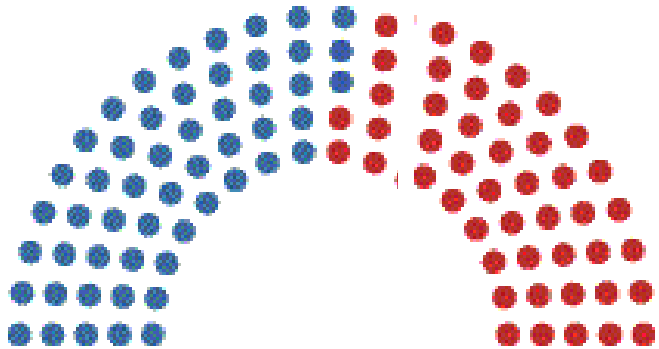
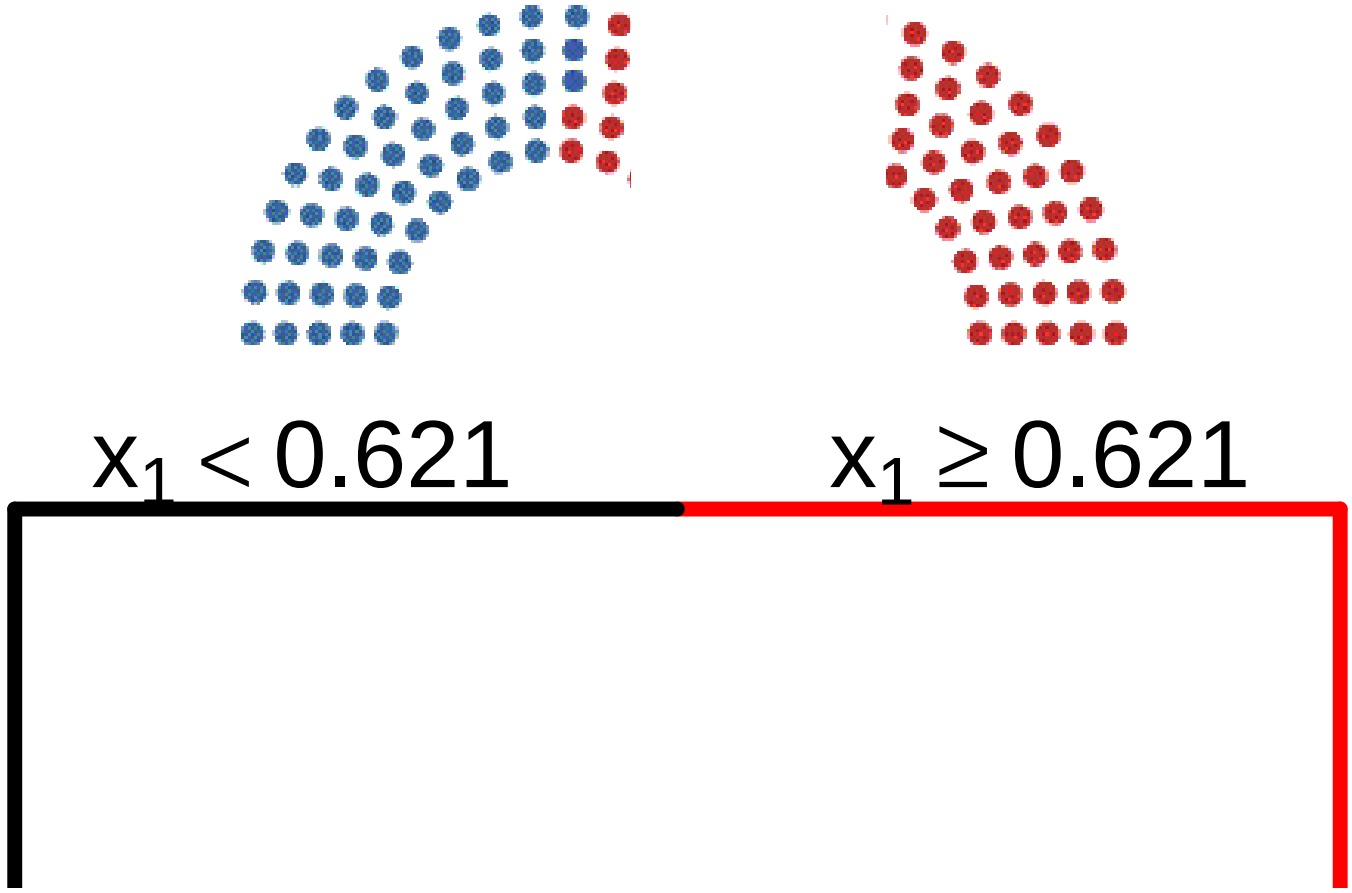
Partitioning the Senators With Trees



Idea our Senators are defined by their attributes. Suppose we (optimally) split ('partition') the Senators with respect to x_1, such that we form two subsets of our training data.

e.g suppose that Republicans generally use '`guns`' more than Democrats, such that grabbing all the observations for which $x_{guns} > 0.621$ captures, say, 80% of the Republicans in our data.

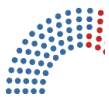
Tree, stage 1



Now, x_2 ...

$\rightarrow$ we now have two subsets of our training set: a bunch of Republicans (classified correctly based on x_1 alone)

and a subset that's still a mix of Republicans and Democrats.

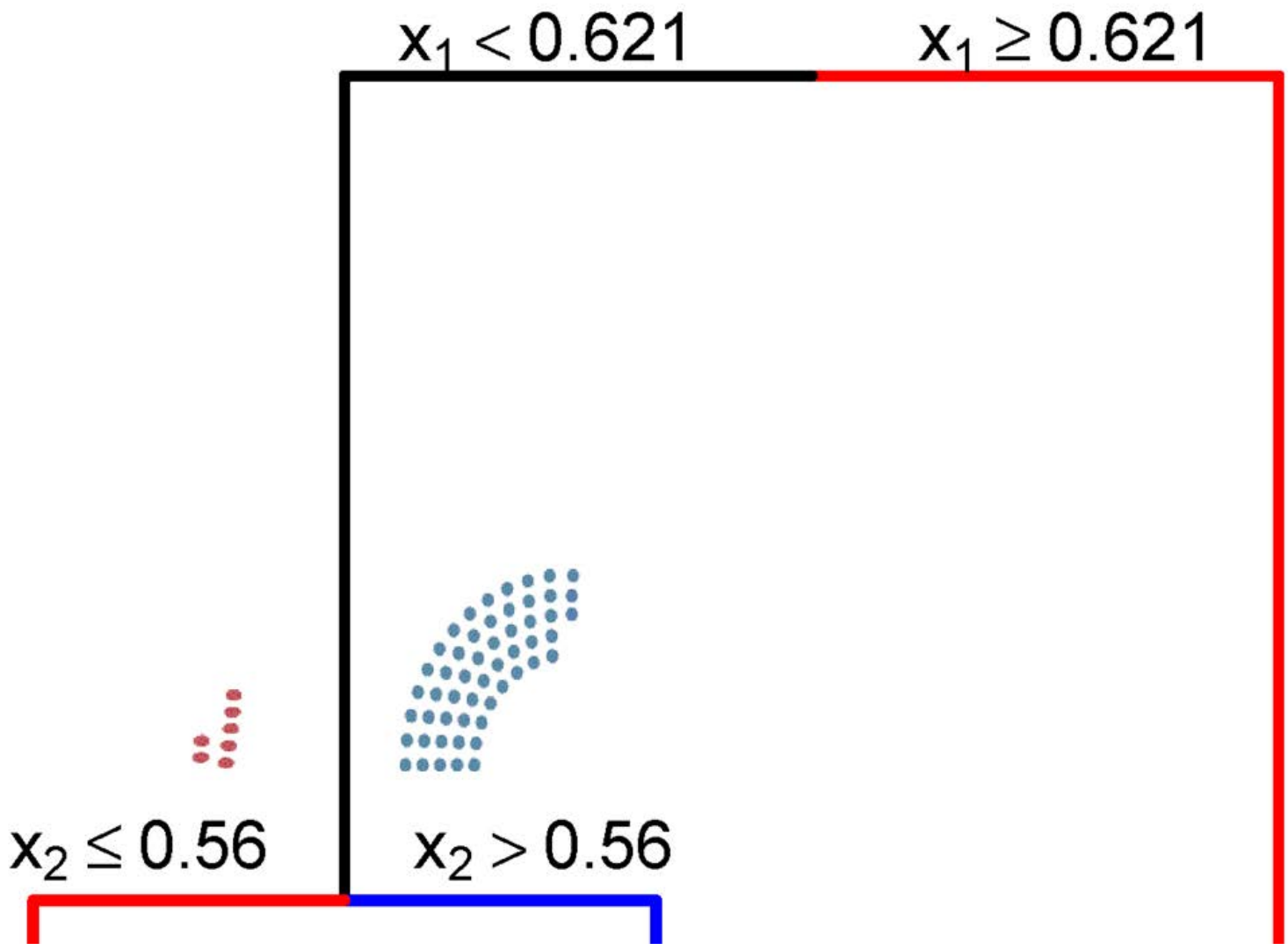


btw The set of Senators we've assigned to Republicans based on their x_1 values are called a leaf.

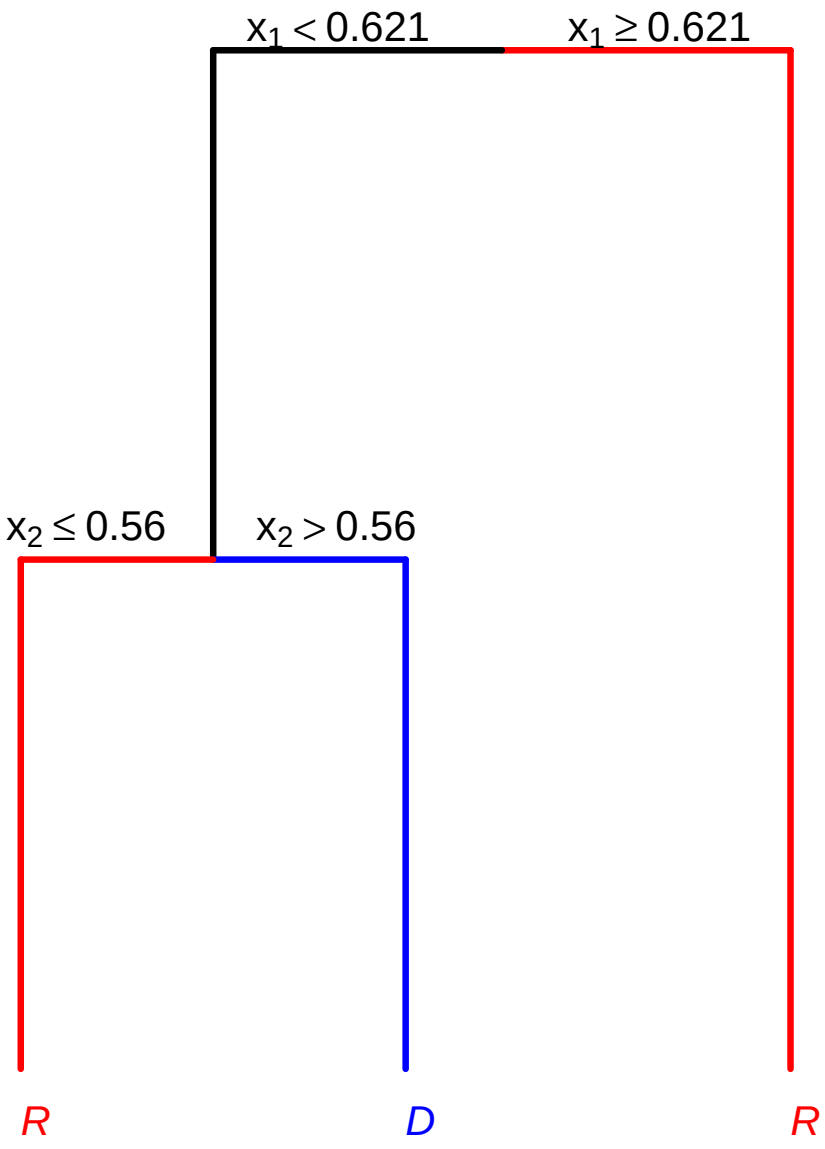
now suppose we take the mixed group remaining ('internal node') and split them based on x_2, which is their use of '`equality`'.

and it turns out that the (remaining) Republicans tend to use this less. So, when we partition according to, say, $x_2 \leq 0.56$ this enables us to perfectly divide this remaining subset into Democrats and Republicans.

Tree, stage 2



Complete Tree



Notes

This classifier is known as a tree, and the recursive partitioning on each derived subset continues until further splits doesn't add 'much' to our classification ability.

→ typically not the case that we can (or want to) classify perfectly into the leaves!

At each node, algorithmic tricks allow fast searching over all the variables to find the one that should be used.

and clearly need a metric for 'best' split in given x: typically based on how homogenous the resulting subset of the data is

e.g. 'Gini impurity' and 'Variance Reduction'

\+ Trees are easy to interpret, and we can report relative variable importance statistics.

But...

These basic CART (Classification and Regression Trees) approaches show instability in practice:

i.e. minor changes to training data can have large consequences for classification decisions, because any error is propagated down the tree.

→ related to problems of overfitting in the training data.

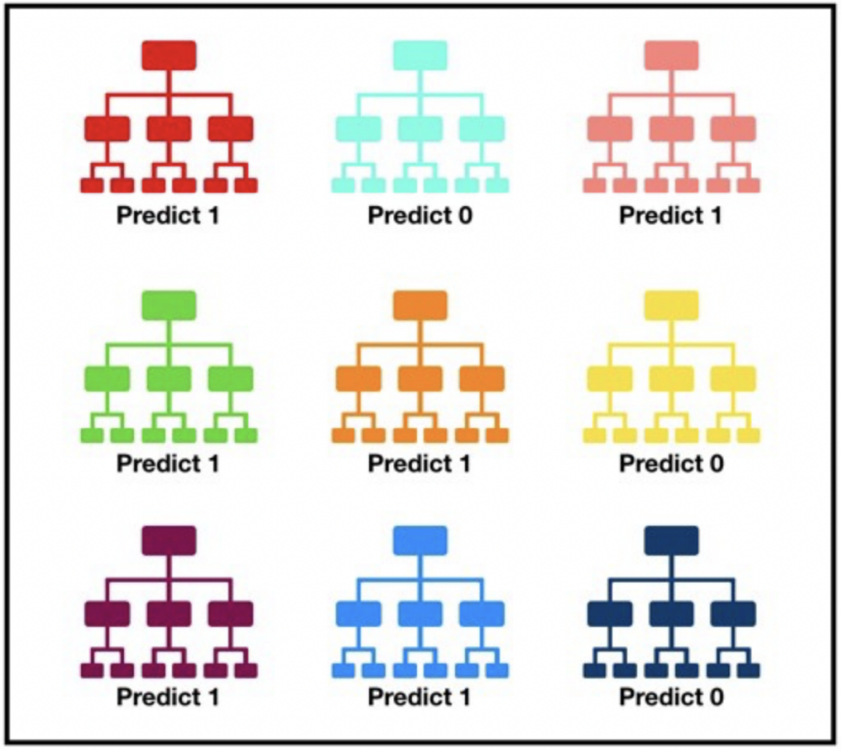
So effort is made to prune the trees back—remove less helpful branches.

Or can construct many trees (from slightly different samples of the data) and average over them: known as bagging ('bootstrap aggregating').

→ random forests combines *many* trees (forest) and at each split a *random sample* of features is considered (rather than all features).

Random Forest Classifier

- Generate many trees and then let them vote.



Tally: Six 1s and Three 0s

Prediction: 1

Assumptions

1. There needs to be some actual signal in our features so that models built using those features do better than random guessing.

2. The predictions (and therefore the errors) made by the individual trees need to have low correlations with each other.

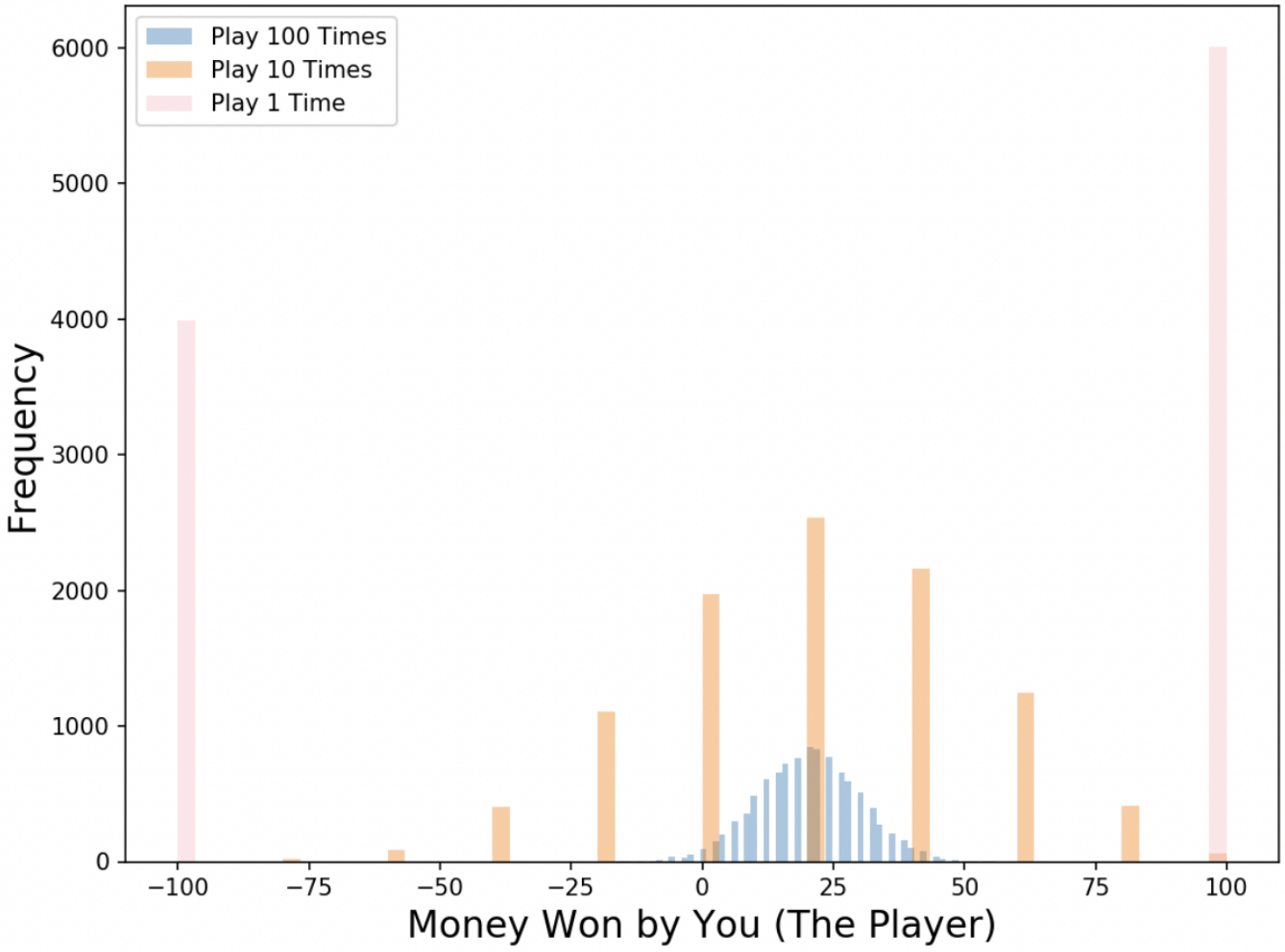
Uncorrelated Outcomes and Random Sampling are Good Things. . .

- I use a uniformly distributed random number generator to produce a number.
- If the number I generate is greater than or equal to 40, you win (so you have a 60% chance of victory) and I pay you some money. If it is below 40, I win and you pay me the same amount.
- Now I offer you the the following choices. We can either:

1. **Game 1** — play 100 times, betting $1 each time.
2. **Game 2**— play 10 times, betting $10 each time.
3. **Game 3**— play one time, betting $100.

Which would you pick? The expected value of each game is the same:

> *Expected Value Game 1 = (0.60*1 + 0.40*-1)*100 = 20*
>
> *Expected Value Game 2= (0.60*10 + 0.40*-10)*10 = 20*
>
> *Expected Value Game 3= 0.60*100 + 0.40*-100 = 20*



Outcome Distribution of 10,000 Simulations for each Game

Tree Bagging and Random Forests

- Bagging—bootstrap aggregating—rests on observation that trees fit to different subsets of data (observations) will give different predictions.
- If those trees are independent, then we get a low bias, low variance estimate of the true response. So, grow the trees (fully) and just take an average over the M samples:
- In practice, this can result in quite correlated trees, so random forests does splits of random subset of variables at each node in a tree.

Unsupervised Learning

Unsupervised vs Supervised Learning:

- Most of this course focuses on *supervised learning* methods such as regression and classification.
- In that setting we observe both a set of features $X_1, X_2, \ldots, X_p$ for each object, as well as a response or outcome variable Y. The goal is then to predict Y using $X_1, X_2, \ldots, X_p$.
- Here we instead focus on *unsupervised learning*, we where observe only the features $X_1, X_2, \ldots, X_p$. We are not interested in prediction, because we do not have an associated response variable Y.

The Goals of Unsupervised Learning

- The goal is to discover interesting things about the measurements: is there an informative way to visualize the data? Can we discover subgroups among the variables or among the observations?
- We discuss two methods:
  - *principal components analysis*, a tool used for data visualization or data pre-processing before supervised techniques are applied, and
  - *clustering*, a broad class of methods for discovering unknown subgroups in data.

The Challenge of Unsupervised Learning

- Unsupervised learning is more subjective than supervised learning, as there is no simple goal for the analysis, such as prediction of a response.
- But techniques for unsupervised learning are of growing importance in a number of fields:
  - subgroups of breast cancer patients grouped by their gene expression measurements,
  - groups of shoppers characterized by their browsing and purchase histories,
  - movies grouped by the ratings assigned by movie viewers.

Another advantage

- It is often easier to obtain *unlabeled data* — from a lab instrument or a computer — than *labeled data*, which can require human intervention.
- For example it is difficult to automatically assess the overall sentiment of a movie review: is it favorable or not?

Principal Components Analysis

- PCA produces a low-dimensional representation of a dataset. It finds a sequence of linear combinations of the variables that have maximal variance, and are mutually uncorrelated.
- Apart from producing derived variables for use in supervised learning problems, PCA also serves as a tool for data visualization.

Clustering

- *Clustering* refers to a very broad set of techniques for finding *subgroups*, or *clusters*, in a data set.
- We seek a partition of the data into distinct groups so that the observations within each group are quite similar to each other,
- It make this concrete, we must define what it means for two or more observations to be *similar* or *different*.
- Indeed, this is often a domain-specific consideration that must be made based on knowledge of the data being studied.

PCA vs Clustering

- PCA looks for a low-dimensional representation of the observations that explains a good fraction of the variance.
- Clustering looks for homogeneous subgroups among the observations.

Principal Components Analysis

PCA: Introduction

Possibly oldest multivariate technique (Pearson, 1901?)

Very popular for data summary, exploration (and analysis?)

Aims:

- extract core/important information from data
- reduce the data/problem down to this information
- simplify data
- analyze data in terms of its patterns/groups

Generally: represent this information as new (and smaller number of) variables known as *principal components*

Overview

Features: these principal components will be uncorrelated (orthogonal) with (to) each other, will be linear combinations of original variables

Result: lower dimensional 'map' of observations in new space:

$\rightarrow$ each observation now has a value on each principal component called its (factor) score, which are projections of (original) observations onto the PCs

Interpretation of given PC: depends on correlation between component and (original) variables—known as loading

Method: (eigen-) decomposition of cov matrix or singular value decomposition of data matrix

Let's see how it works. . .

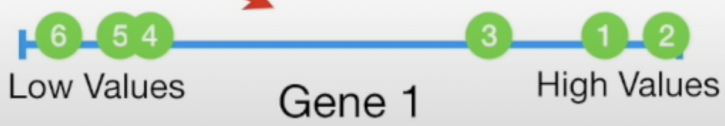
- Here is a link to a good tutorial.

	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

Let's start with a simple data set.

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

If we only measure 1 gene, we can plot the data on a number line…



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

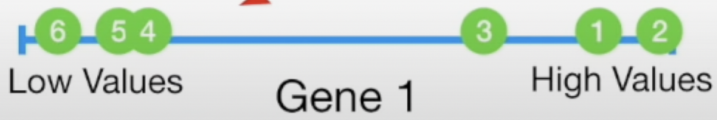
Mice 1, 2 and 3 have relatively high values…

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

…and mice 4, 5 and 6 have relatively low values.

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

Even though it's a simple graph, it shows us that mice 1, 2 and 3 are more similar to each other than they are to mice 4, 5 6.

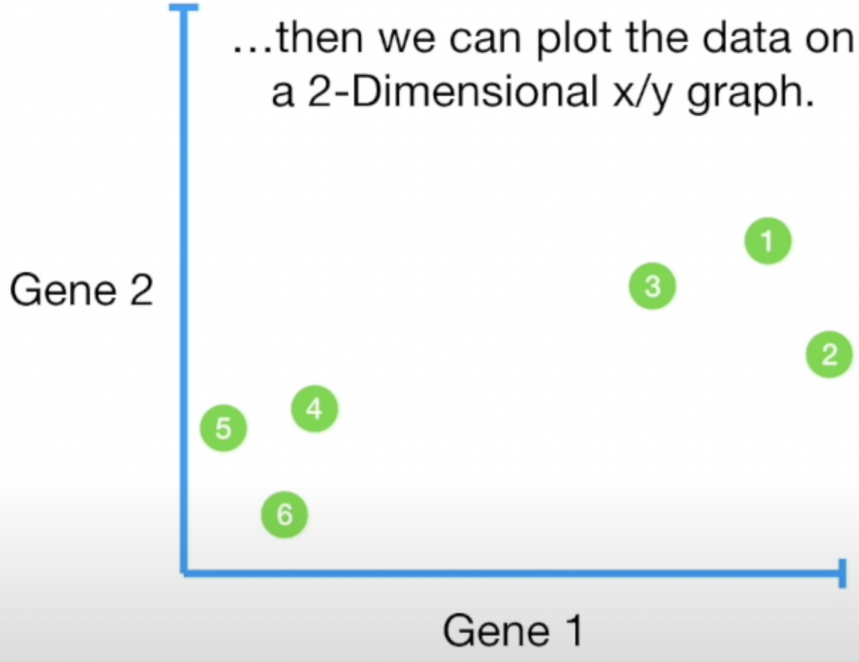


	Mouse 1	Mouse 2	Mouse 3	Mouse 4	Mouse 5	Mouse 6
Gene 1	10	11	8	3	2	1
Gene 2	6	4	5	3	2.8	1

If we measured 2 genes…

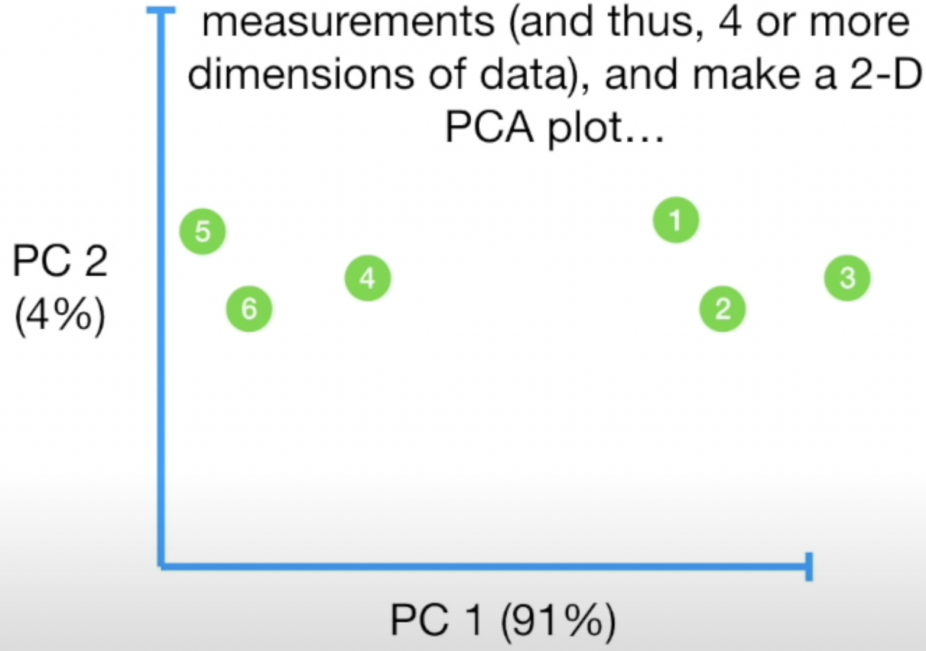
	Mouse 1	Mouse 2	Mouse 3	Mouse 4	Mouse 5	Mouse 6
Gene 1	10	11	8	3	2	1
Gene 2	6	4	5	3	2.8	1

…then we can plot the data on a 2-Dimensional x/y graph.

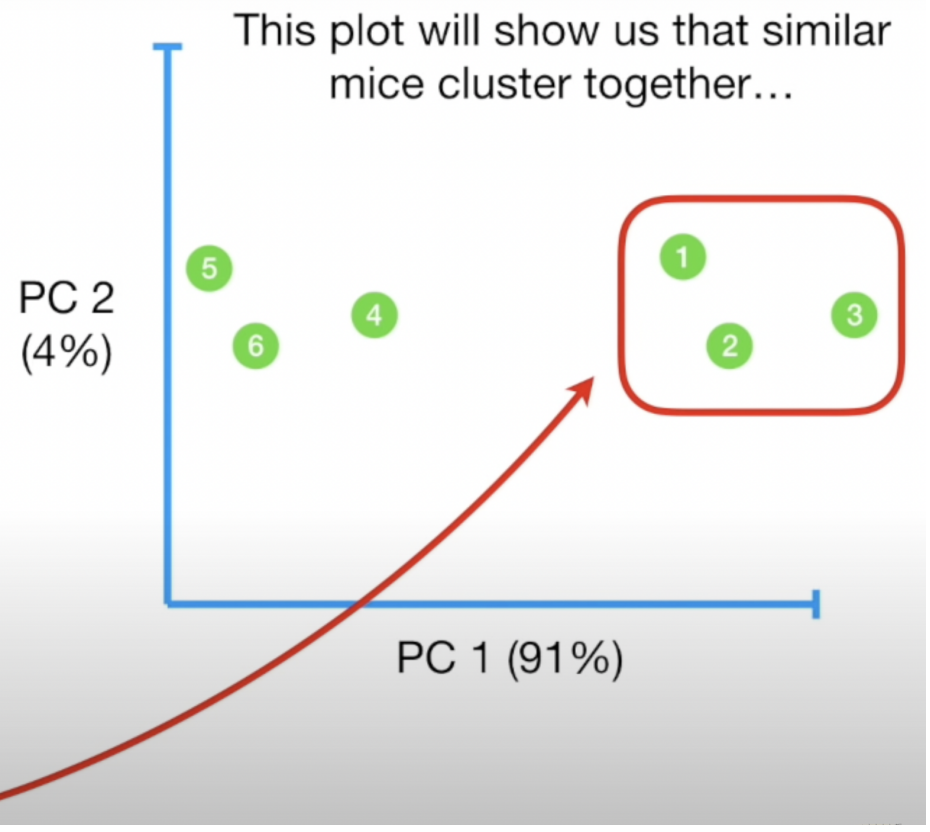


	Mouse 1	Mouse 2	Mouse 3	Mouse 4	Mouse 5	Mouse 6
Gene 1	10	11	8	3	2	1
Gene 2	6	4	5	3	2.8	1
Gene 3	12	9	10	2.5	1.3	2
Gene 4	5	7	6	2	4	7

So we're going to talk about how PCA can take 4 or more gene measurements (and thus, 4 or more dimensions of data), and make a 2-D PCA plot…

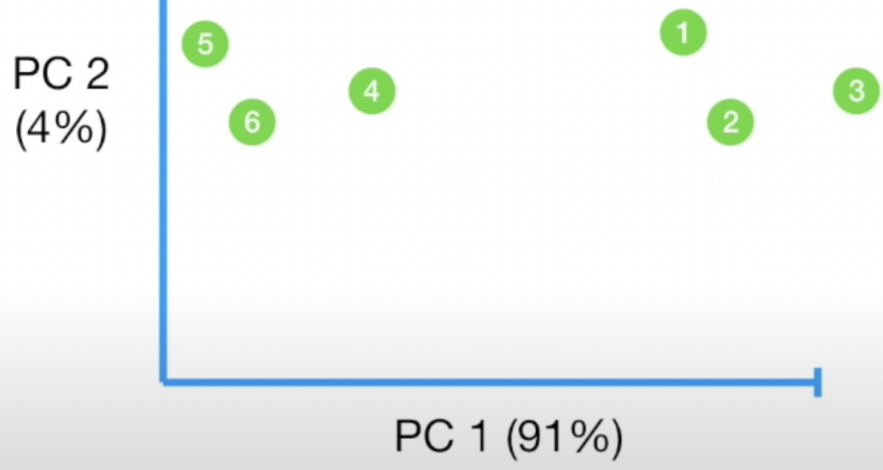


	Mouse 1	Mouse 2	Mouse 3	Mouse 4	Mouse 5	Mouse 6
Gene 1	10	11	8	3	2	1
Gene 2	6	4	5	3	2.8	1
Gene 3	12	9	10	2.5	1.3	2
Gene 4	5	7	6	2	4	7



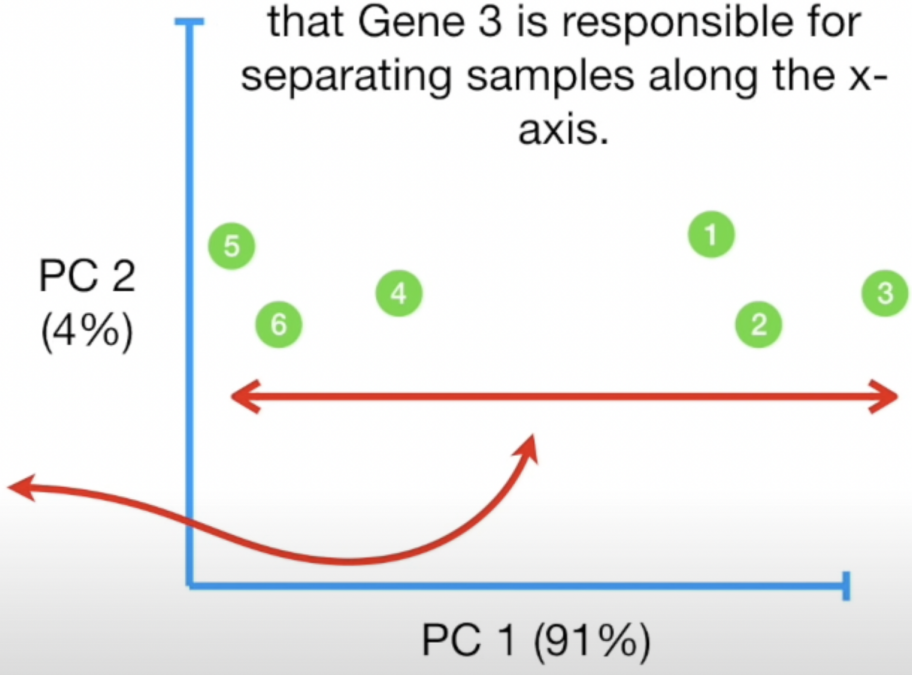
	Mouse 1	Mouse 2	Mouse 3	Mouse 4	Mouse 5	Mouse 6
Gene 1	10	11	8	3	2	1
Gene 2	6	4	5	3	2.8	1
Gene 3	12	9	10	2.5	1.3	2
Gene 4	5	7	6	2	4	7

…We'll also talk about how PCA can tell us which gene (or variable) is the most valuable for clustering the data.



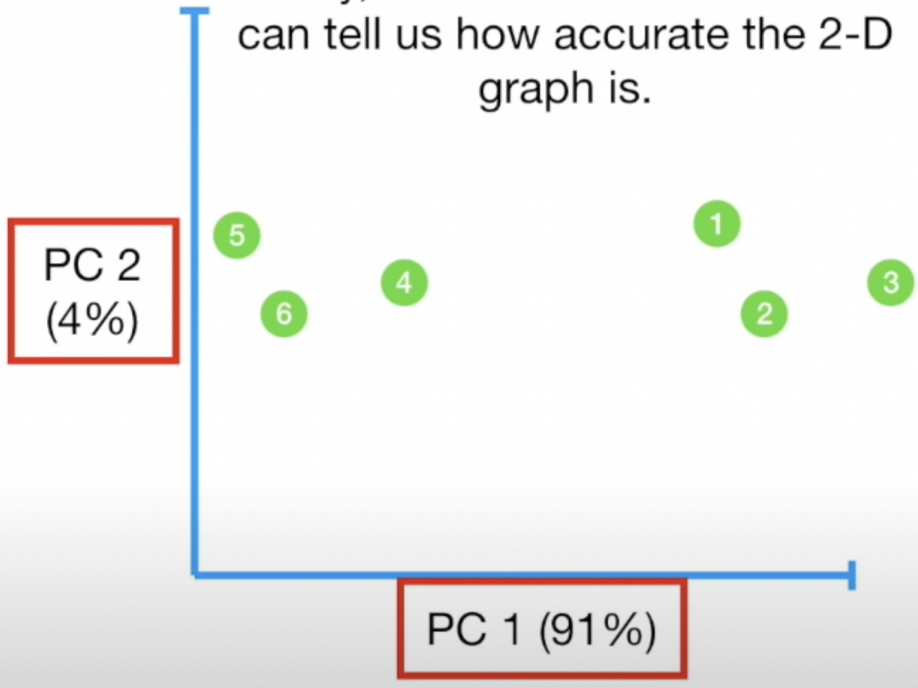
	Mouse 1	Mouse 2	Mouse 3	Mouse 4	Mouse 5	Mouse 6
Gene 1	10	11	8	3	2	1
Gene 2	6	4	5	3	2.8	1
Gene 3	12	9	10	2.5	1.3	2
Gene 4	5	7	6	2	4	7

For example, PCA might tell us that Gene 3 is responsible for separating samples along the x-axis.



	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

Lastly, we'll talk about how PCA can tell us how accurate the 2-D graph is.

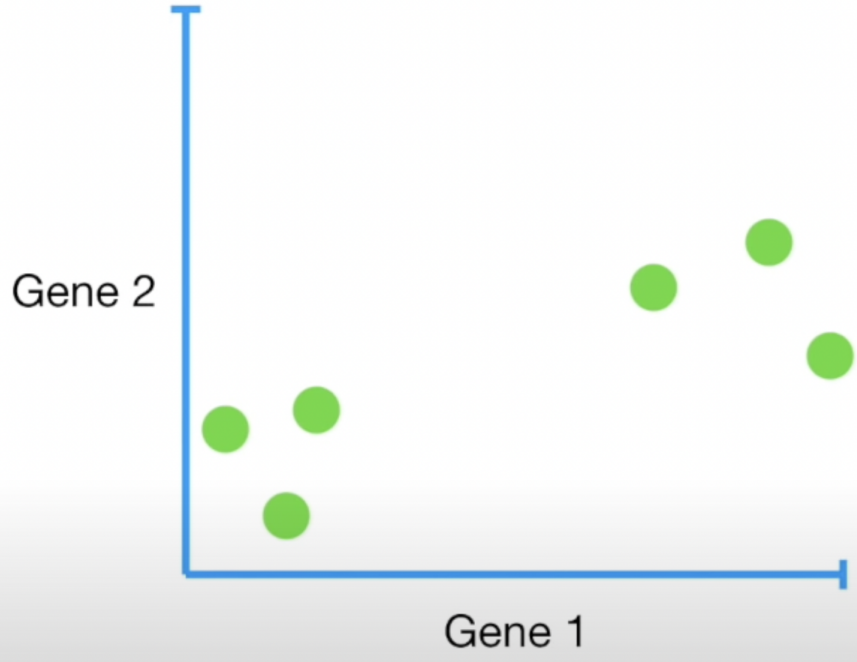


	Mouse 1	Mouse 2	Mouse 3	Mouse 4	Mouse 5	Mouse 6
Gene 1	10	11	8	3	2	1
Gene 2	6	4	5	3	2.8	1

To understand what PCA does and how it works, let's go back to the dataset that only had 2 genes…

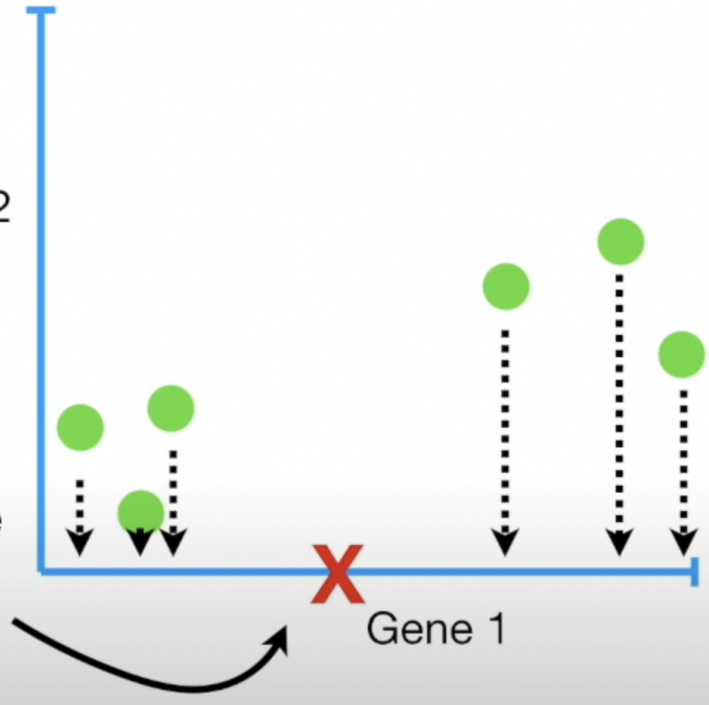
	Mouse 1	Mouse 2	Mouse 3	Mouse 4	Mouse 5	Mouse 6
Gene 1	10	11	8	3	2	1
Gene 2	6	4	5	3	2.8	1

We'll start by plotting the data…

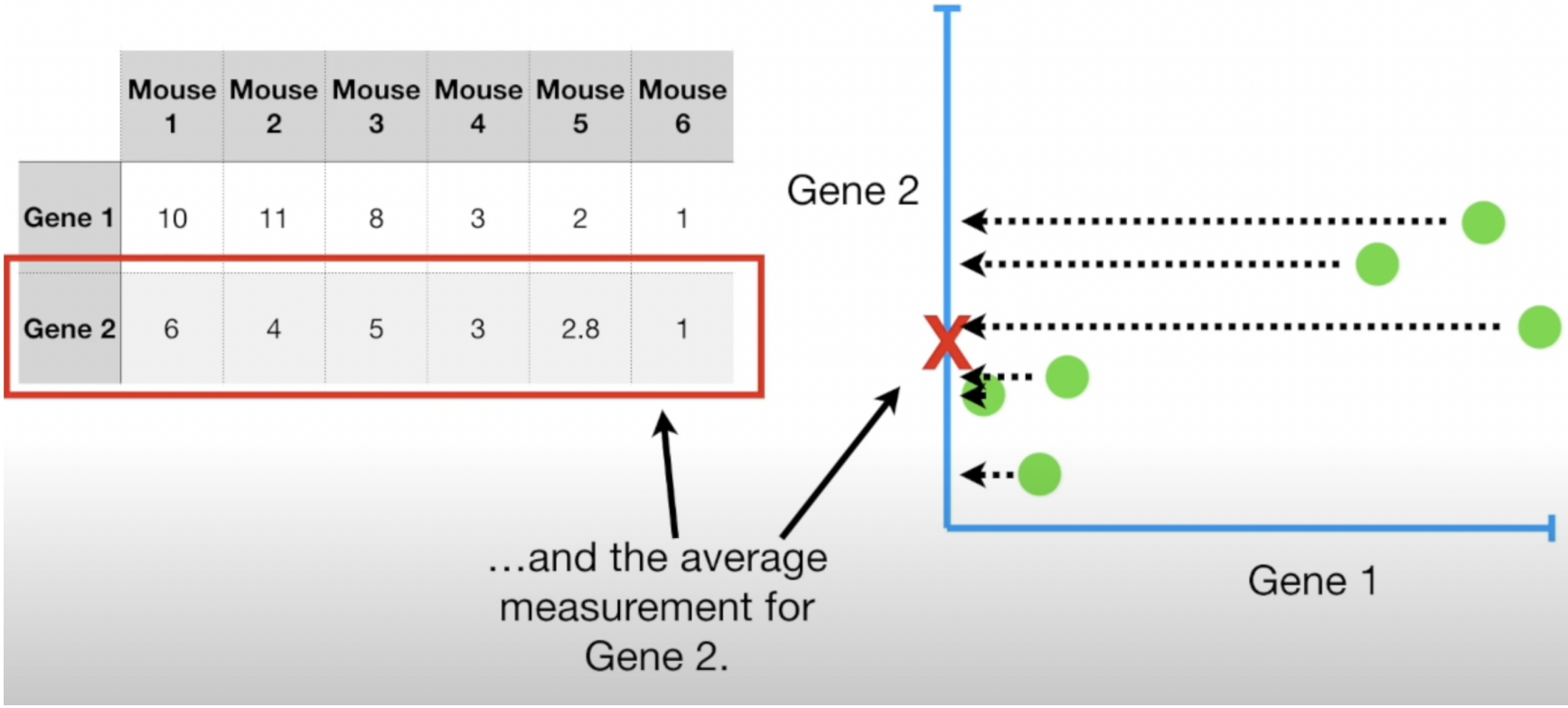


	Mouse 1	Mouse 2	Mouse 3	Mouse 4	Mouse 5	Mouse 6
Gene 1	10	11	8	3	2	1
Gene 2	6	4	5	3	2.8	1

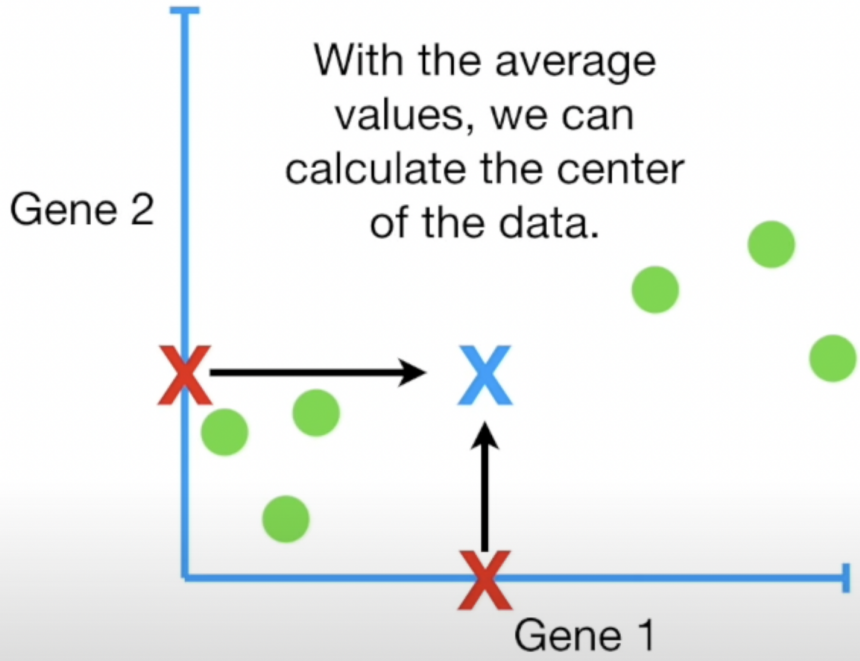
Then we'll calculate the average measurement for Gene 1…

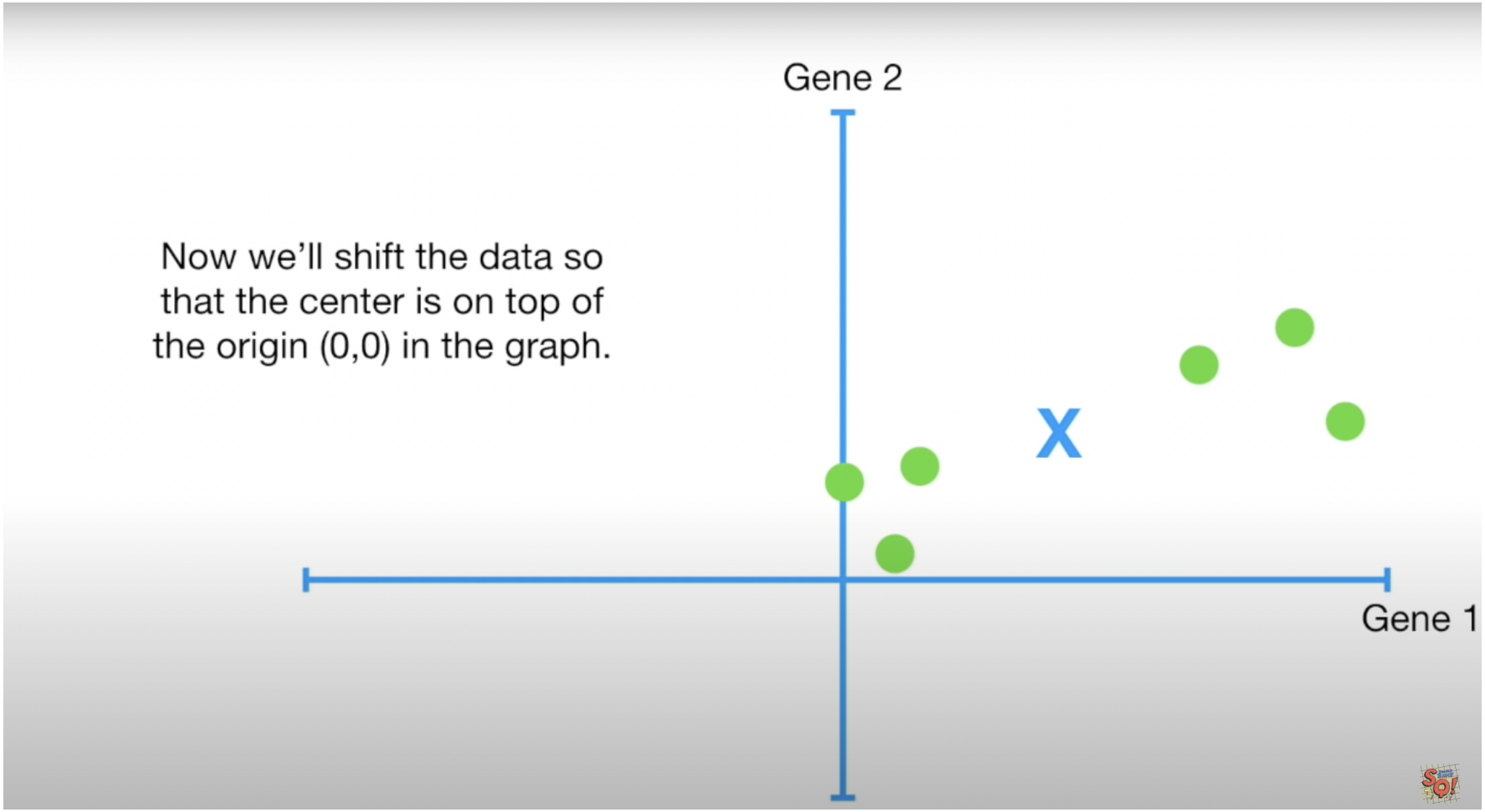


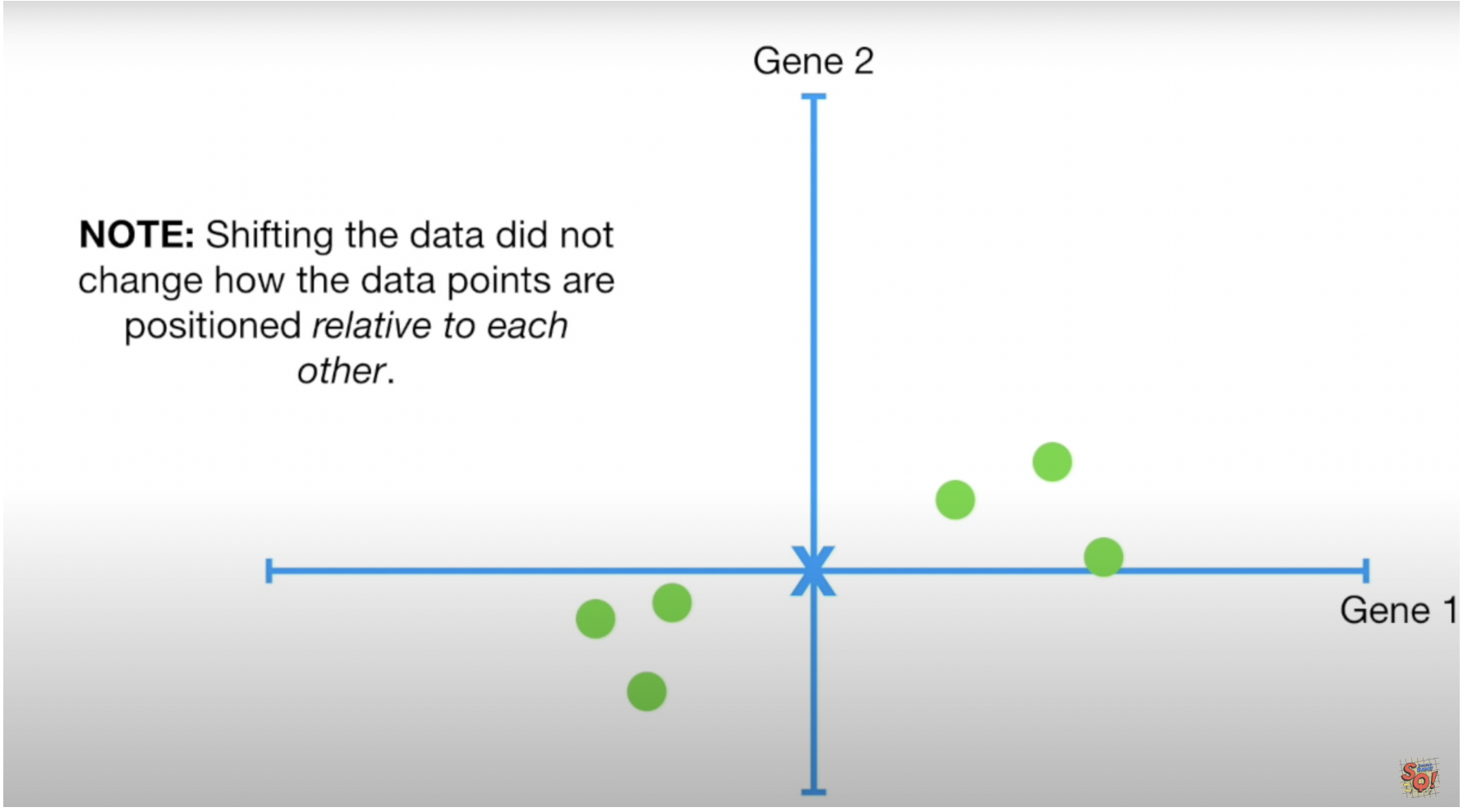
	Mouse 1	Mouse 2	Mouse 3	Mouse 4	Mouse 5	Mouse 6
Gene 1	10	11	8	3	2	1
Gene 2	6	4	5	3	2.8	1

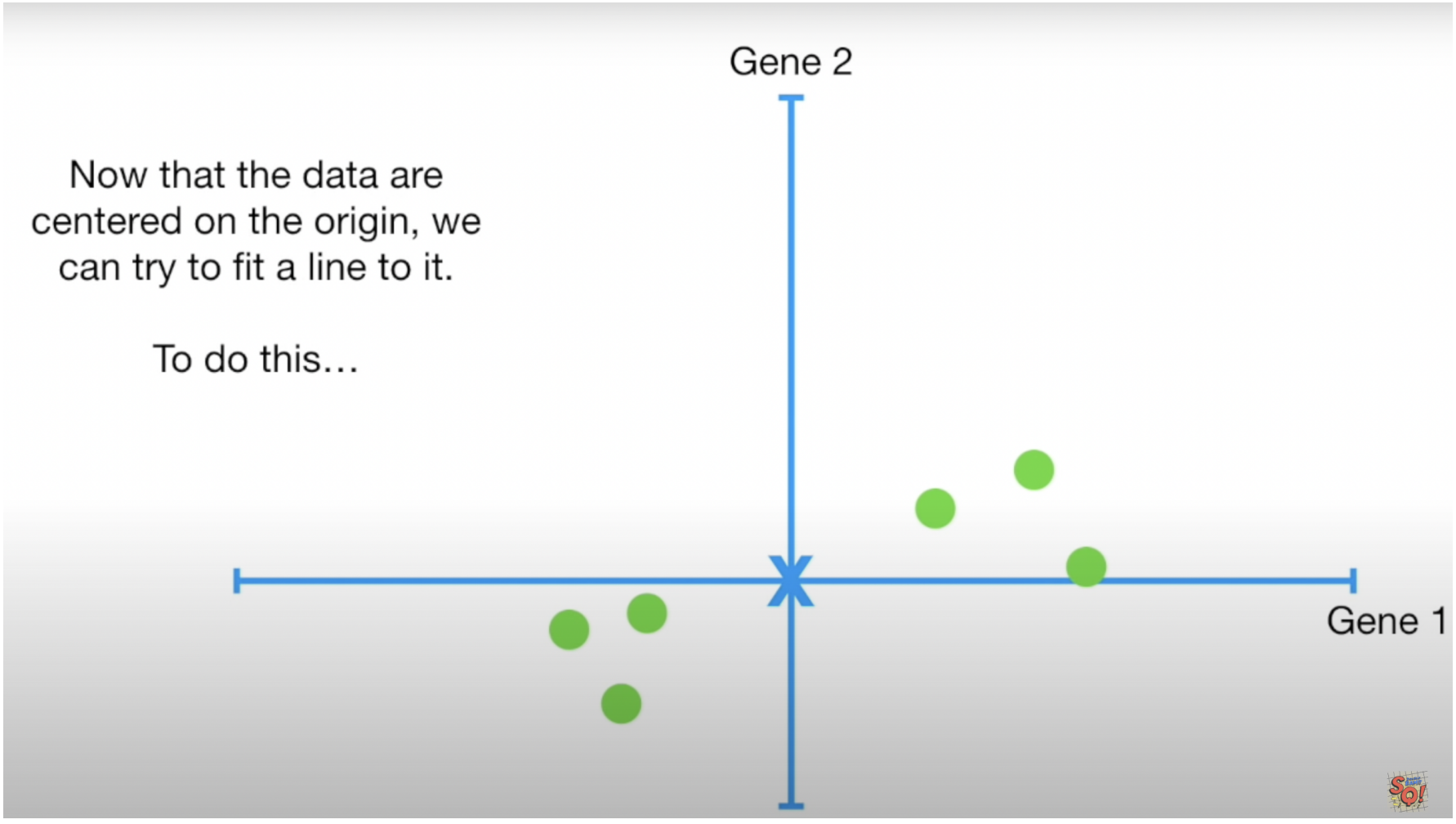


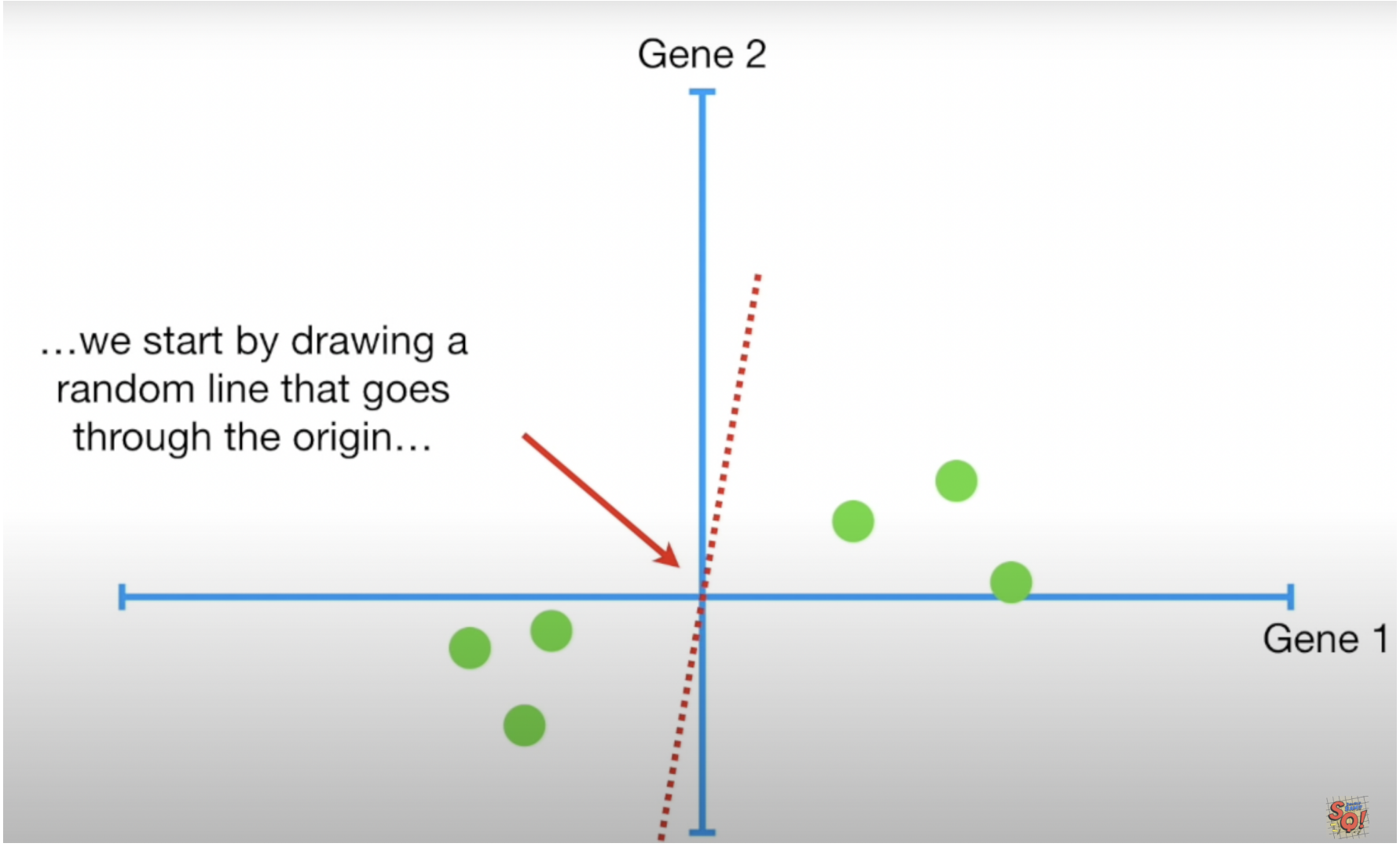
	Mouse 1	Mouse 2	Mouse 3	Mouse 4	Mouse 5	Mouse 6
Gene 1	10	11	8	3	2	1
Gene 2	6	4	5	3	2.8	1

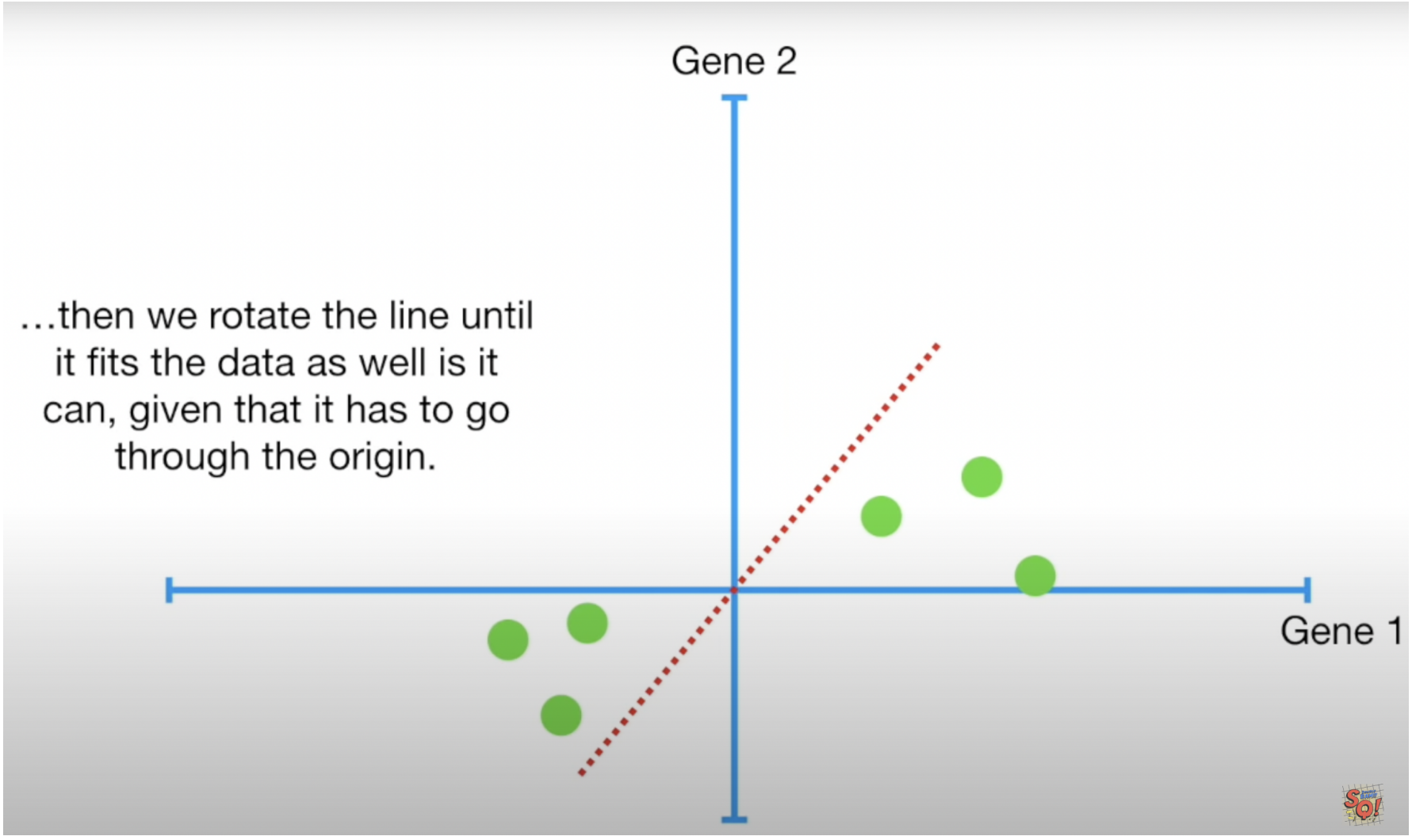


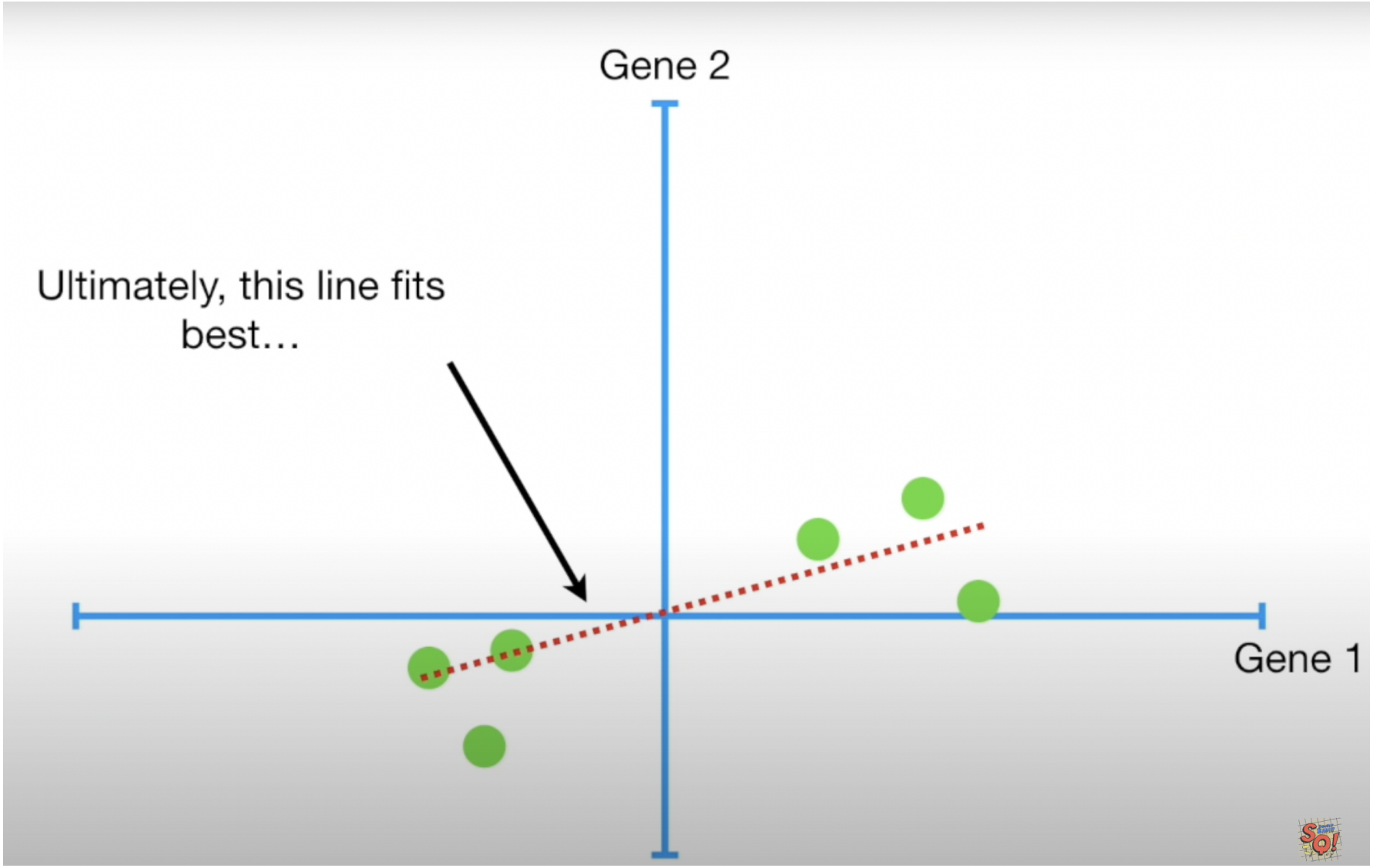

Now we'll shift the data so that the center is on top of the origin (0,0) in the graph.
Gene 2
Gene 1

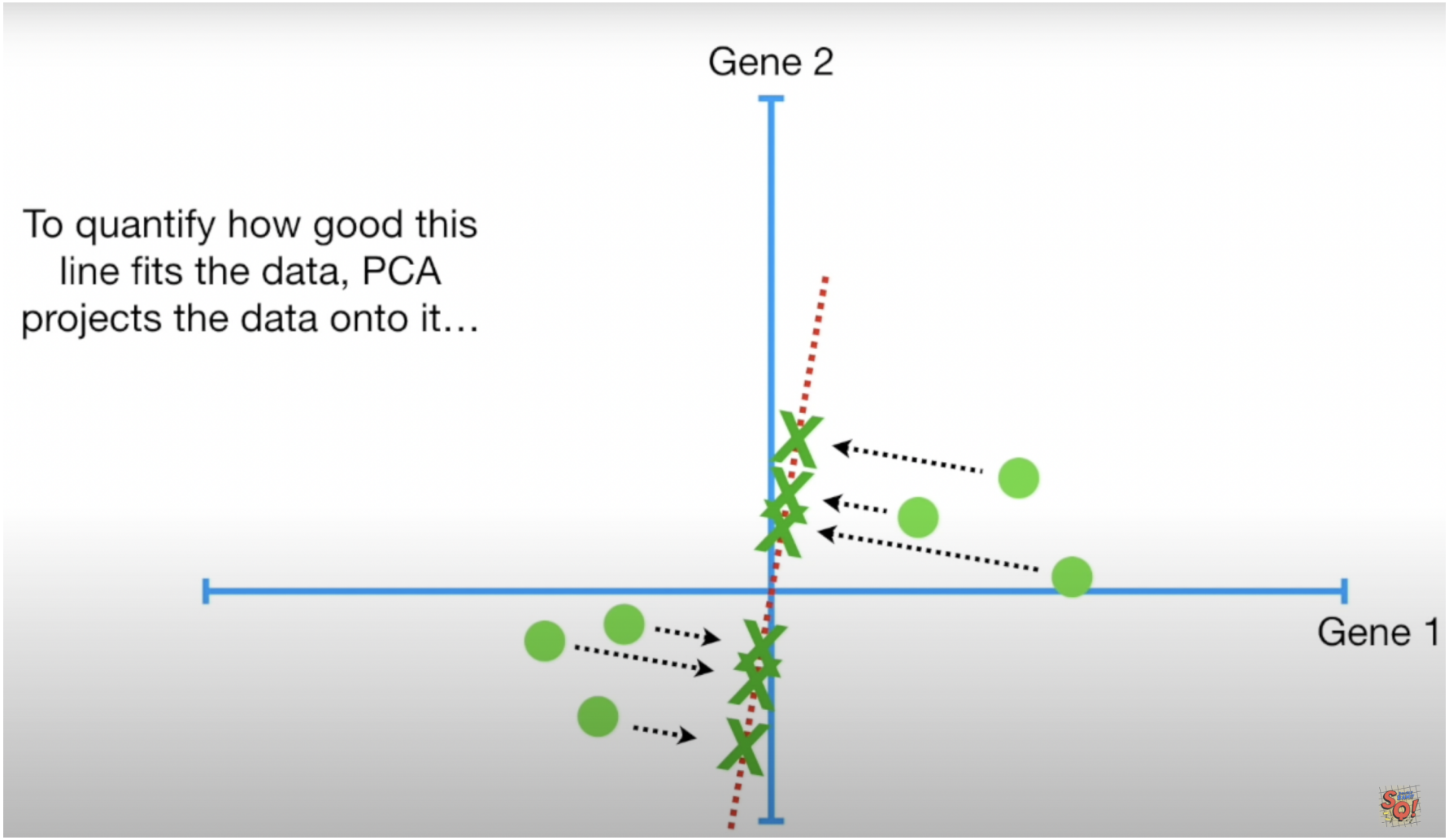

Gene 2
NOTE: Shifting the data did not change how the data points are positioned relative to each other.
Gene 1

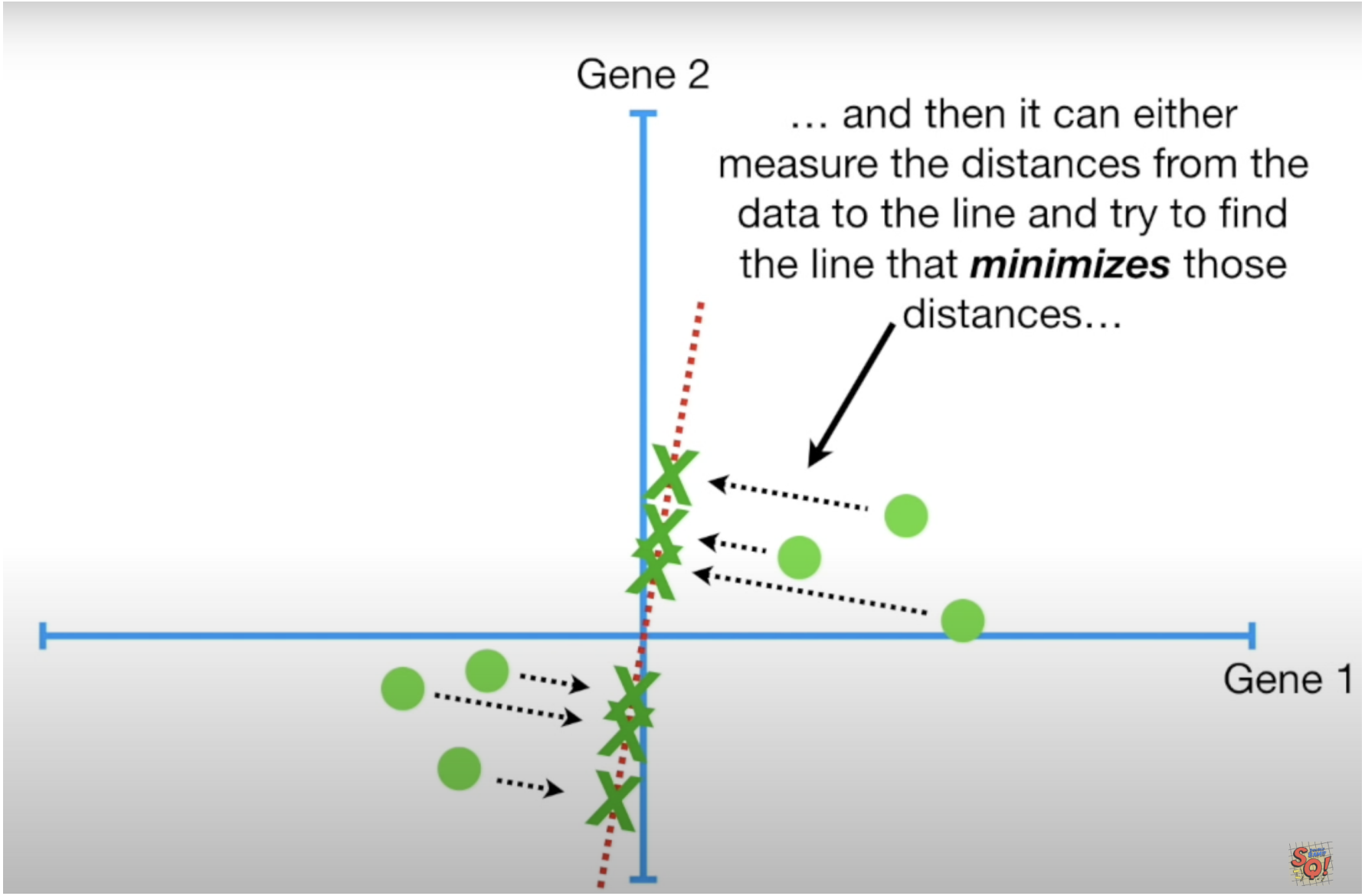

Now that the data are centered on the origin, we can try to fit a line to it.
To do this…
Gene 2
Gene 1

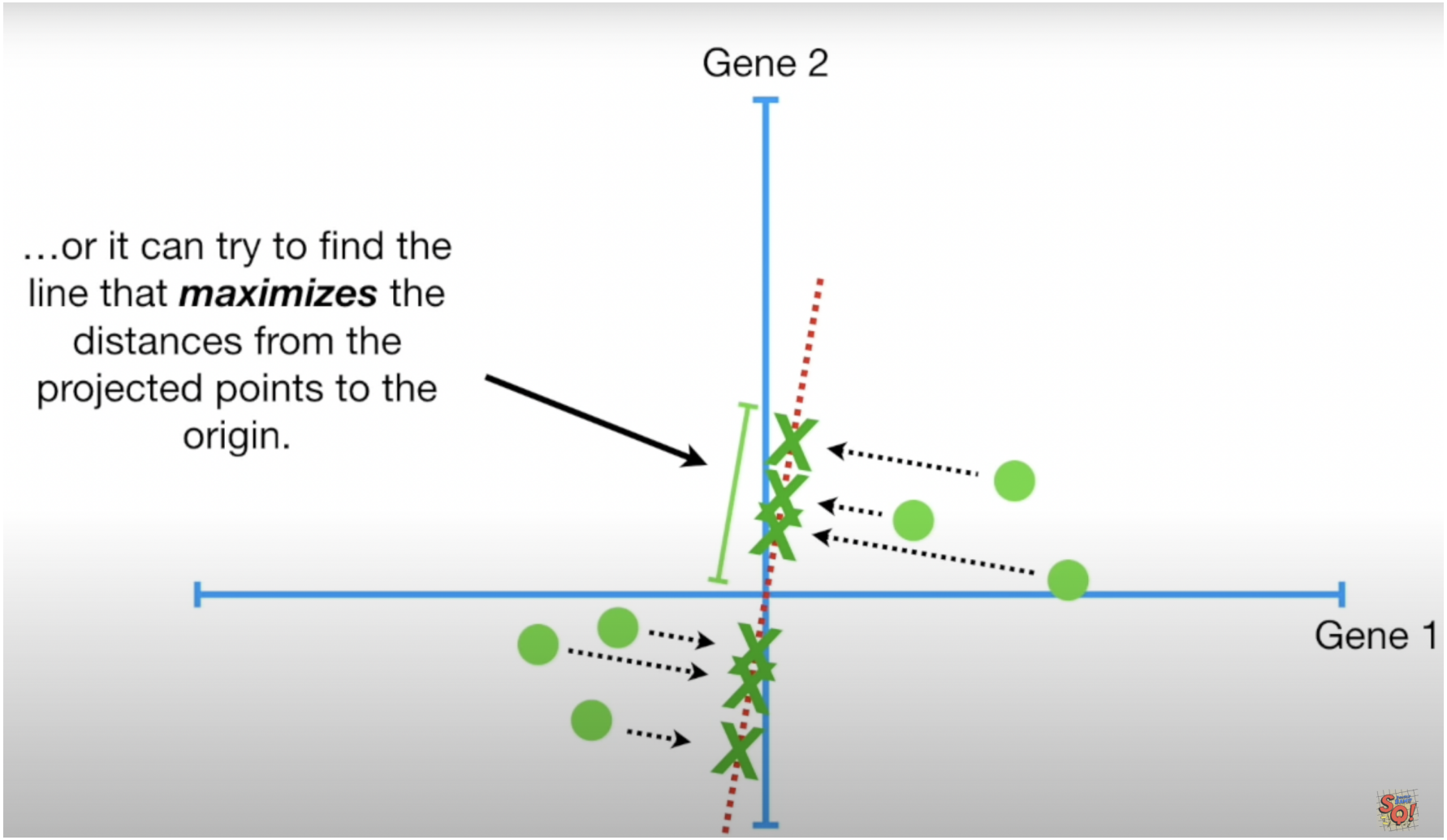

Gene 2
…we start by drawing a random line that goes through the origin…
Gene 1


Gene 2
…then we rotate the line until it fits the data as well is it can, given that it has to go through the origin.
Gene 1

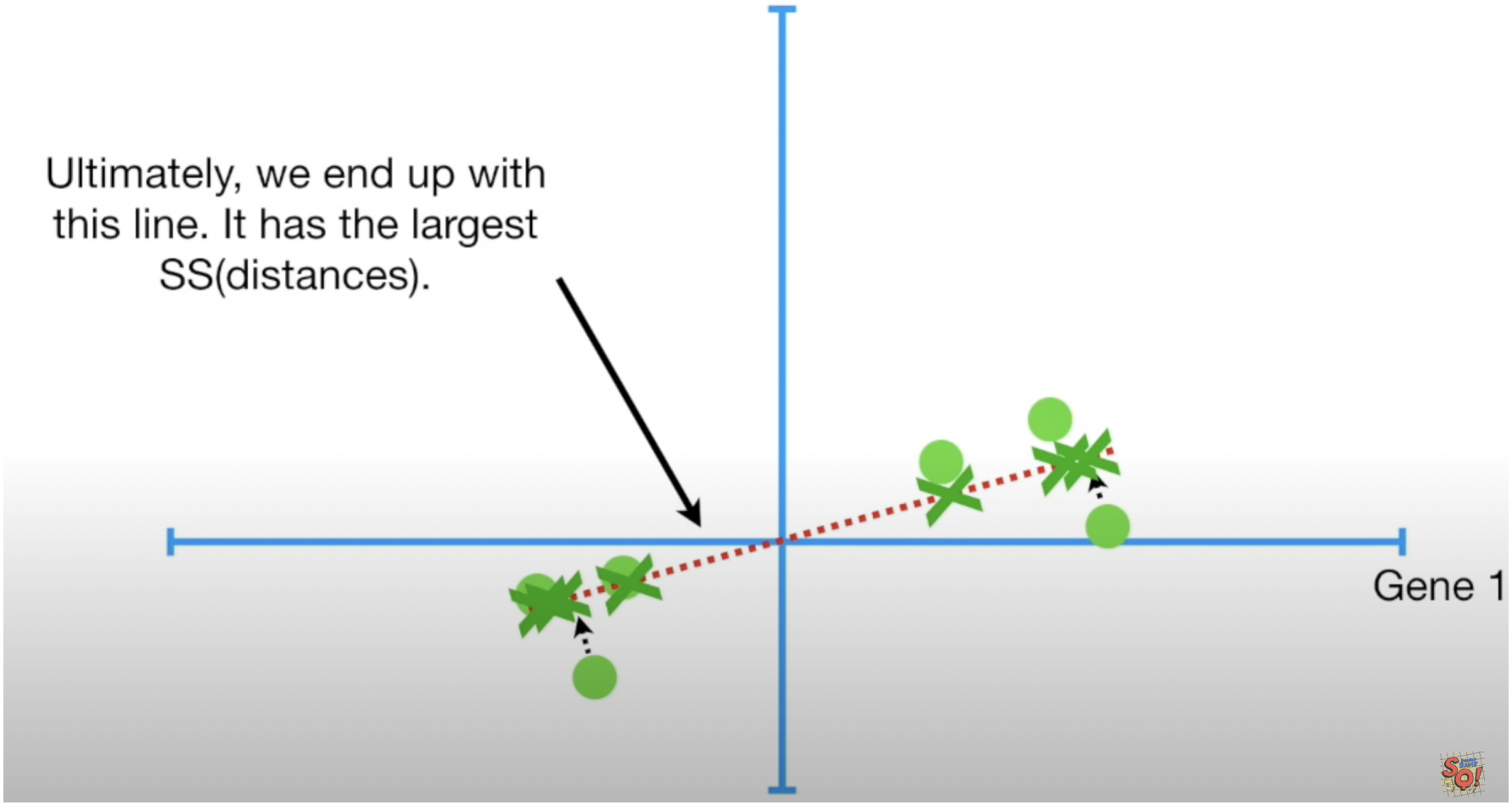

Gene 2
Ultimately, this line fits best…
Gene 1

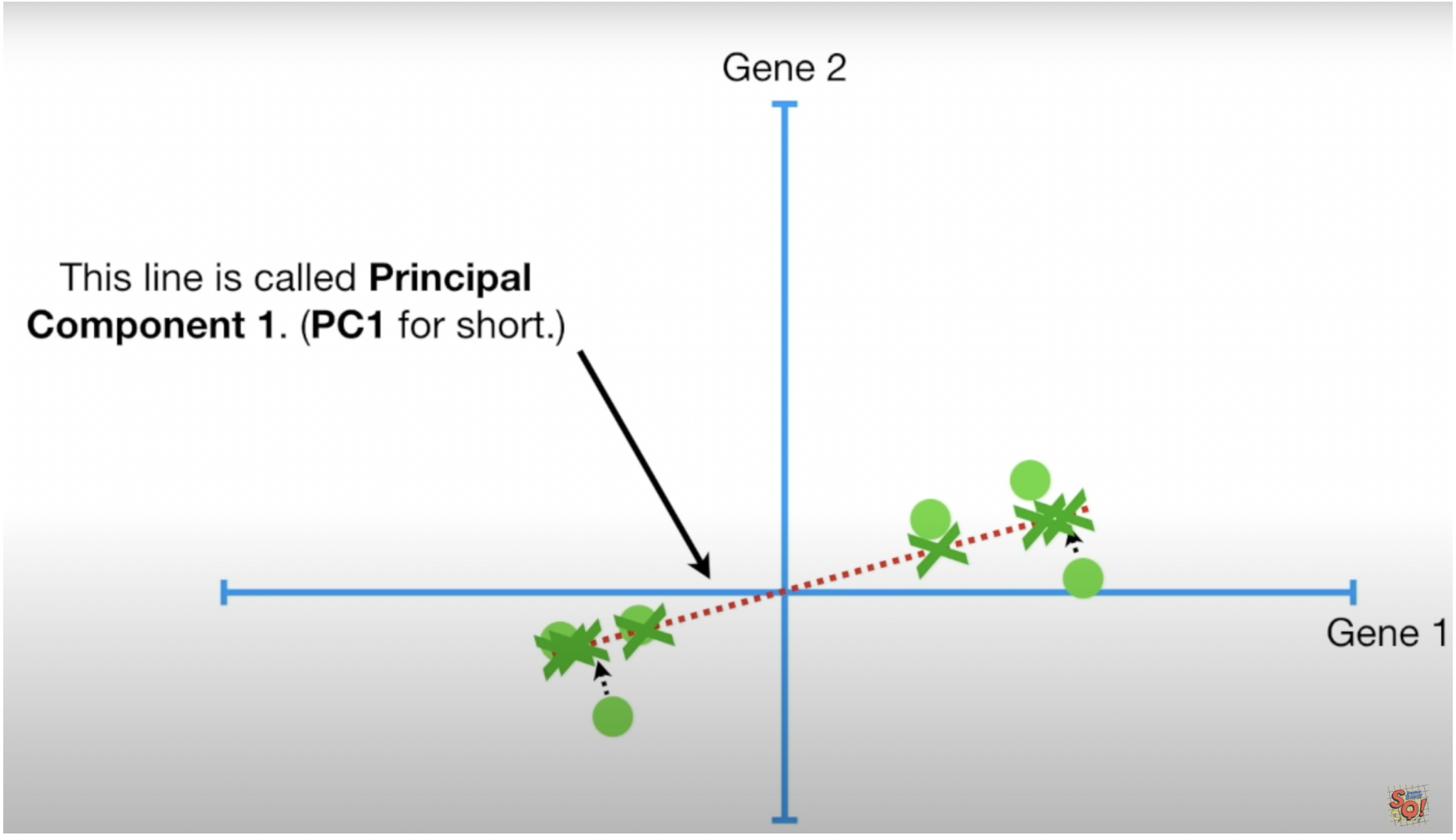

To quantify how good this line fits the data, PCA projects the data onto it…
Gene 2
Gene 1

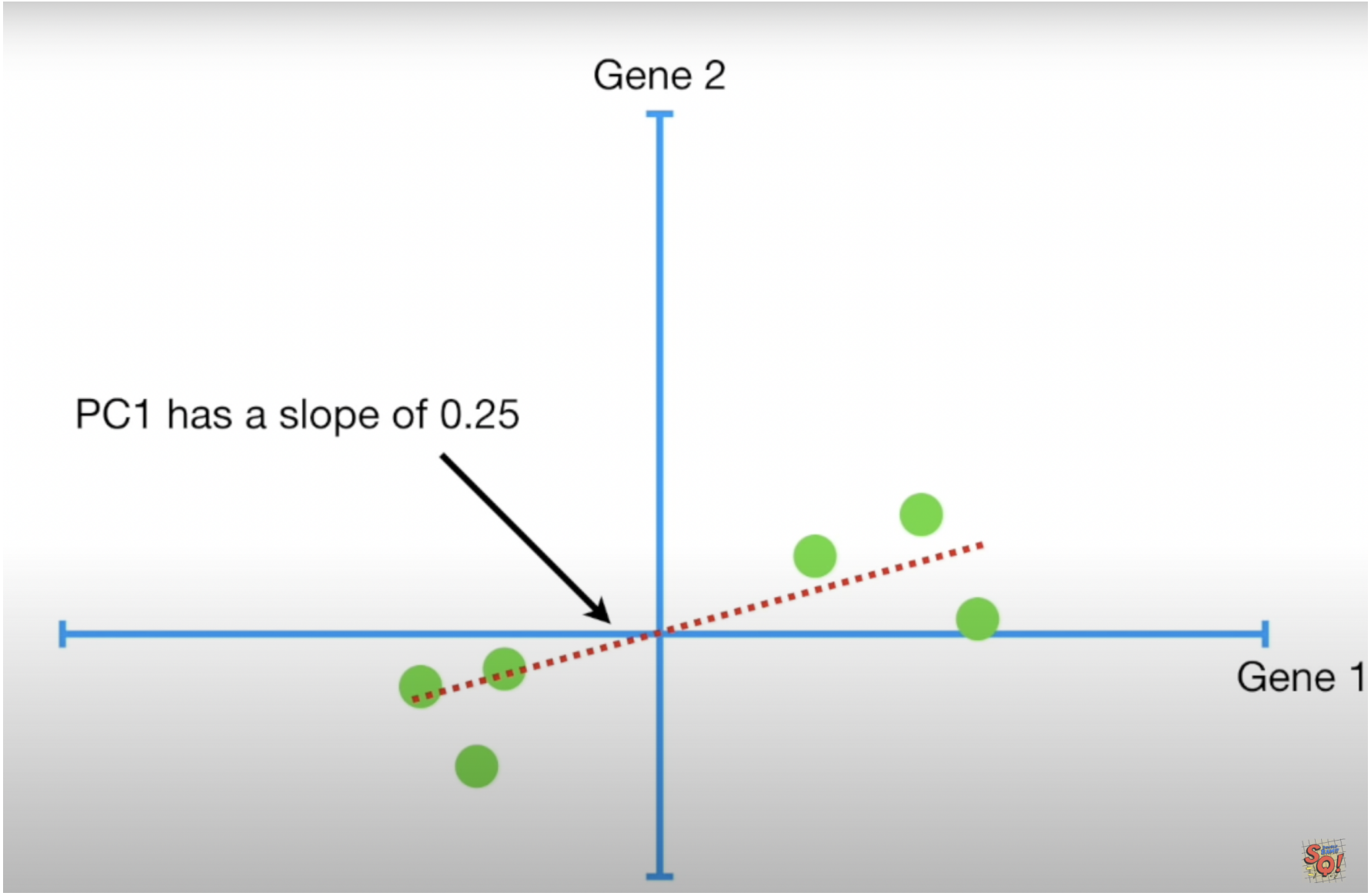

Gene 2
… and then it can either measure the distances from the data to the line and try to find the line that *minimizes* those distances…
Gene 1

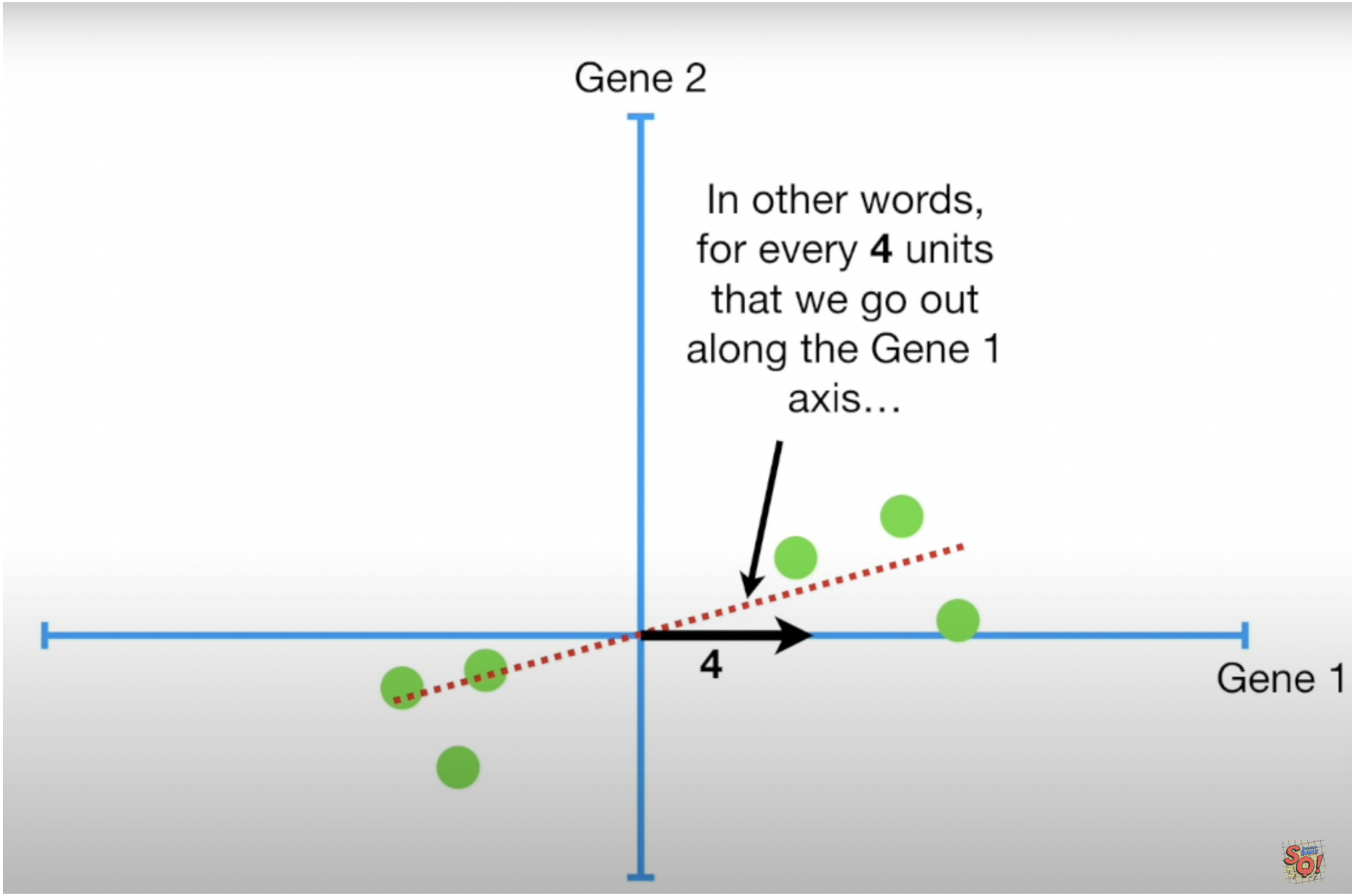

Gene 2
…or it can try to find the line that ***maximizes*** the distances from the projected points to the origin.
Gene 1

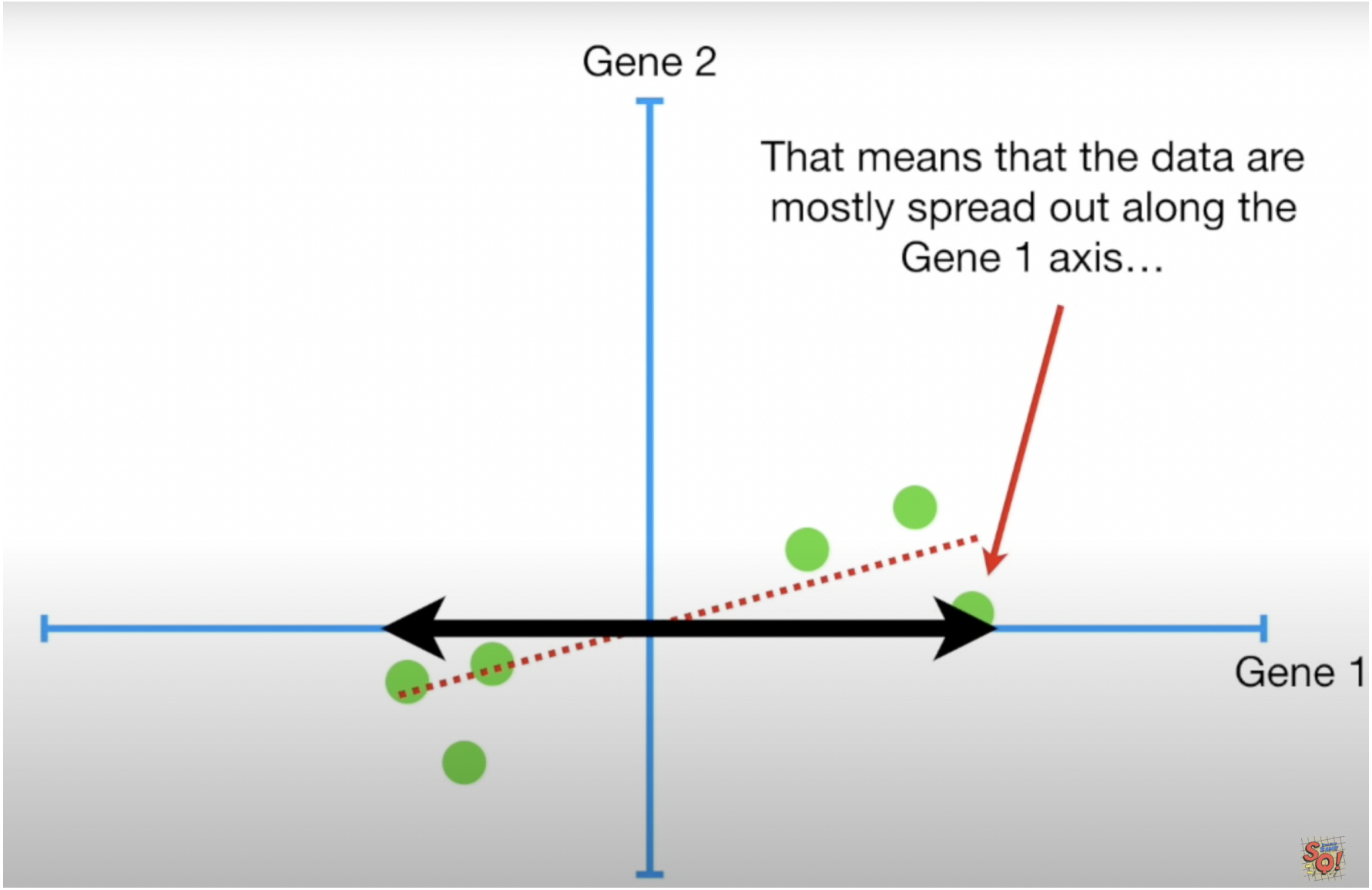
$d_1^2 + d_2^2 + d_3^2 + d_4^2 + d_5^2 + d_6^2$ = sum of squared distances = SS(distances)

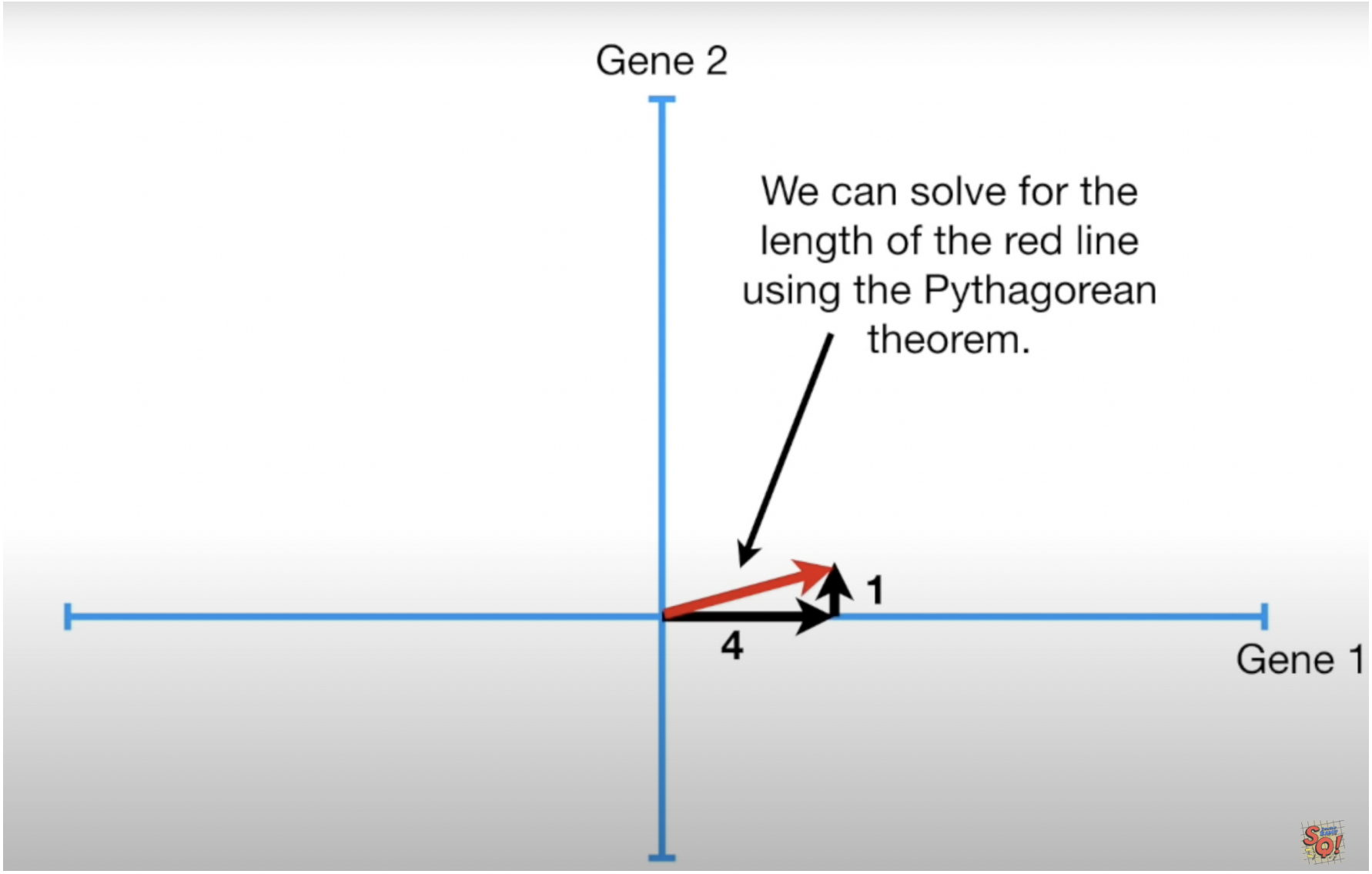


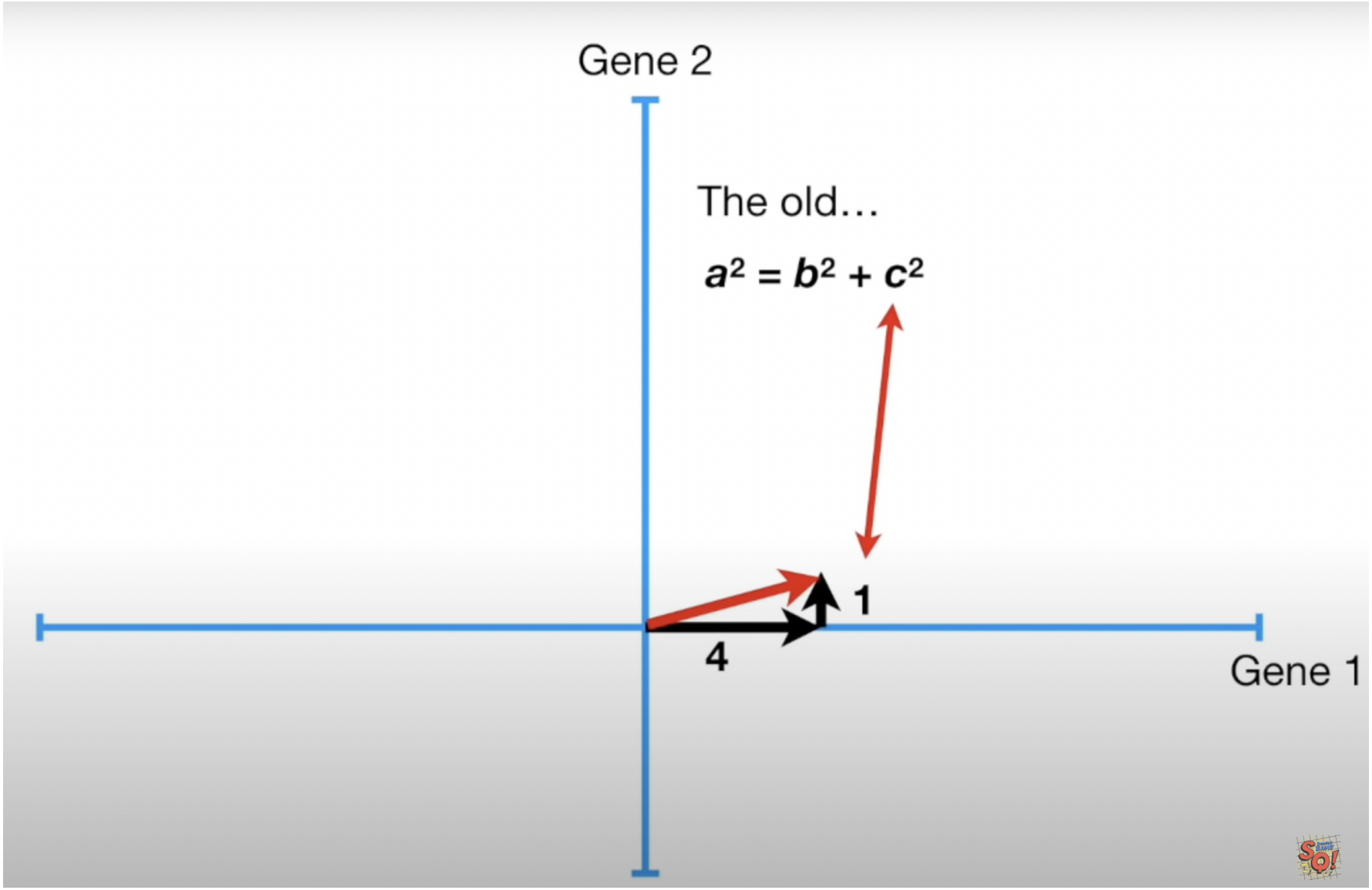

Gene 2
This line is called **Principal Component 1**. (**PC1** for short.)
Gene 1

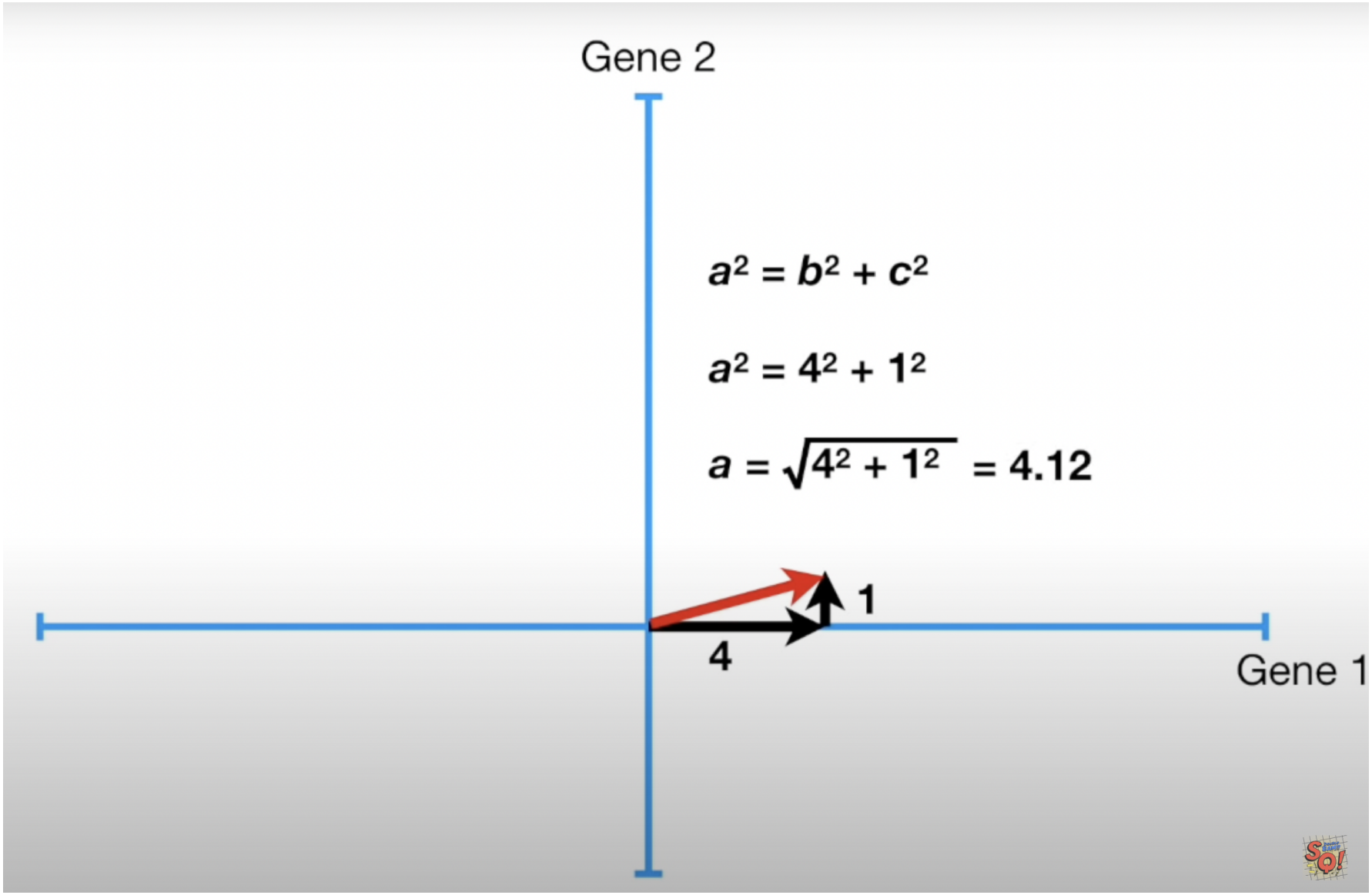

Gene 2
PC1 has a slope of 0.25
Gene 1

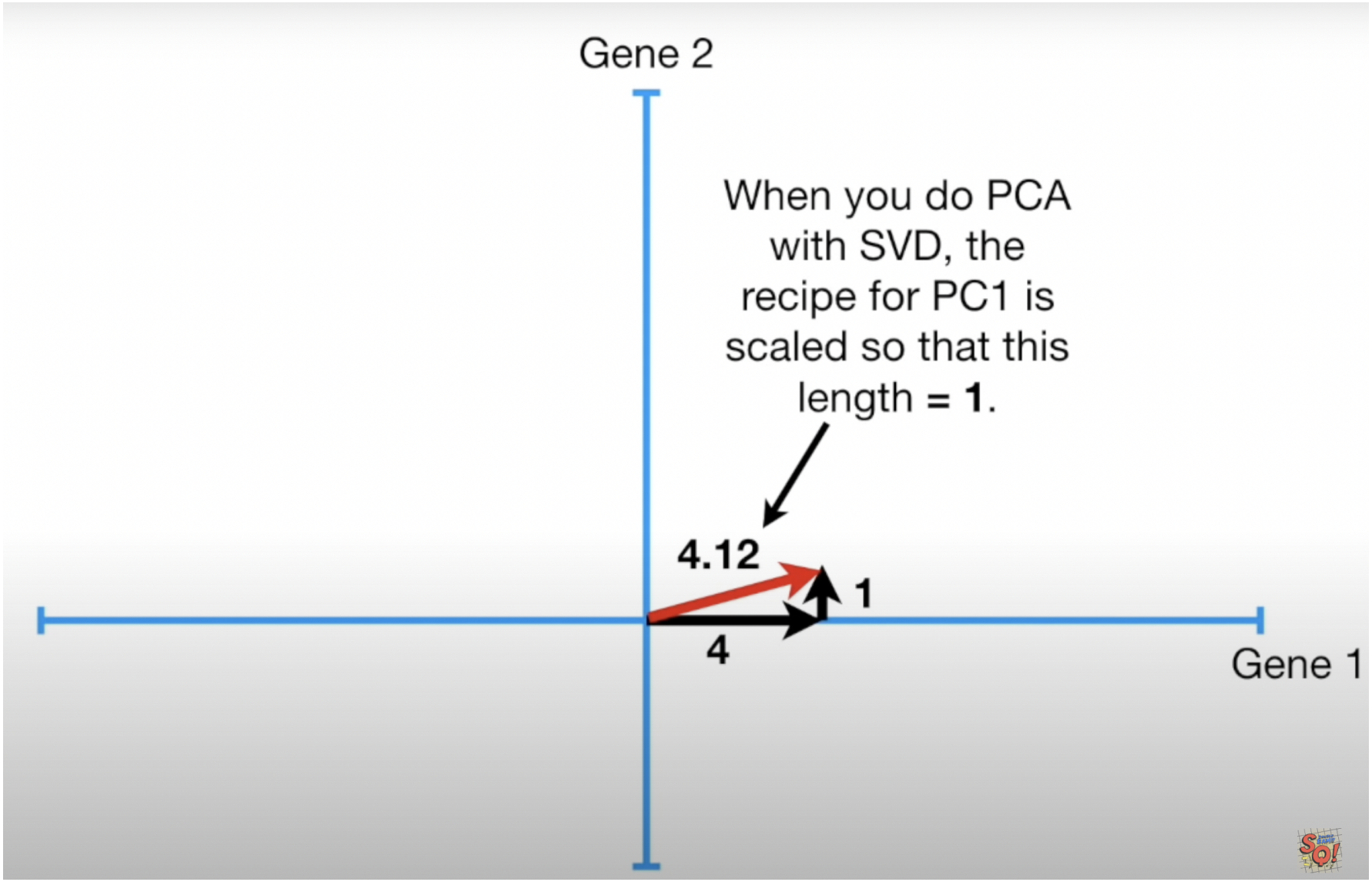

Gene 2
In other words,
for every 4 units
that we go out
along the Gene 1
axis…
4
Gene 1

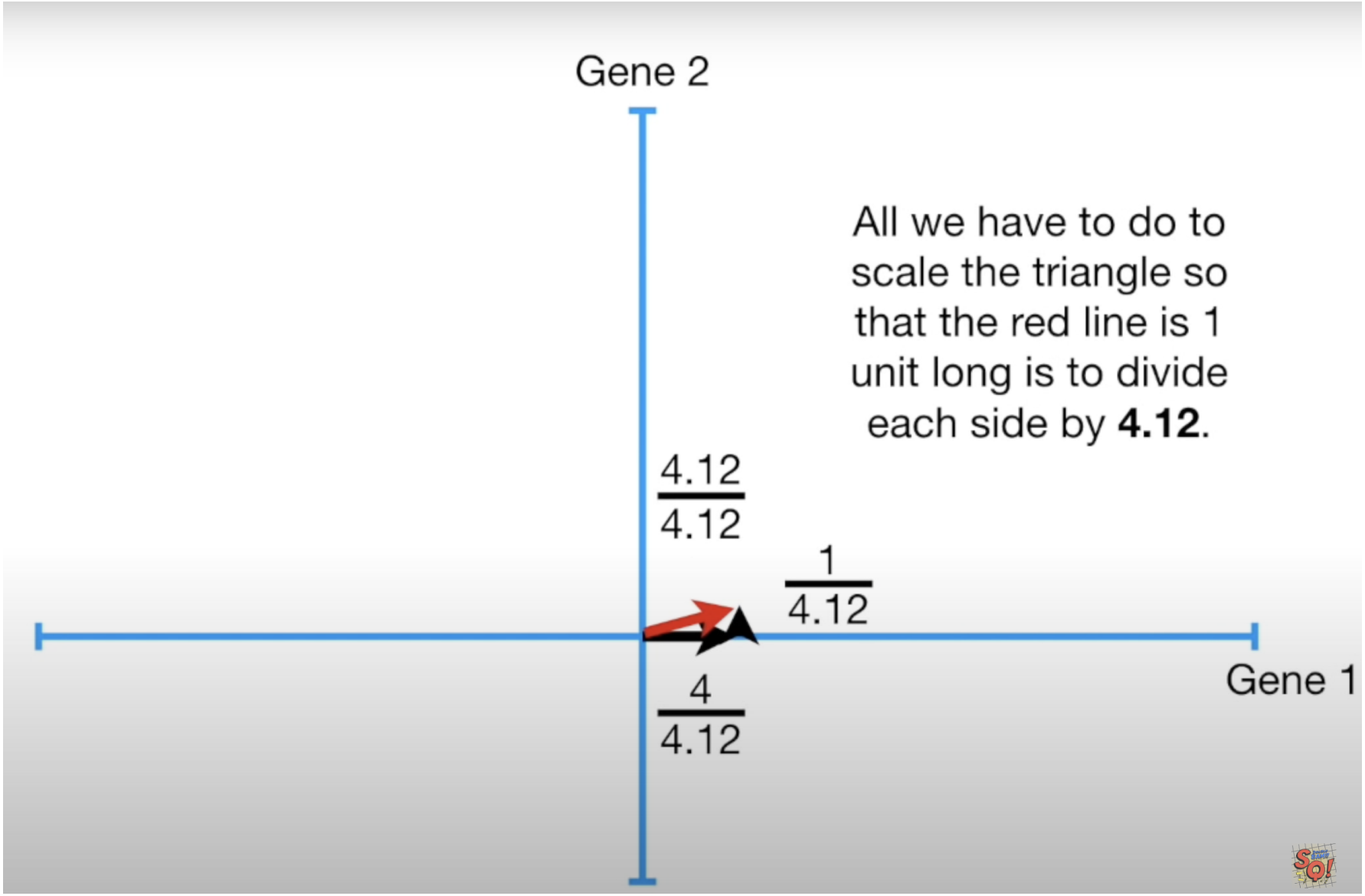

Gene 2
That means that the data are mostly spread out along the Gene 1 axis…
Gene 1

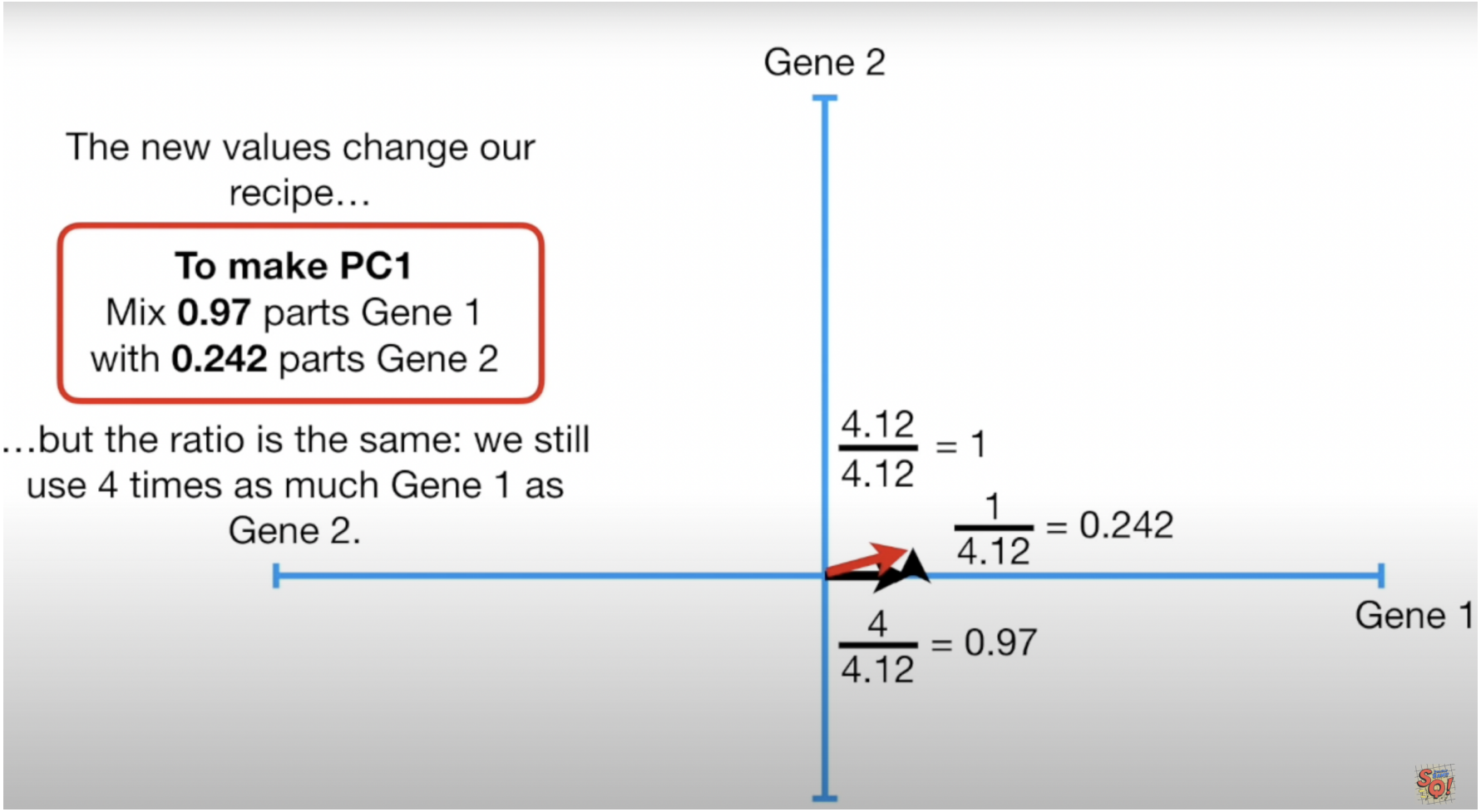

Gene 2
We can solve for the length of the red line using the Pythagorean theorem.
1
4
Gene 1


Gene 2
The old…
$a^2 = b^2 + c^2$
1
4
Gene 1


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

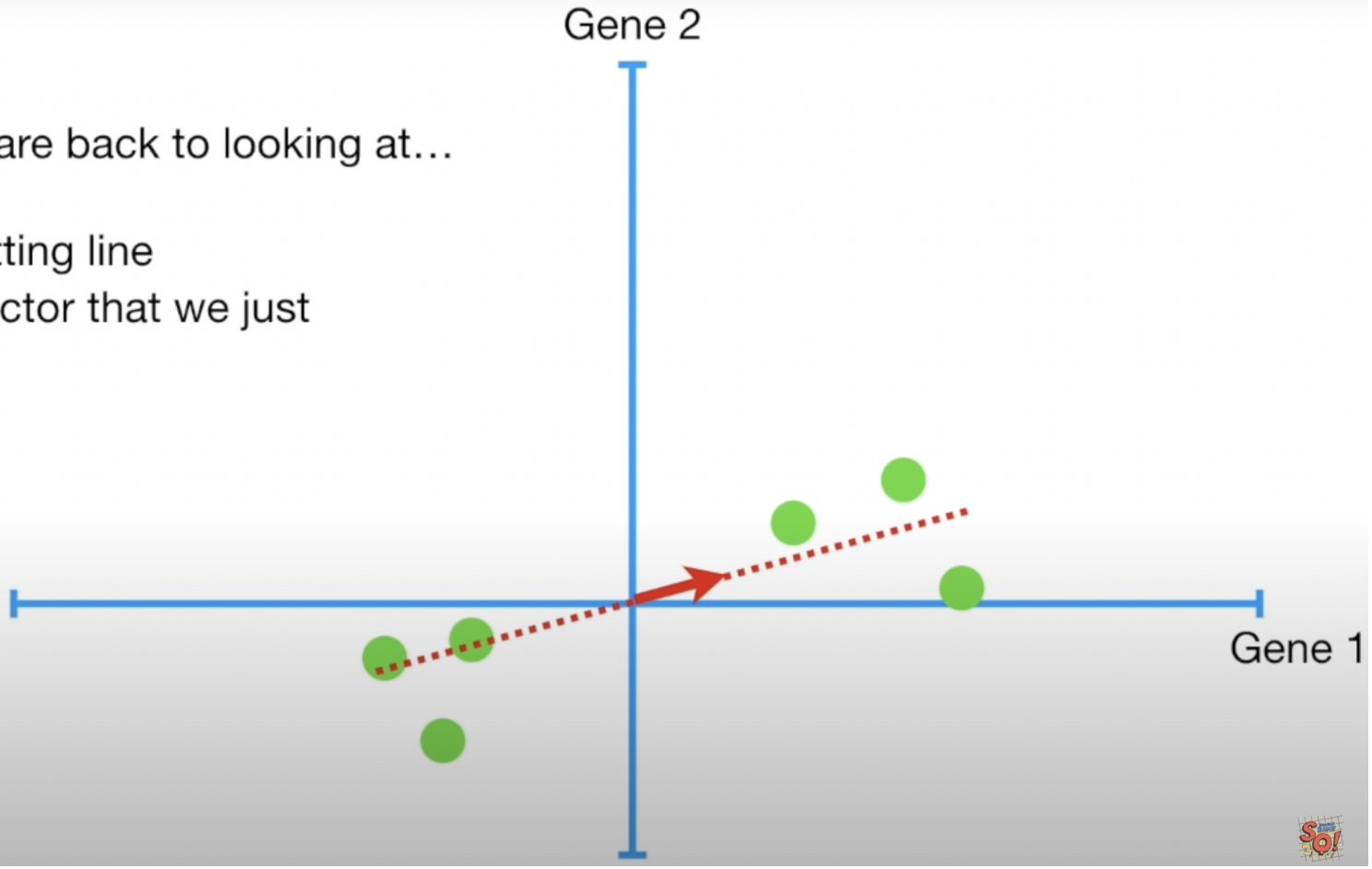

Gene 2
When you do PCA with SVD, the recipe for PC1 is scaled so that this length = 1.
4.12
1
4
Gene 1

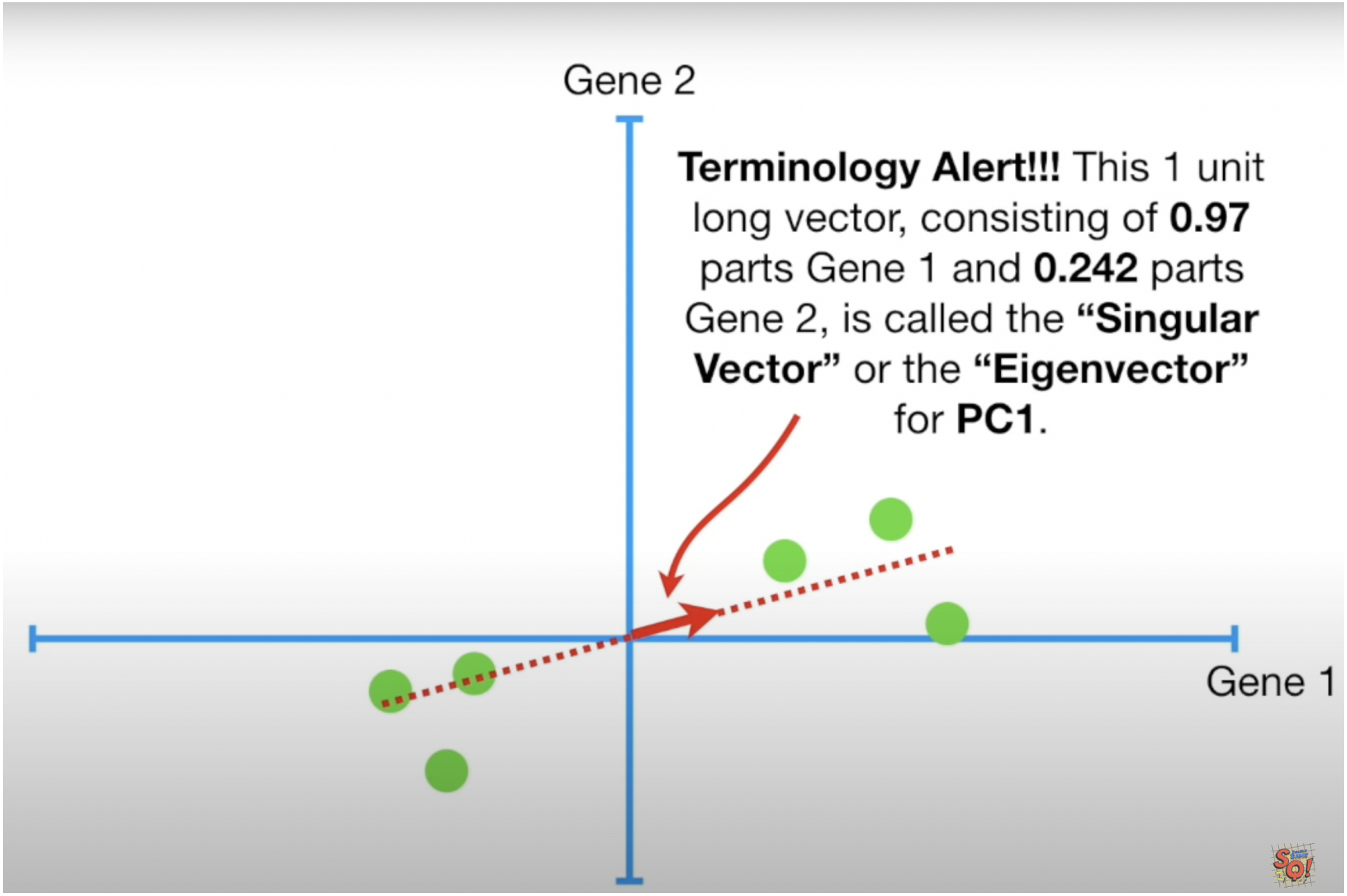

Gene 2
All we have to do to scale the triangle so that the red line is 1 unit long is to divide each side by **4.12**.
4.12
4.12
1
4.12
4
4.12
Gene 1

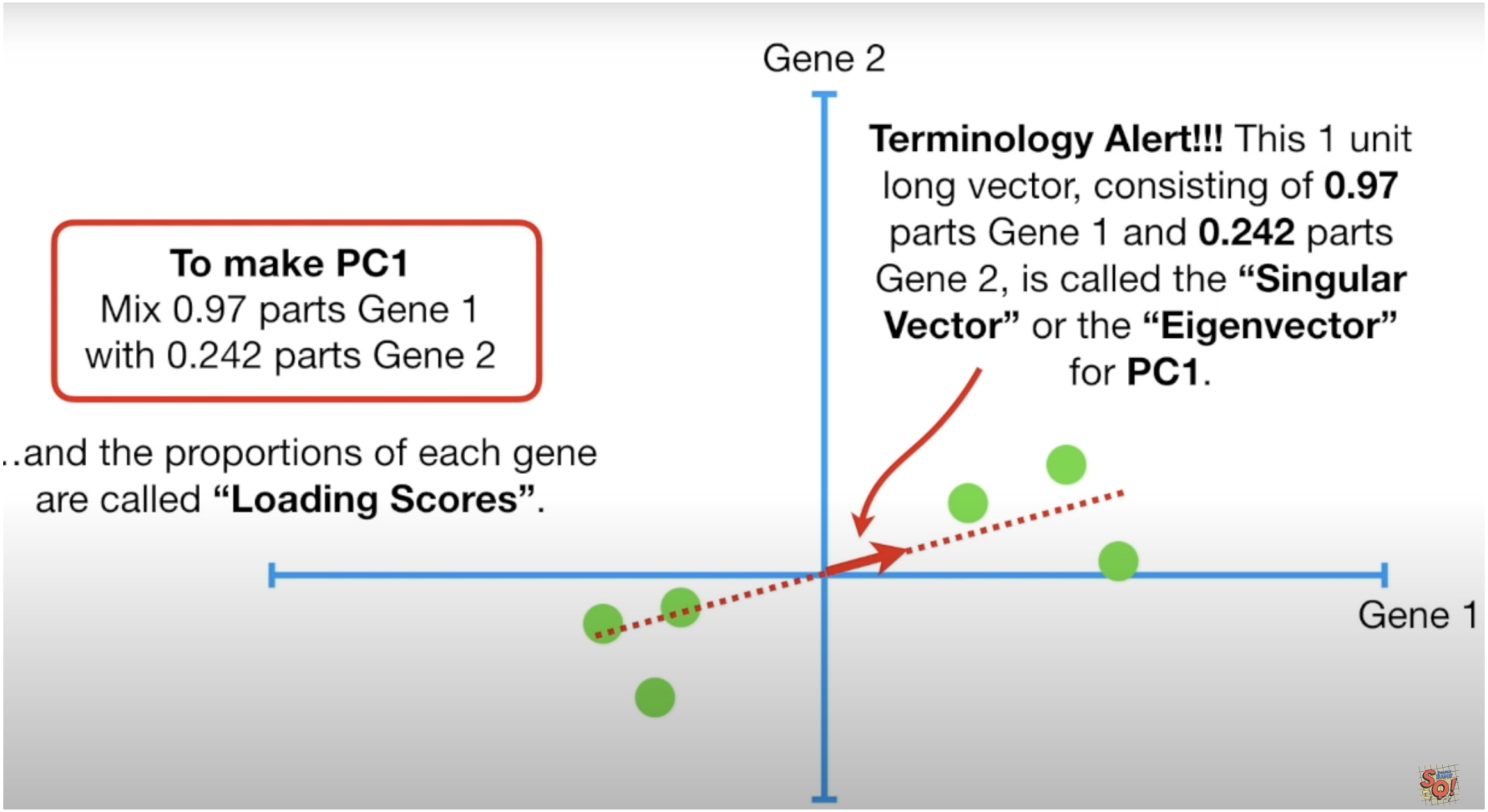

Gene 2
The new values change our recipe…
To make PC1
Mix 0.97 parts Gene 1
with 0.242 parts Gene 2
…but the ratio is the same: we still use 4 times as much Gene 1 as Gene 2.
4.12 / 4.12 = 1
1 / 4.12 = 0.242
4 / 4.12 = 0.97
Gene 1

So now we are back to looking at…

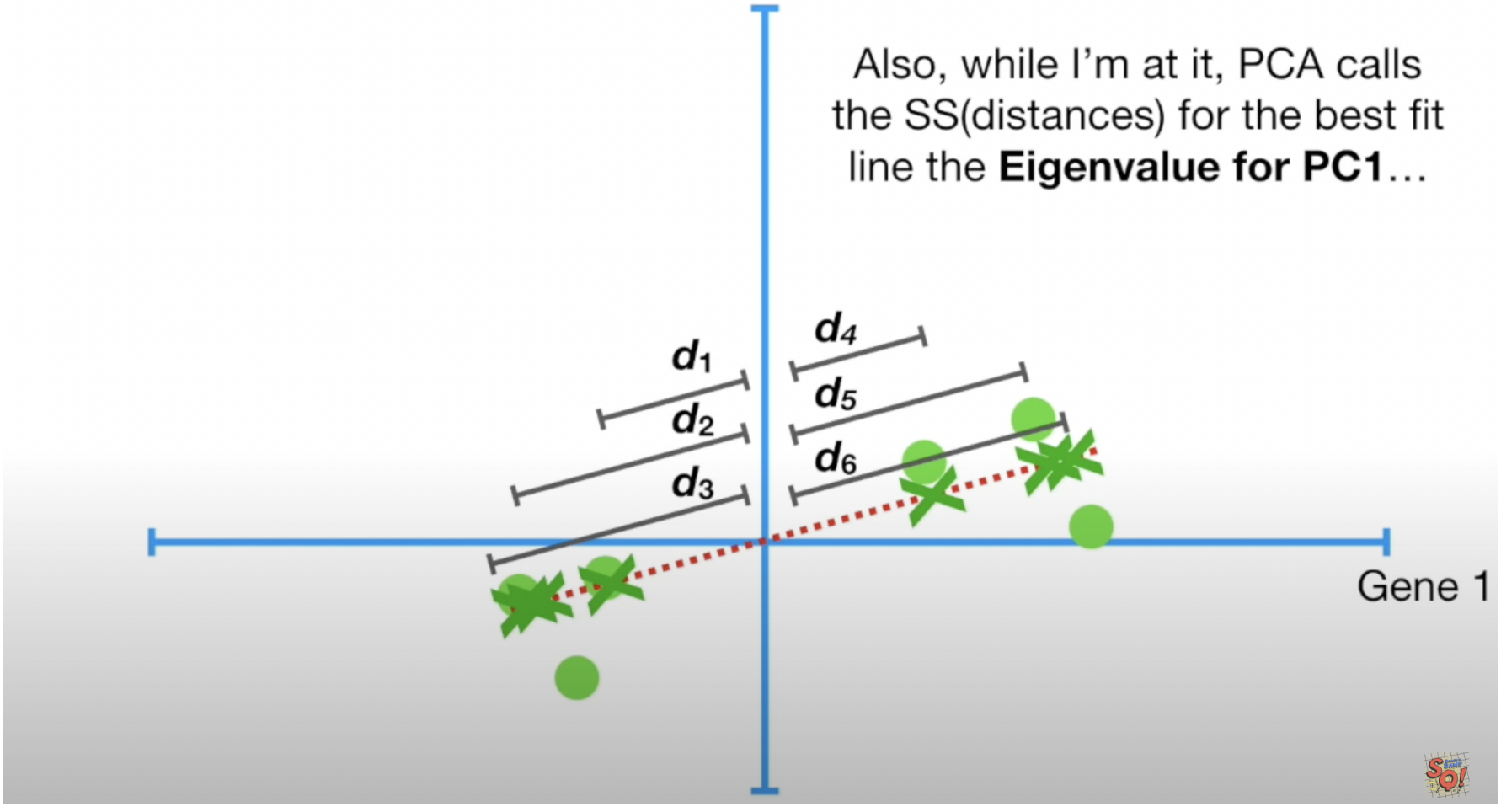
- The data
- The best fitting line
- The unit vector that we just calculated.




Gene 2
Terminology Alert!!! This 1 unit long vector, consisting of **0.97** parts Gene 1 and **0.242** parts Gene 2, is called the **"Singular Vector"** or the **"Eigenvector"** for **PC1**.
Gene 1


Gene 2
Terminology Alert!!! This 1 unit long vector, consisting of 0.97 parts Gene 1 and 0.242 parts Gene 2, is called the "Singular Vector" or the "Eigenvector" for PC1.
To make PC1
Mix 0.97 parts Gene 1
with 0.242 parts Gene 2
..and the proportions of each gene are called "Loading Scores".
Gene 1

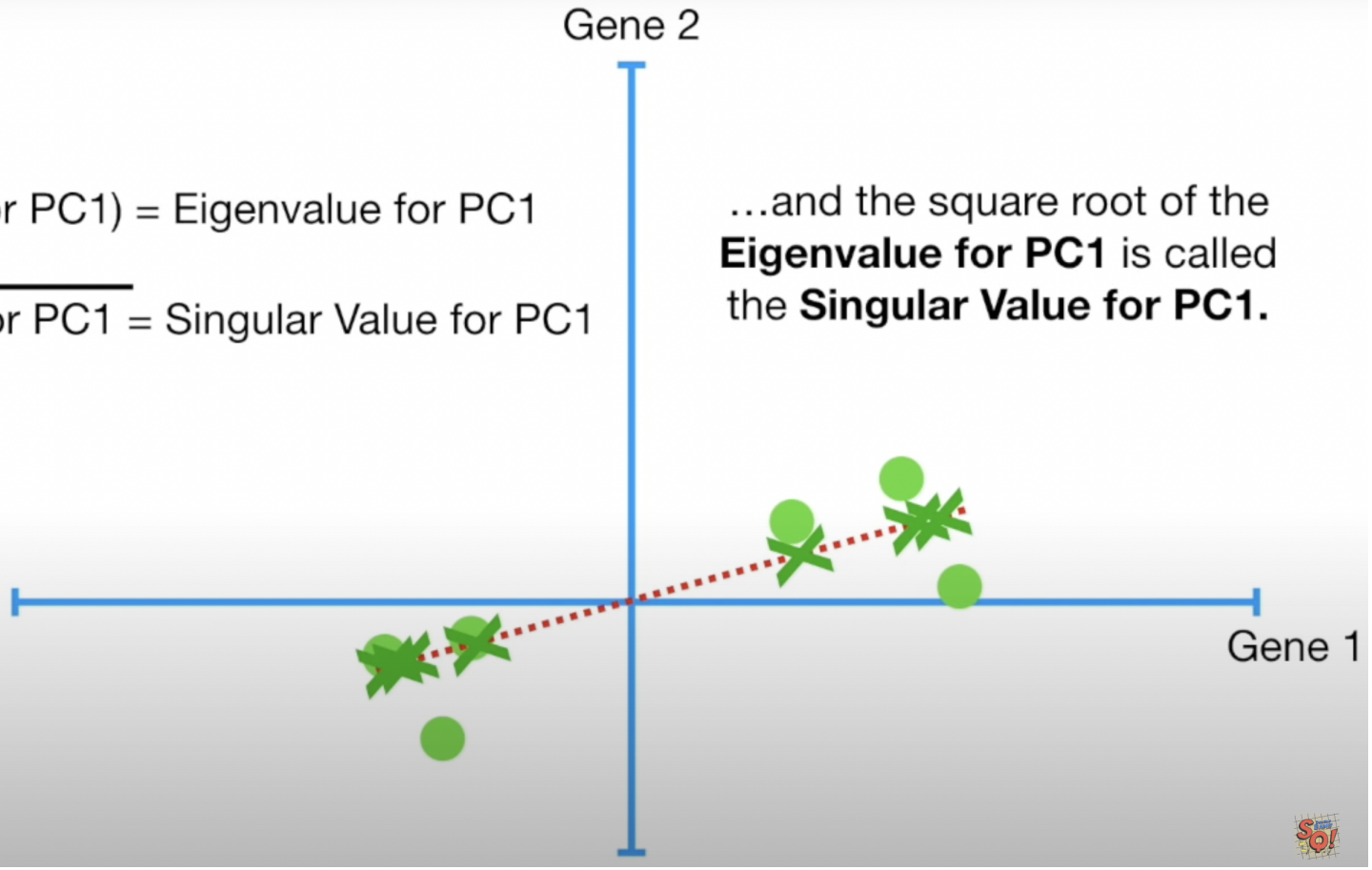
$d_1^2 + d_2^2 + d_3^2 + d_4^2 + d_5^2 + d_6^2$ = sum of squared distances = SS(distances)

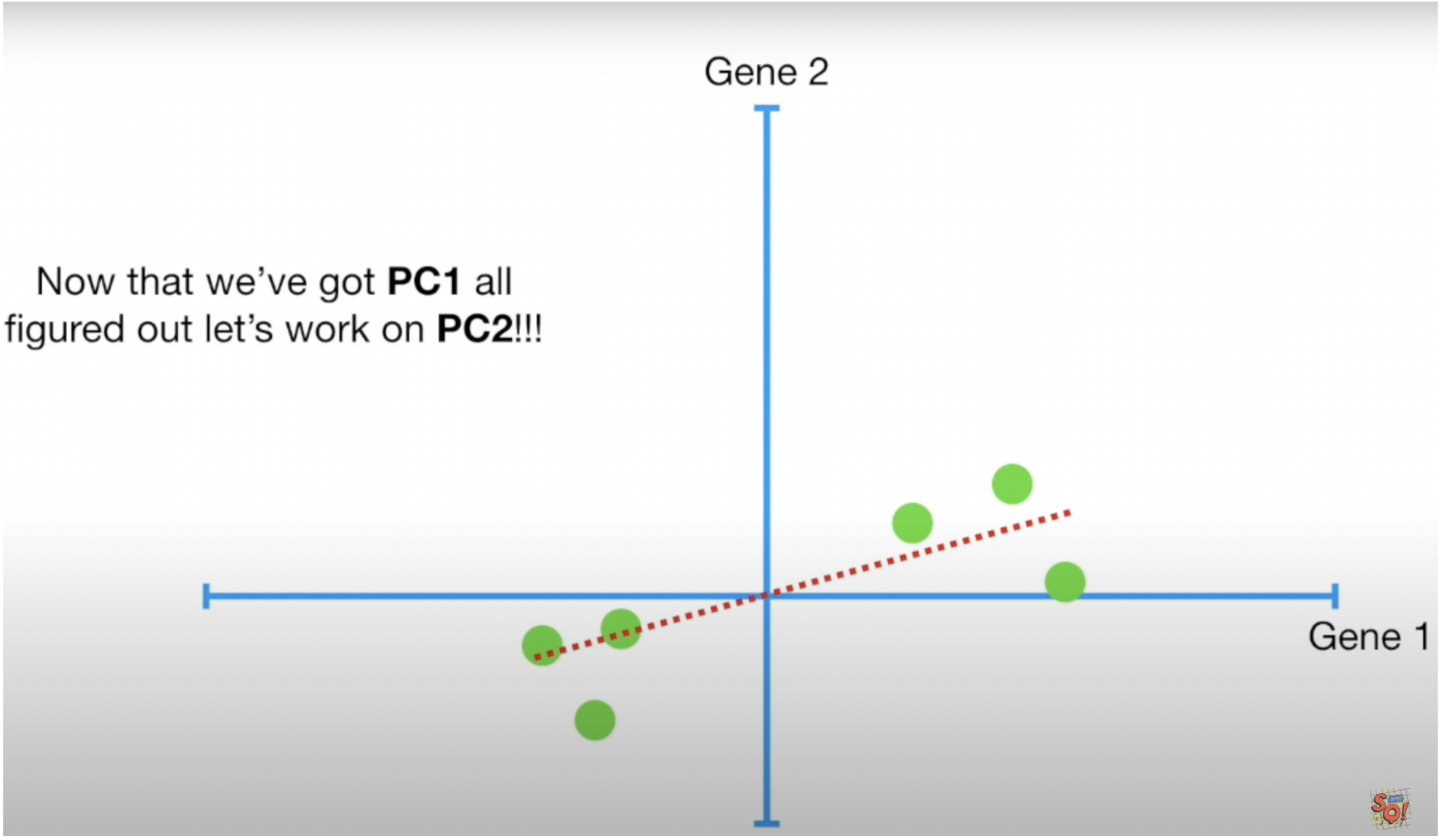


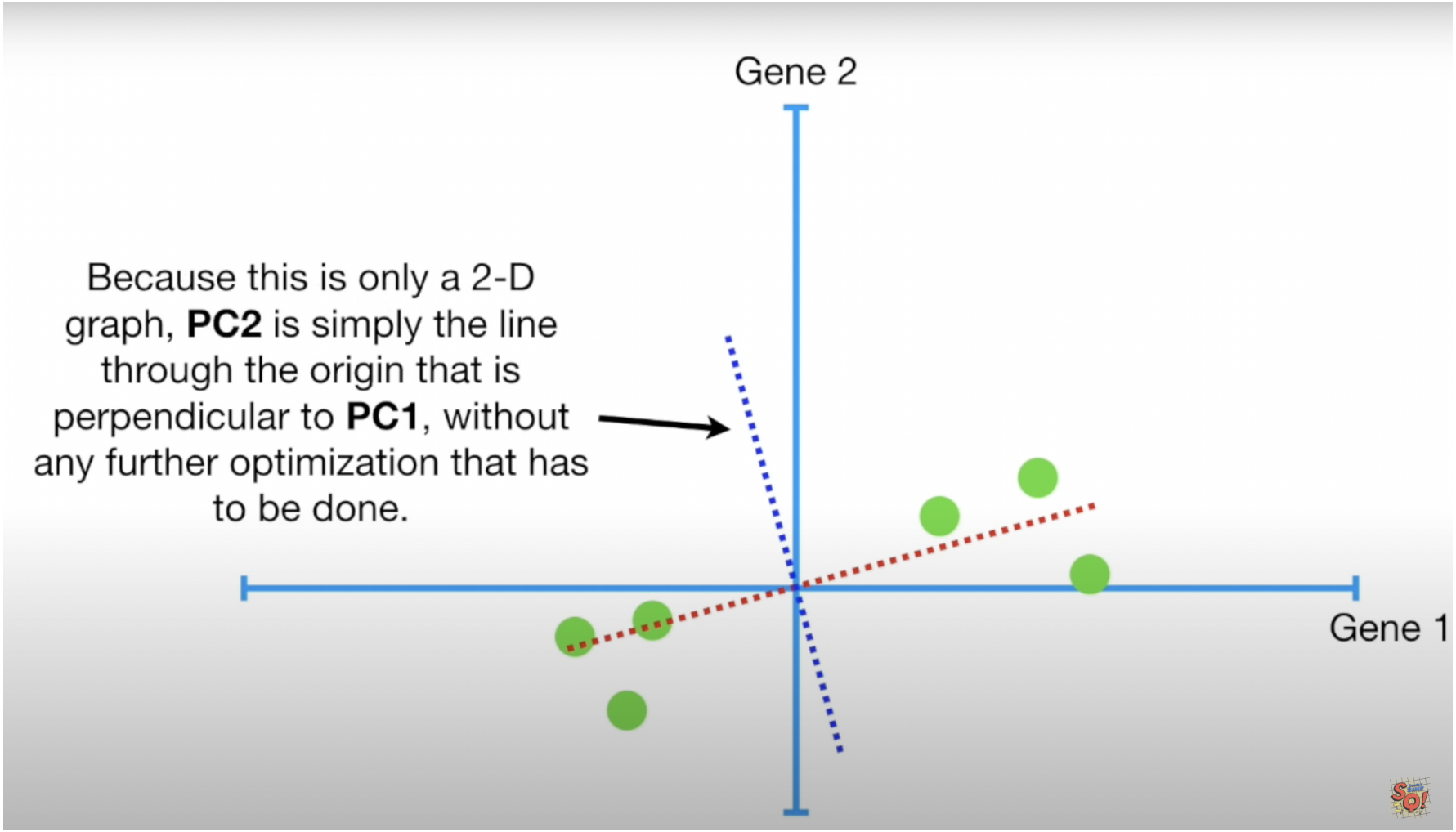
SS(distances for PC1) = Eigenvalue for PC1

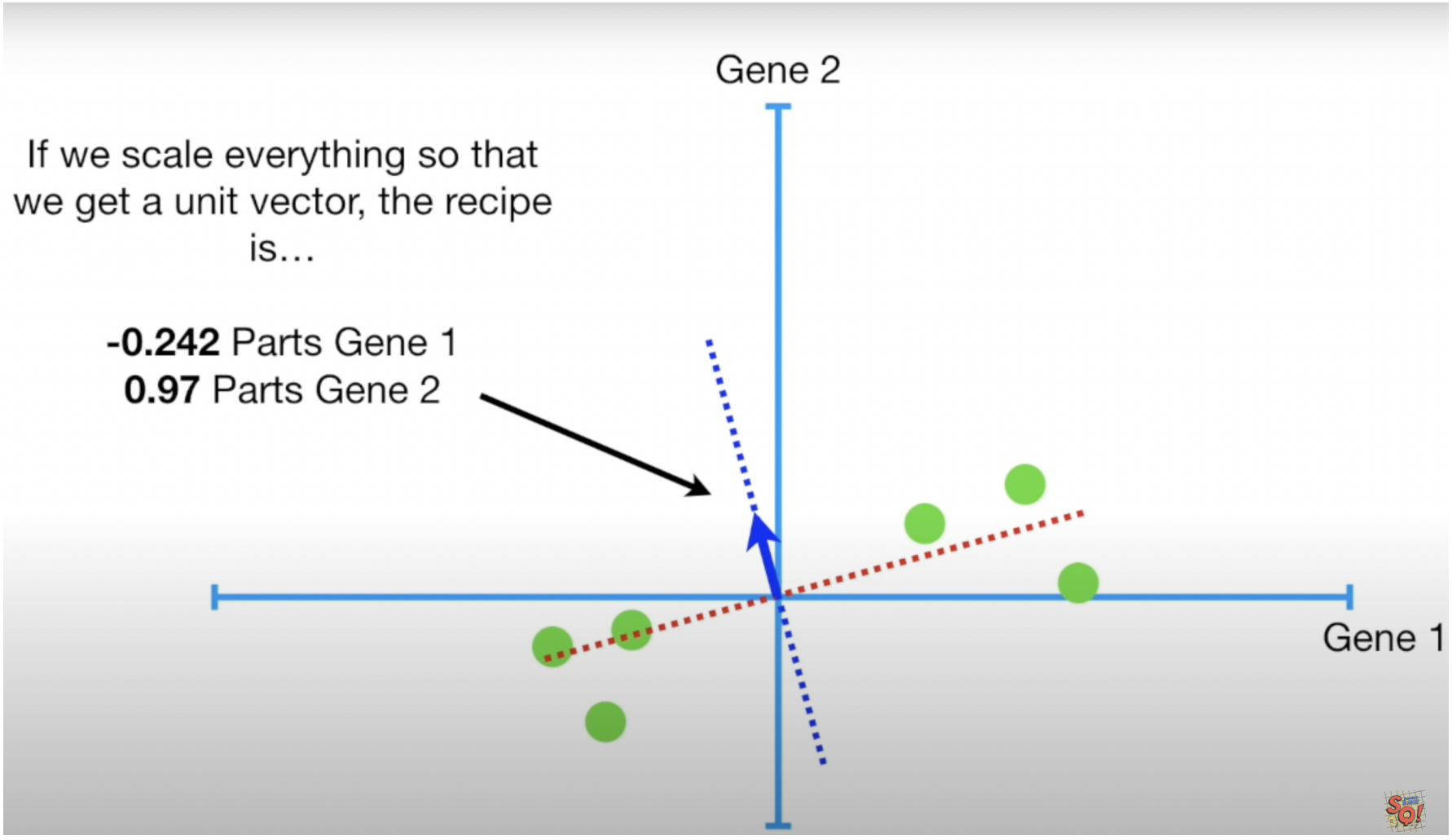
$\sqrt{\text{Eigenvalue for PC1}}$ = Singular Value for PC1

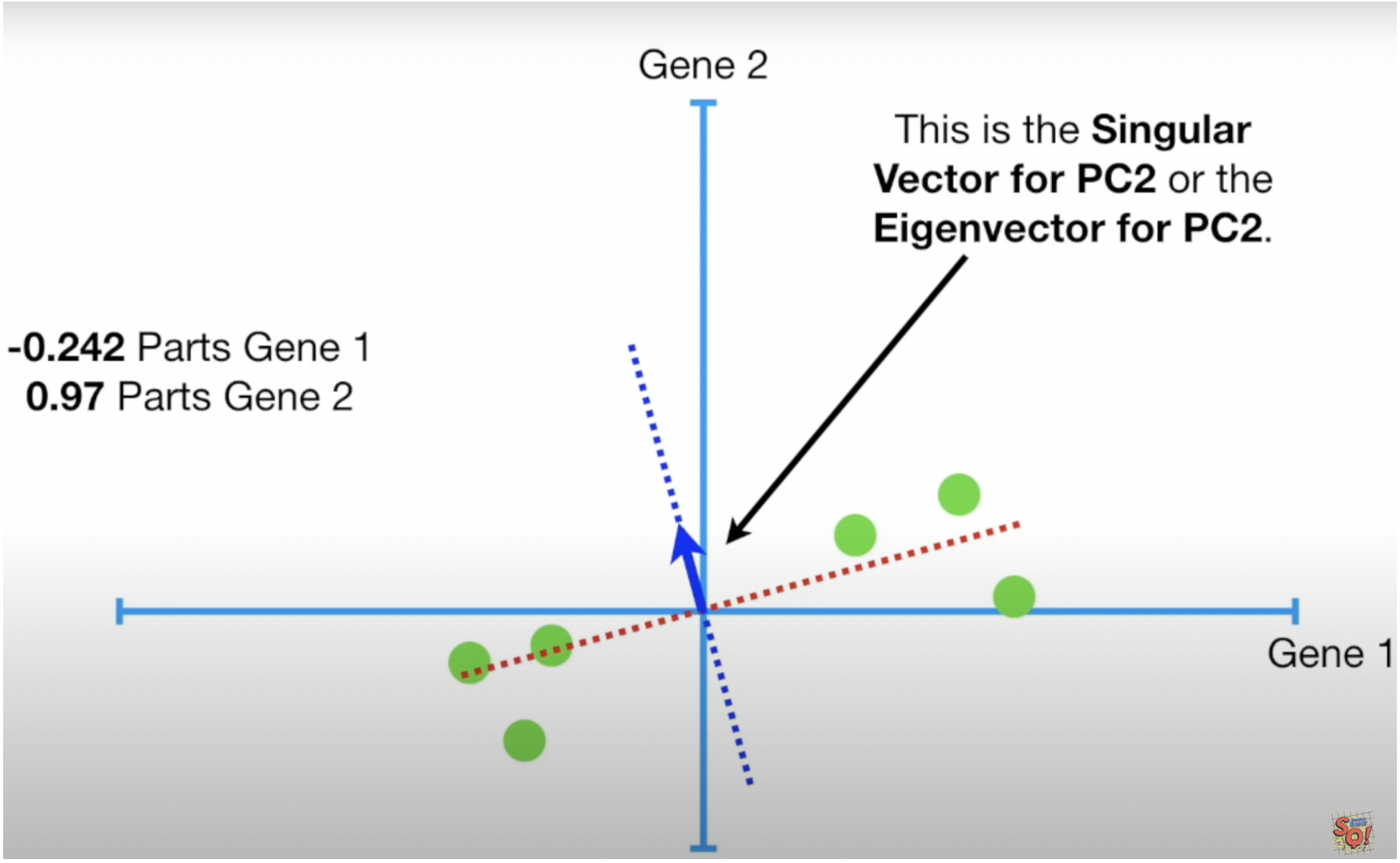
…and the square root of the **Eigenvalue for PC1** is called the **Singular Value for PC1.**

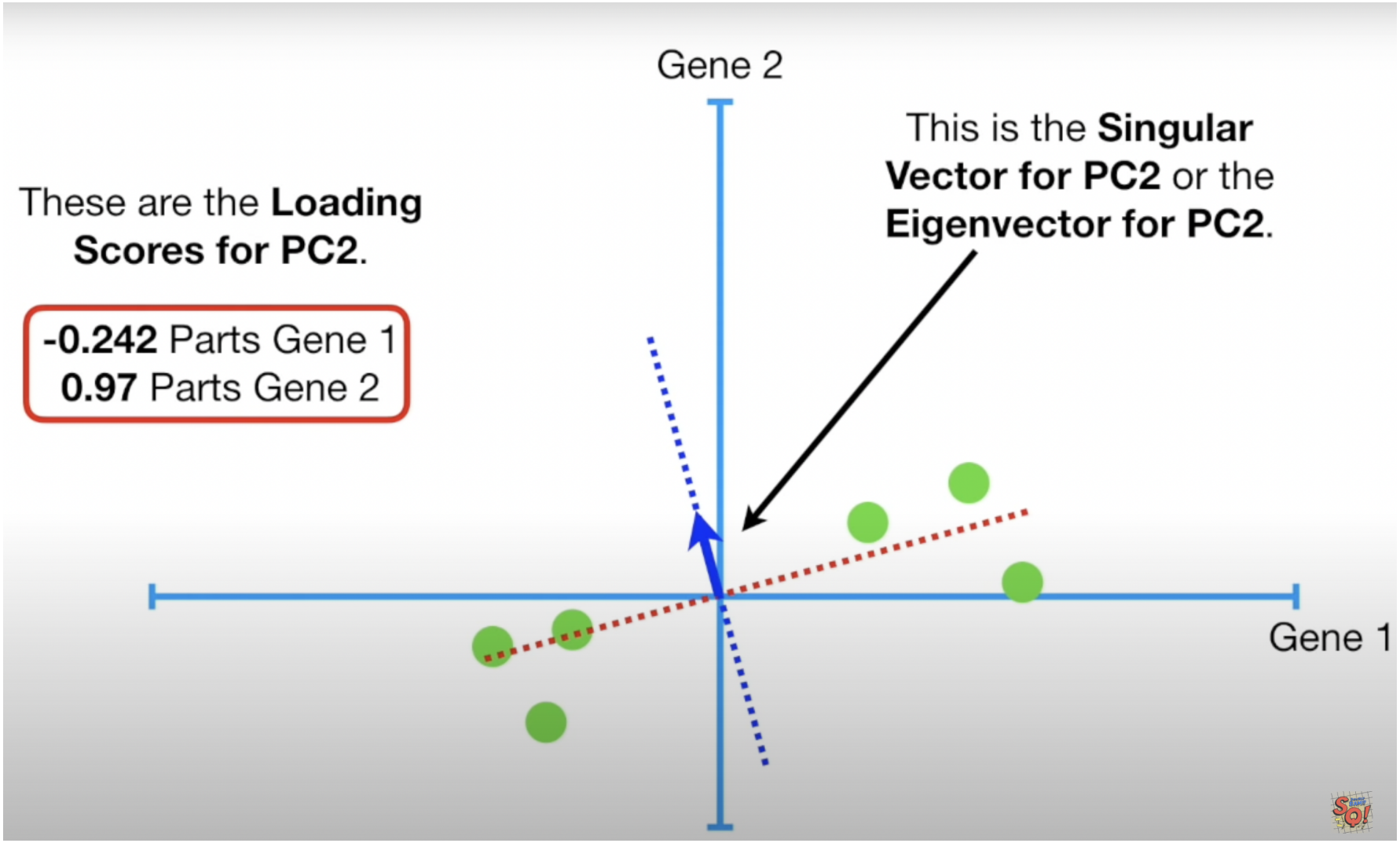



Gene 2
Now that we've got **PC1** all figured out let's work on **PC2**!!!
Gene 1


Gene 2
Because this is only a 2-D graph, **PC2** is simply the line through the origin that is perpendicular to **PC1**, without any further optimization that has to be done.
Gene 1


Gene 2
If we scale everything so that we get a unit vector, the recipe is…
-0.242 Parts Gene 1
0.97 Parts Gene 2
Gene 1

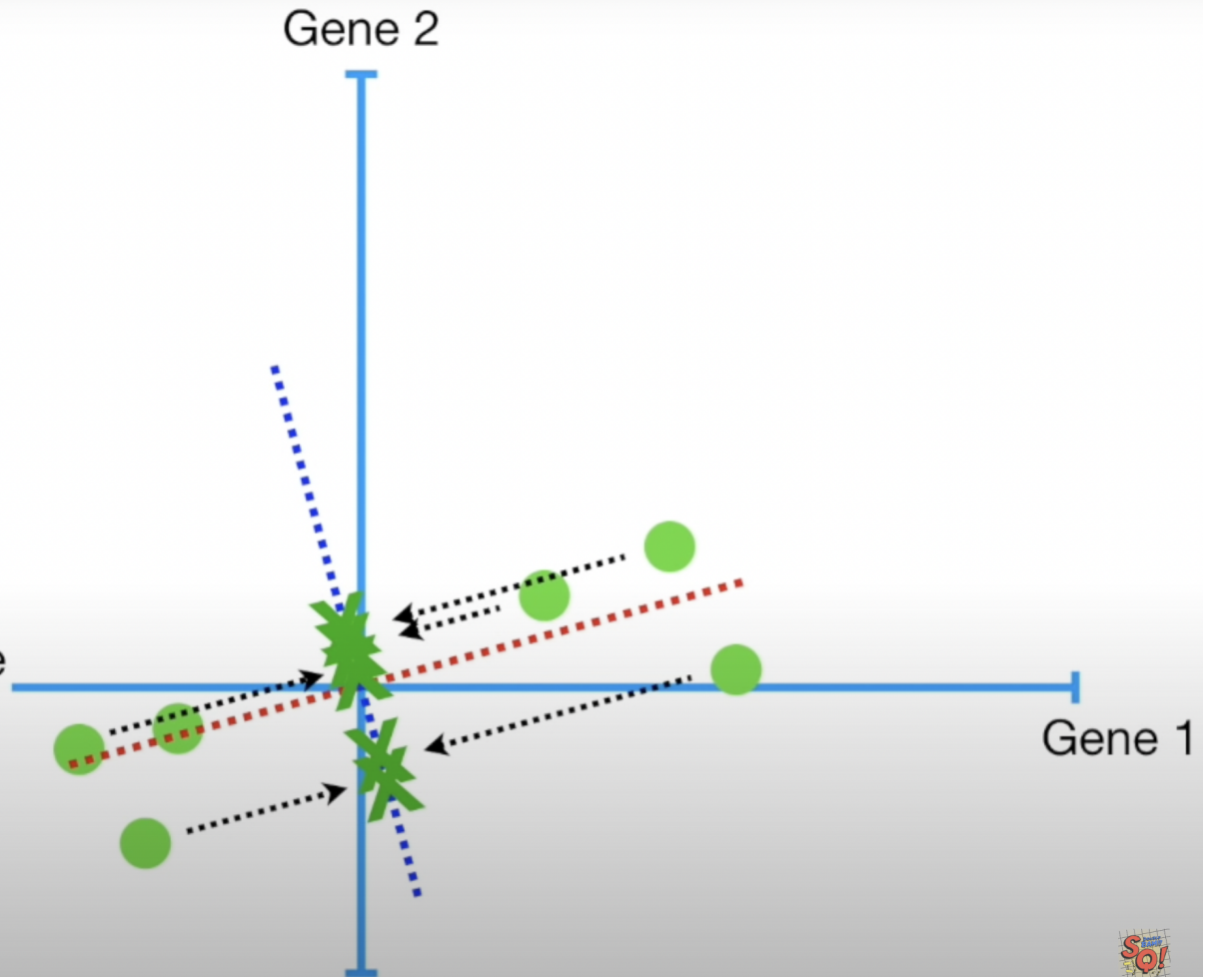

Gene 2
This is the **Singular Vector for PC2** or the **Eigenvector for PC2**.
-0.242 Parts Gene 1
0.97 Parts Gene 2
Gene 1


These are the **Loading Scores for PC2**.
-0.242 Parts Gene 1
0.97 Parts Gene 2
Gene 2
This is the **Singular Vector for PC2** or the **Eigenvector for PC2**.
Gene 1

These are the **Loading Scores for PC2**.

-0.242 Parts Gene 1
0.97 Parts Gene 2

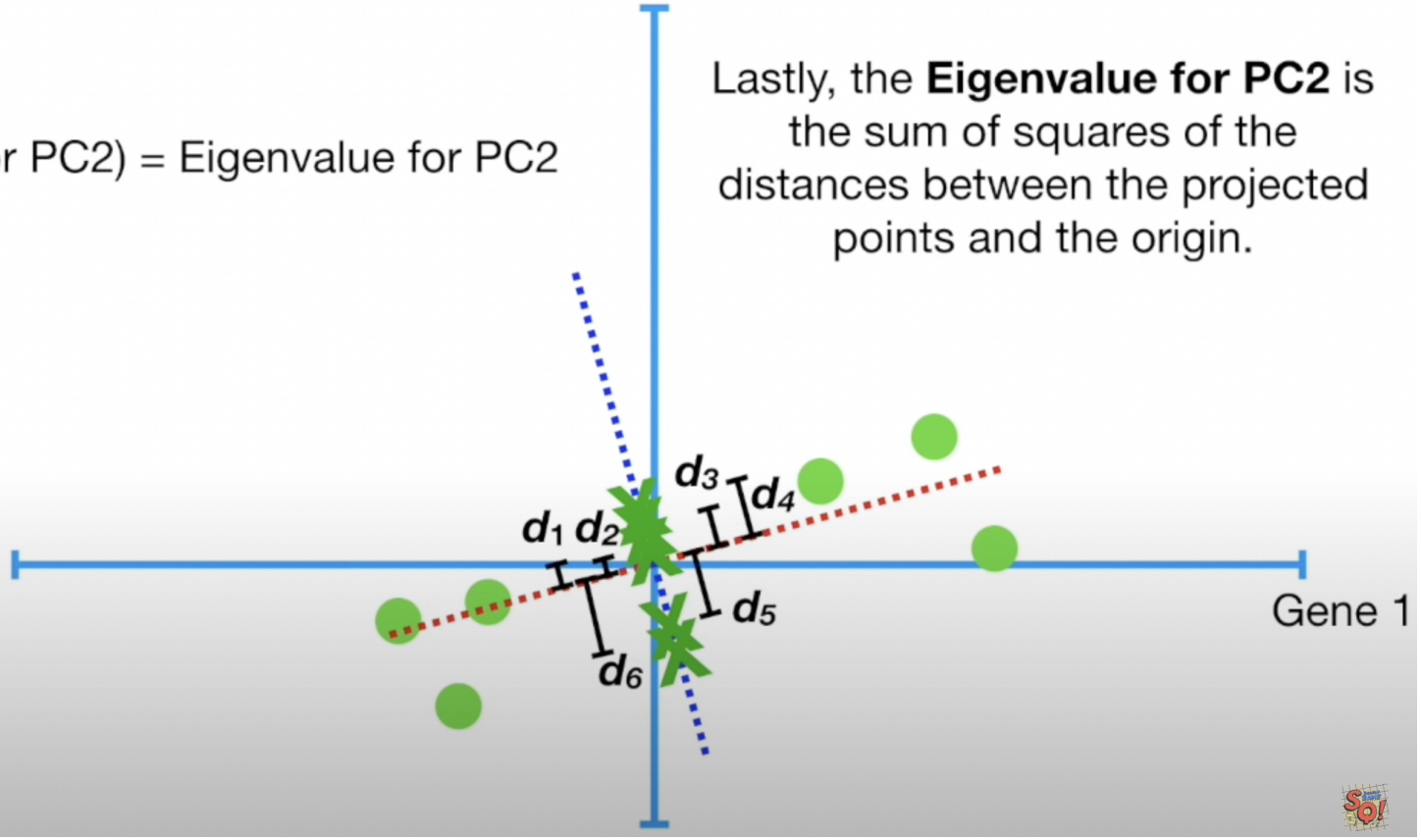
They tell us that, in terms of how the values are projected onto PC2, Gene 2 is 4 times as important as Gene 1.

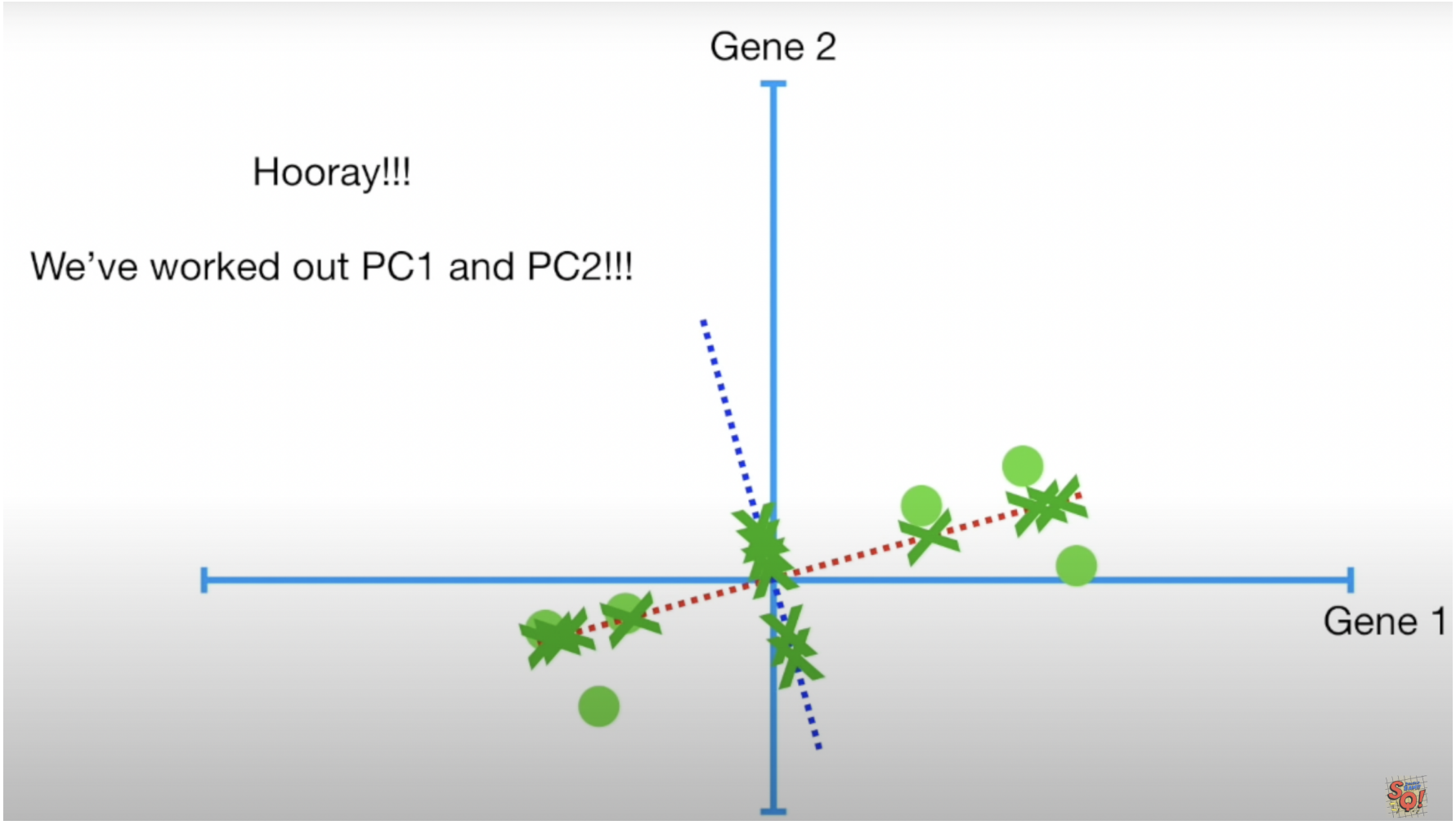


$d_1^2 + d_2^2 + d_3^2 + d_4^2 + d_5^2 + d_6^2$ = sum of squared distances = SS(distances)

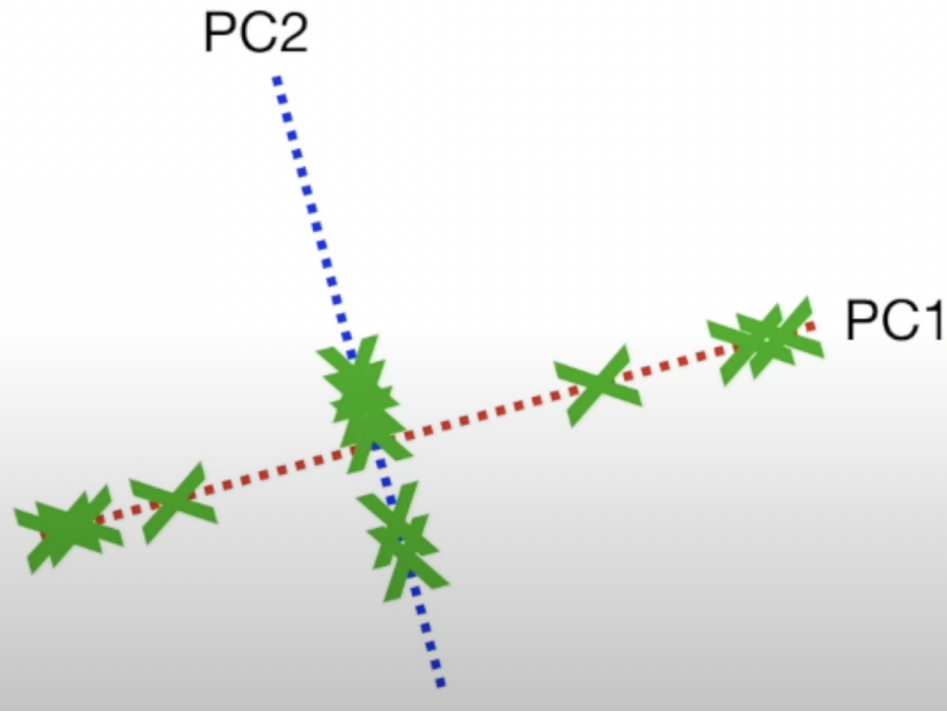
SS(distances for PC2) = Eigenvalue for PC2

Lastly, the **Eigenvalue for PC2** is the sum of squares of the distances between the projected points and the origin.

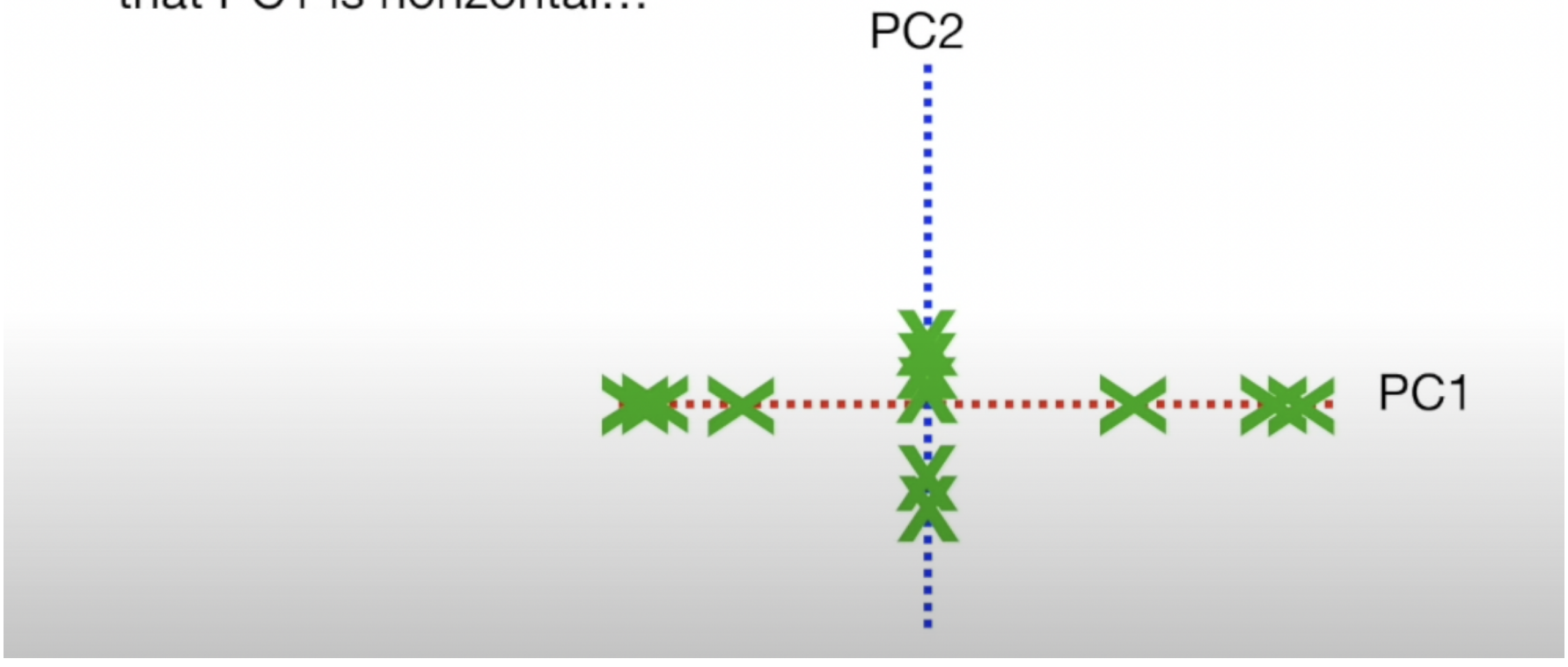



Hooray!!!
We've worked out PC1 and PC2!!!
Gene 2
Gene 1

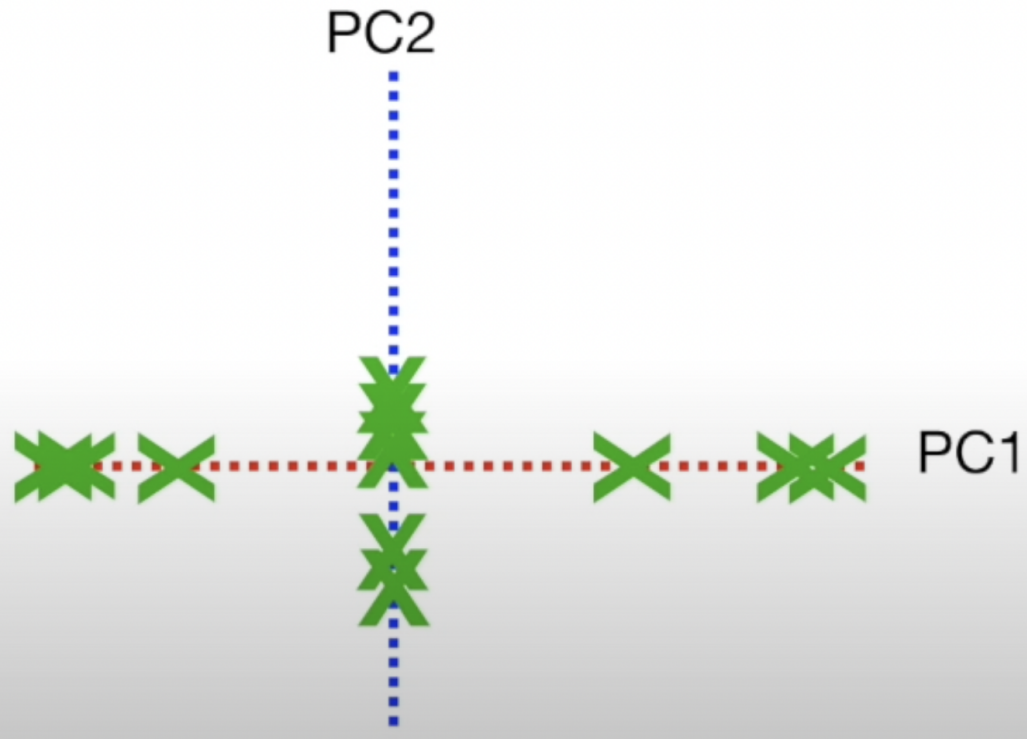
To draw the final PCA plot…

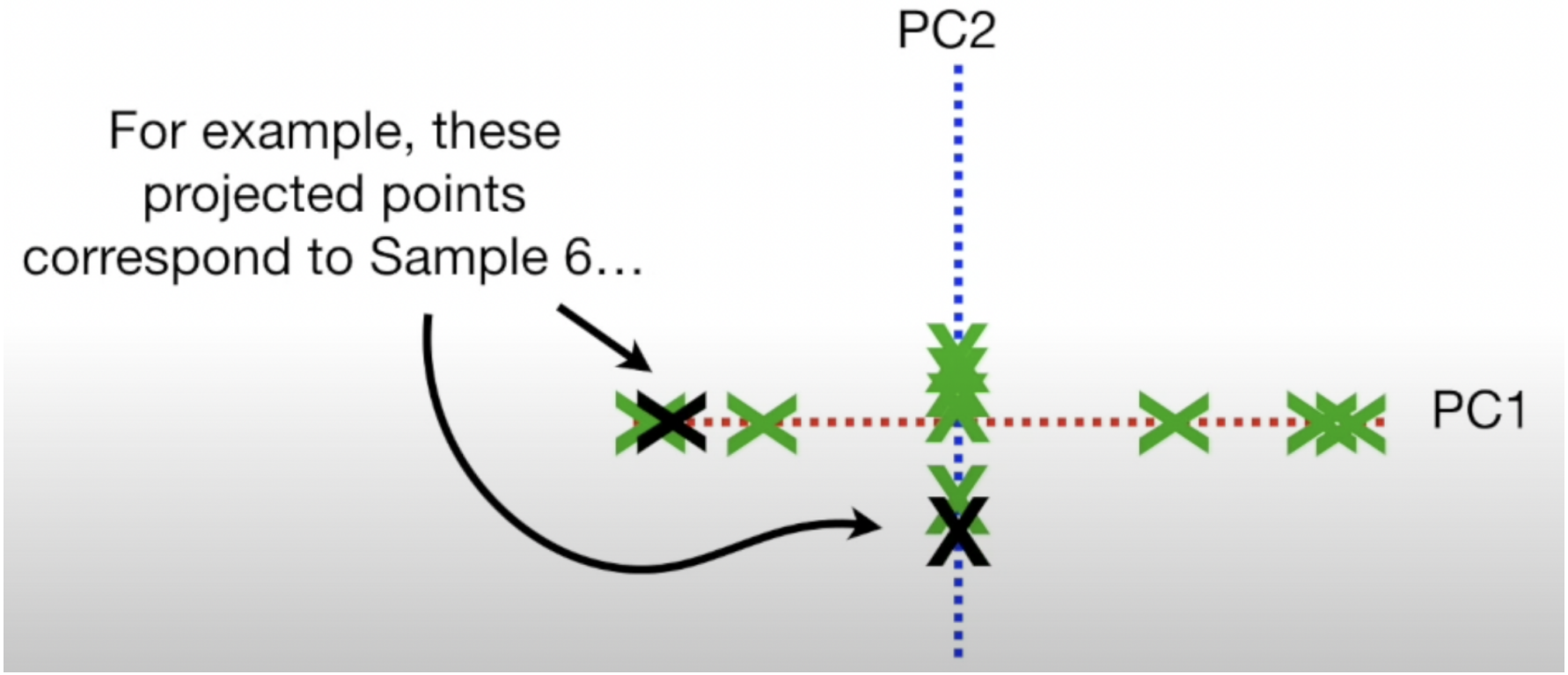


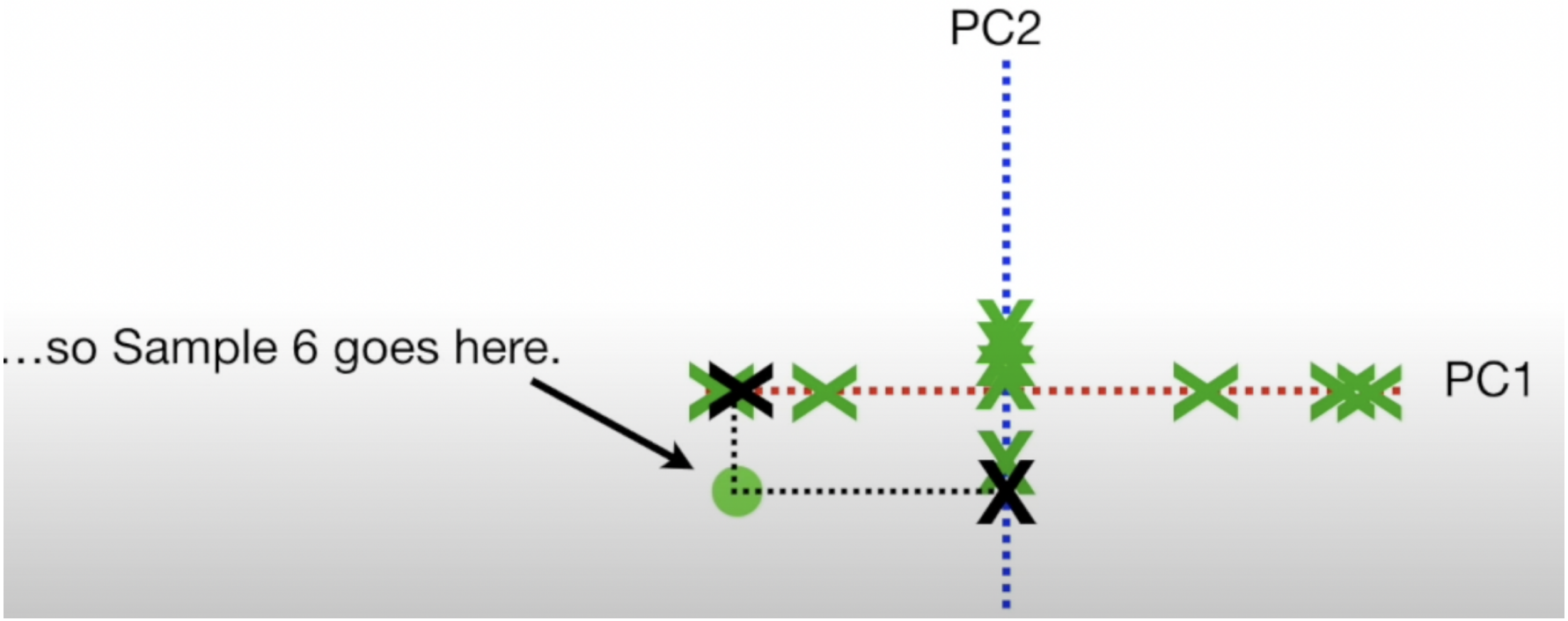
We simply rotate everything so that PC1 is horizontal…

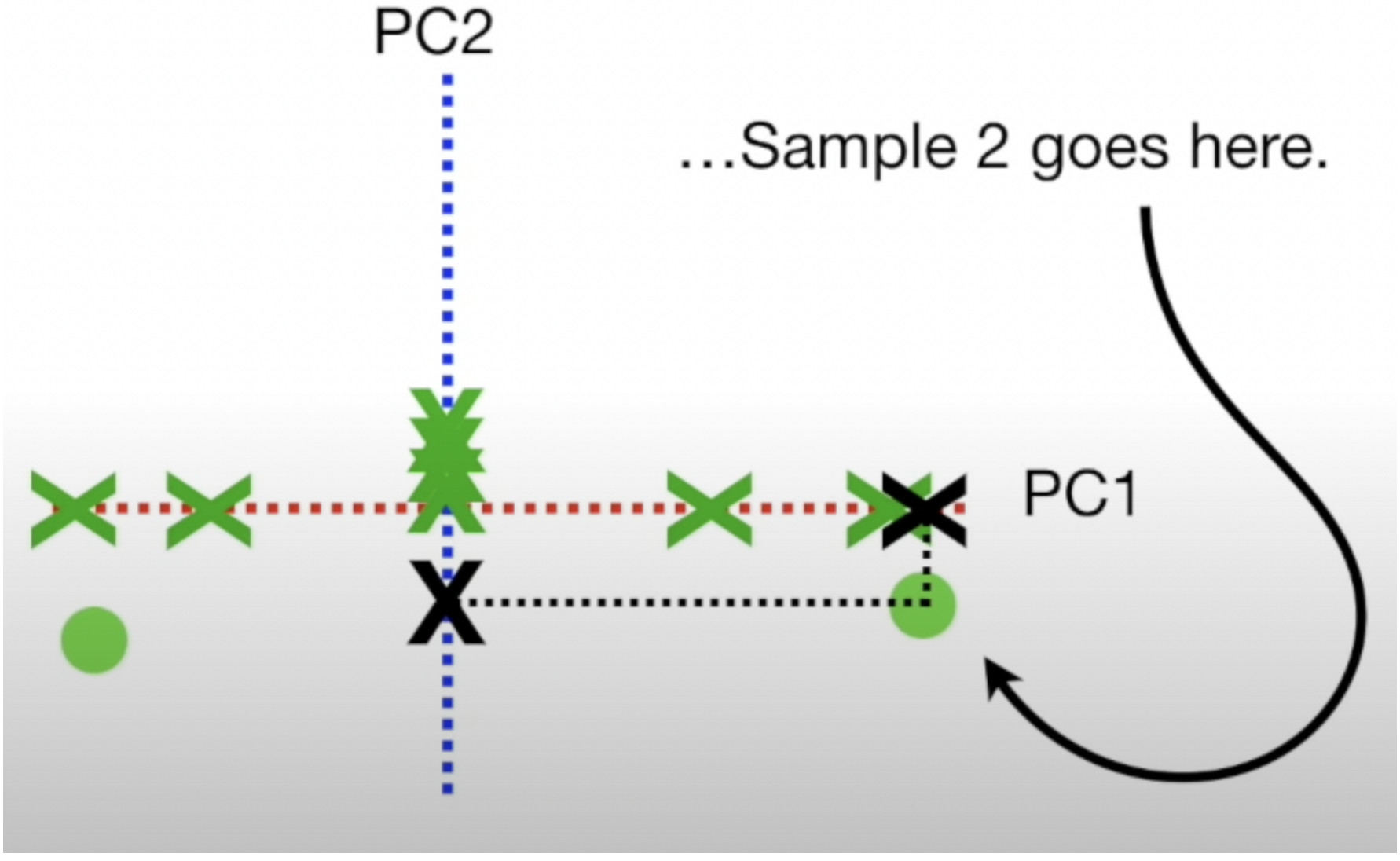


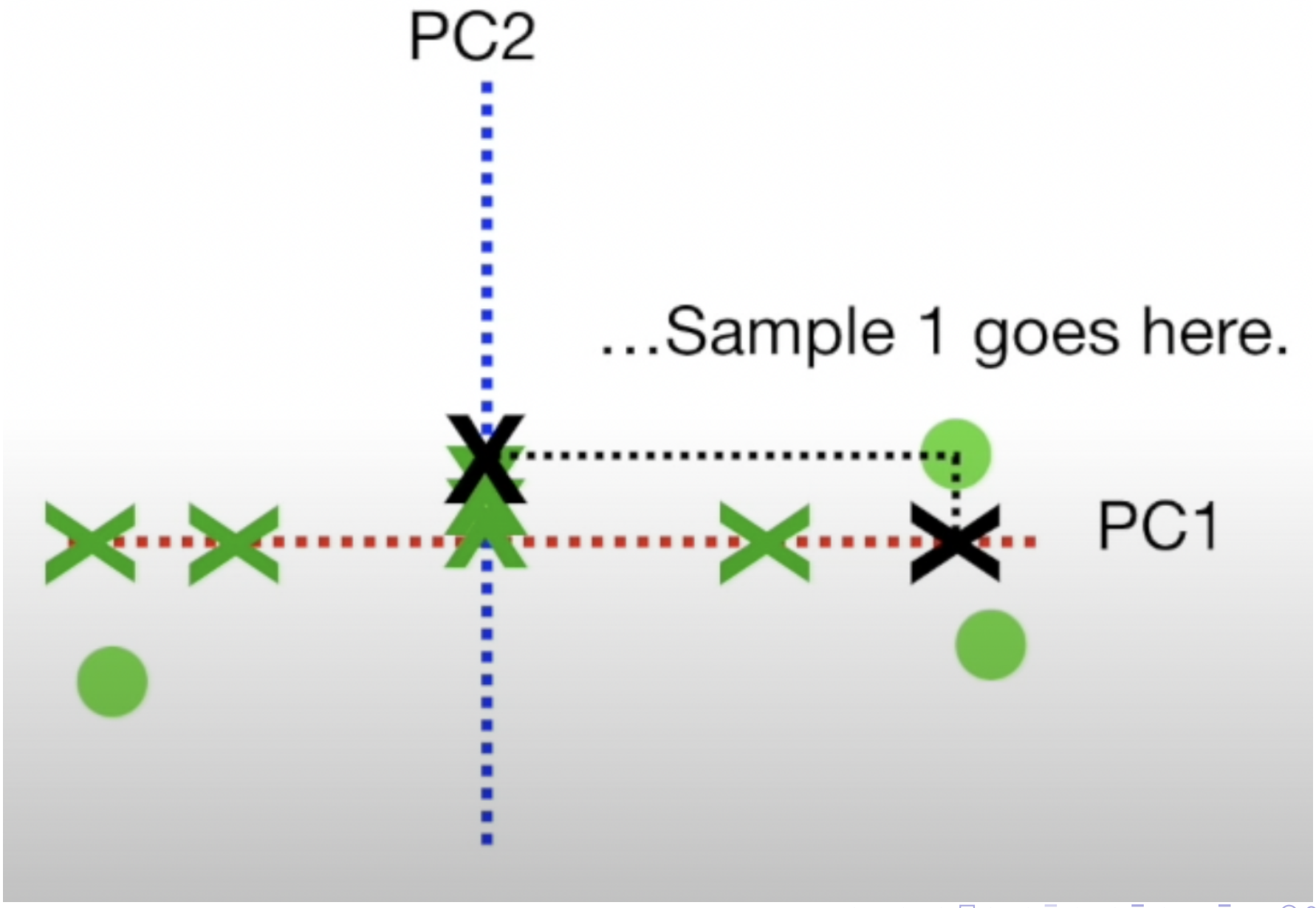
…then we use the projected points to find where the samples go in the PCA plot.




PC2
For example, these
projected points
correspond to Sample 6…
PC1


PC2
…so Sample 6 goes here.
PC1

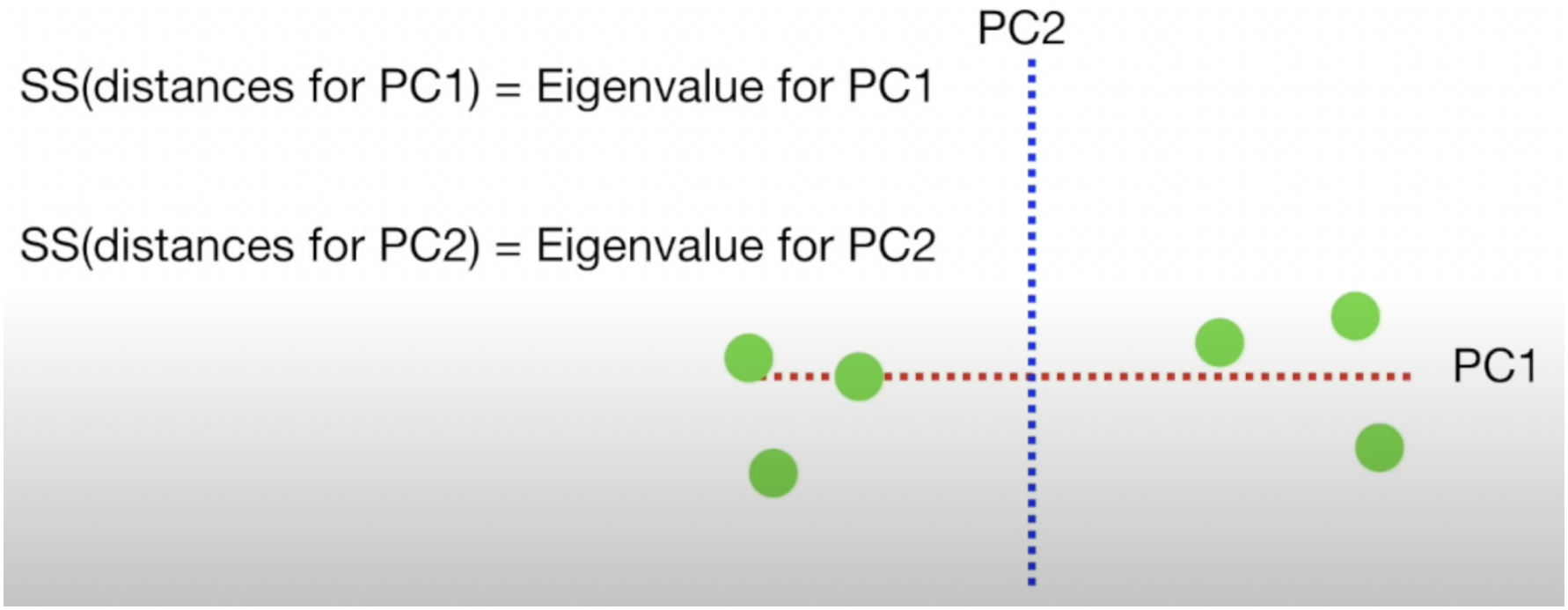

PC2
…Sample 2 goes here.
PC1


PC2
…Sample 1 goes here.
PC1

Remember the eigenvalues?

SS(distances for PC1) = Eigenvalue for PC1

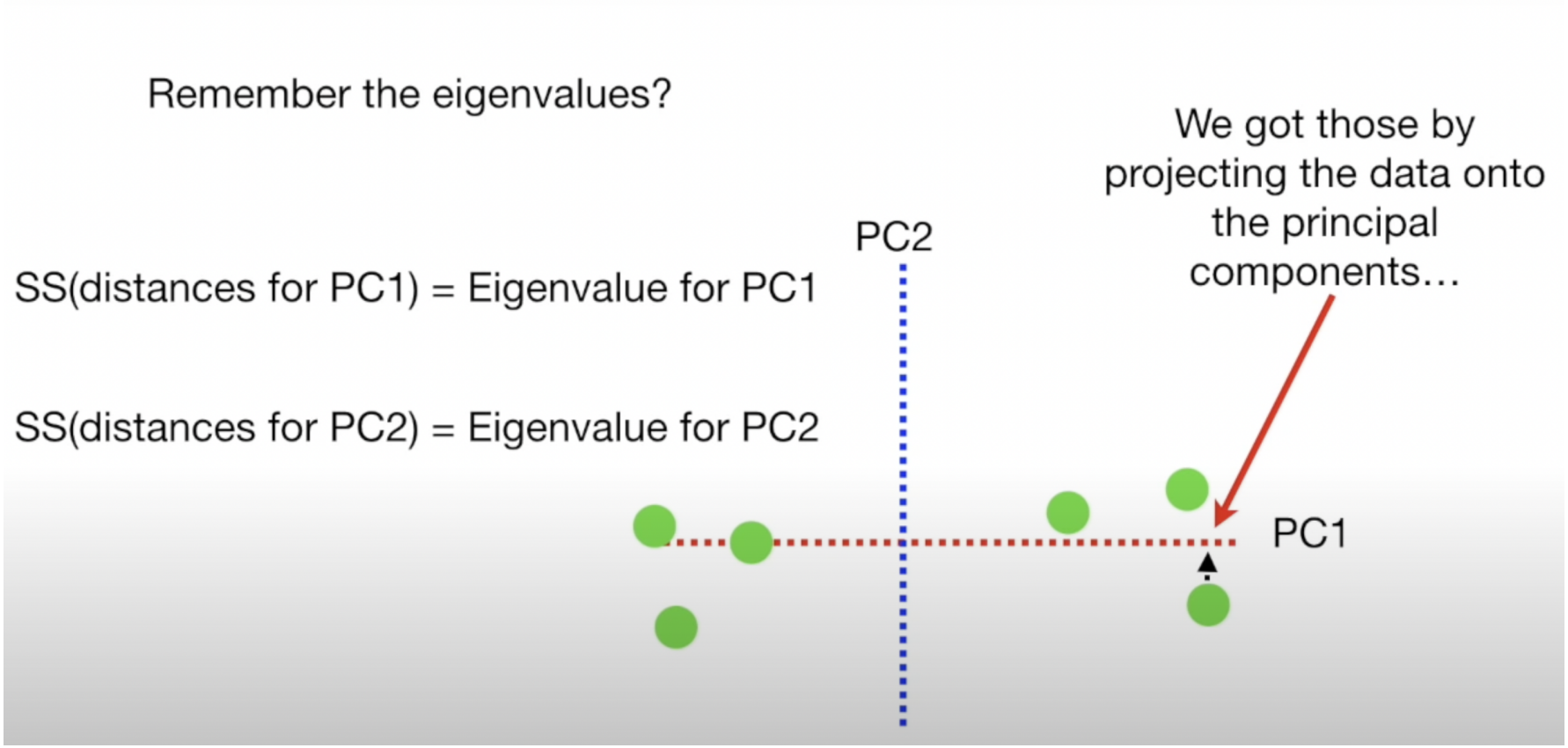
SS(distances for PC2) = Eigenvalue for PC2



Remember the eigenvalues?

SS(distances for PC1) = Eigenvalue for PC1

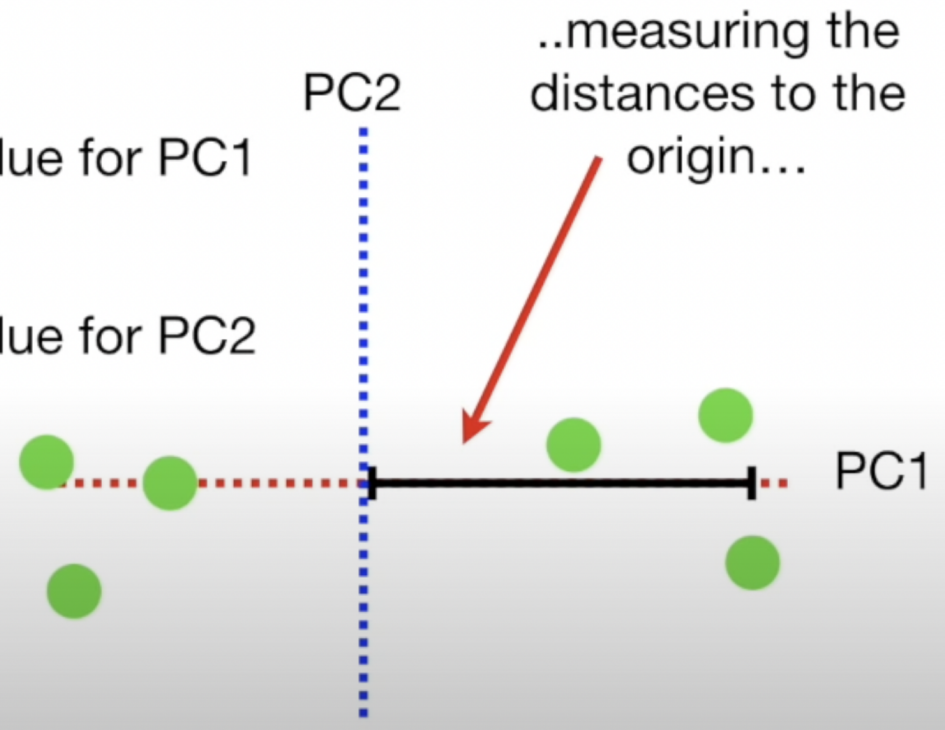
SS(distances for PC2) = Eigenvalue for PC2



Remember the eigenvalues?

SS(distances for PC1) = Eigenvalue for PC1

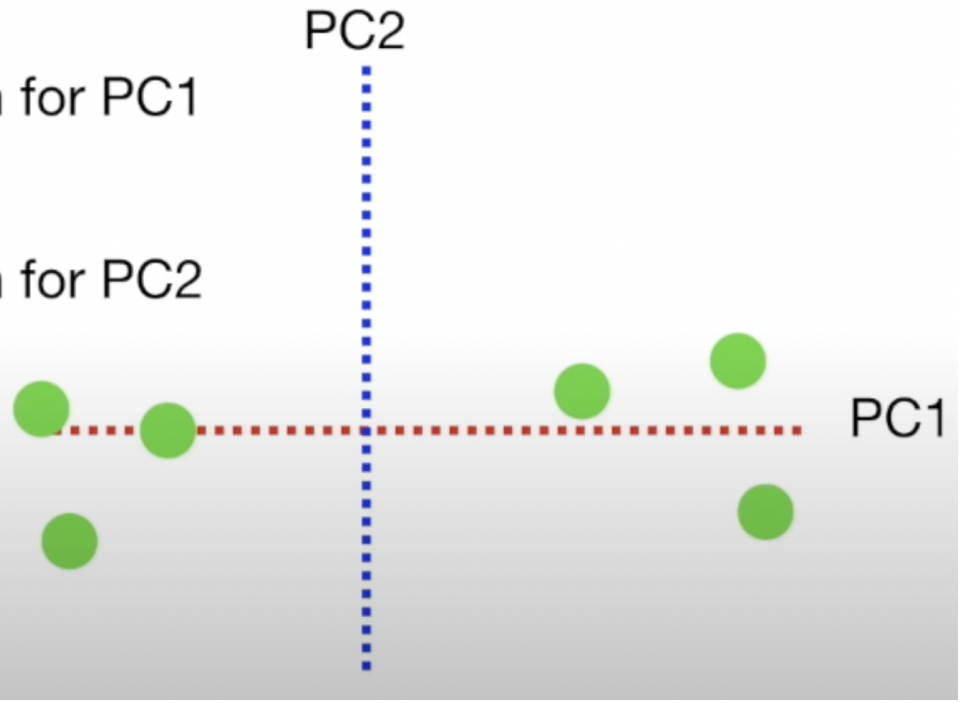
SS(distances for PC2) = Eigenvalue for PC2



We can convert them into variation around the origin (0, 0) by dividing by the sample size minus 1 (i.e. n - 1).

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

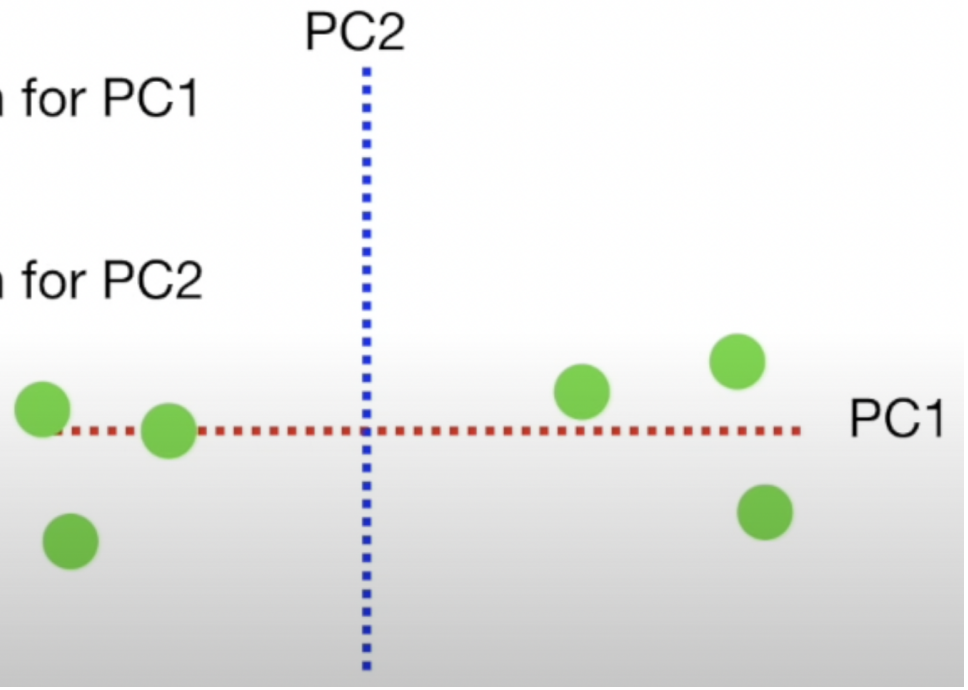
$$\frac{\text{SS(distances for PC2)}}{n - 1} = \text{Variation for PC2}$$



For the sake of the example, imagine that the Variation for **PC1 = 15**, and the variation for **PC2 = 3**.

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

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

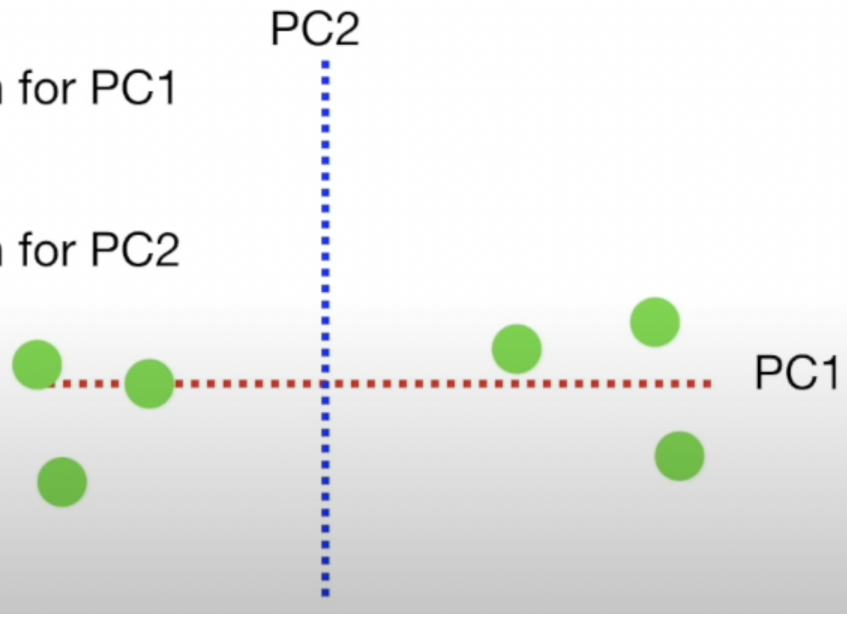


For the sake of the example, imagine that the Variation for **PC1 = 15**, and the variation for **PC2 = 3**.

That means that the total variation around both PCs is **15 + 3 = 18**...

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

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

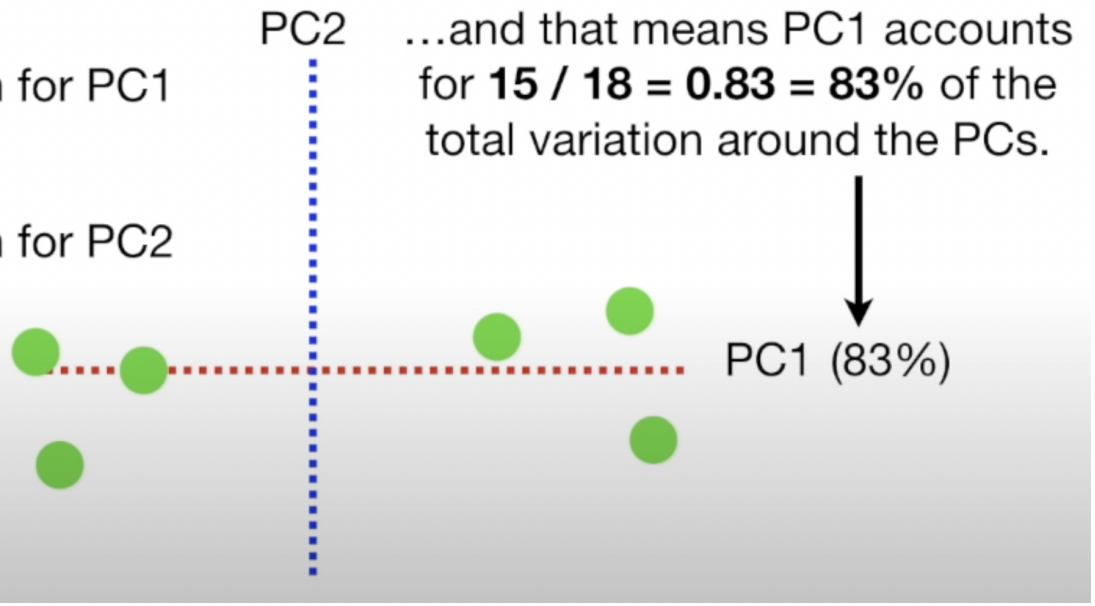


For the sake of the example, imagine that the Variation for **PC1 = 15**, and the variation for **PC2 = 3**.

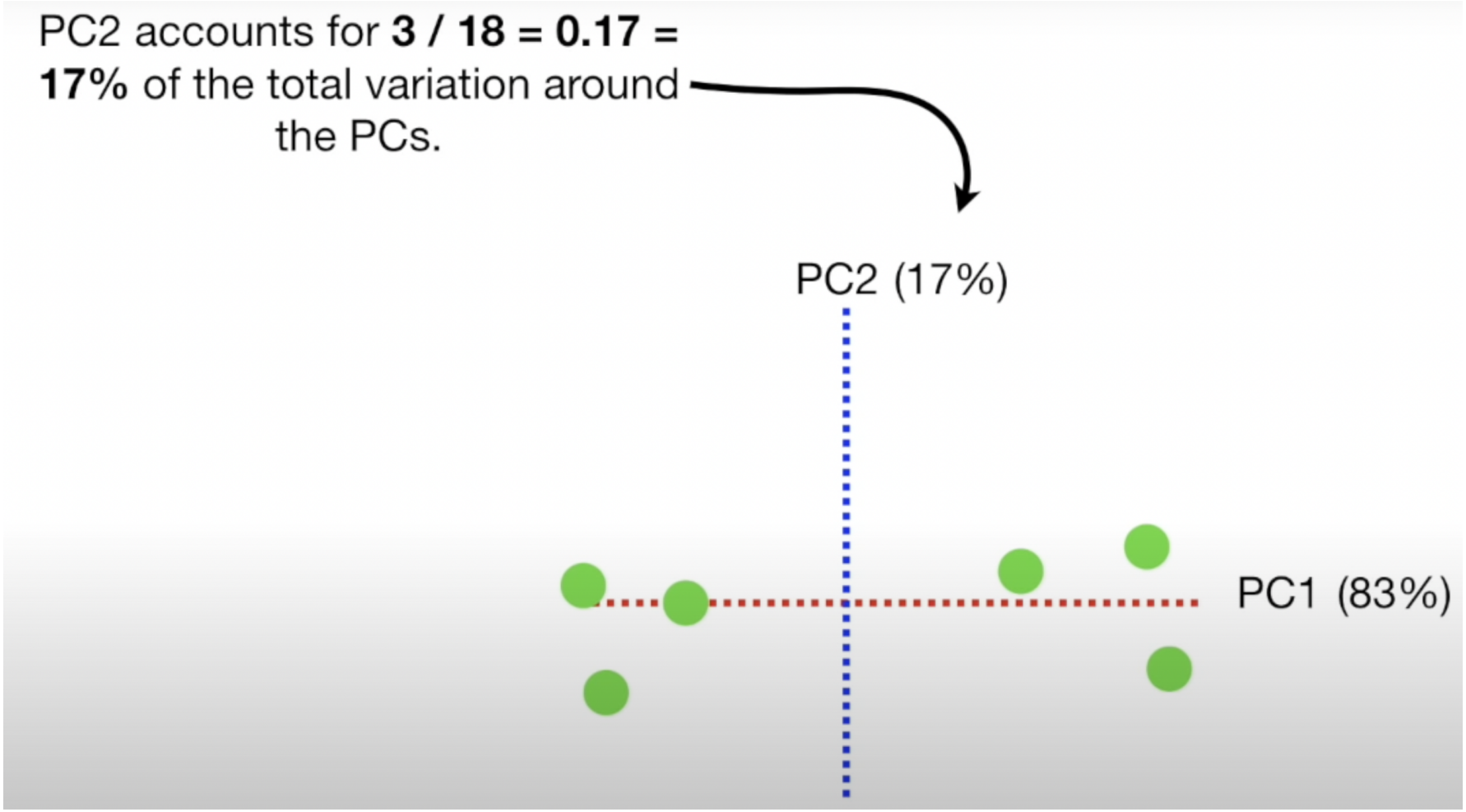
That means that the total variation around both PCs is **15 + 3 = 18**...

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

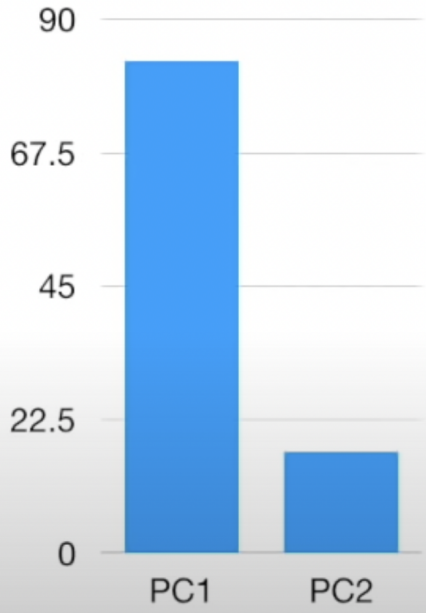
$$\frac{SS(\text{distances for PC2})}{n - 1} = \text{Variation for PC2}$$

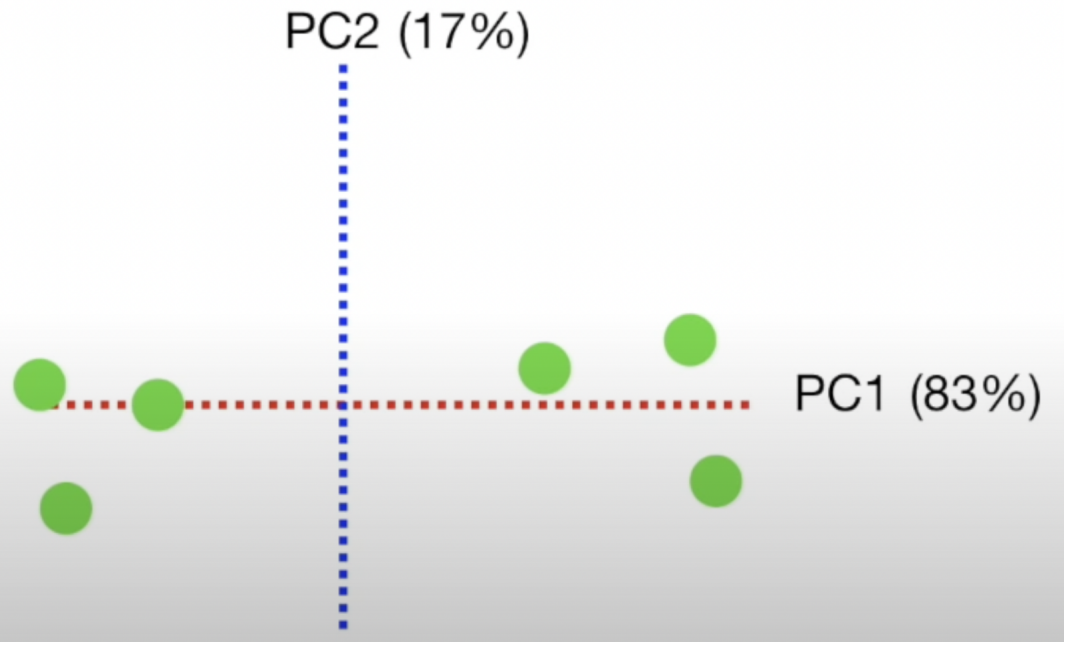


PC2 accounts for **3 / 18 = 0.17 = 17%** of the total variation around the PCs.



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

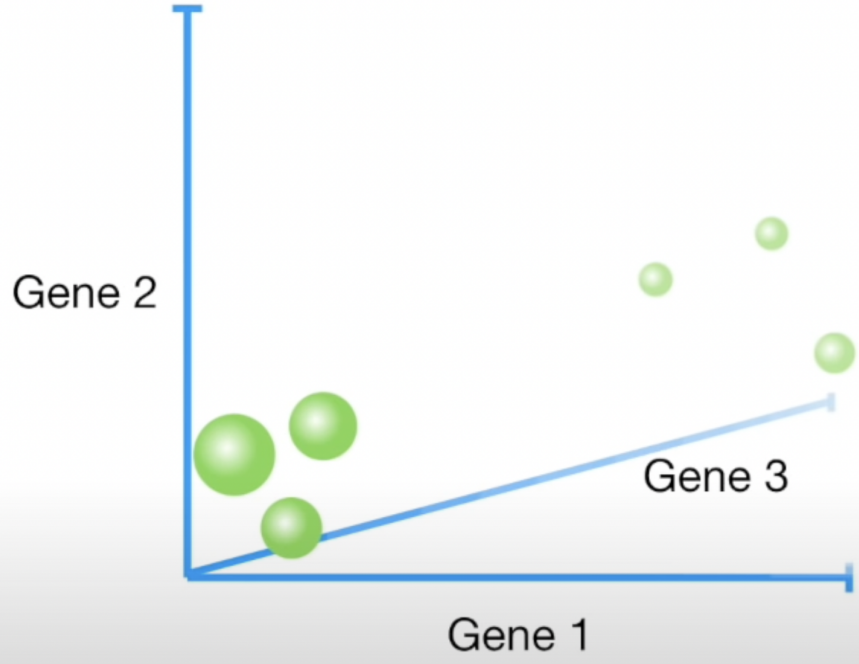


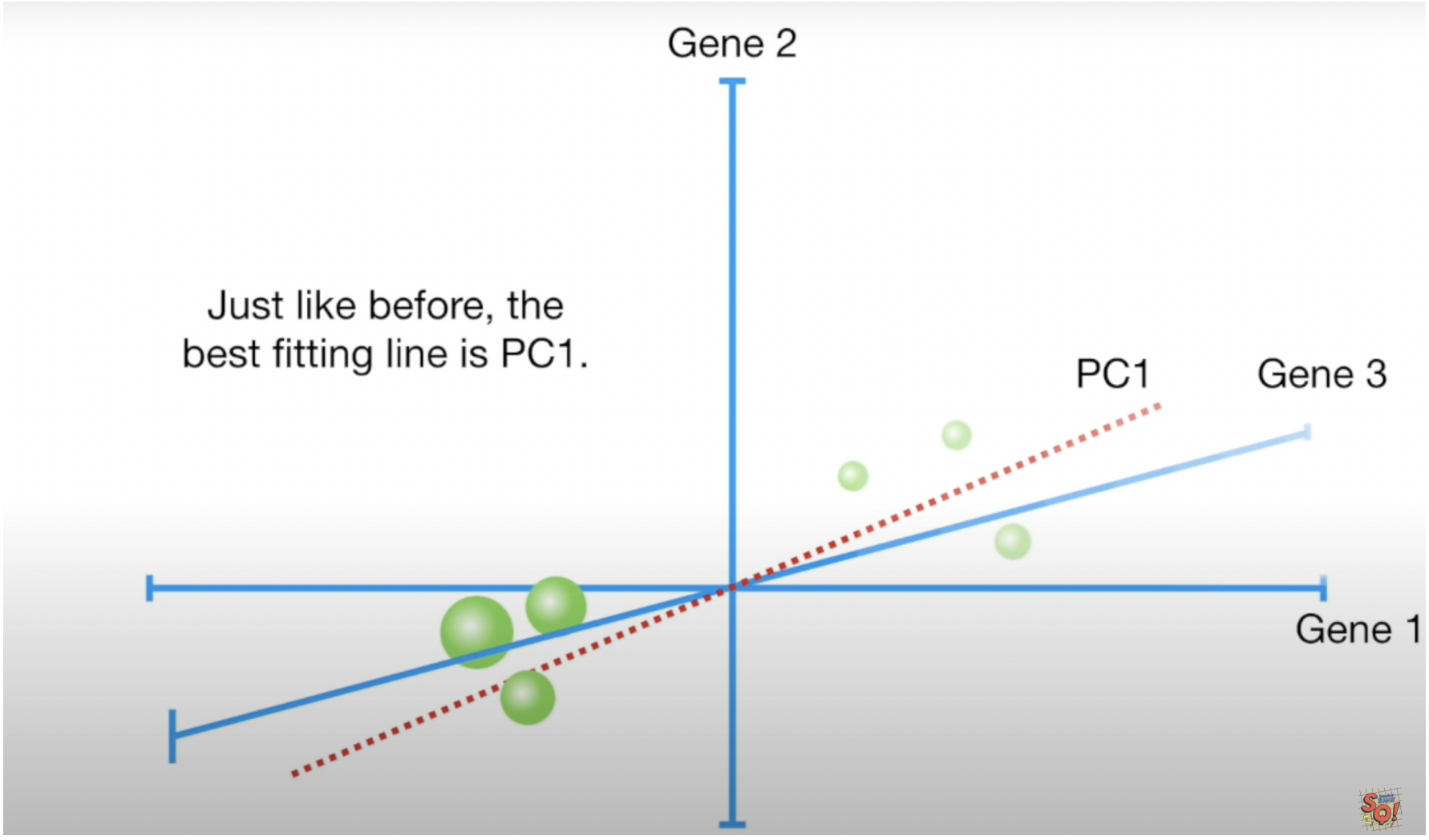


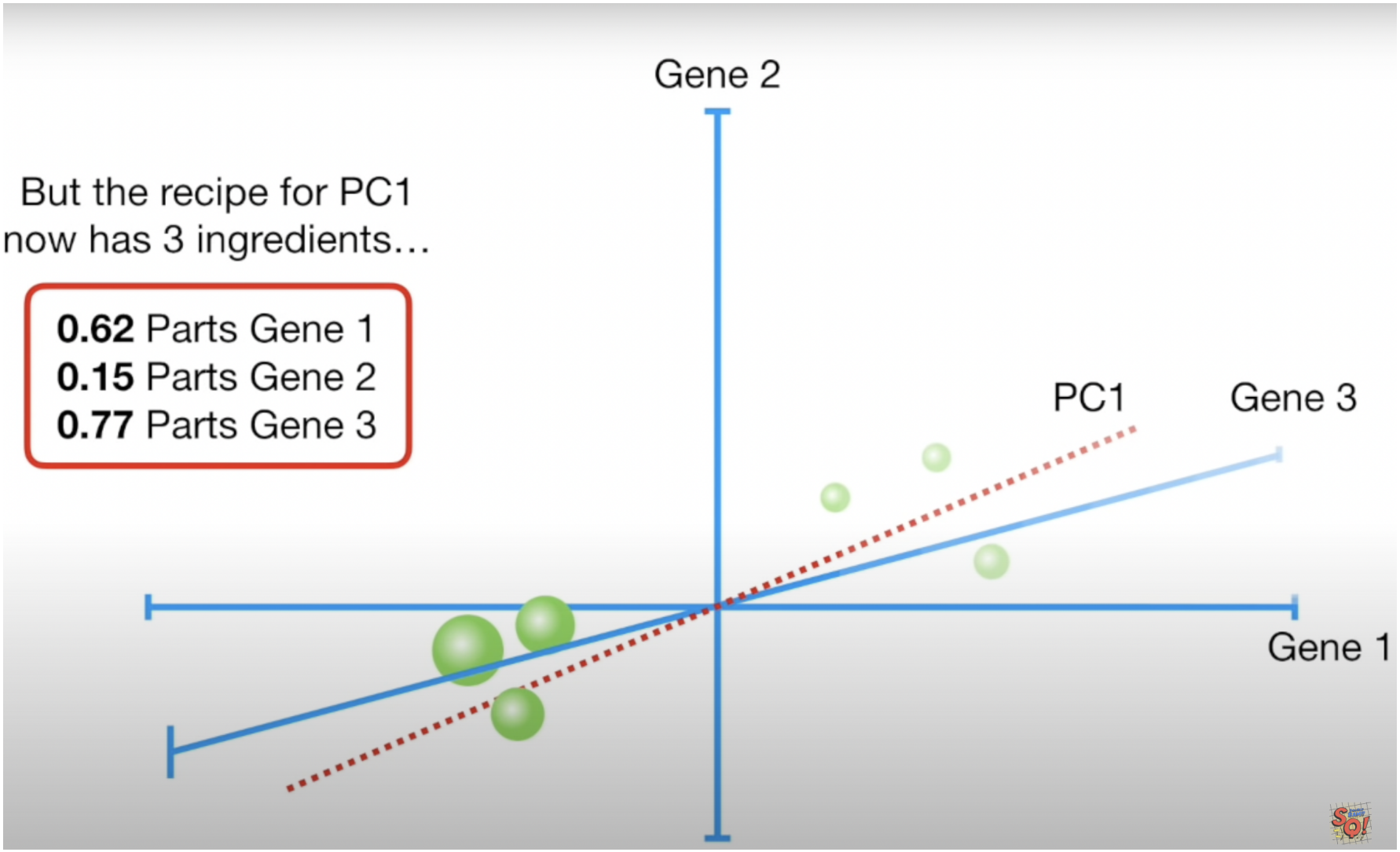
OK - Now let's quickly go through a slightly more complicated example

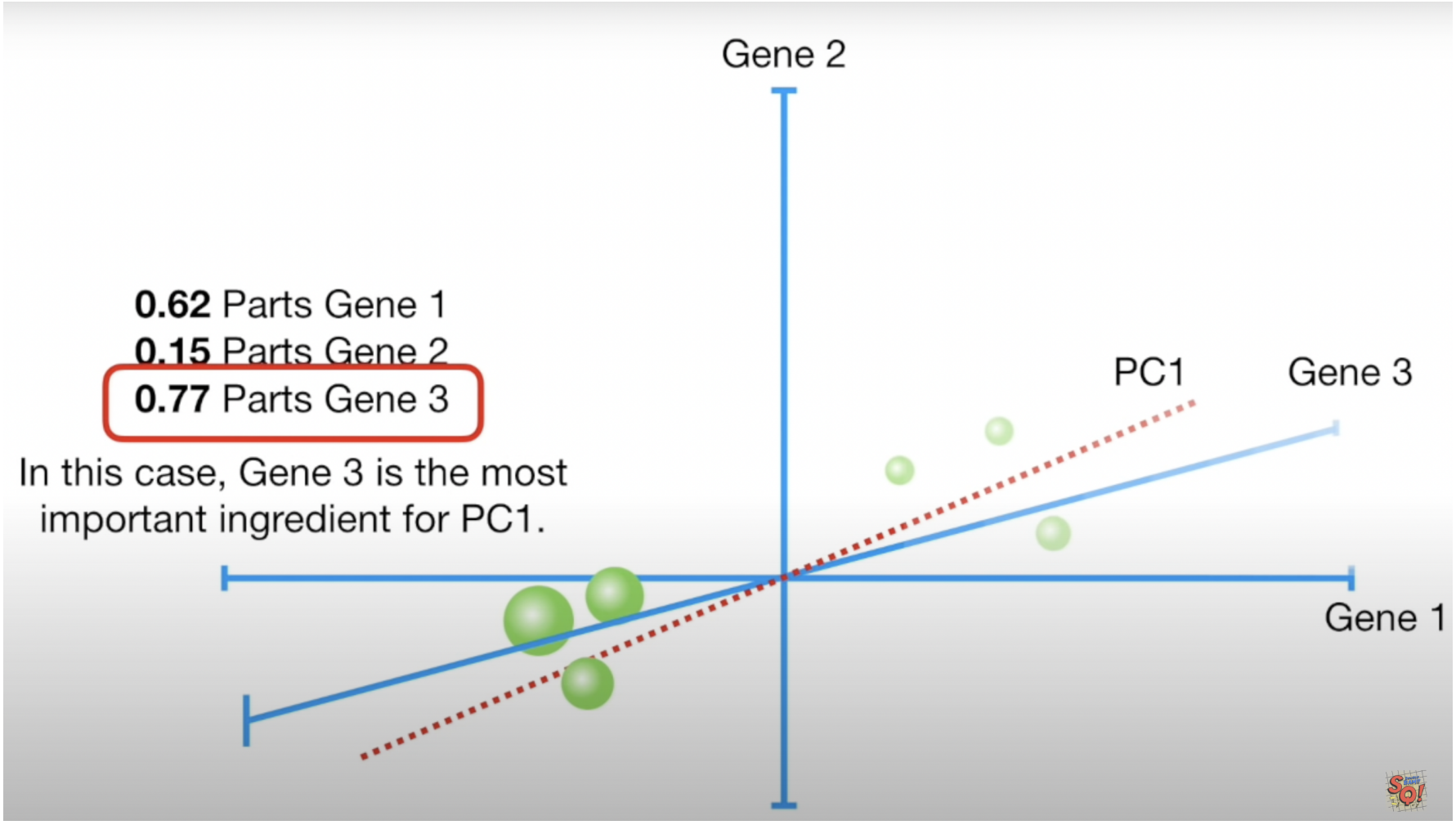
PCA with 3 variables (in this case, that means 3 genes) is pretty much the same as 2 variables…

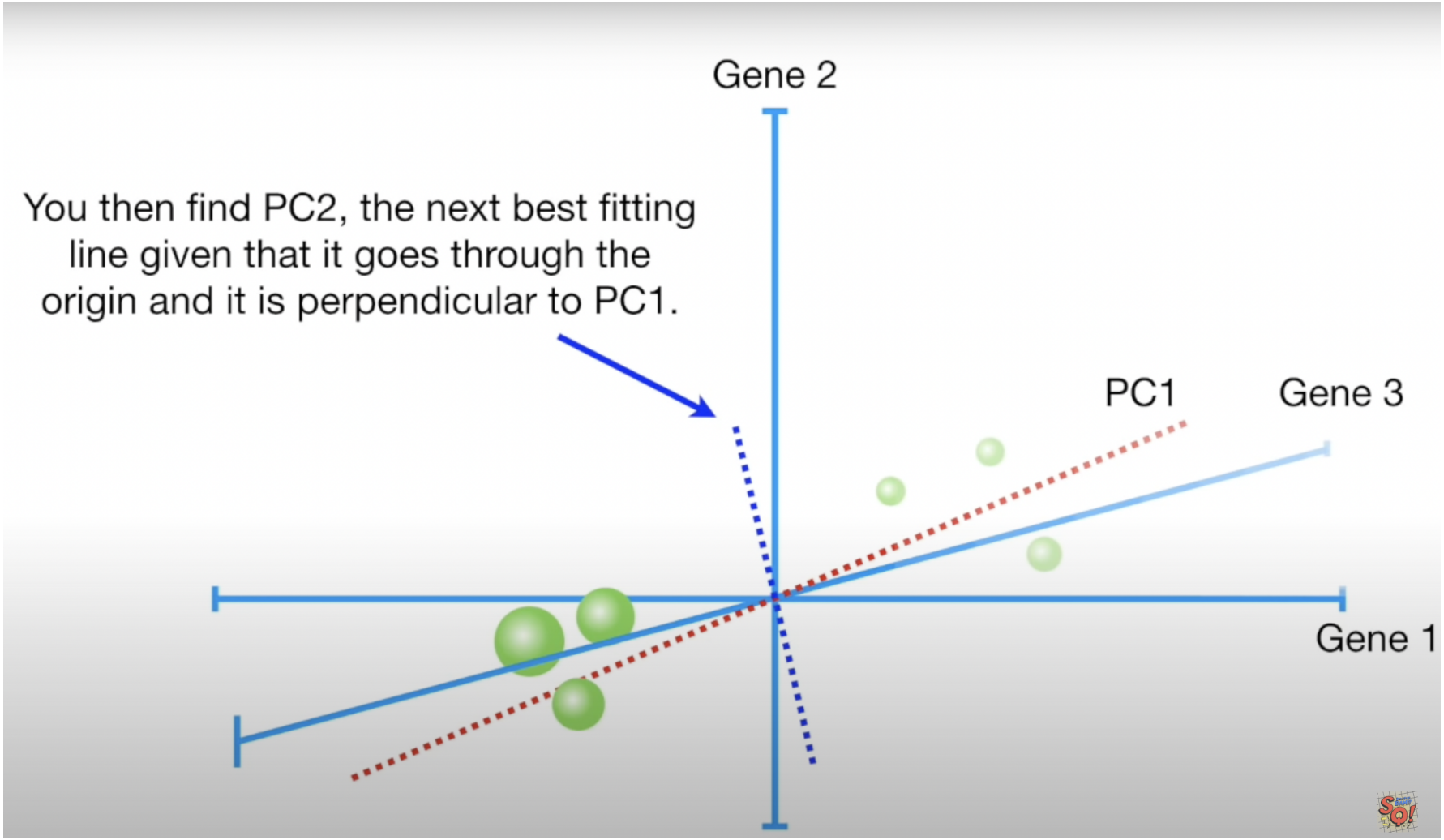
	Mouse 1	Mouse 2	Mouse 3	Mouse 4	Mouse 5	Mouse 6
Gene 1	10	11	8	3	2	1
Gene 2	6	4	5	3	2.8	1
Gene 3	12	9	10	2.5	1.3	2

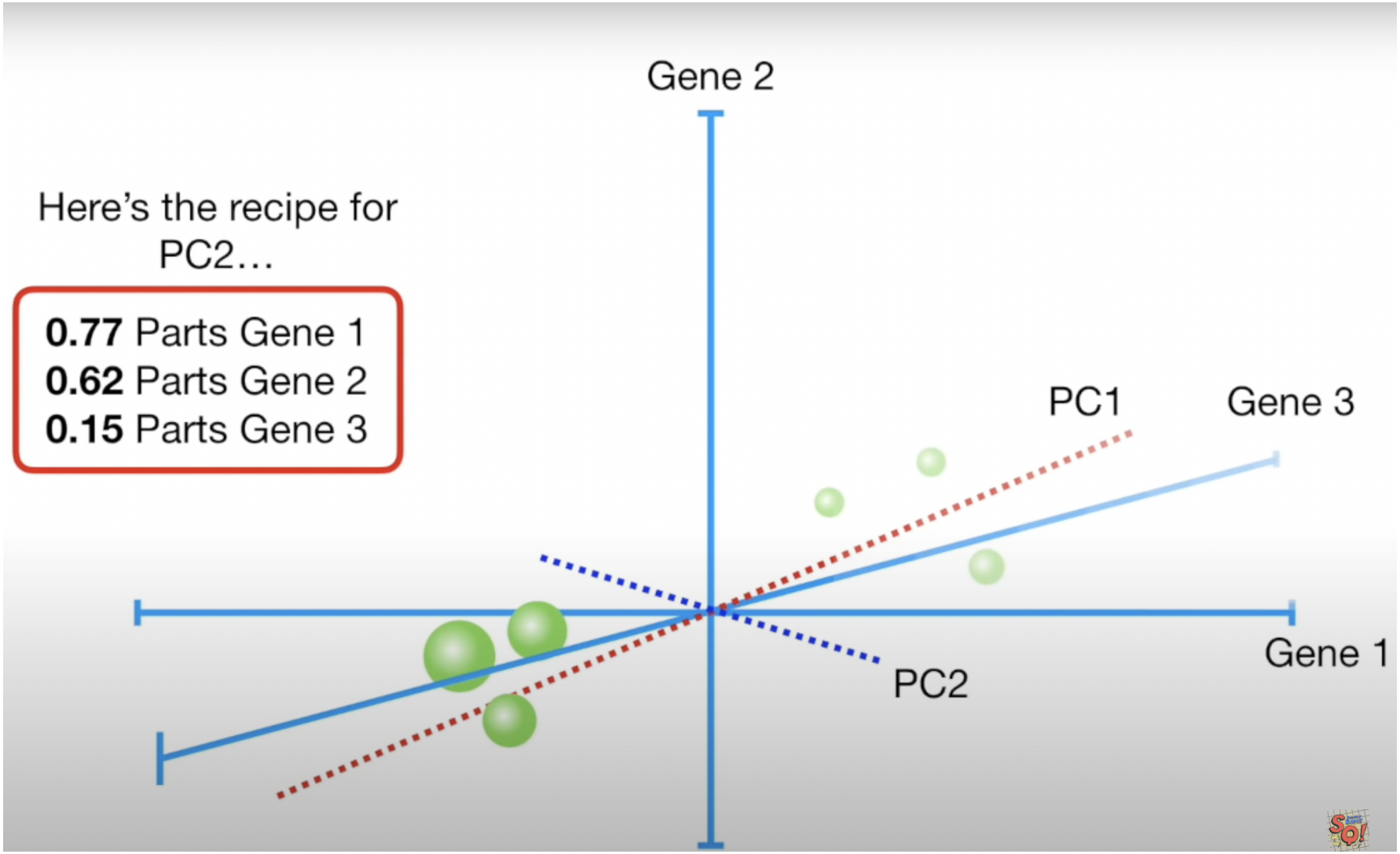


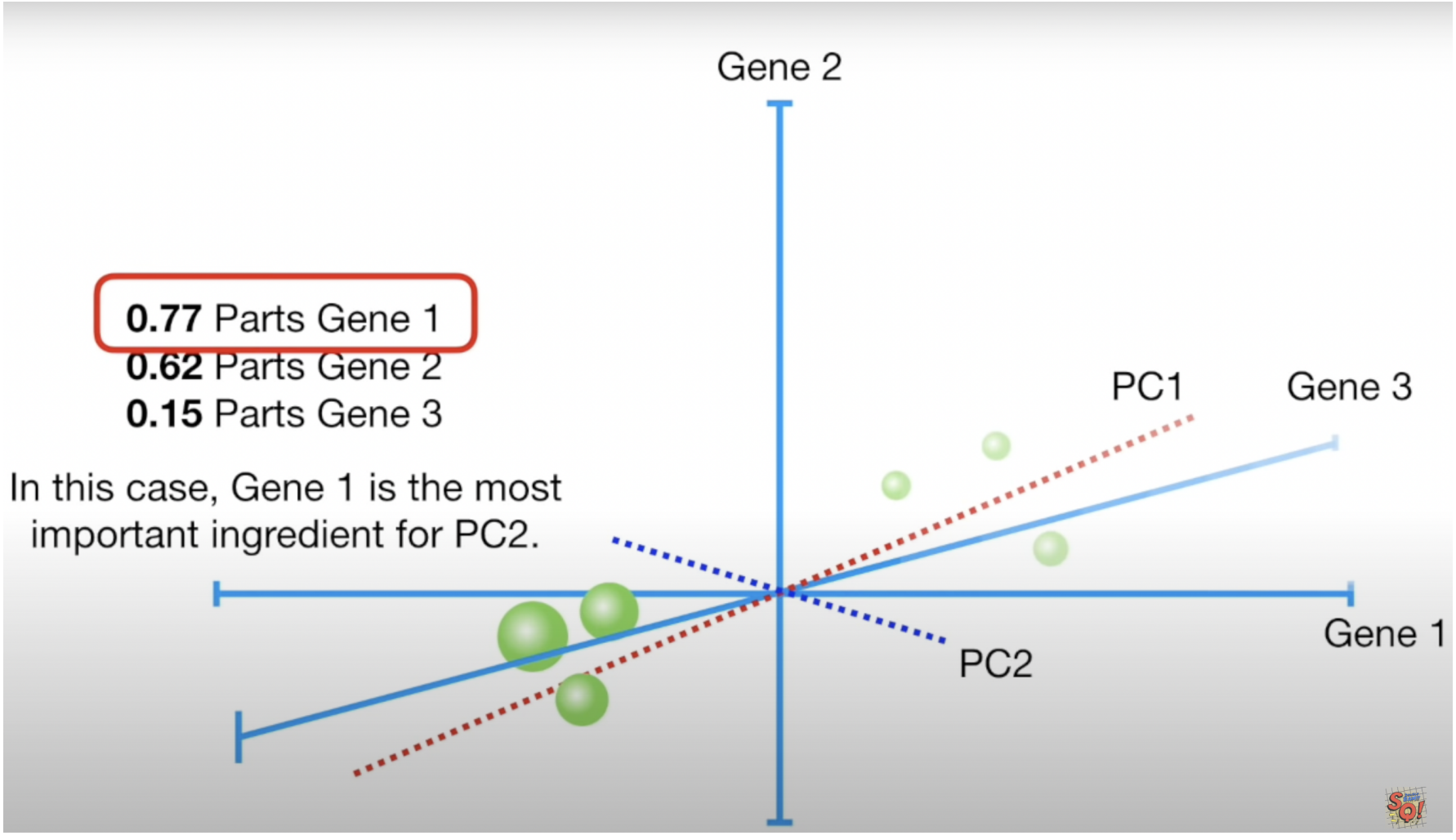

Gene 2
Just like before, the
best fitting line is PC1.
PC1
Gene 3
Gene 1

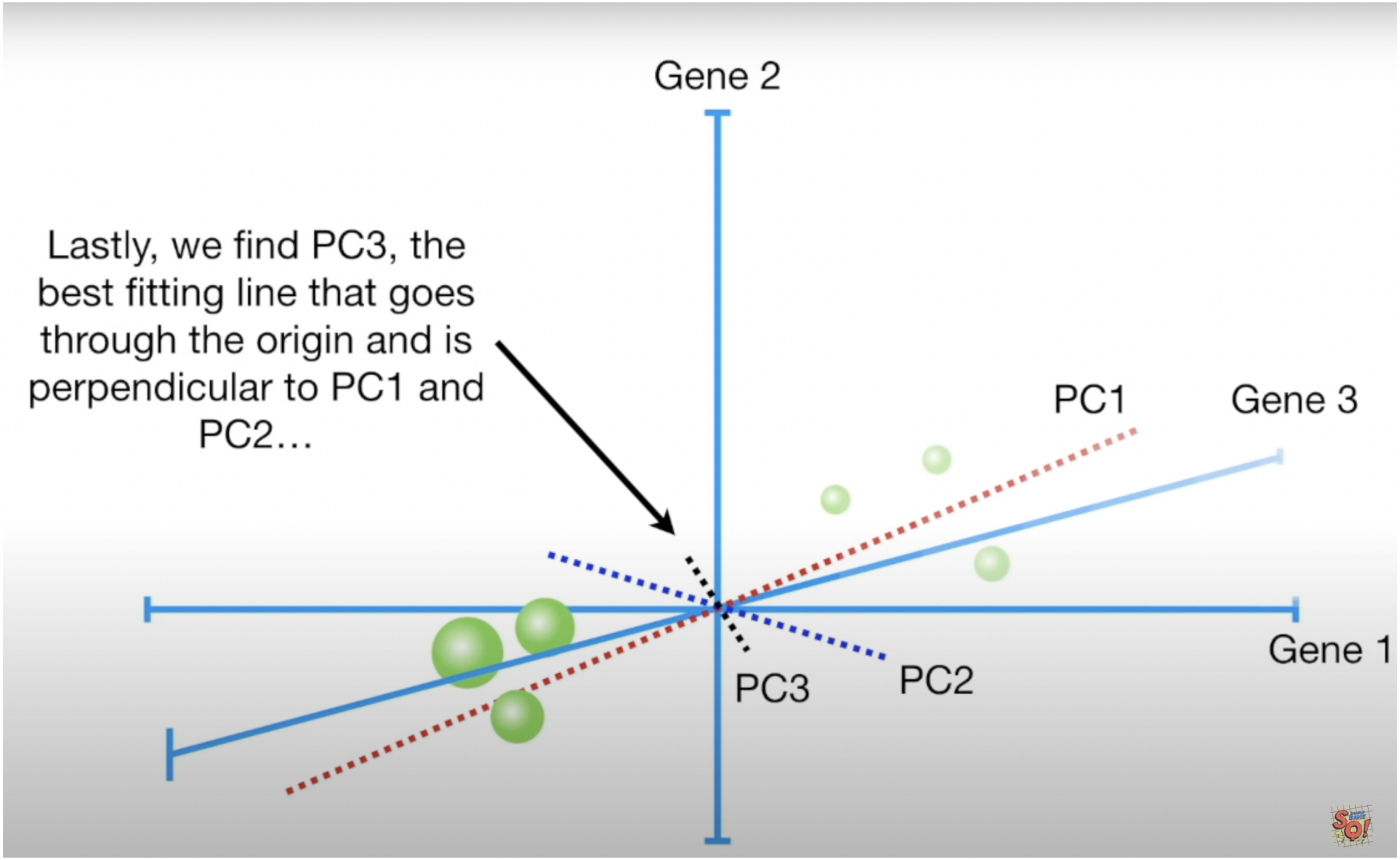

But the recipe for PC1
now has 3 ingredients…
0.62 Parts Gene 1
0.15 Parts Gene 2
0.77 Parts Gene 3
Gene 2
PC1
Gene 3
Gene 1


Gene 2
0.62 Parts Gene 1
0.15 Parts Gene 2
0.77 Parts Gene 3
In this case, Gene 3 is the most important ingredient for PC1.
PC1
Gene 3
Gene 1


Gene 2
You then find PC2, the next best fitting line given that it goes through the origin and it is perpendicular to PC1.
PC1
Gene 3
Gene 1

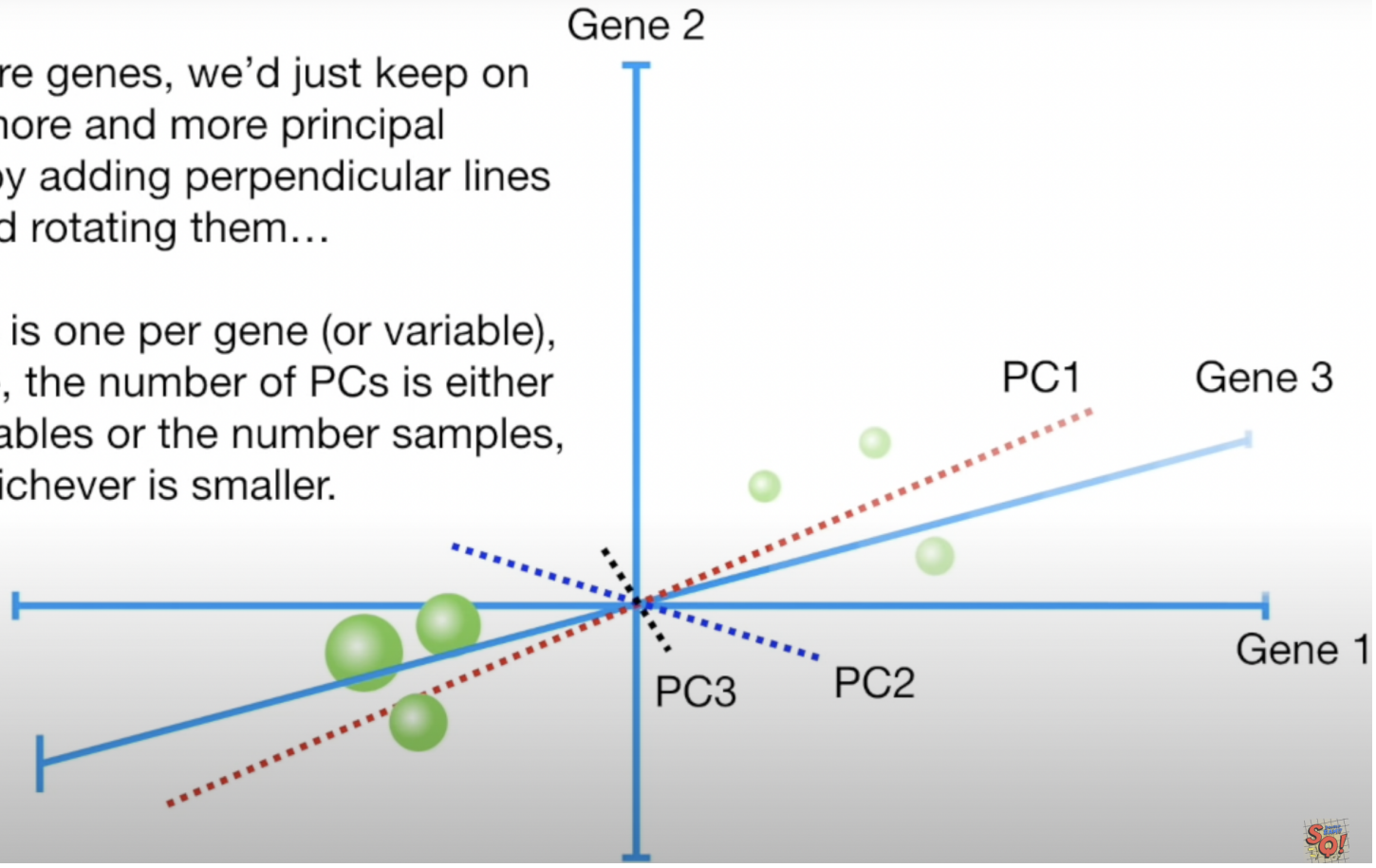

Here's the recipe for PC2…
0.77 Parts Gene 1
0.62 Parts Gene 2
0.15 Parts Gene 3
Gene 2
PC1
Gene 3
Gene 1
PC2


Gene 2
0.77 Parts Gene 1
0.62 Parts Gene 2
0.15 Parts Gene 3
In this case, Gene 1 is the most important ingredient for PC2.
PC1
Gene 3
Gene 1
PC2

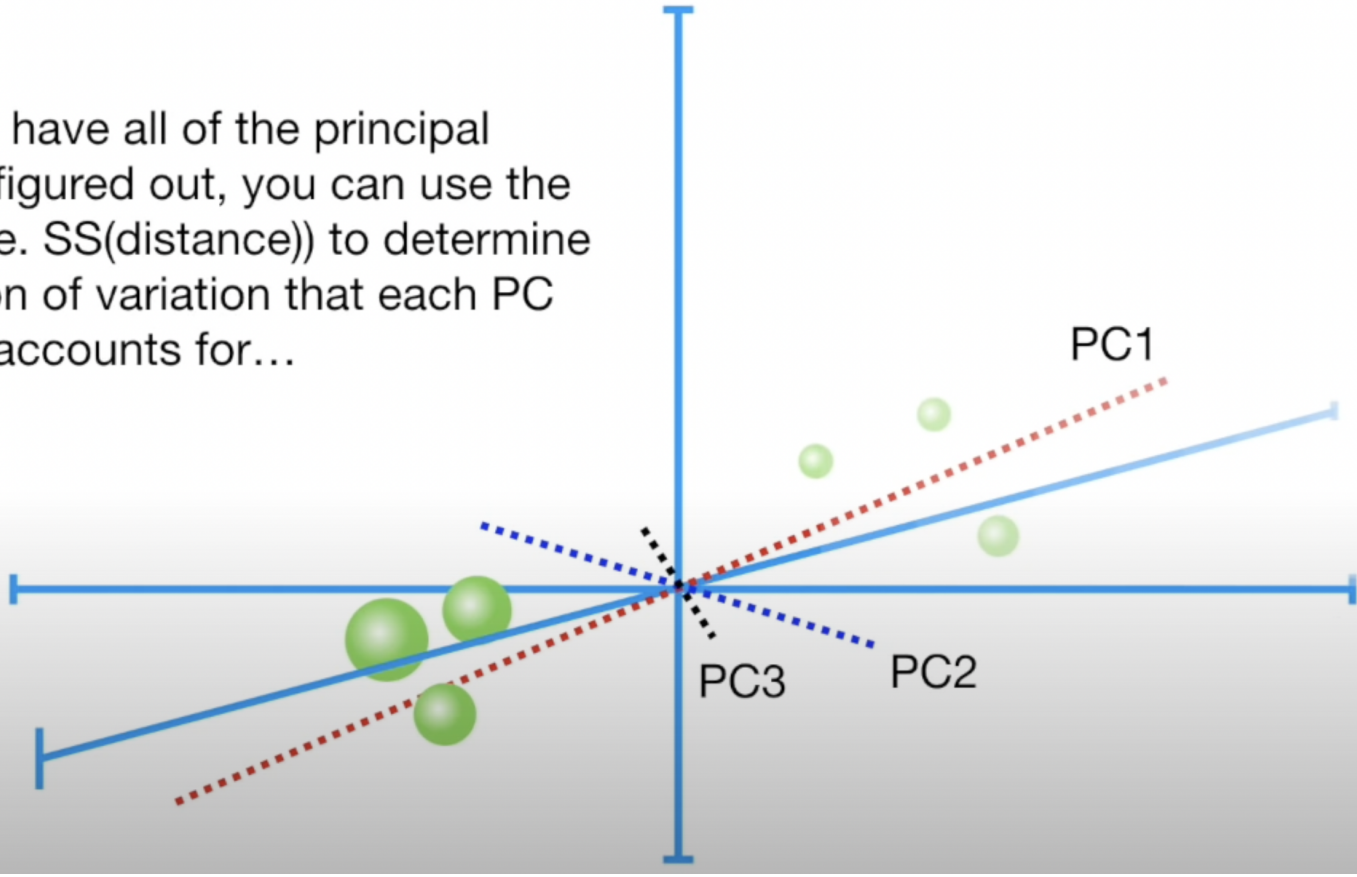

Gene 2
Lastly, we find PC3, the best fitting line that goes through the origin and is perpendicular to PC1 and PC2…
PC1
Gene 3
Gene 1
PC3
PC2

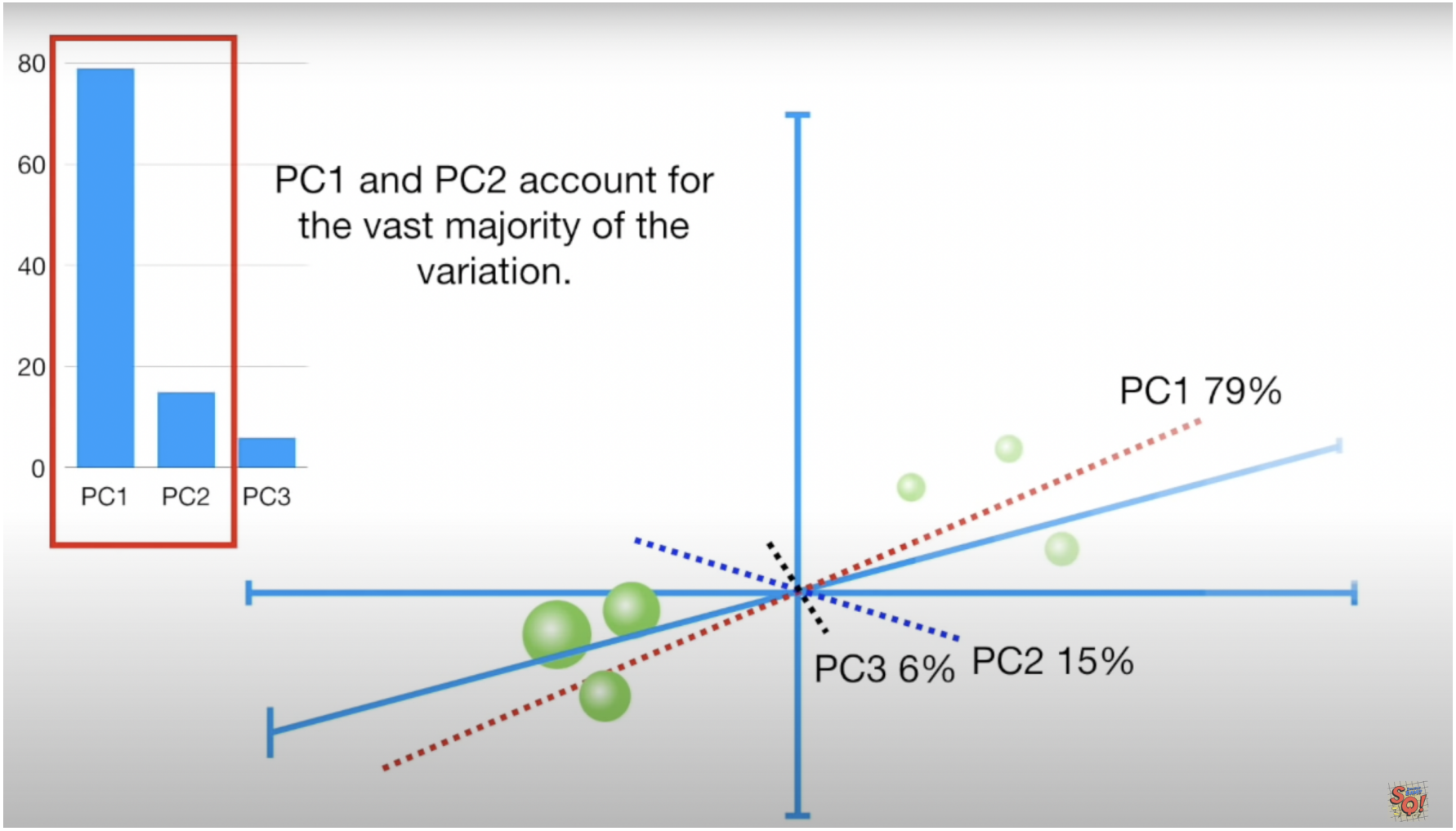
If we had more genes, we'd just keep on finding more and more principal components by adding perpendicular lines and rotating them…

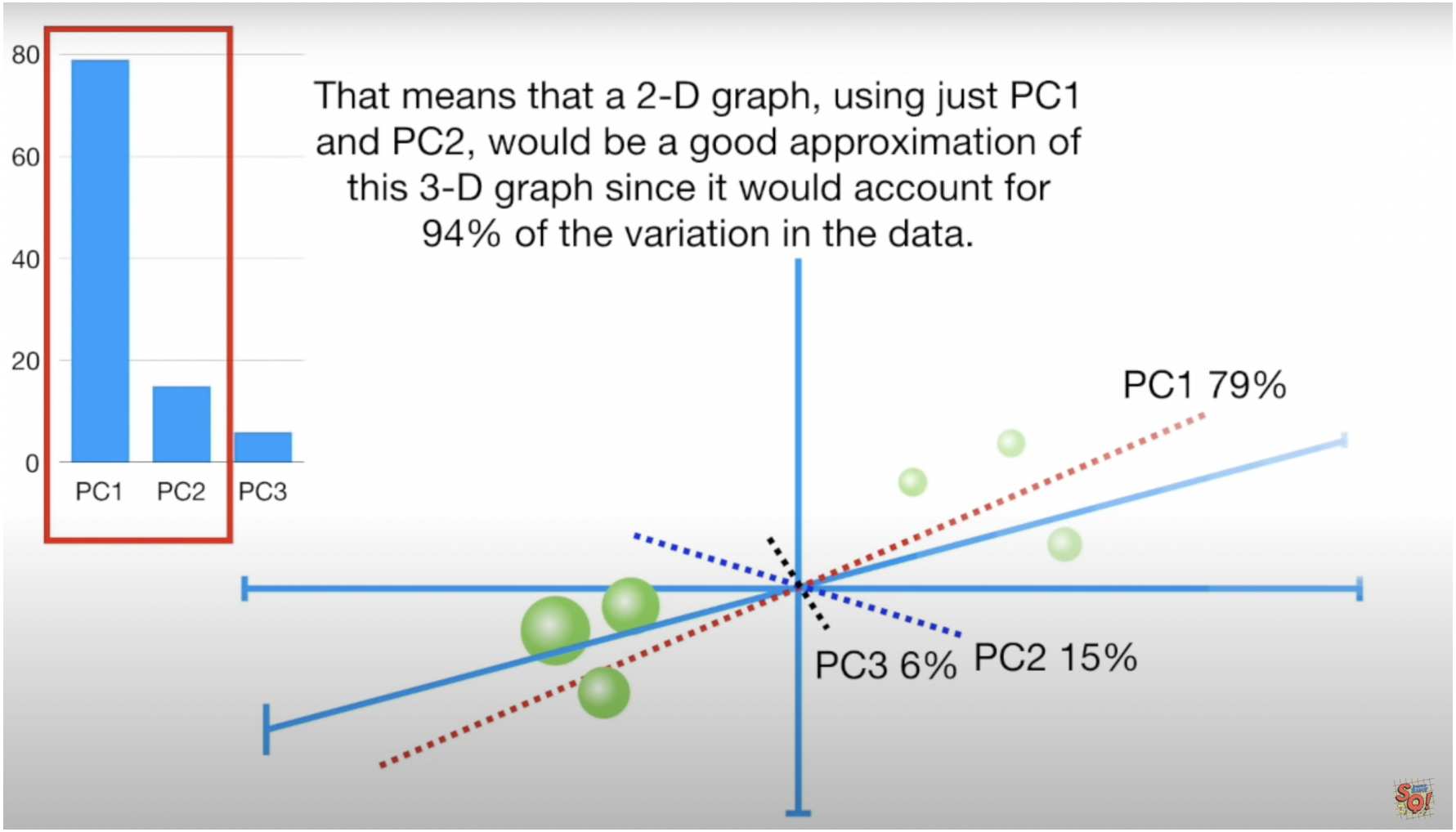
In theory there is one per gene (or variable), but in practice, the number of PCs is either number of variables or the number samples, whichever is smaller.



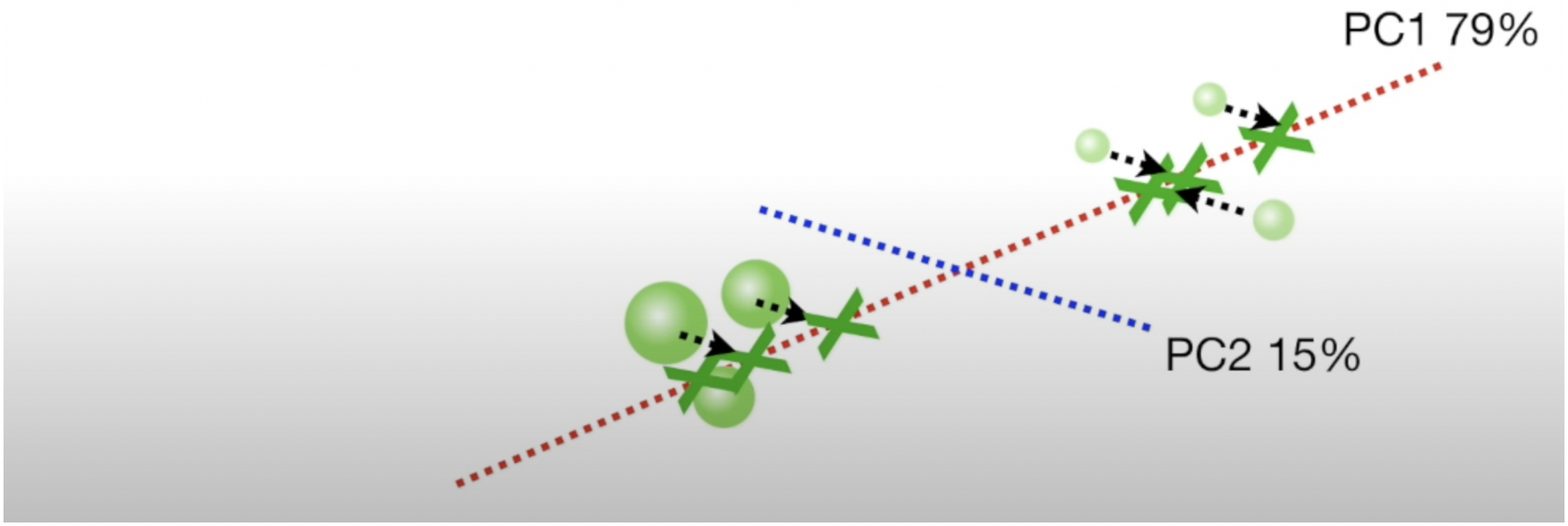
Once you have all of the principal components figured out, you can use the eigenvalues (i.e. SS(distance)) to determine the proportion of variation that each PC accounts for…



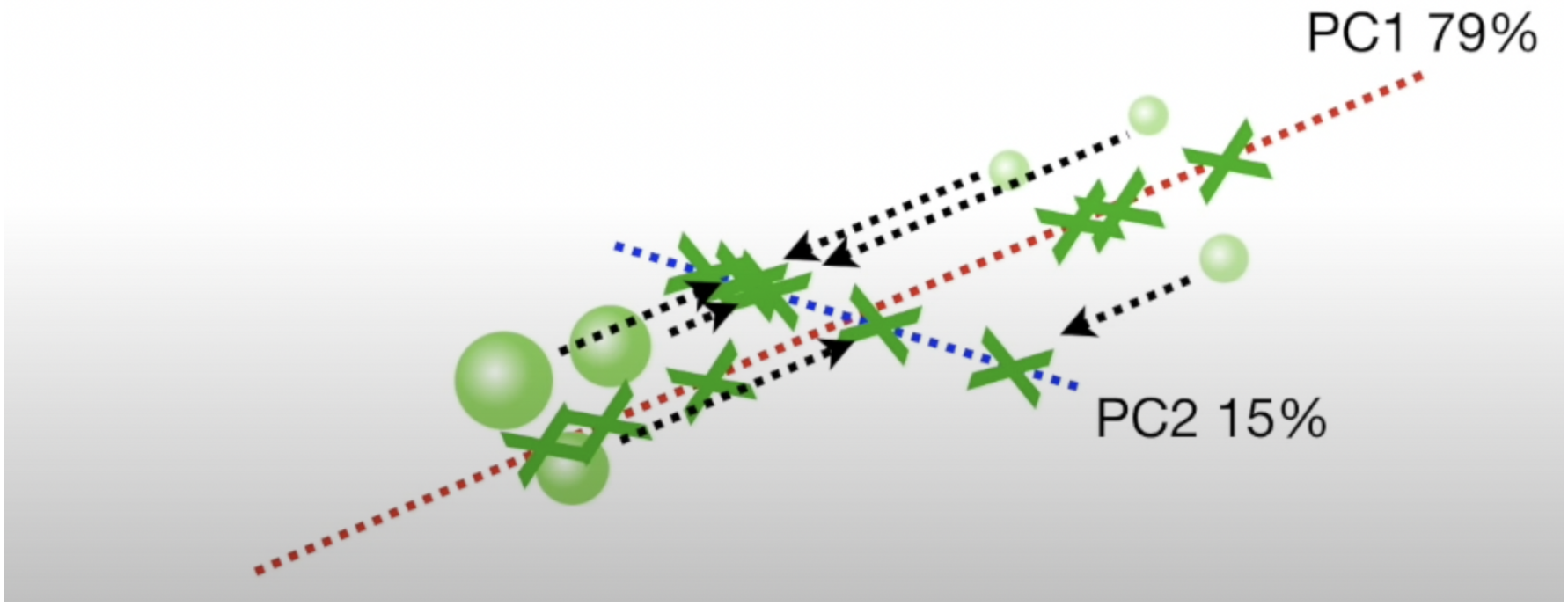

80
60
40
20
0
PC1
PC2
PC3
PC1 and PC2 account for the vast majority of the variation.
PC1 79%
PC3 6%
PC2 15%


80
60
40
20
0
PC1
PC2
PC3
That means that a 2-D graph, using just PC1 and PC2, would be a good approximation of this 3-D graph since it would account for 94% of the variation in the data.
PC1 79%
PC3 6%
PC2 15%

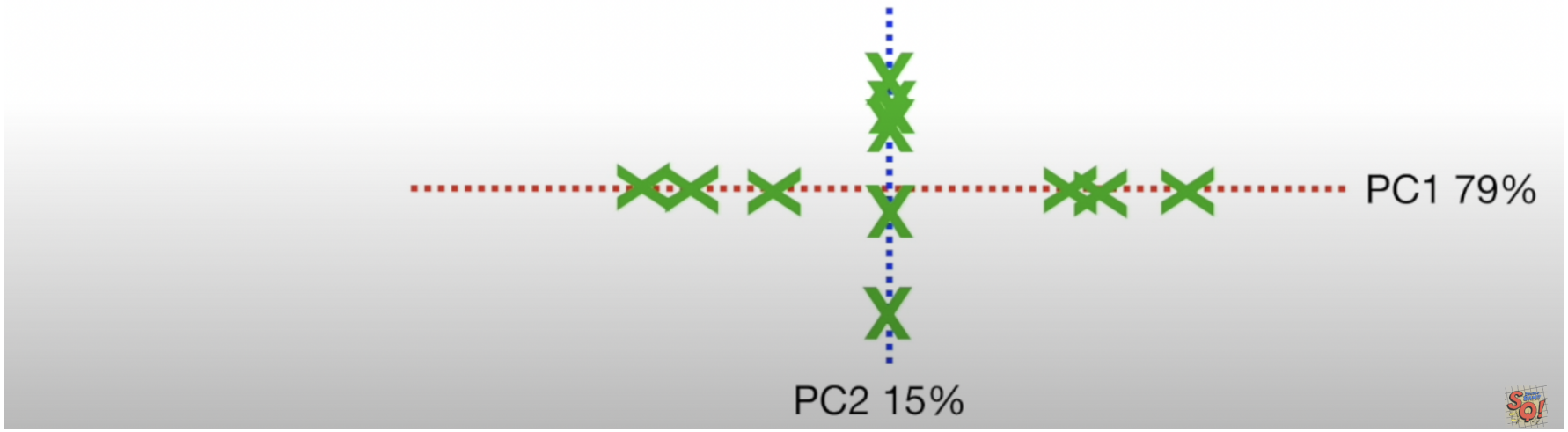
Then project the samples onto PC1…

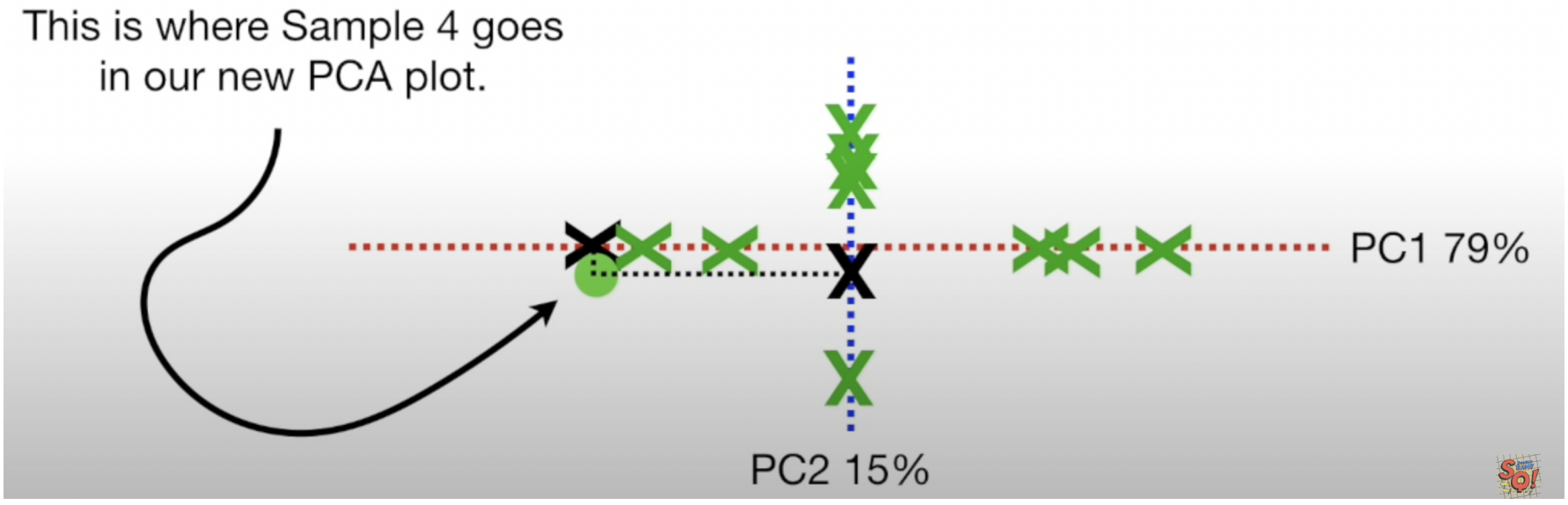


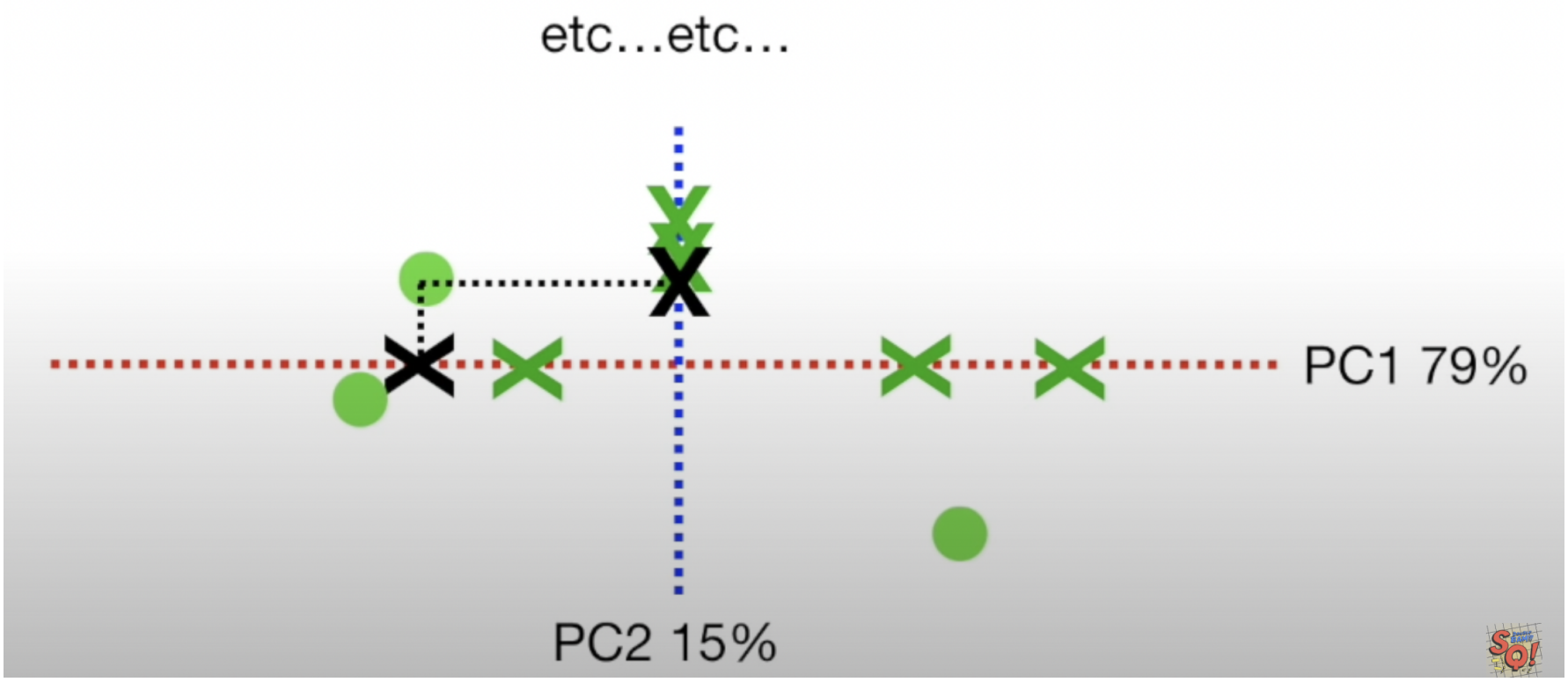
…and PC2.



Then we rotate so that PC1 is horizontal and PC2 is vertical (this just makes it easier to look at).




This is where Sample 4 goes
in our new PCA plot.
PC1 79%
PC2 15%


etc…etc…
PC1 79%
PC2 15%

Narcissistic Personality Disorder

Narcissistic Personality Disorder is a psychological problem in which people's functioning and relationships with others are damaged by excessive self-importance, lack of empathy etc

Assessed by Narcissistic Personality Inventory: 40 'forced choice' questions.

	A	B
1.	○ I have a natural talent for influencing people.	○ I am not good at influencing people.
2.	○ Modesty doesn't become me.	○ I am essentially a modest person.
3.	○ I would do almost anything on a dare.	○ I tend to be a fairly cautious person.
4.	○ When people compliment me I sometimes get embarrassed.	○ I know that I am good because everybody keeps telling me so.
5.	○ The thought of ruling the world frightens the hell out of me.	○ If I ruled the world it would be a better place.
6.	○ I can usually talk my way out of anything.	○ I try to accept the consequences of my behavior.
7.	○ I prefer to blend in with the crowd.	○ I like to be the center of attention.
8.	○ I will be a success.	○ I am not too concerned about success.
9.	○ I am no better or worse than most people.	○ I think I am a special person.
10.	○ I am not sure if I would make a good leader.	○ I see myself as a good leader.
11.	○ I am assertive.	○ I wish I were more assertive.
12.	○ I like to have authority over other people.	○ I don't mind following orders.
13.	○ I find it easy to manipulate people.	○ I don't like it when I find myself manipulating people.
14.	○ I insist upon getting the respect that is due me.	○ I usually get the respect that I deserve.

Loadings: Raskin et al, 1988

Wanted to understand underpinnings of the (standard) NPI

Had 1000 students take test, then fit 7 principal components (explained 52% of response variance)

→ Authority, Self-Sufficiency, Superiority, Exhibitionism, Exploitativeness, Vanity, Entitlement.

NB these labels are not in original data: imposed by researcher after studying!

Then looked at factor *loadings*: correlation between items and factors.

and Squared factor loading is percent of variance in that variable explained by the factor

Narcissistic Personality Inventory Items and Principal-Component Loadings

Items	Loadings					
	1	2	3	4	5	6
47. I would prefer to be a leader.	.83	.00	−.07	.04	−.12	.07
15. I see myself as a good leader.	.83	.16	.09	−.12	.06	.03
13. I will be a success.	.67	.00	−.09	−.14	−.14	.17
46. People always seem to recognize my authority.	.66	.02	.06	−.06	.06	.00
2. I have a natural talent for influencing people.	.66	−.15	.02	−.02	.29	.03
16. I am assertive.	.56	.18	−.02	.22	−.02	−.03
17. I like to have authority over other people.	.56	.08	−.08	.18	.08	.05
50. I am a born leader.	.35	.20	.22	.00	.09	−.14
30. I rarely depend on anyone else to get things done.	.02	.61	−.17	.04	.04	.10
23. I like to take responsibility for making decisions.	.28	.59	−.23	.23	−.12	.00
53. I am more capable than other people.	−.19	.57	.16	.07	.11	.01
45. I can live my life in any way I want to.	−.13	.46	.29	−.02	.05	.05
29. I always know what I am doing.	.15	.46	−.14	−.03	.30	.01
48. I am going to be a great person.	.05	.43	.39	.04	−.03	−.05
54. I am an extraordinary person.	.06	.22	.69	−.07	−.06	.01
7. I know that I am good because everybody keeps telling me so.	−.18	.01	.69	.00	.21	.01
36. I like to be complimented.	.00	−.28	.67	.06	.00	.11
14. I think I am a special person.	.08	.16	.64	−.02	−.09	.17
51. I wish somebody would someday write my biography.	−.06	−.01	.57	.06	−.22	.09
28. I am apt to show off if I get the chance.	−.04	−.02	.04	.71	−.03	.06
3. Modesty doesn't become me.	−.01	.19	−.01	.69	−.16	−.06
52. I get upset when people don't notice how I look when I go out in public.	−.16	.04	.10	.51	.09	.25

Fit

We have variance explained by each PC: can divide out by total variance explained by all PCs.

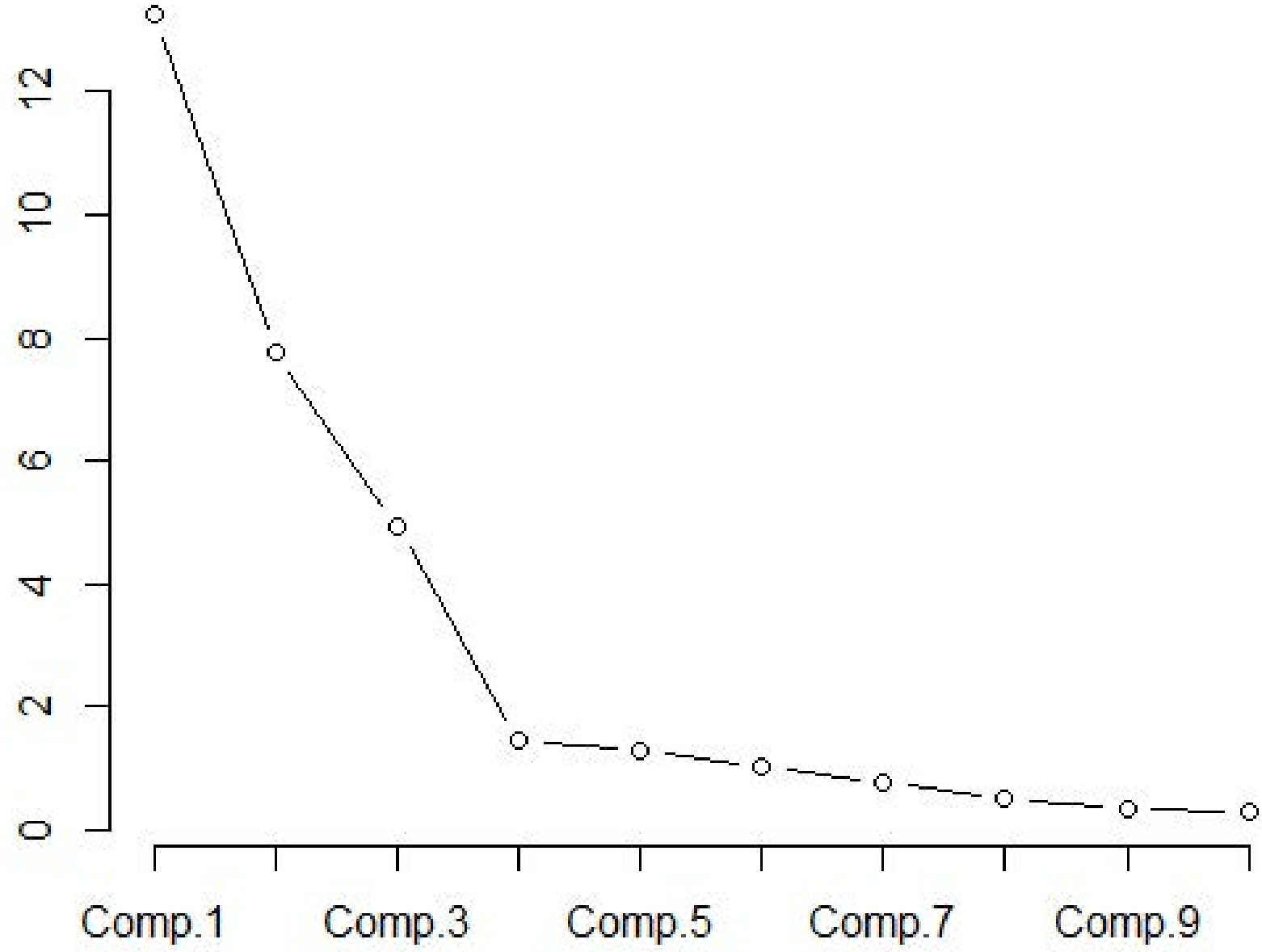
use perhaps (scree) plot, and see when 'next' PC offers 'little' extra variance explained?

or include all PCs up to e.g. 90% of variance explained?

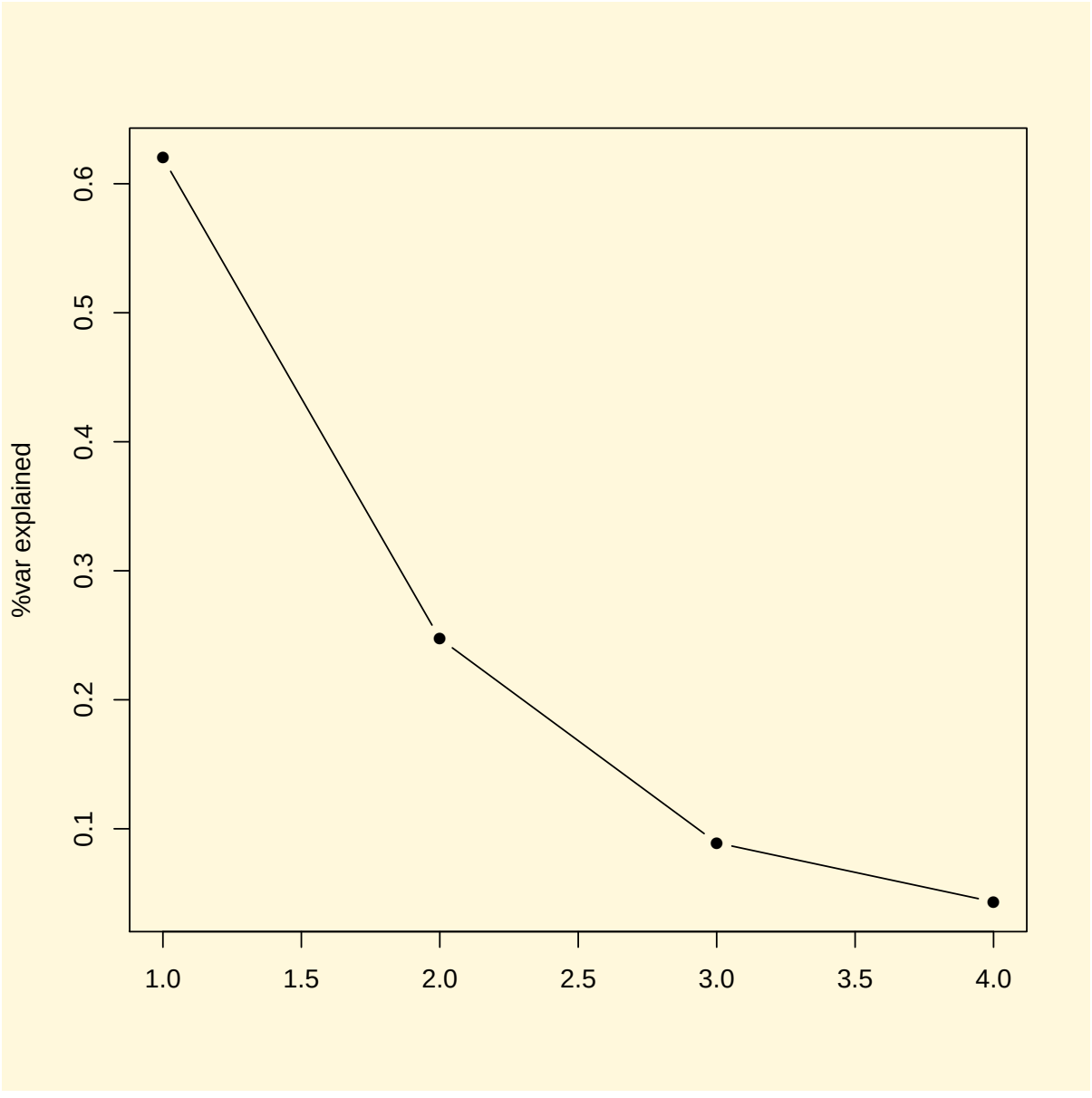
and drop last k PCs whose variances are roughly equal

btw generally like to see an 'elbow'

Good



Bad



Ugly