Methods for assessing clustering stability in single-cell expression datasets[1]

Andi Munteanu [1]

Supervisors
Dr. Liviu Ciortuz [1]
Dr. Irina Mohorianu [2]

[1]Faculty of Computer Science, Alexandru Ioan Cuza University, Iași
[2]Wellcome-MRC Cambridge Stem Cell Institute, University of Cambridge

June 2022

Outline

Table of Contents

asdfda

Table of Contents

PhenoGraph pipeline

PhenoGraph is a pipeline proposed by Levine et al. [2] that applies graph clustering on "normal" datasets.

Consists of three steps:

- dimensionality reduction

PhenoGraph pipeline

PhenoGraph is a pipeline proposed by Levine et al. [2] that applies graph clustering on "normal" datasets.

Consists of three steps:

- dimensionality reduction
- graph construction

PhenoGraph pipeline

PhenoGraph is a pipeline proposed by Levine et al. [2] that applies graph clustering on "normal" datasets.

Consists of three steps:

- dimensionality reduction
- graph construction
- graph clustering

Divergent results

We compared the results of two R packages, Seurat [3] and Monocle [4], that implement the PhenoGraph pipeline to process single-cell data.

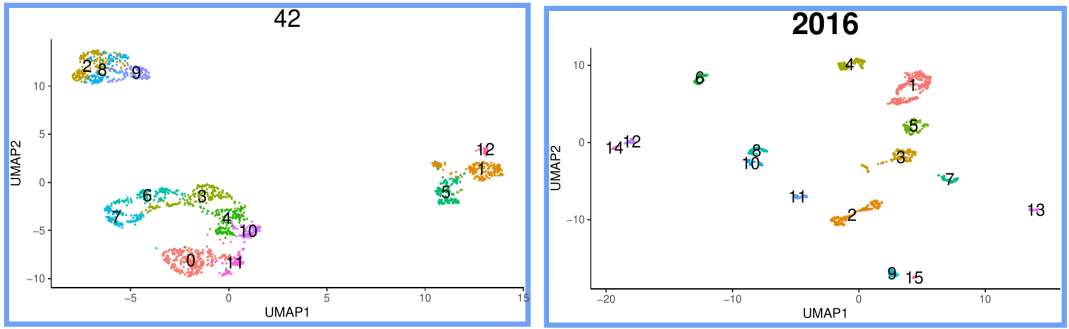

Figure: **Results obtained on default results** Clustering distribution with default parameters for Monocle (right) and Seurat (left). The title indicates the default random seed that is used.

Aligning the results

Parameter name	Step	Monocle value	Seurat value
seed	-	2016	42
feature set	Dim reduction	all genes	HV genes
UMAP min dist	Dim reduction	0.1	0.3
UMAP n neighbours	Dim reduction	15	30
base embedding	Graph building	UMAP	PCA
SNN implementation	Graph building	no self-neighbours only direct neighbours	with self-neighbours direct and indirect neighbours
graph type	Graph building	unweighted	weighted
clustering method	Graph clustering	Leiden	Louvain
quality function	Graph clustering	CPM	RBConfiguration
resolution	Graph clustering	1e-4	0.8
#iterations	Graph clustering	2	10

Table: **Parameters used for aligning the results of the two packages**

Table of Contents

Table of Contents

What is ECS?

- An *affinity matrix* of a partition describe the co-occurrences of different elements by calculating the number of different paths between them.

ECS properties

What is ECS?

- An *affinity matrix* of a partition describe the co-occurrences of different elements by calculating the number of different paths between them.
- For a disjoint partition, the affinity matrix can be calculated using the formula:

$$p_{ij} = \begin{cases} 0, \text{if } i \text{ and } j \text{ don't belong to the same cluster} \\ \frac{\alpha}{|C_\beta|}, \text{if } i \text{ and } j \text{ belong to the cluster } \beta \\ 1 - \alpha + \frac{\alpha}{|C_\beta|}, \text{if } i = j \end{cases}$$

ECS properties

What is ECS?

- An *affinity matrix* of a partition describe the co-occurrences of different elements by calculating the number of different paths between them.
- For a disjoint partition, the affinity matrix can be calculated using the formula:

$$p_{ij} = \begin{cases} 0, \text{if } i \text{ and } j \text{ don't belong to the same cluster} \\ \frac{\alpha}{|C_\beta|}, \text{if } i \text{ and } j \text{ belong to the cluster } \beta \\ 1 - \alpha + \frac{\alpha}{|C_\beta|}, \text{if } i = j \end{cases}$$

- Element-Centric Similarity (ECS) [5] is a clustering comparison score measured using the L1 distance between the affinity matrices: $S_i(\mathcal{A}, \mathcal{B}) = 1 - \frac{1}{2\alpha} \sum_{j=1}^{N} |p_{ij}^{\mathcal{A}} - p_{ij}^{\mathcal{B}}|$

ECS properties

Optimising the calculation of the ECS score for disjoint partitions

- For disjoint partitions, the ECS has the same value for points from the same clusters.
- The number of ECS calculations drops from $N \times N$ to $n \times m$.
- This optimization removes the affinity matrix dependency and should be both time and memory efficient.

Element-Centric Consistency

- Gates et al. also proposed Element-Centric Consistency (or ECC; also named *frustration*) score, used when a list of multiple partitions is provided.
- The ECC score is calculated as the average of the sum of the ECS between every unique pair of partitions from the list:

$$\frac{2}{T(T-1)} \sum_{j=1}^{T} \sum_{k=1}^{j-1} S_i(\mathcal{R}_k, \mathcal{R}_j)$$

Weighted ECC

- If there are multiple occurrences of the same partition, determining the ECC will require rendundant calculation of the ECS of the same pairs.
- To prevent this, we propose a weighted version of ECC, where each unique partition has attached a weight that denotes the number of duplicates.

Weighted ECC pseudocode

Merging identical partitions

The weights are calculated by performing an automatic merge of identical partitions.

To determine if two partitions are identical, we use their contingency table.

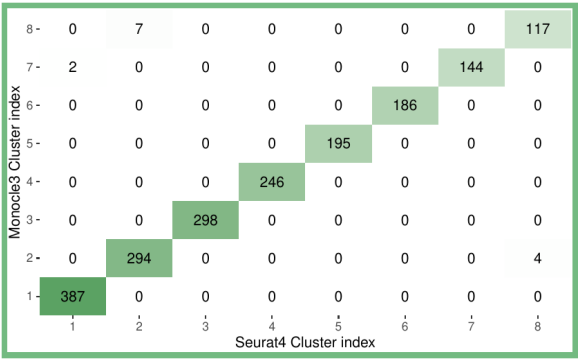

Figure: **Contingency table**. The rows are associated with the clusters from one partition, and the columns with the clusters from the other partitions.

Almost identical partitions - ECS threshold

There can be cases where the difference between two partitions is negligible, as it can be seen in the picture below. To allow the merge of almost identical partitions, we introduced the *ECS threhshold*: the value of this parameter acts as a lower bound and determines whether the partitions should be considered extremely similar and mergeable or not.

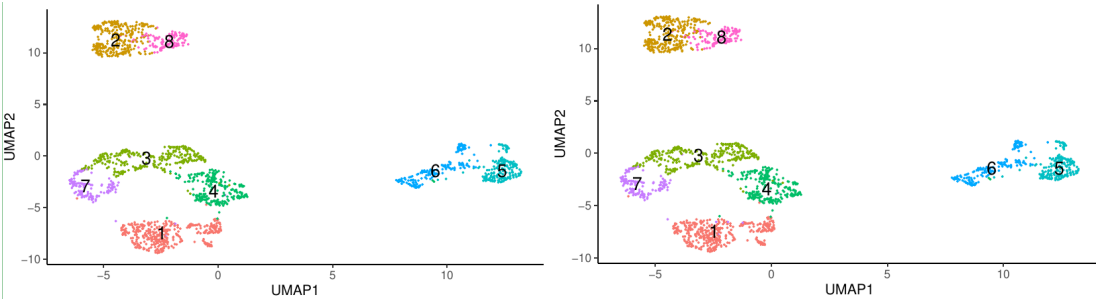

Figure: **Two almost identical partitions** Each panel represents the distribution of a partition on a low-dimensional (UMAP) topology.

Table of Contents

- Random seed is a factor that affects the clustering output.
- We developed a stability pipeline that performs a visual assessment of the robustness of a parameter configuration at the change of seed.
- The robustness will be determined by running the clustering pipeline multiple times with different random seeds. The Element-Centric Consistency of the obtained list of partitions will be an indicator of stability.
- The stability pipeline follows the steps presented in the PhenoGraph algorithm.

Feature stability

The feature set has an important role in obtaining the low-dimensional topology.

We can assess the stability of different feature sets with varying sizes.

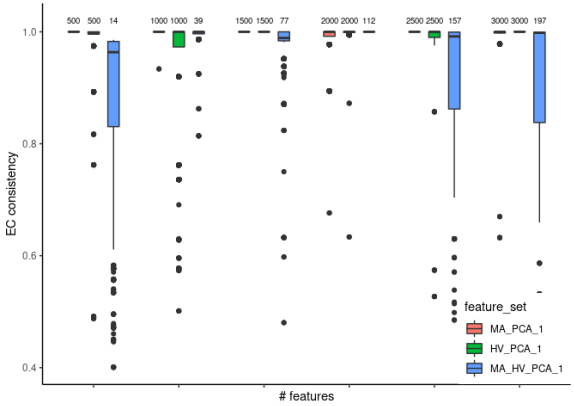

Figure: **Incremental feature stability**. Each colour represents a different feature set. The size of the set is displayed above each boxplot. Each boxplot represents the consistency of the partitions.

Inference of minimum number of clusters

The number of nearest neighbours affect the graph connectivity.

The number of connected components decreases as the number of nearest neighbours increases.

The number of connected components acts as a lower bound for the possible number of clusters.

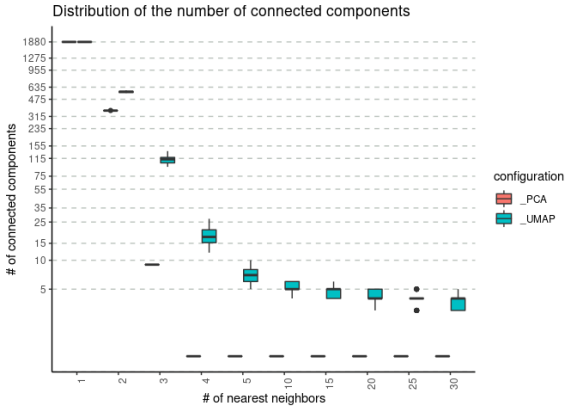

Figure: **Graph connectivity evolution**. Each colour represents a different reduced space used for graph building. The X-axis represents the number of nearest neighbours used. The Y-axis represents the number of connected components.

Resolution - number of clusters stability

For graph clustering methods, the resolution parameter controls the number of clusters.

The stability can be assessed on different parameter configurations.

The colour gradient describes the statistical reliability of the assessment.

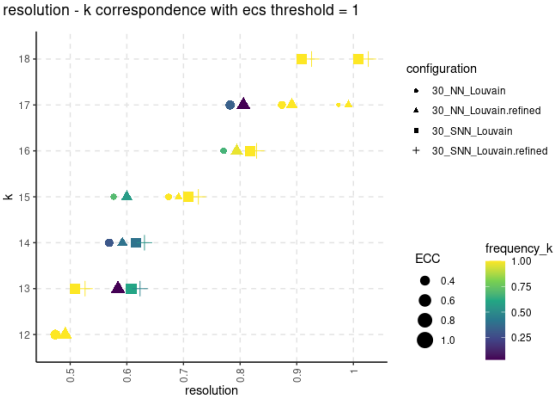

Figure: **Resolution - number of cluster stability**. Each shape of point indicate a different parameter configuration. The point size indicates the stability of the resolution - number of clusters pair. The colour gradient illustrates the frequency of the number of clusters.

Table of Contents

Table of Contents

- To provide an appropriate benchmark of the `ClustAssess` package, we will compare its performance to `Clusim`.

- To provide an appropriate benchmark of the `ClustAssess` package, we will compare its performance to `Clusim`.
- `Clusim` is a Python package that contains the official implementation from the authors of the ECS paper [6].

- To provide an appropriate benchmark of the `ClustAssess` package, we will compare its performance to `Clusim`.
- `Clusim` is a Python package that contains the official implementation from the authors of the ECS paper [6].
- In `Clusim`, the ECS between two disjoint partitions is determined by calculating the affinity matrices.

- To provide an appropriate benchmark of the `ClustAssess` package, we will compare its performance to `Clusim`.
- `Clusim` is a Python package that contains the official implementation from the authors of the ECS paper [6].
- In `Clusim`, the ECS between two disjoint partitions is determined by calculating the affinity matrices.
- The benchmark is done by performing the ECS between two fixed partitions using the implementations from `ClustAssess` and `Clusim`. The size of the partitions varies from 50 to 90 000. The execution is repeated 30 times.

Runtime comparison

`ClustAssess` requires less than a second to calculate the ECS, while the execution time of `Clusim` increases exponentially.

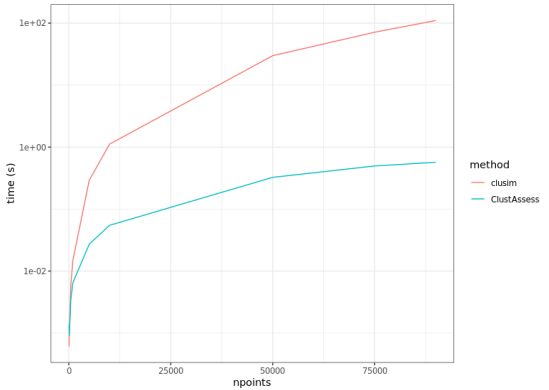

Figure: **Execution time benchmark** The size of the partitions used for the benchmark are displayed on the X-axis. The time (in seconds) required to calculate the ECS using ClustAssess (blue) and Clusim (red) is displayed on the Y-axis. The results are displayed on a logarithmic scale.

Comparison of memory usage

For the optimized version from `ClustAssess` the data storage scales proportionally to the number of clusters.

Memory allocation of $\leq$ 100 MiBs.

The storage of the whole affinity matrix in `Clusim` is inefficient, reaching a memory allocation of 241 GiBs for the largest dataset.

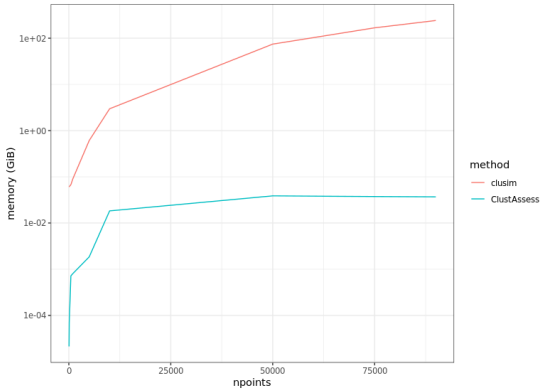

Figure: **Memory usage benchmark** The size of the partitions used for the benchmark are displayed on the X-axis. The amount of memory (in GiBs) required to calculate the ECS using ClustAssess (blue) and Clusim (red) is displayed on the Y-axis. The results are displayed on a logarithmic scale.

Performance with varying number of clusters

- `ClustAssess` calculates a number of unique ECS values proportional to the number of clusters of both partitions.
- It is expected that, as the number of clusters increases, the `ClustAssess` package will require more time and space to calculate the ECS.
- `Clusim` calculates the whole affinity matrix, so its performance shouldn't be affected by the number of clusters.

Performance with varying number of clusters

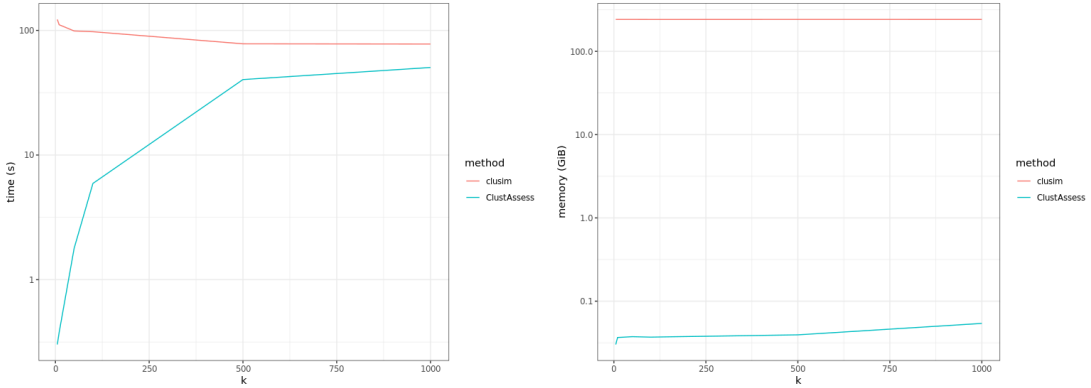

Figure: **Performance comparison for different number of clusters** Left panel - the median time of execution (in seconds) measured after 30 runs for ClustAssess (blue) and Clusim (red). Right panel - the median memory usage (in GiB) measured after 30 runs for ClustAssess (blue) and Clusim (red). The results are displayed on a linear scale.

Table of Contents

How the stability pipeline will be benchmarked

- We will evaluate the performance of the five independent components that provide the data for the plots: `feature_stability`, `nn_n_conn_comps`, `nn_importance`, `clustering_importance`, `resolution_importance`.

How the stability pipeline will be benchmarked

- We will evaluate the performance of the five independent components that provide the data for the plots: `feature_stability`, `nn_n_conn_comps`, `nn_importance`, `clustering_importance`, `resolution_importance`.
- The stability pipeline will be evaluated on the Mende data [7] with subsamples ranging from 1000 to 13000 cells.

How the stability pipeline will be benchmarked

- We will evaluate the performance of the five independent components that provide the data for the plots: `feature_stability`, `nn_n_conn_comps`, `nn_importance`, `clustering_importance`, `resolution_importance`.
- The stability pipeline will be evaluated on the Mende data [7] with subsamples ranging from 1000 to 13000 cells.
- The pipeline allows support for parallelisation, so we will measure the pipeline's performance by varying the number of cores.

How the stability pipeline will be benchmarked

- We will evaluate the performance of the five independent components that provide the data for the plots: `feature_stability`, `nn_n_conn_comps`, `nn_importance`, `clustering_importance`, `resolution_importance`.
- The stability pipeline will be evaluated on the Mende data [7] with subsamples ranging from 1000 to 13000 cells.
- The pipeline allows support for parallelisation, so we will measure the pipeline's performance by varying the number of cores.
- We will also evaluate the impact of using different ECS threshold values: 1, 0.99 and 0.95.

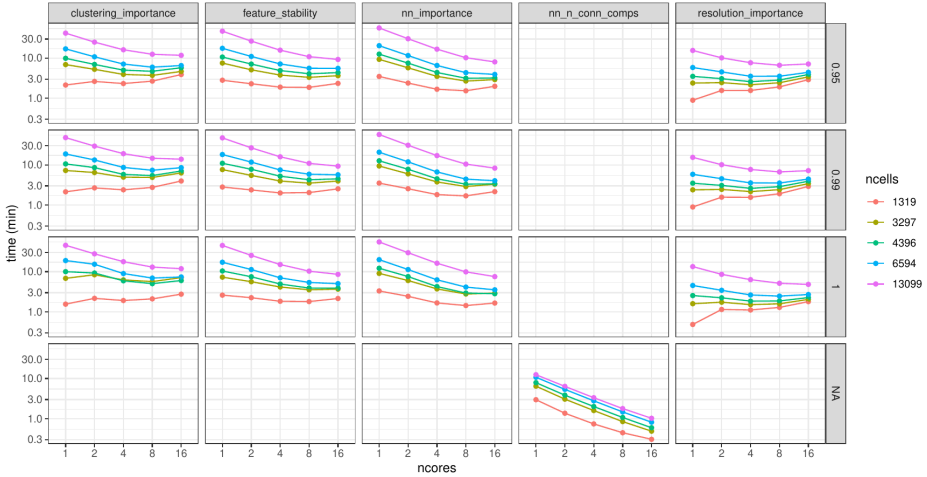

Figure: **Stability pipeline runtime benchmark** The column designate a different component involved in the pipeline. The rows indicate different ECS threshold values. On the X-axis we represent different number of cores. The Y-axis contains the time required (in minutes) for the execution. The colour indicates the size of the dataset.

Table of Contents

Acknwoledgements

The thesis is based on the ClustAssess paper to which I contributed as co-author.
The paper was presented at CSCI 2022 annual retreat and will be presented at ECCB 2022.

References I

[1] Arash Shahsavari, Andi Munteanu, and Irina Mohorianu. Clustassess: tools for assessing the robustness of single-cell clustering. *bioRxiv*, 2022.

[2] Jacob H. Levine, Erin F. Simonds, Sean C. Bendall, Kara L. Davis, El-ad D. Amir, Michelle D. Tadmor, Oren Litvin, Harris G. Fienberg, Astraea Jager, Eli R. Zunder, Rachel Finck, Amanda L. Gedman, Ina Radtke, James R. Downing, Dana Pe'er, and Garry P. Nolan. Data-Driven Phenotypic Dissection of AML Reveals Progenitor-like Cells that Correlate with Prognosis. *Cell*, 162(1):184–197, July 2015.

[3] Yuhan Hao, Stephanie Hao, Erica Andersen-Nissen, William M. Mauck, Shiwei Zheng, Andrew Butler, Maddie J. Lee, Aaron J. Wilk, Charlotte Darby, Michael Zager, Paul Hoffman, Marlon Stoeckius, Efthymia Papalexi, Eleni P. Mimitou, Jaison Jain, Avi Srivastava, Tim Stuart, Lamar M. Fleming, Bertrand Yeung, Angela J. Rogers, Juliana M. McElrath, Catherine A. Blish, Raphael Gottardo, Peter Smibert, and Rahul Satija. Integrated analysis of multimodal single-cell data. *Cell*, 184(13):3573–3587.e29, June 2021.

References II

[4] Junyue Cao, Malte Spielmann, Xiaojie Qiu, Xingfan Huang, Daniel M. Ibrahim, Andrew J. Hill, Fan Zhang, Stefan Mundlos, Lena Christiansen, Frank J. Steemers, Cole Trapnell, and Jay Shendure. The single-cell transcriptional landscape of mammalian organogenesis. *Nature*, 566(7745):496–502, February 2019.

[5] Alexander J. Gates, Ian B. Wood, William P. Hetrick, and Yong-Yeol Ahn. Element-centric clustering comparison unifies overlaps and hierarchy. *Scientific Reports*, 9(1):8574, December 2019.

[6] Alexander J. Gates and Yong-Yeol Ahn. Clusim: a python package for calculating clustering similarity. *Journal of Open Source Software*, 4(35):1264, 2019.

References III

[7] Nicole Mende, Hugo P Bastos, Antonella Santoro, Krishnaa T Mahbubani, Valerio Ciaurro, Emily Francesca Calderbank, Mariana Quiroga Londoño, Kendig Sham, Giovanna Mantica, Tatsuya Morishima, Emily Mitchell, Maria Rosa Lidonnici, Fabienne Meier-Abt, Daniel Hayler, Laura Jardine, Abbie Curd, Muzlifah Haniffa, Giuliana Ferrari, Hitoshi Takizawa, Nicola K Wilson, Bertie Gottgens, Kourosh Saeb-Parsy, Mattia Frontini, and Elisa Laurenti, PhD. Unique molecular and functional features of extramedullary hematopoietic stem and progenitor cell reservoirs in humans. *Blood*, January 2022.

ECS properties

Jump to ECS slide

input : *partList*: the list of disjoint partitions
weights: the weight array that contains the number of duplicates
output : the Element-Centric Consistency of the list

$nPartitions \leftarrow \text{size}(partList)$;
$nTotalPartitions \leftarrow \text{sum}(weights)$; ; // includes duplicates
$ecc \leftarrow \mathbf{0}$;
for $i = 1$ **to** $nPartitions - 1$ **do**
  for $j = i + 1$ **to** $nPartitions$ **do**
    $ecc \leftarrow ecc + \text{ecs}(partList[i], partList[j]) * weights[i] * weights[j]$;
  end
end
$nDuplicatesECS \leftarrow 0$;
for $i = 1$ **to** $nPartitions$ **do**
  $nDuplicatesECS \leftarrow nDuplicatesECS + weights[i] * (weights[i] - 1)/2$;
end
$ecc \leftarrow ecc + nDuplicatesECS * \mathbf{1}$;
$ecc \leftarrow ecc/(nTotalPartitions * (nTotalPartitions - 1)/2)$;
return *ecc*

Jump to Weighted ECC slide