Data pre-processing

Michela Mulas

A quick recap

During the last lecture, we did...

- Introduce the basic procedure for a predictive model design.
- Investigate different datasets.
- Illustrate a case study: predicting fuel economy

Today's goal

Today, we going to discuss about...

Data pre-processing: addition, deletion or transformation of training set data.

- Case study: Cell segmentation in high-content screening.
- Data transformation for individual predictors.
- Data transformation for multiple predictors.
  - Briefly describe the "Principal component analysis".
  - Execute a simple example in R.

Reading list

Max Kuhn and Kjell Johnson. *Applied Predictive Modeling*, Springer (2014)

Today's lecture is mainly based on Chapter 3 of the book.

Motivations

Data pre-processing techniques generally refer to the addition, deletion, or transformation of training set data.

- Data preparation can **make or break a model's predictive ability**.
- Different models have different sensitivities to the type of predictors in the model.
- **Transformations of the data** to reduce the impact of data skewness or outliers can lead to significant improvements in performance.

Feature extraction is one empirical technique for creating surrogate variables that are combinations of multiple predictors.

Motivations

The **need for data pre-processing is determined by the type of model being used**.

- Some procedures, such as tree-based models, are notably insensitive to the characteristics of the predictor data.
- Others, like linear regression, are not.

How the predictors are encoded can have a significant impact on model performance.

- For example, using combinations of predictors can sometimes be more effective than using the individual values: the ratio of two predictors may be more effective than using two independent predictors.
- Often the most effective encoding of the data is informed by the **modeler's understanding of the problem** and thus is not derived from any mathematical technique.

Case study: Cell segmentation in high-content screening

Medical researchers often seek to understand the effects of medicines or diseases on the size, shape, development status, and number of cells in a living organism or plant.

To do this, experts can examine the target serum or tissue under a microscope and **manually assess the desired cell characteristics**.

- This work is tedious and requires expert knowledge of the cell type and characteristics.

Another way to measure the cell characteristics from these kinds of samples is by **using high-content screening**[1]:

- A sample is first dyed with a substance that will bind to the desired characteristic of the cells.
- The sample is then interrogated by an instrument (such as a confocal microscope) where the dye deflects light and the detectors quantify the degree of scattering for that specific wavelength.
- The light scattering measurements are then processed through imaging software to quantify the desired cell characteristics.

[1] Giuliano K, DeBiasio R, Dunlay R, Gough A, Volosky J, Zock J, Pavlakis G, Taylor D (1997). *High-Content Screening: A New Approach to Easing Key Bottlenecks in the Drug Discovery Process.* Journal of Biomolecular Screening, 2(4), 249-259.

Case study: Cell segmentation in high-content screening

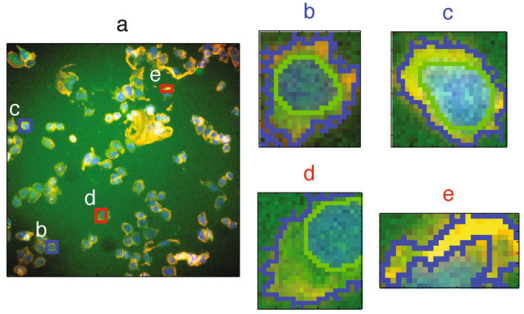
Using an automated, high-throughput approach to assess samples' cell characteristics can sometimes produce misleading results.

Hill et al. (2007)[1] describe a research project that used high-content screening to measure several aspects of cells.

They observed that the imaging software used to determine the location and shape of the cell had difficulty segmenting cells (i.e., defining cells' boundaries).

[1] Hill A, LaPan P, Li Y, Haney S (2007). *Impact of Image Segmentation on High-Content Screening Data Quality for SK–BR-3 Cells.* BMC Bioinformatics, 8(1), 340.

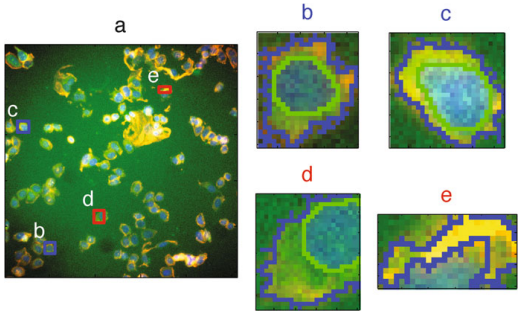
Case study: Cell segmentation in high-content screening



The Figure depicts several example cells from this study.

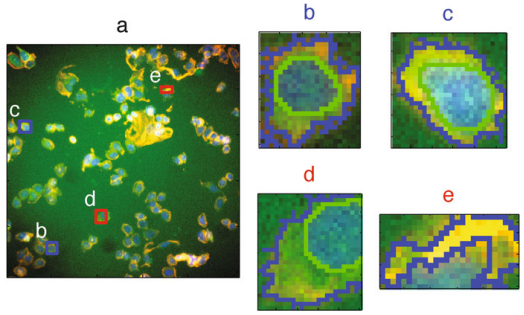
- The bright green boundaries identify the cell nucleus.
- The blue boundaries define the cell perimeter.
- Clearly some cells are well segmented, while others are not.

Case study: Cell segmentation in high-content screening



- Cells that are poorly segmented appear to be damaged, when in reality they are not.
- If cell size, shape, and/or quantity are the endpoints of interest in a study, then it is important that the instrument and imaging software can correctly segment cells.

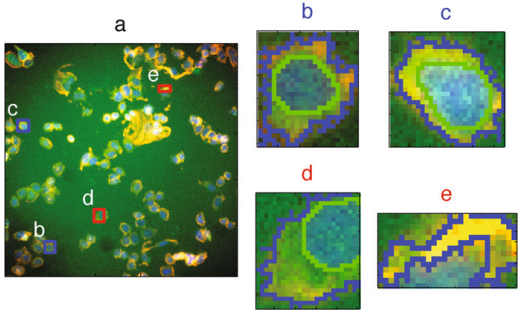
Case study: Cell segmentation in high-content screening



For this research, Hill et al. (2007) assembled a data set consisting of 2019 cells.

- 1300 were judged to be poorly segmented and 719 were well segmented.
- 1009 cells were reserved for the **training set**.

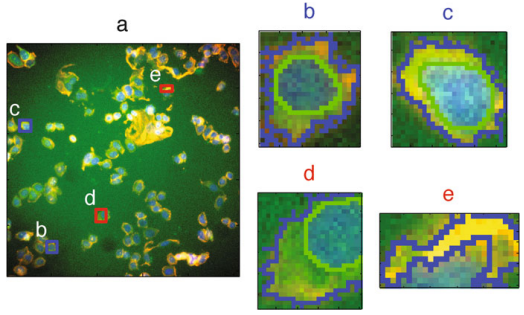
Case study: Cell segmentation in high-content screening



For a particular type of cell, the researchers used different stains that would be visible to different optical channels.

- Channel one was associated with the cell body and can be used to determine the cell perimeter, area, and other qualities.
- Channel two interrogated the cell nucleus by staining the nuclear DNA.
- Channels three and four were stained to detect actin and tubulin, respectively.

Case study: Cell segmentation in high-content screening



Actin and tubulin are two types of filaments that transverse the cells in scaffolds and are part of the cell's cytoskeleton.

For all cells, 116 features (e.g., cell area, spot fiber count) were measured and were used to predict the segmentation quality of cells.2

We will use the training set samples identified by the original authors to **demonstrate data pre-processing techniques**.

Data transformations for individual predictors: Centering and scaling

Centering and scaling are manipulations, generally used to improve the numerical stability of some calculations.

The only real downside to these transformations is a loss of interpretability of the individual values since the data are no longer in the original units.

To center a predictor variable, the average predictor value is subtracted from all the values.

- As a result of centering, **the predictor has a zero mean**.

To scale the data, each value of the predictor variable is divided by its standard deviation.

- As a result of scaling, the data have **common standard deviation equal to one**.

Data transformations for individual predictors: Transformations to Resolve Skewness

An un-skewed distribution is one that is roughly symmetric.

- This means that the probability of falling on either side of the distribution's mean is roughly equal.
- A right-skewed distribution has a large number of points on the left side of the distribution (smaller values) than on the right side (larger values).

For example, the cell segmentation data contain a predictor that measures the standard deviation of the intensity of the pixels in the actin filaments.

Data transformations for individual predictors: Transformations to Resolve Skewness

A general rule of thumb to consider is that **skewed data whose ratio of the highest value to the lowest value is greater than 20** have significant skewness.

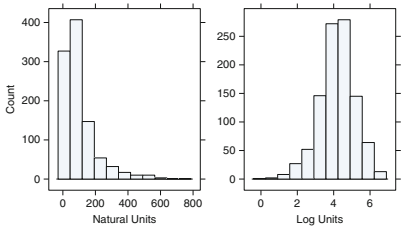
- The predictor distribution is roughly symmetric: the skewness will be close to zero.
- The distribution becomes more right skewed: the skewness becomes larger.
- The distribution becomes more left skewed, the value becomes negative.

$$\text{skewness} = \frac{\sum(x_i - \bar{x})^3}{(n-1)v^{3/2}} \quad \text{where} \quad v = \frac{\sum(x_i - \bar{x})^2}{(n-1)}$$

- x is the predictor variable.
- n is the number of values.
- $\bar{x}$ is the sample mean of the predictor.

Data transformations for individual predictors: Transformations to Resolve Skewness

Case study: Cell segmentation in high-content screening



The cell segmentation data contain a predictor that measures the standard deviation of the intensity of the pixels in the actin filaments.

- **Left**: This predictor has a strong right skewness with a concentration of points with low values. For this variable, the ratio of the smallest to largest value is 870 and a skewness value of 2.39.
- **Right**: The same data after a log transformation. The skewness value for the logged data was -0.4

Data transformations for individual predictors: Transformations to Resolve Skewness

Alternatively, statistical methods can be used to empirically identify an appropriate transformation.

Box and Cox (1964)[1] propose a family of transformations that are indexed by a parameter, denoted as λ:

$$x^* = \begin{cases} \frac{x^\lambda - 1}{\lambda} & \text{if } \lambda \neq 0 \\ \log(x) & \text{if } \lambda = 0 \end{cases}$$

- In addition to the log transformation, this family can identify square transformation ($\lambda = 2$), square root ($\lambda = 0.5$), inverse ($\lambda = -1$), and others in-between.
- Using the training data, λ can be estimated.

[1] Box G, Cox D (1964). *An Analysis of Transformations*. Journal of the Royal Statistical Society. Series B (Methodological), pp. 211-252.

Data transformations for multiple predictors: Transformations to Resolve Outliers

We will generally define outliers as **samples that are exceptionally far from the mainstream of the data**.

- Under certain assumptions, there are formal statistical definitions of an outlier.
- Even with a thorough understanding of the data, **outliers can be hard to define**.
- However, we can often identify an unusual value by looking at a figure.

When one or more samples are suspected to be outliers, the first step is to **make sure that the values are scientifically valid** (e.g., positive blood pressure) and that no data recording errors have occurred.

Great care should be taken not to hastily remove or change values, especially if the sample size is small. With small sample sizes, apparent outliers might be a result of a skewed distribution where there are not yet enough data to see the skewness.

Data transformations for multiple predictors: Transformations to Resolve Outliers

There are several predictive **models that are resistant to outliers**.

- **Tree-based classification models** create splits of the training data and the prediction equation is a set of logical statements such as "if predictor A is greater than X, predict the class to be Y", so the outlier does not usually have an exceptional influence on the model.
- **Support vector machines for classification** generally disregard a portion of the training set samples when creating a prediction equation. The excluded samples may be far away from the decision boundary and outside of the data mainstream.

Data transformations for multiple predictors: Transformations to Resolve Outliers

If a **model is considered to be sensitive to outliers**, one data transformation that can minimize the problem is the **spatial sign**[1].

- The procedure projects the predictor values onto a multidimensional sphere.
- This has the effect of making all the samples the same distance from the center of the sphere. **It does not remove outliers**.
- Mathematically, each sample is divided by its Euclidean norm:

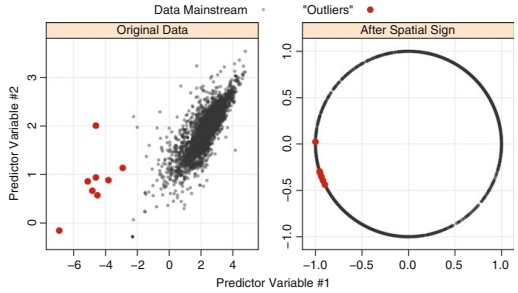
$$x_{ij}^* = \frac{x_{ij}}{\sqrt{\sum_{j=1}^{P} x_{ij}^2}}$$

The denominator measures the distance to the center of the predictor's distribution. Center and scale the predictor data prior to using this transformation is important.

Note that, unlike centering or scaling, this manipulation of the predictors transforms them as a group.

[1] Serneels S, Nolf ED, Espen PV (2006). *Spatial Sign Pre-processing: A Simple Way to Impart Moderate Robustness to Multivariate Estimators*. Journal of Chemical Information and Modeling, 46(3), 1402-1409.

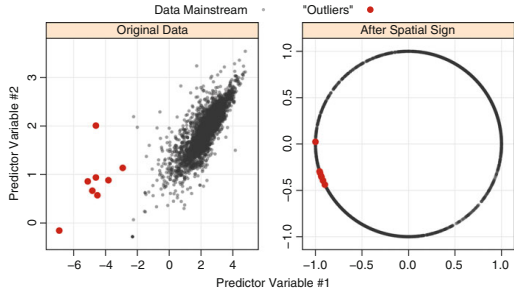
Data transformations for multiple predictors: Transformations to Resolve Outliers



In these data, at least eight samples cluster away from the majority of other data.

- These data points are likely a valid, but poorly sampled subpopulation of the data.
- The modeler would investigate why these points are different; perhaps they represent a group of interest, such as highly profitable customers.

Data transformations for multiple predictors: Transformations to Resolve Outliers



In these data, at least eight samples cluster away from the majority of other data.

- In the spatial sign transformation all the data points are projected to be a common distance away from the origin.
- The outliers still reside in the Northwest section of the distribution but are contracted inwards. This mitigates the effect of the samples on model training.

Data transformations for multiple predictors: Data Reduction and Feature Extraction

Data reduction techniques reduce the data by generating a smaller set of predictors that seek to capture a majority of the information in the original variables.

- In this way, fewer variables can be used that provide reasonable fidelity to the original data.
- For most data reduction techniques, the new predictors are functions of the original predictors; therefore, all the original predictors are still needed to create the surrogate variables.
- This class of methods is often called **signal extraction** or **feature extraction**.

Data transformations for multiple predictors: Data Reduction and Feature Extraction

Principal Component Analysis, PCA, is a commonly used data reduction technique.

The goals of PCA are[1]:

- Extract the most important information from the data;
- Compress the size of the data set by keeping only this important information;
- Simplify the description of the data set;
- Analyze the structure of the observations and the variables.

Data transformations for multiple predictors: Data Reduction and Feature Extraction

In order to achieve its goals, PCA computes new variables called **principal components** which are obtained as linear combinations of the original variables.

- The first principal component is required to have the largest possible variance.
- The second component is computed under the constraint of being orthogonal to the first component and to have the largest possible inertia.
- The other components are computed likewise.

The values of these new variables for the observations are called **factor scores**, and these factors scores can be interpreted geometrically as the projections of the observations onto the principal components.

Principal component analysis

Background of statistics

The entire subject of statistics is based around the idea that we have this big set of data, and we want to analyse that set in terms of the relationships between the individual points in that data set.

Let us assume that our data set is X of which we can calculate the mean:

- **Mean**: $\bar{X} = \frac{\sum_{i=1}^{n} X_i}{n}$

Unfortunately, the mean doesn't tell us a lot about the data except for a sort of middle point. We can use the standard deviation to measure how spread out the data is:

- **Standard deviation**: $s = \sqrt{\frac{\sum_{i=1}^{n}(X_i - \bar{X})^2}{n-1}}$

Variance is another measure of the spread of data in a data set. In fact it is almost identical to the standard deviation.

- **Variance**: $s^2 = \frac{\sum_{i=1}^{n}(X_i - \bar{X})^2}{n-1}$

Principal component analysis

Background of statistics

Standard deviation and variance only operate on 1 dimension:

We could only calculate the standard deviation for each dimension of the data set independently of the other dimensions.

However, it is useful to have a similar measure to find out how much the dimensions vary from the mean with respect to each other.

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

Remember that covariance is always measured between 2 dimensions.

If we have a data set with more than 2 dimensions, there is more than one covariance measurement that can be calculated.

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

Principal component analysis

Background of matrix algebra

Eigenvectors and eigenvalues are numbers and vectors associated to square matrices. Together they provide the **eigen-decomposition of a matrix**, which analyzes the structure of this matrix.

- Even though the eigen-decomposition does not exist for all square matrices, it has a particularly simple expression for matrices such as correlation, covariance, or cross-product matrices.
- Specifically PCA is obtained from the eigen-decomposition of a covariance or a correlation matrix.

There are several ways to define eigenvectors and eigenvalues, the most common approach defines an eigenvector of a matrix as a vector $\mathbf{u}$ that satisfies the following:

$$\mathbf{A}\mathbf{u} = \lambda \mathbf{u}$$
$$(\mathbf{A} - \lambda \mathbf{I})\mathbf{u} = 0$$

- λ are the **eigenvalues** associated to the **eigenvectors**

Principal component analysis

Step by step[1]

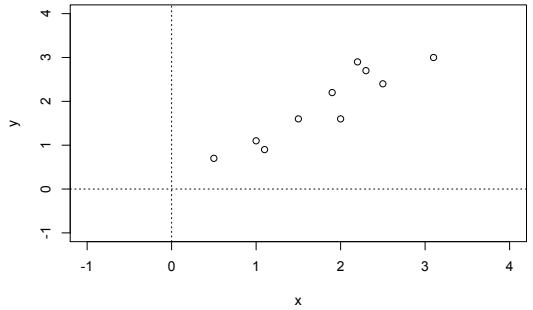
- **Step 1**: Get some data.
- **Step 2**: Subtract the mean.
- **Step 3**: Calculate the covariance matrix.
- **Step 4**: Calculate the eigevectors and eigenvalues of the covariance matrix.
- **Step 5**: Form the feature vectors.
- **Step 6**: Deriving the new dataset.
- **Step 7**: Get the old data back.

[1] http://snobear.colorado.edu/Markw/BioMath/Otis/PCA/principal_components.ps

Principal component analysis

A simple example in R

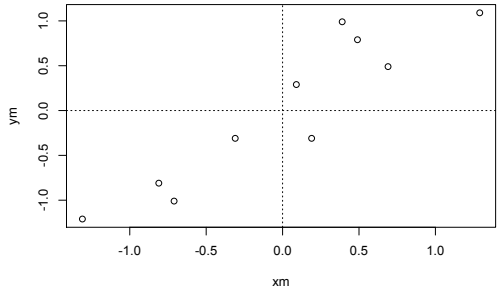
```
## Get some data
> x <- c(2.5,0.5,2.2,1.9,3.1,2.3,2.0,1.0,1.5,1.1)
> y <- c(2.4,0.7,2.9,2.2,3.0,2.7,1.6,1.1,1.6,0.9)
>
> # Plot the data
> plot(x,y,xlim=c(-1,4),ylim=c(-1,4)); abline(h=0,v=0,lty=3)
```



Principal component analysis

A simple example in R

```
##  Subtract the mean
> xm <- x - mean(x)
> ym <- y - mean(y)
>
> # Plot the data
> plot(xm,ym); abline(h=0,v=0,lty=3)
```



dimensions. This produces a **data set whose mean is zero**.

Principal component analysis

A simple example in R

```
##  Make a matrix of the given data and calculate the covariance matrix
> m <- matrix(c(xm,ym),ncol=2)
> m
       [,1]  [,2]
 [1,]  0.69  0.49
 [2,] -1.31 -1.21
 [3,]  0.39  0.99
 [4,]  0.09  0.29
 [5,]  1.29  1.09
 [6,]  0.49  0.79
 [7,]  0.19 -0.31
 [8,] -0.81 -0.81
 [9,] -0.31 -0.31
[10,] -0.71 -1.01
>
> cov.m <- cov(m)
> cov.m
          [,1]      [,2]
[1,] 0.6165556 0.6154444
[2,] 0.6154444 0.7165556
>
```

- The non-diagonal elements in this covariance matrix are positive: both the *x* and *y* increase together.

Principal component analysis
A simple example in R

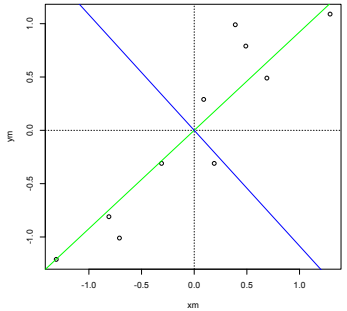
```
## Calculate the eigenvectors and eigenvalues of the covariance matrix
> cov.eig <- eigen(cov.m)
> cov.eig
$values
[1] 1.2840277 0.0490834

$vectors
          [,1]       [,2]
[1,] 0.6778734 -0.7351787
[2,] 0.7351787  0.6778734
```

Notice that the **eigenvalues are quite different values**.

- It turns out that the eigenvector with the highest eigenvalue is the principle component of the data set.
- In our example, the eigenvector with the larges eigenvalue is the one that pointed down the middle of the data. It is the most significant relationship between the data dimensions (see next plot).

Principal component analysis
A simple example in R

```
## Plot the eigenvectors (they represent the new basis) onto the data
> plot(xm,ym); abline(h=0,v=0,lty=3)
>
> abline(a=0,b=(cov.eig$vectors[1,1]/cov.eig$vectors[2,1]),col="green")
> abline(a=0,b=(cov.eig$vectors[1,2]/cov.eig$vectors[2,2]),col="blue")
```



The first eigenvector (green) goes through the middle of the points, like drawing a line of best fit. It is showing us how these two data sets are related along that line.

The second eigenvector (blue) gives us the other, less important, pattern in the data, that all the points follow the main line, but are off to the side of the main line by some amount.

Principal component analysis
A simple example in R

```
## Plot the eigenvectors (they represent the new basis) onto the data
> plot(xm,ym); abline(h=0,v=0,lty=3)
>
> abline(a=0,b=(cov.eig$vectors[1,1]/cov.eig$vectors[2,1]),col="green")
> abline(a=0,b=(cov.eig$vectors[1,2]/cov.eig$vectors[2,2]),col="blue")
```

The eigenvectors are ordered by eigenvalue, highest to lowest.

- This gives us the components in order of significance.

Now, we can decide to ignore the components of lesser significance.

- We do lose some information, but if the eigenvalues are small, we don't lose much.
- If we leave out some components, **the final data set will have less dimensions than the original**.
- If we originally have *n* dimensions in the data, and so we calculate *p* eigenvectors and eigenvalues, and then we choose only the first *p* eigenvectors, then the final data set has only *p* dimensions.

Principal component analysis
A simple example in R

```
> ## Form the feature vectors
> # Feature vector with just one component
> feature.vector1 <- as.matrix(cov.eig$vectors[,1],ncol=1)
> feature.vector1
          [,1]
[1,] 0.6778734
[2,] 0.7351787
>
> # Feature vector with both components
> feature.vector2 <- as.matrix(cov.eig$vectors[,c(1,2)],ncol=2)
> feature.vector2
          [,1]       [,2]
[1,] 0.6778734 -0.7351787
[2,] 0.7351787  0.6778734
```

Here, we formed the **feature vector**, just a fancy name for a matrix of vectors.

- This is constructed by taking the eigenvectors that you want to keep from the list of eigenvectors, and forming a matrix with these eigenvectors in the columns.

Principal component analysis
A simple example in R

```
## Derive the new transformed dataset
> # New dataset for feature vector 1
> final1 <- t(feature.vector1) %*% t(m)
> final1
     [,1] [,2] [,3] [,4] [,5] [,6]   [,7] [,8]   [,9] [,10]
[1,] 0.83 -1.8 0.99 0.27  1.7 0.91 -0.099 -1.1 -0.44  -1.2
>
> # New dataset for feature vector 2
> final2 <- t(feature.vector2) %*% t(m)
> final2
      [,1]  [,2] [,3] [,4]  [,5] [,6]   [,7]   [,8]   [,9] [,10]
[1,]  0.83 -1.78 0.99 0.27  1.68 0.91 -0.099 -1.145 -0.438 -1.22
[2,] -0.18  0.14 0.38 0.13 -0.21 0.18 -0.350  0.046  0.018 -0.16
```

$$FinalData = RowFeatureVector \times RowDataAdjust$$

- *RowFeatureVector* is the matrix with the eigenvectors in the columns transposed so that the eigenvectors are now in the rows (the most significant eigenvect. at the top).
- *RowDataAdjust* is mean-adjusted data transposed (the data items are in each column, with each row holding a separate dimension).

Principal component analysis
A simple example in R

```
## Derive the new transformed dataset
> # New dataset for feature vector 1
> final1 <- t(feature.vector1) %*% t(m)
> final1
     [,1] [,2] [,3] [,4] [,5] [,6]   [,7] [,8]   [,9] [,10]
[1,] 0.83 -1.8 0.99 0.27  1.7 0.91 -0.099 -1.1 -0.44  -1.2
>
> # New dataset for feature vector 2
> final2 <- t(feature.vector2) %*% t(m)
> final2
      [,1]  [,2] [,3] [,4]  [,5] [,6]   [,7]   [,8]   [,9] [,10]
[1,]  0.83 -1.78 0.99 0.27  1.68 0.91 -0.099 -1.145 -0.438 -1.22
[2,] -0.18  0.14 0.38 0.13 -0.21 0.18 -0.350  0.046  0.018 -0.16
```

This give us the original data solely in terms of the vectors we chose.

- Our original data set had two axes, x and y, so our data was in terms of them.
- It is possible to express data in terms of any two axes that you like. If these axes are perpendicular, like the eigenvectors, then the expression is the most efficient.

Principal component analysis
A simple example in R

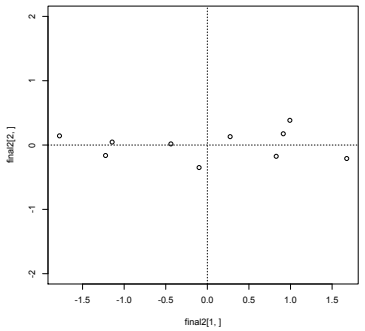
```
## Derive the new transformed dataset
> # New dataset for feature vector 1
> final1 <- t(feature.vector1) %*% t(m)
> final1
     [,1] [,2] [,3] [,4] [,5] [,6]   [,7] [,8]   [,9] [,10]
[1,] 0.83 -1.8 0.99 0.27  1.7 0.91 -0.099 -1.1 -0.44  -1.2
>
> # New dataset for feature vector 2
> final2 <- t(feature.vector2) %*% t(m)
> final2
      [,1]  [,2] [,3] [,4]  [,5] [,6]   [,7]   [,8]   [,9] [,10]
[1,]  0.83 -1.78 0.99 0.27  1.68 0.91 -0.099 -1.145 -0.438 -1.22
[2,] -0.18  0.14 0.38 0.13 -0.21 0.18 -0.350  0.046  0.018 -0.16
```

This give us the original data solely in terms of the vectors we chose.

- We have changed our data from being in terms of the axes x and y, and now they are in terms of our 2 eigenvectors.
- In the case of the new data set has reduced dimensionality (that is, we have left some of the eigenvectors out), the new data is only in terms of the vectors that we decided to keep.

Principal component analysis
A simple example in R

```
> ## Derive the new transformed dataset
> # final2 has 2 dimensions, we can plot it:
plot(final2[1,],final2[2,],ylim=c(-2,2));abline(h=0,v=0,lty=3)
```



In the case we keep both eigenvalues we get the data and the plot in the figure.

This plot is basically the original data, rotated so that the **eigenvectors are the axes**.

We have lost no information in this decomposition.

Principal component analysis
A simple example in R

```
> ## Derive the new transformed dataset
> # final1 has 1 dimension
> final1
     [,1] [,2] [,3] [,4] [,5] [,6]   [,7] [,8]   [,9] [,10]
[1,] 0.83 -1.8 0.99 0.27  1.7 0.91 -0.099 -1.1 -0.44  -1.2
```

The other transformation we can make is by taking only the eigenvector with the largest eigenvalue. As expected, it only has a single dimension.

If we compare this data set with the one resulting from using both eigenvectors, we notice that this data set is exactly the first column of the other.

We have effectively thrown away the whole other axis, which is the other eigenvector.

Principal component analysis
A simple example in R

What we have done?

We have transformed our data so that is **expressed in terms of the patterns between them**, where the patterns are the lines that most closely describe the relationships between the data.

Initially, we had the simple *x* and *y* axes.

- This is fine, but the and values of each data point don?t really tell us exactly how that point relates to the rest of the data.

Now, the values of the data points tell us exactly where (ie. above/below) the trend lines the data point sits.

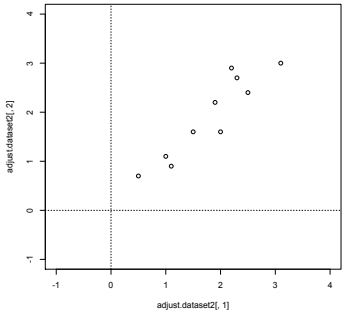
- In the case of the transformation using both eigenvectors, we have simply altered the data so that it is in terms of those eigenvectors instead of the usual axes.
- The single-eigenvector decomposition has removed the contribution due to the smaller eigenvector and left us with data that is only in terms of the other.

Principal component analysis
A simple example in R

```
> ## Get the old data back
> # If we keep 2 components, we can recover it by 100% (like in final2)
> adjust.dataset2 <- t(feature.vector2 %*% final2)
>
> adjust.dataset2[,1] <- adjust.dataset2[,1] + mean(x)
> adjust.dataset2[,2] <- adjust.dataset2[,2] + mean(y)
> adjust.dataset2
      [,1] [,2]
 [1,]  2.5  2.4
 [2,]  0.5  0.7
 [3,]  2.2  2.9
 [4,]  1.9  2.2
 [5,]  3.1  3.0
 [6,]  2.3  2.7
 [7,]  2.0  1.6
 [8,]  1.0  1.1
 [9,]  1.5  1.6
[10,]  1.1  0.9
```


Principal component analysis
A simple example in R

```
> ## Get the old data back
> # If we keep 2 components, we can recover it by 100% (like in final2)
> plot(adjust.dataset2[,1],adjust.dataset2[,2],xlim=c(-1,4),ylim=c(-1,4))
> abline(h=0,v=0,lty=3)
```



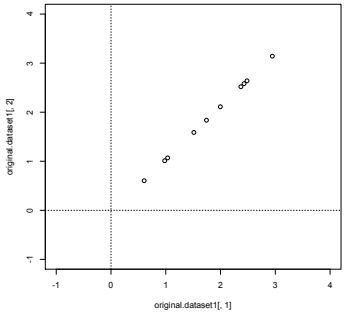
The result is exactly the data we started with.

Principal component analysis
A simple example in R

```
> ## Get the old data back
> # If we keep just one component, we do the same
> # We cannot expect the original dataset
> adjust.dataset1 <- t(feature.vector1 %*% final1)
>
> adjust.dataset1[,1] <- adjust.dataset1[,1] + mean(x) # re-add means
> adjust.dataset1[,2] <- adjust.dataset1[,2] + mean(y)
> adjust.dataset1
      [,1] [,2]
 [1,] 2.37  2.5
 [2,] 0.61  0.6
 [3,] 2.48  2.6
 [4,] 2.00  2.1
 [5,] 2.95  3.1
 [6,] 2.43  2.6
 [7,] 1.74  1.8
 [8,] 1.03  1.1
 [9,] 1.51  1.6
[10,] 0.98  1.0
```


Principal component analysis
A simple example in R

```
> ## Get the old data back
> # If we keep just one component, we do the same
> # We cannot expect the original dataset
> plot(adjust.dataset1[,1],adjust.dataset1[,2],xlim=c(-1,4),ylim=c↩
    (-1,4))
> abline(h=0,v=0,lty=3)
```



Comparing the figure to the original data plot we can notice how, while the variation along the principle eigenvector has been kept, the variation along the other component (the other eigenvector that we left out) has gone.

Principal component analysis
Built-in PCA in R

```
> ## In R the library stats include prcomp()
> ## prcomp() performs a principal components analysis on the given ↩
    data matrix and returns the results as an object of class prcomp.
> df = data.frame(x=x, y=y)
>
> # prcomp() does the mean centering (option center=TRUE) and it also ↩
    scales the variables so that all have unit variance (scale=TRUE).
> # This is necessary if the data have different units.
> # In this case, the units are the same and we use (scale=FALSE):
> pca.eg<-prcomp(df,center=TRUE,scale=FALSE)
>
> pca.eg
Standard deviations:
[1] 1.13 0.22

Rotation:
    PC1   PC2
x -0.68  0.74
y -0.74 -0.68
```

The rotation attributes are equal to `cov.eig` above (except for the minus sign which is not relevant).

Principal component analysis
Built-in PCA in R

```
> summary(pca.eg)
Importance of components:
                          PC1   PC2
Standard deviation      1.388 0.272
Proportion of Variance 0.963 0.037
Cumulative Proportion  0.963 1.000
> par(mfrow=c(1,2))
> plot(pca.eg)
> biplot(pca.eg)
```