HOW DO YOU DO A PCA?

1. Standardize the range of continuous initial variables
2. Compute the covariance matrix to identify correlations
3. Compute the eigenvectors and eigenvalues of the covariance matrix to identify the principal components
4. Create a feature vector to decide which principal components to keep
5. Recast the data along the principal components axes

What Is Principal Component Analysis?

Principal Component Analysis, or PCA, is a dimensionality-reduction method that is often used to reduce the dimensionality of large data sets, by transforming a large set of variables into a smaller one that still contains most of the information in the large set.

Reducing the number of variables of a data set naturally comes at the expense of accuracy, but the trick in dimensionality reduction is to trade a little accuracy for simplicity. Because smaller data sets are easier to explore and visualize and make analyzing data much easier and faster for machine learning algorithms without extraneous variables to process.

So to sum up, the idea of PCA is simple — reduce the number of variables of a data set, while preserving as much information as possible.

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 one of them contributes equally to the analysis.

More specifically, the reason why it is critical to perform standardization prior to PCA, is that the latter is quite sensitive regarding the variances of the initial variables. That is, 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. So, transforming the data to comparable scales can prevent this problem.

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 understand how the variables of the input data set are varying from the mean with respect to each other, or in other words, to see if there is any relationship between them. Because sometimes, variables are highly correlated in such a way that they contain redundant information. So, in order to identify these correlations, we compute the covariance matrix.

The covariance matrix is a *p* × *p* symmetric matrix (where *p* is the number of dimensions) that has as entries the covariances associated with all possible pairs of the initial variables. For example, for a 3-dimensional data set with 3 variables *x*, *y*, and *z*, the covariance matrix is a 3×3 matrix of this from:

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

Covariance Matrix for 3-Dimensional Data

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.

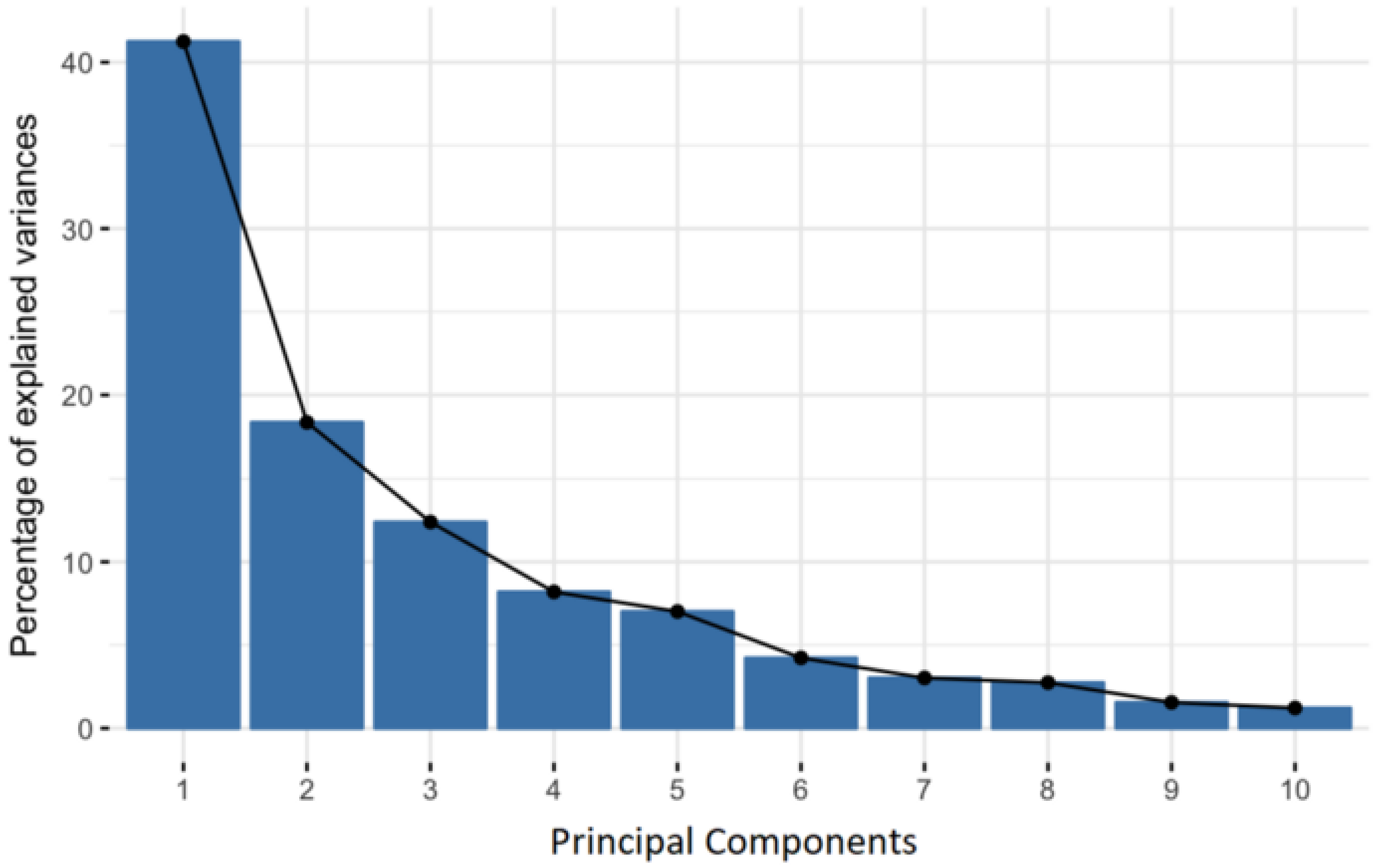
What do the covariances that we have as entries of the matrix tell us about the correlations between the variables?
It's actually the sign of the covariance that matters :

- if positive then : the two variables increase or decrease together (correlated)
- if negative then : One increases when the other decreases (Inversely correlated)

Now, that we know that the covariance matrix is not more than a table that summaries the correlations between all the possible pairs of variables, let's move to the next step.

STEP 3: COMPUTE THE EIGENVECTORS AND EIGENVALUES OF THE COVARIANCE MATRIX TO IDENTIFY THE PRINCIPAL COMPONENTS

Eigenvectors and eigenvalues are the linear algebra concepts that we need to compute from the covariance matrix in order to determine the ***principal components*** of the data. Before getting to the explanation of these concepts, let's first understand what do we mean by principal components.
Principal components are new variables that are constructed as linear combinations or mixtures of the initial variables. These combinations are done in such a way that the new variables (i.e., principal components) are uncorrelated and most of the information within the initial variables is squeezed or compressed 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 scree plot below.



Percentage of Variance (Information) for each by PC

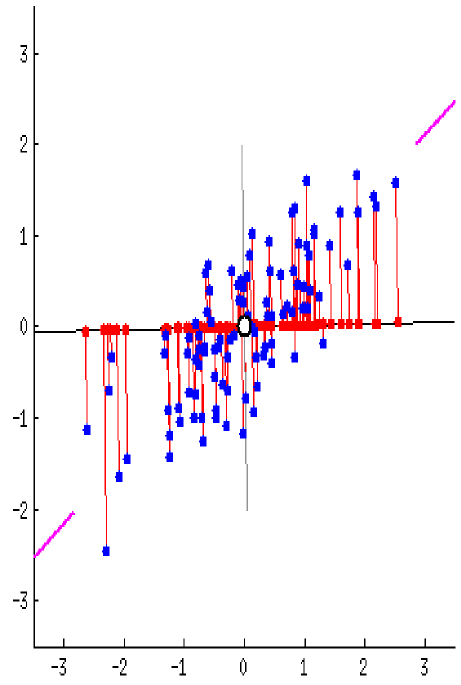
Organizing information in principal components this way, will allow you to reduce dimensionality without losing much information, and this by discarding the components with low information and considering the remaining components as your new variables.

An important thing to realize here is that, the principal components are less interpretable and don't have any real meaning since they are constructed as linear combinations of the initial variables.

Geometrically speaking, principal components represent the directions of the data that explain a **maximal amount of variance**, that is to say, the lines that capture most information of the data. The relationship between variance and information here, is that, the larger the variance carried by a line, the larger the dispersion of the data points along it, and the larger the dispersion along a line, the more the information it has. To put all this simply, just think of principal components as new axes that provide the best angle to see and evaluate the data, so that the differences between the observations are better visible.

How PCA Constructs the Principal Components

As there are as many principal components as there are variables 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. For example, let's assume that the scatter plot of our data set is as shown below, can we guess the first principal component ? Yes, it's approximately the line that matches the purple marks because it goes through the origin and it's the line in which the projection of the points (red dots) is the most spread out. Or mathematically speaking, it's the line that maximizes the variance (the average of the squared distances from the projected points (red dots) to the origin).



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.

This continues until a total of p principal components have been calculated, equal to the original number of variables.

Now that we understood what we mean by principal components, let’s go back to eigenvectors and eigenvalues. What you firstly need to know about them is that they always come in pairs, so that every eigenvector has an eigenvalue. And their number is equal to the number of dimensions of the data. For example, for a 3-dimensional data set, there are 3 variables, therefore there are 3 eigenvectors with 3 corresponding eigenvalues.

Without further ado, it is eigenvectors and eigenvalues who are behind all the magic explained above, because the eigenvectors of the Covariance matrix are actually *the directions of the axes where there is the most variance*(most information) and that we call Principal Components. And eigenvalues are simply the coefficients attached to eigenvectors, which 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 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% and 4% of the variance of the data.

STEP 4: FEATURE VECTOR

As we saw in the previous step, computing the eigenvectors and ordering them by their eigenvalues in descending order, allow us to find the principal components in order of significance. In this step, what we do is, to choose whether to keep all these components or discard those of lesser significance (of low eigenvalues), and form with the remaining ones a matrix of vectors that we call *Feature vector*.

So, the feature vector is simply a matrix that has as columns the eigenvectors of the components that we decide to keep. This makes it the first step towards dimensionality reduction, because if we choose to keep only ***p*** eigenvectors (components) out of ***n***, the final data set will have only ***p*** dimensions.

Example:

Continuing with the example from the previous step, we can either form a feature vector with both of the eigenvectors *v*1 and *v*2:

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

Or discard the eigenvector *v*2, which is the one of lesser significance, and form a feature vector with *v*1 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 *v*2 was carrying only 4% of the information, the loss will be therefore not important and we will still have 96% of the information that is carried by *v*1.

So, as we saw in the example, it’s up to you to choose whether to keep all the components or discard the ones of lesser significance, depending on what you are looking for. Because if you just want to describe your data in terms of new variables (principal components) that are uncorrelated without seeking to reduce dimensionality, leaving out lesser significant components is not needed.

LAST STEP: RECAST THE DATA ALONG THE PRINCIPAL COMPONENTS AXES

In the previous steps, apart from standardization, you do not make any changes on the data, you just select the principal components and form the feature vector, but the input data set remains always in terms of the original axes (i.e, in terms of the initial variables).

In this step, which is the last one, the aim is to use the feature vector formed using the eigenvectors of the covariance matrix, to reorient the data from the original axes to the ones represented by the principal components (hence the name Principal Components Analysis). This can be done by multiplying the transpose of the original data set by the transpose of the feature vector.

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

* * *

What is PCA?

Let’s say that you want to predict what the gross domestic product (GDP) of the United States will be for 2017. You have lots of information available: the U.S. GDP for the first quarter of 2017, the U.S. GDP for the entirety of 2016, 2015, and so on. You have any publicly-available economic indicator, like the unemployment rate, inflation rate, and so on. You have U.S. Census data from 2010 estimating how many Americans work in each industry and American Community Survey data updating those estimates in between each census. You know how many members of the House and Senate belong to each political party. You could gather stock price data, the number of IPOs occurring in a year, and how many CEOs seem to be mounting a bid for public office. Despite being an overwhelming number of variables to consider, this *just scratches the surface*.

TL;DR — you have *a lot* of variables to consider.

If you’ve worked with a lot of variables before, you know this can present problems. Do you understand the relationships between each variable? Do you have so many variables that you are in danger of overfitting your model to your data or that you might be violating assumptions of whichever modeling tactic you’re using?

You might ask the question, “How do I take all of the variables I’ve collected and focus on only a few of them?” 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 and you are less likely to overfit your model. (Note: This doesn't immediately mean that overfitting, etc. are no longer concerns — but we're moving in the right direction!)

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. In the GDP example above, instead of considering every single variable, we might drop all variables except the three we think will best predict what the U.S.'s gross domestic product will look like. 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. If we only use last year's GDP, the proportion of the population in manufacturing jobs per the most recent American Community Survey numbers, and unemployment rate to predict this year's GDP, 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.

Feature extraction, however, doesn't run into this problem. Say we have ten independent variables. In feature extraction, we create ten "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.

You might say, "Where does the dimensionality reduction come into play?" Well, we keep as many of the new independent variables as we want, but we drop the "least important ones." Because we ordered the new variables by how well they predict our dependent variable, we know which variable is the most important and least important. But — and here's the kicker — because these new independent variables are combinations of our old ones, we're still keeping the most valuable parts of our old variables, even when we drop one or more of these "new" variables!

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. If we decide to fit a linear regression model with these “new” variables (see “principal component regression” below), this assumption will necessarily be satisfied.

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?

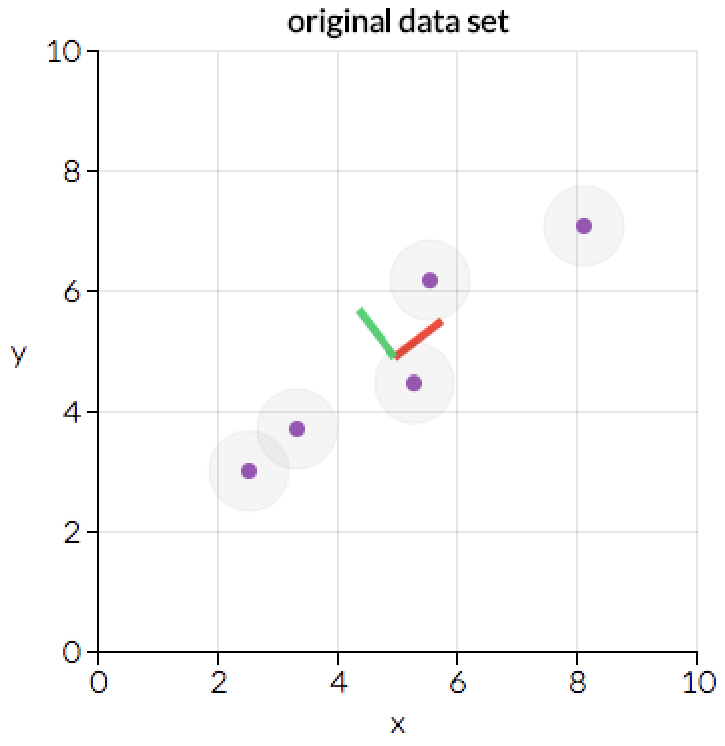
If you answered “yes” to all three questions, then PCA is a good method to use. If you answered “no” to question 3, you **should not** use PCA.

How does PCA work?

The section after this discusses *why* PCA works, but providing a brief summary before jumping into the algorithm may be helpful for context:

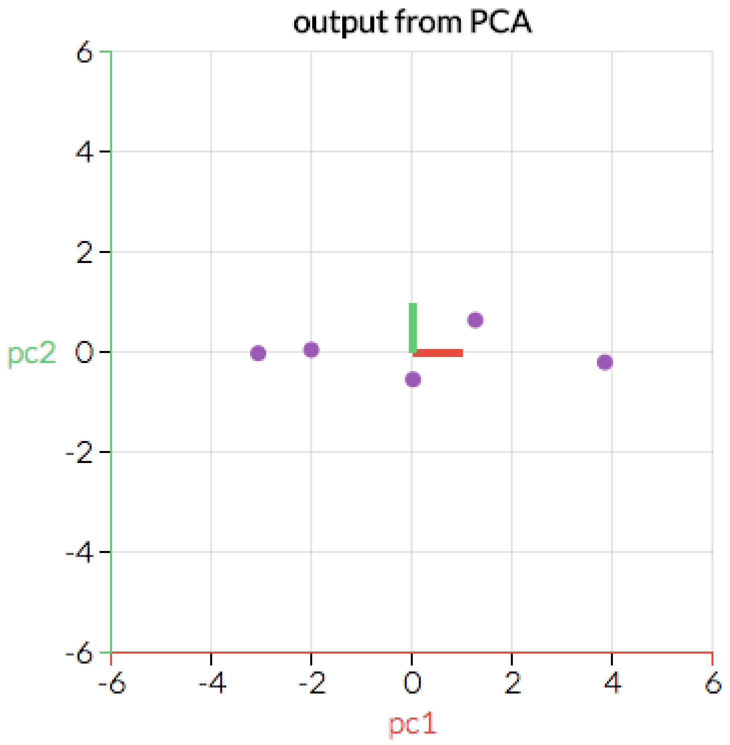
- We are going to calculate a matrix that summarizes how our variables all relate to one another.

- 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). The screenshot below, from the setosa.io applet, displays the two main directions in this data: the "red direction" and the "green direction." In this case, the "red direction" is the more important one. We'll get into why this is the case later, but given how the dots are arranged, can you see why the "red direction" looks more important than the "green direction?" (*Hint: What would fitting a line of best fit to this data look like?*)



Our original data in the xy-plane. (Source.)

- We will transform our original data to align with these important directions (which are combinations of our original variables). The screenshot below (again from setosa.io) is the same exact data as above, but transformed so that the x- and y-axes are now the "red direction" and "green direction." What would the line of best fit look like here?



Our original data transformed by PCA. (Source.)

- While the visual example here is two-dimensional (and thus we have two "directions"), think about a case where our data has more dimensions. By identifying which "directions" are most "important," we can compress or project our data into a smaller space by dropping the "directions" that are the "least important." **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!**

Here, I walk through an algorithm for conducting PCA. I try to avoid being too technical, but it's impossible to ignore the details here, so my goal is to walk through things as explicitly as possible. A deeper intuition of *why* the algorithm works is presented in the next section.

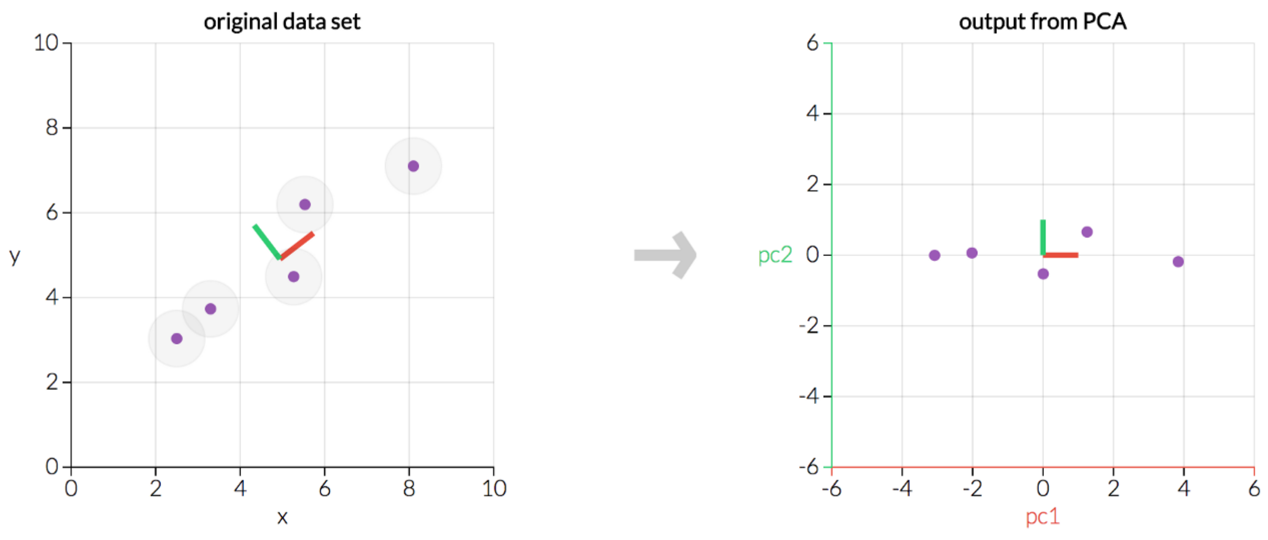
Before starting, you should have tabular data organized with *n* rows and likely *p+1* columns, where one column corresponds to your dependent variable (usually denoted ***Y***) and *p* columns where each corresponds to an independent variable (the matrix of which is usually denoted ***X***).

1. If a ***Y*** variable exists and is part of your data, then separate your data into ***Y*** and ***X***, as defined above — we'll mostly be working with ***X***. (Note: if there exists no column for **Y**, that's okay — skip to the next point!)
2. Take the matrix of independent variables ***X*** and, for each column, subtract the mean of that column from each entry. (This ensures that each column has a mean of zero.)
3. Decide whether or not to standardize. Given the columns of ***X***, are features with higher variance more important than features with lower variance, or is the importance of features independent of the variance? (In this case, importance means how well that feature predicts ***Y***.) **If the importance of features is independent of the variance of the features, then divide each observation in a column by that column's standard deviation.** (This, combined with step 2, standardizes each column of *X* to make sure

each column has mean zero and standard deviation 1.) Call the centered (and possibly standardized) matrix $\boldsymbol{Z}$.

4. Take the matrix $\boldsymbol{Z}$, transpose it, and multiply the transposed matrix by $\boldsymbol{Z}$. (Writing this out mathematically, we would write this as $\boldsymbol{Z}^T\boldsymbol{Z}$.) The resulting matrix is the covariance matrix of $\boldsymbol{Z}$, up to a constant.
5. (This is probably the toughest step to follow — stick with me here.) Calculate the eigenvectors and their corresponding eigenvalues of $\boldsymbol{Z}^T\boldsymbol{Z}$. This is quite easily done in most computing packages— in fact, the eigendecomposition of $\boldsymbol{Z}^T\boldsymbol{Z}$ is where we decompose $\boldsymbol{Z}^T\boldsymbol{Z}$ into $\boldsymbol{PDP}^{-1}$, where $\boldsymbol{P}$ is the matrix of eigenvectors and $\boldsymbol{D}$ is the diagonal matrix with eigenvalues on the diagonal and values of zero everywhere else. The eigenvalues on the diagonal of $\boldsymbol{D}$ will be associated with the corresponding column in $\boldsymbol{P}$ — that is, the first element of $\boldsymbol{D}$ is λ_1 and the corresponding eigenvector is the first column of $\boldsymbol{P}$. This holds for all elements in $\boldsymbol{D}$ and their corresponding eigenvectors in $\boldsymbol{P}$. We will always be able to calculate $\boldsymbol{PDP}^{-1}$ in this fashion. (Bonus: for those interested, we can always calculate $\boldsymbol{PDP}^{-1}$ in this fashion because $\boldsymbol{Z}^T\boldsymbol{Z}$ is a symmetric, positive semidefinite matrix.)
6. Take the eigenvalues $\lambda_1, \lambda_2, \ldots, \lambda_p$ and sort them from largest to smallest. In doing so, sort the eigenvectors in $\boldsymbol{P}$ accordingly. (For example, if λ_2 is the largest eigenvalue, then take the second column of $\boldsymbol{P}$ and place it in the first column position.) Depending on the computing package, this may be done automatically. Call this sorted matrix of eigenvectors $\boldsymbol{P^*}$. (The columns of $\boldsymbol{P^*}$ should be the same as the columns of $\boldsymbol{P}$, but perhaps in a different order.) **Note that these eigenvectors are independent of one another.**

7. Calculate ***Z**** = ***ZP****. This new matrix, ***Z****, is a centered/standardized version of ***X*** but now each observation is a combination of the original variables, where the weights are determined by the eigenvector. **As a bonus, because our eigenvectors in *P** are independent of one another, each column of *Z** is also independent of one another!**



An example from setosa.io where we transform five data points using PCA. The left graph is our original data *X; the right graph would be our transformed data Z*.*

Note two things in this graphic:

- The two charts show the exact same data, but the right graph reflects the original data transformed so that our axes are now the principal components.
- In both graphs, the principal components are perpendicular to one another. **In fact, every principal component will ALWAYS be orthogonal** (a.k.a. official math term for perpendicular) **to every other principal component.** (Don't believe me? Try to break the applet!)

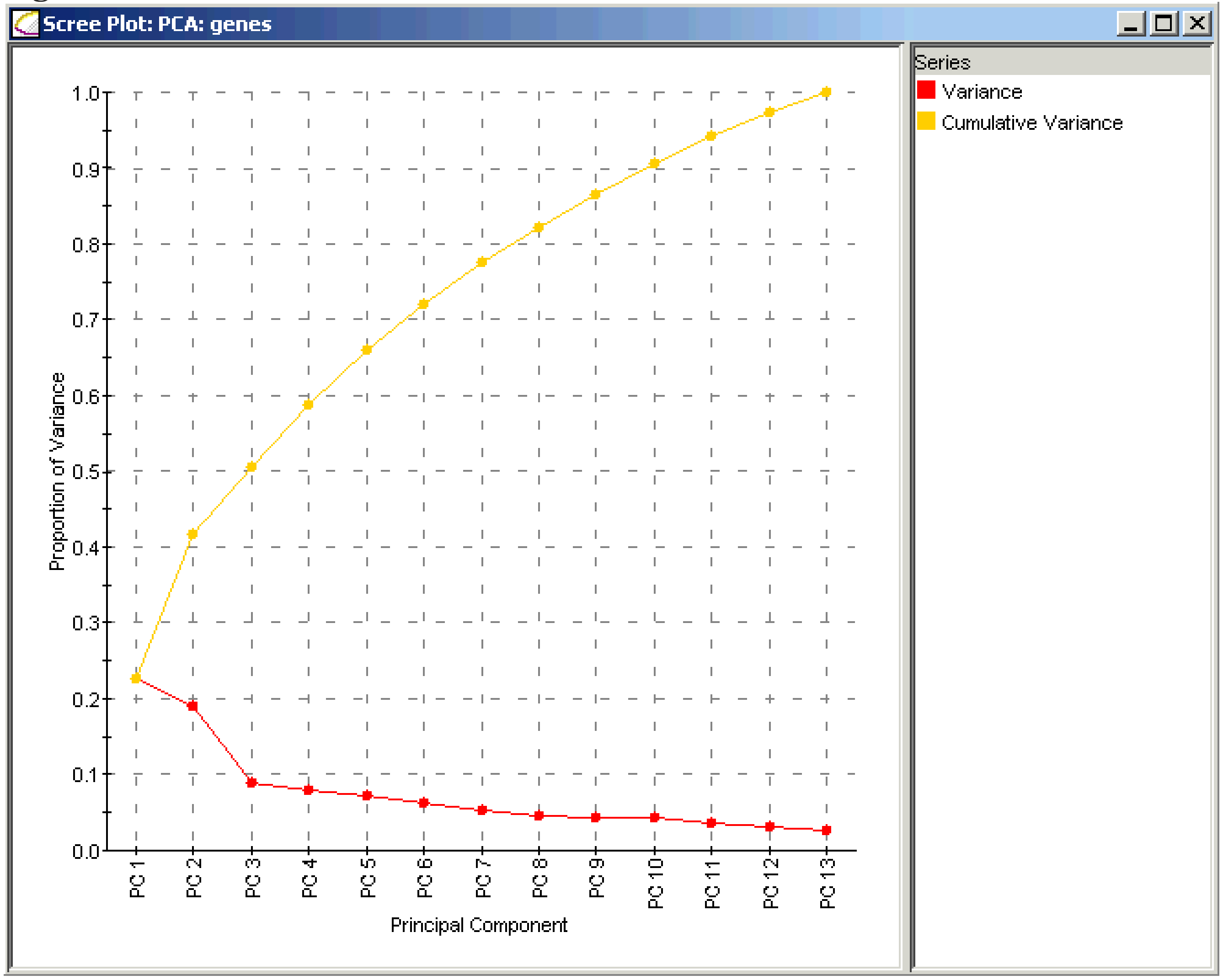
Because our principal components are orthogonal to one another, they are statistically linearly independent of one another… which is why our columns of *Z are linearly independent of one another!**

8. 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 (briefly explained below) 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. (This plot is called a scree plot, shown below.) 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.)

Because each eigenvalue is roughly the importance of its corresponding eigenvector, the proportion of variance explained is the sum of the eigenvalues of the features you kept divided by the sum of the eigenvalues of all features.



Scree Plot for Genetic Data. (Source.)

Consider this scree plot for genetic data. (Source: here.) The red line indicates the proportion of variance explained by each feature, which is calculated by taking that principal component's eigenvalue divided by the sum of all eigenvalues. The proportion of variance explained by including only principal component 1 is $\lambda_1/(\lambda_1 + \lambda_2 + \ldots + \lambda_p)$, which is about 23%. The proportion of variance explained by including only principal component 2 is $\lambda_2/(\lambda_1 + \lambda_2 + \ldots + \lambda_p)$, or about 19%.

The proportion of variance explained by including both principal components 1 and 2 is $(\lambda_1 + \lambda_2)/(\lambda_1 + \lambda_2 + \ldots + \lambda_p)$, which is about 42%. This is where the yellow line comes in; the yellow line indicates the cumulative proportion of variance explained if you included all principal components up to that point. For example, the yellow dot above PC2 indicates that including principal components 1 and 2 will explain about 42% of the total variance in the model.

Now let's go through some examples:

- Method 1: We arbitrarily select a number of principal components to include. Suppose I wanted to keep five principal components in my model. In the genetic data case above, these five principal components explains about 66% of the total variability that would be explained by including all 13 principal components.
- Method 2: Suppose I wanted to include enough principal components to explain 90% of the total variability explained by all 13 principal components. In the genetic data case above, I would

include the first 10 principal components and drop the final three variables from $\boldsymbol{Z}^*$.

- Method 3: Here, we want to "find the elbow." In the scree plot above, we see there's a big drop in proportion of variability explained between principal component 2 and principal component 3. In this case, we'd likely include the first two features and drop the remaining features. As you can see, this method is a bit subjective as "elbow" doesn't have a mathematically precise definition and, in this case, we'd include a model that explains only about 42% of the total variability.

Once we've dropped the transformed variables we want to drop, we're done! That's PCA.

But, like, *why* does PCA work?

While PCA is a very technical method relying on in-depth linear algebra algorithms, it's a relatively intuitive method when you think about it.

- First, the covariance matrix $\boldsymbol{Z}^T\boldsymbol{Z}$ is a matrix that contains estimates of how every variable in $\boldsymbol{Z}$ relates to every other variable in $\boldsymbol{Z}$. Understanding how one variable is associated with another is quite powerful.
- Second, eigenvalues and eigenvectors are important. Eigenvectors represent directions. Think of plotting your data on a multidimensional scatterplot. Then one can think of an individual eigenvector as a particular "direction" in your scatterplot of data.

Eigenvalues represent magnitude, or importance. Bigger eigenvalues correlate with more important directions.

- Finally, we make an assumption that more variability in a particular direction correlates with explaining the behavior of the dependent variable. Lots of variability usually indicates signal, whereas little variability usually indicates noise. Thus, the more variability there is in a particular direction is, theoretically, indicative of something important we want to detect. (The setosa.io PCA applet is a great way to play around with data and convince yourself why it makes sense.)

Thus, PCA is a method that brings together:

1. A measure of how each variable is associated with one another. (Covariance matrix.)
2. The directions in which our data are dispersed. (Eigenvectors.)
3. The relative importance of these different directions. (Eigenvalues.)

PCA combines our predictors and allows us to drop the eigenvectors that are relatively unimportant.