Hypercluster: a flexible tool for parallelized unsupervised clustering optimization

Lili Blumenberg[1,2], Kelly V. Ruggles[1,2]

[1]Institute of Systems Genetics, New York University School of Medicine, New York, NY 10016 [2]Department of Medicine, New York University School of Medicine, New York, NY 10016

Abstract

Unsupervised clustering is a common and exceptionally useful tool for large biological datasets. However, clustering requires the selection of an algorithm and hyperparameters, which can introduce bias into the final clustering labels. It is therefore common and advisable to obtain a range of clustering results from multiple models and hyperparameters, which is cumbersome and slow. To aid this process, we present hypercluster, a python package and SnakeMake pipeline for flexible and parallelized clustering evaluation and selection. Hyptercluster is available on pip and bioconda; installation and usage documentation can be found at: https://github.com/liliblu/hypercluster.

Introduction

Unsupervised clustering is commonly used for the interpretation of ‘omics datasets. It provides an objective and intuitive measure of similarity and difference between samples. Clustering can be used to determine biologically relevant subgroups of samples, find co-regulated molecular features, or provide objective support for the phenotypic similarity of biological perturbations. Clustering is a key step in the analysis of many emerging technologies; for instance a fundamental challenge in the analysis of single-cell measurement data is determining robust clusters of phenotypically similar cells (1–3). Additionally clustering is increasingly being used alongside traditional diagnostic techniques to establish new classifications of patient samples into disease-relevant subgroups (4–7) and is increasingly used for patient subgroup classification and risk stratification (6,8–12). The near-future of personalized medicine depends on researchers identifying unsupervised clustering-based disease subtypes. Therefore, it is essential that high-quality clustering results are easily and robustly obtainable, without user-selected hyperparameters introducing bias and impeding rapid analysis.

Currently, researchers robustly employing unsupervised clustering must choose specific algorithms and hyperparameters that are appropriate to their experiment type and data. Although some efforts have been made to advise researchers on optimal selection of both (13), biological datasets vary between batches, days, labs and researchers, underlining the importance of context- and experiment-dependent analysis

tuning. Software packages for automatic hyperparameter tuning and model selection for regression and classification machine learning techniques do exist, notably auto-sklearn from AutoML (14), but there are not yet published packages for automated unsupervised clustering optimization.

Usually the effect of hyperparameter choice on the quality of clustering results cannot be described with a convex function, meaning that when searching the landscape of hyperparameter choices, there are often local maxima that may appear to be optimal results if broad choices of hyperparameters are not considered. Therefore it is unlikely that a sequential approach which uses, for instance, gradient descent from a single initialized set of hyperparameters, will produce optimal results for most clustering problems (15). Exhaustive (i.e. grid) search is the most likely to obtain optimal results from unsupervised clustering.

Here we present hypercluster, a python package and SnakeMake pipeline for parallelized clustering calculations and comparison. The hypercluster package allows users to calculate results from many hyperparameters of one or many algorithms in series, and easily calculate and visualize evaluation metrics (19) for each result. The accompanying SnakeMake pipeline allows parallelization on a single computer, across a high performance computing cluster, or on cloud based services (16,17), speeding up optimization, especially for large datasets. In addition, our pipeline has all the advantages of the SnakeMake framework, e.g. easily adding new datasets to analyze, keeping track of progress and simplified bug tracking. Currently, hypercluster can compare all clustering algorithms and evaluation metrics from scikit-learn (18). In

addition, hypercluster can be extended to calculate user-supplied clustering algorithm or evaluation metrics. Given a metric to maximize, hypercluster identifies “best” labels and optionally provides comparisons of labeling results. As far as we are aware, hypercluster provides researchers the first and only pipeline for flexible, parallelized, distributed and user-friendly algorithm selection and hyper-parameter tuning for unsupervised clustering.

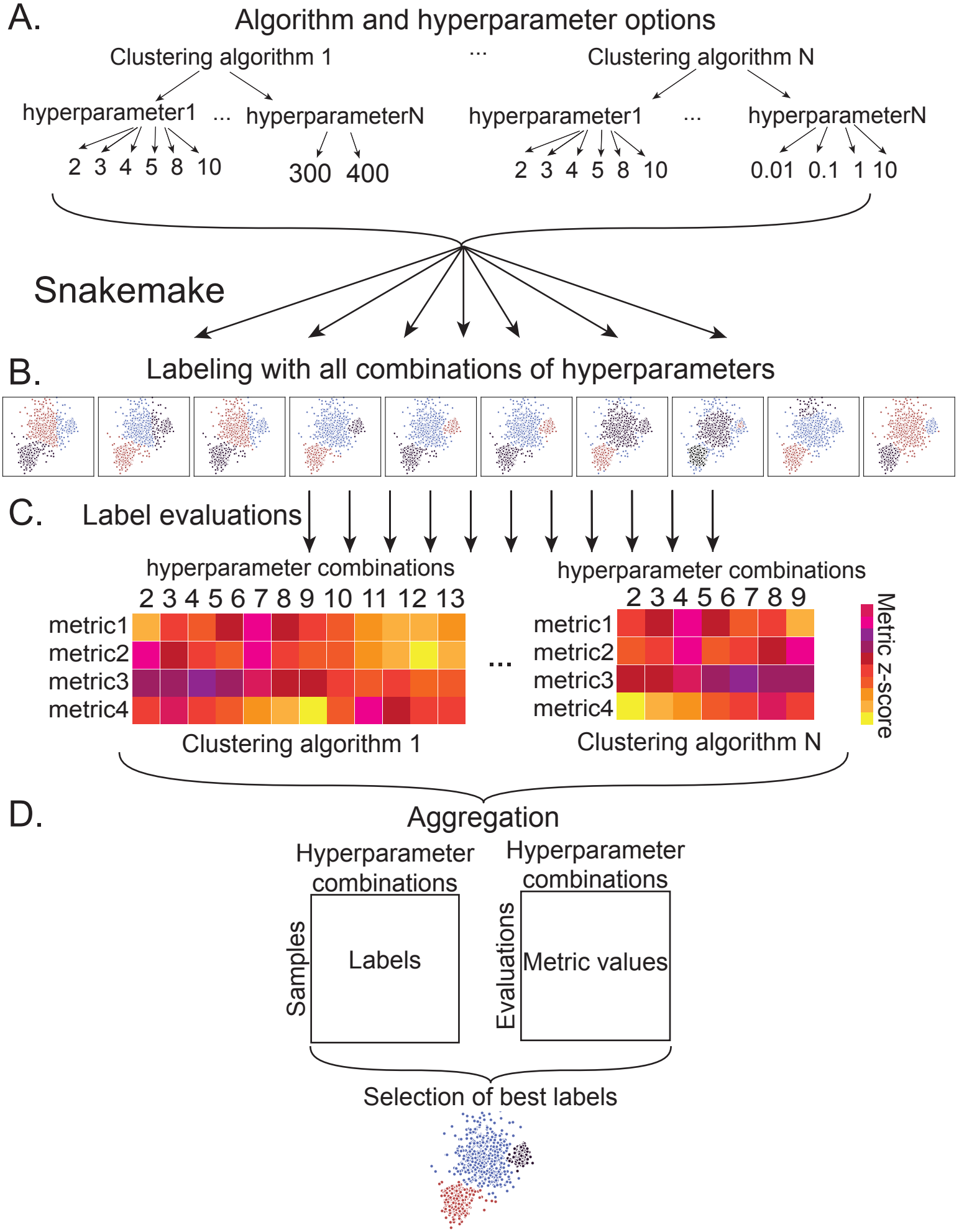
Design and implementation

The hypercluster package uses scikit-learn (18) and optionally SnakeMake (16) to calculate and compare clustering algorithms and hyperparameters to find optimal clusters for any given input data (Fig. 1). Hypercluster requires python3, pandas (20), numpy (21), scipy (22), matplotlib (23), seaborn (24), scikit-learn (18) and SnakeMake (16). Hypercluster can also be run independently of SnakeMake, simply as a standalone python package. Inputs, outputs and an example workflow are described below, but additional example workflows, including how to add custom clustering algorithms and evaluation metrics are provided at https://github.com/liliblu/hypercluster/examples.

Inputs

For local running, AutoClusterer and MultiAutoClusterer objects can be instantiated with default or user-defined values. To run through hyperparameters for a dataset, users simply provide a pandas DataFrame to the “fit” method on either object. Users evaluate the labeling results by running the “evaluate” method.

Fig. 1: Workflow



config.yml

SnakeMake allows users to parallelize clustering calculations. To configure the SnakeMake pipeline, users edit a config.yml file (Table 1). In that file, users can specify input and output folders (Table 1, lines 2-3, 6-8) and the hyperparameter search space (Fig 1A, Table 1, line 18). Users can specify whether to use exhaustive grid search or random search; if random search is selection, they can specify probability weights for each hyperparameter (Table 1, line 9). Snakemake then schedules each clustering algorithm as a separate job (Fig. 1B). Users can specify which evaluation metrics to apply (Fig. 1C, Table 1, line 10) and add keyword arguments to tune several steps in the process (Table 1, lines 4, 8-9, 11-16). Clustering and evaluation results are then aggregated into final tables (Fig. 1D). However, other than the location and names of the input files, everything has a predefined default that allows the pipeline to be used "out of the box." Users can reference the documentation and examples for more information.

Input data and execution

After specifying the config.yml file, users provide a data table with samples to be clustered as the rows and features as the columns. Users can then simply run "snakemake -s hypercluster.smk" in the command line, with any additional SnakeMake flags appropriate for their system. Applying the same configuration to new files or adjusting algorithms and hyperparameter options simply requires editing the config.yml

Table 1

	config.yml parameter	Explanation	Example from scRNA-seq workflow
1	input_data_folder	Path to folder in which input data can be found. No / at the end.	/input_data
2	input_data_files	List of prefixes of data files. Exclude extension, .csv, .tsv and . txt allowed.	['input_data1', 'input_data2']
3	gold_standard_file	File name of gold_standard_file. Must have same pandas. read_csv kwargs as the corresponding input file. Must be in input_data_folder.	{'input_data': 'gold_standard_file.txt'}
4	read_csv_kwargs	Per input data file, keyword args to put into pandas.read_csv. If specifying multiindex, also put the same in output_kwargs [‘labels’]	{'test_input': {'index_col':[0]}}
5	output_folder	Path to folder in which results will be written. No / at the end.	/hypercluster_results
6	intermediates_folder	Name of the folder within the output_folder to put intermediate results, such as labels and evaluations per condition. No need to change this usually.	clustering_intermediates
7	clustering_results	Name of the folder within the output_folder to put final results. No need to change this usually.	clustering
8	clusterer_kwargs	Additional static keyword arguments to pass to individual clusterers, not for optimization	KMeans: {'random_state':8}}
9	generate_parameters_addtl_kwargs	Additonal keyword arguments for the hypercluster. AutoClusterer class.	{‘KMeans’: {‘random_search’:true, ‘param_weights’: {‘n_clusters’: {5: 0.25, 6: 0.25, 7: 0.5}}}
10	evaluations	Names of evaluation metrics to use. See hypercluster.constants.inherent_metrics or hypercluster.constants.need_ground_truth	['silhouette_score', 'number_clustered']
11	eval_kwargs	Additional kwargs per evaluation metric function.	{'silhouette_score': {'random_state': 8}}
12	metric_to_choose_best	If picking best labels, which metric to maximize to choose the labels. If not choosing best labels, leave as empty string (‘’).	silhouette_score
13	metric_to_compare_labels	If comparing labeling result pairwise similarity, which metric to use. To not generate this comparison, leave blank/or empty string.	adjusted_rand_score
14	compare_samples	Whether to made a table and figure with counts of how often two samples are in the same cluster.	TRUE
15	output_kwargs	pandas.to_csv and pandas.read_csv kwargs per output type. Generally, don’t need to change the evaluations kwargs, but labels index_col have to match index_col like in the read_csv_kwargs.	{'evaluations': {'index_col':[0]}, 'labels': {'index_col':[0]}}
16	heatmap_kwargs	Additional kwargs for seaborn.heatmap for visualizations.	{'vmin':-2, 'vmax':2}
17	optimization_parameters	Fun part! This is where you put which hyperparameters per algorithm to try.	{'KMeans': {'n_clusters': [5, 6, 7]}}

file and rerunning SnakeMake; SnakeMake will detect which files to create and which have already been generated.

Extending hypercluster

Currently, hypercluster can optimize any clustering algorithm and calculate any evaluation available in scikit-learn (18,26). Additional clustering class instances and evaluation metrics can also be added by users in the additional_clusterer.py and additional_metrics.py files, respectively, if written to accommodate the same input (see examples).

Outputs

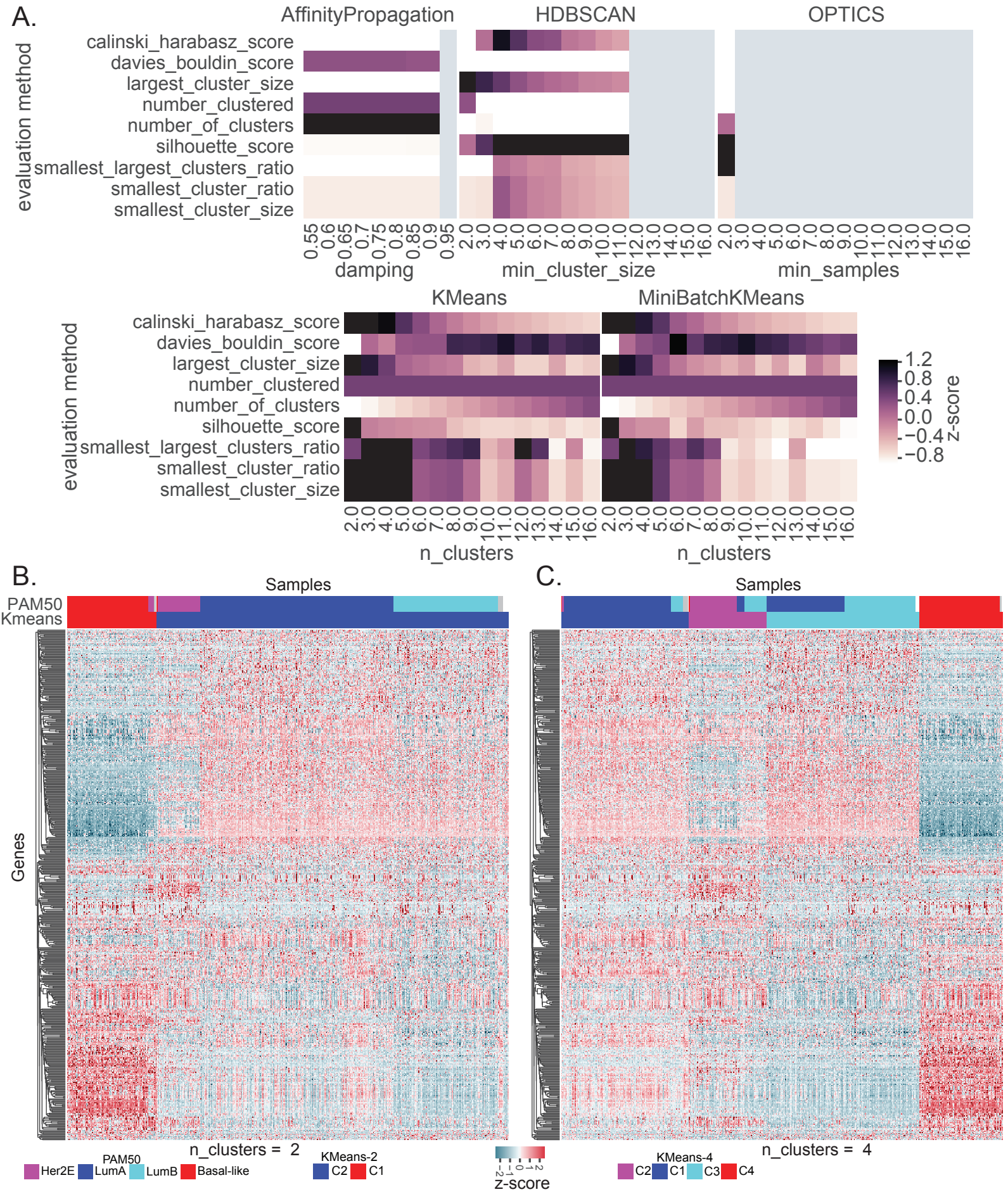
By default, hypercluster outputs a yaml file containing all configurations of the clustering algorithms and hyperparameters that are being searched. For each configuration, it generates a file containing labels and a file containing all specified evaluations. It also outputs aggregated tables of all labels and evaluations. Finally, given a metric to maximize, hypercluster writes files containing the “best” labels. Optionally, hypercluster will also output a table and heatmap of pairwise comparisons of labeling similarities with a user-specified metric (Figure S1). It can also optionally output a table and heatmap showing how often each pair of samples were assigned the same cluster (Figure S2).

Example workflow

Unsupervised clustering of RNA-seq on patient samples

To illustrate the utility of hypercluster in a disease-relevant context, we applied our method to RNA-seq data from 526 breast cancer patient samples from the Cancer Genome Atlas (TCGA) (27). RNA-seq can be used to classify breast cancer patients into four major PAM50 subtypes (Basal-like, LuminalA, LuminalB, and Her2-enriched), which are based on the expression of 50 particular genes (7,28,29). We applied 120 different hyperparameter conditions to find optimal clustering of samples using the most variable genes to compare with results from the PAM50 classifier. The whole workflow can be found at https://github.com/liliblu/hypercluster/examples. Briefly, RNA-seq data was downloaded from (27). That data has already been normalized per sample and log transformed. Genes with missing values were removed, and the top 500 genes with the highest standard deviation were retained. The data was then z-scored per gene. We used the config.yml file to point SnakeMake to our input files, and left the 120 default hyperparameter combinations. Hypercluster outputs a visualization of evaluation metrics for all hyperparameter combinations (Fig. 2A), which allows users to quickly see how changing hyperparameters affects clustering result quality. These results highlight how evaluation metrics are not generally convex over ranges of hyperparameters, highlighting the utility of exhaustive grid search approach. In addition, the pipeline optionally creates a pairwise comparison of labeling, with a specified user metric. An

Figure 2


A.
AffinityPropagation
HDBSCAN
OPTICS
evaluation method
calinski_harabasz_score
davies_bouldin_score
largest_cluster_size
number_clustered
number_of_clusters
silhouette_score
smallest_largest_clusters_ratio
smallest_cluster_ratio
smallest_cluster_size
0.55 0.6 0.65 0.7 0.75 0.8 0.85 0.9 0.95
damping
2.0 3.0 4.0 5.0 6.0 7.0 8.0 9.0 10.0 11.0 12.0 13.0 14.0 15.0 16.0
min_cluster_size
min_samples
KMeans
MiniBatchKMeans
n_clusters
1.2 0.8 0.4 0.0 −0.4 −0.8
z-score
B.
C.
Samples
PAM50
Kmeans
Genes
n_clusters = 2
n_clusters = 4
PAM50
Her2E LumA LumB Basal-like
KMeans-2
C2 C1
KMeans-4
C2 C1 C3 C4
z-score

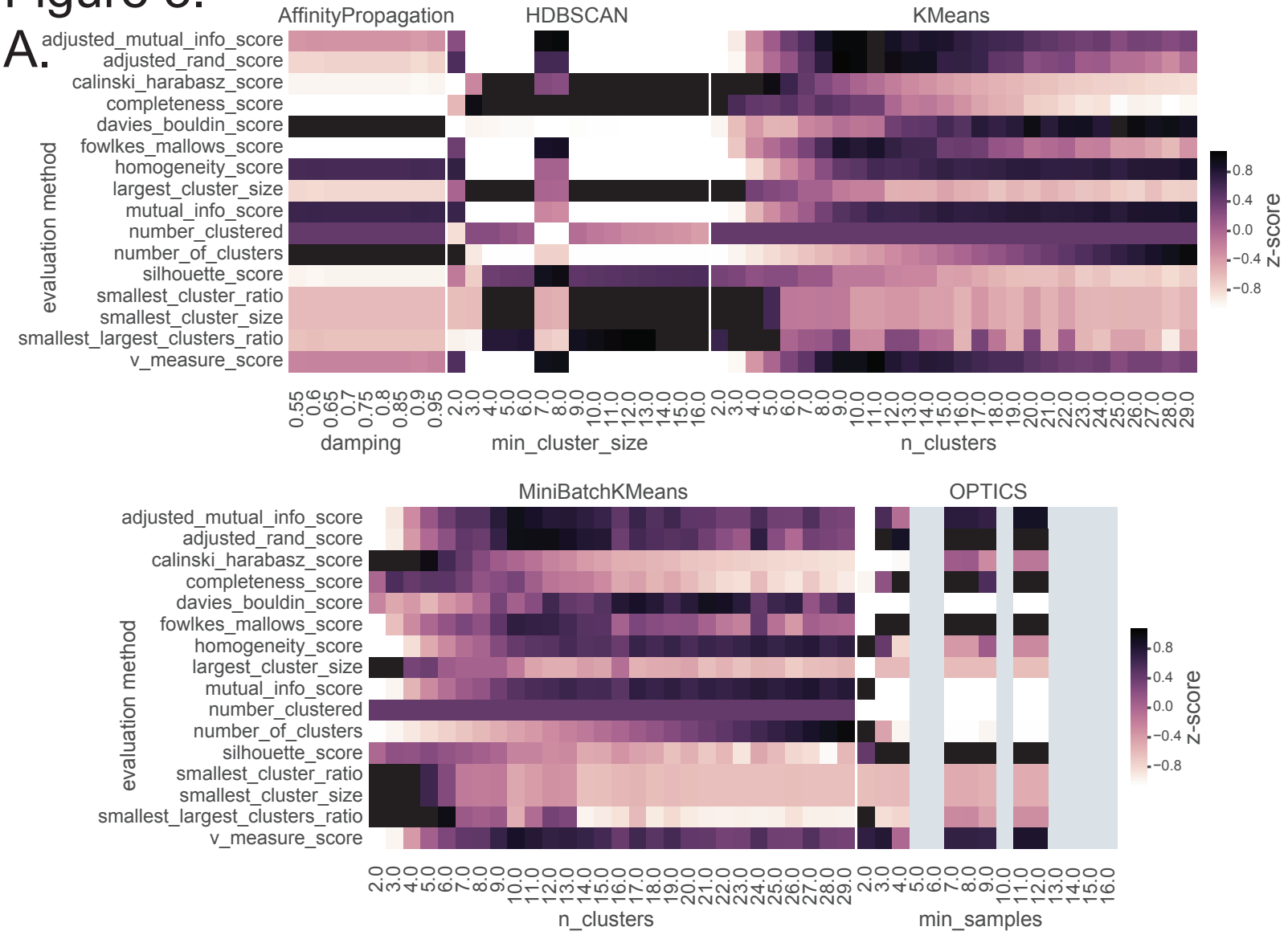
example from this workflow using the adjusted rand index as the metric is in Figure S1A.

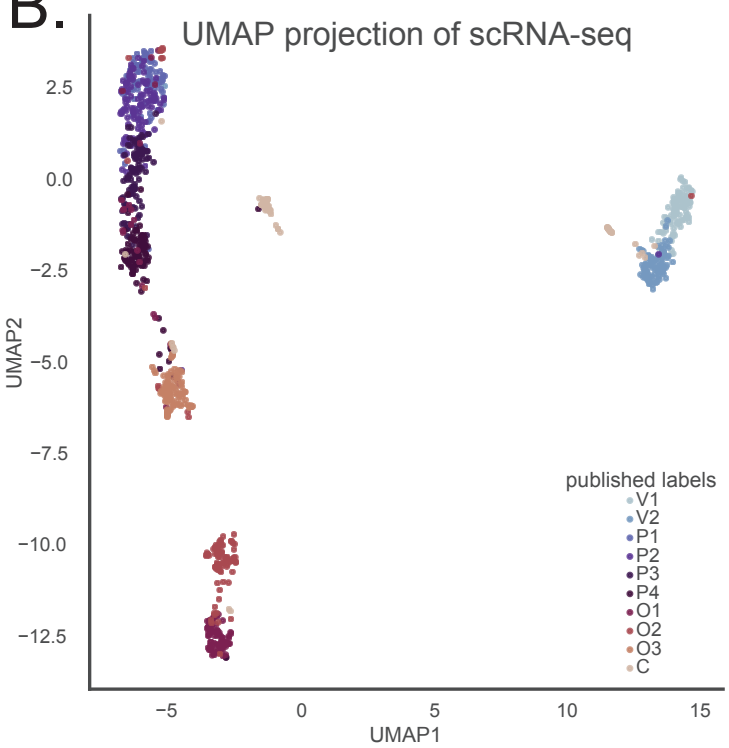
We chose the highest quality clustering by first filtering for conditions which labeled all samples, which is important when classifying patient samples as some algorithms discard poorly fitting samples. We then chose labels with the highest silhouette score: KMeans with 2 clusters (Fig. 2B). This clustering largely separated samples into Basal-like and non-Basal-like which is with breast cancer literature, as basal-like breast cancer is known to have an extremely different molecular phenotype from the other subtypes, likely due to the lack of hormone receptor expression (30,31). Hypercluster retains all clustering results; it is therefore simple to observe other results. To see if more granular subtypes match with the PAM50 classification, we visualized results from KMeans run with 4 groups. These labels largely matches PAM50 classification (adj rand score = 0.44), with the exception of a subset of PAM50 Luminal A samples that cluster with Luminal B samples (Fig. 2C). Comparison of sample metadata and clustering results is extremely easy with hypercluster; this simple example highlights how hypercluster allows rapid and parallelized clustering optimization which allows users to find biologically relevant patterns in high dimensional data.

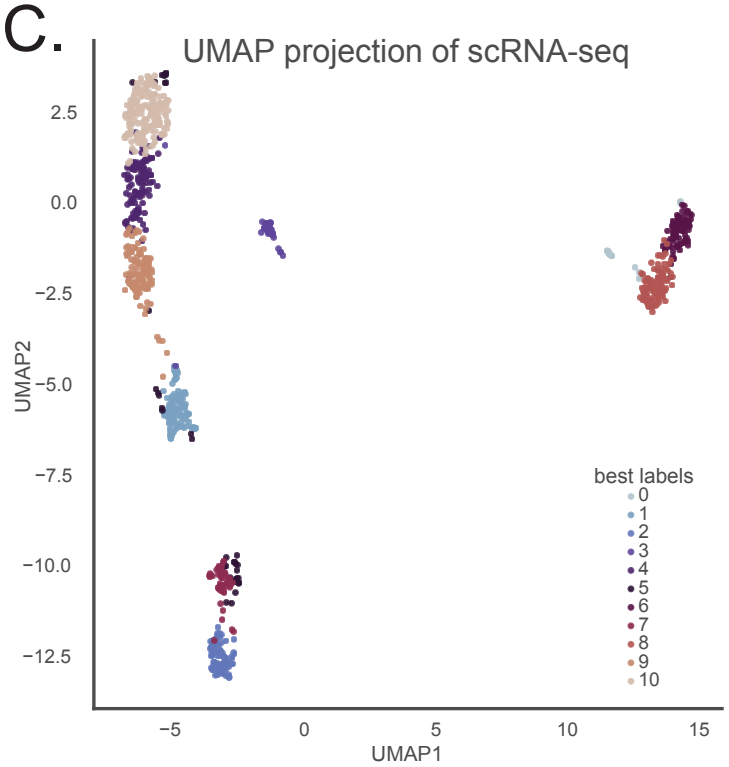
Exploration of bone marrow microenvironment scRNA-seq

We obtained single cell RNA seq data from a study investigating the hematopoietic stem cell microenvironment (32) and performed comparative analysis of several clustering algorithms in parallel on a high performance computing cluster with

Figure 3.


A.
AffinityPropagation
HDBSCAN
KMeans
MiniBatchKMeans
OPTICS
evaluation method
adjusted_mutual_info_score
adjusted_rand_score
calinski_harabasz_score
completeness_score
davies_bouldin_score
fowlkes_mallows_score
homogeneity_score
largest_cluster_size
mutual_info_score
number_clustered
number_of_clusters
silhouette_score
smallest_cluster_ratio
smallest_cluster_size
smallest_largest_clusters_ratio
v_measure_score
damping
min_cluster_size
n_clusters
min_samples
z-score


B.
UMAP projection of scRNA-seq
UMAP1
UMAP2
published labels
V1
V2
P1
P2
P3
P4
O1
O2
O3
C


C.
UMAP projection of scRNA-seq
UMAP1
UMAP2
best labels
0
1
2
3
4
5
6
7
8
9
10

the Slurm scheduler (Yoo et al. 2003) (see examples). We downloaded supplementary table 1 from that study, which contains the normalized expression of genes characteristic to each cluster identified in the study. We then used hypercluster to determine which method and hyperparameters would best recover the labels achieved with graph-based clustering in that study. Hypercluster was then run with all default parameters, measuring the labels with every available evaluation (Fig. 3A). We also compared how similar all labeling outputs were to each other, which is an optional output of the pipeline (Figure S1B). We chose the algorithm and hyperparameters that resulted in the most similar labels to published labels (Fig. 3B) using the adjusted rand score, visualized in Fig. 3C. This example workflow is available in the github examples repository as well.

Discussion

Defining groups of molecularly similar patient samples is key to personalizing safe and effective treatments making unsupervised clustering a workhorse for researchers advancing personalized medicine. It is therefore essential that unsupervised clustering is rigorous and not biased by arbitrary hyperparameter selection. While extremely high quality open-source tools such as scikit-learn make unsupervised clustering accessible to many, exhaustively and reproducibly comparing hyperparameters is still challenging; hypercluster solves these issues.

Almost every step in data analysis pipelines require some hyperparameter selection, during which biased or arbitrary parameter selection can affect results.

Preprocessing data, during which researchers filter datasets to remove low quality or low coverage samples or features (e.g. removing genes with very few reads in RNA-seq). A subpackage of AutoML, auto-sklearn (14) provides functionality for automating pre-processing of data tables, which could easily be incorporated upstream of hypercluster. Snakemake is already widely used for optimizing pipelines (33–39). If clustering is a downstream analysis of interest, determining which parameters to select will be cumbersome, and possibly inaccurate without hypercluster. This package allows users to attach clustering to their SnakeMake or python workflows, increasing reproducibility and preventing arbitrary hyperparameter choices. In addition to helping researchers choose hyperparameters, hypercluster will aid computational biologists who are benchmarking new clustering algorithms, evaluation metrics and pre- or post-processing steps (3).

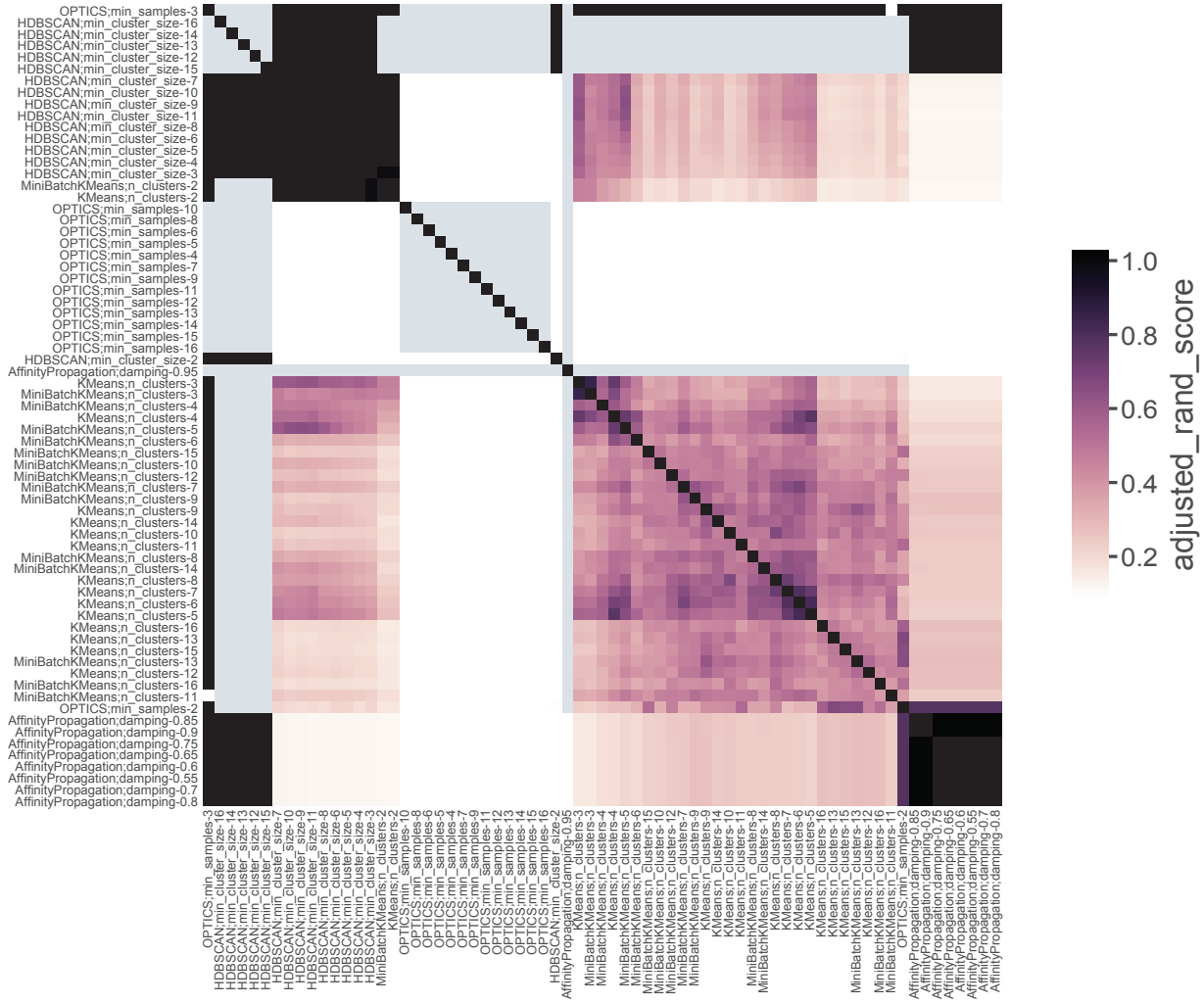
Hypercluster distributes clustering calculations in parallel and aggregates results using SnakeMake, making it is extremely easy for users to add additional datasets, (e.g. a single dataset preprocessed in a variety of ways), augment hyperparameters to test and investigate several evaluation metrics to choose optimal clustering. Hypercluster streamlines the use of unsupervised clustering to derive biologically relevant structure within data. Most importantly, it eases the prioritization of rigor and reproducibility for researchers using these techniques.

Ethics approval and consent to participate

Not Applicable

Figure S1

A.


adjusted_rand_score
1.0
0.8
0.6
0.4
0.2

B.

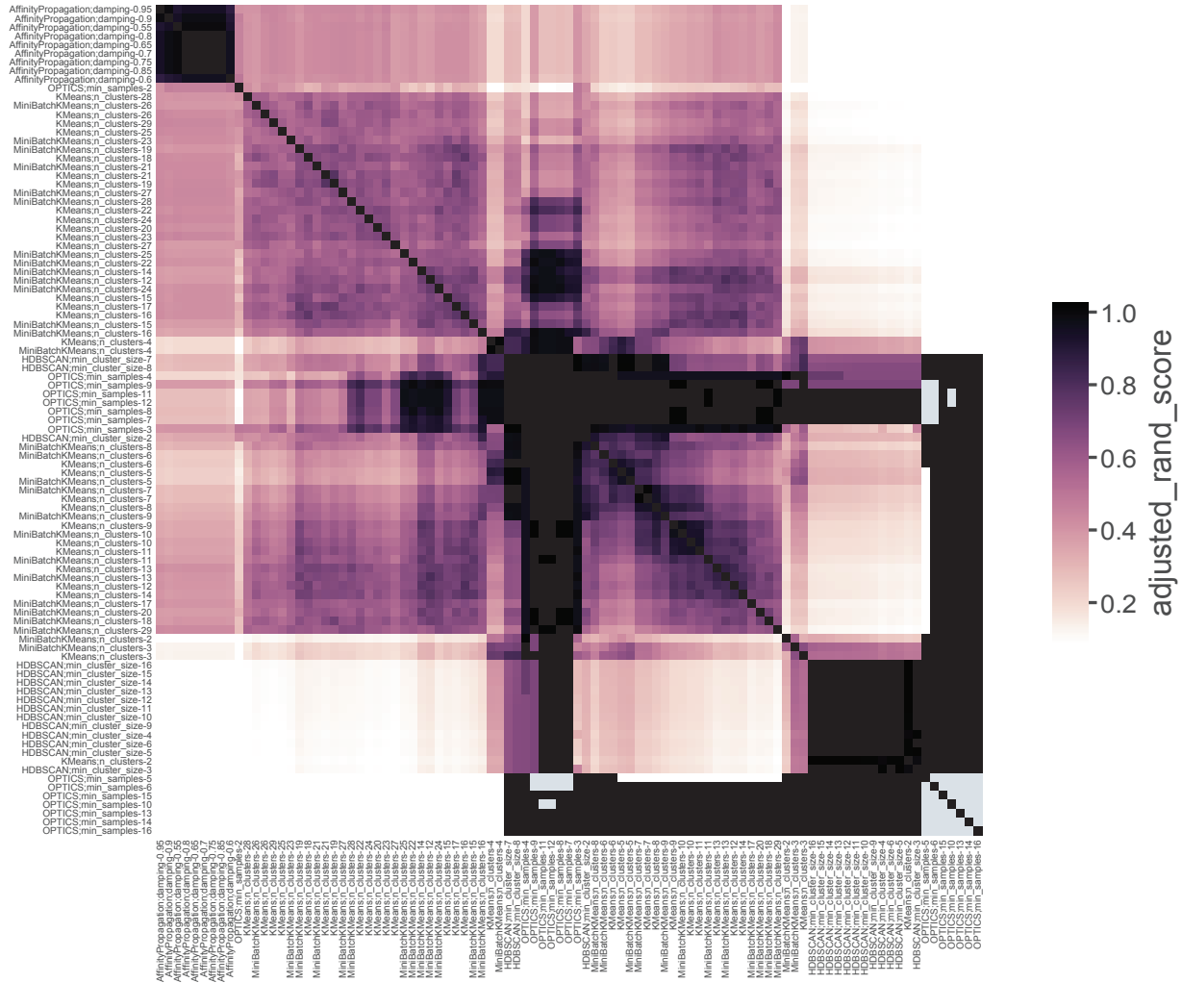
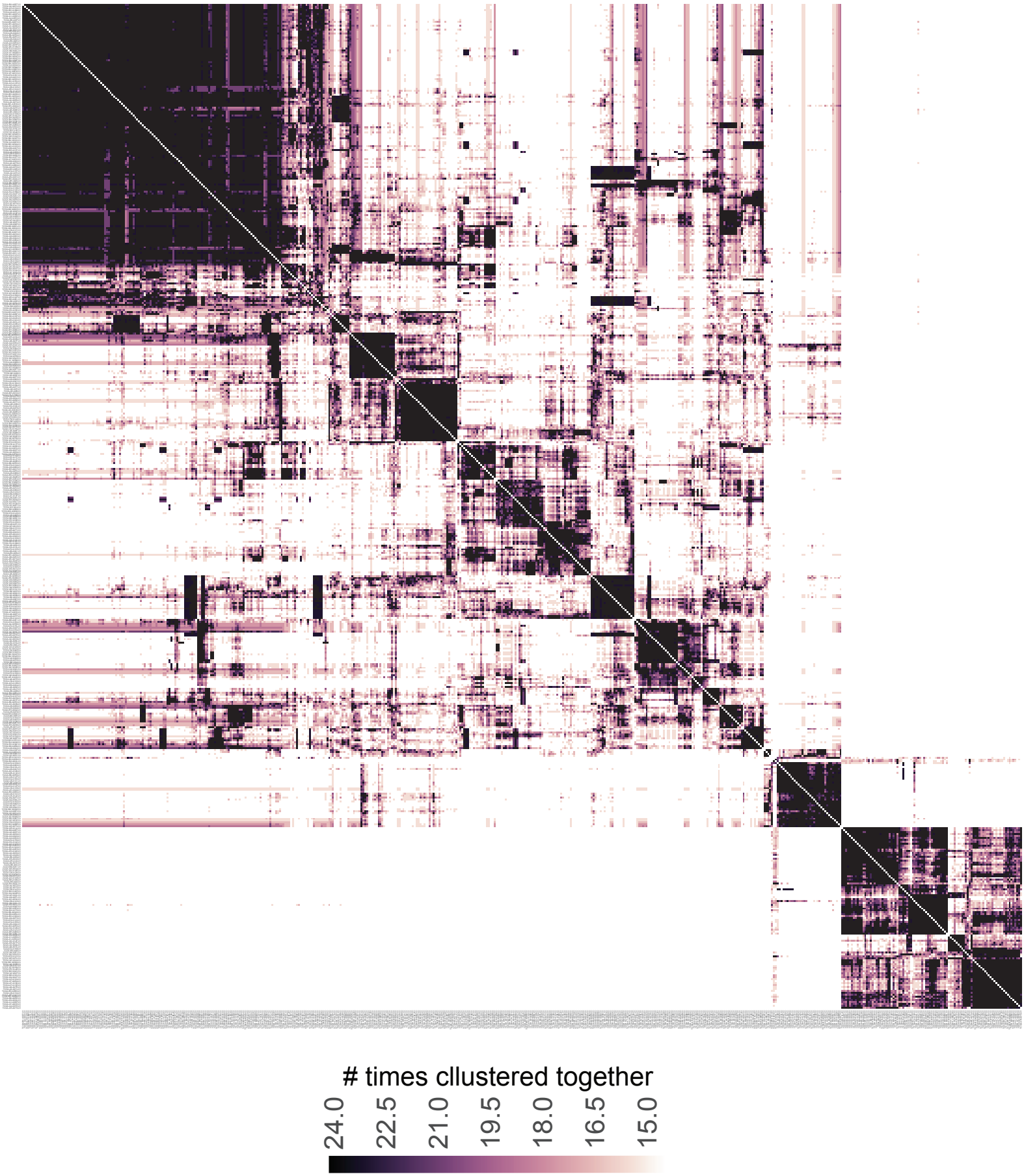

adjusted_rand_score
1.0
0.8
0.6
0.4
0.2

Figure S2



Consent for publication

Not Applicable

Availability of data and materials

Hypercluster is released on pip (pip install hypercluster) and conda (conda install -c bioconda hypercluster). Development versions and installation instructions can be found at our github (https://github.com/liliblu/hypercluster/), tutorials and examples, including data, can be found here: https://github.com/liliblu/hypercluster/examples, and documentation can be found here: https://hypercluster.readthedocs.io/en/latest/. Hypercluster is written in python and was developed and tested on MacOS and Linux. Requirements are listed on the github and in the documentation. Hypercluster is available with the MIT licence.

Competing interests

The authors declare no competing interests.

Funding

This work has been supported by the National Cancer Institute (NCI) through CPTAC award U24 CA210972

Authors' contributions

LB and KVR conceived of the ideas and wrote the manuscript. LB developed the method and wrote the code. KVR supervised the project.

Acknowledgements

We thank the members of Ruggles and Fenyö labs for their helpful discussions and input. We would like to thank MacIntosh Cornwell for his advice with the SnakeMake pipeline.

Figure legends

Fig. 1 Diagram of hypercluster's workflow.

a) Clustering algorithms and their respective hyperparameters are user-specified. Hypercluster then uses those combinations to create exhaustive configurations, and if selected a random subset is chosen. b) Snakemake is then used to distribute each clustering calculation into different jobs. c) Each set of clustering labels is then evaluated in a separate job by a user-specified list of metrics. d) Finally all clustering results and evaluation results are aggregated into tables. Best labels can also be chosen by a user-specified metric.

Fig. 2

a) Example automatic output from hypercluster, showing z-scored evaluation values of evaluation metrics for each clustering algorithm and hyperparameter set. Evaluations

applied to clustering from TCGA breast cancer samples. b, c) Clustered heatmap showing z-scores log2 gene expression data from top 500 most variable genes and TCGA samples labeled by b) KMeans;n_clusters-2 and c) KMeans;n_clusters-4. Top row indicates PAM50 labeling, second row indicates KMeans labeling.

Fig. 3

a) Evaluation metrics for clustering conditions, automatically generated by hypercluster for single cell RNA-seq data UMAP embedding of single cells with b) published labels and c) best KMeans clustering.

Table 1

Line-by-line explanation of a toy config.yml file snakemake.

Figure S1

a, b) Example pairwise comparison of labeling automatically generated using hypercluster of a) breast cancer samples b) and single cell RNA-seq results.

Figure S2

Example pairwise comparison of BRCA RNA-seq sample labeling similarity, automatically generated using hypercluster.

References

1. Kiselev VY, Andrews TS, Hemberg M. Publisher Correction: Challenges in unsupervised

clustering of single-cell RNA-seq data. Nat Rev Genet [Internet]. 2019 May;20(5):310. Available from: http://dx.doi.org/10.1038/s41576-019-0095-5

2. Sun S, Zhu J, Ma Y, Zhou X. Accuracy, robustness and scalability of dimensionality reduction methods for single-cell RNA-seq analysis [Internet]. Vol. 20, Genome Biology. 2019. Available from: http://dx.doi.org/10.1186/s13059-019-1898-6

3. Liu X, Song W, Wong BY, Zhang T, Yu S, Lin GN, et al. A comparison framework and guideline of clustering methods for mass cytometry data. Genome Biol [Internet]. 2019 Dec 23;20(1):297. Available from: http://dx.doi.org/10.1186/s13059-019-1917-7

4. Parker JS, Mullins M, Cheang MCU, Leung S, Voduc D, Vickery T, et al. Supervised risk predictor of breast cancer based on intrinsic subtypes. J Clin Oncol [Internet]. 2009 Mar 10;27(8):1160–7. Available from: http://dx.doi.org/10.1200/JCO.2008.18.1370

5. Ohnstad HO, Borgen E, Falk RS, Lien TG, Aaserud M, Sveli MAT, et al. Prognostic value of PAM50 and risk of recurrence score in patients with early-stage breast cancer with long-term follow-up. Breast Cancer Res [Internet]. 2017 Nov 14;19(1):120. Available from: http://dx.doi.org/10.1186/s13058-017-0911-9

6. Ali HR, Rueda OM, Chin S-F, Curtis C, Dunning MJ, Aparicio SA, et al. Genome-driven integrated classification of breast cancer validated in over 7,500 samples. Genome Biol [Internet]. 2014 Aug 28;15(8):431. Available from: http://dx.doi.org/10.1186/s13059-014-0431-1

7. Perou CM, Sørlie T, Eisen MB, van de Rijn M, Jeffrey SS, Rees CA, et al. Molecular portraits of human breast tumours. Nature [Internet]. 2000 Aug 17;406(6797):747–52.

Available from: http://dx.doi.org/10.1038/35021093

8. Capper D, Jones DTW, Sill M, Hovestadt V, Schrimpf D, Sturm D, et al. DNA methylation-based classification of central nervous system tumours. Nature [Internet]. 2018 Mar 22;555(7697):469–74. Available from: http://dx.doi.org/10.1038/nature26000

9. Sturm D, Orr BA, Toprak UH, Hovestadt V, Jones DTW, Capper D, et al. New Brain Tumor Entities Emerge from Molecular Classification of CNS-PNETs. Cell [Internet]. 2016 Feb 25;164(5):1060–72. Available from: http://dx.doi.org/10.1016/j.cell.2016.01.015

10. Hoadley KA, Yau C, Hinoue T, Wolf DM, Lazar AJ, Drill E, et al. Cell-of-Origin Patterns Dominate the Molecular Classification of 10,000 Tumors from 33 Types of Cancer. Cell [Internet]. 2018 Apr 5;173(2):291–304.e6. Available from: http://dx.doi.org/10.1016/j.cell.2018.03.022

11. Aure MR, Vitelli V, Jernström S, Kumar S, Krohn M, Due EU, et al. Integrative clustering reveals a novel split in the luminal A subtype of breast cancer with impact on outcome. Breast Cancer Res [Internet]. 2017 Mar 29;19(1):44. Available from: http://dx.doi.org/10.1186/s13058-017-0812-y

12. Curtis C, Shah SP, Chin S-F, Turashvili G, Rueda OM, Dunning MJ, et al. The genomic and transcriptomic architecture of 2,000 breast tumours reveals novel subgroups. Nature [Internet]. 2012 Apr 18;486(7403):346–52. Available from: http://dx.doi.org/10.1038/nature10983

13. Jaskowiak PA, Costa IG, Campello RJGB. Clustering of RNA-Seq samples: Comparison study on cancer data. Methods [Internet]. 2018 Jan 1;132:42–9. Available from:

http://dx.doi.org/10.1016/j.ymeth.2017.07.023

14. Feurer M, Klein A, Eggensperger K, Springenberg J, Blum M, Hutter F. Efficient and Robust Automated Machine Learning. In: Cortes C, Lawrence ND, Lee DD, Sugiyama M, Garnett R, editors. Advances in Neural Information Processing Systems 28 [Internet]. Curran Associates, Inc.; 2015. p. 2962–70. Available from: http://papers.nips.cc/paper/5872-efficient-and-robust-automated-machine-learning.pdf

15. Barber RF, Ha W. Gradient descent with non-convex constraints: local concavity determines convergence. Inf Inference [Internet]. 2018 Dec 11 [cited 2020 Jan 5];7(4):755–806. Available from: https://academic.oup.com/imaiai/article-abstract/7/4/755/4904162

16. Köster J, Rahmann S. Snakemake--a scalable bioinformatics workflow engine. Bioinformatics [Internet]. 2012 Oct 1;28(19):2520–2. Available from: http://dx.doi.org/10.1093/bioinformatics/bts480

17. Cluster and cloud execution — Snakemake 5.9.1+0.g138720f.dirty documentation [Internet]. [cited 2020 Jan 5]. Available from: https://snakemake.readthedocs.io/en/stable/executing/cluster-cloud.html

18. Pedregosa F, Varoquaux G, Gramfort A, Michel V, Thirion B, Grisel O, et al. Scikit-learn: Machine Learning in Python. J Mach Learn Res [Internet]. 2011 [cited 2019 Dec 23];12(Oct):2825–30. Available from: http://www.jmlr.org/papers/v12/pedregosa11a

19. Van Craenendonck T, Blockeel H. Using internal validity measures to compare clustering algorithms. Benelearn 2015 Poster presentations (online) [Internet]. 2015;1–8. Available

from: https://lirias.kuleuven.be/1656512?limo=0

20. McKinney W, Others. Data structures for statistical computing in python. In: Proceedings of the 9th Python in Science Conference [Internet]. Austin, TX; 2010. p. 51–6. Available from: https://pdfs.semanticscholar.org/f6da/c1c52d3b07c993fe52513b8964f86e8fe381.pdf

21. Walt S van der, Colbert SC, Varoquaux G. The NumPy Array: A Structure for Efficient Numerical Computation. Comput Sci Eng [Internet]. 2011 Mar 1;13(2):22–30. Available from: https://aip.scitation.org/doi/abs/10.1109/MCSE.2011.37

22. Virtanen P, Gommers R, Oliphant TE, Haberland M, Reddy T, Cournapeau D, et al. SciPy 1.0--Fundamental Algorithms for Scientific Computing in Python [Internet]. arXiv [cs.MS]. 2019. Available from: http://arxiv.org/abs/1907.10121

23. Hunter JD. Matplotlib: A 2D Graphics Environment. Comput Sci Eng [Internet]. 2007 May 1;9(3):90–5. Available from: https://aip.scitation.org/doi/abs/10.1109/MCSE.2007.55

24. Waskom M, Botvinnik O, O'Kane D, Hobson P, Lukauskas S, Gemperline DC, et al. mwaskom/seaborn: v0.8.1 (September 2017) [Internet]. 2017. Available from: https://zenodo.org/record/883859

25. Yoo AB, Jette MA, Grondona M. SLURM: Simple Linux Utility for Resource Management. In: Job Scheduling Strategies for Parallel Processing [Internet]. Springer Berlin Heidelberg; 2003. p. 44–60. Available from: http://dx.doi.org/10.1007/10968987_3

26. 2.3. Clustering — scikit-learn 0.22 documentation [Internet]. [cited 2019 Dec 23]. Available from: https://scikit-learn.org/stable/modules/clustering.html

27. Cancer Genome Atlas Network. Comprehensive molecular portraits of human breast

tumours. Nature [Internet]. 2012 Oct 4;490(7418):61–70. Available from: http://dx.doi.org/10.1038/nature11412

28. Sorlie T, Tibshirani R, Parker J, Hastie T, Marron JS, Nobel A, et al. Repeated observation of breast tumor subtypes in independent gene expression data sets. Proc Natl Acad Sci U S A [Internet]. 2003 Jul 8;100(14):8418–23. Available from: http://dx.doi.org/10.1073/pnas.0932692100

29. Sørlie T, Perou CM, Tibshirani R, Aas T, Geisler S, Johnsen H, et al. Gene expression patterns of breast carcinomas distinguish tumor subclasses with clinical implications. Proc Natl Acad Sci U S A [Internet]. 2001 Sep 11;98(19):10869–74. Available from: http://dx.doi.org/10.1073/pnas.191367098

30. Rakha EA, Ellis IO. Triple-negative/basal-like breast cancer: review. Pathology [Internet]. 2009 Jan;41(1):40–7. Available from: http://dx.doi.org/10.1080/00313020802563510

31. Badve S, Dabbs DJ, Schnitt SJ, Baehner FL, Decker T, Eusebi V, et al. Basal-like and triple-negative breast cancers: a critical review with an emphasis on the implications for pathologists and oncologists. Mod Pathol [Internet]. 2011 Feb;24(2):157–67. Available from: http://dx.doi.org/10.1038/modpathol.2010.200

32. Tikhonova AN, Dolgalev I, Hu H, Sivaraj KK, Hoxha E, Cuesta-Domínguez Á, et al. The bone marrow microenvironment at single-cell resolution. Nature [Internet]. 2019 May;569(7755):222–8. Available from: http://dx.doi.org/10.1038/s41586-019-1104-8

33. Wang D. hppRNA—a Snakemake-based handy parameter-free pipeline for RNA-Seq analysis of numerous samples. Brief Bioinform [Internet]. 2018 Jul 20 [cited 2020 Jan 5];19(4):622–6. Available from:

https://academic.oup.com/bib/article-abstract/19/4/622/2918128

34. Pranzatelli TJF, Michael DG, Chiorini JA. ATAC2GRN: optimized ATAC-seq and DNase1-seq pipelines for rapid and accurate genome regulatory network inference. BMC Genomics [Internet]. 2018 Jul 31;19(1):563. Available from: http://dx.doi.org/10.1186/s12864-018-4943-z

35. Abdelaal T, Michielsen L, Cats D, Hoogduin D, Mei H, Reinders MJT, et al. A comparison of automatic cell identification methods for single-cell RNA sequencing data [Internet]. Vol. 20, Genome Biology. 2019. Available from: http://dx.doi.org/10.1186/s13059-019-1795-z

36. Dirmeier S, Emmenlauer M, Dehio C, Beerenwinkel N. PyBDA: a command line tool for automated analysis of big biological data sets. BMC Bioinformatics [Internet]. 2019 Nov 12;20(1):564. Available from: http://dx.doi.org/10.1186/s12859-019-3087-8

37. single-cell-rna-seq [Internet]. Github; [cited 2020 Jan 8]. Available from: https://github.com/snakemake-workflows/single-cell-rna-seq

38. Lun ATL, McCarthy DJ, Marioni JC. A step-by-step workflow for low-level analysis of single-cell RNA-seq data with Bioconductor [Internet]. Vol. 5, F1000Research. 2016. p. 2122. Available from: http://dx.doi.org/10.12688/f1000research.9501.2

39. Soneson C, Robinson MD. Bias, robustness and scalability in single-cell differential expression analysis. Nat Methods [Internet]. 2018 Apr;15(4):255–61. Available from: http://dx.doi.org/10.1038/nmeth.4612