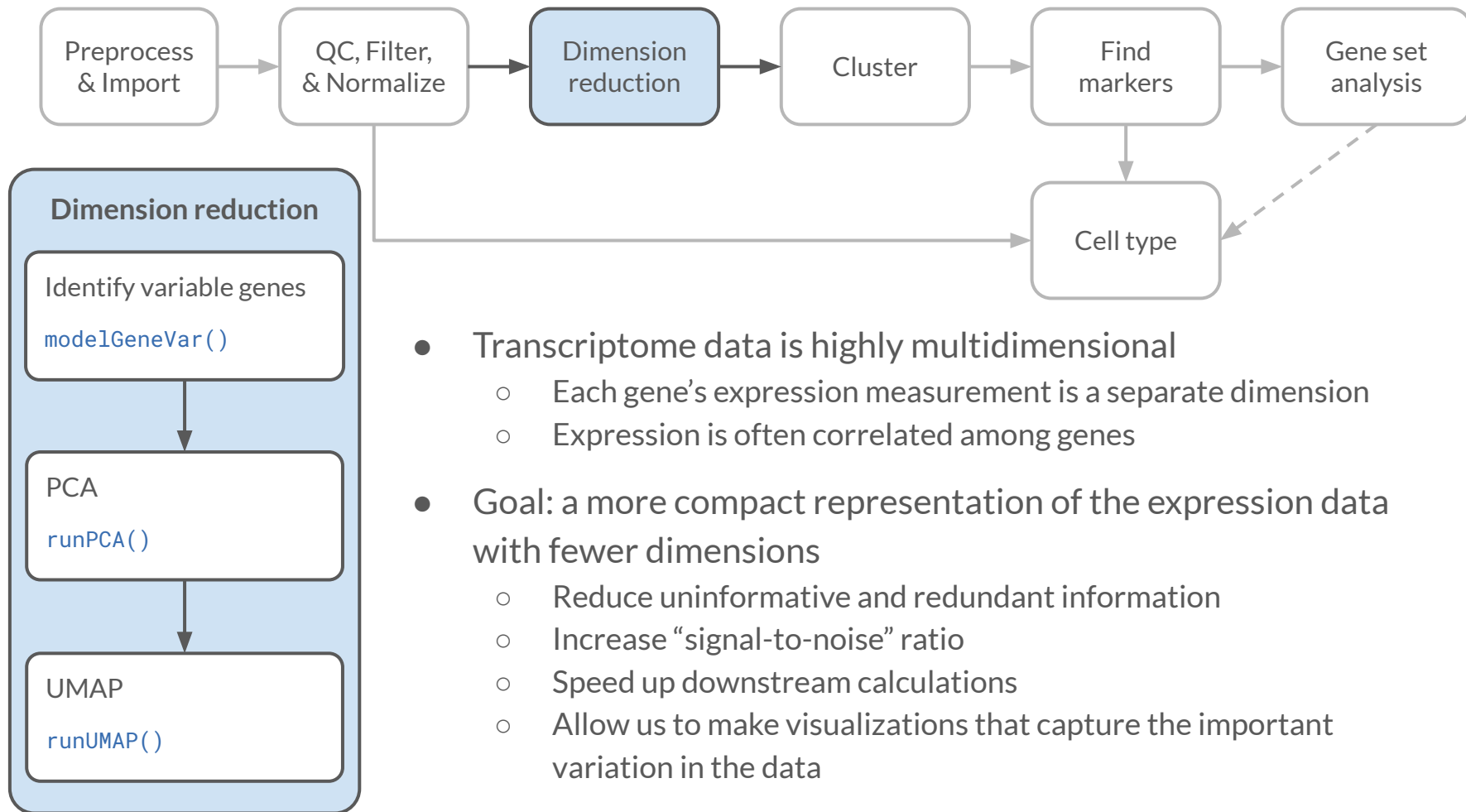




Dimensionality Reduction and Clustering of Single-cell Data

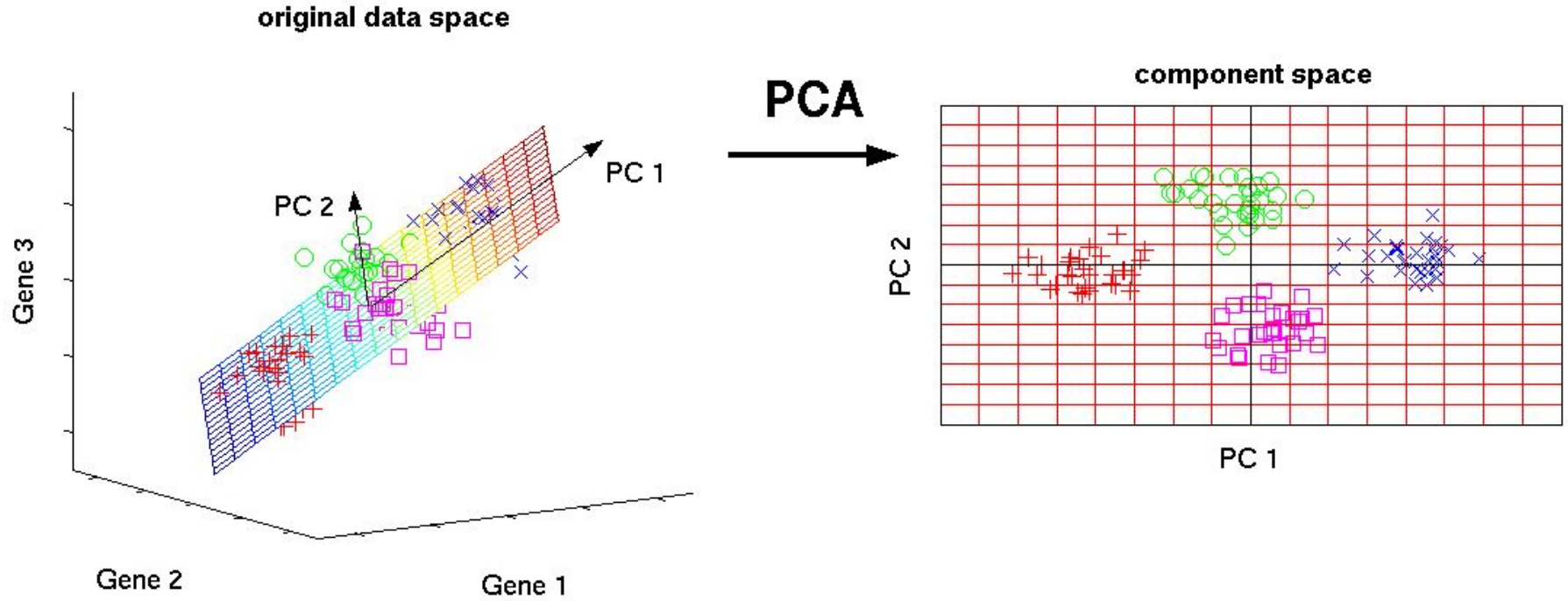
The Data Lab

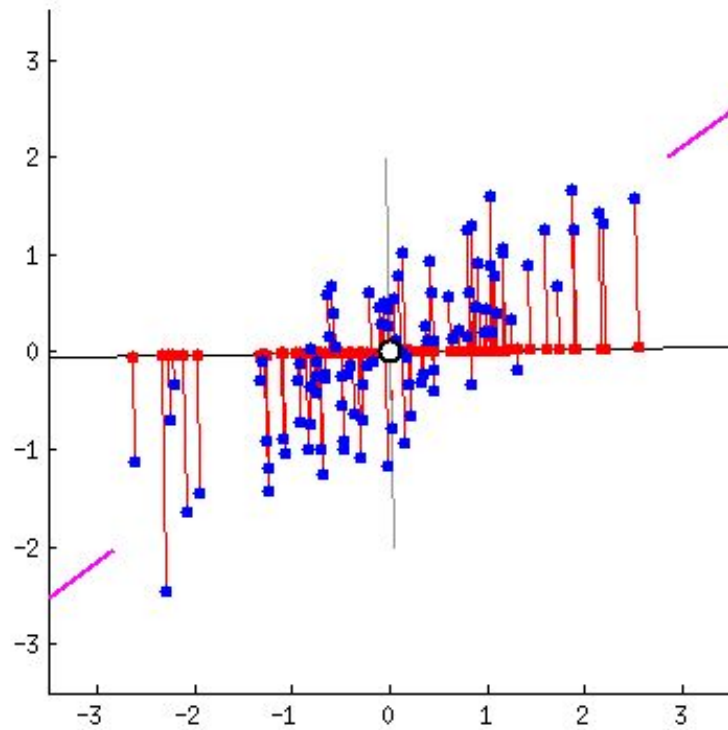


Dimensionality Reduction Methods

- Feature selection
 - Select the most (biologically) variable genes
- Principal Components Analysis
 - linear transformation of input data
 - usually to tens of dimensions
 - removes much of the noise; retains most of the signal
 - useful as input to many downstream analyses (clustering, etc.)
- UMAP and/or tSNE
 - reduce down to 2 or 3 dimensions
 - transformation is highly non-linear
 - much slower than PCA

Principal Components Analysis (PCA)





<https://builtin.com/data-science/step-step-explanation-principal-component-analysis>

Assumptions/Limitations of PCA

- PCA is a linear transformation of the input data
 - Fast!
 - Reversible if we keep all dimensions
 - Usually we don't keep everything... removing higher dimensions reduces effects of noise
- Assumes \sim normal distributions for error
 - For scRNA-seq count data, this can be approximated with log-scale normalization
- Sensitive to outliers



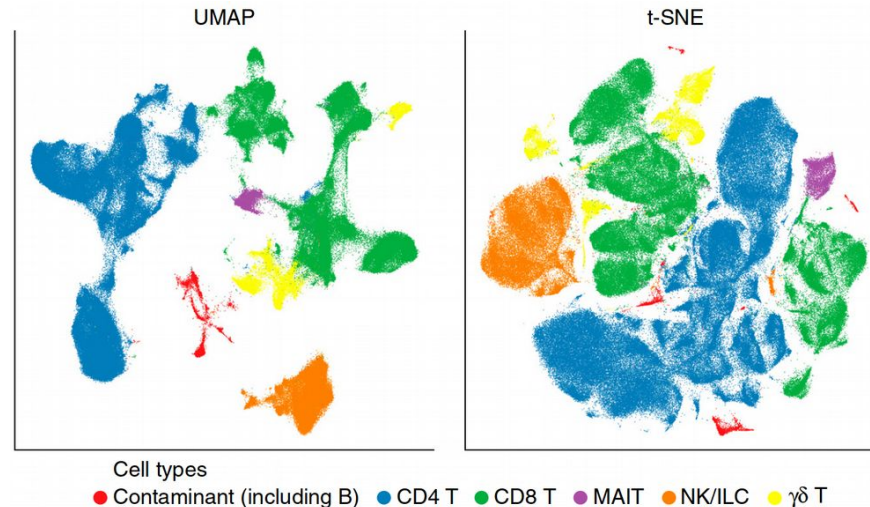
UMAP and tSNE

Machine learning methods for dimensionality reduction

Details are beyond the scope of this course, but the basic steps are these:

- Calculate the similarity between pairs of data points
- Find a representation in low dimensionality space (mapping) that recapitulates the similarity matrix
 - How? Start with a mapping then progressively update it by how well the distances in the low dimension space match the original distances

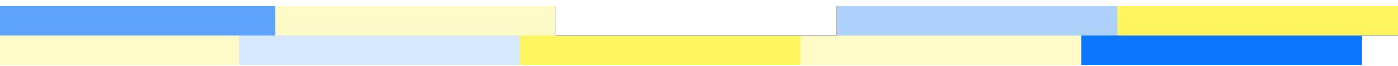
A nice visualization/playground for tSNE: <https://distill.pub/2016/misread-tsne/>



<https://doi.org/10.1038/nbt.4314>

Assumptions/Limitations of UMAP & tSNE

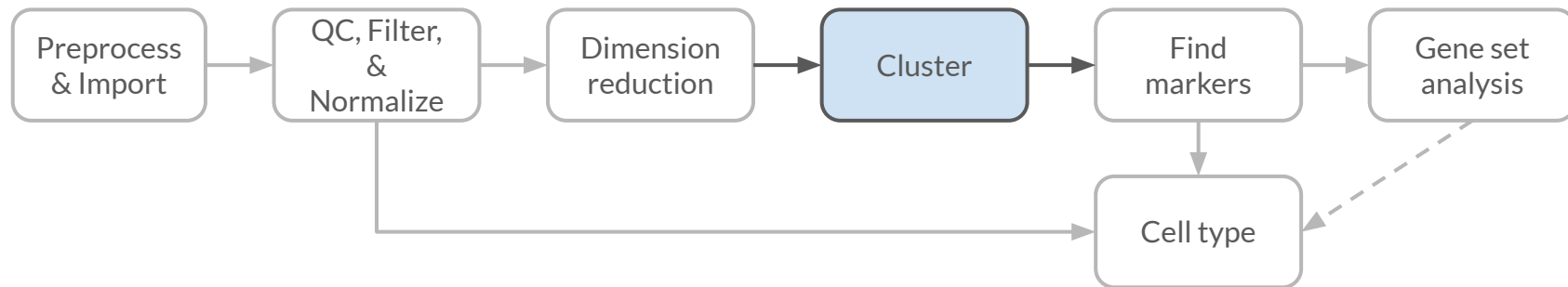
- No assumptions about shape of data
 - Performs better when structures may not have “normal” distributions
- Tends to produce more visually distinct clustering
 - Nice for visualization, but be careful!
 - Distances between points may be misleading
 - Similar challenge to squashing a globe onto a flat map... but more extreme!
- Non-reversible (can't infer original data from mapping)
 - Don't use the resulting coordinates for analysis!
- Can be slow
 - Common to use PCA first for partial dimension reduction, then UMAP/tSNE on that
 - UMAP is (usually) faster



To the notebooks, Batman!



Clustering Cells



Dimensionality reduction often results in visible “clusters”, but how do we define those? *Many methods!*

- hierarchical clustering
 - Join closest points/groups recursively
- k-means clustering
 - Pick a number k , then find the “best” way to divide cells into that many groups
 - Assumes clusters are “spherical”
- graph-based clustering
 - Connect cells to other cells with similar expression, then divide up the graph into clusters

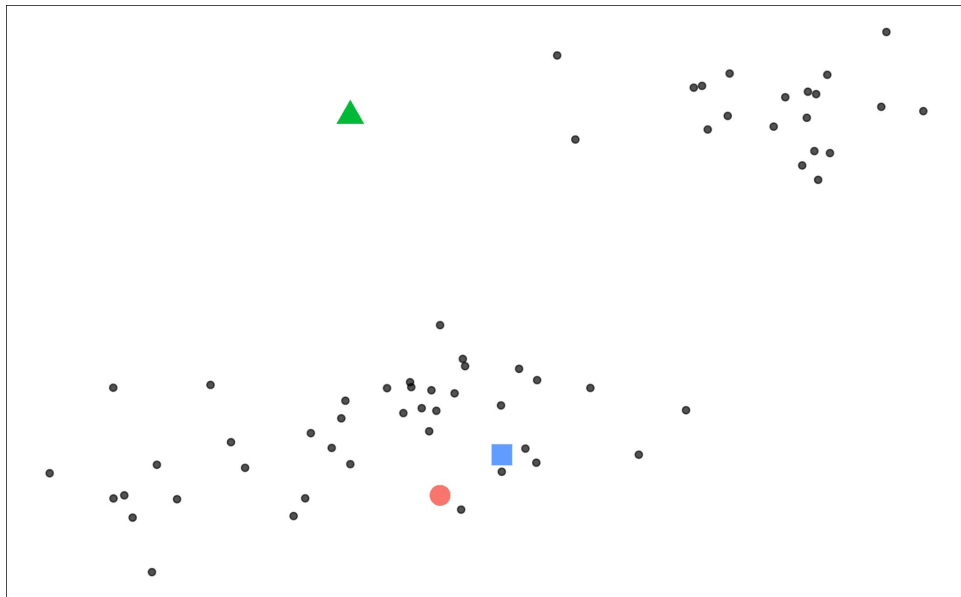
k-means clustering

Step 1: Pick k random centers

Step 2: Assign points to clusters by which center is closest

Step 3: Find new centers as the mean locations of all points in a cluster

Repeat Steps 2 and 3 until the clusters are stable



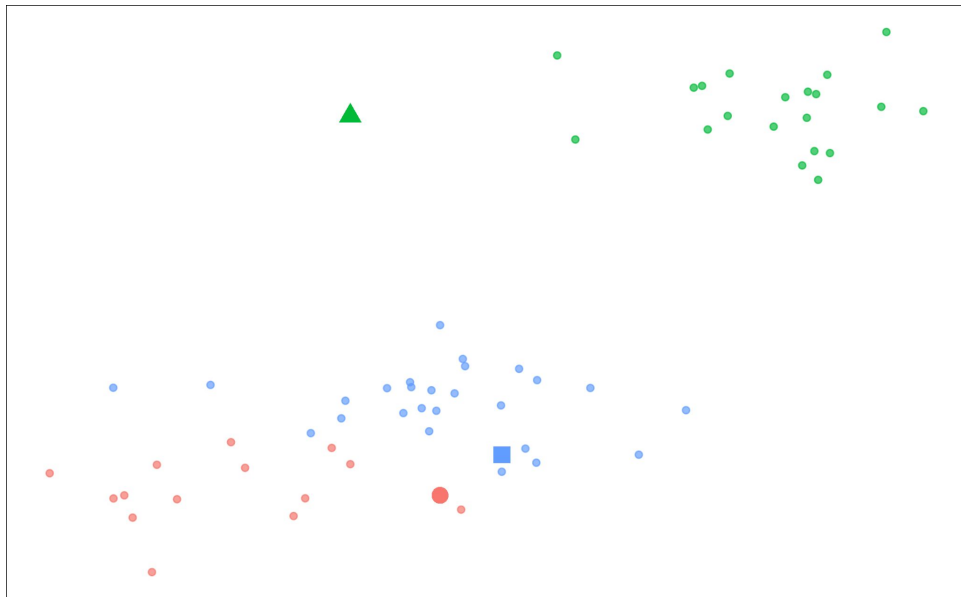
k-means clustering

Step 1: Pick k random centers

Step 2: Assign points to clusters by which center is closest

Step 3: Find new centers as the mean locations of all points in a cluster

Repeat Steps 2 and 3 until the clusters are stable



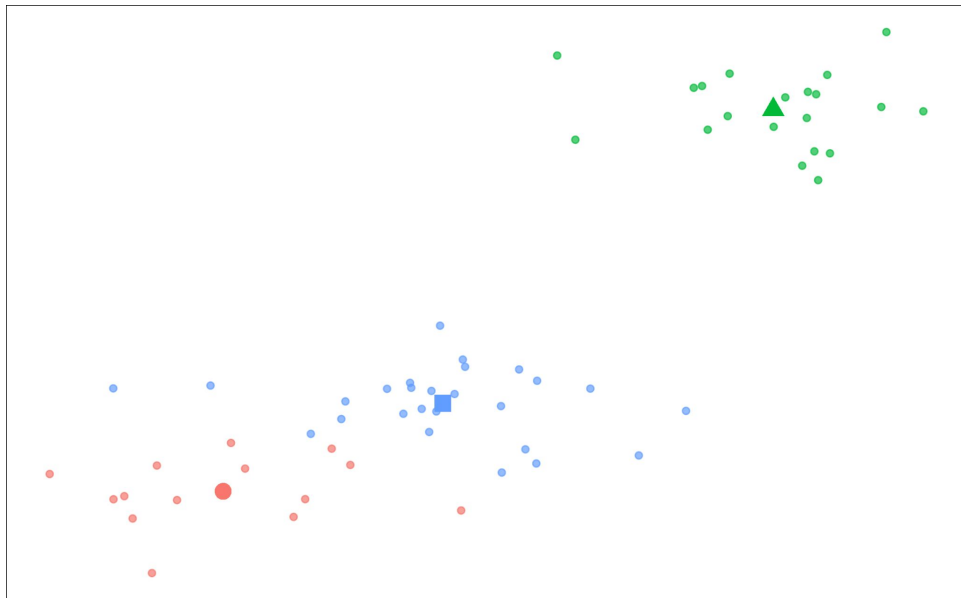
k-means clustering

Step 1: Pick k random centers

Step 2: Assign points to clusters by which center is closest

Step 3: Find new centers as the mean locations of all points in a cluster

Repeat Steps 2 and 3 until the clusters are stable



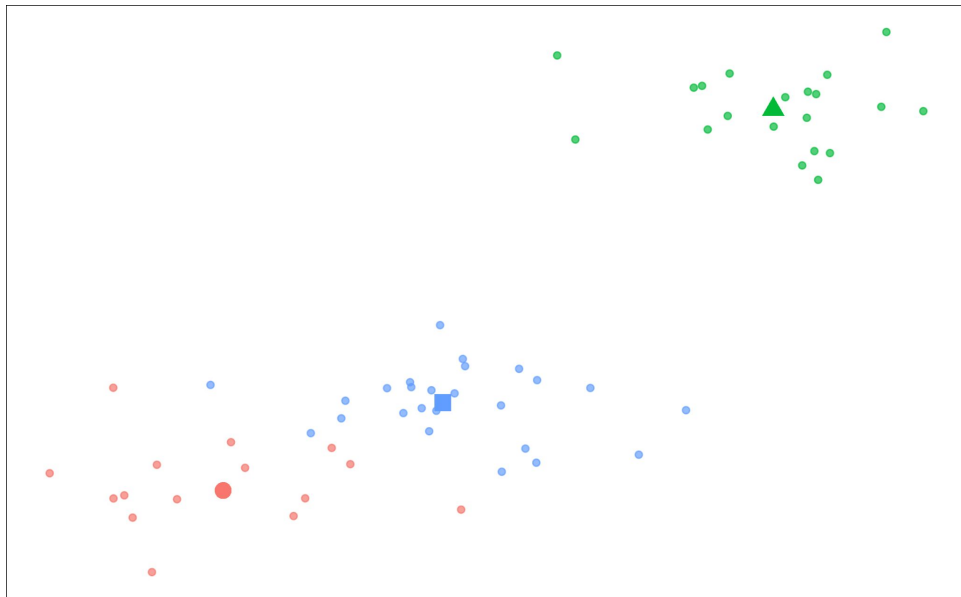
k-means clustering

Step 1: Pick k random centers

Step 2: Assign points to clusters by which center is closest

Step 3: Find new centers as the mean locations of all points in a cluster

Repeat Steps 2 and 3 until the clusters are stable



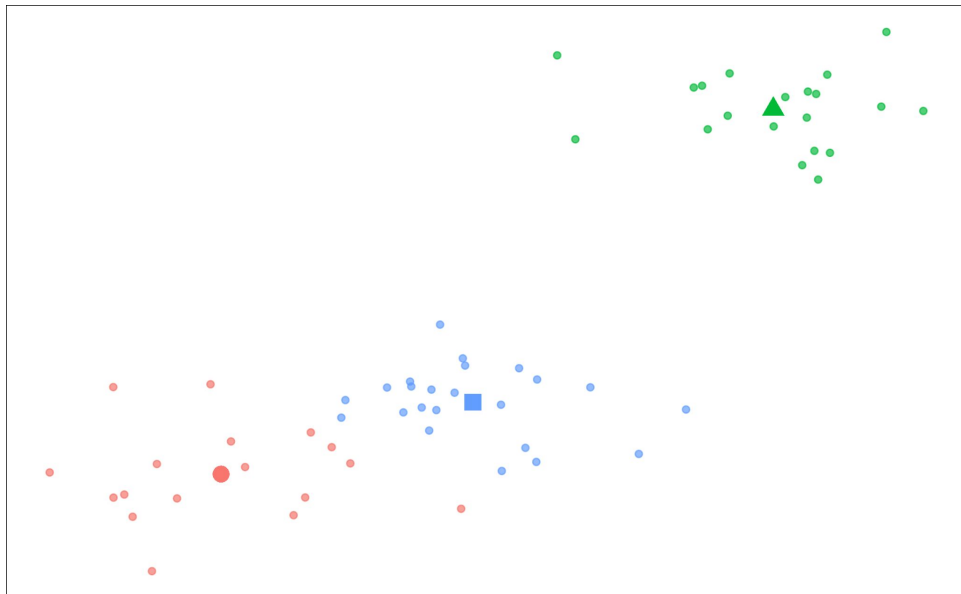
k-means clustering

Step 1: Pick k random centers

Step 2: Assign points to clusters by which center is closest

Step 3: Find new centers as the mean locations of all points in a cluster

Repeat Steps 2 and 3 until the clusters are stable



Graph-based Clustering

Step 1: Calculate similarity matrix among points

Step 2: Build a weighted network graph connecting points to their neighbors

Step 3: Divide network graph into “neighborhoods” based on connection patterns

Many options at each step! The algorithms can determine how many clusters to assign.

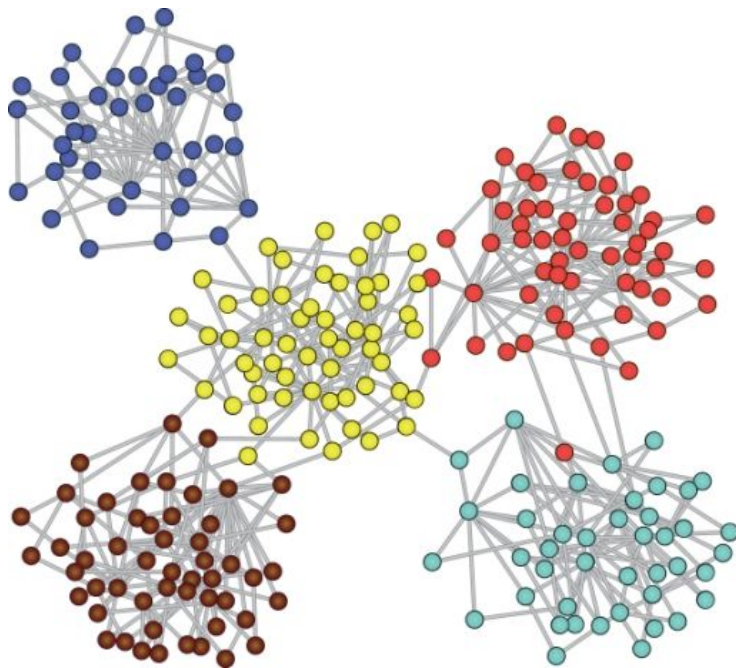
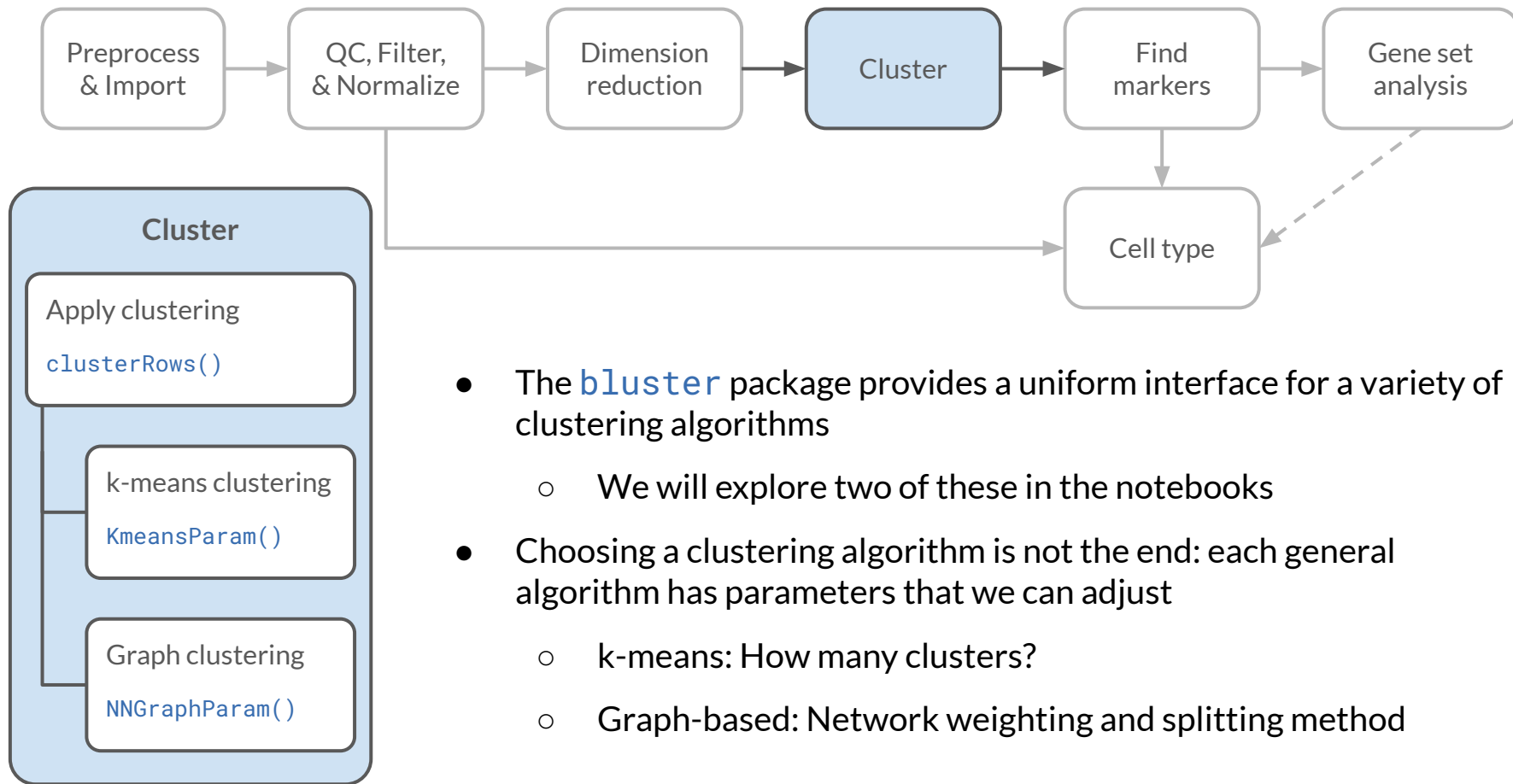


Image from:

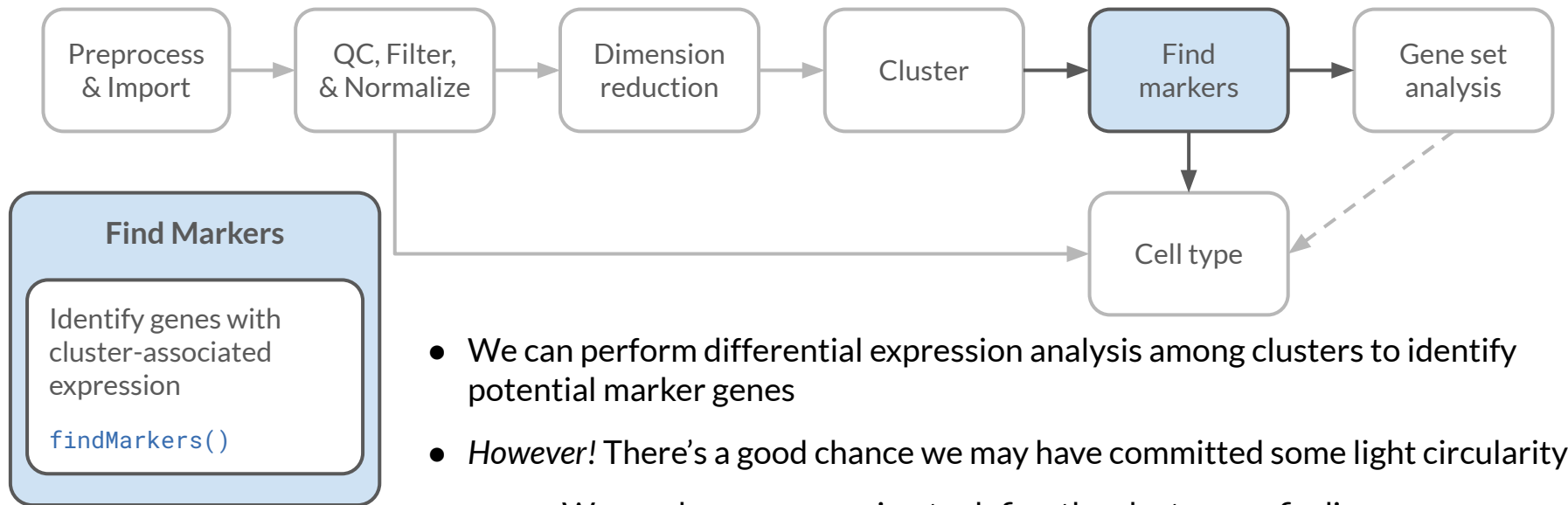
<https://github.com/benedekrozemberczki/awesome-community-detection>



What do the clusters represent?

- Groups of cells with distinct gene expression patterns
- What does that mean?
 - maybe cell types?
 - sometimes cell states?
 - perhaps perturbations?
- Interpretation will vary based on the sample you are using!
 - do not expect a simple mapping of clusters to cell types
- Clustering is usually somewhat stochastic
 - parameter choice and random seeds will affect clusters
 - use caution when interpreting clustering results!

Identifying marker genes



- We can perform differential expression analysis among clusters to identify potential marker genes
- *However!* There's a good chance we may have committed some light circularity
 - We used gene expression to define the clusters, so finding gene expression differences between clusters is expected!
 - Don't rely too much on the specific statistics we calculate (for more, see the [OSCA section on p values](#))