Dimensionaly Reduction

PCA(Principal Componenet Analysis)
[unsupervised method]

Q.

Why need singular (unit) vector

Why need to rotate all points to make PC1 horizontal

Number of PCA is equal to number of variables or numbe rof observations which is smaller.

Introduction

If you work with a lot of variables, this can present problems. In technical terms, you want to "reduce the dimension of your feature space." By reducing the dimension of your feature space, you have fewer relationships between variables to consider.

Somewhat unsurprisingly, ***reducing*** the ***dimension*** of the feature space is called "***dimensionality reduction***." There are many ways to achieve dimensionality reduction, but most of these techniques fall into one of two classes:

- Feature Elimination
- Feature Extraction

Feature elimination is what it sounds like: we reduce the feature space by eliminating features. Advantages of feature elimination methods include simplicity and maintaining interpretability of your variables.

As a disadvantage, though, you gain no information from those variables you've dropped. We're missing out on whatever the dropped variables could contribute to our model. By eliminating features, we've also entirely eliminated any benefits those dropped variables would bring.

- Step wise forward selection
- Step wise backward selection
- Combination of both forward and backward
- Descision tree induction

Feature extraction, however, doesn't run into this problem. Say we have ten independent variables. In feature extraction, we create few (may less than 10) "new" independent variables, where each "new" independent variable is a combination of each of the ten "old" independent variables. However, we create these new independent variables in a specific way and order these new variables by how well they predict our dependent variable.

Principal component analysis is a technique for *feature extraction*—so it combines our input variables in a specific way, then we can drop the "least important" variables while still retaining the most valuable parts of all of the variables! *As an added benefit, each of the "new" variables after PCA are all independent of one another*. This is a benefit because the assumptions of a linear model require our independent variables to be independent of one another.

PCA takes the dataset with a lot of dimensions and flattens it to less (2 or 3) dimensions. It tries to find a meaningful way to flatten the data by focusing on the things that are different between cells.

When should I use PCA?

1. Do you want to reduce the number of variables, but aren't able to identify variables to completely remove from consideration?
2. Do you want to ensure your variables are independent of one another?
3. Are you comfortable making your independent variables less interpretable?

Pre-requisit for PCA

- Matrix operations/linear algebra (matrix multiplication, matrix transposition, matrix inverses, matrix decomposition, eigenvectors/eigenvalues)
- Statistics/machine learning (standardization, variance, covariance, correlation, independence, linear regression, feature selection).

How does PCA work?

- We are going to calculate a matrix that summarizes how our variables all relate to one another.
- We'll then break this matrix down into two separate components: direction and magnitude. We can then understand the "directions" of our data and its "magnitude" (or how "important" each direction is).

- Transform our original data to align with these important directions (which are combinations of our original variables). **By projecting our data into a smaller space, we're reducing the dimensionality of our feature space… but because we've transformed our data in these different "directions," we've made sure to keep all original variables in our model!**

Understand PCA through less technical example

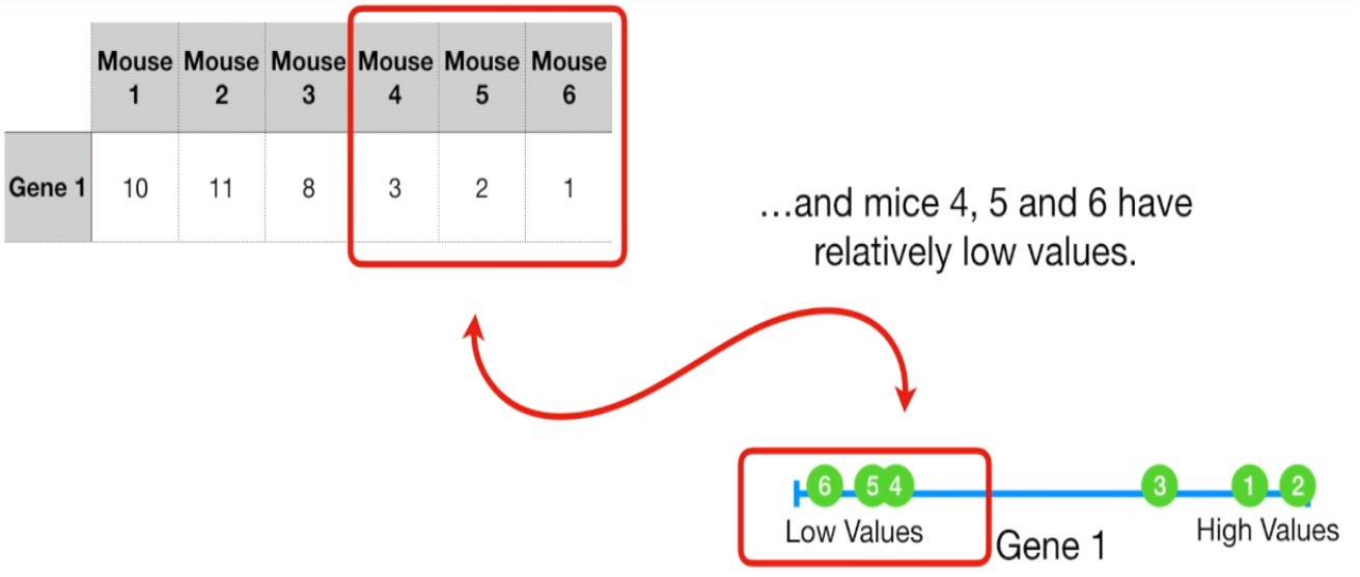
	Mouse 1	Mouse 2	Mouse 3	Mouse 4	Mouse 5	Mouse 6
Gene 1	10	11	8	3	2	1
Gene 2	6	4	5	3	2.8	1

Mouse 1…n = Records

Gene1 and Gene2 =Features

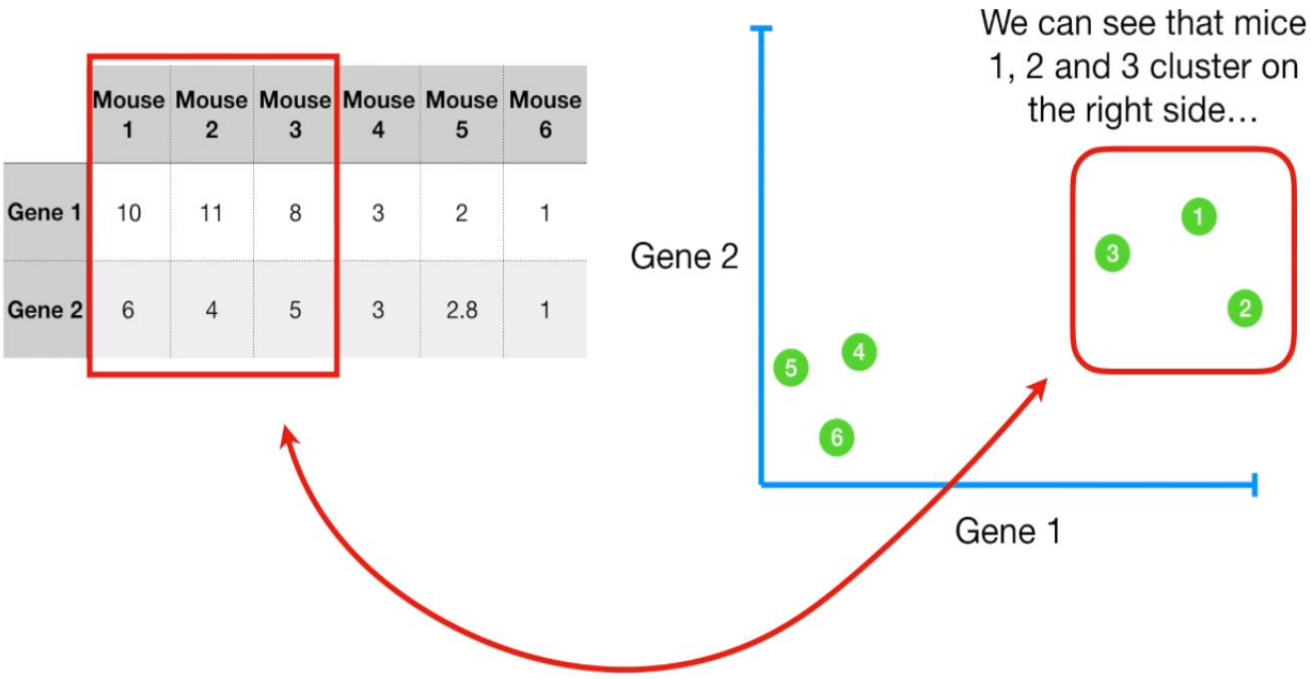
	Student 1	Student 2	Student 3	Student 4	...
Math	95	88	93	75	...
Reading	96	79	98	81	...

If we measure the records against single Gene, then we can plot the data on single number line. Mice 1,2,3 have relatively high values and mice 4,5,6 have relatively low values. In other words, mice 1,2,3 are similar to each other than mice 4,5,6.

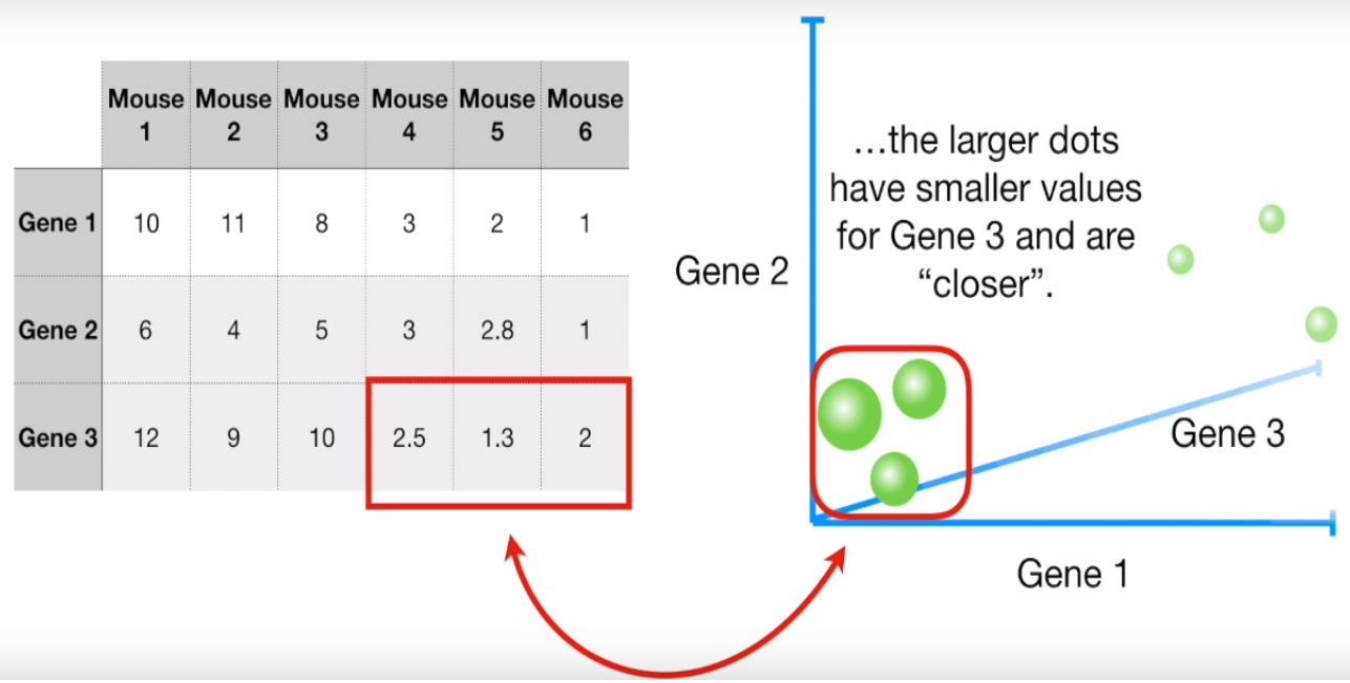


If we measured two genes then we can plot the data in on a two-dimensional x/y graph, i.e. Gene-1 is on the x-axis and Gene-2 on the y-axis.

We can see mice 1,2,3 are clustered in right hand side and mice 4,5,6 cluster on the lower left hand side.

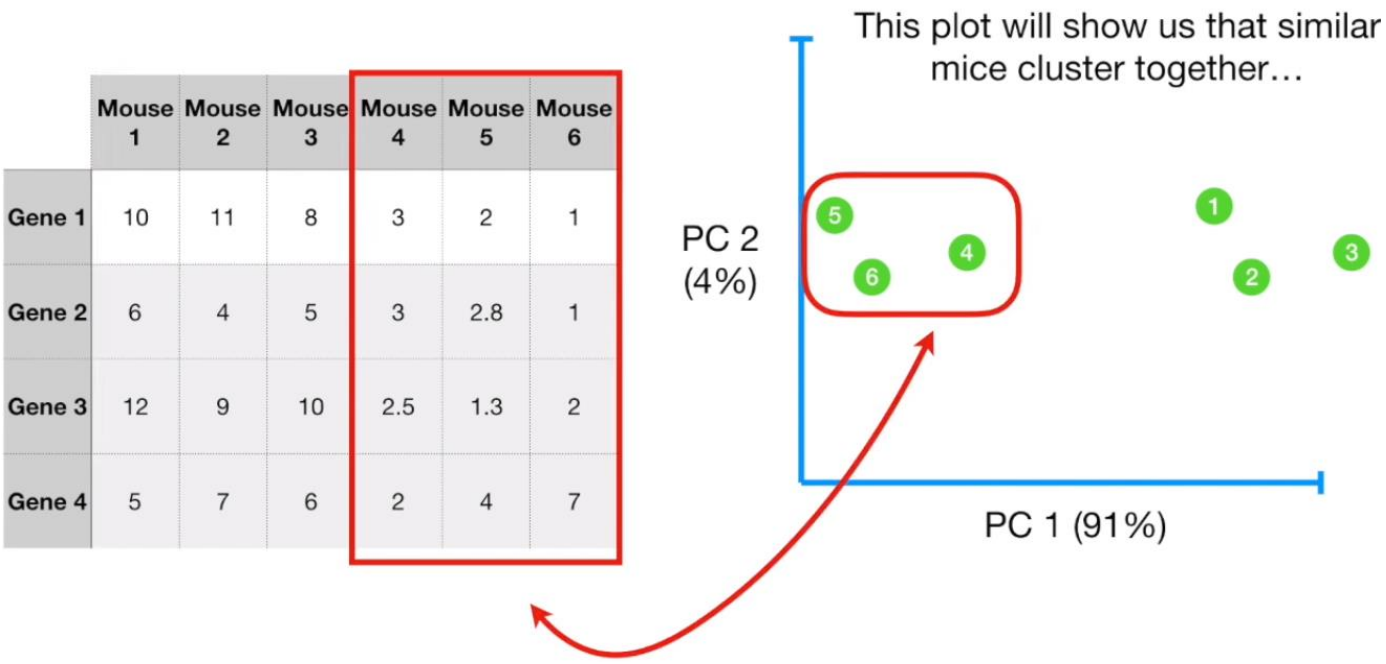


If we measured three Genes, we would add another axis to the graph and it look like 3-Dimensional.



If we measured 4- Genes and it requires 4-dimenstion….no longer able to plot.

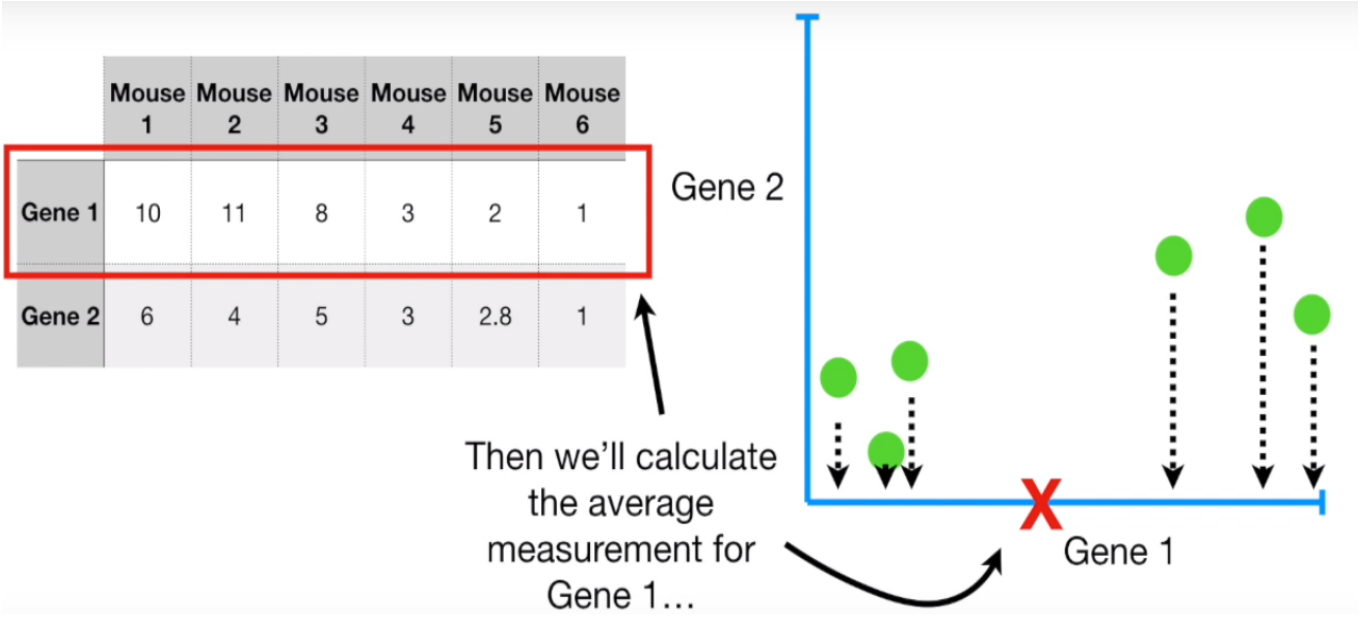
So, we are going to solve this problem with PCA that can take 4 or mores Gene measurements (features) and make a 2-D (not fixed) PCA plot…



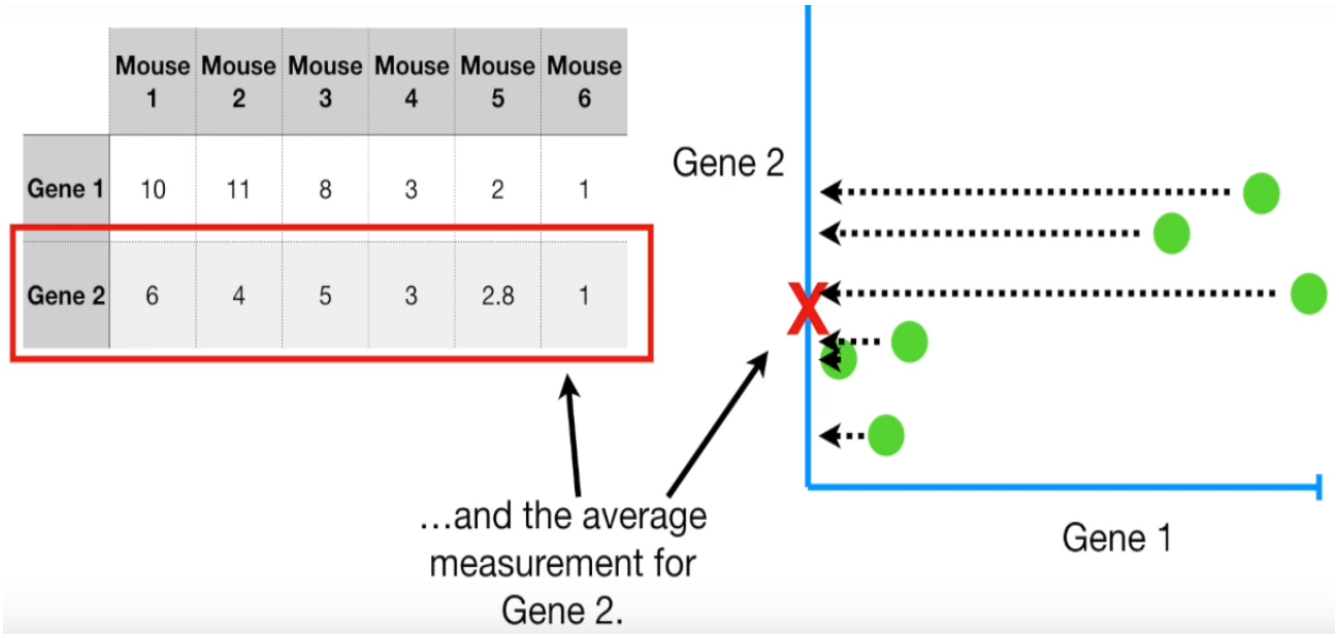
	Mouse 1	Mouse 2	Mouse 3	Mouse 4	Mouse 5	Mouse 6
Gene 1	10	11	8	3	2	1
Gene 2	6	4	5	3	2.8	1
Gene 3	12	9	10	2.5	1.3	2
Gene 4	5	7	6	2	4	7

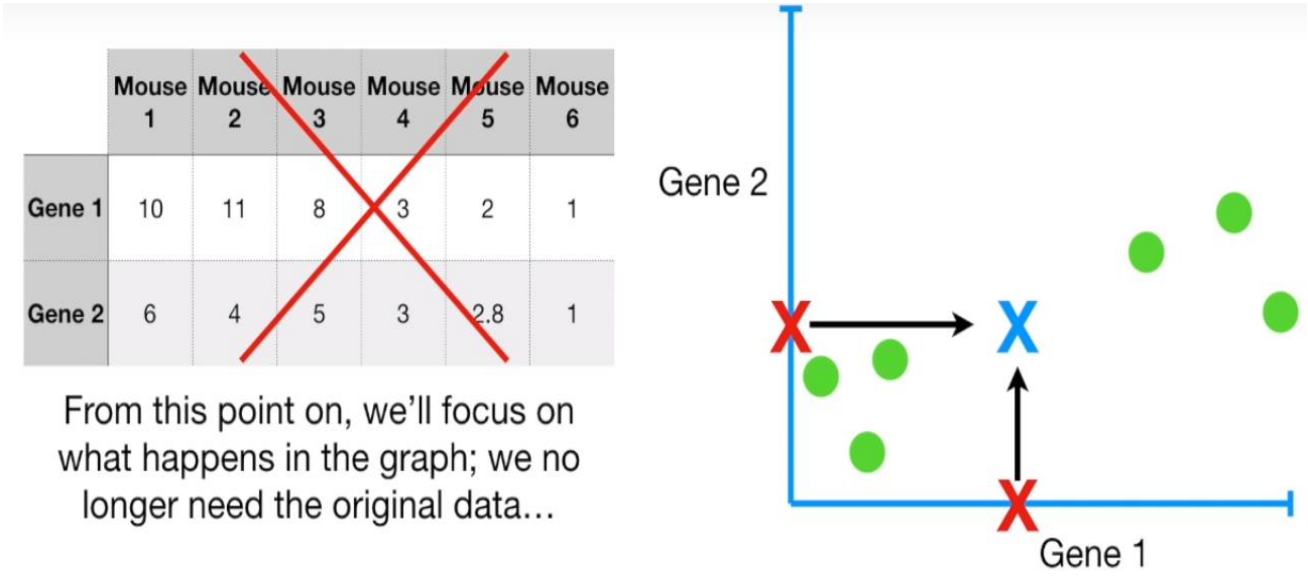
X-axis explain 91% variation and y-axis 4% of data.

To understand PCA working, take 2-D data:

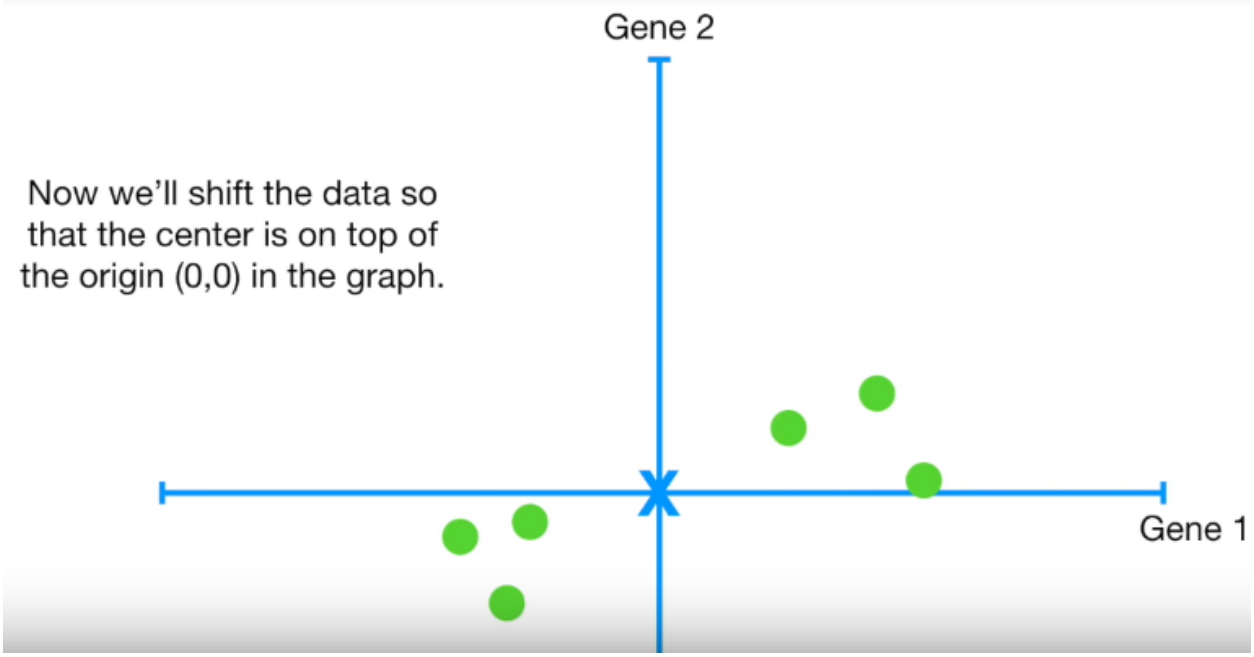


	Mouse 1	Mouse 2	Mouse 3	Mouse 4	Mouse 5	Mouse 6
Gene 1	10	11	8	3	2	1
Gene 2	6	4	5	3	2.8	1



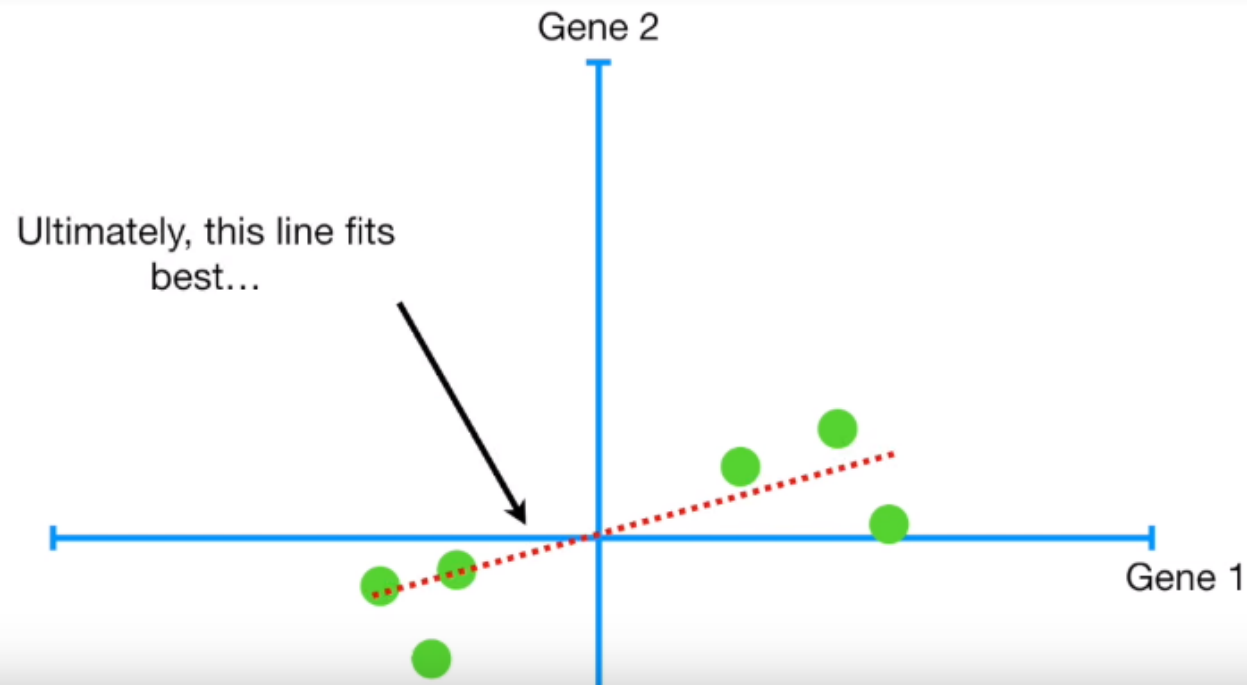


With the average value, we can calculate the center of the data.



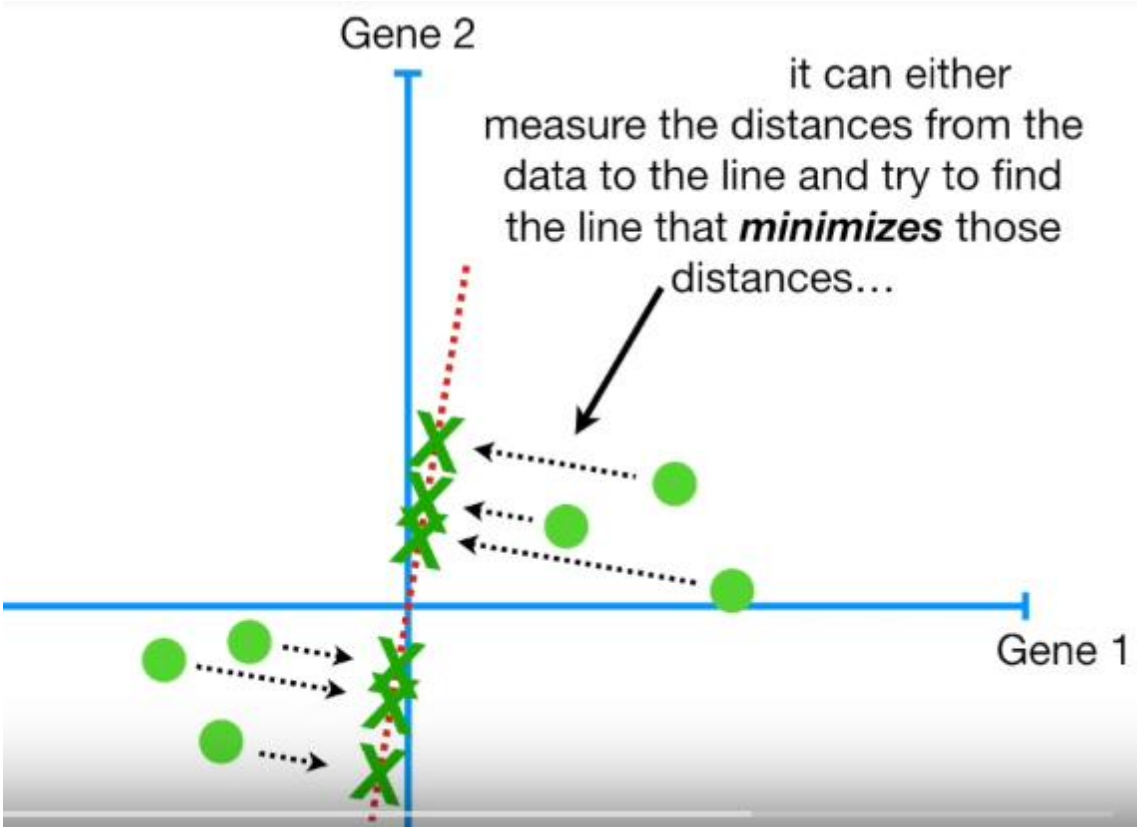
Note: Shifting the data did not change how the data points are positioned *relative* to each other.

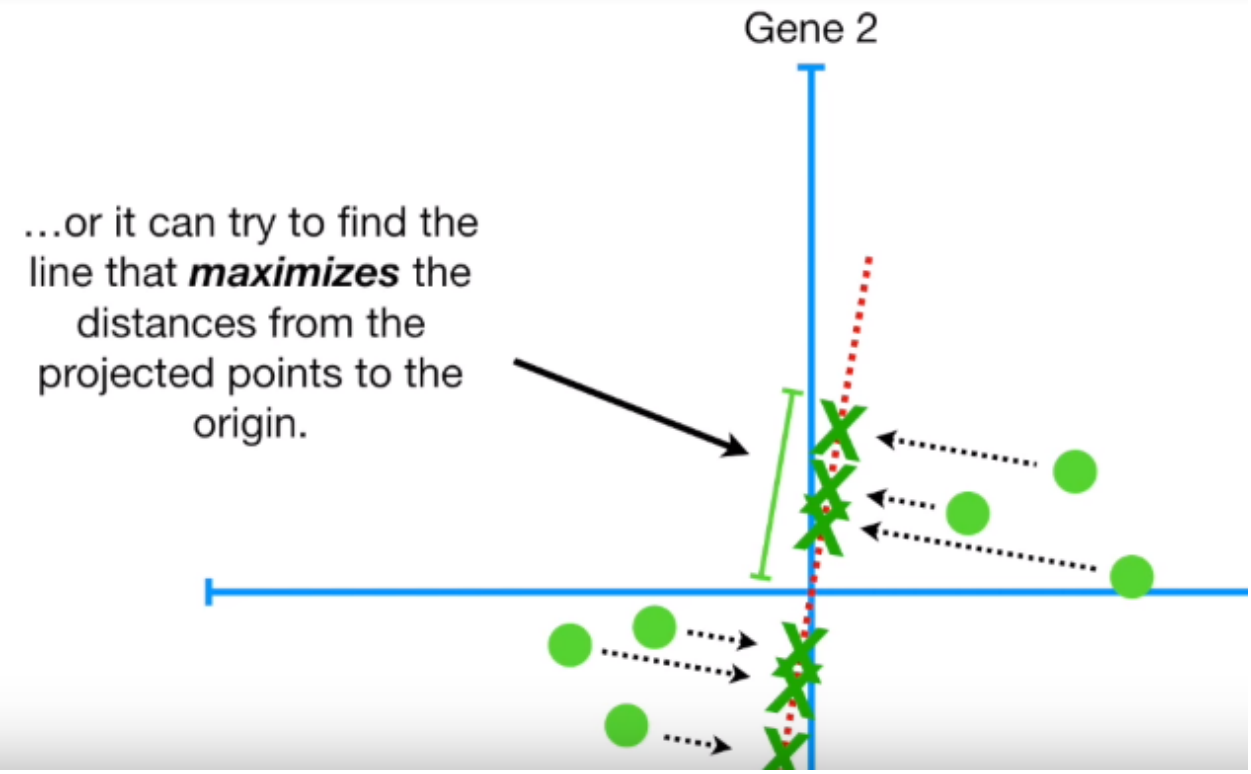
Now, the data are centered on the origin, we can try to fit a line to it. We start by drawing the random line that goes through the origin. Then we rotate the line until it best fits the data.

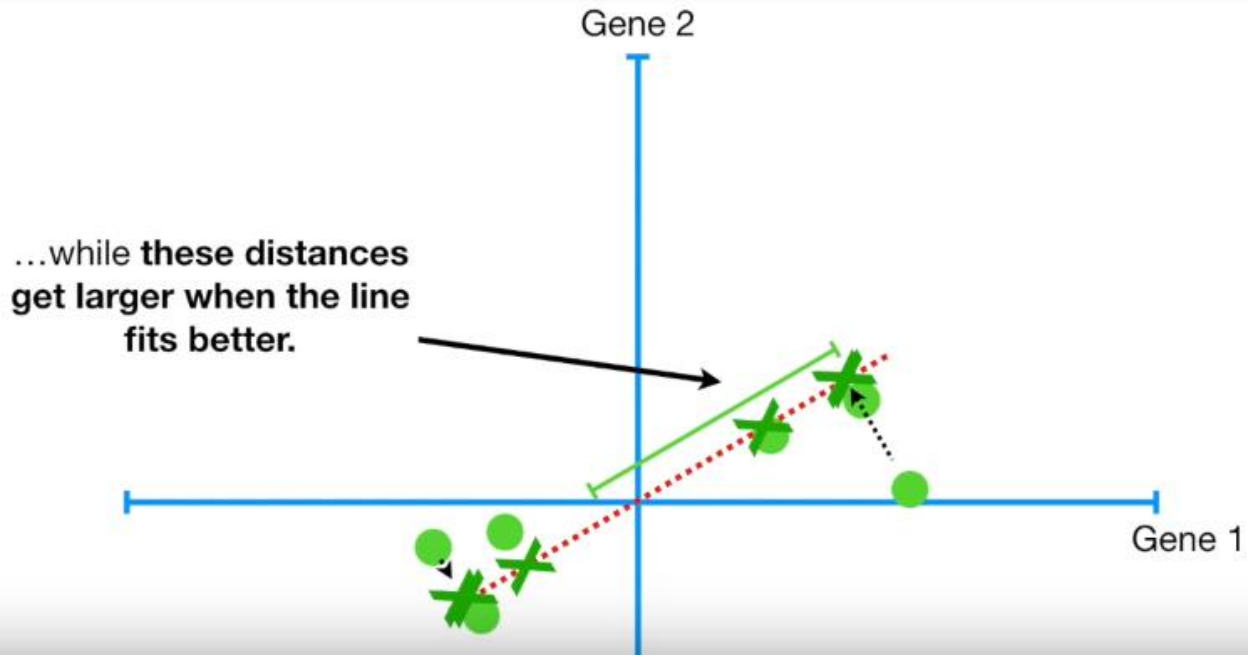


Q. How PCA decides if a fit is good or not?

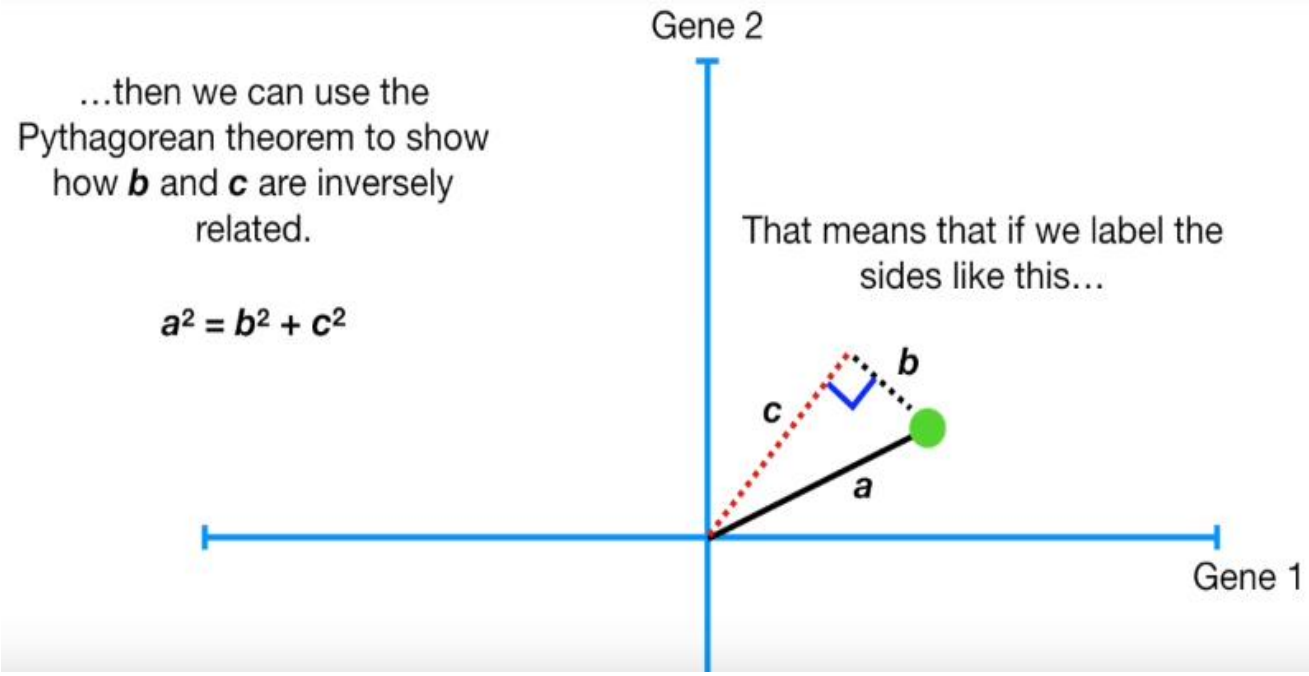
Ans: There are two ways:





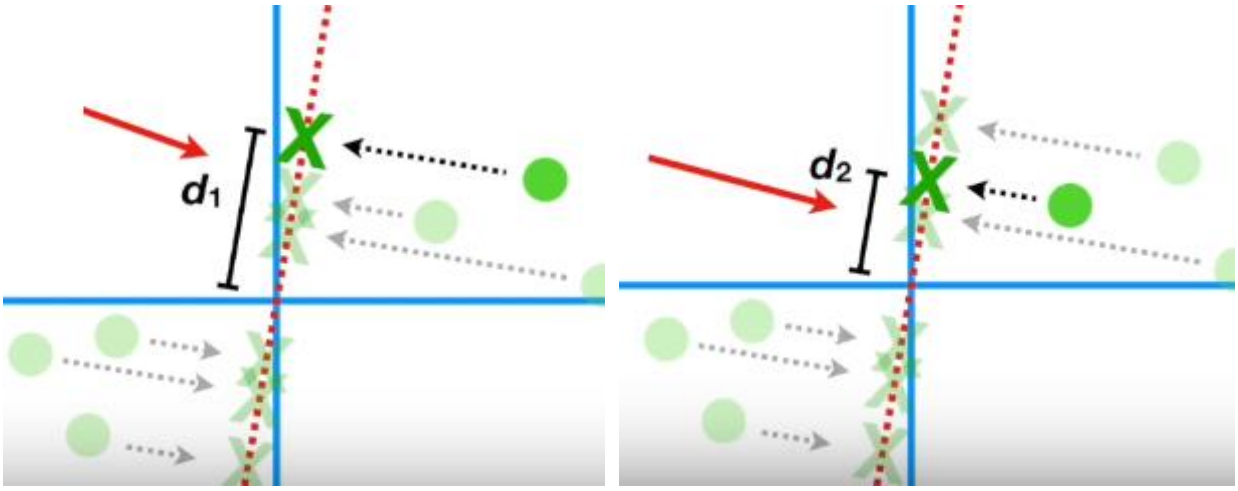


Let’s understand in mathematical way:

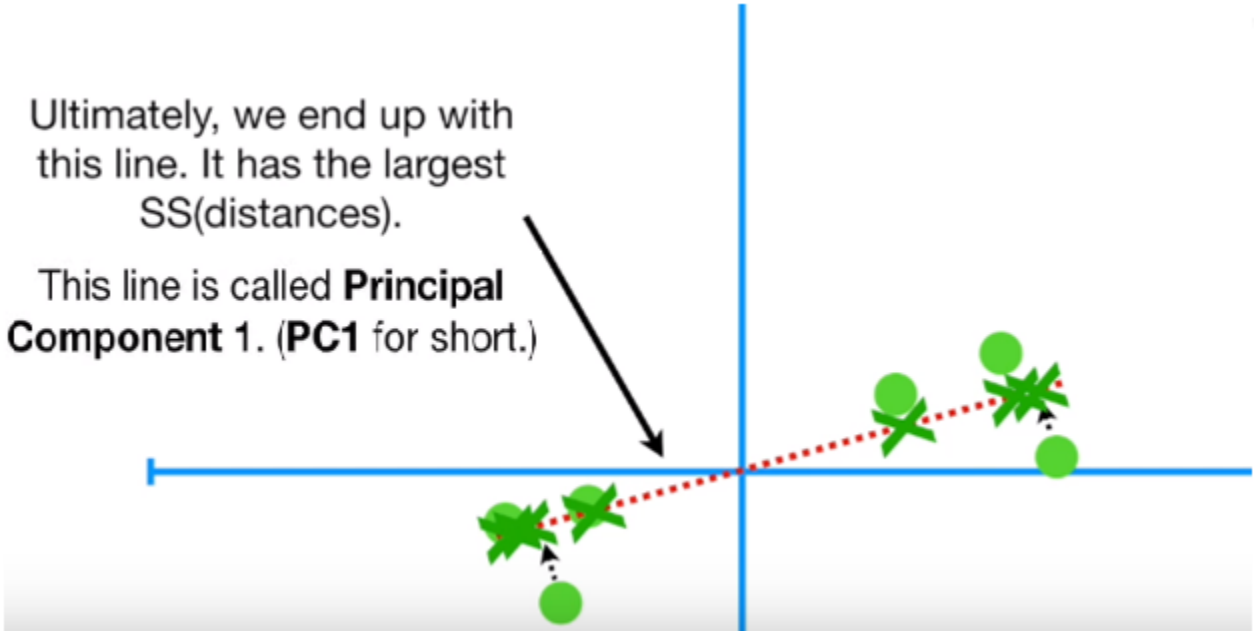


a^2 doesn't changes.

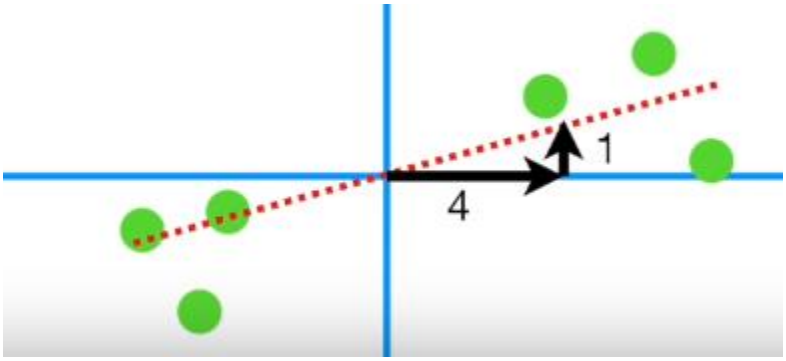
Note: Its actually easier to calculate 'C', the distance from the projected point to the origin, so PCA finds the best fitting line by maximizing the sum of the squared distance from the projected points to the origin.



$d_1^2 + d_2^2 + d_3^2 + d_4^2 + d_5^2 + d_6^2$ = sum of squared distances = SS(distances)



PC1 has the slop of 0.25. In other words for every 4 units go out along the x-axis (Gene-1), we go up 1-unit along the y-axis (Gene-2). That means data are mostly spread out along the x-axis (Gene-1) and only little bit spread out along the y-axis (Gene-2).

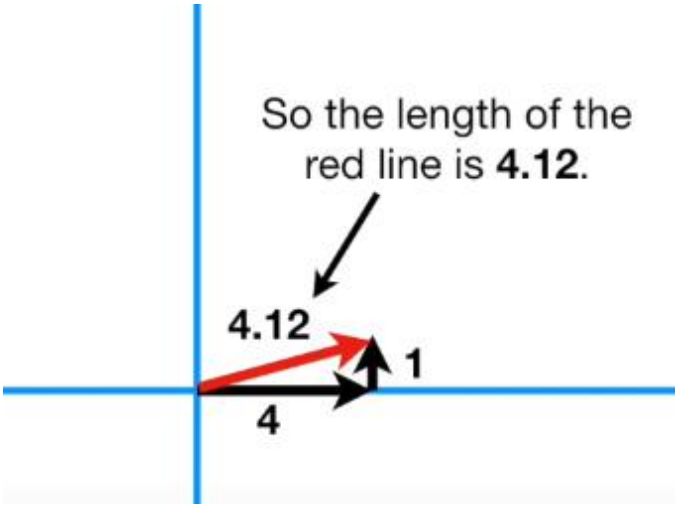


To make PC-1

Mix 4-part of gene-1

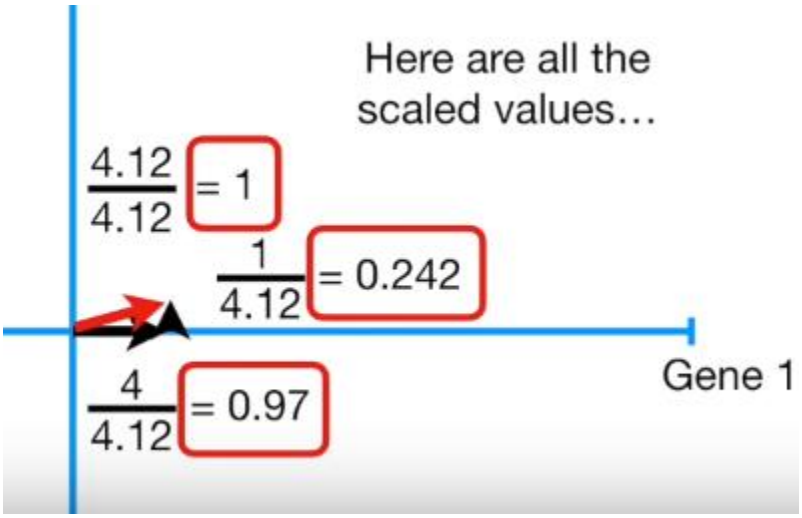
With 1-part of Gene-2

Mathematically, we call it linear combination of Gene-1 and Gene-2 (PCA is a liner combination of variables).



$$a = \sqrt{4^2 + 1^2} = 4.12$$

We do PCA with SVD (single value decomposition), we scaled PCA so that length =1 (unit vector).



To make PC-1

Mix 0.97-part of gene-1

With 0.242-part of Gene-2

Note: ration is the same

This one-unit long vector, consisting of 0.97 parts of gene-1 and 0.242 parts of Gene-2, is called the **SINGULAR VECTOR** or **EIGENVECTOR** for PC1. The proportion of each gene are called **LOADING SCORES**.

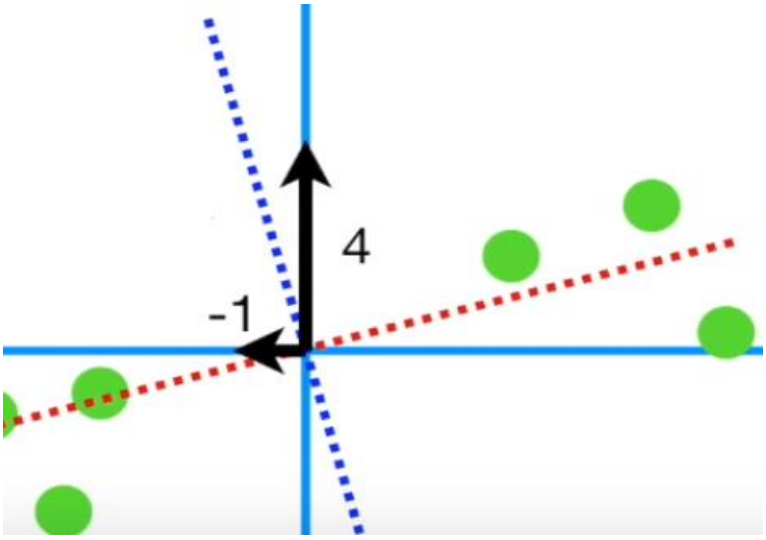
SS(distances for PC1) = Eigenvalue for PC1

$\sqrt{\text{Eigenvalue for PC1}}$ = Singular Value for PC1

…and the square root of the **Eigenvalue for PC1** is called the **Singular Value for PC1.**

Lets work on PC2

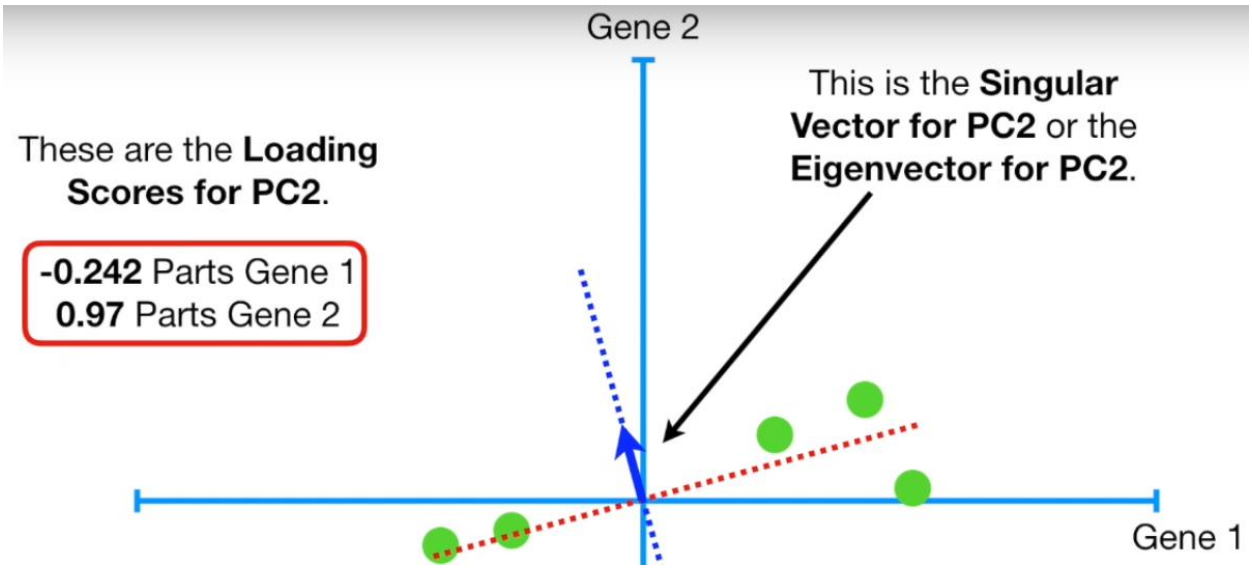
PC2 is simply the line through the origin that is perpendicular to PC1.



To make PC-2

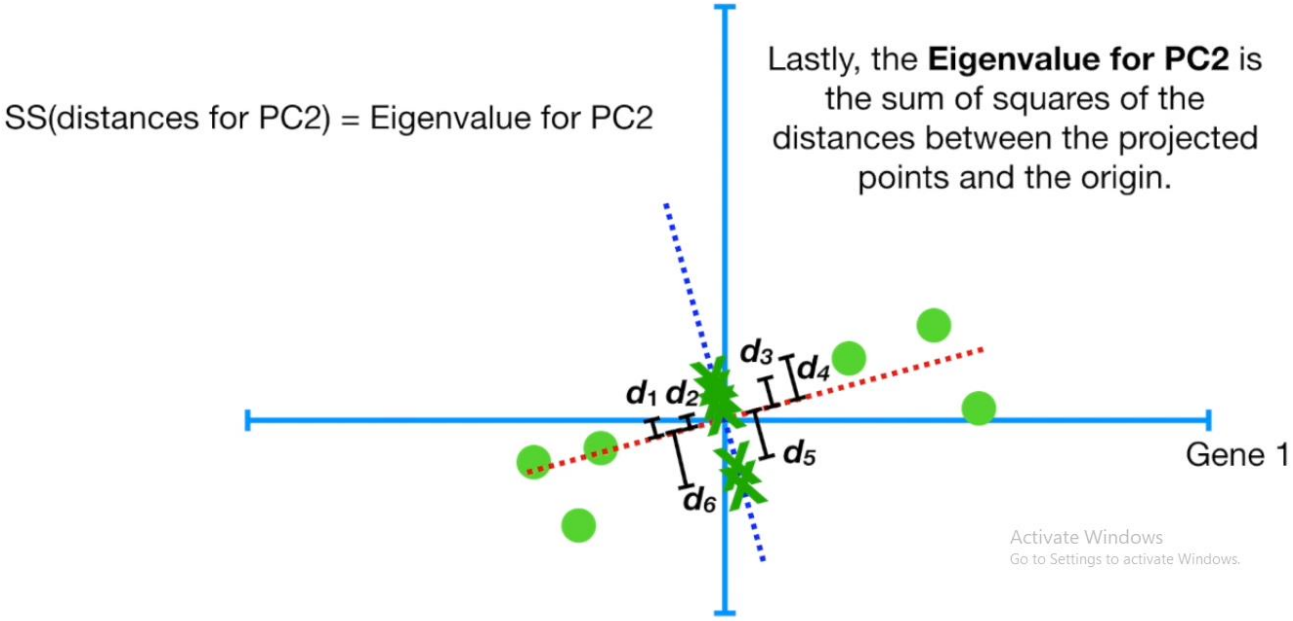
-1 part of gene-1

4 part of Gene-2

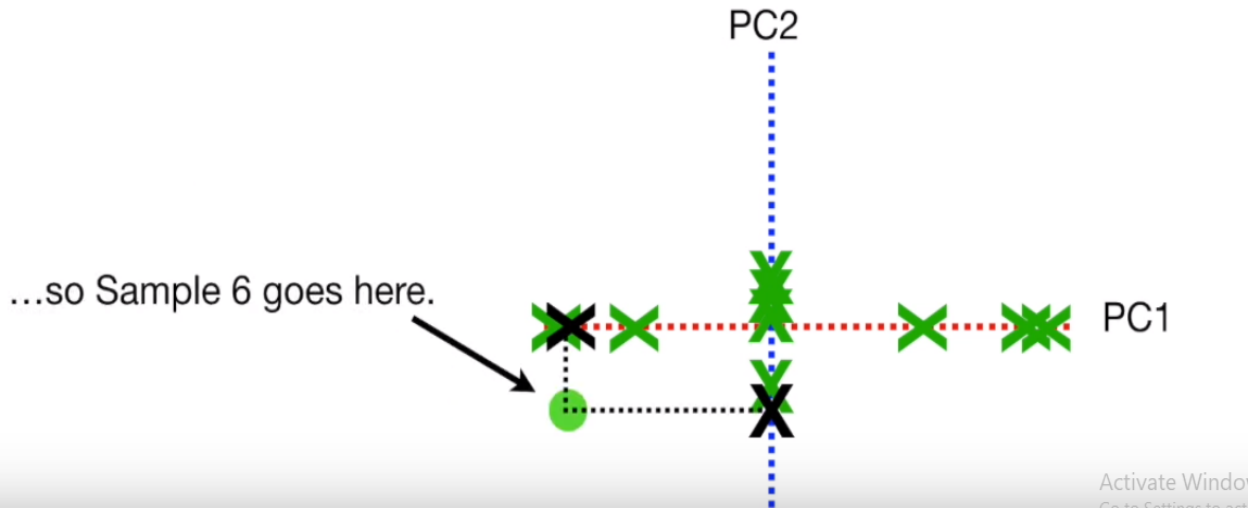


Gene-2 is 4 times as important as Gene-1.

$d_1^2 + d_2^2 + d_3^2 + d_4^2 + d_5^2 + d_6^2$ = sum of squared distances = SS(distances)



Now, we rotate everything such that PC1 become horizontal and points are project accordingly.



Calculate total variance calculated by PCA

$$\frac{\text{SS(distances for PC1)}}{n - 1} = \text{Variation for PC1}$$

$$\frac{\text{SS(distances for PC2)}}{n - 1} = \text{Variation for PC2}$$

Suppose,

The variation of PC-1 is = 15

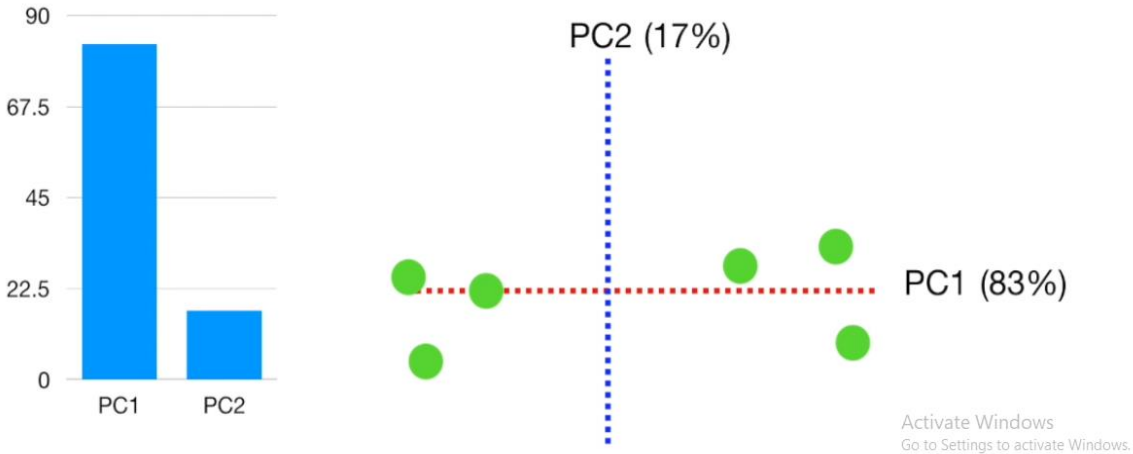
The variation of PC-2 is = 3

Total variations around both PCs=18

PC-1 account 15/18 (83%) of total variations around the PCs

Similarly, PC-2, 17%

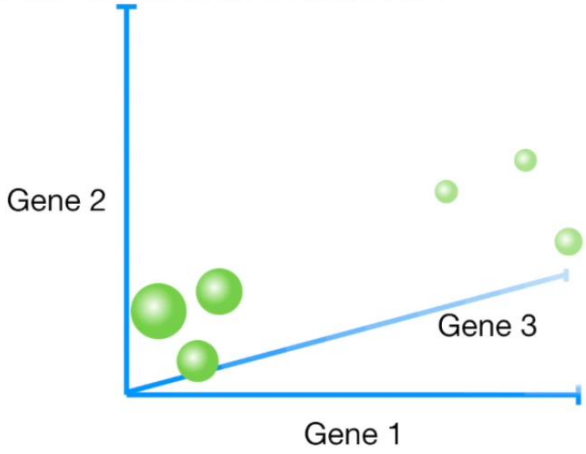
TERMINOLOGY ALERT!!!! A **Scree Plot** is a graphical representation of the percentages of variation that each PC accounts for.

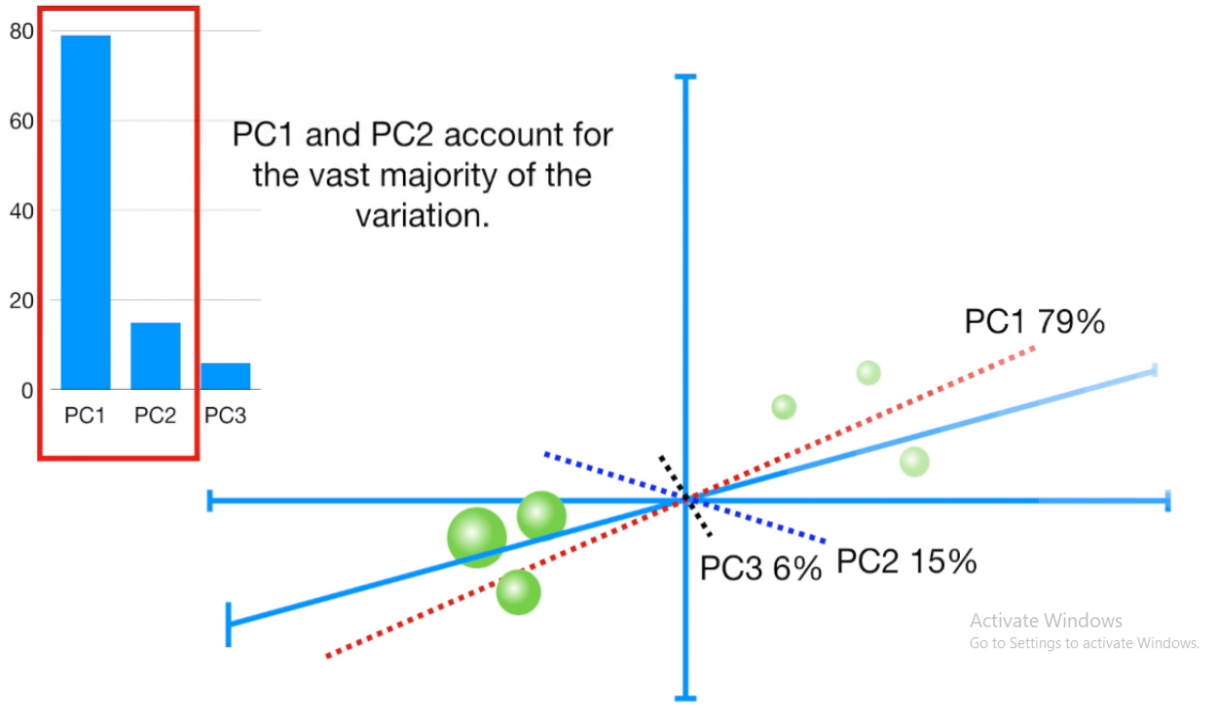


Consider little more complex example (3-D)

PCA with 3 variables (in this case, that means 3 genes) is pretty much the same as 2 variables…

	Mouse 1	Mouse 2	Mouse 3	Mouse 4	Mouse 5	Mouse 6
Gene 1	10	11	8	3	2	1
Gene 2	6	4	5	3	2.8	1
Gene 3	12	9	10	2.5	1.3	2



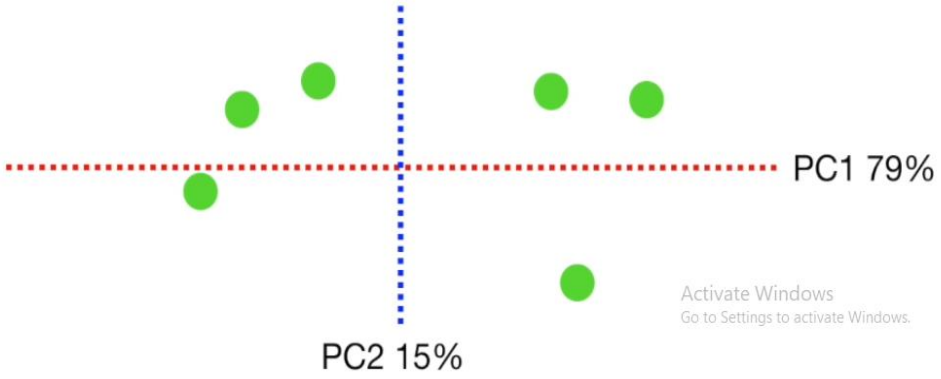


To make PC-1
0.62 part of gene-1
0.15 part of Gene-2
0.77 part of Gene-3

To make PC-2
0.77 part of gene-1
0.62 part of Gene-2
0.15 part of Gene-3

Dimensionality reduction: we determined that PC1 and PC2 (data in 2-D) still very informative, i.e. 94% variations defined by PC1 and PC2. In this view, we can drop (exclude PC3) PC3 in our model.

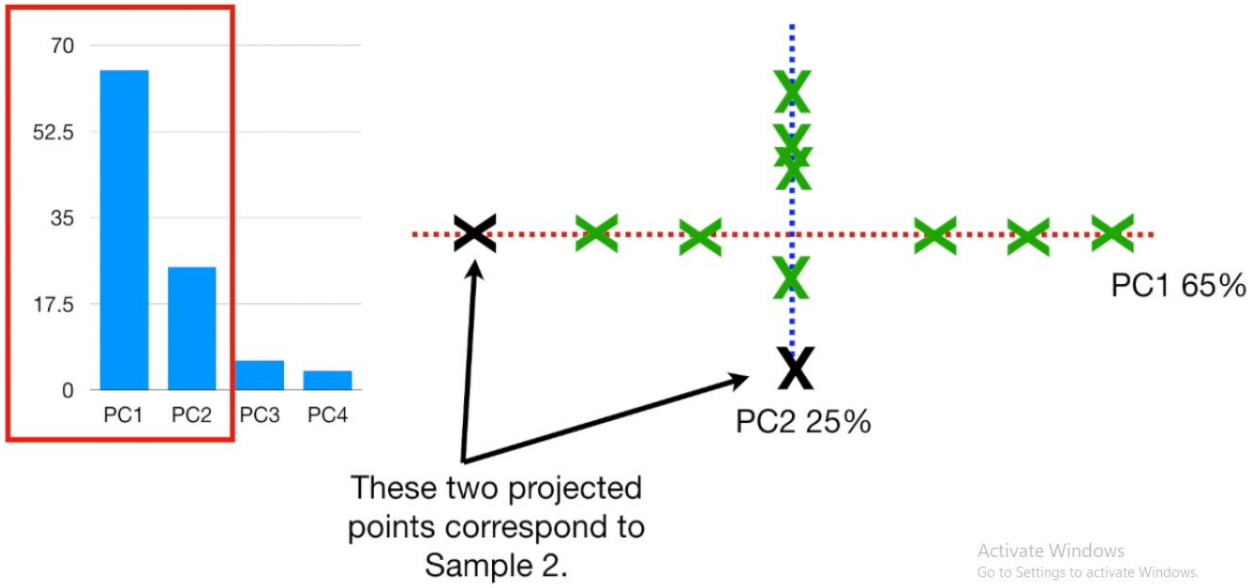
…lastly, we used PC1 and PC2 to draw a 2-Dimensional graph with the data.



Consider data with more features

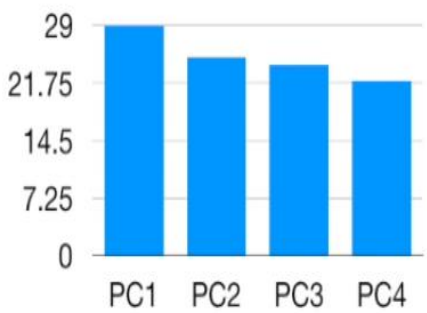
	Mouse 1	Mouse 2	Mouse 3	Mouse 4	Mouse 5	Mouse 6
Gene 1	10	11	8	3	2	1
Gene 2	6	4	5	3	2.8	1
Gene 3	12	9	10	2.5	1.3	2
Gene 4	5	20	6	2	18	19

If we measured 4 genes per mouse, we would not be able to draw a 4-dimensional graph of the data…



Still two PCs are determine more than 90% variations and so keep them only.

Suppose, PCs contributions is almost evenly distributed:



NOTE: If the scree plot looked like this, where PC3 and PC4 account for a substantial amount of variation, then just using the first 2 PCs would not create a very accurate representation of the data.

Finally, we need to determine how many features to keep versus how many to drop. There are three common methods to determine this, discussed below and followed by an explicit example:

- **Method 1**: We arbitrarily select how many dimensions we want to keep. Perhaps I want to visually represent things in two dimensions, so I may only keep two features. This is use-case dependent and there isn't a hard-and-fast rule for how many features I should pick.
- **Method 2**: Calculate the proportion of variance explained for each feature, pick a threshold, and add features until you hit that threshold. (For example, if you want to explain 80% of the total variability possibly explained by your model, add features with the largest explained proportion of variance until your proportion of variance explained hits or exceeds 80%.)
- **Method 3**: This is closely related to Method 2. Calculate the proportion of variance explained for each feature, sort features by proportion of variance explained and plot the cumulative proportion of variance explained as you keep more features. One can pick how many features to include by identifying the point where adding a new feature has a significant drop in variance explained relative to the previous feature, and choosing features up until that point. (I call this the "find the elbow" method, as looking at the "bend" or "elbow" in the scree plot determines where the biggest drop in proportion of variance explained occurs.)

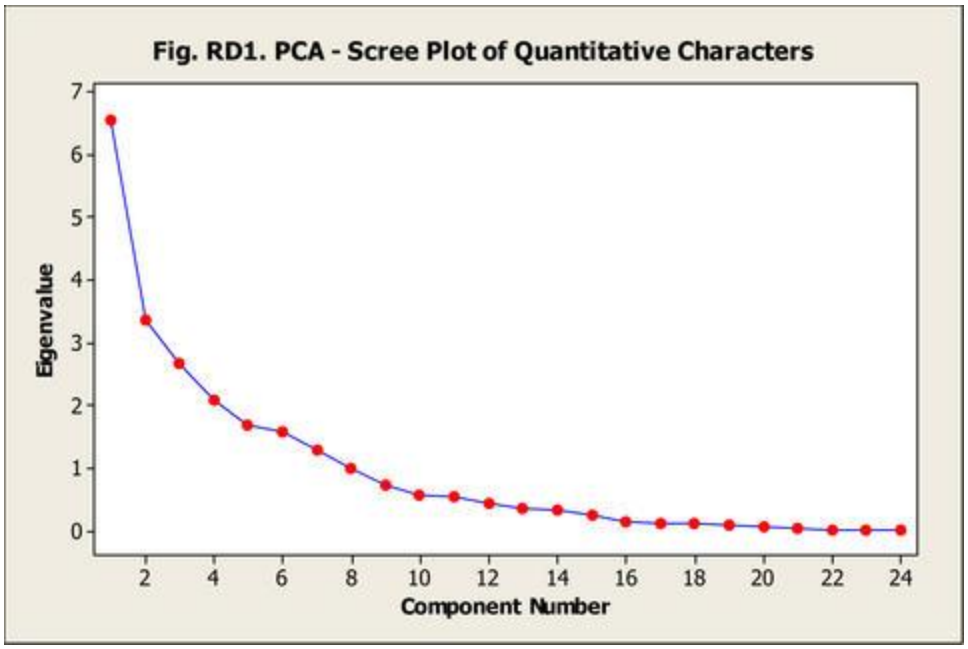
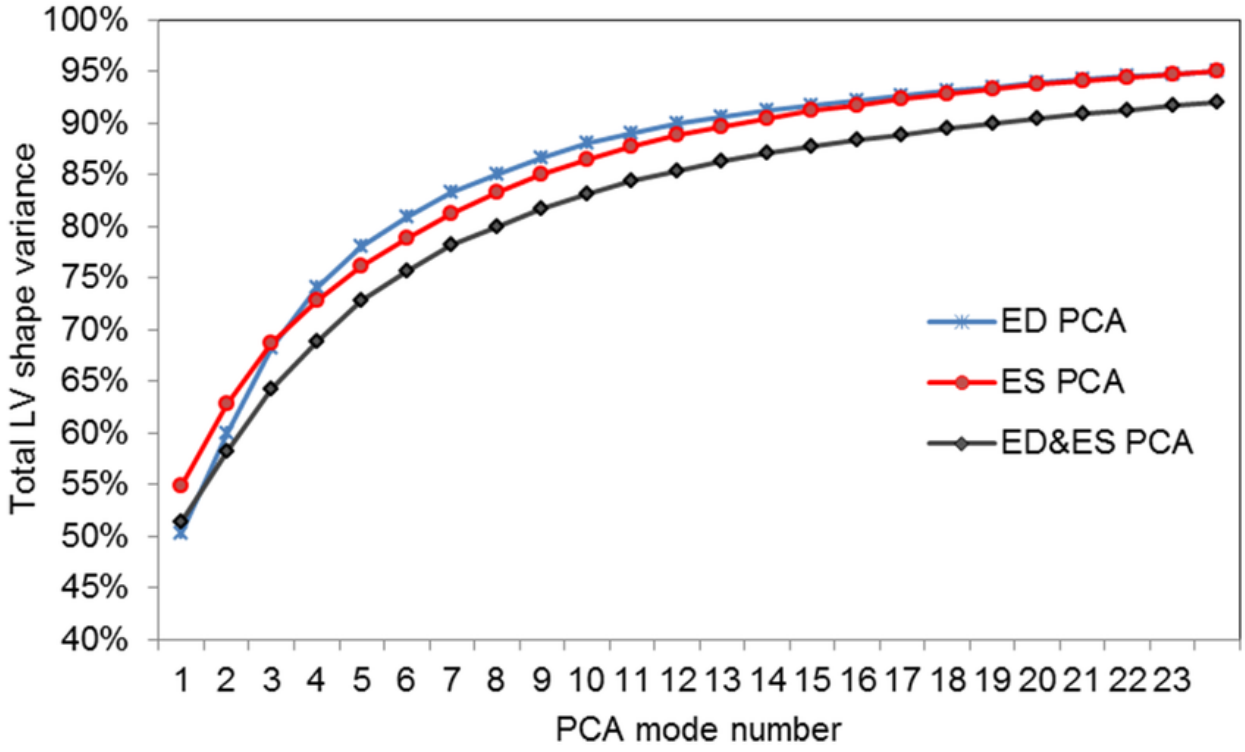
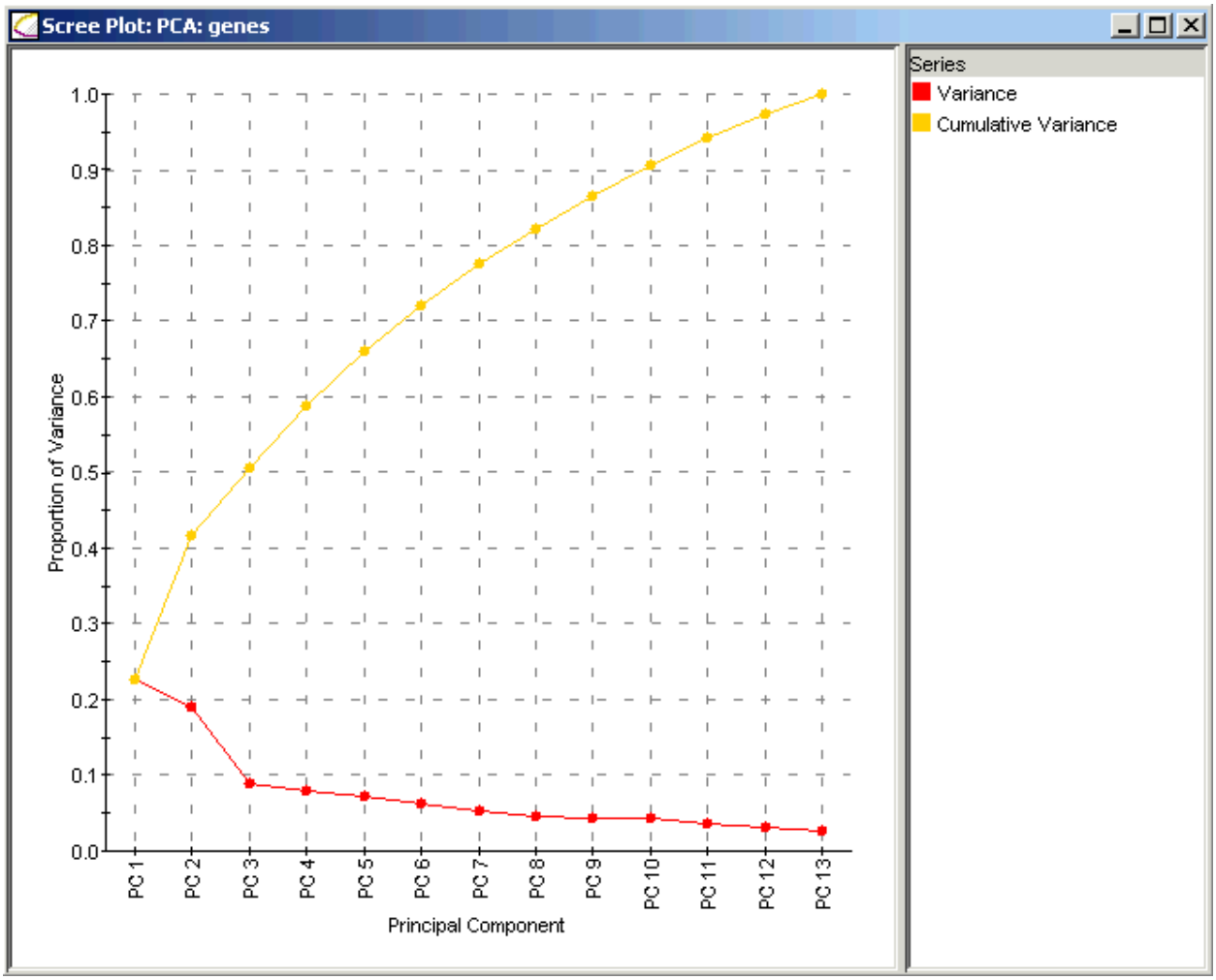


Fig. RD1. PCA - Scree Plot of Quantitative Characters
Eigenvalue
Component Number
0
1
2
3
4
5
6
7
2
4
6
8
10
12
14
16
18
20
22
24


Total LV shape variance
PCA mode number
ED PCA
ES PCA
ED&ES PCA
100%
95%
90%
85%
80%
75%
70%
65%
60%
55%
50%
45%
40%
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23



PCA is a method that brings together:

1. Standardize the data
2. A measure of how each variable is associated with one another. (Covariance matrix.)

   [Obtain the Eigenvectors and Eigenvalues from the covariance matrix or correlation matrix, or perform Singular Vector Decomposition (SVD)]

3. The directions in which our data are dispersed. (Eigenvectors.)

4. The relative importance of these different directions. (Eigenvalues.)
5. Sort eigenvalues in descending order and choose the k eigenvectors that correspond to the k largest eigenvalues where k is the number of dimensions of the new feature subspace (k≤d).
6. Construct the projection matrix W from the selected k eigenvectors.

Covariance Matrix

Covariance is the measure of how two different variables change together. The covariance between two variables, X and Y, can be given by the following formula.

$$cov = \frac{\sum_{i=1}^{n}(X_i - \bar{X})(Y_i - \bar{Y})}{(n-1)}$$

where $\bar{\mathbf{x}}$ is the mean vector $\bar{\mathbf{x}} = \sum_{k=1}^{n} x_i$.

Now, if we wanted to look at all the possible covariance in a dataset, we can compute the covariance matrix, which has this form –

$$C = \begin{pmatrix} cov(x,x) & cov(x,y) & cov(x,z) \\ cov(y,x) & cov(y,y) & cov(y,z) \\ cov(z,x) & cov(z,y) & cov(z,z) \end{pmatrix}$$

Notice that this matrix will be **symmetric** ($A=A^T$)

Eigenvalue and Eigen vector formula

Given matrix ‘A’ find out matrix $\bar{X}$ and λ such that-

$$A\bar{X} = \lambda\bar{X}$$

A- Matrix

$\bar{X}$ - Eigen vector

λ - Eigenvalue

Ex.

$$\begin{bmatrix} 30 \\ 12 \end{bmatrix}\begin{bmatrix} 1 \\ 1 \end{bmatrix} = \begin{bmatrix} 3 \\ 3 \end{bmatrix} = 3\begin{bmatrix} 1 \\ 1 \end{bmatrix}$$

$$A\bar{X} = \lambda\bar{X}$$

$$(A-\lambda I)\bar{X} = 0$$

$\bar{X}$ can not be zero since it is direction, so

$$\det(A-\lambda I) = 0$$

Consider the example

$$\begin{pmatrix} -2 & -4 & 2 \\ -2 & 1 & 2 \\ 4 & 2 & 5 \end{pmatrix}.$$

The characteristic equation is

$$\det \begin{pmatrix} -2-\lambda & -4 & 2 \\ -2 & 1-\lambda & 2 \\ 4 & 2 & 5-\lambda \end{pmatrix} = 0.$$

Expanding the determinant,

$$(-2-\lambda)[(1-\lambda)(5-\lambda)-2\times 2]+4[(-2)\times(5-\lambda)-4\times 2]+2[(-2)\times 2-4(1-\lambda)]=0.$$

Expanding the brackets and simplifying:

$$-\lambda^3+4\lambda^2+27\lambda-90=0,$$

or, equivalently

$$\lambda^3-4\lambda^2-27\lambda+90=0.$$

meaning that the eigenvalues are 3, −5 and 6.

We now go on to solve

$$\begin{pmatrix} -2-\lambda & -4 & 2 \\ -2 & 1-\lambda & 2 \\ 4 & 2 & 5-\lambda \end{pmatrix} \begin{pmatrix} X \\ Y \\ Z \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}$$

More details:

http://wwwf.imperial.ac.uk/metric/metric_public/matrices/eigenvalues_and_eigenvectors/eigenvalues2.html

https://www.youtube.com/watch?v=cHOsd2PhkqE

Implementation

https://plot.ly/ipython-notebooks/principal-component-analysis/

https://www.analyticsvidhya.com/blog/2016/03/practical-guide-principal-component-analysis-python/