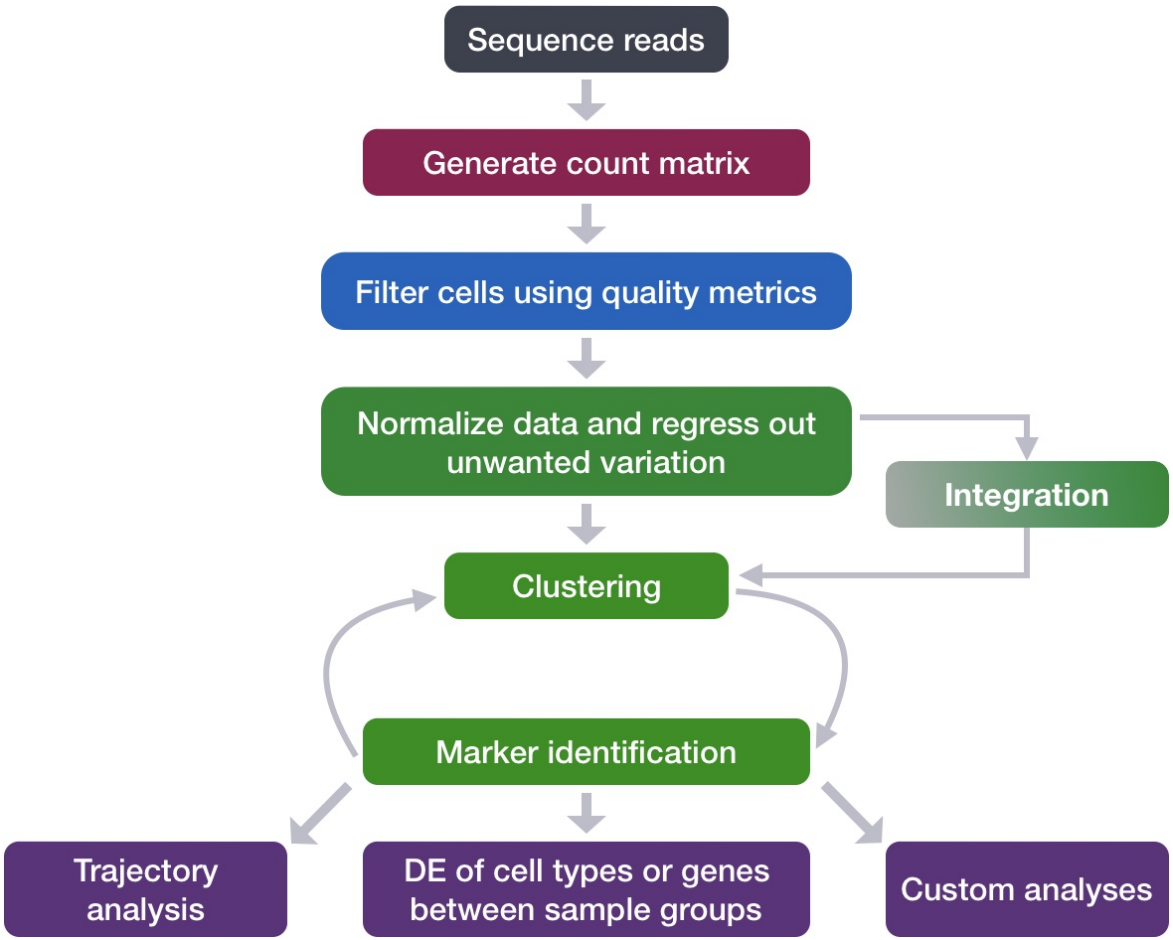
STAT718/BIOL793: Genomic Data Science

Single-cell RNA-seq Normalization and Clustering

Yen-Yi Ho (hoyen@stat.sc.edu)

scRNAseq Workflow



Normalization

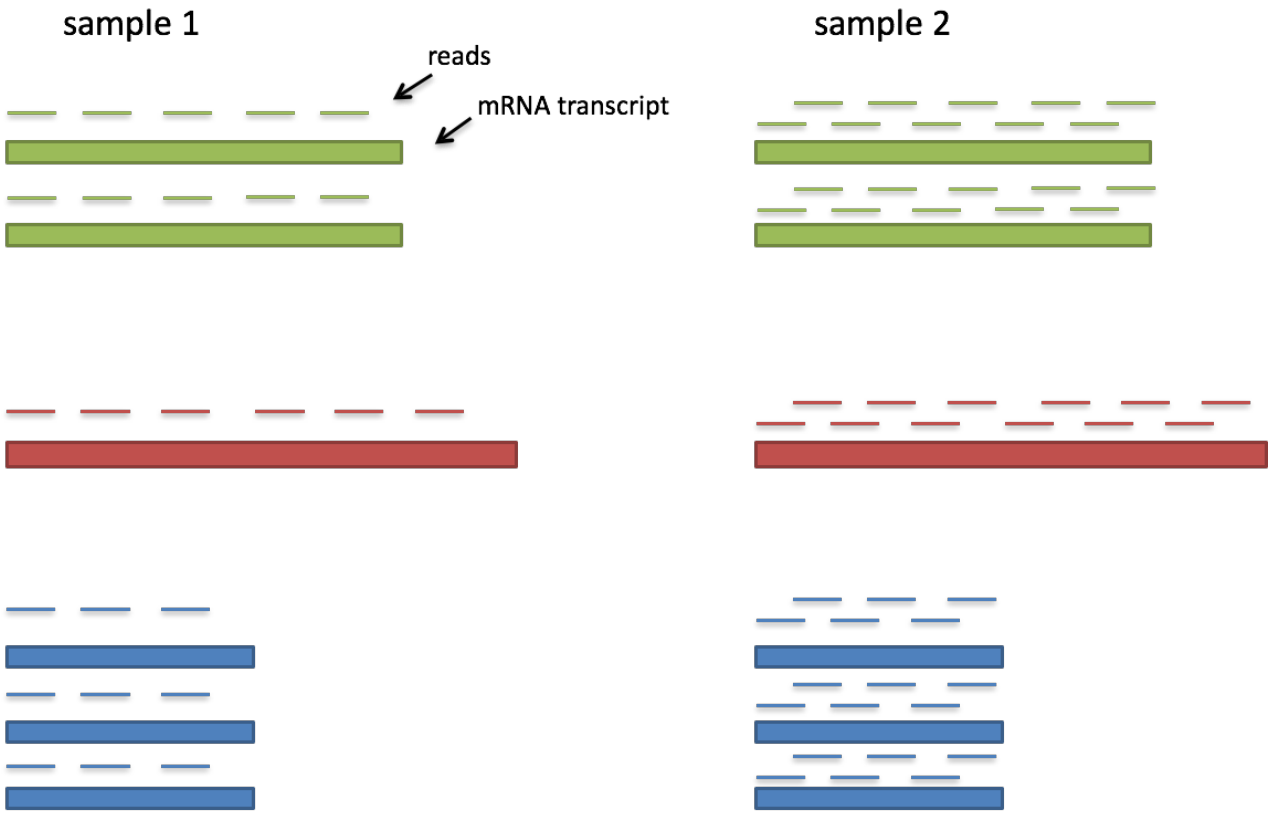
The goal of normalization is to

- make expression counts comparable across genes and/or samples.
- help identify the most variant genes likely to be indicative of the different cell types present.

scRNAseq Normalization

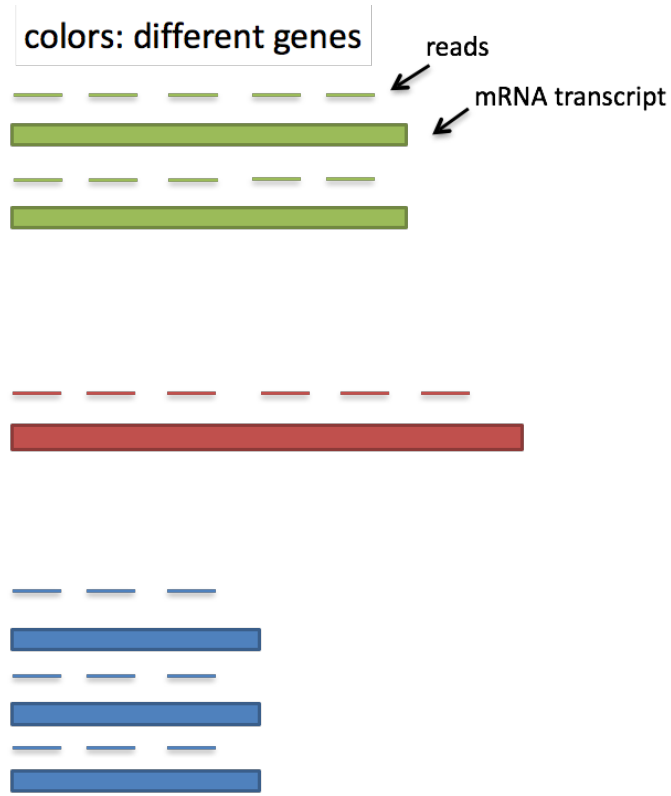
- Sequencing Depth
- Gene Length
- Mitochondria Ratio

Sequencing Depth



MI Love: RNA-seq statistical analysis

Gene Length



Slide adapted from MI Love: RNA-seq statistical analysis

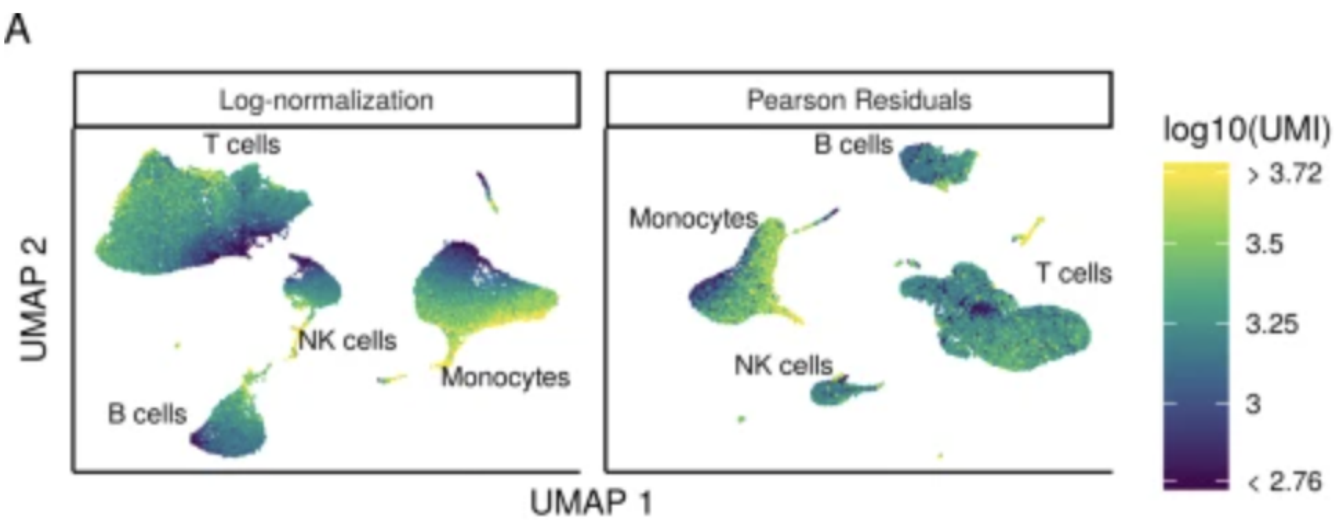
Normalization Approach

- Scaling: multiply each UMI count by a cell specific factor to get all cells to have the same UMI counts
- Transformation
  - Simple: log transformation
  - Complex: apply correction on a per-gene basis

Normalization

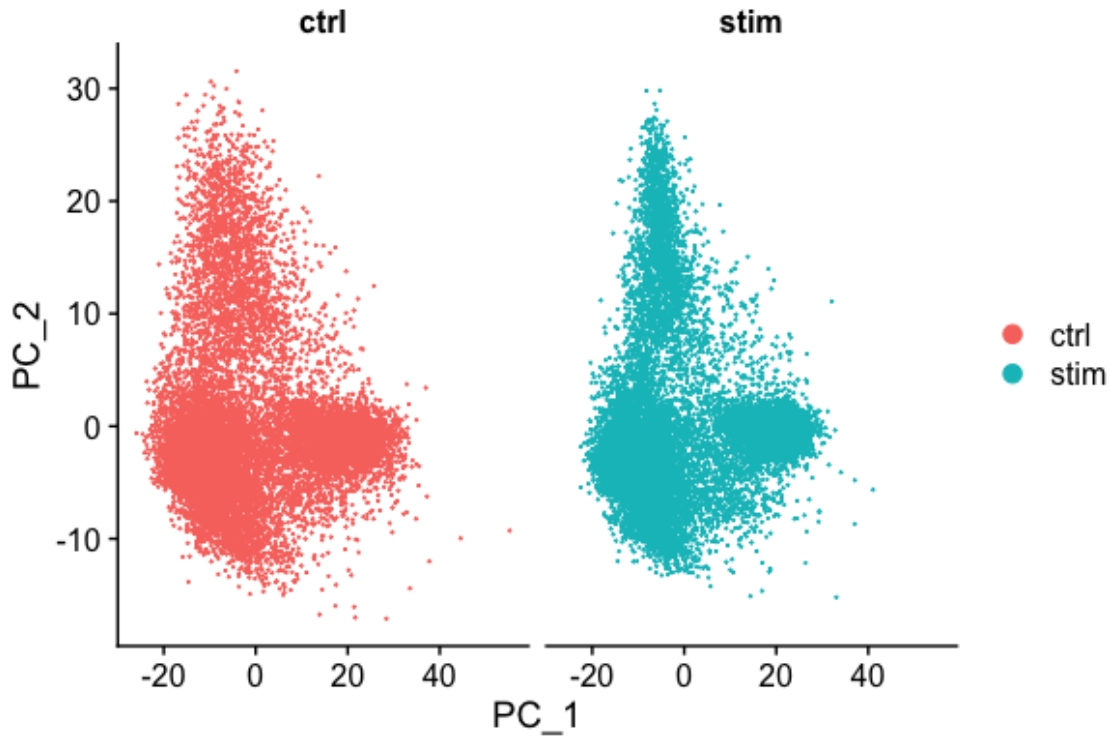
SCTransform method constructs a generalized linear model (GLM) for each gene with UMI counts as the response and sequencing depth as the explanatory variable. The R code for performing SCTransform can be found on SC3.R on the course website.

Fig. 6

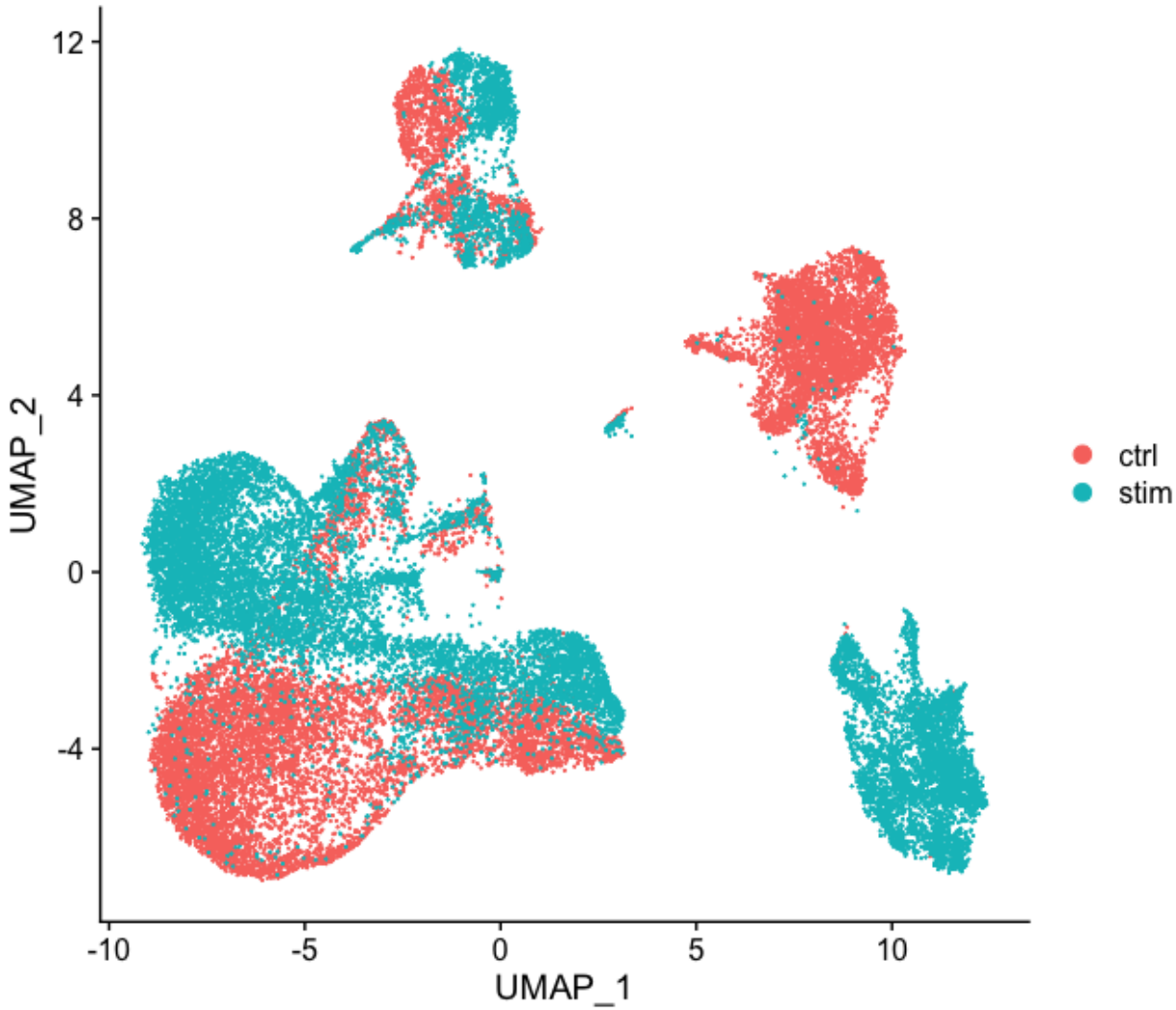


UMAP

A popular dimensionality reduction technique for scRNAseq is called **Uniform Manifold Approximation and Projection (UMAP)**. UMAP takes the information from any number of top PCs to arrange the cells in this multidimensional space. It will take those distances in multidimensional space and plot them in two dimensions working to preserve local and global structure.

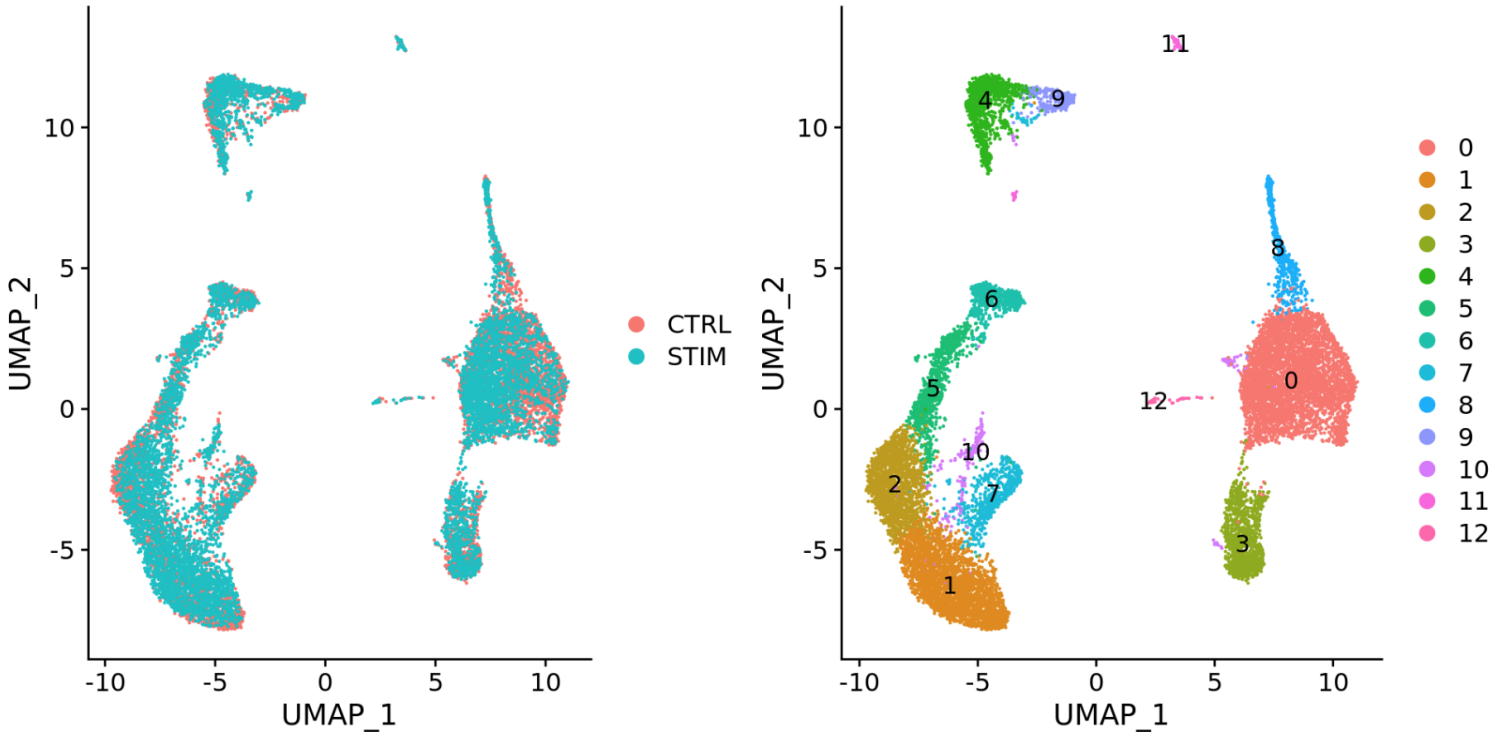


To integrate or not to integrate?

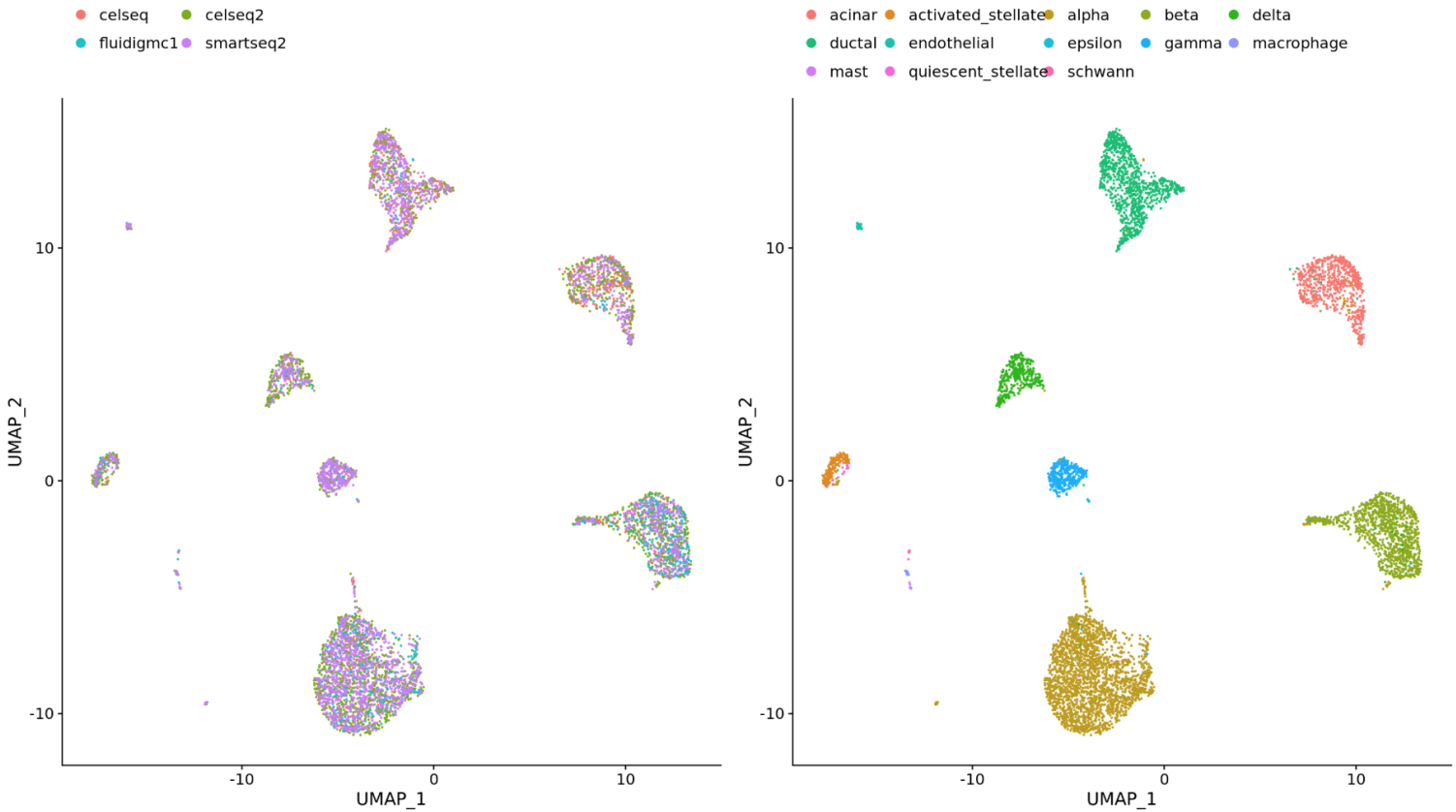


Integrate samples across conditions

Often `integration" orharmonization"` is performed during the analysis to identify a "common set of biological features" between groups.

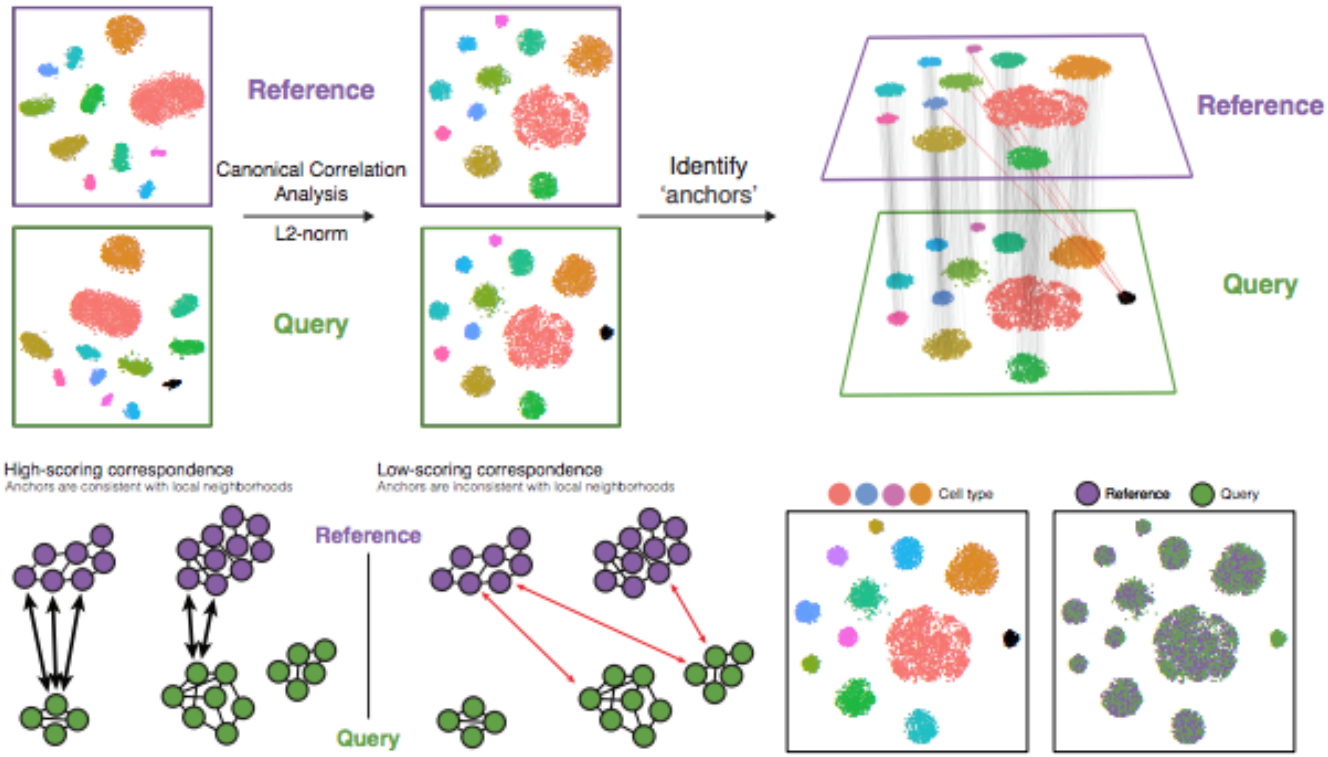


Integrate samples across datasets or batches

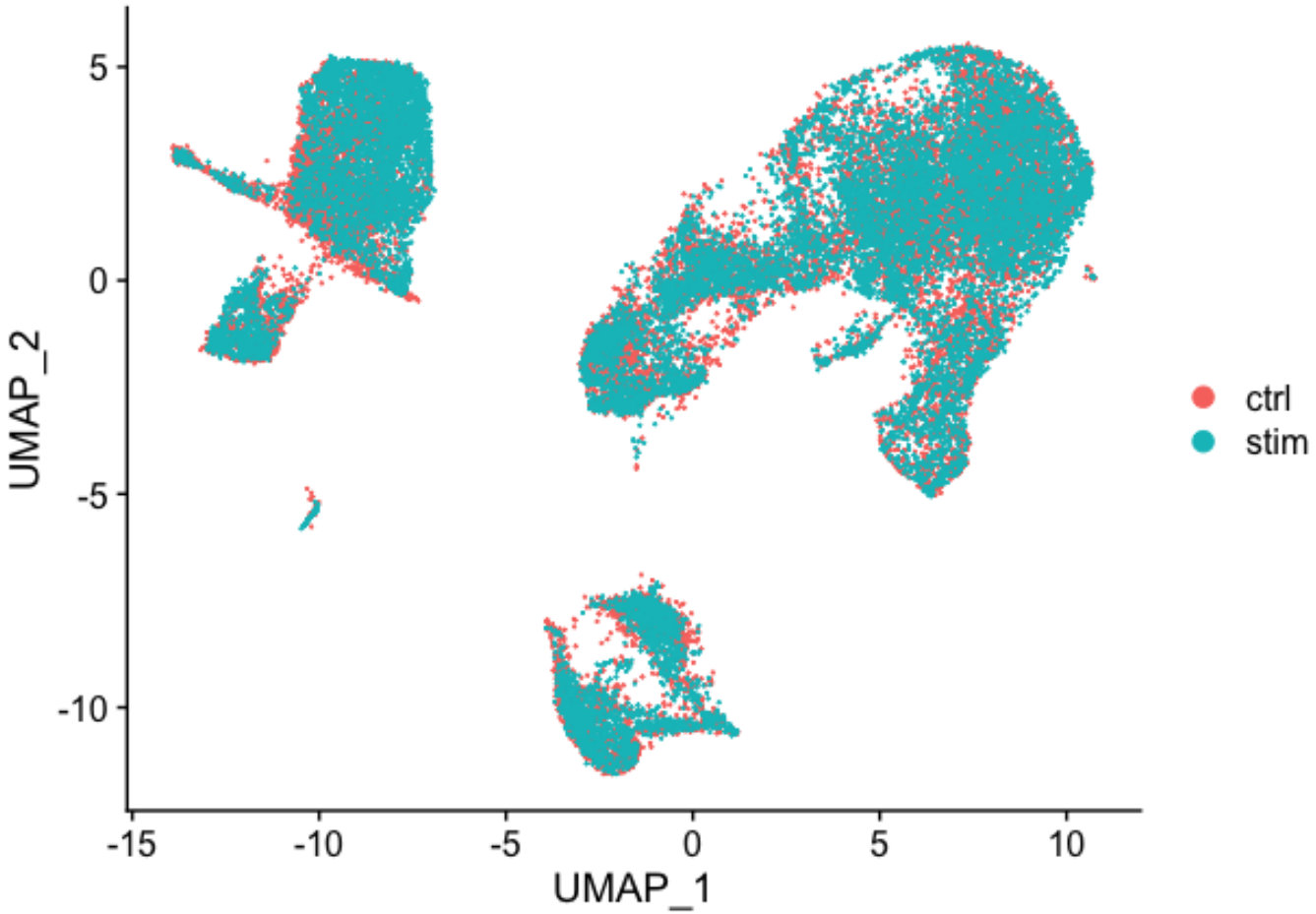


CCA

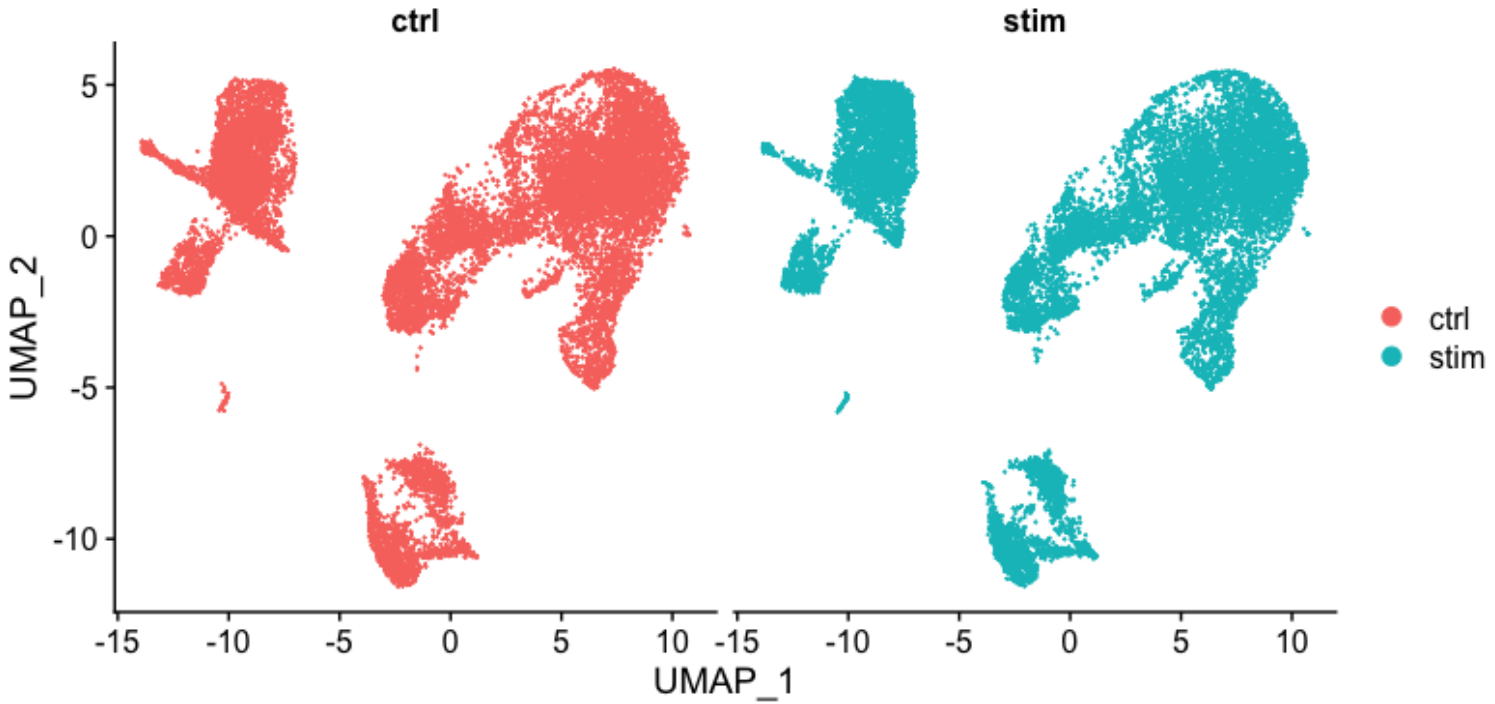
The integration method in the pacakge Seurat uses the canonical correlation analysis (CCA). This method expects "correspondences" or shared biological states among at least a subset of single cells across the groups. The steps in the Seurat integration workflow are outlined in the figure below:



UMAP Visualization



Side-by-side comparison of clusters



The other popular algorithm alternative to the Seurat integration workflow is called Harmony.

Clustering

Goals:

- To **generate cell type-specific clusters** and use known cell type marker genes to determine the identities of the clusters.
- To **determine whether clusters represent true cell types or cluster due to biological or technical variation**, such as clusters of cells in the S phase of the cell cycle, clusters of specific batches, or cells with high mitochondrial content.

Challenges

- **Identifying poor quality clusters** that may be due to uninteresting biological or technical variation
- **Identifying the cell types** of each cluster
- Maintaining patience as this can be a highly iterative process between clustering and marker identification (sometimes even going back to the QC filtering)

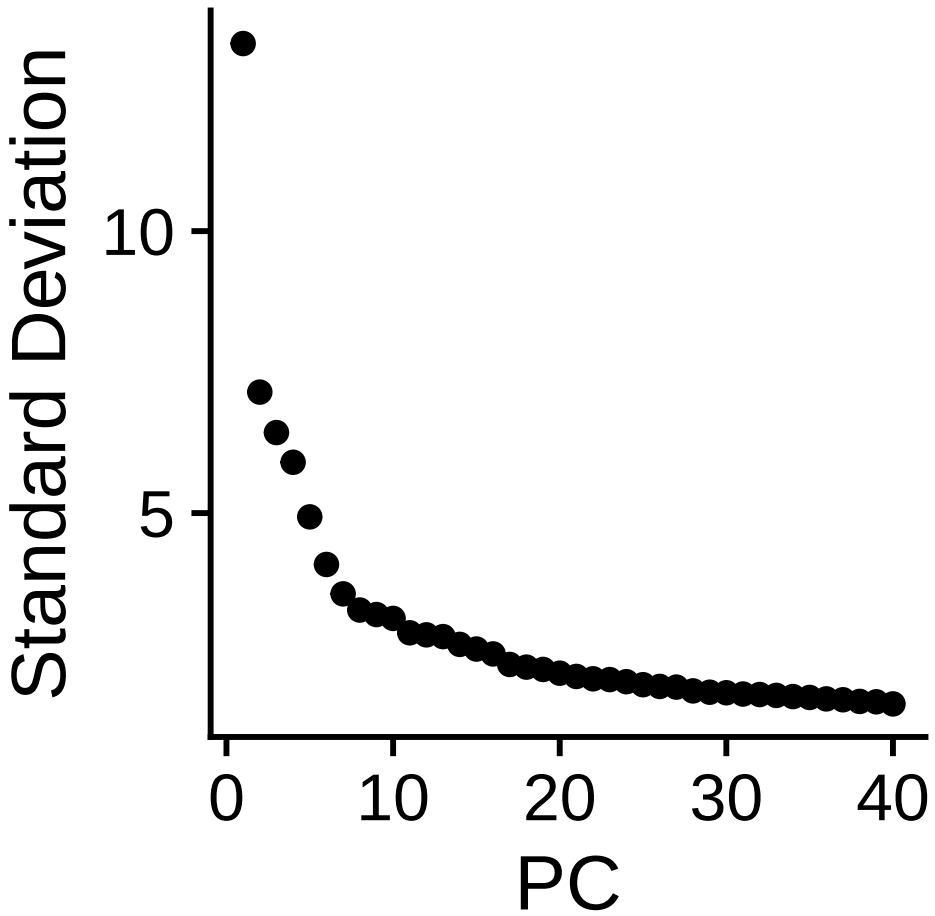
Recommendations:

- Have a good idea of your expectations for the **cell types to be present** prior to performing the clustering. Know whether you expect cell types of low complexity or higher mitochondrial content AND whether the cells are differentiating
- If you have **more than one condition**, it's often helpful to perform integration to align the cells
- **Regress out** number of UMIs (by default with sctransform), mitochondrial content, and cell cycle, if needed and appropriate for experiment, so not to drive clustering
- Identify any junk clusters for removal or re-visit QC filtering. Possible junk clusters could include those with high **mitochondrial content** and low UMIs/genes. If comprised of a lot of cells, then may be helpful to go back to QC to filter out, then re-integrate/cluster.
- **not detecting all cell types as separate clusters**, try changing the resolution or the number of PCs used for clustering

Clustering cells based on top PCs

```
## PC_ 1
## Positive:  FTL, TIMP1, FTH1, C15orf48, CXCL8
## Negative:  RPL3, RPL13, RPS6, RPS18, RPL10
## PC_ 2
## Positive:  GNLY, CCL5, NKG7, GZMB, FGFBP2
## Negative:  CD74, IGHM, IGKC, HLA-DRA, CD79A
## PC_ 3
## Positive:  CD74, IGKC, HLA-DRA, IGHM, HLA-DRB1
## Negative:  TRAC, FTL, CCL2, PABPC1, S100A8
## PC_ 4
## Positive:  CD74, IGHM, CCL5, GNLY, IGKC
## Negative:  HSPB1, CACYBP, HSPH1, HSP90AB1, HSPA8
## PC_ 5
## Positive:  VMO1, FCGR3A, MS4A7, TIMP1, TNFSF10
## Negative:  CCL2, FTL, CXCL8, S100A8, S100A9
## PC_ 6
## Positive:  IGHM, IGKC, CD79A, CCL2, MS4A1
## Negative:  HLA-DQA1, TXN, HLA-DRA, HLA-DPA1, LYZ
## PC_ 7
## Positive:  TIMP1, S100A8, LYZ, IGHM, MARCKSL1
## Negative:  CCL2, CCL3, CCL4, CCL4L2, MIR155HG
## PC_ 8
## Positive:  CCL3, CCL4, CXCL8, IL1B, CCL4L2
## Negative:  CCL2, LGALS3, FTL, ISG15, CTSL
## PC_ 9
## Positive:  MIR155HG, NME1, HERPUD1, FTH1, DUSP4
```

The number of PCs for clustering



Find clusters

Seurat will iteratively group cells together with the goal of optimizing the standard modularity function

```
# Determine the clusters for various resolutions
seurat_integrated <- FindClusters(object = seurat_integrated,
                                  resolution = c(0.4, 0.6, 0.8, 1.0, 1.4))
```

```
## Modularity Optimizer version 1.3.0 by Ludo Waltman and Nees Jan van Eck
##
## Number of nodes: 29629
## Number of edges: 1113933
##
## Running Louvain algorithm...
## Maximum modularity in 10 random starts: 0.9206
## Number of communities: 13
## Elapsed time: 4 seconds
## Modularity Optimizer version 1.3.0 by Ludo Waltman and Nees Jan van Eck
##
## Number of nodes: 29629
## Number of edges: 1113933
##
## Running Louvain algorithm...
## Maximum modularity in 10 random starts: 0.9016
## Number of communities: 15
## Elapsed time: 4 seconds
## Modularity Optimizer version 1.3.0 by Ludo Waltman and Nees Jan van Eck
##
```

Explore resolutions

```
# Assign identity of clusters
Idents(object = seurat_integrated) <- "integrated_snn_res.0.8"
# Plot the UMAP
DimPlot(seurat_integrated,
        reduction = "umap",
        label = TRUE,
        label.size = 1)
```