16th Annual Midsouth Computational Biology & Bioinformatics Society Conference

MCBIOS 2019

Tutorial #1: Single-cell data analysis

March 28, 2019 1:00PM - 4:00PM

Instructors

Min Gao, PhD, Informatics Institute, UAB, USA

Shanrun Liu, PhD, CFCC single cell core, UAB, USA

Jake Chen, PhD, Informatics Institute, UAB, USA

Christopher Fucile, MS, Informatics Institute, UAB, USA

MCBIOS.ORG

ISCB INTERNATIONAL SOCIETY FOR COMPUTATIONAL BIOLOGY AFFILIATED GROUP

UAB THE UNIVERSITY OF ALABAMA AT BIRMINGHAM

Knowledge that will change your world

Single-cell data analysis tutorial

Schedule

1:00 PM – 1:10 PM Introduction and overview
Min Gao, Ph.D., Bioinformatics Scientist, Informatics Institute

1:10 PM – 1:40 PM Introduction of single-cell analysis
Shanrun Liu, Ph.D., CFCC single cell core manager,
Department of Biochemistry and Molecular Genetics

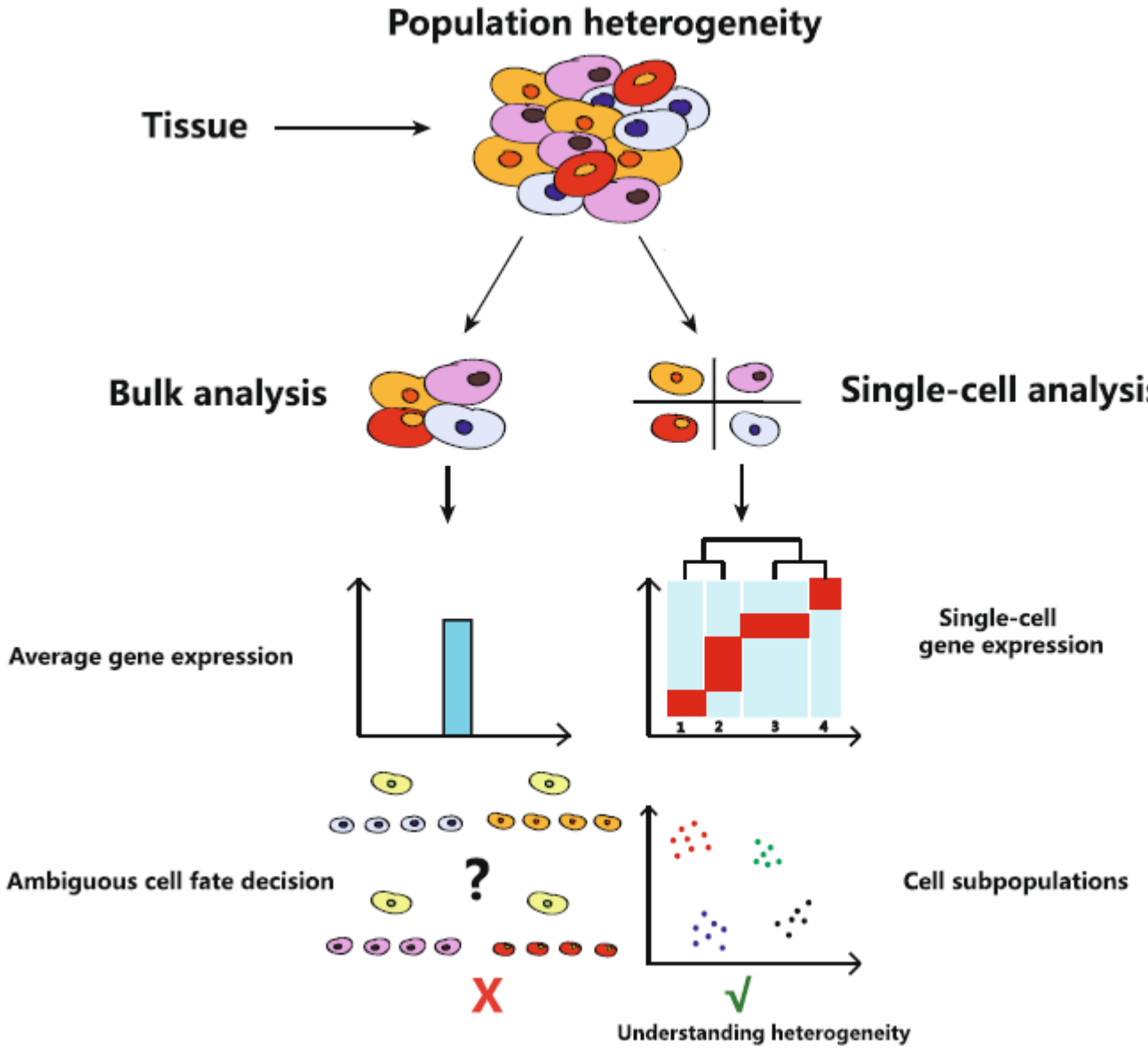
1:40 PM – 2:20 PM Computational techniques for single-cell data analysis
Jake Chen, Ph.D., Associate Director, Informatics Institute
Professor of Genetics and Computer Science

2:20 PM – 2:30 PM Break

2:30 PM – 3:00 PM Single-cell RNA sequencing in immunology
Christopher Fucile, MS, Scientist, Informatics Institute

3:00 PM – 4:00 PM Single-cell RNAseq data analysis – Case Study (hands-on)
Min Gao, Ph.D., Bioinformatics Scientist, Informatics Institute

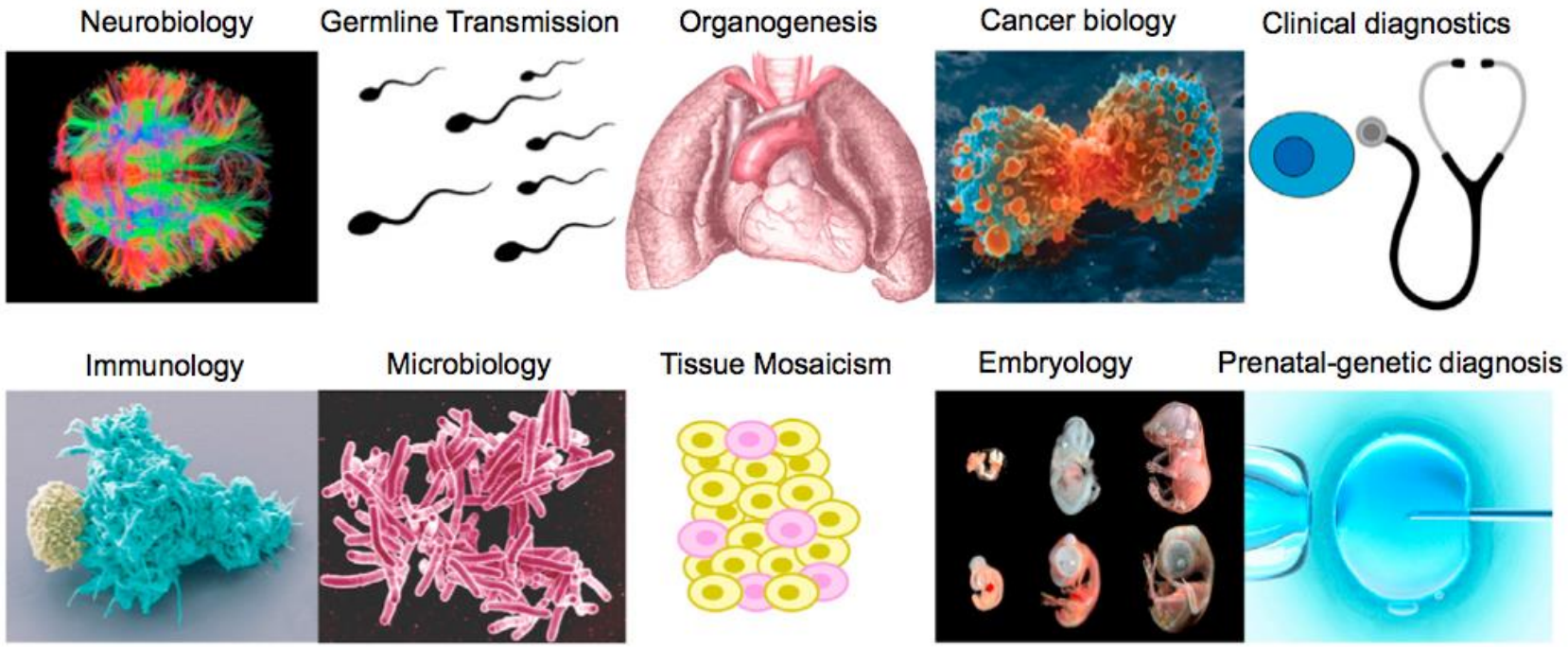
Why single-cell analysis?



Single-cell analysis reveals heterogeneity

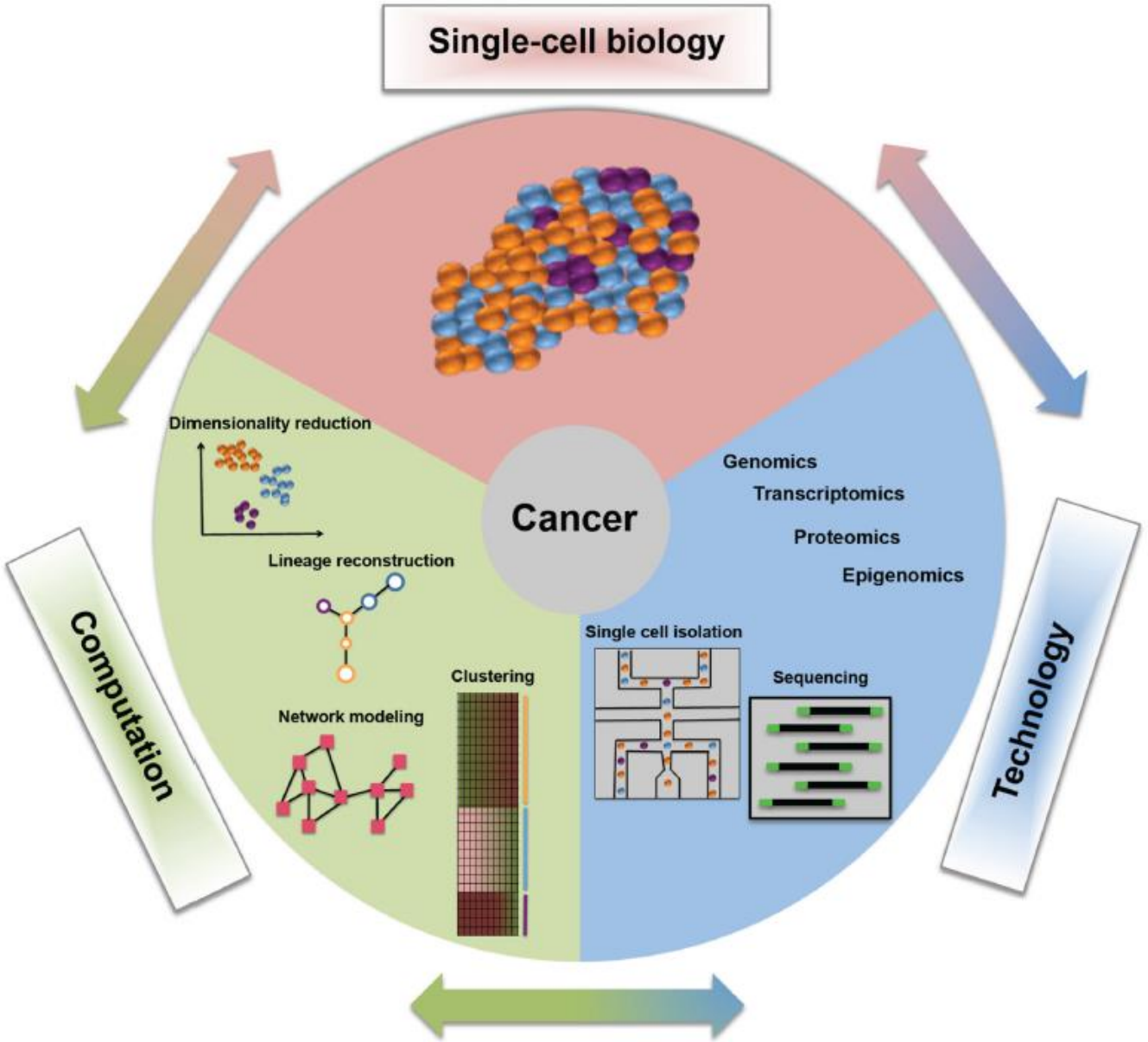
J Hematol Oncol. 2017 Jan 21;10(1):27

Single cell analysis affects diverse areas of biological research



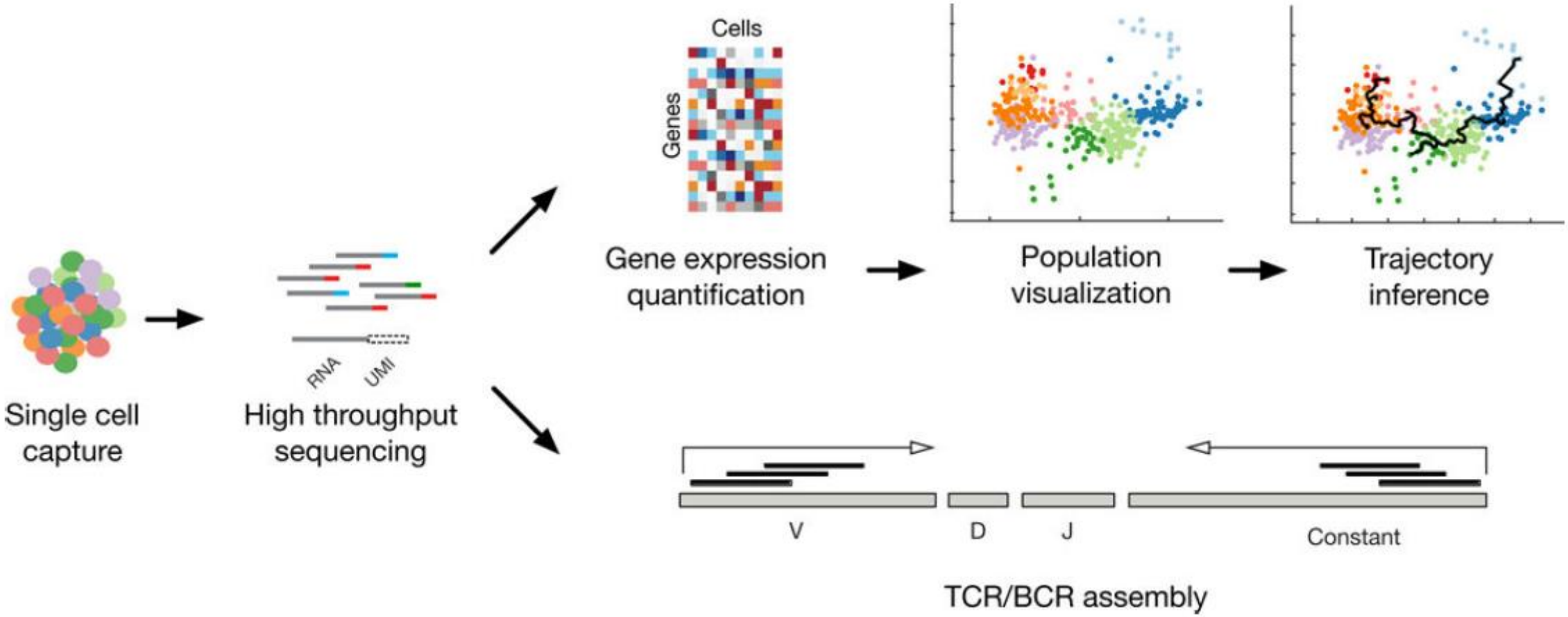
Wang Y.; Navin N. E. Mol. Cell 2015

Single cell analysis in cancer genomics



Trends Genet. 2015 Oct; 31(10): 576–586.

Single cell analysis in immunology



Trends Immunol. 2017 February ; 38(2): 140–149

Challenges in processing single-cell data

Variety in and of data is a classic biological problem pertaining also to big data. While there are clear opportunities in bigger volumes of data, there are technical, statistical and interpretative challenges rising alongside.

- **Basic programming needed to interpret data**
- **The information contained in single-cell data needs to be transformed into relevant biological knowledge**

Single-cell data analysis tutorial

Schedule

1:00 PM – 1:10 PM Introduction and overview
Min Gao, Ph.D., Bioinformatics Scientist, Informatics Institute

1:10 PM – 1:40 PM Introduction of single-cell analysis
Shanrun Liu, Ph.D., CFCC single cell core manager,
Department of Biochemistry and Molecular Genetics

1:40 PM – 2:20 PM Computational techniques for single-cell data analysis
Jake Chen, Ph.D., Associate Director, Informatics Institute
Professor of Genetics and Computer Science

2:20 PM – 2:30 PM Break

2:30 PM – 3:00 PM Single-cell RNA sequencing in immunology
Christopher Fucile, MS, Scientist, Informatics Institute

3:00 PM – 4:00 PM Single-cell RNAseq data analysis – Case Study (hands-on)
Min Gao, Ph.D., Bioinformatics Scientist, Informatics Institute

16th Annual Midsouth Computational Biology & Bioinformatics Society Conference

MCBIOS 2019

Single-cell RNAseq data analysis – Case Study

Min Gao, PhD

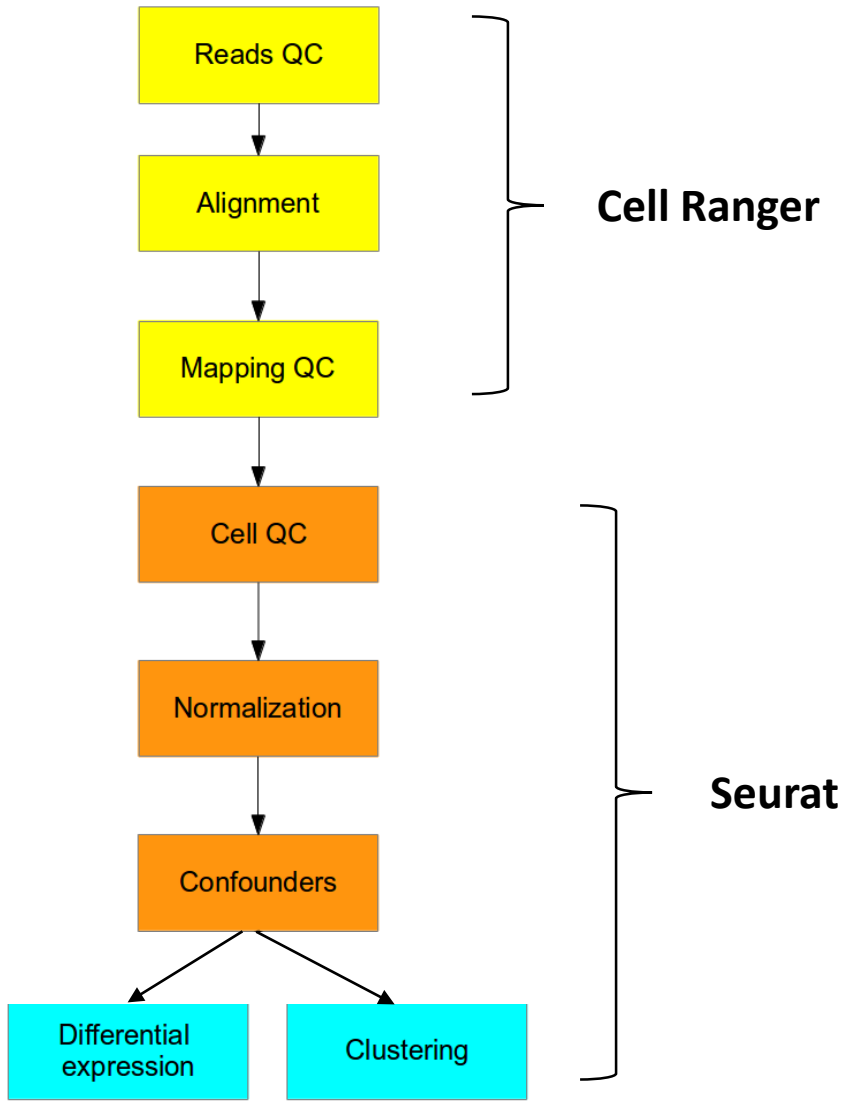
Informatics Institute

UAB, USA

Outline

- Single-cell RNA-seq analysis work flow
- Dataset and tools
- Functions in Seurat
- Hands-on single-cell RNAseq data analysis using Seurat
- Steps for single-cell RNAseq data analysis

Flowchart of the single-cell RNA-seq analysis



Dataset and tools

1. Dataset: 3k PBMCs(Peripheral Blood Mononuclear Cells) from a Healthy Donor

Raw reads was downloaded from https://support.10xgenomics.com/single-cell-gene-expression/datasets

2. tools:

Cell Ranger Pipeline (Runs on Linux, easy to download and run)
https://support.10xgenomics.com/single-cell-gene-expression/software/pipelines/latest/what-is-cell-ranger

Cell Ranger is a set of analysis pipelines that process Chromium single-cell RNA-seq output to align reads, generate feature-barcode matrices and perform clustering and gene expression analysis.

The single-cell gene expression matrix was generated from this step. We will use this matrix for further analysis

Seurat (Runs on R) https://satijalab.org/seurat/

Seurat is an R package designed for QC, analysis, and exploration of single cell RNA-seq data. Seurat aims to enable users to identify and interpret sources of heterogeneity from single cell transcriptomic measurements, and to integrate diverse types of single cell data.

Functions in Seurat – load data to Seurat

CreateSeuratObject

Initialize And Setup The Seurat Object

Initializes the Seurat object and some optional filtering

Usage

```
CreateSeuratObject(raw.data, project = "SeuratProject", min.cells = 0,
  min.genes = 0, is.expr = 0, normalization.method = NULL,
  scale.factor = 10000, do.scale = FALSE, do.center = FALSE,
  names.field = 1, names.delim = "_", meta.data = NULL,
  display.progress = TRUE, ...)
```

Functions in Seurat – Cell QC and filter cells

AddMetaData

Add Metadata

Adds additional data for single cells to the Seurat object. Can be any piece of information associated with a cell (examples include read depth, alignment rate, experimental batch, or subpopulation identity). The advantage of adding it to the Seurat object is so that it can be analyzed/visualized using FetchData, VlnPlot, GenePlot, SubsetData, etc.

Usage

```
AddMetaData(object, metadata, col.name = NULL)
```

FilterCells

Return A Subset Of The Seurat Object

Creates a Seurat object containing only a subset of the cells in the original object. Takes either a list of cells to use as a subset, or a parameter (for example, a gene), to subset on.

Usage

```
FilterCells(object, subset.names, low.thresholds, high.thresholds,
  cells.use = NULL)
```

Functions in Seurat – Data Normalization

NormalizeData

Normalize Assay Data

Normalize data for a given assay

Usage

```
NormalizeData(object, assay.type = "RNA",
  normalization.method = "LogNormalize", scale.factor = 10000,
  display.progress = TRUE)
```

Functions in Seurat – Detection of highly variable genes

FindVariableGenes

Identify Variable Genes

Identifies genes that are outliers on a 'mean variability plot'. First, uses a function to calculate average expression (mean.function) and dispersion (dispersion.function) for each gene. Next, divides genes into num.bin (deafult 20) bins based on their average expression, and calculates z-scores for dispersion within each bin. The purpose of this is to identify variable genes while controlling for the strong relationship between variability and average expression.

Usage

```
FindVariableGenes(object, mean.function = ExpMean,
  dispersion.function = LogVMR, do.plot = TRUE, set.var.genes = TRUE,
  x.low.cutoff = 0.1, x.high.cutoff = 8, y.cutoff = 1,
  y.high.cutoff = Inf, num.bin = 20, binning.method = "equal_width",
  selection.method = "mean.var.plot", top.genes = 1000, do.recalc = TRUE,
  sort.results = TRUE, do.cpp = TRUE, display.progress = TRUE, ...)
```

Functions in Seurat – Scaling the data and removing uninteresting sources of variation

ScaleData

Scale And Center The Data.

Scales and centers genes in the dataset. If variables are provided in vars.to.regress, they are individually regressed against each gene, and the resulting residuals are then scaled and centered.

Usage

```
ScaleData(object, genes.use = NULL, data.use = NULL, vars.to.regress,
  model.use = "linear", use.umi = FALSE, do.scale = TRUE,
  do.center = TRUE, scale.max = 10, block.size = 1000,
  min.cells.to.block = 3000, display.progress = TRUE, assay.type = "RNA",
  do.cpp = TRUE, check.for.norm = TRUE, do.par = FALSE, num.cores = 1)
```

Functions in Seurat – **Perform linear dimensional reduction(PCA)**

RunPCA

Run Principal Component Analysis On Gene Expression Using IRLBA

Run a PCA dimensionality reduction. For details about stored PCA calculation parameters, see `PrintPCAParams` .

Usage

```
RunPCA(object, pc.genes = NULL, pcs.compute = 20, use.imputed = FALSE,
  rev.pca = FALSE, weight.by.var = TRUE, do.print = TRUE,
  pcs.print = 1:5, genes.print = 30, reduction.name = "pca",
  reduction.key = "PC", assay.type = "RNA", seed.use = 42, ...)
```

Functions in Seurat – Cell clustering

FindClusters

Cluster Determination

Identify clusters of cells by a shared nearest neighbor (SNN) modularity optimization based clustering algorithm. First calculate k-nearest neighbors and construct the SNN graph. Then optimize the modularity function to determine clusters. For a full description of the algorithms, see Waltman and van Eck (2013) *The European Physical Journal B*. Thanks to Nigel Delaney (evolvedmicrobe@github) for the rewrite of the Java modularity optimizer code in Rcpp!

Usage

```
FindClusters(object, genes.use = NULL, reduction.type = "pca",
  dims.use = NULL, k.param = 30, plot.SNN = FALSE, prune.SNN = 1/15,
  print.output = TRUE, distance.matrix = NULL, save.SNN = FALSE,
  reuse.SNN = FALSE, force.recalc = FALSE, nn.eps = 0,
  modularity.fxn = 1, resolution = 0.8, algorithm = 1, n.start = 100,
  n.iter = 10, random.seed = 0, temp.file.location = NULL,
  edge.file.name = NULL)
```

Functions in Seurat – Run Non-linear dimensional reduction (tSNE)

RunTSNE

Run T-Distributed Stochastic Neighbor Embedding

Run t-SNE dimensionality reduction on selected features. Has the option of running in a reduced dimensional space (i.e. spectral tSNE, recommended), or running based on a set of genes. For details about stored TSNE calculation parameters, see `PrintTSNEParams` .

Usage

```
RunTSNE(object, reduction.use = "pca", cells.use = NULL, dims.use = 1:5,
  genes.use = NULL, seed.use = 1, tsne.method = "Rtsne", add.iter = 0,
  dim.embed = 2, distance.matrix = NULL, reduction.name = "tsne",
  reduction.key = "tSNE_", ...)
```

Functions in Seurat – Finding differentially expressed genes (cluster biomarkers)

FindMarkers

Gene Expression Markers Of Identity Classes

Finds markers (differentially expressed genes) for identity classes

Usage

```
FindMarkers(object, ident.1, ident.2 = NULL, genes.use = NULL,
  logfc.threshold = 0.25, test.use = "wilcox", min.pct = 0.1,
  min.diff.pct = -Inf, print.bar = TRUE, only.pos = FALSE,
  max.cells.per.ident = Inf, random.seed = 1, latent.vars = NULL,
  min.cells.gene = 3, min.cells.group = 3, pseudocount.use = 1,
  assay.type = "RNA", ...)
```

Other functions in Seurat

Name	Description
VlnPlot	Single cell violin plot
GenePlot	Scatter plot of single cell data
VizPCA	Visualize PCA genes
PCAPlot	Plot PCA map
PCHeatmap	Principal component heatmap
TSNEPlot	Plot tSNE map
FeaturePlot	Visualize 'features' on a dimensional reduction plot
DotPlot	Dot plot visualization
DoHeatmap	Gene expression heatmap

Hands-on single-cell RNAseq data analysis using Seurat

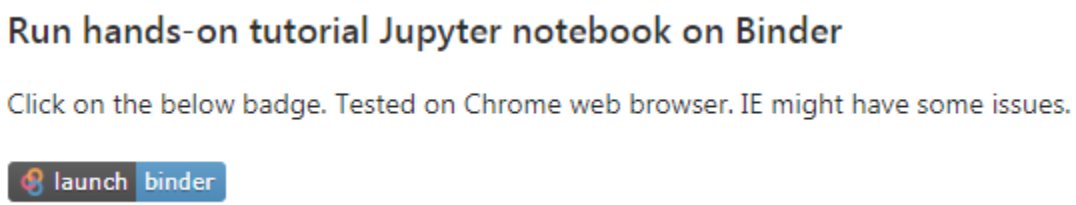
- **Prerequisites:**

  Install R version 3.3 or later (R).
  Install.packages('Seurat')

- **Single cell tutorial GitLab:**

  Please search “MCBIOS19” on Chrome web browser, go to “program”-- “Tutorial/Workshop”— “Tutorial 1”
  Single-cell RNAseq data analysis – Case Study (hands-on)

- **U-BRITE Binder link :(**Tested on Chrome web browser. IE might have some issues)



- **UAB OnDemand :** (It provides an integrated, single access point for all of your HPC resources. Login in using your Cheaha account.) https://rc.uab.edu/

Load Seurat package

jupyter pbmc_small_case2 (unsaved changes)

File Edit View Insert Cell Kernel Widgets Help Not Trusted

Run Code

In [1]:
```
library(Seurat)
library(dplyr)
library(Matrix)
```

```
Loading required package: ggplot2
Loading required package: cowplot

Attaching package: 'cowplot'

The following object is masked from 'package:ggplot2':

    ggsave

Loading required package: Matrix

Attaching package: 'dplyr'

The following objects are masked from 'package:stats':

    filter, lag

The following objects are masked from 'package:base':

    intersect, setdiff, setequal, union
```

Load PBMC data to Seurat

In [2]:
```
# Load the PBMC dataset
pbmc.data <- Read10X("./dataset/2.7kPBMC_filtered_gene_bc_matrices/hg19")
#Examine the memory savings between regular and sparse matrices
dense.size <- object.size(as.matrix(pbmc.data))
dense.size
```

```
709548272 bytes
```

In [3]:
```
# Initialize the Seurat object with the raw (non-normalized data).  Keep all
# genes expressed in >= 3 cells (~0.1% of the data). Keep all cells with at
# least 200 detected genes
pbmc <- CreateSeuratObject(raw.data = pbmc.data, min.cells = 3, min.genes = 200,
    project = "10X_PBMC")
pbmc
```

```
An object of class seurat in project 10X_PBMC
 13714 genes across 2700 samples.
```

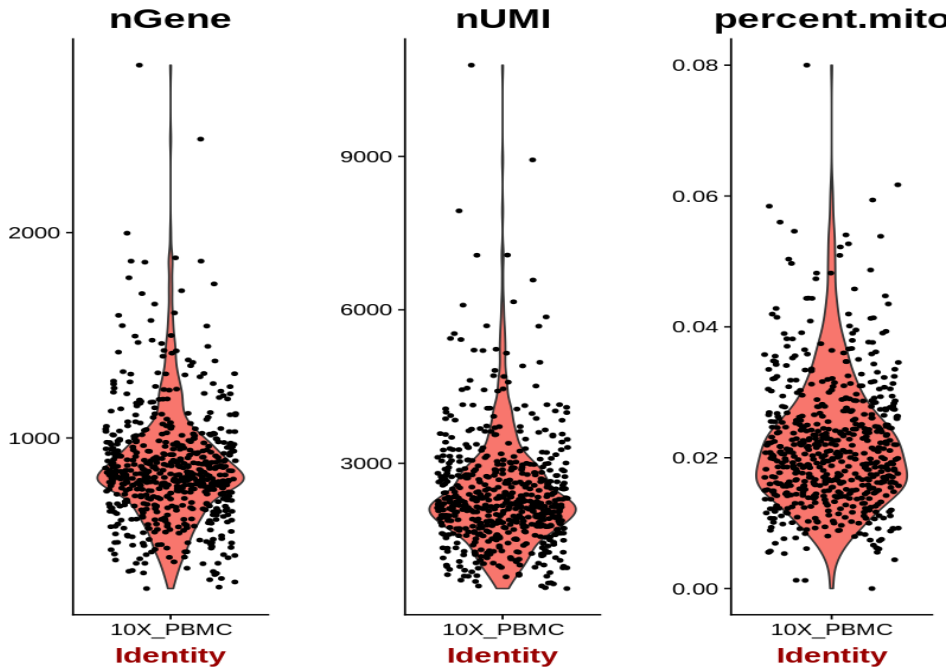
In [4]:
```
pbmc_small <- SubsetData(object = pbmc, cells.use = pbmc@cell.names[1:600])
```

In [5]:
```
pbmc_small
```

```
An object of class seurat in project 10X_PBMC
 13714 genes across 600 samples.
```

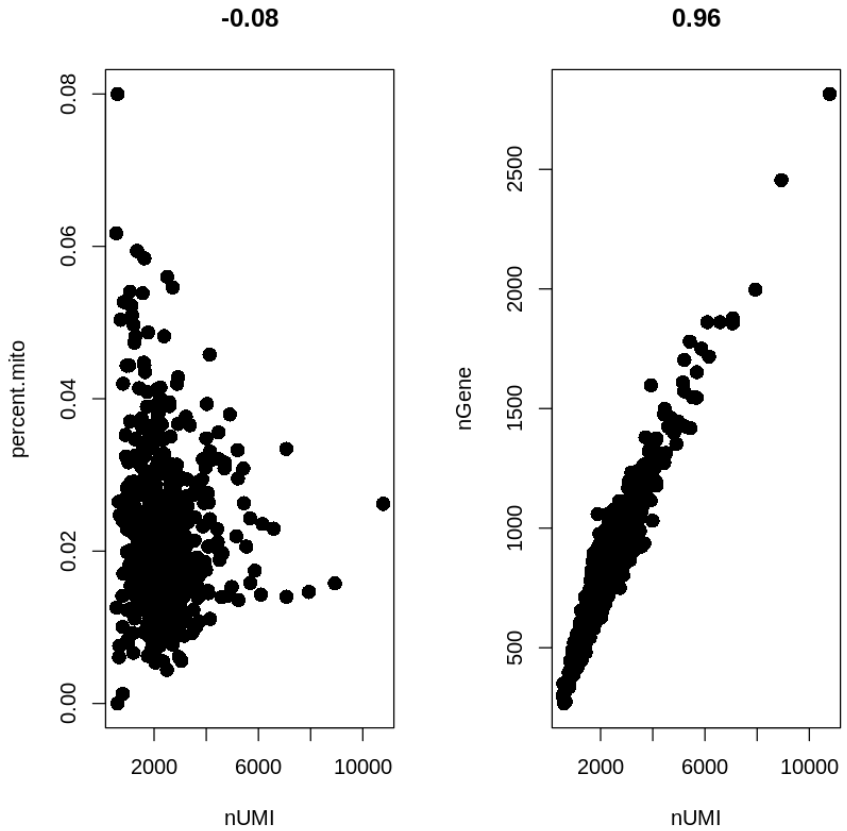
Cell QC

In [6]:
```
# The number of genes and UMIs (nGene and nUMI) are automatically calculated
# for every object by Seurat.  For non-UMI data, nUMI represents the sum of
# the non-normalized values within a cell We calculate the percentage of
# mitochondrial genes here and store it in percent.mito using AddMetaData.
# We use object@raw.data since this represents non-transformed and
# non-log-normalized counts The % of UMI mapping to MT-genes is a common
# scRNA-seq QC metric.
mito.genes <- grep(pattern = "^MT-", x = rownames(x = pbmc_small@data), value = TRUE)
percent.mito <- Matrix::colSums(pbmc_small@raw.data[mito.genes, ])/Matrix::colSums(pbmc_small@raw.data)

# AddMetaData adds columns to object@meta.data, and is a great place to
# stash QC stats
pbmc_small <- AddMetaData(object = pbmc_small, metadata = percent.mito, col.name = "percent.mito")
VlnPlot(object = pbmc_small, features.plot = c("nGene", "nUMI", "percent.mito"), nCol = 3)
```



Cell QC

In [7]:
```
# GenePlot is typically used to visualize gene-gene relationships, but can
# be used for anything calculated by the object, i.e. columns in
# object@meta.data, PC scores etc.  Since there is a rare subset of cells
# with an outlier level of high mitochondrial percentage and also low UMI
# content, we filter these as well
par(mfrow = c(1, 2))
GenePlot(object = pbmc_small, gene1 = "nUMI", gene2 = "percent.mito")
GenePlot(object = pbmc_small, gene1 = "nUMI", gene2 = "nGene")
```



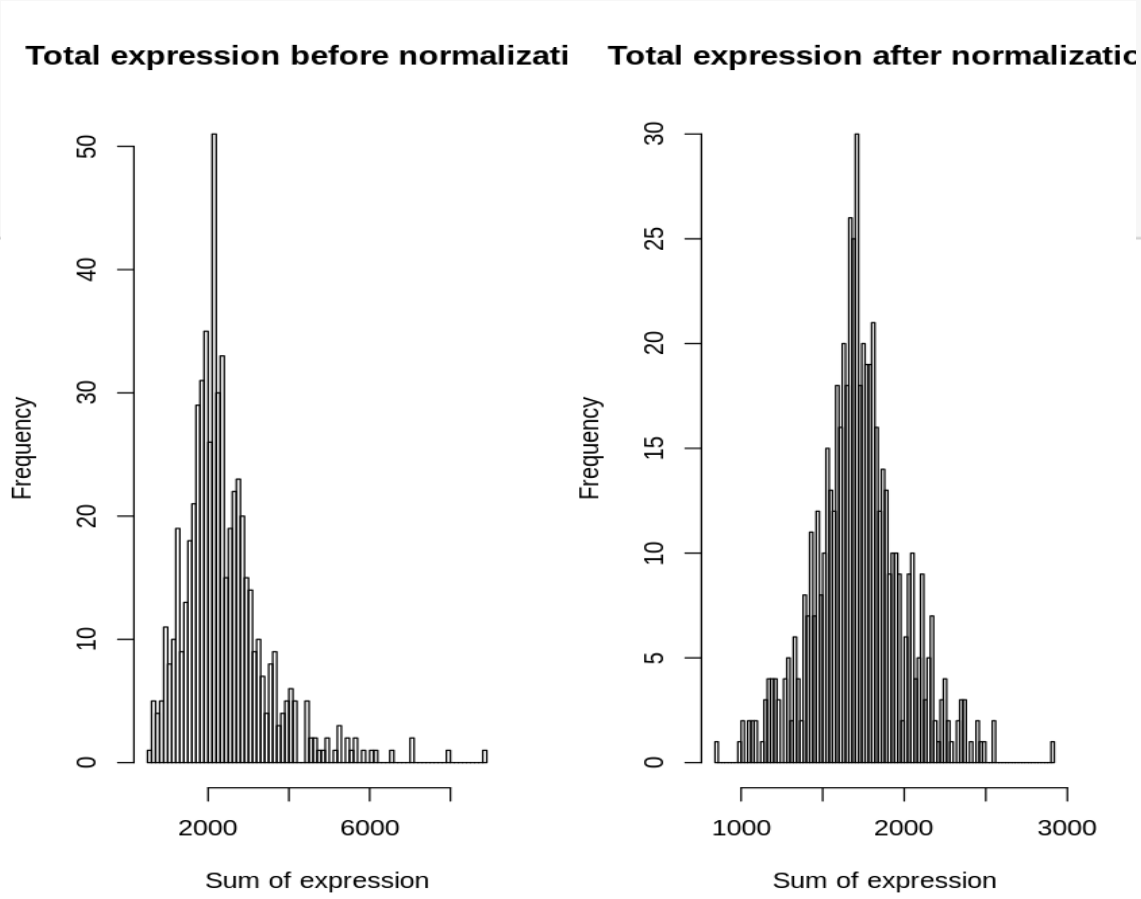
Cell QC – filter cells

In [8]:
```
# We filter out cells that have unique gene counts over 2,500 or less than
# 200 Note that low.thresholds and high.thresholds are used to define a
# 'gate'.  -Inf and Inf should be used if you don't want a lower or upper
# threshold.
pbmc_small <- FilterCells(object = pbmc_small, subset.names = c("nGene", "percent.mito"),
    low.thresholds = c(200, -Inf), high.thresholds = c(2500, 0.05))

pbmc_small
```

```
An object of class seurat in project 10X_PBMC
 13714 genes across 587 samples.
```

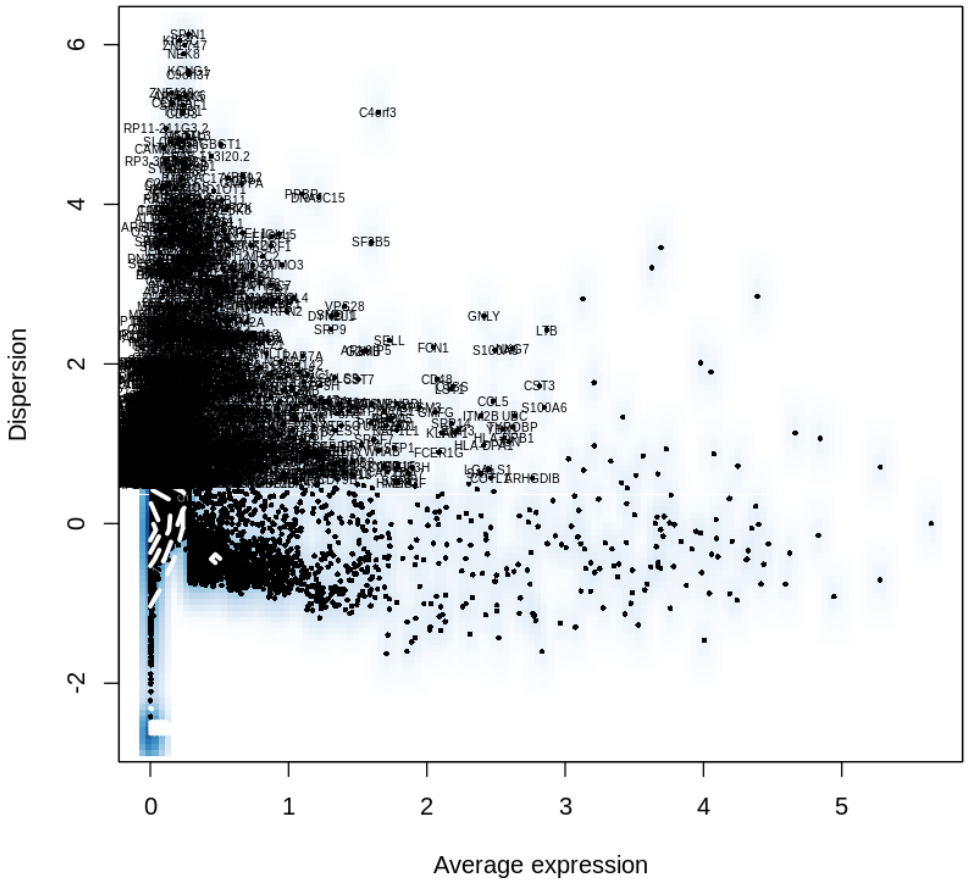
Data normalization

In [9]:
```
# Normalizing the data
par(mfrow = c(1, 2))

hist(colSums(pbmc_small@data),
     breaks = 100,
     main = "Total expression before normalization",
     xlab = "Sum of expression")

pbmc_small <- NormalizeData(object = pbmc_small, normalization.method = "LogNormalize",
    scale.factor = 10000)

hist(colSums(pbmc_small@data),
     breaks = 100,
     main = "Total expression after normalization",
     xlab = "Sum of expression")
```



Detection of highly variable genes

In [10]:
```
# Detection of variable genes across the single cells
pbmc_small <- FindVariableGenes(object = pbmc_small, mean.function = ExpMean, dispersion.function = LogVMR,
    x.low.cutoff = 0.0125, x.high.cutoff = 3, y.cutoff = 0.5)
```



In [11]:
```
length(x = pbmc_small@var.genes)
```

2596

Scaling the data and removing uninteresting sources of variation

In [12]:
```
# Scaling the data and removing unwanted sources of variation

pbmc_small <- ScaleData(object = pbmc_small, vars.to.regress = c("nUMI", "percent.mito"))
pbmc_small
```

```
Regressing out: nUMI, percent.mito


Time Elapsed:  11.2599799633026 secs

Scaling data matrix

An object of class seurat in project 10X_PBMC
 13714 genes across 587 samples.
```

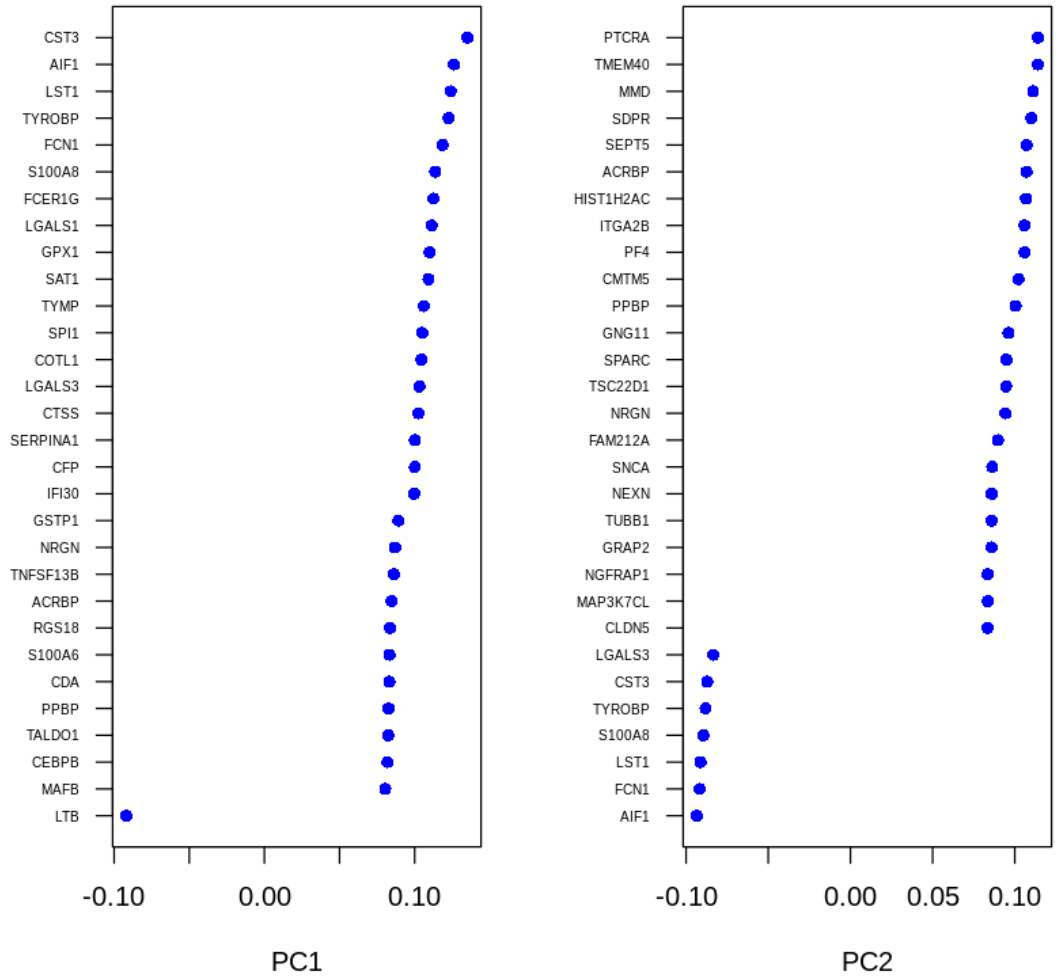
Perform linear dimensional reduction(PCA)

In [13]:
```
# Perform linear dimensional reduction
pbmc_small <- RunPCA(object = pbmc_small, pc.genes = pbmc_small@var.genes, do.print = TRUE, pcs.print = 1:5,
    genes.print = 5)
```

```
[1] "PC1"
[1] "LTB"    "ACAP1" "CD69"   "CD27"   "CD247"
[1] ""
[1] "CST3"   "AIF1"   "LST1"   "TYROBP" "FCN1"
[1] ""
[1] ""
[1] "PC2"
[1] "AIF1"   "FCN1"   "LST1"   "S100A8" "TYROBP"
[1] ""
[1] "PTCRA"  "TMEM40" "MMD"    "SDPR"   "SEPT5"
[1] ""
[1] ""
[1] "PC3"
[1] "NKG7"   "GZMB"   "PRF1"   "CST7"   "FGFBP2"
[1] ""
[1] "CD79A"     "MS4A1"     "HLA-DQA1"  "LINC00926" "CD79B"
[1] ""
[1] ""
[1] "PC4"
[1] "RGS10"   "CD27"    "MAL"     "NGFRAP1" "TRABD2A"
[1] ""
[1] "HLA-DQA1" "HLA-DPB1" "CD79A"    "CD79B"    "HLA-DPA1"
[1] ""
[1] ""
[1] "PC5"
[1] "AP001189.4" "GP9"        "CLU"        "C1orf198"   "SENCR"
```

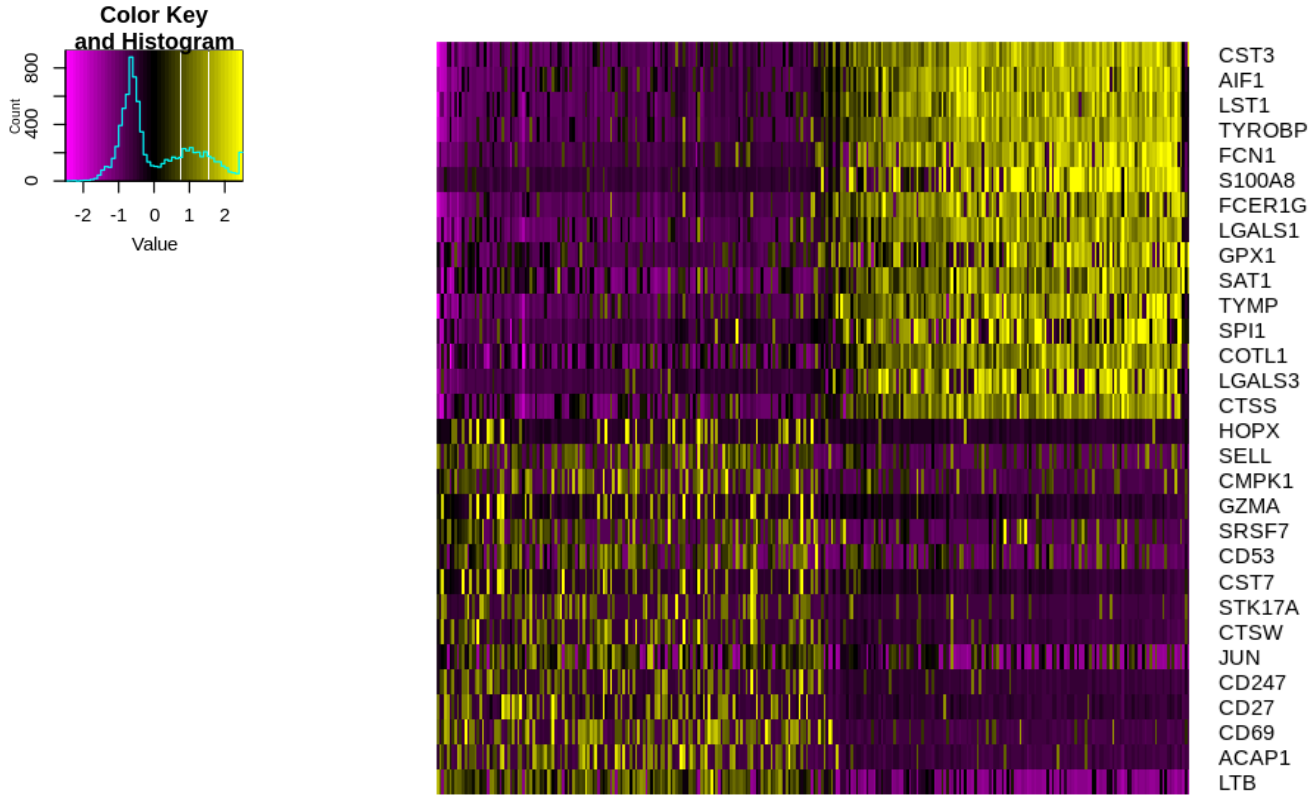
Visualizing both cells and genes that define the PCA

```
In [14]: VizPCA(object = pbmc_small, pcs.use = 1:2)
```



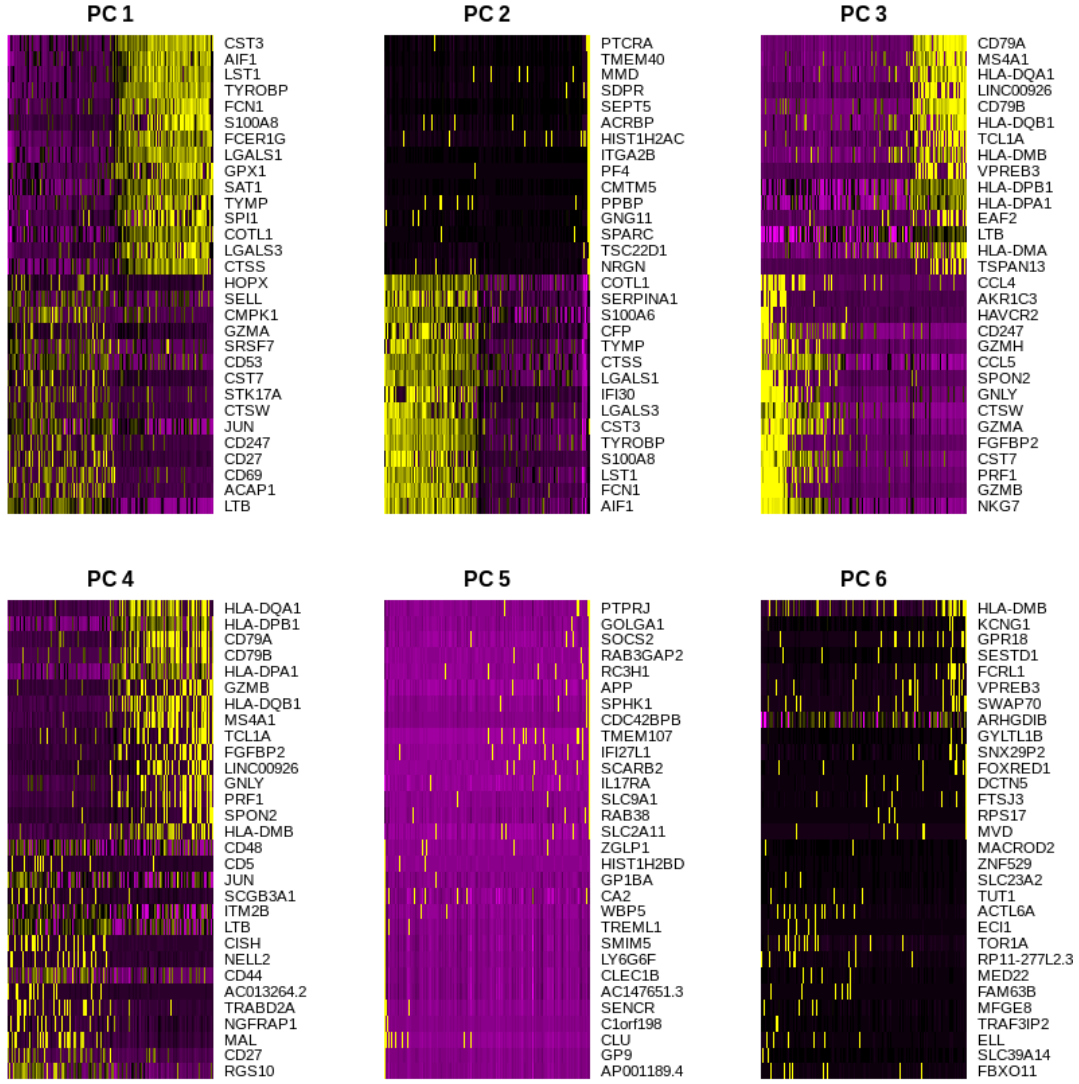
Visualizing both cells and genes that define the PCA

In [16]:
```
# ProjectPCA scores each gene in the dataset (including genes not included
# in the PCA) based on their correlation with the calculated components.
# Though we don't use this further here, it can be used to identify markers
# that are strongly correlated with cellular heterogeneity, but may not have
# passed through variable gene selection.  The results of the projected PCA
# can be explored by setting use.full=T in the functions above
pbmc_small <- ProjectPCA(object = pbmc_small, do.print = FALSE)
PCHeatmap(object = pbmc_small, pc.use = 1, cells.use = 300, do.balanced = TRUE, label.columns = FALSE)
```



Visualizing both cells and genes that define the PCA

```
In [17]: PCHeatmap(object = pbmc_small, pc.use = 1:6, cells.use = 300, do.balanced = TRUE,
             label.columns = FALSE, use.full = FALSE)
```



Cell clustering

In [18]:
```
# save.SNN = T saves the SNN so that the clustering algorithm can be rerun
# using the same graph but with a different resolution value (see docs for
# full details)
pbmc_small <- FindClusters(object = pbmc_small, reduction.type = "pca", dims.use = 1:10,
    resolution = 0.6, print.output = 0, save.SNN = TRUE)
```

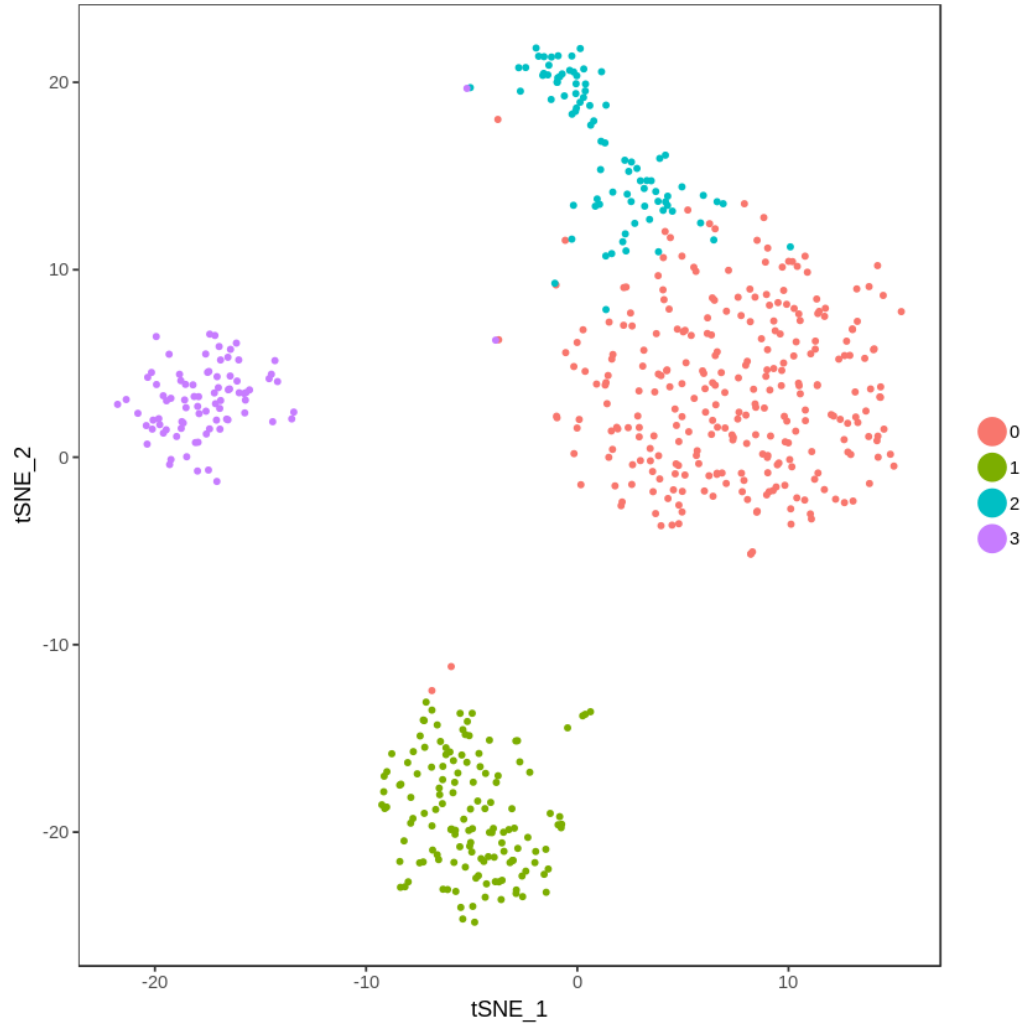
In [19]:
```
PrintFindClustersParams(object = pbmc_small)
```

```
Parameters used in latest FindClusters calculation run on: 2019-03-02 16:35:11
=============================================================================
Resolution: 0.6
-----------------------------------------------------------------------------
Modularity Function    Algorithm         n.start         n.iter
     1                   1                 100             10
-----------------------------------------------------------------------------
Reduction used          k.param          prune.SNN
     pca                 30                0.0667
-----------------------------------------------------------------------------
Dims used in calculation
=============================================================================
1 2 3 4 5 6 7 8 9 10
```

Run Non-linear dimensional reduction (tSNE)

In [20]:
```
#Run Non-linear dimensional reduction (tSNE)
pbmc_small <- RunTSNE(object = pbmc_small, dims.use = 1:10, do.fast = TRUE)
```

In [21]:
```
# note that you can set do.label=T to help label individual clusters
TSNEPlot(object = pbmc_small)
```



Finding differentially expressed genes (cluster biomarkers)

In [22]:
```
# Finding differentially expressed genes (cluster biomarkers)
# find all markers of cluster 1
cluster1.markers <- FindMarkers(object = pbmc_small, ident.1 = 1, min.pct = 0.25)
print(x = head(x = cluster1.markers, n = 5))
```

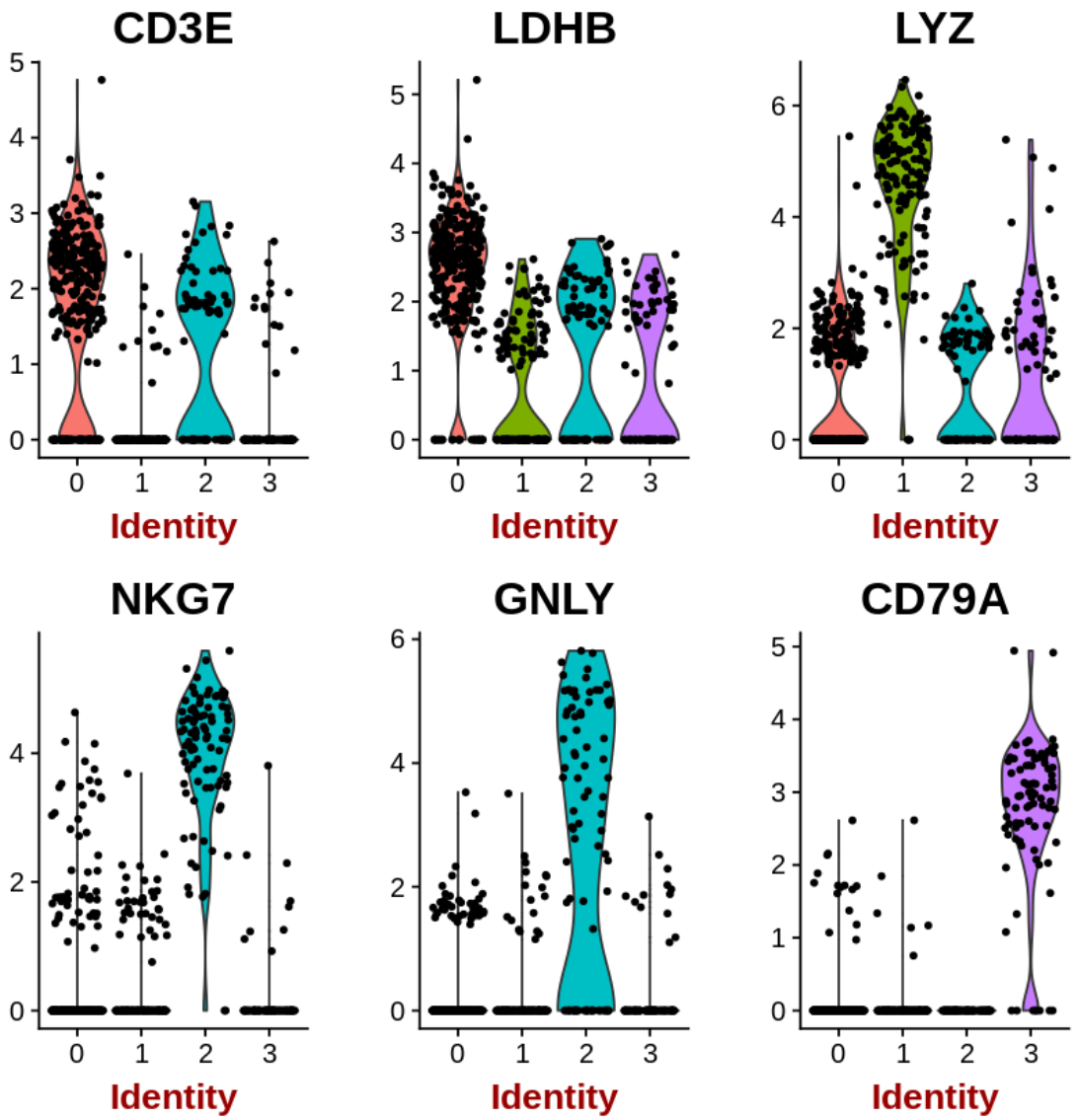
```
              p_val avg_logFC pct.1 pct.2    p_val_adj
CFD    1.036921e-88  2.214294 0.812 0.027 1.422033e-84
AIF1   1.840404e-88  2.791903 0.978 0.167 2.523930e-84
LST1   7.096445e-86  2.681395 0.978 0.180 9.732065e-82
TYROBP 8.110049e-85  2.663135 0.978 0.196 1.112212e-80
CST3   2.127291e-83  2.733195 0.993 0.214 2.917367e-79
```

In [23]:
```
# find markers for every cluster compared to all remaining cells, report
# only the positive ones
pbmc.markers <- FindAllMarkers(object = pbmc_small, only.pos = TRUE, min.pct = 0.25,
    thresh.use = 0.25)
pbmc.markers %>% group_by(cluster) %>% top_n(3, avg_logFC)
```

p_val	avg_logFC	pct.1	pct.2	p_val_adj	cluster	gene
1.192146e-54	1.305500	0.892	0.211	1.634910e-50	0	CD3D
2.276763e-54	1.293197	0.914	0.500	3.122352e-50	0	LDHB
8.099153e-39	1.235470	0.746	0.244	1.110718e-34	0	CD3E
1.053782e-82	3.729337	0.942	0.160	1.445157e-78	1	S100A9
8.915330e-75	3.470103	0.783	0.062	1.222648e-70	1	S100A8
2.481547e-67	3.207607	0.978	0.481	3.403193e-63	1	LYZ
4.860895e-66	3.147831	0.977	0.194	6.666231e-62	2	NKG7
3.071513e-44	2.917535	0.547	0.042	4.212274e-40	2	GZMB
7.867607e-34	3.543229	0.651	0.144	1.078964e-29	2	GNLY
4.453360e-92	2.921570	0.905	0.042	6.107338e-88	3	CD79A
4.387094e-71	2.459229	0.917	0.105	6.016461e-67	3	HLA-DQA1
3.332300e-52	2.364653	0.571	0.030	4.569916e-48	3	TCL1A

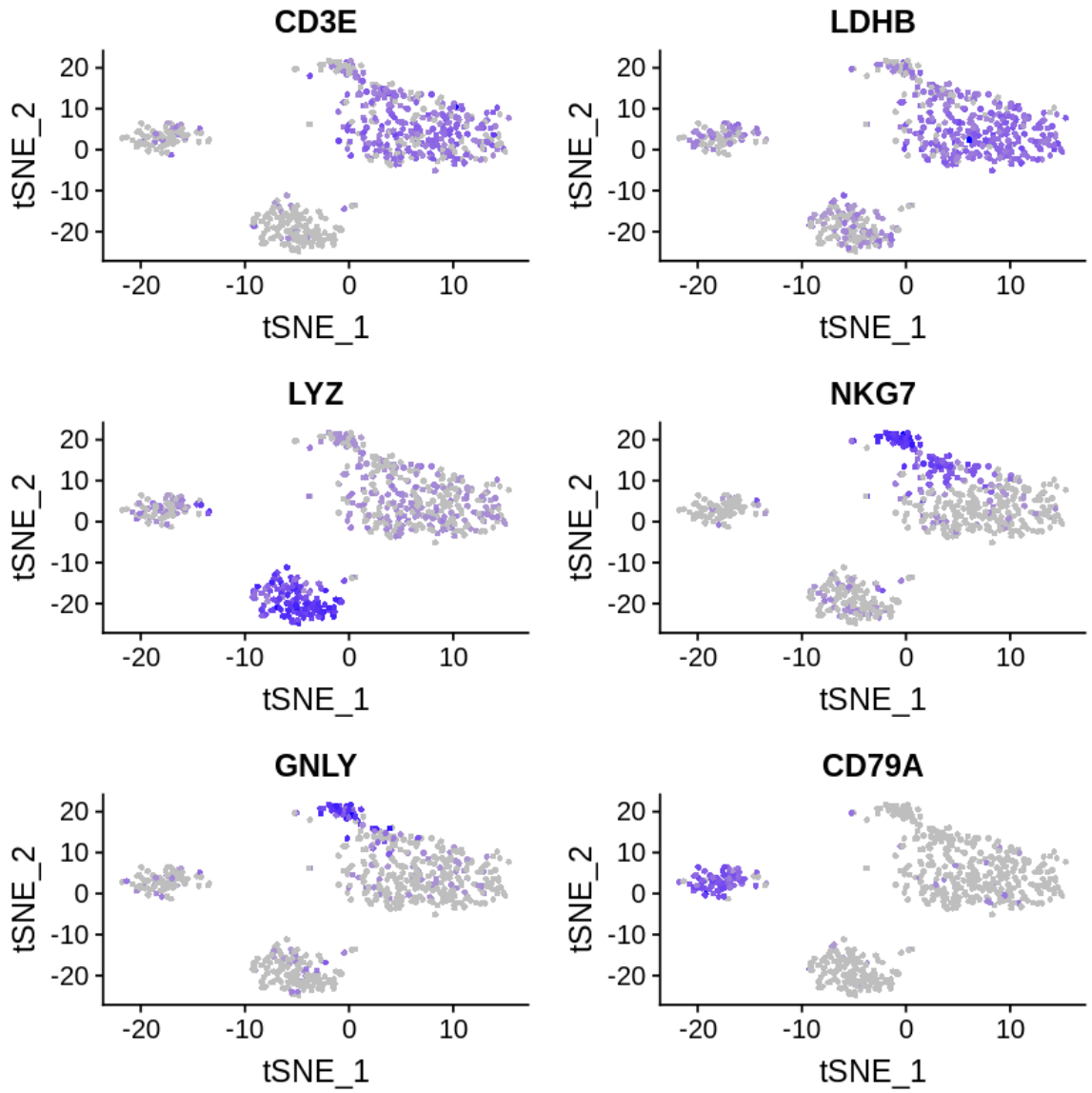
Visualizing marker genes

```
In [24]: VlnPlot(object = pbmc_small, features.plot = c("CD3E","LDHB", "LYZ", "NKG7", "GNLY", "CD79A"))
```



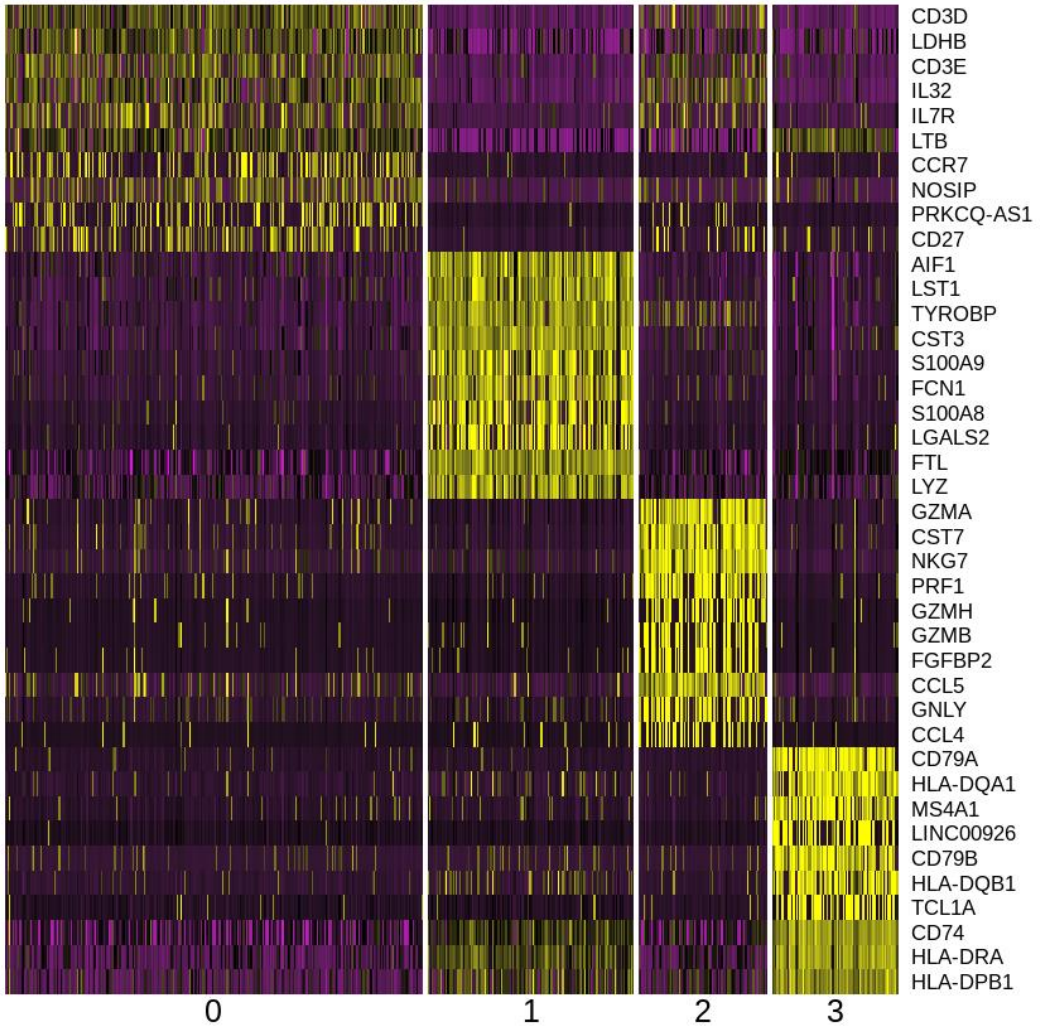
Visualizing marker genes

```
In [25]: FeaturePlot(object = pbmc_small, features.plot = c("CD3E","LDHB", "LYZ", "NKG7", "GNLY", "CD79A"),
                     cols.use = c("grey", "blue"), reduction.use = "tsne")
```



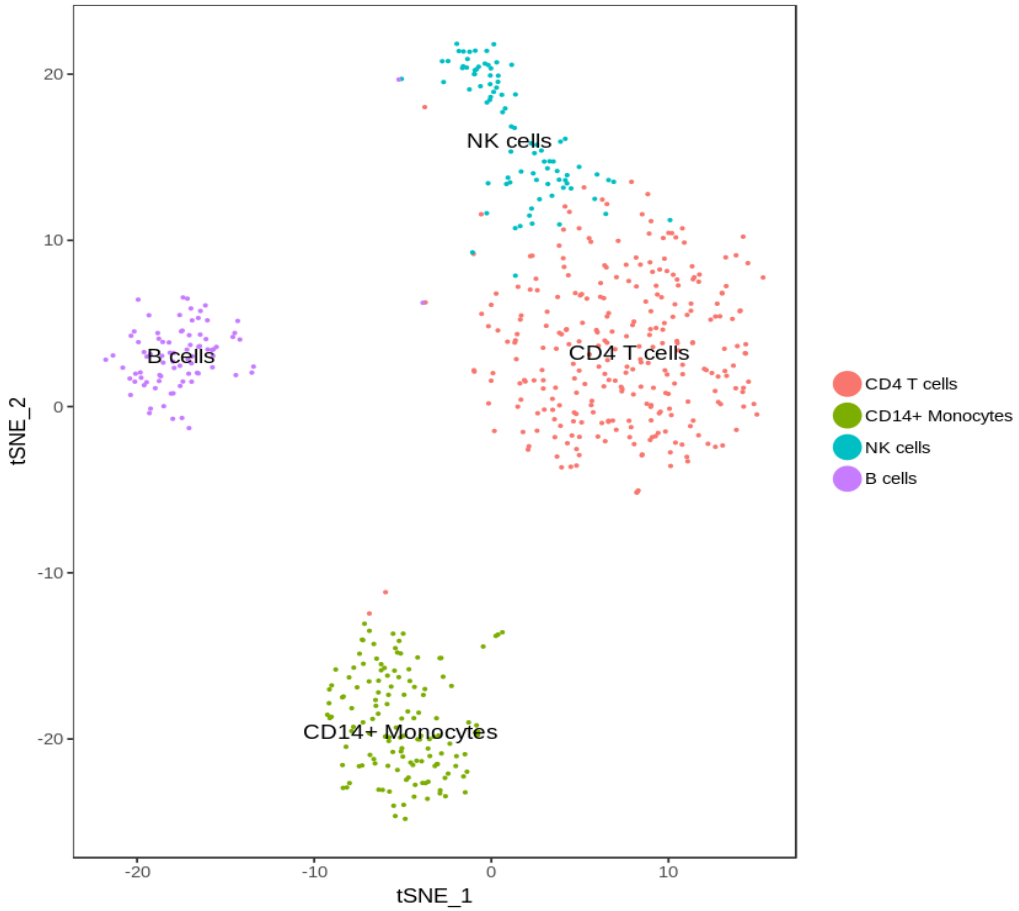
Visualizing marker genes

In [26]:
```
# we are plotting the top 10 markers (or all markers if less than 10) for each cluster.
top10 <- pbmc.markers %>% group_by(cluster) %>% top_n(10, avg_logFC)
# setting slim.col.label to TRUE will print just the cluster IDS instead of
# every cell name
DoHeatmap(object = pbmc_small, genes.use = top10$gene, slim.col.label = TRUE, remove.key = TRUE)
```



Assigning cell type identity to clusters

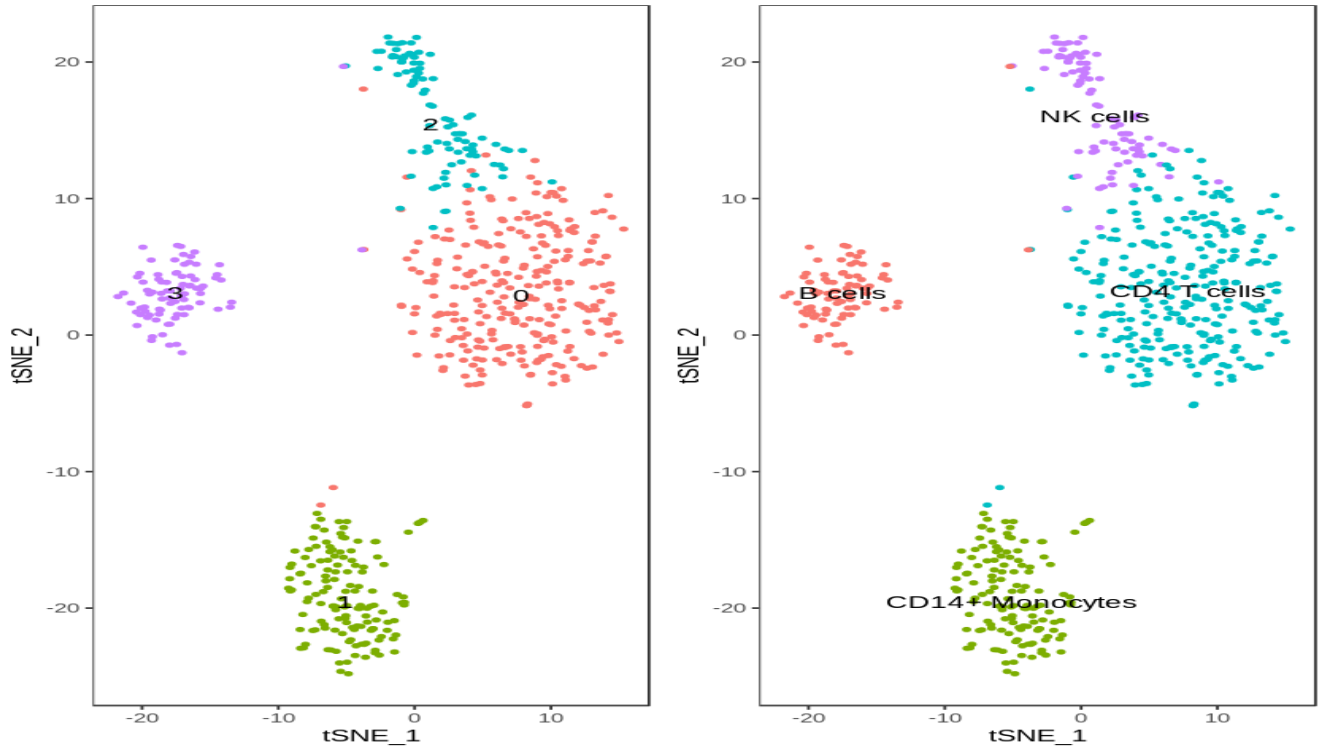
In [27]:
```
# Assigning cell type identity to clusters
# Cluster ID ⟶ Markers ⟶ Cell Type
#0 ⟶ IL7R ⟶ CD4 T cells
#1 ⟶ CD14, LYZ ⟶ CD14+ Monocytes
#2 ⟶ GNLY, NKG7 ⟶ NK cells
#3 ⟶ MS4A1 ⟶ B cells
current.cluster.ids <- c(0, 1, 2, 3)
new.cluster.ids <- c("CD4 T cells", "CD14+ Monocytes", "NK cells", "B cells")
pbmc_small@ident <- plyr::mapvalues(x = pbmc_small@ident, from = current.cluster.ids, to = new.cluster.ids)
TSNEPlot(object = pbmc_small, do.label = TRUE, pt.size = 0.5)
```



Further subdivisions within cell types

In [28]:
```
#Further subdivisions within cell types
# First lets stash our identities for later
pbmc_small <- StashIdent(object = pbmc_small, save.name = "ClusterNames_0.6")

# Note that if you set save.snn=T above, you don't need to recalculate the
# SNN, and can simply put: pbmc <- FindClusters(pbmc,resolution = 0.8)
pbmc_small <- FindClusters(object = pbmc_small, reduction.type = "pca", dims.use = 1:10,
    resolution = 0.8, print.output = FALSE)
```

In [29]:
```
# Demonstration of how to plot two tSNE plots side by side, and how to color
# points based on different criteria
plot1 <- TSNEPlot(object = pbmc_small, do.return = TRUE, no.legend = TRUE, do.label = TRUE)
plot2 <- TSNEPlot(object = pbmc_small, do.return = TRUE, group.by = "ClusterNames_0.6",
    no.legend = TRUE, do.label = TRUE)
plot_grid(plot1, plot2)
```



Find discriminating markers

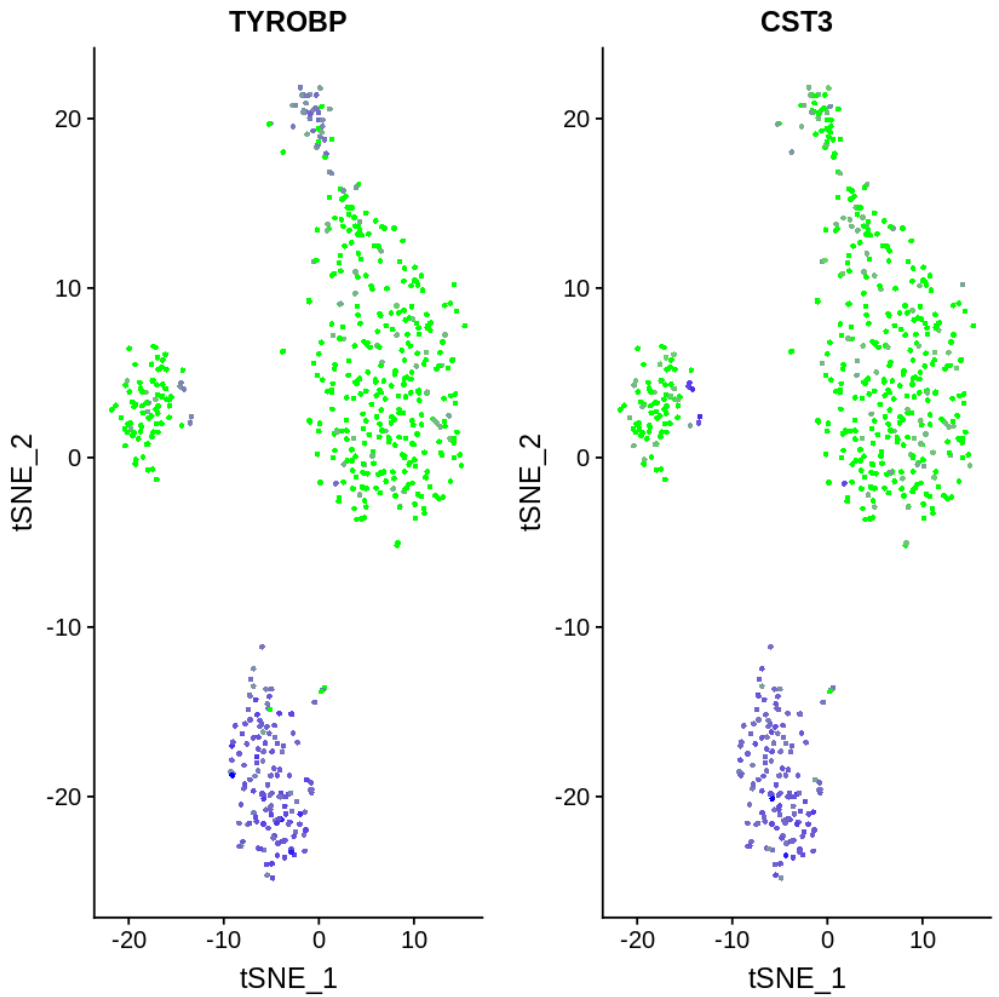
In [30]:
```
# Find discriminating markers
tcell.markers <- FindMarkers(object = pbmc_small, ident.1 = 0, ident.2 = 1)
```

In [31]:
```
print(x = head(x = tcell.markers , n = 20))
```

```
               p_val avg_logFC pct.1 pct.2    p_val_adj
TYROBP  3.380143e-73 -3.328775 0.118 0.978 4.635528e-69
CST3    2.441656e-70 -3.128186 0.173 0.993 3.348487e-66
LST1    6.314975e-69 -2.767073 0.162 0.978 8.660357e-65
AIF1    1.692267e-66 -2.706547 0.210 0.978 2.320774e-62
S100A9  1.163818e-64 -3.652983 0.147 0.942 1.596060e-60
LGALS1  1.226384e-61 -2.398197 0.261 0.978 1.681862e-57
FCER1G  1.520521e-61 -2.627984 0.121 0.906 2.085242e-57
FTH1    7.278900e-61 -2.168327 0.993 1.000 9.982283e-57
FTL     1.246951e-60 -2.419805 0.982 1.000 1.710069e-56
FCN1    1.723035e-60 -2.849617 0.118 0.884 2.362970e-56
CFD     2.865916e-60 -2.192691 0.037 0.812 3.930317e-56
LYZ     3.814118e-58 -3.354936 0.482 0.978 5.230681e-54
CD68    5.093519e-58 -1.990388 0.029 0.790 6.985253e-54
CTSS    4.100613e-57 -2.294424 0.335 0.957 5.623581e-53
RPS27   1.711225e-55  1.019949 1.000 0.993 2.346774e-51
SAT1    1.172186e-54 -2.083349 0.404 0.978 1.607536e-50
HLA-DRA 2.264931e-54 -2.434184 0.327 0.942 3.106127e-50
PSAP    2.471197e-54 -1.970408 0.305 0.942 3.389000e-50
COTL1   5.180494e-54 -1.754621 0.544 0.993 7.104529e-50
OAZ1    5.618498e-54 -1.391211 0.871 1.000 7.705208e-50
```

Visualizing marker genes

In [32]:
```
# Most of the markers tend to be expressed in C1 (i.e. S100A4). However, we
# can see that CCR7 is upregulated in C0, strongly indicating that we can
# differentiate memory from naive CD4 cells.  cols.use demarcates the color
# palette from low to high expression
FeaturePlot(object = pbmc_small, features.plot = c("TYROBP", "CST3"), cols.use = c("green",
    "blue"))
```



Hands-on single-cell RNAseq data analysis using Seurat

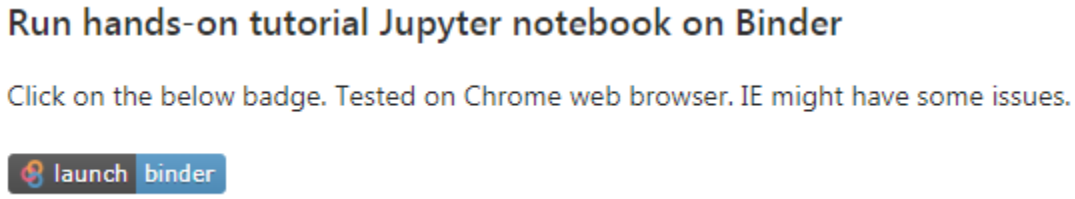
- **Prerequisites:**

  Install R version 3.3 or later (R).
  Install.packages('Seurat')

- **Single cell tutorial GitLab:** https://gitlab.rc.uab.edu/mcbios19_single_cell/single_cell_rnaseq_hands-on_1

  Please search “MCBIOS19” on Chrome web browser, go to “program”-- “Tutorial/Workshop”— “Tutorial 1”– gitlab link

- **U-BRITE Binder link :(**Tested on Chrome web browser. IE might have some issues)

  https://mybinder.org/v2/git/https%3A%2F%2Fgitlab.rc.uab.edu%2Fmcbios19_single_cell%2Fsingle_cell_rnaseq_hands-on_1.git/a20f707fc0b67f6eb4f9bf85a5daacc52c125df6

- **UAB OnDemand :** (It provides an integrated, single access point for all of your HPC resources. Login in using your Cheaha account.) https://rc.uab.edu/

Acknowledgement

Informatics Institute
(UAB) School of Medicine

Dr. Jake Chen
Christopher Fucile
Dr. Alexander Rosenberg
Dr. Andrew Crouse
Jelai Wang
Nafisah Ajala

Division of Clinical Immunology and Rheumatology
(UAB) School of Medicine

Dr. Shanrun Liu
Dr. John D. Mountz
Dr. Hui-Chen Hsu