Lecture 4: Exploratory Data Analysis (EDA)

Olexandr Isayev

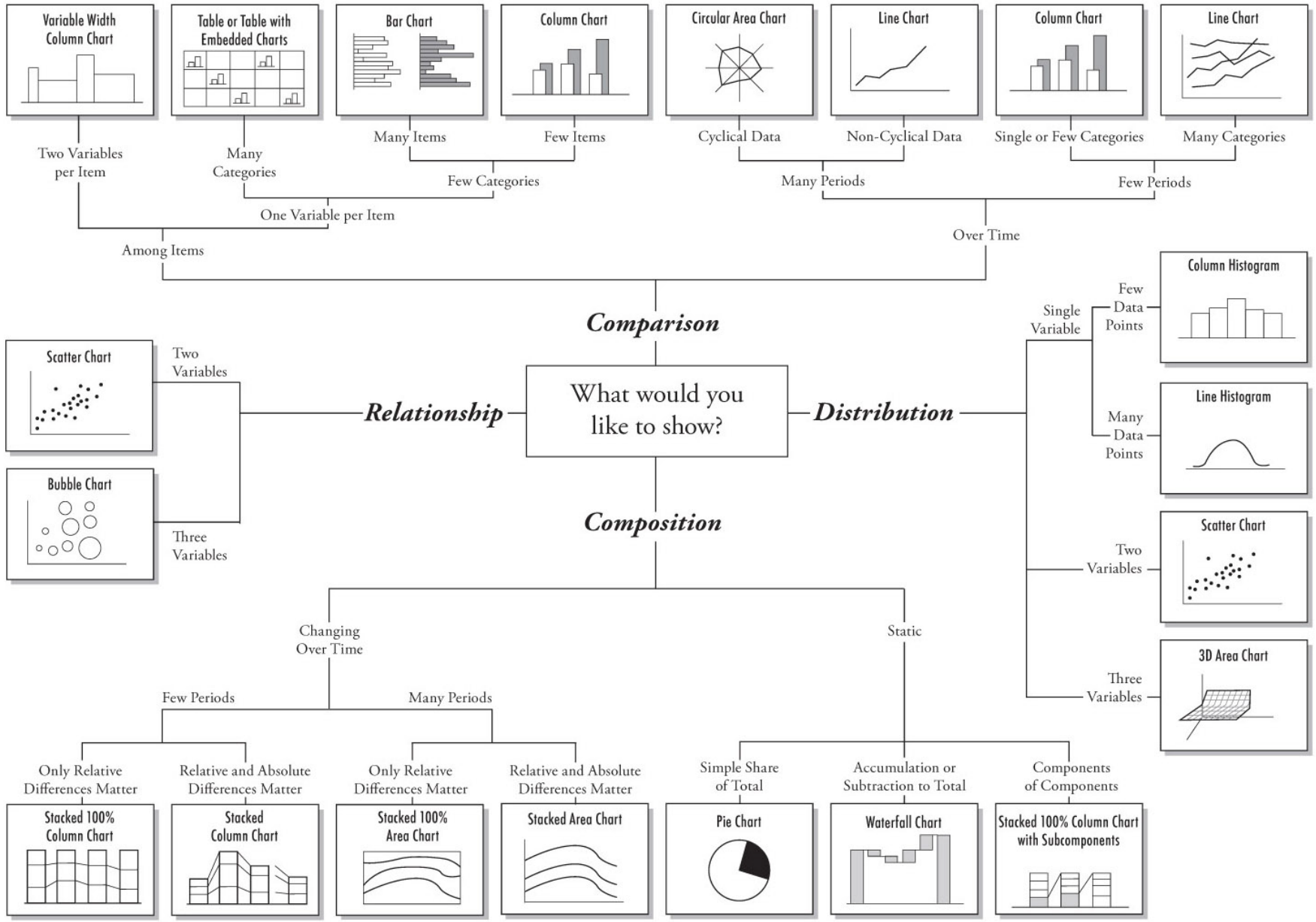
Department of Chemistry, CMU

olexandr@cmu.edu

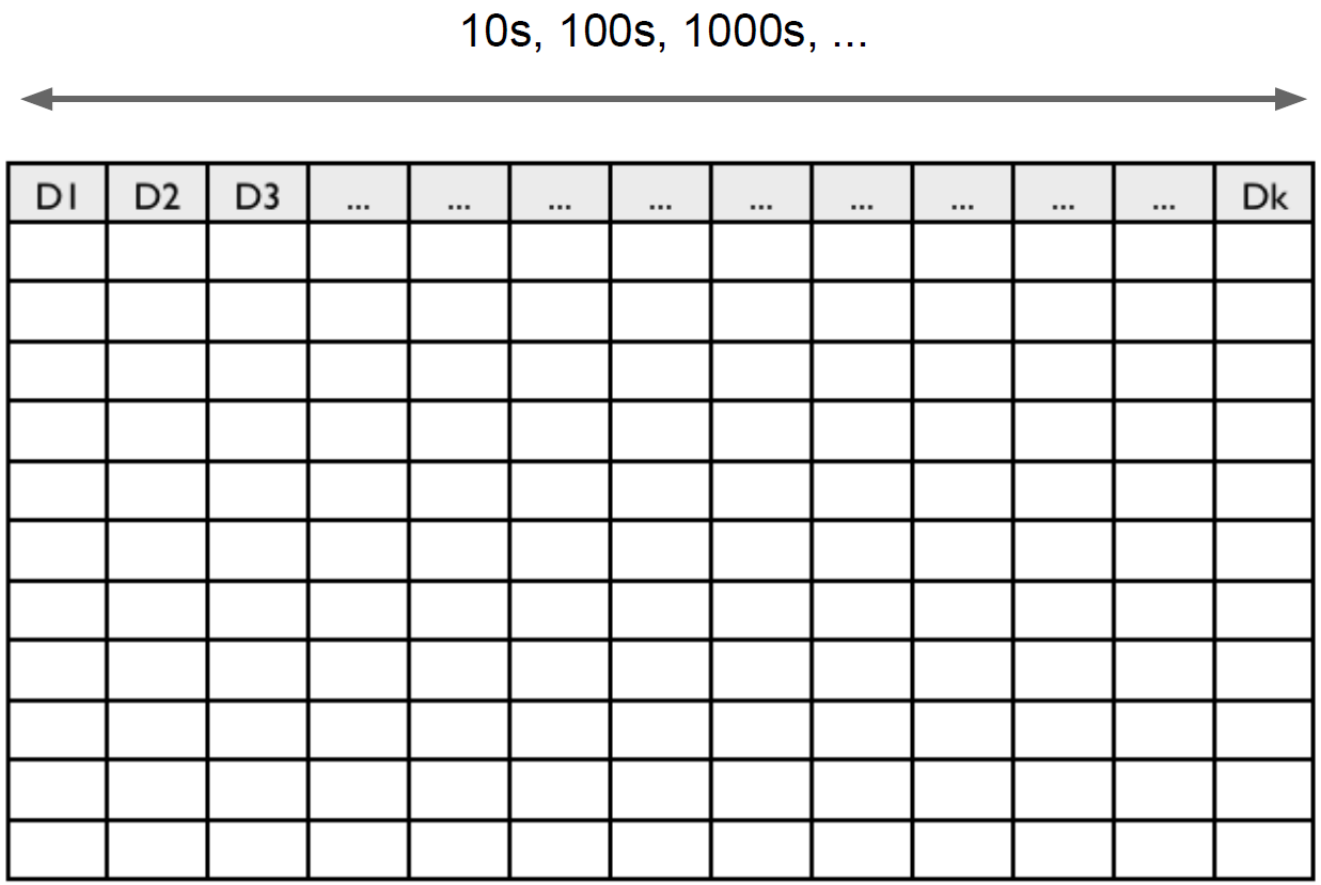
Outline

- Part 1: Data, data types, formats, basic analysis
- Part 2: Data Visualization (last week)
- Part 3: Dimensionality reduction (today)

Chart Suggestions—A Thought-Starter



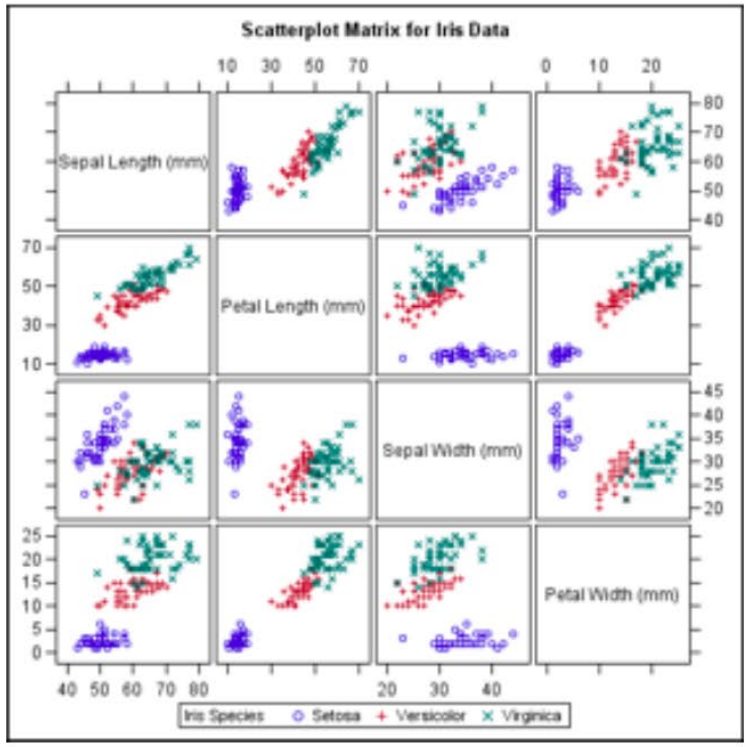
High-Dimensional Data Visualization



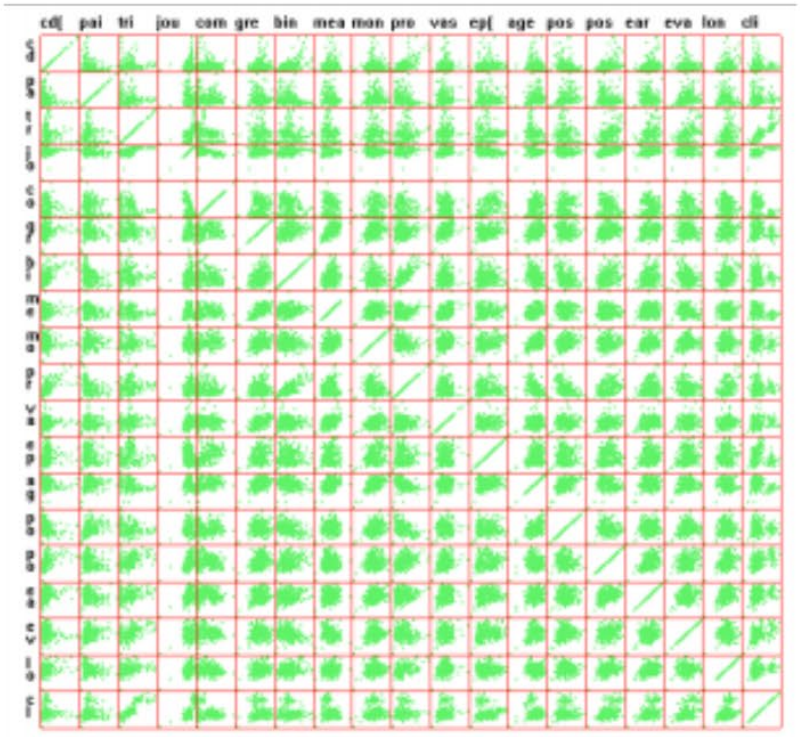
- Set of visual features very limited
- Resolution very limited
- Ability to make sense of it very limited!

Example: Scatter Plot Matrix

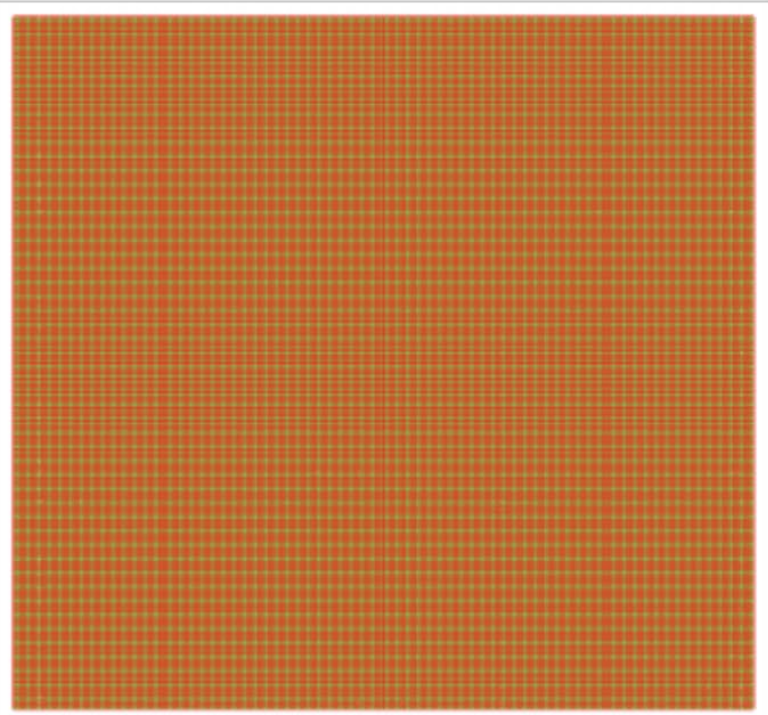
4 dimensions



20 dimensions

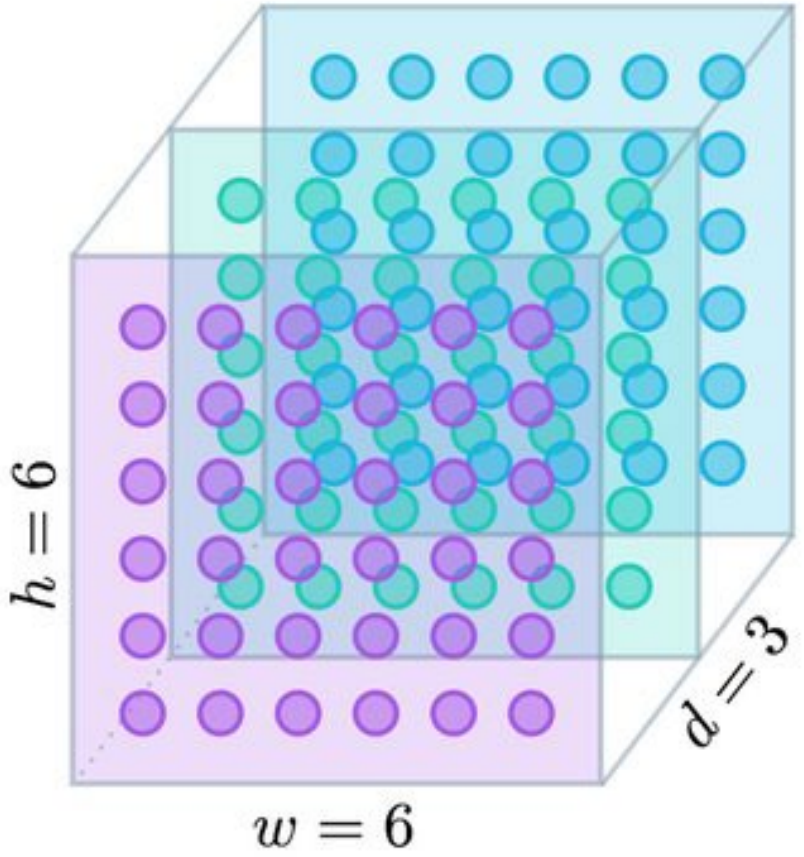


100 dimensions



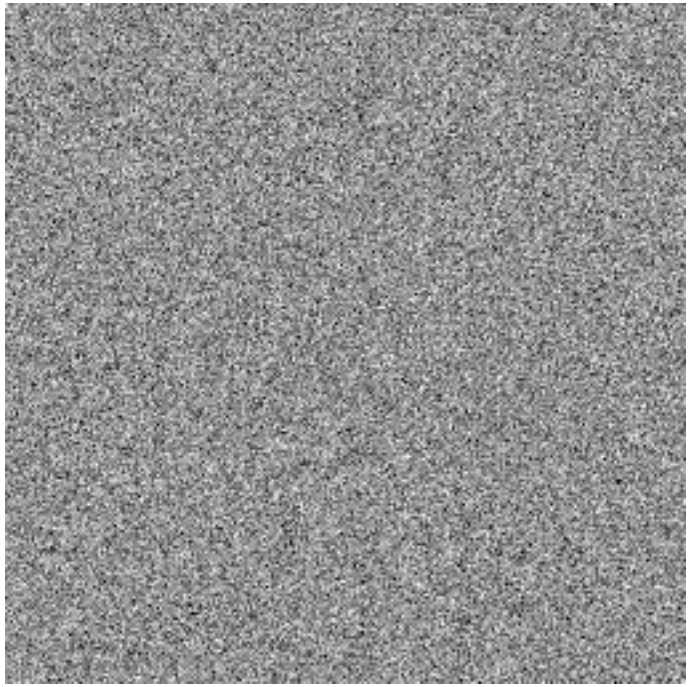
Image

- X, Y – dimensions of pixals
- 3 channels for RGB color



Observation from the real world

You will have to wait a long time before a sample of white noise looks like a natural image.



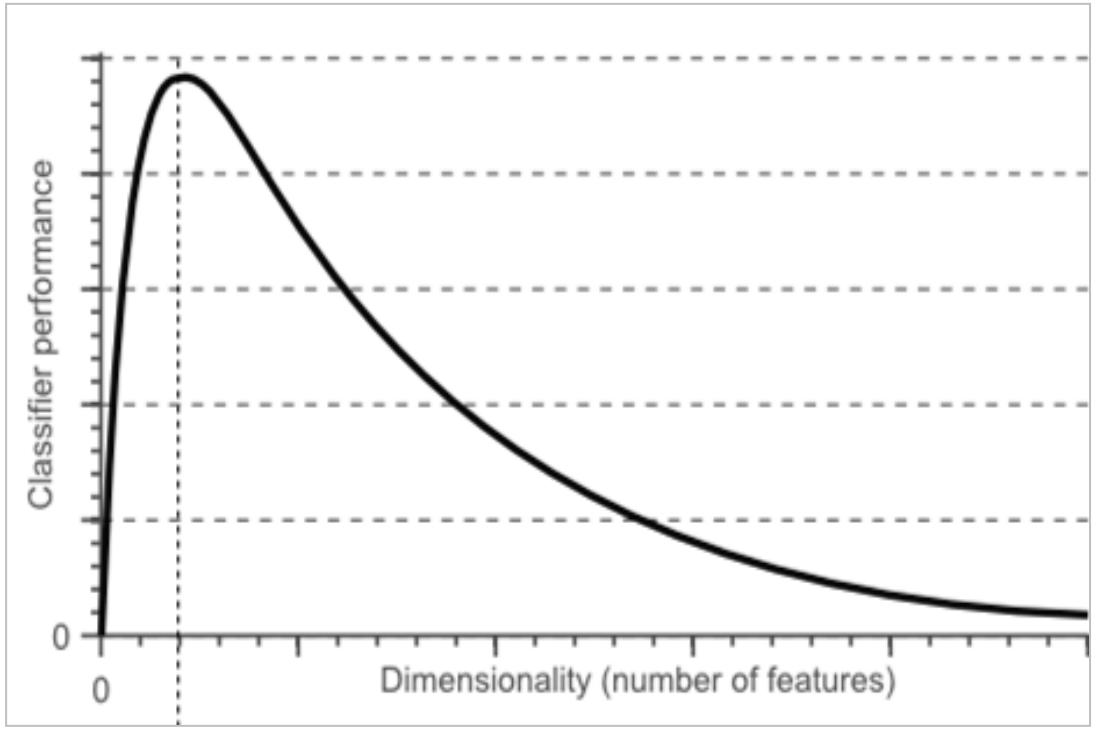
Huge number of dimensions causes problems

Data becomes very sparse, some algorithms become meaningless (e.g. density based clustering)

The complexity of several algorithms depends on the dimensionality and they become infeasible.

Curse of Dimensionality

Machine learning algorithms tend to over-fit data when the sample has a lot of predictor variables. There is a number of features above which the performance of a ML will degrade rather than improve.



Dimensionality Reduction

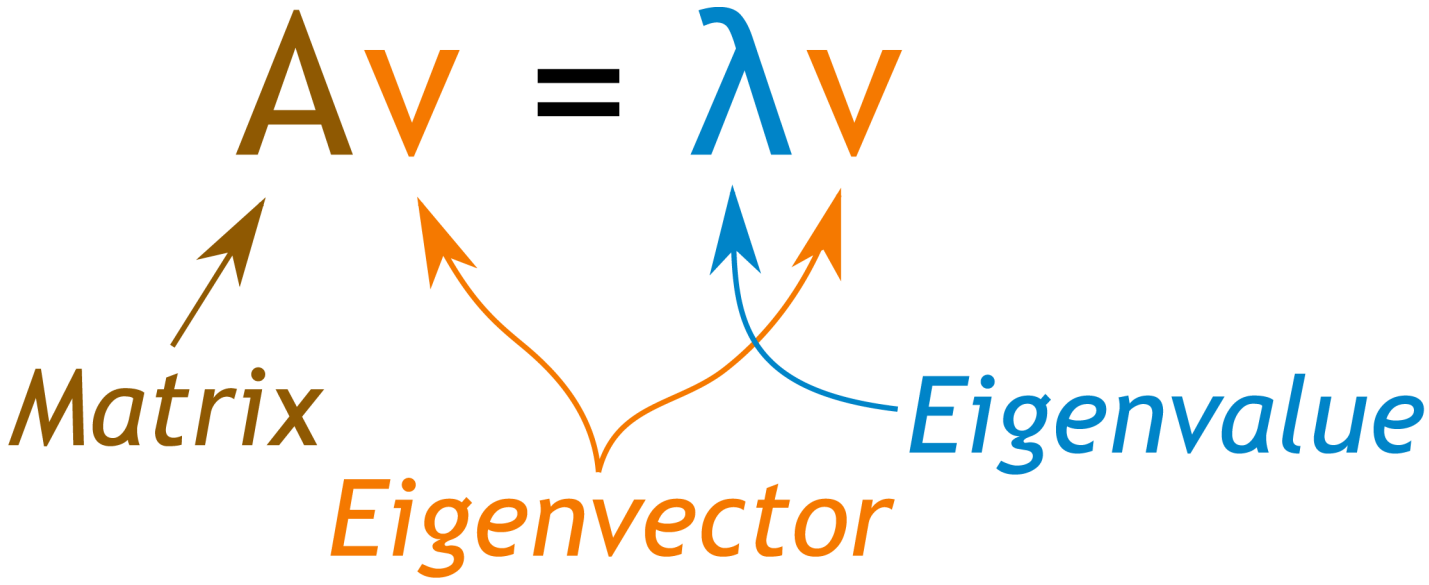
Visualization

Discovery of structure

Data Compression

Noise/Artifact detection

Eigenvalues and Eigenvectors



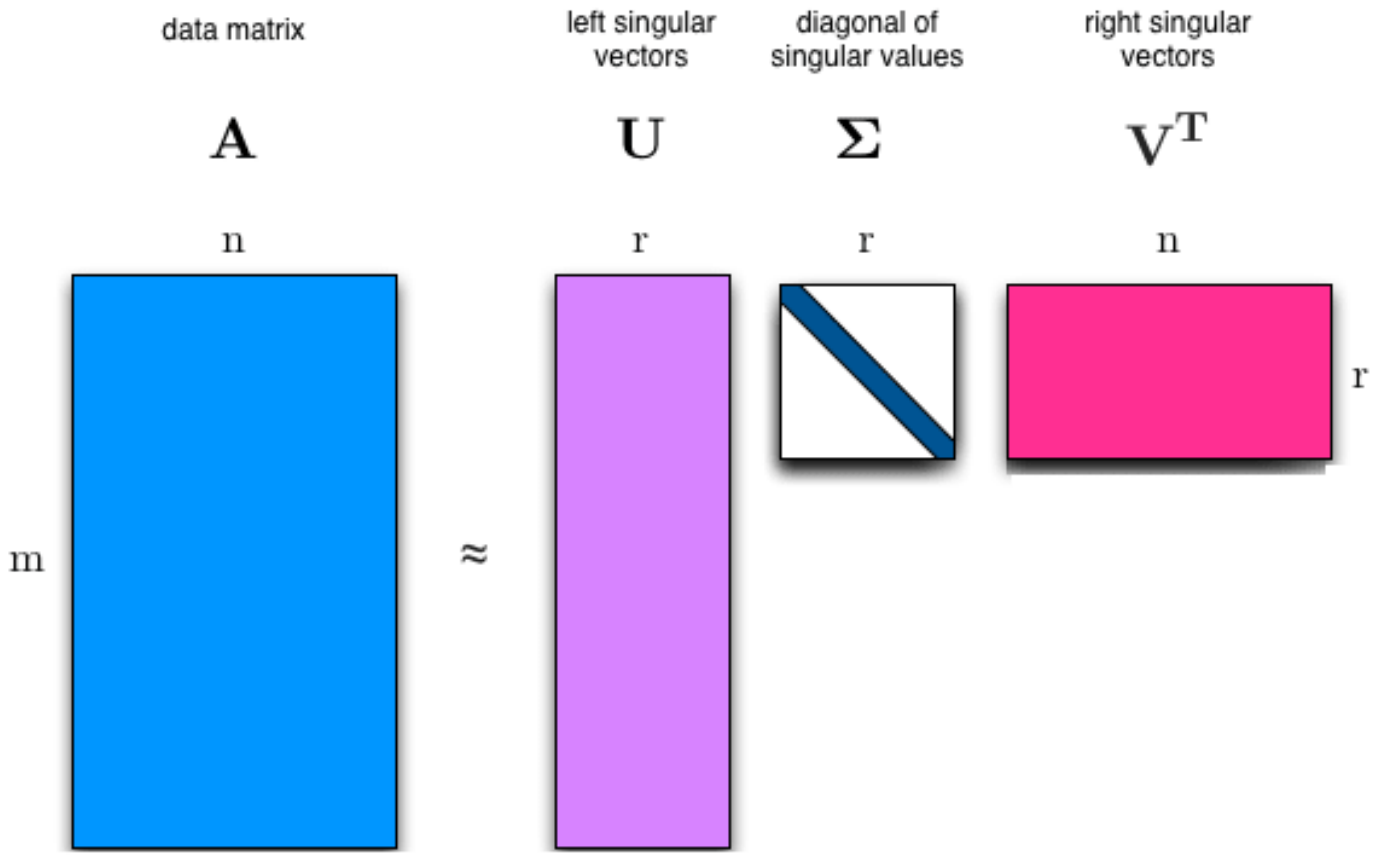
Eigenvalue Decomposition of a matrix.

- We can think of a matrix **A** as a transformation that acts on a vector **x** by multiplication to produce a new vector **Ax**

- If A is an m×p matrix and B is a p×n matrix, the matrix product **C=AB** (which is an m×n matrix) is defined as:

$$[\mathbf{C}]_{ij} = c_{ij} = \sum_{k=1}^{p} a_{ik} b_{kj}$$

Singular Value Decomposition



The singular value decomposition (SVD) is a factorization of a real or complex matrix that generalizes the eigendecomposition of a square normal matrix to any m×n matrix.

PCA and SVD

- PCA is SVD done on **centered** data
- PCA looks for such a direction that the data projected to it has the maximal variance
- PCA/SVD continues by seeking the next direction that is orthogonal to all previously found directions
- All directions are orthogonal

How to compute the PCA

- Data matrix **A**, ***rows = data points***, ***columns = variables*** (attributes, features, parameters)

1. Center the data by subtracting the mean of each column

2. Compute the SVD of the centered matrix **A'**
   (i.e., find the first **k** singular values/vectors) **$A' = U\Sigma V^T$**

3. The principal components are the columns of **V**, the coordinates of the data in the basis defined by the principal components are **$U\Sigma$**

Definition

PCA is defined as an **orthogonal** linear transformation that transforms the data to a new coordinate system such that the greatest variance by some scalar projection of the data comes to lie on the first coordinate, the second greatest variance on the second coordinate, and so on

Singular values tell us something about the variance

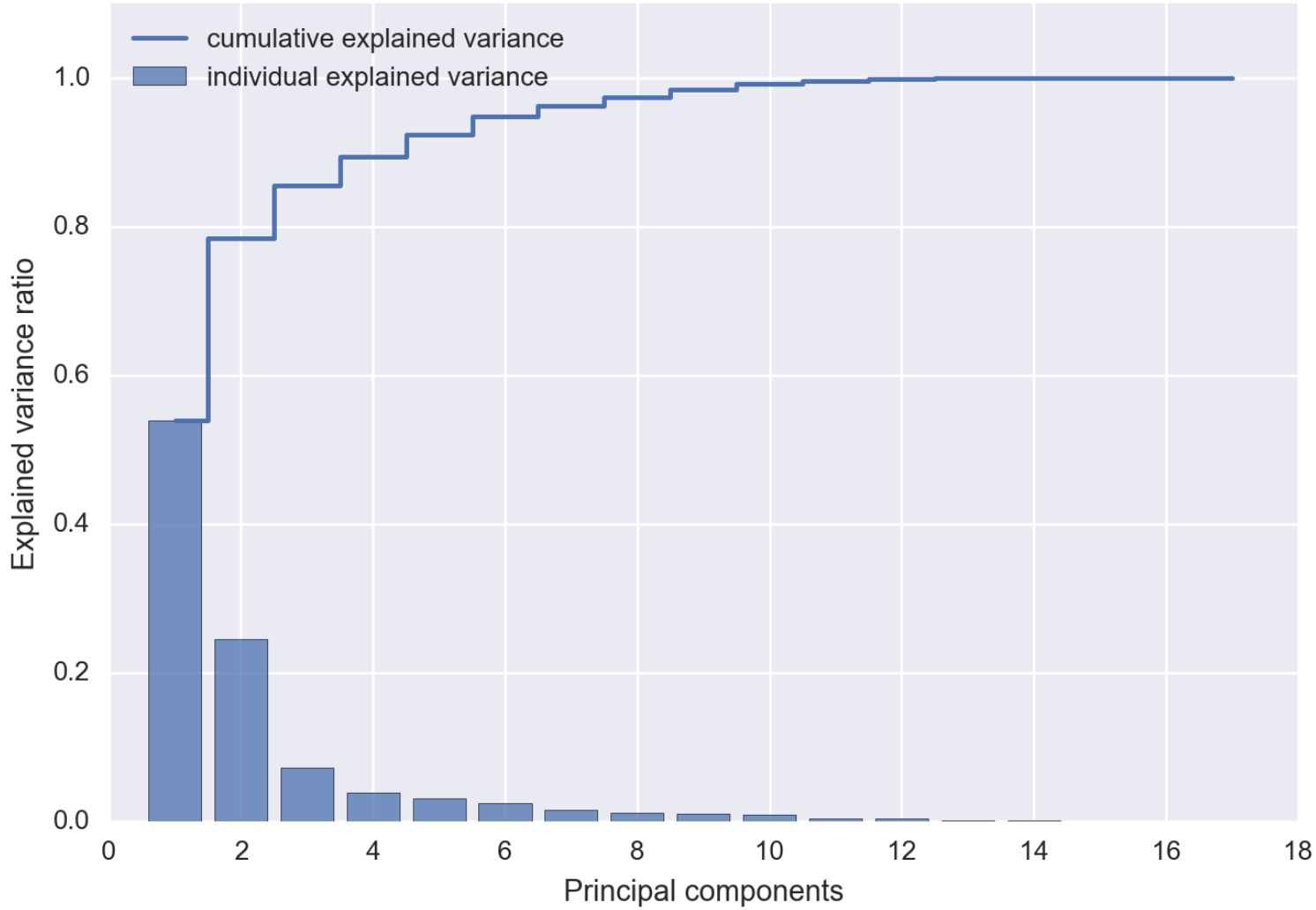
- The variance in the direction of the **k**-th principal component is given by the corresponding singular value σ_k^2

- Singular values can be used to estimate how many components to keep

- ***Rule of thumb:*** keep enough to explain ***85-90%*** of the variation:

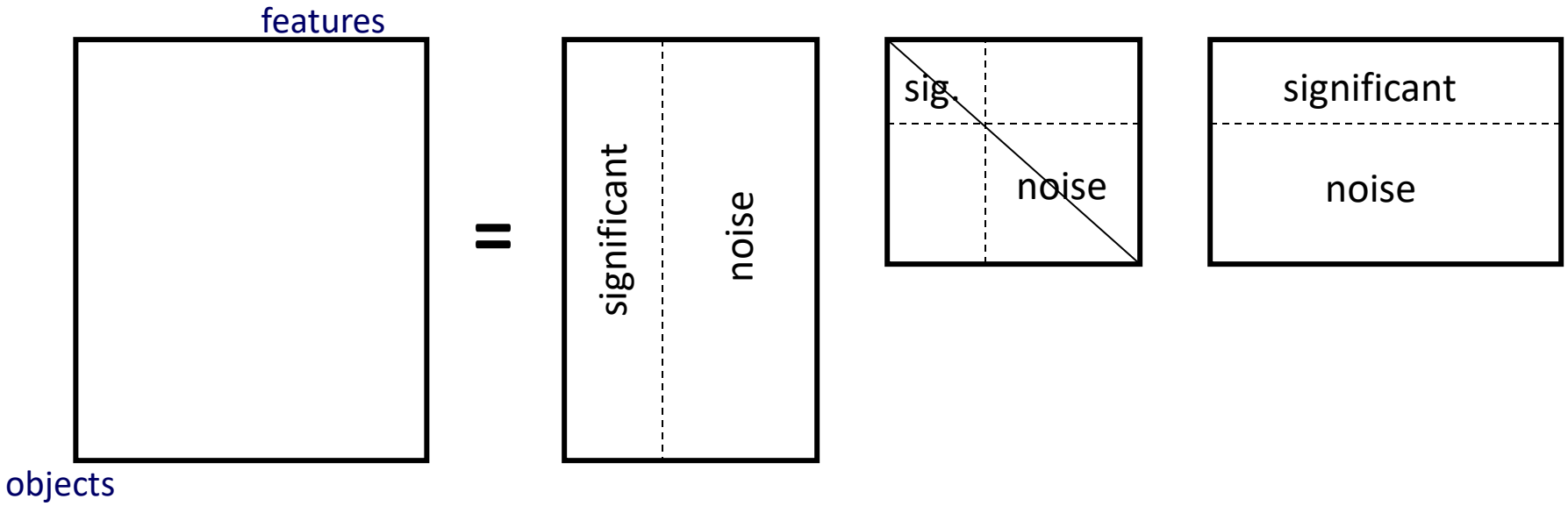
$$\frac{\sum_{j=1}^{k} \sigma_j^2}{\sum_{j=1}^{n} \sigma_j^2} \approx 0.85$$

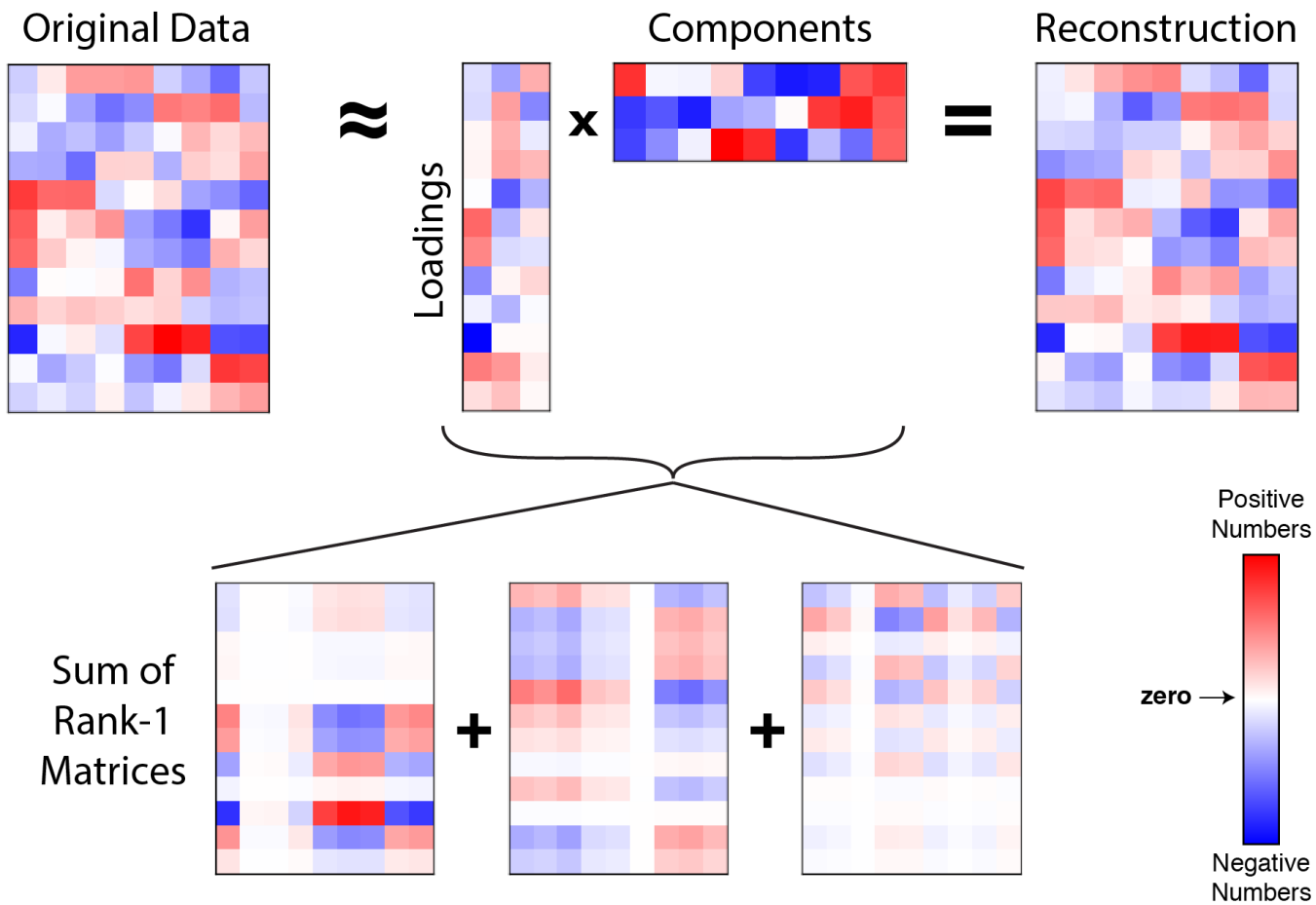
Explained Variance



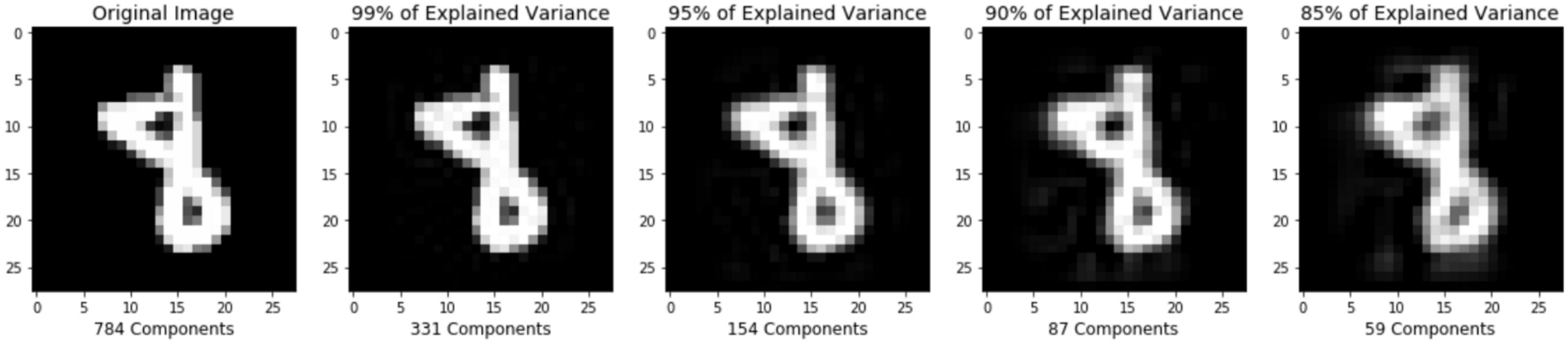
SVD/PCA and Rank-k approximations

$$\mathbf{A} = \mathbf{U} \Sigma \mathbf{V}^T$$




Original Data
≈
Loadings
x
Components
=
Reconstruction
Sum of
Rank-1
Matrices
+
+
Positive
Numbers
zero
Negative
Numbers

Data Compression

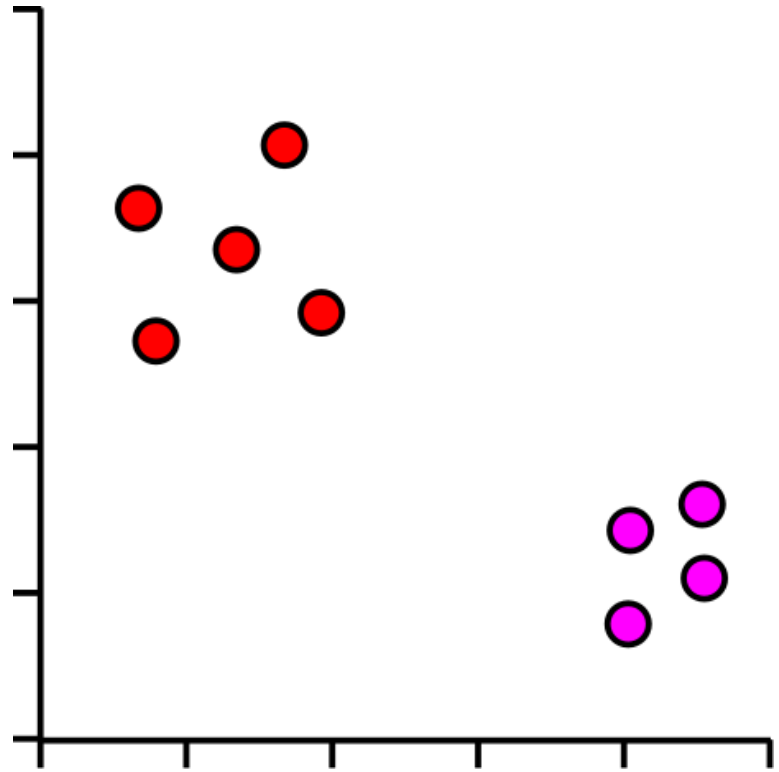


PCA in Scikit Learn

```
from sklearn.decomposition import PCA

pca = PCA(n_components=2)
#ProTip: specify float value for % explained variance
#pca = PCA(0.95)

principalComponents = pca.fit_transform(x)



pca.explained_variance_ratio_
```

Example of the hypothetical dataest

Gene	Description	Cell 1	Cell 2	Cell 3	Cell 4	Cell 5
Inpp5d	inositol polyphosphate-5-phosphatase D	7.00	5.45	5.89	6.03	5.75
Aim2	absent in melanoma 2	3.01	4.37	4.59	4.38	4.18
Gldn	gliomedin	3.48	3.63	4.61	4.70	4.74
Frem2	Fras1 related extracellular matrix protein 2	4.75	4.66	3.46	3.74	3.45
Rps3a1	ribosomal protein S3A1	6.10	7.23	7.44	7.36	7.34
Slc38a3	solute carrier family 38, member 3	1.90	3.16	3.52	3.61	3.19
Mt1	metallothionein 1	5.07	6.49	6.46	6.04	6.05
C1s1	complement component 1, s subcomponent 1	2.74	3.02	3.86	4.10	4.10
Cds1	CDP-diacylglycerol synthase 1	4.55	4.22	3.80	3.16	3.12
Ifi44	interferon-induced protein 44	4.82	4.52	3.87	3.42	3.59
Lefty2	left-right determination factor 2	6.95	6.28	5.88	5.60	5.61
Fmr1nb	fragile X mental retardation 1 neighbor	4.28	2.78	3.10	3.25	2.57
Tagln	transgelin	7.93	7.91	7.20	7.02	6.68



Each dot is a cell

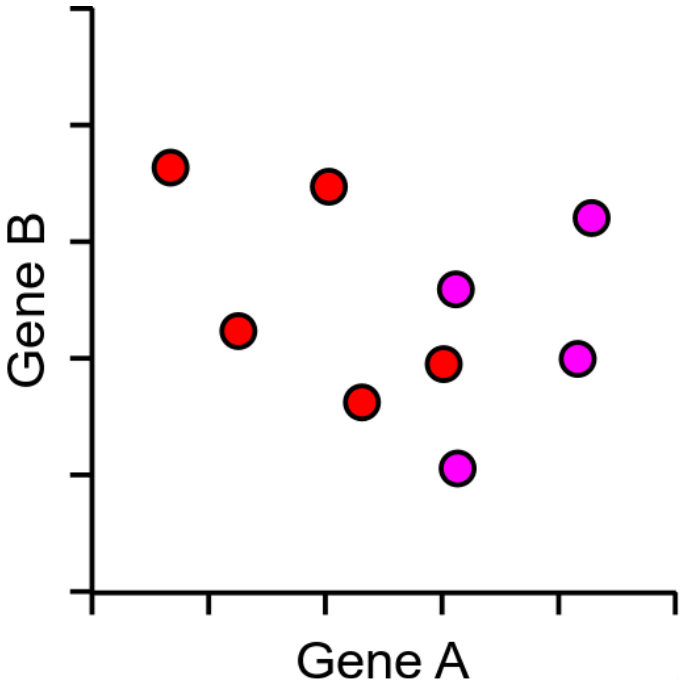
Groups of dots are similar cells

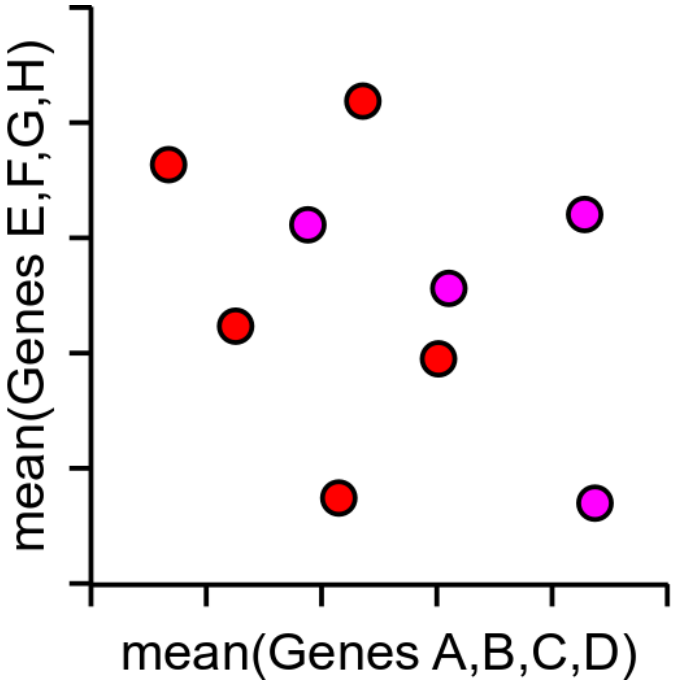
Separation of groups could be interesting biology

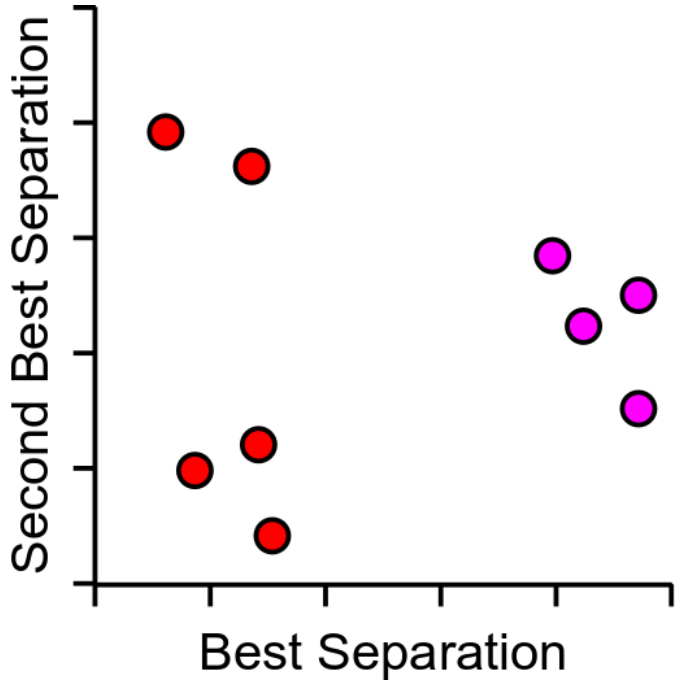
Example by: Simon Andrews, Babraham Bioinformatics

Too much data!

- 5000 cells and 2500 measured genes
- Realistically only 2 dimensions we can plot (x,y)

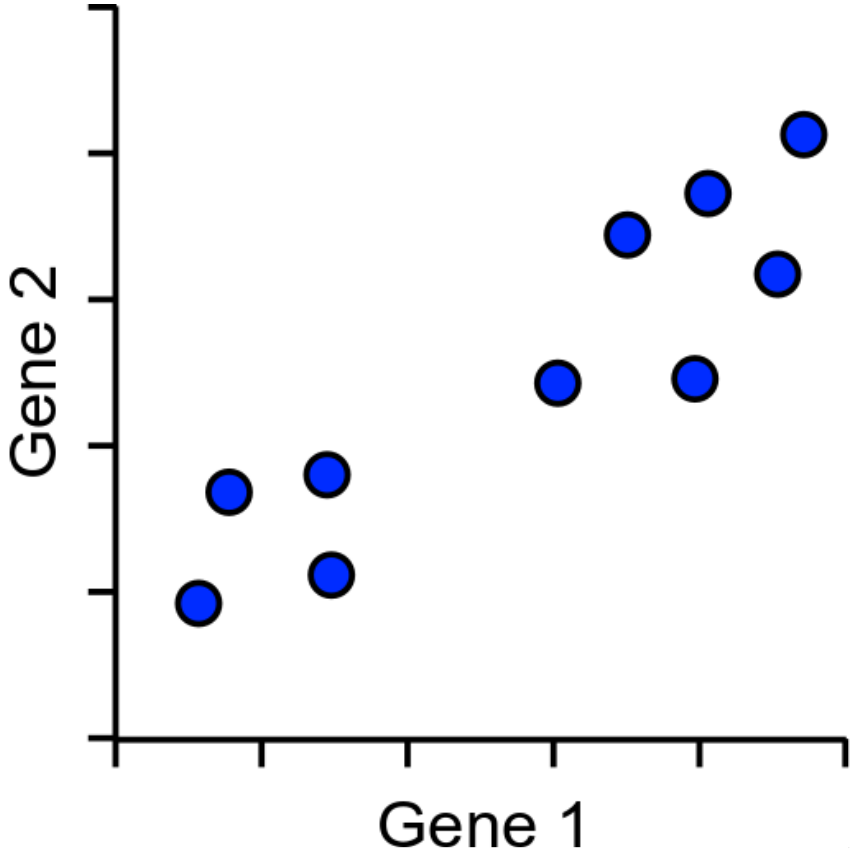


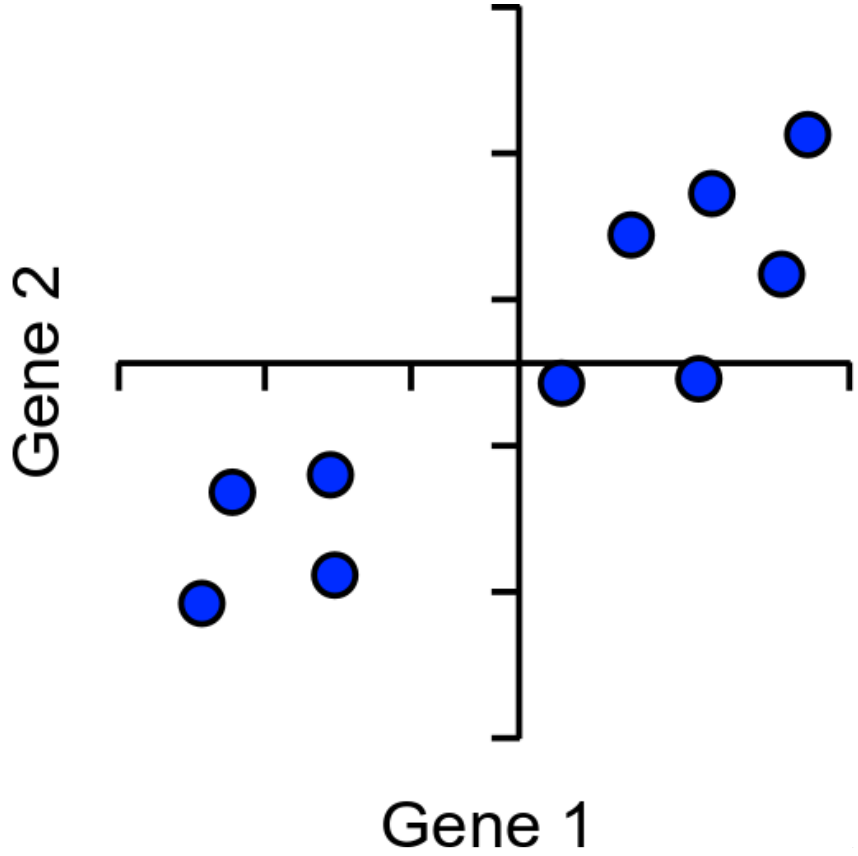




How does PCA work?

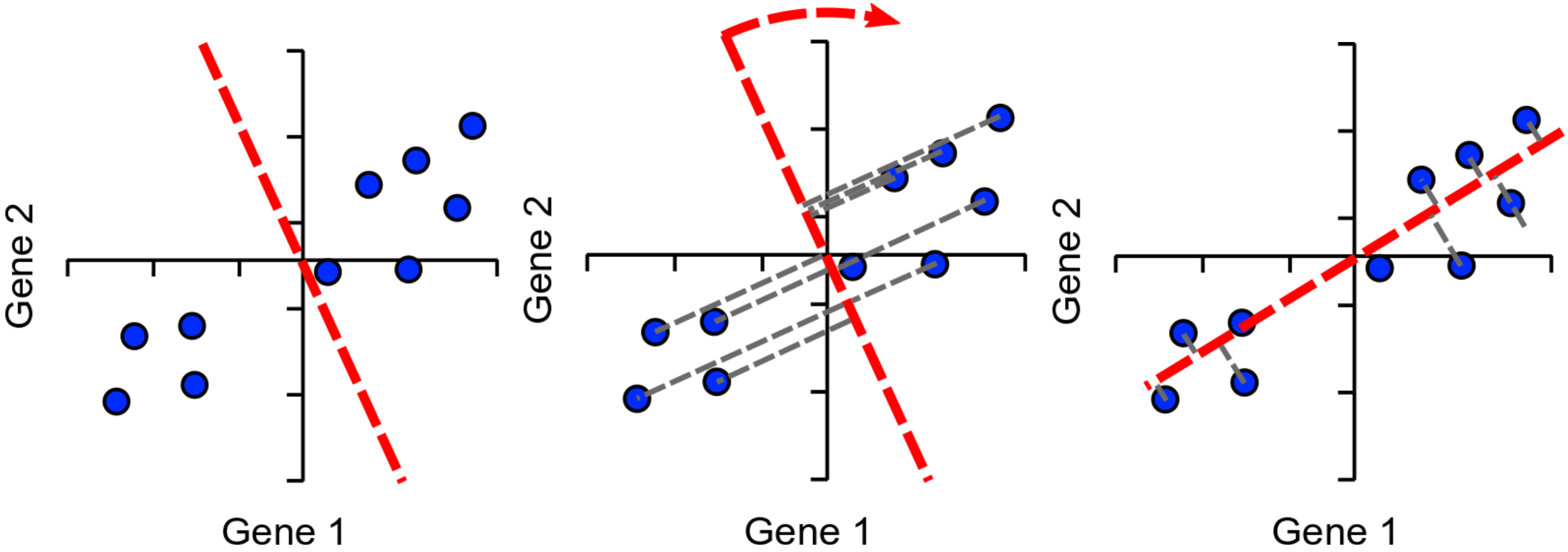
- Simple example using 2 genes and 10 cells



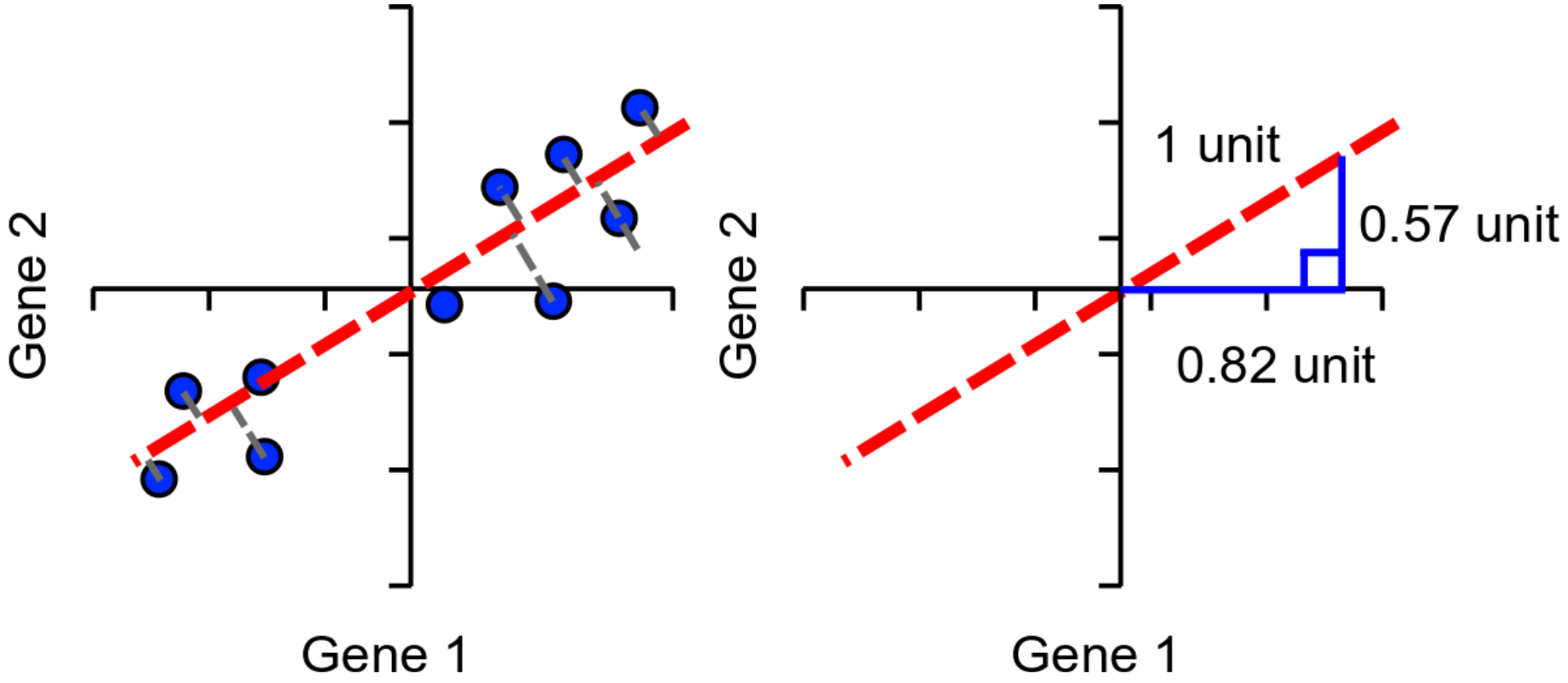


How does PCA work?

- Find line of best fit, passing through the origin



Assigning Loadings to Genes



Single Vector or '**eigenvector**'

Loadings:

- Gene1 = 0.82
- Gene2 = 0.57

Higher loading equals more influence on PC

More dimensions

- The same idea extends to larger numbers of dimensions (n)

- Calculation of first PC rotates in (n-1) -dimensions
  - Next PC is perpendicular to PC2, but rotated similarly (n-2)
  - Last PC is remaining perpendicular (no choice)
  - Same number of PCs as features in the dataset.

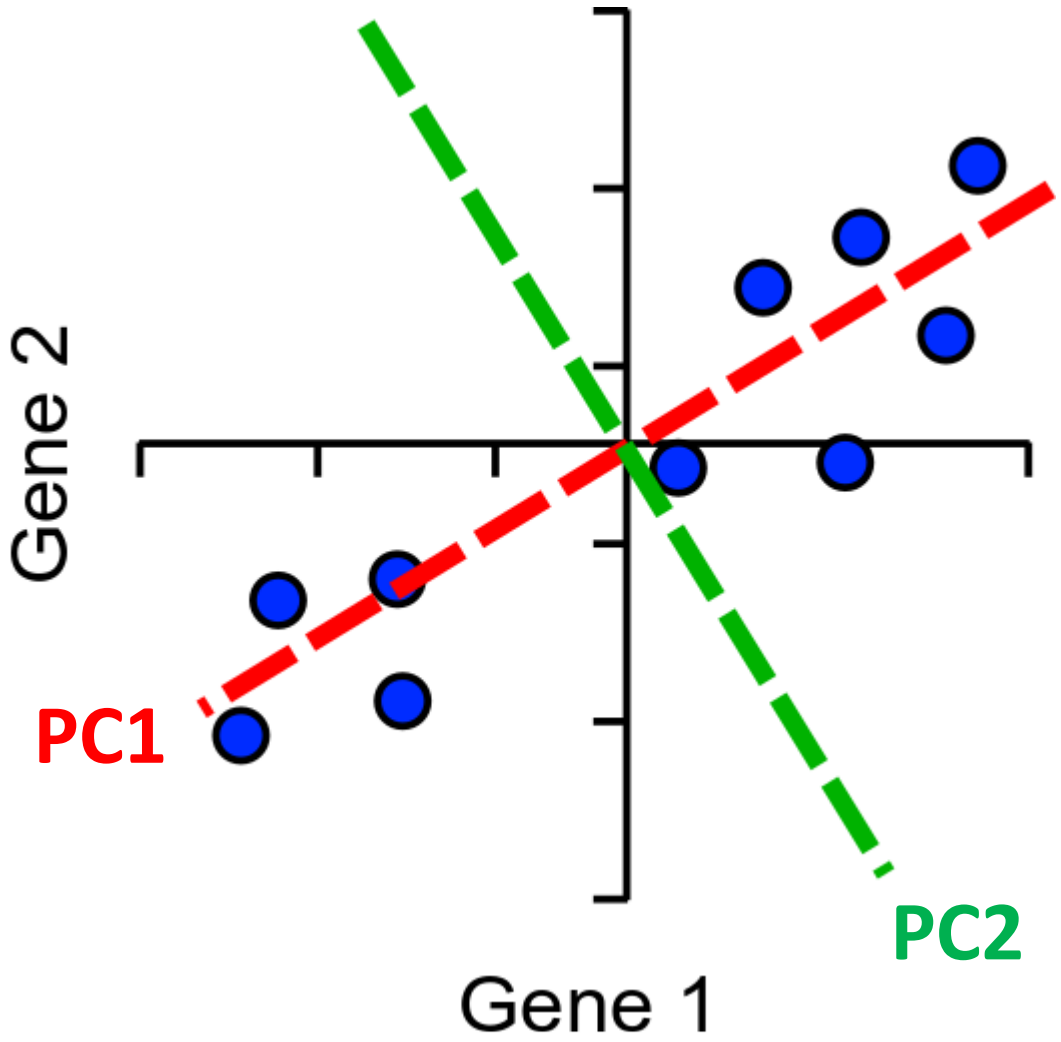
Explaining Variance

- Each PC always explains some proportion of the total variance in the data. Between them they explain everything
  - PC1 always explains the most
  - PC2 is the next highest etc. etc.

- Since we only plot 2 dimensions we'd like to know that these are a good explanation

- How do we calculate this?

Explaining variance



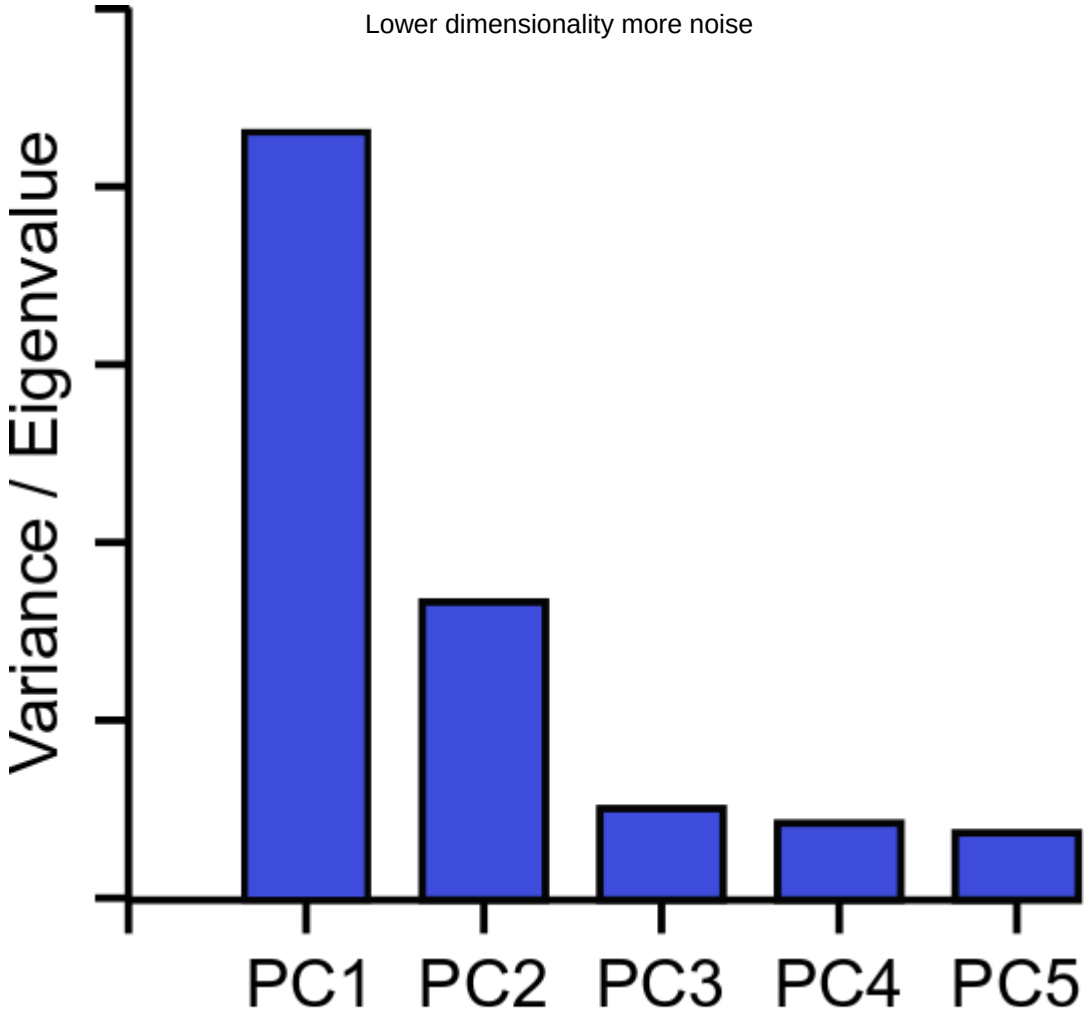
- Project onto PC
- Calculate distance to the origin

- Calculate sum of squared differences (SSD)
  - This is a measure of variance called the ‘eigenvalue’

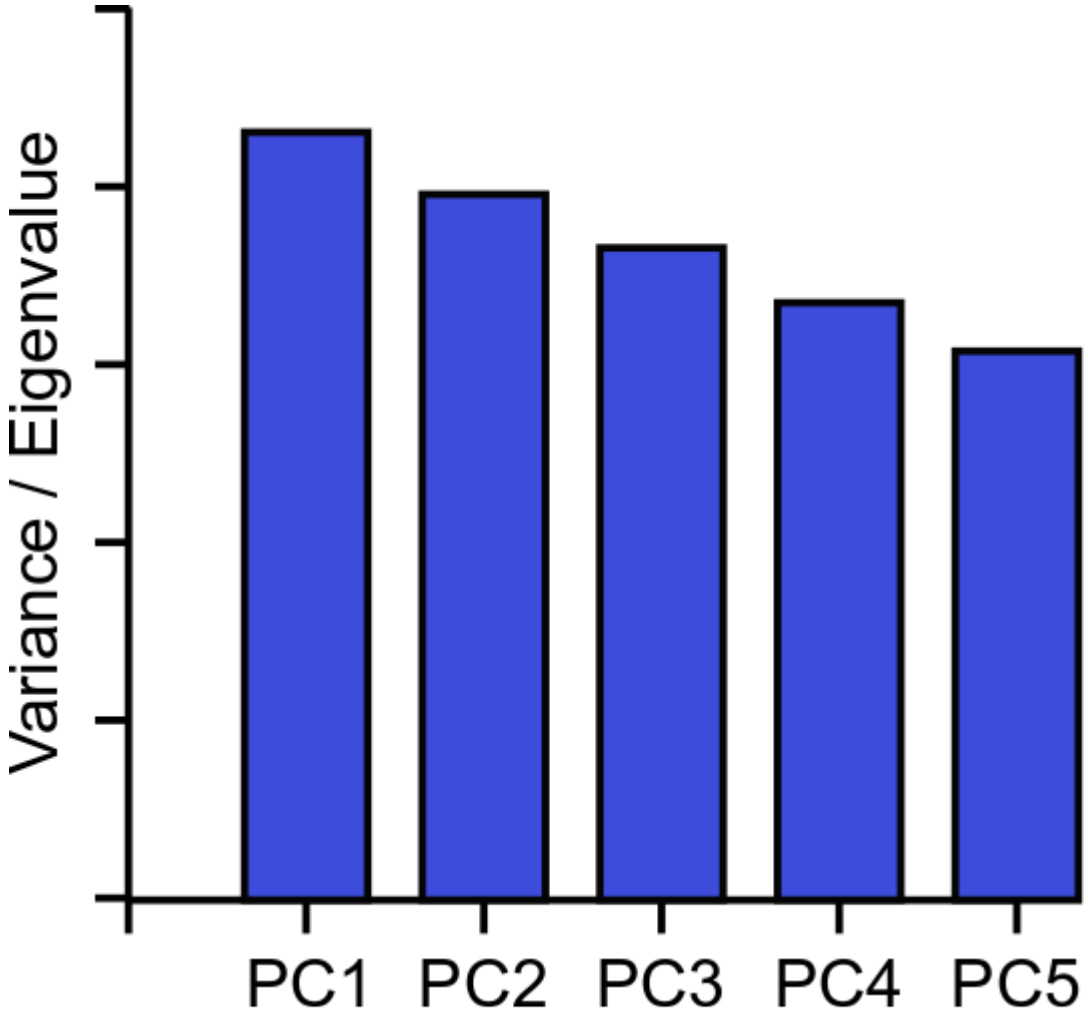
  - Divide by (points-1) to get actual variance

Explaining Variance – Scree Plots

Lower dimensionality more noise

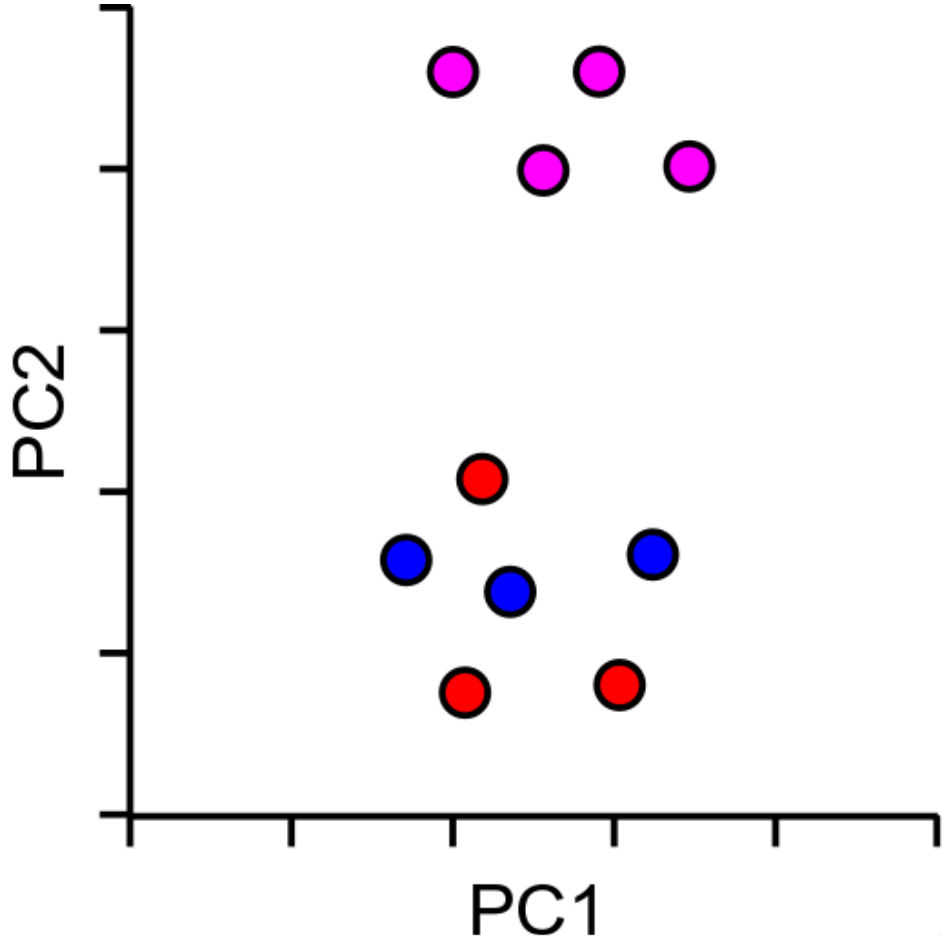


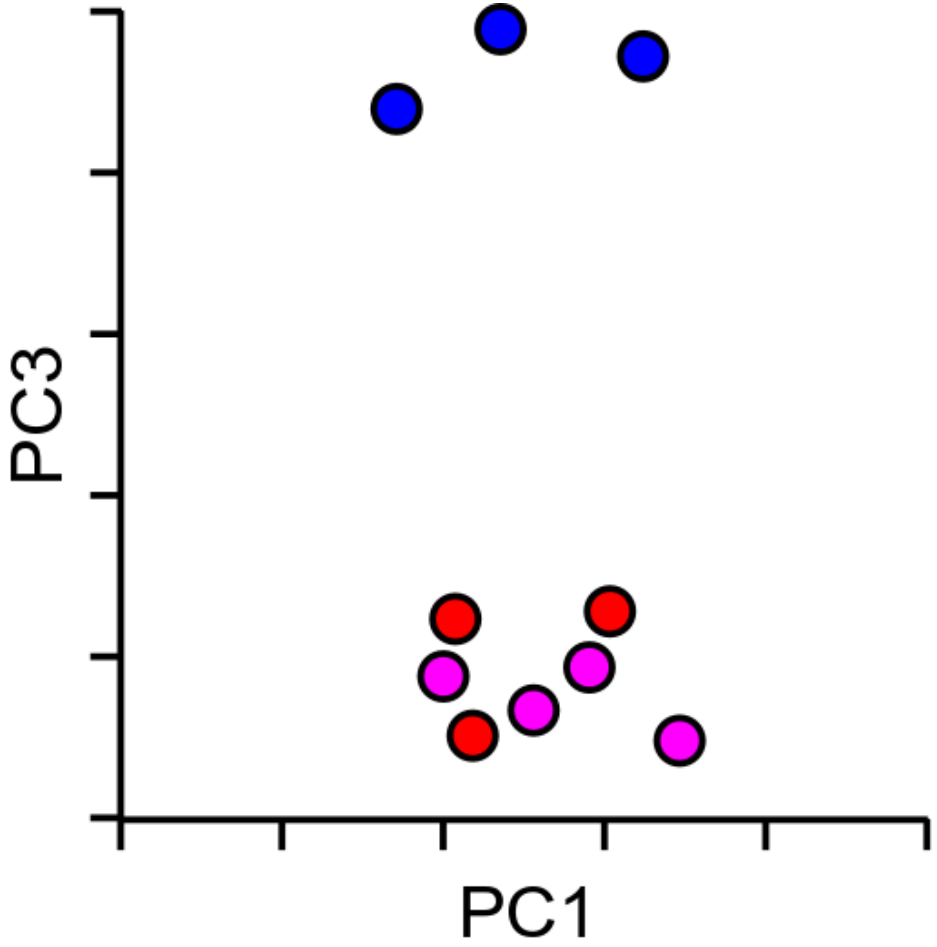
Higher dimensionality less noise

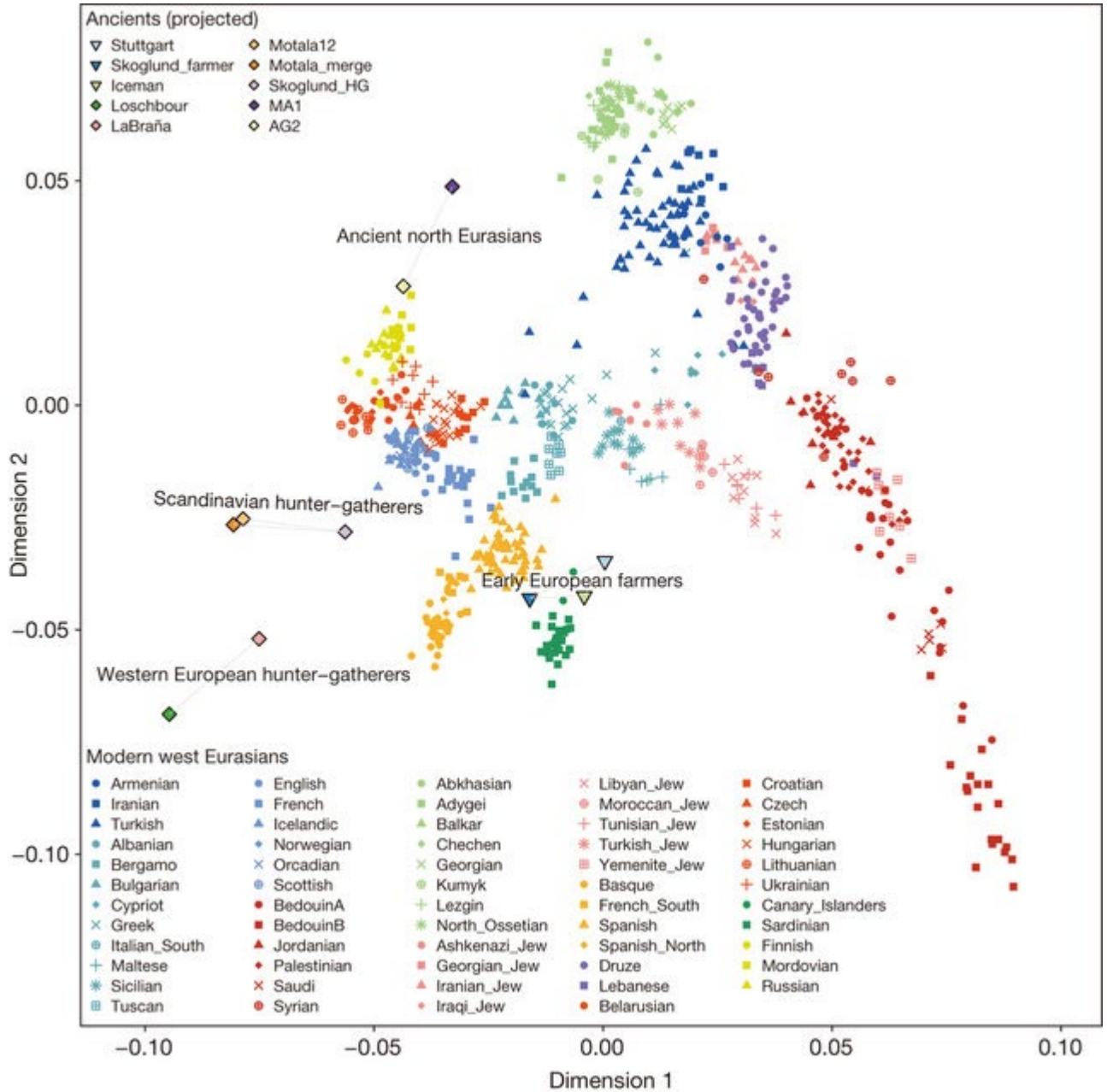


So PCA is great then?

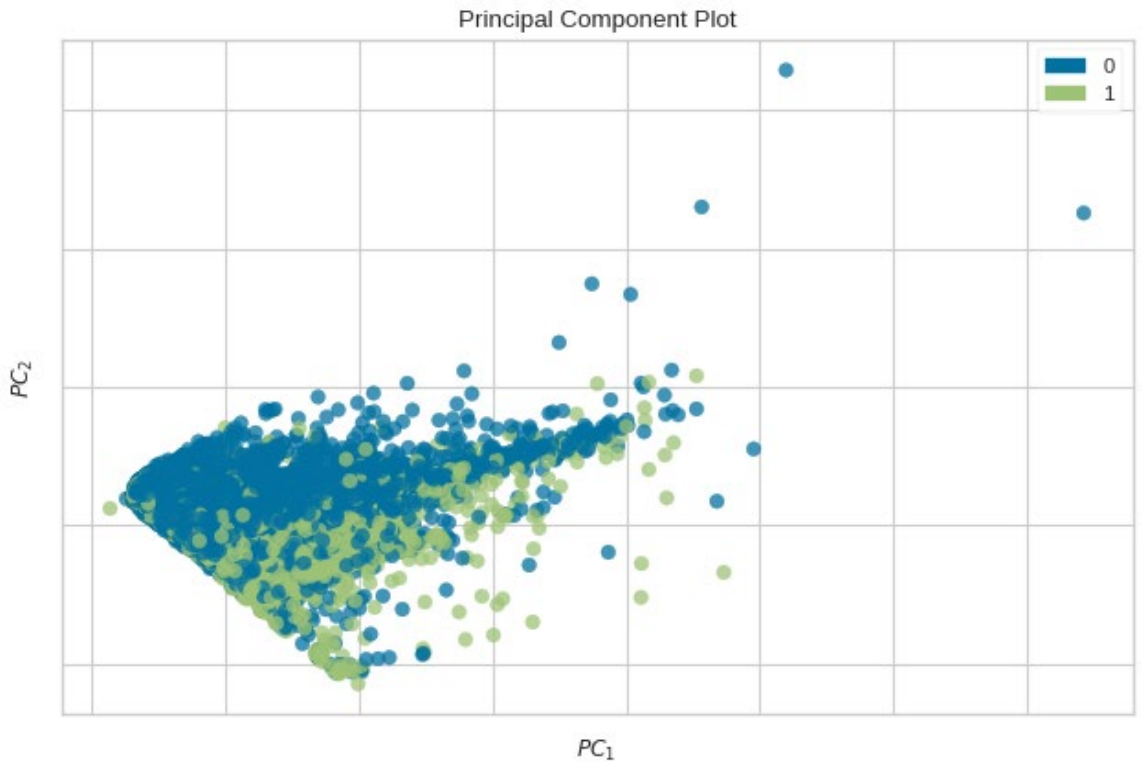
- Kind of…

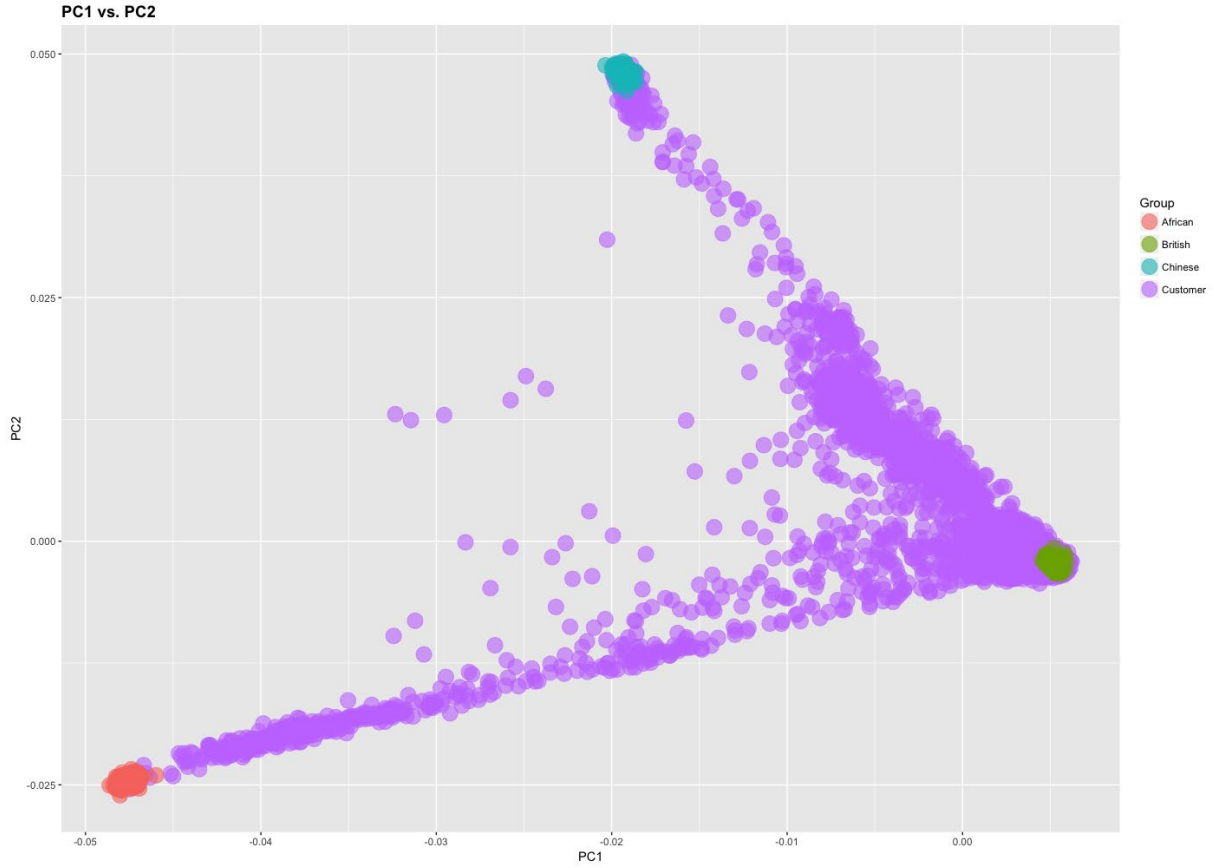





Ancients (projected)
Stuttgart
Skoglund_farmer
Iceman
Loschbour
LaBraña
Motala12
Motala_merge
Skoglund_HG
MA1
AG2
Ancient north Eurasians
Scandinavian hunter−gatherers
Early European farmers
Western European hunter−gatherers
Modern west Eurasians
Armenian
Iranian
Turkish
Albanian
Bergamo
Bulgarian
Cypriot
Greek
Italian_South
Maltese
Sicilian
Tuscan
English
French
Icelandic
Norwegian
Orcadian
Scottish
BedouinA
BedouinB
Jordanian
Palestinian
Saudi
Syrian
Abkhasian
Adygei
Balkar
Chechen
Georgian
Kumyk
Lezgin
North_Ossetian
Ashkenazi_Jew
Georgian_Jew
Iranian_Jew
Iraqi_Jew
Libyan_Jew
Moroccan_Jew
Tunisian_Jew
Turkish_Jew
Yemenite_Jew
Basque
French_South
Spanish
Spanish_North
Druze
Lebanese
Belarusian
Croatian
Czech
Estonian
Hungarian
Lithuanian
Ukrainian
Canary_Islanders
Sardinian
Finnish
Mordovian
Russian
0.05
0.00
−0.05
−0.10
Dimension 2
−0.10
−0.05
0.00
0.05
0.10
Dimension 1

More (realistic) Examples



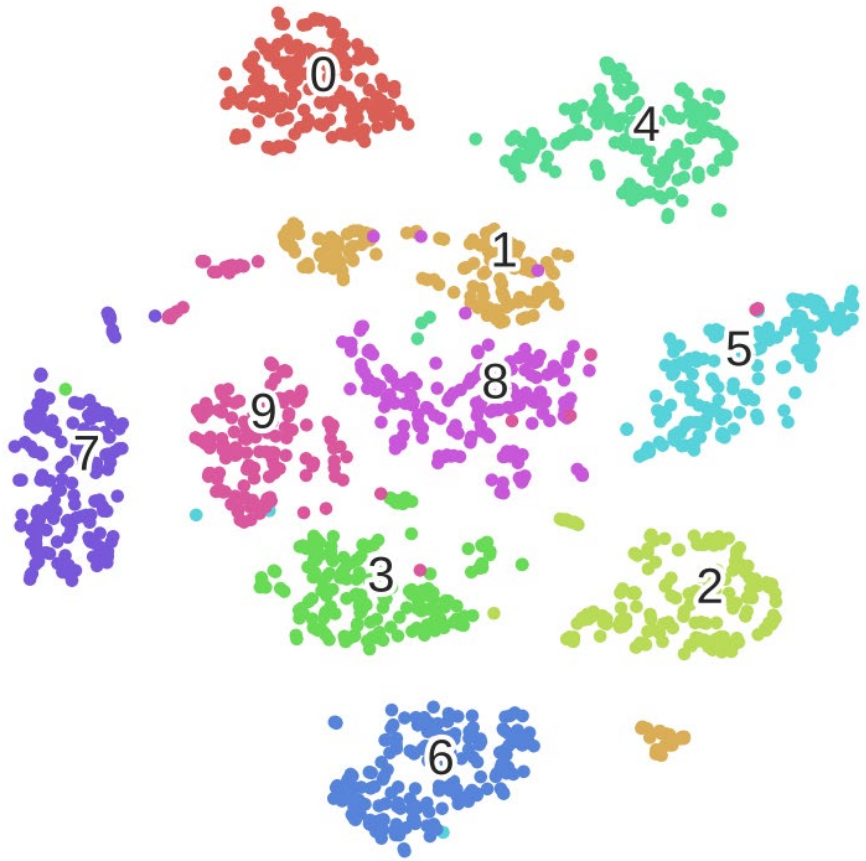


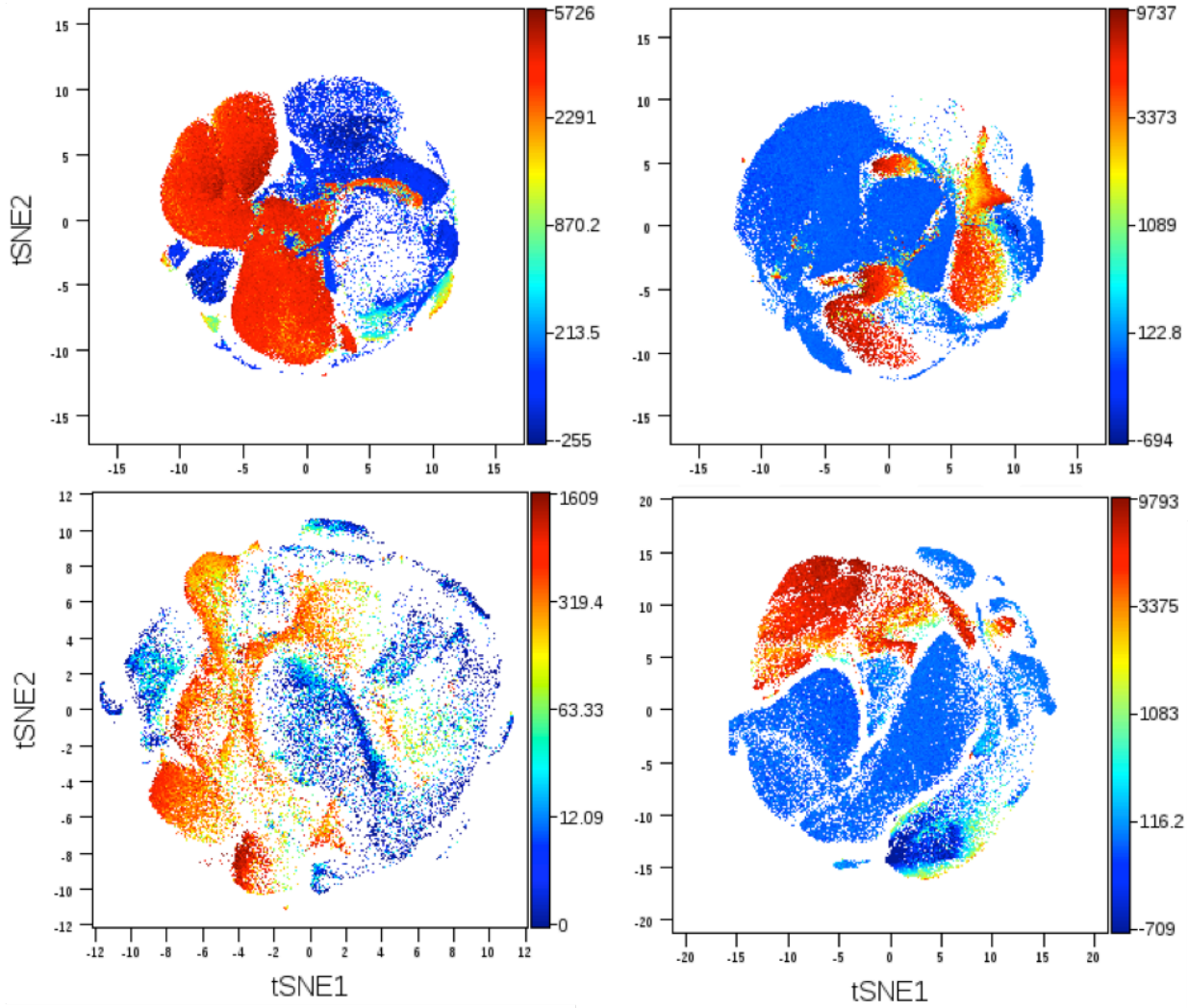
tSNE to the rescue…

- T-Distributed Stochastic Neighbour Embedding

- Aims to solve the problems of PCA
  - Non-linear scaling to represent changes at different levels

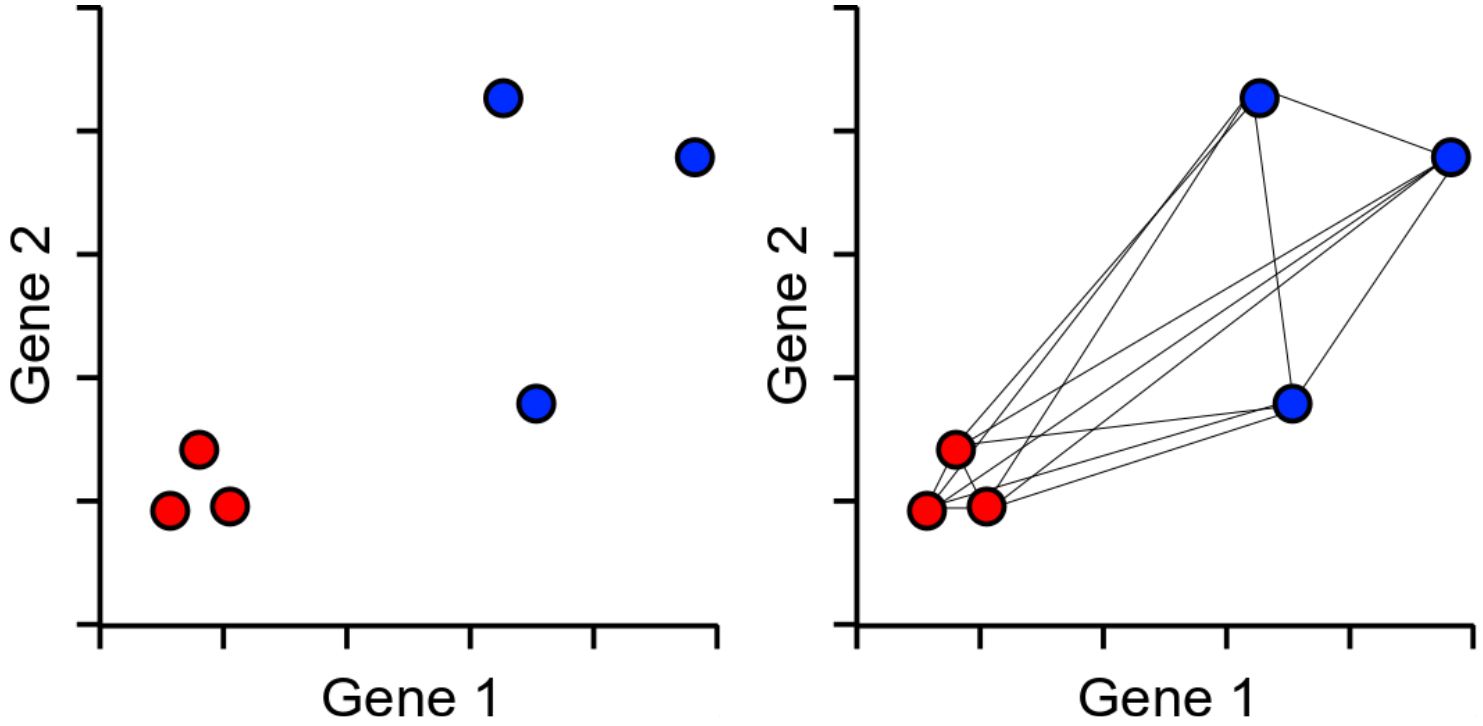
  - Optimal separation in 2-dimensions


0
4
1
5
8
9
7
3
2
6


tSNE2
tSNE1
5726
2291
870.2
213.5
-255
9737
3373
1089
122.8
-694
1609
319.4
63.33
12.09
0
9793
3375
1083
116.2
-709

How does tSNE work?

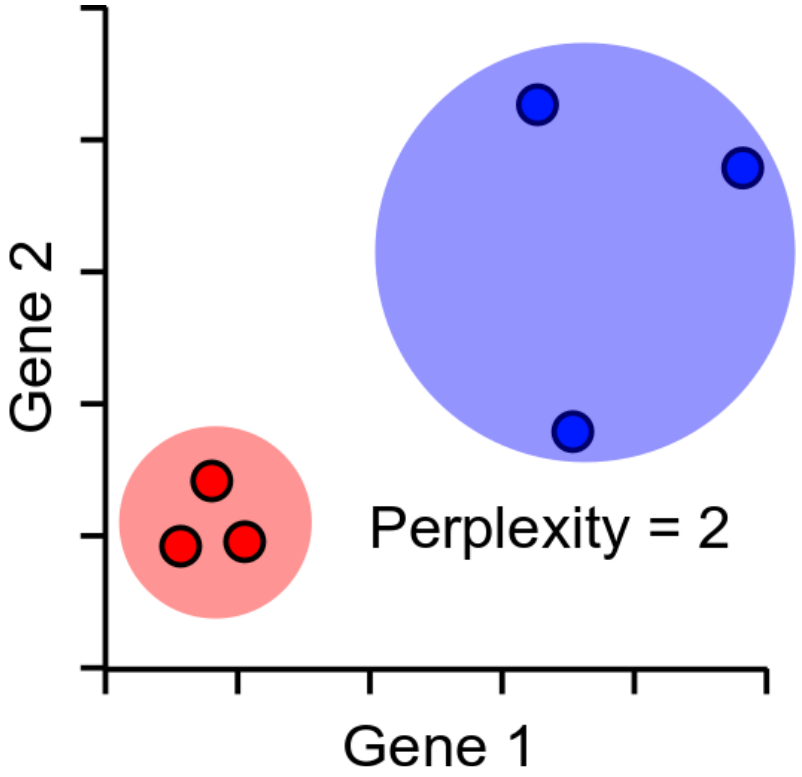
- Based around all-vs-all table of pairwise cell to cell distances



	0	10	10	295	158	153
	9	0	1	217	227	213
	1	8	0	154	225	238
	205	189	260	0	23	45
	248	227	246	44	0	54
	233	176	184	41	36	0

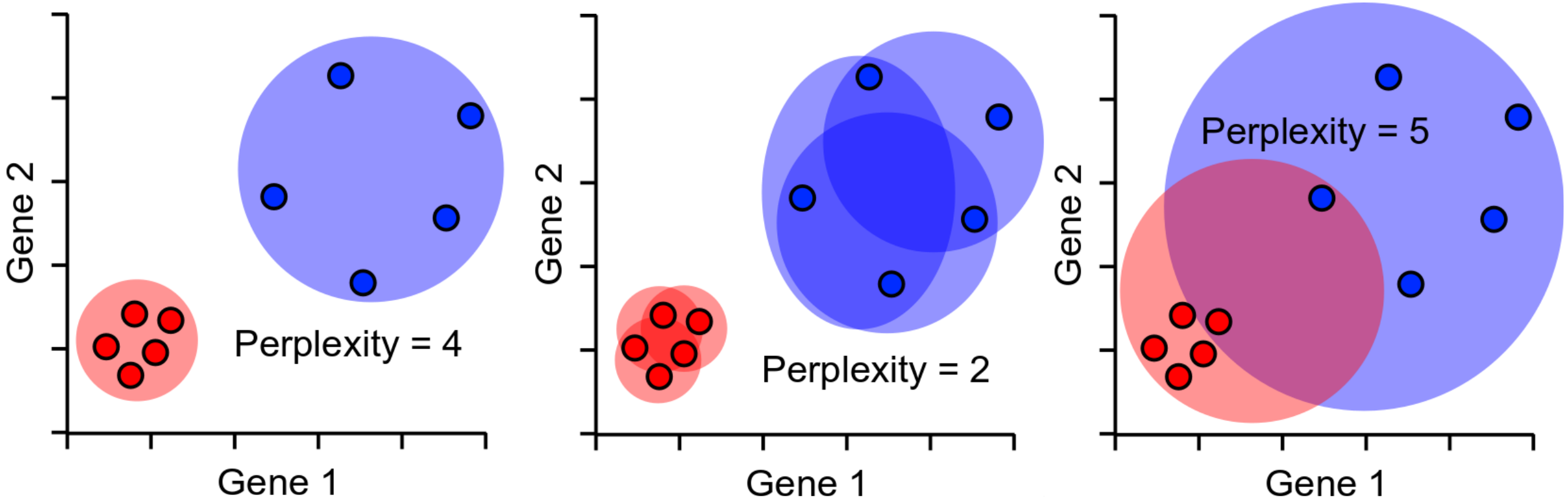
Distance scaling and perplexity

- Perplexity = expected number of neighbours within a cluster
- Distances scaled relative to perplexity neighbours



	0	4	6	586	657	836
	4	0	4	815	527	776
	9	3	0	752	656	732
	31	28	29	0	4	7
	31	24	25	4	0	7
	40	37	32	8	8	0

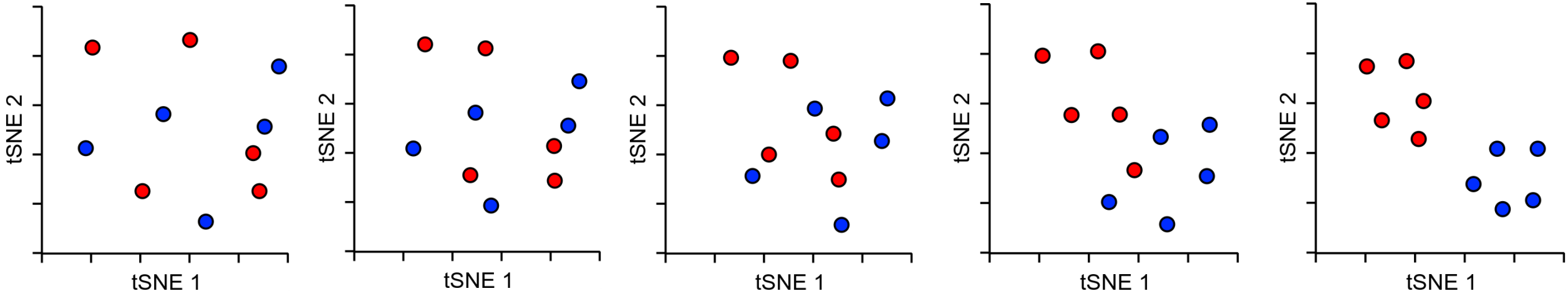
Perplexity Robustness



tSNE Projection

- Normally 2D, but can be any number of dimensions
- Randomly scatter all points within the space
- Start a simulation
  - Aim is to make the point distances match the distance matrix
  - Shuffle points based on how well they match
  - Stop after fixed number of iterations, or
  - Stop after distances have converged

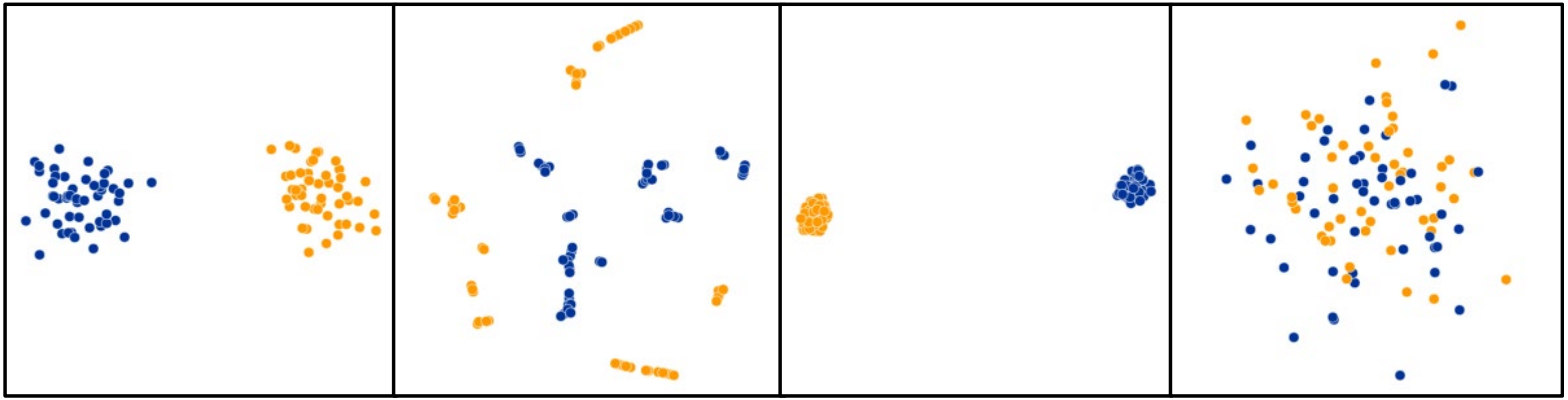
tSNE Projection



- X and Y don’t mean anything (unlike PCA)
- Distance doesn’t mean anything (unlike PCA)
- Close proximity is highly informative
- Distant proximity isn’t very interesting
- Can’t rationalise distances, or add in more data

tSNE Practical Examples

Perplexity Settings Matter

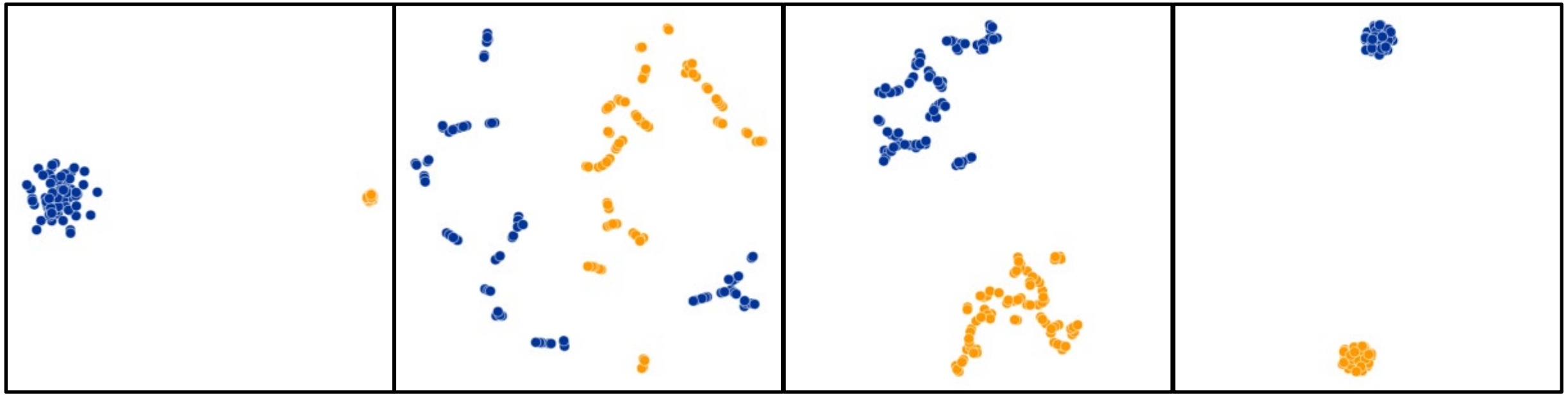


Original | Perplexity = 2 | Perplexity = 30 | Perplexity = 100

https://distill.pub/2016/misread-tsne/

tSNE Practical Examples

Cluster Sizes are Meaningless

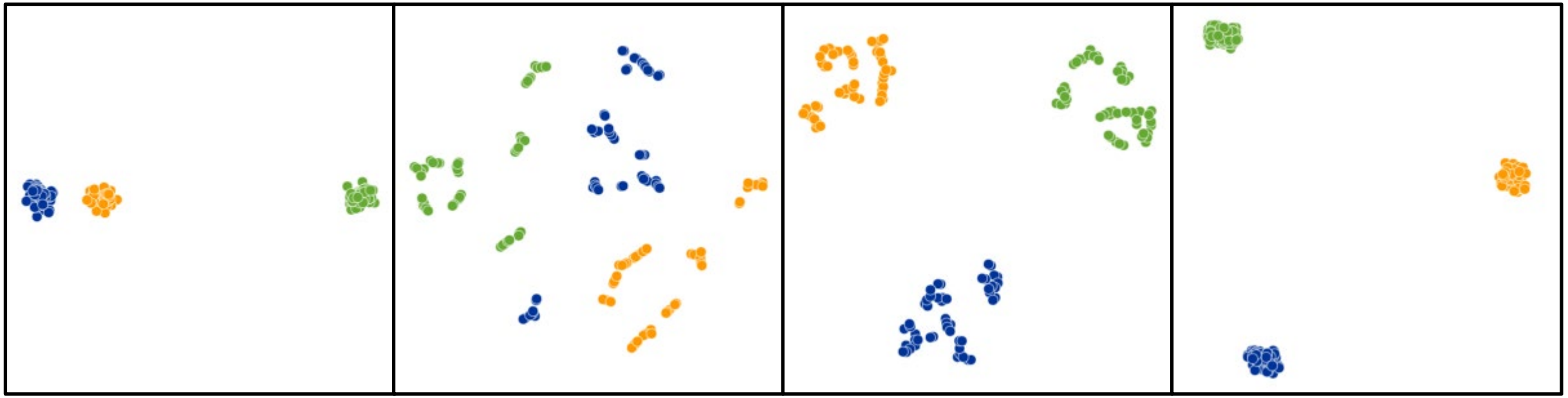


Original | Perplexity = 2 | Perplexity = 5 | Perplexity = 50

https://distill.pub/2016/misread-tsne/

tSNE Practical Examples

Distances between clusters can’t be trusted

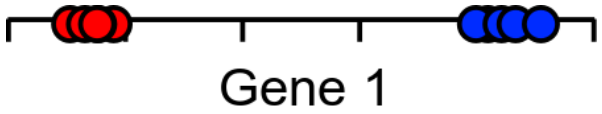


Original | Perplexity = 2 | Perplexity = 5 | Perplexity = 30

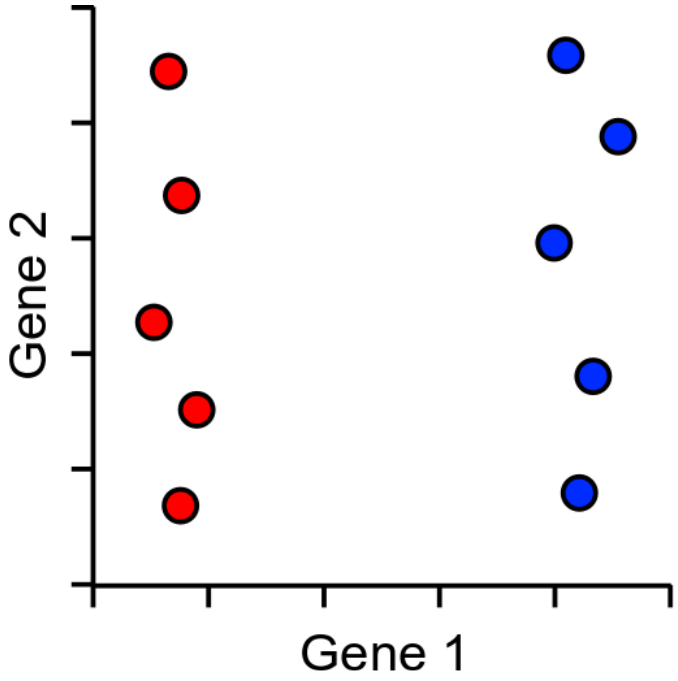
https://distill.pub/2016/misread-tsne/

So tSNE is great then?

- Kind of…
- Imagine a dataset with only one super informative gene



Distance within cluster = low
Distance between clusters = high



Distance within cluster = higher
Distance between clusters = lower

- Now 3 genes
- Now 3,000 genes

- Everything is the same distance from everything

So everything sucks?

- PCA
  - Requires more than 2 dimensions
  - Thrown off by quantised data
  - Expects linear relationships
- tSNE
  - Can’t cope with noisy data
  - Loses the ability to cluster

Answer: Combine the two methods, get the best of both worlds

- PCA
  - Good at extracting signal from noise
  - Extracts informative dimensions
- tSNE
  - Can reduce to 2D well
  - Can cope with non-linear scaling

Practical PCA + tSNE

- Filter data heavily before starting

- Do PCA
  - Extract most interesting signal
  - Take top PCs. Reduce dimensionality (but not to 2)

- Do tSNE
  - Calculate distances from PCA projections
  - Scale distances and project into 2-dimensions

tSNE is Dead. Long Live UMAP!

UMAP: **Uniform Manifold Approximation and Projection**

- UMAP uses the **number of nearest neighbors** instead of perplexity.
- Deal differently with high-dimensional probabilities
- Better cost function
- Scale to BIG DATA!
- https://pair-code.github.io/understanding-umap/

- https://arxiv.org/abs/1802.03426
- UMAP: Uniform Manifold Approximation and Projection for Dimension Reduction

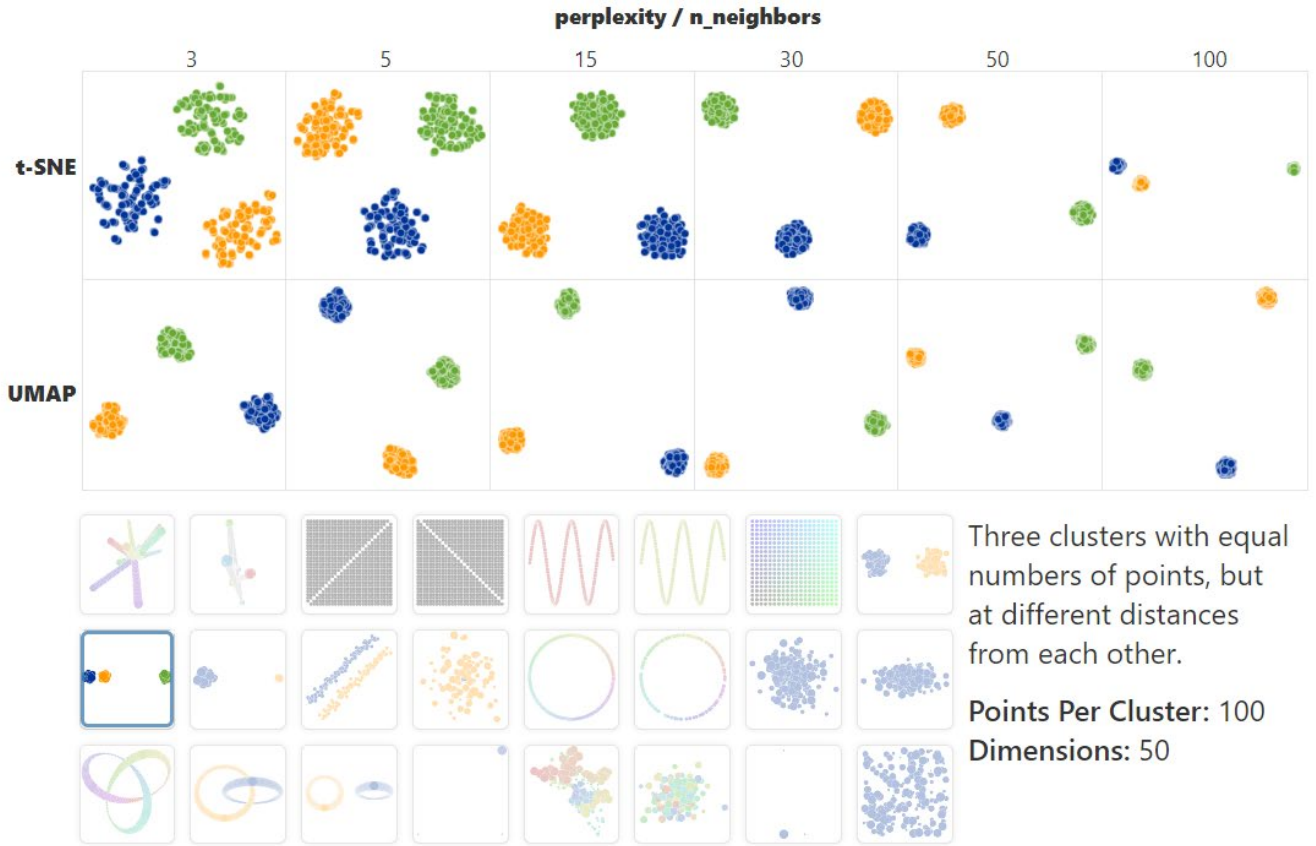
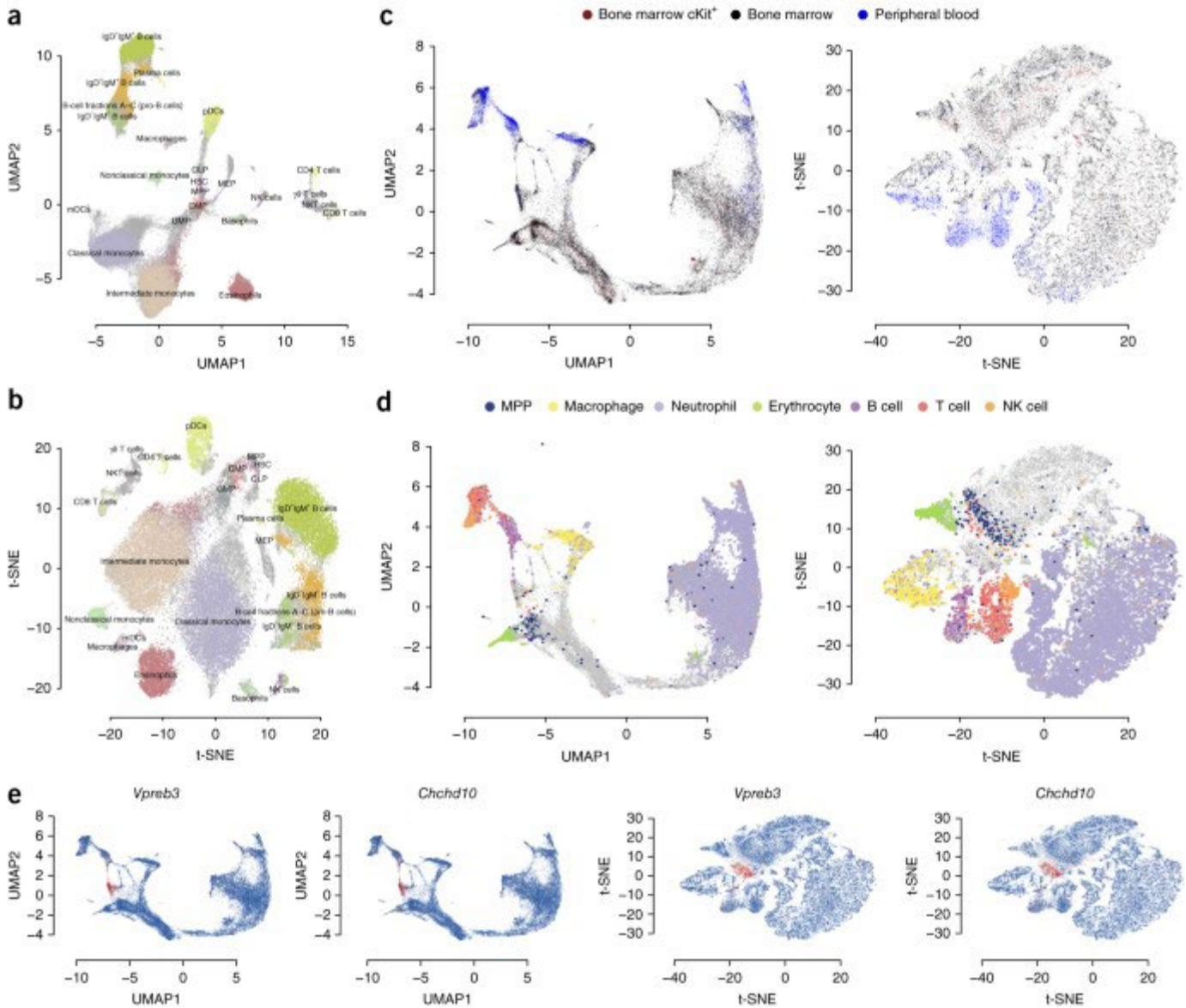


Figure 7: *Comparison between UMAP and t-SNE projecting various toy datasets.*

https://pair-code.github.io/understanding-umap/



Becht, E., McInnes, L., Healy, J. *et al.* Dimensionality reduction for visualizing single-cell data using UMAP. *Nat Biotechnol* **37,** 38–44 (2019). https://doi.org/10.1038/nbt.4314

Dimensionality Reduction Taxonomy

Supervised	Unsupervised
Fisher LDA, Neural Network	PCA/SVD, ICA, t-SNE, ISOMAP, Neural Network

Linear	Non Linear
PCA/SVD, ICA, LDA	t-SNE, UMAP, ISOMAP, MDS

Out of Sample Extension
Given new sample, can you reduce its dimension with a pre-learned mapping?

Mapping	Visualization
PCA, ICA, LDA	t-SNE, UMAP, ISOMAP, MDS

https://en.wikipedia.org/wiki/Nonlinear_dimensionality_reduction

ICA: Independent Component Analysis

MDS: Multidimensional Scailing

LDA: Linear Discriminant Analysis

Non-linear Dimension Reduction

What if the mapping: F: X -> Y was not a linear mapping?

Hard to optimize over “all non-linear functions”

What are we even trying to do?

- Leads to many heuristics

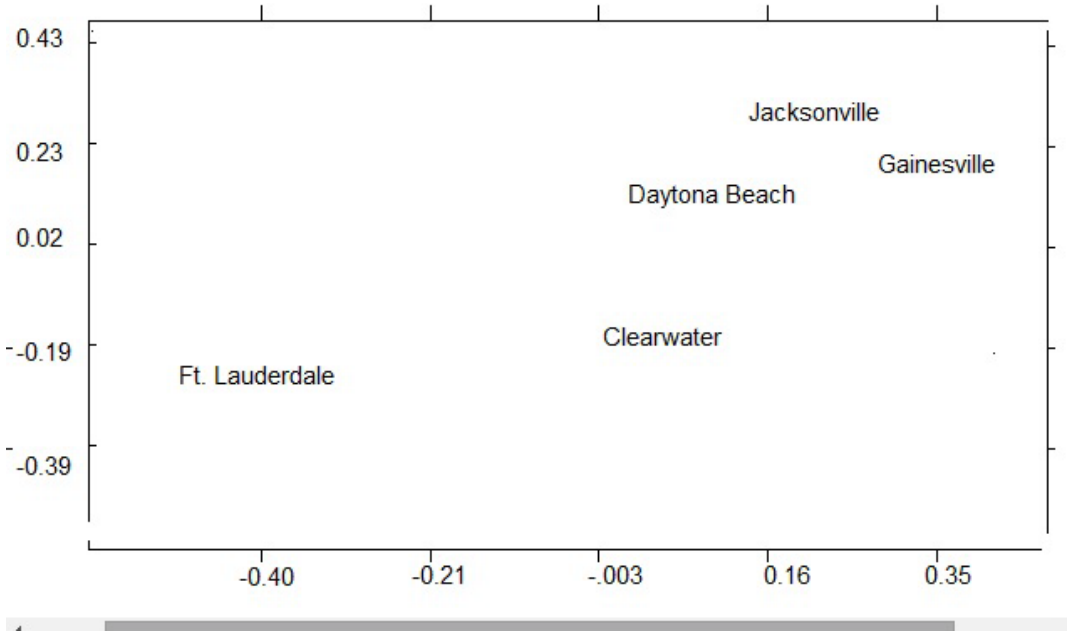
When would this even work?

- Requires the introduction of manifolds

MDS

- Distances from city to city

CITY	Clearwater	Daytona Beach	Ft. Lauderdale	Gainesville	Jacksonville
Clearwater	0	159	247	131	197
Daytona Beach	159	0	230	97	89
Ft. Lauderdale	247	230	0	309	317
Gainesville	131	97	309	0	68
Jacksonville	197	89	317	68	0



MDS map

MDS Algo

Assign a number of points to coordinates in n-dimensional space.

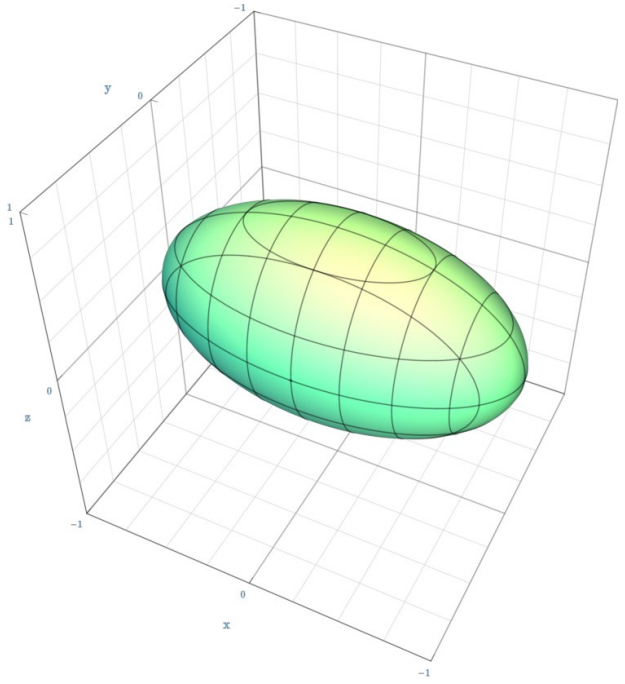
Calculate Euclidean distances for all pairs of points.

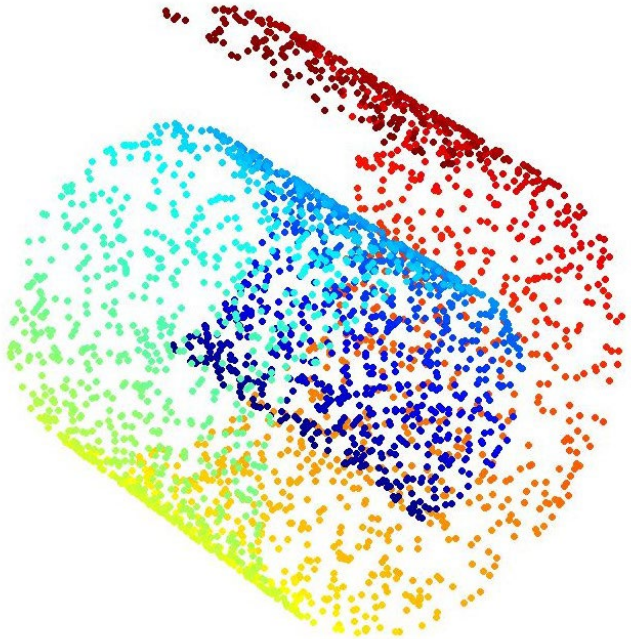
Compare the distance matrix with the original input matrix by evaluating the *stress* function.

Stress is a goodness-of-fit measure, based on differences between predicted and actual distances.

Adjust coordinates, if necessary, to minimize stress.

What's a Manifold?

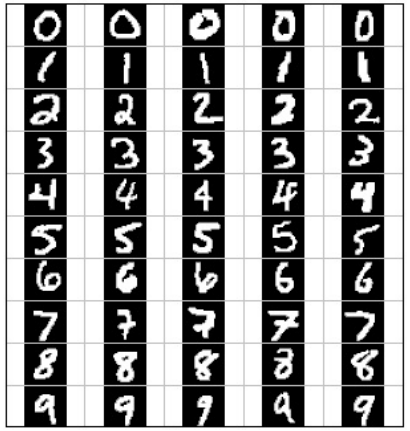




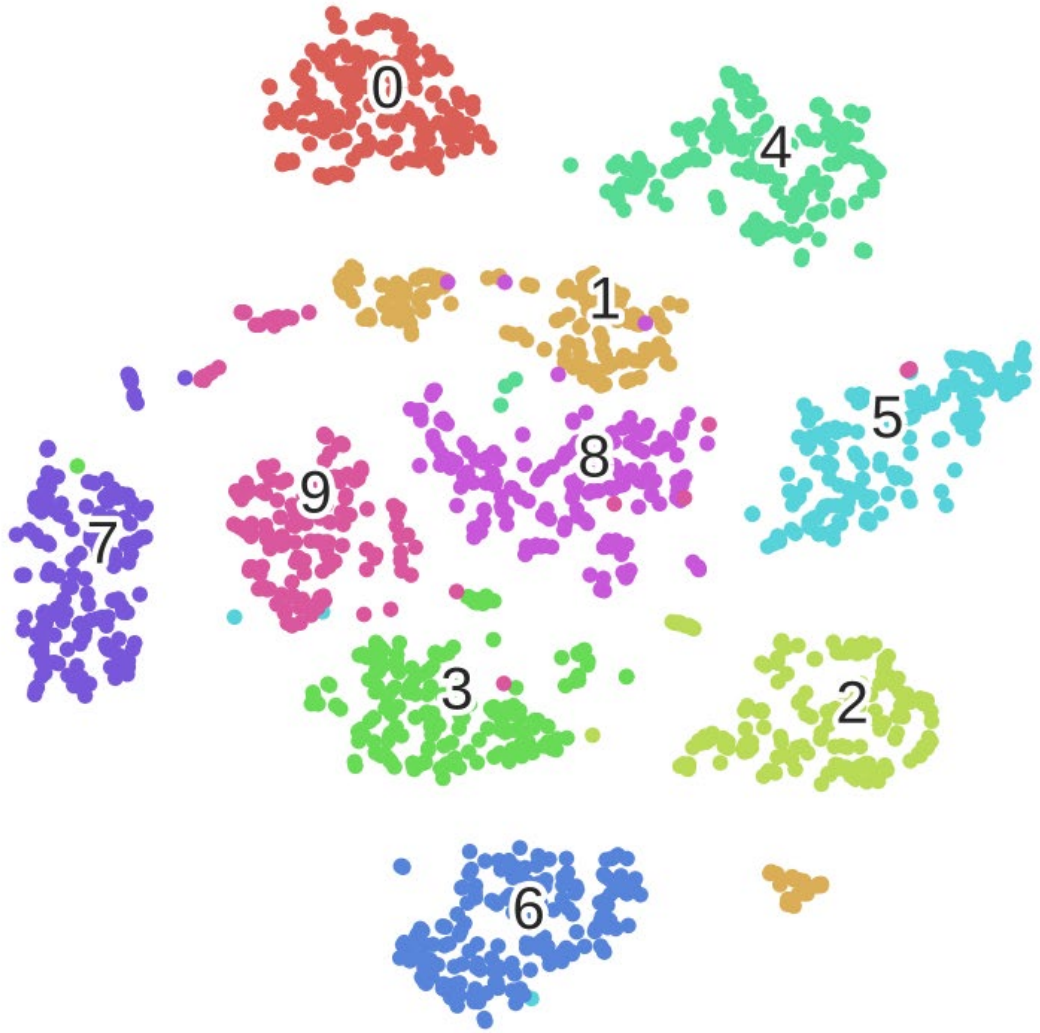
- Regular surfaces: "two dimensional object living in a three dimensional world."
- We don't have to restrict ourselves to 3 dimensions and 2 dimensions
- Manifold extends regular surfaces to: "d dimensional object living in a D dimensional world."

Why Should I Care?

- AKA: does my data really lie on (or near) a manifold?
- Some examples of where this might be true.
- Sometimes linear dimension reduction doesn't work



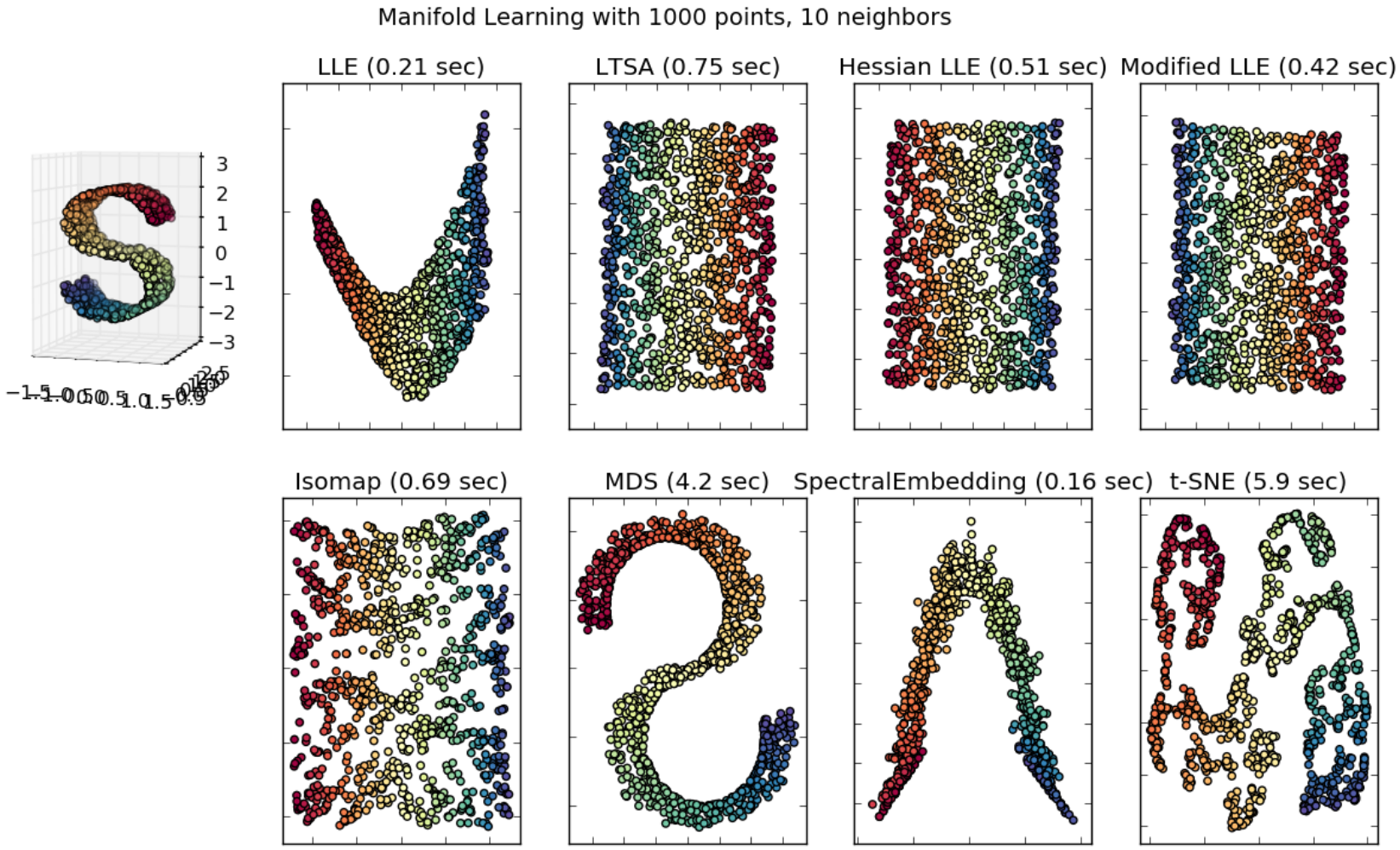
tSNE for MNIST data

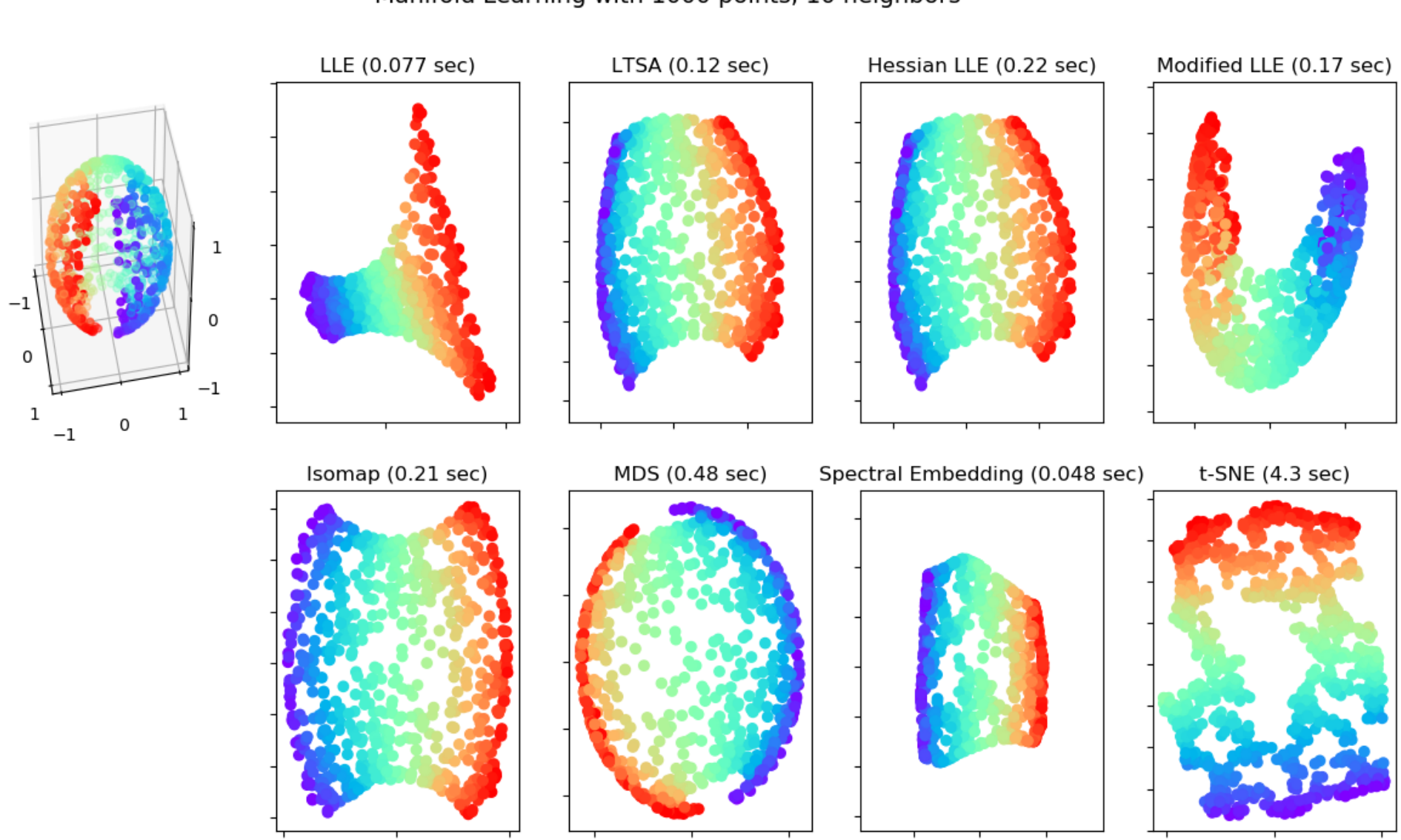


So how do we do it?

- There are a lot of different heuristics:
  - **Isomap**: preserve pairwise graph distances calculated using a sparse neighbhorhood graph
  - **Locally Linear Embedding**: preserve the local regression weights that reconstruct the original data with the new data.
  - **Local Tangent Space Alignment**: Estimate the tangent space at each point and put a restriction to allow for a global alignment step
  - **Spectral Embedding**: take the eigenvectors associated with the largest eigenvalues of the Laplacian Matrix
- They all result in a different embedding

Example Manifold Learning Results

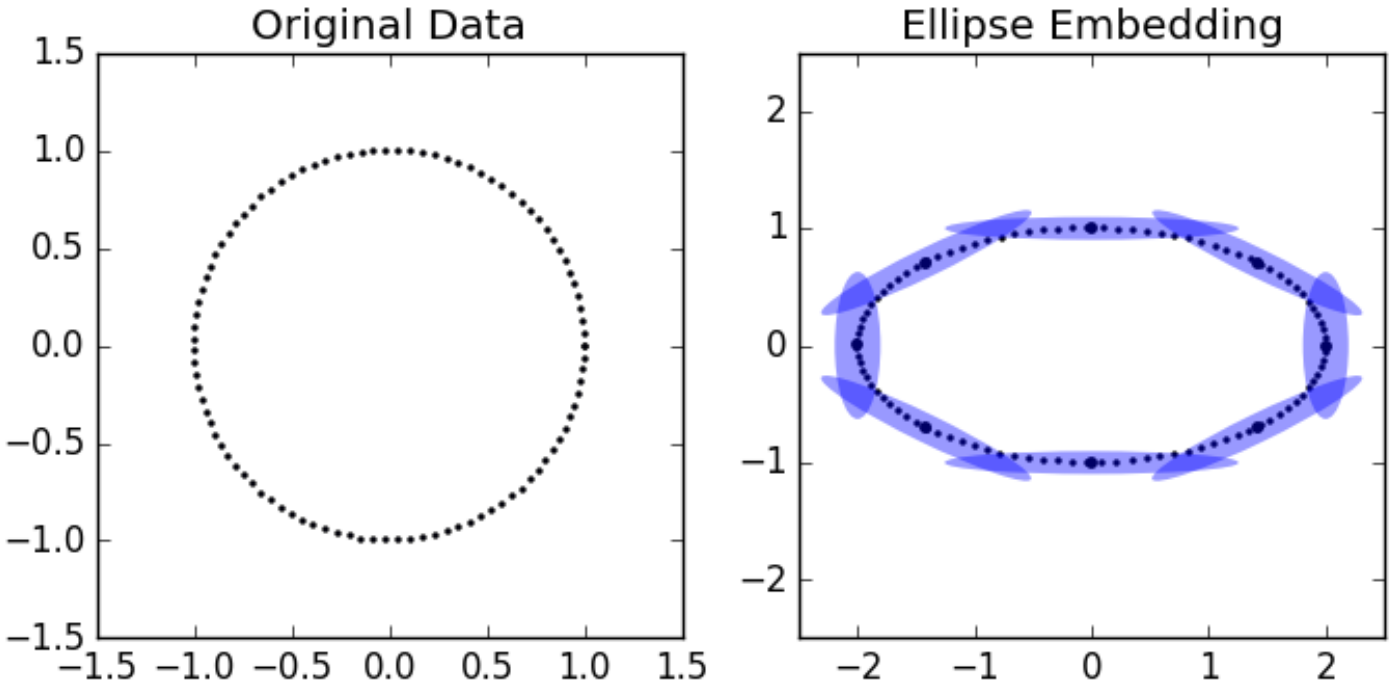



Manifold Learning with 1000 points, 10 neighbors
LLE (0.077 sec)
LTSA (0.12 sec)
Hessian LLE (0.22 sec)
Modified LLE (0.17 sec)
Isomap (0.21 sec)
MDS (0.48 sec)
Spectral Embedding (0.048 sec)
t-SNE (4.3 sec)

Challenges of Manifold Learning

- Each Method is based off a Heuristic – there's no agreement
- In almost all cases there is significant distortion even when an isometric embedding is possible
  - How do you measure distortion?
  - Potentially this is due to insufficient data
  - So-called "curse of dimensionality"
- Many Manifold Learning algorithms either do not scale well (fundamentally in their design) or have not been implemented in a scalable fashion
- Problem: you need lots of data for good estimates but methods don't scale to large data sets.

Distortion & the Riemannian Metric

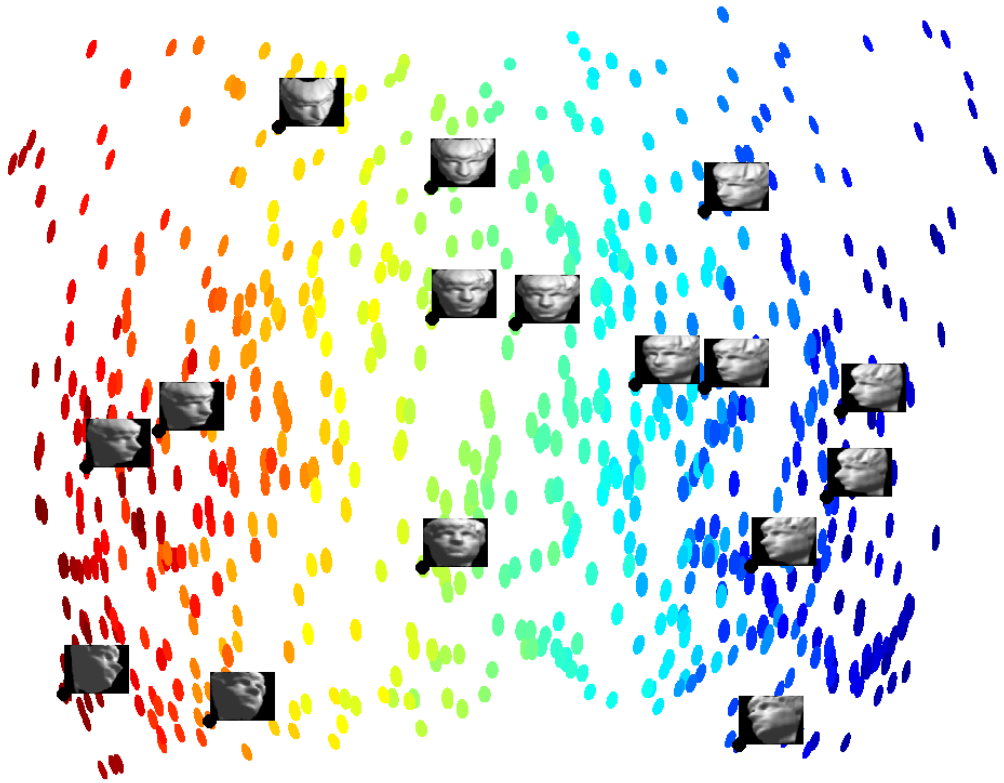


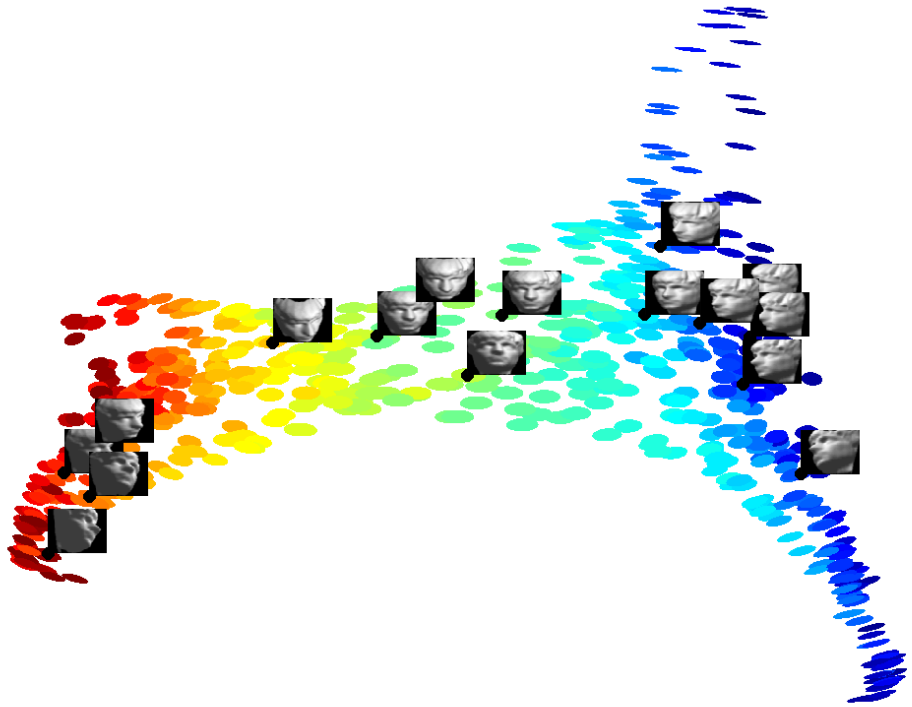
“Riemannian Metric” used to measure geometric quantities on a manifold

Creates a “push-forward” metric when we transform a data set.

If we use this metric then we achieve Isometry.

Distortion visually





Scaling Manifold Learning

- Most manifold learning algorithms are not implemented to scale.
- Sci-kit learn has most of the algorithms implemented but the primary objective of sci-kit learn is usability.
- Bottlenecks:
  - Neighborhood graphs
  - Eigendecomposition
- Solution:
  - Approximate Neighbors Graph (FLANN)
  - Sparse Symmetric Positive Definite Methods (LOBPCG)

Megaman

Scalable Manifold Learning in Python

Harnesses Cython, FLANN, LOBPCG & pre-conditioning to scale up-manifold learning algorithms.

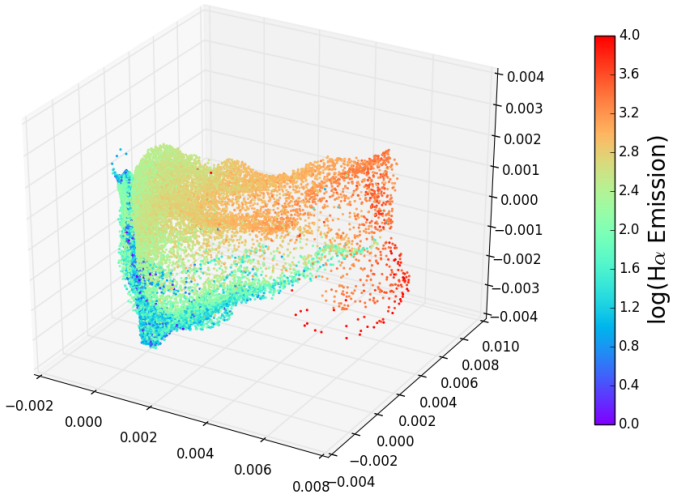
Smart with memory and designed with research in mind.

Github: https://github.com/mmp2/megaman

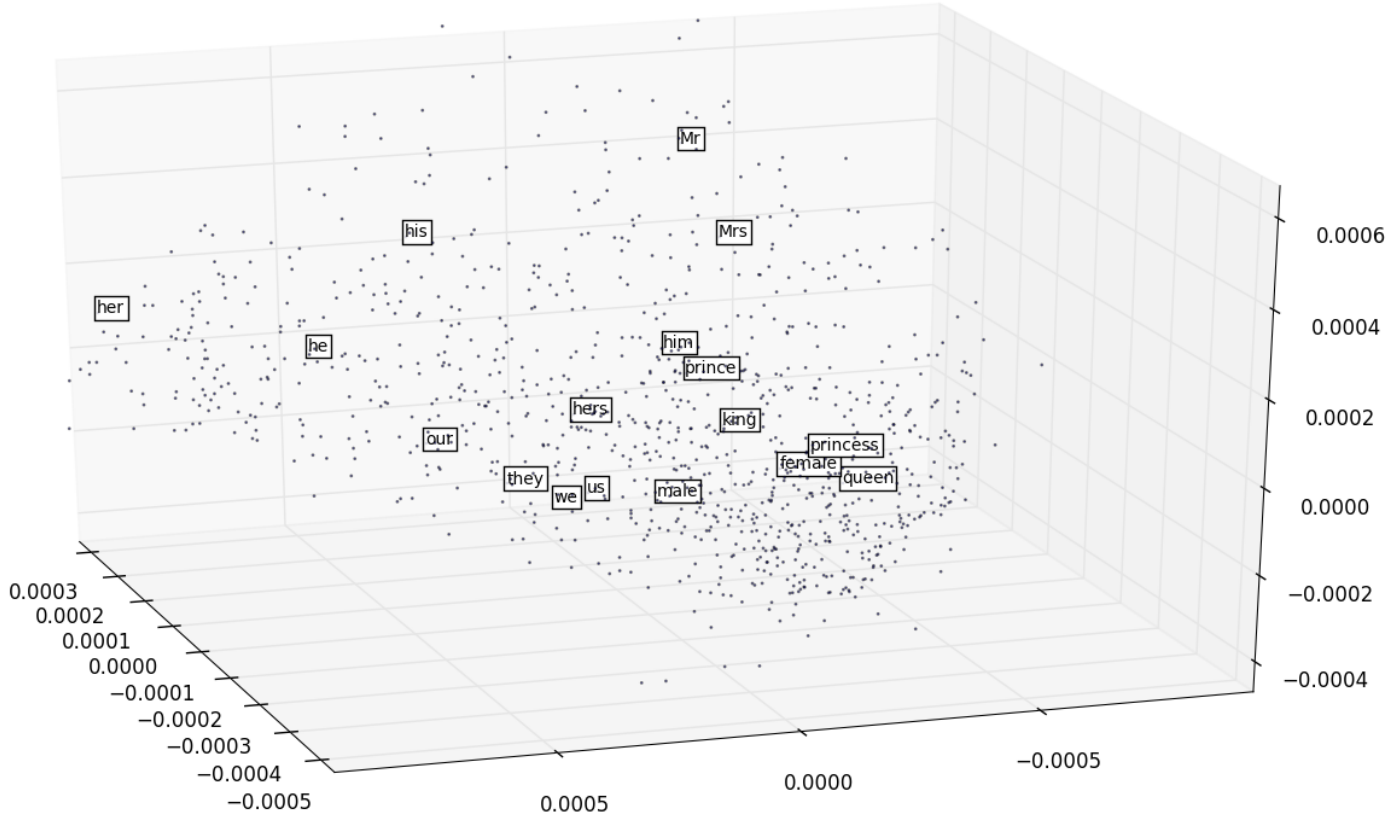
Website/Documentation: http://mmp2.github.io/megaman/

Question: how many of you are familiar with the character Megaman?

Megaman Results:



Sample of 675,000 observed spectra in 3750 dimensions. Colors indicate strength of Hydrogen alpha emissions (common in emission nebulae)



A sample of embedding 3M words & phrases in 300 dimensions taken using Word2Vec on Google News