Dimensionality reduction and clustering

in 

Kevin Rue-Albrecht
University of Oxford (for the African Institute of Biomedical Science and Technology)
2022-11-15 (updated: 2022-11-04)

Learning objectives

Learning goals

- Describe dimensionality reduction and clustering methods.
- Explain the difference between dimensionality reduction and clustering.
- Learn how to use those methods as part of analytical workflow in R.

Learning objectives

- Apply dimensionality reduction methods to data; visualise and compare results.
- Identify and visualise features contributing most to principal components.
- Apply clustering methods to data; visualise and compare results.
- Integrate clustering and dimensionality reduction results for visualisation.

Prerequisites

- A computer with Microsoft Windows.
- A working installation of R.
- A working installation of RStudio Desktop.

Lessons

- Introduction to base R.
- Introduction to *ggplot2*.

Set up

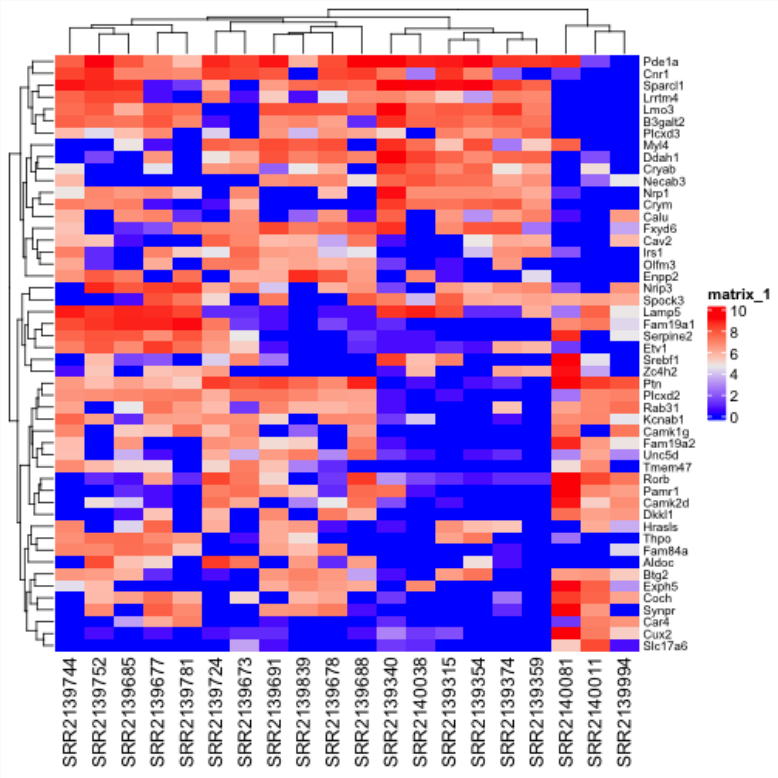
- Launch RStudio Desktop on your Windows computer.
- Make a copy of the template notebook for this lesson in your **git** repository.
- Make a symbolic link to your copy of the notebook in the RStudio project for this week.
- Open the notebook and follow along, editing and running the code as needed.

Visually extracting information from data

Data

	sample 1	sample 2	sample 3	sample 4	sample 5
gene 1	1	1	1	0	1
gene 2	1	1	1	0	1
gene 3	2	1	0	1	2
gene 4	0	1	1	0	1
gene 5	1	1	2	0	1
gene 6	1	1	1	1	0
gene 7	1	1	0	1	1
gene 8	0	3	1	1	2
gene 9	1	1	2	0	0
gene 10	1	2	1	0	1

Information



Sources of variation in data

Difference in data (e.g., expression) can come from multiple sources:

- Biological
- Technical

Either of those sources could be either:

- Of interest to study
- Considered a confounding covariate

Signal and noise both depend on your research question.

Confounding

Experimental design is crucial to ensure that sources of interesting variation are not confounded with independent sources of uninteresting variation (e.g. technical).

Confounded

Cell	Site	Treatment
1	S1	A
2	S1	A
3	S1	A
4	S1	A
5	S2	B
6	S2	B
7	S2	B
8	S2	B

Balanced

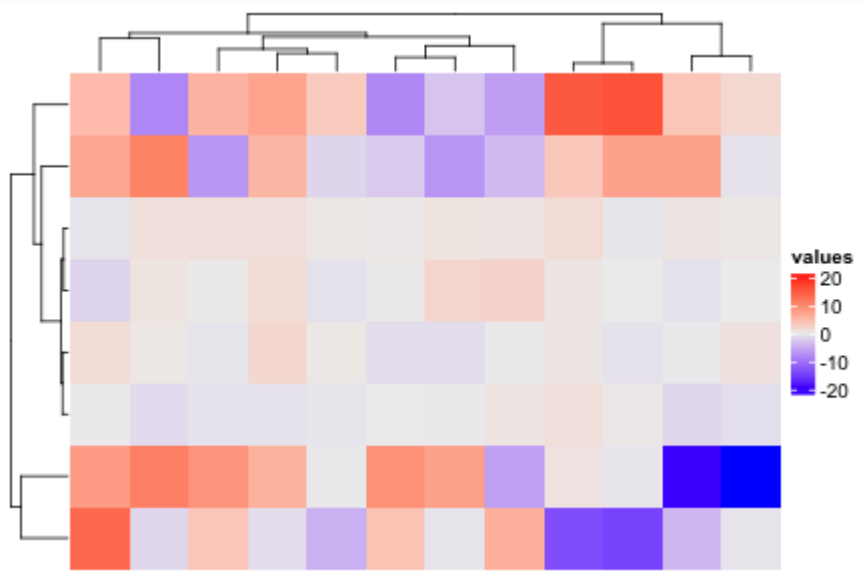
Cell	Site	Treatment
1	S1	A
2	S1	B
3	S1	A
4	S1	B
5	S2	A
6	S2	B
7	S2	A
8	S2	B

Feature selection

Many genes are not interesting because they don't vary much, or they don’t have enough counts.

Filtering for feature selection is needed to:

- Select genes that display useful variation.
- Reduce memory usage and computational cost/time.



Dimensionality reduction

We use dimensionality reduction methods to:

- Find structure in the data.
- Aid in visualization.

Unsupervised learning helps finding groups of homogeneous items

- Many approaches to do this (e.g. PCA, t-SNE, UMAP)

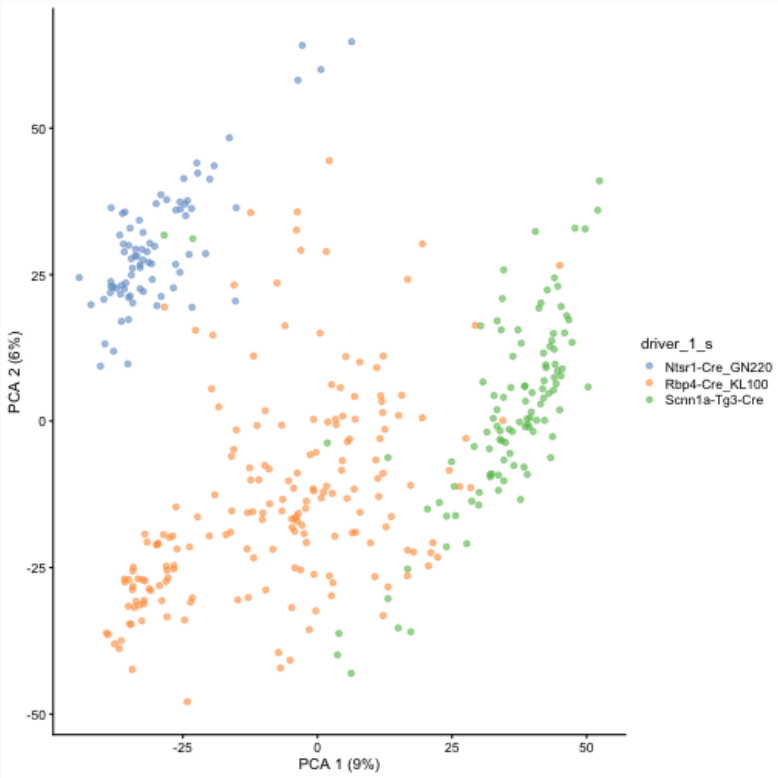
	sample 1	sample 2	sample 3	sample 4	sample 5
gene 1	0	1	2	0	1
gene 2	3	0	3	1	1
gene 3	0	0	2	3	1
gene 4	1	1	0	1	1
gene 5	0	1	1	0	1
gene 6	2	0	3	0	1
gene 7	0	0	3	1	1
gene 8	2	1	0	2	1
gene 9	0	3	0	0	0
gene 10	0	0	0	0	1

	dim 1	dim 2
sample 1	13.430388	-0.7356440
sample 2	-2.145794	-0.3763417
sample 3	-1.795565	-6.8166048
sample 4	-1.001907	-3.2427027
sample 5	7.126663	0.6016044

Principal component analysis (PCA)

Goals

- Find linear combination of variables to create principal components (PCs).
- Maintain most variance in the data (for given number of PCs).
- PCs are uncorrelated (orthogonal to each other) and ordered with respect to the percentage of variance explained.

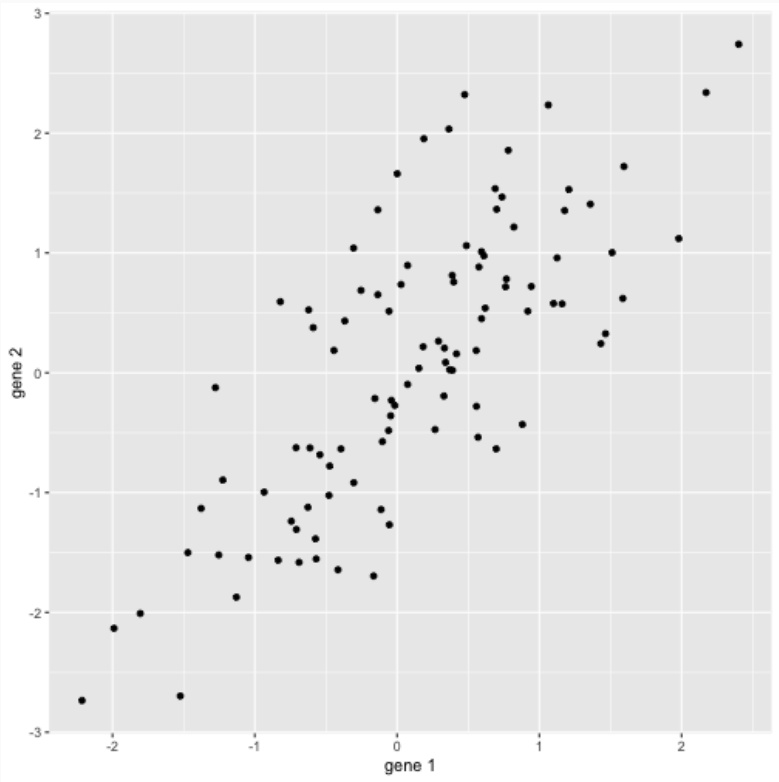


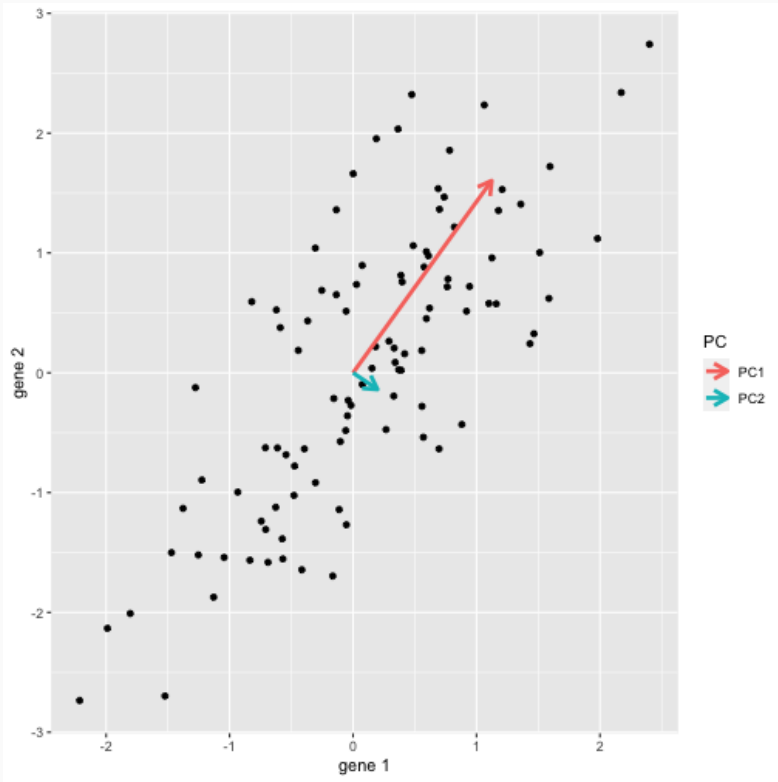
Assumptions

- Relationship between variables is linear!
- Not optimal for non-linear data structures.

```
pca <- prcomp(x, center = TRUE, scale. = FALSE, ...)
```

PCA example



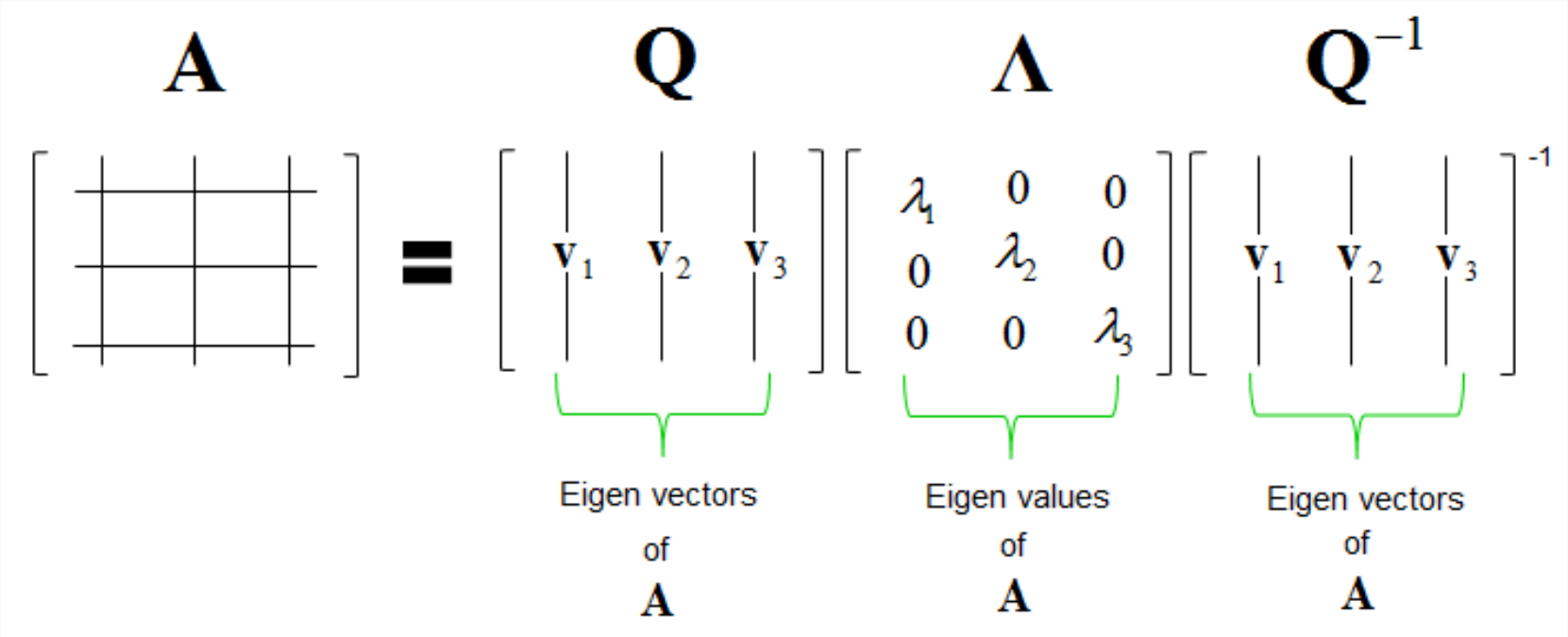


$$PC1 = \beta_{(1,1)} * gene_1 + \beta_{(1,2)} * gene_2$$

$$PC2 = \beta_{(2,1)} * gene_1 + \beta_{(2,2)} * gene_2$$

Eigenvalue decomposition

Eigenvalue decomposition is matrix factorization algorithm.



In the context of PCA:

- An eigenvector represents a direction or axis.
- The corresponding eigenvalue represents variance along that eigenvector.

PCA

- First, center data.
  - Always best, unless you have a good reason not to.
- If comparing different units, scale data.
  - i.e., using correlation matrix instead of covariance matrix[1]
  - Genes have very different dynamic ranges!

Spectral decomposition = Eigen decomposition

- More intuitive, but computationally slower

Singular Value Decomposition (SVD)

- Equivalent, faster

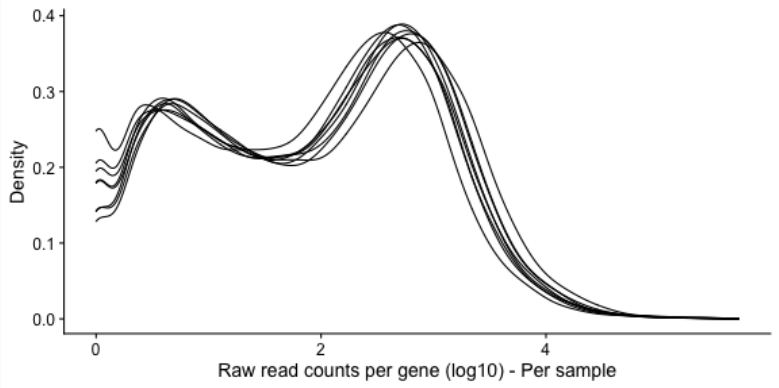
Approach:

- The idea is to select a smaller number of dimensions by taking the first k out of n eigenvectors that explain as much of the variability of the data as possible.
- How to choose k?

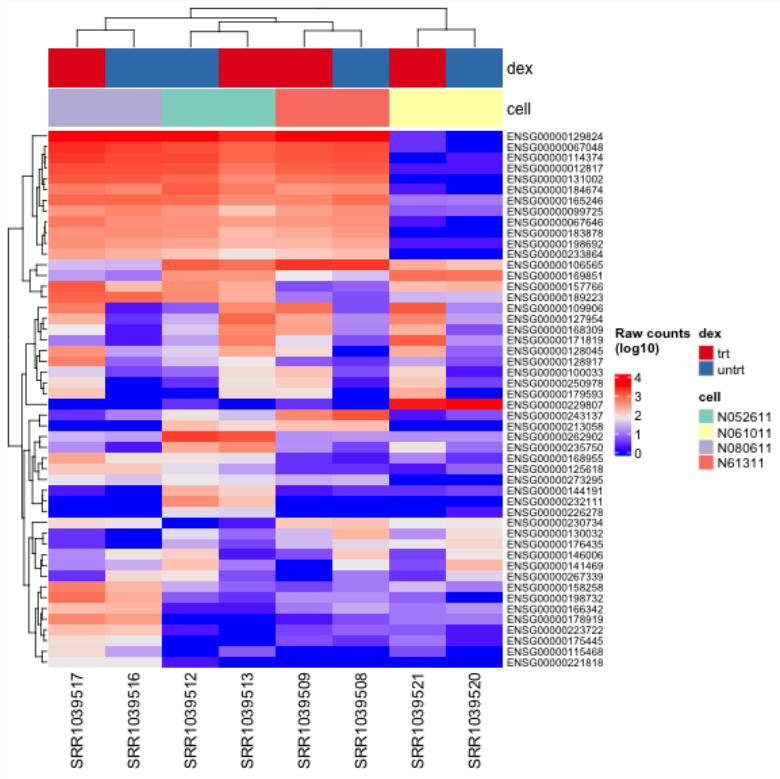
See also: Towards data science, "Correlation matrix and covariance matrix"

Expression data example

Airway smooth muscle cells expression profiling by high throughput sequencing; GSE52778.

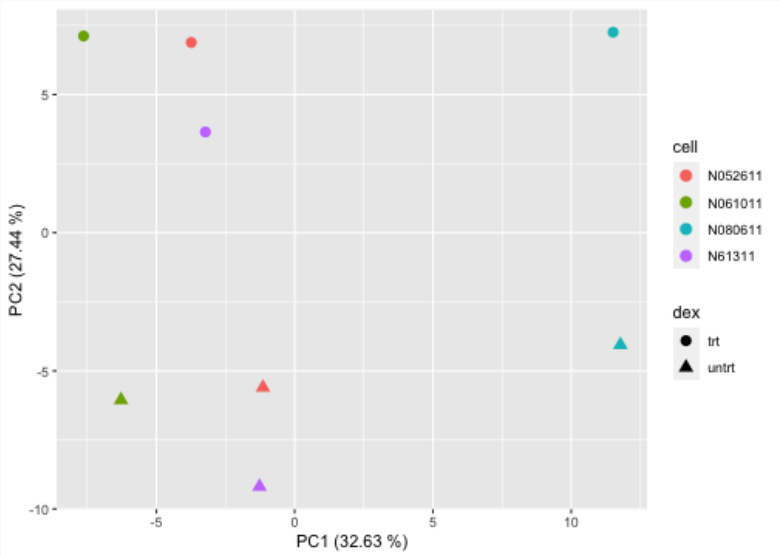


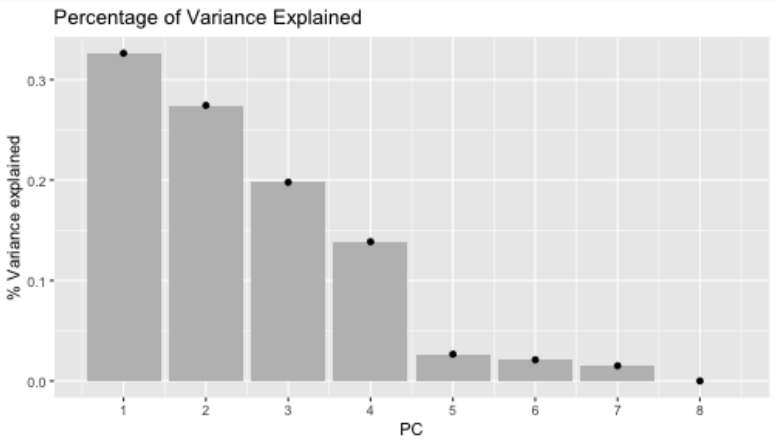
- `dex` : treatment with dexamethasone
- `cell` : cell line

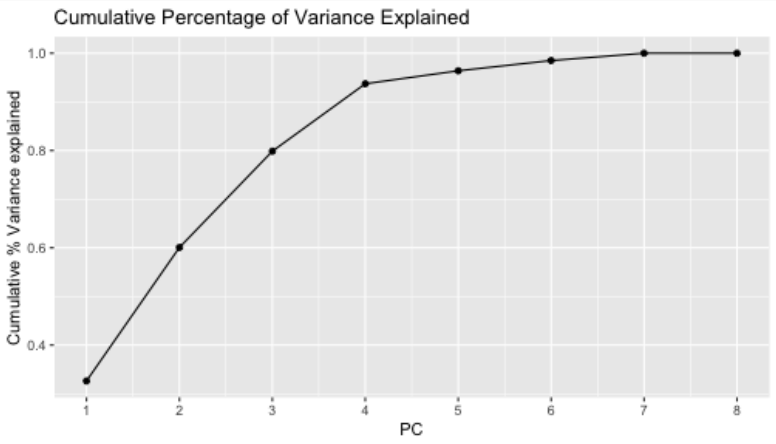


Expression data example

Airway smooth muscle cells expression profiling by high throughput sequencing; GSE52778.







Percentage variance explained:

$$pct\_var = sdev^2 / sum(sdev^2)$$

PCA - Loadings / Rotation matrix

```
# Visualise loadings for the first five genes and principal components
pca$rotation[1:5, 1:5]
```

```
##                        PC1          PC2        PC3          PC4          PC5
## ENSG00000129824  0.1255530 -0.007276016  0.2151036  0.006015145 -0.001731216
## ENSG00000229807 -0.1293194  0.006308624 -0.2077728 -0.013730312 -0.088395451
## ENSG00000114374  0.1139619 -0.018080785  0.1851760  0.014011127  0.001171676
## ENSG00000067048  0.1134158 -0.001955943  0.1805077  0.004824954 -0.003910270
## ENSG00000131002  0.1140993 -0.012775024  0.1746800  0.010215936 -0.003090937
```

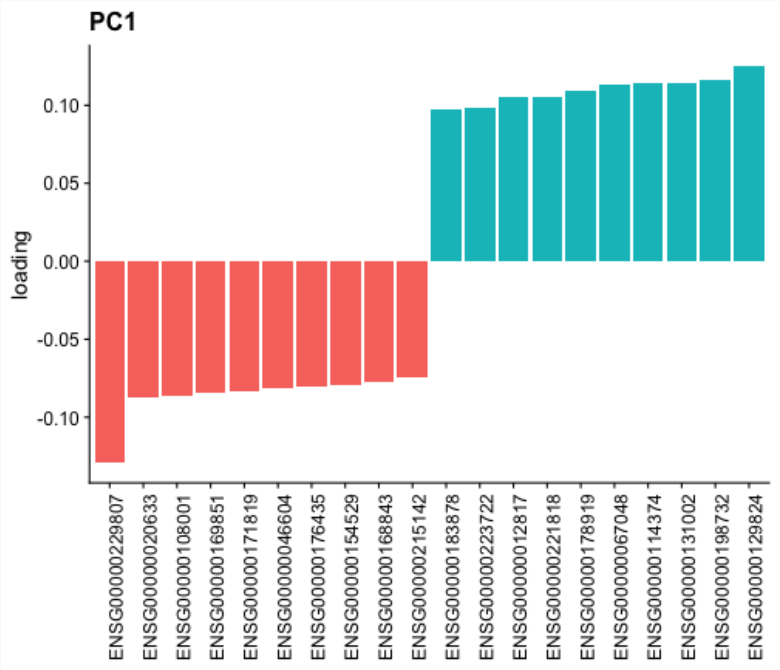
Meaning that for each cell:

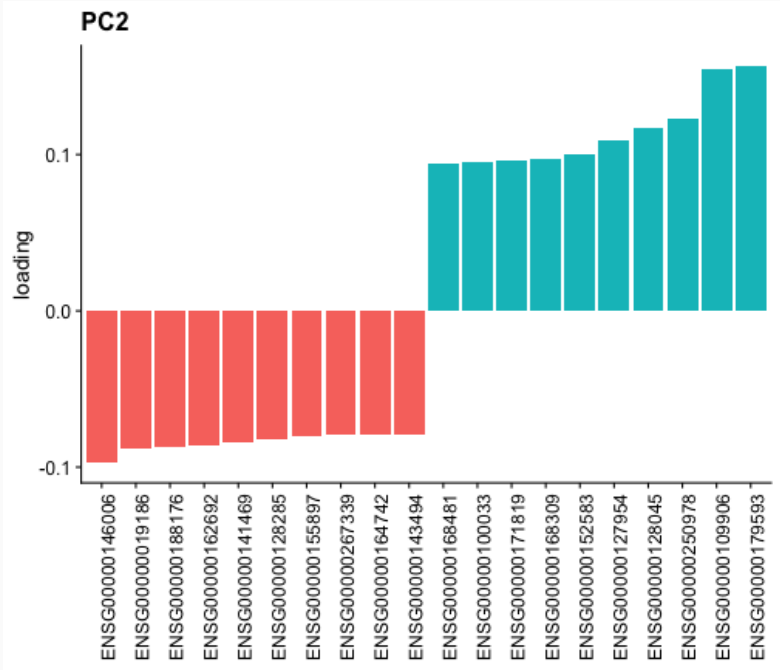
$$PC1_{(cell)} = 0.1255530 \times ENS00000129824_{(cell)} - 0.1293194 \times \ldots$$

Visualize top genes

Airway smooth muscle cells expression profiling by high throughput sequencing; GSE52778.

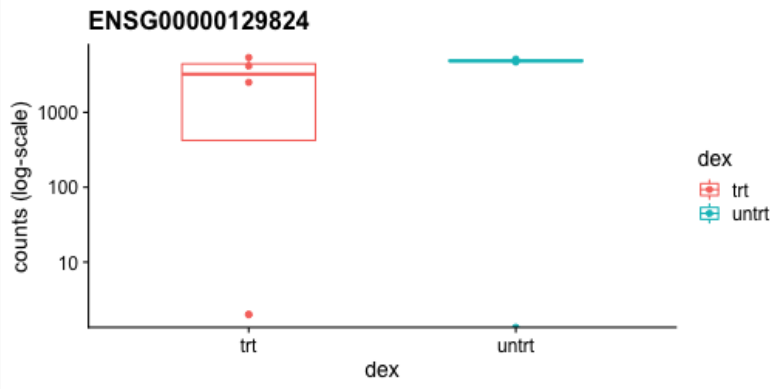
Top / Bottom loadings

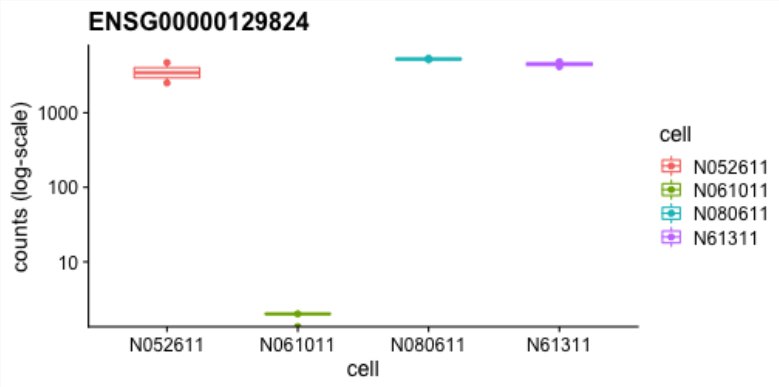




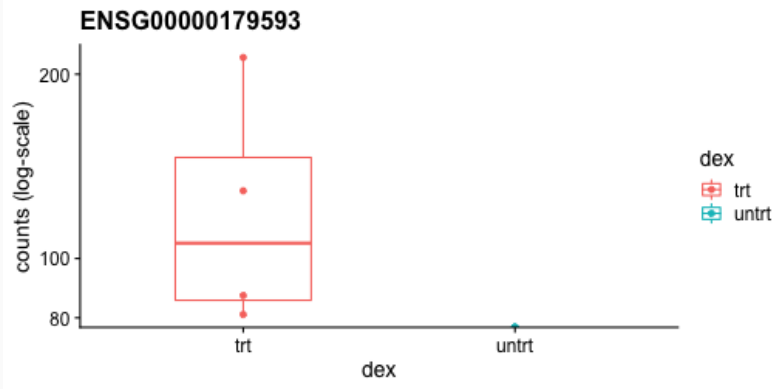
Visualize top genes - expression

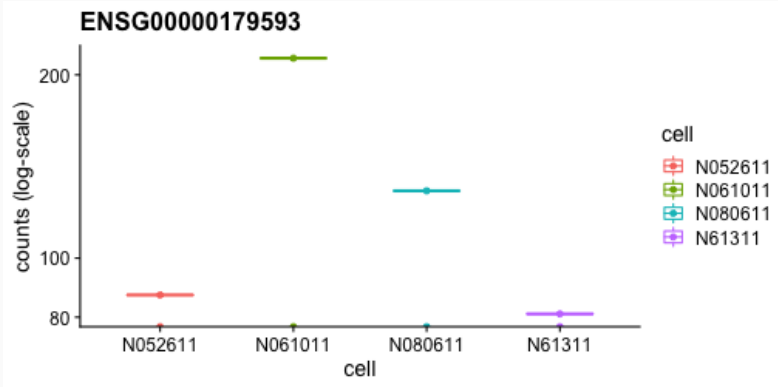
PC1





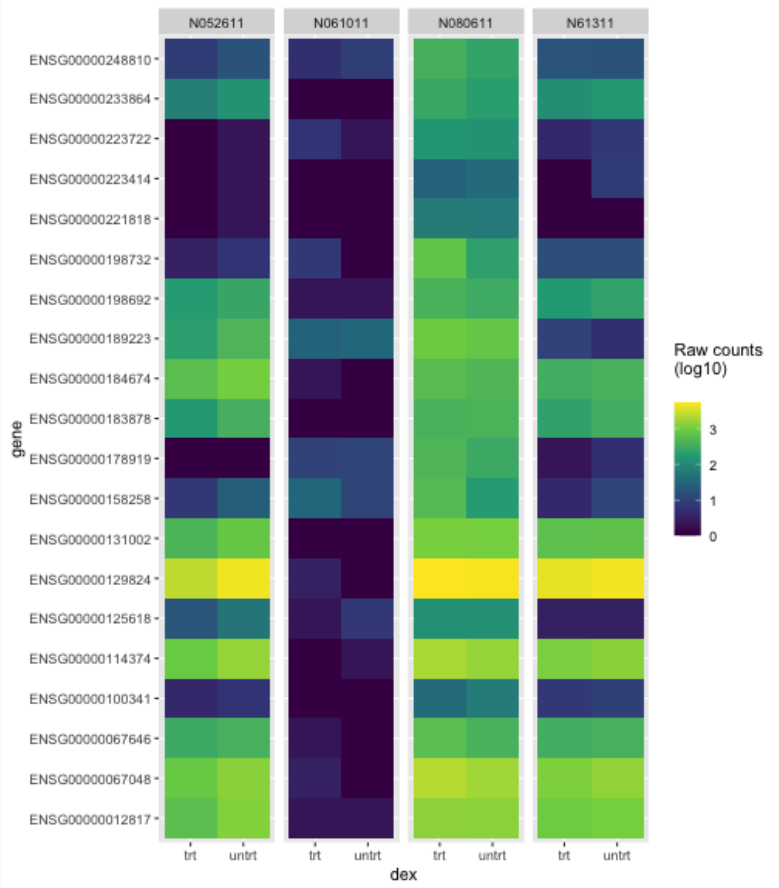
PC2



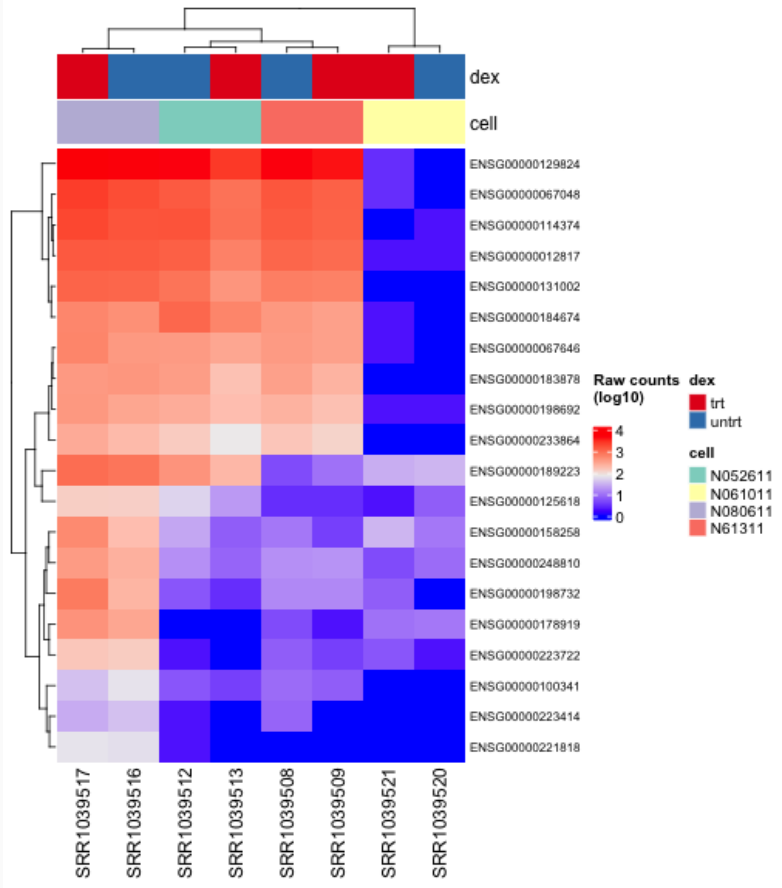


Visualize top genes - expression

ggplot2::geom_tile()

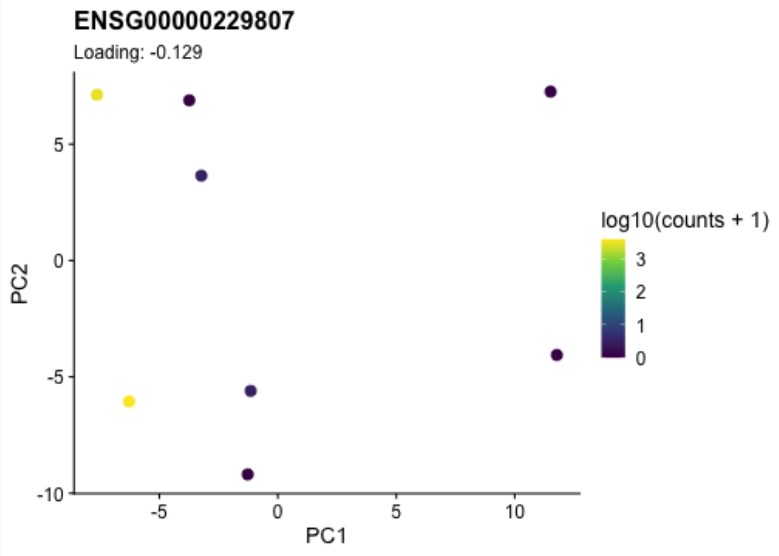


ComplexHeatmap::Heatmap()

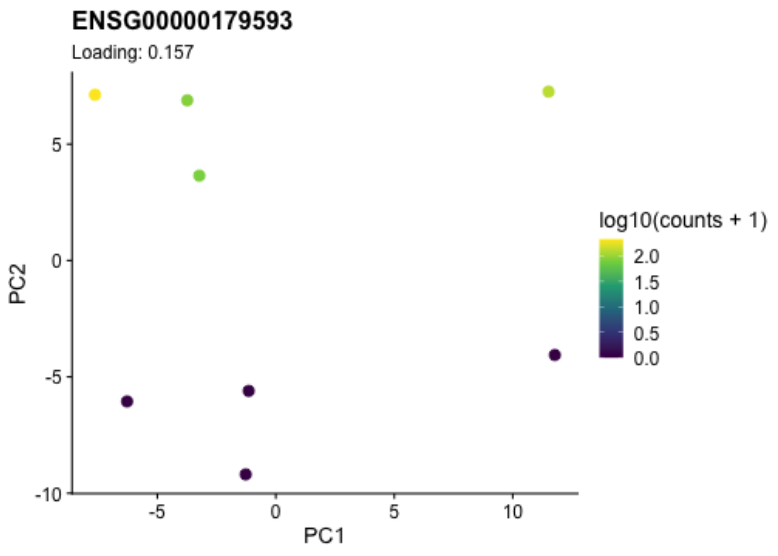


Visualize top genes - expression

PC1



PC2



Exercise

Setup

- Import the `iris` data set.
- Separate the matrix of measurements in a new object named `iris_features`.

Exercise

Apply Principal Components Analysis (PCA)

The `prcomp()` function allows you to standardise the data as part of the principal components analysis itself.

- Apply PCA while centering and scaling the matrix of features.
- Examine the PCA output. Display the loading of each feature on each principal component.
- Use the return value of the PCA to create a `data.frame` called `pca_iris_dataframe` that contains the coordinates projected on principal components.
- Visualise the PCA projection using `ggplot2::geom_point()`.

Bonus point

- Color data points according to their class label.
- Store the PCA plot as an object named `pca_iris_species`.

Exercise

Variable loading

- Color a scatter plot of PC1 and PC2 by the value of the variable most strongly associated with the first principal component.

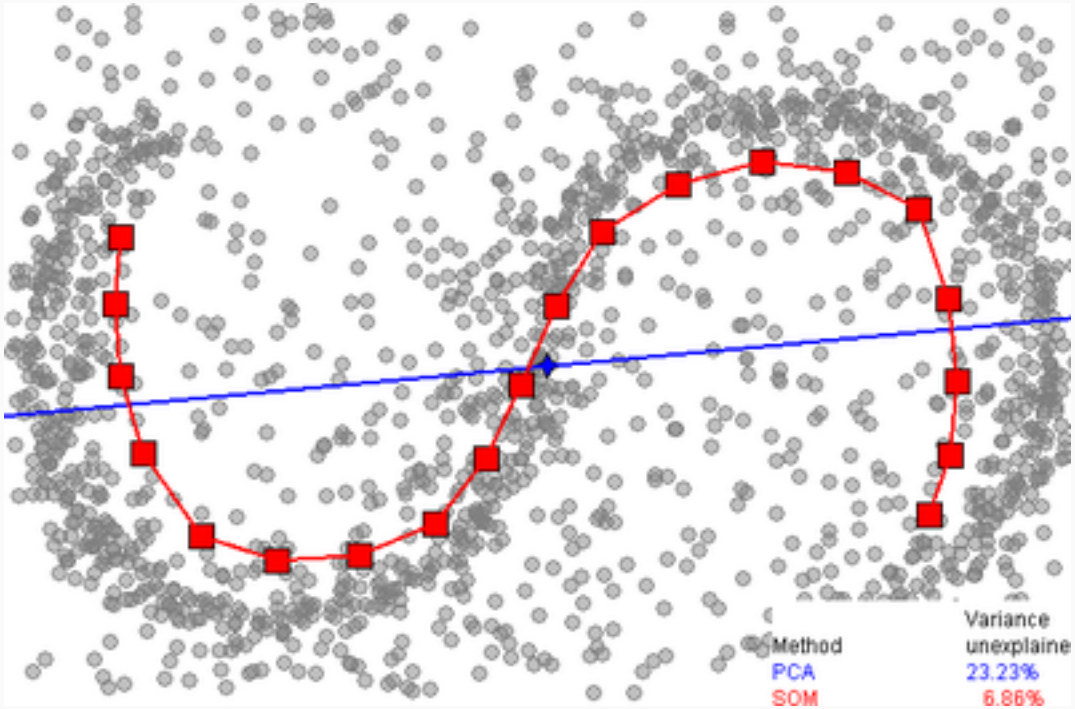
What do you observe?

Variance explained

- Compute the variance explained by principal components, using information present in the return value of the `prcomp()` function.
- Visualise the variance explained by each principal component using `ggplot2::geom_col()`.

Non-linear dimensionality reduction techniques

In many cases, the relationship between features is not linear.



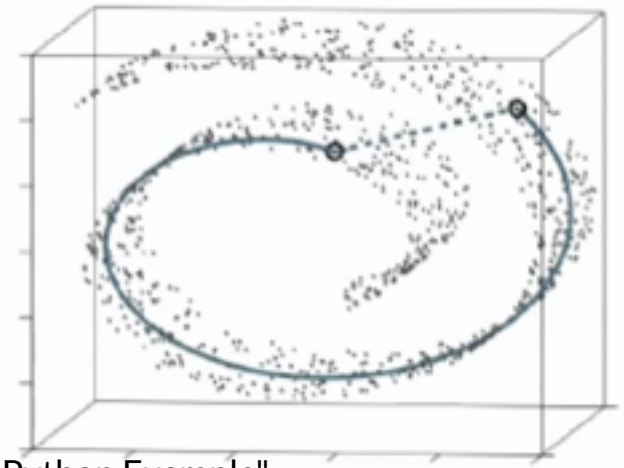
- Linear dimensionality reduction techniques like PCA (in blue) will fit their model as best as they can.
- But non-linear techniques will be able to accurately capture deviations non-linear patterns.
    - e.g., self organising map (SOM), t-SNE, UMAP.

t-SNE

t-Distributed Stochastic Neighbor Embedding

- Technique for dimensionality reduction that is particularly well suited for the visualization of high-dimensional datasets.
- Aims to place cells with similar local neighbourhoods in high-dimensional space together in low-dimensional space.
- Non-linear dimensionality reduction (as opposed to PCA).
- R implementation https://lvdmaaten.github.io/tsne/

- Preserve local structure / small pairwise distances / local similarities

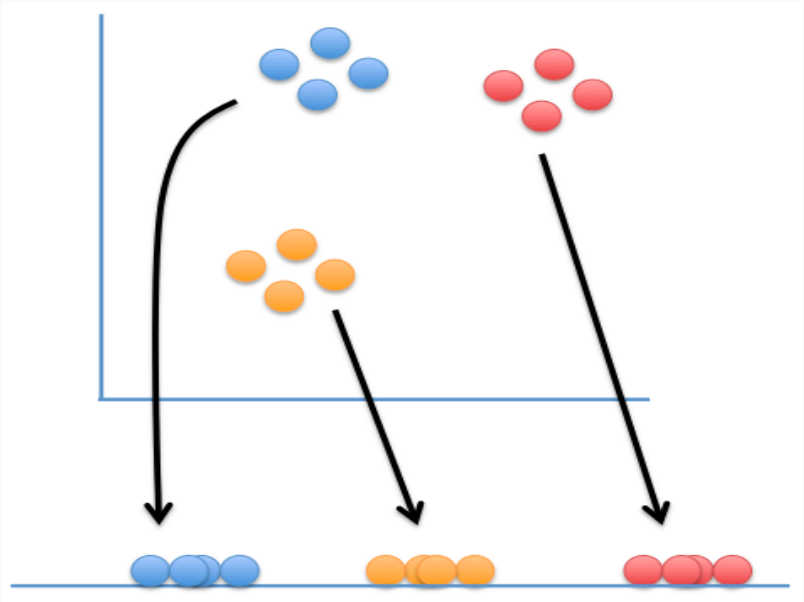


See also:

1. Towards data science, "An Introduction to t-SNE with Python Example".

t-SNE

Finds a way to project data into a low-dimension space (here, 1-D line), so that the clustering in the high-dimension space (here, 2-D scatter plot) is preserved.



See also:

1. StatQuest, "t-SNE, clearly explained!".
2. younesse.net, "Dimensionality reduction & visualization of representations".

UMAP

- Concept comparable to t-SNE.
- Faster than t-SNE, especially for large data sets.
- Better preservation of the global structure in the data.

There is no wrong choice. It doesn't hurt to run both and pick the best-looking one.

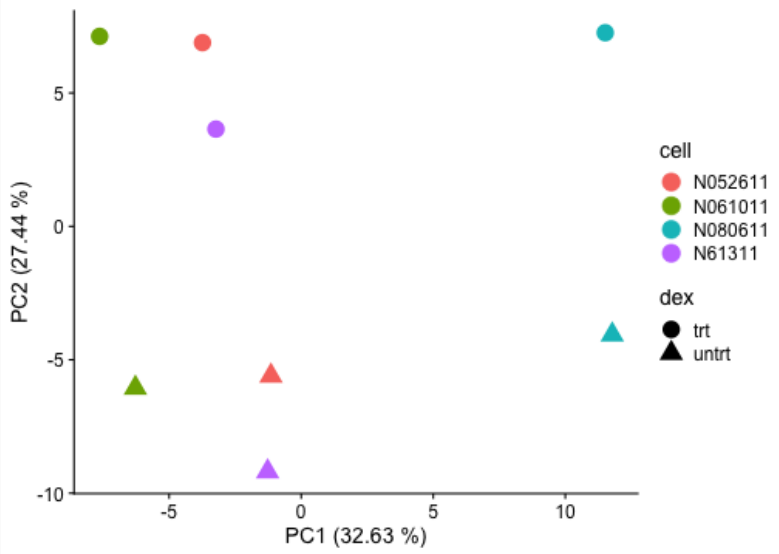
See also:

1. Understanding UMAP

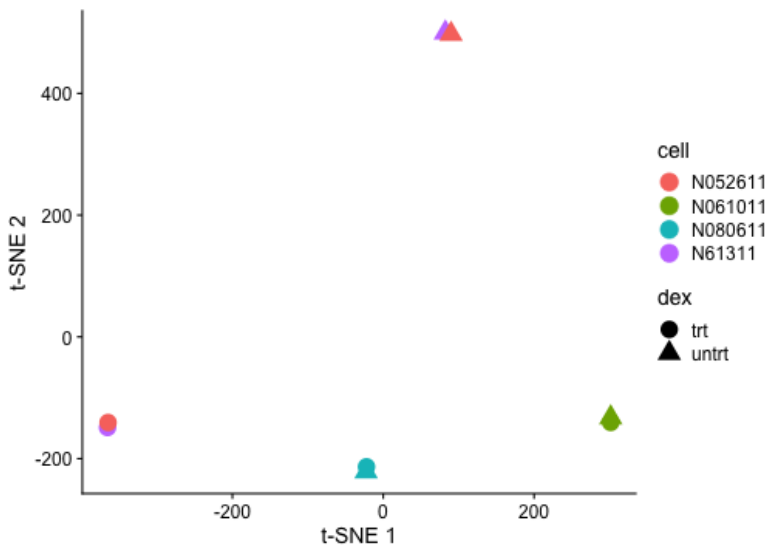
Expression data example

Airway smooth muscle cells expression profiling by high throughput sequencing; GSE52778.

PCA



t-SNE



Exercise

UMAP

- Apply UMAP on the output of the PCA.
- Inspect the UMAP output.
- Visualise the UMAP projection using `ggplot2::geom_point()`.

Bonus point

- Color data points according to their class label.
- Store the UMAP plot as an object named `umap_iris_species`.

Exercise

t-SNE

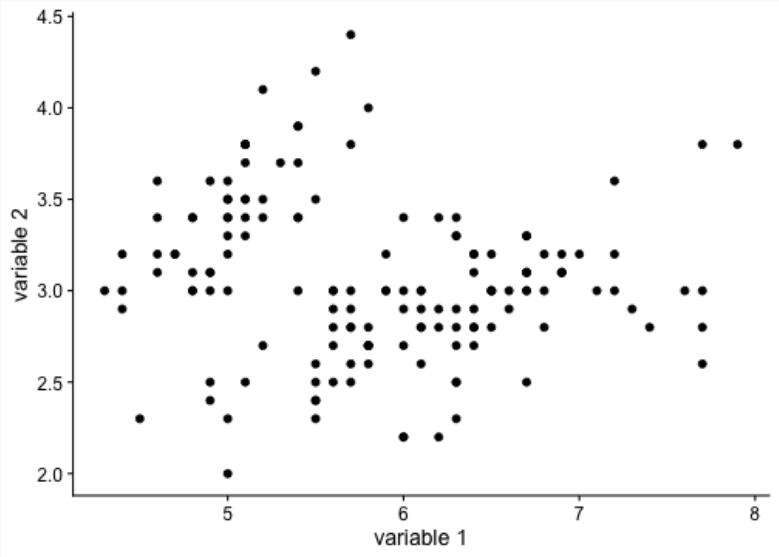
- Apply t-SNE and inspect the output.
- Use the return value of the t-SNE to create a `data.frame` called `tsne_iris_dataframe` that contains the coordinates.
- Visualise the t-SNE projection.

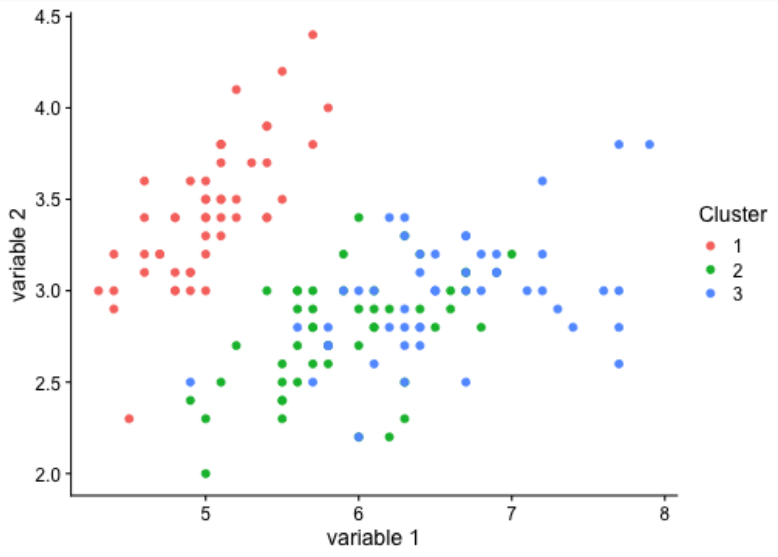
Bonus points

- Color data points according to their class label.
- Store the t-SNE plot as an object named `tsne_iris_species`.
- Combine PCA, UMAP and t-SNE plots in a single figure.

Clustering

- Technique for grouping of given data points and classification into groups.
  - In theory, points with similar features should belong to the same group.
  - Points with dissimilar features should belong to different groups.
  - Method of unsupervised learning (no known labels).
- Yields valuable insights from seeing what groups fall into after clustering
- Many methods (e.g. K-means clustering, hierarchical clustering)





K-Means Clustering

- Probably the most well known clustering algorithm (unsupervised).
- Easy to understand and implement.

Pros

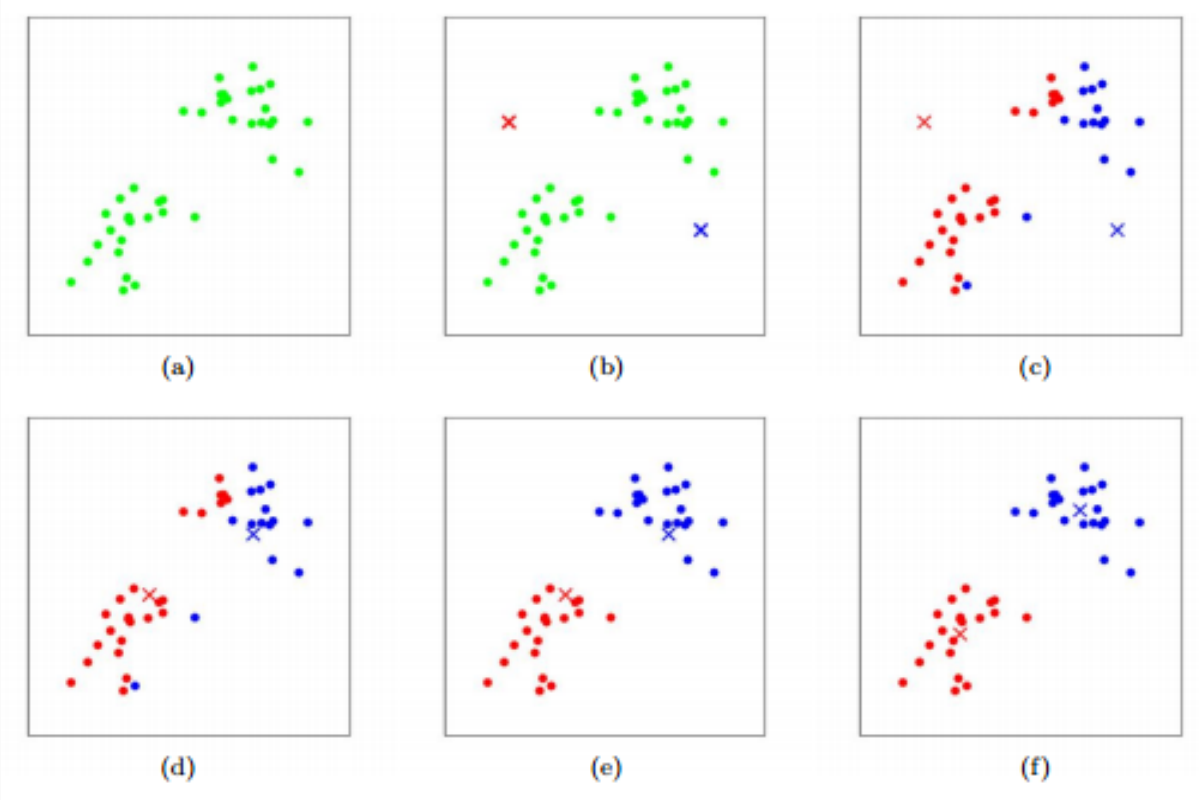
- Very fast

Cons

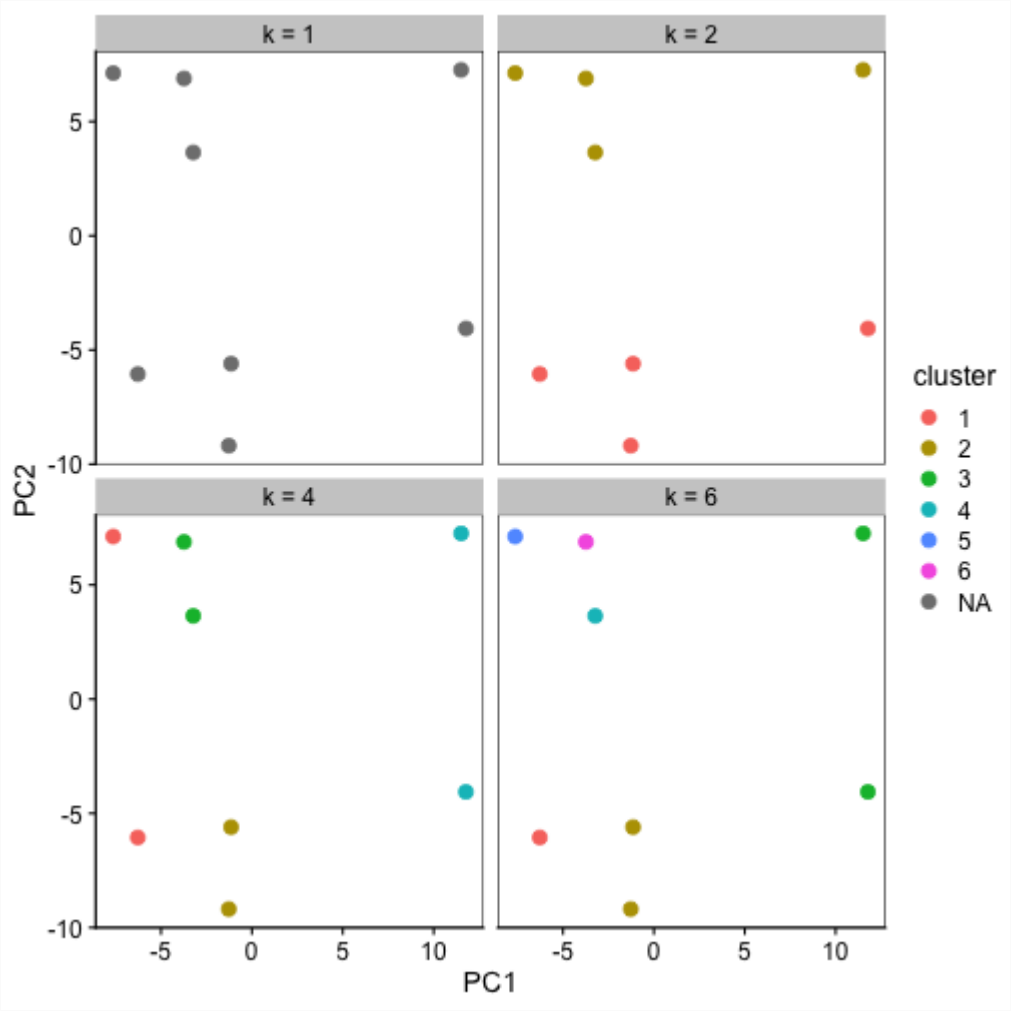
- Need to preselect the number of groups/classes – not always trivial
- Random choice of cluster centers can yield different clustering results on different attempts.

K-means clustering - Iterations

1. Initialise k centroids randomly.
2. Assign each data points to the nearest centroid.
3. Compute new centroid coordinates.
4. Repeat (2) and (3) until convergence, or for a maximum number of iterations allowed.



K-Means Clustering - How many clusters?

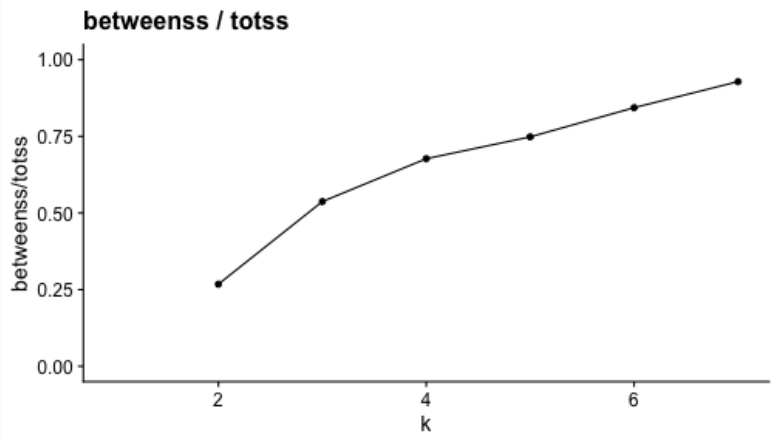


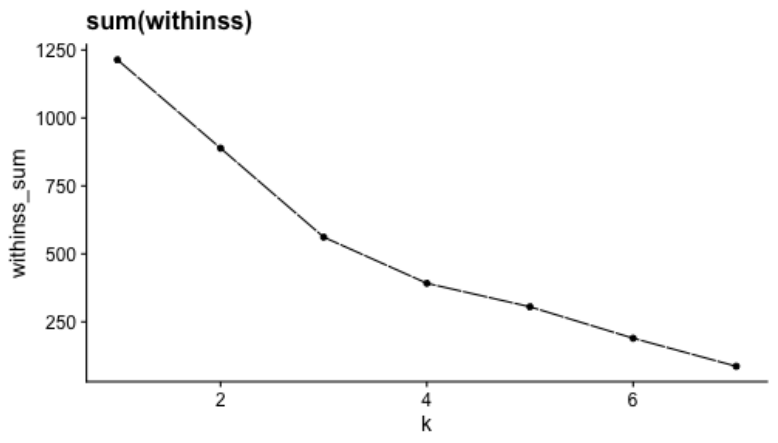
K-means clustering

To choose k, run multiple values and try to maximise `betweenss / totss`, which is a measure of how well the data is clustered.

- **Sum of squares between clusters:** how far points are *between* clusters (separation).
- **Sum of squares within clusters:** how close points are *within* clusters (compactness).

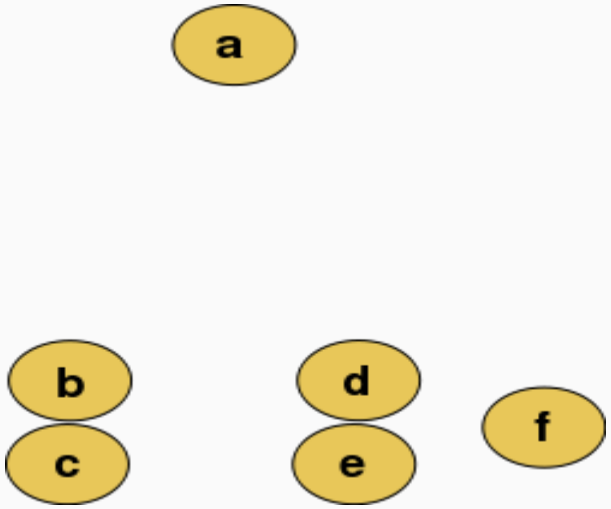
For good clustering we want small `sum(withinss)` and large `betweenss`, so this ratio we want to be as large as possible.

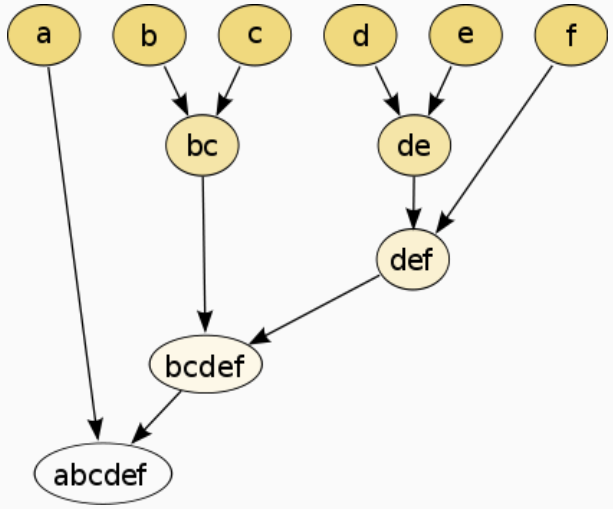




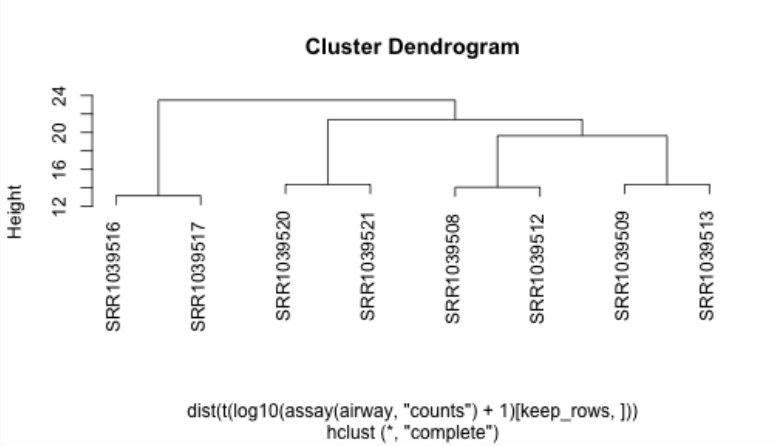
Hierarchical clustering

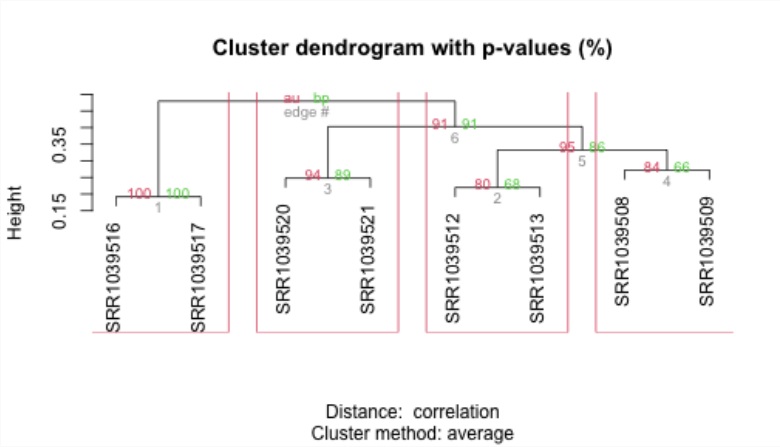
- Aims to build a hierarchy of classes
- To decide which clusters are similar/dissimilar, use a metric (distance between observations), e.g. Euclidean distance
- Either a bottom-up ('agglomerative') or a top-down ('divisive') approach.
  - **Agglomerative:** each cell is initially assigned to its own cluster and pairs of clusters are subsequently merged to create a hierarchy.
  - **Divisive:** starts with all observations in one cluster and then recursively split each cluster to form a hierarchy.

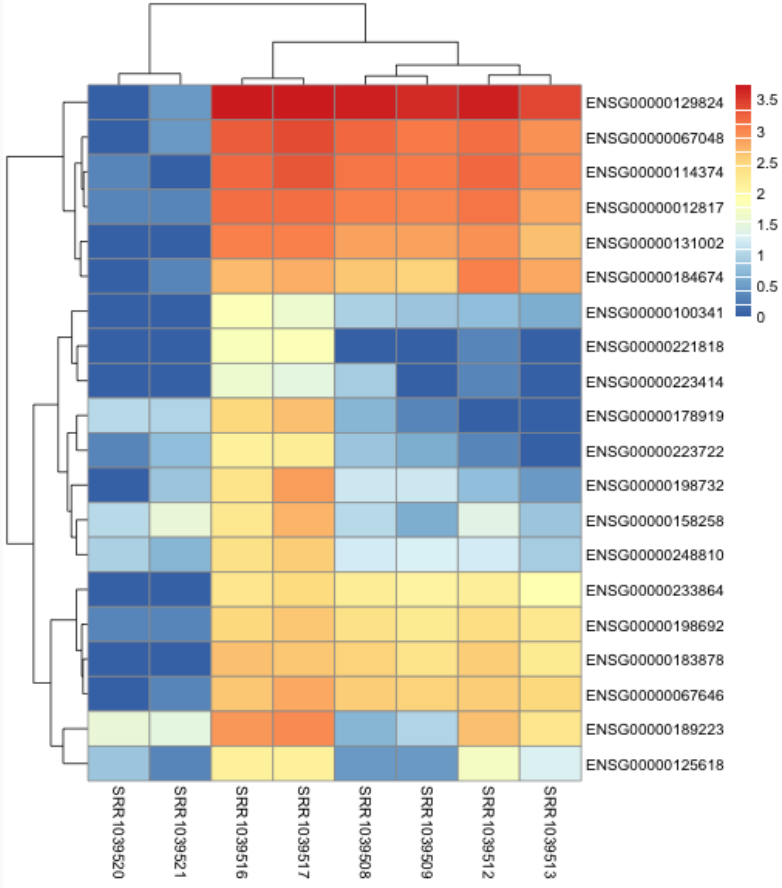




Hierarchical clustering



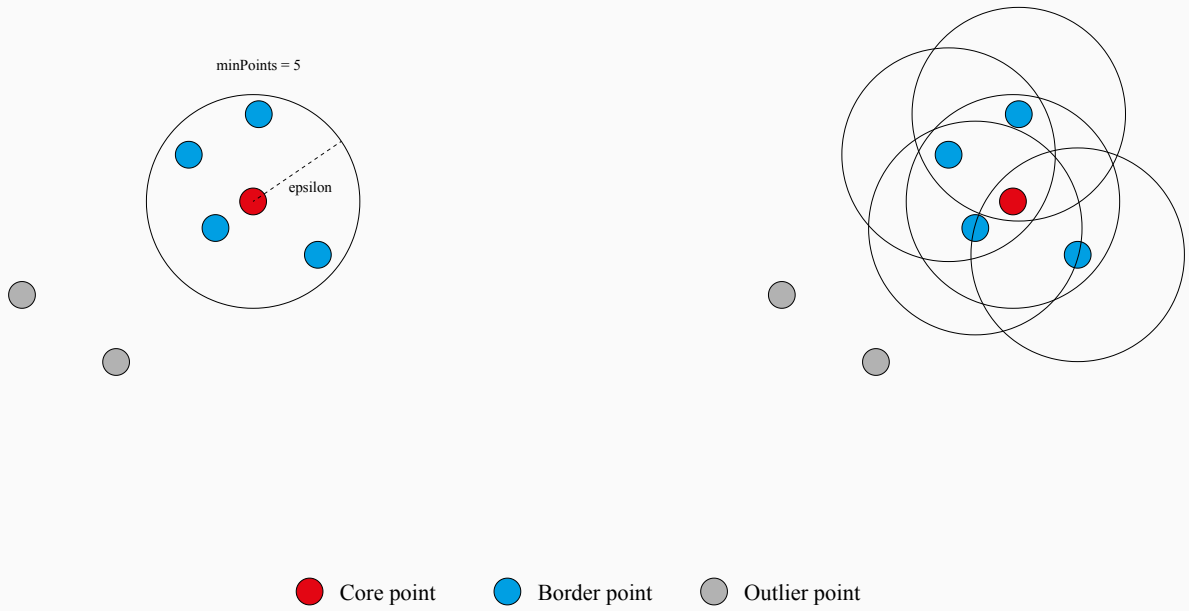




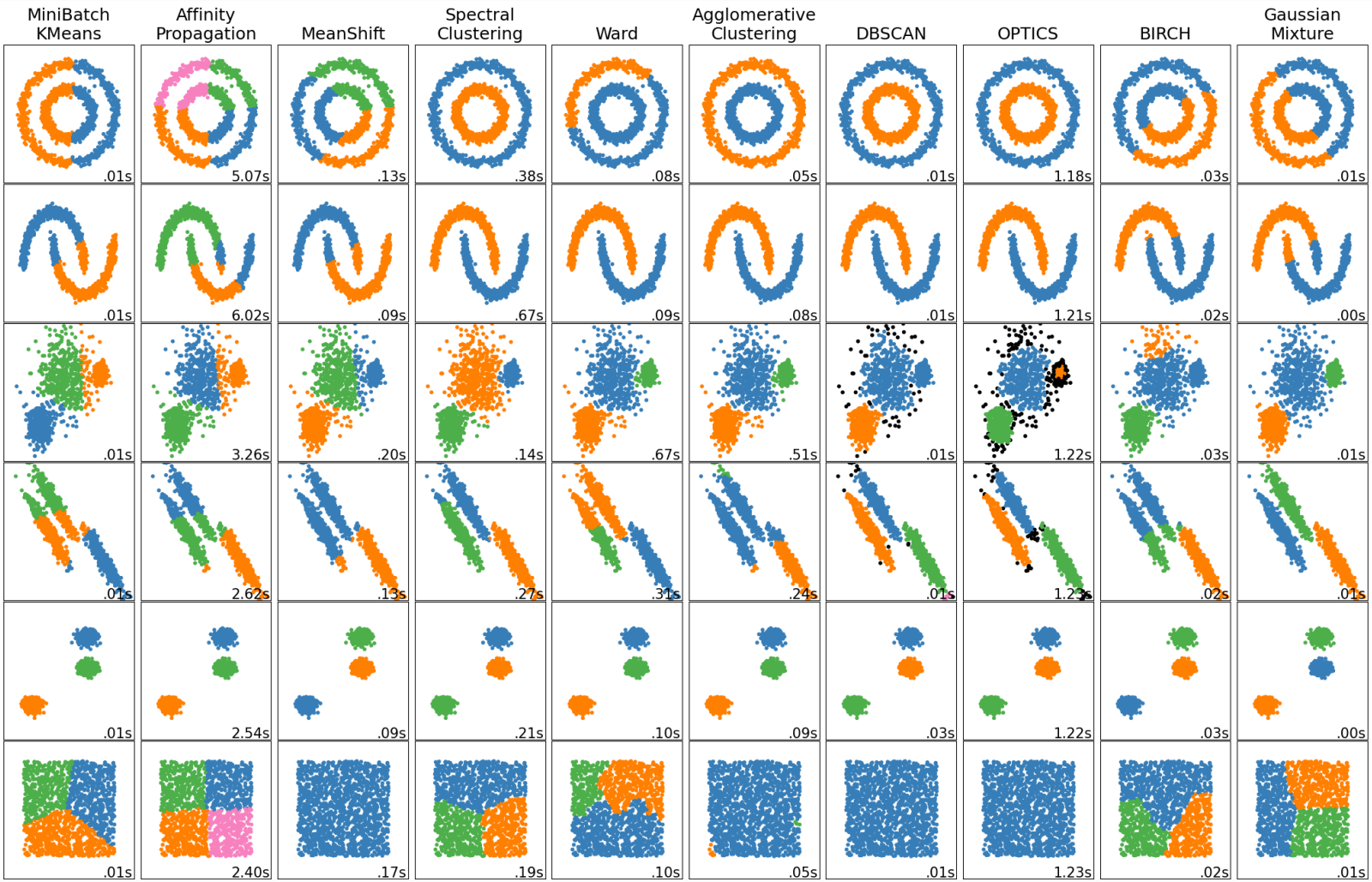
pvclust (CRAN)

Density-based clustering

- Two parameters:
  - Minimum number of points to initiate a cluster.
  - Maximum distance to search for neigbouring points.
- Core points have at least $minPoints - 1$ neigbours within $epsilon$ distance.
- Border points are located within $epsilon$ of a core point (without being a core point themselves).
- Outlier points are further than $epsilon$ from any other point.



Comparing clustering algorithms on toy datasets



Exercise

Hierarchical clustering

- Perform hierarchical clustering on the `iris_features` data set, using the `euclidean` distance and method `ward.D2`. Use the functions `dist()` and `hclust()`.
- Plot the clustering tree. Use the function `plot()`.

How many clusters would you call from a visual inspection of the tree?

- **Bonus point:** Color leaves by known species (use `dendextend`).
- Cut the tree in 3 clusters and extract the cluster label for each flower. Use the function `cutree()`.
- Repeat clustering using 3 other agglomeration methods:
  - `complete`
  - `average`
  - `single`
- Compare clustering results on scatter plots of the data.

Exercise

dbscan

- Apply `dbscan` to the `iris_features` data set.
- Visualise the `dbscan` cluster label on a scatter plot of the data.

hdbscan

- Apply `hdbscan` to the `iris_features` data set.
- Visualise the `hdbscan` cluster label on a scatter plot of the data.

Bonus point

- Combine the plots of `dbscan` and `hdbscan` into a single plot.

Exercise

K-means

- Apply K-means clustering to `iris_features` with K set to 3 clusters.
- Inspect the output.
- Extract the cluster labels.
- Extract the coordinates of the cluster centers.
- Construct a data frame that combines the `iris` dataset and the cluster label.
- Plot the data set as a scatter plot.
    - Color by cluster label.

Bonus point

- Add cluster centers as points in the plot.

Exercise

Cross-tabulation with ground truth

- Cross-tabulate cluster labels with known labels.

How many observations are mis-classified by K-means clustering?

Elbow plot

- Plot the "total within-cluster sum of squares" for K ranging from 2 to 10.

Do you agree that 3 is the optimal number of clusters for this data set?

Further reading

- *dimRed* vignette.
- Hitchhiker's Guide to Matrix Factorization and PCA

References

Wickham, H., M. Averick, et al. (2019NA). "Welcome to the tidyverse". In: *Journal of Open Source Software* 4.43, p. 1686. DOI: 10.21105/joss.01686.