iCellR: Combined Coverage Correction and Principal Component Alignment for Batch Alignment in Single-Cell Sequencing Analysis

Alireza Khodadadi-Jamayran[1]*, Joseph Pucella[2], Hua Zhou[1], Nicole Doudican[3], John Carucci[3], Adriana Heguy[4], Boris Reizis[2], Aristotelis Tsirigos[1,2]*

[1] Applied Bioinformatics Laboratories, NYU School of Medicine, New York, NY, USA
[2] Department of Pathology, NYU School of Medicine, New York, NY, USA
[3] Ronald O. Perelman Department of Dermatology, New York University Langone Medical Center, New York, NY
[4] Genome Technology Center (GTC), NYU School of Medicine, New York, NY, USA
**Corresponding authors: alireza.Khodadadi-Jamayran@nyulangone.org (A.K.), aristotelis.Tsirigos@nyulangone.org (A.T.)*

SUMMARY

Under-sampling RNA molecules and low-coverage sequencing in some single cell sequencing technologies introduce zero counts (also known as drop-outs) into the expression matrices. This issue may complicate the processes of dimensionality reduction and clustering, often forcing distinct cell types to falsely resemble one another, while eliminating subtle, but important differences. Considering the wide range in drop-out rates from different sequencing technologies, it can also affect the analysis at the time of batch/sample alignment and other downstream analyses. Therefore, generating an additional harmonized gene expression matrix is important. To address this, we introduce two separate batch alignment methods: Combined Coverage Correction Alignment (CCCA) and Combined Principal Component Alignment (CPCA). The first method uses a coverage correction approach (analogous to imputation) in a combined or joint fashion between multiple samples for batch alignment, while also correcting for drop-outs in a harmonious way. The second method (CPCA) skips the coverage correction step and uses k nearest neighbors (KNN) for aligning the PCs from the nearest neighboring cells in multiple samples. Our results of nine scRNA-seq PBMC samples from different batches and technologies shows the effectiveness of both these methods. All of our algorithms are implemented in R, deposited into CRAN, and available in the iCellR package.

INTRODUCTION

Single-cell sequencing of transcripts (RNAs) has become extremely popular in recent years [1], and many large-scale projects such as Human Cell Atlas (HCA) [2] and Mouse Cell Atlas (MCA) [3] are examples of accelerating growth. While many labs produce such data, integrating these data and correcting for batch differences has emerged as a crucial step in the process. Thus, researchers have developed a variety of methods, such as Mutual Nearest Neighbors (MNN) [4], Batch Balanced K Nearest Neighbors (BBKNN) [5], Harmony [6] and Seurat Multiple Canonical Correlation Analysis (MultiCCA) [7], to correct batch effects by aligning different samples.

In addition, single-cell sequencing technologies suffer from low capture rates or under-sampling of RNA molecules, also known as drop-outs [8]. Different sequencing technologies have varying drop-out and gene coverage rates. Although, some labs have developed methods, such as Markov Affinity-based Graph Imputation of Cells (MAGIC) [8], DrImpute [9], Variability-preserving ImPutation for Expression Recovery (VIPER) [10] and scImpute [11], none have developed a method to simultaneously harmonize the drop-out and gene converge rates across different technologies and batches.

Here, we introduce a method to solve both problems in one step. Our Combined Coverage Correction Alignment (CCCA) algorithm not only performs an accurate batch alignment but also harmonizes the gene coverage and drop-out rates. Our Coverage Correction (CC) algorithm is analogous to imputation and performs with a higher accuracy than the other methods mentioned above. Furthermore, we also introduce Combined Principal Component Alignment (CPCA), which skips coverage correction and employs principle components for alignment instead of imputed expression matrix.

METHODS

Coverage Correction (CC)

To fill the drop-outs using this method, we perform a Principle Component Analysis (PCA) on the expression matrix and then calculate the distances (Euclidian by default) between the cells using the first few PCs (10 PCs by default). Then, we use KNN [12] to find the nearest neighboring cells (k=10 by default) per each cell (root cell). We then average the expression values of neighboring cells and apply the averaged values to the root cell. This process is repeated for each single cell, creating a high-resolution expression smoothing. Unlike the current imputation methods where the resulted matrices have multiple layers of data transformation (scaling, log, etc.), coverage correction has expression values very close to the ones in the original matrix. This means that fold changes remain similar (in the concept of differential expression analysis) and the data follows a similar pattern. This can be useful if one chooses the option of using coverage corrected data for differential expression and downstream analyses.

Combined Coverage Correction Alignment (CCCA)

Here, we use coverage correction as an alignment technique and a coverage harmonization method. To do this, we perform a Principle Component Analysis (PCA) on the expression matrix and then calculate the distances (Euclidian by default) between the cells using the first few PCs (30 PCs by default). Then, we collect 'equal' numbers of nearest neighboring cells (k=10 by default) for each cell (root cells) 'from every batch' and then average the expression values and apply them to the root cell. This is repeated for all the cells. In this process, every cell is considered a root cell only once. This method results in a harmonization among the expression values and gene coverages from different batches and ensures that the drop-out rates are much improved.

Combined Principle Component Alignment (CPCA)

This method is similar to CCCA; however, it skips the coverage correction part and instead aligns the Principle Components (PCs). In the CPCA method, the expression matrix is replaced with the PC matrix, and PCs are averaged instead of the expression values. CPCA is faster than CCCA, and the batch alignment results are similar. However, CPCA will not create a coverage corrected matrix.

RESULTS

Coverage Correction performs better than Markov Affinity-based Graph Imputation of Cells and a few other imputation methods

MAGIC (Markov Affinity-based Graph Imputation of Cells) is a widely used imputation method which employs diffusion data to denoise the expression matrix by filling the drop-outs [8]. To compare CC with MAGIC, we used a commonly analyzed PBMC sample dataset provided by 10x Genomics. Comparing heatmaps of gene expression matrices showed denoising for drop-outs with MAGIC introduced additional noise in a way that the resulting matrix does not show similar expression patterns seen in the original data (Figure 1). Furthermore, MAGIC failed to accurately measure a true gene-gene correlation. For instance, NKG7 and GNLY, two highly correlated genes, are expressed in NK cells and CD8-positive cells and not the other cell types. This pattern can be seen in the original data which includes drop-outs and is much improved using CC, but after MAGIC imputation CD8-positive cells seem to be mixed with other cells (Figure 2). Because the expression matrices imputed with MAGIC undergo a few levels of data transformation, such matrices cannot be used to calculate true fold changes in the concept of differential expression analysis. Our results also show that coverage corrected data, if used for clustering and dimensionality reduction, can illuminate more details by separating more cell types and clusters. For instance, it can separate CD8-positive T cells with distinctive distance from CD4-positive T cells (Figure 2). A few other imputation methods were also compared to CC and while some performed better than others CC was most consistent in following the patterns seen in the original data (Figure S2).

Combined Coverage Correction normalizes for gene coverage difference across different technologies

We analyzed nine PBMC sample datasets provided by the Broad Institute to detect batch differences [13]. These datasets were generated using varying technologies, including 10x Chromium v2 (3 samples), 10x Chromium v3, CEL-Seq2, Drop-seq, inDrop, Seq-Well and SMART-Seq [13]. Comparing the gene coverages across these technologies shows that combined coverage correction not only corrects for drop-outs but also harmonizes the gene coverages across different technologies (Figure 3).

CCCA and CPCA are both effective at batch alignment

All nine PBMC sample datasets mentioned above were used to perform both CPCA and CCCA alignment. We first performed Principal Component Analysis (PCA) then calculated cell-cell distance for all the cells in the nine samples. Each cell was considered a root cell once and then ten neighboring cells from each sample were found for each root cell. For CCCA, the expression matrix from the 10 samples was used to average the expression values. While for CPCA, the PCs were used. All the averaged values were then applied to the root cell, and this data was then used to perform PCA, T-distributed Stochastic Neighbor Embedding (t-SNE) and Uniform Manifold Approximation and Projection (UMAP). After performing batch alignment using both CCCA and CPCA, we observed that both methods could align the cell types from different technologies correctly (Figure 4 and S3). This was also evident in the heatmaps of marker genes (Figure S1).

Data Accession and Codes

All the PBMC data we used for batch alignment is available from the Broad Single Cell Portal (https://singlecell.broadinstitute.org/single_cell).

The PBMC data used for coverage correction can be downloaded from the 10x genomics website (https://support.10xgenomics.com/single-cell-gene-expression/datasets).

For connivance all these datasets can also be downloaded from here (https://genome.med.nyu.edu/results/external/iCellR/data/).

All the codes and algorithms are written in R and are available from iCellR package from CRAN (https://cran.r-project.org/package=iCellR).

In addition, the pipelines generating the figures in the results are in supplementary R scripts and the GitHub page for iCellR is (https://github.com/rezakj/iCellR).

REFERENCES

1- Svensson V, Vento-Tormo R, Teichmann SA. Exponential scaling of single-cell RNA-seq in the past decade. Nat Protoc. 2018 Apr;13(4):599-604.
2- Regev A, Teichmann SA, Lander ES, Amit I, Benoist C, Birney E, Bodenmiller B, Campbell P, Carninci P, Clatworthy M, Clevers H, Deplancke B, Dunham I, Eberwine J, Eils R, Enard W, Farmer A, Fugger L, Göttgens B, Hacohen N, Haniffa M, Hemberg M, Kim S, Klenerman P, Kriegstein A, Lein E, Linnarsson S, Lundberg E, Lundeberg J, Majumder P, Marioni JC, Merad M, Mhlanga M, Nawijn M, Netea M, Nolan G, Pe'er D, Phillipakis A, Ponting CP, Quake S, Reik W, Rozenblatt-Rosen O, Sanes J, Satija R, Schumacher TN, Shalek A, Shapiro E, Sharma P, Shin JW, Stegle O, Stratton M, Stubbington MJT, Theis FJ, Uhlen M, van Oudenaarden A, Wagner A, Watt F, Weissman J, Wold B, Xavier R, Yosef N; Human Cell Atlas Meeting Participants. The Human Cell Atlas. Elife. 2017 Dec 5;6. pii: e27041.
3- Han X, Wang R, Zhou Y, Fei L, Sun H, Lai S, Saadatpour A, Zhou Z, Chen H, Ye F, Huang D, Xu Y, Huang W, Jiang M, Jiang X, Mao J, Chen Y, Lu C, Xie J, Fang Q, Wang Y, Yue R, Li T, Huang H, Orkin SH, Yuan GC, Chen M, Guo G. Mapping the Mouse Cell Atlas by Microwell-Seq. Cell. 2018 Feb 22;172(5):1091-1107.e17.
4- Haghverdi L, Lun ATL, Morgan MD, Marioni JC. Batch effects in single-cell RNA-sequencing data are corrected by matching mutual nearest neighbors. Nat Biotechnol. 2018 Jun;36(5):421-427.
5- Polański K, Young MD, Miao Z, Meyer KB, Teichmann SA, Park JE. BBKNN: fast batch alignment of single cell transcriptomes. Bioinformatics. 2020 Feb 1;36(3):964-965.
6- Korsunsky I, Millard N, Fan J, Slowikowski K, Zhang F, Wei K, Baglaenko Y, Brenner M, Loh PR, Raychaudhuri S. Fast, sensitive and accurate integration of single-cell data with Harmony. Nat Methods. 2019 Dec;16(12):1289-1296.
7- Butler A, Hoffman P, Smibert P, Papalexi E, Satija R. Integrating single-cell transcriptomic data across different conditions, technologies, and species. Nat Biotechnol. 2018 Jun;36(5):411-420.
8- Van Dijk D, Sharma R, Nainys J, Yim K, Kathail P, Carr AJ, Burdziak C, Moon KR, Chaffer CL, Pattabiraman D, Bierie B, Mazutis L, Wolf G, Krishnaswamy S, Pe'er D. Recovering Gene Interactions from Single-Cell Data Using Data Diffusion. Cell. 2018 Jul 26;174(3):716-729.e27.
9- Gong W, Kwak IY, Pota P, Koyano-Nakagawa N, Garry DJ. DrImpute: imputing dropout events in single cell RNA sequencing data. BMC Bioinformatics. 2018 Jun 8;19(1):220.
10- Chen M, Zhou X. VIPER: variability-preserving imputation for accurate gene expression recovery in single-cell RNA sequencing studies. Genome Biol. 2018 Nov 12;19(1):196.
11- Li WV, Li JJ. An accurate and robust imputation method scImpute for single-cell RNA-seq data. Nat Commun. 2018 Mar 8;9(1):997.

12- Altman, Naomi S. An introduction to kernel and nearest-neighbor nonparametric regression. The American Statistician. 1992, 46 (3): 175–185.

13- Jiarui Ding, Xian Adiconis, Sean K. Simmons, Monika S. Kowalczyk, Cynthia C. Hession, Nemanja D. Marjanovic, Travis K. Hughes, Marc H. Wadsworth, Tyler Burks, Lan T. Nguyen, John Y. H. Kwon, Boaz Barak, William Ge, Amanda J. Kedaigle, Shaina Carroll, Shuqiang Li, Nir Hacohen, Orit Rozenblatt-Rosen, Alex K. Shalek, Alexandra-Chloé Villani, Aviv Regev, View ORCID ProfileJoshua Z. Levin. Systematic comparative analysis of single cell RNA-sequencing methods. 2019 Mar23; doi: https://doi.org/10.1101/632216

Figures

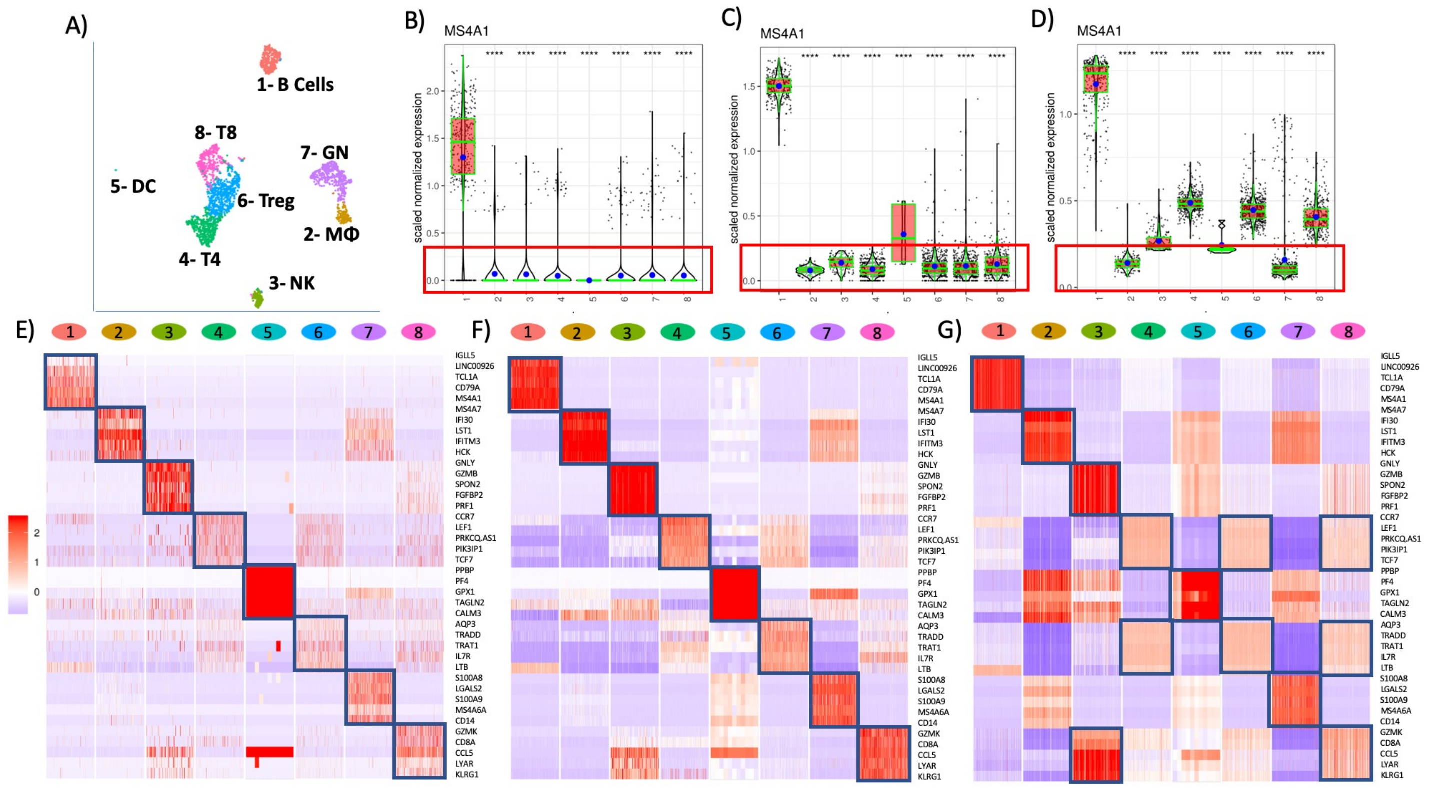

Figure 1

A) UMAP plot of a sample PBMC data showing 8 clusters/cell types.

B) Boxplot showing the expression of MS4A1 in the raw expression matrix. Big blue dots in the box plots are mean and the green lines show the median. This plot is indicating many cells with zero expression (drop-outs) and comparable average expressions in all the clusters except cluster one as seen in the red box.

C) Boxplot showing the expression of MS4A1 in the coverage corrected (CC) data showing similar average expression points (blue dot) and medians (green line) as seen in the red box. Only cluster five is out of the red box and it’s because there are only seven cells in this cluster.

D) Boxplot showing the expression of MS4A1 in the data imputed with MAGIC showing varying average expression points (blue dot) and medians (green line). As seen in the plot the means and medians don’t fit in the red box.

E) Heatmap showing the expression patterns of the top 5 markers per cluster in the raw data which includes many drop-outs.
F) Heatmap showing the expression patterns of the top 5 markers per cluster in the coverage corrected (CC) data showing similar patters to the raw data.
G) Heatmap showing the expression patterns of the top 5 markers per cluster in the data imputed with MAGIC showing differing patters to the raw data. As shown, markers for clusters 4, 6 and 8 are not distinguishable, and the markers for cluster 8 are expressed more in cluster 3. This indicates that MAGIC imputation introduces noise to the data.

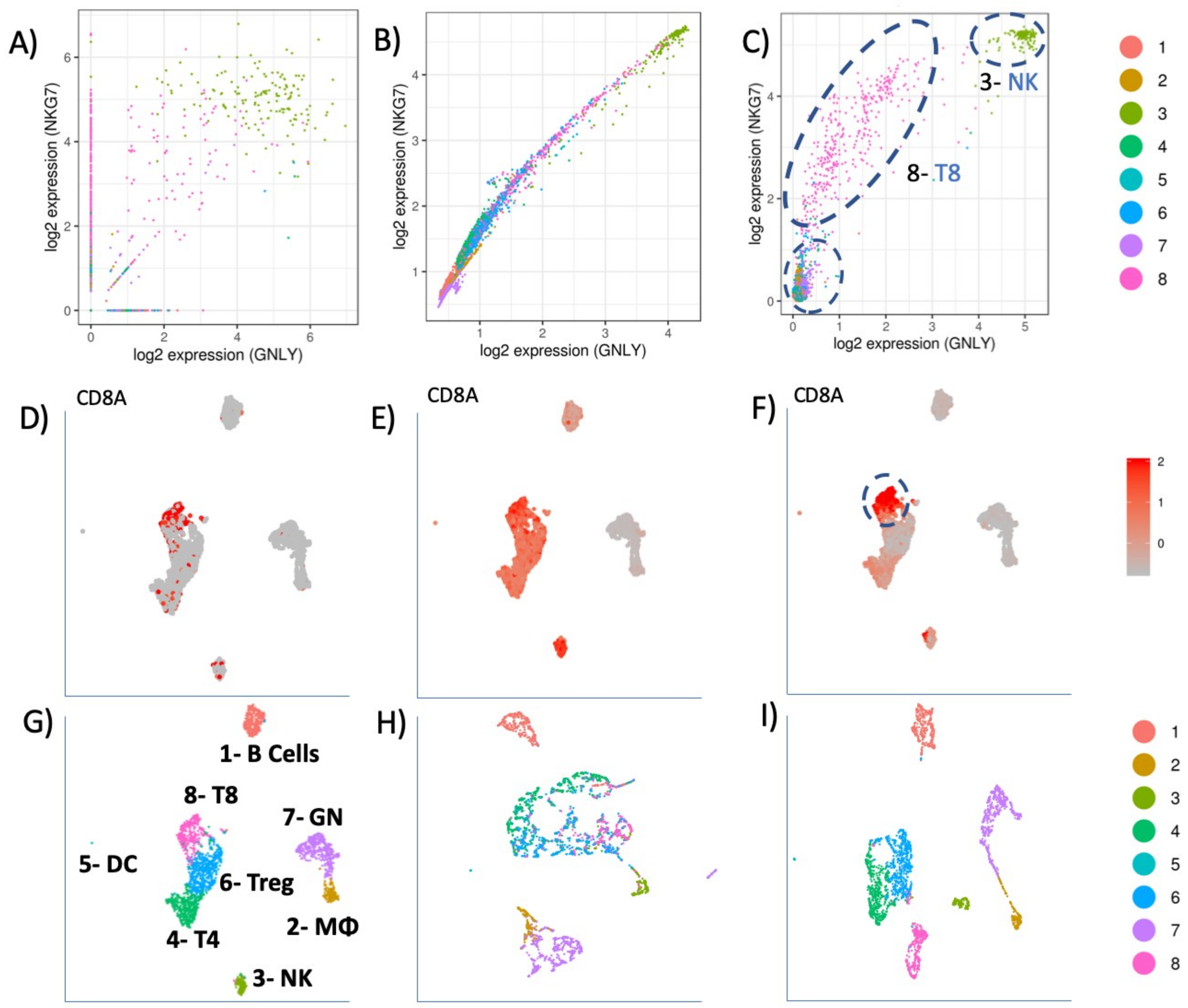

Figure 2

A) Gene-gene correlation for GNLY and NKG7 in the raw data showing many genes with zero counts making it hard to calculate correlation.
B) Gene-gene correlation for GNLY and NKG7 in the data imputed with MAGIC showing that NK cells and CD8+ T cells are mixed with other clusters.

C) Gene-gene correlation for GNLY and NKG7 in the coverage corrected (CC) data showing that NK cells and CD8+ T cells could be clearly distinguished as they express these genes at much higher levels compared to other clusters.
D) UMAP plot showing the expression of CD8A gene in the raw data which includes many drop-outs.
E) UMAP plot showing the expression of CD8A gene in the data imputed with MAGIC, showing that it is hard to distinguish CD8-positive cells from other cells.
F) UMAP plot showing the expression of CD8A gene in the coverage corrected (CC) data, showing CD8-positve T cells can be clearly distinguished.
G) UMAP plot showing the clustering based on non-imputed data.
H) UMAP plot showing the clustering based on imputed data using MAGIC. As seen in the plot CD8-positive T cells are mixed with CD4-positive T cells.
I) UMAP plot showing the clustering based on coverage corrected (CC) data using iCellR. As seen in the plot CD8-positive T cells are well distinguished from CD4-positive cells also showing more resolution so if clustered again more clusters and details can be seen.

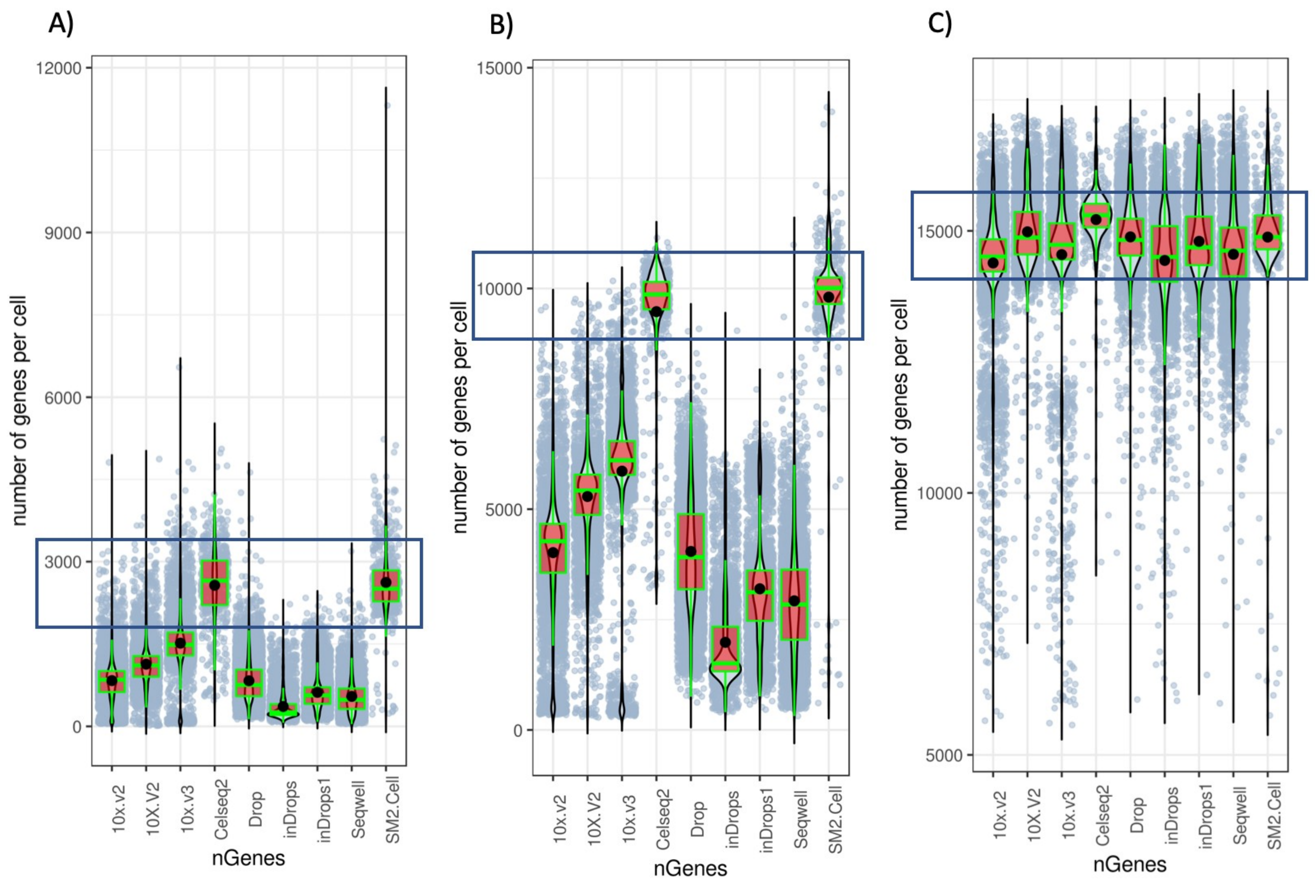

Figure 3

A) Box plots of the gene coverages per cell in the raw data, showing that some technologies have more coverage.
B) Box plots of the gene coverages per cell in the coverage corrected (CC) data, showing that the coverages are higher but similar patterns or trends are maintained. For instance, Cel-Seq2 and SMART-Seq2 have the highest coverages.
C) Box plots of the gene coverages per cell in the combined coverage corrected (CCCA) data, showing that the coverages across all the technologies are harmonized.

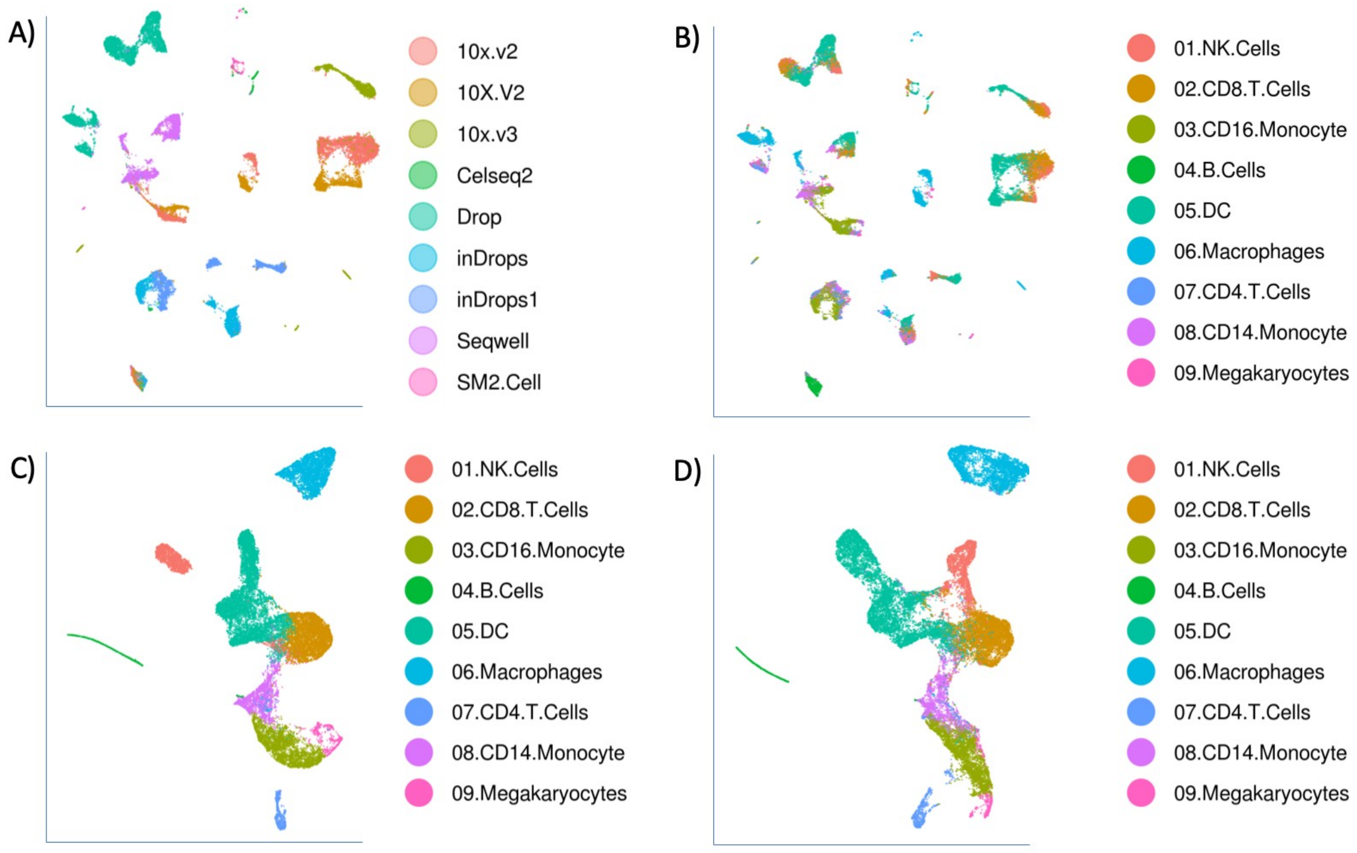

Figure 4
A) UMAP plot of unaligned data, showing the clusters and technologies.
B) UMAP plot of unaligned data, showing the cell types from different technologies are not aligned and some colors are seen in multiple clusters.
C) UMAP plot of CPCA aligned data, showing that cell types from different technologies are well aligned.
D) UMAP plot of CCCA aligned data, showing that cell types from different technologies are well aligned.

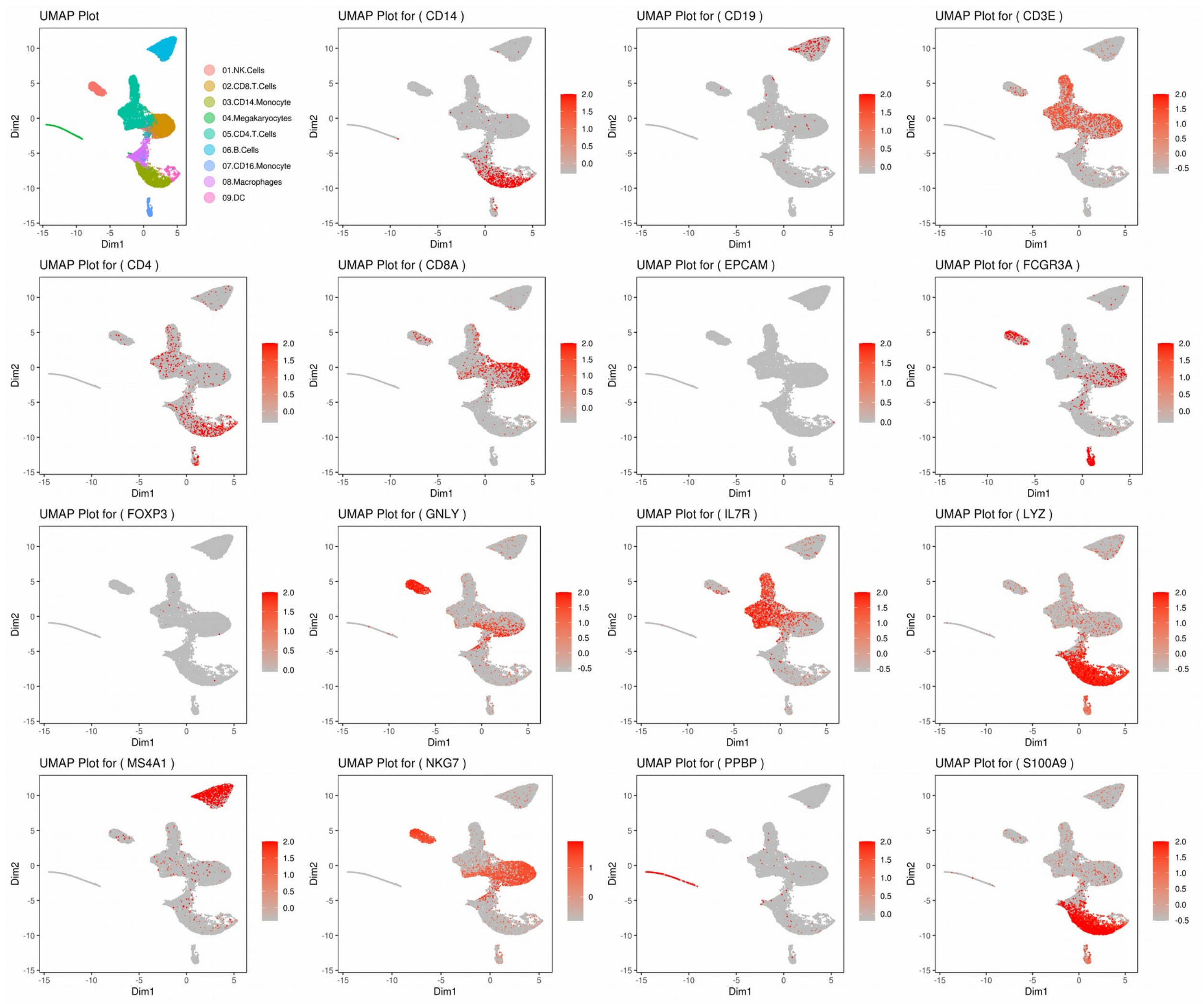

Figure S1

UMAP plots of some gene markers after alignment (CPCA) showing that the samples are properly aligned based on cell types.

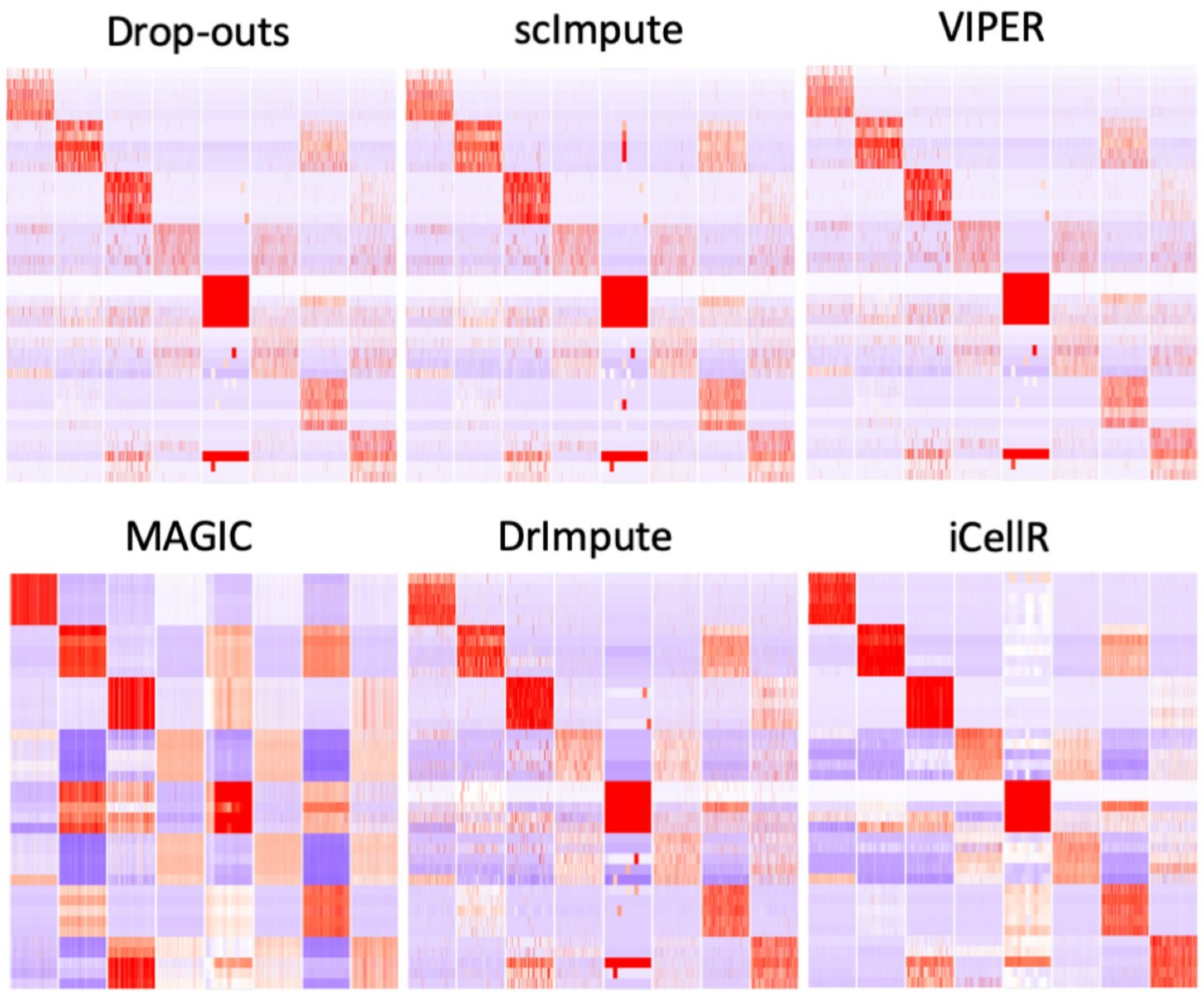

Figure S2

Heatmaps of marker genes before (with high drop-outs) and after imputation (reduced drop-outs). As shown in the figure, iCellR not only fills more drop-outs but also follows similar expression patterns as seen in the original data (Drop-outs).

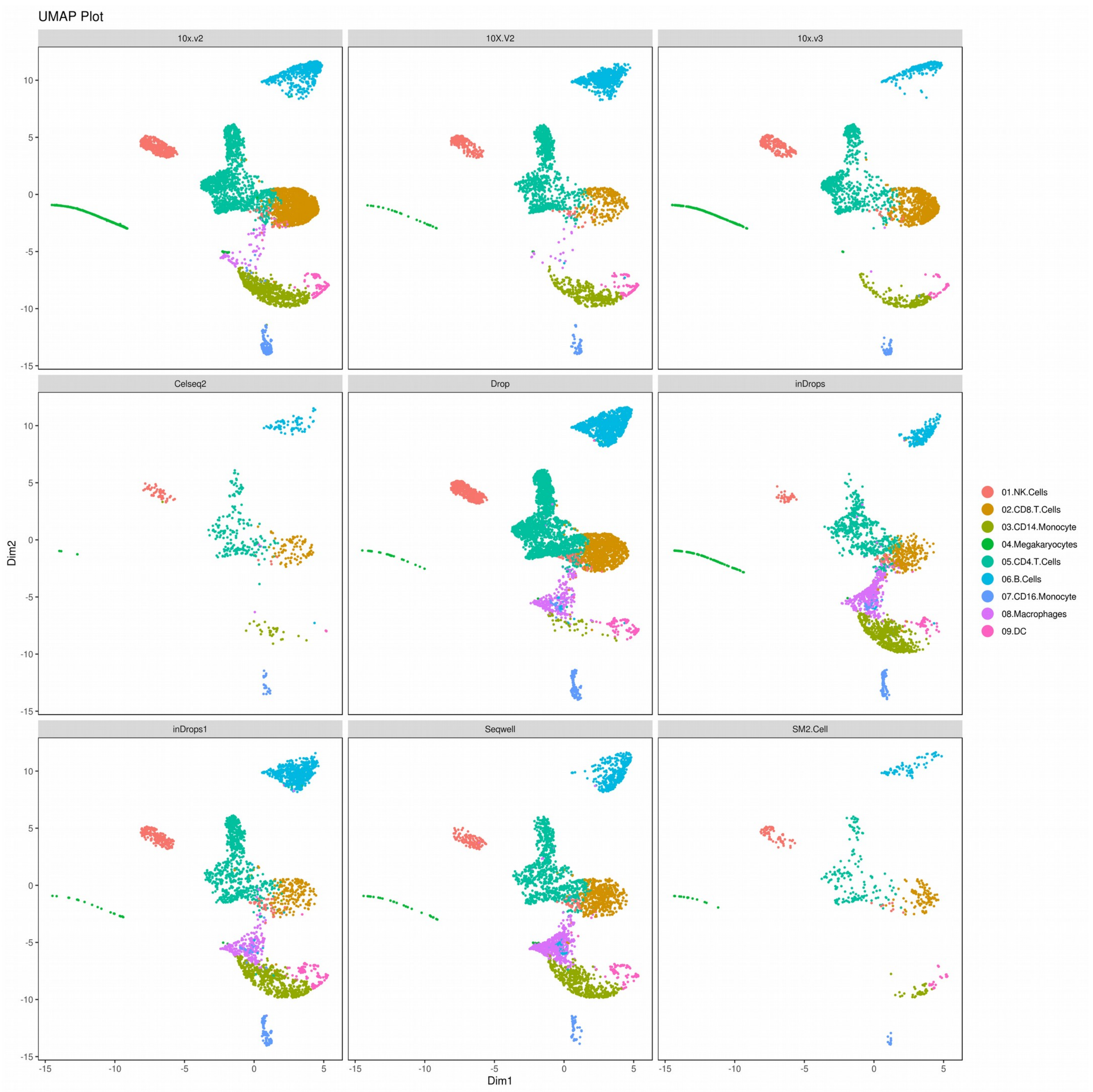

Figure S3

UMAP plots showing the alignments of the nine samples.

Supplementary Codes 1

```
sample.file.url =
"https://genome.med.nyu.edu/results/external/iCellR/data/pbmc3k_
filtered_gene_bc_matrices.tar.gz"

# download the file
download.file(url = sample.file.url,
     destfile = "pbmc3k_filtered_gene_bc_matrices.tar.gz",
     method = "auto")

# unzip the file.
untar("pbmc3k_filtered_gene_bc_matrices.tar.gz")

library("iCellR")
my.data <- load10x("filtered_gene_bc_matrices/hg19/")

my.obj <- make.obj(my.data)

my.obj <- qc.stats(my.obj,
    s.phase.genes = s.phase,
    g2m.phase.genes = g2m.phase)


png('plot1_QC_stats.png',width = 15, height = 6, units = 'in',
res = 300)
stats.plot(my.obj,
    plot.type = "all.in.one",
    out.name = "UMI-plot",
    interactive = FALSE,
    cell.color = "slategray3",
    cell.size = 1,
    cell.transparency = 0.5,
    box.color = "red",
    box.line.col = "green")
dev.off()

my.obj <- cc(my.obj, s.genes = s.phase, g2m.genes = g2m.phase)

png("cellCycle.png")
pie(table(my.obj@stats$Phase))
dev.off()

# filter
my.obj <- cell.filter(my.obj,
min.mito = 0,
max.mito = 0.05 ,
min.genes = 200,
max.genes = 4000,
min.umis = 0,
max.umis = Inf)

# normalize
my.obj <- norm.data(my.obj, norm.method = "ranked.glsf",
top.rank = 500)

################################################################
# make model
my.obj <- gene.stats(my.obj, which.data = "main.data")
#head(my.obj@gene.data[order(my.obj@gene.data$numberOfCells,
decreasing = T),])
my.gene.data <- my.obj@gene.data
head(my.gene.data)
write.table((my.gene.data), file="gene.data.tsv", sep="\t",
row.names =F)

png('plot4_gene.model.png',width = 6, height = 6, units = 'in',
res = 300)
make.gene.model(my.obj, my.out.put = "plot",
    dispersion.limit = 1.5,
    base.mean.rank = 500,
    no.mito.model = T,
    mark.mito = T,
    interactive = F,
    no.cell.cycle = T,
    out.name = "gene.model")
dev.off()

my.obj <- make.gene.model(my.obj, my.out.put = "data",
    dispersion.limit = 1.5,
    base.mean.rank = 500,
    no.mito.model = T,
    mark.mito = T,
    interactive = F,
    no.cell.cycle = T,
    out.name = "gene.model")

# run PCA
my.obj <- run.pca(my.obj, method = "gene.model", gene.list =
my.obj@gene.model,data.type = "main")
```

```
png('plot5_Opt_Number_Of_PCs.png',width = 6, height = 6, units =
'in', res = 300)
opt.pcs.plot(my.obj)
dev.off()

#### run 2 pass PCA
####
length(my.obj@gene.model)
my.obj <- find.dim.genes(my.obj, dims = 1:20,top.pos = 20,
top.neg = 20)
length(my.obj@gene.model)

# second round PC
my.obj <- run.pca(my.obj, method = "gene.model", gene.list =
my.obj@gene.model,data.type = "main")

# cluster
library(Rphenograph)
my.obj <- run.phenograph(my.obj,k = 100,dims = 1:10)

# tSNE, UMAP
my.obj <- run.pc.tsne(my.obj, dims = 1:10)
my.obj <- run.umap(my.obj, dims = 1:10)
my.obj <- run.impute(my.obj,dims = 1:10,data.type = "pca", nn =
10)
########################### cluster avrage expression
my.obj <- clust.avg.exp(my.obj)
head(my.obj@clust.avg)
write.table((my.obj@clust.avg),file="clust.avg.tsv",sep="\t",
row.names =F)

# save object
save(my.obj, file = "my.obj.Robj")

# you can load object to continue in the future!
load("my.obj.Robj")

# plot

A= cluster.plot(my.obj,plot.type = "pca",interactive =
F,cell.size = 0.2)
B= cluster.plot(my.obj,plot.type = "tsne",interactive =
F,cell.size = 0.2)
C= cluster.plot(my.obj,plot.type = "umap",interactive =
F,cell.size = 0.2)
```

```
D= cluster.plot(my.obj,plot.type = "umap",col.by =
"conditions",interactive = F,cell.size = 0.2)

library(gridExtra)
png('AllClusts.png', width = 8, height = 8, units = 'in', res =
300)
grid.arrange(A,B,C,D)
dev.off()

### plot markers
png('heatmap_gg.png', width = 20, height = 20, units = 'in', res
= 300)
heatmap.gg.plot(my.obj, gene = readLines("genes.txt"),
interactive = F, cluster.by = "clusters", cell.sort = TRUE)
dev.off()

png('heatmap_gg_imputed.png', width = 20, height = 20, units =
'in', res = 300)
heatmap.gg.plot(my.obj, gene = readLines("genes.txt"),
interactive = F, cluster.by = "clusters", data.type = "imputed",
cell.sort = TRUE)
dev.off()

png('heatmap_gg_imputed_sudo.png', width = 20, height = 20,
units = 'in', res = 300)
heatmap.gg.plot(my.obj, gene = readLines("genes.txt"),
interactive = F, cluster.by = "none", data.type = "imputed",
cell.sort = TRUE)
dev.off()

### more plots

A <- gene.plot(my.obj, gene = "MS4A1",
    plot.type = "scatterplot",
    interactive = F,
    data.type = "imputed",
    out.name = "scatter_plot")
# PCA 2D
B <- gene.plot(my.obj, gene = "MS4A1",
    plot.type = "scatterplot",
    interactive = F,
    out.name = "scatter_plot",
    data.type = "imputed",
    plot.data.type = "umap")

# Box Plot
```

```
C <- gene.plot(my.obj, gene = "MS4A1",
    box.to.test = 0,
    box.pval = "sig.signs",
    col.by = "clusters",
    plot.type = "boxplot",
    interactive = F,
    data.type = "imputed",
    out.name = "box_plot")

# Bar plot (to visualize fold changes)
D <- gene.plot(my.obj, gene = "MS4A1",
    col.by = "clusters",
    plot.type = "barplot",
    interactive = F,
    data.type = "imputed",
    out.name = "bar_plot")

library(gridExtra)
png('gene.plots.png', width = 8, height = 8, units = 'in', res =
300)
grid.arrange(A,B,C,D)
dev.off()

###################
dir.create("MAGIC")
setwd("MAGIC/")

DATA <- my.obj@main.data
DATA <- DATA[ rowSums(DATA) > 0, ]
DATA <- as.data.frame(t(DATA))
library(Rmagic)
library(phateR)
library(viridis)
data_MAGIC <- magic(DATA,
                        genes = "all_genes", k = 10, alpha = 15,
                        init = NULL, t.max = 20,
                        verbose = 1, n.jobs = 1,
                        seed = NULL)
DATA <- as.data.frame(t(data_MAGIC$result))
DATA <- round(DATA, digits = 3)
my.obj@imputed.data <- DATA
save(my.obj, file = "my.obj.Robj")

pdf("MS4A1.pdf")
gene.plot(my.obj, gene = "MS4A1",
    plot.type = "boxplot",
    interactive = F, data.type = "imputed")
dev.off()

pdf("CD8A.pdf")
gene.plot(my.obj, gene = "CD8A",plot.data.type = "umap",
    interactive = F, data.type = "imputed")
dev.off()

png('heatmap_gg_imputed_genes.png', width = 20, height = 20,
units = 'in', res = 300)
heatmap.gg.plot(my.obj, gene = readLines("genes.txt"),
interactive = F, cluster.by = "clusters", data.type =
"imputed",cell.sort = F)
dev.off()
```

Supplementary Codes 2

```
sample.file.url =
"https://genome.med.nyu.edu/results/external/iCellR/data/pbmc_da
ta/my.obj.Robj"

# download the file
download.file(url = sample.file.url,
     destfile = "my.obj.Robj",
     method = "auto")

library(iCellR)
load("my.obj.Robj")

my.obj

#####
#### QC
my.obj <- qc.stats(my.obj,
s.phase.genes = s.phase,
g2m.phase.genes = g2m.phase, which.data = "raw.data")

summary(my.obj@stats)

png('plot1_QC_stats.png',width = 15, height = 6, units = 'in',
res = 300)
stats.plot(my.obj,
    plot.type = "all.in.one",
    out.name = "UMI-plot",
    interactive = FALSE,
    cell.color = "slategray3",
    cell.size = 1,
    cell.transparency = 0.5,
    box.color = "red",
    box.line.col = "green")
dev.off()
###

my.obj <- cc(my.obj, s.genes = s.phase, g2m.genes = g2m.phase)

png("cellCycle.png")
pie(table(my.obj@stats$Phase))
dev.off()
```

```
########## PCA
my.obj <- run.pca(my.obj, top.rank = 2000)

my.obj <- find.dim.genes(my.obj, dims = 1:30,top.pos = 20,
top.neg = 20)
length(my.obj@gene.model)

########### Batch alignment

my.obj <- iba(my.obj,dims = 1:30, k = 10,ba.method = "CPCA",
method = "gene.model", gene.list = my.obj@gene.model)

my.obj <- run.impute(my.obj,dims = 1:10,data.type = "pca", nn =
10)

# or CCCA
# my.obj <- iba(my.obj,dims = 1:30, k = 10, ba.method = "CCCA",
method = "gene.model", gene.list = my.obj@gene.model)

# or PCA
# my.obj <- run.pca(my.obj, method = "gene.model", gene.list =
my.obj@gene.model,data.type = "main")

##############################################
####

my.obj <- run.pc.tsne(my.obj, dims = 1:10)
my.obj <- run.umap(my.obj, dims = 1:10)

save(my.obj, file = "my.obj.Robj")

 library(gridExtra)
A= cluster.plot(my.obj,plot.type = "umap",interactive =
F,cell.size = 0.1)
B= cluster.plot(my.obj,plot.type = "tsne",interactive =
F,cell.size = 0.1)
C= cluster.plot(my.obj,plot.type = "umap",col.by =
"conditions",interactive = F,cell.size = 0.1)
D=cluster.plot(my.obj,plot.type = "tsne",col.by =
"conditions",interactive = F,cell.size = 0.1)

png('AllClusts.png', width = 12, height = 12, units = 'in', res
= 300)
grid.arrange(A,B,C,D)
dev.off()
```

```
png('AllConds_clusts.png', width = 15, height = 15, units =
'in', res = 300)
cluster.plot(my.obj,
              cell.size = 0.5,
              plot.type = "umap",
              cell.color = "black",
              back.col = "white",
              cell.transparency = 1,
              clust.dim = 2,
              interactive = F,cond.facet = T)
dev.off()

png('AllConds_clusts_tsne.png', width = 15, height = 15, units =
'in', res = 300)
cluster.plot(my.obj,
              cell.size = 0.5,
              plot.type = "tsne",
              cell.color = "black",
              back.col = "white",
              cell.transparency = 1,
              clust.dim = 2,
              interactive = F,cond.facet = T)
dev.off()


genelist =
c("PPBP","LYZ","MS4A1","GNLY","FCGR3A","NKG7","CD14","S100A9","C
D3E","CD8A","CD4","CD19","IL7R","FOXP3","EPCAM")

rm(list = ls(pattern="PL_"))

cond.to.plot = NULL

for(i in genelist){
    MyPlot <- gene.plot(my.obj, gene = i,
     interactive = F,
     conds.to.plot = conds.to.plot,
     cell.size = 0.1,
     data.type = "main",
     plot.data.type = "umap",
     scaleValue = T,
     min.scale = -2.5,max.scale = 2.0,
     cell.transparency = 1)
    NameCol=paste("PL",i,sep="_")
    eval(call("<-", as.name(NameCol), MyPlot))
}

UMAP = cluster.plot(my.obj,plot.type = "umap",interactive =
F,cell.size = 0.1, anno.size=5)
library(cowplot)
filenames <- ls(pattern="PL_")
filenames <- c("UMAP", filenames)

png('genes.png',width = 18, height = 15, units = 'in', res =
300)
plot_grid(plotlist=mget(filenames))
dev.off()
```