Published in Towards Data Science

Casey Cheng Follow
Feb 3, 2022 · 12 min read · ✦ · Listen

Save

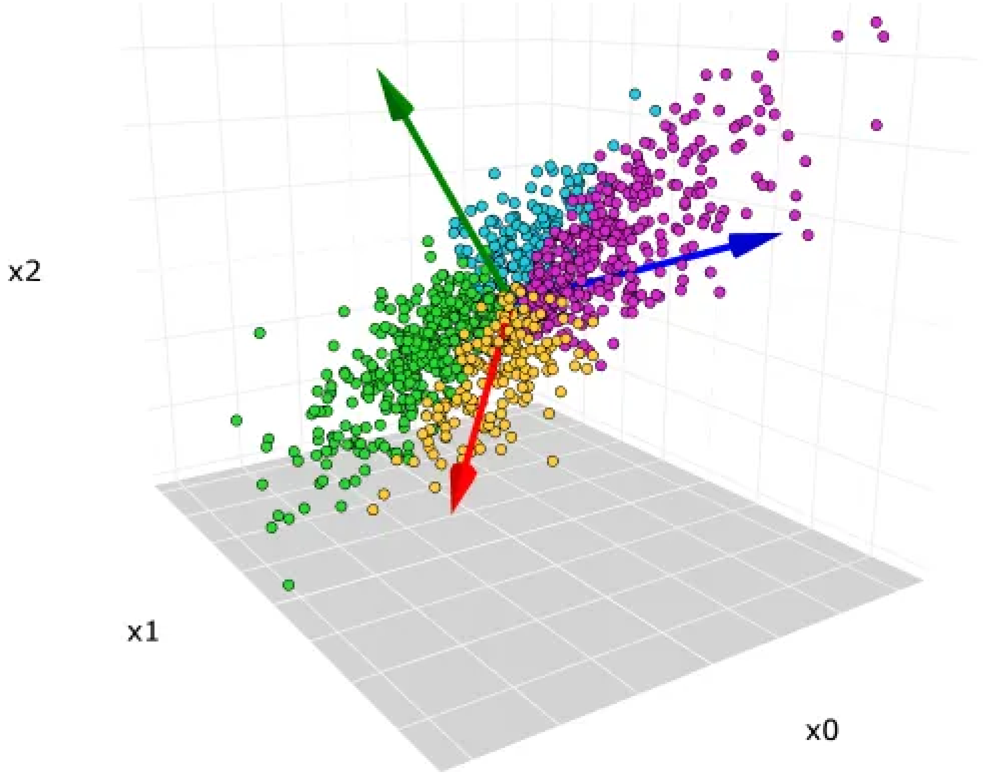
Principal Component Analysis (PCA) Explained Visually with Zero Math

Principal Component Analysis (PCA) is an indispensable tool for visualization and dimensionality reduction for data science but is often buried in complicated math. It was tough-, to say the least, to wrap my head around the whys and that made it hard to appreciate the full spectrum of its beauty.

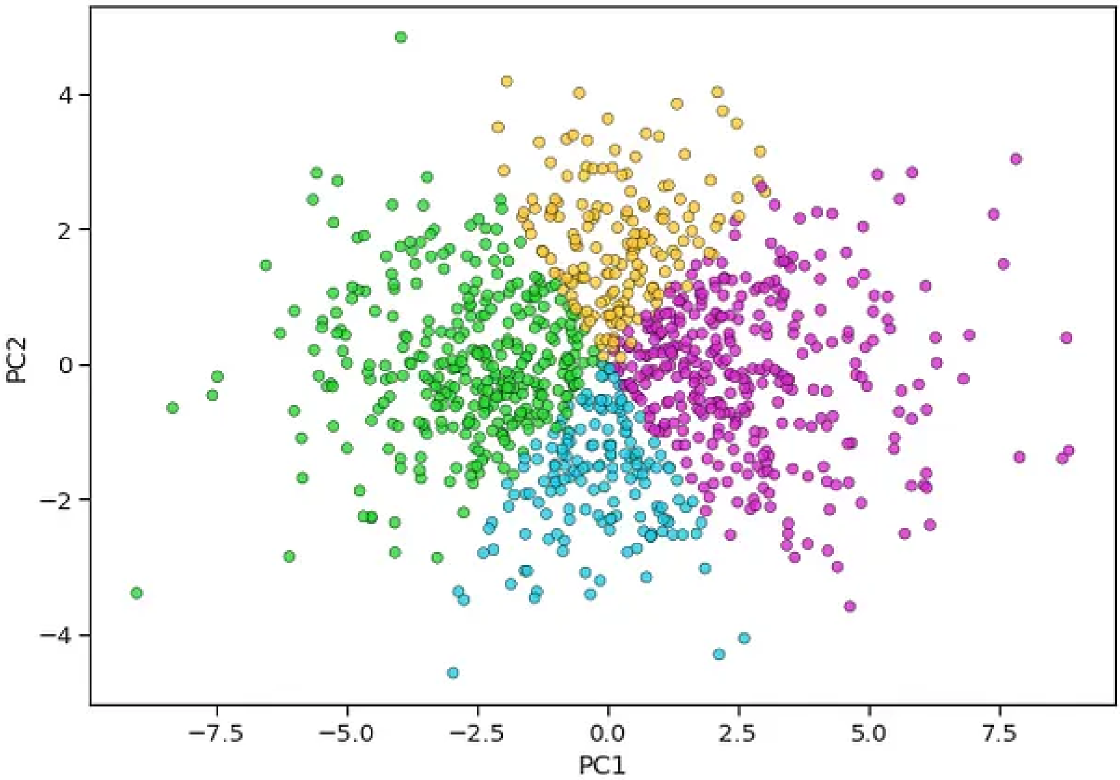
While numbers are important to prove the validity of a concept, I believe it's equally important to share the narrative behind the numbers — with a story.

What is PCA?

Principal component analysis (PCA) is a technique that transforms high-dimensions data into lower-dimensions while retaining as much information as possible.



The original 3-dimensional data set. The red, blue, green arrows are the direction of the first, second, and third principal components, respectively. Image by the author.



Scatterplot after PCA reduced from 3-dimensions to 2-dimensions. Image by the author.

PCA is extremely useful when working with data sets that have a lot of features. Common applications such as image processing, genome research always have to deal with thousands-, if not tens of thousands of columns.

While having more data is always great, sometimes they have so much information in them, we would have impossibly long model training time and the curse of dimensionality starts to become a problem. Sometimes, less is more.

I like to compare PCA with writing a book summary.

Finding the time to read a 1000-pages book is a luxury that few can afford. Wouldn't it be nice if we can summarize the most important points in just 2 or 3 pages so that the information is easily digestible even by the busiest person? We may lose some information in the process, but hey, at least we get the big picture.

How does PCA work?

It's a two-step process. We can't write a book summary if we haven't read or understood the content of the book.

PCA works the same way — understand, then summarize.

Understanding data the PCA way

Human understands the meaning of a storybook through the use of expressive language. Unfortunately, PCA doesn't speak English. It has to find meaning within our data through its preferred language, mathematics.

The million-dollar question here is…

- Can PCA understand which part of our data is important?
- Can we mathematically quantify the amount of information embedded within the data?

Well, *variance* can.

The greater the variance, the more the information. Vice versa.

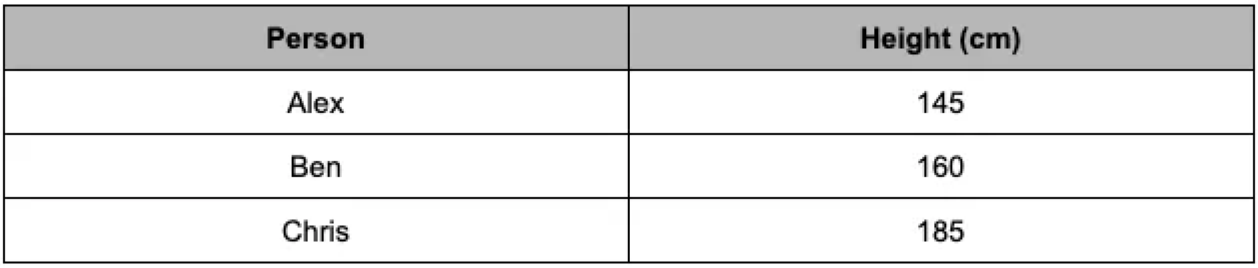
For most people out there, variance is not an unfamiliar term. We've learned in high school that variance measures the average degree to which each point differs from the mean.

$$S^2 = \frac{\sum (x_i - \bar{x})^2}{n-1}$$

The formula of variance.

But nowhere does it associate variance with information. So where does this association comes from? And why should it make sense?

Suppose that we are playing a guessing game with our friends. The game is simple. Our friends would cover their faces and we need to guess who's who based solely on their height. Being the good friends that we are, we remember how tall everyone is.



Person	Height (cm)
Alex	145
Ben	160
Chris	185

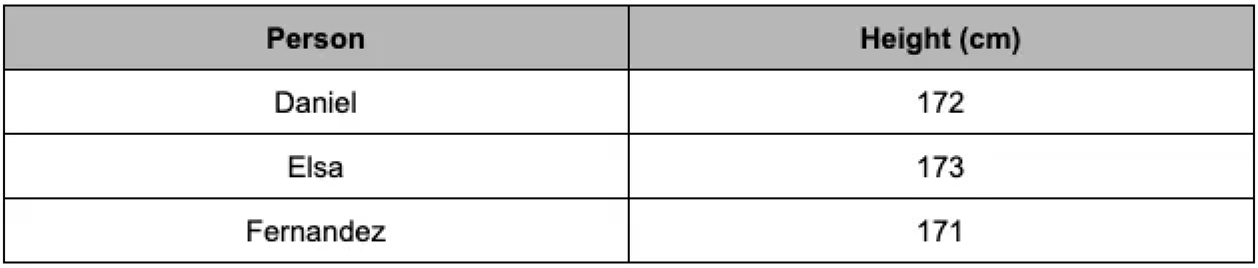
Our friends' heights from memory. Image by the author.

I'll go first.

The silhouette of three identical friends whom we need to identify based on their height differences. Image by 7089643 from Pixabay, edited with permission by the author.

Without a doubt, I am going to say that Person A is Chris, Person B is Alex, and Person C is Ben.

Now, let's try and guess a different group of friends.



Person	Height (cm)
Daniel	172
Elsa	173
Fernandez	171

Another group of our friends' heights that we remember by heart. Image by the author.

Your turn.

The silhouette of three similarly-tall friends whom we need to identify. Image by 7089643 from Pixabay, edited with permission by the author.

Can you guess who's who? It's tough when they are very similar in height.

Earlier, we had no trouble differentiating a 185cm person from a 160cm and 145cm person because their height *varies* a lot.

In the same way, when our data has a higher variance, it holds more information. This is why we keep hearing PCA and maximum variance in the same sentence. I'd like to formalize this by quoting a snippet from Wikipedia.

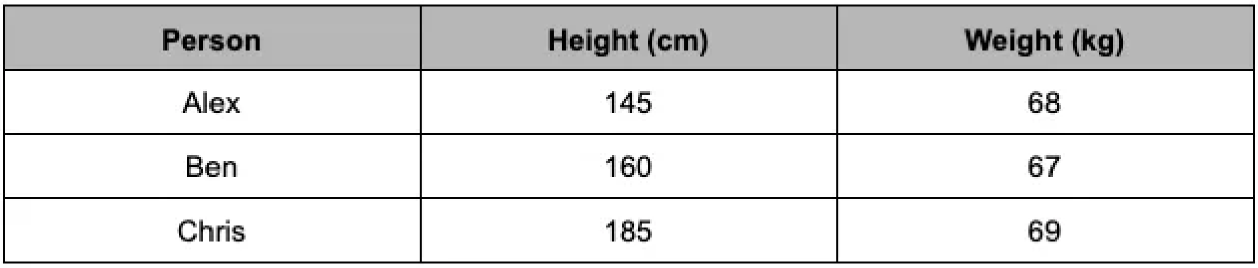
> *PCA is defined as an orthogonal linear transformation that transforms the data to a new coordinate system such that the* **greatest variance** *by some scalar projection of the data comes to lie on the first coordinate (called the first principal component), the second greatest variance on the second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

In the eyes of PCA, variance is an objective and mathematical way to quantify the amount of information in our data.

Variance *is* information.

To drive the point home, I propose a rematch for the guessing game, only that this time, we get to guess who's who based on their height and weight.

Round two.



Person	Height (cm)	Weight (kg)
Alex	145	68
Ben	160	67
Chris	185	69

The same set of friends and their respective height and weight. Image by the author.

In the beginning, we only had height. Now, we have essentially doubled the amount of data on our friends. Would that change your guessing strategy?

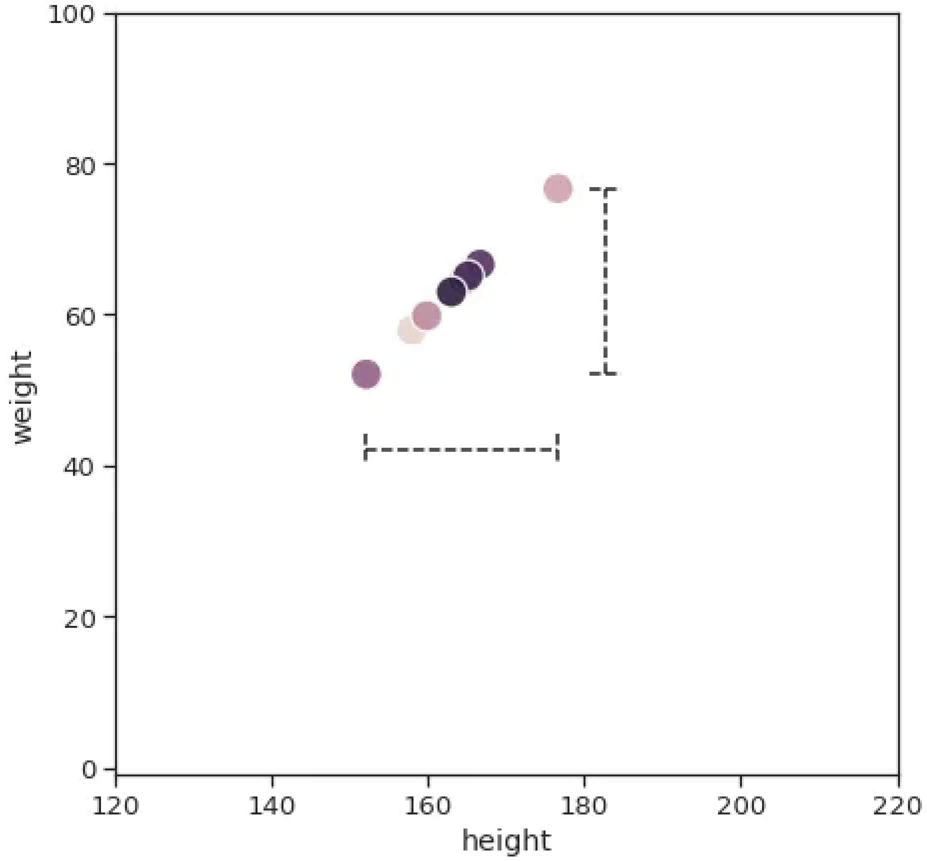
This is a nice little segue into the next section — how PCA summarizes our data, or more accurately, reduces dimensionality.

Summarizing data with PCA

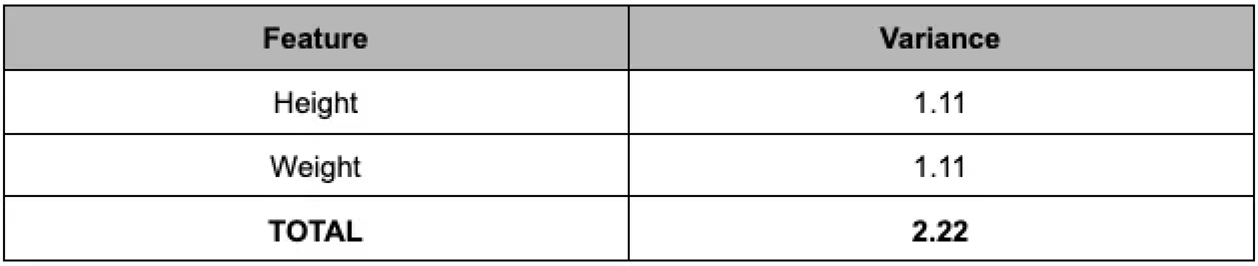
Personally, the weight differences are so small (a.k.a small variance), it doesn't help me differentiate our friends at all. I still had to rely mostly on height to make my guesses.

Intuitively, we have just reduced our data from 2-dimensions to 1-dimension. The idea is that we can selectively keep the variables with higher variances and then forget about the variables with lower variance.

But what if-, just what if height and weight have the *same* variance? Does it mean we can no longer reduce the dimensionality of this data set? I'd like to illustrate this with a sample dataset.



The dotted line represents the variance of height and weight. Image by the author.



Feature	Variance
Height	1.11
Weight	1.11
TOTAL	**2.22**

All the features are standardized to the same scale for a fair comparison. Image by the author.

In this case, it’s very difficult to choose the variables we want to delete. If I throw away either one of the variables, we are throwing away half of the information.

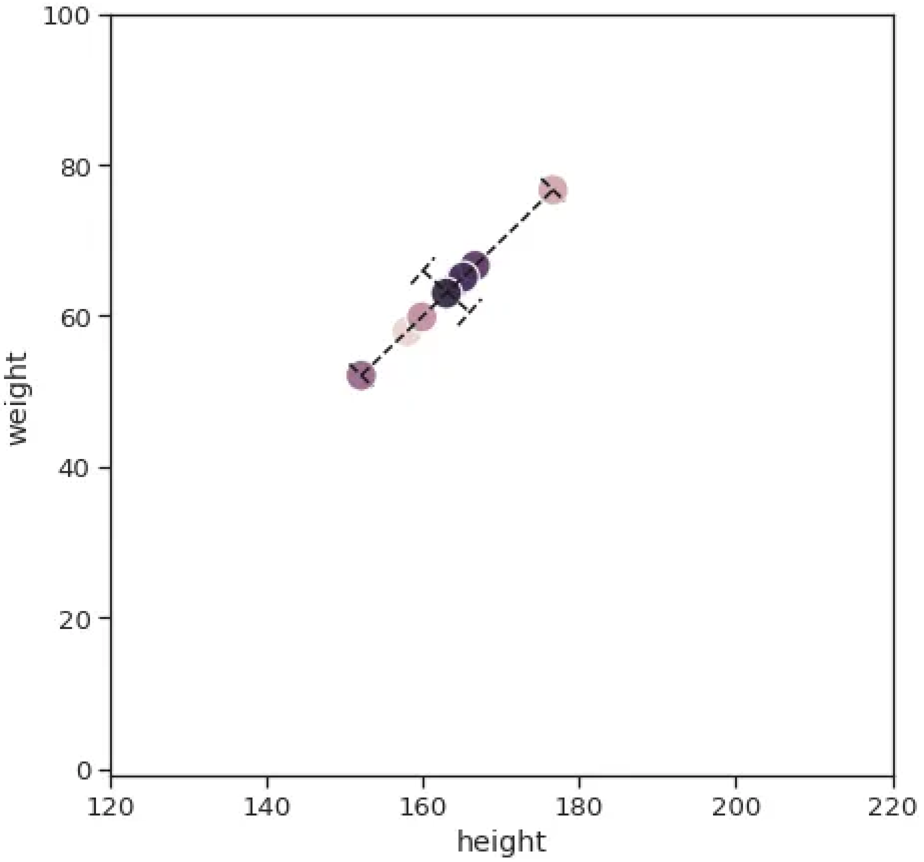
Can we keep *both*?

Perhaps, with a different perspective.

The best storybooks always have hidden themes that are not written but *implied*. Reading each chapter individually wouldn’t make sense. But if we read all of it, it gives us enough context to piece the puzzles together — the underlying plot emerges.

Up until now, we have only been looking at the variance of height and weight individually. Instead of limiting ourselves to choose just one or the other, why not combine them?

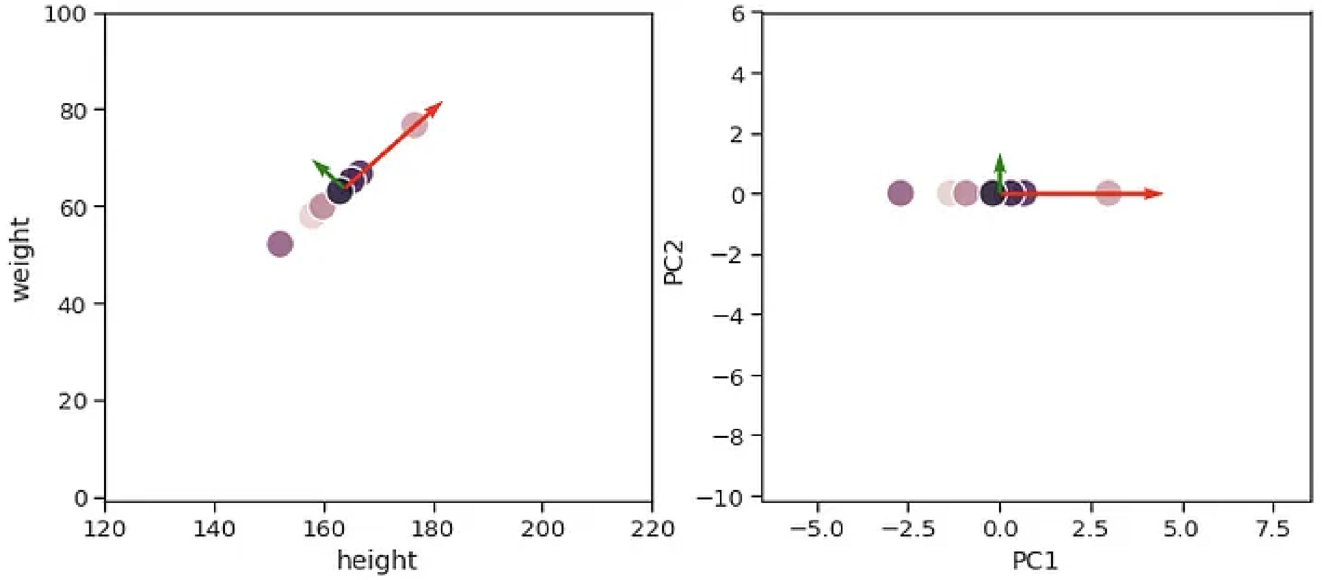
When we look closer at our data, the maximum amount of variance lies not in the x-axis, not in the y-axis, but a diagonal line across. The second-largest variance would be a line 90 degrees that cuts through the first.



The dotted line shows the direction of maximum variance. Image by the author.

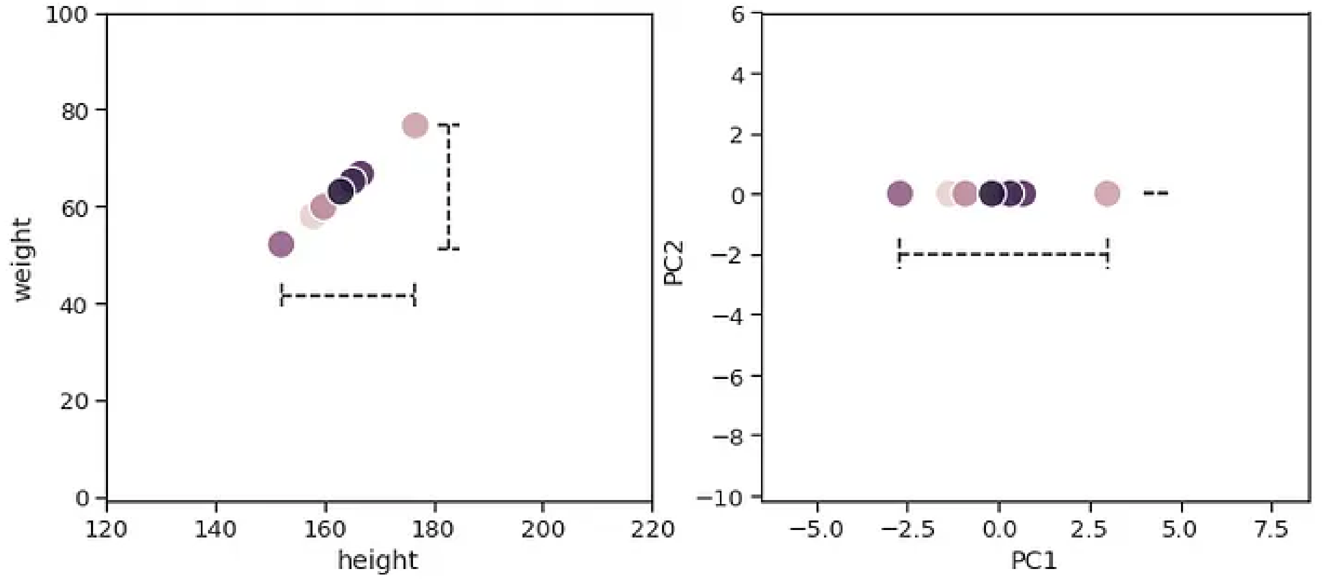
To represent these 2 lines, PCA combines both height and weight to create two brand new variables. It could be 30% height and 70% weight, or 87.2% height and 13.8% weight, or any other combinations depending on the data that we have.

These two new variables are called the **first principal component (PC1)** and the **second principal component (PC2)**. Rather than using height and weight on the two axes, we can use PC1 and PC2 respectively.

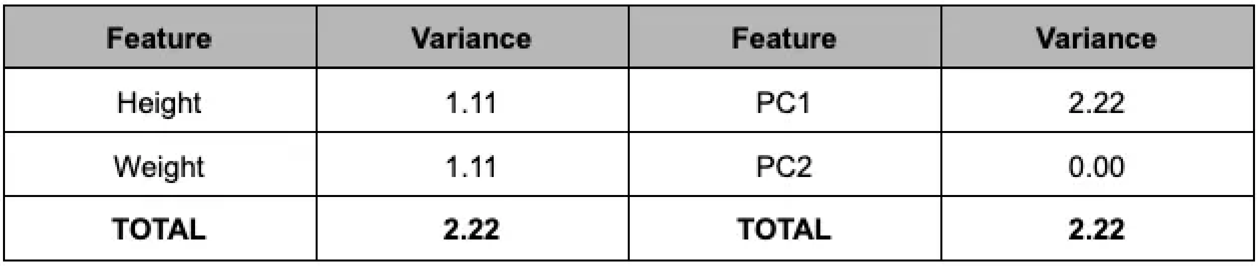


(Left) The red and green arrows are the principal axes in the original data. Image by the author. | **(Right)** The direction of the principal axes have been rotated to become the new x- and y-axis. Image by the author.

After all the shenanigans, let's take a look at the variances again.



(Left) The variance of height and weight are similar in the original data. Image by the author. | **(Right)** After PCA transformation, all of the variances are shown in the PC1 axis. Image by the author.



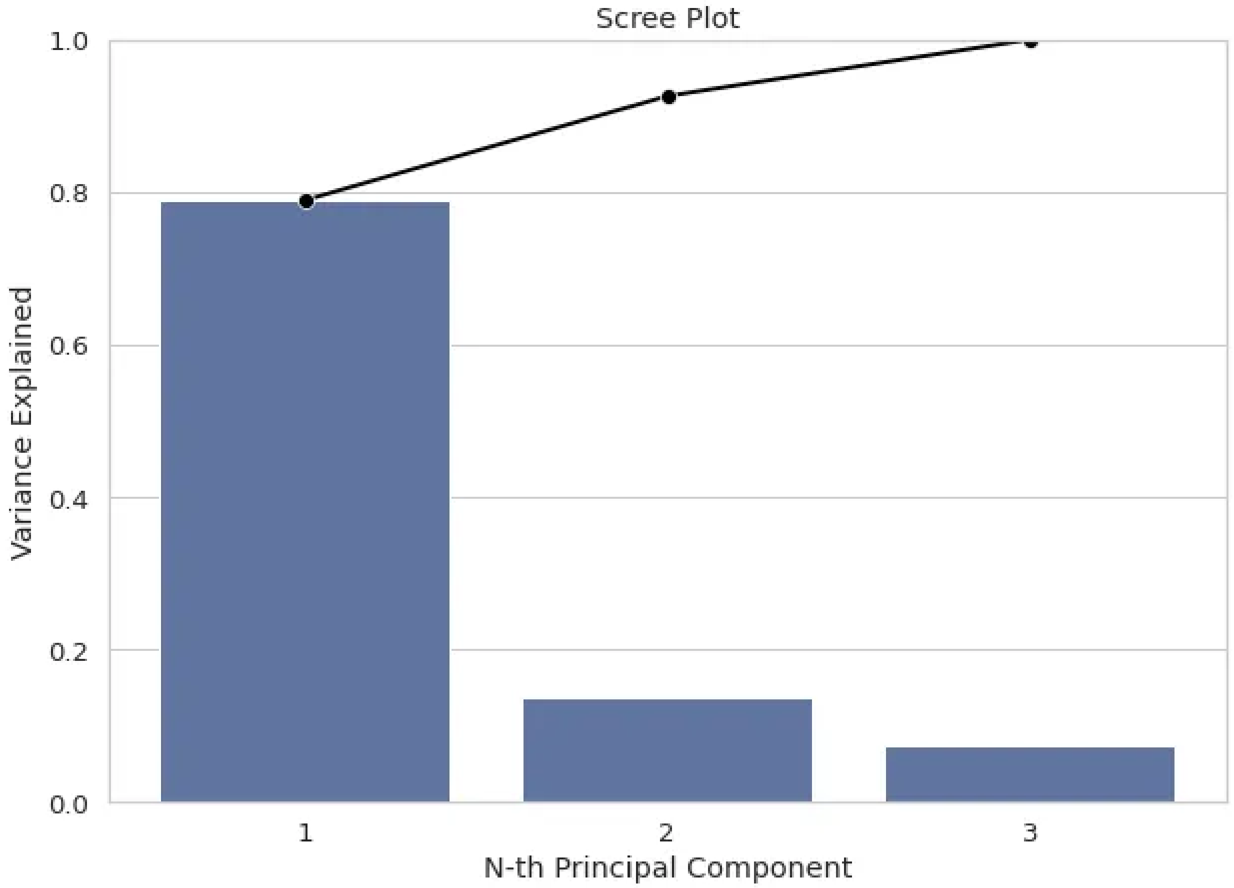
Feature	Variance	Feature	Variance
Height	1.11	PC1	2.22
Weight	1.11	PC2	0.00
TOTAL	**2.22**	**TOTAL**	**2.22**

All the variables are standardized to the same scale for a fair comparison. Image by the author.

PC1 alone can capture the total variance of Height and Weight combined. Since PC1 has all the information, you already know the drill — we can be very comfortable in removing PC2 and know that our new data is still representative of the original data.

When it comes to real data, more often than not, we won't get a principal component that captures 100% of the variances. Performing a PCA will give us N number of principal components, where N is equal to the dimensionality of our original data. From this list of principal components, we generally choose the least number of principal components that would explain the most amount of our original data.

A great visual aid that will help us make this decision is a **Scree Plot**.



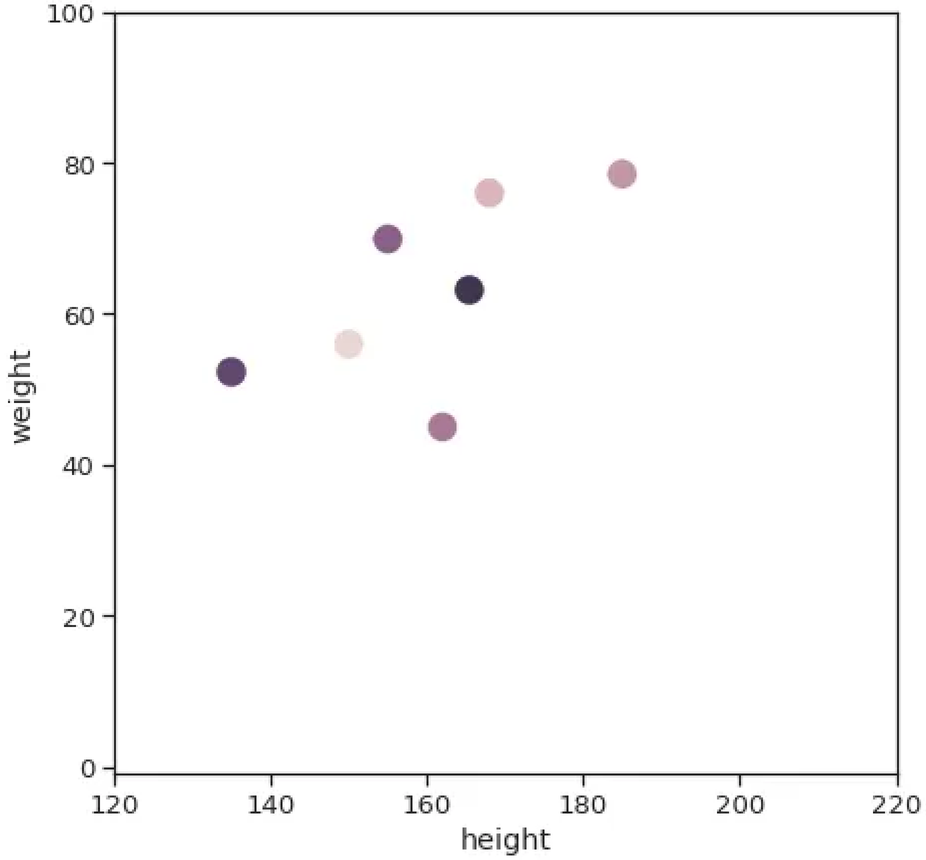
An example of a Scree Plot for a 3-dimensional data set. Image by the author.

The bar chart tells us the proportion of variance explained by each of the principal components. On the other hand, the superimposed line chart gives us the cumulative sum of explained variance up until N-th principal component. Ideally, we want to get at least 90% variance with just 2- to 3-components so that enough information is retained while we can still visualize our data on a chart.

Looking at the chart, I would feel comfortable using 2 principal components.

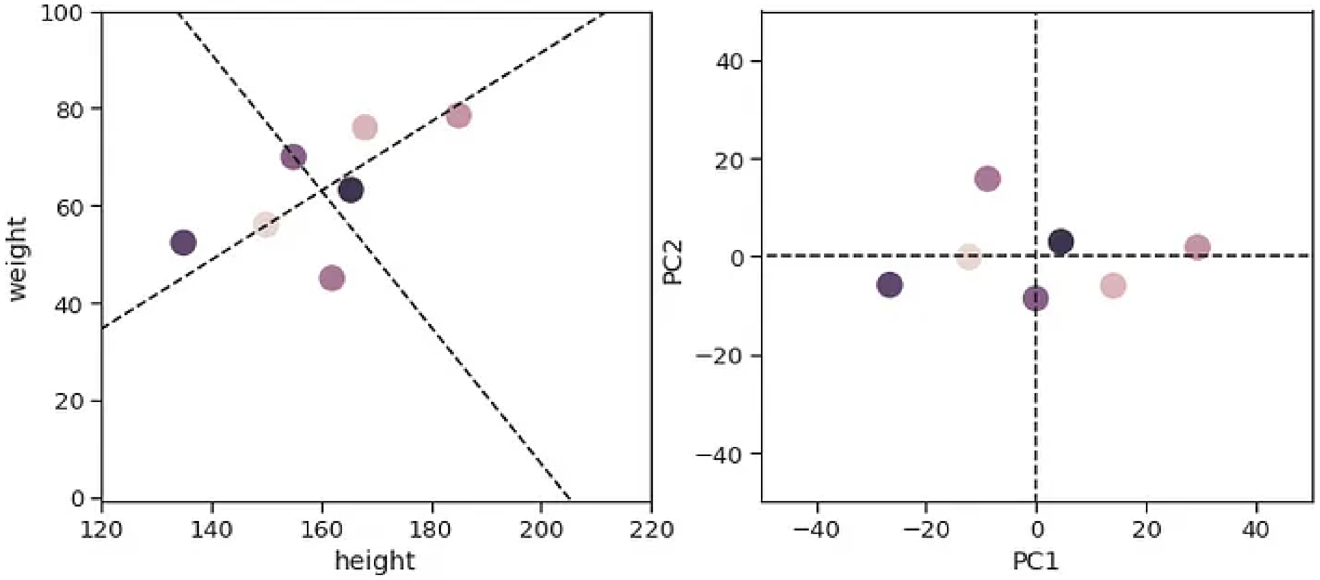
The PC That Got Away

Since we're not choosing all the principal components, we inevitably lose some information. But we haven't exactly described what we are losing. Let's dive deeper into that with a new toy example.



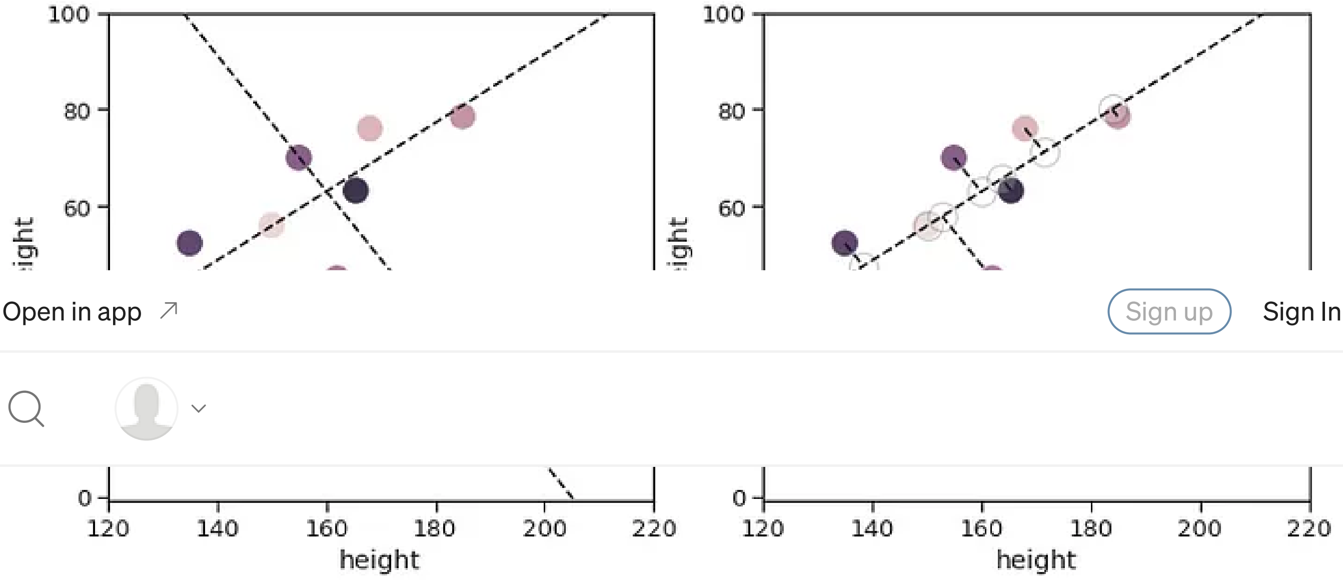
The points are scattered but we can still see some positive correlation in a diagonal line across. Image by the author.

If we feed our data through the PCA model, it would start by drawing the First Principal Component followed by the Second Principal Component. When we transform our original data from 2-dimensions to 2-dimensions, everything stays the same except the orientation. We just rotated our data so that the maximum variance is in PC1. Nothing new here.



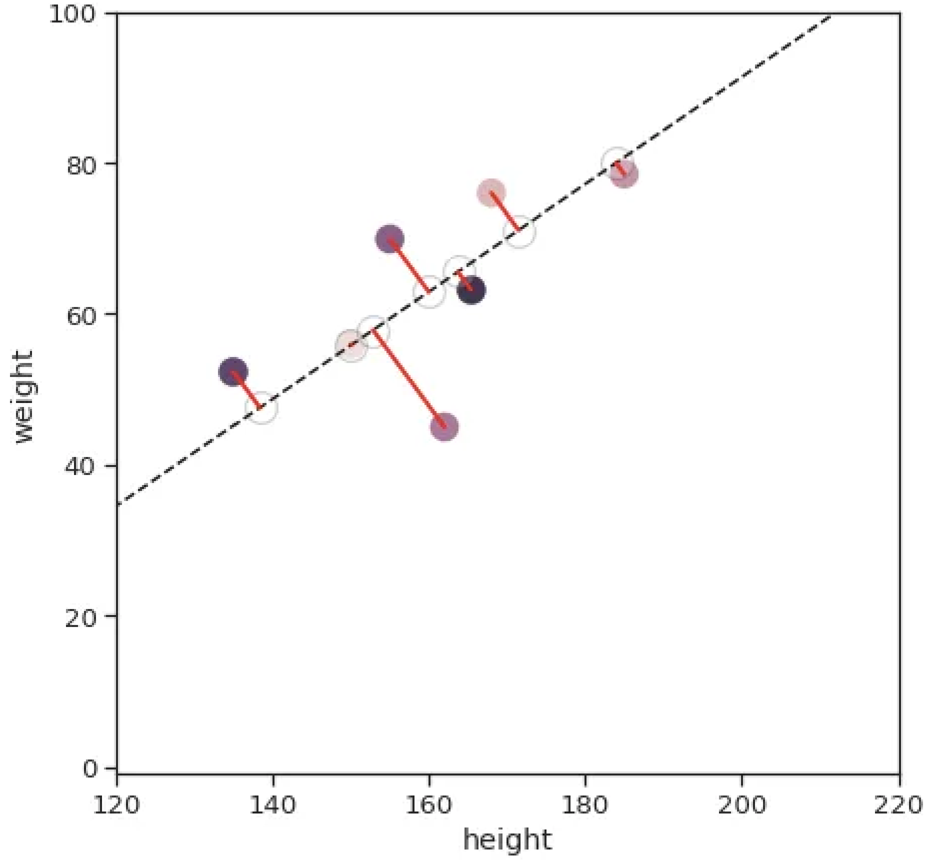
(Left) The dotted lines are the direction of the first and second principal components. Image by the author. | **(Right)** PCA rotates the data hence putting the maximum variance on PC1, followed by PC2. Image by the author.

However, suppose that we have decided to keep only the First Principal Component, we would have to project all our data points onto the First Principal Component because we no longer have the y-axis.



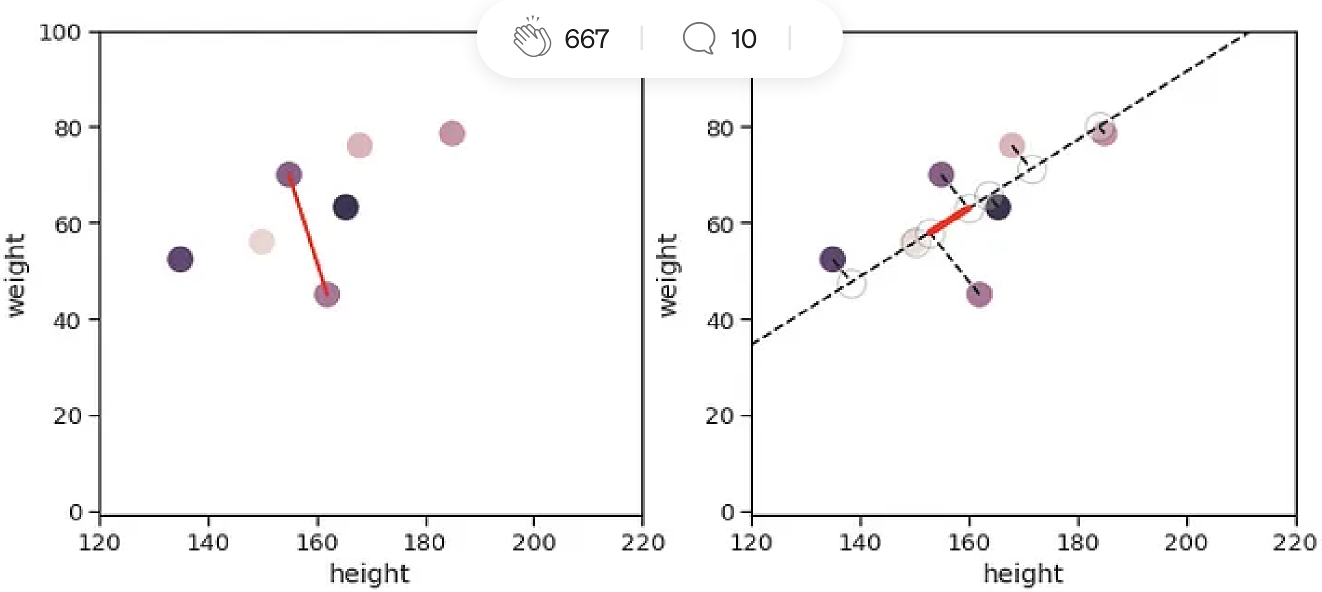
(Left) The dotted lines are the direction of the first and second principal components. Image by the author. | **(Right)** All the dots now sit on the dotted line because we removed the 2nd principal component. Image by the author.

What we would lose is the distance in the Second Principal Component, highlighted with the red color line below.



All the red lines are values in the 2nd principal component and they have been removed. Image by the author.

This has implications on the perceived distance of each data point. If we look at the Euclidean distance between two specific points (a.k.a pairwise distance), you will notice that some points are much farther in the original data than in the transformed data.

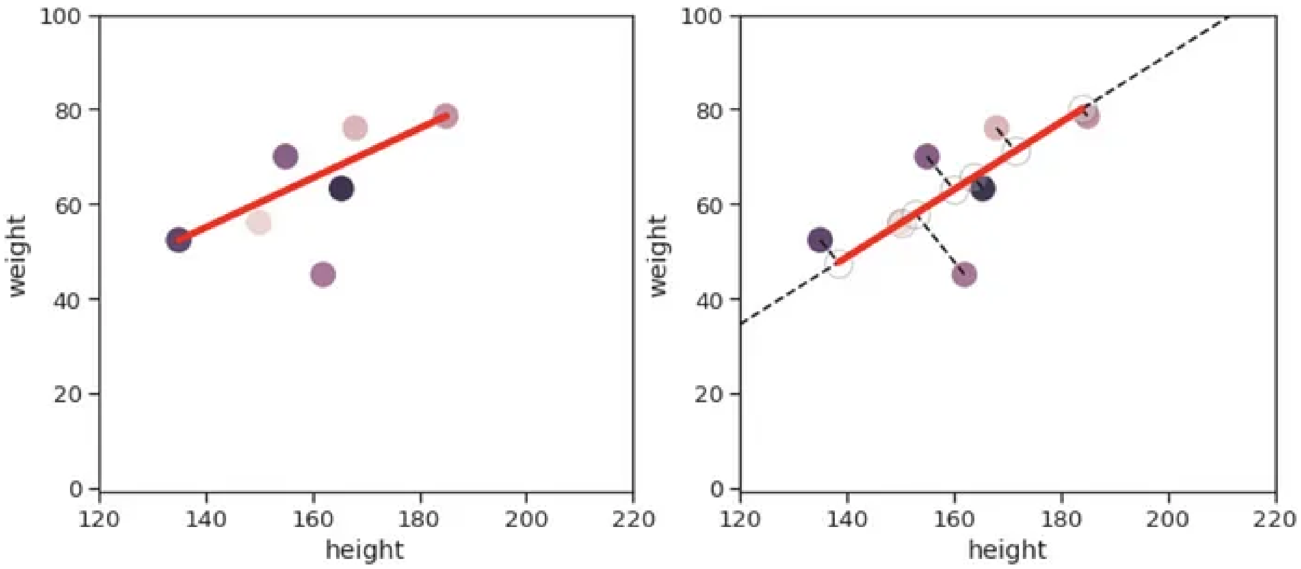


The comparison of pairwise Euclidean distance between two points before **(Left)** and after **(Right)** dimensionality are reduced from 2- to 1-dimension. Image by the author.

The PCA is a linear transformation so in and of itself does not alter distances, but when we start removing dimensions, the distances get distorted.

It gets trickier— not all pairwise distance gets affected equally.

If we take the two furthest points, you will see that they are almost parallel to the principal axes. Although their Euclidean distance is still distorted, it is to a much lesser degree.



The pairwise Euclidean distance between these two points before **(Left)** and after **(Right)** dimensionality reduction remains fairly identical. Image by the author.

The reason is that principal component axes are drawn in the direction where we have the largest variance. By definition, variance increases when the data points are further apart. So naturally, the points furthest apart would align themselves better with the principal axes.

To sum it all up, reducing dimensions with PCA changes the distances of our data. It does so in a way that preserves large pairwise distance better than small pairwise distance.

This is one of the few drawbacks of reducing dimensions with PCA and we need to be aware of that, especially when working with Euclidean distance-based algorithm.

Sometimes, it may be more beneficial to run your algorithm on the original data instead. That’s where you, a Data Scientist need to make a decision based on your data and your use case.

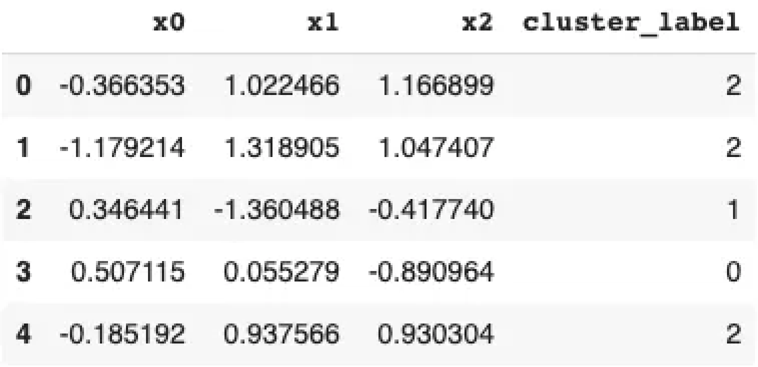
After all, data science is as much art as it is science.

Implementing PCA in Python

There is much more to PCA beyond the premise of this article. The only way to truly appreciate the beauty of PCA is to experience it yourself. Hence, I would love to share some code snippets here for anyone that wants to get their hands dirty. The full code can be accessed here with Google Colab.

First things first, let's get the imports out of the way and generate some data that we will be working with.

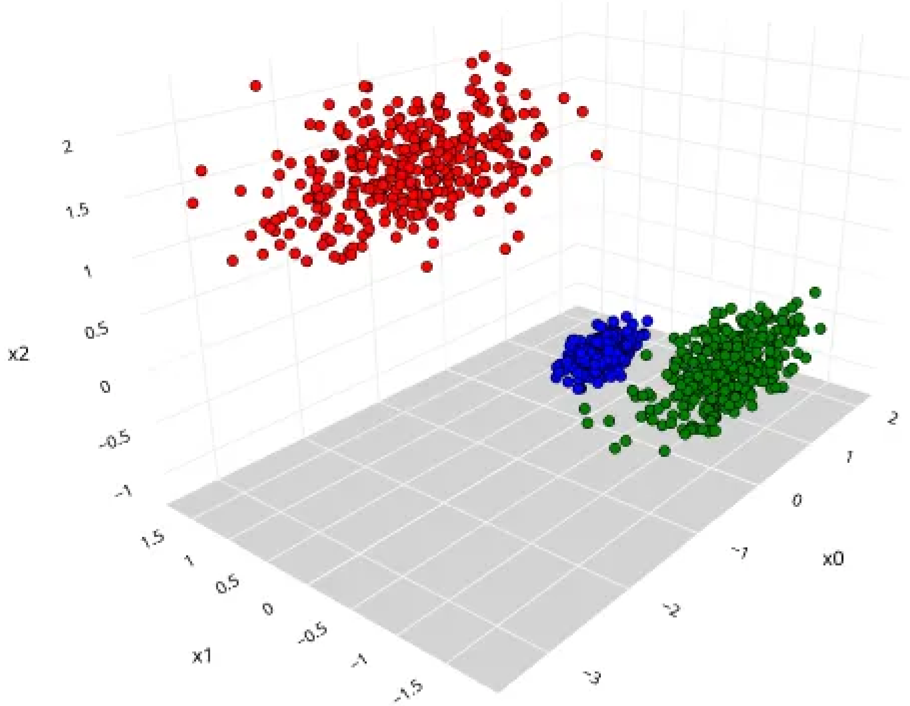
Our toy data set has 3 variables — x0, x1, and x2 and they are distributed in a way that clumps together in 3 different clusters. The “cluster_label” tells us which cluster the data point belongs to.



	x0	x1	x2	cluster_label
0	-0.366353	1.022466	1.166899	2
1	-1.179214	1.318905	1.047407	2
2	0.346441	-1.360488	-0.417740	1
3	0.507115	0.055279	-0.890964	0
4	-0.185192	0.937566	0.930304	2

The first 5 rows of our toy example data set. Image by the author.

It’s always a good idea to visualize them whenever possible.



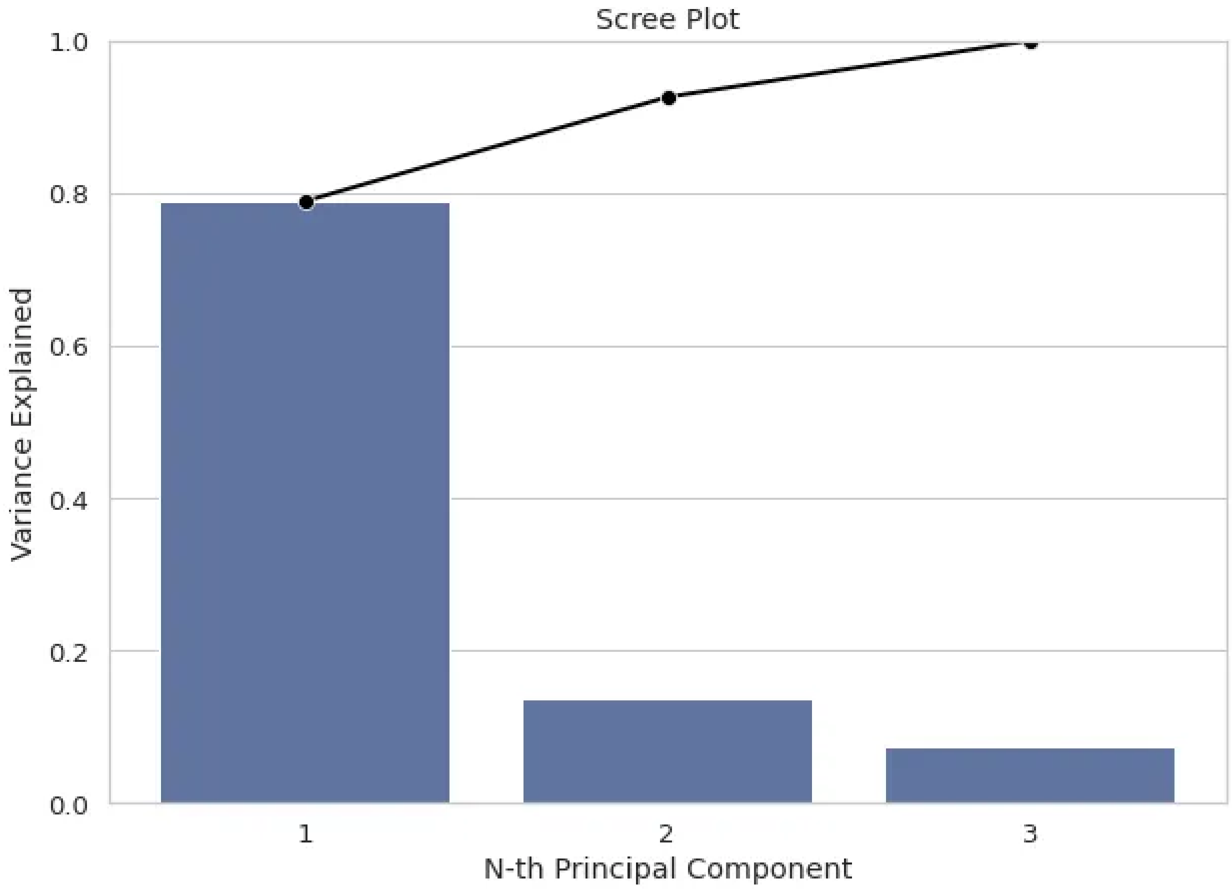
Our toy data on a 3-D chart. Image by the author.

The data seems ready for PCA. We're going to try and reduce its dimensionality. Fortunately, Sklearn made PCA very easy to execute. Even though it took us over 2000 words to explain PCA, we only needed 3 lines to run it.

There are a couple of moving parts here. When we fit our data to Sklearn's PCA function, it does all of the heavy liftings to return us a PCA model and the transformed data.

The model gives us access to multitudes of attributes such as eigenvalues, eigenvectors, mean of original data, variance explained, and the list goes on. These are incredibly insightful if we want to understand what the PCA has done with our data.

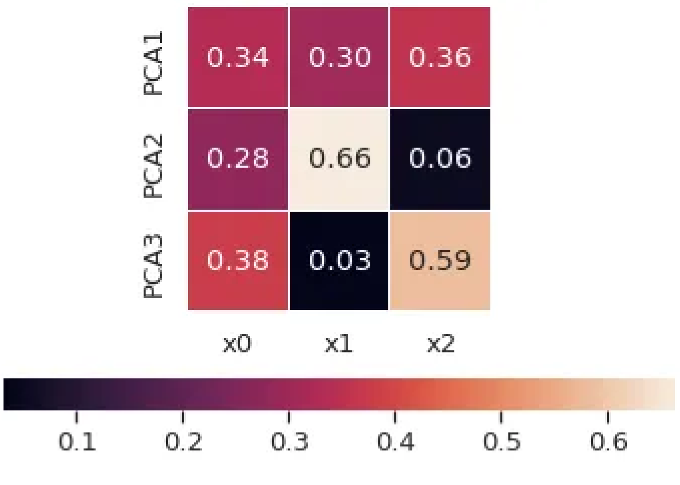
One attribute I'd like to highlight is the `pca.explained_variance_ratio_` which tells us the proportion of variance explained by each principal component. We could visualize this with a Scree Plot.



Line plot superimposed on a bar plot to show the proportion of variance for each PC. Image by the author.

The chart informs us that using 2 principal components instead of 3 is fine because they can capture 90%+ of the variance.

On top of that, we can also look at the combinations of variables that created each principal component with `pca.components_**2`. We could use a heat map to showcase this.



Each PC is made from combinations of multiple variables. Image by the author.

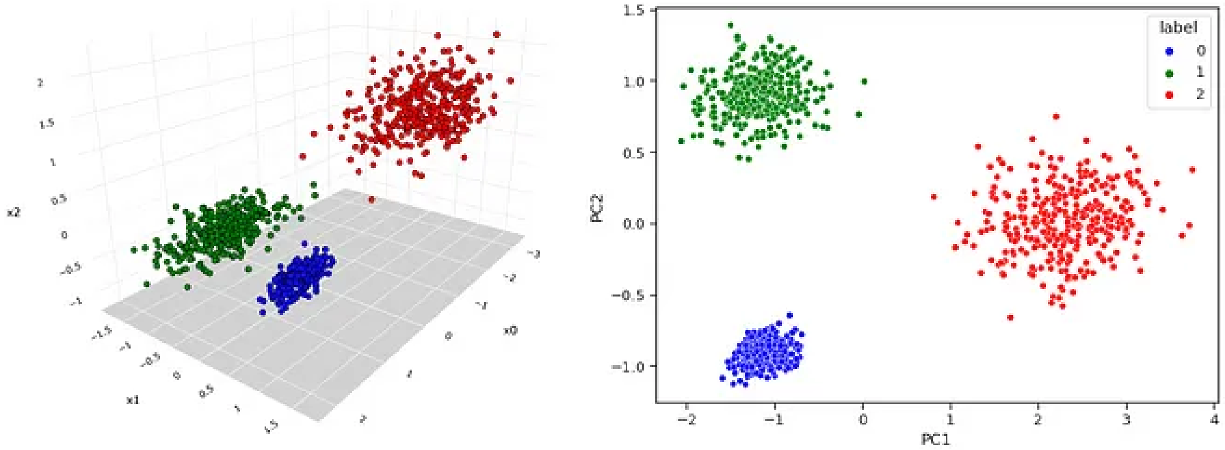
In our example, we can see that PCA1 is made from 34% of x0, 30% of x1, and 36% of x2. PCA2 is primarily dominated by x1.

There are a lot more useful attributes that are made available by Sklearn. For those who are interested, I recommend having a look at the attributes section of PCA on Sklearn documentation.

Now that we understand the principal components a little bit better, we can make a final decision on the number of principal components we want to keep. In this case, I feel that 2 principal components are sufficient.

So, we can re-run the PCA model, but this time with the `n_components=2` argument, which tells the PCA to keep only the top 2 principal components for us.

This will return us a DataFrame with the first two principal components. Finally, we can plot a scatterplot to visualize our data.



(Left) The original data. Image by the author. | **(Right)** The same data but reduced to 2-D with PCA. Image by the author.

Final Remarks

PCA is a mathematically beautiful concept and I hope I was able to convey it in a casual tone so it doesn't feel overwhelming. For those that are eager to look into the nitty-gritty details, I have attached some interesting discussions/resources below for your perusal.

Thank you for your time, and have a great day.

[1]: Medium, Farhad Malik (Jan 7, 2019). *What are Eigenvalues and Eigenvectors?* https://medium.com/fintechexplained/what-are-eigenvalues-and-eigenvectors-a-must-know-concept-for-machine-learning-80d0fd330e47

[2]: GitHub. *In-Depth: Principal Component Analysis* https://jakevdp.github.io/PythonDataScienceHandbook/05.09-principal-component-analysis.html

[3]: StackExchange, whuber (Feb 21, 2013). *Should one remove highly correlated variables before doing PCA?* https://stats.stackexchange.com/questions/50537/should-one-remove-highly-correlated-variables-before-doing-pca

[4]: StackExchange, ttnphns (Apr 13, 2017). *PCA and proportion of variance explained* https://stats.stackexchange.com/questions/22569/pca-and-proportion-of-variance-explained

[5]: StackExchange, amoeba (Apr 13, 2017). *What is meant by PCA preserving only large pairwise distances?* https://stats.stackexchange.com/questions/176672/what-is-meant-by-pca-preserving-only-large-pairwise-distances

[6]: StackExchange, amoeba (6 Mar, 2015). *Making sense of principal component analysis, eigenvectors & eigenvalues* https://stats.stackexchange.com/questions/2691/making-sense-of-principal-component-analysis-eigenvectors-eigenvalues/140579#140579

Principal Component Data Science Data Visualization Python Deep Dives