Clustering Methods in Single-Cell Analysis

Graph-based Clustering

Build kNN graph then find communities

Leiden Algorithm

Improved Louvain with better guarantees

K-means Adaptations

SC3 uses consensus clustering

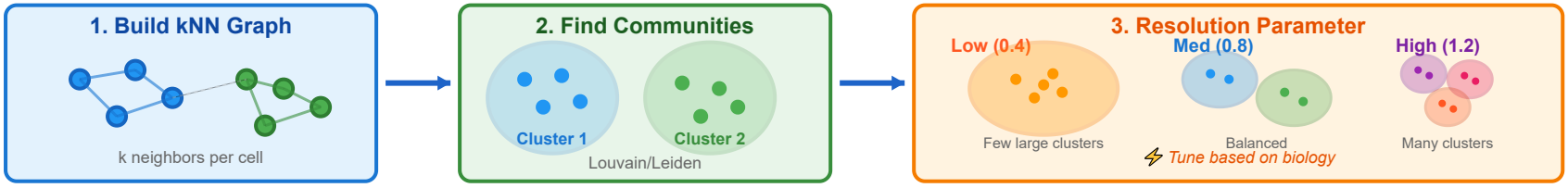
Resolution Selection

Higher resolution = more clusters

Stability Analysis

Bootstrap to assess cluster robustness

💡 No single correct clustering - depends on biological question



Detailed Method Explanations

1 Graph-based Clustering

Graph-based clustering represents cells as nodes in a network where edges connect similar cells. This approach naturally captures the manifold structure of single-cell data.

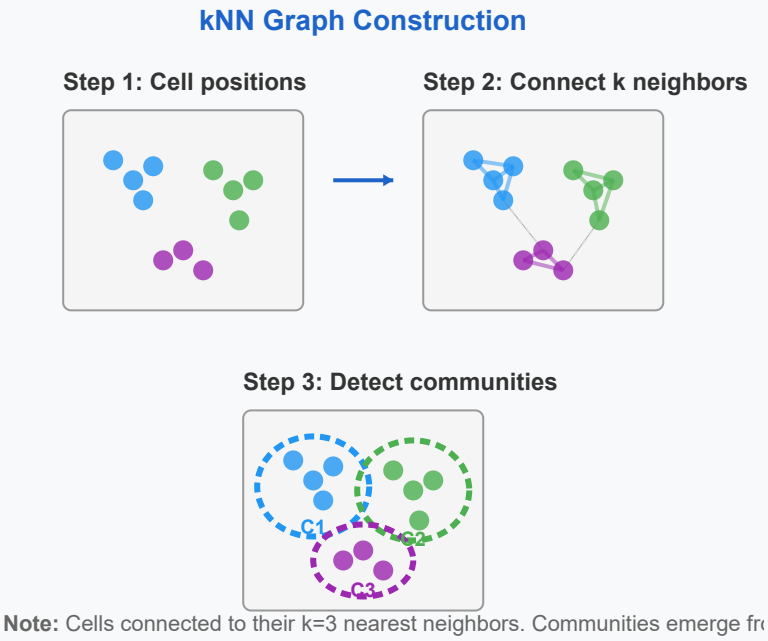
Key Steps:

- **Graph Construction:** Build a k-nearest neighbor (kNN) graph where each cell is connected to its k most similar cells based on gene expression distances
- **Edge Weighting:** Edges are weighted by similarity (e.g., Euclidean distance in PCA space or cosine similarity)
- **Community Detection:** Apply algorithms like Louvain or Leiden to identify densely connected groups (communities)

Advantages:

- Handles complex, non-convex cluster shapes naturally
- Computationally efficient for large datasets
- Robust to noise and outliers
- Standard approach in Seurat and Scanpy

> **Typical Parameter:** k = 10-30 neighbors is common for most scRNA-seq datasets. Higher k creates more connections but may merge distinct populations.



Note: Cells connected to their k=3 nearest neighbors. Communities emerge fr

Leiden Algorithm

The Leiden algorithm is an improved community detection method that addresses key limitations of the popular Louvain algorithm, particularly ensuring well-connected communities.

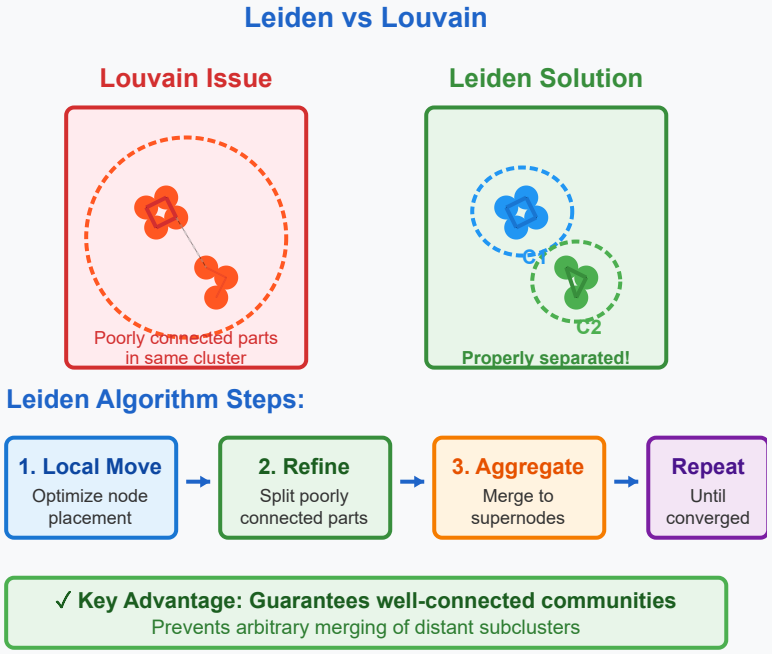
Improvements over Louvain:

- **Guarantees Connectivity:** Ensures communities are internally connected, preventing poorly connected subcommunities
- **Faster Convergence:** Typically requires fewer iterations to reach optimal partitioning
- **Quality Function:** Optimizes modularity while maintaining community quality
- **Refinement Phase:** Includes additional refinement step to split disconnected parts

Algorithm Phases:

- **Local Moving:** Move nodes between communities to increase modularity
- **Refinement:** Split weakly connected subcommunities
- **Aggregation:** Collapse communities into supernodes
- **Repeat:** Iterate until no improvement

Why it matters: Leiden is the recommended algorithm in Scanpy and provides more reliable clustering results,



especially for complex datasets with hierarchical structure.

```
# Python (Scanpy)
sc.tl.leiden(adata, resolution=0.8)

# R (Seurat) - uses Louvain by default
adata <- FindClusters(adata, resolution=0.8,
algorithm=4) # 4 = Leiden
```

3 K-means Adaptations (SC3)

SC3 (Single-Cell Consensus Clustering) adapts k-means clustering for single-cell data through consensus approaches and careful parameter selection.

SC3 Methodology:

- **Multiple Transformations:** Apply different distance metrics and dimensionality reductions
- **Consensus Clustering:** Run k-means multiple times with different initializations
- **Similarity Matrix:** Build consensus across multiple runs to stabilize results

- **Optimal k Selection:** Uses silhouette width and other metrics to estimate cluster number

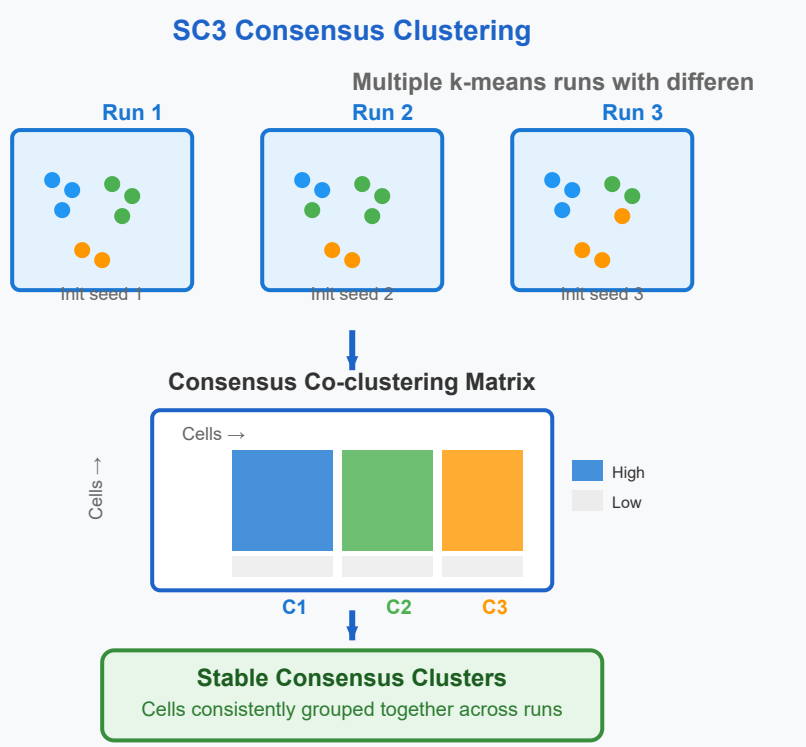
Why Consensus?

- K-means is sensitive to initialization (local minima problem)
- Different random starts can produce different results
- Consensus approach averages over many runs for stability
- Identifies core cluster memberships that persist across runs

Limitations:

- Assumes spherical clusters (unlike graph-based methods)
- Requires specifying k (number of clusters) in advance
- Computationally intensive for large datasets
- Less popular than graph-based methods for scRNA-seq

> **When to use:** SC3 is useful for smaller datasets where you want robust cluster number estimation and are willing to invest computational time.



4 Resolution Parameter Selection

The resolution parameter controls the granularity of clustering in community detection algorithms. This is one of the most important tuning parameters in single-cell clustering.

How Resolution Works:

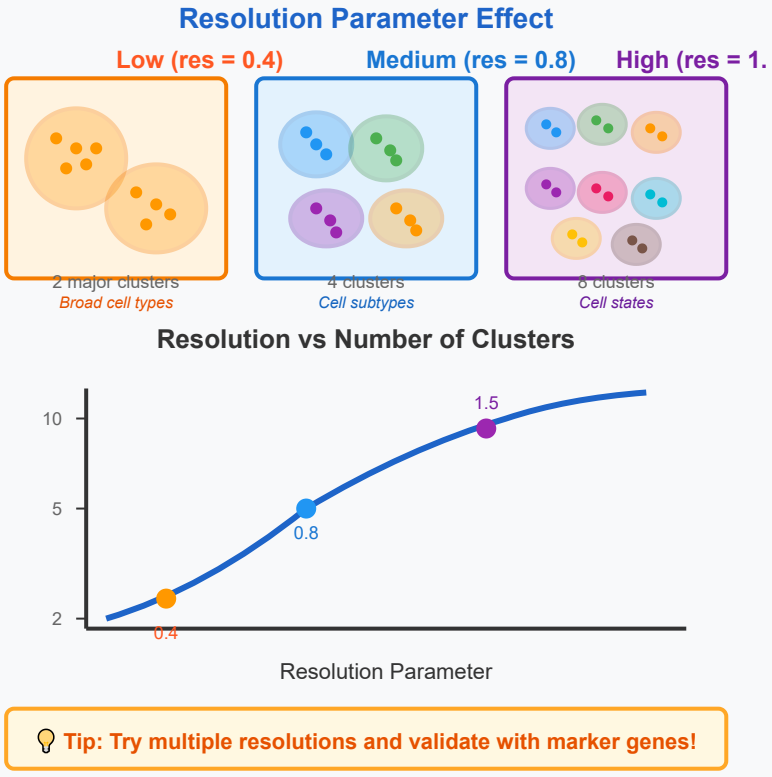
- **Low Resolution (0.1-0.5):** Favors larger communities, merges similar populations
- **Medium Resolution (0.6-1.0):** Balanced granularity, commonly used default
- **High Resolution (1.0-2.0+):** Produces more fine-grained clusters, separates subtle differences

Biological Interpretation:

- **Cell Type Level:** Lower resolution (0.4-0.6) for major cell types (T cells, B cells, monocytes)
- **Subtype Level:** Medium resolution (0.8-1.2) for cell subtypes (CD4+ T cells, CD8+ T cells)
- **State Level:** Higher resolution (1.5-2.0) for cell states (activated, resting, proliferating)

Selection Strategies:

- Start with default (0.8-1.0) and adjust based on results
- Use multiple resolutions and compare with biological knowledge
- Check cluster stability at each resolution
- Consider the biological question being asked



Tip: Try multiple resolutions and validate with marker genes!

Important: There is no "correct" resolution - it depends on your research question. Are you interested in broad cell types or fine-grained states?

5 Cluster Stability Analysis

Stability analysis assesses how robust clustering results are to perturbations in the data. This helps identify reliable clusters versus those that may be artifacts.

Bootstrap Resampling:

- **Process:** Randomly sample cells with replacement, recluster multiple times
- **Stability Score:** Measure how often cells cluster together across bootstrap runs
- **Interpretation:** High stability (>0.8) indicates robust clusters; low stability suggests uncertain boundaries

Subsampling Analysis:

- Remove random subsets of cells and recluster
- Check if major clusters persist
- Identifies clusters dependent on specific cells

Metrics for Stability:

- **Adjusted Rand Index (ARI):** Measures agreement between clusterings (0=random, 1=perfect)
- **Jaccard Index:** Overlap between cluster memberships
- **Co-clustering Frequency:** How often cell pairs are grouped together

Practical Applications:

- Identify stable core clusters to report in publications
- Flag uncertain cells at cluster boundaries
- Compare stability across different resolutions
- Validate that biological conclusions don't depend on unstable clusters

> **Best Practice:** Always perform stability analysis before biological interpretation. Don't over-interpret unstable clusters!

```
# Python example with clustree-like approach
for i in range(100): # 100 bootstrap runs
    subsample =
adata[np.random.choice(adata.n_obs,
                size=adata.n_obs, replace=True)]
    sc.tl.leiden(subsample, resolution=0.8)
    # Calculate co-clustering matrix
```