Bulk analysis:

1. Quality control
2. Count table
3. Dimensionality reduction (PCA)
4. Differential expression (DESeq)
   comparing treated to untreated

Single-cell analysis:

1. Quality control analysis
2. Count table
3. Dimensionality reduction (tSNE or UMAP)
4. Differential expression (DESeq)
   Next step: **clustering**

CLUSTERING

To identify subpopulations of cells within a sample (e.g., treated sample) in scRNA-seq experiments. Key points:

- **Number of clusters** we want to obtain
- **Approach** to adopt to obtain clustering (there are many) → the approach is chosen based on the parameters that drive the most relevant differences. Clustering is based on differential genes' expression, so that if cells are similar in their transcriptomic profiles the analysis will be more difficult. Approaches we'll see:
  1. **K-means clustering**
  2. **Louvain modularity-based clustering**
  3. **Random projection-based clustering**

---

1. K-means clustering

It requires the user to know how many clusters they want. With this approach the user sets a K value, let's suppose K=3 → 3 random points in the space are taken: these will be the centre of our clusters. Each cell (based on its transcriptome) is assigned to a cluster, i.e. the one that is less distant from the considered cell. This is done cell by cell. Then, after this first round of centre-assignation, the algorithm repeats again the procedure after having calculated the mean of each cluster: basically in the second round each cell is assigned to one of the newly calculated clusters (centres). This is repeated until cells do not move anymore from a cluster to another. The problem is if centres are problematic from the very beginning (when they're initialized). The quality information about the clustering is related to the max variation between the elements of each cluster.

Ho to define the optimal number of clusters? Basic concept: the more the clusters, the smaller the variance associated to each cluster (the more the number of clusters approaches the number of cells, of course the variance shrinks, because in the extreme case each cluster corresponds to one cell → if K = number of data points, variance = 0).

Working mechanisms of K-means algorithm. The users provides the number of clusters: K=3. These are the steps that follow:

1. Random centroid selection → 3 random data points are taken: centroids.
2. Distance measurement → the distance between each centroid and each actual data point is calculated to assign the data points to their closest centroid.
3. Cluster mean measurement and reassignment → for each previously defined cluster, mean distances between the points within the cluster are calculated to obtain a new centroid (cluster by cluster). Now the algorithm calculates the distance between each data point and each newly calculated centroid to reassign data points to their closest cluster.

4. The whole process is reiterated until data points stay still in their positions.
5. Variance can be now calculated (variance = how much points are dispersed from the centroid of their clusters). Less variance means a better clustering.
6. Steps 1-5 are repeated starting from new 3 K points (centroids), so we end up with a variance value again. Normally 10k repetitions are carried out and in the end the variances obtained in each are ranked to select the best clustering considering the given data sets (the one with smaller variance).

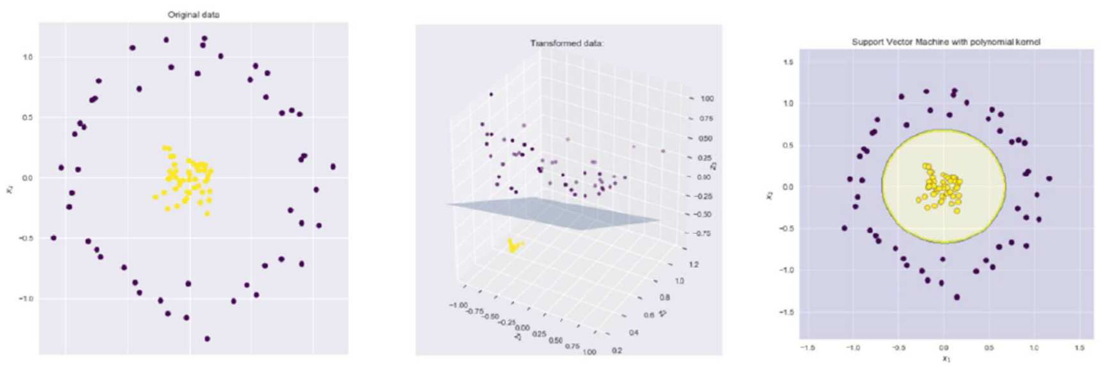
More complex data require more repetitions. Also, bigger datasets generally give greater overlap between clusters (more difficult to differentiate them).

How to select the best K at the beginning? **Elbow plot** → every time K increases by 1, variation reduces (of course), until a K value is reached where variance stops decreasing significantly. At that value, called "elbow point", we can stop increasing our K (and we use that threshold value of K).

Single-cell interpreter with multi-kernel learning (SIMLR)

Sometimes K-means clustering isn't enough because the prior data reduction step wasn't sufficient to well represent data. SIMLR is a K-mean with different modality of data reduction, particularly it's a kernel-based approach (**kernel-based approach** = original data are represented in a different way, i.e. projected on a different plane, and the so-obtained projections are called *kernels*. Similar to the t-SNE approach). SIMLR allows to find the best data representation possible (to better separate them). SIMLR algorithm generates a hyperplane to change the representation. After every projection attempt, SIMLR proceeds with clustering: only those kernels (projections) that better fit the cluster number you gave as input are selected. With the info from the selected kernel we make a similarity matrix that we'll use for dimensional reduction (to retrieve a good graphical representation): data are kind of subset to make a better representation.

This approach is limited by the number of cells (no more than 2000) and by the fact that the number of genes should exceed that of cells.



Silhouette method

There are many ways to find the best number of clusters. One of these is the **Silhouette coefficient (SC)**. SC ranges from -1 to 1 and it's based on 2 parameters:

- Cohesion between cells of the same cluster → **a(x)**
- Separation between cells (points) of a cluster and cells from another cluster → **b(x)**

High cohesion and high separation are ideal (Silhouette score closer to 1).

$$\text{Silhouette:} \quad s(x) = \frac{b(x) - a(x)}{\max\{a(x), b(x)\}}$$

$$\text{Silhouette coefficient (SC):} \quad \text{SC} = \frac{1}{N}\sum_{i=1}^{N} s(x)$$

This method works well only if you've a good separation of the clusters.

Cell stability score

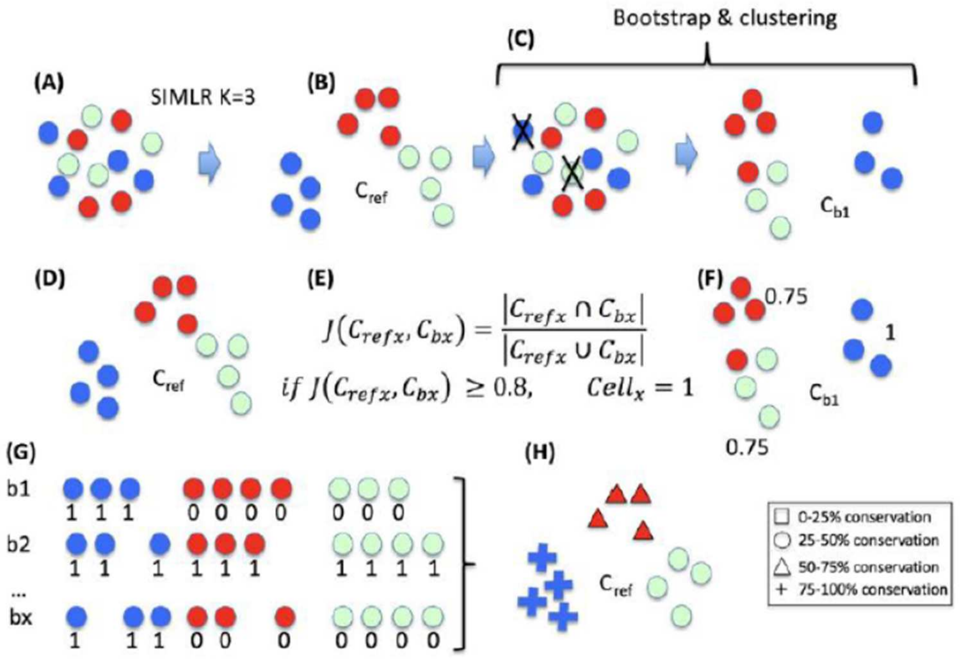
After clustering you make sort of a test (**perturbation process**): you take a random subset of cells out of the original dataset and you repeat the clustering only with the remaining cells. The result is the same as before if the cells that were picked were not important; some clusters might be perturbed instead, because they were less stable than others. This perturbation process is performed several times to calculate how many times cells do not change cluster.

More in detail: from the original (called *reference*) cluster, we do a *first permutation cluster* (when we remove a subset of cells from the dataset for the first time, could be 10-15% of the data points). We use the **Jaccard index** to compare the composition of cells in the first permutation cluster with that of the reference cluster. Jaccard score = 1 if cells did not move and are in the same cluster. No score is assigned (=0 i guess) when the cluster is not robust enough to contain the same cells. Jaccard scoring system gives info about stability of clusters. Jaccard score is a cell stability score (CSS).

- CSS > 75% is obtained if the cluster hosts at least 90-93% cells of the same type.

Stability gets worse when a cluster contains different cell types. Generally speaking, CSS is inversely correlated to cells' heterogeneity within the same cluster. If CSS is bad we can either change the clustering tools or the number of clusters we set. *Transition cells* are those at the boundaries of two different clusters and have a lower CSS.

By increasing perturbation (i.e. you remove more cells from the dataset) you're going to generate a new dataset and the overall stability will drop.



2. Louvain modularity-based methods

They include **Griph** and **Seurat clustering** → number of clusters is estimated and there's no need to give it as input. Quick approach, thus often used to have a rough idea of which could be the ideal number of cluster to give as input using different clustering methods (e.g., SIMLR).

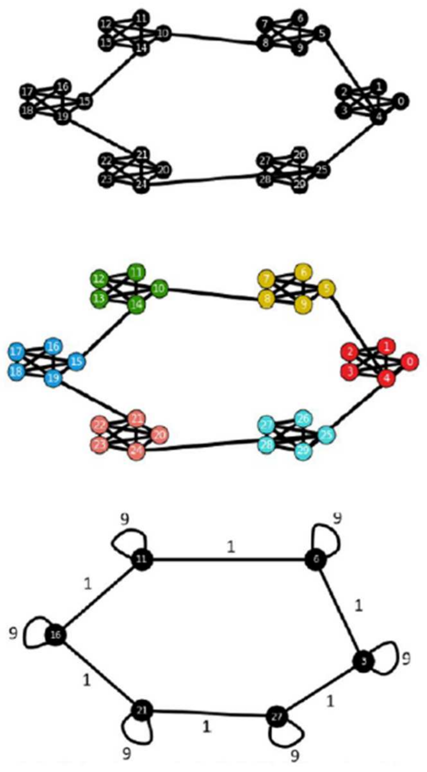
- Seurat method does that but it doesn't use the full dataset (it extrapolates) and PCA as dimensional reduction of data
- Griph method uses the full dataset I guess and a tSNE visualization as data dimensional reduction approach

Louvain modularity method quantifies the quality of the assignment of nodes to *communities* (that are made of cells that tend to stay together, i.e. groups of nodes within a network that are more densely connected to each other rather than to other nodes) by evaluating how much condensed these nodes are compared to the average, expected connection in a random network instead.

- *Heuristic approach*: any approach to problem solving or self-discovery that is not (necessarily) the best, but allows to reach an immediate, short-term goal. It basically speeds up the process of finding a satisfactory solution. So the global optima solution is impractical to be identified and this approach at max tries to find local optimal solution at each step of the process.
- *Greedy algorithm*: any algorithm following the heuristic approach to find a global optimum in a reasonable amount of time.
- *Coarse-Grained modelling*: representation of a network (simplified representation of data).

Louvain modularity approach is based on this:

1. *Greedy* assignment of nodes to communities (as a greedy approach, local optimizations are favoured to speed up the process), i.e., definition of potential groups of nodes that likely make communities.
    - Random aggregation of a certain number of nodes to form communities (also called cliques). We know nothing about connection now, we're just expecting to have the highest possible number of connections between nodes fallen in the same community (i.e. highest modularity possible).
2. Definition of a new *coarse-grained* network based on the communities previously identified.
    - Each community is converted to a "main" node with an associated value that is the number of connections the nodes establish in that community. The choice of a main node is just to label the various communities.



Points are moved until the best association between points (highest modularity possible) is obtained, meaning that steps 1 and 2 are repeated multiple times (community reassignment is done several times).