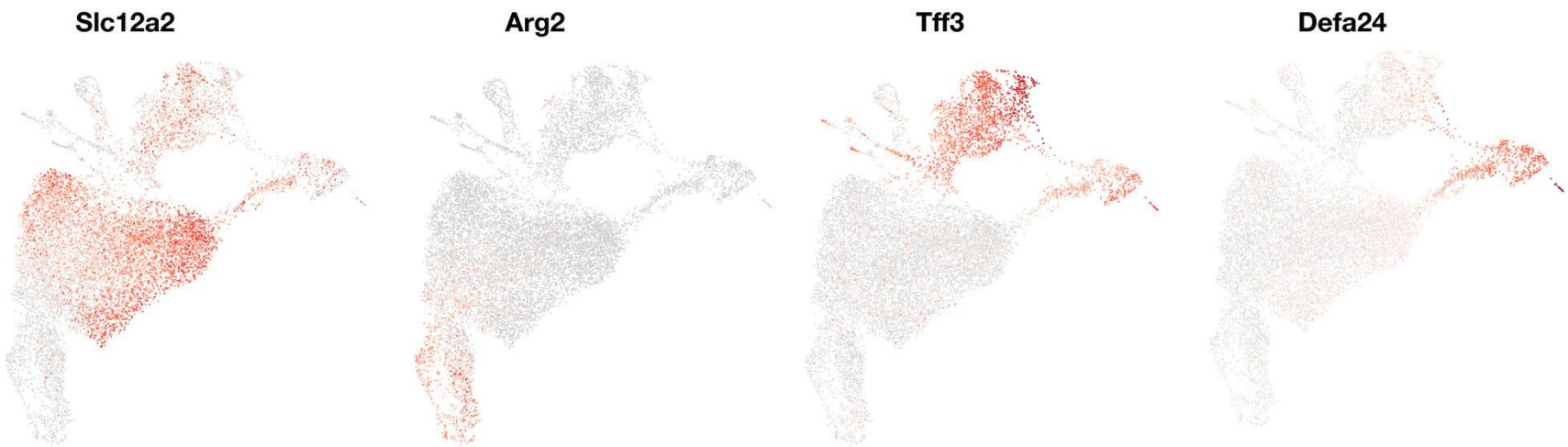
Single Cell Data Analysis

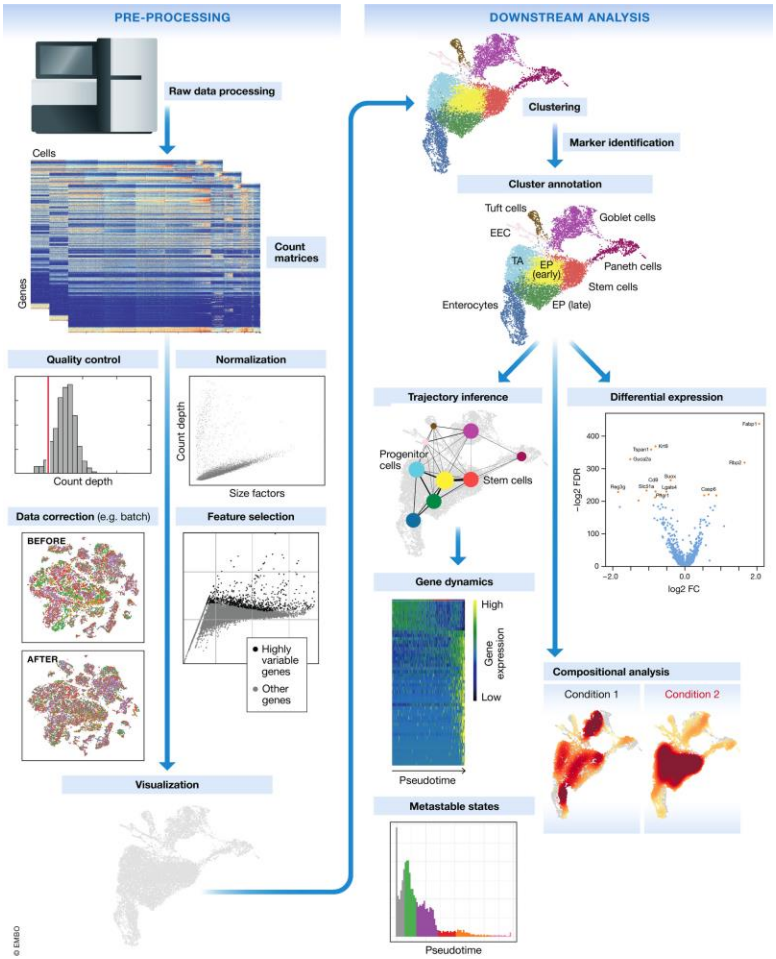
CMM262-2022
Rob Morey
remorey@eng.ucsd.edu



Typical Single-cell RNA-seq Analysis Workflow

Workflow:

- Filter for good cells and detected genes (arbitrary cutoffs)
- Normalize/Scale Data
- Remove unwanted sources of variation (batch/cell cycle)
- Feature selection, dimensionality reduction and visualization
- Feature selection (highly variable genes)
- Dimensionality Reduction – summarization (describe data in as few dimensions as possible for downstream)
- Dimensionality Reduction – visualization (describe data in 2D or 3D)
- Clustering (grouping cells based on expression profiles)
- Define Cell-Type specific signatures through cluster annotation.
- Trajectory analysis (transitions between cell identities)
- Unification between clustering and trajectory inference - (partition-based graph abstraction - PAGA)



Wolf *et al. Genome Biology* (2018) 19:15
https://doi.org/10.1186/s13059-017-1382-0

Genome Biology

SOFTWARE Open Access

SCANPY: large-scale single-cell gene expression data analysis

F. Alexander Wolf[1*], Philipp Angerer[1] and Fabian J. Theis[1,2*]

Review

OPEN ACCESS

molecular systems biology

Current best practices in single-cell RNA-seq analysis: a tutorial

Malte D Luecken[1] & Fabian J Theis[1,2,*]

https://github.com/theislab/single-cell-tutorial/

https://scanpy.readthedocs.io/en/latest/index.html

Scanpy
latest

Search docs

Tutorials
Usage Principles
Installation
API
References

Docs » Scanpy – Single-Cell Analysis in Python

Edit on GitHub

pypi v1.3.7 | docs passing | build passing | install with bioconda

Scanpy – Single-Cell Analysis in Python

Scanpy is a scalable toolkit for analyzing single-cell gene expression data. It includes preprocessing, visualization, clustering, pseudotime and trajectory inference and differential expression testing. The Python-based implementation efficiently deals with datasets of more than one million cells.

Report issues and see the code on GitHub. If Scanpy is useful for your research, consider citing Genome Biology (2018).

Usage Principles

Import the Scanpy API as:

```
import scanpy.api as sc
```

Workflow

The typical workflow consists of subsequent calls of data analysis tools in `sc.tl`, e.g.:

```
sc.tl.tsne(adata, **tool_params)  # embed the data using tSNE
```

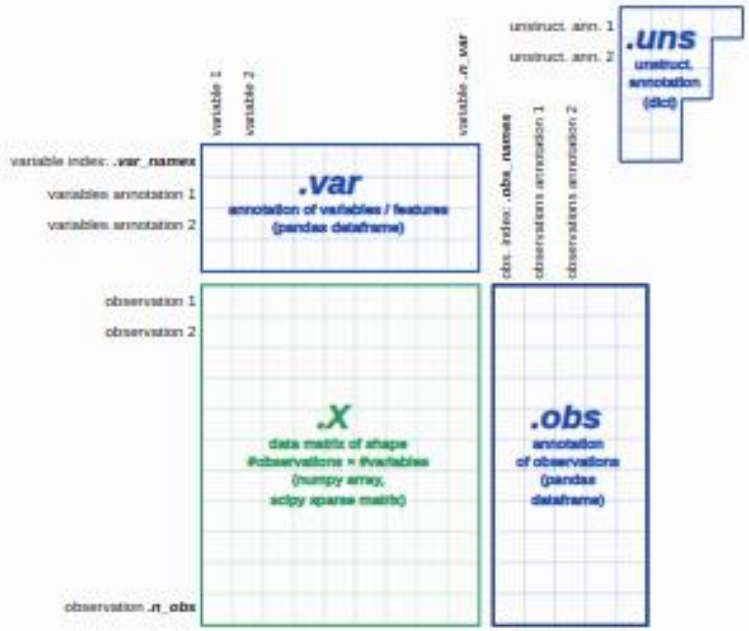
where `adata` is an **`AnnData`** object. Each of these calls adds annotation to an expression matrix *X*, which stores *n_obs* observations (cells) of *n_vars* variables (genes). For each tool, there typically is an associated plotting function in `sc.pl`:

```
sc.pl.tsne(adata, **plotting_params)
```

If you pass `show=False`, a **`matplotlib.axes.Axes`** instance is returned and you have all of matplotlib's detailed configuration possibilities.

AnnData

Scanpy is based on `anndata`, which provides the `AnnData` class.



At the most basic level, an AnnData object adata stores a data matrix (adata.X), dataframe-like annotation of observations (adata.obs) and variables (adata.var) and unstructured dict-like annotation (adata.uns). Values can be retrieved and appended via adata.obs['key1'] and adata.var['key2']. Names of observations and variables can be accessed via adata.obs_names and adata.var_names, respectively. AnnData objects can be sliced like dataframes, for example, adata_subset = adata[:, list_of_gene_names].

http://falexwolf.de/blog/171223_AnnData_indexing_views_HDF5-backing/

https://www.ncbi.nlm.nih.gov/geo/query/acc.cgi?acc=GSE150637

Article

Reprogramming roadmap reveals route to human induced trophoblast stem cells

https://doi.org/10.1038/s41586-020-2734-6

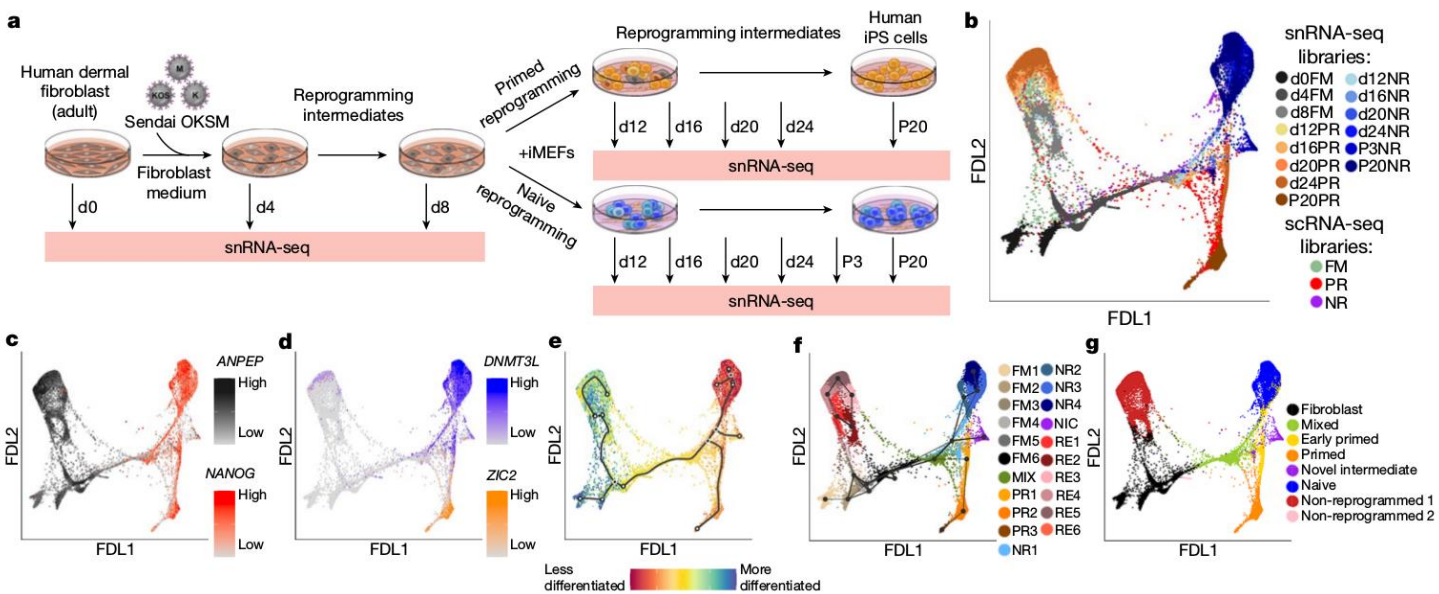
Received: 5 February 2019

Accepted: 24 June 2020

Published online: 16 September 2020

Check for updates

Xiaodong Liu[1,2,3,19], John F. Ouyang[4,19], Fernando J. Rossello[1,2,3,16,19], Jia Ping Tan[1,2,3], Kathryn C. Davidson[1,2,3], Daniela S. Valdes[1,2,3], Jan Schröder[1,2,3], Yu B. Y. Sun[1,2,3], Joseph Chen[1,2,3], Anja S. Knaupp[1,2,3], Guizhi Sun[1,2,3], Hun S. Chy[3,5], Ziyi Huang[3,5], Jahnvi Pflueger[6,7], Jaber Firas[1,2,3], Vincent Tano[1,2,3], Sam Buckberry[6,7], Jacob M. Paynter[1,2,3], Michael R. Larcombe[1,2,3], Daniel Poppe[6,7], Xin Yi Choo[1,2,3], Carmel M. O'Brien[3,5], William A. Pastor[8,9,17], Di Chen[8,9], Anna L. Leichter[10], Haroon Naeem[11], Pratibha Tripathi[1,2], Partha P. Das[1,2], Alexandra Grubman[1,2,3], David R. Powell[11], Andrew L. Laslett[3,5], Laurent David[12,13], Susan K. Nilsson[3,5], Amander T. Clark[8,9,14,15], Ryan Lister[6,7], Christian M. Nefzger[1,2,3,18], Luciano G. Martelotto[10], Owen J. L. Rackham[4] ✉ & Jose M. Polo[1,2,3] ✉



Preprocessing: Raw data processing and filtering

- Raw data processing pipelines like Cell Ranger assign reads to cells, align reads to genome, and create count matrices.
- Cell QC performed on three QC covariates:
- 1) Count Depth (number of reads per cell barcode)
- 2) Number of genes per barcode (cell)
- 3) Fraction of counts from mitochondrial genes

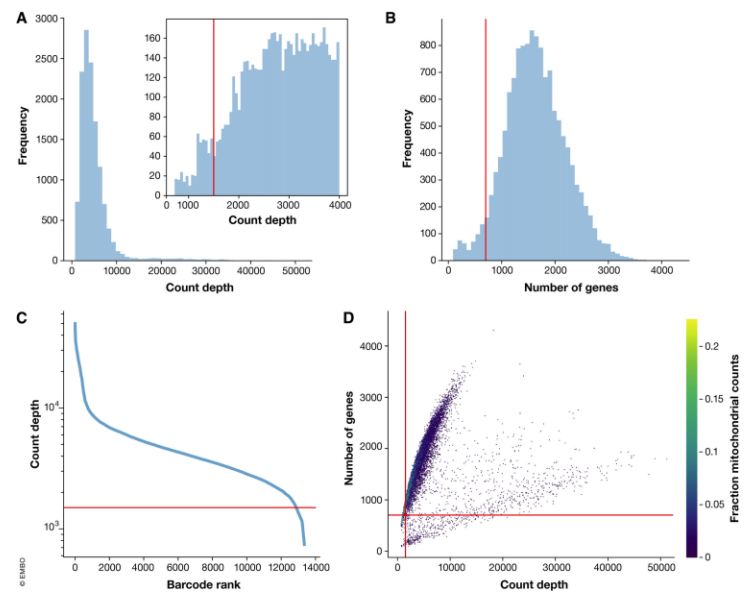


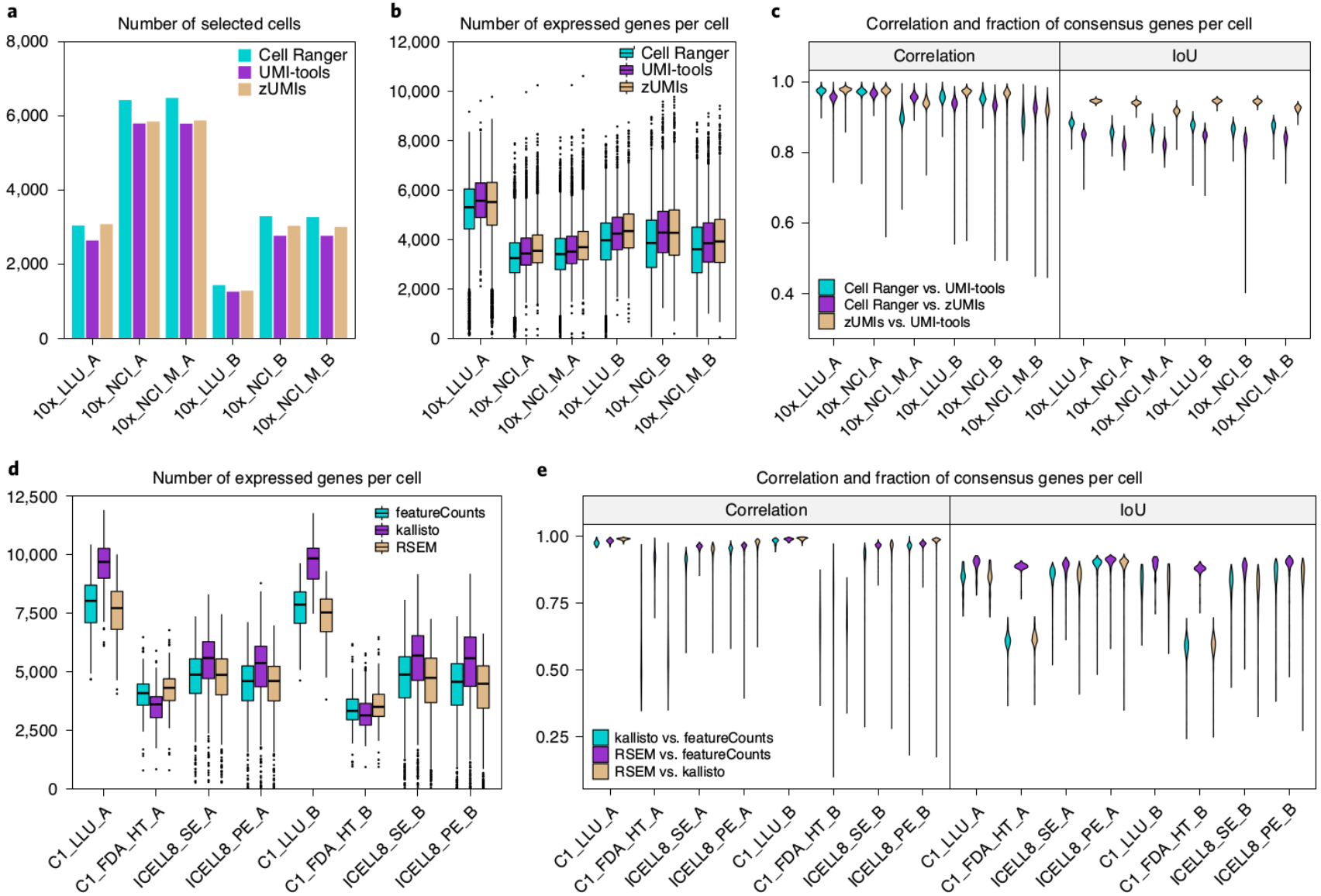
Table 1 | **FASTQ processing tools**

Method	Description	Documentation	Detects empty barcodes	Ref.
CellRanger	Default 10X genomics software package for processing data generated on the 10X platform	https://support.10xgenomics.com/single-cell-gene-expression/software/pipelines/latest/what-is-cell-ranger	Yes	4
DropEst	Improves on quantification accuracy compared with CellRanger. Supports 10X, Split-seq, Drop-seq, inDrop, iCLIP and Seq-Well	https://github.com/hms-dbmi/dropEst	Yes	20
Kallisto–BUStools	Extremely efficient memory and CPU usage through the use of the BUStools file formats. Supports any platform that uses cell barcodes	https://www.kallistobus.tools/getting_started	No	21
Alevin	Extension of the Salmon pseudo-aligner for scRNA-seq data. Supports 10X and Drop-seq platforms	https://salmon.readthedocs.io/en/latest/alevin.html	Yes	26
STARSolo	Extension of the STAR read aligner for processing single-cell data. Supports the 10X platform	https://github.com/alexdobin/STAR/blob/master/doc/STARmanual.pdf	Yes	25
UMI-Tools	Models potential errors in UMIs and corrects them to improve gene expression accuracy	https://github.com/CGATOxford/UMI-tools	Yes	23

CPU, central processing unit; scRNA-seq, single-cell RNA sequencing; UMI, unique molecular index.

Wu et al., 2020 https://doi.org/10.1038/s41581-020-0262-0

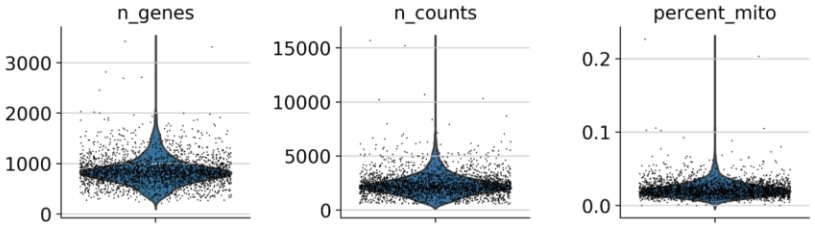
Preprocessing variation

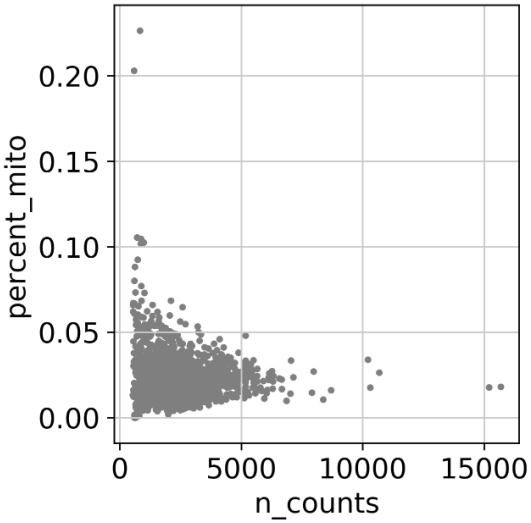


Chen et al. Nature Biotech 2020

Remove outlier cells/genes

- A guideline to setting gene thresholds is to use the minimum cell cluster size that is of interest and leaving some leeway for dropout effects.
- Be permissive and revisit QC after clustering (judged based on downstream analysis, but not "p-hacking")
- Do not look at any of the covariates in isolation

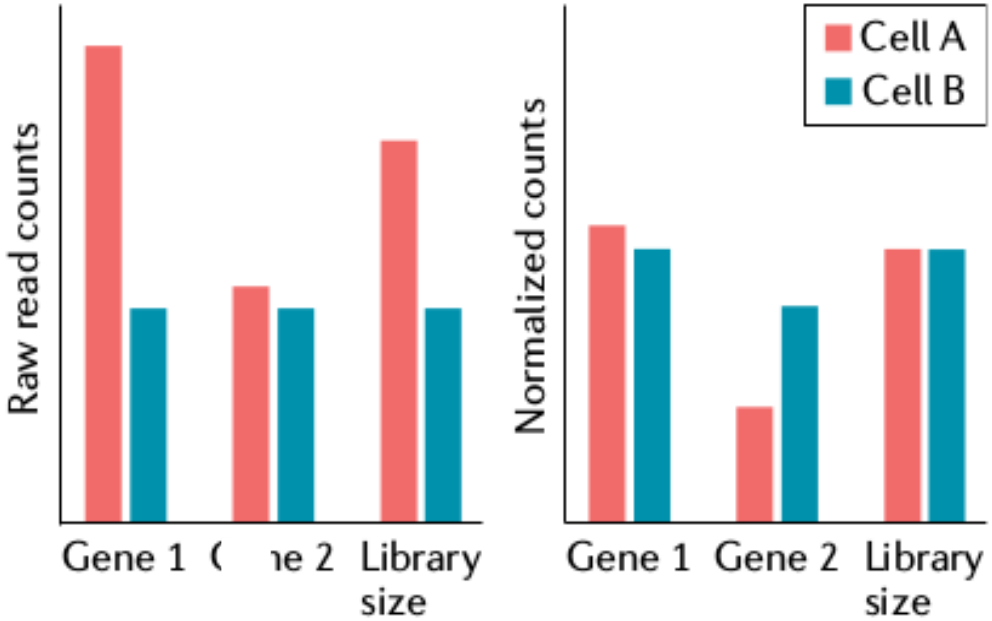




Normalization

- How do you normalize for sequencing depth?
- Should you normalize for sequencing depth in single cell data?
- Is length an important normalization step here?
- Should you perform gene scaling (weight genes equally)?
- Log transformation? - allows for normal dist. Assumption
- Scone tool can be used to select appropriate normalization method

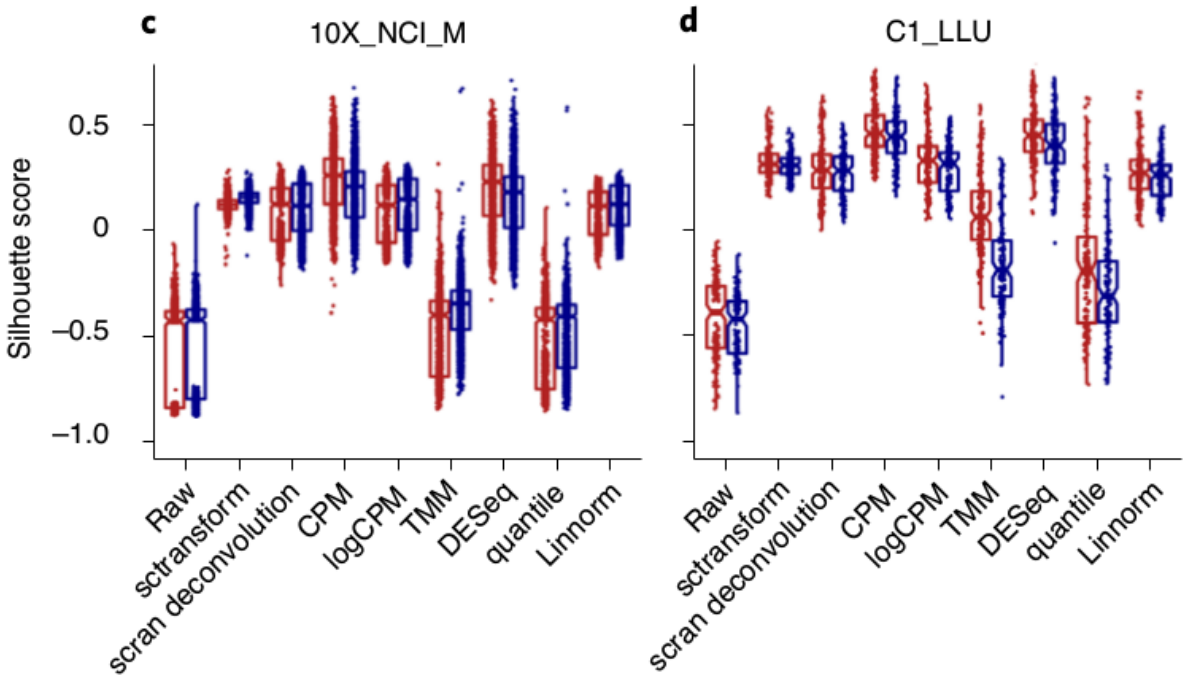
	Cell-specific effects	Gene-specific effects	Not removed by UMIs
Sequencing depth	✓		✓
Amplification	✓	✓	
Capture & RT efficiency	✓	✓	✓
Gene length		✓	
GC-content	✓	✓	✓
mRNA content	✓		✓



Wu et al., 2020 https://doi.org/10.1038/s41581-020-0262-0

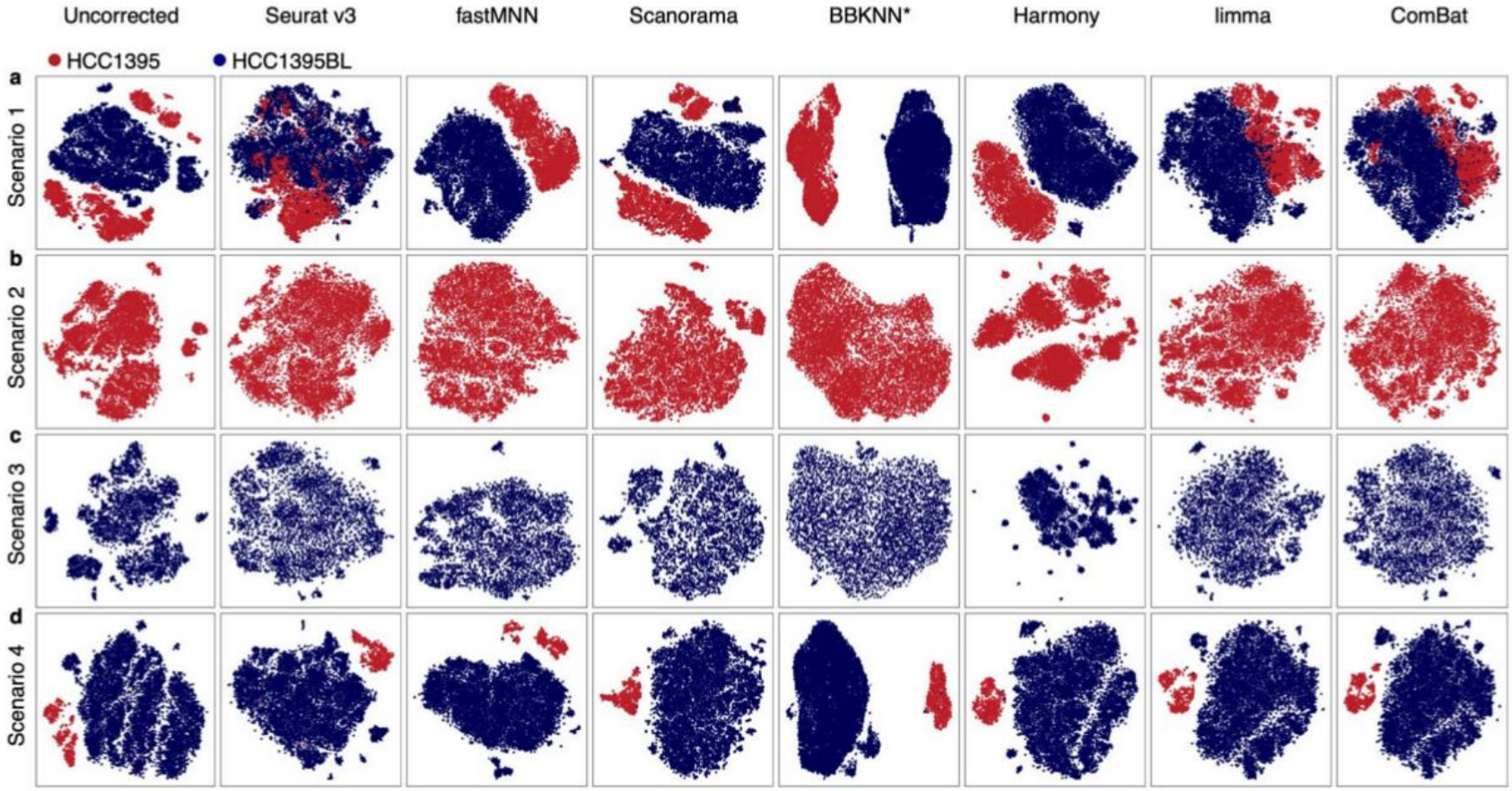
Effects of Normalization

Silhouette width score quantifies how well two different types of cells are separated from each other. The larger the silhouette width values, the better the performance of the normalization method



Chen et al. Nature Biotech 2020

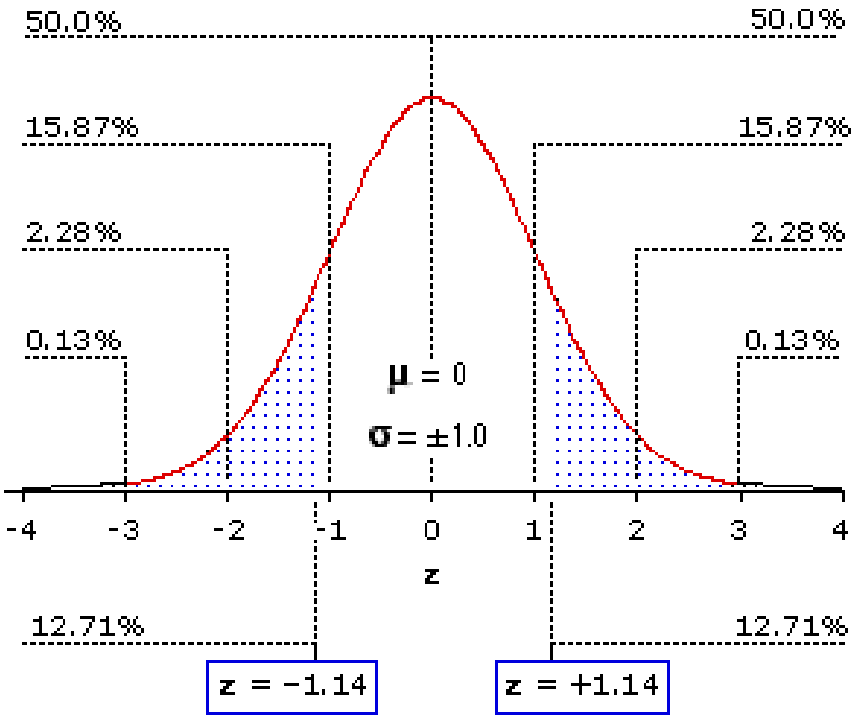
Effects of Normalization



Chen et al. Nature Biotech 2020

Z-score scaling for comparing between genes:

- Z-score is the number of standard deviations away from the mean
- Purpose is to scale expression of each gene relative to all the cells
- The preference between the two choices revolves around whether all genes should be weighted equally for downstream analysis, or whether the magnitude of expression of a gene is an informative proxy for the importance of the gene.



Overcoming systematic errors caused by log-transformation of normalized single-cell RNA sequencing data

Aaron Lun[1,*]

1 Cancer Research UK Cambridge Institute, University of Cambridge, Li Ka Shing Centre, Robinson Way, Cambridge CB2 0RE, United Kingdom

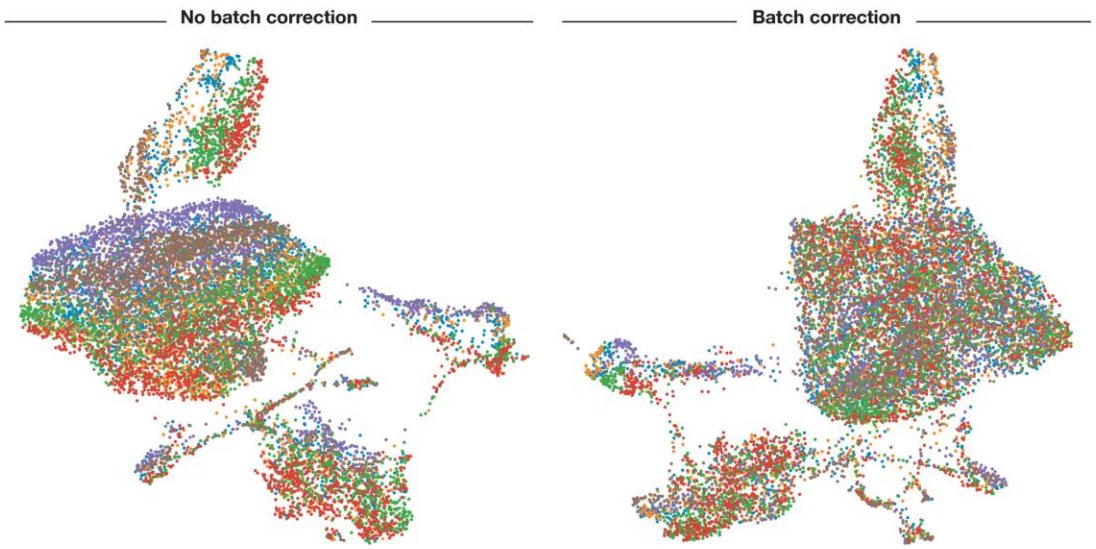
*** Email: aaron.lun@cruk.cam.ac.uk**

Abstract

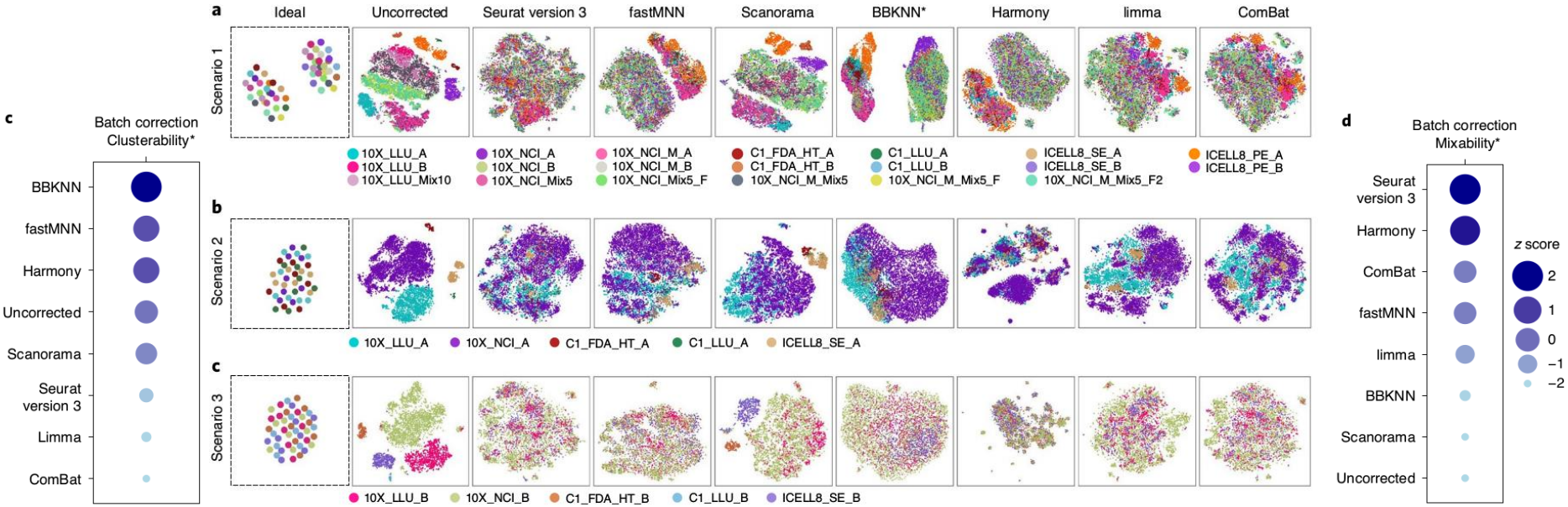
Applying a log-transformation to normalized expression values is one of the most common procedures in exploratory analyses of single-cell RNA sequencing (scRNA-seq) data. Normalization removes systematic biases in sequencing coverage between cells, while the log-transformation ensures that downstream computational procedures operate on relative rather than absolute differences in expression. We show that the log-transformation can introduce systematic errors when cells vary in sequencing coverage, leading to spurious non-zero differences in expression and artificial population structure in simulations. We observe similar effects in real scRNA-seq data where the difference in transformed values between groups of cells is not an accurate proxy for the log-fold change. We provide some practical recommendations to overcome this effect and analytically derive an expression for a larger pseudo-count that controls the transformation-induced error to a specified threshold.

Data Correction and Integration

- Correct for biological (e.g. cell cycle) covariates – *warning: Regress out biological covariates only for trajectory inference and if other biological processes of interest are not masked by the regressed out biological covariate*
- Correct for technical covariates (e.g. batch effects – data integration)



Batch Effect Comparison

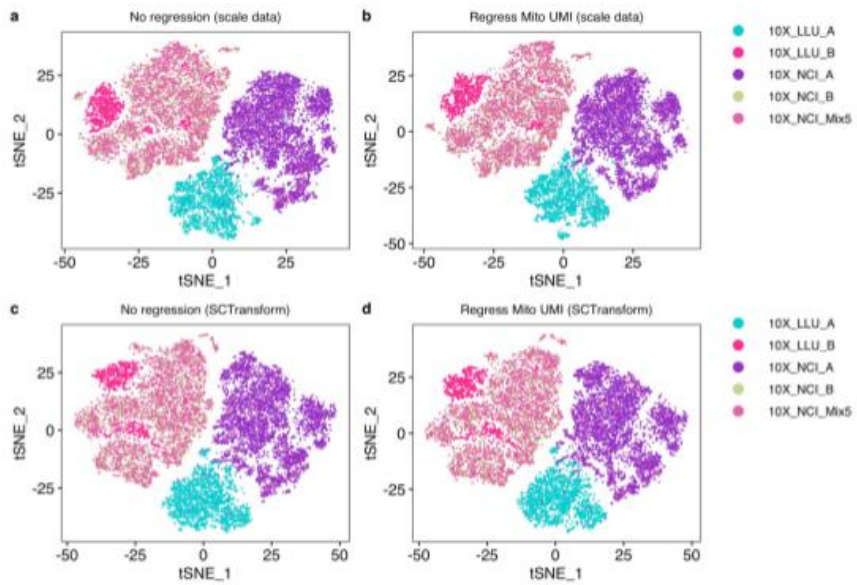


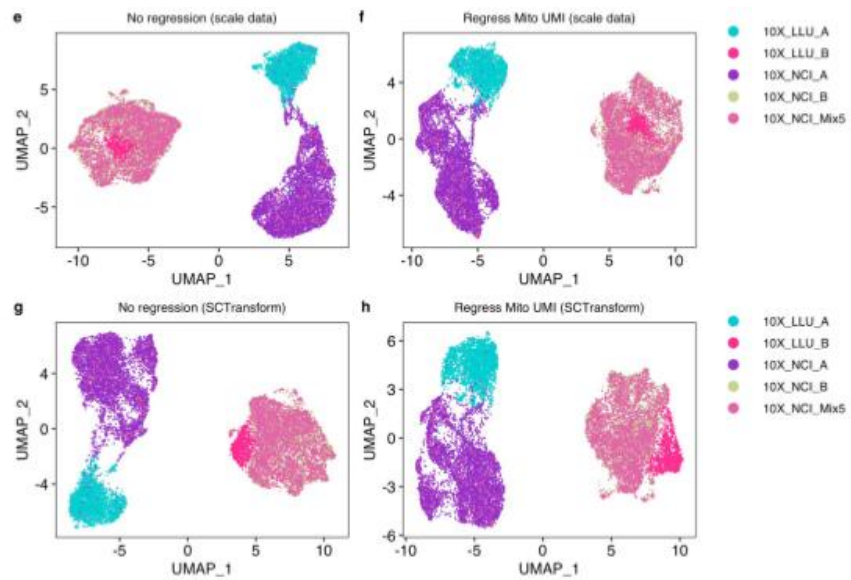
Clusterability - ability to separate different cell types from each other

Mixability - ability to group similar cells together across datasets

Chen et al. Nature Biotech 2020

Regressing out mitochondrial genes does not seem to help downstream clustering





Chen et al. Nature Biotech 2020

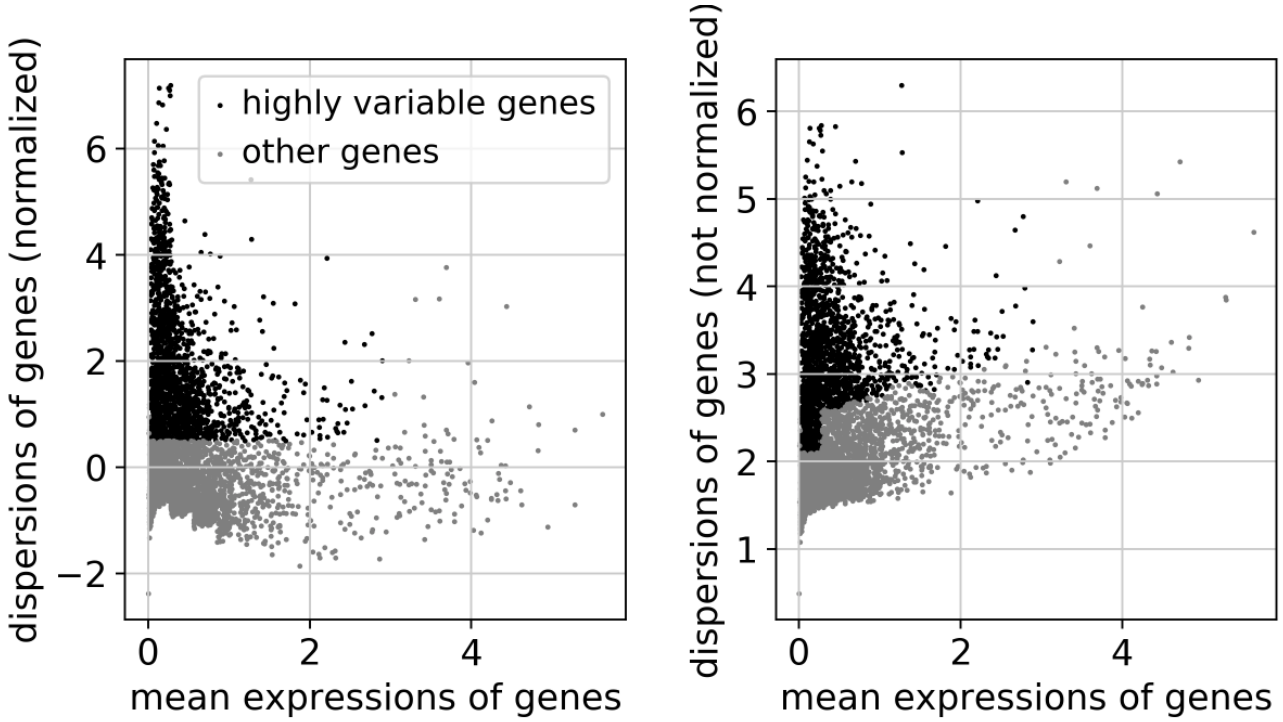
Chen et al. Nature Biotech 2020 – best practices summary

Box 1 | Best practice recommendations

We summarize below 11 best practice recommendations for the community based on our analysis.

1. There were large variations across different scRNA-seq platforms and centers.
2. While most of the genes and cells detected were consistent between the different methods, we observed variations for low-expression genes and cells with low mRNA content across different methods. However, these differences did not affect our analyses of cell classification or mixability.
3. Normalization algorithms alone could not remove batch effects.
4. Different normalization strategies performed differently across datasets and platforms; sctransform, scran, logCPM and Linnorm performed well for either 3′- or full-length-transcript scRNA-seq platforms, but TMM and quantile performed poorly and are not recommended.
5. Seurat version 3, Harmony, BBKNN, fastMNN and Scanorama all could correct and remove batch variations in specific sample and dataset scenarios; we recommend users apply appropriate batch-effect correction methods depending on the characteristics of their datasets (for example, cellular and sample heterogeneity and composition, platforms used; Fig. 6e).
6. BBKNN, fastMNN and Harmony ranked best for clusterability/cell type classification, whereas Seurat version 3, Harmony and fastMNN performed best for mixability.
7. fastMNN, BBKNN and Harmony removed batch variations effectively across different platforms, including both mixed and unmixed distinct samples, but the order of importing the datasets into the pipeline and the requirement for a mixed sample was critical for MNN and fastMNN, whereas BBKNN and Harmony performed well regardless of the inclusion of mixed heterogeneous biologically distinct samples across platforms and batches; thus, for MNN and fastMMN, we recommend including a mixed sample and importing the mixed data into the pipeline first.
8. CCA/Seurat version 3 had superior mixability for biologically similar samples but overcorrected batch effects and misclassified cells (that is, poor clusterability/cell type classification) when large proportions of distinct cell types were present. However, Seurat version 3 performed well both for clusterability and mixability for datasets when only a small fraction of dissimilar cells (for example, 5–10%) was present. Thus, we do not recommend using CCA Seurat version 3 for scenarios containing large fractions of biologically distinct cell types.
9. BBKNN performed best in clusterability and cell type classification, but it ranked low in mixability, particularly in heterogeneous cell samples.
10. The current version of Scanorama performed well only for the 10x Genomics data and did not work for non-10x platforms; thus, we do not recommend it for data from non-10x platforms.
11. We observed good consistency between Cell Ranger 3.1 and 2.0 preprocessed data; however, Cell Ranger 3.1 can detect some extra cells with very few transcripts; this may affect batch-effect corrections in certain scenarios.

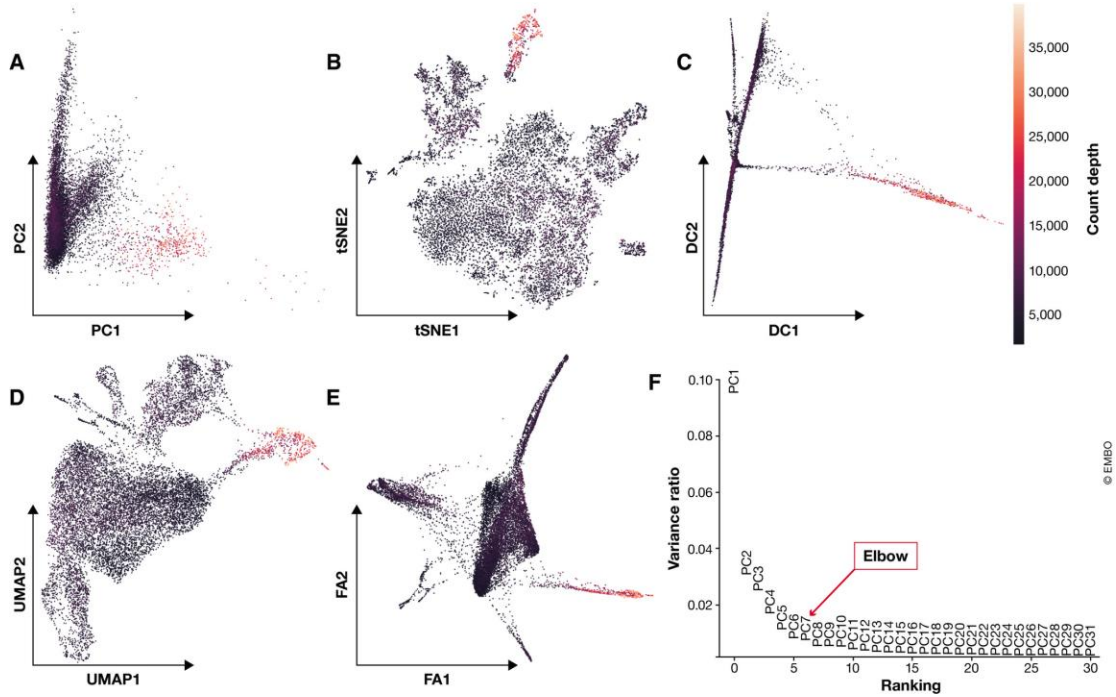
Feature Selection: Filter for variable genes to control the relationship between variability and average expression



Preliminary results from Klein et al (2015) suggest that downstream analysis is robust to the exact choice of the number of HVGs (between 200-2400).

Break Time!

Dimensionality Reduction



Goal: Embed expression matrix into low-dimensional space that still captures the underlying structure of the data

Two Goals:

- 1) Visualization – describe data in 2D or 3D
- 2) Summarization – reduce data to essential components - used downstream

Dimensionality Reduction Methods

Table 2 | **Methods of dimensionality reduction**

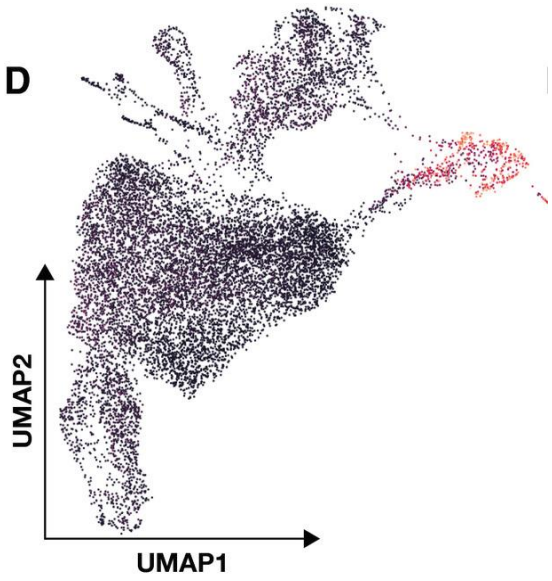
Method	Description	Documentation	Ref.
PCA	Default dimensionality reduction method for most single-cell pipelines	Implemented in Seurat, SCANPY and Pagoda2; https://github.com/ujjwalkarn/DataScienceR/blob/master/PCA.R	56
ZIFA	Variation of PCA that accounts for zero inflation in the counts matrix	https://github.com/epierson9/ZIFA	57
f-scLVM	Uses latent variable modelling and gene sets to generate interpretable lower dimensional factors	https://github.com/bioFAM/slalom	58
Pagoda2	Runs PCA on gene sets to identify interpretable components and find the ones with the highest variability for the given dataset	https://github.com/hms-dbmi/pagoda2	39
NMF	Generates a more interpretable dimensional reduction in which each dimension typically corresponds to a group of genes expressed in a group of cells	https://github.com/linxihui/NNLM	60
LLE	Generates a piecewise locally linear dimensional reduction that can capture non-linearity in the data. Works well for capturing trajectories	https://github.com/jw156605/SLICER	63
Dmaps	Generates a smooth dimensional reduction under the assumption that the cells follow a continuous path	https://github.com/theislab/destiny	62
DCA	Uses a deep neural network to encode the dataset into lower dimensions	https://github.com/theislab/dca	43
scScope	Uses a recurrent neural network to remove technical noise and then encode the dataset into lower dimensions	https://github.com/AltschulerWu-Lab/scScope	65
scVI	Uses probabilistic modelling with deep neural networks to generate a lower dimensional embedding of the dataset	https://github.com/YosefLab/scVI	42

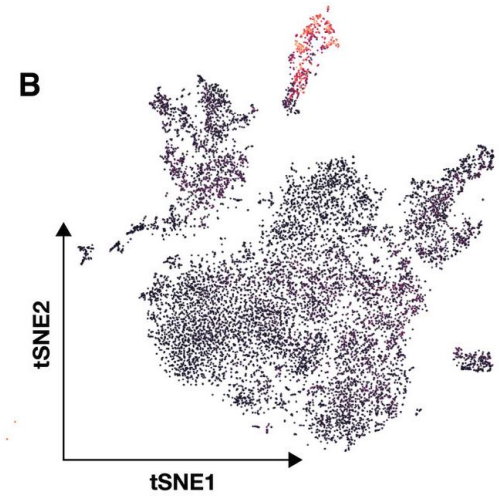
DCA, deep count autoencoder; Dmaps, diffusion maps; f-scLVM, single-cell latent variable model; LLE, locally linear embedding; NMF, non-negative matrix factorization; PCA, principal component analysis; scVI, single-cell variational inference; ZIFA, zero-inflated factor analysis.

Wu et al., 2020 https://doi.org/10.1038/s41581-020-0262-0

Visualization

- Standard Practice – non-linear dimensionality reduction methods
- t-SNE dimensions focus on capturing local similarity at the expense of global structure. Thus, these visualizations may exaggerate differences between cell populations and over- look potential connections between these populations.
- Uniform Approximation and Projection method (UMAP; preprint: McInnes & Healy, 2018) - arguably represent the best approximation of the underlying topology
- Recently published - den-SNE and densMAP (Jan 18, 2021 - Nature Biotech)



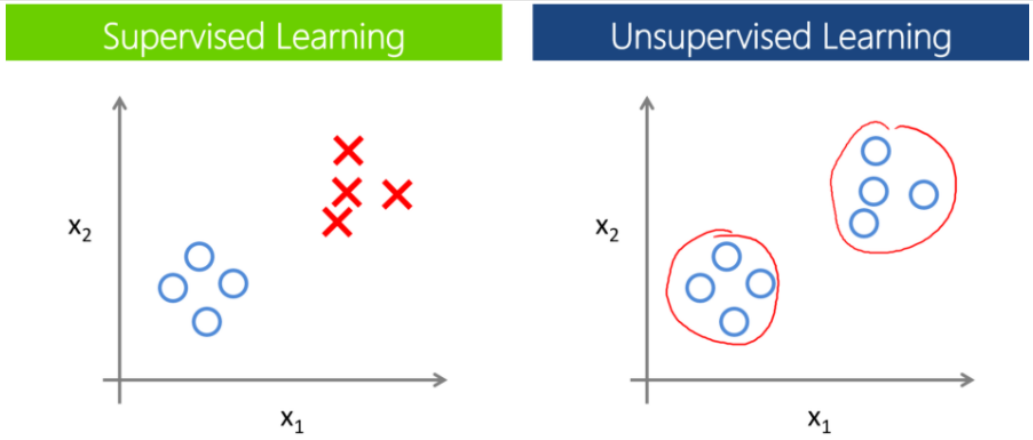


Stages of pre-processed data

Table 1. Stages of data processing and appropriate downstream applications.

Pre-processing layer	Stage of data processing	Appropriate applications
Measured	1) Raw	Statistical testing (Differential expression: marker genes, genes over condition, genes over time)
	2) Normalized (+ log transformed)	
Corrected	3.1) Corrected (technical correction)	Visual comparison of data (plotting)
	3.2) Corrected (biological correction)	Pre-processing for trajectory inference
Reduced	4) Feature selected	Visualization, trajectory inference
	5) Dimensionality reduced (summarized)	Visualization, clustering, KNN graph inference, trajectory inference

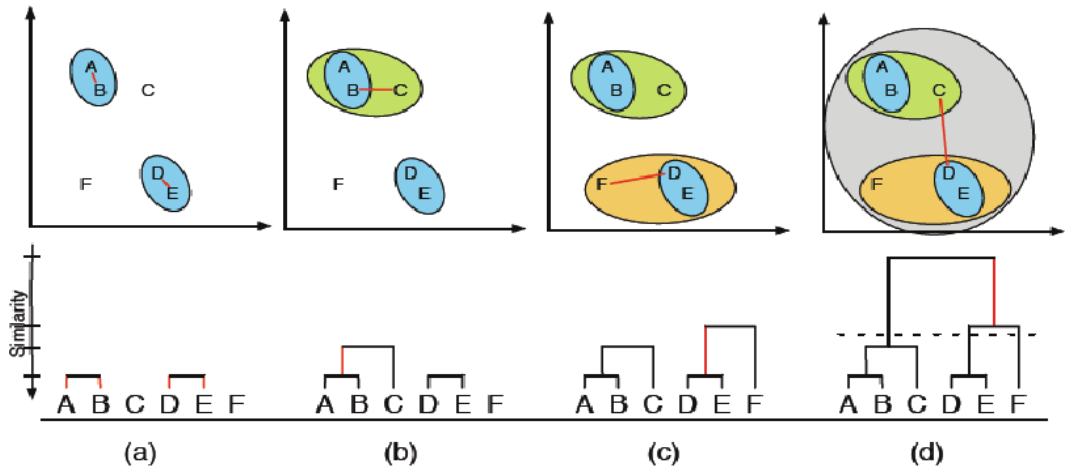
Cluster Analysis



Clustering – group cells based on similarity of expression profiles determined by distance metrics

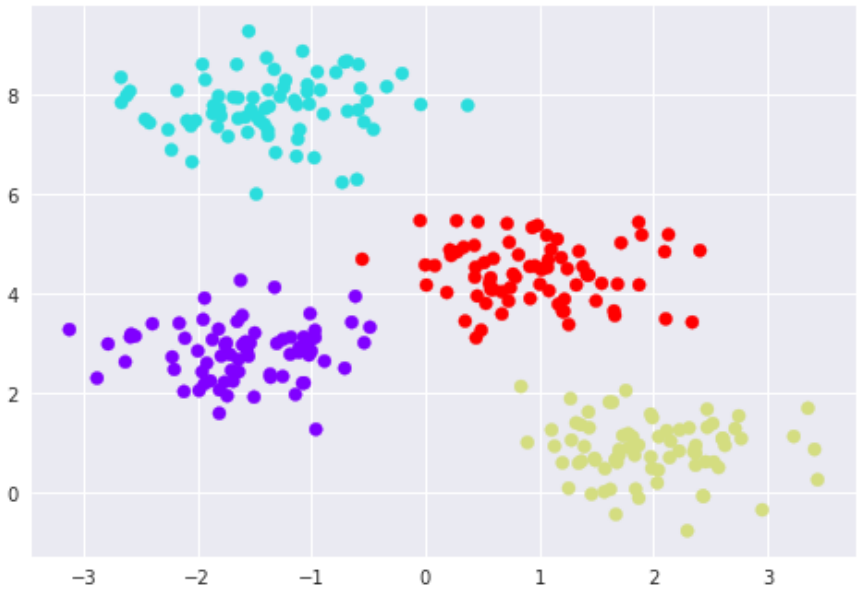
- Supervised:
  - Have prior knowledge of the groups.
- Unsupervised:
  - Have no priors. Looking for substructure to find new patterns
- Cells are assigned to clusters by minimizing intra-cluster distances or finding dense regions in the reduced expression space.

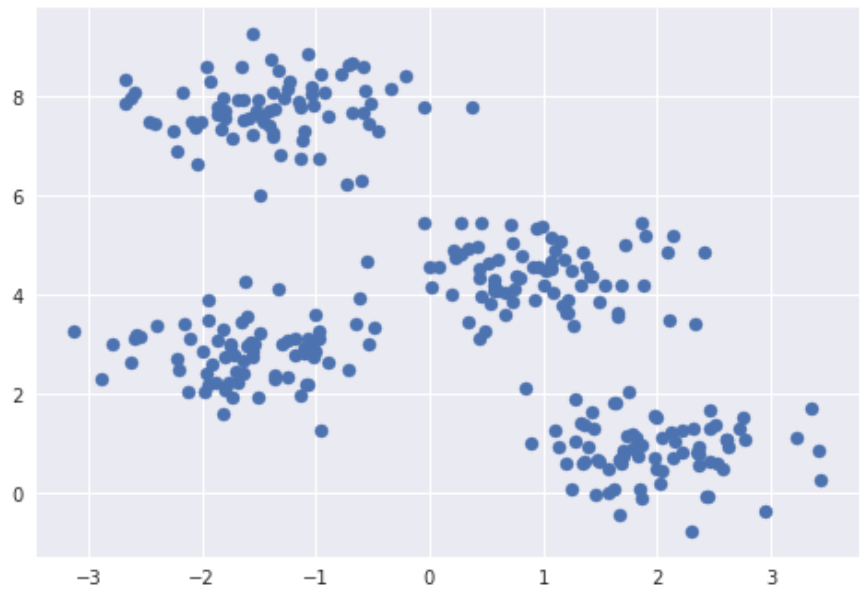
Hierarchical Clustering:



- Grouping cells together based on similarity
- Bottom up: Start with individual points and add most similar data points
- Top down: Start with one big group, and drop out one at a time based on similarity
- Cluster groups are defined based on your chosen similarity metric
- Often used in subclustering but is slow compared to graph-based methods

https://www.researchgate.net/figure/Example-of-hierarchical-clustering-clusters-are-consecutively-merged-with-the-most_273456906





How many clusters exist in this dataset?

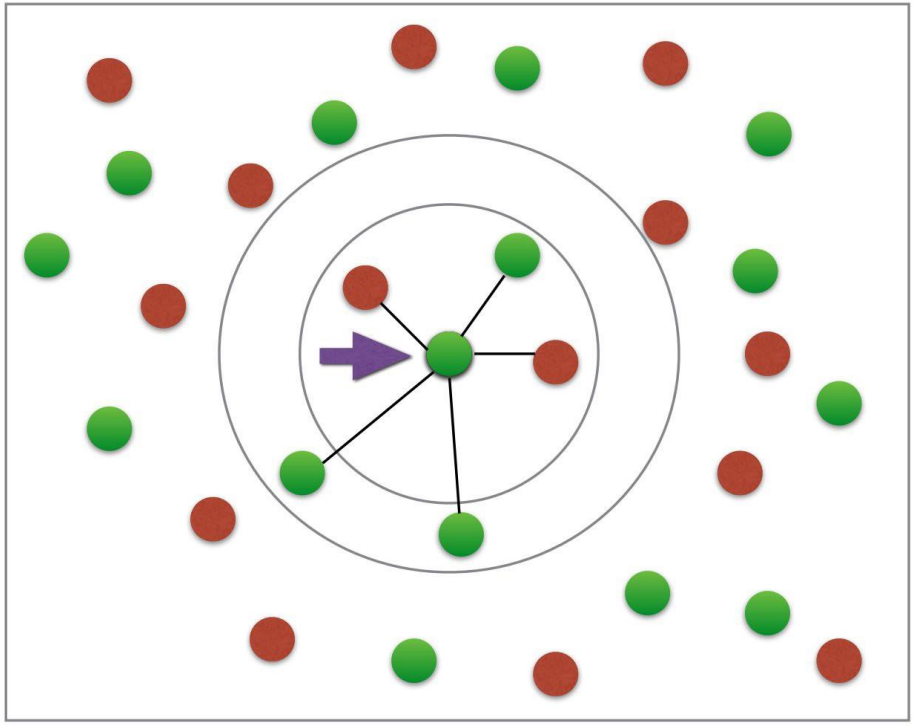
K-means clustering

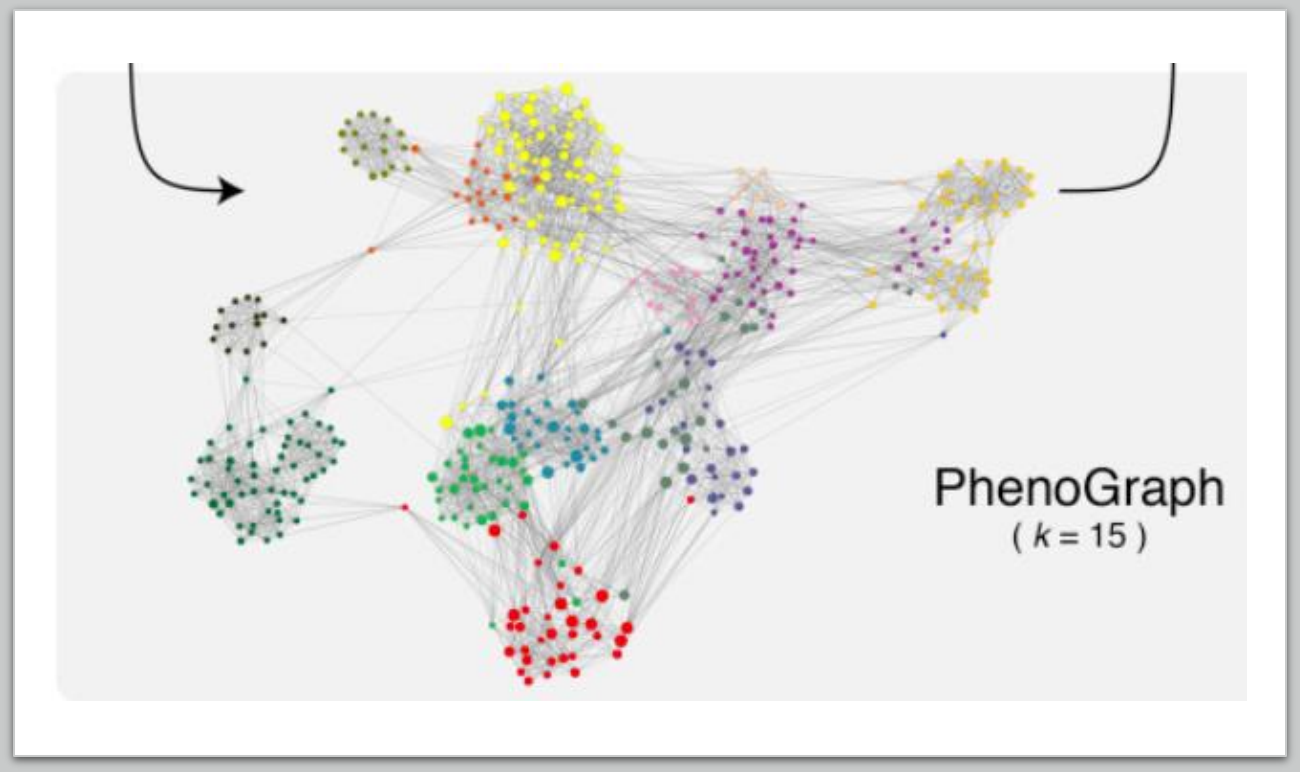
- Randomly assign points to guess the cluster centers (number of points = k)
- Assign data points to the nearest cluster center.
- Move the cluster assignment to the mean of each group
- Repeat

http://stanford.edu/class/ee103/visualizations/kmeans/kmeans.html

Graph-based clustering (k-nearest neighbors)

- Measure Euclidian distances between cells up to K nearest neighbors (edges)
  - Weight of edges scales with similarity metric (Euclidian distance)
- Calculate "connectedness" of nodes (cells) by the number of shared neighbors
- Refine densely connected modules as communities





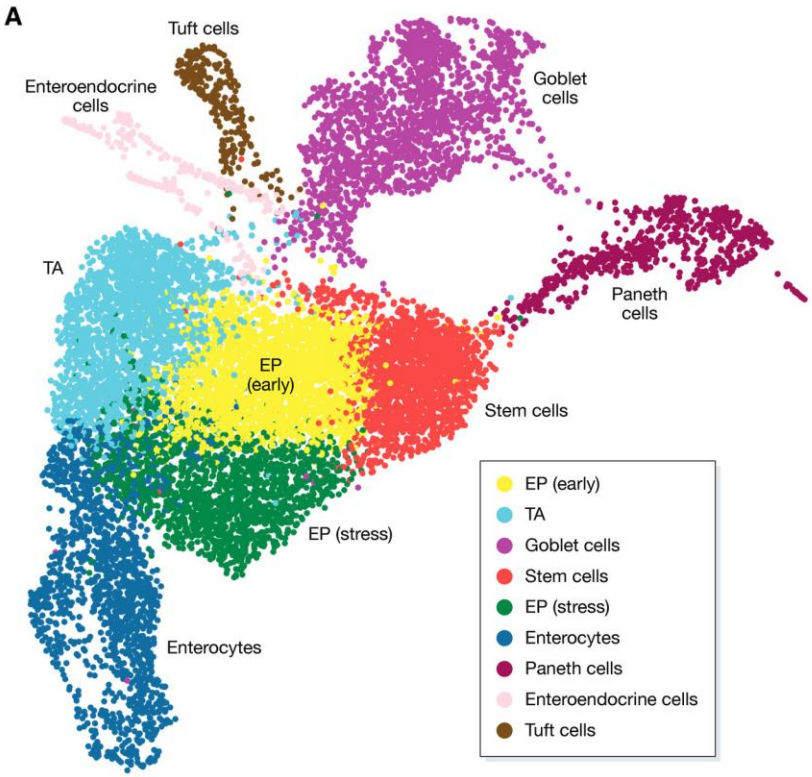
Graph-based clustering (k-nearest neighbors)

Cells are represented as nodes in the graph. Each cell is connected to its K most similar cells, which are typically obtained using Euclidean distances on the PC-reduced expression space. Depending on the size of the dataset, K is commonly set to be between 5 and 100 nearest neighbors. The resulting graph captures the underlying topology of the expression data

Levine et al. 2015 *Cell*

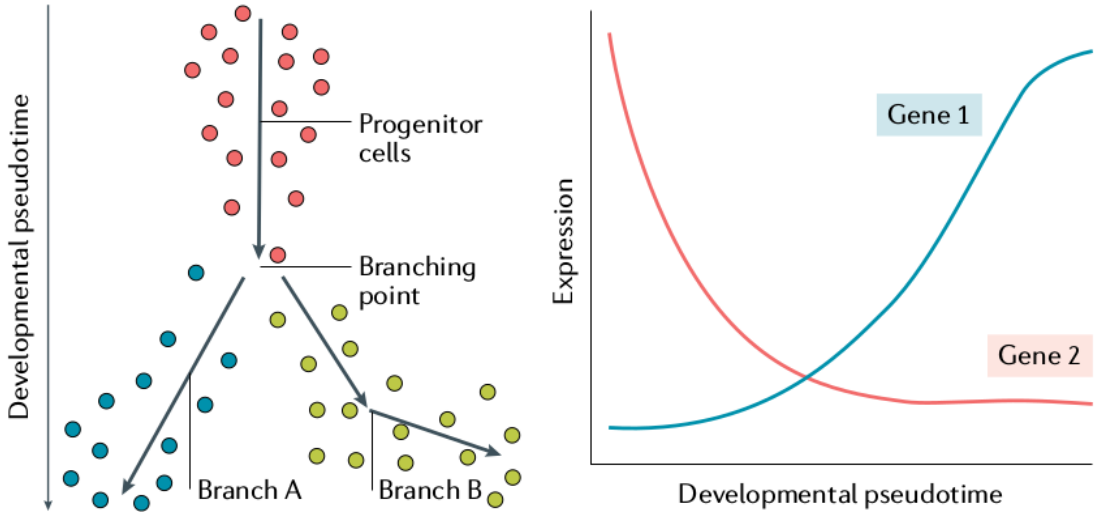
Cluster Annotation

Compare marker genes in data to marker genes in reference dataset



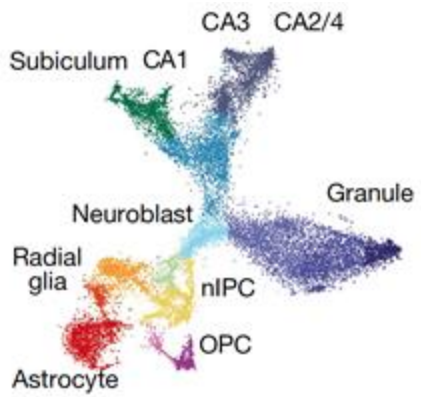
Trajectory analysis

- Clustering groups cells into cell identities and trajectory analysis captures transitions between cell identities, differentiation processes, or changes in biological function.
- **Inferring pseudotemporal orderings or trajectories of cells**: assumes that data lie on a connected manifold and labels cells with a continuous variable
- Multiple methods should be tested

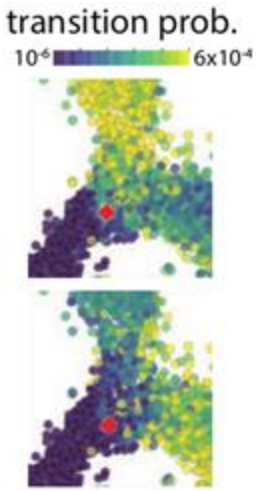


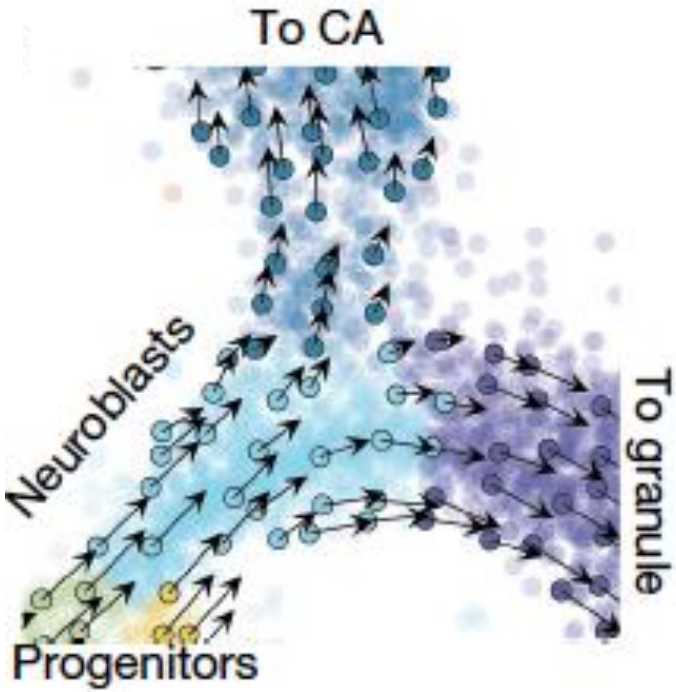
Wu et al., 2020 https://doi.org/10.1038/s41581-020-0262-0

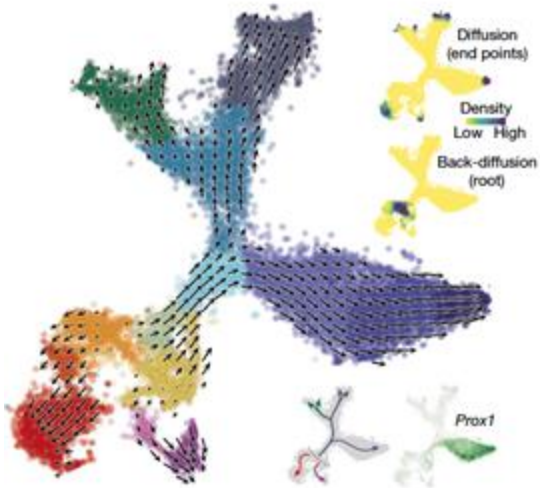
Fate Decisions of Major Neural Lineages



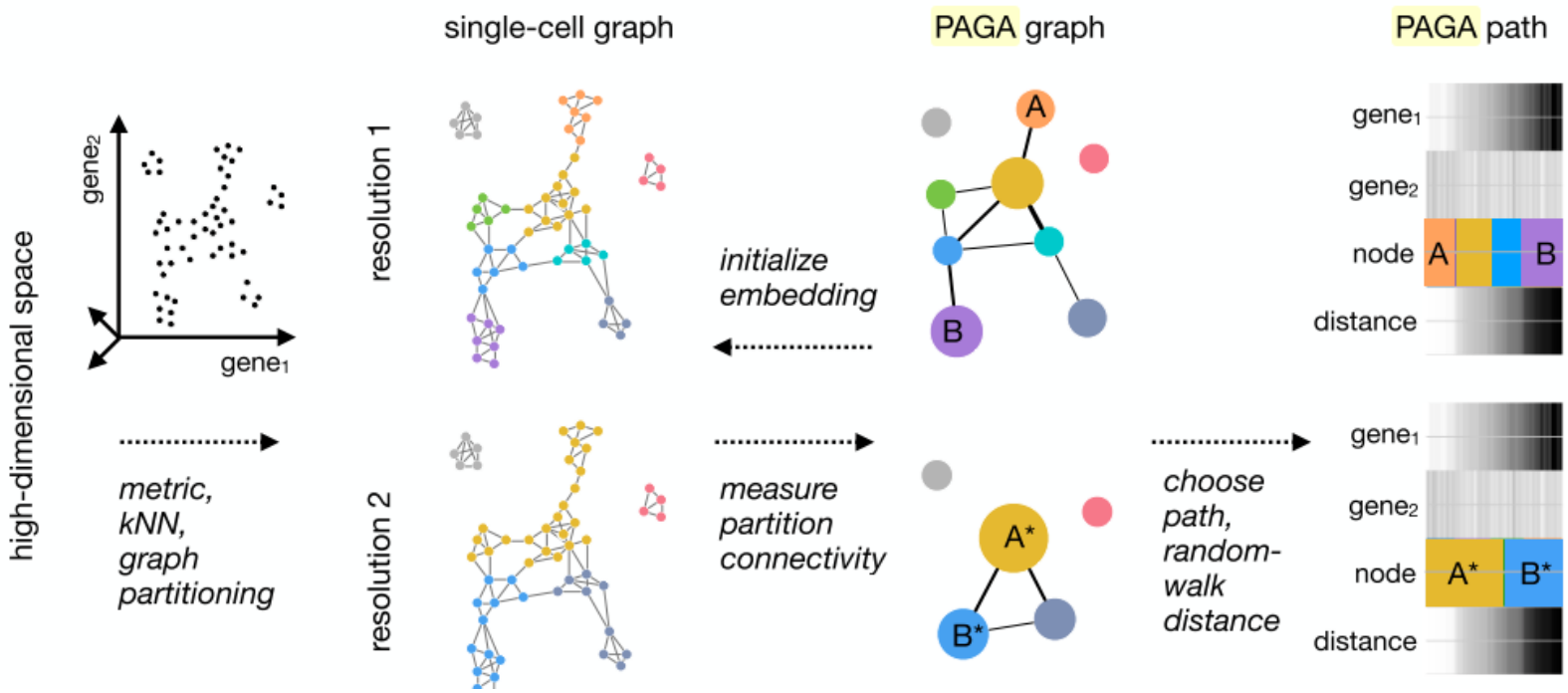
t-SNE







RNA Velocity can orient a lineage tree (predict future) without prior knowledge of the developmental process by assessing the ratio of spliced to unspliced RNA molecules present. High ratio of unspliced/spliced = increasing gene expression



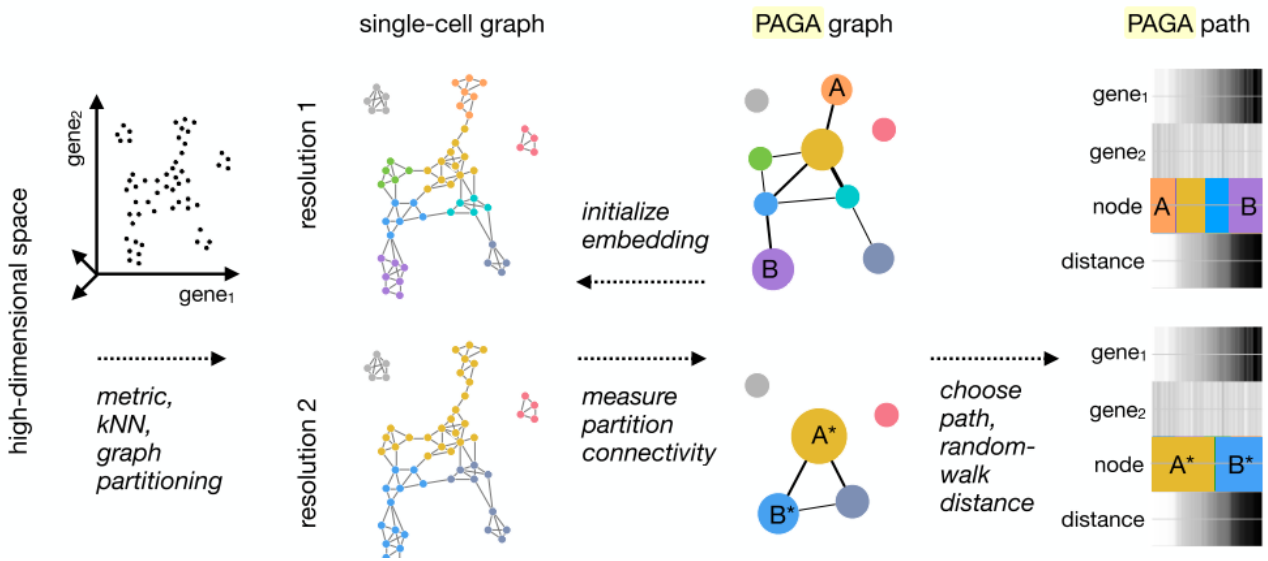
F. A. Wolf et al., Graph abstraction reconciles clustering with trajectory inference through a topology preserving map of single cells. bioRxiv 208819 [Preprint]. 25 October 2017.

PAGA (Partition-based graph abstraction)

- PAGA is a statistical model for the connectivity of groups of cells whose nodes correspond to cell groups and whose edge weights quantify the connectivity between groups. Groups are connected if their number of inter-edges exceeds a fraction of the number of inter-edges expected under random assignment.
- By quantifying the connectivity of partitions (groups, clusters) of the single-cell graph, PAGA generates a much simpler abstracted graph (PAGA graph) of partitions
- By averaging over single-cell paths, it becomes possible to trace a putative biological process from a progenitor to fates

Partition-based graph abstraction (PAGA)

- PAGA generates graph-like maps of cells that preserve both the continuous and disconnected structure in single cell mRNA-seq data, reconciling clustering and pseudotemporal ordering algorithms



F. A. Wolf et al., Graph abstraction reconciles clustering with trajectory inference through a topology preserving map of single cells. bioRxiv 208819 [Preprint]. 25 October 2017.