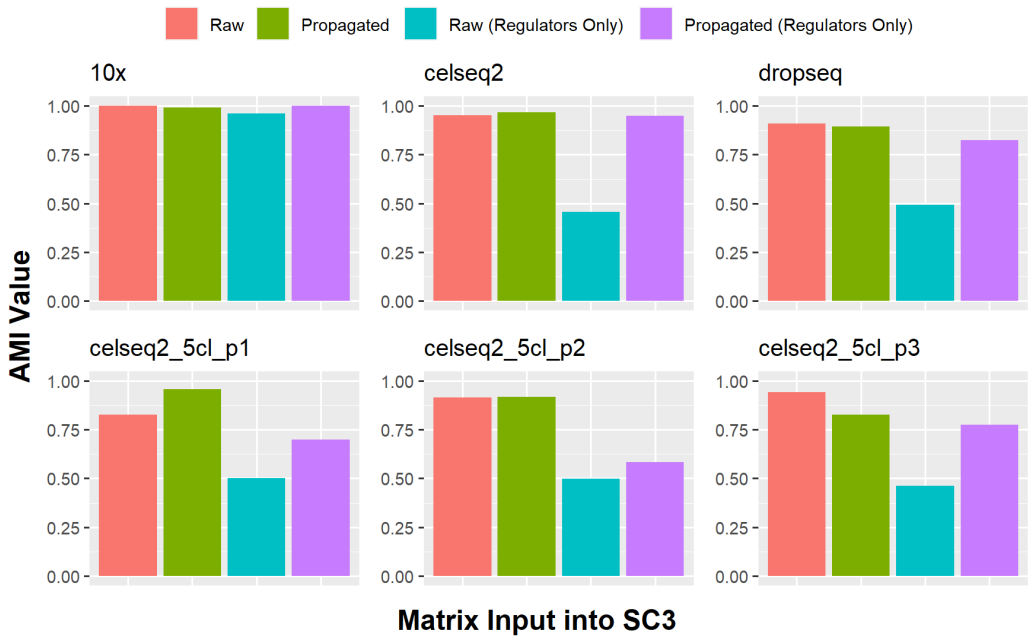


(a) Average AMI of SC3 Clustering on all 6 Datasets for the Raw and Propagated Data Matrices, Both Full and with Regulators Only

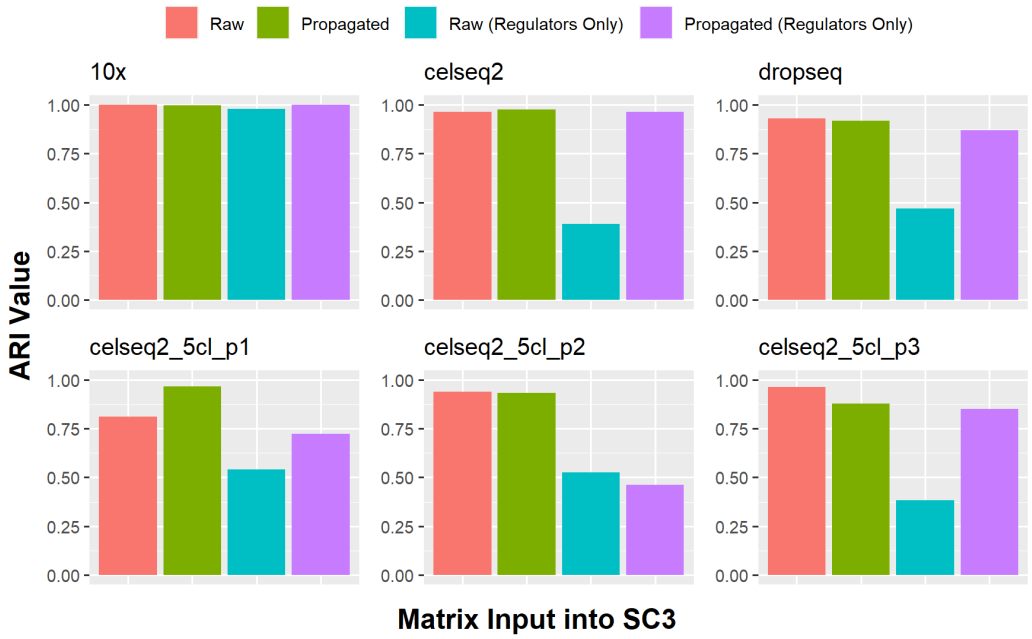


(b) Average ARI of SC3 Clustering on all 6 Datasets for the Raw and Propagated Data Matrices, Both Full and with Regulators Only

Figure 4.3. Summarizing of the Effect of Network Propagation on the Performance of SC3 Clustering



(a) AMI of SC3 Clustering on Each of the 6 Datasets for the Raw and Propagated Data Matrices, Both Full and with Regulators Only



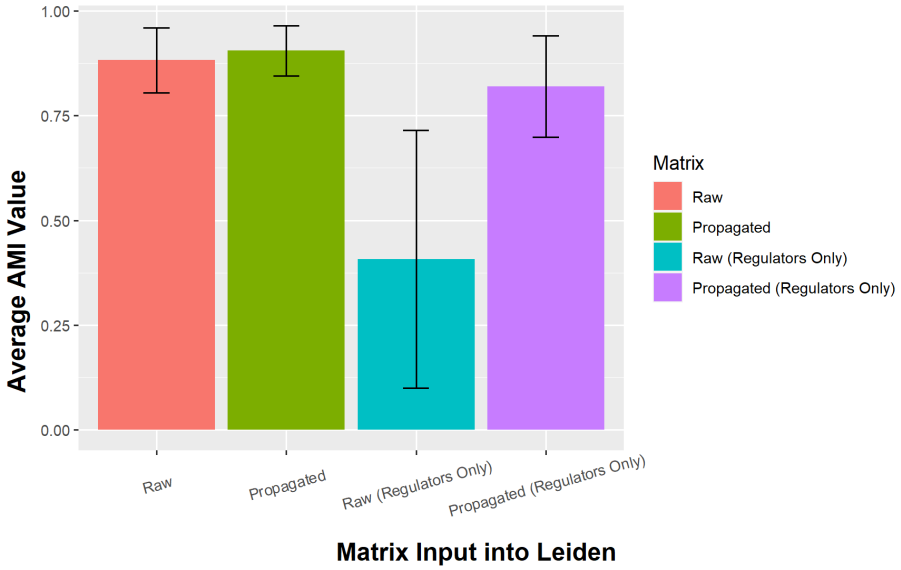
(b) ARI of SC3 Clustering on Each of the 6 Datasets for the Raw and Propagated Data Matrices, Both Full and with Regulators Only

Figure 4.4. The Effect of Network Propagation on the Performance of SC3 Clustering on Each Dataset

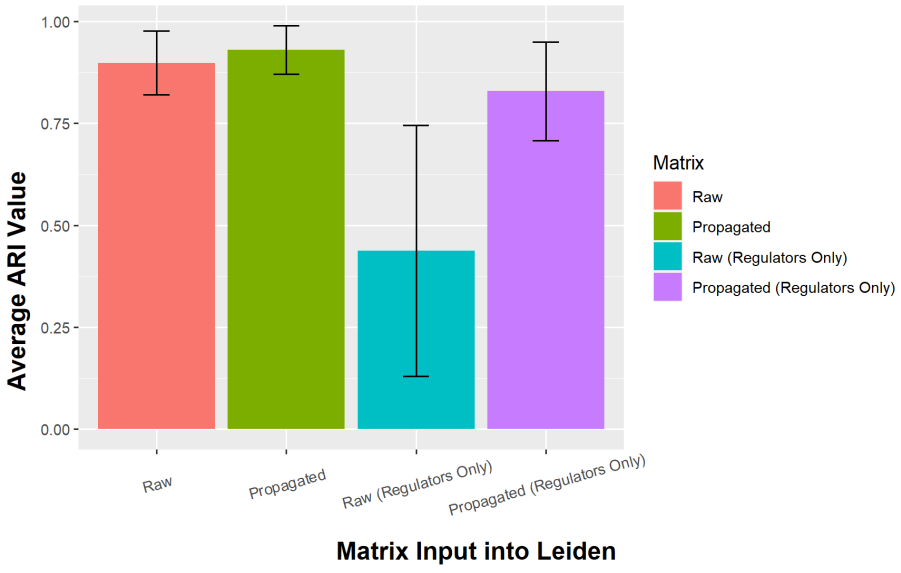
Looking at Figures 4.3 and 4.4 with the different matrix results for SC3 across the datasets, there are a few different things to notice. First, looking at just the Raw and Propagated bars in both figures, they appear to average about the same AMI and ARI across the datasets, meaning that sometimes one will be a bit more than the other, but there is never an extreme difference, and one is not always higher than the other. This means that the propagation on the full matrices seems to make a difference, but that difference is sometimes positive, and sometimes negative. After seeing these results, we wanted to dive deeper into when the propagation makes a positive difference, and when it makes a negative difference. If we are able to find that out, then we can use the propagation only when it helps, and not when it hurts. More about this will come shortly, but first we needed to confirm that results for Leiden, the other reliable clustering algorithm, were consistent with these.

Before that, however, another thing to note in Figures 4.3 and 4.4 are that looking at just the two regulators only bars, the Propagated matrix performed better for six out of six datasets when comparing AMI values and five out of six datasets when comparing ARI values. So even though the two matrices with only regulators performed worse than the two full matrices, it appears as though when only clustering regulators that the propagation is making a positive difference, which is what we wanted to look further into regarding the propagation. Again, after confirming Leiden results are consistent with these, the next section will look into confirming that the propagation makes a positive difference, but only on regulators.

4.3.2 Leiden

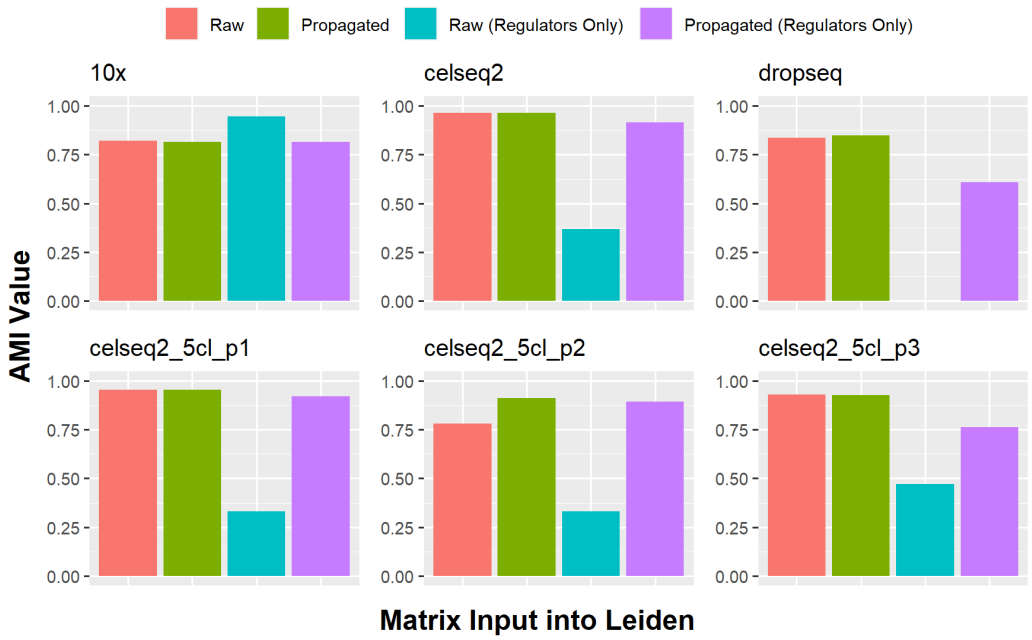


(a) Average AMI of Leiden Clustering on all 6 Datasets for the Raw and Propagated Data Matrices, Both Full and with Regulators Only

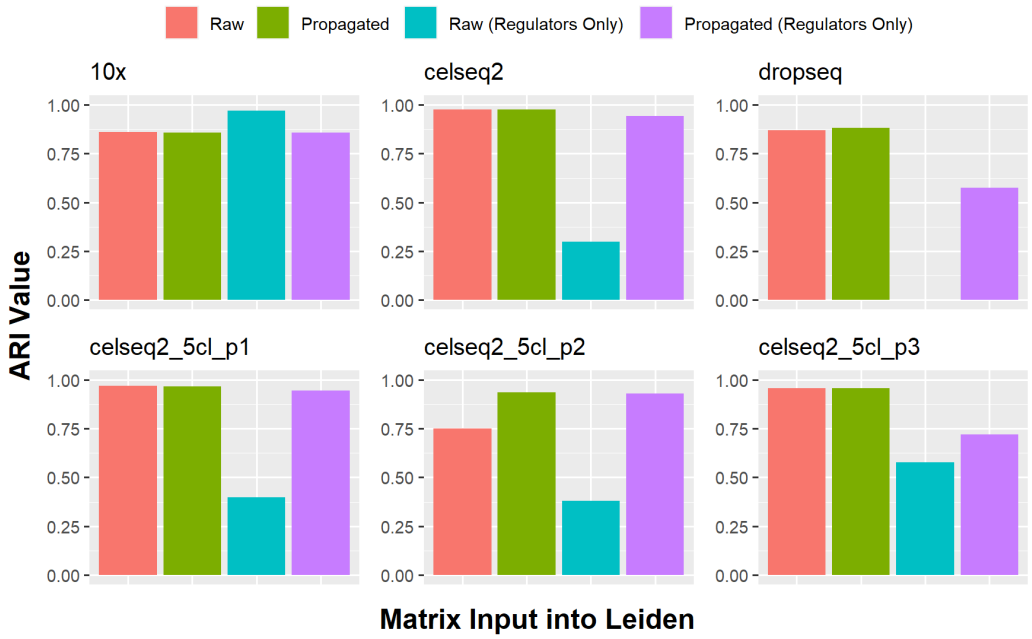


(b) Average ARI of Leiden Clustering on all 6 Datasets for the Raw and Propagated Data Matrices, Both Full and with Regulators Only

Figure 4.5. Summarizing of the Effect of Network Propagation on the Performance of Leiden Clustering



(a) AMI of Leiden Clustering on Each of the 6 Datasets for the Raw and Propagated Data Matrices, Both Full and with Regulators Only



(b) ARI of Leiden Clustering on Each of the 6 Datasets for the Raw and Propagated Data Matrices, Both Full and with Regulators Only

Figure 4.6. The Effect of Network Propagation on the Performance of Leiden Clustering on Each Dataset

The Leiden results for the full raw matrix, the full propagated matrix, the raw matrix with just regulators, and the propagated matrix with just regulators can be seen in Figures 4.5 and 4.6, and as we had hoped, they are very consistent with the SC3 results. In fact, they are actually slightly better. Looking at only the Raw and Propagated bars for the six datasets in Figure 4.6, they are both either the same, or the Propagated is higher. Similar to with SC3, this means that sometimes the propagation is making a positive difference, and sometimes it just is not doing anything. Again, we would like to find out what in the propagation is making the positive difference so we can isolate it and only use that part, but we will talk about that in the next section.

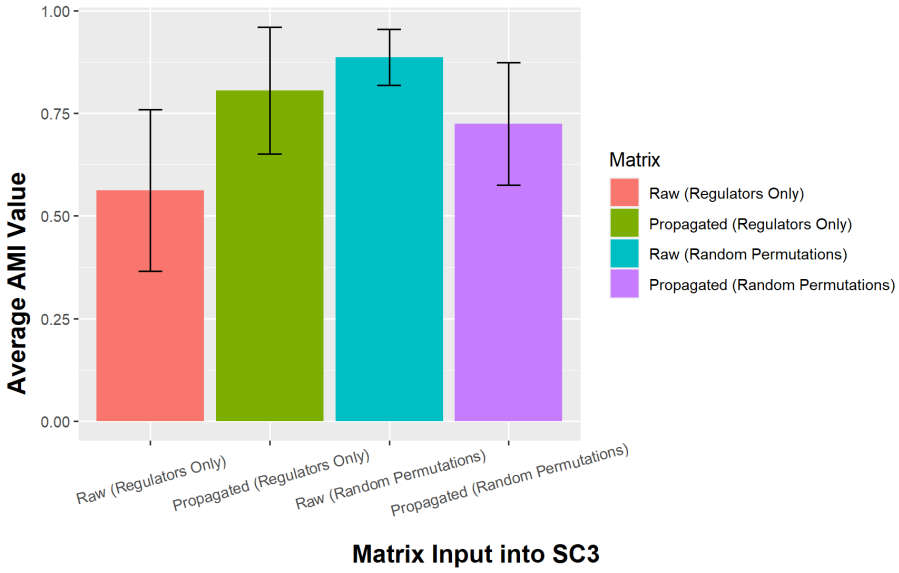
The other thing to notice in Figure 4.6 is that looking at just the two regulators only bars, the Propagated matrix performed better for five out of six datasets when comparing AMI values and five out of six datasets when comparing ARI values. This keeps consistent with our SC3 results pointing to the propagation most likely contributing in a positive way when performed on regulators.

4.4 Assessment of Value Added by Regulatory Networks

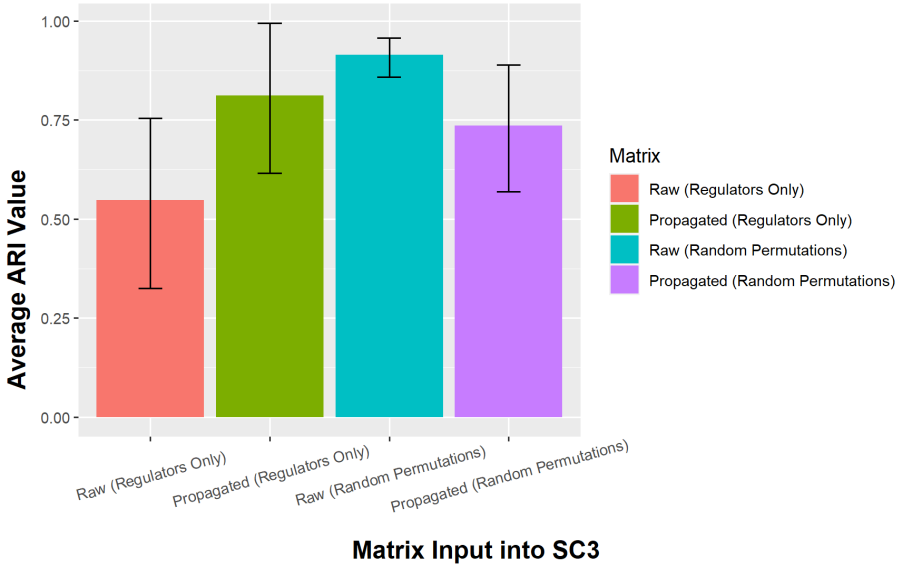
Now that it appears as though the propagation is enhancing clustering with regulators, we want to confirm that this is the case and that our results are not due to simply clustering on a subset of data. To do this, we compare the results for the Raw and Propagated matrices with only regulators to the average results of taking the same size subset of genes from both the full Raw and Propagated matrices,

although each time that subset being a random subset of genes rather than just regulators.

4.4.1 SC3

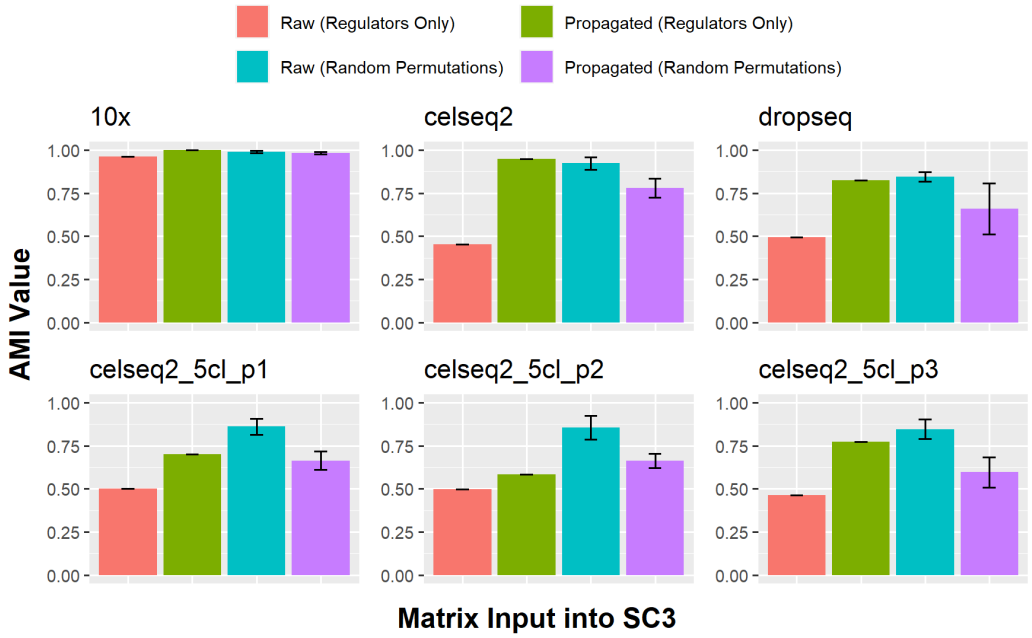


(a) Average AMI of SC3 Clustering on all 6 Datasets for the Raw and Propagated Data Matrices, Both with Regulators Only and Random Permutations of Genes

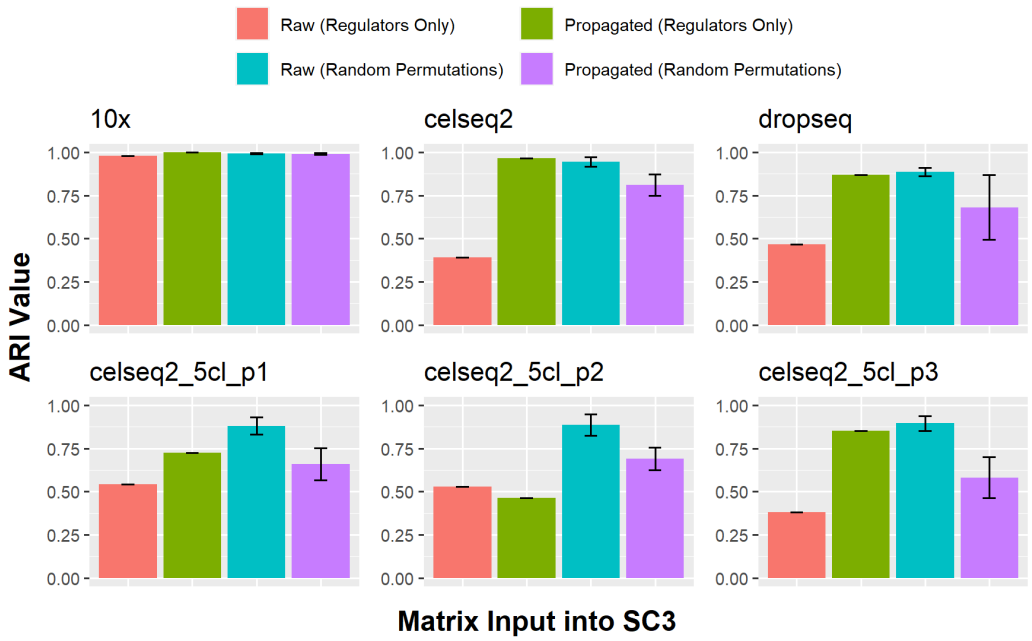


(b) Average ARI of SC3 Clustering on all 6 Datasets for the Raw and Propagated Data Matrices, Both with Regulators Only and Random Permutations of Genes

Figure 4.7. Summarizing SC3 Clustering Performance with Only a Subset of Genes



(a) Average AMI of SC3 Clustering on Each of the 6 Datasets for the Raw and Propagated Data Matrices, Both with Regulators Only and Random Permutations of Genes



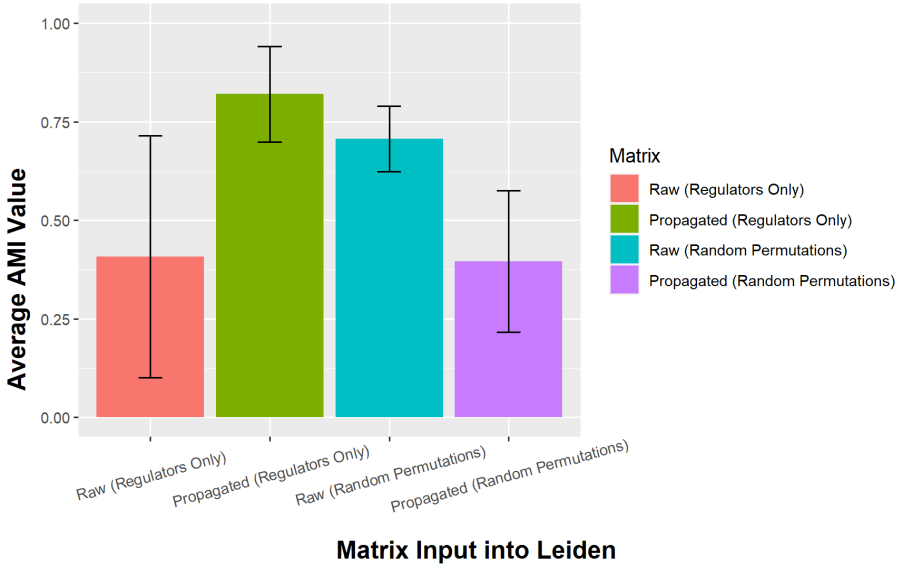
(b) Average ARI of SC3 Clustering on Each of the 6 Datasets for the Raw and Propagated Data Matrices, Both with Regulators Only and Random Permutations of Genes

Figure 4.8. SC3 Clustering Performance with Only a Subset of Genes on Each Dataset

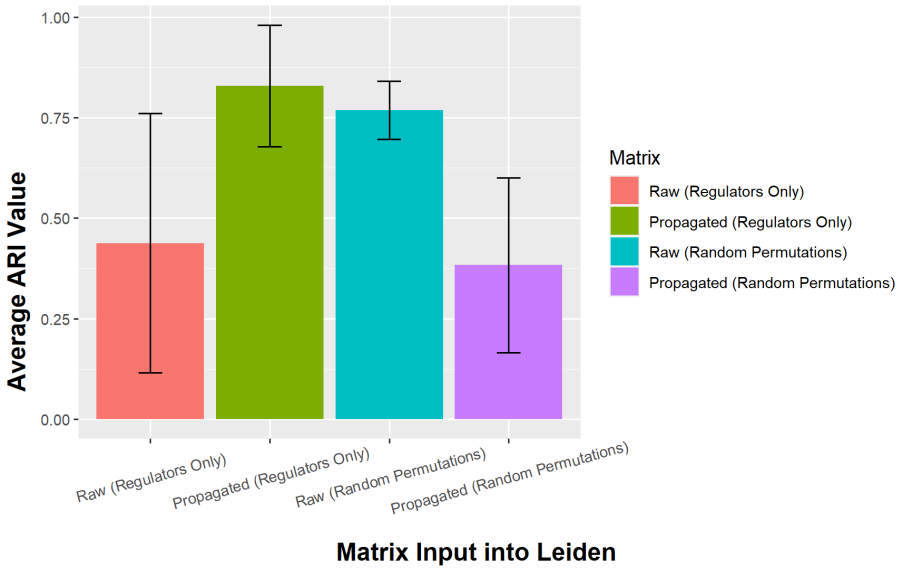
Looking at Figures 4.7 and 4.8, you can see the results of these random permutations compared to the Raw and Propagated matrices with just regulators for SC3 clustering. If you compare the two Raw matrix bars across the different datasets in Figure 4.8, so the first and third bars, you'll see that six out of six times for both AMI and ARI, the average with the random permutations performs better than the Raw matrix with just regulators. This is really interesting to note because in the previous section, we saw that when comparing the full matrices, sometimes the Propagated performs better than the Raw, and sometimes it performs worse. We also saw that the Propagated matrix with just regulators performed better than the Raw matrix with just regulators, leading us to believe that the propagation has a positive impact when performed on regulators. This new information just adds to this hypothesis, leading us to believe that the propagation has negative results when performed on non-regulators, which explains the full Propagated matrix sometimes performing worse than the full Raw matrix.

The other thing to note in Figure 4.8 is that if you compare the two Propagated matrix bars across the different datasets, so the second and fourth bars, you'll see the opposite trend than we saw with the two Raw matrix bars - that five out of six times for both AMI and ARI, the average with the random permutations performs worse than the Propagated matrix with just regulators. Again, this stays consistent with the results we discussed in the previous section, and further proves that propagating regulators leads to enhanced clustering.

4.4.2 Leiden

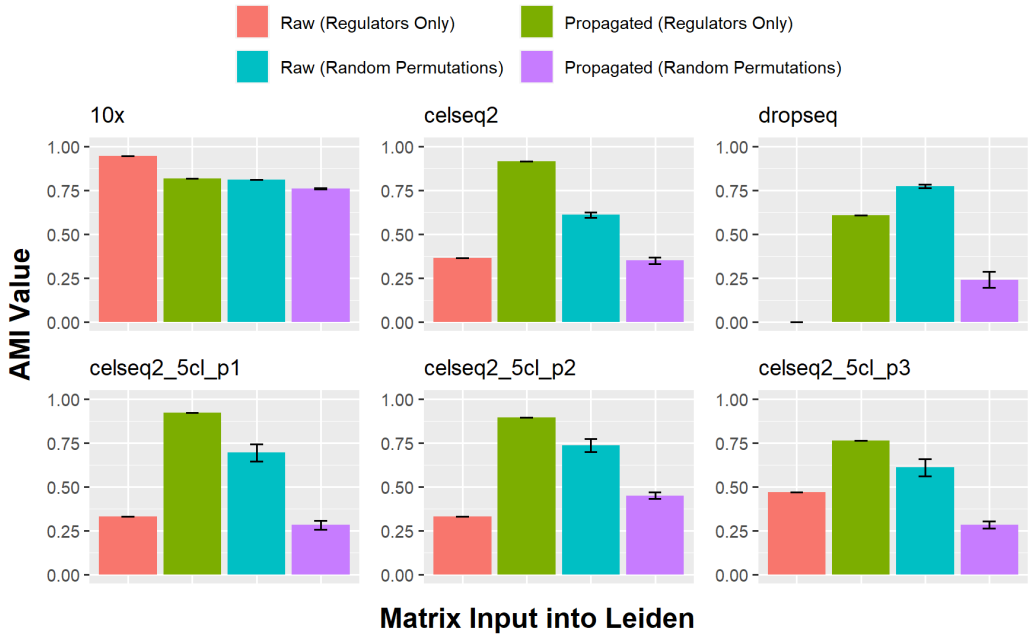


(a) Average AMI of Leiden Clustering on all 6 Datasets for the Raw and Propagated Data Matrices, Both with Regulators Only and Random Permutations of Genes

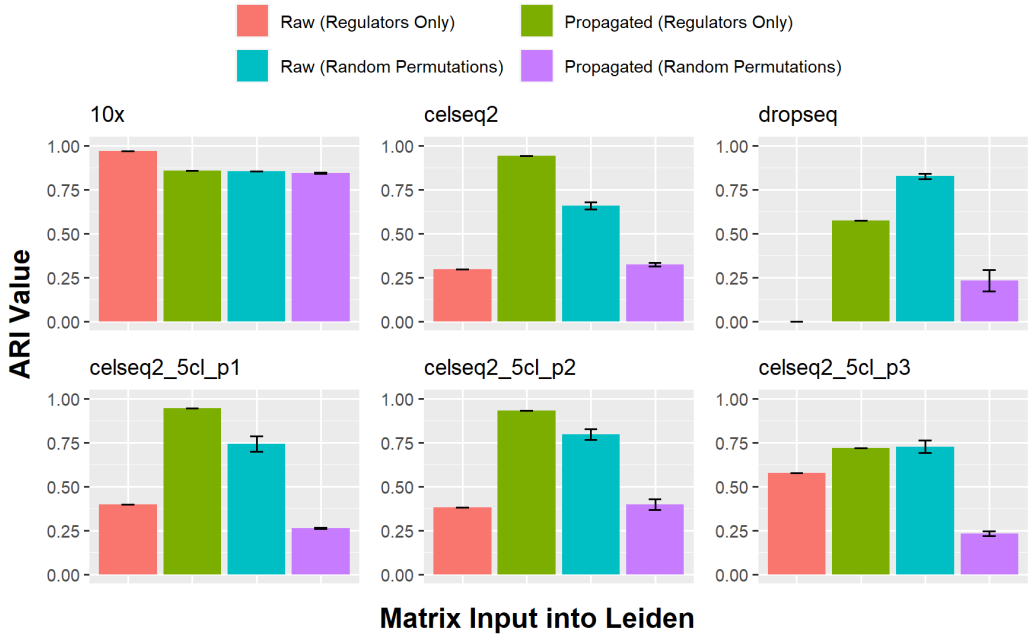


(b) Average ARI of Leiden Clustering on all 6 Datasets for the Raw and Propagated Data Matrices, Both with Regulators Only and Random Permutations of Genes

Figure 4.9. Summarizing Leiden Clustering Performance with Only a Subset of Genes



(a) Average AMI of Leiden Clustering on Each of the 6 Datasets for the Raw and Propagated Data Matrices, Both with Regulators Only and Random Permutations of Genes



(b) Average ARI of Leiden Clustering on Each of the 6 Datasets for the Raw and Propagated Data Matrices, Both with Regulators Only and Random Permutations of Genes

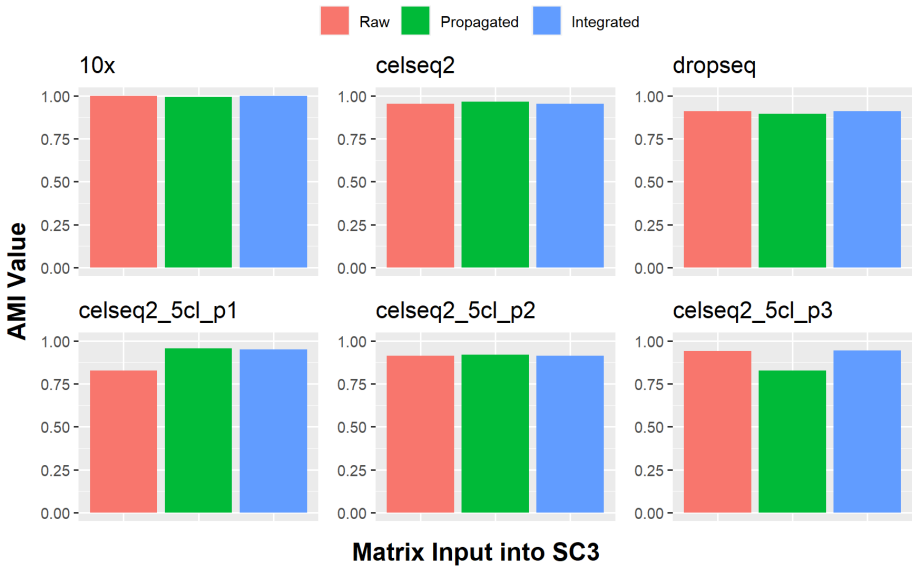
Figure 4.10. Leiden Clustering Performance with Only a Subset of Genes on Each Dataset

The Leiden results look very similar to the SC3 results when comparing the Raw and Propagated matrices with just regulators to the average results of the random permutations of genes. Looking at just the Raw matrix bars in Figure 4.10, you'll see that for five out of six datasets for both AMI and ARI, the average of the random permutations outperformed the regulators only, which is to be expected. Also, when looking at just the Propagated matrix bars, you'll again see the opposite results - that the average of the random permutations is outperformed by the regulators.

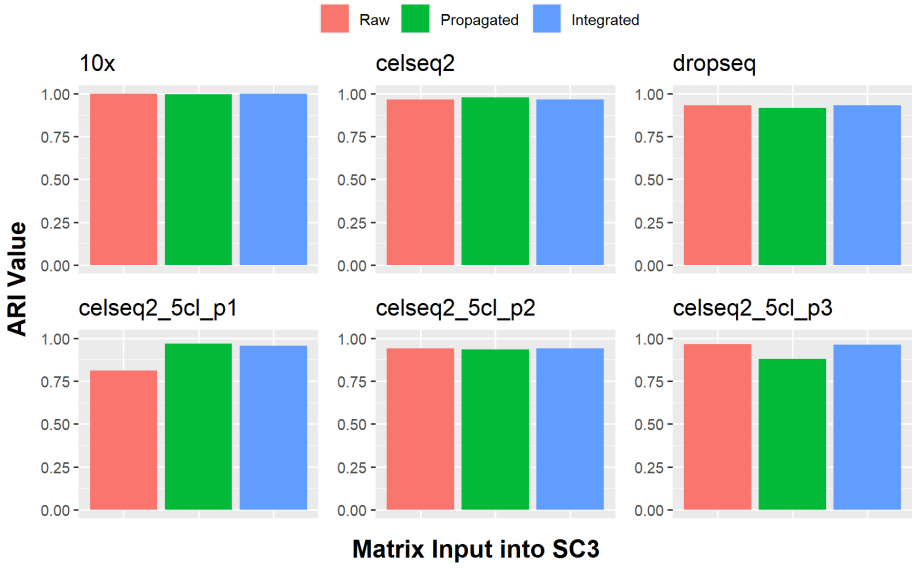
4.5 Comparison of the Integrated Matrix to the Raw and Propagated Matrices

After finding that the propagation appears to enhance clustering for regulators and impair it for non-regulators, we decided to create what we call an "Integrated" matrix. This Integrated matrix is essentially just a hybrid of the full Raw and Propagated matrices: we include the row from the full Raw matrix if the gene is just a target, while if the gene is listed as a regulator in the regulatory network we're using, we use the row from the full Propagated matrix. This can be seen in the top right corner of Figure 3.1. Also, we wanted to bring SIMLR and kmeans results back into this comparison to see how the Integrated matrix performs with less accurate and precise clustering algorithms since not everyone will always be able to cluster using these better techniques such as SC3 and Leiden.

4.5.1 SC3



(a) AMI of SC3 Clustering on Each of the 6 Datasets for the Raw, Propagated, and Integrated Data Matrices



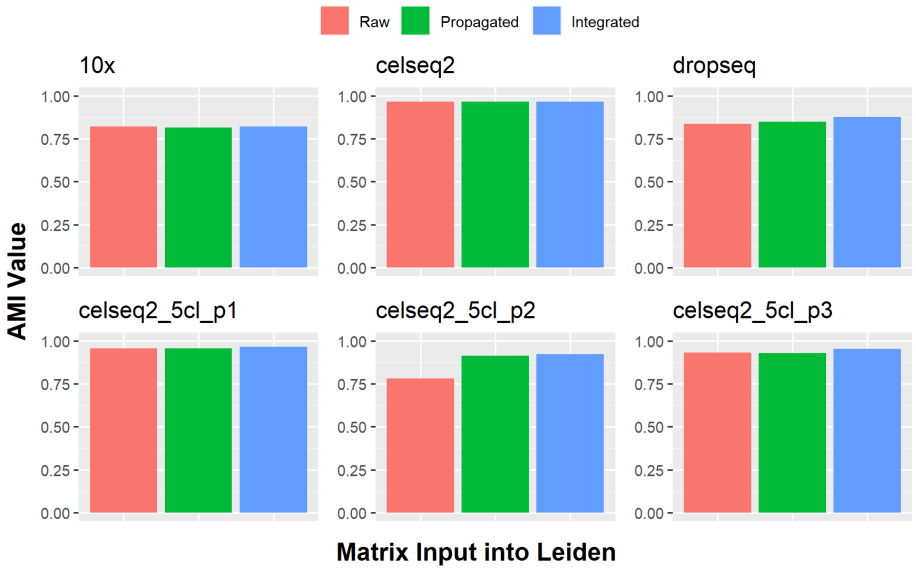
(b) ARI of SC3 Clustering on Each of the 6 Datasets for the Raw, Propagated, and Integrated Data Matrices

Figure 4.11. The Effect of the Integrated Matrix on the Performance of SC3 Clustering

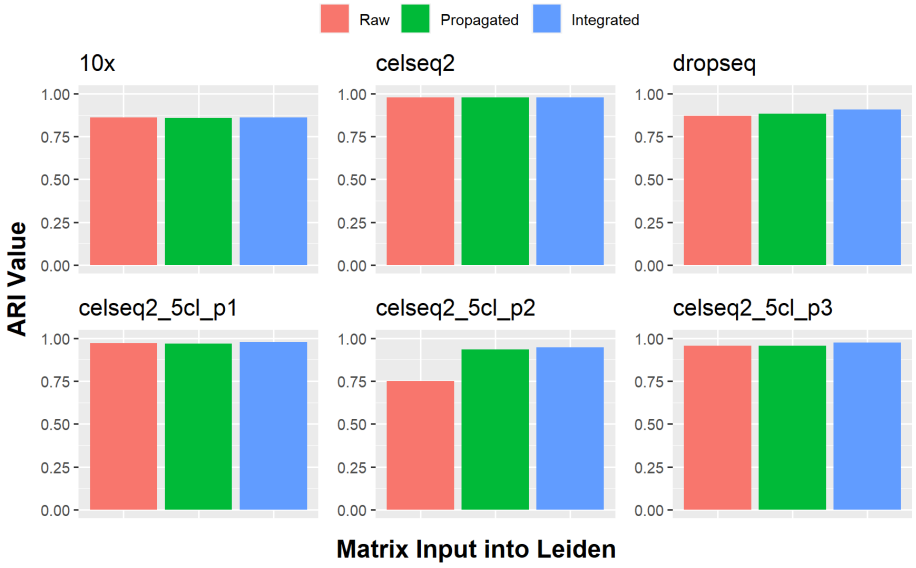
Looking at Figure 4.11, you will see the comparison of AMI and ARI values for SC3 clustering for the three full matrices: Raw, Propagated, and Integrated. The first thing to notice is that with the first two bars for the full Raw matrix and the full Propagated matrix, these are the same as the first two bars when we initially compared the different matrices for SC3 and Leiden. Sometimes the Propagated bar is higher than the Raw, and sometimes the Raw is higher than the Propagated, so this is why we made the third Integrated bar that is a mixture. Since the third bar is a mixture of the better-performing parts of the first two, you'll notice that it is consistent with whichever between the raw matrix and the inferred matrix has a higher AMI and ARI value. Again, this makes complete sense because the Integrated matrix takes rows form the Raw matrix for target genes since the propagation impaired these results, while it takes rows from the Propagated matrix for regulator genes because the propagation enhances these results.

We believe this to be an improvement since clustering with the raw data, which is the state of the art, would get us the Raw matrix results, but there are some instances where the Propagated matrix outperforms the Raw matrix, which is where the Integrated matrix shines.

4.5.2 Leiden



(a) AMI of Leiden Clustering on Each of the 6 Datasets for the Raw, Propagated, and Integrated Data Matrices

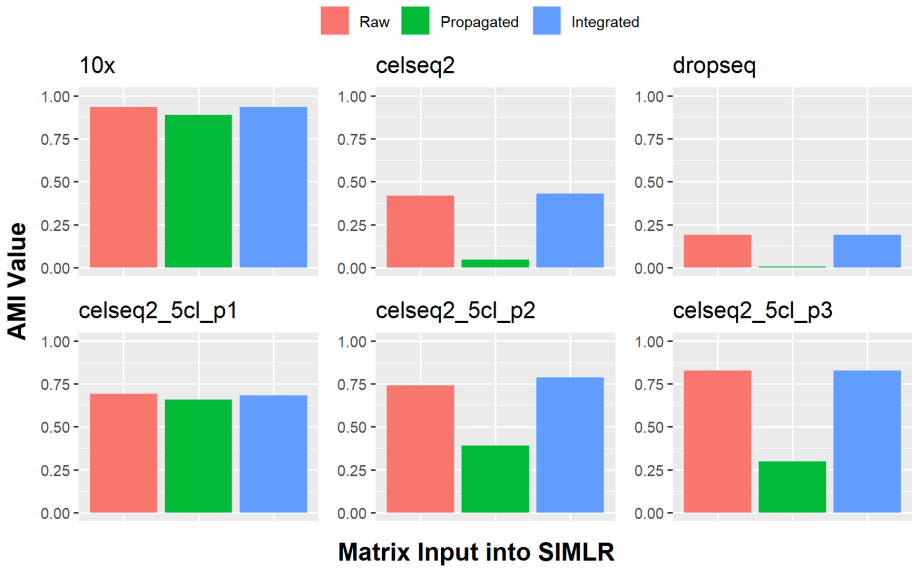


(b) ARI of Leiden Clustering on Each of the 6 Datasets for the Raw, Propagated, and Integrated Data Matrices

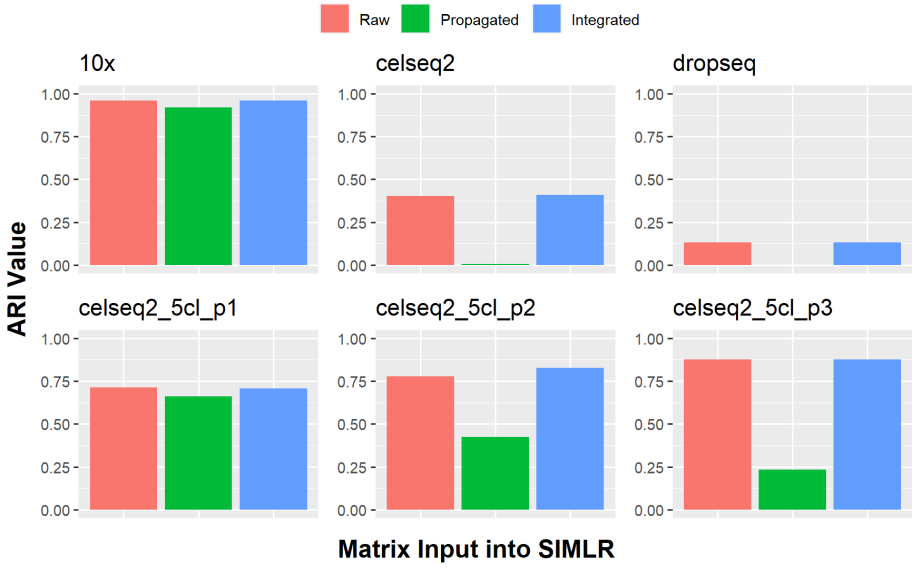
Figure 4.12. The Effect of the Integrated Matrix on the Performance of Leiden Clustering

The Leiden results with three full matrices are consistent with the SC3 results as can be seen in Figure 4.12. Again, if you look at the third bar which is for the Integrated matrix results, it's either as high as the others, or is the highest on its own, which is what we we're looking for.

4.5.3 SIMLR



(a) AMI of SIMLR Clustering on Each of the 6 Datasets for the Raw, Propagated, and Integrated Data Matrices

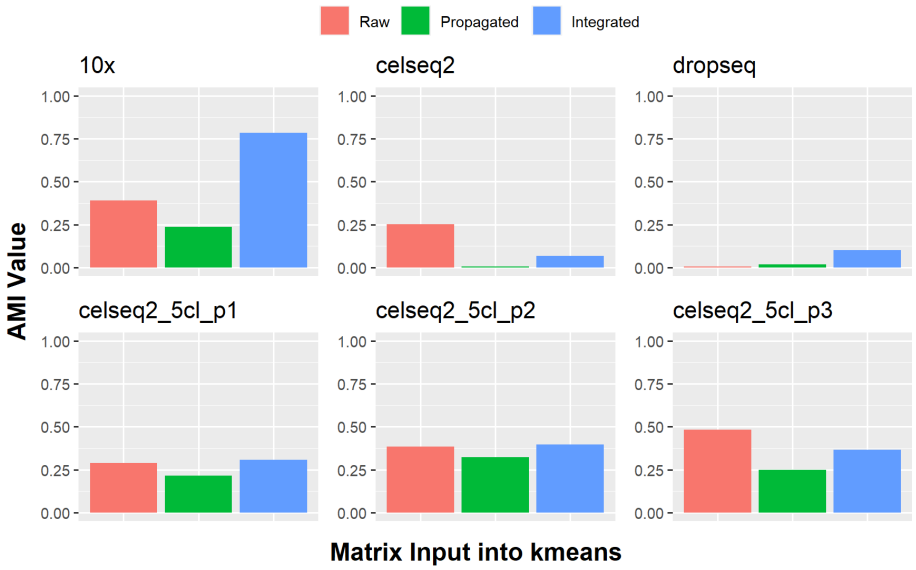


(b) ARI of SIMLR Clustering on Each of the 6 Datasets for the Raw, Propagated, and Integrated Data Matrices

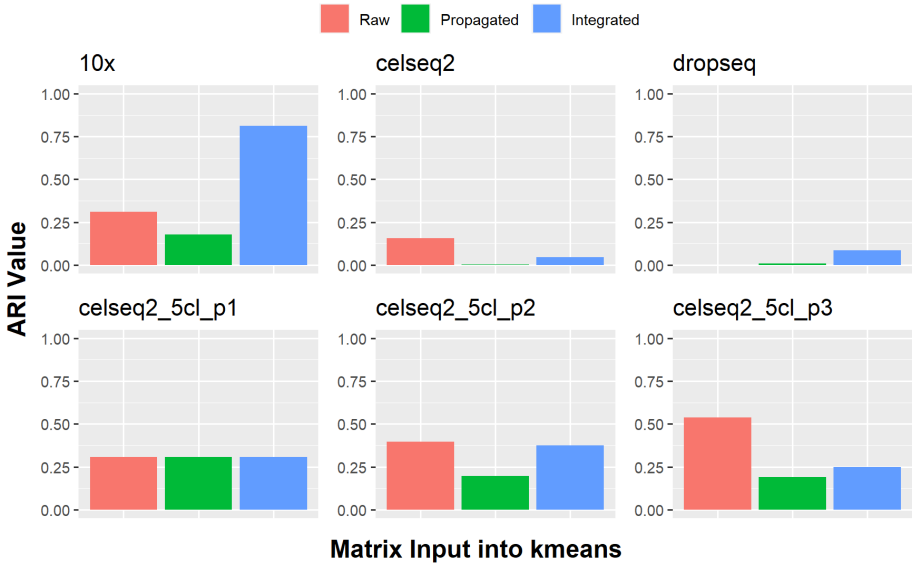
Figure 4.13. The Effect of the Integrated Matrix on the Performance of SIMLR Clustering

The SIMLR results are very interesting, which makes sense since its AMI and ARI results on the raw data alone was not particularly accurate or consistent. Start by looking at just the first two bars in Figure 4.13, so the bars for the full Raw and Propagated matrices. For both AMI and ARI, six out of six times the Raw matrix outperforms the Propagated matrix. Again, don't think too much into these results as SIMLR does not perform amazingly on this data anyways, but the thing to notice here is that even though the Raw matrix outperformed the Propagated matrix every time, it performed about the same as the Integrated matrix because of the way it's setup. So even though in this particular case it doesn't seem like the propagation does any enhancing with this clustering algorithm, the Integrated matrix doesn't impair the results at all, so there is no negative to it here, while it does appear to enhance in other situations.

4.5.4 kmeans



(a) AMI of kmeans Clustering on Each of the 6 Datasets for the Raw, Propagated, and Integrated Data Matrices

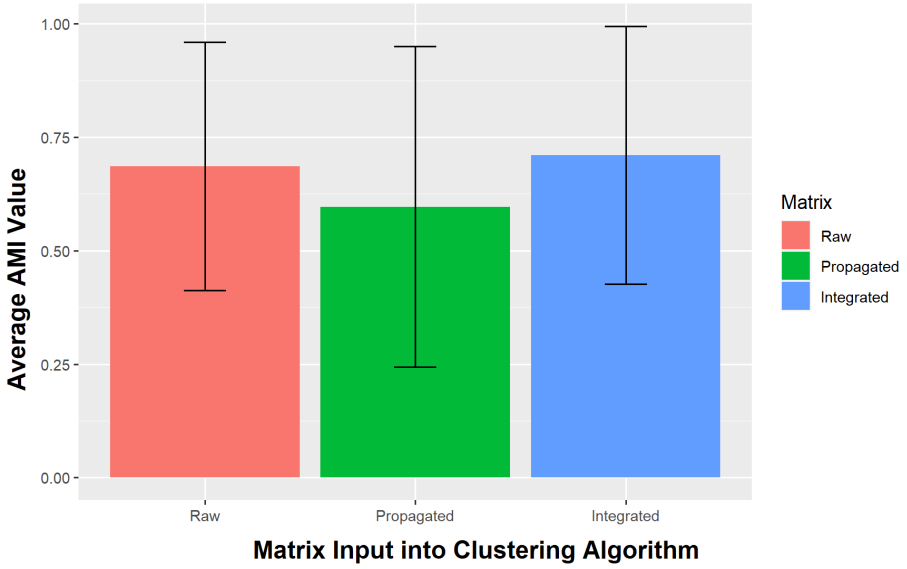


(b) ARI of kmeans Clustering on Each of the 6 Datasets for the Raw, Propagated, and Integrated Data Matrices

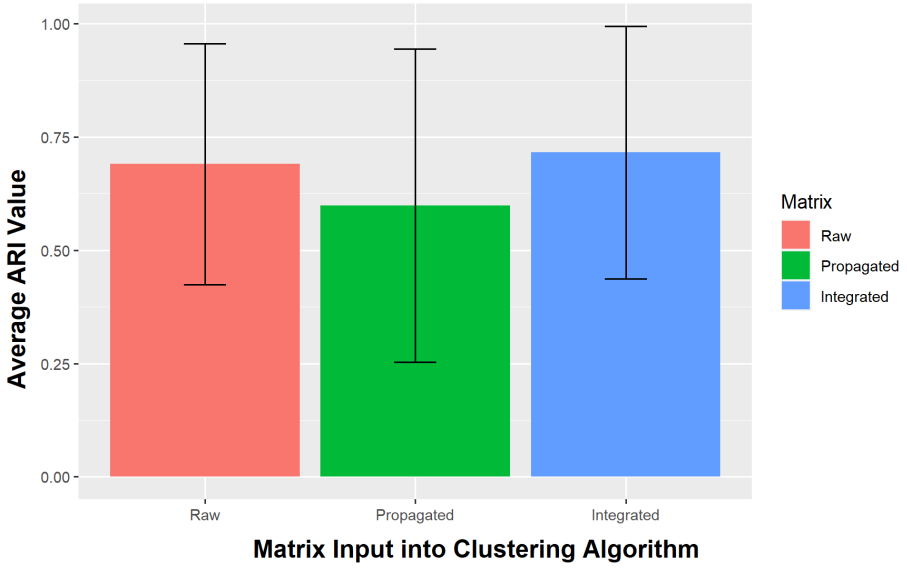
Figure 4.14. Summarizing the Effect of the Integrated Matrix on the Performance of kmeans Clustering

The kmeans results in Figure 4.14 are very similar to what we saw with the SIMLR results. This means that when comparing the bars for the Raw and Propagated matrices, the Raw outperforms the Propagated each time. When comparing the Raw bar to the Integrated bar, however, the Integrated bar looks to be about the same as the Raw bar on average, which again shows that the Integrated matrix has no disadvantages in situations where it doesn't improve performance.

4.5.5 Overall



(a) Average AMI of All 4 Clustering Algorithms on All 6 Datasets for the Raw, Propagated, and Integrated Data Matrices



(b) Average ARI of All 4 Clustering Algorithms on All 6 Datasets for the Raw, Propagated, and Integrated Data Matrices

Figure 4.15. Summarizing the Effect of the Integrated Matrix on the Performance of All Clustering Algorithms

You'll see the average results for all clustering algorithms for the three full matrices in Figure 4.15. As you can see, the Integrated bar is the highest of the three bars for both AMI and ARI. This means that overall, the Integrated matrix performs the best of the three.

5 Conclusions

Based on our results, we conclude that the Integrated data matrix is the best matrix to use when clustering on single-cell data because it appears to be consistently tied to having the best results, which can't be said for the Raw and Propagated data matrices. As is described in Chapter 4 and can be seen in Figures 4.11, 4.12, 4.13, and 4.14, the Integrated matrix consistently performs the same as the better between the Raw and Propagated matrices, which is important because neither the Raw nor the Progapagated matrix consistently outperforms the other.

Since the Integrated matrix outperforms both the Raw matrix and the Propagated matrix, we can also conclude that incorporating a gene-to-gene regulatory interaction network based off of transcription factors into the single-cell clustering process allows us to get enhanced results. Now, as can also be seen in Figures 4.11, 4.12, 4.13, and 4.14, and what was our main reason for creating the Integrated matrix, the Propagated matrix does not always outperform the Raw matrix. This is significant because it proves that incorporating regulatory interactions, specifically in regulators, leads to enhanced results, rather than incorporating interactions with all genes, which leads to mixed results.

Lastly, as incorporating a gene-to-gene regulatory interaction network based off of transcription factors allows us to get better results when clustering single-cell RNA-sequencing data, this leads us conclude that the relationships between

different genes does, in fact, play a role in how well single-cell RNA-sequencing data is able to be clustered.

References

[1] Irène Baccelli, Andreas Schneeweiss, Sabine Riethdorf, Albrecht Stenzinger, Anja Schillert, Vanessa Vogel, Corinna Klein, Massimo Saini, Tobias Bäuerle, Markus Wallwiener, et al. Identification of a population of blood circulating tumor cells from breast cancer patients that initiates metastasis in a xenograft assay. *Nature biotechnology*, 31(6):539–544, 2013.

[2] Vincent D Blondel, Jean-Loup Guillaume, Renaud Lambiotte, and Etienne Lefebvre. Fast unfolding of communities in large networks. *Journal of statistical mechanics: theory and experiment*, 2008(10):P10008, 2008.

[3] Junyue Cao, Malte Spielmann, Xiaojie Qiu, Xingfan Huang, Daniel M Ibrahim, Andrew J Hill, Fan Zhang, Stefan Mundlos, Lena Christiansen, Frank J Steemers, et al. The single-cell transcriptional landscape of mammalian organogenesis. *Nature*, 566(7745):496–502, 2019.

[4] Ashraful Haque, Jessica Engel, Sarah A Teichmann, and Tapio Lönnberg. A practical guide to single-cell rna-sequencing for biomedical research and clinical applications. *Genome medicine*, 9(1):1–12, 2017.

[5] Sophie Helaine, Angela M Cheverton, Kathryn G Watson, Laura M Faure, Sophie A Matthews, and David W Holden. Internalization of salmonella by macrophages induces formation of nonreplicating persisters. *Science*, 343 (6167):204–208, 2014.

[6] Ming-Wen Hu, Dong Won Kim, Sheng Liu, Donald J Zack, Seth Blackshaw, and Jiang Qian. Panoview: An iterative clustering method for single-cell rna sequencing data. *PLoS computational biology*, 15(8):e1007040, 2019.

[7] Ruth Huh, Yuchen Yang, Yuchao Jiang, Yin Shen, and Yun Li. Same-clustering: S ingle-cell a ggregated clustering via m ixture model e nsemble. *Nucleic acids research*, 48(1):86–95, 2020.

[8] Vladimir Yu Kiselev, Kristina Kirschner, Michael T Schaub, Tallulah Andrews, Andrew Yiu, Tamir Chandra, Kedar N Natarajan, Wolf Reik, Mauricio Barahona, Anthony R Green, et al. Sc3: consensus clustering of single-cell rna-seq data. *Nature methods*, 14(5):483–486, 2017.

[9] Vladimir Yu Kiselev, Tallulah S Andrews, and Martin Hemberg. Challenges in unsupervised clustering of single-cell rna-seq data. *Nature Reviews Genetics*, 20(5):273–282, 2019.

[10] Peijie Lin, Michael Troup, and Joshua WK Ho. Cidr: Ultrafast and accurate clustering through imputation for single-cell rna-seq data. *Genome biology*, 18(1):1–11, 2017.

[11] Zhi-Ping Liu, Canglin Wu, Hongyu Miao, and Hulin Wu. Regnetwork: an integrated database of transcriptional and post-transcriptional regulatory networks in human and mouse. *Database*, 2015:bav095, 2015.

[12] Evan Z. Macosko, Anindita Basu, Rahul Satija, James Nemesh, Karthik Shekhar, Melissa Goldman, Itay Tirosh, Allison R. Bialas, Nolan Kamitaki, Emily M. Martersteck, John J. Trombetta, David A. Weitz, Joshua R. Sanes, Alex K. Shalek, Aviv Regev, and Steven A. McCarroll. Highly parallel genome-wide expression profiling of individual cells using nanoliter droplets. *Cell*, 161:1202–1214, 2015.

[13] Elisabetta Mereu, Atefeh Lafzi, Catia Moutinho, Christoph Ziegenhain, Davis J McCarthy, Adrian Alvarez-Varela, Eduard Batlle, Sagar, Dominic Gruen, Julia K Lau, et al. Benchmarking single-cell rna-sequencing protocols for cell atlas projects. *Nature biotechnology*, 38(6):747–755, 2020.

[14] Simone Romano, Nguyen Xuan Vinh, James Bailey, and Karin Verspoor. Adjusting for chance clustering comparison measures. *The Journal of Machine Learning Research*, 17(1):4635–4666, 2016.

[15] RStudio Team. *RStudio: Integrated Development Environment for R*. RStudio, PBC., Boston, MA, 2020. URL http://www.rstudio.com/.

[16] Rahul Satija, Jeffrey A Farrell, David Gennert, Alexander F Schier, and Aviv Regev. Spatial reconstruction of single-cell gene expression data. *Nature biotechnology*, 33(5):495–502, 2015.

[17] Sreenath V Sharma, Diana Y Lee, Bihua Li, Margaret P Quinlan, Fumiyuki Takahashi, Shyamala Maheswaran, Ultan McDermott, Nancy Azizian, Lee Zou, Michael A Fischbach, et al. A chromatin-mediated reversible drug-tolerant state in cancer cell subpopulations. *Cell*, 141(1):69–80, 2010.

[18] Berend Snijder, Raphael Sacher, Pauli Rämö, Eva-Maria Damm, Prisca Liberali, and Lucas Pelkmans. Population context determines cell-to-cell variability in endocytosis and virus infection. *Nature*, 461(7263):520–523, 2009.

[19] Tim Stuart, Andrew Butler, Paul Hoffman, Christoph Hafemeister, Efthymia Papalexi, William M Mauck, Yuhan Hao, Marlon Stoeckius, Peter Smibert, and Rahul Satija. Comprehensive integration of single-cell data. *Cell*, 177(7):1888–1902, 2019.

[20] Luyi Tian, Xueyi Dong, Saskia Freytag, Kim-Anh Lê Cao, Shian Su, Abolfazl JalalAbadi, Daniela Amann-Zalcenstein, Tom S. Weber, Azadeh Seidi, Jafar S. Jabbari, Shalin H. Naik, and Matthew E. Ritchie. Benchmarking single cell rna-sequencing analysis pipelines using mixture control experiments. *Nature Methods*, 16:479–487, 2019.

[21] Vincent A Traag, Ludo Waltman, and Nees Jan Van Eck. From louvain to leiden: guaranteeing well-connected communities. *Scientific reports*, 9(1):5233, 2019.

[22] Bang Tran, Duc Tran, Hung Nguyen, Seungil Ro, and Tin Nguyen. sccan: single-cell clustering using autoencoder and network fusion. *Scientific Reports*, 12 (1):10267, 2022.

[23] Duc Tran, Bang Tran, Hung Nguyen, and Tin Nguyen. A novel method for single-cell data imputation using subspace regression. *Scientific Reports*, 12 (1):2697, 2022.

[24] Shibiao Wan, Junil Kim, and Kyoung Jae Won. Sharp: hyperfast and accurate processing of single-cell rna-seq data via ensemble random projection. *Genome research*, 30(2):205–213, 2020.

[25] Bo Wang, Junjie Zhu, Emma Pierson, Daniele Ramazzotti, and Serafim Batzoglou. Visualization and analysis of single-cell rna-seq data by kernel-based similarity learning. *Nature methods*, 14(4):414–416, 2017.

[26] F Alexander Wolf, Philipp Angerer, and Fabian J Theis. Scanpy: large-scale single-cell gene expression data analysis. *Genome biology*, 19:1–5, 2018.

[27] Lu Yang, Jiancheng Liu, Qiang Lu, Arthur D Riggs, and Xiwei Wu. Saic: an iterative clustering approach for analysis of single cell rna-seq data. *BMC genomics*, 18:9–17, 2017.

[28] Yuchen Yang, Ruth Huh, Houston W Culpepper, Yuan Lin, Michael I Love, and Yun Li. Safe-clustering: single-cell aggregated (from ensemble) clustering for single-cell rna-seq data. *Bioinformatics*, 35(8):1269–1277, 2019.

[29] Serhan Yılmaz, Marzieh Ayati, Daniela Schlatzer, A Ercüment Çiçek, Mark R Chance, and Mehmet Koyutürk. Robust inference of kinase activity using functional networks. *Nature communications*, 12(1):1177, 2021.

[30] Amit Zeisel, Ana B Muñoz-Manchado, Simone Codeluppi, Peter Lönnerberg, Gioele La Manno, Anna Juréus, Sueli Marques, Hermany Munguba, Liqun He, Christer Betsholtz, et al. Cell types in the mouse cortex and hippocampus revealed by single-cell rna-seq. *Science*, 347(6226):1138–1142, 2015.

[31] Grace X. Y. Zheng, Jessica M. Terry, Phillip Belgrader, Paul Ryvkin, Zachary W. Bent, Ryan Wilson, Solongo B. Ziraldo, Tobias D. Wheeler, Geoff P. McDermott, Junjie Zhu, Mark T. Gregory, Joe Shuga, Luz Montesclaros, Jason G. Underwood, Donald A. Masquelier, Stefanie Y. Nishimura, Michael Schnall-Levin, Paul W. Wyatt, Christopher M. Hindson, Rajiv Bharadwaj, Alexander Wong, Kevin D. Ness, Lan W. Beppu, H. Joachim Deeg, Christopher McFarland, Keith R. Loeb, William J. Valente, Nolan G. Ericson, Emily A. Stevens, Jerald P. Radich, Tarjei S. Mikkelsen, Benjamin J. Hindson, and Jason H. Bielas. Massively parallel digital transcriptional profiling of single cells. *Nature Communications*, 8, 2017.

[32] Xiaoshu Zhu, Jian Li, Hong-Dong Li, Miao Xie, and Jianxin Wang. Sc-gpe: A graph partitioning-based cluster ensemble method for single-cell. *Frontiers in Genetics*, 11:604790, 2020.

[33] Justina Žurauskienė and Christopher Yau. pcareduce: hierarchical clustering of single cell transcriptional profiles. *BMC bioinformatics*, 17:1–11, 2016.