Visualization

Slides by Damon Wischik

LECTURE ONE — chart literacy

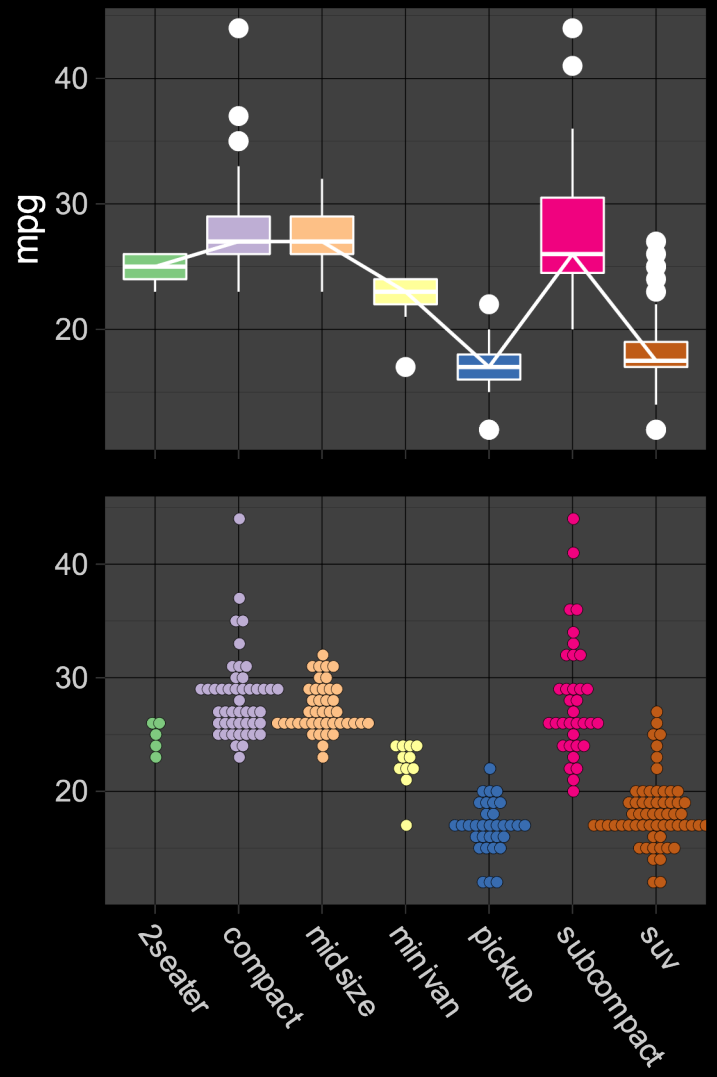
1. anatomy of a plot
2. scale theory
3. scale perception
4. making comparisons
5. atomic plots

LECTURE TWO — embedding

6. unsupervised learning
7. dimension reduction
8. self-supervised learning
9. content scales

6. Supervised versus unsupervised plotting

Some plots are associated with supervised learning, some with unsupervised.

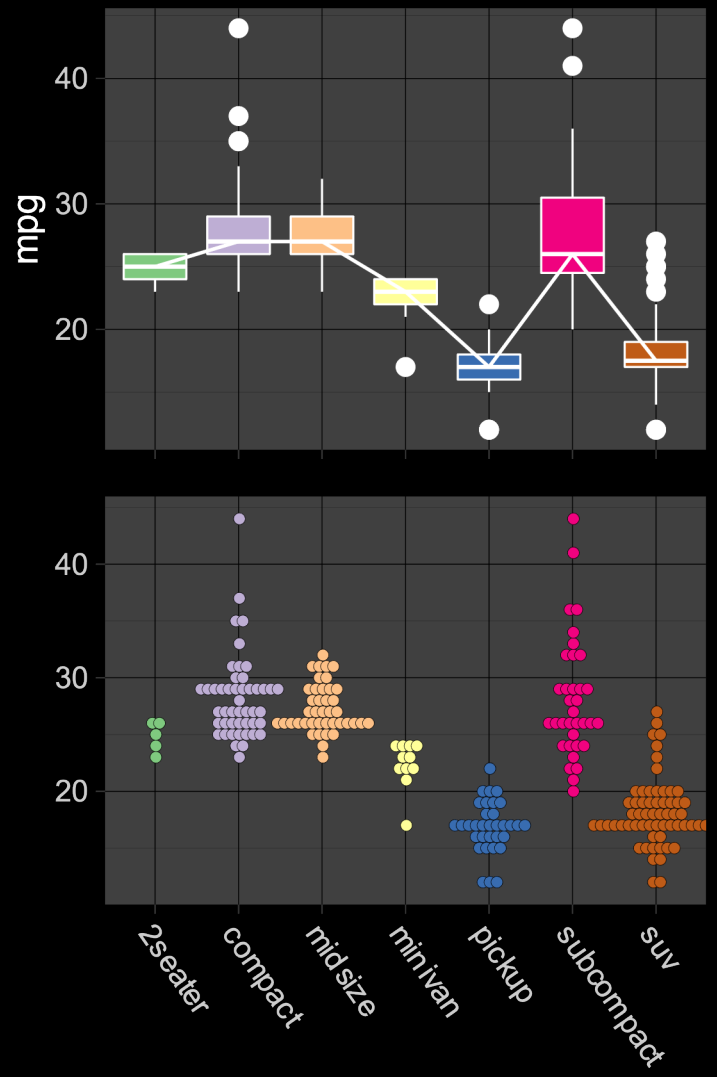


Supervised learning

- What is the conditional distribution of $Y|X$?
- How do I build a predictor / classifier?

6. Supervised versus unsupervised plotting

Some plots are associated with supervised learning, some with unsupervised.



Supervised learning

- What is the conditional distribution of $Y|X$?
- How do I build a predictor / classifier?

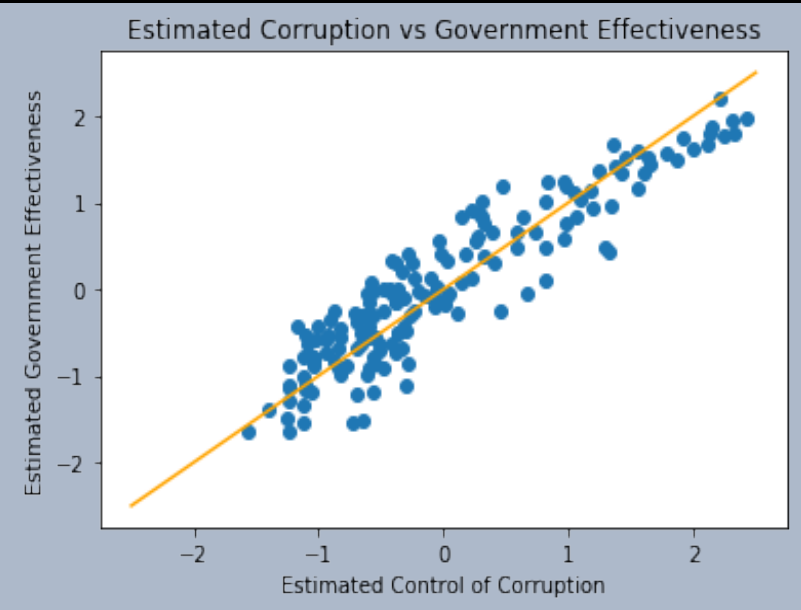
Generative (unsupervised) learning

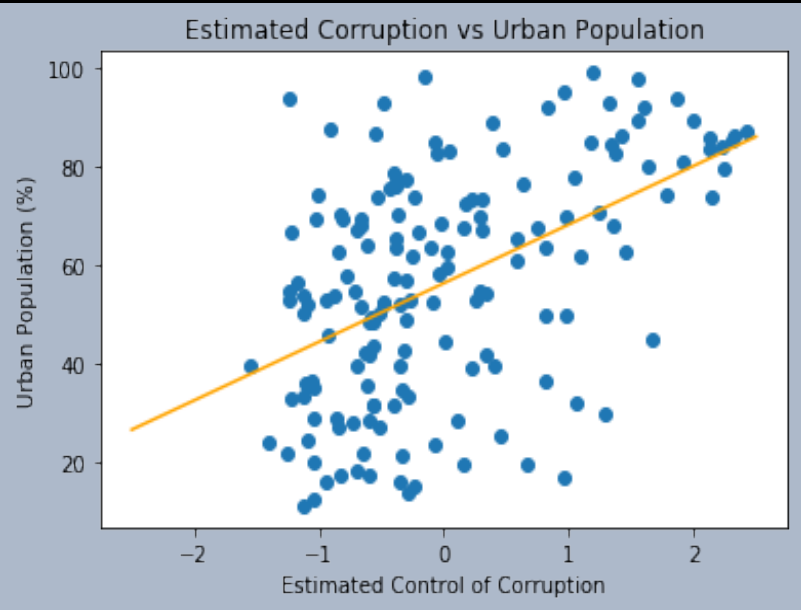
- What is the joint distribution of (X, Y)?
- How do I generate new random datapoints?

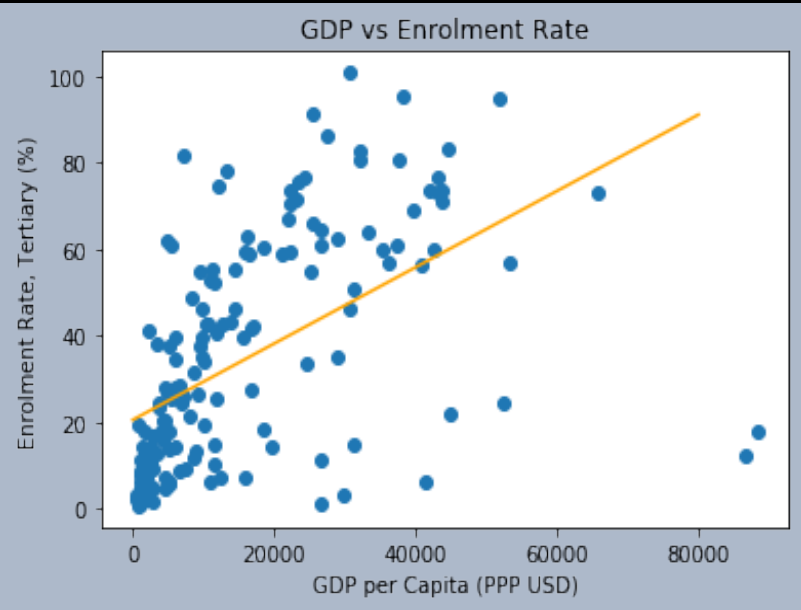
Who decides which variable is to be the predictor and which is to be the response?

Data set: World Bank country / demographic data

Country Name	GDP per Capita (PPP USD)	Population Density (persons per sq km)	Population Growth Rate (%)	Urban Population (%)	Life Expectancy at Birth (avg years)
Afghanistan	1560.67	44.62	2.44	23.86	60.07
Albania	9403.43	115.11	0.26	54.45	77.16
Algeria	8515.35	15.86	1.89	73.71	70.75

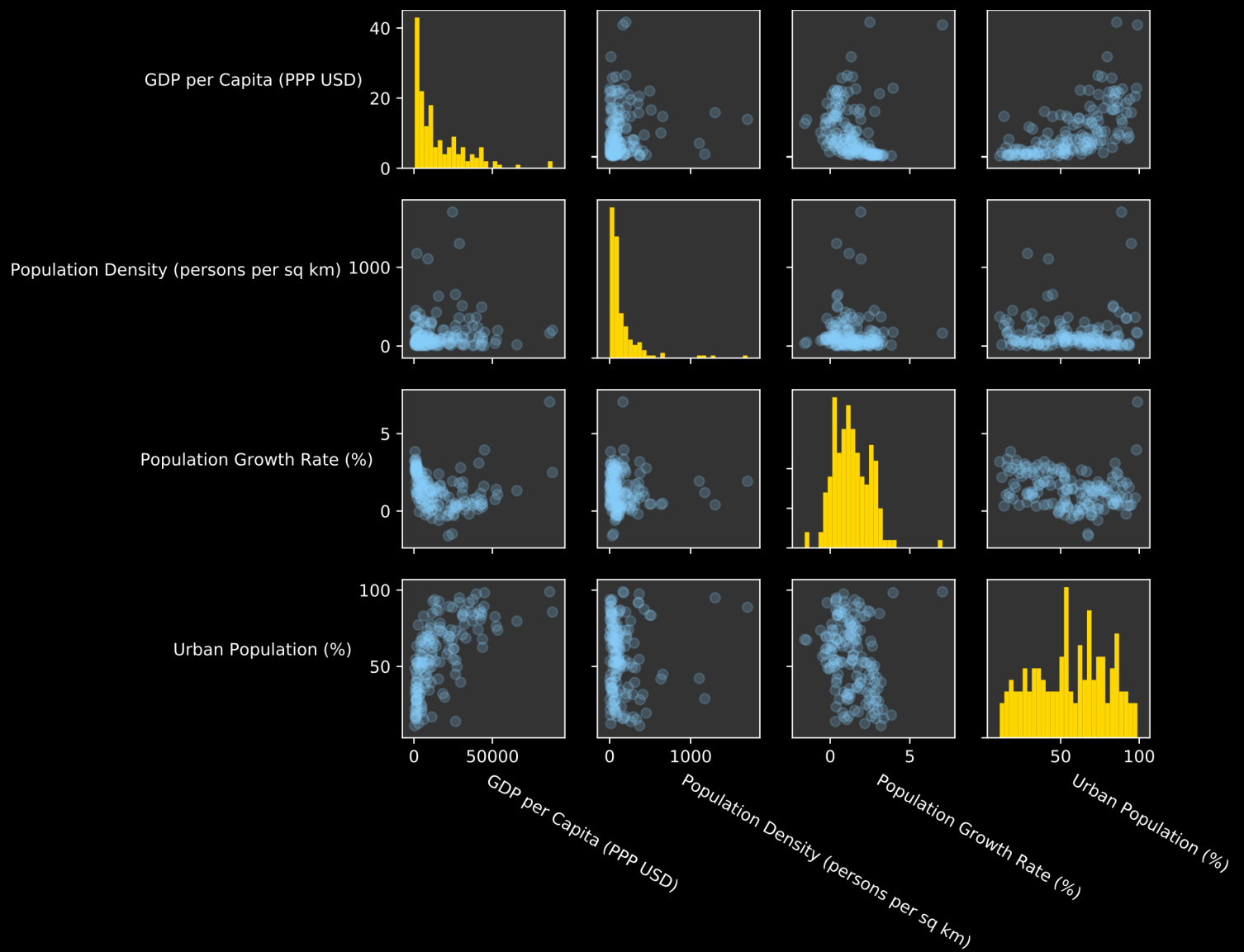




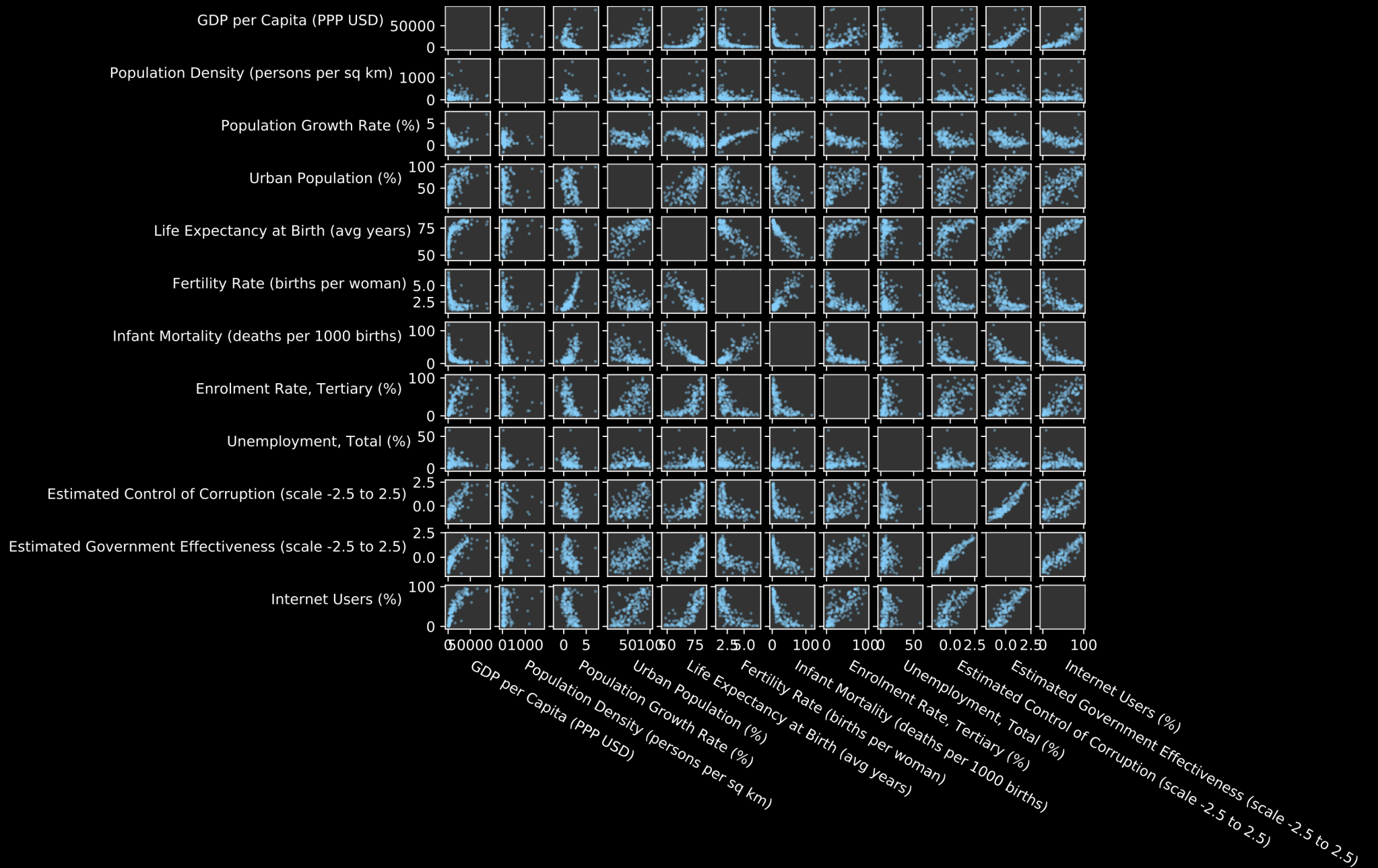


Some of the linear models from Lecture 2

A splom (scatter plot matrix) shows all pairs of variables. If you don’t have a definite prediction task in mind, this is a good starting point.

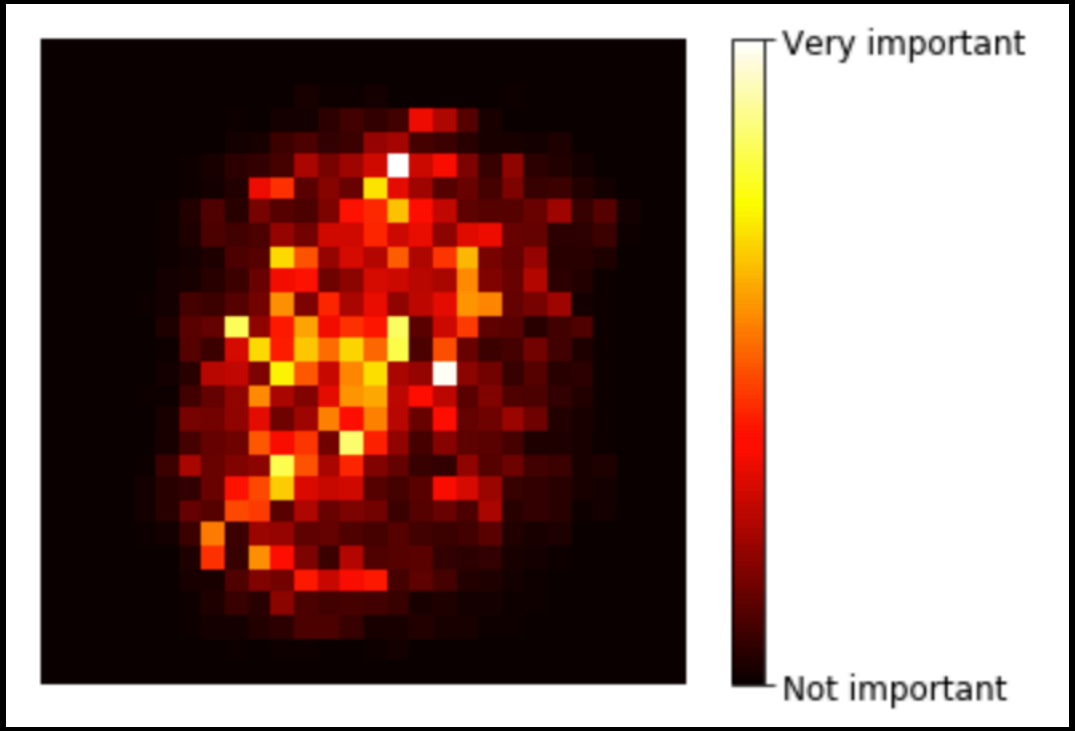


A splom is unwieldy for 12 variables. And what if we had 1200 or 12000?



The curse of dimensionality

- A model that relies on 10,000 features may be slow to train
- Is there a way to make it more tractable?
- **Solution** – dimensionality reduction. Speeds up training + helps visualization
- **Caution**: may help filter out some noise and result in better performance, but generally it won't as it loses information

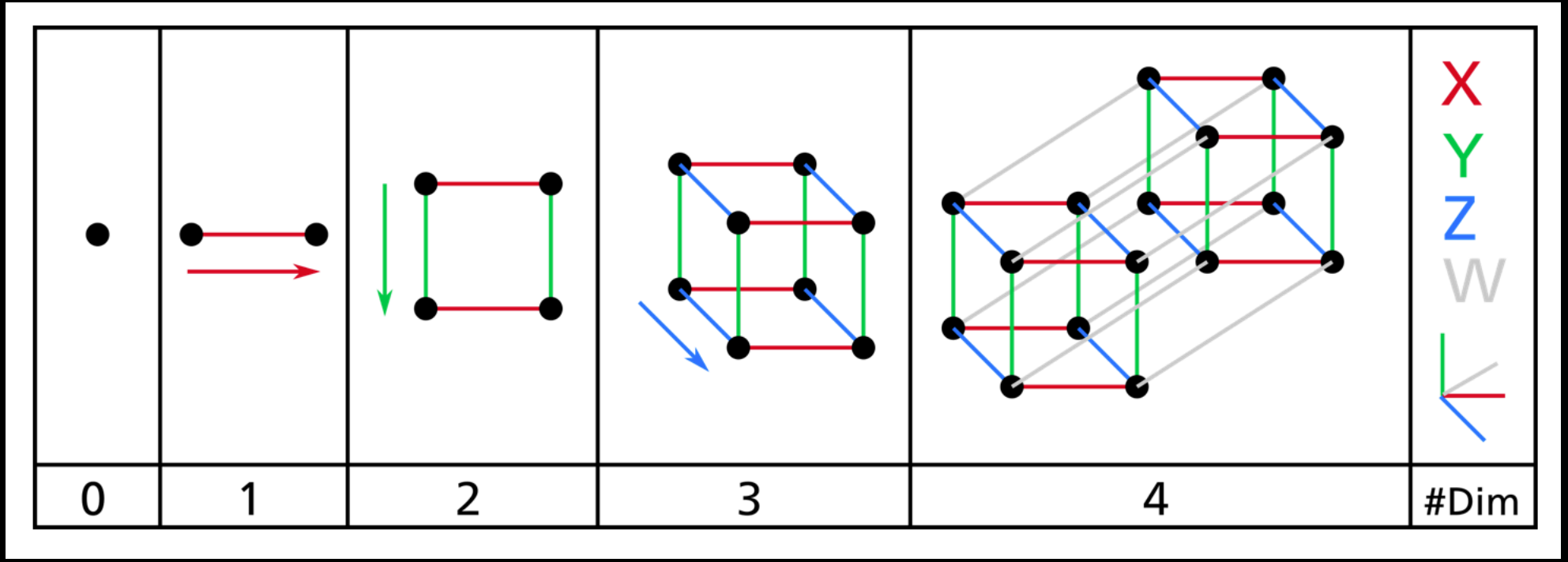


Reminder: what you might have found out for the digits dataset – not all pixels are equally informative

+ neighboring pixels are often highly correlated ⟶ can merge them?

The curse of dimensionality

- A random point in a unit square (1 × 1) will have less than a 0.4% chance of being located less than 0.001 from a border (“extreme”) vs. 99.99% for a 10,000-dimensional unit hypercube[1]
- A distance between 2 random points in a 2D unit space ≈0.52, in 3D ≈0.66, in 1,000,000-D ≈408.25 [2] ⇒ sparse, prone to overfitting



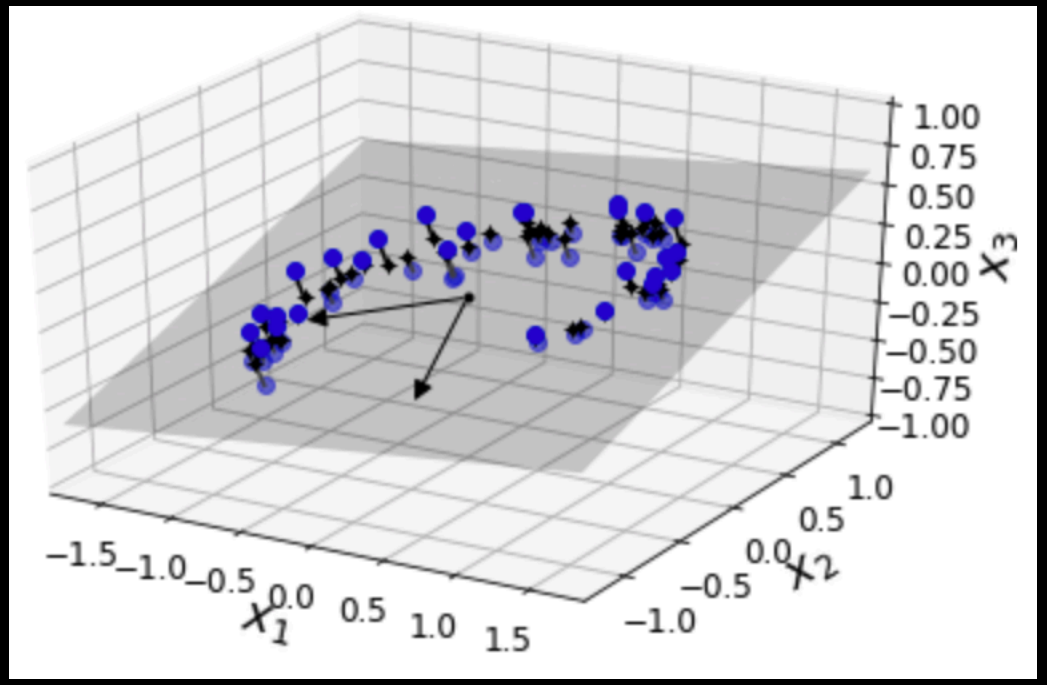
[1] https://datascience.stackexchange.com/questions/45690/many-things-behave-differently-in-high-dimensional-space
[2] https://mathworld.wolfram.com/HypercubeLinePicking.html
https://en.wikipedia.org/wiki/Hypercube

7. Dimension reduction / PCA

PCA is a tool that takes in high-dimensional data and compresses it lossily into fewer dimensions.

7. Dimension reduction / PCA

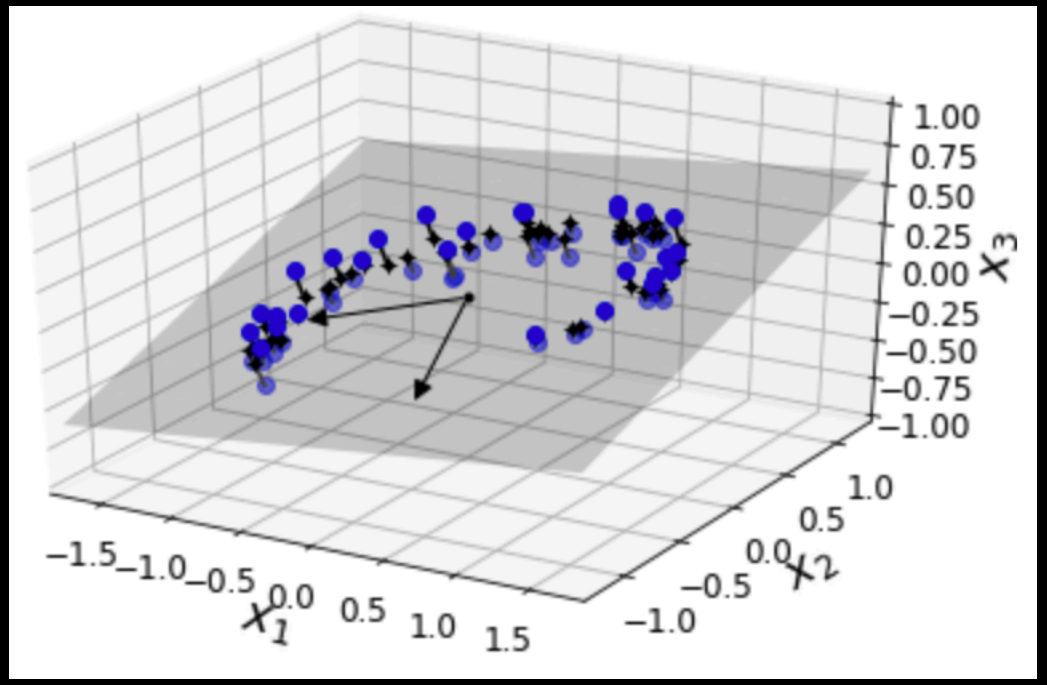
PCA is a tool that takes in high-dimensional data and compresses it lossily into fewer dimensions.

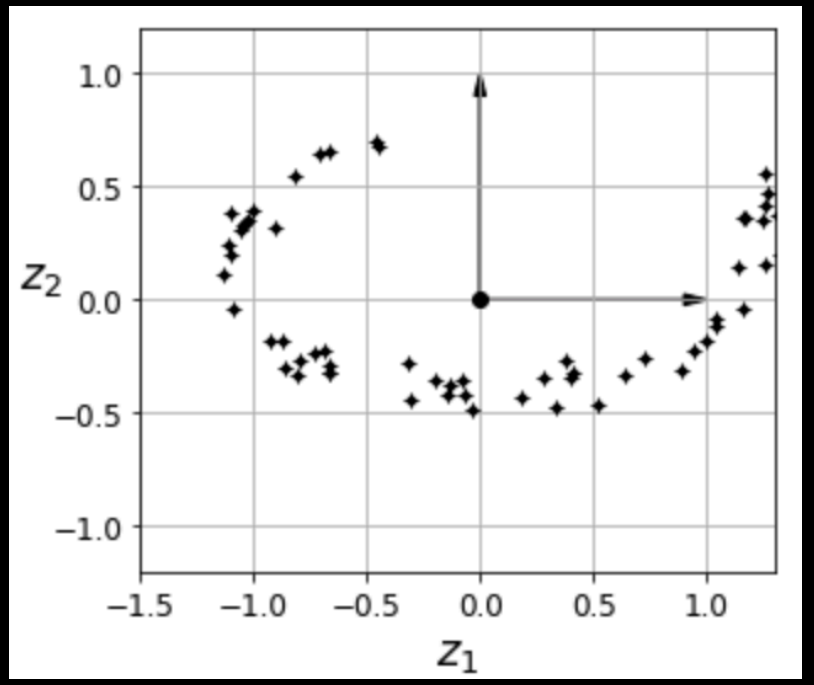


Geron (2017). *Hands-on Machine Learning with Scikit-Learn & TensorFlow*

7. Dimension reduction / PCA

PCA is a tool that takes in high-dimensional data and compresses it lossily into fewer dimensions.

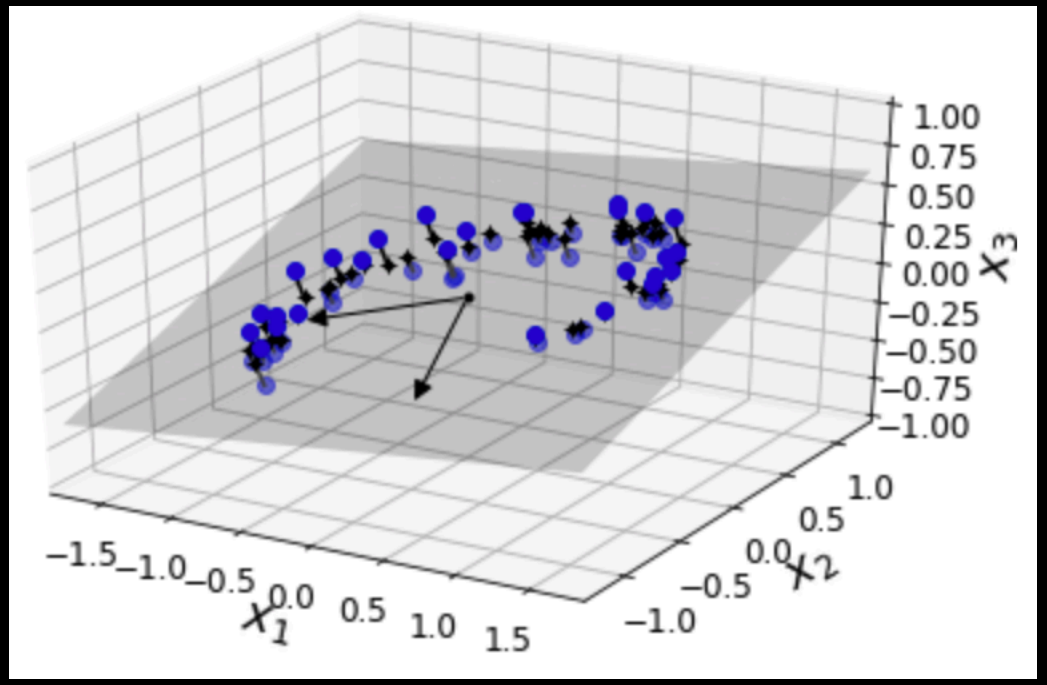


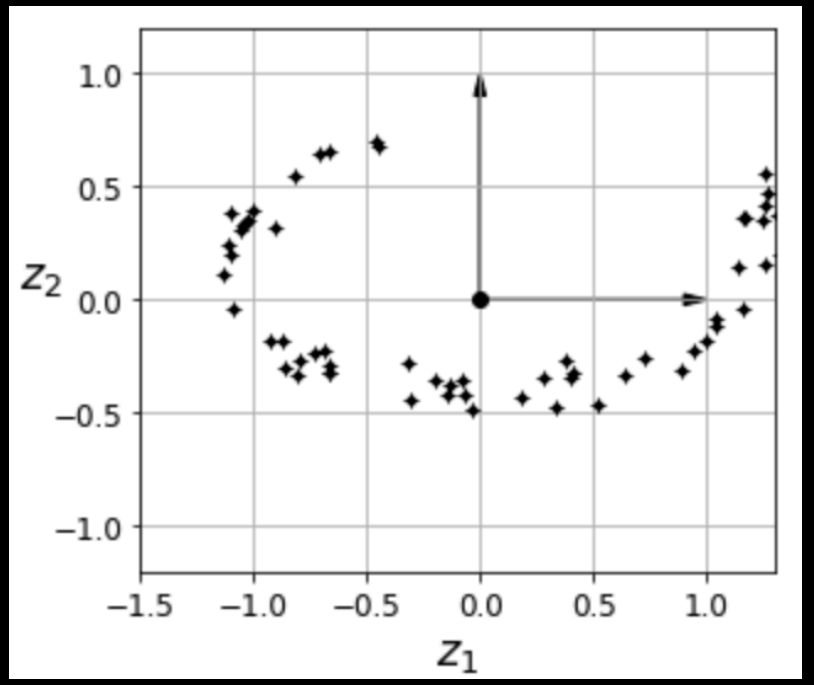


Geron (2017). *Hands-on Machine Learning with Scikit-Learn & TensorFlow*

7. Dimension reduction / PCA

Principal Component Analysis (PCA) first identifies a hyperplane that lies closest to the data, and then projects the data onto it.

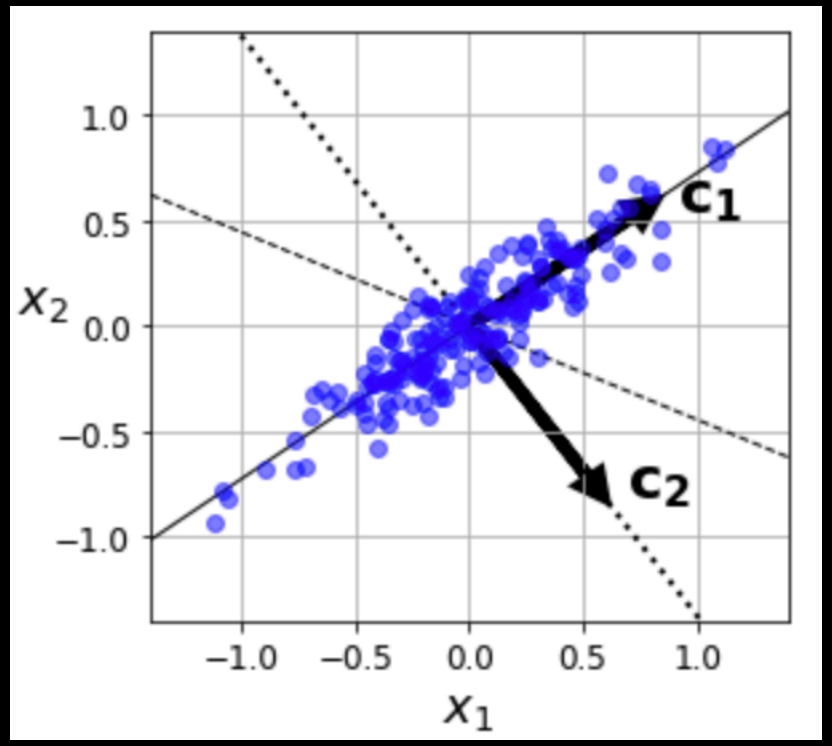




Geron (2017). *Hands-on Machine Learning with Scikit-Learn & TensorFlow*

How to choose the right hyperplane: preserving the variance

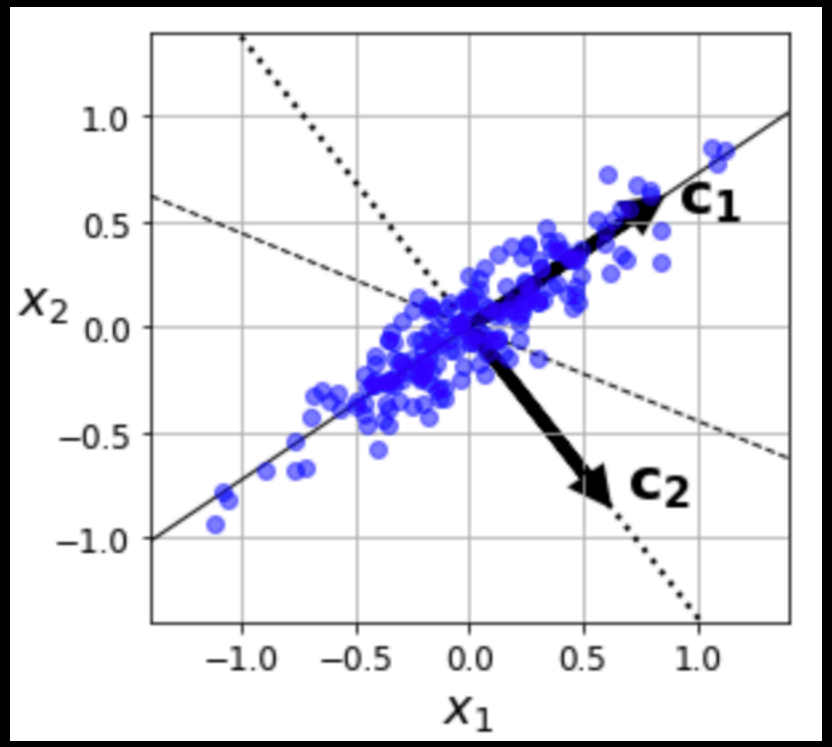
Which line preserves maximum variance (i.e., likely loses less information)?

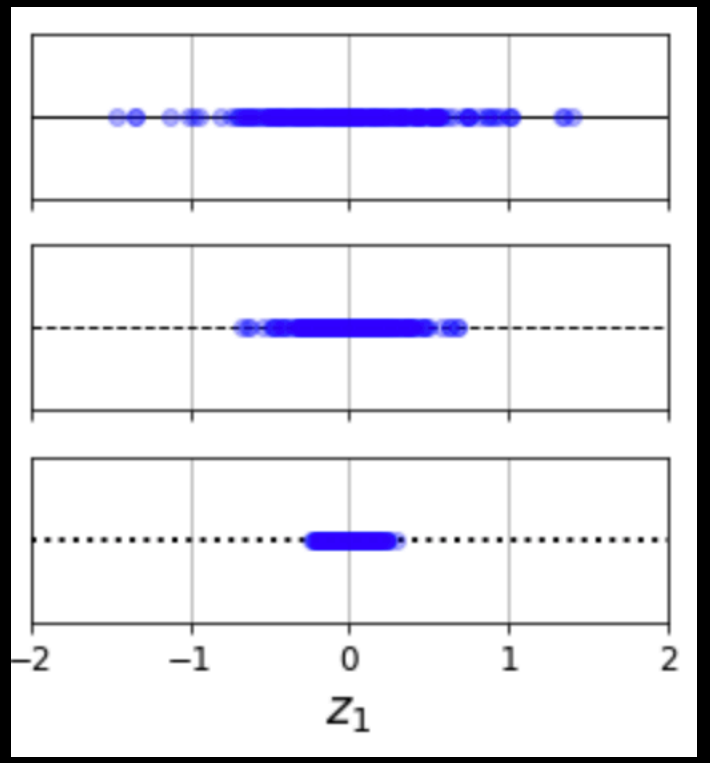


Geron (2017). *Hands-on Machine Learning with Scikit-Learn & TensorFlow*

How to choose the right hyperplane: preserving the variance

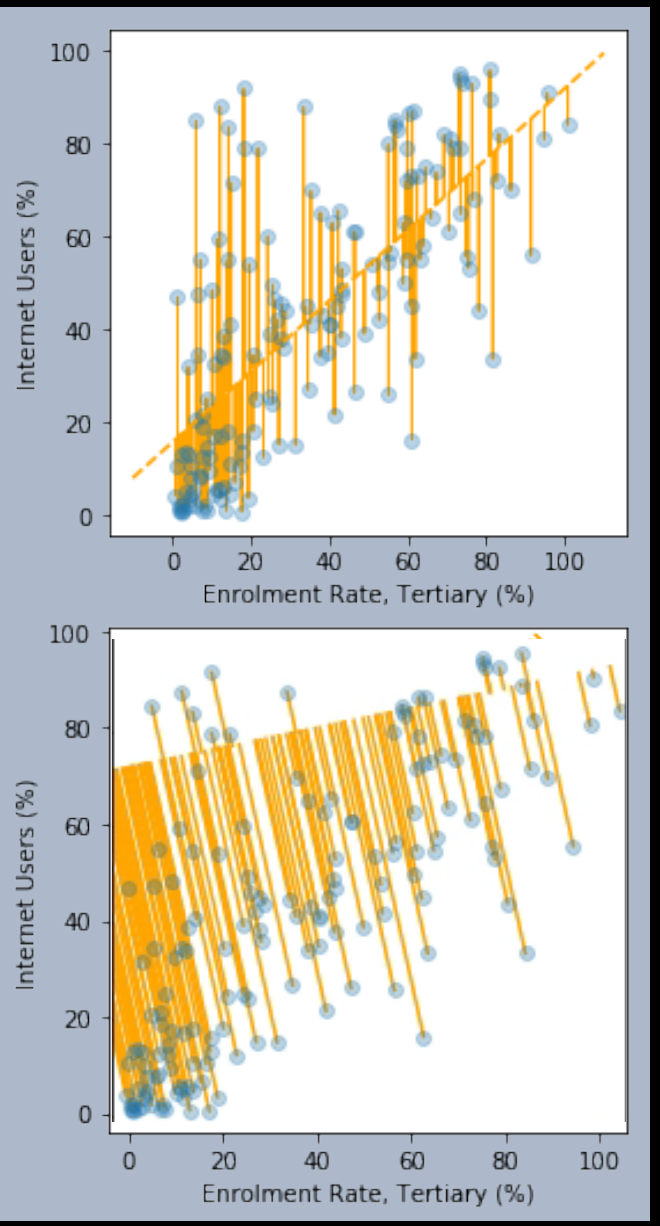
Which line preserves maximum variance? Solid line (c_1)





Geron (2017). *Hands-on Machine Learning with Scikit-Learn & TensorFlow*

7. Dimension reduction / PCA



In linear regression, we model the data by

$$y_i = \alpha\, x_i + \beta + \varepsilon_i$$

and learning consists in:
choosing the parameters α and β to minimize

$$L(\alpha, \beta) = \frac{1}{2}\sum_{i=1}^{n} \varepsilon_i^2$$

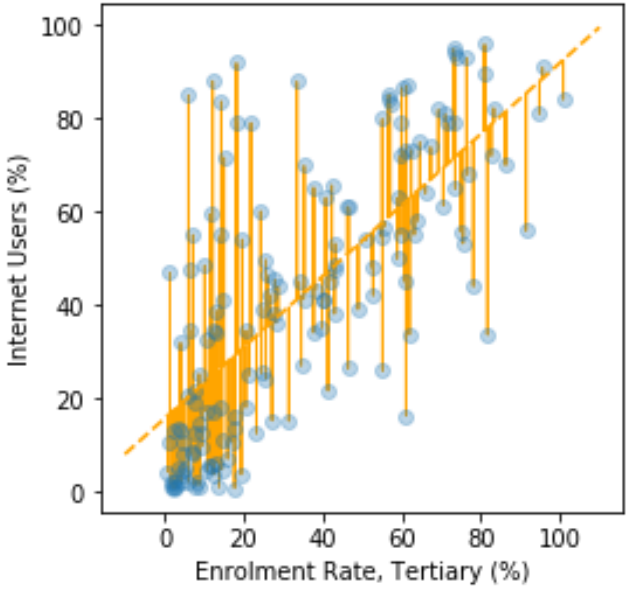
$$L(\alpha, \beta) = \frac{1}{2}\sum_{i=1}^{n} (\alpha\, x_i + \beta - y_i)^2$$

predicted

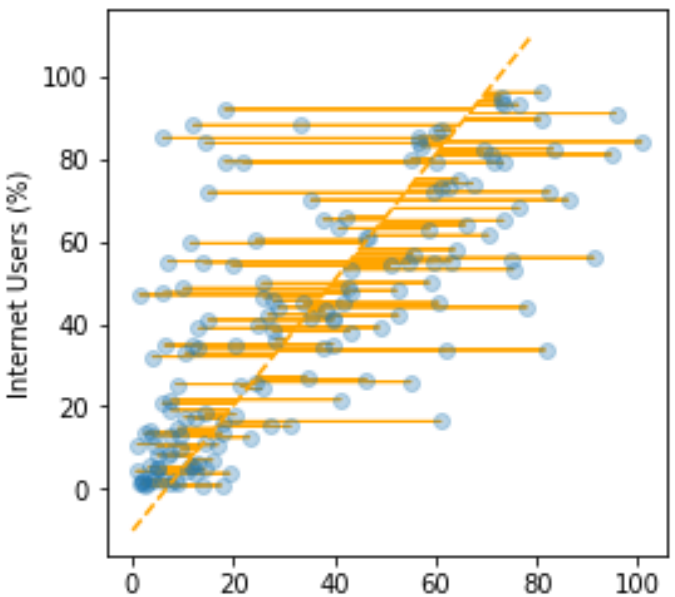
observed

7. Dimension reduction / PCA

$$y = 15.81 + 0.7619x$$



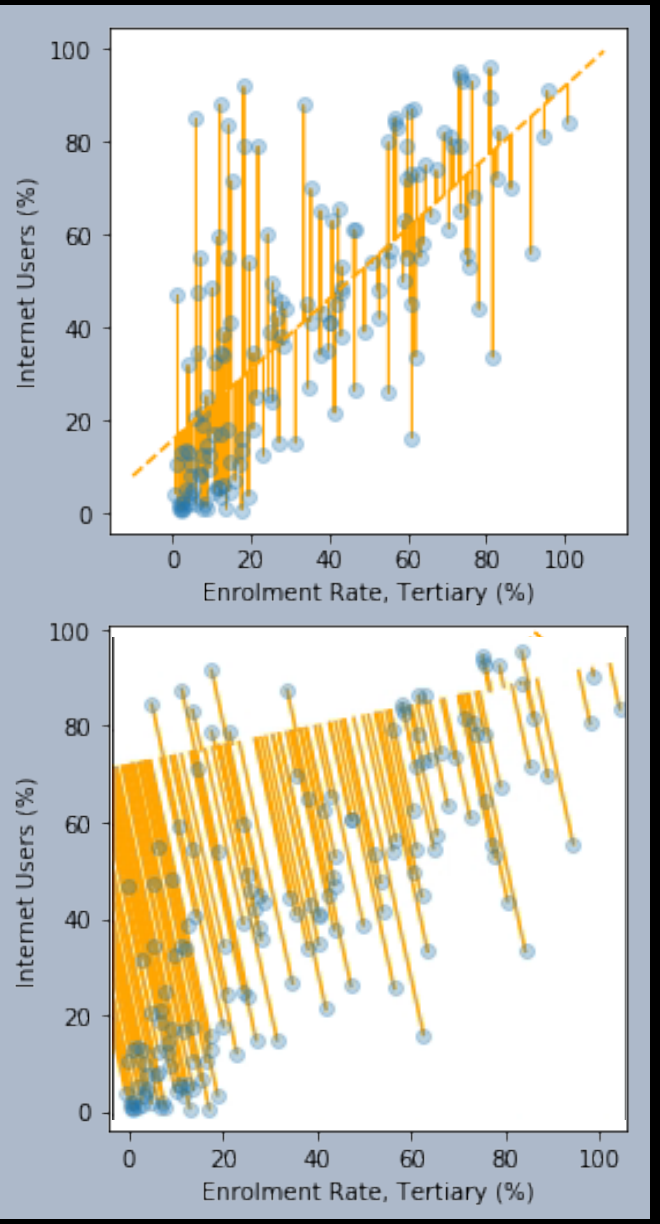
$$x = 6.728 + 0.6583y$$
$$\Rightarrow \quad y = -10.22 + 1.519x$$



If we regress y against x, should we get the same answer as regressing x against y?

If not, is there a way to express the relationship between x and y that doesn't require an arbitrary choice of response versus predictor?

7. Dimension reduction / PCA



Optimal projection: model the data by

$$y_i = \alpha\, x_i + \beta$$

choosing the parameters α and β to minimize:

$$L(\alpha, \beta) = \frac{1}{2}\sum_{i=1}^{n} \left\| \begin{pmatrix} x_i \\ y_i \end{pmatrix} - proj_{\alpha,\beta} \begin{pmatrix} x_i \\ y_i \end{pmatrix} \right\|^2$$

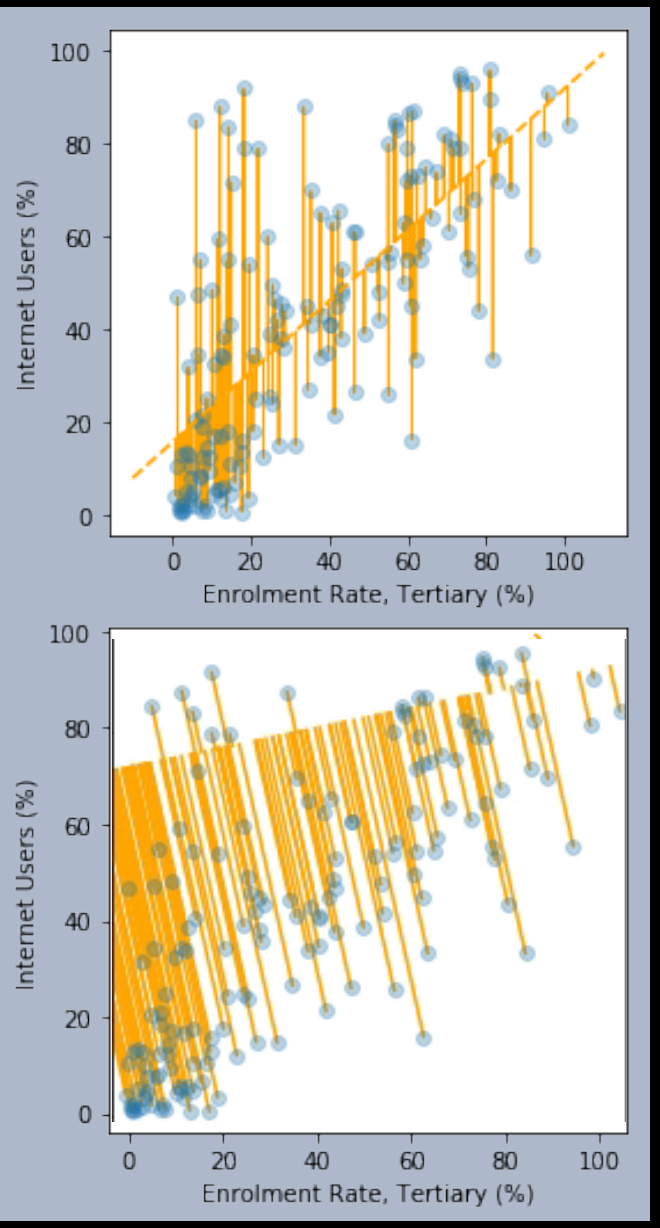
A better way to define this subspace: in 1d PCA, we model the data by

$$\begin{pmatrix} x_i \\ y_i \end{pmatrix} = \begin{pmatrix} \mu_x \\ \mu_y \end{pmatrix} + \lambda_i \begin{pmatrix} \delta_x \\ \delta_y \end{pmatrix} + \begin{pmatrix} \varepsilon_{x,i} \\ \varepsilon_{y,i} \end{pmatrix}$$

and learning consists in:
choosing the parameters μ and δ to minimize

$$L(\mu, \delta) = \frac{1}{2}\sum_{i=1}^{n} \left\| \begin{pmatrix} \varepsilon_{x,i} \\ \varepsilon_{y,i} \end{pmatrix} \right\|^2$$

7. Dimension reduction / PCA



In linear regression, we model the data by

$$y_i = \alpha\, x_i + \beta + \varepsilon_i$$

and learning consists in:
choosing the parameters α and β to minimize

$$L(\alpha, \beta) = \frac{1}{2}\sum_{i=1}^{n} \varepsilon_i^2$$

This model says: let's represent a 2d datapoint (x_i, y_i) using a 1d summary (λ_i).

In 1d PCA, we model the data by

$$\begin{pmatrix} x_i \\ y_i \end{pmatrix} = \begin{pmatrix} \mu_x \\ \mu_y \end{pmatrix} + \lambda_i \begin{pmatrix} \delta_x \\ \delta_y \end{pmatrix} + \begin{pmatrix} \varepsilon_{x,i} \\ \varepsilon_{y,i} \end{pmatrix}$$

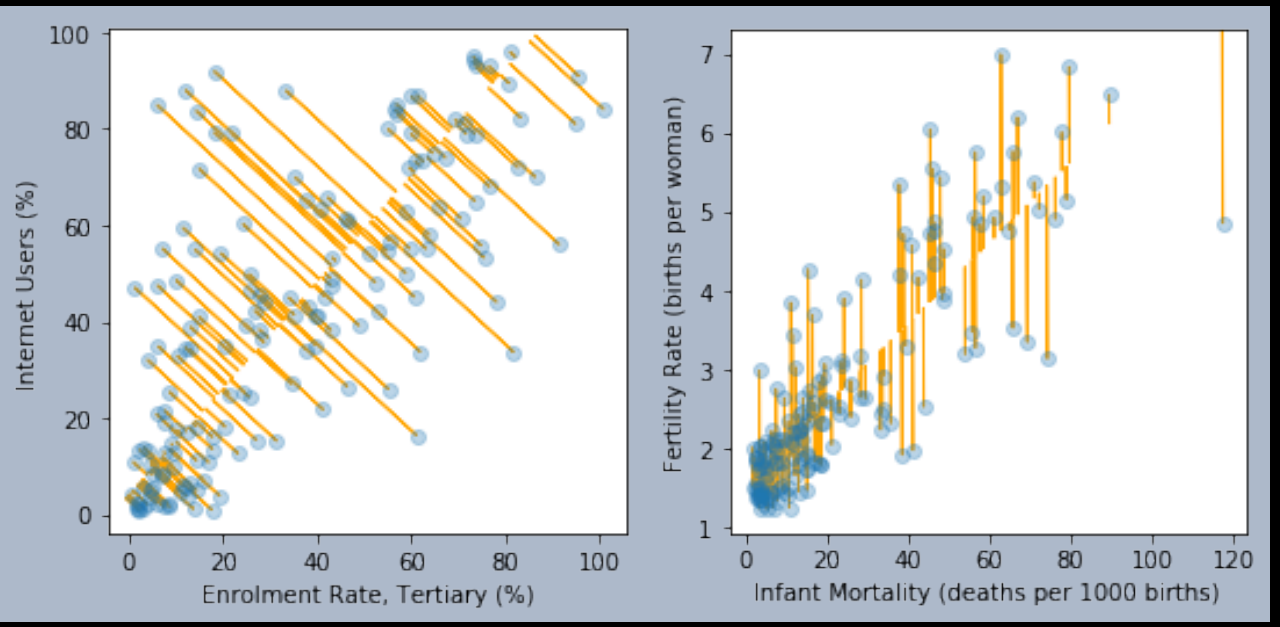
and learning consists in:
choosing the parameters μ and δ to minimize

$$L(\mu, \delta) = \frac{1}{2}\sum_{i=1}^{n} \left\| \begin{pmatrix} \varepsilon_{x,i} \\ \varepsilon_{y,i} \end{pmatrix} \right\|^2$$

7. Dimension reduction / PCA

A PCA gotcha:

Why do these two PCA plots, on two different data features, come out so different?



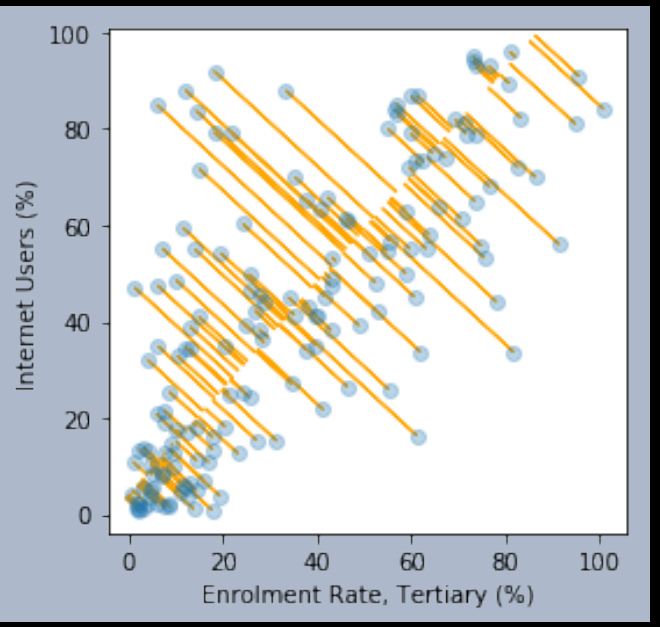
Because we're minimizing

$$\frac{1}{2}\sum_{i=1}^{n}(\varepsilon_{x,i}+\varepsilon_{y,i})^2$$

and the x and y units are so different.

Solution: scale the x and y columns so they have the same standard deviation, before doing the fit.

7. Dimension reduction / PCA



- PCA is a tool that takes in high-dimensional data and compresses it lossily into fewer dimensions.
- This avoids an arbitrary choice of predictor vs response variables.

- We've shown it for 2d → 1d compression
  but it also works for arbitrary dimensions $N \rightarrow K$.

How to run PCA

We model N-dimensional data by picking a K-dimensional subspace, call it S_K, and representing each point by its projection onto that subspace,

$$\vec{x}_i = \vec{\mu} + \sum_{k=1}^{K} \lambda_{k,i}\, \vec{\delta}_k + \vec{\varepsilon}_i$$

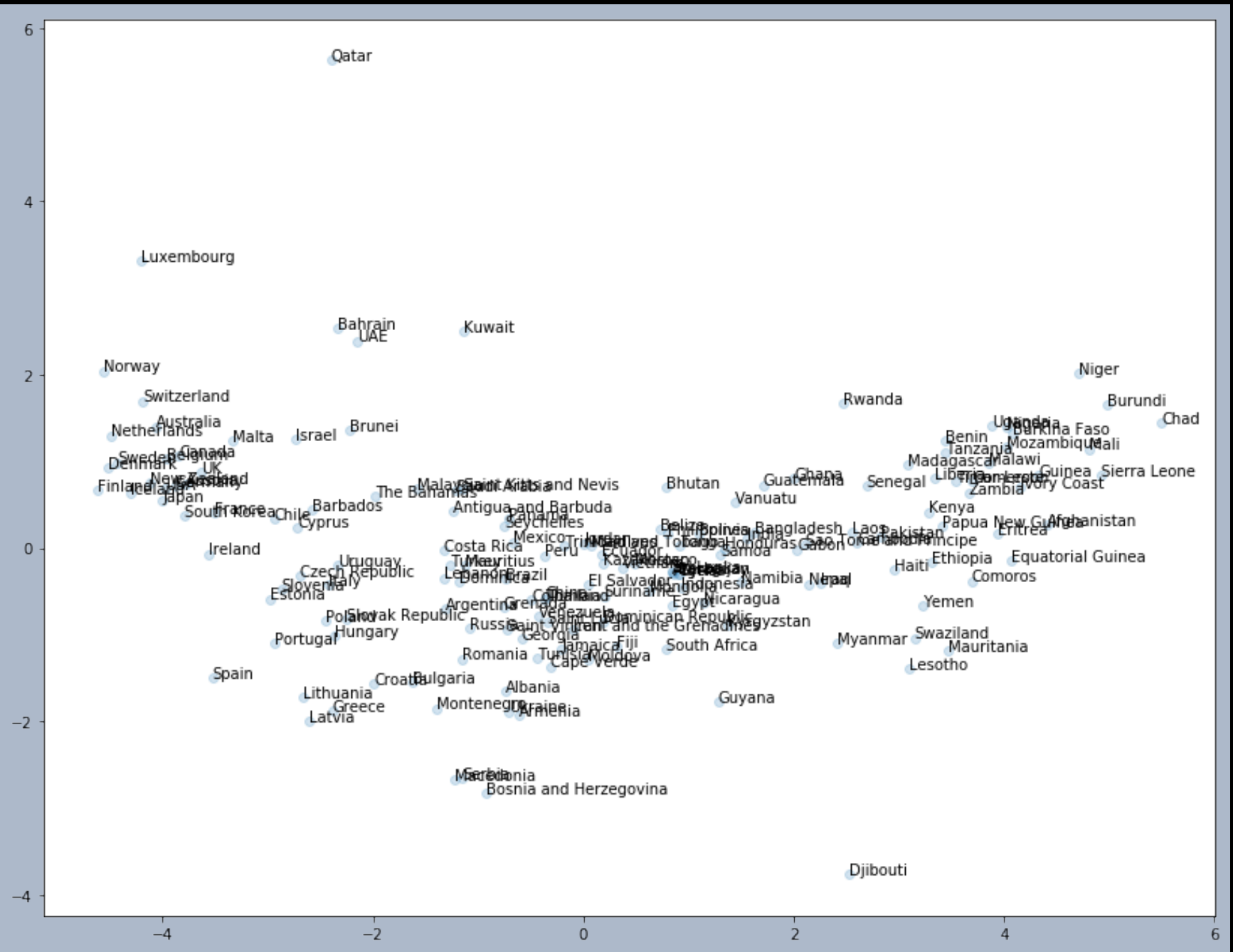
The subspace S_K is specified by an offset $\vec{\mu}$ and a basis $\{\vec{\delta}_1, \ldots, \vec{\delta}_K\}$. We pick the subspace so as to minimize the mean square error $\sum \|\vec{\varepsilon}_i\|^2$.

TECHNICAL NOTE
There are multiple bases that could be chosen to specify a given subspace. How does PCA pick a particular basis to specify S_K? It turns out, via the magic of linear algebra, that the optimal subspaces are nested ($S_1 \subset S_2 \subset \cdots \subset S_N$) and so a sensible choice is to pick the components $\vec{\delta}_k$ be an ordered basis such that

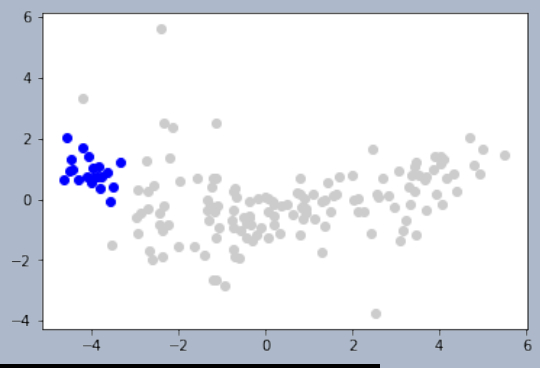
$$S_1 = \text{span}(\vec{\delta}_1), \qquad S_2 = \text{span}(\vec{\delta}_1, \vec{\delta}_2), \qquad \ldots$$

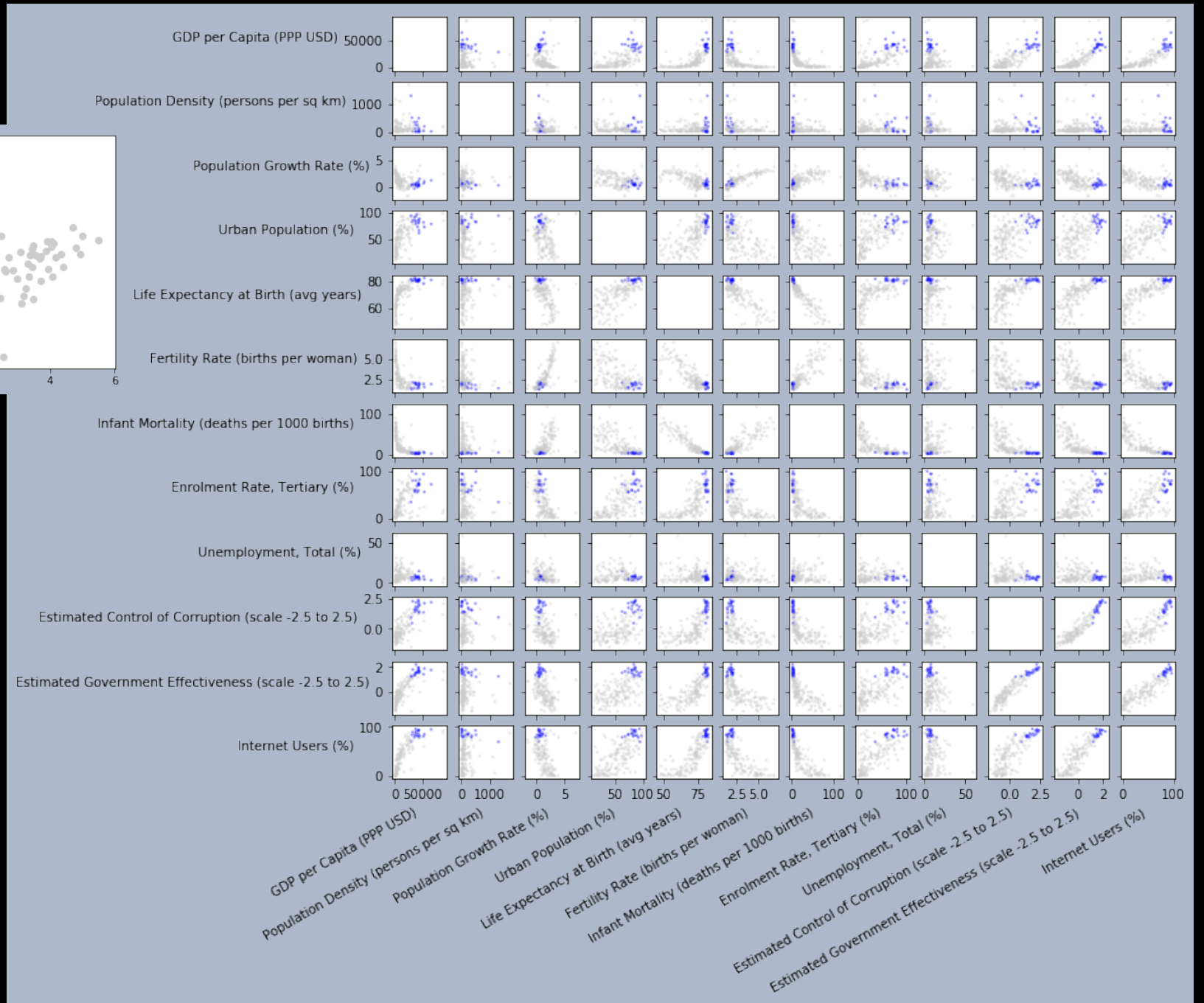
```
1   X = countries[features].values
2   pca = sklearn.decomposition.PCA()
3   pca_result = pca.fit_transform(X)
4
5   μ = pca.mean_
6   pred = μ + np.zeros_like(pca_result)
7   for k in range(L):      # L = number of PCA components to use
8      λk = pca_result[:,k]
9      δk = pca.components_[k]
10     pred = pred + λk.reshape((-1,1)) * δk.reshape((1,-1))
```

Run PCA, then plot the first two components $(\lambda_{1,i}, \lambda_{2,i})$ to get a nice visualization.



Nearby points in the PCA diagram should represent records with similar features.





Reasons for using dimension reduction

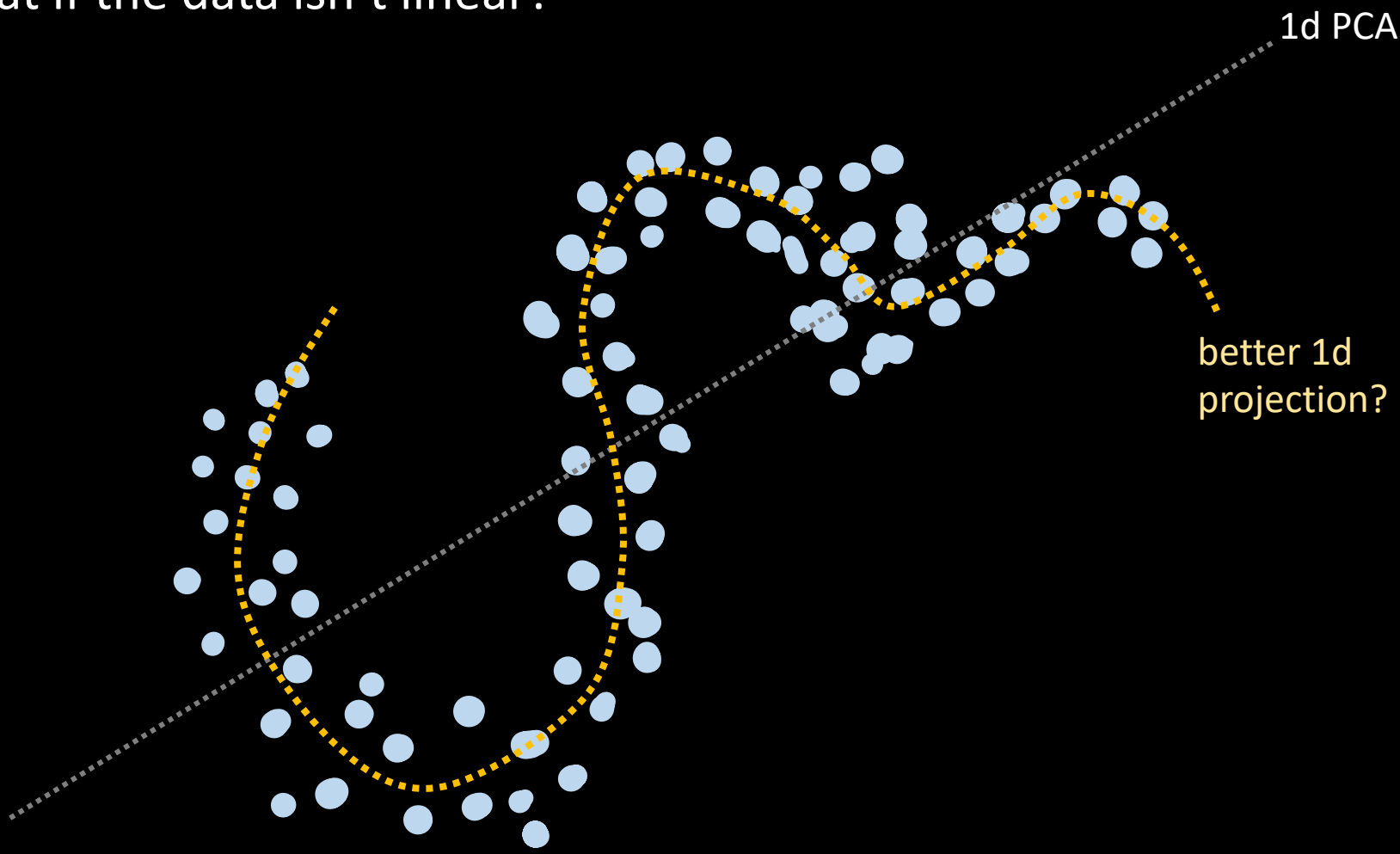
- To test generalization error for an ML system
  - Reduce the dataset to 1d. Take off 20% at one end, and use it as a validation set.
  - This leads to a validation set that is qualitatively different to the training set. If we want to measure how well our ML system generalizes, this is helpful.

- To reduce the number of features, to speed up an ML system
  - Suppose we have a categorical feature with 1000 levels, and we one-hot encode it. This gives 1000 features, which may be too much for our ML system to digest.
  - Use dimension reduction to make the features more palatable.

- To find clusters, which we hope will be useful for data exploration
  - However, if you know what you want to predict, it's a supervised learning problem. There's no point treating it as an unsupervised problem.

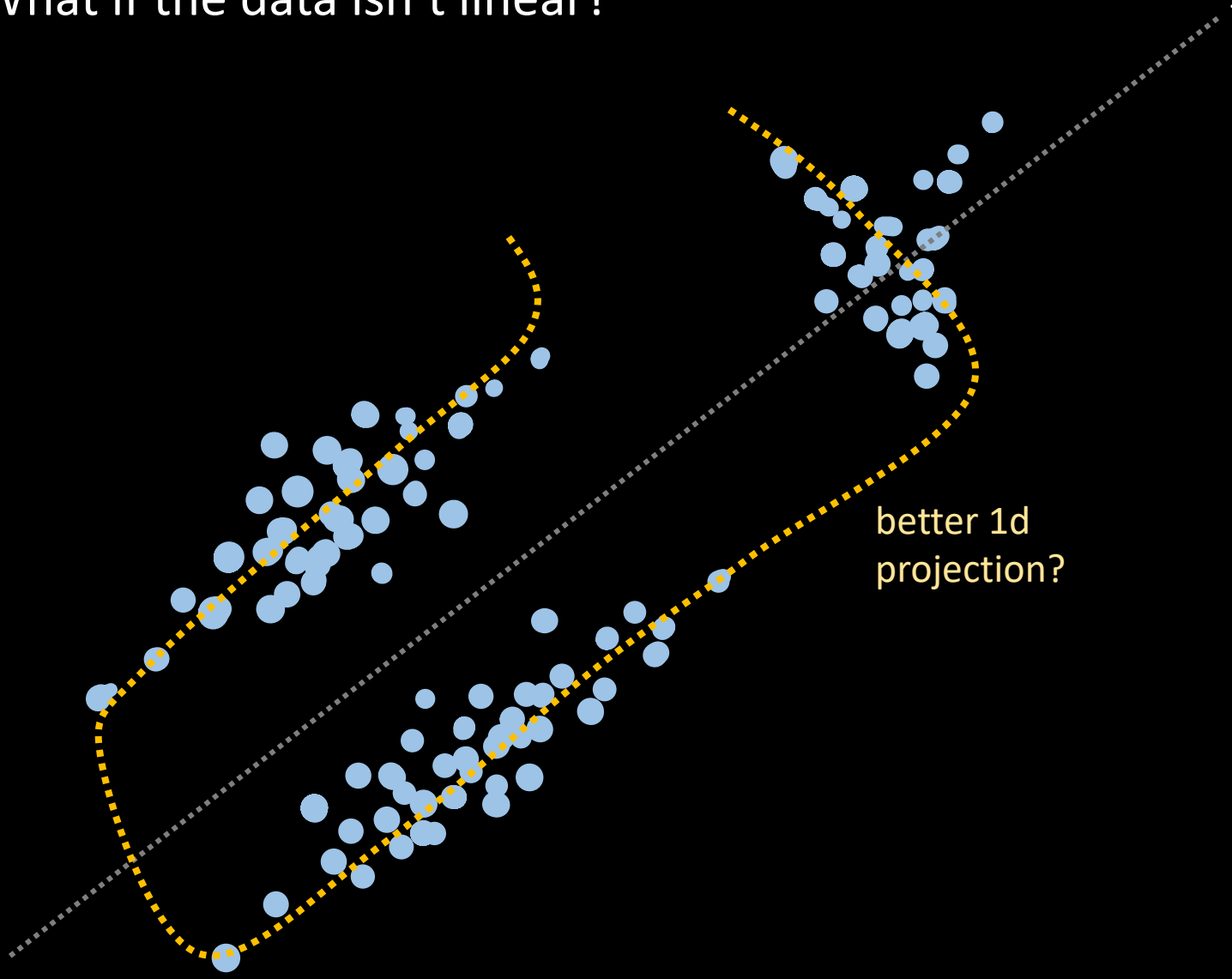
7. Dimension reduction / tSNE

What if the data isn’t linear?



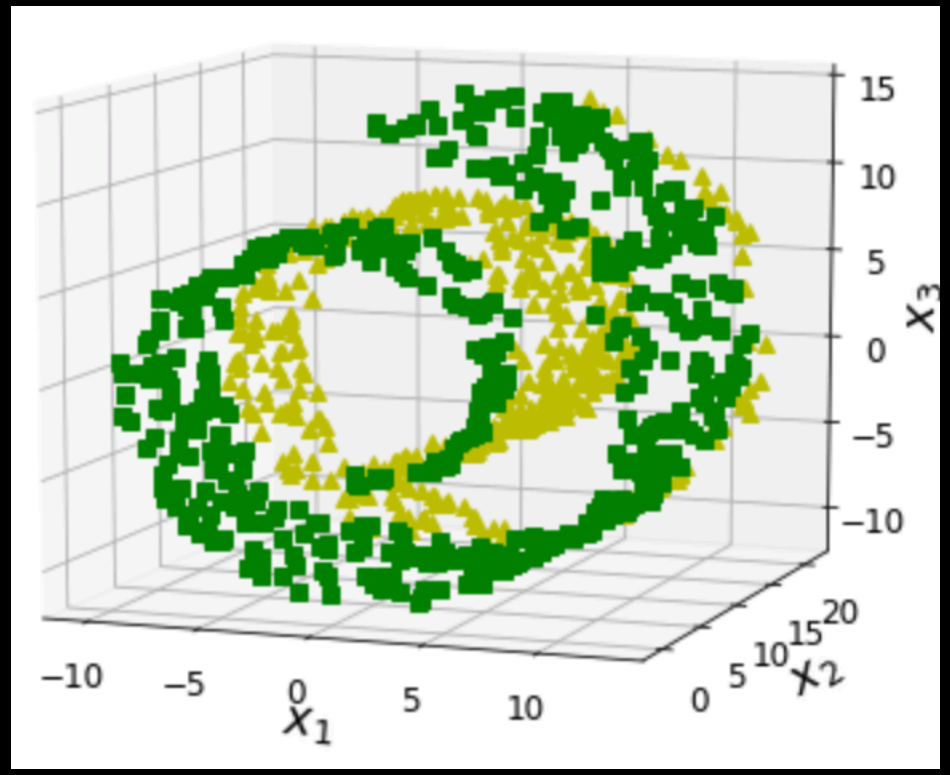
7. Dimension reduction / tSNE

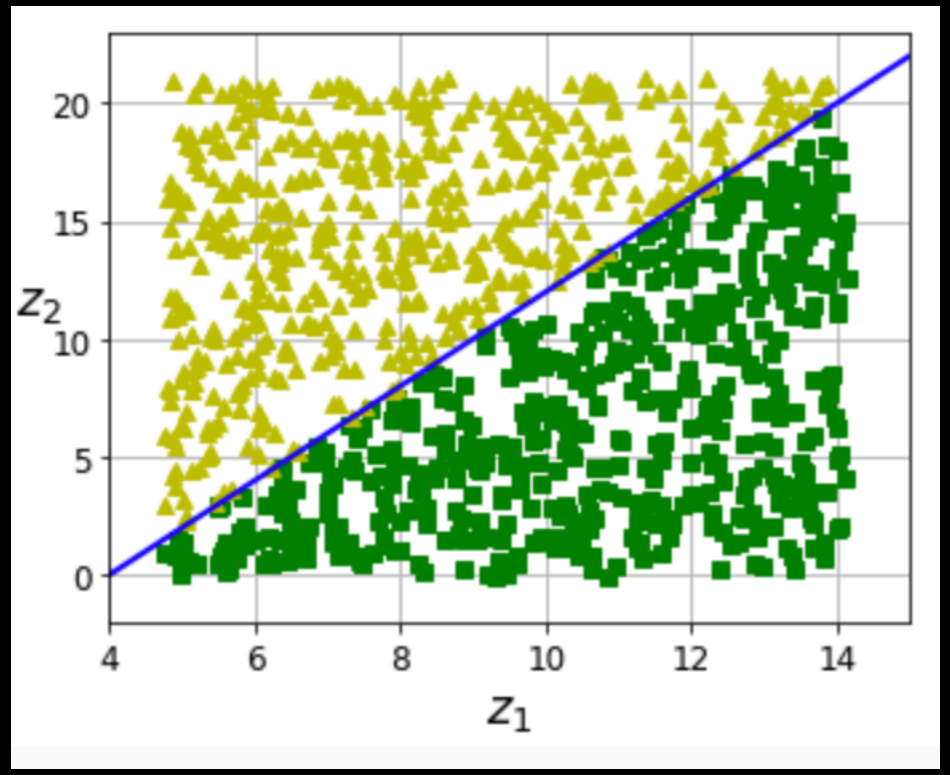
What if the data isn’t linear?



7. Dimension reduction / tSNE

Manifold Learning



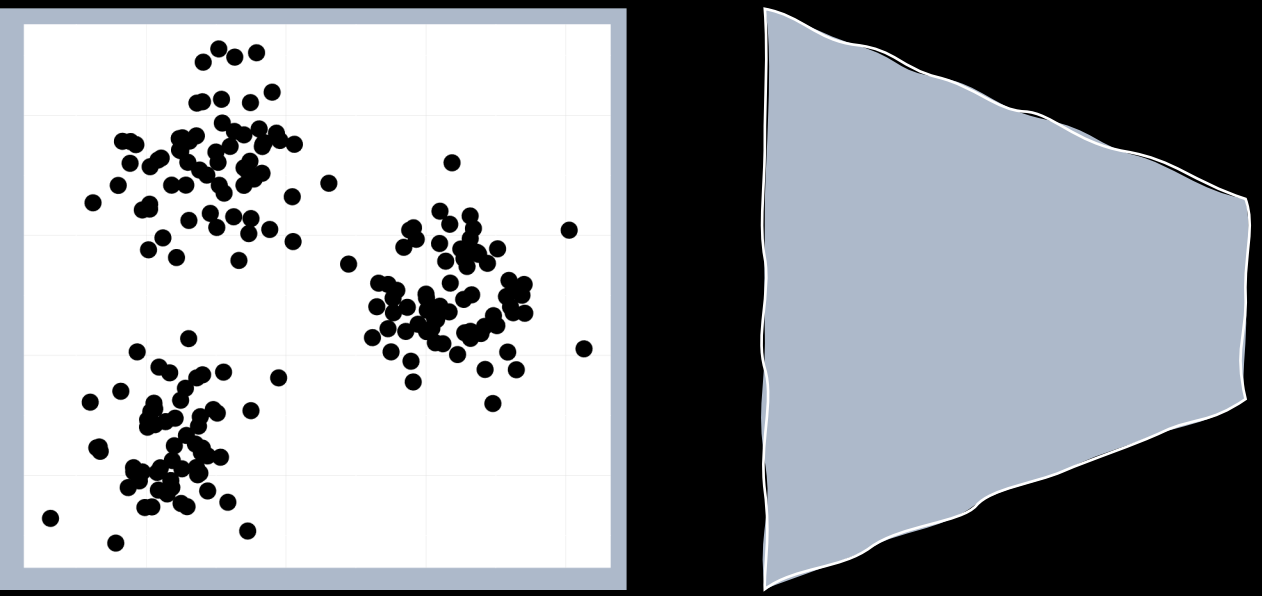


Geron (2017). *Hands-on Machine Learning with Scikit-Learn & TensorFlow*

7. Dimension reduction / tSNE

tSNE is another tool for dimension reduction. It (sometimes) gives sensible results for nonlinear data.

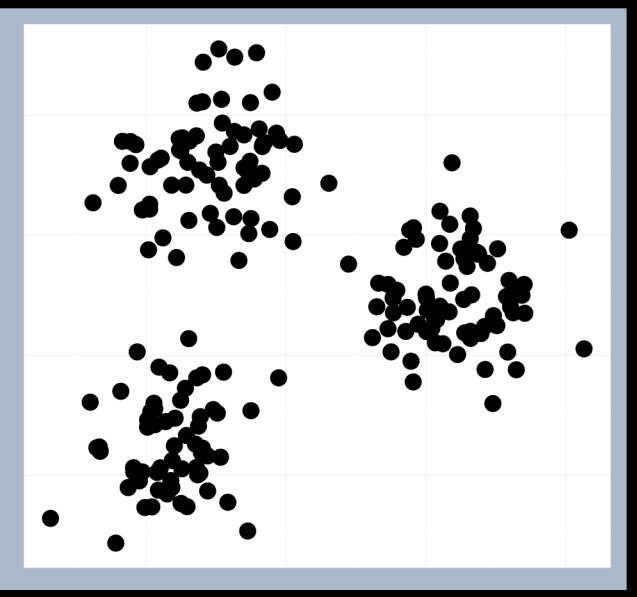
the data

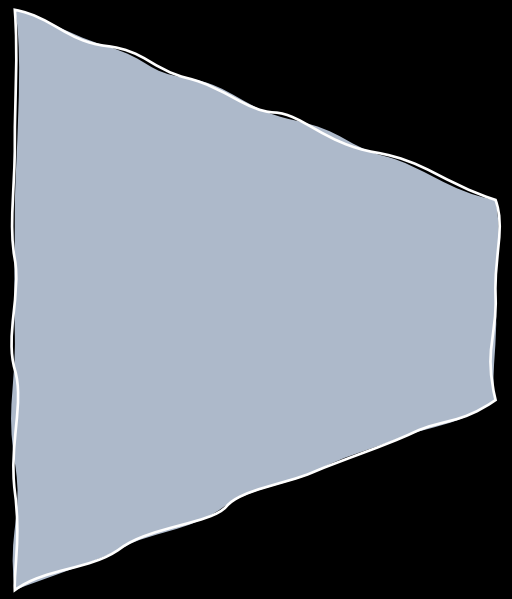


7. Dimension reduction / tSNE

tSNE is another tool for dimension reduction. It (sometimes) gives sensible results for nonlinear data.

the data



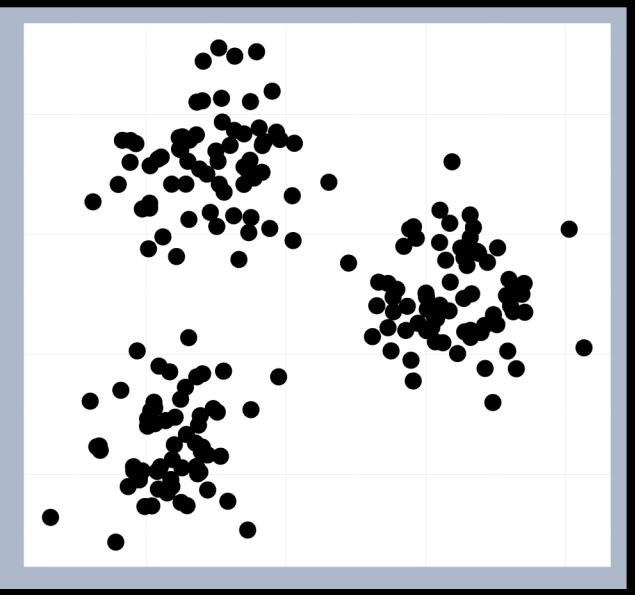


lower-dimensional
version of the dataset

7. Dimension reduction / tSNE

tSNE is another tool for dimension reduction. It (sometimes) gives sensible results for nonlinear data.

the data



tSNE

lower-dimensional
version of the dataset

Using tSNE

1. Each datapoint $\vec{x}_i$ is to be encoded to a K-dimensional latent variable λ_i
2. Each $\vec{x}_i$ has a certain set of nearest-P neighbours in the data space. Likewise each λ_i has a certain set of nearest-P neighbours in the latent space.

   We'd like these two sets of neighbours to be as similar as possible, across all datapoints. (The exact method isn't quite this, but it's close.)
3. We find the optimal encoding, using gradient descent.
4. The hyperparameter P is the *perplexity.* It affects the fit.

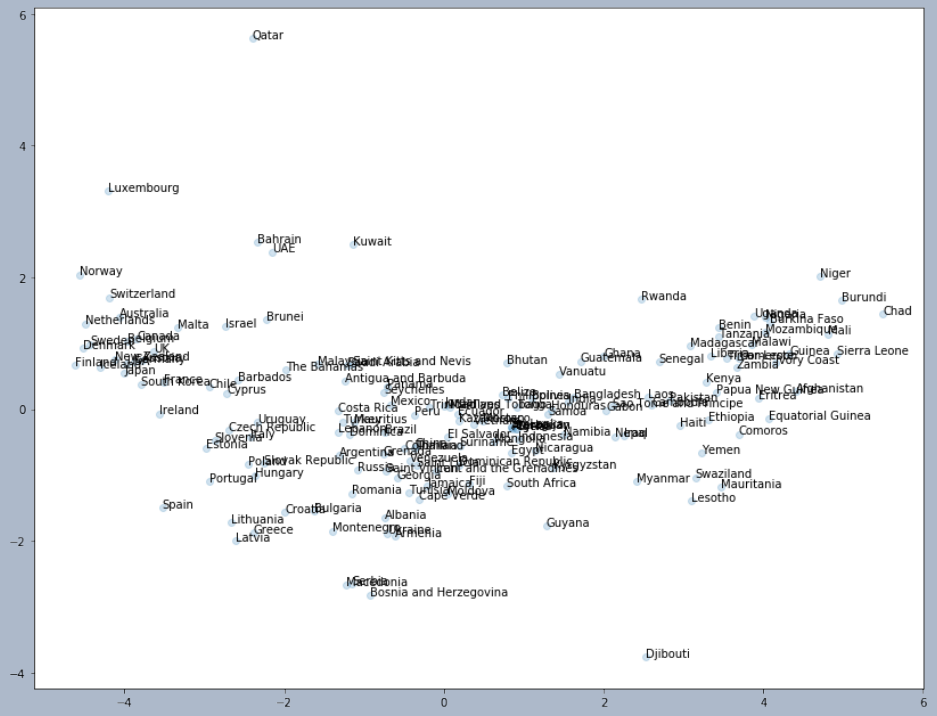
```
X = countries[features].values
# scale the columns, so they have the same variance
for i in range(len(features)):
    X[:,i] = X[:,i] / np.std(X[:,i])

# K = number of dimensions to reduce to
tsne = sklearn.manifold.TSNE(n_components=K)
tsne_results = tsne.fit_transform(X)

# K=2 gives us a nice easy 2d plot
p1,p2 = tsne_results[:,0], tsne_results[:,1]
plt.scatter(p1, p2, alpha=.2)
```

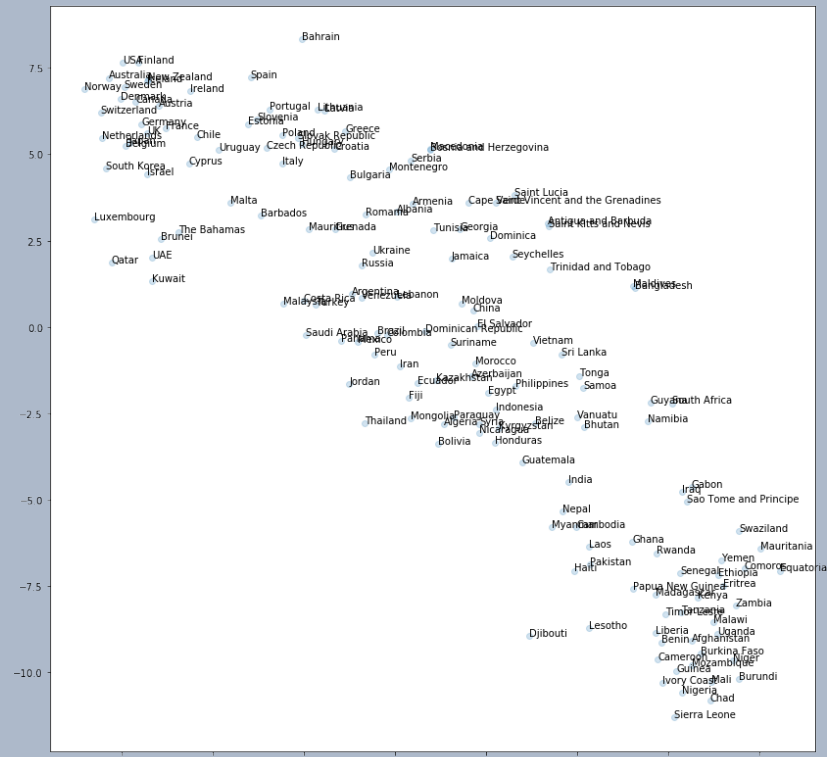
PCA looks for a linear coordinate system with which to represent the data.

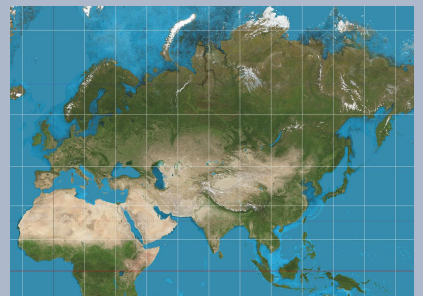
Each coordinate in a PCA plot measures “distance along a PCA axis” — i.e. each coordinate is an interval scale.



tSNE tries to embed the data so as to preserve distances between datapoints.

The coordinates in a tSNE plot don’t mean anything per se, it’s only distances that are important — distance is a ratio scale.



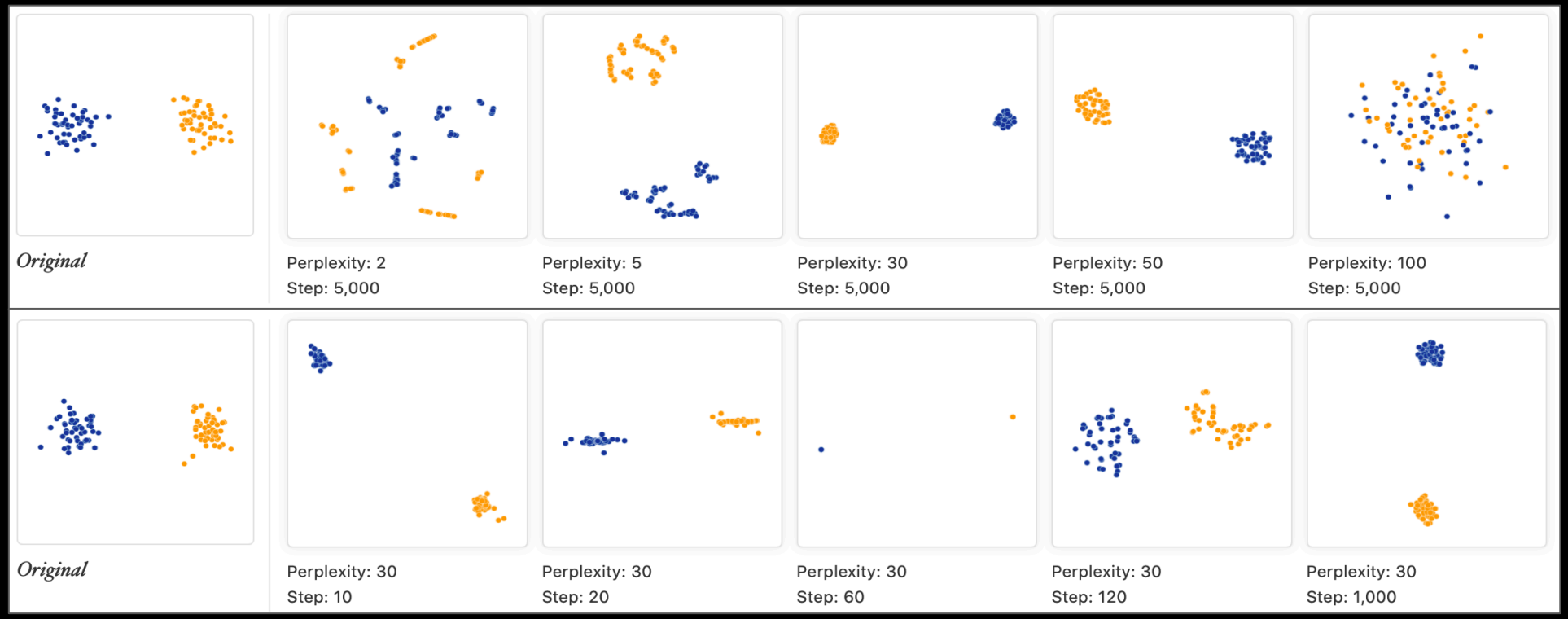


In the Mercator projection, you can read off longitude from the x-axis and latitude from the y-axis

There’s no way to embed the entire globe in 2d and preserve distances, but there are projections that do a reasonable job of preserving distances over smaller areas

Using tSNE effectively

1. tSNE is very effective for visualisation and finding structure in high-dimensional data.
2. It is often tricky to interpret and can be misleading.
3. Study its behaviour on simple cases and in steps.



Wattenberg et al. (2016). *How to Use t-SNE Effectively*
Visualisations: https://distill.pub/2016/misread-tsne/

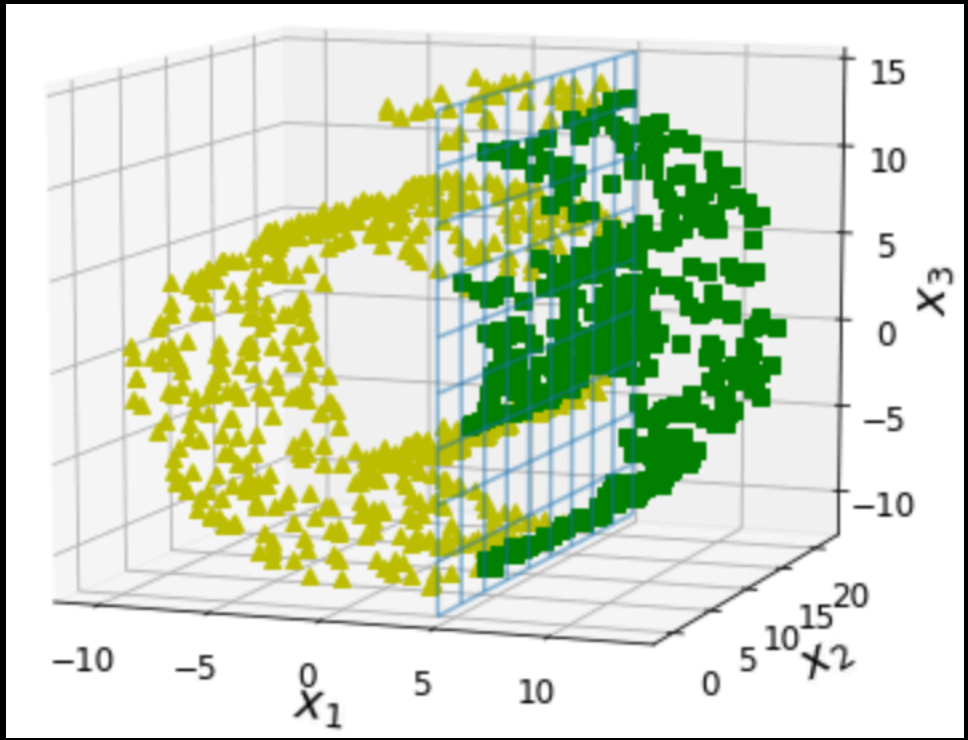
Other dimensionality reduction techniques

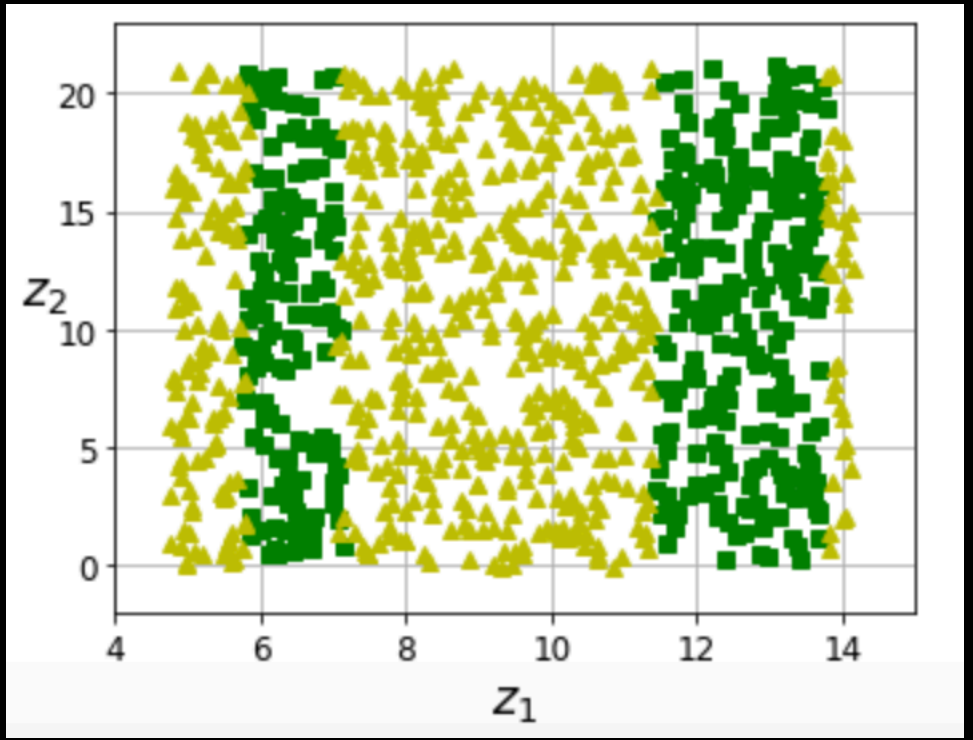
- Incremental PCA
- Randomized PCA
- Kernel PCA
- Locally Linear Embedding (LLE)
- Multidimensional Scaling
- Isomap
- Linear Discriminant Analysis (LDA)

  - Note: Many algorithms (e.g., t-SNE) are computationally expensive. It is possible to chain several of them: e.g., run PCA to quickly get rid of a large number of less useful dimensions, then apply another, more powerful, computationally expensive and slower algorithm. The results will be on a par with the latter algorithm, but achieved in a fraction of the time.

https://scikit-learn.org/stable/modules/manifold.html

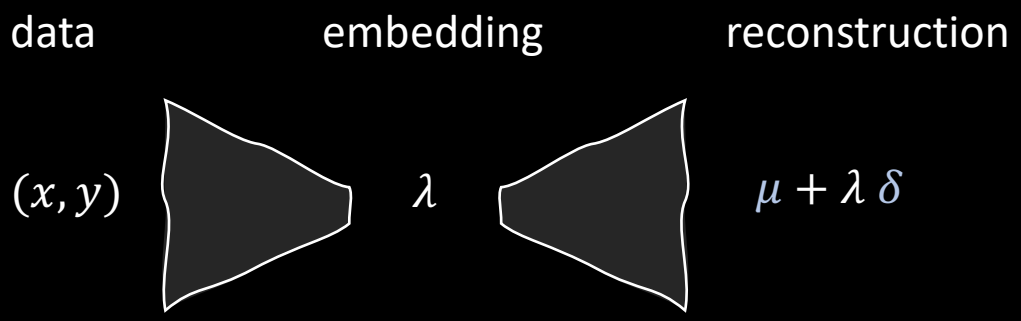
When Manifold Learning doesn't help





Geron (2017). *Hands-on Machine Learning with Scikit-Learn & TensorFlow*

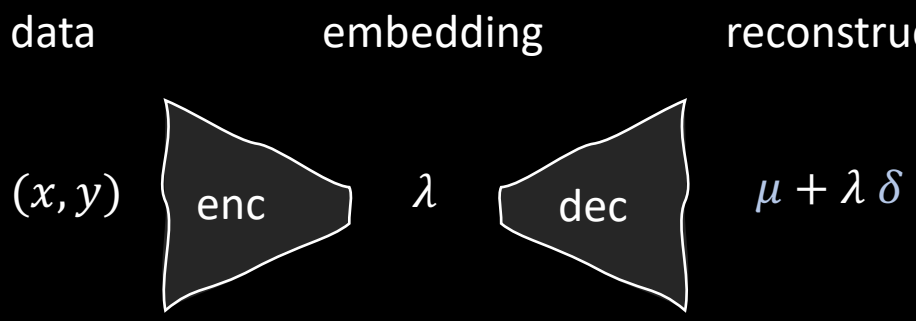
8. Self-supervised learning and embedding



- Don’t think of PCA and tSNE as simply
  “run an algorithm on a dataset, get a dimension-reduced version”

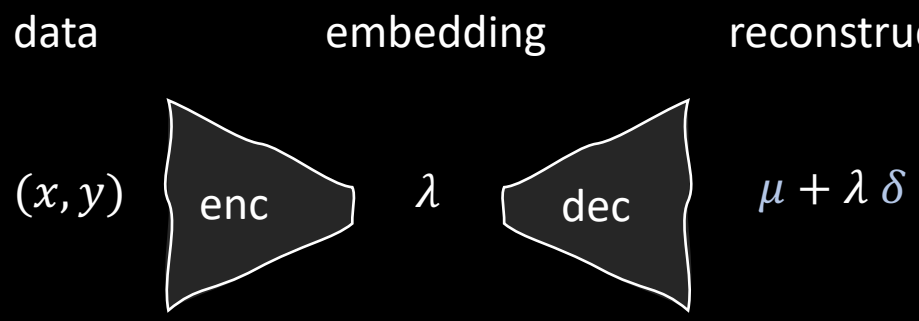
- Think of them as learning an *embedding* from the dataset;
  this embedding can then be applied to *new unseen data*

8. Self-supervised learning and embedding



- PCA learns a subspace (specified by μ and δ)
- The subspace entails an *encoder* or *embedding* function $(x, y) \mapsto \lambda$
- and also a *decoder* or *reconstruction* function $\lambda \mapsto \mu + \lambda\,\delta$

8. Self-supervised learning and embedding



- PCA learns a subspace (specified by μ and δ)
- The subspace entails an *encoder* or *embedding* function
  $(x, y) \mapsto \lambda$
  $\in \mathbb{R}^N$ $\in \mathbb{R}^k$
- and also a *decoder* or *reconstruction* function
  $\lambda \mapsto \mu + \lambda\,\delta$
  $\in \mathbb{R}^k$ $\in \mathbb{R}^N$

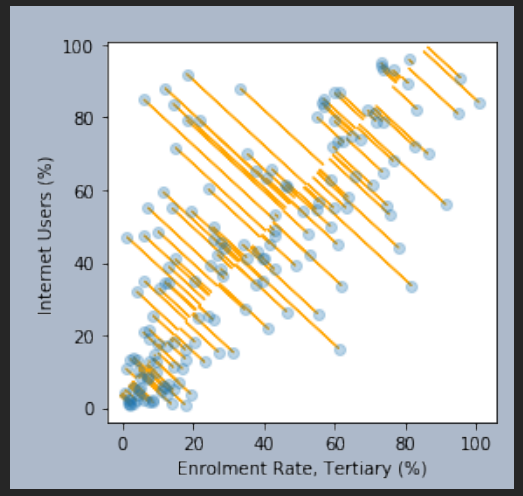
TECHNICAL NOTE
How do we compute λ from a datapoint $\vec{x}$?
PCA produces an orthonormal basis $(\vec{\delta}_1, \ldots, \vec{\delta}_K)$, and so we can compute the components of the embedding very easily:

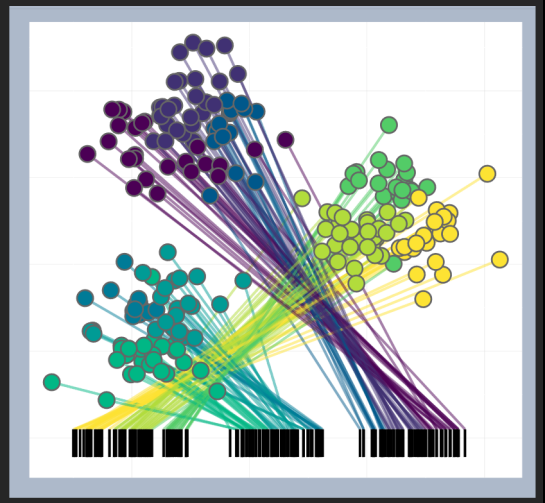
$$\lambda_k = (\vec{x} - \mu) \cdot \vec{\delta}_k$$

PCA embedding

tSNE embedding

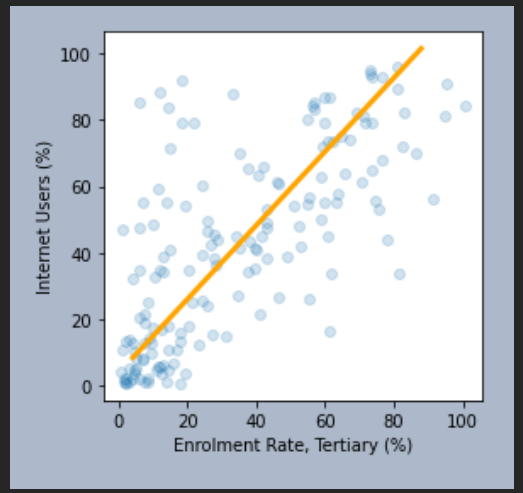
encoding / embedding
(from 2d data space to
1d embedding)

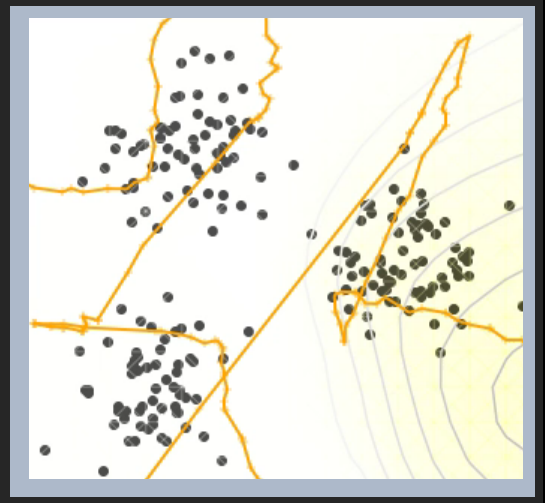




embedded values

decoding
(from 1d
back to 2d data space)





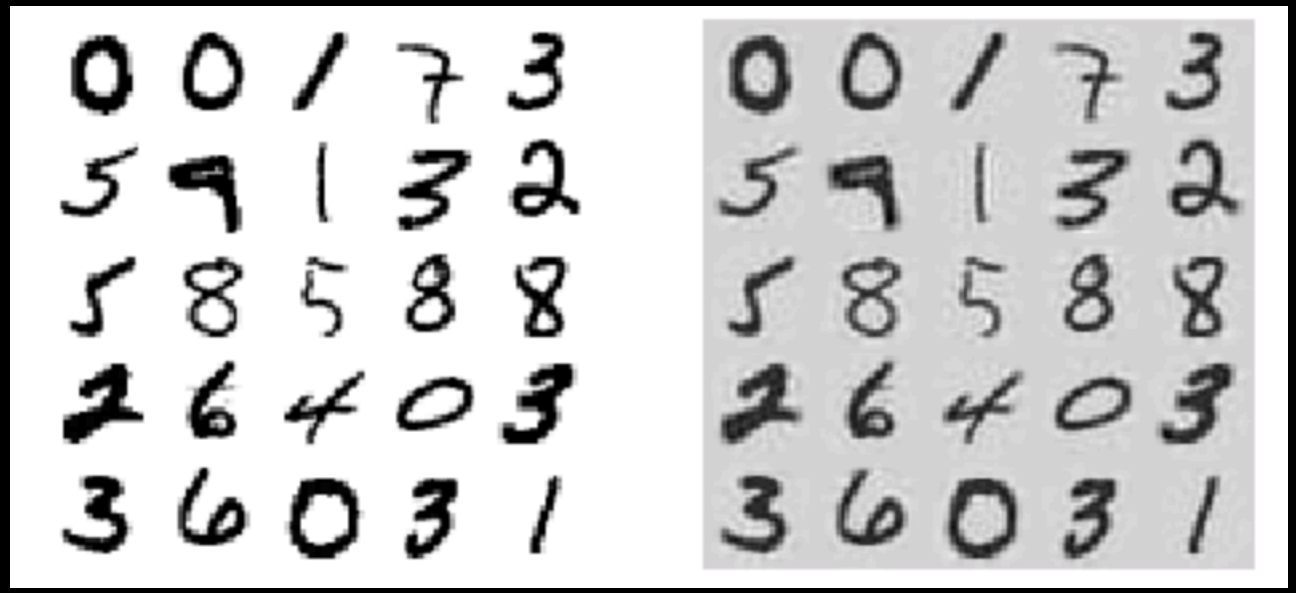
I don’t know any library that provides a tSNE encoder. It’s fairly easy to implement it ourselves, if we look into the tSNE maths.

I don’t know of any publication that reports the decoder for tSNE.

It takes some mathematical skill to define the decoder—it doesn’t produce a *value* in the dataspace, it produces a *distribution* over the dataspace.

The decoder is also tricky to implement. Better to use autoencoder neural networks.

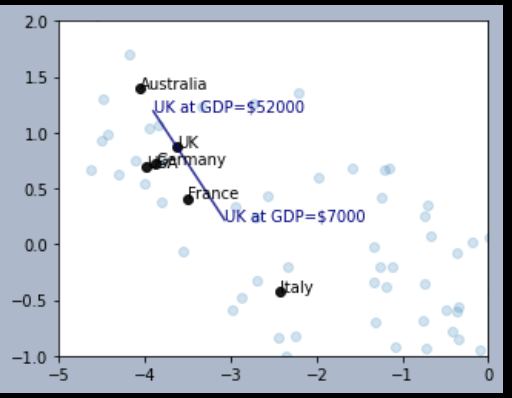
PCA and inverse transform



This is a compressed version of the MNIST data that preserves 95% variance

```
pca = PCA(n_components = 154)
X_reduced = pca.fit_transform(X_train)
X_recovered = pca.inverse_transform(X_reduced)
```

Here are some questions we can answer with an encoder/decoder pair (shown here for PCA)



"If the UK's GDP were different, all else being equal, what country would it be most like?"

- Let UK_g be like the UK but with GDP=g, and plot $\text{enc}(UK_g)$ as g varies
- For this dataset, "all else being equal" is daft. But for other datasets it might be a useful question.

```
uk = countries.loc[countries['Country Name']=='UK']
uk – dec(enc(uk))
```

GDP: -1204
Unemployment%: 2.4
Life expectancy: -0.78
Internet users%: 4.95

"In what respects does the UK stand out, compared to what we'd expect from other countries?"

```
usa = countries.loc[countries['Country Name']=='USA']
china = countries.loc[countries['Country Name']=='China']
dec(0.5*enc(usa) + 0.5*enc(china))
```

"What would a country half way between the USA and China look like?"

- It's pointless to use PCA enc+dec to answer this, because PCA is linear. But with a nonlinear embedding like tSNE it can be informative.

Reasons for using embeddings

- To make predictions when you have lots of unlabelled data and only a little labelled data, e.g. millions of images, only a few of them labelled. This is called *semi-supervised learning*.
  - Train an embedding using all the data. Hopefully we'll learn a useful low-dimensional embedding, that picks out the important features.
  - Train a predictor that predicts the label from the embedding. If the features we found are useful, then the predictor won't need so much training data.

- To be able to share the learning between many different tasks. This is called *transfer learning.*
  - Widely used in NLP (see word embeddings) and computer vision

9. Content scales

Nominal: no comparison is meaningful

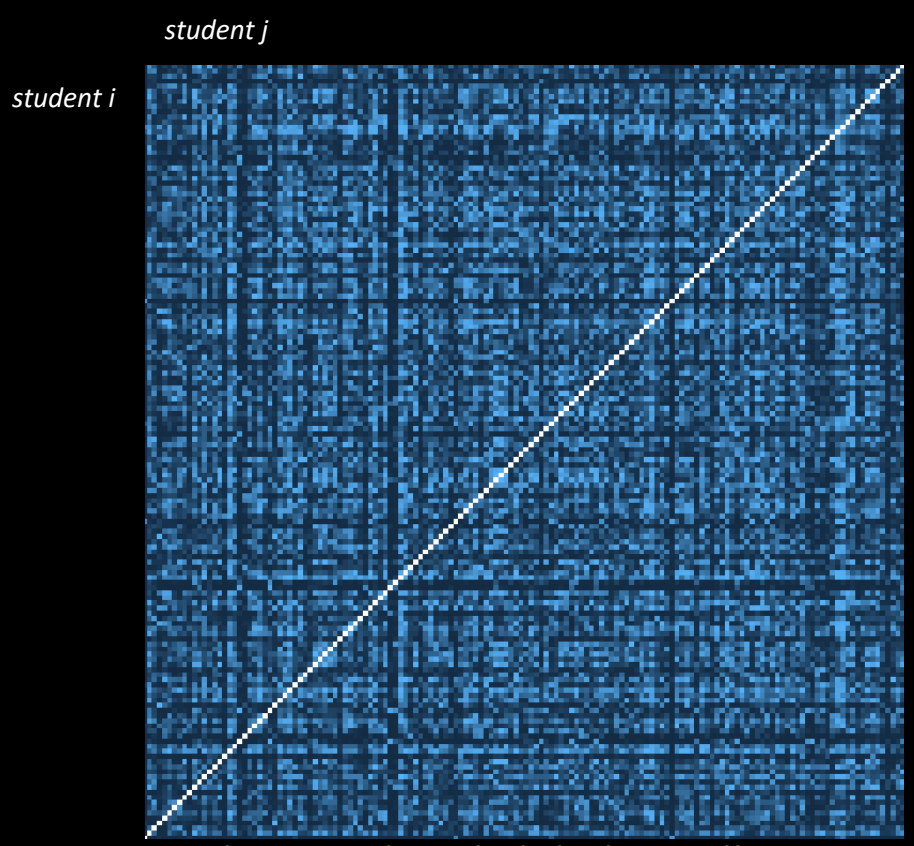
This is dumb.

How can I say “no comparison is meaningful” — and at the same time render onto a y scale?

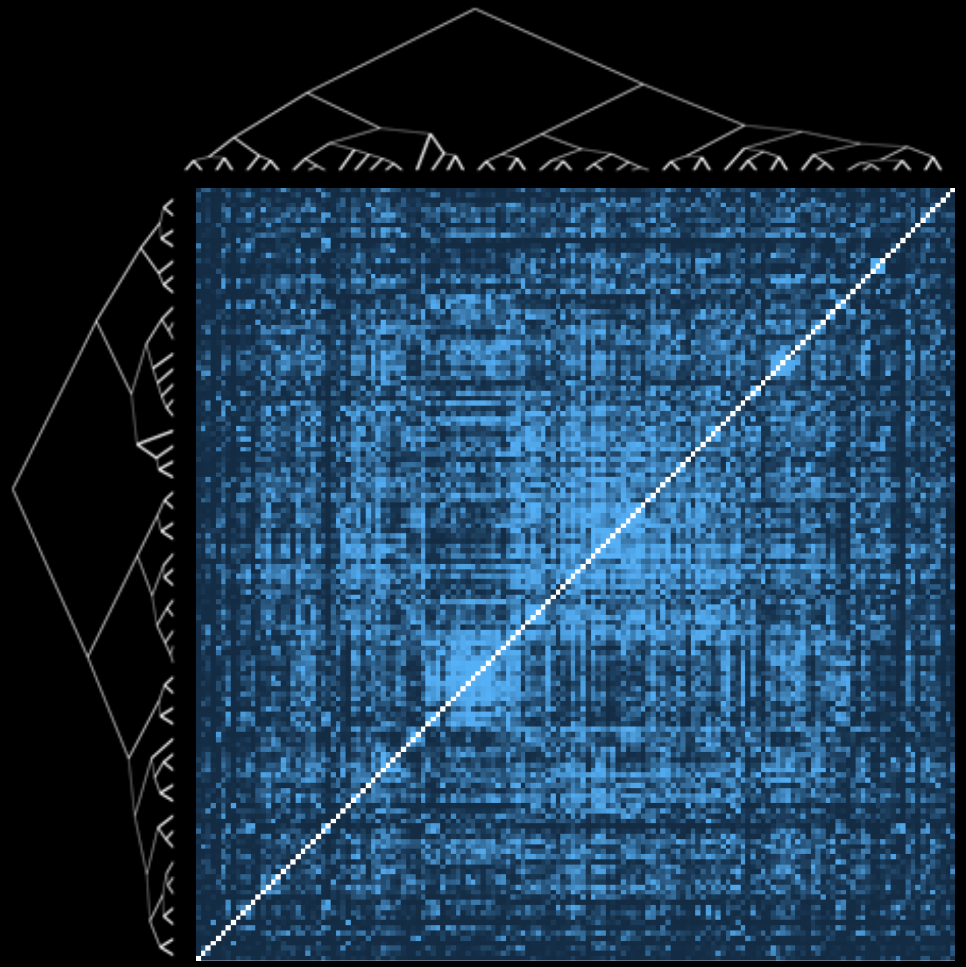
Actually, what we've plotted is an embedding from countries onto $\mathbb{R}$. We've chosen an embedding such that countries near to each other alphabetically get embedded into nearby locations on the scale.

Content scales

- We've seen how to create a content scale, i.e. an embedding, for high-dimensional numerical data.
- When we have nominal or ordinal data, we should pick an embedding that's a useful reflection of the content we're interested in.



Students ordered alphabetically



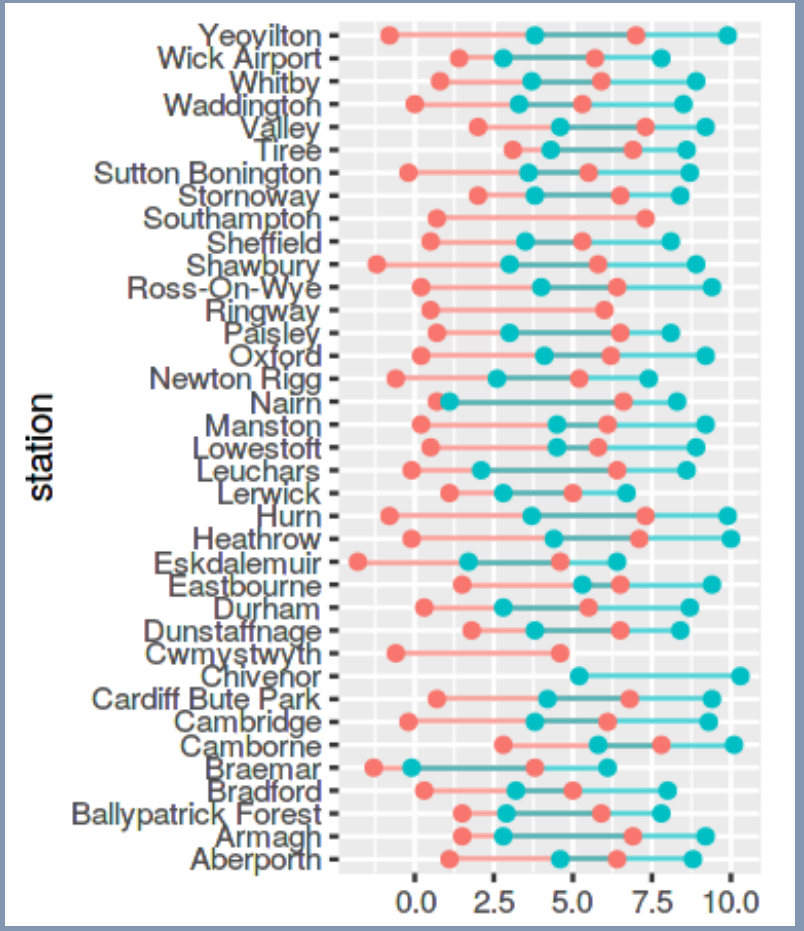
Students ordered by clustering
—reveals cliques of cheaters

Similarity between student coursework submissions

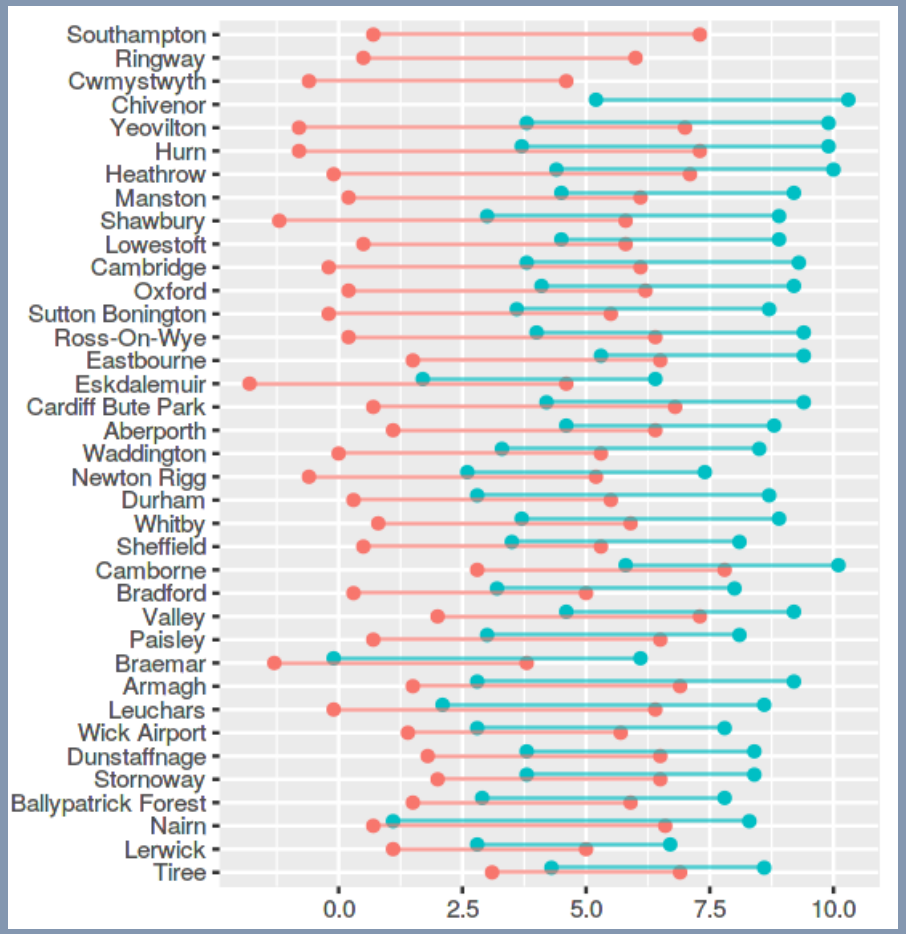
high similarity between work by *i* and *j*

low similarity

Content scales



A good scale, if the content I'm interested in is the name of the station (for easy lookup)



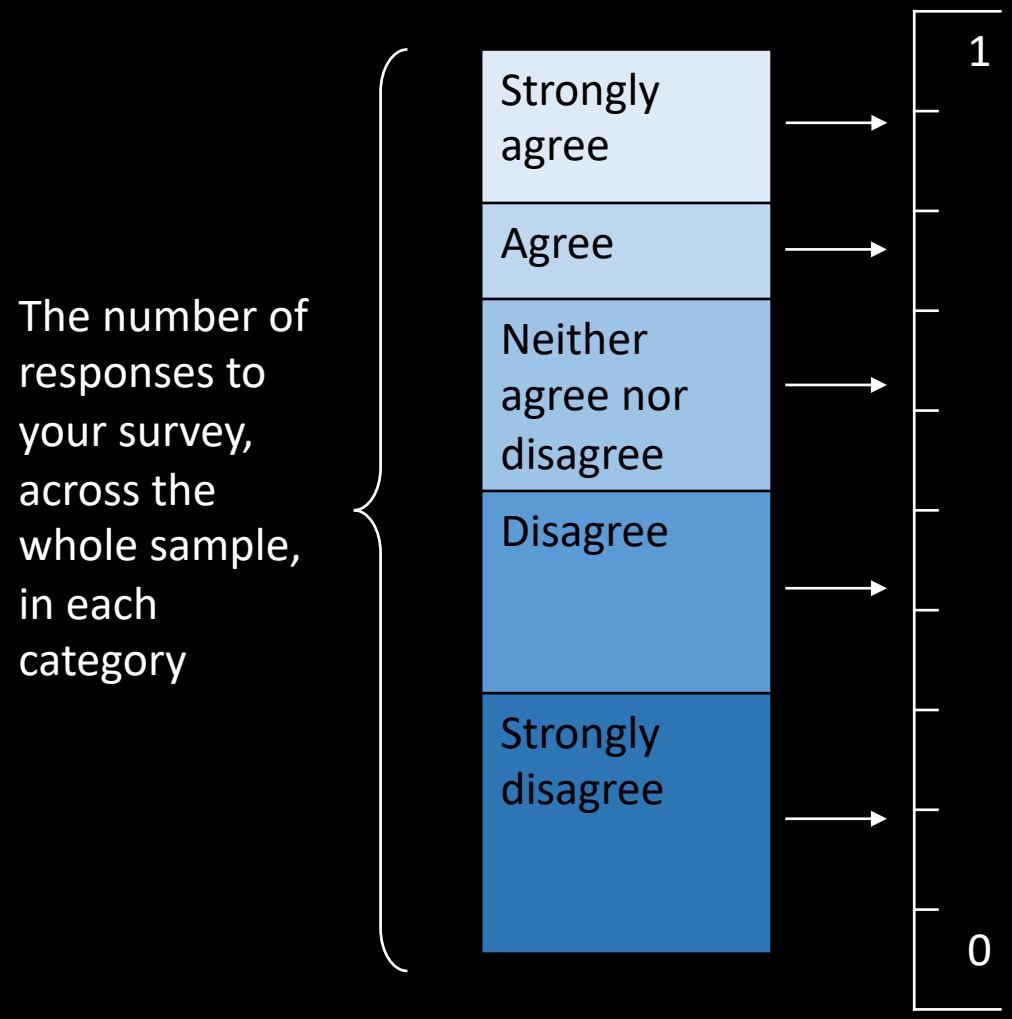
min and max February temperature at stations across the UK

1981
2017

A good scale, if the content I'm interested in is temperature increase from 1981 to 2017

Embedding an ordinal scale, such as Likert items

Here's a handy trick for getting useful numbers out of ordinal data. It's not deep and rigorous, but it is helpful.

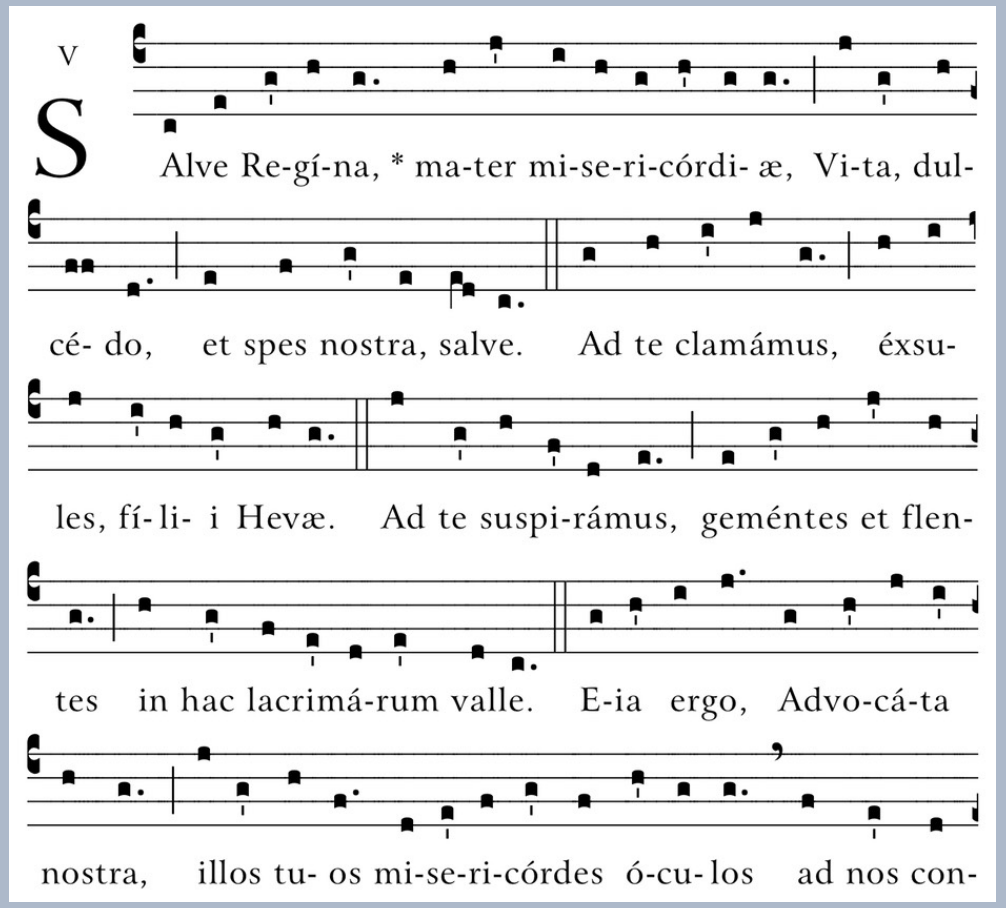


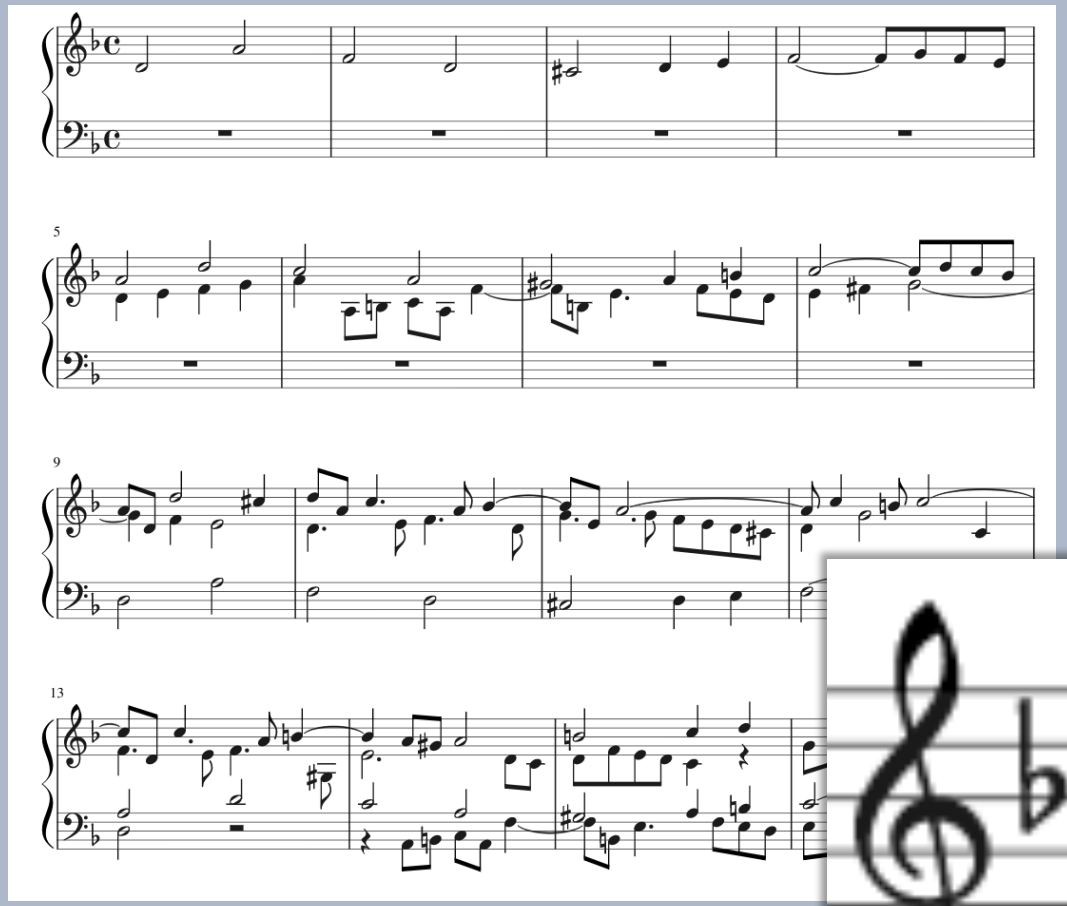
Map each ordinal answer to a number in the range $[0,1]$, based on its quantile within the whole sample.

This trick lets you compute average responses, run linear regression, etc.

Content versus metrical scales

- In a remarkable period from 1250–1350, we started to use metrical scales, systematically. This changed how we see the world.
  See *The measure of reality: quantification and western society, 1250–1600,* by Alfred W. Crosby, 1997.
- Computer power is letting us go back to useful content scales / embeddings.





Visualization

LECTURE ONE — chart literacy

1. anatomy of a plot
2. scale theory
3. scale perception
4. making comparisons
5. atomic plots

LECTURE TWO — embedding

6. unsupervised learning
7. dimension reduction
8. self-supervised learning
9. content scales

EPILOGUE next steps

the "grammar" of plots

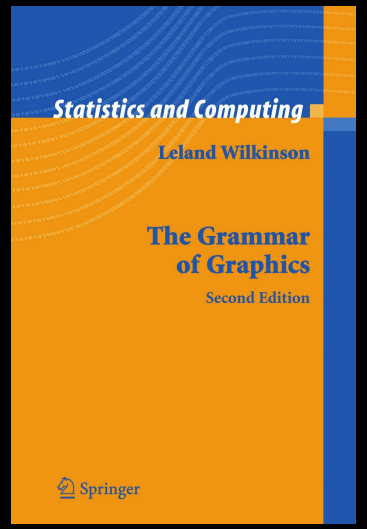
plots for data exploration

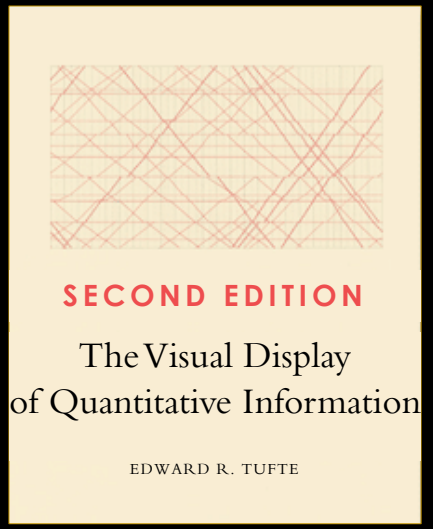
We have studied the grammar of graphics.

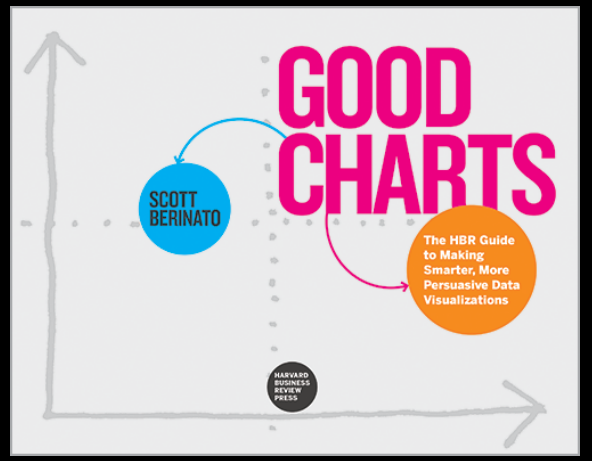
Grammar doesn't tell you how to create great charts. But it does give you tools to think systematically about your charts.

You also need • style • the skill to tell a story • good software libraries.

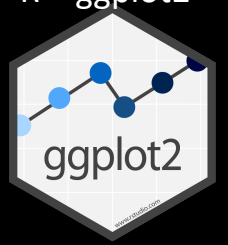
rhetoric = grammar + style + reason / arrangement







R + ggplot2



Javascript + D3

Vega Lite

and many more libraries

matplotlib

Practical 6

- Data and code for Practical 6 can be found on: Github (https://github.com/ekochmar/cl-datasci-pnp-2021/tree/main/DSPNP_practical6)
- It's mainly about PCA and tSNE
- Practical ('ticking') session over Zoom at the time allocated by your demonstrator
- At the practical, be prepared to discuss the task and answer the questions about the code to get a 'pass'
- Upload your solutions (Jupyter notebook or Python code) to Moodle by the deadline (Tuesday 1 December, 4pm)