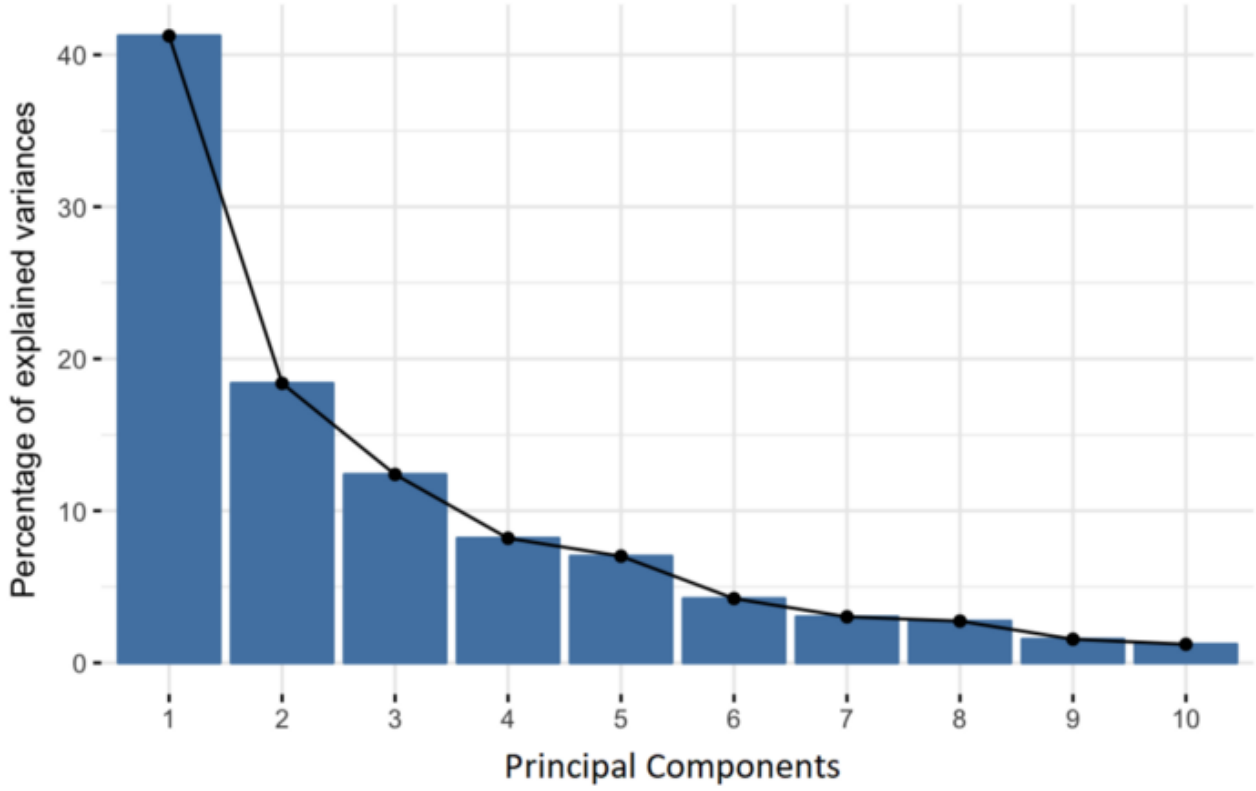
Principal Component Analysis (PCA) is a dimensionality reduction method used to simplify a large data set into a smaller set while still maintaining significant patterns and trends.

What are principal components?

Principal components are new variables that are constructed as linear combinations of the initial variables.

These combinations are done in such a way that the new variables are uncorrelated and most of the information is squeezed into the first components.

So, the idea is 10-dimensional data gives you 10 principal components, but PCA tries to put maximum possible information in the first component, then maximum remaining information in the second and so on, until having something like shown in the plot below.



Percentage of Variance (Information) for each by PC.

It is important to note that principal components are less interpretable and don't have any meaning.

Principal components represent the directions of the data that explain a maximal amount of variance, i.e. the lines that capture the most information.

(Q: what is variance? what does a large variance mean?)

How are principal components constructed?

The number of principal components is equal to the number of features in the data.

Principal components are constructed in such a manner that the first principal component accounts for the largest possible variance in the data set.

The second principal component is calculated in the same way, with the condition that it is uncorrelated with (i.e. perpendicular to) the first principal component and that it accounts for the next highest variance.

Step-by-step explanation of PCA

Step 1. Standardization

The aim of this step is to standardize the range of the continuous initial variables so that each of them contributes equally to the analysis.

We do so because PCA is sensitive to the variances of the initial variables. If there are large differences between the ranges of initial variables, those variables with larger ranges will dominate over those with small ranges.

For example, a variable that ranges between 0 and 100 will dominate over a variable that ranges between 0 and 1, which will lead to biased results.

Mathematically, this can be done by subtracting the mean and dividing by the standard deviation for each value of each variable.

$$z = \frac{value - mean}{standard\ deviation}$$

Once the standardization is done, all the variables will be transformed to the same scale.

Step 2. Covariance Matrix computation

The aim of this step is to see if there is any relationship between the initial variables.
This is because some variables are highly correlated in such a way that they contain redundant information. We compute the covariance matrix to identify these correlations.

The covariance matrix is an *n x n* matrix (where n is the number of dimensions).
It has as entries the covariances associated with all possible pairs of the initial variables.

For example, a 3-dimensional data set with 3 variables x, y and z will have a 3x3 covariance matrix of the form:

$$\begin{bmatrix} Cov(x,x) & Cov(x,y) & Cov(x,z) \\ Cov(y,x) & Cov(y,y) & Cov(y,z) \\ Cov(z,x) & Cov(z,y) & Cov(z,z) \end{bmatrix}$$

Since the covariance of a variable with itself is its variance (Cov(a,a)=Var(a)), in the main diagonal (Top left to bottom right) we actually have the variances of each initial variable. And since the covariance is commutative (Cov(a,b)=Cov(b,a)), the entries of the covariance matrix are symmetric with respect to the main diagonal, which means that the upper and the lower triangular portions are equal.

(Q: prove mathematically that the covariance of a variable with itself is the same as its variance)

Step 3. Compute the eigenvectors and eigenvalues of the covariance matrix to identify the principal components

We need to compute the eigenvectors and eigenvalues from the covariance matrix in order to determine the principal components of the data.

Some things to note about eigenvectors and eigenvalues:

- They always come in pairs i.e. every eigenvector has an eigenvalue
- The number of eigenvectors is equal to the number of dimensions of the data i.e. a 3-dimensional dataset will have 3 eigenvectors and 3 eigenvalues

The eigenvectors of the covariance matrix are the directions of the axes where there is the most variance and are the principal components.
The larger the variance of a line, the more information it has.

Eigenvalues are simply the coefficients attached to eigenvectors. They give the amount of variance carried in each principal component.

By ranking your eigenvectors in order of their eigenvalues, highest to lowest, you get the principal components in order of significance.

Example
Let’s suppose that our data set is 2-dimensional with 2 variables ***x,y*** and that the eigenvectors

and eigenvalues of the covariance matrix are as follows:

$$v1 = \begin{bmatrix} 0.6778736 \\ 0.7351785 \end{bmatrix} \qquad \lambda_1 = 1.284028$$

$$v2 = \begin{bmatrix} -0.7351785 \\ 0.6778736 \end{bmatrix} \qquad \lambda_2 = 0.04908323$$

If we rank the eigenvalues in descending order, we get λ1>λ2, which means that the eigenvector that corresponds to the first principal component (PC1) is *v1* and the one that corresponds to the second principal component (PC2) is *v2.*

After having the principal components, to compute the percentage of variance (information) accounted for by each component, we divide the eigenvalue of each component by the sum of eigenvalues. If we apply this on the example above, we find that PC1 and PC2 carry respectively 96 percent and 4 percent of the variance of the data.

Step 4. Create a Feature vector

In this step, we decide which eigenvectors we want to keep.

We then form a matrix from the chosen vectors which we call a feature vector.

The feature vector is simply a matrix that has as columns the eigenvectors of the components we decide to keep.

If we decide to keep only **p** eigenvectors (principal components) out of n, the final data set will have only ***p*** dimensions.

Example
Continuing from the previous step, we can either form a feature vector with both of the eigenvectors v1 and v2:

$$\begin{bmatrix} 0.6778736 & -0.7351785 \\ 0.7351785 & 0.6778736 \end{bmatrix}$$

Or discard the eigenvector v2, which of lesser significance and form a feature vector with v1 only:

$$\begin{bmatrix} 0.6778736 \\ 0.7351785 \end{bmatrix}$$

Discarding the eigenvector *v2* will reduce dimensionality by 1, and will consequently cause a loss of information in the final data set. But given that v2 was carrying only 4 percent of the information, the loss will be therefore not important and we will still have 96 percent of the information that is carried by v1.

Step 5. Recast the data along the principal component axes

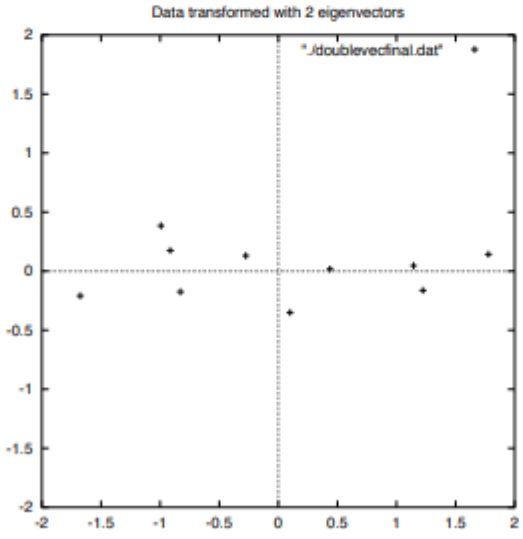
This is the last step, we aim to use the feature vector created above to reorient the data from its original axes to the ones represented by the principal components.

This can be done by multiplying the transpose of the original data set by the transpose of the feature vector.

$$FinalDataSet = FeatureVector^T * StandardizedOriginalDataSet^T$$

Extra: getting our data back

	x	y
	-.827970186	-.175115307
	1.77758033	.142857227
	-.992197494	.384374989
	-.274210416	.130417207
Transformed Data=	-1.67580142	-.209498461
	-.912949103	.175282444
	.0991094375	-.349824698
	1.14457216	.0464172582
	.438046137	.0177646297
	1.22382056	-.162675287



Data transformed with 2 eigenvectors

Transformed Data (Single eigenvector)

x
-.827970186
1.77758033
-.992197494
-.274210416
-1.67580142
-.912949103
.0991094375
1.14457216
.438046137
1.22382056

Data transformed with only the most significant eigenvector

We can only get the original data back if we took all the eigenvectors in our transformation. If the number of eigenvectors were reduced in the final transformation, then the retrieved data has lost some information.

Recall that the final transform is this:

$$FinalData = RowFeatureVector \times RowDataAdjust,$$

Which can be turned around so that, to get the original back,

$$RowDataAdjust = RowFeatureVector^{-1} \times FinalData$$

Where $RowFeatureVector^{-1}$ is the inverse of $RowFeatureVector$. However, when we take all the eigenvectors in our feature vector, it turns out that the inverse of our feature vector is actually equal to the transpose of our feature vector. This is only true because the elements of the matrix are all the unit eigenvectors of our data set. The equation becomes

$$RowDataAdjust = RowFeatureVector^{T} \times FinalData$$

But to get the original data back, we have to multiply by the standard deviation and add the means we subtracted earlier, thus our formula becomes:
RowDataAdjust = ((RowFeatureVector^T x FinalData) * OriginalStandardDeviation) + Origina

Why should we reduce the number of dimensions?

The number of dimensions of a data set refer to the number of features (typically columns) that data set has.

Reducing the dimensionality of a dataset can be beneficial for several reasons, depending on the context. Here are some key motivations:

1. Improved performance and efficiency

- Algorithms often perform better and faster on datasets with fewer dimensions because the computational cost typically grows with the number of dimensions.
  Training a machine learning model on 10 features is generally faster than training it on 1,000 features.
- Large datasets with many features require significant memory and storage. Dimensionality reduction can compress data while retaining its essential characteristics.

2. Improved visualization
   Humans can naturally understand 2D and 3D visualizations. Dimensionality reduction techniques like PCA enable us to project high-dimensional data into two or three dimensions, making it easier to detect patterns, clusters, or trends.

   Customer purchase data with 50 features (e.g., spending habits across different categories) can be visualized in 2D after dimensionality reduction to identify distinct customer segments.
3. Noise Reduction
   High-dimensional datasets often contain features that are noisy or carry little information. These features can obscure patterns or relationships in the data.
   Techniques like PCA prioritize components with the most variance, effectively filtering out features dominated by noise.
4. Avoiding Overfitting
   In machine learning, having too many features relative to the number of observations can lead to overfitting, where the model learns patterns specific to the training data but fails to generalize to unseen data.

   By reducing the number of dimensions, we reduce the risk of overfitting and improve the model's ability to generalize.
5. Data Compression
   Dimensionality reduction techniques can summarize a large dataset with fewer features while retaining most of the information, making it ideal for scenarios with limited storage or bandwidth.

   Images with thousands of pixels can be compressed into a few principal components for efficient storage or transmission.