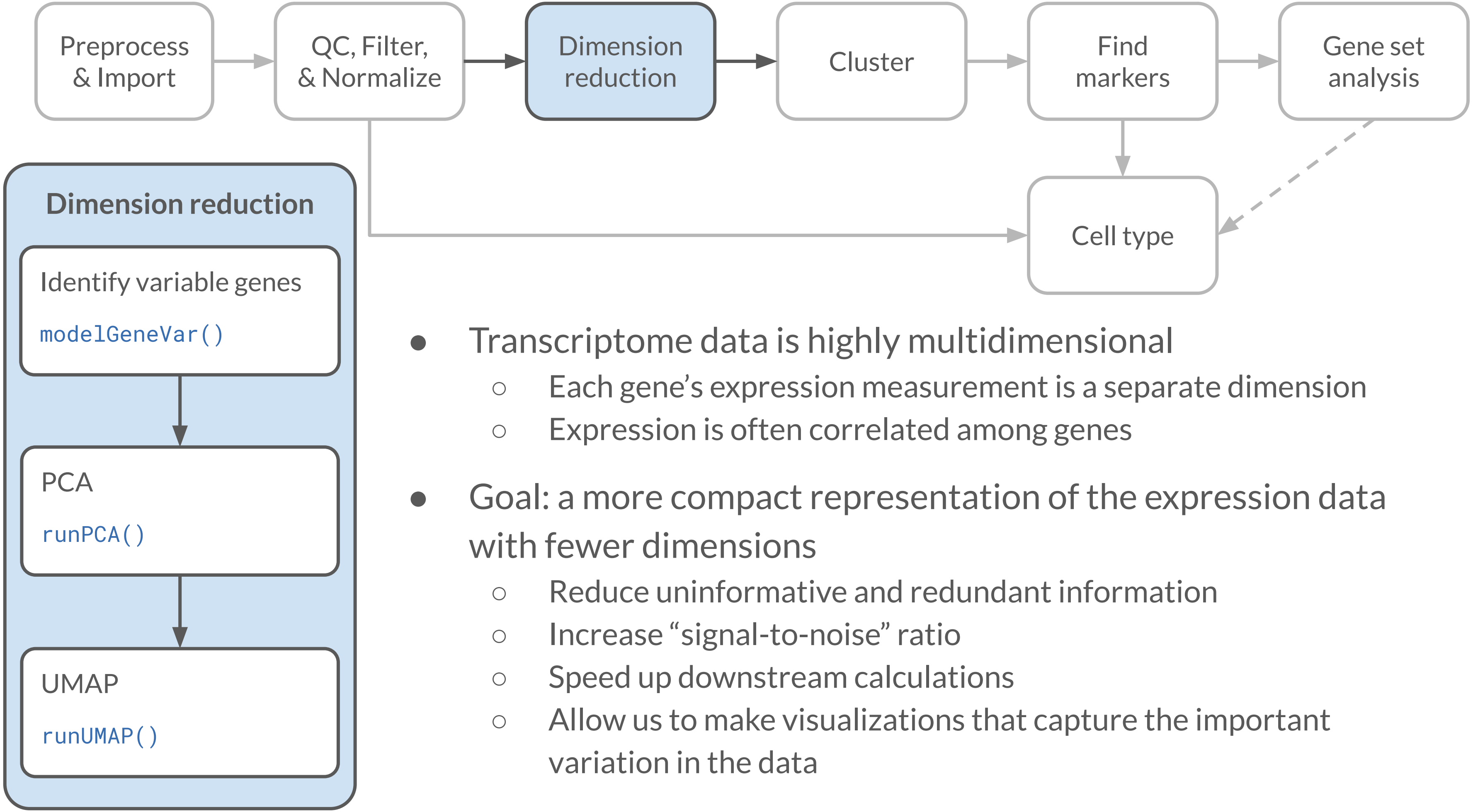
Dimensionality Reduction and Clustering of Single-cell Data

The Data Lab



Preprocess & Import → QC, Filter, & Normalize → Dimension reduction → Cluster → Find markers → Gene set analysis

Cell type

Dimension reduction

Identify variable genes

`modelGeneVar()`

PCA

`runPCA()`

UMAP

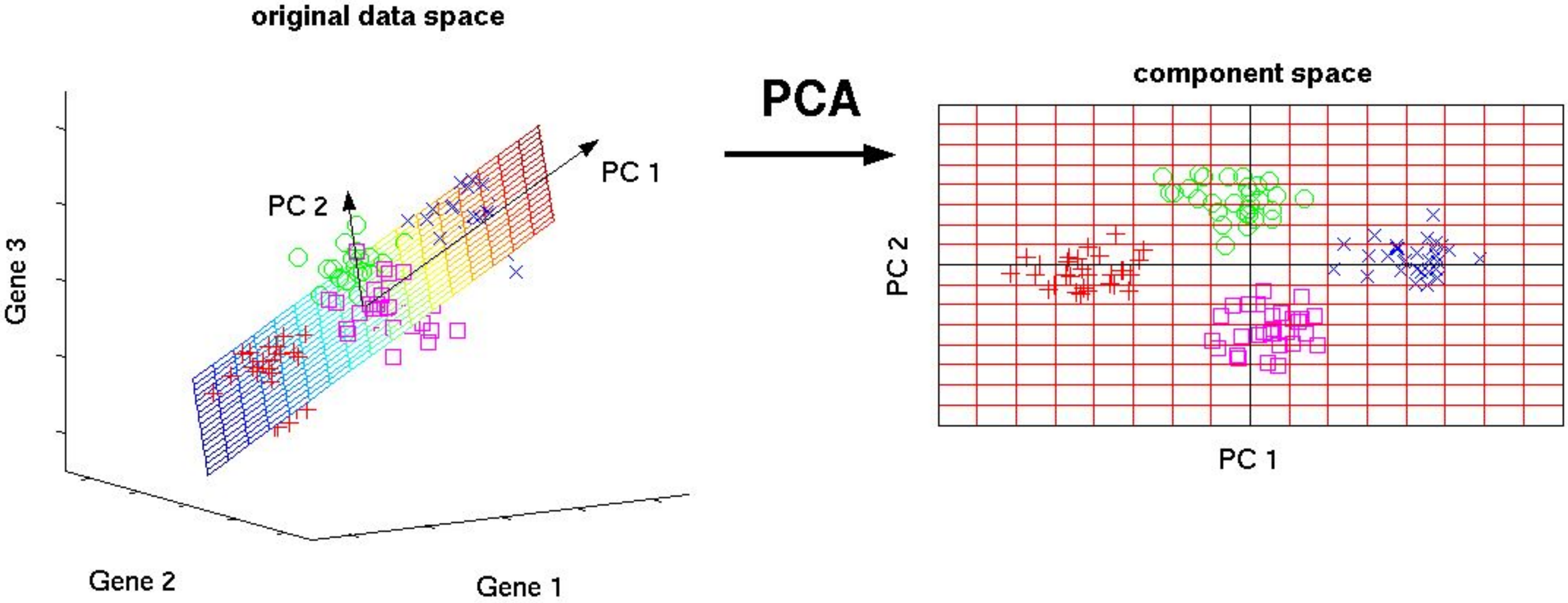
`runUMAP()`

- Transcriptome data is highly multidimensional
  - Each gene's expression measurement is a separate dimension
  - Expression is often correlated among genes
- Goal: a more compact representation of the expression data with fewer dimensions
  - Reduce uninformative and redundant information
  - Increase "signal-to-noise" ratio
  - Speed up downstream calculations
  - Allow us to make visualizations that capture the important variation in the data

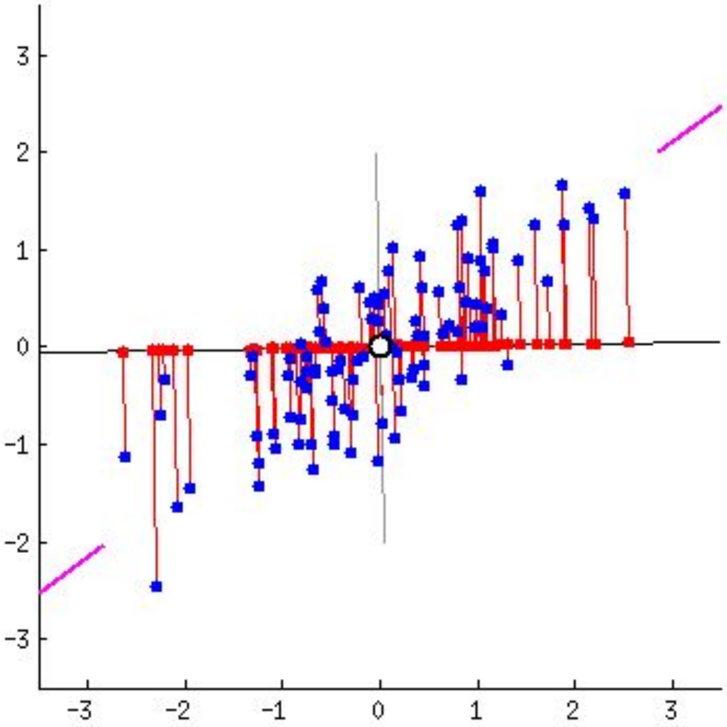
Dimensionality Reduction Methods

- 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)



http://www.nlpca.org/pca_principal_component_analysis.html



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

Assumptions/Limitations of PCA

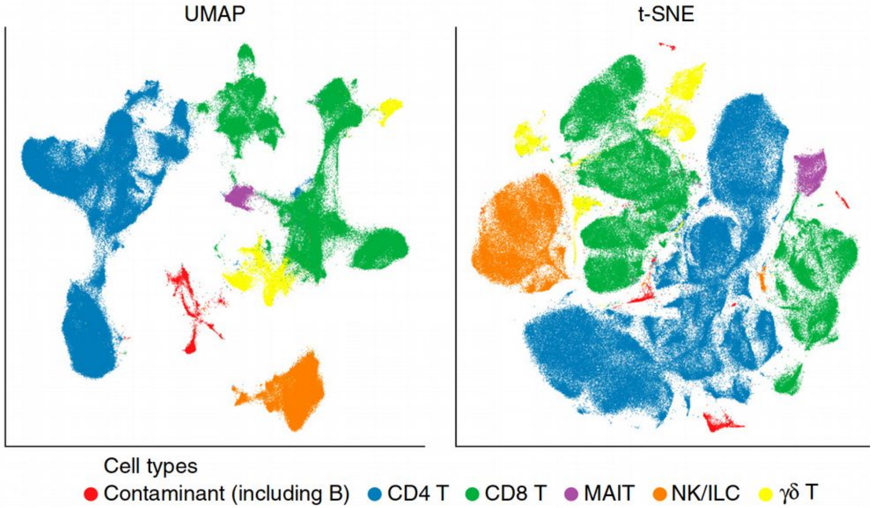
- PCA is a linear transformation of the input data
  - Fast!
  - Reversible if we keep all dimensions
  - Usually we don’t keep everything… removing higher dimensions reduces effects of noise
- Assumes ~ normal distributions for error
  - For scRNA-seq count data, this can be approximated with log-scale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

- GLM-PCA may solve many of these limitations, but is not in wide use: (Townes *et al.* 2019 https://doi.org/10.1186/s13059-019-1861-6)

UMAP and tSNE

Machine learning methods for dimensionality reduction

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



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

- 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/

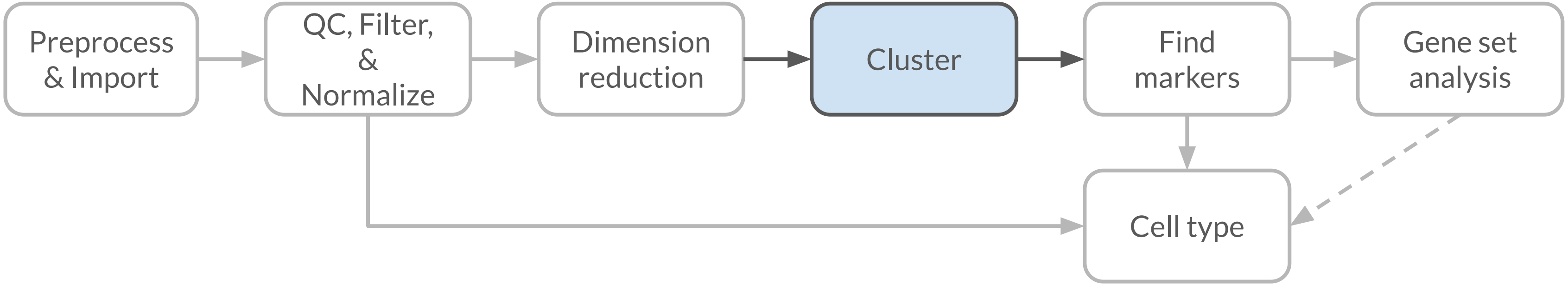
Assumptions/Limitations of UMAP & tSNE

- 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

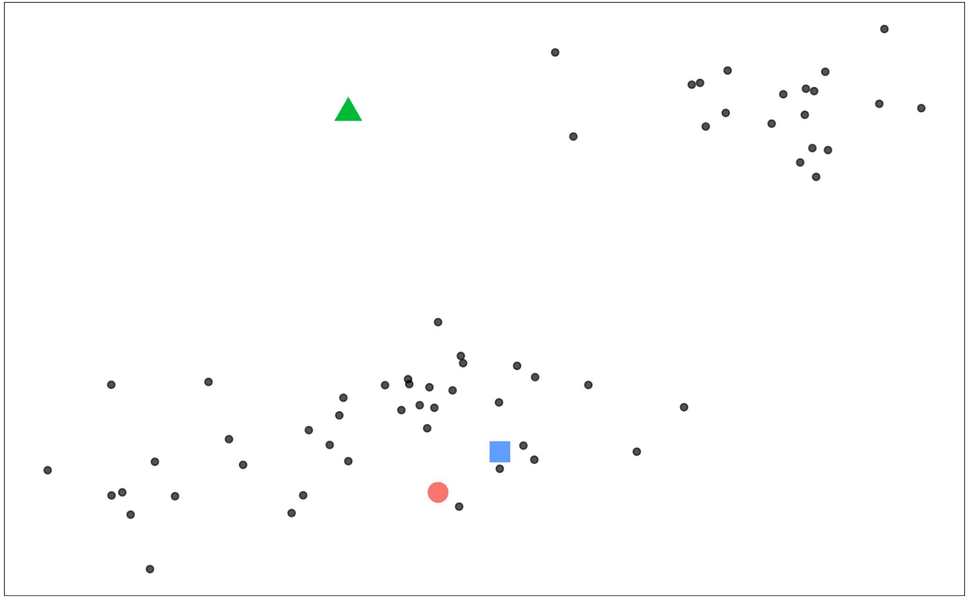
k-means clustering

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



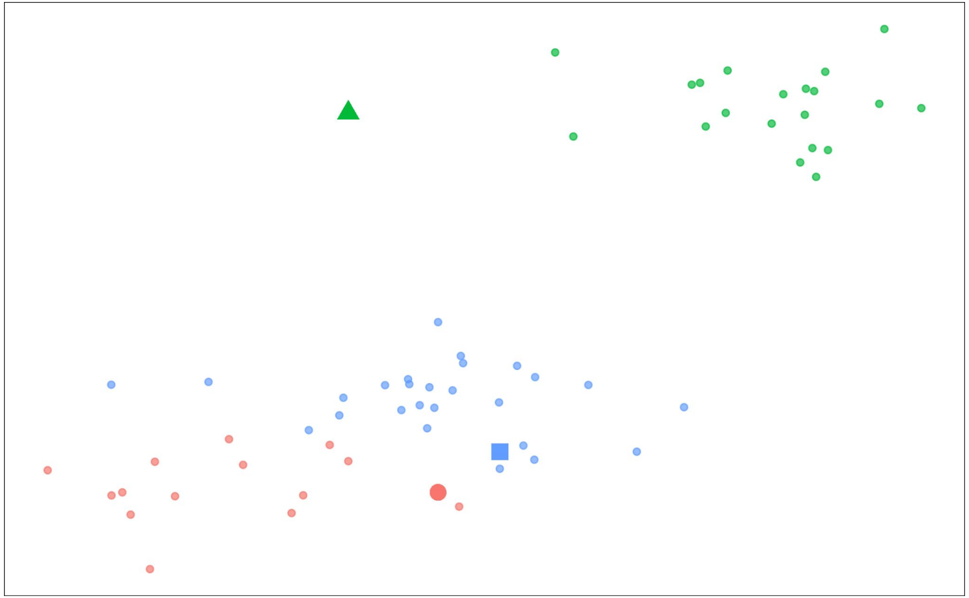
k-means clustering

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



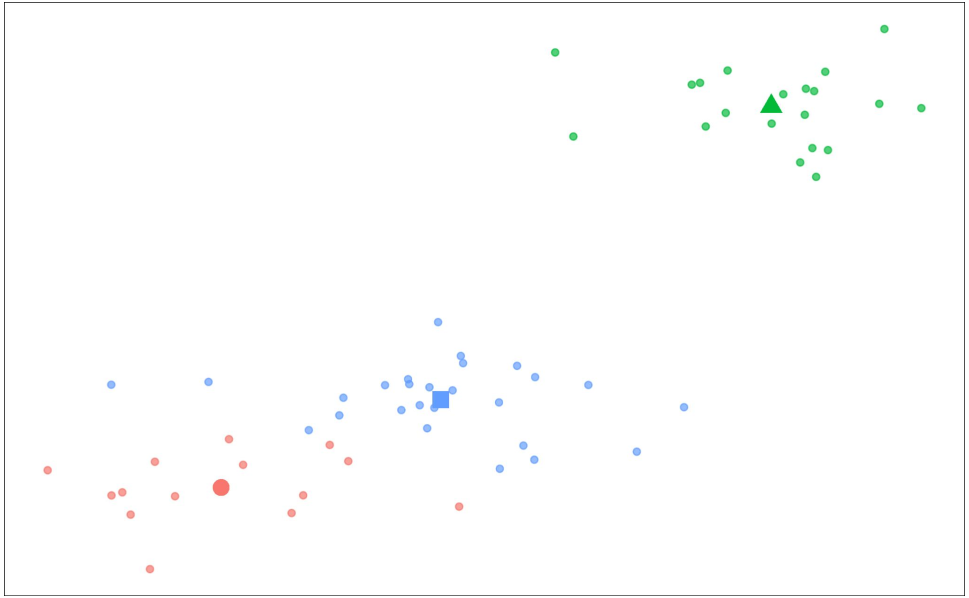
k-means clustering

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



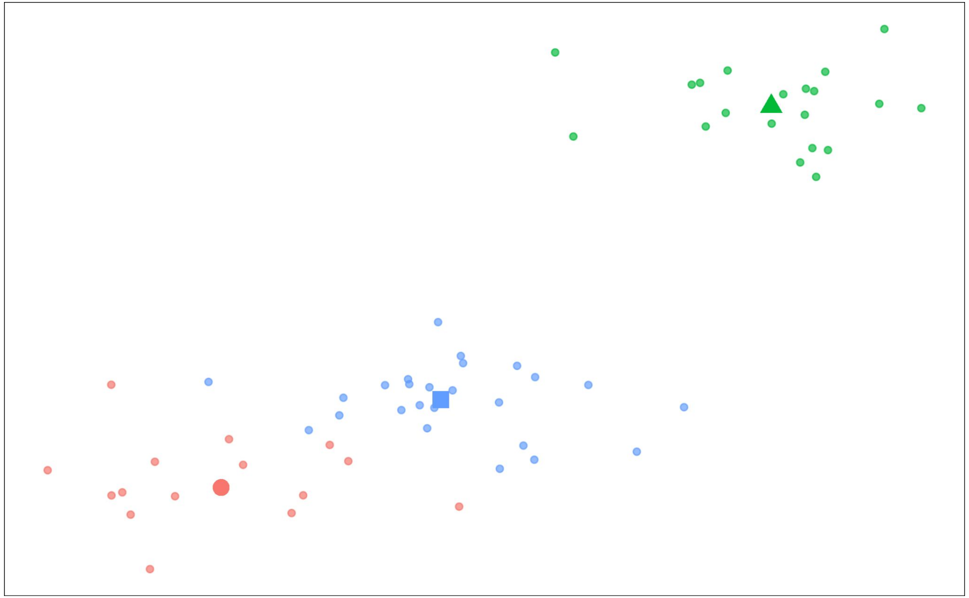
k-means clustering

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



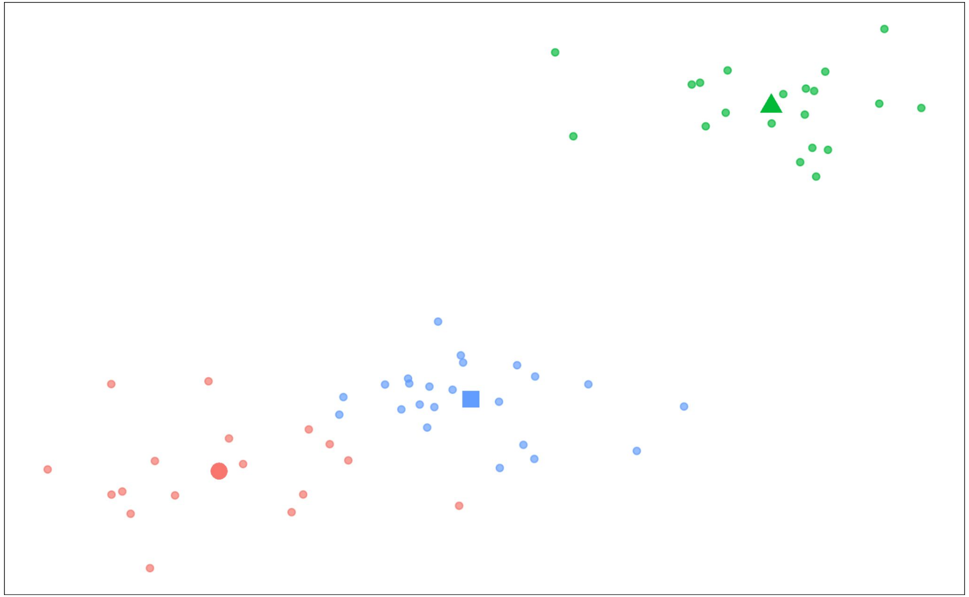
k-means clustering

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.

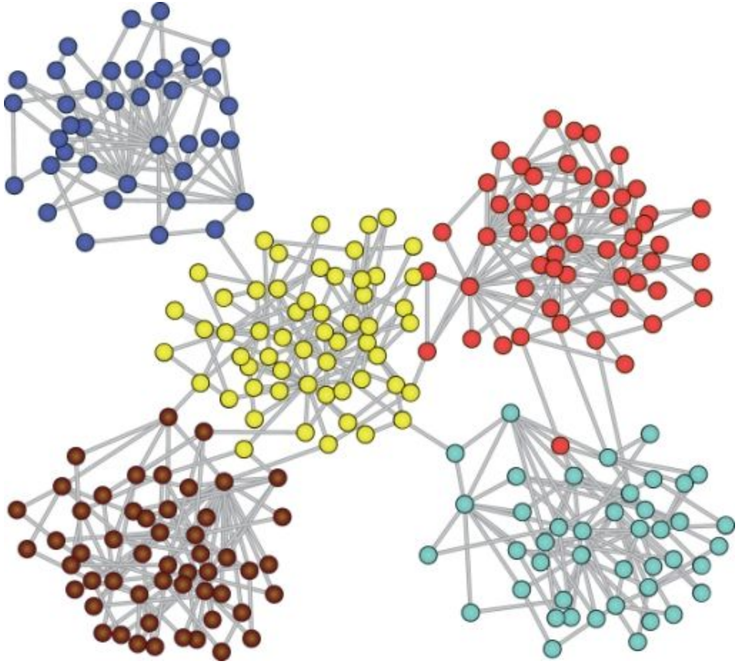
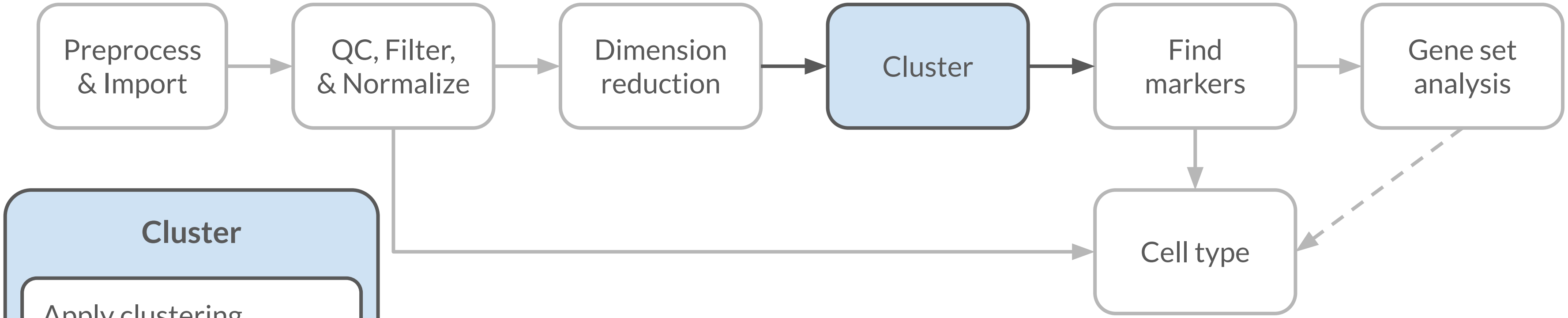


Image from:
https://github.com/benedekrozemberczki/awesome-community-detection



Cluster

Apply clustering

`clusterRows()`

- k-means clustering

  `KmeansParam()`

- Graph clustering

  `NNGraphParam()`

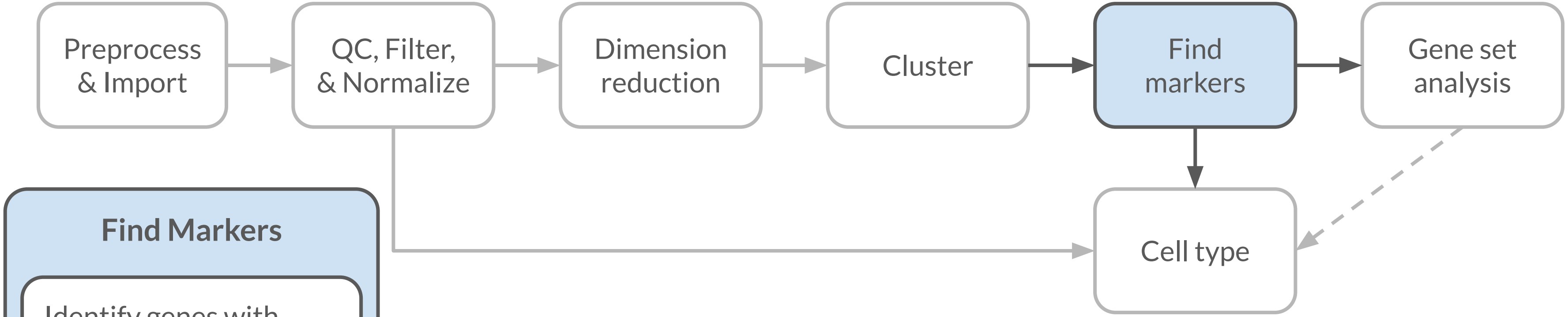
- The `bluster` package provides a uniform interface for a variety of clustering algorithms
  - We will explore two of these in the notebooks
- Choosing a clustering algorithm is not the end: each general algorithm has parameters that we can adjust
  - k-means: How many clusters?
  - Graph-based: Network weighting and splitting method

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



Preprocess & Import

QC, Filter, & Normalize

Dimension reduction

Cluster

Find markers

Gene set analysis

Cell type

Find Markers

Identify genes with cluster-associated expression

`findMarkers()`

- 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)