Single cell RNA sequencing analysis (BT samples)

Shiju Sisobhan

2024-07-18

Load required packages

```
library(dplyr)
library(Seurat)
library(patchwork)
library(ggplot2)
library(Matrix)
library(data.table)
```

Set the working directory

```
setwd('X:/Sequencing_data/Clark_seq_data_6-12-2024/01.RawData/BT_all')
```

Read the .mtx file created by kallisto-bus tools and transpose it such that genes are in the row and cells are in the column. Then rename the row and column using gene symbol and cell bar code. Here we again rename the column by meaningfull names susch as cell1, cell2....

```
# Read the .mtx file
sparse_matrix <- readMM("KB_BT_all/counts_unfiltered/cells_x_genes.mtx")
sparse_matrix<-t(sparse_matrix)

# Read the genes and barcodes files
genes <- fread("KB_BT_all/counts_unfiltered/cells_x_genes.genes.names.txt", header = FALSE)
barcodes <- fread("KB_BT_all/counts_unfiltered/cells_x_genes.barcodes.txt", header = FALSE)

# Convert to vectors
gene_names <- genes$V1
barcode_names <- barcodes$V1

# Create a vector of meaningful names
meaningful_names <- paste0("Cell", seq_along(barcode_names))

# Assign row names (genes) and column names (barcodes) to the matrix
rownames(sparse_matrix) <- gene_names
colnames(sparse_matrix)<-meaningful_names
Count_matrix <- as.matrix(sparse_matrix)
head(Count_matrix)
```

```
##             Cell1 Cell2 Cell3 Cell4 Cell5 Cell6
## INE-1{}6078     0     0     0     0     0     0
```

```
## CG5846          0     0     0     0     0     0
## TBC1D23         0  1104     0     0     0     0
## Fatp1           0   150     0     0     0     0
## INE-1{}2524     0     0     0     0     0     0
## ApepP           0  1102     0    36     0    29
```

Create a seurat object with the non-normalized count matrix

```
seurat <- CreateSeuratObject(Count_matrix, project="BT")
```

Quality control

filter out :

1.Cells with too few genes (empty droplets or low quality of cell)

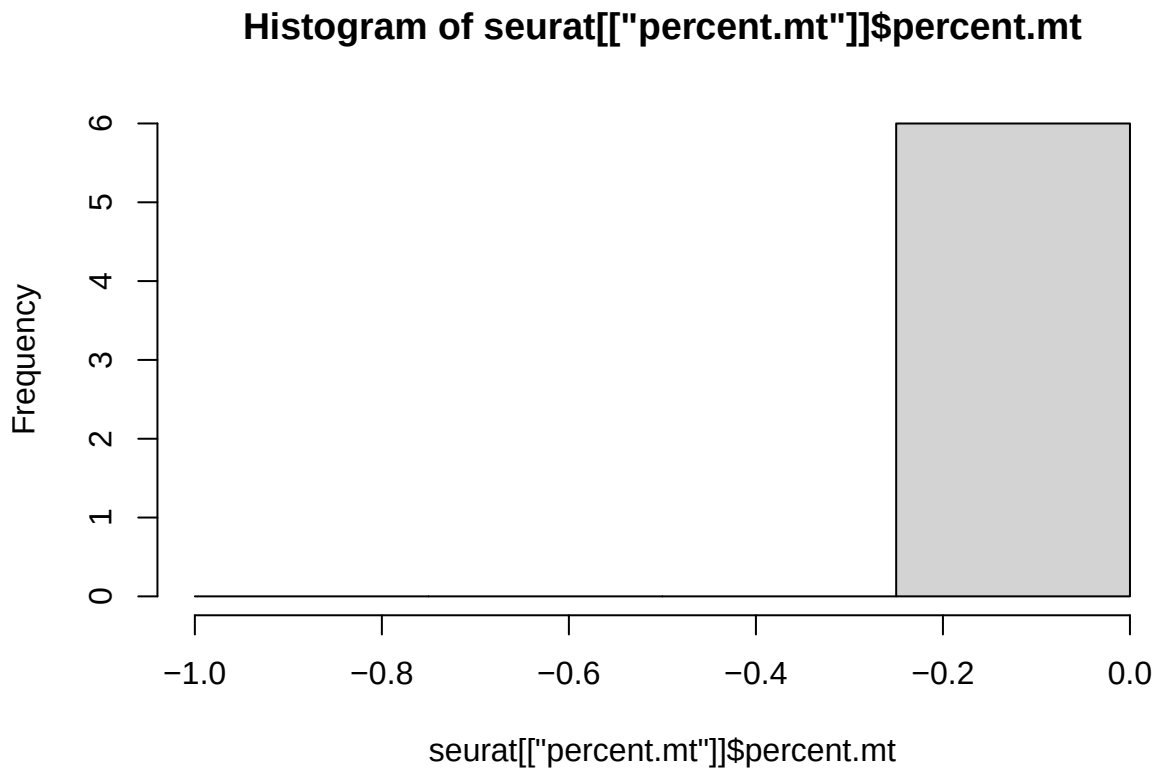
2.Cells with too many genes (duplets or multiplets- with in a droplet more than one cell)

3.Cells with high mitochondrial transcript percentage

Metrics considered for quality control: RNA count (UMI), Feature count (Gene count), Mitochondria content.

Approach : Identify and discard outliers (Different samples may require different cutoffs)

```
seurat[["percent.mt"]] <- PercentageFeatureSet(seurat, pattern = "^MT[-\\.]")
hist(seurat[["percent.mt"]]$percent.mt, breaks = c(-1,-0.75,-0.5,-0.25, 0))
```



Most of the mitochondrial transcript percentage are less than 5%, which is good.

```
seurat@meta.data
```

```
##       orig.ident nCount_RNA nFeature_RNA percent.mt
## Cell1         BT    4037678         2115          0
## Cell2         BT    5830950         3944          0
## Cell3         BT    3051329         2712          0
## Cell4         BT    5895798         2982          0
## Cell5         BT    2630004         1881          0
## Cell6         BT    3572528         2286          0
```

nCount_RNA : Number of unique RNA molecule

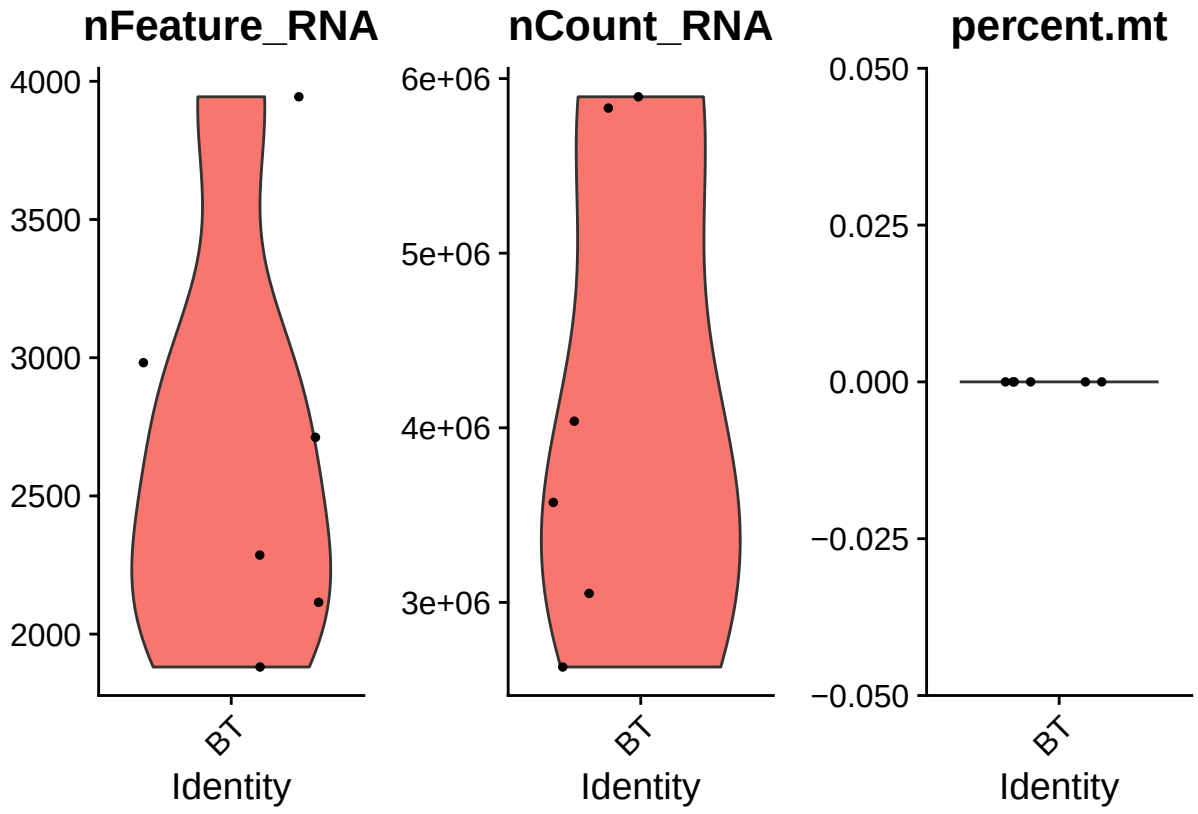
nFeature_RNA : Number of genes

percent.mt : mitochondrial transcript percentage which are 0 (High quality data)

For good quality percent.mt should be between 0-5%

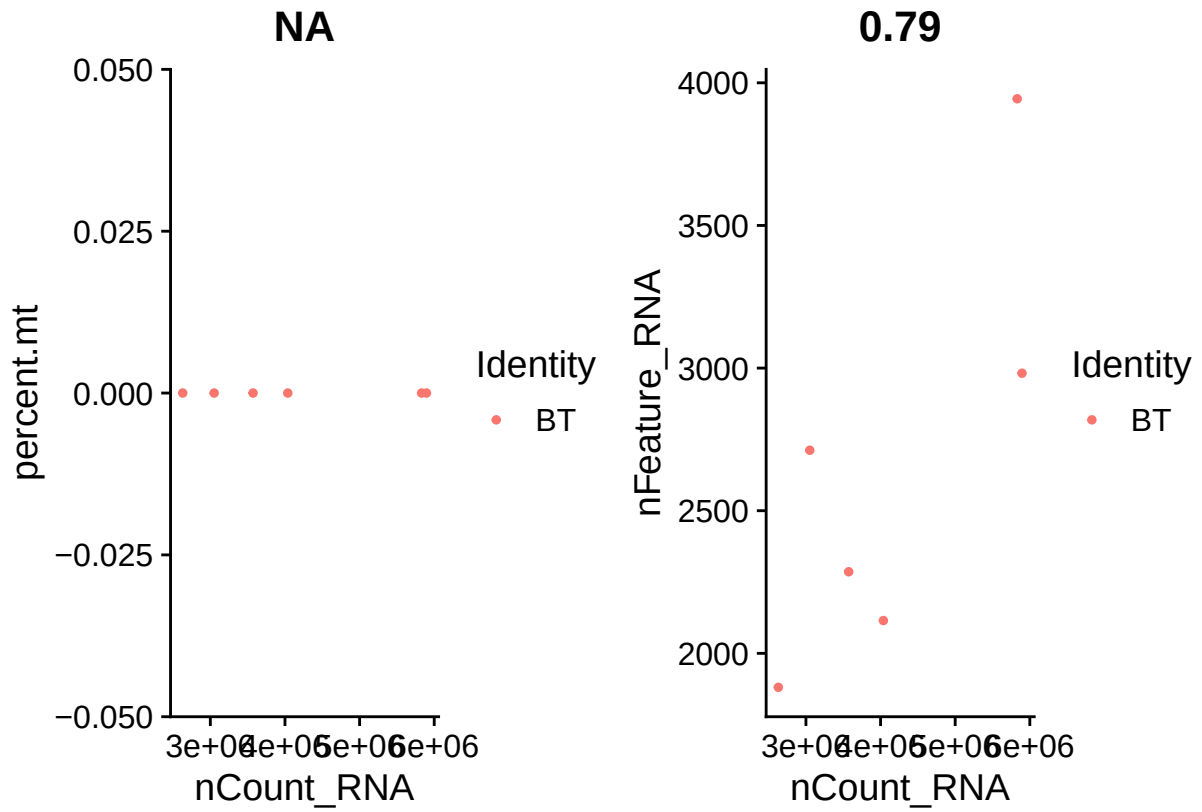
Visualise nCount_RNA, nFeature_RNA and percent.mt

```
VlnPlot(seurat, features = c("nFeature_RNA", "nCount_RNA", "percent.mt"), ncol = 3, pt.size=1)
```



Correlation plot Visualize the correlation between nFeature_RNA", "nCount_RNA", "percent.mt"

```
plot1 <- FeatureScatter(seurat, feature1 = "nCount_RNA", feature2 = "percent.mt")
plot2 <- FeatureScatter(seurat, feature1 = "nCount_RNA", feature2 = "nFeature_RNA")
plot1 + plot2
```



nFeature_RNA", "nCount_RNA" correlated & these 2 with "percent.mt" are not correlated.Due to the correlation of gene number and transcript number, we only need to set a cutoff to either one of these metrics, combined with an upper threshold of mitochondrial transcript percentage, for the QC.

```
seurat <- subset(seurat, subset = nFeature_RNA > 500 & nFeature_RNA < 5000 & percent.mt < 5)
```

In this experiment, the above line does not filter out any cells. However user can set any threshold value according to your need.

Normalization

Aim : To make the data comparable across cells.

Amount of captured RNA is different from cell to cell. Therefore not directly compare the number of captured transcripts for each gene between cells.

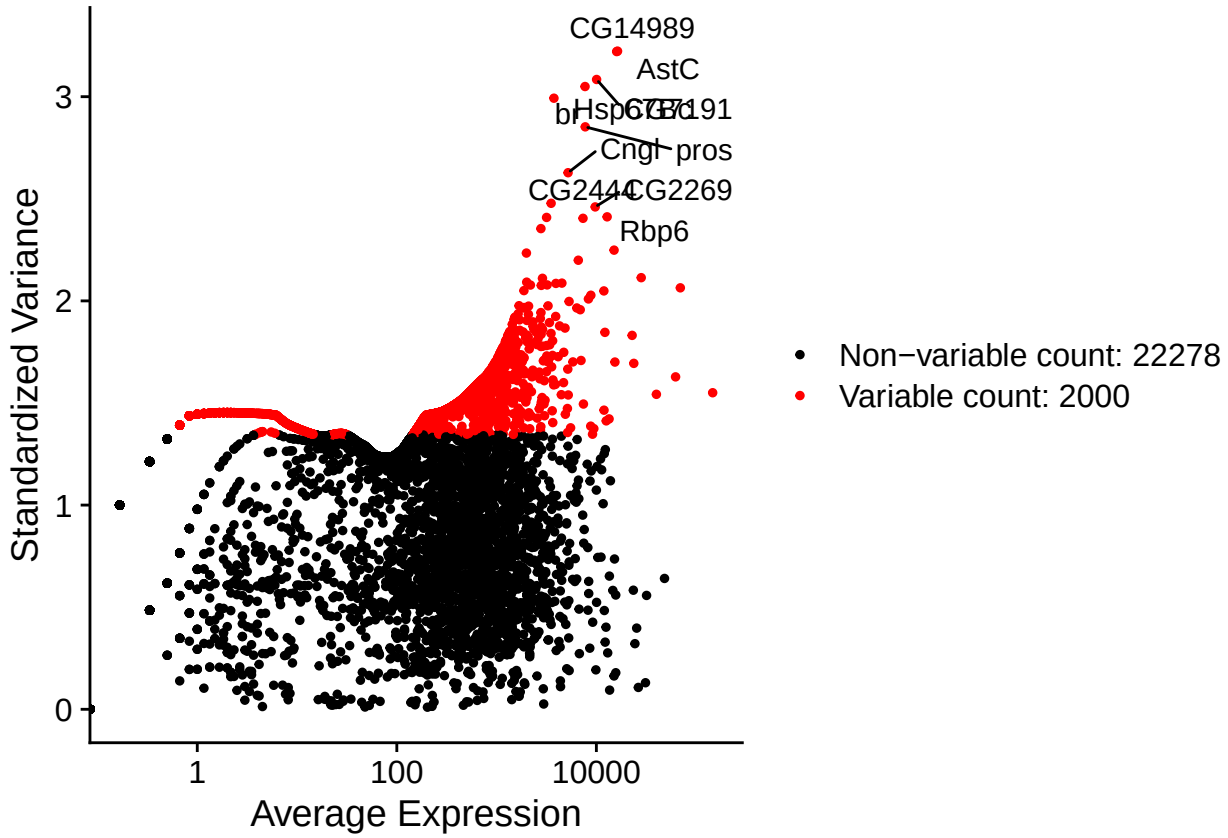
count per million (CPM) Normalization : (gene UMI count/total UMI count) x 1,000,000

```
seurat <- NormalizeData(seurat)
```

Select Highly variable genes

Cellular heterogeneity of samples. Genes with low expression levels, and those with similar expression levels across all cells are not very informative. Identification of highly variable features/genes, which are genes with the most varied expression levels across cells. Following code will select top 2000 most variable genes, which are our genes of interest.

```
seurat <- FindVariableFeatures(seurat, nfeatures = 2000)

top_features <- head(VariableFeatures(seurat), 10)
plot1 <- VariableFeaturePlot(seurat)
plot2 <- LabelPoints(plot = plot1, points = top_features, repel = TRUE)
plot2
```



Data scaling

Since different genes have different base expression levels and distributions, a scaling is applied to the data using the selected features.

```
seurat <- ScaleData(seurat)
```

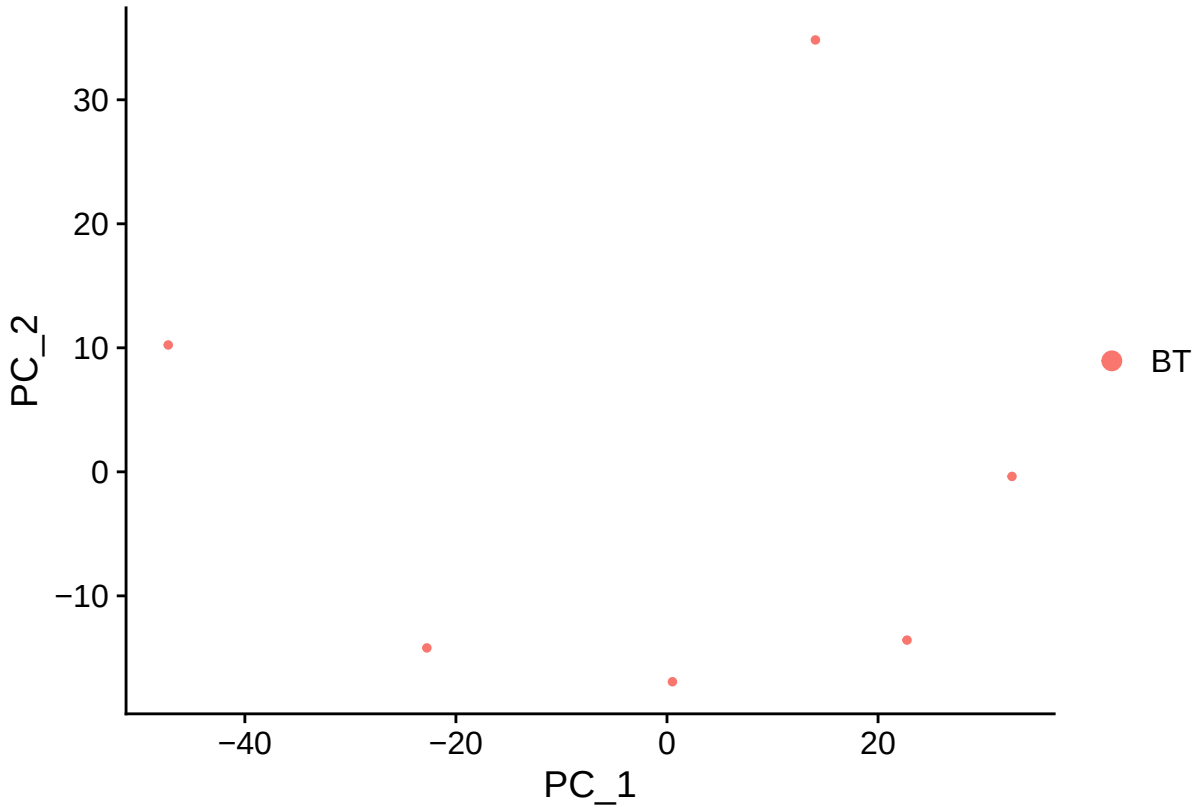
one can also remove unwanted sources of variation from the data set by setting the parameter

```
seurat <- ScaleData(seurat, vars.to.regress = c("nFeature_RNA", "percent.mt"))
```

Linear dimensionality reduction- principal component analysis (PCA)

The number of principal components (PCs) that one can calculate for a data set is equal to the number of highly variable genes or the number of cells, whichever value is smaller.

```
seurat <- RunPCA(seurat, npcs = 3) # adding PCA to seurat object
# Plot the cells in the 2D PCA projection
DimPlot(seurat, reduction = "pca")
```



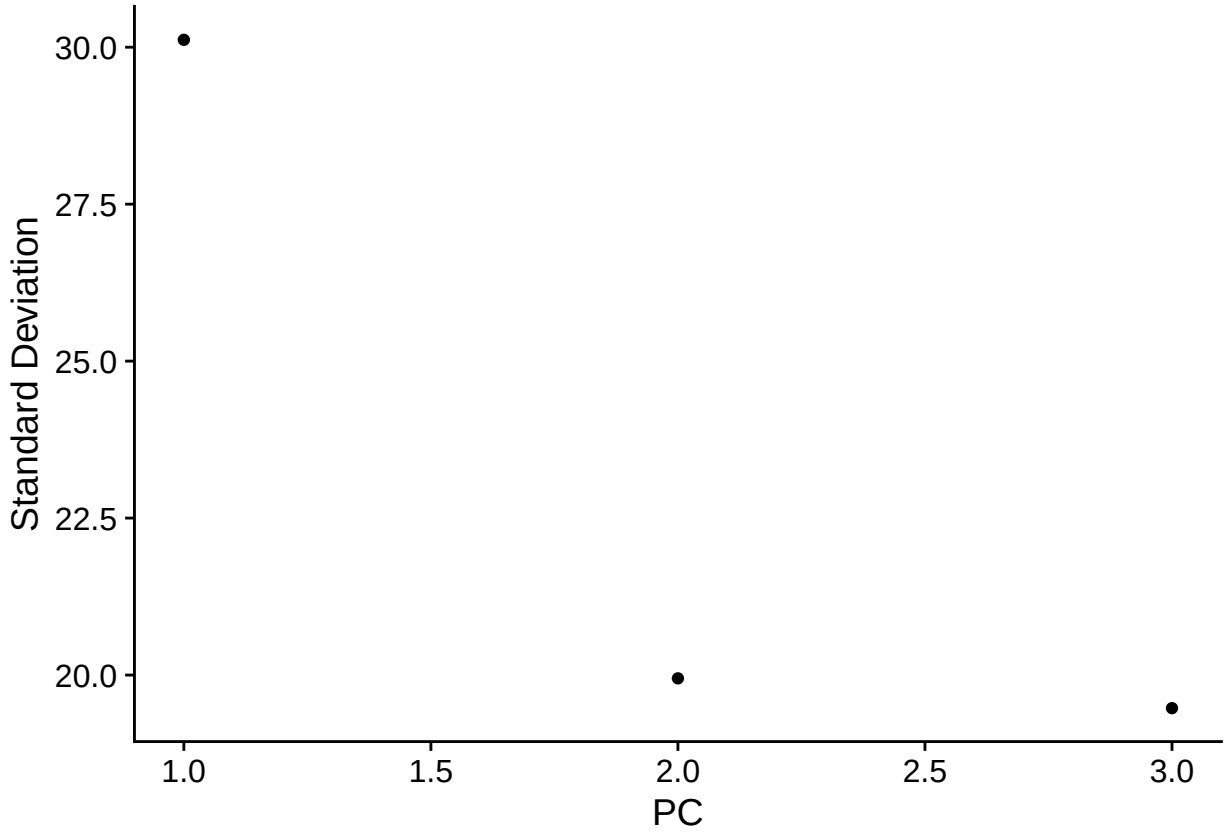
Print top genes contributing PC1, PC2

```
print(seurat[['pca']],dims=1:2, nfeatures=10)
```

```
## PC_ 1
## Positive:  5-HT1A, oys, D19A, Top1, Map60, HPS1, CG4972, PR-Set7, dome, sotv
## Negative:  Asator, aPKC, zfh2, kn, CG14535, CG8248, CG17807, rdgA, cyst, CG32700
## PC_ 2
## Positive:  tral, ytr, Vamp7, Prp40, prd1, CG4390, CG3309, Usp7, Jafrac1, CG9395
## Negative:  Fer1HCH, Vha26, Ank, jim, mamo, C3G, Ggamma30A, Pdp1, Tbc1d15-17, BuGZ
```

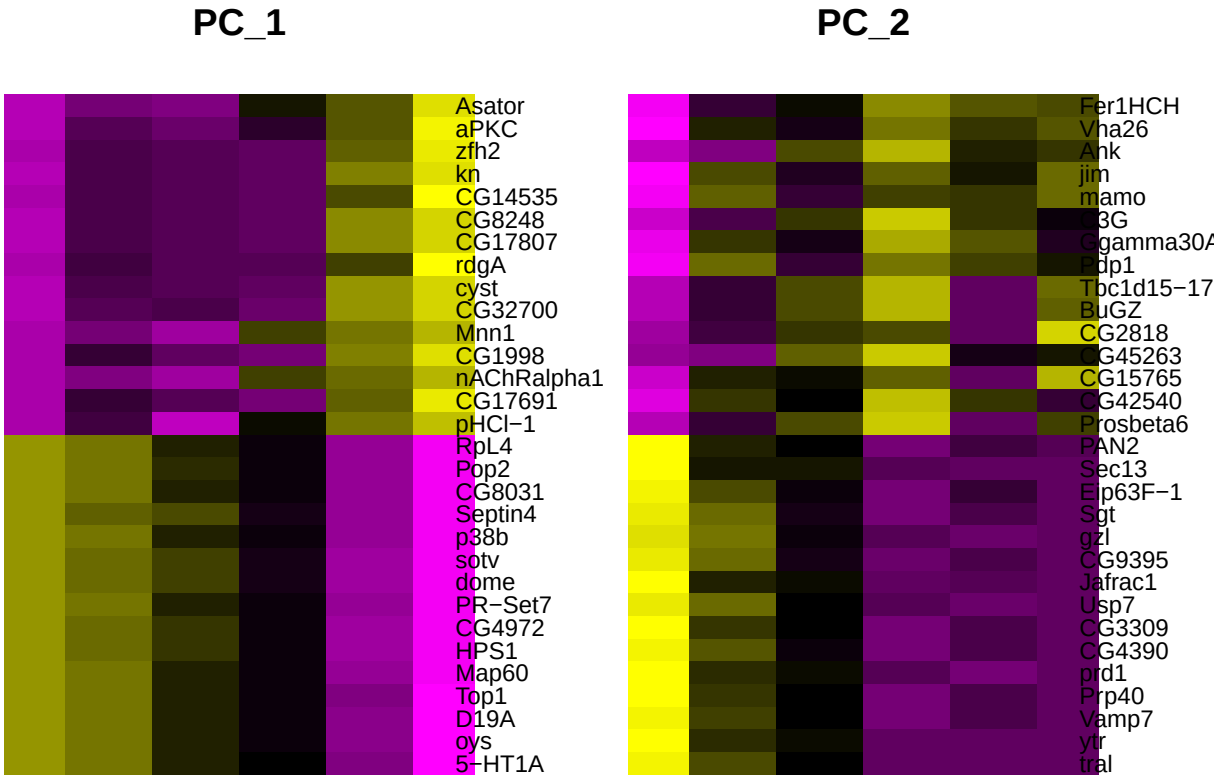
Elbowplot plotting the explained variation as a function of each PCA, which will tell as how many PCAs are important.

```
ElbowPlot(seurat, ndims = ncol(Embeddings(seurat, "pca")))
```



check which genes are mostly contributing to each of the PC1

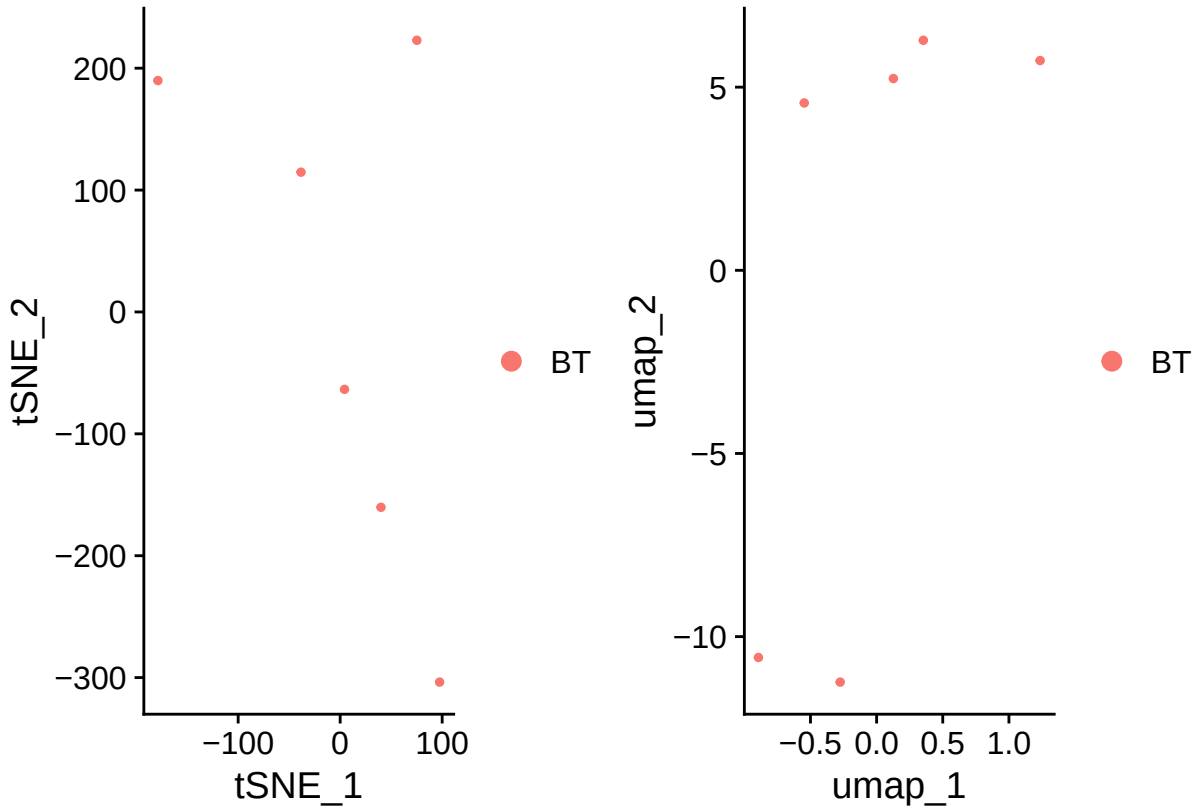
```
PCHeatmap(seurat, dims = 1:2, cells = 6, balanced = TRUE, ncol = 2)
```



Non-linear dimension reduction for visualization

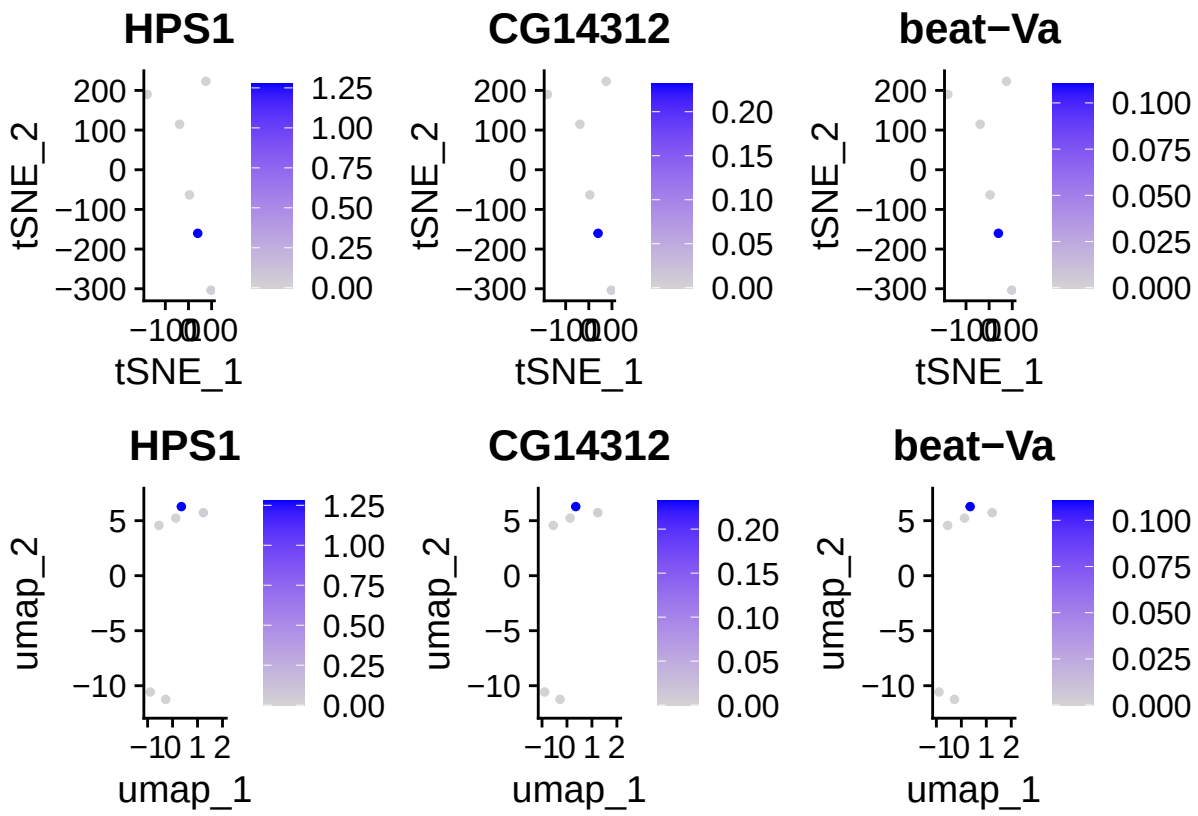
t-distributed Stochastic Neighbor Embedding (t-SNE) and Uniform Manifold Approximation and Projection (UMAP).The top PCs in the PCA analysis are used as the input to create a tSNE and UMAP embedding of the data (dims = 1: number of top pca)

```
seurat <- RunTSNE(seurat, dims = 1:3, resolution = 0.5, perplexity = 1)

seurat <- RunUMAP(seurat, dims = 1:3,  n.neighbors = 3)

plot1 <- TSNEPlot(seurat)
plot2 <- UMAPPlot(seurat)
plot1 + plot2
```



Check whether certain cell types or cell states exist in the data. feature plots of some known canonical markers of the cell types of interest (genes)

```
plot1 <- FeaturePlot(seurat, c("HPS1", "CG14312", "beat-Va"),
                     ncol=3, reduction = "tsne")

plot2 <- FeaturePlot(seurat, c("HPS1", "CG14312", "beat-Va"),
                     ncol=3, reduction = "umap")
plot1 / plot2
```



Cluster the cells

To understand the underlying heterogeneity in the data, it is necessary to identify cell groups. Following steps are involved in the cluster identification:

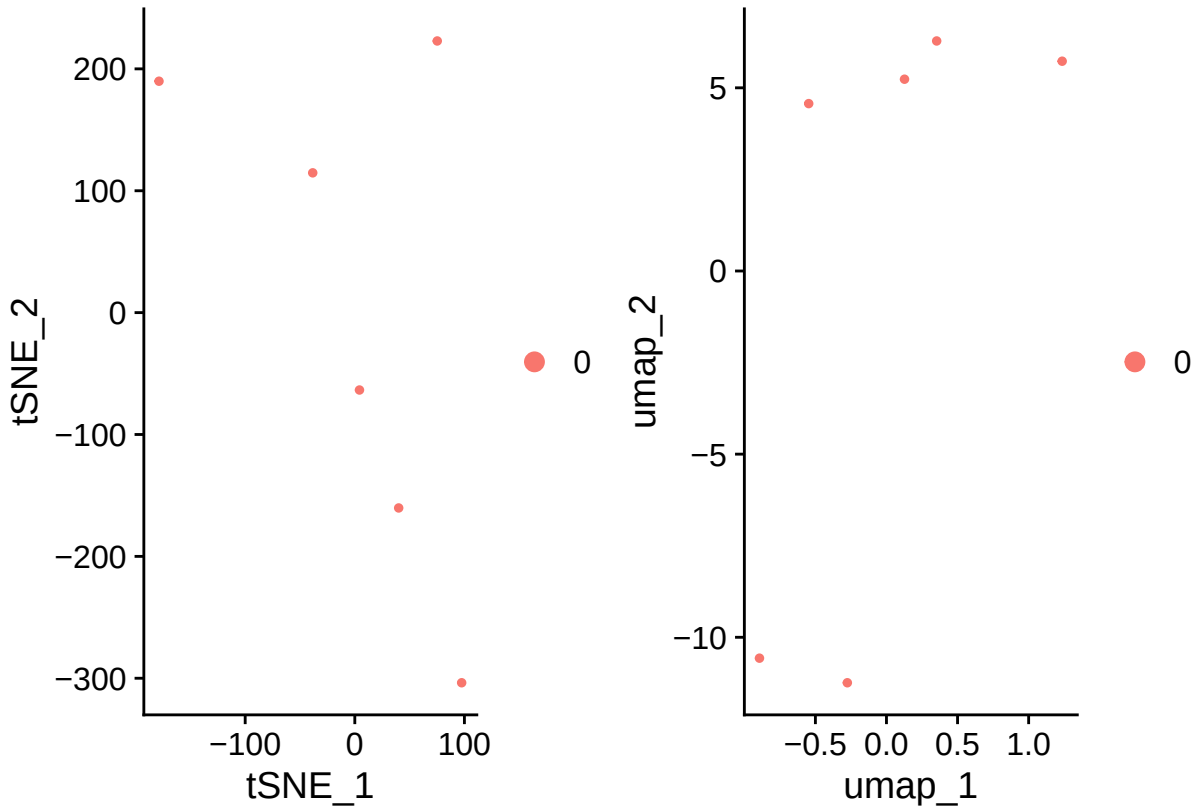
1. A k-nearest neighbor network of cells is generated
2. Every cells is firstly connected to cells with the shortest distances, based on their corresponding PC values
3. Only cell pairs which are neighbors of each other are considered as connected

```
seurat <- FindNeighbors(seurat, dims = 1:3)
seurat <- FindClusters(seurat, resolution = 1)
```

```
## Modularity Optimizer version 1.3.0 by Ludo Waltman and Nees Jan van Eck
##
## Number of nodes: 6
## Number of edges: 15
##
## Running Louvain algorithm...
## Maximum modularity in 10 random starts: 0.0000
## Number of communities: 1
## Elapsed time: 0 seconds
```

Next is to visualize the clustering result using the tSNE and UMAP

```
plot1 <- DimPlot(seurat, reduction = "tsne", label = F)
plot2 <- DimPlot(seurat, reduction = "umap", label = F)
plot1 + plot2
```



There is only one cluster which is labeled as 0.Cells with the same label are similar to each other and therefore can be seen to be of the same cell type or cell state.

In this experiment number of cells are small. So we apply clustering-independent method for finding differentially expressed genes as descibed in [https://doi.org/10.1038/s41467-020-17900-3] It uses the distribution of cells in an input space to predict DEGs as follows:

1. it infers a reference distribution of cells in the space (distribution Q).
2. SingleCellHaystack estimates the distribution of cells in which a gene G is detected.
3. The divergence of gene G, DKL(G), is calculated.
4. The significance of DKL(G) is evaluated

```
# Extract the gene expression matrix
gene_expression_matrix <- GetAssayData(seurat, slot = "data")

# Convert to a data frame
gene_expression_df <- as.data.frame(as.matrix(gene_expression_matrix))
```

```
# PCA coordinates are stored in the @reductions$pca@cell.embeddings slot
pca_coordinates <- Embeddings(seurat, reduction = "pca")

# Convert to a data frame if needed
pca_df <- as.data.frame(pca_coordinates)

### Extract t-SNE Coordinates ###
# t-SNE coordinates are stored in the @reductions$tsne@cell.embeddings slot
tsne_coordinates <- Embeddings(seurat, reduction = "tsne")

# Convert to a data frame
tsne_df <- as.data.frame(tsne_coordinates)

library(singleCellHaystack)
res.pca3 <- haystack(x = pca_df, expression = gene_expression_matrix,grid.points=3)
show_result_haystack(res.haystack = res.pca3, n = 10)
```

```
##               D_KL log.p.vals log.p.adj
## CG1161    2.242630  -2.021750         0
## mRpL3     2.242438  -2.021709         0
## Pex23     2.242924  -2.021517         0
## Hs2st     2.242010  -2.021070         0
## CG14232   2.241897  -2.020774         0
## GlyS      2.241788  -2.020435         0
## 412{}1368 2.244905  -2.008603         0
## CG32066   2.239450  -1.996961         0
## CG1371    2.247264  -1.950430         0
## CG11858   2.237652  -1.945201         0
```

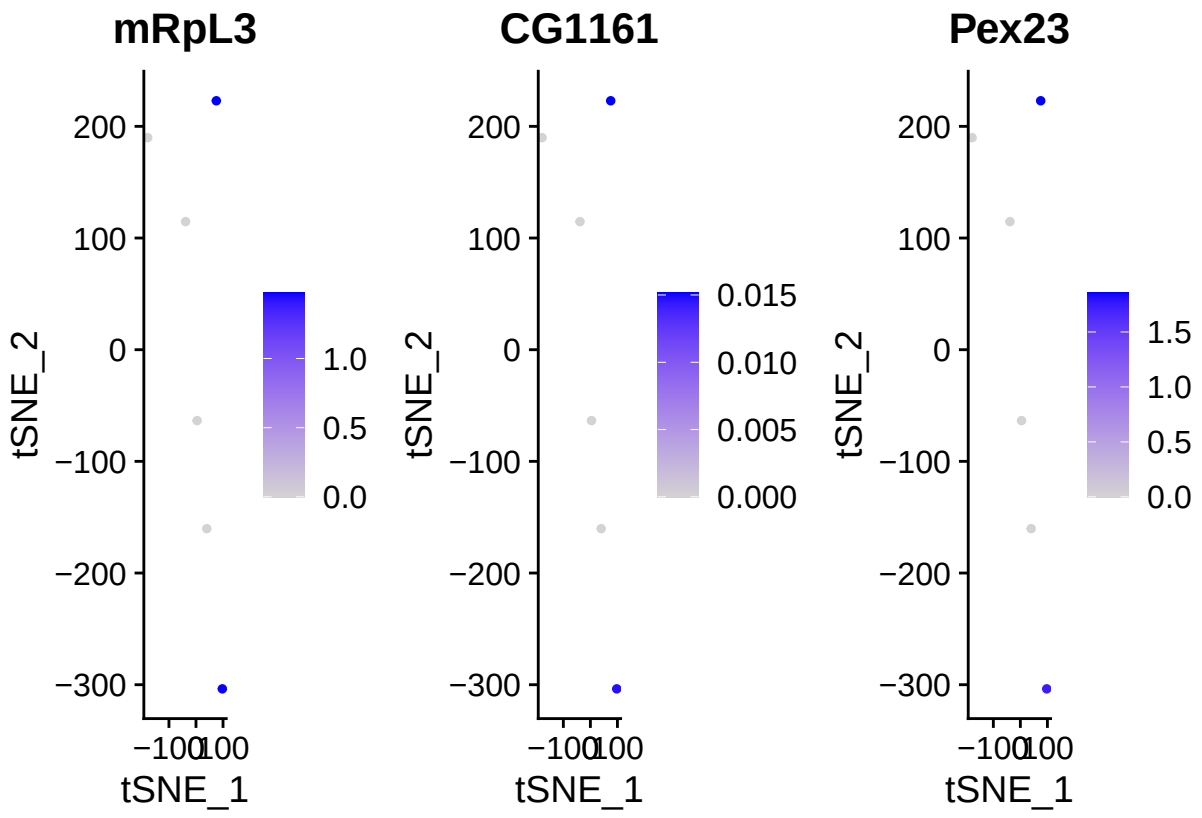
Calculate p vales and BH. Q values

```
DEG<-res.pca3$results

DEG$Pval<-10^(DEG$log.p.vals)
DEG$Qval<-p.adjust(DEG$Pval, method = "BH")

DEG<- DEG[order(DEG$Pval),]
head(DEG)
```

```
##             D_KL log.p.vals log.p.adj        Pval      Qval
## CG1161  2.242630  -2.021750         0 0.009511527 0.7784153
## mRpL3   2.242438  -2.021709         0 0.009512425 0.7784153
## Pex23   2.242924  -2.021517         0 0.009516624 0.7784153
## Hs2st   2.242010  -2.021070         0 0.009526417 0.7784153
## CG14232 2.241897  -2.020774         0 0.009532926 0.7784153
## GlyS    2.241788  -2.020435         0 0.009540375 0.7784153
```

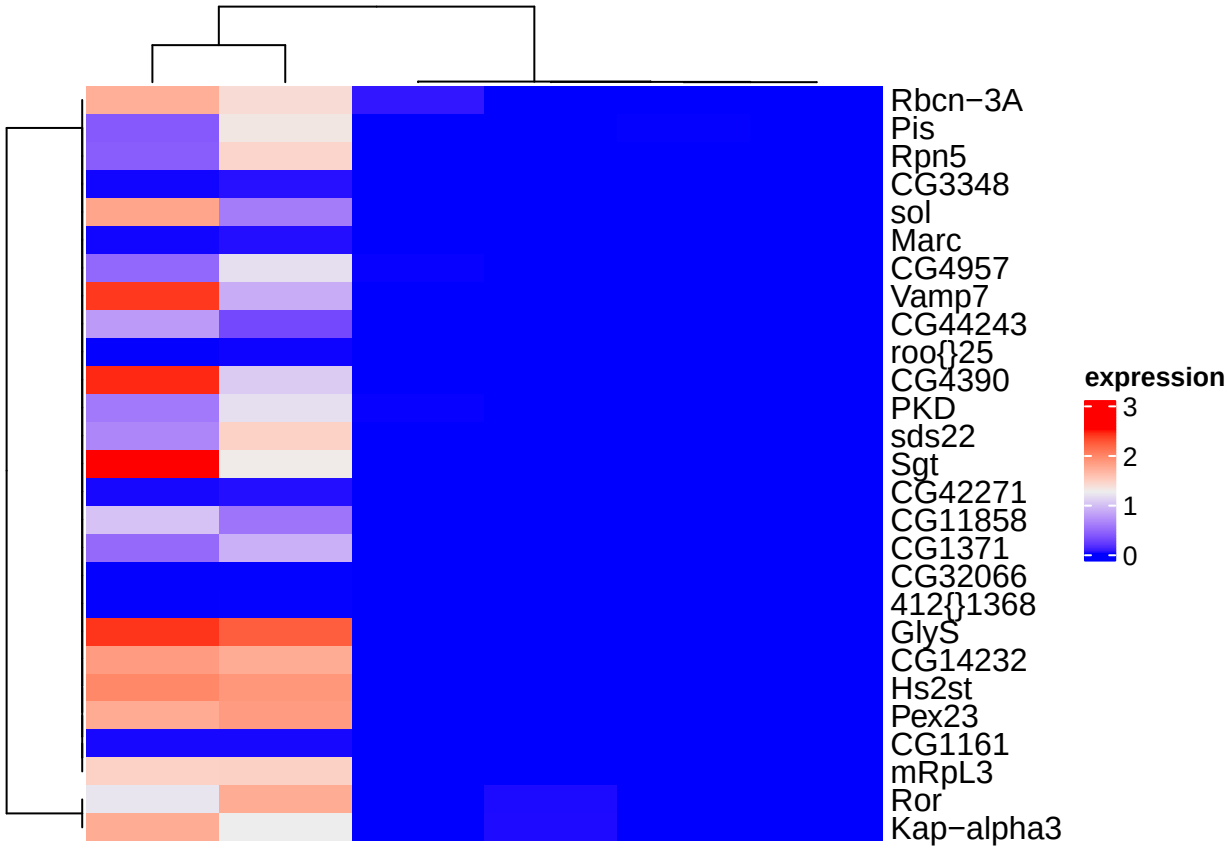
The most significant DEGs are mRpL3, CG1161, Pex23. Here we visualize their expression in the t-SNE plot.

```
plot1 <- FeaturePlot(seurat, c("mRpL3", "CG1161", "Pex23"),
                     ncol=3, reduction = "tsne")

plot1
```



Get the top most significant genes, and cluster them by their distribution pattern in the 2D plot

```
library(ComplexHeatmap)
sorted.table <- show_result_haystack(res.haystack = res.pca3, p.value.threshold = 0.02)
gene.subset <- row.names(sorted.table)
 hm <- hclust_haystack(pca_df, gene_expression_matrix[gene.subset, ], grid.coordinates=res.pca3$info$gr
 ComplexHeatmap::Heatmap(gene_expression_df[gene.subset, ], show_column_names=FALSE, cluster_rows=hm, na
```



Write the DEG to a csv file

```
write.csv(DEG[,c(1,4:5)], 'Single_Cell_DEG-BT.csv')
```