Imputation Methods in High-dimensional Sparse Data

Jurijs Nazarovs[*]

May 11, 2018

1 Motivation

1.1 Explanation of high-dimensional sparse data

Nowadays, we have an access to huge amount of data. However, not every time all data is meaningful. Data might contain technical noise, produced by a device, which collects data, or data might be simply missed for some reasons. Especially data with technical errors is common in biology. On of the example is single-cell RNA-seq (scRNA) data, produced by 10x machine. Sometimes if device gets a measure close to zero, but not zero, data is still reported as zero. Such effect calls dropout and similar data is considered as sparse data.

Genetics data, scRNA, is a very good example of high-dimensional data. Typical results of an experiment after preprocessing is represented by a matrix with around 25000 rows - genes and 4500 columns - cells. If we would like to perform an analysis of cells, considering genes as covariates, then dimension p is much higher than n.

In the end we gets that scRNA data is usually high-dimensional sparse data. For more visible explanation of sparsity Figure 1 displays distribution of portion of 0-expressed genes across cells for 4 different real scRNA experiments. That is, we calculate a proportion of 0-expressed genes per cell and plot the distribution.

Based on Figure 1[1] we can say that on average per cell at least 87% of genes are 0-expressed. Which means that per observation 87% of covariates are 0.

1.2 Data-science objective

One of the most common goals in data analysis is to get a structure of data. Usually, to complete this task, some sort of clustering approach is required, for example, k-means. However, a lot of clustering methods try to gather together data-points based on a distance in high-dimension. If the distance looks the same, then points are similar. The problem arises when such approach is applied to high-dimensional sparse data, because data-points might be completely different, but the presence of huge number of 0s, provides the same value for distance. Thus, very different points are clustered together.

Another desired step in data-analysis is data visualization. However, it is hard to visualize a matrix with dimension $[25000 \times 4500]$ and great number of 0. For this purpose the common

[*]UW Madison, Department of Statistics. Email: nazarovs@wisc.edu

[1]4 replicates from Bresnick Lab: mutant(A and B) and wild type (F and I) and conduct an initial comparison with a published 10X data (Peripheral blood mononuclear cells (PBMCs)) to see the quality and similarity between data sets (our and published)

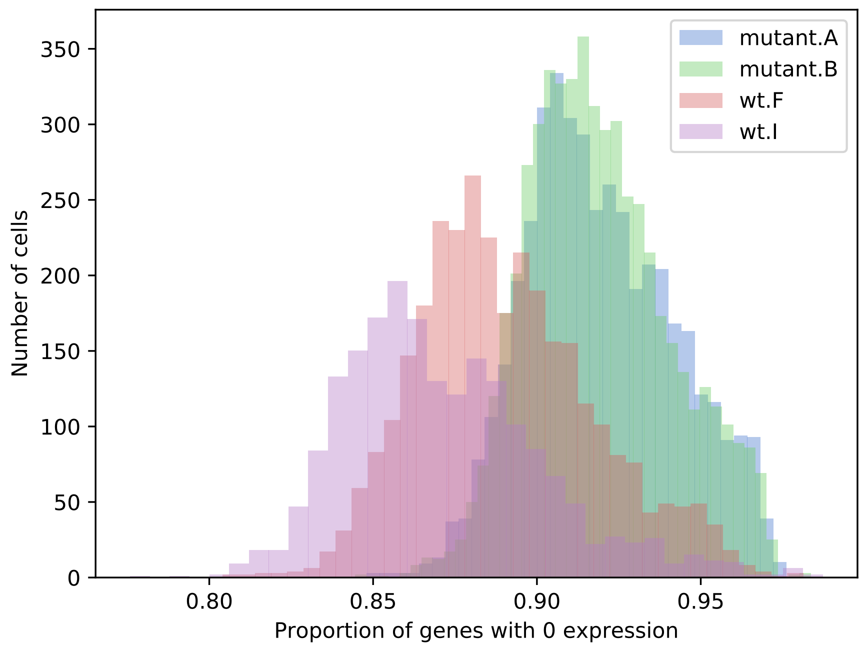


Figure 1: Distribution of proportion of 0-expressed genes per cell for 4 replicates

approach is to apply some dimension reduction method. Popular methods are PCA and state of art t-sne. Unfortunately, both methods were not designed to deal with enormous sparsity of the data and might provide poor results.

To sum up previous paragraphs, sparsity of data is a trouble. One way to solve it to decrease amount of 0s, by replacing 0 with some non-zero values, based on prior information or extracted from original data. Such approach is called imputation and this paper covers 3 imputation methods:

1. CIDR [1]
2. MAGIC [3]
3. netSmooth [2]

2 Imputation methods

2.1 Common idea

All imputation methods mentioned above are same in sense that they change 0 values for non-zero, and they do that based on some sort of dissimilarity/affinity matrix. The difference is in approach of crating a dissimilarity/affinity matrix. Some methods are using just data by itself, other implement an injection of some prior data. Important to notice that usually imputation methods are not dimension-reduction methods in their origins and are dedicated to change 0s for other variable. Thus, it is still a good idea to apply dimension-reduction method to results of imputation methods.

The following part of this section describes the main idea with mathematical development of each method.

2.2 CIDR

CIDR is the earliest model among three analyzed here, and probably the most intuitive. The idea is to create a dissimilarity matrix between data-points (cells in case of scRNA), using modified euclidian distance. Where the euclidian distance is calculated given that not all 0 are real 0. From mathematical perspective dissimilarity matrix is:

$$[D(c_i, c_j)]^2 = \sum_{k=1}^{n}(o_{ki} - o_{kj}) = \sum_{k \in \{\text{no zeros}\}}(o_{ki} - o_{kj}) + \sum_{k \in \{\text{both zeros}\}}(o_{ki} - o_{kj}) + \sum_{k \in \{\text{one zero}\}}(o_{ki} - o_{kj}),$$

where o_{ki} is expression of gene k in cell i. Problems appear when we have one zero value - dropout. Thus, authors are going to make an imputation just in this case. The detailed procedure is following:

1. Detect threshold to select possible dropouts. Thus, for gene expression per cell being a dropout is not necessary be zero. It is possible to be some other small positive value, but less than threshold.
2. Estimate a probability of gene expression being a dropout by least square estimates of decreasing logistic function: (empirical dropout rate vs average of expressed entries) recorder for every gene, where

   $$\text{empirical dropout rate} = \frac{\#\text{cells w/ dropout expression}}{\#cells}$$

   $$\text{average of expressed entries} = \text{mean}(o_{ki} : o_{ki} \neq 0 \text{ and } \neq \text{ dropout})$$

   The decrease logistic function is used because higher average expression leads to a lower chance to get a dropout.
3. Calculate a dissimilarity matrix between data points mentioned above, but in case when one value is dropout and another is not, the dropout value o_{ki} is replaced by

   $$\hat{o}_{ki} = \hat{P}(o_{ki})o_{ki} + (1 - \hat{P}(o_{ki}))o_{ki}$$

Once we got a dissimilarity matrix, we can proceed a downstream analysis, like a dimension reduction prior to clustering and visualization.

2.3 MAGIC

The second chronological method is the MAGIC. The idea is very different from the one described in CIDR and more complicated. Instead of calculating dissimilarity matrix, method is focusing on affinity matrix and implements an idea from thermodynamic about heat distribution - diffusion process (Markov-normalized affinity), but applying it to other type of data. Apparently, cells are attracted to stable cell-states, and MAGIC approach is trying to find this state. Since a cellular state space forms a smooth low-dimensional structure - manifold, such manifold can be represented by a nearest neighbor graph. Each node is a cell, which is connected to the nearest neighbor, based on affinity. Once such graph is created, values of 0 - expressed genes are imputed by values from a neighborhood. The detailed mathematical procedure is described below:

1. Consider D as a filtered matrix with rows being genes and columns being cells.
2. Compute a dissimilarity matrix, using euclidian distance between columns - cells.
3. Compute an affinity matrix using Gaussian kernel: $k(x, y) = exp(\frac{-||x-y||^2}{\sigma^2})$. It allows to find a continuous manifold and has a property that "higher distance $\rightarrow$ lower affinity".
4. Compute M - Markov affinity matrix, which is just normalization of affinity matrix, such that row-sum is 1. Values of M are similar to probabilities to go from one state to another, where state is gene expression of a cell.
5. Detect t, such that M^t gives a stationary state, which leads to clusters of states.
6. Compute imputed matrix according to: $D_{imputed} = M^t D$.
7. Rescale imputed D.

Once we get the final version of imputed matrix D, we can proceed with a downstream analysis.

2.4 netSmooth

The netSmooth method is the last one discussed in this paper. It was inspired by MAGIC, but instead of finding stationary state based on neighborhood data, the method uses other prior information, for example, protein-protein interaction matrix. Procedure is very similar to the MAGIC method:

1. Consider D as a filtered matrix with rows being genes and columns being cells
2. Compute M - Markov affinity matrix, by normalization, such that row-sum is 1, based on prior information. For example, protein-protein network
3. Use a process of random walk with restart to get a stationary distribution of matrix D. This is a diffusion process again as in Magic, which gives us imputed values

Resulting imputed matrix is same as initial Matrix (genes $\times$ cells), but with some previous values of 0 different from 0 now.

2.5 Packages availability

All three methods are available at least in R as fully-functional packages from the following links:

1. CIDR: https://github.com/VCCRI/CIDR
2. MAGIC: https://github.com/KrishnaswamyLab/MAGIC
3. netSmooth: https://github.com/BIMSBbioinfo/netSmooth/

2.6 Summary

The main goal of imputation methods is to change 0 values for some non-zero, using available data. The derived matrix might be used for clustering immediately, since it should be less 0 values, but dimension-reduction methods still might be helpful.

3 Simulation study to compare imputation methods

This section shows the comparison of two imputation methods CIDR and MAGIC, based on accuracy of detected clusters, using same clustering method and doing a PCA reduction prior to a clustering procedure. Since the comparison study is based on simulation, we do not include the netSmooth method, because we do not have a prior information and not sure what is a good way to simulate it.

3.1 Simulated dataset

To make a comparison of imputation methods in respect of an ability to impute data to detect clusters, we simulate data similar to described in Section 1, with percentage of 0 expressed genes varies in the interval $(75\%, 90\%)$ per cell. Every cell is assigned to one of 5 clusters, and every cluster has the same number of data points.

Non-zero values are draw from a Poison distribution with mean equals to a cluster number. That is, $o_{ki} \sim Po(\lambda_j)$ - value in k-th row and i-th column, and $\lambda_j \in \{1, \ldots, 5\}$. Same time, since we are interested in imputation in high-dimensional data, $k = 1, \ldots, 25000$ and $i = 1, \ldots, 4500$.

3.2 Downstream analysis after imputation

We apply clustering to two different sets per imputation method: one is clustering based on imputed data, another is clustering of imputed data after dimension reduction method - PCA with 2 components.

For clustering columns - cells, we use k-means clustering, with $k = 5$.

3.3 Performance comparison

Since we know real labels of clusters, which are assigned in simulation study, we compute accuracy for both methods according to the following formula

$$mean(y_{real} == y_{predicted}).$$

Since sometimes it is possible that specific method performs better than on average, we do a comparison for 50 times (it takes quite long to run on such big data), and take a mean accuracy as a final value of the analysis. We got following results:

CIDR	CDIR+PCA	MAGIC	MAGIC + PCA
0.2	0.19	0.3	0.3

Based on the simulation study we select MAGIC to apply to a real data set and will perform clustering and visualization.

4 Real data analysis

In this section we consider an analysis of real data, which represents a gene expression of a single-cell RNA-seq experiment, obtained by 10x machine. The data contains information about 27,877 genes and 3,119 cells. On average data has 88% of 0-expressed genes per cell, with minimum amount of 80% and maximum of 98% of 0-expressed genes per cell.

Based on our simulation study we select the MAGIC method as an imputation approach. Since in real data we do not know cells association to any of specific groups, it is hard to judge how good the clustering is. For clustering we consider two procedures:

1. hierarchical cluster and plot the corresponding heat map
2. k-means with several different k and following comparison of average silhouette measure. And following plot of clustered labels on 2d representation of the data, using t-sne method.

4.1 Hierarchical clustering

To see the difference in imputation, it is good to compare results of clustering using the same scheme and hyper parameters. For comparison of hierarchical clustering I chose a heat map with an order of clustered cells (Figure 2) and heatmap where rows and columns are clustered (Figure 3).

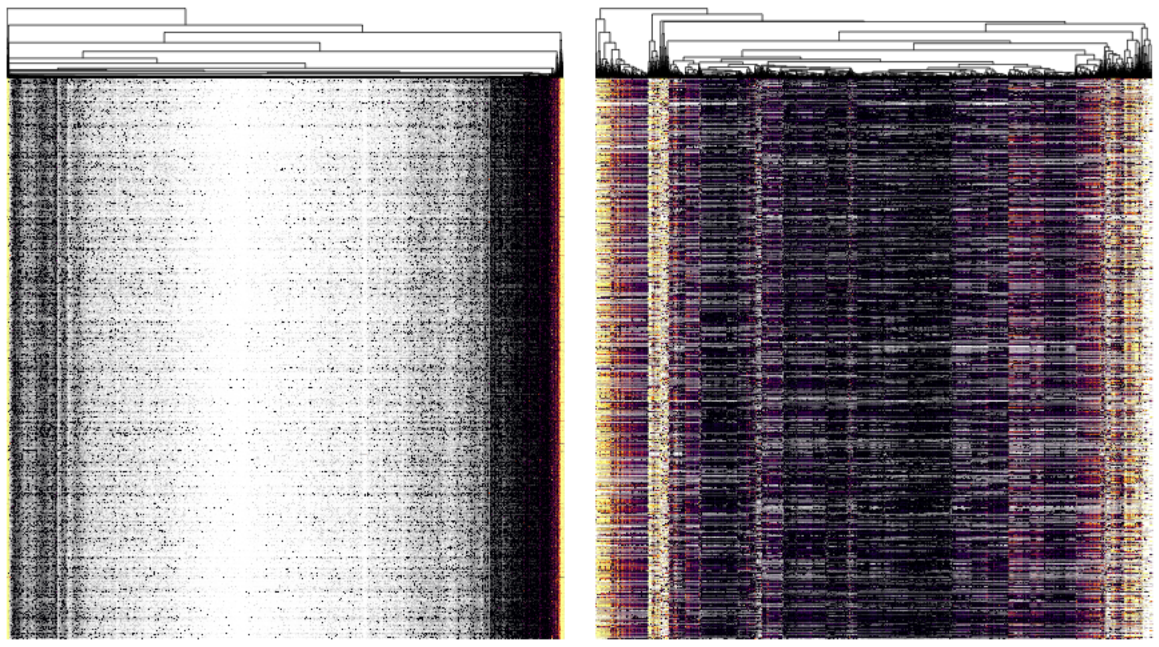
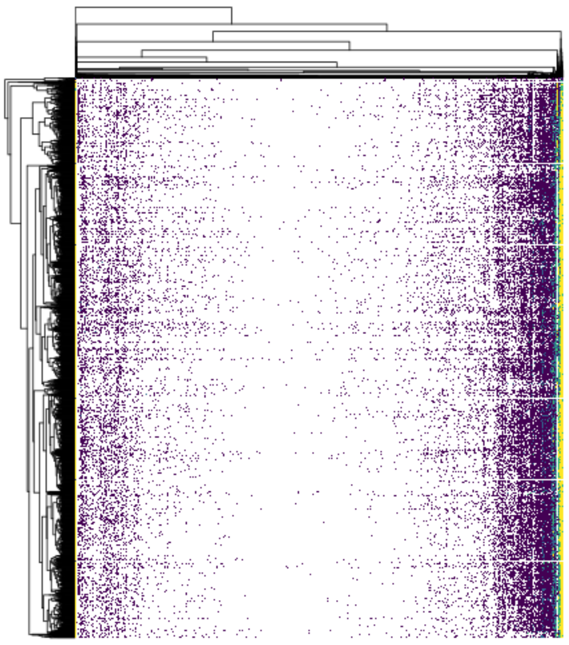

(a) Raw data (b) Imputed data

Figure 2: Heatmap of 2 datasets: raw and imputed. Heatmap's columns are ordered according to results of hierarchical clustering

Figure 2 and Figure 3 shows that after imputation hierarchical clustering algorithm is able to catch higher amount of clusters. However, based on Figure 2 I still have doubts that imputation works well. Since it does not look that clusters are so different. It looks more like visualization illusion, because we have more colors, which are further from white, since zeros are non-zero values now. But once we apply clustering also to rows, we see a huge difference in visualization, which is shown on Figure 3.

Based on plots above we could say that imputation does make a huge difference at least in visualization. In the next section we continue an analysis based on k-means.


(a) Raw data

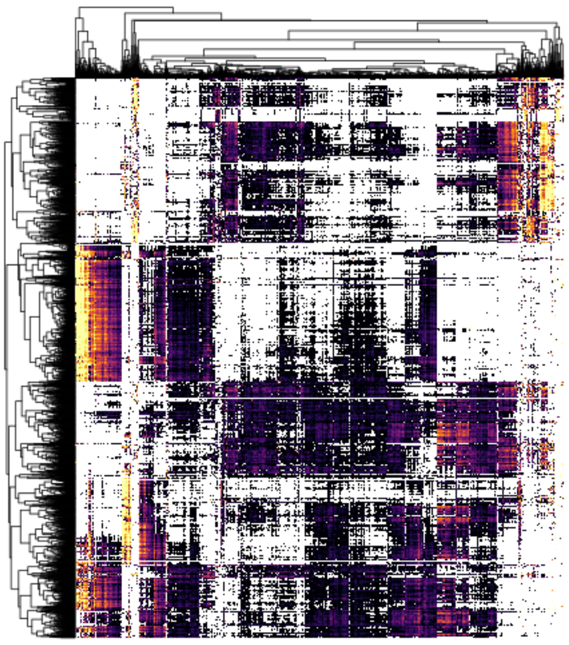

(b) Imputed data

Figure 3: Heatmap of 2 datasets: raw and imputed. Heatmap's columns and rows are ordered according to results of hierarchical clustering

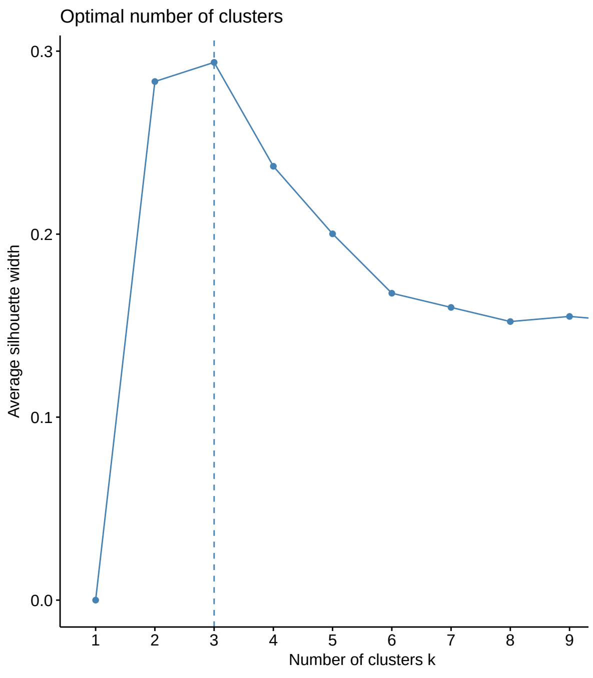
4.2 k-means

Since we do not have any prior information about the number of clusters in cells, we are running k-means for $k = 1, \dots, 10$ and calculating an average silhouette measure for every k. Figure 4 presents an average silhouette measure for every k in 2 data sets: raw and imputed.

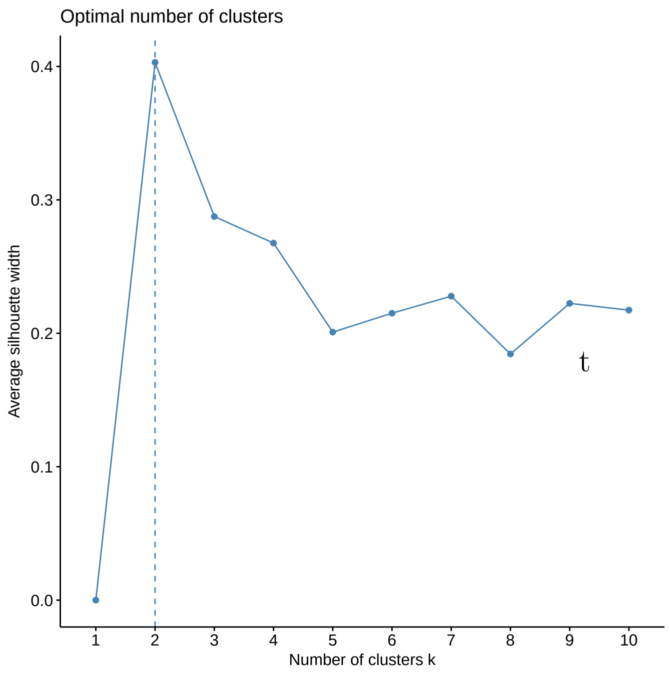
From Figure 4 we see that for raw data the best number of clusters is 3 and for imputed is 2. In the following subsection we are plotting these clusters on 2d representation of a data to see, if the structure of derived clusters is saved after t-sne procedure.

4.3 t-sne

To make a 2d visualization we apply t-sne with parameter 2, and plot labels of clusters derived in a previous subsection. Figure 5 represents visualization of clusters with an ideal k, based on average silhouette measure.

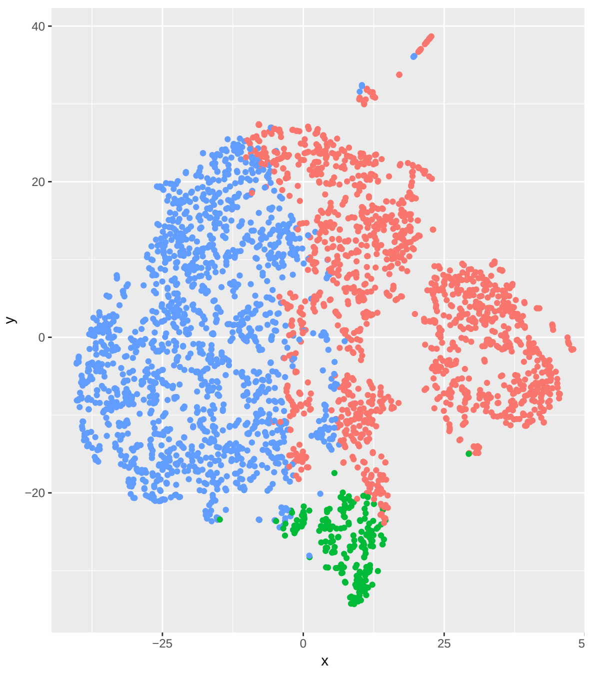


(a) Raw data

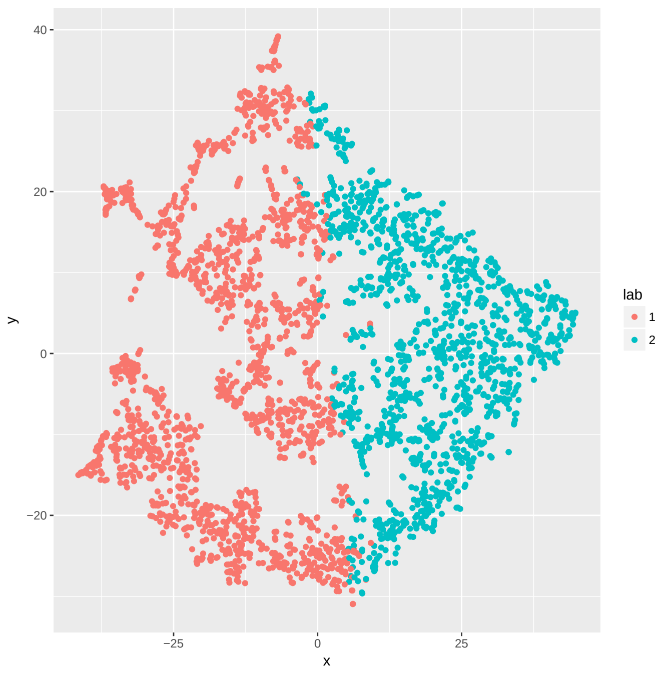


(b) Imputed data

Figure 4: The average silhouette measure for different k in k-means clustering



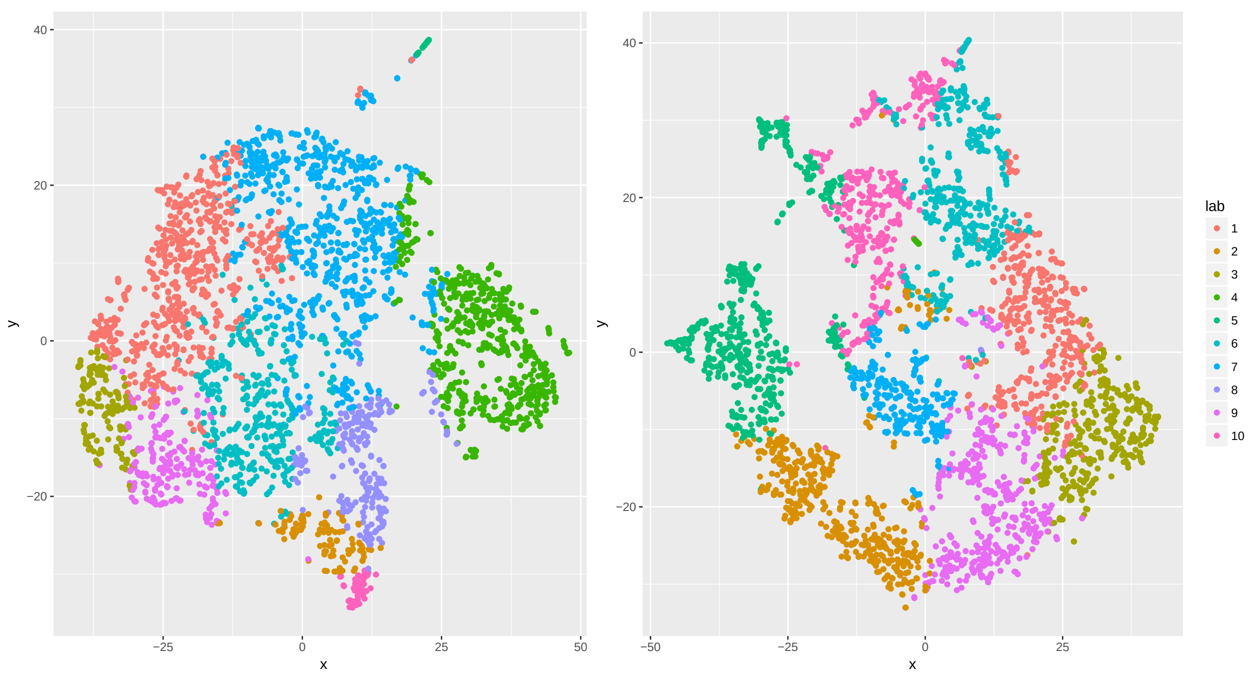
(a) Raw data, k = 3



(b) Imputed data, k = 2

Figure 5: t-sne representation of 2 datasets: raw and imputed, based on the best number of cluster according to the average silhouette measure

Based on Figure 5 we can say that t-sne keeps the information about clusters, when it is doing low-dimensional representation. However, it doesn't look like imputation provides any significant improvements. it is possible that the selected ks are very small, thus, lets plot same graph, but with $k = 10$ for both data sets.



(a) Raw data, k = 10 (b) Imputed data, k = 10

Figure 6: t-sne representation of 2 datasets: raw and imputed

Unfortunately, Figure 6 does not show any significant difference in cluster distribution between raw and imputed data. There is no mix in clusters, that one class is split by another. And both pictures looks identical in respect of clustering, just the image corresponded to an imputed data set is flipped.

4.4 Result of imputation

To understand if predicted classes are the same between raw and imputed data, we consider the proportion of match between predictions, based on raw data and imputed data. For $k = 3$ we have 23% match, and for $k = 10$, just 4% match. Based on our observation and simulation study, we would say that imputation does influence on clustering.

5 Reference

References

[1] P. Lin, M. Troup, and J. W. Ho. Cidr: Ultrafast and accurate clustering through imputation for single-cell rna-seq data. *Genome biology*, 18(1):59, 2017.

[2] J. Ronen and A. Akalin. netsmooth: Network-smoothing based imputation for single cell rna-seq. *F1000Research*, 7, 2018.

[3] D. van Dijk, J. Nainys, R. Sharma, P. Kathail, A. J. Carr, K. R. Moon, L. Mazutis, G. Wolf, S. Krishnaswamy, and D. Pe'er. Magic: A diffusion-based imputation method reveals gene-gene interactions in single-cell rna-sequencing data. *BioRxiv*, page 111591, 2017.

6 Supplementary material

6.1 Simulation study

```
library(cidr)
library(Rmagic)
library(precrec)

## Prior parameters
p <- 25000
n <- 4500
n_labels <- 5
labels <- sample(rep(seq(1, n_labels, length.out = n_labels), n/n_labels))

## Data generation
data <- matrix(0, nrow = p, ncol = n)
for (j in 1:dim(data)[2]){
  ind_tmp <- rep(TRUE, dim(data)[1])
  ind_of_0 <- sample(1:dim(data)[1], size = round(runif(1,
                                              min = 0.75*dim(data)[1],
                                              max = 0.90*dim(data)[1])))
  ind_tmp[ind_of_0] <- FALSE
  data[ind_tmp, j] <- rpois(n = sum(ind_tmp), lambda = labels[j])
}

## CIDR
sData <-  scDataConstructor(data)
sData <- determineDropoutCandidates(sData)
sData <- wThreshold(sData)
sData <- scDissim(sData)
data_cidr <- slot(sData, "dissim")

# raw
labels_cidr <- kmeans(data_cidr, n_labels)$cluster
accuracy_cidr <- mean(labels == labels_cidr)
# PCA
data_cidr_pca <- prcomp(data_cidr)$x[, 1:2]
labels_cidr_pca <- kmeans(data_cidr_pca, n_labels)$cluster
accuracy_cidr_pca <- mean(labels == labels_cidr_pca)

## MAGIC
data_magic <- run_magic(data, rescale_percent=0.99)
# raw
labels_magic <- kmeans(t(data_magic), n_labels)$cluster
accuracy_magic <- mean(labels == labels_magic)
# PCA
data_magic_pca <- prcomp(t(data_magic))$x[, 1:2]
labels_magic_pca <- kmeans(data_magic_pca, n_labels)$cluster
accuracy_magic_pca <- mean(labels == labels_magic_pca)
```

6.2 Real data analysis

```
setwd("/Users/owner/Box Sync/UW/_stat771/hw/project/")
library(pheatmap)
library(ggplot2)
library(cluster)
library(factoextra)
library(Rtsne)
library(viridis)

experiment_name <- "wt.F_rg.magic"
load(paste0(experiment_name, ".RData"))
#data_MAGIC <- read.csv(paste0(experiment_name, ".csv"), stringsAsFactors = F,
#                        header = T, row.names = 1)
data_magic <- t(data_MAGIC)


#plot(silhouette(kmm$cluster, data_magic), col=1:n_clusters, border=NA)

# Silhouette
pdf(paste0(experiment_name, '_silhouette.pdf'))
fviz_nbclust(data_magic, kmeans, method = "silhouette")
dev.off()

# k-means
n_clusters <- 3
kmm <- kmeans(data_magic, n_clusters)$cluster

# Compare k-means of raw data
# data_hui <- t(read.csv(paste0("wt.F_rg", ".csv"), stringsAsFactors = F,
#                        header = T, row.names = 1))
# kmm_raw <- kmeans(data_hui, n_clusters)$cluster
# print(mean(kmm == kmm_raw))


# t-sne
data_tsne <- Rtsne(data_magic, 2)$Y
data_plot <- data.frame(x = data_tsne[, 1], y = data_tsne[, 2], lab = factor(kmm))
pdf(paste0(experiment_name, "_tsne_kmeans_", n_clusters, ".pdf"))
ggplot(data_plot, aes(x=x, y=y, color = lab)) +
  geom_point()
dev.off()

## Create breaks for pheatmap
quantile_breaks <- function(xs, n = 10) {
  breaks <- quantile(xs, probs = seq(0, 1, length.out = n))
  breaks[!duplicated(breaks)]
}
magic_breaks <- quantile_breaks(t(as.matrix(data_MAGIC)), n = 20)
```

```
png(paste0(experiment_name, '_heatmap.png'))
pheatmap(t(as.matrix(data_MAGIC)),
         show_colnames = FALSE,
         show_rownames = FALSE,
         scale = "row", #kmeans_k = 5,
         cluster_rows = FALSE,
         color             = inferno(length(magic_breaks) - 1),
         breaks            = magic_breaks,
         border_color      = NA,
         drop_levels       = TRUE,
         fontsize          = 14)
dev.off()
```