bioinformatics.ca

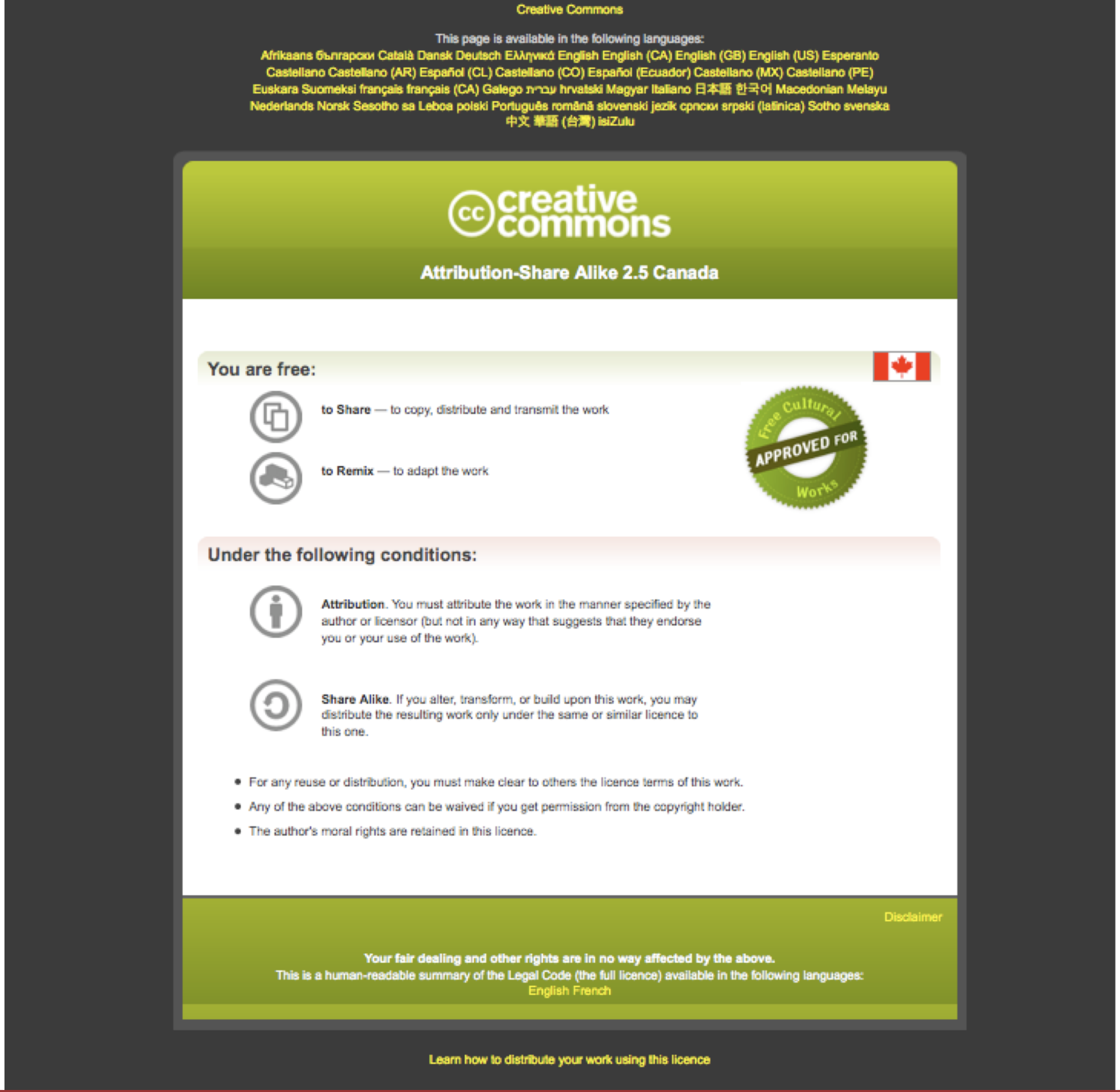
Canadian Bioinformatics Workshops

www.bioinformatics.ca

bioinformaticsdotca.github.io

Supported by

McGill UNIVERSITY



Creative Commons

This page is available in the following languages:
Afrikaans български Català Dansk Deutsch Ελληνικά English English (CA) English (GB) English (US) Esperanto Castellano Castellano (AR) Español (CL) Castellano (CO) Español (Ecuador) Castellano (MX) Castellano (PE) Euskara Suomeksi français français (CA) Galego עברית hrvatski Magyar Italiano 日本語 한국어 Macedonian Melayu Nederlands Norsk Sesotho sa Leboa polski Português română slovenski jezik српски srpski (latinica) Sotho svenska 中文 華語 (台灣) isiZulu

creative commons

Attribution-Share Alike 2.5 Canada

You are free:

to Share — to copy, distribute and transmit the work

to Remix — to adapt the work

Under the following conditions:

Attribution. You must attribute the work in the manner specified by the author or licensor (but not in any way that suggests that they endorse you or your use of the work).

Share Alike. If you alter, transform, or build upon this work, you may distribute the resulting work only under the same or similar licence to this one.

- For any reuse or distribution, you must make clear to others the licence terms of this work.
- Any of the above conditions can be waived if you get permission from the copyright holder.
- The author's moral rights are retained in this licence.

Disclaimer

Your fair dealing and other rights are in no way affected by the above.
This is a human-readable summary of the Legal Code (the full licence) available in the following languages:
English French

Learn how to distribute your work using this licence

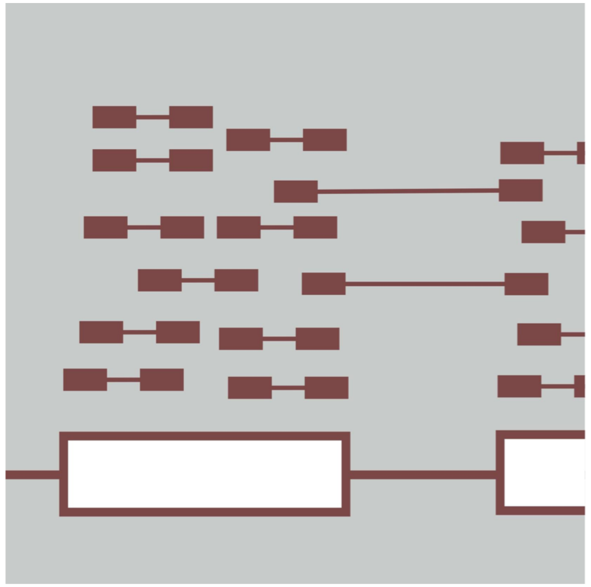
single-cell RNA-sequencing

Allegra Petti

Informatics for RNA-seq Analysis

June 17-19, 2020

Washington University School of Medicine in St. Louis



Learning Objectives for Part II: Analysis of the gene-barcode matrix

- By the end of this lecture, you will:
  - Understand the rationale underlying downstream scRNAseq analysis steps
  - Understand how to assess (and improve) the quality of scRNAseq data
  - Understand to interpret scRNA-seq data from a biological standpoint
  - Learn how to perform custom analyses of scRNAseq data using R
  - Implement a complete gene-barcode matrix analysis pipeline in R

Analysis of the gene-barcode matrix

Governed by two overarching principles:

1. Single-cell RNA-seq data is very high-dimensional

2. And very sparse. Fraction of transcripts captured per cell:
10x V2: 14-15%
10x V3: 30-32%

Methods Galore

Number of single cell tools ~ Number of single cell studies ~ 500 (https://www.scrna-tools.org/)

Point and Click:
Loupe Browser
Partek Flow ($$$)
Flow-Jo ($$$)

Large data sets:
scSVA
SAUCIE

Pseudotemporal Ordering:
Monocle 3 (R)
Slingshot (R)
PAGA (Python)
pCreode (Python)

General-purpose:
Seurat V3
Monocle V3
scran
LIGER (NMF)
CellHarmony
scAlign
Scanorama

Predicting the future:
RNAvelocity
scVelo

Cell type assignment:
SingleR
Cellassign
CellHarmony
Moana
Garnet

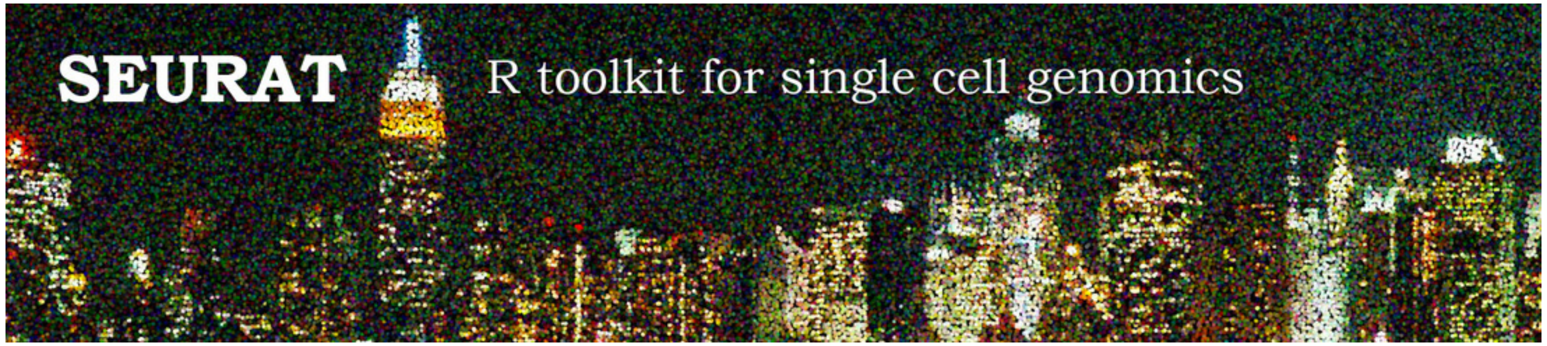
Mutation Detection:
CONICSmat (CNV)
HoneyBadger (CNV, LOH)
cb_sniffer (SNVs, Indels)
Vartrix (SNVs, Indels)

→ Need thoughtful, creative application of existing tools to extract new biology

Analysis of the gene-by-cell matrix: Overview

<u>Bold type indicates functionality not available in cellranger</u>

- For multiple samples, optionally subsample to achieve comparable sequencing depth using 'cellranger aggr' function
- **Read in data and perform initial gene and cell filtering:**
  - **Retain genes present in >= x cells**
  - **Retain cells containing >= y genes**
- Merge samples if not done in cellranger
- **Batch correction, if necessary (cellranger does <u>not</u> do this properly)**
- **Data quality overview for subsequent filtering**
  - **Plot distribution of Genes/cell, UMIs/cell, mitochondrial percentage per cell, ribosomal percentage per cell**
- **Secondary cell filtering (depends on data set, questions):**
  - **Genes and/or UMIs**
  - **Mitochondrial transcript %**
  - **Ribosomal transcript %**
- **Calculate G1/S and G2/M scores for each cell**
- **Identify variable genes, normalize, then scale the data (or use the one-step alternative, SCTransform)**
- **Remove unwanted sources of variation**
  - **Cell cycle (total cell cycle, not "Cell cycle difference")**
  - **Mitochondrial percentage**
  - **Ribosomal percentage**
  - **Cell cycle**
  - **Combinations of these variables**
- Principal Component Analysis (PCA) on variable genes
- **Retain and plot key information about each principal component (PC):**
  - Percentage of standard deviation explained
  - P-value (obtained from bootstrapping "Jackstraw")
  - Plot gene expression heatmaps for each of the top ~12 principal components
- **Choose Principal Components**:
  - Purpose: choose relative importance of minor expression signatures
  - Discontinuity in elbow plot (of standard deviation explained by each PC)
  - All PCs that explain >= 2% of SD
  - P-value from JackStraw analysis < 1×10^{-100}
  - Clarity of PC heatmaps
- Compute t-SNE and **UMAP** layouts on **n** Principal Components (NB: not raw data)
  - Single samples: 5-10
  - Multiple samples: 20-50
  - Cellranger default: 10
  - Partek default: 50
- Clustering
  - A tool for finding patterns in the data
  - Graph-based (unsupervised, must specify resolution (0.7))
  - Alternative: k-means (supervised, must specify k)
- Characterizing Clusters in terms of individual genes
  - Differentially expressed genes (numerous methods)
  - PC-perspective
    - Choose genes that contribute heavily to top principal components
    - Plot heatmaps of these genes in each cluster
    - Independent of clustering
    - Shows relationships of clusters to each other
- Cell type inference



- R package (R3.5+) containing functions and data structures for single-cell data, including scRNA-seq, scATAC-seq, CITE-seq, etc.
- Useful references:
  - https://satijalab.org/seurat/
  - Basic workflow and command list: https://satijalab.org/seurat/essential_commands.html
  - Tutorial on newest pipeline (using SCTransform): https://satijalab.org/seurat/v3.1/sctransform_vignette.html
  - Older pipeline: https://satijalab.org/seurat/v3.1/multimodal_vignette.html
  - Details on getting information and data into and out of the Seurat object: https://github.com/satijalab/seurat/wiki
- Key terms:
  - Features = Genes (and/or proteins if using CITE-seq)
  - Counts = UMIs
  - Barcodes = Cells

Getting Help on R (and Seurat) functions

1. In R terminal, type:

    ?FunctionName

    example: ?FindAllMarkers

2. Code available on github, e.g.:
https://github.com/satijalab/seurat/blob/master/man/FindClusters.Rd

Seurat V3 workflow overview
(for a single sample or pre-integrated samples)

Seurat functions in boldface blue font

```
scrna.counts <- Read10X(data.dir = "/path/to/SampleID/outs/filtered_feature_bc_matrix")
scrna <- CreateSeuratObject(counts = scrna.counts)
scrna <- NormalizeData(object = scrna)
scrna <- FindVariableFeatures(object = scrna)
scrna <- ScaleData(object = scrna)
scrna <- RunPCA(object = scrna) # Principal Component Analysis
scrna <- FindNeighbors(object = scrna) # build K-Nearest Neighbor network
scrna <- FindClusters(object = scrna) # cluster the data
scrna <- RunTSNE(object = scrna)
scrna <- RunUMAP(object = scrna)
DimPlot(object = scrna, reduction = "tsne")
UMAPPlot(object = scrna)
```

picky

Read data, create a Seurat object, and perform initial filtering

Purpose: eliminate genes with essentially no expression, and cells with very few genes

Caution!

- Rare genes may be expressed in only a few cells
- Different cell types have different numbers of genes. Example: Red blood cells express only ~200 genes in some 10x data sets.
    - Don't filter out potentially important cells.
    - Consider cell-type-specific filtering thresholds

picky

```
scrna.counts <- Read10X(data.dir = "/yourpath/outs/filtered_feature_bc_matrix")
scrna <- CreateSeuratObject(counts = scrna.counts, min.cells = 10, min.features = 100, pro
ject = Project1)
```

Multiplexed version: Combining multiple samples in Seurat

Purpose: Read, filter, and merge an arbitrary number of samples into a single Seurat object

```
# matrix.dirs is a list of directories containing 10x data

data.10x = list(); # declare a list of 10x data objects

for (i in 1:nsamples) { # nsamples = number of samples
    data.10x[[i]] <- Read10X(data.dir = matrix.dirs[i]);

}

scrna.list = list(); # a list of Seurat objects

for (i in 1:length(data.10x)) {
     scrna.list[[i]] = CreateSeuratObject(counts = data.10x[[i]], min.cells=10, min.feat
ures=100, project = “Project1”);
     scrna.list[[i]][["Sample"]] = samples[i]; # optional: assign a sample name from a v
ector ‘samples’, e.g. samples = c(“A”,”B”,”C”)

}
scrna <- merge(x=scrna.list[[1]], y=c(scrna.list[[2]],scrna.list[[3]]), add.cell.ids = c
(”name1",”name2",”name3”…)) # create merged Seurat object
```

Alternative: Combining multiple samples in cellranger

cellranger downsamples the data sets to the same [lowest] sequencing depth

```
cellranger aggr --id=$AggOutName --csv=$af --normalize=mapped --mempercore=64"
```

Definitions:
$AggOutName = Name of the output directory for the aggregated samples
$af = Full path to aggregation file, e.g. /path/to/aggregationfile.csv

Aggregation file format:

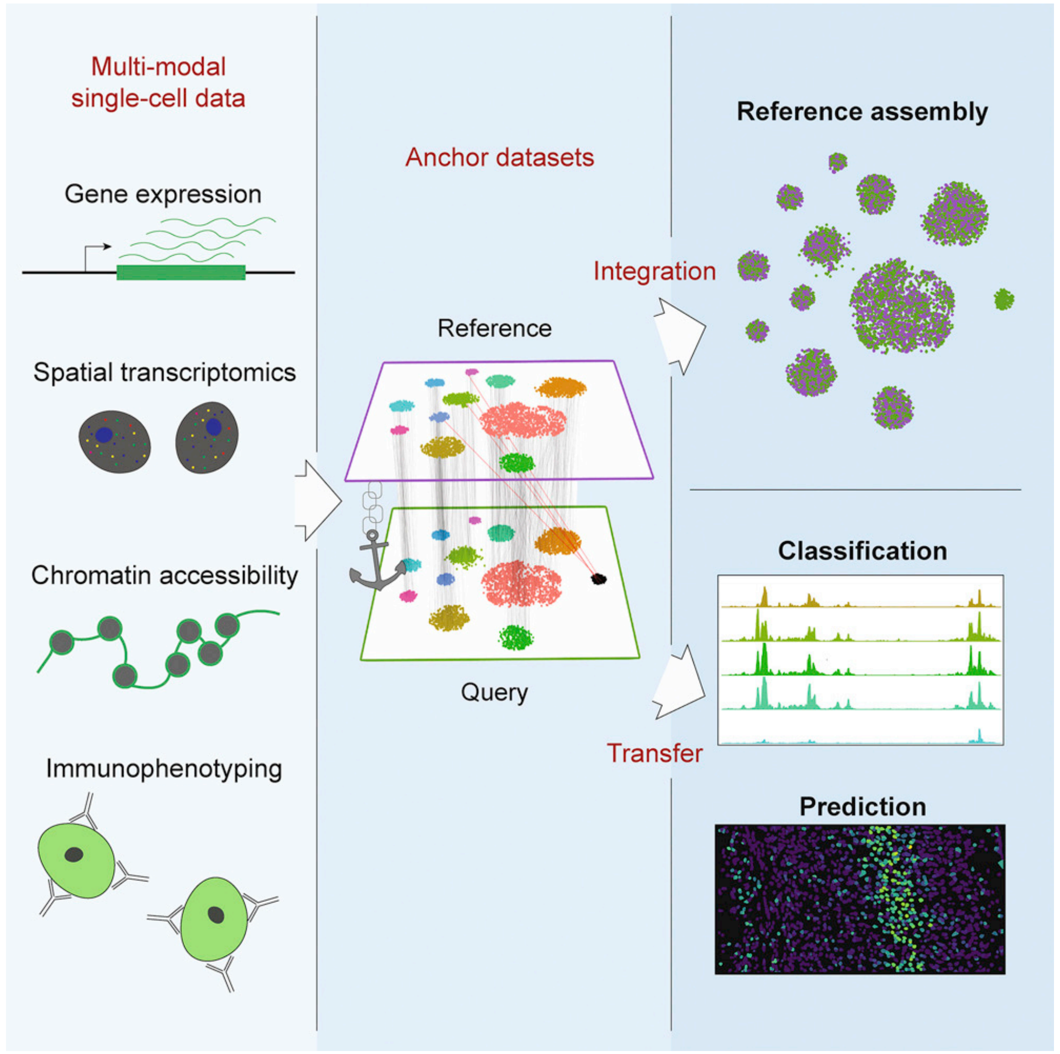
```
library_id,molecule_h5
Name1,/PathToData/Name1/outs/molecule_info.h5
Name2,/PathToData/Name2/outs/molecule_info.h5
```

Batch correct/integrate instead of merging (optional)

Avoid batch correction unless absolutely necessary

When to use it:

- Correct for different technologies (e.g. 3’ and 5’)
- Correct for known different batches
- Discover conserved biology by finding corresponding cells across different data sets
- Combining data of different types (e.g. scRNA-seq, ATAC-seq)



Batch correct/integrate instead of merging (cont'd)

```
# matrix.dirs is a list of directories containing 10x data
data.10x = list(); # declare list of 10x data sets
for (i in 1:length(matrix.dirs.)) { # for each data set
     data.10x[[i]] <- Read10X(data.dir = matrix.dirs[i]); # add it to the list
}
scrna.list = list(); # declare list of Seurat objects
for (i in 1:length(data.10x)) { # for each data set…
    scrna.list[[i]] = CreateSeuratObject(counts = data.10x[[i]], min.cells=10, min.features=100,project
=batch[i]); # …create a Seurat object for each data set
    scrna.list[[i]][["Batch"]] = batch[i]; # assign a batch label from 'batch'
    scrna.list[[i]][["Sample"]] = samples[i]; # assign a sample name from 'samples'
}
# Not shown: Filter each sample separately and normalize (using same method for all samples)
anchors <- FindIntegrationAnchors(object.list = scrna.list, dims = 1:30) # find anchors
scrna.int <- IntegrateData(anchorset = anchors, dims = 1:30) # Integrate data
DefaultAssay(object = scrna.int) <- "integrated" # make integrated data the default for downstream anal
yses
```

Batch correction and integration: The fine print

- Integrated values not intended for use with differential expression calculations.
  - We recommend running your differential expression tests on the “unintegrated” data. By default this is stored in the “RNA” Assay. There are several reasons for this.
  - The integration procedure inherently introduces dependencies between data points. This violates the assumptions of the statistical tests used for differential expression.
- SCTransformed data requires a different batch-correction workflow. (DE is not supported yet). See https://satijalab.org/seurat/v3.0/integration.html
- TransferData function uses data integration to classify cells based on a reference data set.
- Cellranger does faux batch-correction (corrected values are discarded), but batch-corrected tSNE can be visualized in the loupe browser.

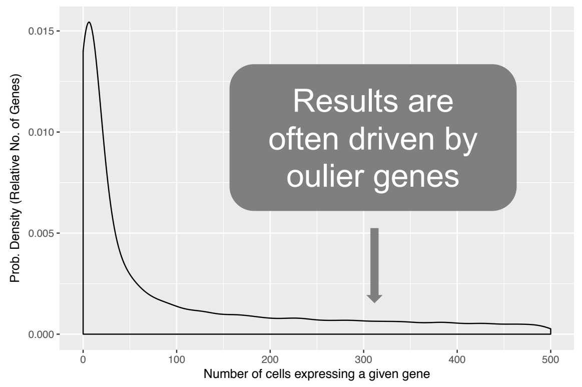
Plot key parameters (per sample) to choose filtering cutoffs

- Goals of Plotting and Filtering:
- Eliminate apoptosing or leaky cells (high percentage of mitochondrial transcripts)
- Eliminate free-floating transcripts or under-sequenced cells (too few UMIs or genes per cell)
- Eliminate doublets (too many UMIs or genes per cell)
- Know what's in your data
    - Is it dominated by a few genes?
    - Is it dominated by ribosomal protein-encoding genes?
    - How heterogeneous are the cells?
    - Does the data suggest that the sample preparation protocol need to be adjusted?

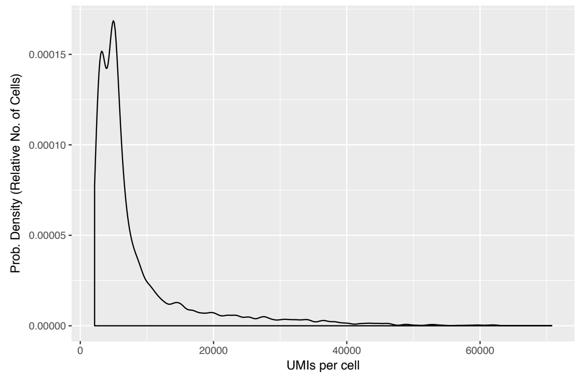
Properties of the data influence interpretation and analysis

Key variables have *wide* ranges, multimodal distributions

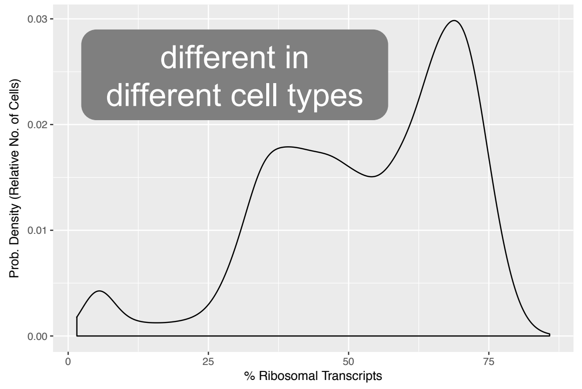
Most genes measured in a few cells!



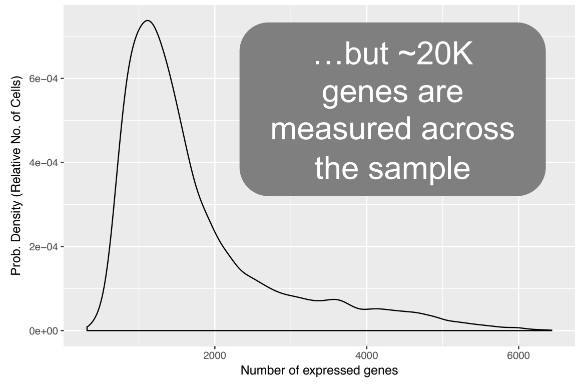
Distribution of UMIs/cell



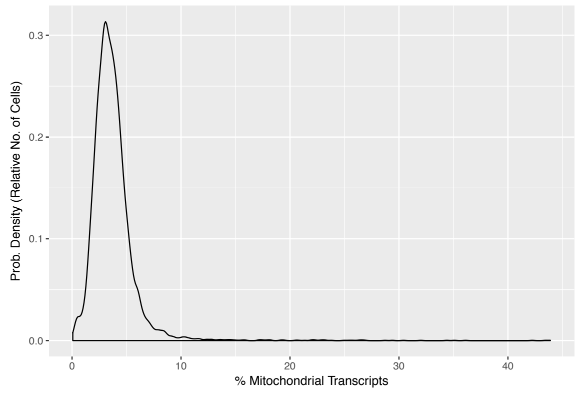
Distribution of Ribosomal transcripts/cell



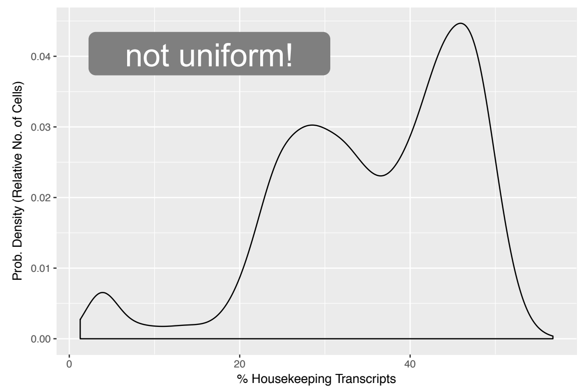
Most cells: ~1200 genes/cell



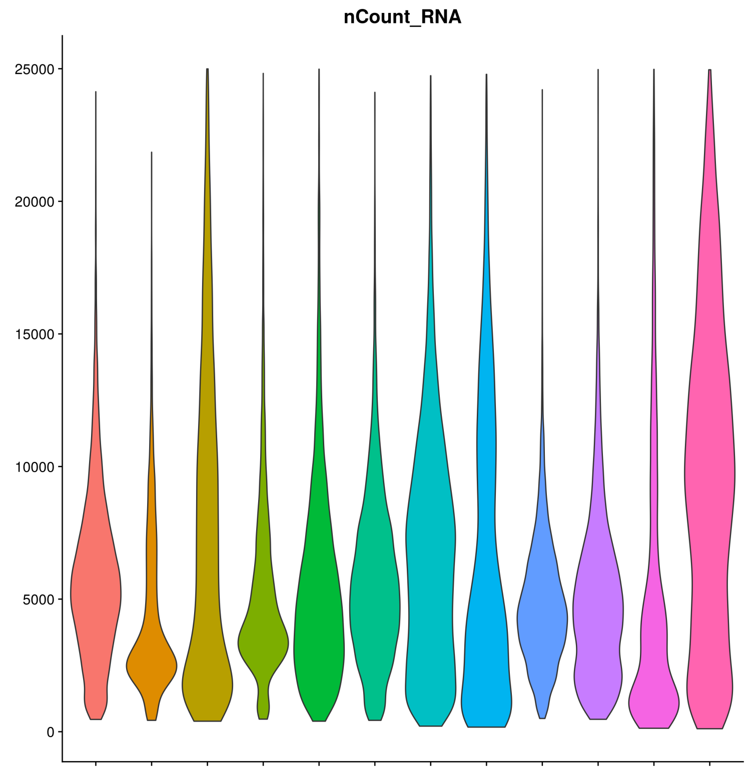
Distribution of MC transcripts/cell

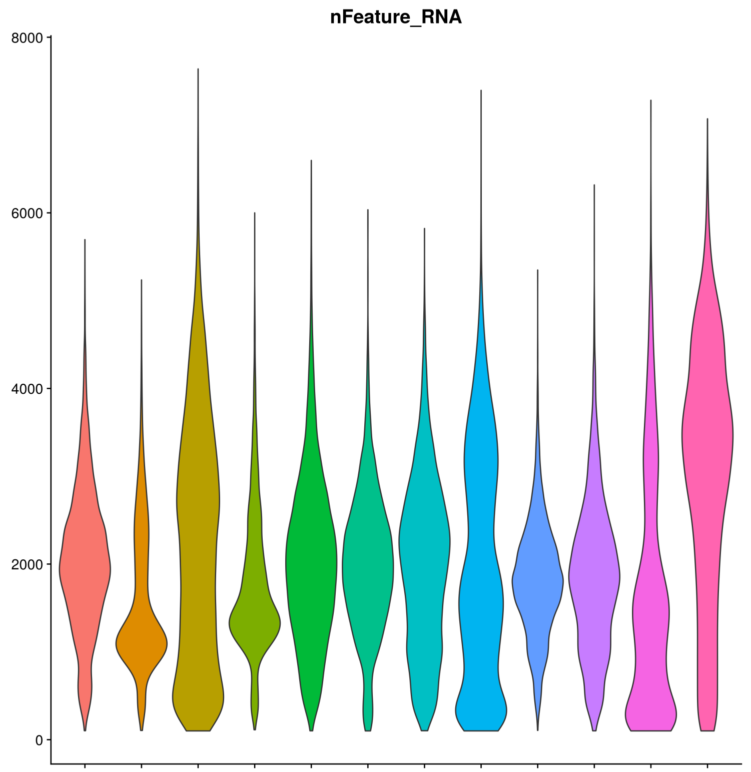


Distribution of HK transcripts/cell



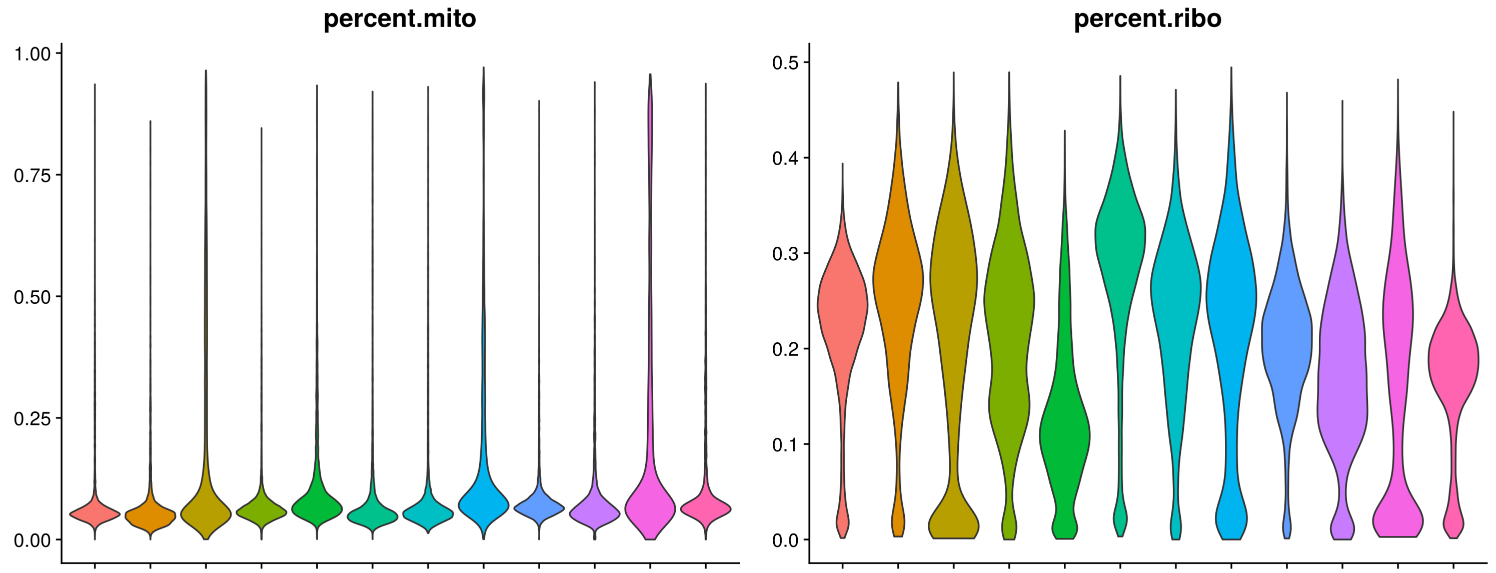
Plot counts (UMIs) and features (genes) for every data set





Very heterogeneous, filter separately

Plot Mitochondrial and Ribosomal Protein content for every data set



- High MC content implies cellular damage (resulting in loss of cytoplasmic transcripts) or impending apoptosis
  - Ilicic, et al. Classification of low quality cells from single-cell RNA-seq data. (2016) Genome Biology 17:29
  - Marquez-Jurado et al. Mitochondrial levels determine variability in cell death by modulating apoptotic gene expression. (2018) Nat. Comm. 9:389
- Ribosomal content: Reflects transcriptional diversity, cell type diversity, and probably proliferative potential. (I do not recommend filtering for ribosomal content.)

Sample R and Seurat code for parameter plots

NB: The number of genes and UMIs (nGene and nUMI) are automatically calculated for every object by Seurat and stored in the meta data.

```
# Calculate percentage of mitochondrial genes
mito.genes <- grep(pattern = "^MT-", x = rownames(x = scrna), value = TRUE);
percent.mito <- Matrix::colSums(x = GetAssayData(object = scrna, slot = 'counts')[mito.genes, ]) /
Matrix::colSums(x = GetAssayData(object = scrna, slot = 'counts'));
scrna[['percent.mito']] <- percent.mito; # assign it to the meta data

# ribosomal genes
ribo.genes <- grep(pattern = "^RP[SL][[:digit:]]", x = rownames(x = scrna), value = TRUE);
percent.ribo <- Matrix::colSums(x = GetAssayData(object = scrna, slot = 'counts')[ribo.genes, ]) /
Matrix::colSums(x = GetAssayData(object = scrna, slot = 'counts'));
scrna[['percent.ribo']] <- percent.ribo; # assign it to the meta data

vln <- VlnPlot(object = scrna, features = c("percent.mito", "percent.ribo"), pt.size=0, ncol = 2,
group.by="Batch"); # make a violin plot, and color by batch or sample

vln <- VlnPlot(object = scrna, features = "nCount_RNA", pt.size=0, group.by="Batch", y.max=25000)

vln <- VlnPlot(object = scrna, features = "nFeature_RNA", pt.size=0, group.by="Batch")
```

Some numbers
(50K reads/cell, 3’ V2 kit, cellranger V2, circa 2017)

The average gene is detected in ~150 cells

~30-50% of the reads are from transcripts that encode ribosomal proteins

When detected, a gene is represented by one read on average, but the range is huge!

Metric	Sample1	Sample2	Sample3
gene.total	21342	21036	22377
gene.per.cell.mean	1499	1454	1751
gene.per.cell.med	1381	1438	1395
gene.per.cell.min	431	383	317
gene.per.cell.max	4447	4059	6435
gene.per.cell.sd	524	478	1052
cell.per.gene	168	123	123
umi.per.cell.mean	4186	4361	8412
umi.per.cell.med	3570	4144	5296
umi.per.cell.min	1655	1278	2143
umi.per.cell.max	21273	18114	70759
umi.per.cell.sd	2253	2049	8574
umi.per.gene.mean	1	1	1
umi.per.gene.max	1150	2397	17067
umi.per.gene.sd	2	2	5

More numbers (50K reads/cell, 3’ kit)

~50% of the reads come from just 100 genes

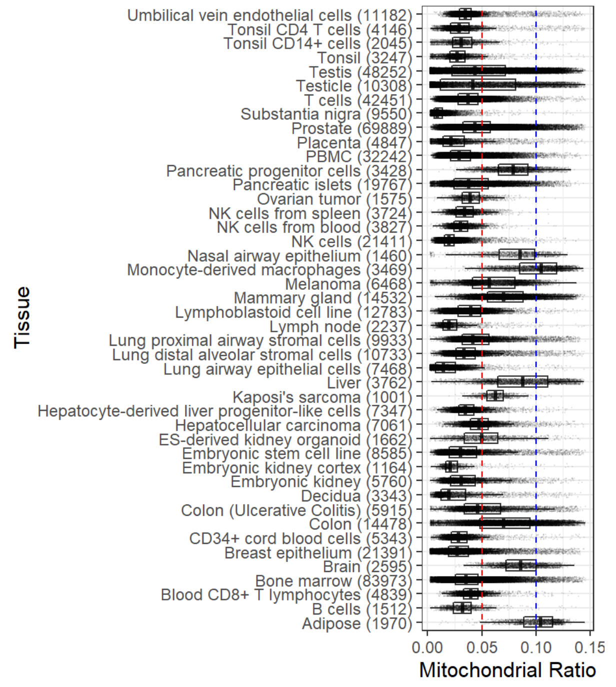
~30-50% of the reads are from transcripts that encode ribosomal proteins

~20-40% of the reads are from housekeeping genes, which are not uniformly expressed

Metric	Sample1	Sample2	Sample3
Topgene %	48	56	57
Ribogene %	32	53	51
ribo.mean %	30	51	53
ribo.med %	29	53	56
ribo.min %	12	14	2
ribo.max %	68	73	86
ribo.sd %	8	8	17
mt.mean %	7	8	4
mt.med %	6	7	3
mt.min %	2	2	0
mt.max %	56	51	44
mt.sd %	2	4	2
hkgene %	22	35	34
hk.mean %	10	16	31
hk.med %	8	15	20
hk.min %	2	2	1
hk.max %	71	72	232
hk.sd %	7	9	31

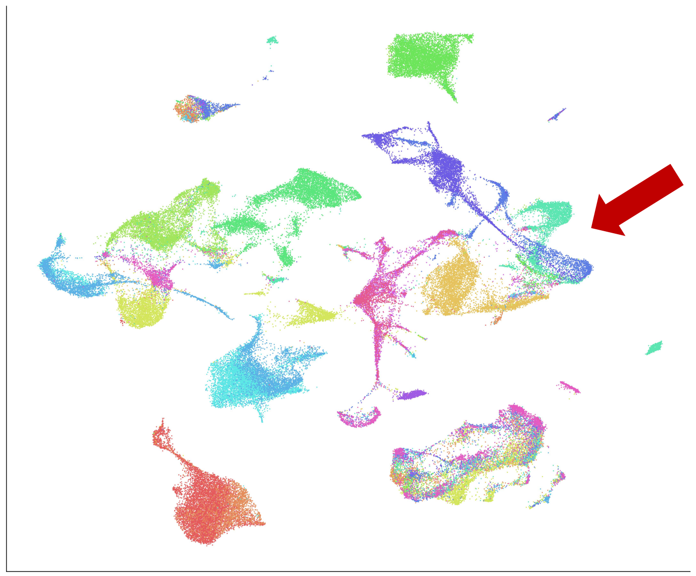
Filter data

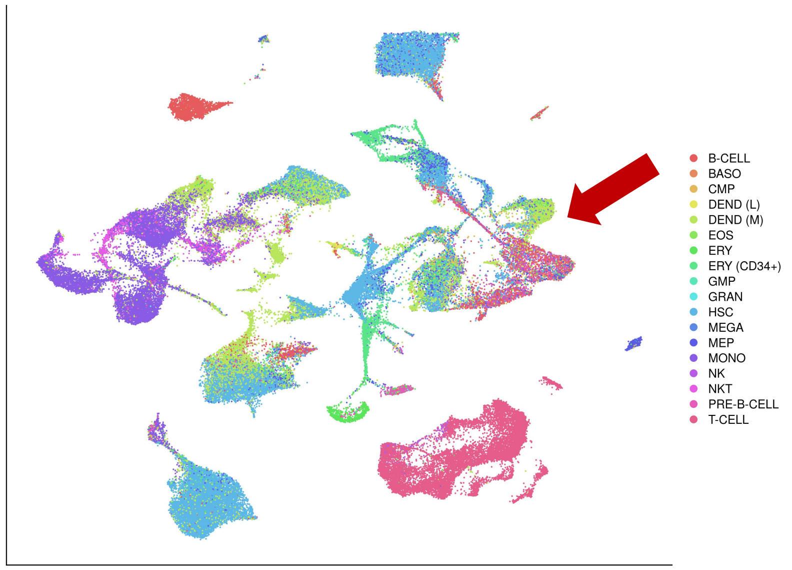
- Use violin plots and/or preliminary clustering results to choose appropriate thresholds for your data.
- Some example thresholds:
  - nFeature_RNA: between 500 and 4000, or between fixed minimum and 95th percentile (for example)
  - nFeature > x, nUMI < y percentile (based on predicted doublet rate of 0.9% per 1000 cells)
  - Mitochondrial content cutoff 5-10%, but is tissue-specific
    - AML 10%
    - 5% for mice, 10% for humans (Osorio and Cai. Systematic determination of the mitochondrial proportion in human and mice tissues for sin- gle-cell RNA sequencing data quality control (2020) BioRxiv)
  - Consider sample specific and/or cell-type specific thresholds

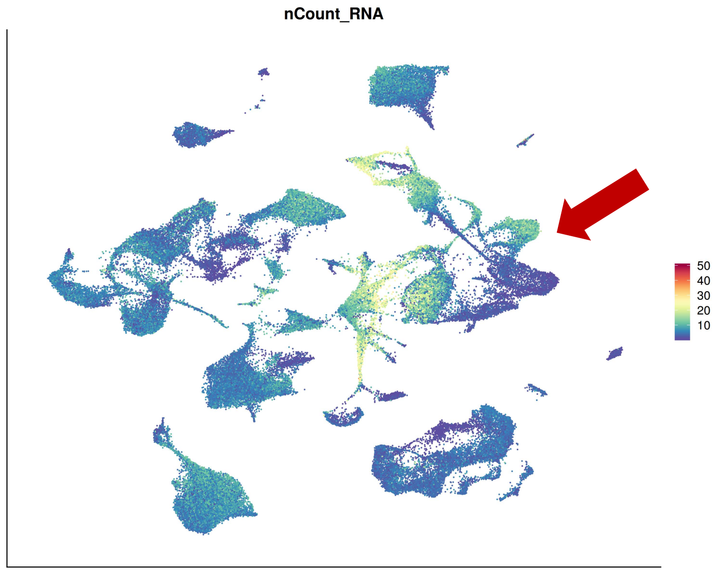


```
scrna <- subset(x = scrna, subset = nFeature_RNA > x & nUMI_RNA < y & percent.mito < z)
```

Example of possibly inadequate filtering



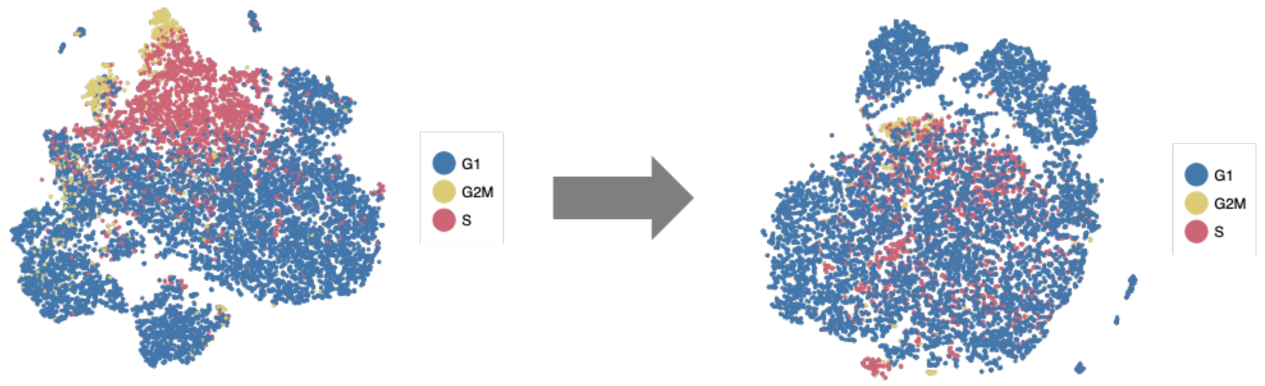




Metric	Cutoff/threshold/range
Genes	>700
UMI	< 93 percentile
Mitochondrial %	<10%

Calculate cell cycle phase of each cell

- Purpose: Cell cycle signature can dominate tSNE/UMAP plots and may need to be removed
- Seurat has a built-in function that calculates relative expression level of G1/S and G2/M genes defined in Tirosh et al 2016



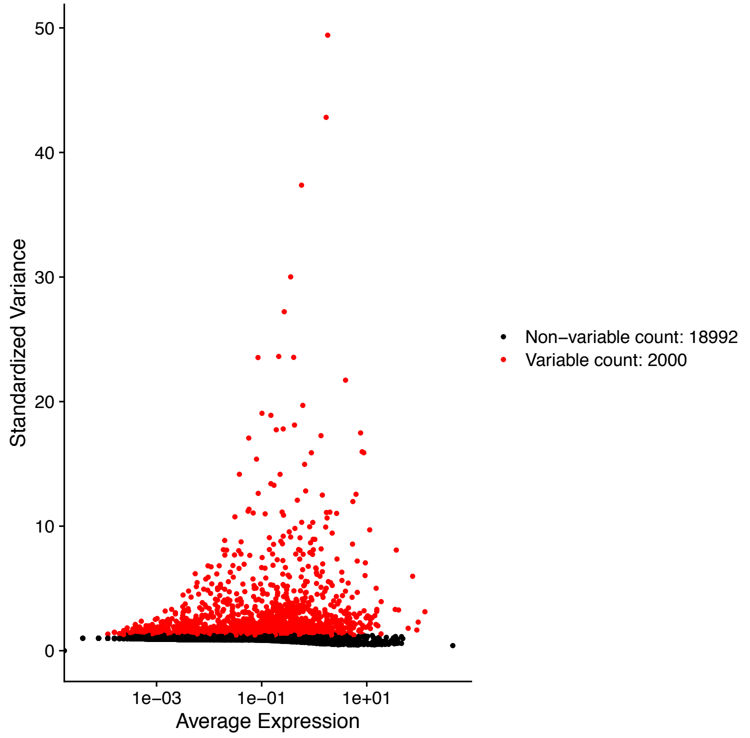
```
cell.cycle.tirosh <- read.table("CellCycleTirosh.txt", sep='\t', header=FALSE);
s.genes = cell.cycle.tirosh$V2[which(cell.cycle.tirosh$V1 == "G1/S")];
g2m.genes = cell.cycle.tirosh$V2[which(cell.cycle.tirosh$V1 == "G2/M")];
scrna <- CellCycleScoring(object=scrna, s.features=s.genes, g2m.features=g2m.genes, set.ident=FALSE)
```

Step 5: Normalize, scale, control for unwanted variation

- **Goal: remove technical effects while preserving biological variation**
- Main technical variable: sequencing depth of a cell = total UMI (nCounts)
- Even in the same experiment, different cells have very different sequencing depths
- Expression level of a gene in a cell is proportional to the sequencing depth of the cell, unless the data is normalized to sequencing depth
- Older normalization approach(es) scale every gene in the cell by the same factor: the sequencing depth
  - **NormalizeData**:
    - Divide by total counts in each cell
    - Scale to fixed counts (default is $1x10^4$ use $1x10^6$ for CPM)
    - Add 1
    - natural log
  - **FindVariableFeatures:** use a variance stabilizing transformation
  - **ScaleData:** Optionally subtract mean, divide by standard deviation, remove unwanted signal using multiple regression

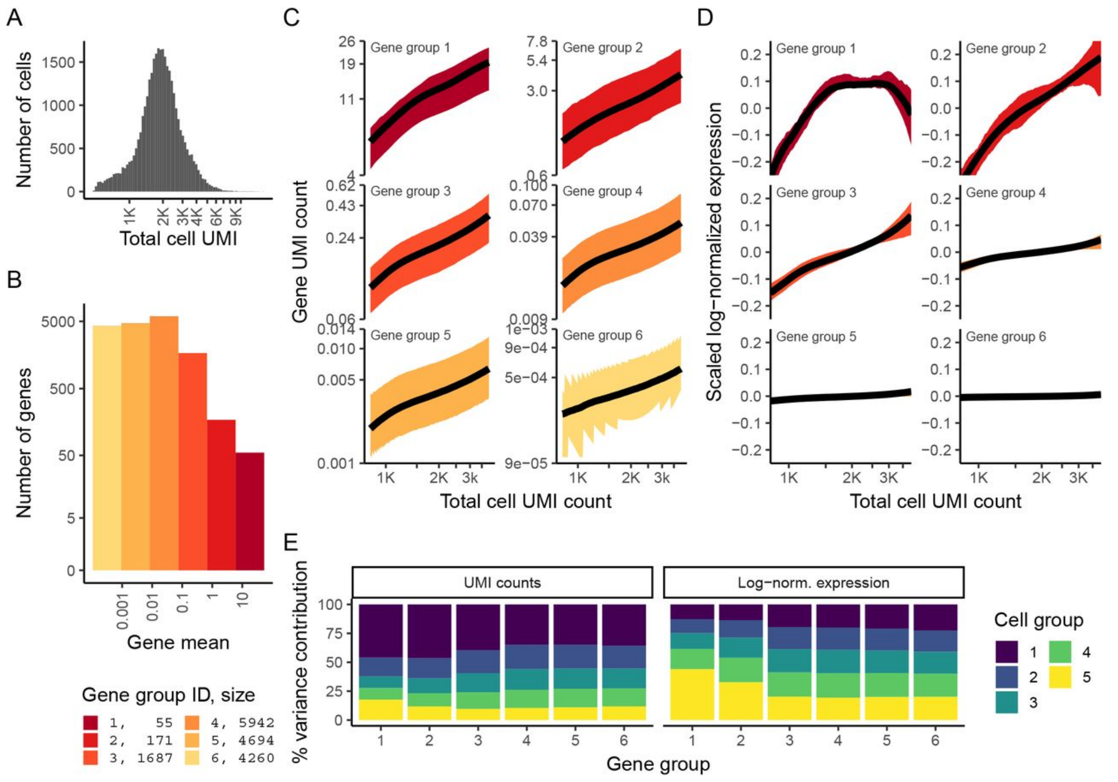
Note on FindVariableFeatures

- vst: Variance-Stabilizing Transformation
    - Standardize the expression of each gene so that its variance matches the calculated expected variance
    - Plot standardized variance vs log(mean) and choose outliers
    - Choose 2000-3000 outliers



Regularized negative binomial normalization with the SCTransform function

- Variance stabilizing transformation used in FindVariableFeatures affects different genes differently
- Highly and lowly expressed genes (B, C) are affected differently by scaling factor (D)
- Variance related to sequencing depth, and traditional normalization unevenly changes contribution of each gene to overall variance
- Solution: Use negative binomial regression (with parameters derived from groups of genes with similar expression) to remove impact of sequencing depth. Seurat function SCTransform.



Hafemeister and Satija. bioRxiv 2019

Two approaches to normalization and scaling in Seurat:

```
# I. Variance Stabilizing Transformation (with removal of cell cycle signal):

scrna <- NormalizeData(object = scrna, normalization.method = "LogNormalize", scale.factor
= 1e4) # feature counts divided by total, multiplied by scale factor, add 1, ln-transforme
d

scrna <- FindVariableFeatures(object = scrna, selection.method = 'vst', mean.cutoff = c(0.
1,8), dispersion.cutoff = c(1, Inf)) # designed to find ~2000 variable genes

scrna <- ScaleData(object = scrna, features = rownames(x = scrna), vars.to.regress = c("S.
Score","G2M.Score"), display.progress=FALSE) # center and regress out unwanted variation

# II. SCTransform (with removal of cell cycle signal):

scrna <- SCTransform(scrna, vars.to.regress = c("S.Score", "G2M.Score"), verbose=FALSE)
```

The fine print:

• scrna[["SCT"]]@scale.data contains the residuals (normalized values), and is used directly as input to PCA. To save memory, we store these values only for variable genes, by setting the return.only.var.genes = TRUE by default in the SCTransform function call.
• To assist with visualization and interpretation, we also convert Pearson residuals back to ‘corrected’ UMI counts. You can interpret these as the UMI counts we would expect to observe if all cells were sequenced to the same depth.
• The ‘corrected’ UMI counts are stored in scrna[["SCT"]]@counts. We store log-normalized versions of these corrected counts in pbmc[["SCT"]]@data, which are very helpful for visualization.
• You can use the corrected log-normalized counts for differential expression and integration. However, in principle, it would be most optimal to perform these calculations directly on the residuals (stored in the scale.data slot) themselves. This is not currently supported in Seurat v3, but will be soon.

Practical Exercise Steps 1-9

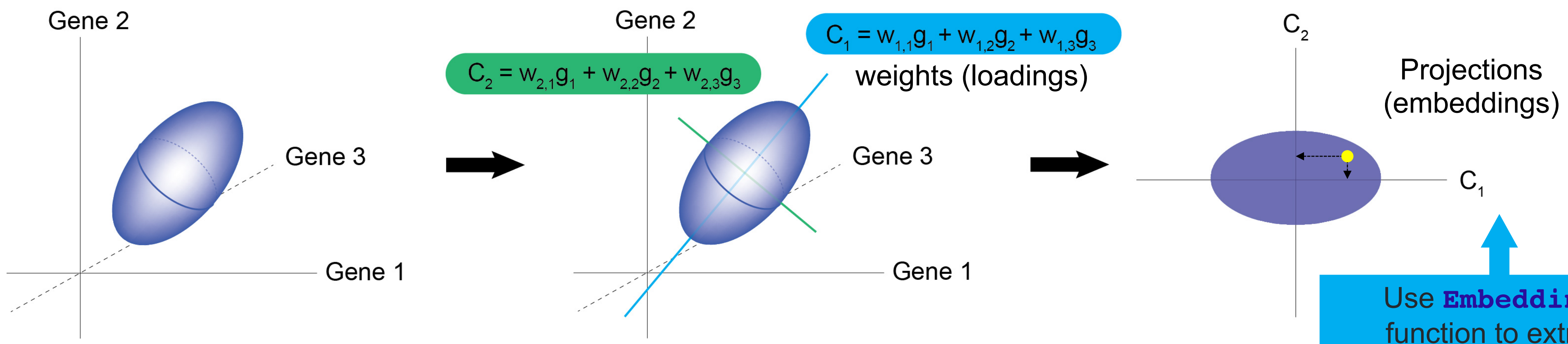
- https://rnabio.org/module-08-scrna/0008/02/01/scRNA/

We will return to the lecture during the saveRDS step at the end of step 9

Dimensionality reduction using PCA

Purpose: Approximate original data using fewer dimensions. Define new axes that capture as much "information" as possible in as few dimensions as possible.

PCA = Principal Component Analysis (similar to SVD - Singular Value Decomposition)
Principal Axes, Eigen Decomposition: Euler (1751)…Cauchy (1829)
SVD: Eugenio Beltrami, 1873 (etc)
PCA: Karl Pearson, 1901
Computation: Gene Golub, Christian Reinsch, 1970
Gene Expression: Orly Alter, Patrick Brown, David Botstein, 2000 (PNAS 97 (18): 10101-10106)
Other applications: image processing, video games, math, statistics, computer science, machine learning, finance, etc.


Gene 2
Gene 3
Gene 1
$C_2 = w_{2,1}g_1 + w_{2,2}g_2 + w_{2,3}g_3$
$C_1 = w_{1,1}g_1 + w_{1,2}g_2 + w_{1,3}g_3$
weights (loadings)
Gene 2
Gene 3
Gene 1
C_2
Projections (embeddings)
C_1

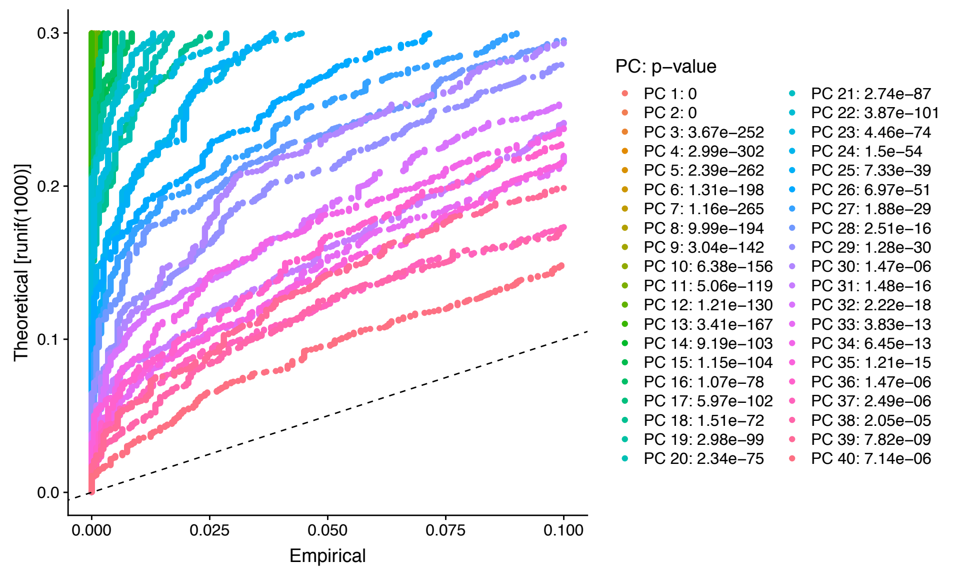
Use **`Embeddings`** function to extract embeddings and work in PCA space

Non-negative matrix factorization: Lee and Seung, 1999. (Nature 401 (6755): 788–791)
Similar to PCA, but axes are not mutually orthogonal, and clustering and factorization are coupled.

bioinformatics.ca

Selecting PCs: Jackstraw analysis

- Calculate statistical significance of each PC
- Randomly permute data
- Recalculate PCs of randomized data
- Compare “real” PCs to “random” PCs to derive significance



```
# NB: jackstraw analysis is slow
scrna <- JackStraw(object = scrna, num.replicate = 100, dims=N) # specify N: 30, 50, etc
scrna <- ScoreJackStraw(object = scrna, dims = 1:50)
js <- JackStrawPlot(object = scrna, dims = 1:20) # make plot shown above
pc.pval <- scrna@reductions$pca@jackstraw@overall.p.values # get overall pvalues for each PC
```

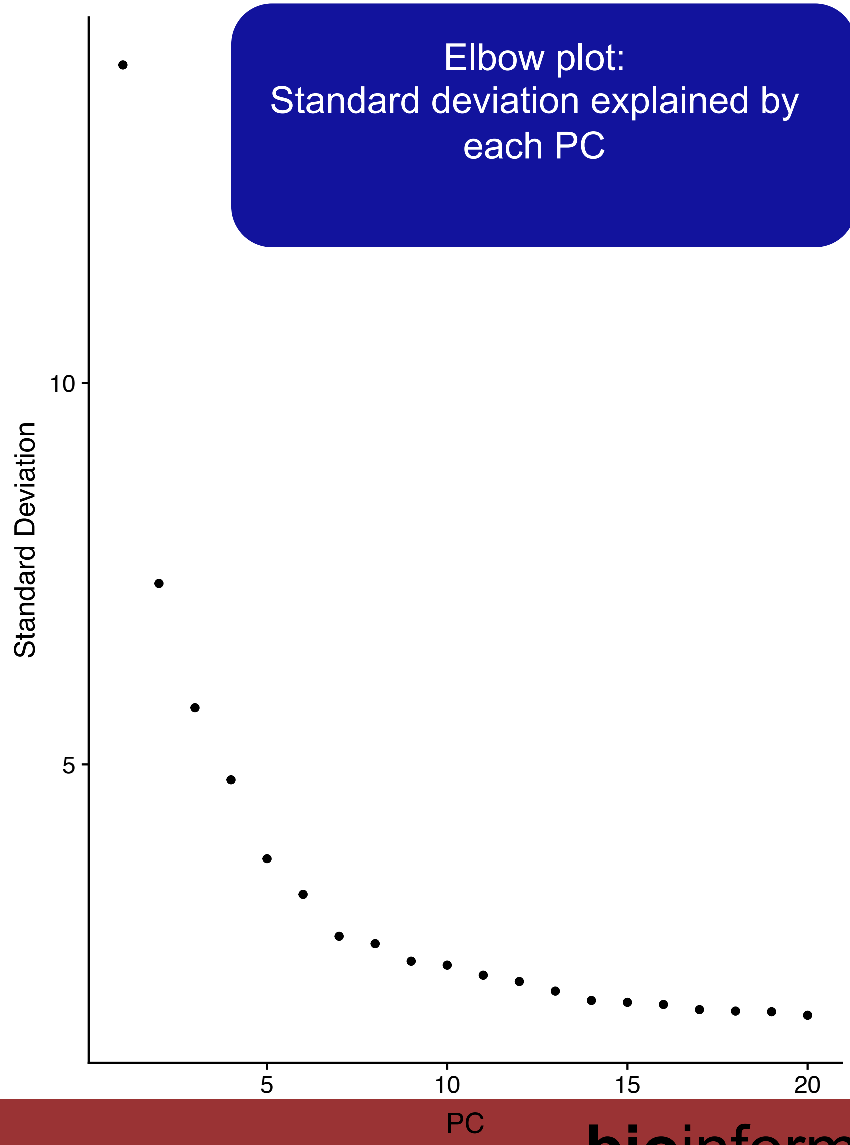
Chung and Storey. Statistical significance of variables driving systematic variation in high-dimensional data. Bioinformatics 2015

Selecting Principal Components

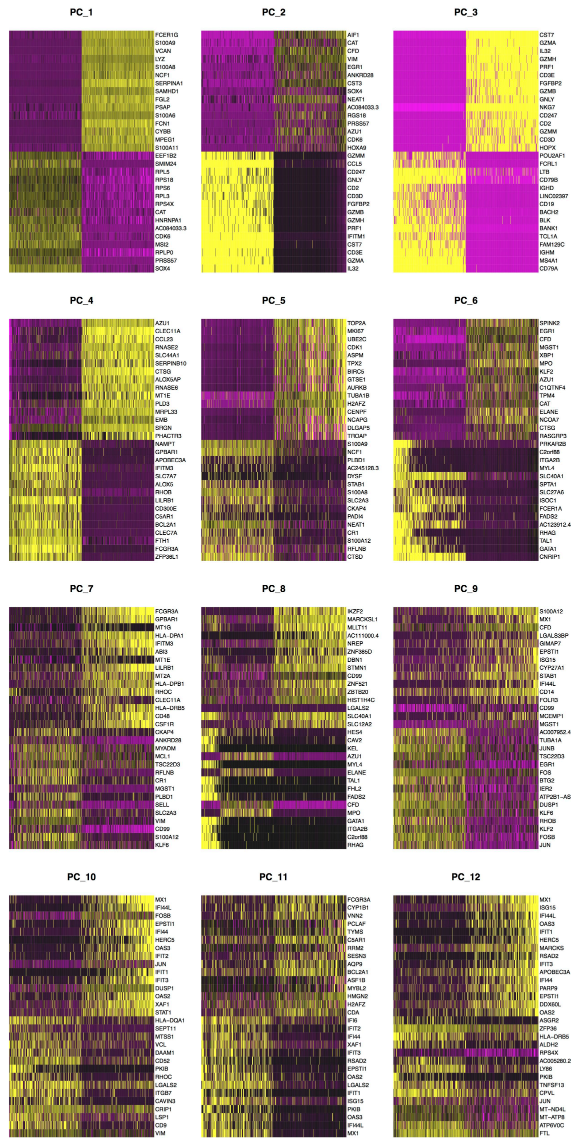
All downstream calculations are done on PCs, not raw data

- Retain and plot key information about each principal component (PC):
  - Percentage of standard deviation explained
  - P-value (obtained from bootstrapping "Jackstraw")
  - Plot gene expression heatmaps for each of the top ~12 principal components
- Choose Principal Components:
  - Purpose: choose relative importance of minor expression signatures
  - Discontinuity in elbow plot
  - All PCs that explain >= x% of SD (e.g. 2%)
  - P-value from JackStraw analysis < 1×10^{-n} (e.g. 1×10^{-100})
  - Clarity of PC heatmaps
- Number of PCs:
  - Single samples: 5-10
  - Multiple samples: 20-50
  - Cellranger default: 10
  - Partek default: 50

Elbow plot:
Standard deviation explained by each PC



Minor components contain signal that is not represented in t-SNE/UMAP



PC Tradeoff: More components → more signal, more noise

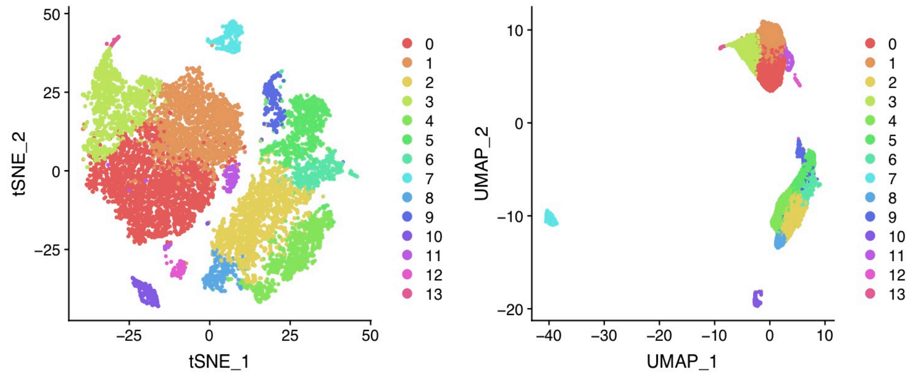
Plotting using t-SNE/UMAP

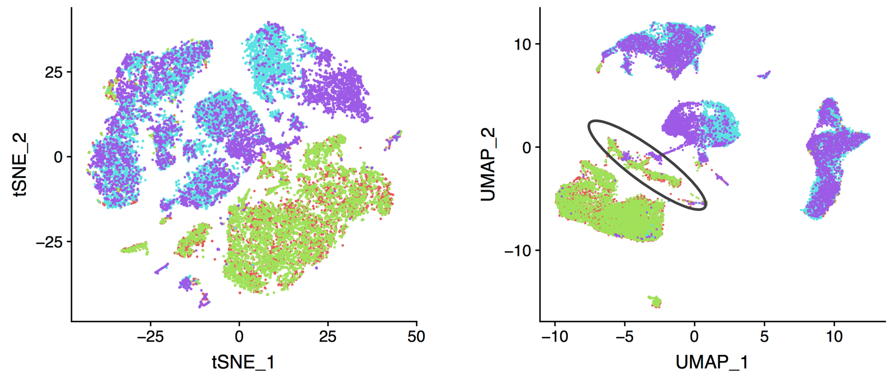
t-SNE = *t*-distributed Stochastic Neighbor Embedding
UMAP = Uniform Manifold Approximation and Projection

Goal: Embed high-dimensional data in low-dimensional space

End product: 2D plot where cells are positioned near each other if they have similar gene expression profiles. "Units" are relative and data-dependent.

- Expression "distances" between points (ie cells) in high-dimensional space are modeled using a gaussian distribution.
- Distances in low-D space are modeled using a t-distribution.
- Operates in "PCA space"
- t-SNE preserves local structure only.
- UMAP preserves local AND global structure.
- Implication: In UMAP, distances between points *and* clusters in UMAP are more interpretable in terms of expression distances/similarity.



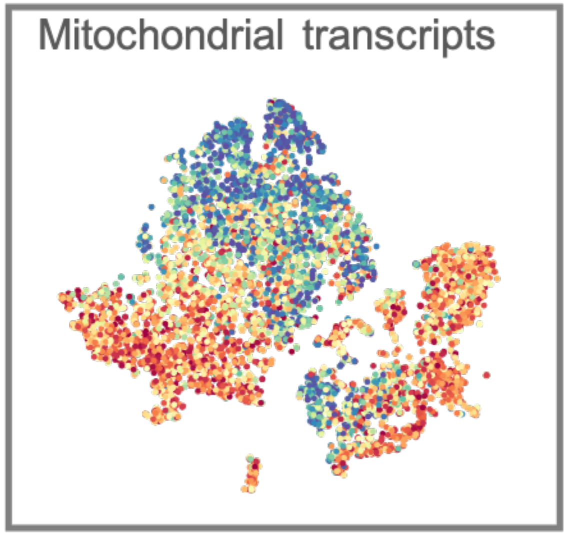


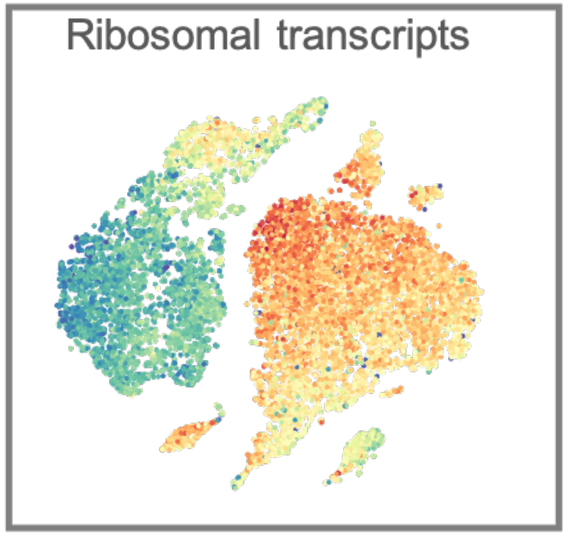
Plotting using t-SNE/UMAP

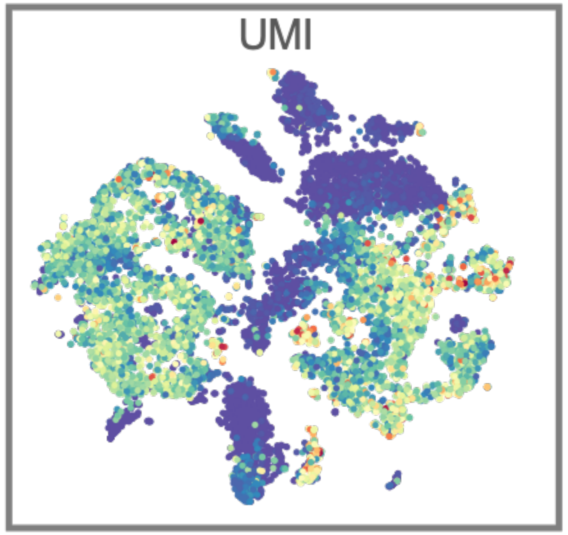
```
n = 10; # choose number of dimensions - experiment-specific!
scrna <- RunUMAP(object = scrna, reduction = "pca", dims = 1:n) # calculate UMAP
scrna <- RunTSNE(object = scrna, reduction = "pca", dims = 1:n) # calculate tSNE

# make some plots:
DimPlot(object = scrna, reduction = "tsne", group.by = ”Batch", pt.size=0.1) # color by
Batch

FeaturePlot(object = scrna, features = c("nCount_RNA"), reduction=“tsne”) # plot one or
more genes or variable on t-SNE
```

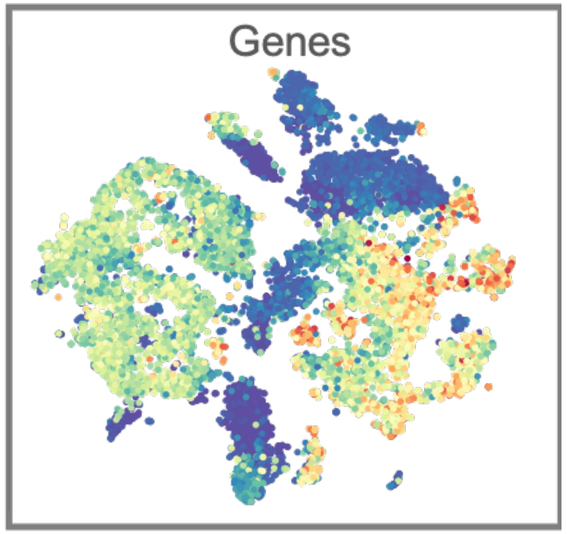
“How to use t-SNE effectively” https://distill.pub/2016/misread-tsne/

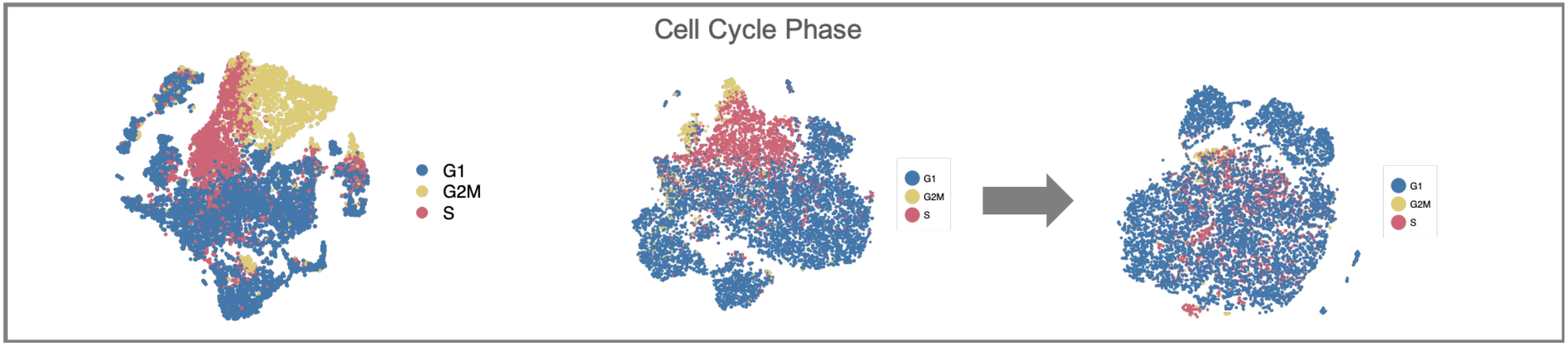
Interpreting the t-SNE/UMAP, Part I: Potentially misleading sources of variation





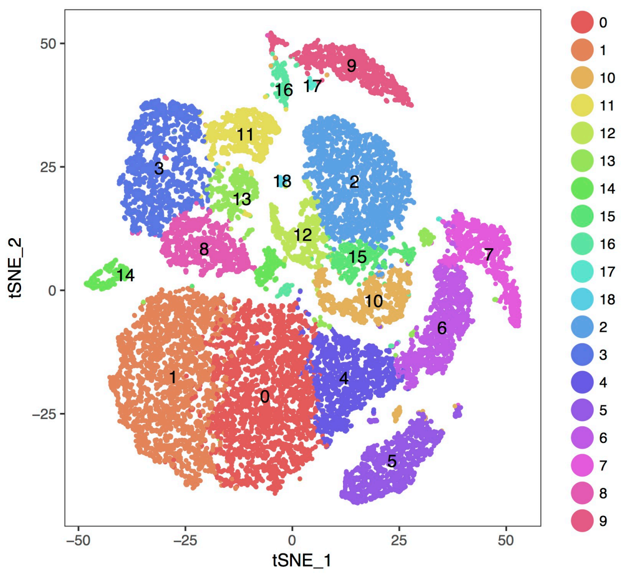


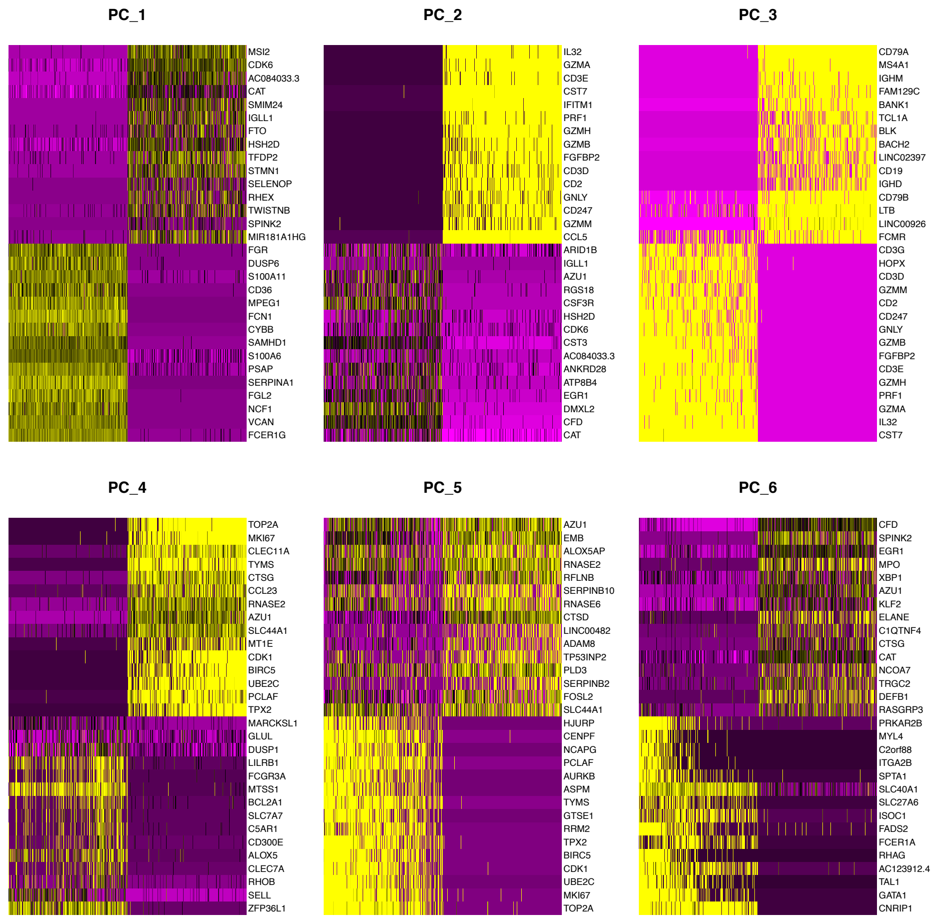




Interpreting the t-SNE/UMAP, Part II: Systematic analysis of variation

- What is driving the t-SNE/UMAP layout?
- Find genes that vary:
  - Principal components
  - Individual cluster-specific genes
- Examine across clusters/t-SNE/UMAP



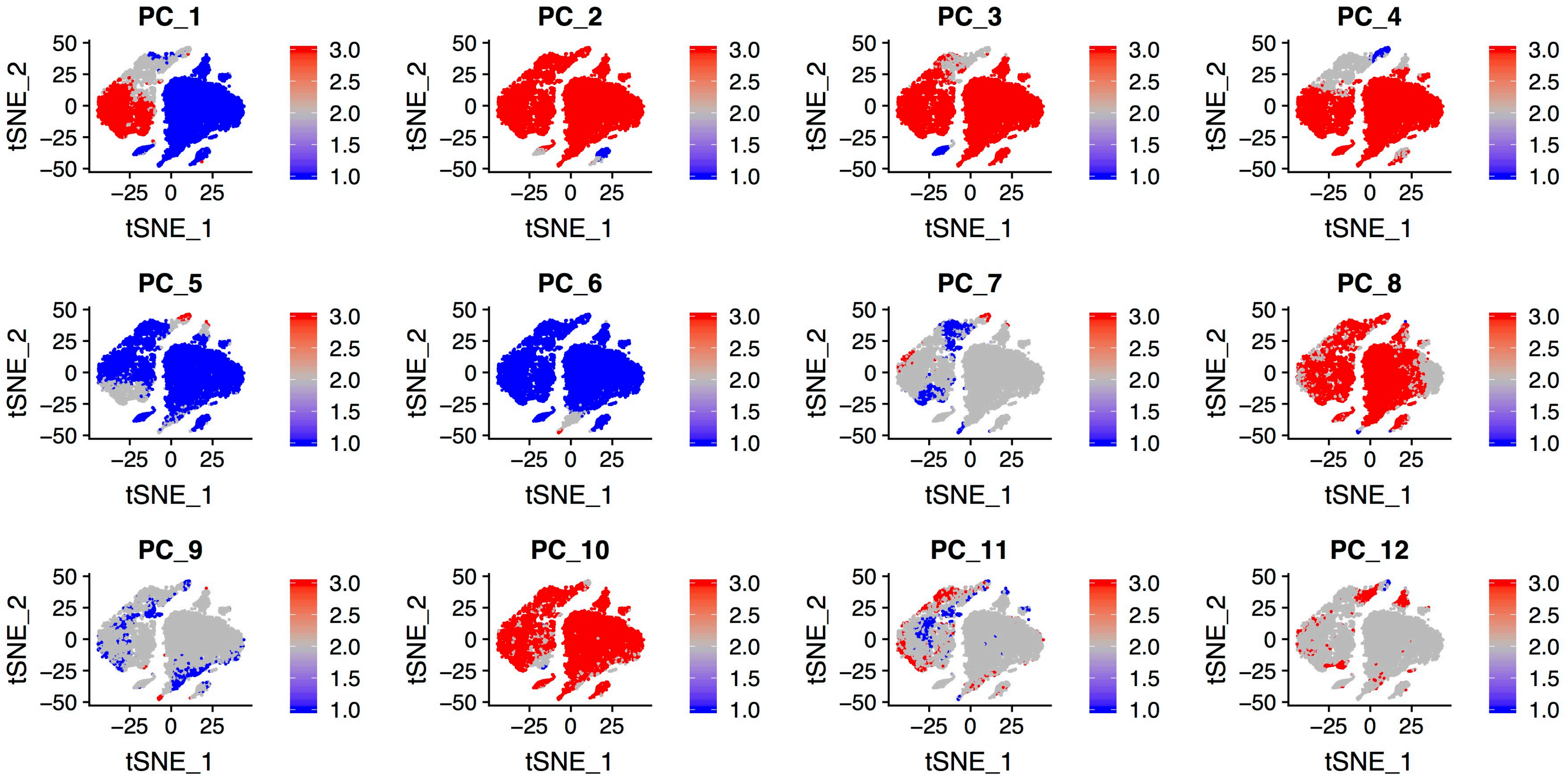


```
DimHeatmap(object = scrna, dims = 1, cells = 500, balanced = TRUE)
```

Part II, cont'd: Visualizing sources of variation

Small but distinct cell clusters may contribute heavily to the layout (may need to be removed)

PC #1 captures the biggest source of variation

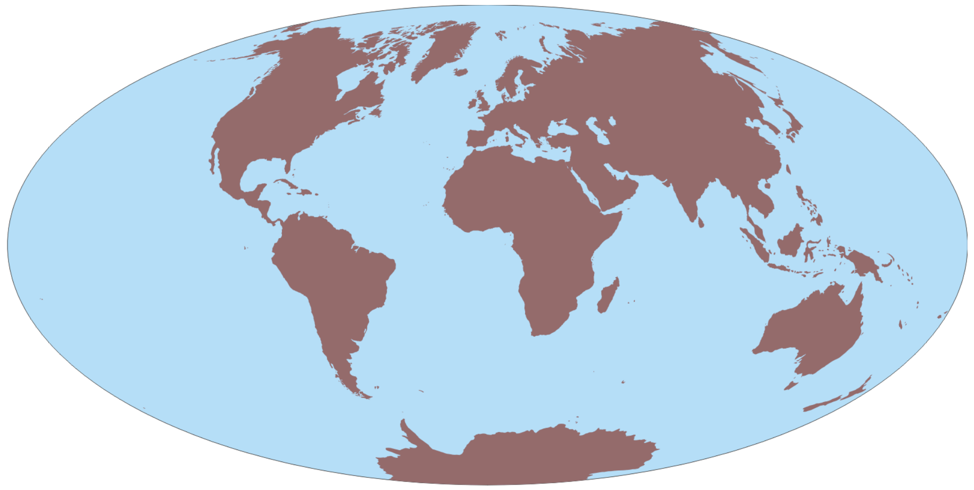


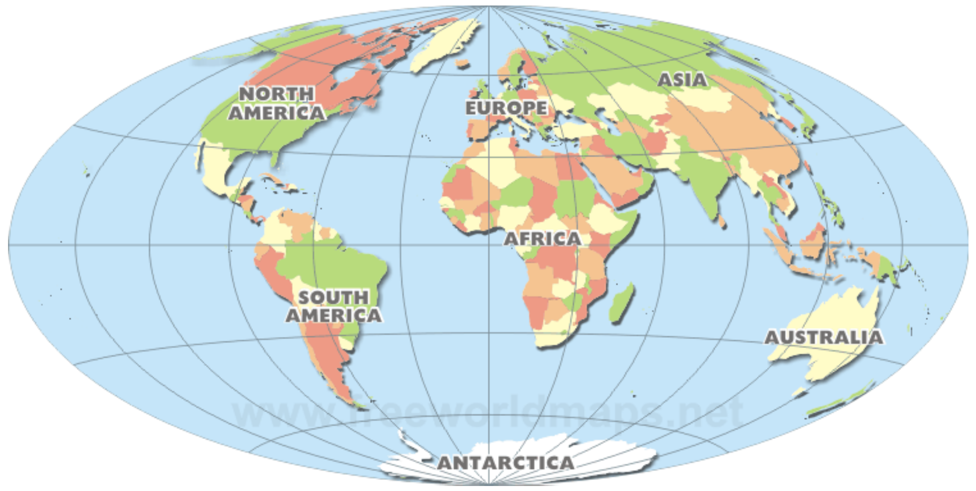
Post-PCA analysis of the gene x cell matrix

- Compute t-SNE and UMAP layouts on *n* Principal Components (NB: not raw data)
- **Clustering**
  - A tool for finding patterns in the data
  - Graph-based (unsupervised, must specify resolution (0.7))
  - Alternative: k-means (supervised, must specify k)
- Characterizing Clusters in terms of individual genes
  - Differentially expressed genes (numerous methods)
  - PC-perspective
    - Choose genes that contribute heavily to top principal components
    - Plot heatmaps of these genes in each cluster
    - Independent of clustering
    - Shows relationships of clusters to each other
- Cell lineage inference

2-D layout vs. Clustering

- tSNE and UMAP reflects natural organization of data by approximating high-dimensional relationships in low-dimensional space
- Clustering imposes structure by assigning cells to non-overlapping groups based on relative expression similarity





Clustering [Cells]

- Clustering helps organize and identify patterns in data.
- There is no “correct” or “perfect” clustering of any data set.
- Corollary: Even the best clustering may be misleading (aka “wrong”).
- Don’t take clusters too seriously – they don’t prove anything.

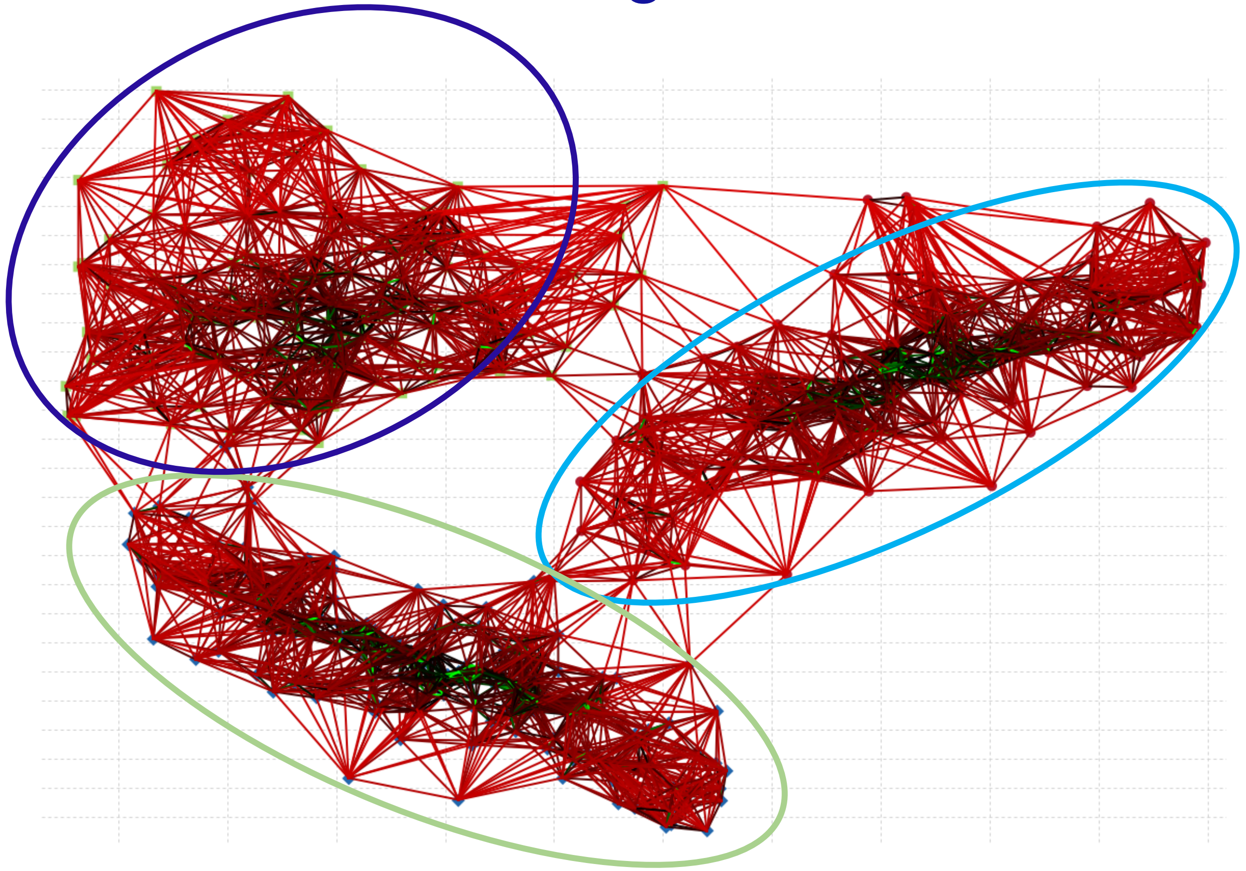
Common methods:

- Graph-based: Unsupervised, but “adjustable” using ”resolution” parameter
  - Calculates k-nearest neighbors for each cell using Euclidian distance in PCA-space; constructs shared nearest-neighbor (SNN) graph; optimize graph modularity (Waltman and van Eck algorithm)
- K-means: Supervised: specify number of clusters (available in cellranger, not Seurat).

Clustering

Step 1: Build KNN graph using Euclidian diastnce in PCA-space

Step 2: Find clusters, or cliques, or modules using Louvain modularity optimization algorithm (alternatives available)



```
nPC = 20; # specify number of PCs to use
cluster.res = 0.7; # specify resolution of clustering.
scrna <- FindNeighbors(object=scrna, dims=1:nPC);
scrna <- FindClusters(object=scrna, resolution=cluster.res);
scrna <- StashIdent(object = scrna, save.name = sprintf("ClusterNames_%.1f_%dPC",
cluster.res, nPC)) # save cluster names in a new identity (ClusterNames_0.7_20 in this
example) if desired
```

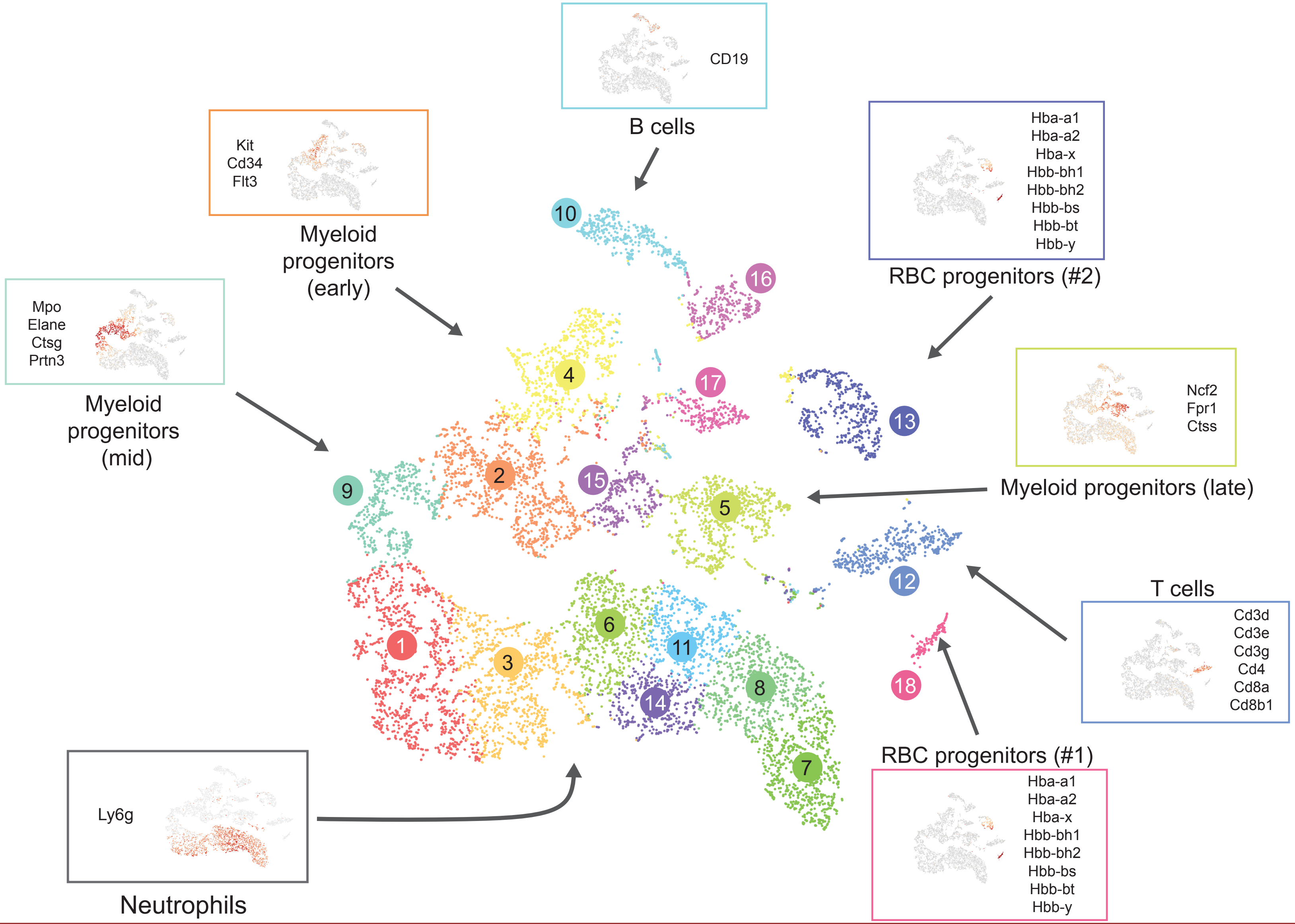
Practical Exercise Steps 10-12

- https://rnabio.org/module-08-scrna/0008/02/01/scRNA/

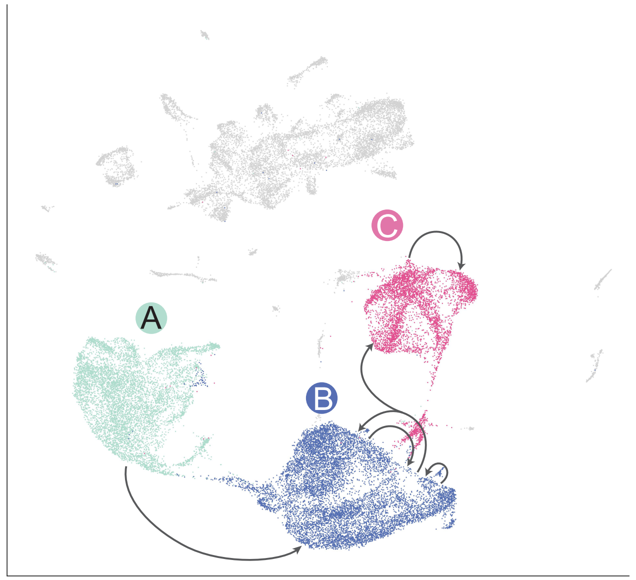
We will return to the lecture during the clustering step (Step 12)

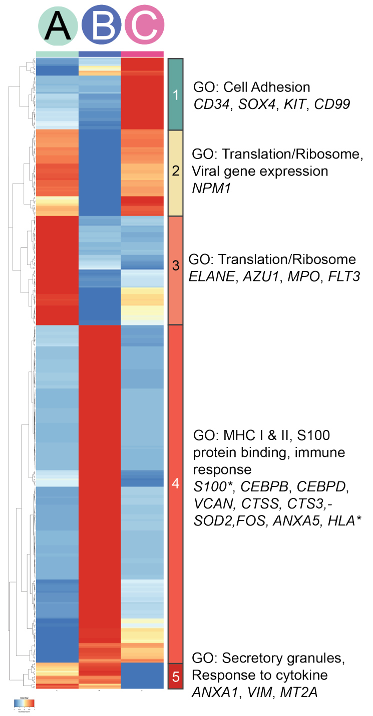
Characterizing clusters using marker genes

Cells colored by graph-based cluster



Characterizing clusters using differential gene expression



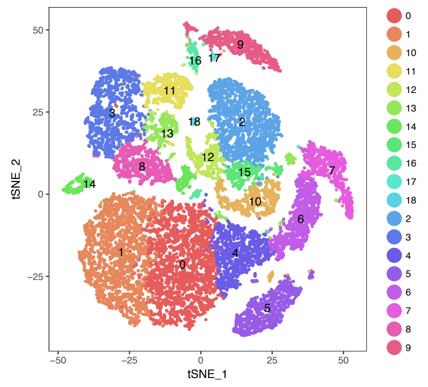


- Data has zero-inflated negative binomial distribution (lots of zeros, overdispersed) so can't use bulk methods
- Default in Seurat: Wilcoxon rank-sum test
- Nonparametric version of t-test
- For two clusters (A and B), and one gene, rank each cell in each cluster according to expression
- Determine whether sum-of-ranks for cluster A is significantly different than sum-of-ranks for cluster B
- Clear explanation of Wilcoxon rank-sum test: http://statweb.stanford.edu/~susan/courses/s141/hononpara.pdf
- Numerous other tests in Seurat and other packages

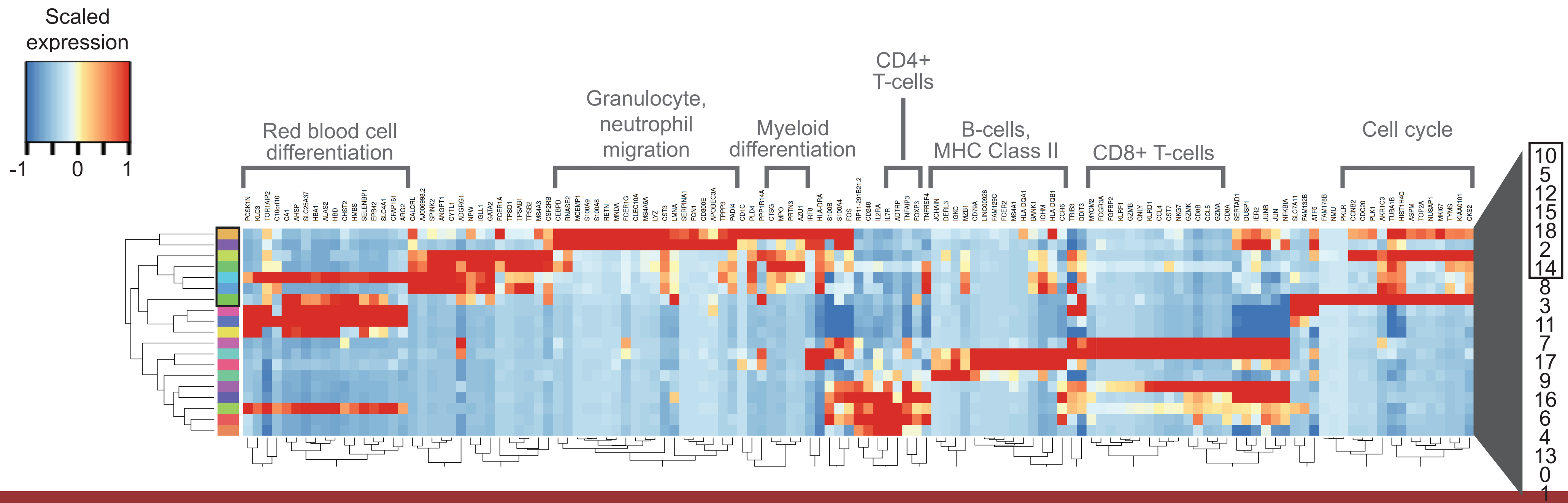
Characterizing clusters using differential gene expression

```
DEGs <- FindAllMarkers(object=scrna); # Compare each clusters to all other cells. output is a matrix.
de.markers <- FindMarkers(scrna, ident.1 = ″1", ident.2 = ″2") # compare identity 1 to identity 2
# NB: May first need to set default identities, e.g.:
Idents(object = scrna) <- ″seurat_clusters″; # sets the default identity to Seurat_clusters

# Do the DEGs make sense? Plot them
FeaturePlot(object = scrna, features = ‘CD34’)
# Prettier version:
FeaturePlot(object = scrna, features = genesToPlot, cols = c("gray","red"), ncol=2, reduction = "umap") + theme(axis.title.x=element_blank(),axis.title.y=element_blank(),axis.text.x=element_blank(),axis.text.y=element_blank(),axis.ticks.x=element_blank(),axis.ticks.y=element_blank())
```

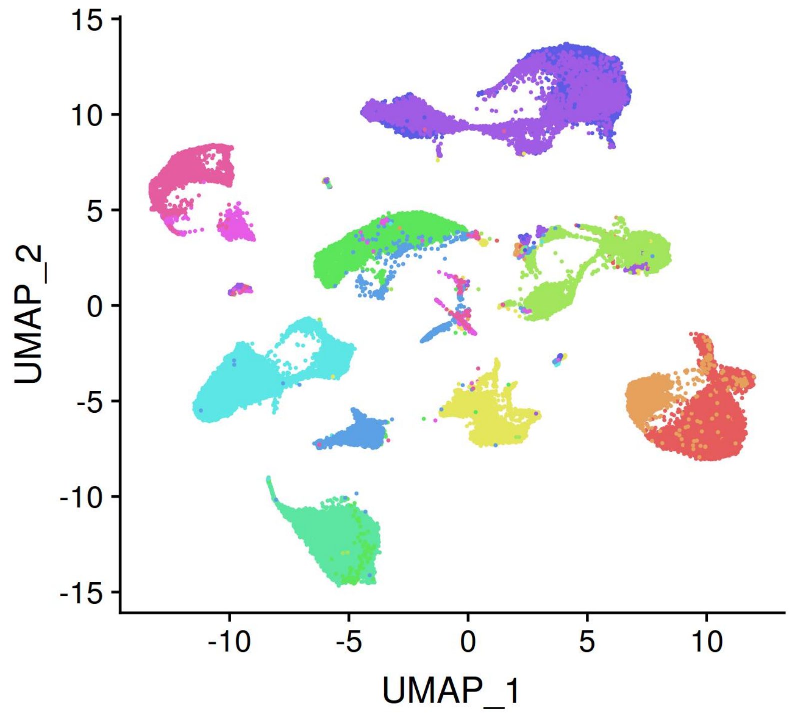
Characterizing clusters using principle components

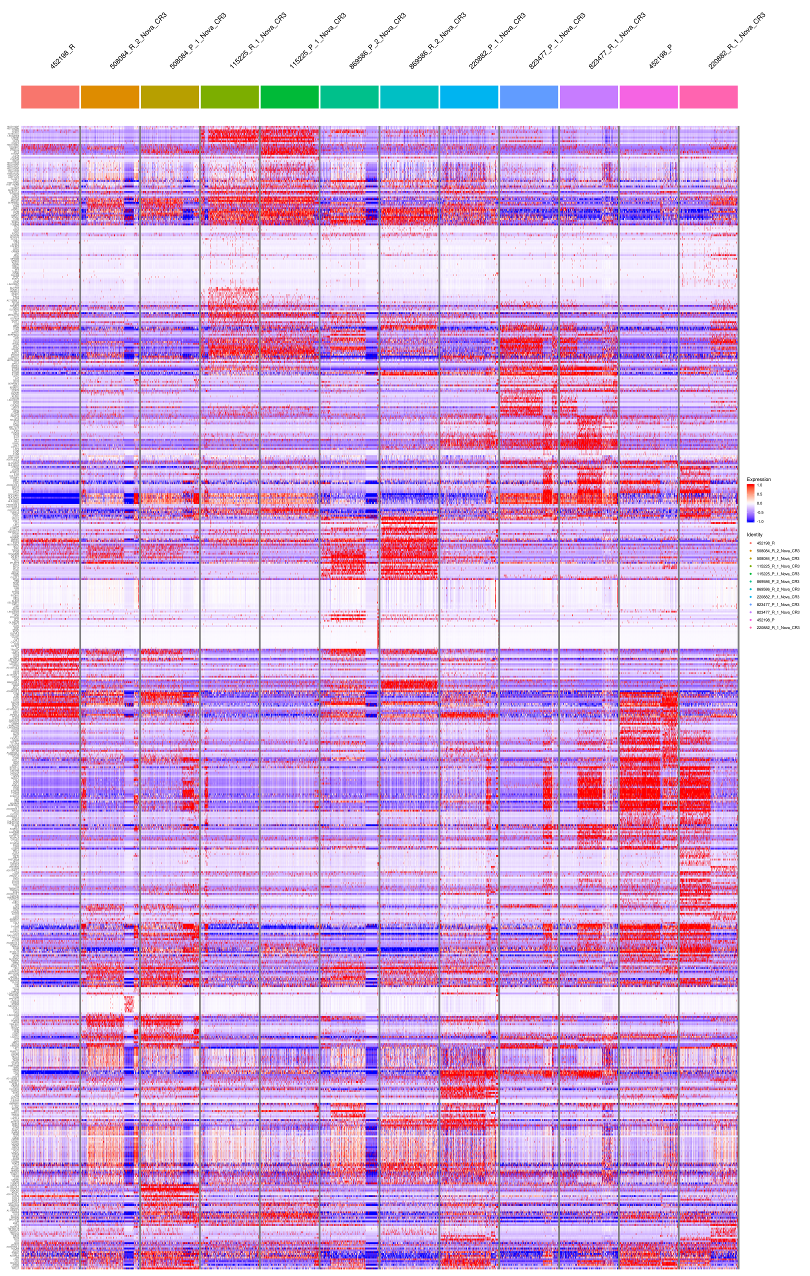


- Cluster-specific DEGs are problematic: require “perfect” clusters, oversimplify, can miss signal, too stringent for the data.
- Instead, visualize principle components across clusters to view relationships among clusters
- Choose most highly-weighted genes in the top principal components
- Calculate average expression of each gene in each cluster
- Use hierarchical clustering to cluster the clusters based on these genes



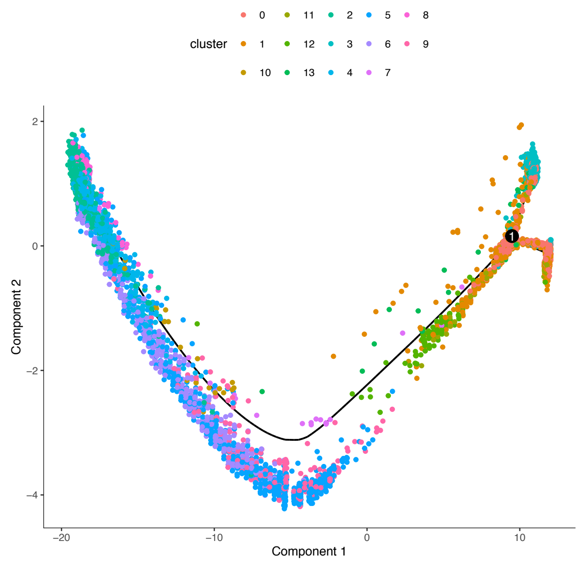
Characterizing samples using principle components

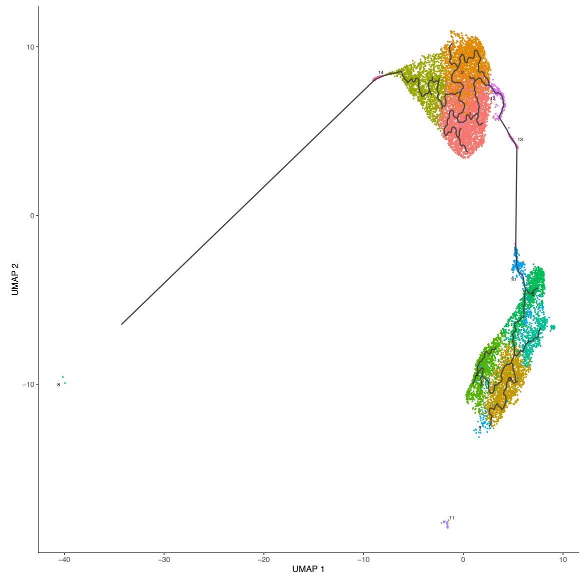




Characterizing gene expression changes with respect to pseudotime

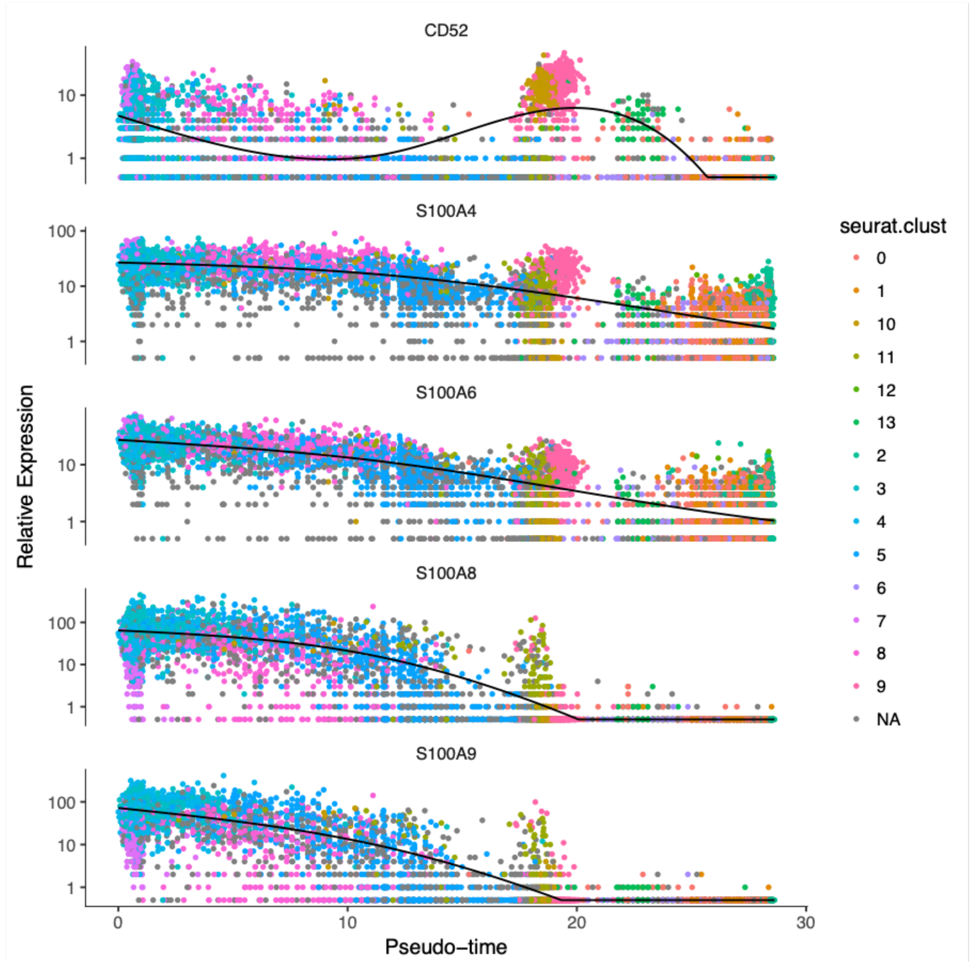
(Monocle2 vs Monocle3, Slingshot)

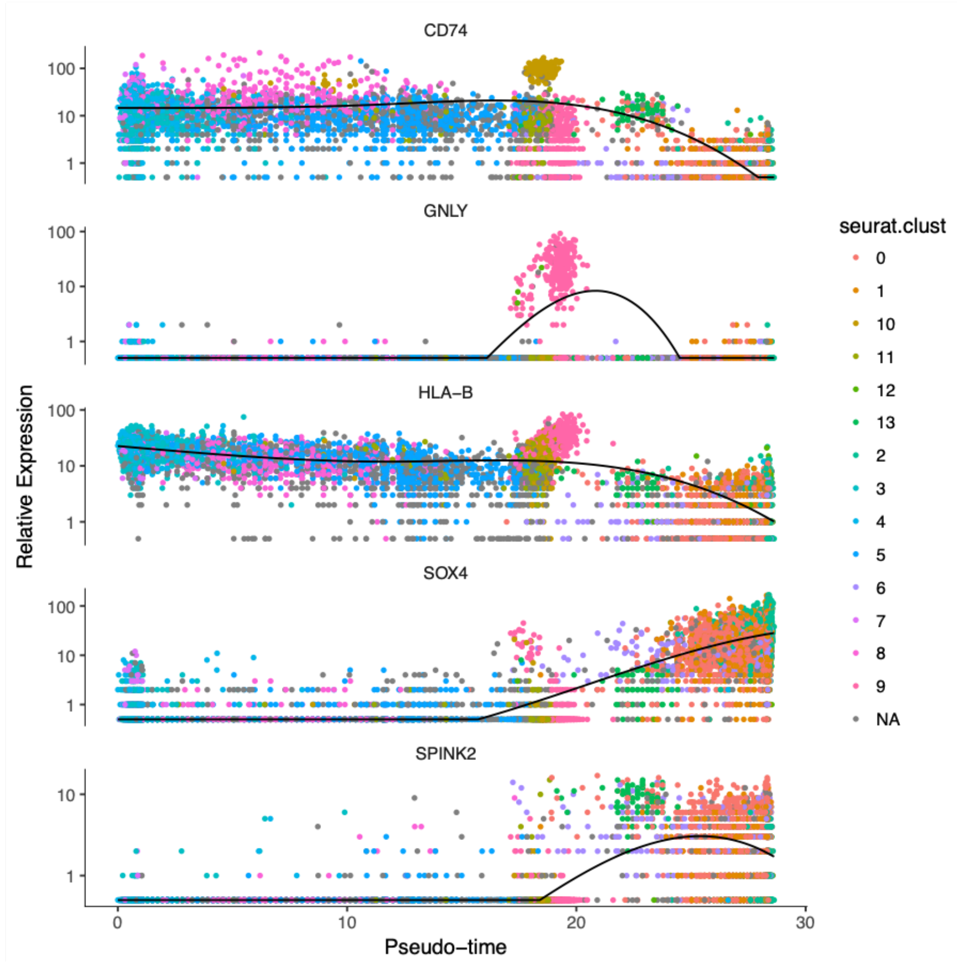




Characterizing gene expression changes with respect to pseudotime

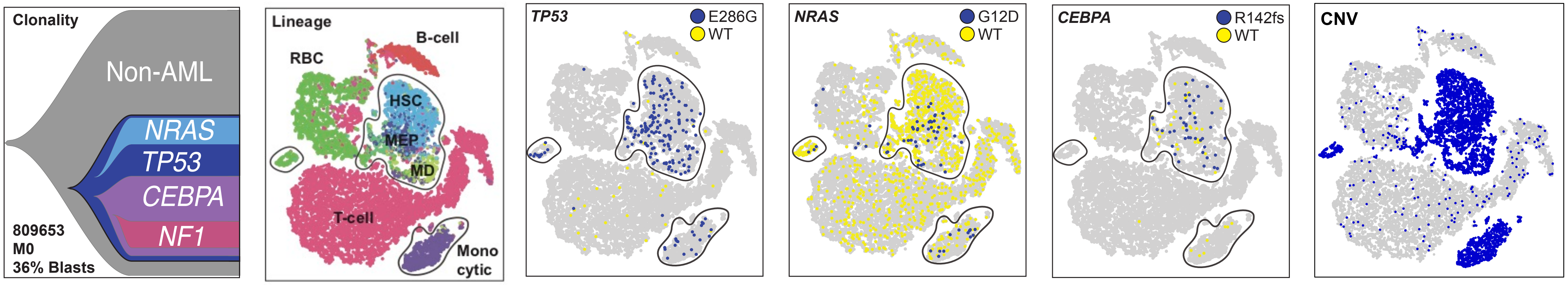
(Monocle)





Mutation detection in scRNA-seq data

- scRNAseqMutations clarify scRNA-seq data interpretation by marking tumor cell clusters



Founding clone
=> AML clusters*

Independent subclones, overlapping signatures

CNVs mark AML clusters

- vartrix
- cb_sniffer
- CONICSmat (CNV only)
- HoneyBADGER (CNV, LOH)

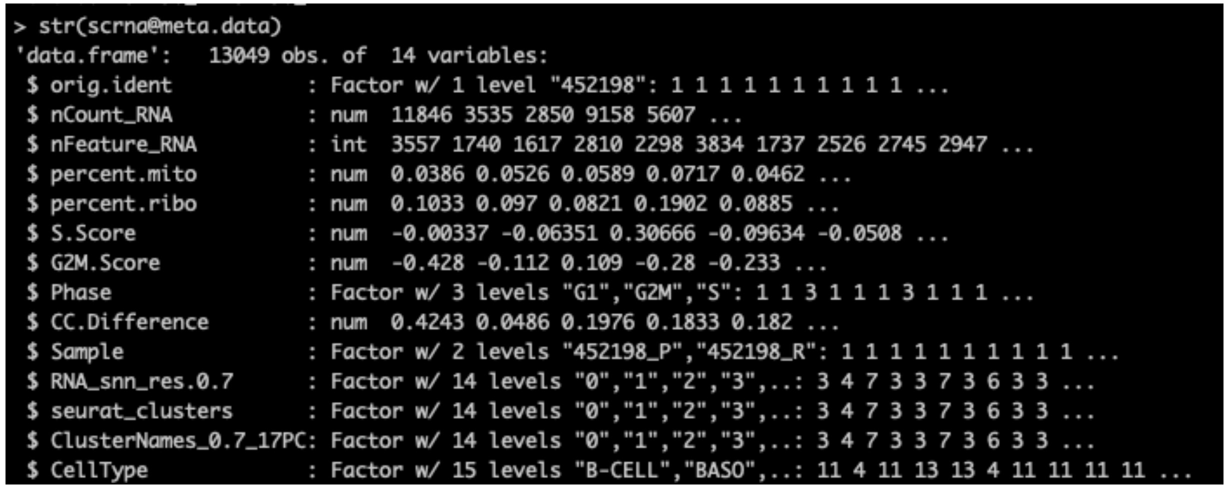
*AML clusters: significantly enriched for somatic mutations (Fisher exact test)

Return to the exercise to complete steps 12-14

- https://rnabio.org/module-08-scrna/0008/02/01/scRNA/

Part III:
Useful Seurat functions for future reference

Seurat object: meta data



```
> str(scrna@meta.data)
'data.frame':   13049 obs. of  14 variables:
 $ orig.ident           : Factor w/ 1 level "452198": 1 1 1 1 1 1 1 1 1 1 ...
 $ nCount_RNA           : num  11846 3535 2850 9158 5607 ...
 $ nFeature_RNA         : int  3557 1740 1617 2810 2298 3834 1737 2526 2745 2947 ...
 $ percent.mito         : num  0.0386 0.0526 0.0589 0.0717 0.0462 ...
 $ percent.ribo         : num  0.1033 0.097 0.0821 0.1902 0.0885 ...
 $ S.Score              : num  -0.00337 -0.06351 0.30666 -0.09634 -0.0508 ...
 $ G2M.Score            : num  -0.428 -0.112 0.109 -0.28 -0.233 ...
 $ Phase                : Factor w/ 3 levels "G1","G2M","S": 1 1 3 1 1 1 3 1 1 1 ...
 $ CC.Difference        : num  0.4243 0.0486 0.1976 0.1833 0.182 ...
 $ Sample               : Factor w/ 2 levels "452198_P","452198_R": 1 1 1 1 1 1 1 1 1 1 ...
 $ RNA_snn_res.0.7      : Factor w/ 14 levels "0","1","2","3",..: 3 4 7 3 3 7 3 6 3 3 ...
 $ seurat_clusters      : Factor w/ 14 levels "0","1","2","3",..: 3 4 7 3 3 7 3 6 3 3 ...
 $ ClusterNames_0.7_17PC: Factor w/ 14 levels "0","1","2","3",..: 3 4 7 3 3 7 3 6 3 3 ...
 $ CellType             : Factor w/ 15 levels "B-CELL","BASO",..: 11 4 11 13 13 4 11 11 11 11 ...
```

Meta data (scrna@meta.data) contains:
- summary statistics
- sample name
- cluster membership for each cell
- cell cycle phase for each cell
- batch or sample for each cell
- other custom labels for each cell

Access using:
```
scrna[[]]
scrna@meta.data
str(scrna@meta.data)
```
Combine with R commands such as head and str, e.g. str(scrna[[]])

Example: Access number of genes (“Features”) for each cell:
```
head(scrna@meta.data$nFeature_RNA)
```
Example: Access number of UMIs for each cell:
```
head(scrna@meta.data$nCount_RNA)
```
What are the items in the current default cell identity class?
```
levels(x=scrna)
```
How many clusters are there?
```
length(unique(scrna@meta.data$seurat_clusters))
levels(x=scrna)
```
What batches are included in this data set?
```
unique(scrna@meta.data$Batch)
```
Can add meta data

Seurat object: Assay data and Dimensionality reduction

- Assay data (i.e. RNA-seq measurements): e.g. ‘RNA’
  - Assay = data type
  - Each assay has multiple “slots” that hold the raw data (‘counts’), scaled data (‘scale.data’) or normalized data (‘data)
  - Data transformation, such as scaling and normalization, adds new ‘slots’ to the assay data
  - Access values in the slots as follows:
    - raw RNA counts: `scrna[['RNA']]@counts[1:3,1:3]`
    - scaled data after SCTransform: `scrna[['SCT']]@scale.data[1:3,1:3]`
    - corrected UMI count data after SCTransform: `scrna[['SCT']]@counts[1:3,1:3]`
    - log-normalized data after SCTransform: scrna[['SCT']]@data[1:3,1:3]
  - Alternatively, use the function GetAssayData:
    - `GetAssayData(object = scrna, slot = 'scale.data')[1:3, 1:3]`
    - `GetAssayData(object = scrna, slot = 'counts')[1:3, 1:3]`
  - It’s often important to know what ‘slot’ a function is using. Sometimes you can change it.
- Dimensionality reduction data, e.g. PCA, tSNE, UMAP data
  - Access using scrna[[‘pca’]], scrna[[‘tsne’]], scrna[[‘umap’]]

Summary of common functions for the Seurat object

```
GetAssayData(object = scrna, slot = "counts")
GetAssayData(object = scrna, slot = "scale.data")
FetchData(object = scrna) # returns a data frame
colnames(x = scrna)
rownames(x = scrna)
VariableFeatures(object = scrna)
HVFInfo(object = scrna)
scrna[["assay.name"]] eg scrna[[“RNA”]]
scrna[["pca"]]
Embeddings(object = scrna, reduction = "pca")
Loadings(object = scrna, reduction = "pca")
scrna$name <- vector # assign a vector of identities
scrna$name # e.g. scrna$seurat_clusters
Idents(object = scrna) # get default cell identities
Idents(object = scrna) <- "new.idents” # change the identities
Idents(object = scrna, cells = 1:10) <- "new.idents” # change the identities for specific cells
scrna$saved.idents <- Idents(object = scrna) # change current identities for an existing identity class
levels(x = scrna) # get a list of the default identities (e.g. a list of clusters)
RenameIdents(object = scrna, "old.ident" = "new.ident") # change name of identity classes
WhichCells(object = scrna, idents = "ident.keep") # which cells are in cluster x?
WhichCells(object = scrna, idents = "ident.remove", invert = TRUE)
WhichCells(object = scrna, downsample = 500)
WhichCells(object = scrna, expression = name > low & name < high)
subset(x = scrna, cells = cellist, idents=‘keep’); # subset seurat object and return seurat object
subset(x = scrna, subset = name > low & name < high)
merge(x = object1, y = object2)
```

Additional ways to access data

```
Cells(scrna) # get list of cells
# choose cells that have a given characteristic, here, that are in “sample1”:
subcells <- Cells(scrna)[(which(scrna[["Sample"]]$Sample == sample1))]
# alternative approach, choosing cells in cluster 4:
Idents(object=scrna) <- "ClusterNames” # set default identity to ClusterNames
WhichCells(scrna,idents="4") # use WhichCells
# extract raw expression matrix for all cells in cluster 4
as.matrix(GetAssayData(scrna, slot = "counts")[, WhichCells(scrna, ident = '4')])
FetchData # subset Seurat object and return a data frame
subset # subset Seurat object and return a Seurat object
levels(scrna) # reorder the columns in Do.Heatmap
Command(scrna) # lists the functions that were applied to the Seurat object
```

We are on a Coffee Break & Networking Session

Workshop Sponsors:

compute canada | calcul canada

OICR Ontario Institute for Cancer Research

Canadian Centre for Computational Genomics

MiCM McGill initiative in Computational Medicine