

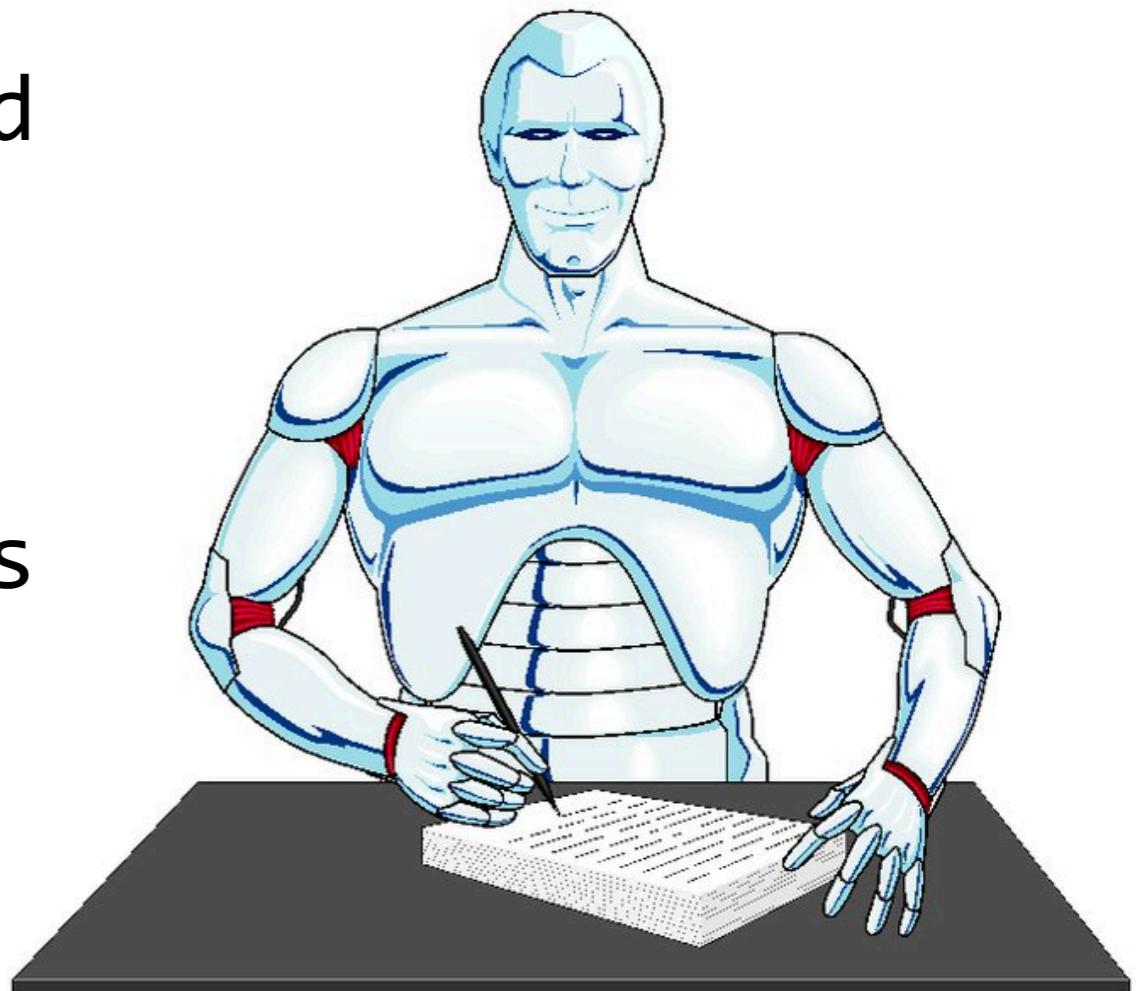
PRINCIPAL COMPONENTS ANALYSIS

12.10.2018



FINAL

- * the final will be posted today (i.e. before midnight)
- * it will be due (and this is a HARD deadline) on Wednesday, December 19 at 10:59 AM



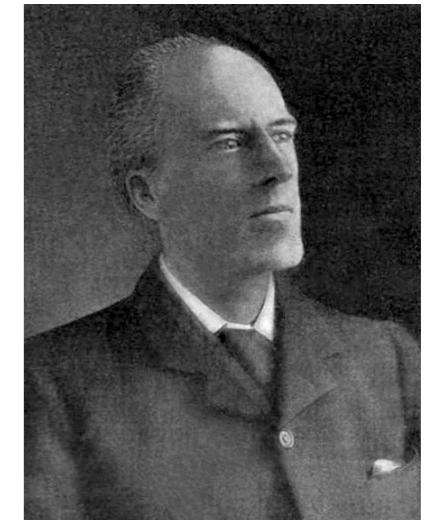
FINAL

- * the final will be SELF-TIMED (honor system!) for 4 hours
 - * your time starts when you first look at it, and you should stop working on it 4 hours later
- * it is OPEN BOOK, OPEN DOCUMENTATION, & OPEN INTERNET
 - * but don't discuss it with anyone else until you have both finished it

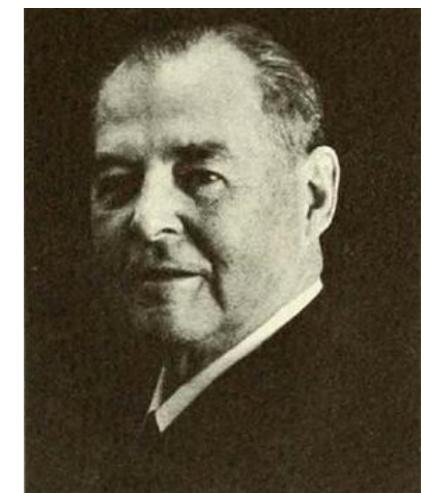
FINAL

- * the final will cover all the topics we've touched on in class
- * it will be EASIER THAN THE HOMEWORKS
 - * i.e. the vast majority of you should finish in 4 hours
- * good luck :)

PCA



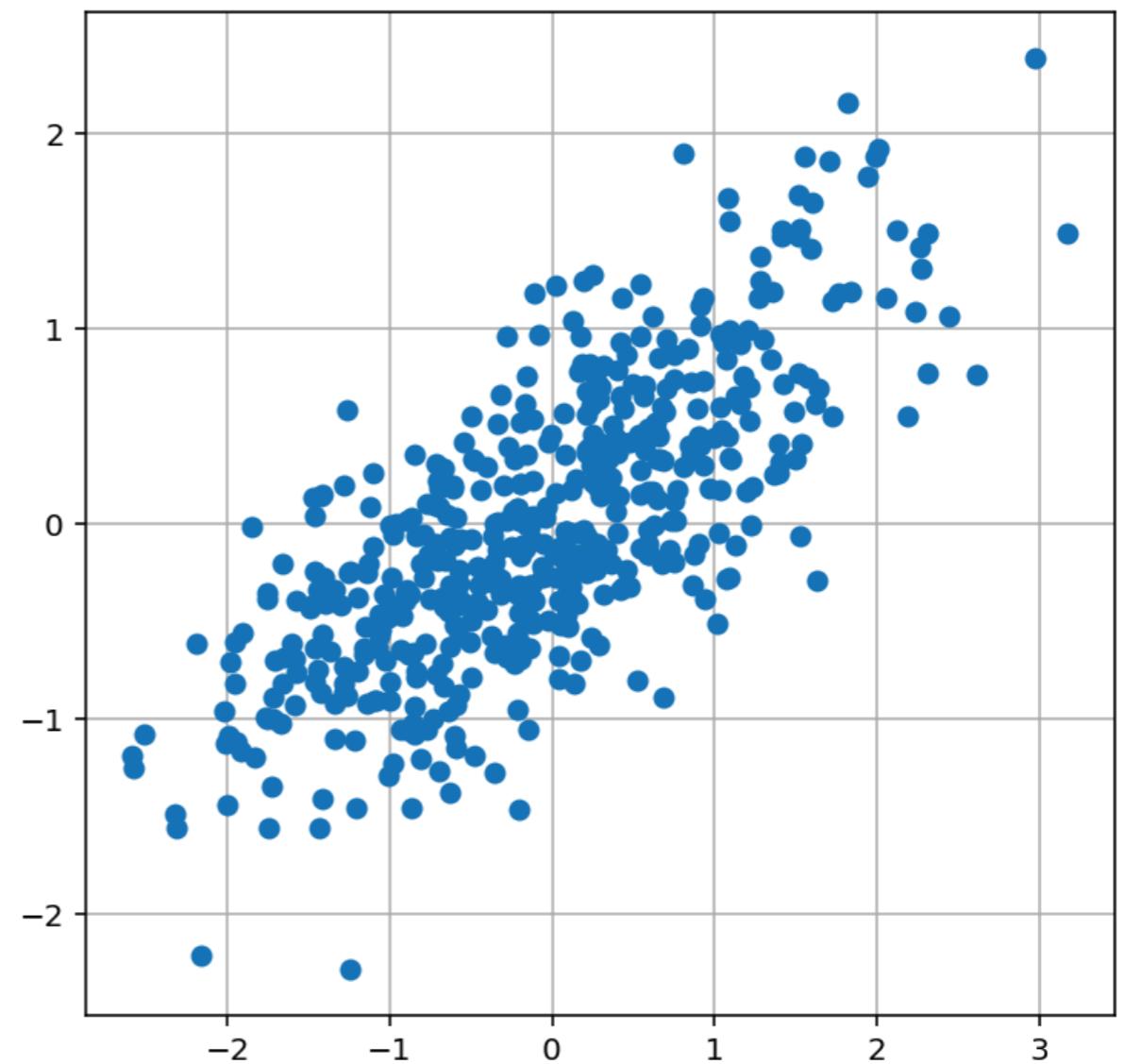
- * **Principal Components Analysis** is an *unsupervised* method for finding structure in datasets
- * It was developed (independently!) by Karl Pearson and Harold Hotelling



Harold
Hotelling

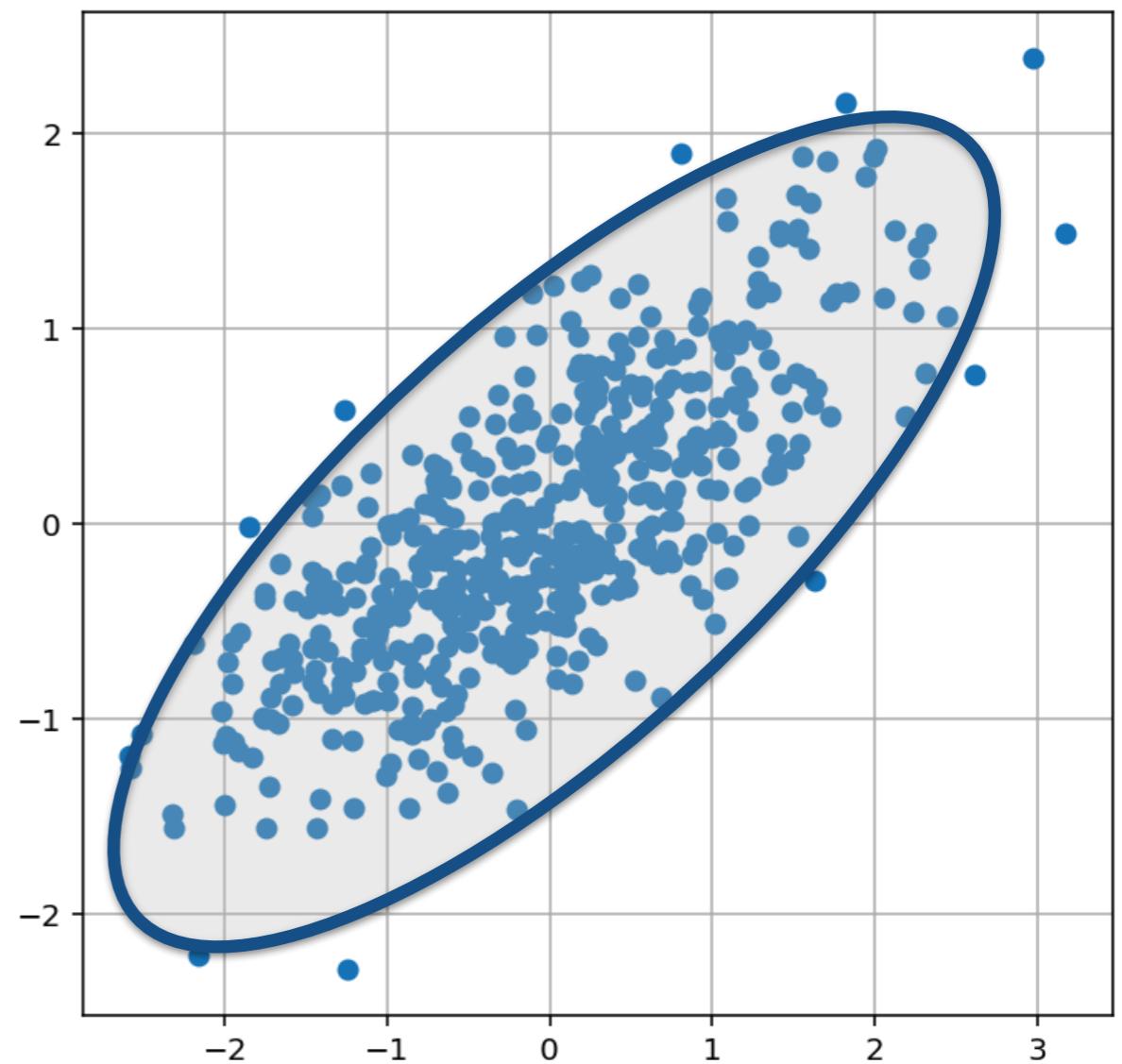
PCA

- * The typical explanation of PCA:
- * PCA “fits an ellipse” to a cloud of datapoints



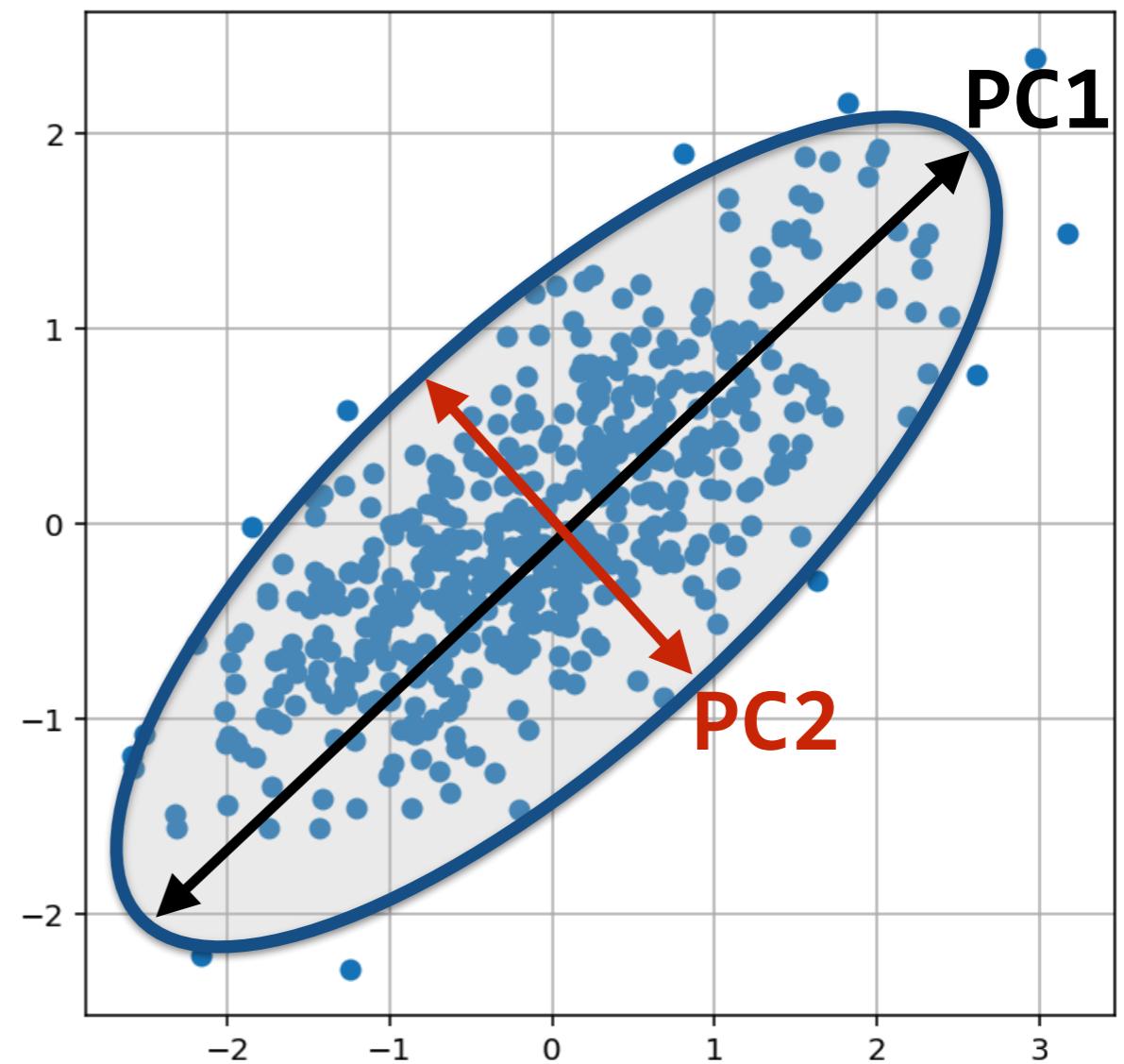
PCA

- * The typical explanation of PCA:
- * PCA “fits an ellipse” to a cloud of datapoints



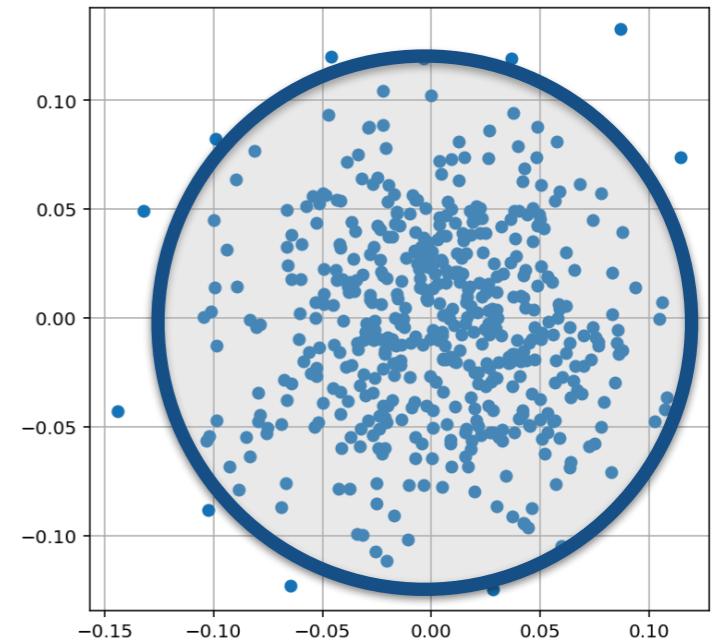
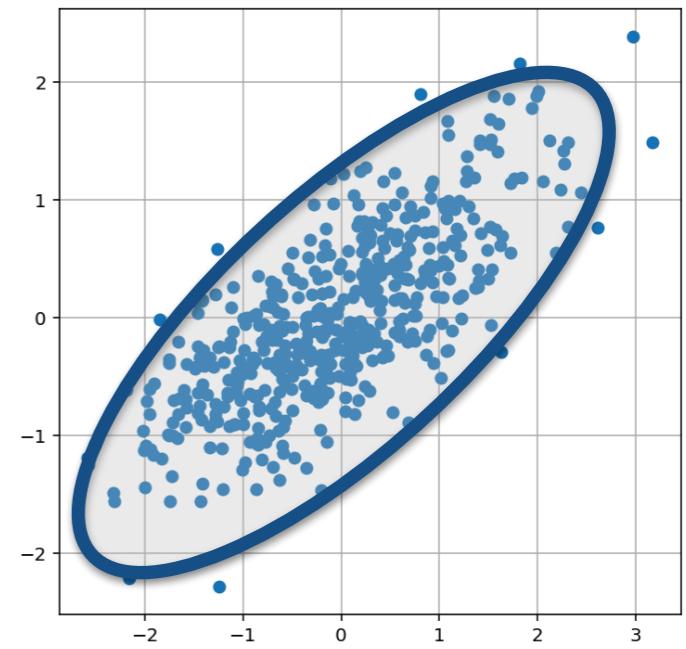
PCA

- * The axes of the ellipse are the “principal components”



PCA

- * It also gives you a way to *transform* the data so that the ellipse becomes a perfect circle

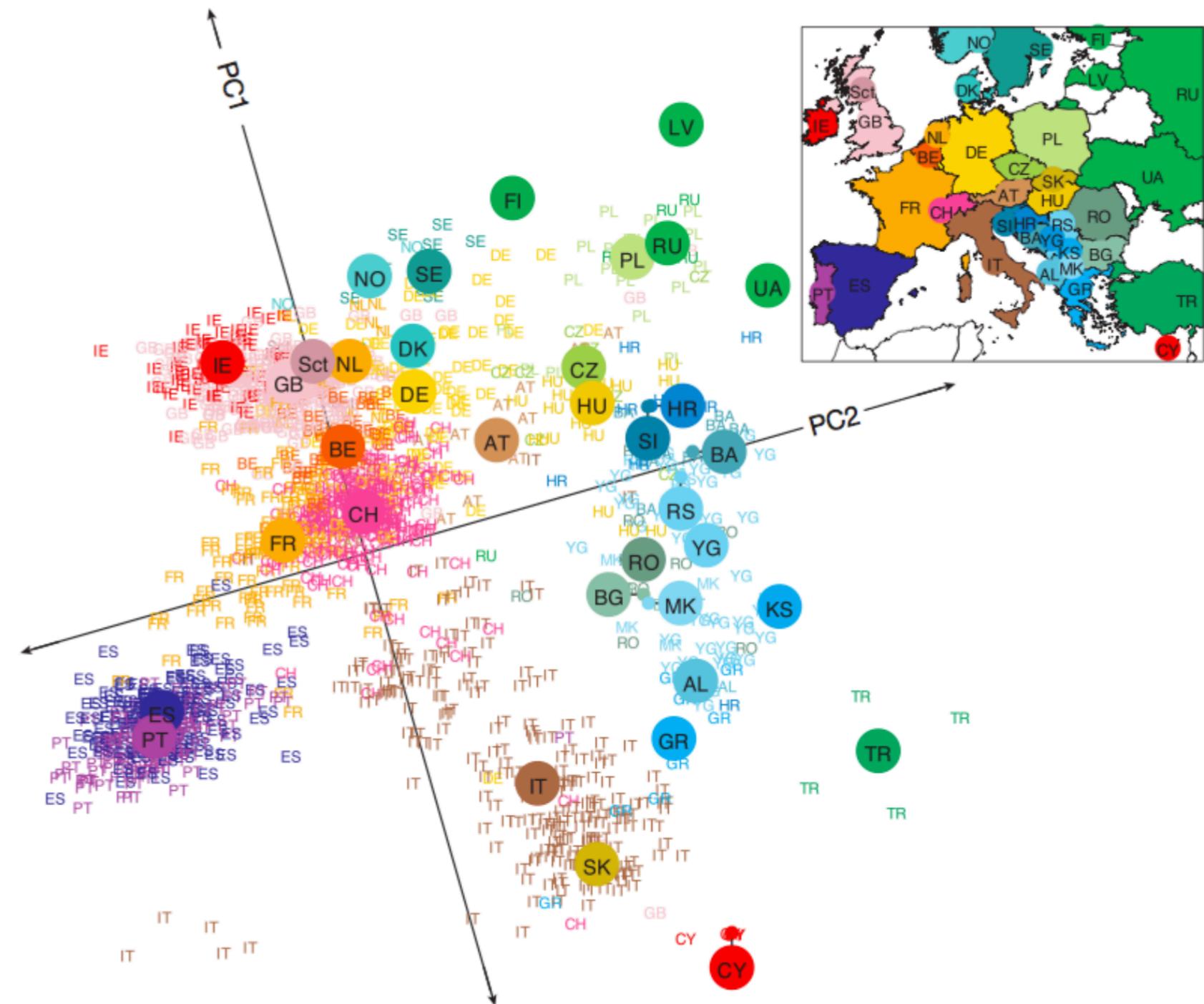


PCA

- * But the key use case of PCA is *dimensionality reduction*
 - * If your data has lots of dimensions, PCA finds new dimensions along which the data vary a lot (the long axis of the ellipse)
 - * and dimensions along which the data don't vary much at all, and can be ignored (the short axis)

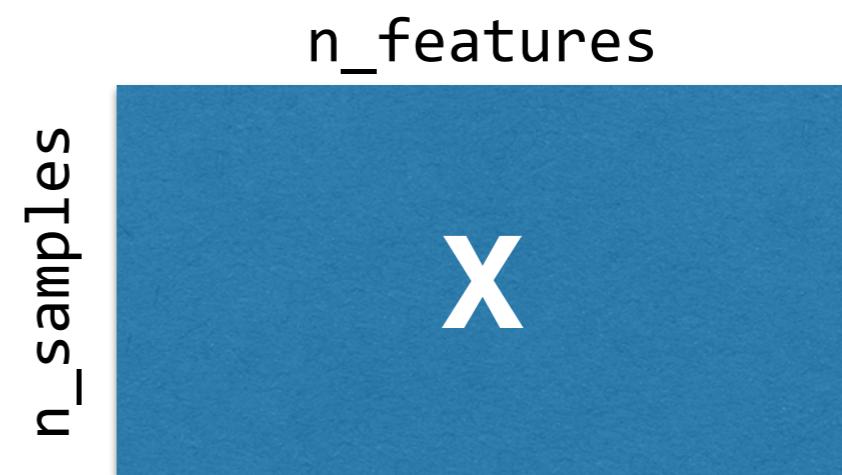
EXAMPLE: HUMAN GENETICS

- * Hundreds of Europeans are genotyped for thousands of genes
 - * This info is reduced to 2 dimensions using PCA



HOW DO YOU USE IT

- * The input to PCA is a matrix X with shape $(n_samples, n_features)$
- * (Assume that each column of X has zero mean. If this isn't true, you can make it true!)



HOW DO YOU USE IT

- * PCA represents \mathbf{X} as a product of 3 matrices: \mathbf{U} , Σ , and \mathbf{W}
- * (We assume that $n_{\text{features}} > n_{\text{samples}}$)

$$\begin{matrix} n_{\text{samples}} \\ \mathbf{X} \end{matrix} = \begin{matrix} n_{\text{features}} \\ \mathbf{U} \end{matrix} \Sigma \begin{matrix} n_{\text{features}} \\ \mathbf{WT} \end{matrix}$$

The diagram illustrates the decomposition of a data matrix \mathbf{X} into three components. The matrix \mathbf{X} has dimensions $n_{\text{samples}} \times n_{\text{features}}$. It is shown as a blue rectangle. To its right is an equals sign. To the left of the equals sign is a green rectangle labeled \mathbf{U} with dimensions $n_{\text{samples}} \times n_{\text{samples}}$. To the right of the equals sign is a pink rectangle labeled Σ with a red diagonal line through it, representing a diagonal matrix. To the right of Σ is a blue rectangle labeled \mathbf{WT} with dimensions $n_{\text{features}} \times n_{\text{features}}$.

HOW DO YOU USE IT

- * W contains the **principal components** (each row is one)
- * U is the **score matrix**, which tells you how much of each PC is in each sample
- * **Sigma** is a diagonal matrix of **singular values**, which tell you how “big” (or important) each PC is

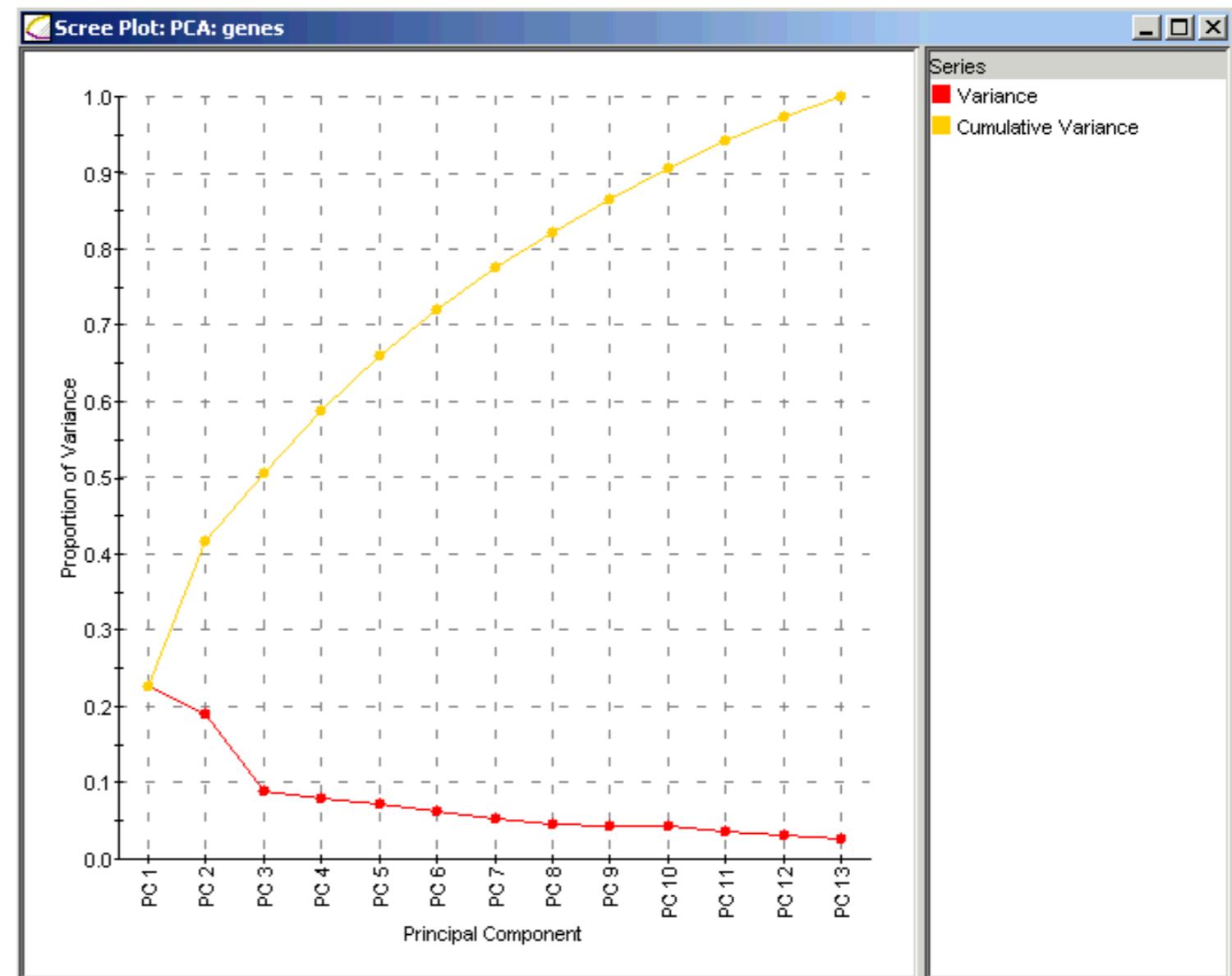
$$\begin{matrix} n_samples & & \\ & X & \\ \end{matrix} = \begin{matrix} n_features & & \\ & U & \\ n_samples & & \end{matrix} = \begin{matrix} n_samples & & \\ & \Sigma & \\ \end{matrix} = \begin{matrix} n_features & & \\ & WT & \\ \end{matrix}$$

HOW DO YOU USE IT

- * You can then visualize the scores (data projected onto PCs), the PCs themselves, and the singular values

HOW DO YOU USE IT

- * Singular values are typically visualized as a “Scree plot”
- * this shows how much variance each component accounts for in the dataset



HOW DO YOU USE IT

- * (If you're familiar with the **singular value decomposition (SVD)**, this is exactly the same thing)
- * you can use the SVD directly with `np.linalg.svd`

HOW DO YOU USE IT

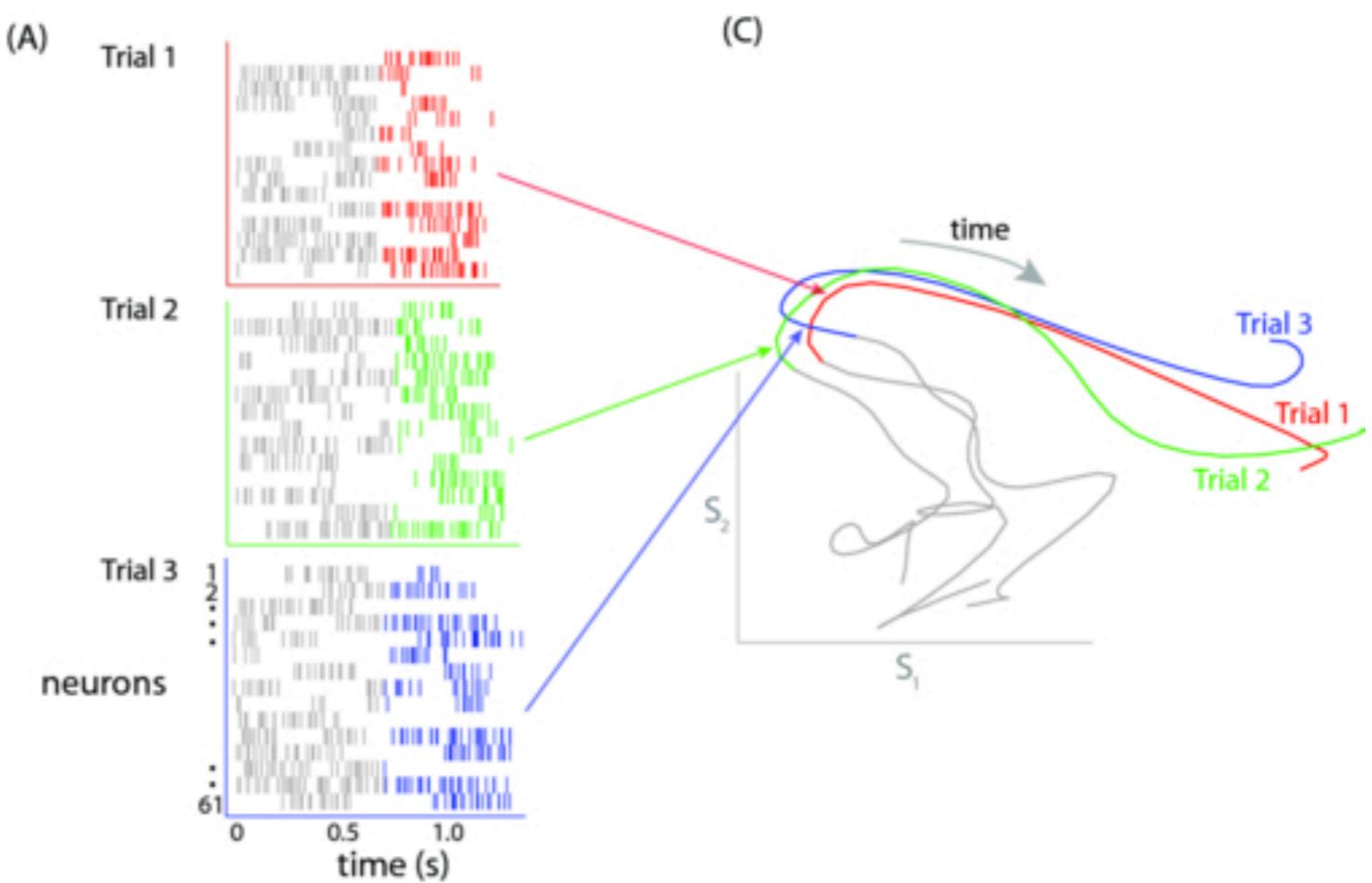
- * it's also possible to do PCA through scikit-learn with **sklearn.decomposition.PCA**
- * (This has more options, and will do nice things like subtract the column means for you.)

PCA APPLICATIONS

- * PCA can be used for either *data analysis* or *data visualization* (and often both)
- * It should probably be your #1 go-to visualization tool whenever you have lots of features and don't know how to look at them

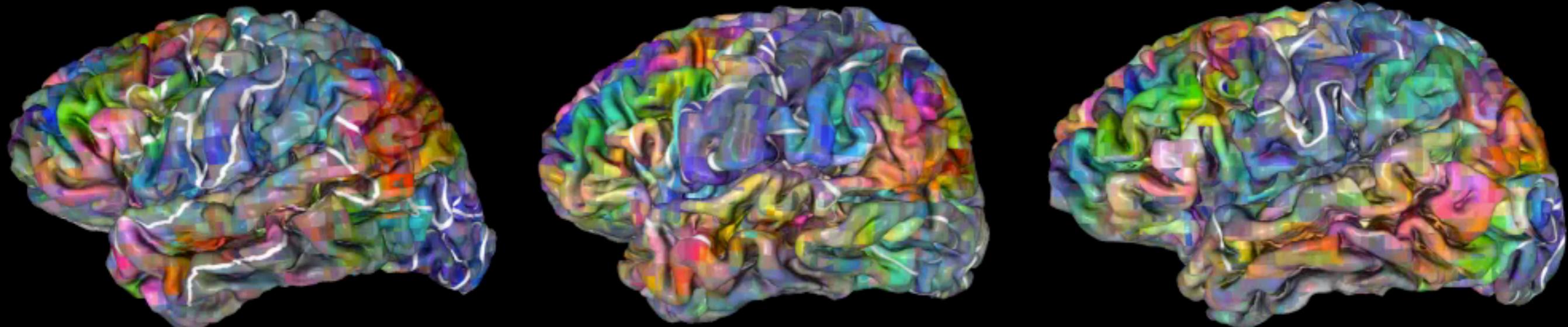
PCA APPLICATIONS

- * population analyses of neural activity
- * PCA gives a “neural state space”



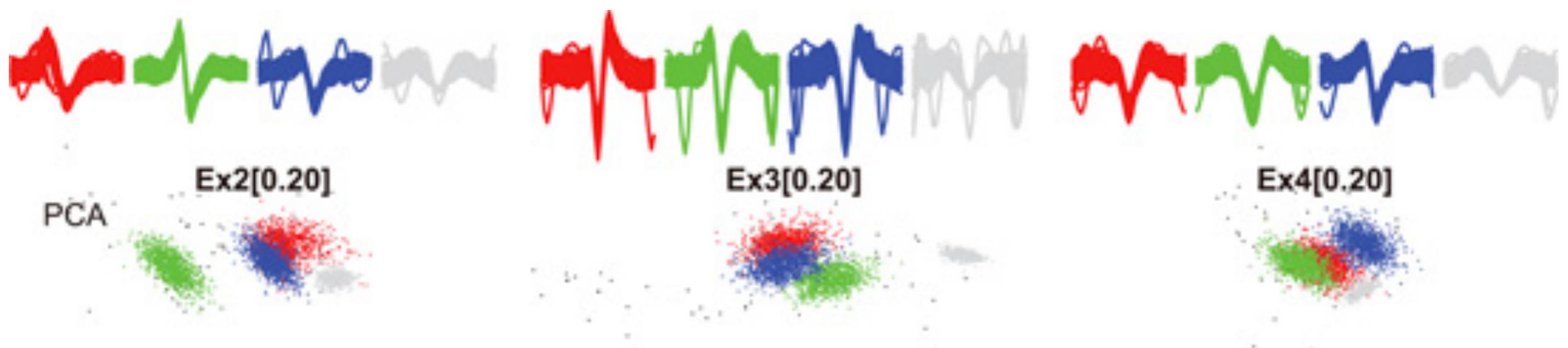
PCA APPLICATIONS

- * visualization of high-dimensional regression models



PCA APPLICATIONS

- * spike sorting!

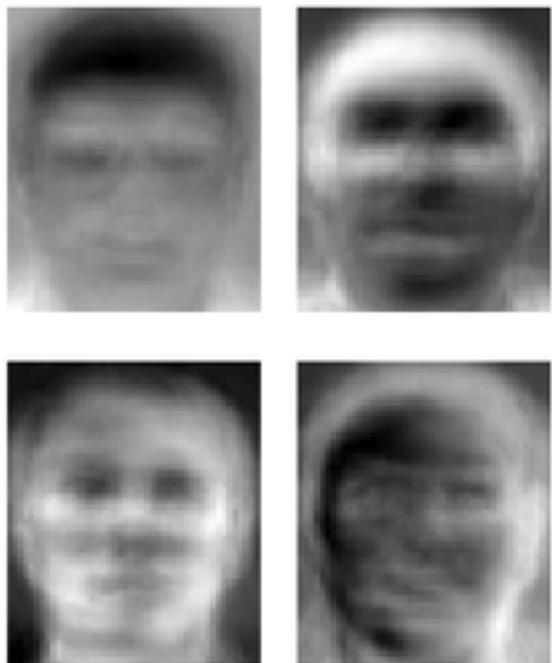


PCA APPLICATIONS

- * it's even possible that PCA is something that *individual neurons do*
- * Oja's rule is a slight modification to Hebb's rule for learning synaptic weights (it normalizes the weights so they can't grow without bound)
- * Oja's rule *IS PRINCIPAL COMPONENTS ANALYSIS*

PCA APPLICATIONS

- * do PCA on a whole bunch of (aligned) face images to get “eigenfaces” – basis functions that all face images are built out of
- * it turns out this might fundamental to how individual neurons represent faces! (Chang & Tsao, *Cell* 2017)



PCA EXTENSIONS: AUTOENCODERS

- * autoencoders are artificial neural networks that push each sample through a “bottleneck layer” and then try to reconstruct the original sample
- * simple autoencoders compute exactly PCA
- * more complicated ones can do very cool stuff

PCA EXTENSIONS: AUTOENCODERS

anger disgust fear happy sad surprise neutral



from Cheung, Livezey, Bansal, & Olshausen (2015)

MORE RESOURCES

- * If you want to learn more about PCA, Jake vanderPlas's book has a great chapter:
[https://jakevdp.github.io/
PythonDataScienceHandbook/05.09-
principal-component-analysis.html](https://jakevdp.github.io/PythonDataScienceHandbook/05.09-principal-component-analysis.html)

THANK YOU!