Dimensionality Reduction

Machine Learning II

Master in Business Analytics and Big Data

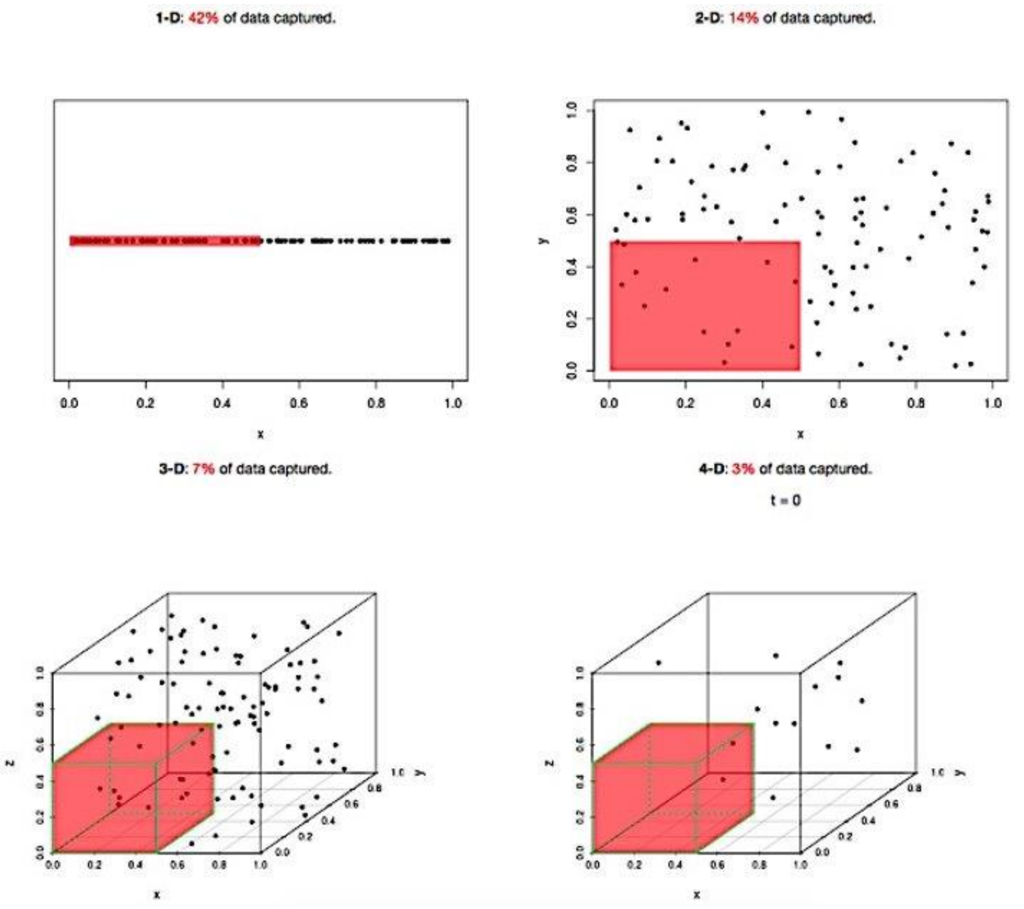
acastellanos@faculty.ie.edu

Reduce the dimension of the data set ➔ **smaller** data set

…to capture *most of the interesting behavior* of the data

feature selection
or
dimensionality reduction.

Why care about high dimensions?

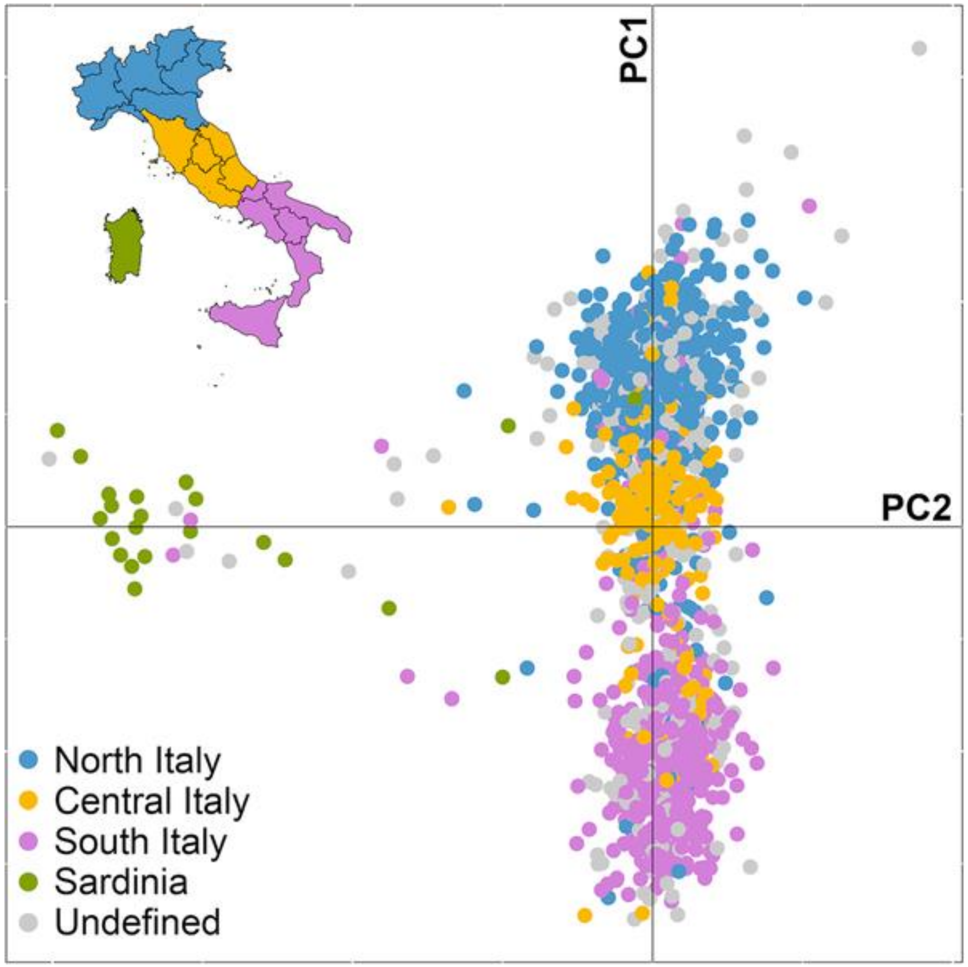


Why care about high dimensions?

- **Traditional** statistical techniques apply to problems where the **number of observations (*n*) is much greater than the number of features (*p*)**.
- In **recent years, technology changed the game**
  - Example: Understanding people’s online shopping patterns considers as *features* all the search terms and interactions with the web by every user. In this case:

$$\boxed{\begin{array}{c} n \approx 1000 \\ p >> n \end{array}}$$

Principal Component Analysis



Source:
Wikipedia Commons

PCA (2)

- PCA is a technique for **reducing the dimension** of a dataset
- This method seeks to find a **smaller number of representative variables** (**linear combinations of the predictors,** known as principal components PCs), which capture the most possible variance.
- **The first PC captures the most variability of all possible linear combinations.**
  - Then, subsequent PCs are derived such that these linear combinations capture the most remaining variability while also being uncorrelated with all previous PCs.

PCA (3)

- PCA works with the correlation between variables.
- **If the variables are *uncorrelated*, there is no point in computing PCA!**
- PCA is a **technique** (not an algorithm). It is absolutely *exploratory* or *preparatory*.
    - With the appropriate tools it becomes almost ridiculously easy to perform.
    - It's important to **understand the basics** behind to ensure that this method is applied properly.

Advantages & Caveats

- It creates components that are **uncorrelated**.
- PCA **does not consider the response variable** when summarizing variability. It is **unsupervised.**
  - For a supervised equivalent, use “LDA/QDA”.
- Sensitive to **scales** and (**skewed**) distributions
  - PCA focuses on **identifying the data structure** based on measurement scales rather than the important relationships within the data.

Scaling

```r
# load libraries
library(caret)
# load the dataset
data(iris)
# summarize data
summary(iris[,1:4])
```

```
  Sepal.Length    Sepal.Width     Petal.Length    Petal.Width   
 Min.   :4.300   Min.   :2.000   Min.   :1.000   Min.   :0.100  
 1st Qu.:5.100   1st Qu.:2.800   1st Qu.:1.600   1st Qu.:0.300  
 Median :5.800   Median :3.000   Median :4.350   Median :1.300  
 Mean   :5.843   Mean   :3.057   Mean   :3.758   Mean   :1.199  
 3rd Qu.:6.400   3rd Qu.:3.300   3rd Qu.:5.100   3rd Qu.:1.800  
 Max.   :7.900   Max.   :4.400   Max.   :6.900   Max.   :2.500  
```

Approach to standardization **(scaling and centering)** using CARET package

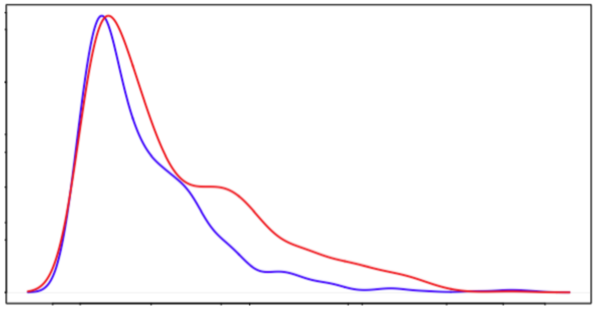
```r
# calculate the pre-process parameters from the dataset
preprocessParams <- preProcess(iris[,1:4], method=c("center", "scale"))
# transform the dataset using the parameters
transformed <- predict(preprocessParams, iris[,1:4])
# summarize the transformed dataset
summary(transformed)
```

```
  Sepal.Length       Sepal.Width       Petal.Length      Petal.Width     
 Min.   :-1.86378   Min.   :-2.4258   Min.   :-1.5623   Min.   :-1.4422  
 1st Qu.:-0.89767   1st Qu.:-0.5904   1st Qu.:-1.2225   1st Qu.:-1.1799  
 Median :-0.05233   Median :-0.1315   Median : 0.3354   Median : 0.1321  
 Mean   : 0.00000   Mean   : 0.0000   Mean   : 0.0000   Mean   : 0.0000  
 3rd Qu.: 0.67225   3rd Qu.: 0.5567   3rd Qu.: 0.7602   3rd Qu.: 0.7880  
 Max.   : 2.48370   Max.   : 3.0805   Max.   : 1.7799   Max.   : 1.7064  
```

Skewness

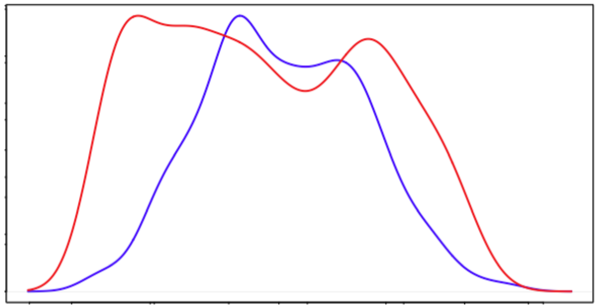
```
# load libraries
library(mlbench)
library(caret)
# load the dataset
data(PimaIndiansDiabetes)
# summarize pedigree and age
summary(PimaIndiansDiabetes[,7:8])
```

```
    pedigree             age       
 Min.   :0.0780   Min.   :21.00  
 1st Qu.:0.2437   1st Qu.:24.00  
 Median :0.3725   Median :29.00  
 Mean   :0.4719   Mean   :33.24  
 3rd Qu.:0.6262   3rd Qu.:41.00  
 Max.   :2.4200   Max.   :81.00  
```



Approach to fixing skewness, using CARET package

```
# calculate the pre-process parameters from the dataset
preprocessParams <- preProcess(PimaIndiansDiabetes[,7:8], method=c("BoxCox"))
# transform the dataset using the parameters
transformed <- predict(preprocessParams, PimaIndiansDiabetes[,7:8])
# summarize the transformed dataset (note pedigree and age)
summary(transformed)
```



```
    pedigree             age        
 Min.   :-2.5510   Min.   :0.8772  
 1st Qu.:-1.4116   1st Qu.:0.8815  
 Median :-0.9875   Median :0.8867  
 Mean   :-0.9599   Mean   :0.8874  
 3rd Qu.:-0.4680   3rd Qu.:0.8938  
 Max.   : 0.8838   Max.   :0.9019  
```

How?

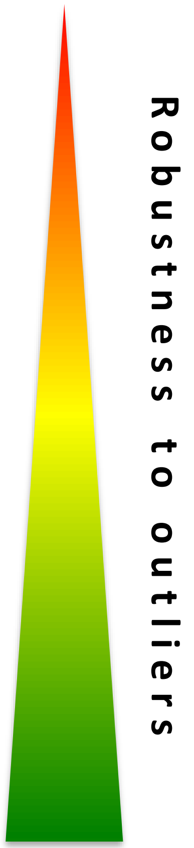
```
> Result <- prcomp( data )
```

Summary Stats

Measures of **LOCATION**	Measures of **SPREAD**	Measures of **ASSOCIATION**
Mean	Variance	Covariance
Median	Standard Deviation	Correlation
Percentiles	Interquartile Range	

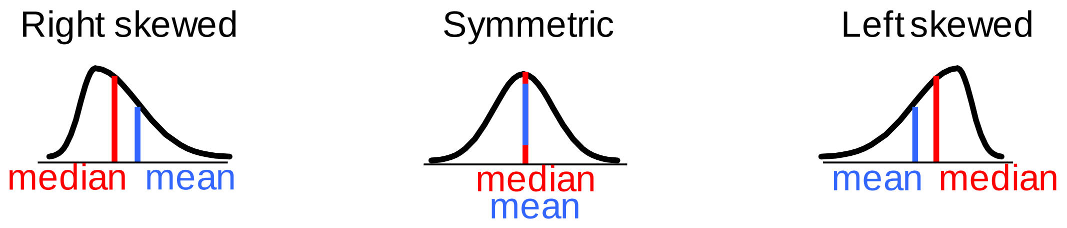
Mean, median and percentiles

- **Mean** is the average
- **Median** in the middle value in the sorted list of values.
- **Percentiles**: 25th, 50th and 75th percentiles respectively
  - *value below which a given percentage of observations in a group of observations fall*

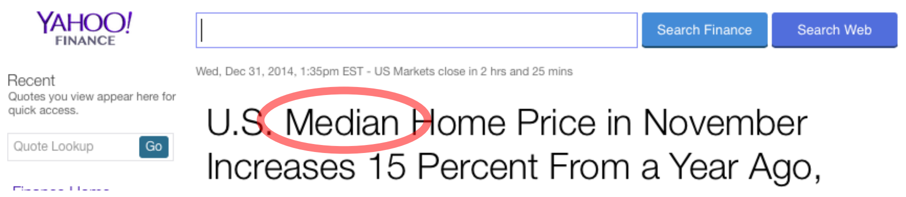


Mean or Median?

- **Median is preferred as, mean is unstable**, and whenever there's **skewness**, median is more stable.



Most typical:
prices, salaries, etc.



Recent
Quotes you view appear here for quick access.

Wed, Dec 31, 2014, 1:35pm EST - US Markets close in 2 hrs and 25 mins

U.S. Median Home Price in November Increases 15 Percent From a Year Ago,

Variance, Stdev and Interquartile range

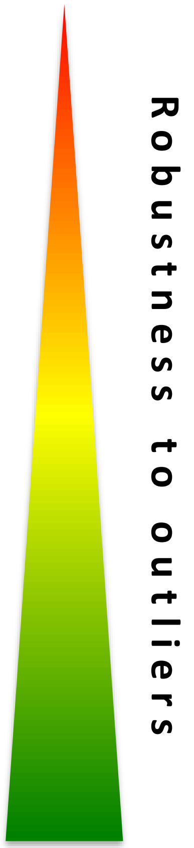
- **Variance** (*n* is sample size)

  Always positive, measures *if the points are very close to each other* and to the mean (small), or spread out (high).

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

- **Standard deviation**: Square root of the variance. It's **on the scale of the original data**. Represents the average distance from the mean.

- The **interquartile range** is the 3rd quartile minus the 1st quartile.



Covariance & Correlation

- **Covariance** between x and y

  Measures **how two variables change together**

  Positive: Y increases as X increases

  Negative: Y decreases as X increases

  WARN: Range is the same as the variables.

$$\frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{n-1}$$

- **Correlation** (coefficient of) is covariance divided by the product of the two standard deviations
  - Ranges between -1 and 1.

Theory

- **Covariance Matrix**

$$\Sigma = \begin{pmatrix} \mathrm{cov}(x,x) & \mathrm{cov}(x,y) & \cdots \\ \mathrm{cov}(y,x) & \mathrm{cov}(y,y) & \\ \vdots & & \ddots \end{pmatrix}$$

- Because of the symmetric nature of correlation (and covariance), **this matrix is equals to its transpose** (reflects elements along its diagonal).

Reminder on Matrix Operations

Inverse Matrix

$$A = \begin{bmatrix} a & b \\ c & d \end{bmatrix} \rightarrow A^{-1} = \frac{1}{ad - bc} \begin{bmatrix} d & -b \\ -c & a \end{bmatrix}$$

Transpose Matrix

$$\begin{pmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{pmatrix}^T = \begin{pmatrix} a_{11} & a_{21} & a_{31} \\ a_{12} & a_{22} & a_{32} \\ a_{13} & a_{23} & a_{33} \end{pmatrix}$$

Theory (2)

- So,… if we invoke the ***spectral decomposition theorem*** at this point, we find that
  - *For any symmetric $N \times N$ matrix A, there exists another orthogonal* matrix U,* such that:

$$B = \begin{pmatrix} \lambda_1 & & & \\ & \lambda_2 & & \\ & & \ddots & \\ & & & \lambda_N \end{pmatrix} = U^{-1} \cdot A \cdot U$$

  - λ_i are the **eigenvalues**
  - Columns of U, are the **eigenvectors** of A

*Orthogonal: $U^T = U^{-1}$; $U^T U = U U^T = 1$

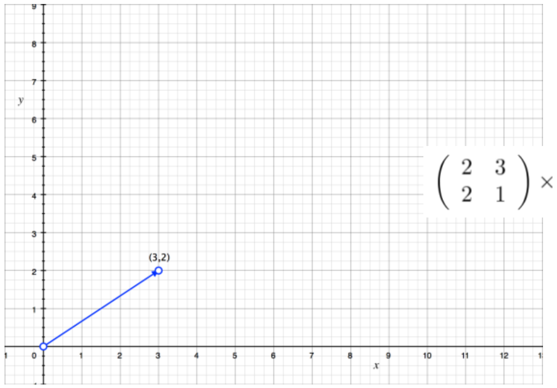
What is an eigenvector?

$$\begin{pmatrix} 2 & 3 \\ 2 & 1 \end{pmatrix} \times \begin{pmatrix} 1 \\ 3 \end{pmatrix} = \begin{pmatrix} 11 \\ 5 \end{pmatrix}$$

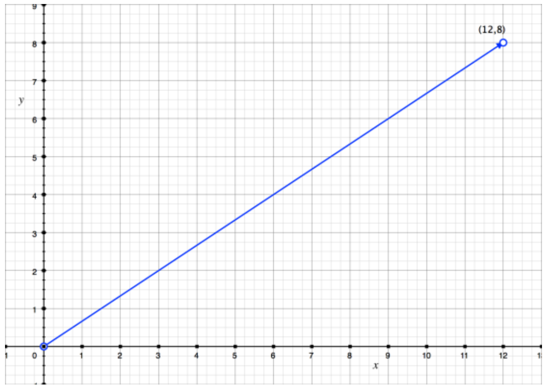
$$\begin{pmatrix} 2 & 3 \\ 2 & 1 \end{pmatrix} \times \begin{pmatrix} 3 \\ 2 \end{pmatrix} = \begin{pmatrix} 12 \\ 8 \end{pmatrix} = 4 \times \begin{pmatrix} 3 \\ 2 \end{pmatrix}$$

In the first line,
the resulting vector is not an integer multiple of the original vector,
whereas in the second line, the example is exactly **4** times the vector we began with.

Why is this?



$$\begin{pmatrix} 2 & 3 \\ 2 & 1 \end{pmatrix} \times \begin{pmatrix} 3 \\ 2 \end{pmatrix} = \begin{pmatrix} 12 \\ 8 \end{pmatrix} = 4 \times \begin{pmatrix} 3 \\ 2 \end{pmatrix}$$



$\begin{pmatrix} 2 & 3 \\ 2 & 1 \end{pmatrix}$ Is a **transformation matrix**. If you multiply this matrix on the left of a vector, the answer is another vector that is transformed from it’s original position.

Eigenvectors

Eigenvectors (of a given matrix) are vectors that suffer no transformation, just scaling, when multiplied.

$\begin{pmatrix} 3 \\ 2 \end{pmatrix}$ is an **eigenvector** of the *transformation matrix* $\begin{pmatrix} 2 & 3 \\ 2 & 1 \end{pmatrix}$

Properties

1) Eigenvectors can only be found for **square matrices**. Not every square matrix has eigenvectors. $n \times n$ matrices has (if any) n eigenvectors.
2) If vector is **scaled** by some amount before transforming it, the result is a multiple of it, too.

$$2 \times \begin{pmatrix} 3 \\ 2 \end{pmatrix} = \begin{pmatrix} 6 \\ 4 \end{pmatrix}$$

$$\begin{pmatrix} 2 & 3 \\ 2 & 1 \end{pmatrix} \times \begin{pmatrix} 6 \\ 4 \end{pmatrix} = \begin{pmatrix} 24 \\ 16 \end{pmatrix} = 4 \times \begin{pmatrix} 6 \\ 4 \end{pmatrix}$$

3) Eigenvectors are perpendicular (**orthogonal**). That means that we express data in terms of the eigenvectors instead of the x and y axes.
4) Due to their scaling property (2), we're interested only in eigenvectors of **length = 1**.

Eigenvalues

Remember the amount by which the original vector was scaled after multiplication by the square matrix?

Remember that amount was always the same?

That is the **eigenvalue**.

$$\begin{pmatrix} 2 & 3 \\ 2 & 1 \end{pmatrix} \times \begin{pmatrix} 3 \\ 2 \end{pmatrix} = \begin{pmatrix} 12 \\ 8 \end{pmatrix} = 4 \times \begin{pmatrix} 3 \\ 2 \end{pmatrix}$$

No matter what multiple of the eigenvector we took before we multiplied it by the square matrix, we would always get 4 times the scaled vector as our result

Eigenvalues

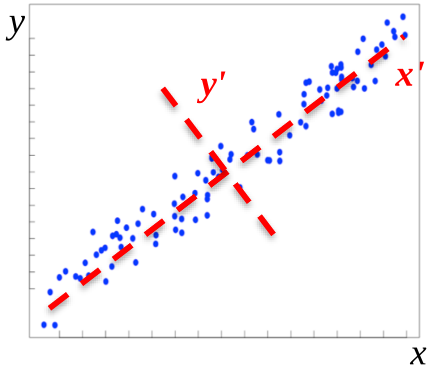
?

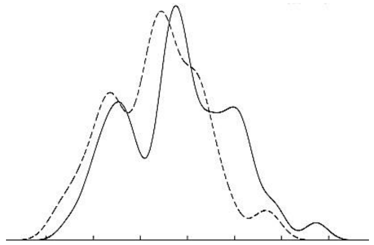
Why is this important?

- We can change the variables in our observations from a square matrix A into a diagonal matrix, B, which contains the same information.

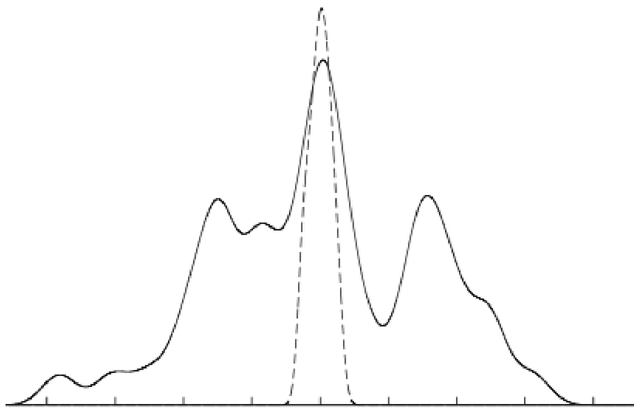
- The purpose is to **find the directions** in a new coordinate system, **where is more suitable to describe the data** than was in the original coordinate system.

Example





Distribution of the two variables along x and y.



Distribution of the two variables along x' and y'.

Original pictures taken from “Data analysis with open source tools”

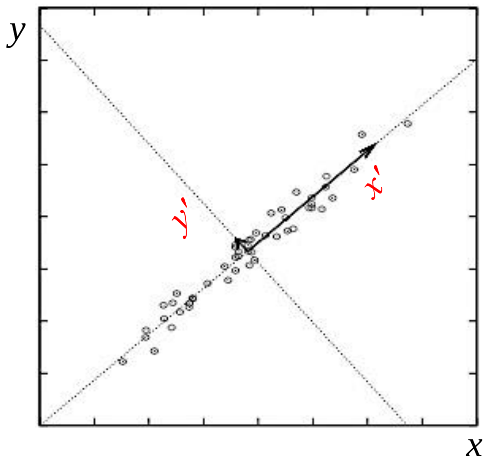
Interpretation

- ***Eigenvalues*** point along the directions of greatest variance.
    - The eigenvalue (corresponding to each eigenvector) is a measure of the **width of the distribution** along this direction.
    - Variables measured along the principal directions are **uncorrelated** with each other (*take a look to the correlation matrix*).

- If the data points are distributed as a globular cloud, ***eigenvectors*** will give us the directions of the principal axes of the cloud of data points and the ***eigenvalues*** will give us the length of the cloud along each of these directions

Why is this useful?

- The PCA uses the information contained in the mutual correlations between variables to **identify** those that are **redundant**.

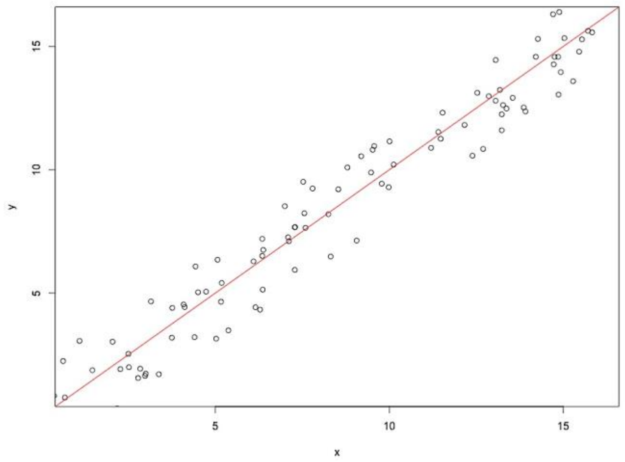


- The **irrelevant** variables are those corresponding to ***small eigenvalues***

Example

- Let's produce some sample data with a strong correlation.
- 2 variables (dimensions), X and Y.
- Y is simply, X ± noise between 0 and 2.

```r
x <- runif(100, 0, 16)
for(i in seq(along=x)) {
  y[i] = x[i] + (randSign()*runif(1, 0, 2))
}
data <- data.frame(x,y)
```



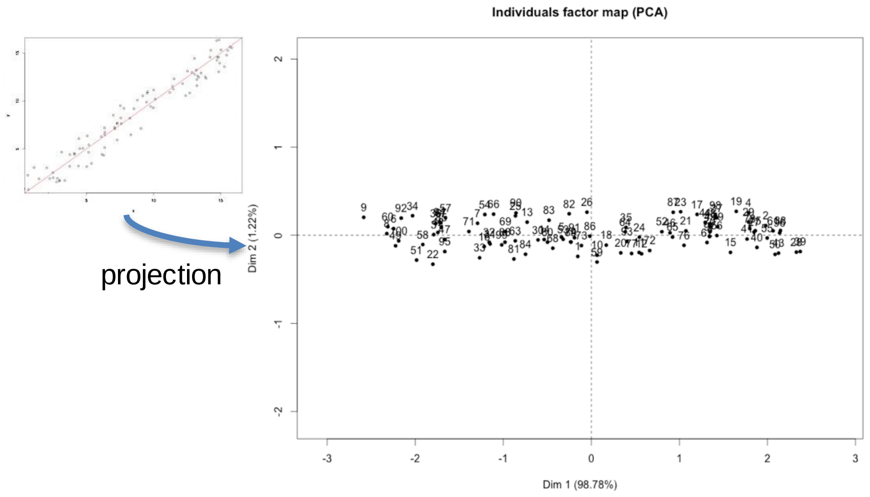
PCA on sample data

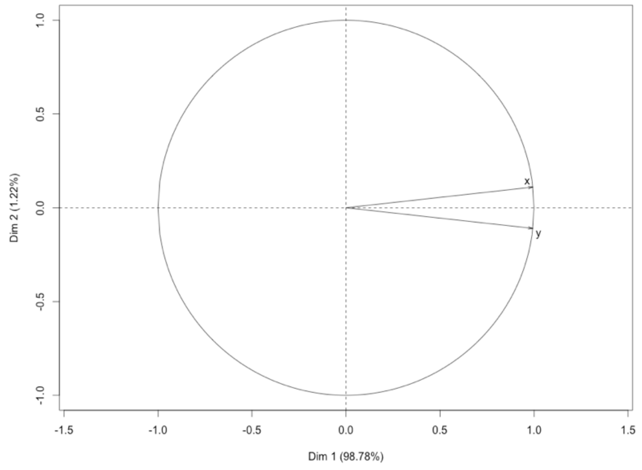
```
> pca <- prcomp(data)
> print(pca)
Standard deviations:
[1] 6.9073683 0.7656859

Rotation:
        PC1        PC2
x 0.6959660 -0.7180748
y 0.7180748  0.6959660

> summary(pca)
Importance of components:
                          PC1     PC2
Standard deviation     6.9074 0.76569
Proportion of Variance 0.9879 0.01214
Cumulative Proportion  0.9879 1.00000

> pca$sd^2
[1] 47.7117365  0.5862748
```

- **PC1 stdev is much greater than PC2**, which means that **PC1 explains most of the variance**.
- In the summary, you can see how **PC1 dimension explains 98,79%** of the variance.
- `Rotation` **is the matrix of eigenvectors**. PC1 dimension presents almost the same value for X and Y, which means that they're correlated.

- And finally, just for curiosity, what you can check is the eigenvalues themselves, where λ_1 is a lot bigger than λ_2, explaining its importance.

And PCA, graphically





The **projection** of the variables onto the 2 principal components result in the expected stretched ellipsoid.

- X and Y, are parallel to the first PC → **they are important variables.**
- Both vectors are almost identical (parallel) → **they're strongly correlated**.
- Most important take away is that X and Y are redundant.

A more complex example

Wine quality dataset: http://archive.ics.uci.edu/ml/datasets/Wine+Quality

```
1 - fixed acidity
2 - volatile acidity
3 - citric acid
4 - residual sugar
5 - chlorides
6 - free sulfur dioxide
7 - total sulfur dioxide
8 - density
9 - pH
10 - sulphates
11 - alcohol
12 - quality (score between 0 and 10)
```

```r
wine <- read.csv( "winequality-white.csv",
sep=';', header=TRUE )
pc <- prcomp( wine )
plot( pc )
```

Same example, explained in: http://blog.haunschmid.name/dimensionality-reduction-1-understanding-pca-and-ica-using-r/

Exercise

Do the values in dataset range within the same scales?

What variables are more representative?

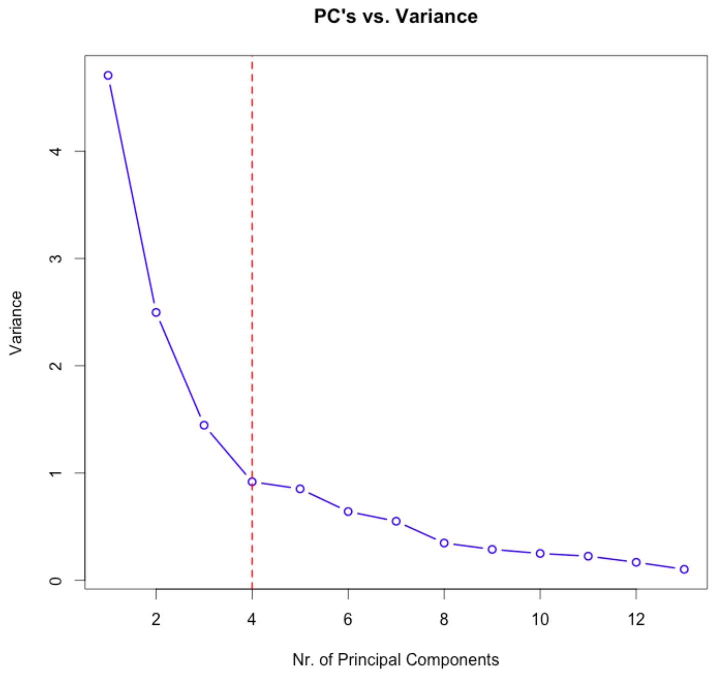
How many variables do we need to explain, at least, 75% of the variance?

Ranges

```
> head(wine)
  V1    V2   V3   V4   V5  V6   V7   V8   V9  V10  V11  V12  V13  V14
1  1 14.23 1.71 2.43 15.6 127 2.80 3.06 0.28 2.29 5.64 1.04 3.92 1065
2  1 13.20 1.78 2.14 11.2 100 2.65 2.76 0.26 1.28 4.38 1.05 3.40 1050
3  1 13.16 2.36 2.67 18.6 101 2.80 3.24 0.30 2.81 5.68 1.03 3.17 1185
4  1 14.37 1.95 2.50 16.8 113 3.85 3.49 0.24 2.18 7.80 0.86 3.45 1480
5  1 13.24 2.59 2.87 21.0 118 2.80 2.69 0.39 1.82 4.32 1.04 2.93  735
6 1 14.20 1.76 2.45 15.2 112 3.27 3.39 0.34 1.97 6.75 1.05 2.85 1450

> std_wine <- as.data.frame(scale(wine[2:14]))
   V2    V3    V4    V5   V6   V7   V8    V9   V10   V11   V12  V13   V14
1 1.51 -0.56  0.23 -1.17 1.91 0.81 1.03 -0.66  1.22  0.25  0.36 1.84  1.01
2 0.25 -0.50 -0.83 -2.48 0.02 0.57 0.73 -0.82 -0.54 -0.29  0.40 1.11  0.96
3 0.20  0.02  1.11 -0.27 0.09 0.81 1.21 -0.50  2.13  0.27  0.32 0.79  1.39
4 1.69 -0.35  0.49 -0.81 0.93 2.48 1.46 -0.98  1.03  1.18 -0.43 1.18  2.33
5 0.29  0.23  1.84  0.45 1.28 0.81 0.66  0.23  0.40 -0.32  0.36 0.45 -0.04
6 1.48 -0.52  0.30 -1.29 0.86 1.56 1.36 -0.18  0.66  0.73  0.40 0.34  2.23
```

How many principal components?

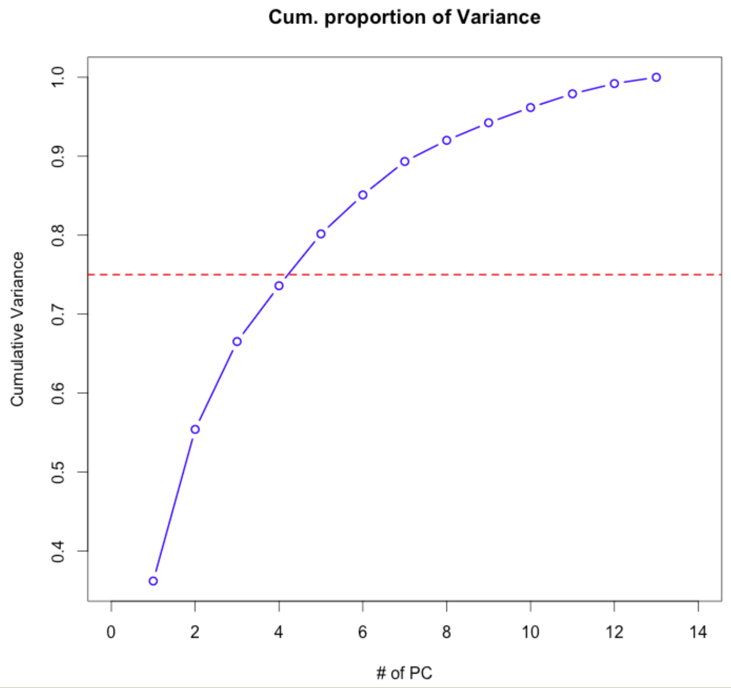
- **Criteria #1: Slope change in variance**



```
plot(c(1:13), (pca$sdev)^2, type="b", col="blue", lwd=2)
abline(v=4, lwd=2, lty=2, col="red")
```

How many principal components?

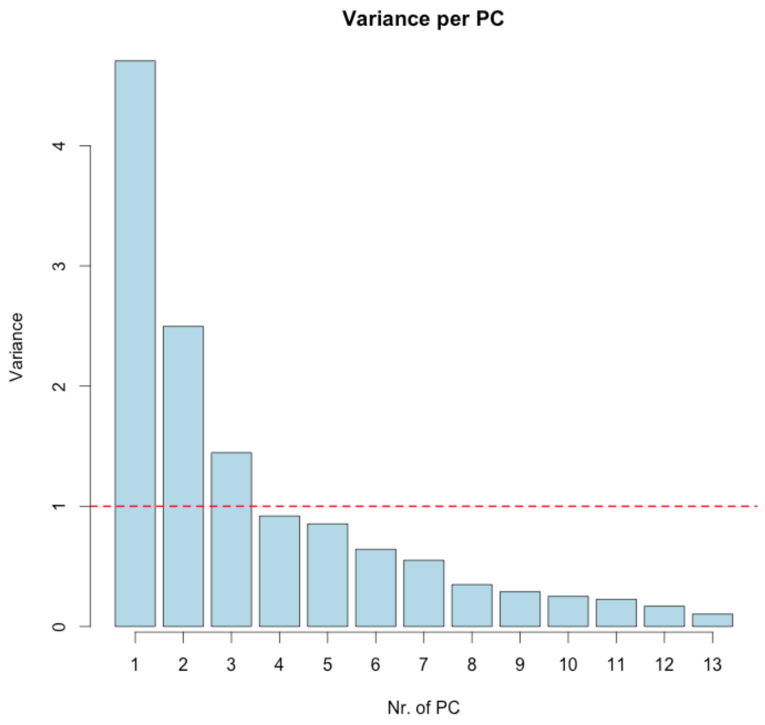
- **Criteria #2: Cumulative variance**



```
plot(cumsum((pca$sdev^2)/sum(pca$sdev^2)), type="b", lwd=2)
abline(h=0.75, col="red", lty=2, lwd=2)
```

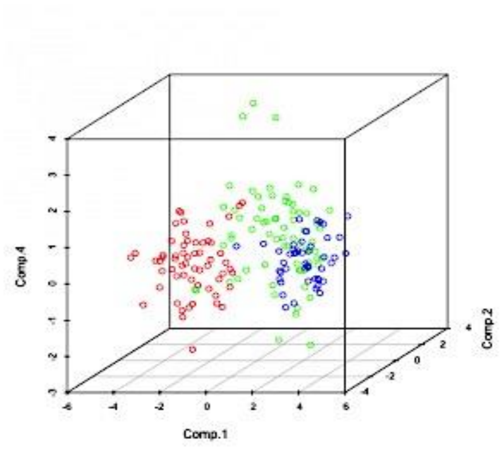
How many principal components?

- **Criteria #3: Variance > 1**

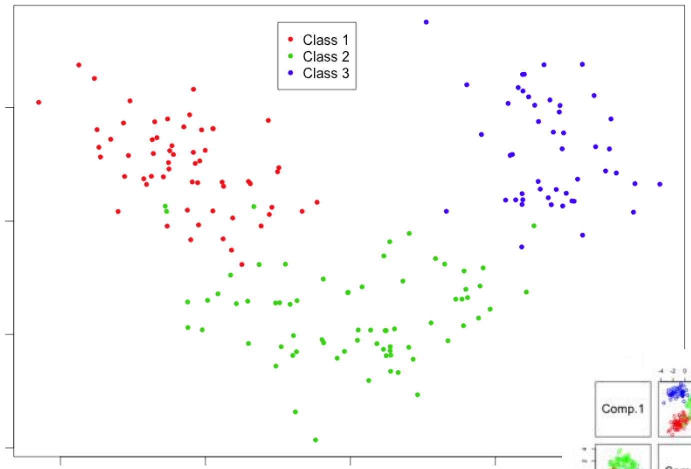


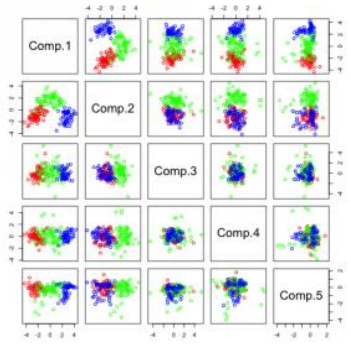
```
bp <- barplot(pca$sdev^2)
abline(h=1, col="red", lwd=2, lty=2)
```

PC1 vs. PC2 vs. PC4



PC1 vs. PC2

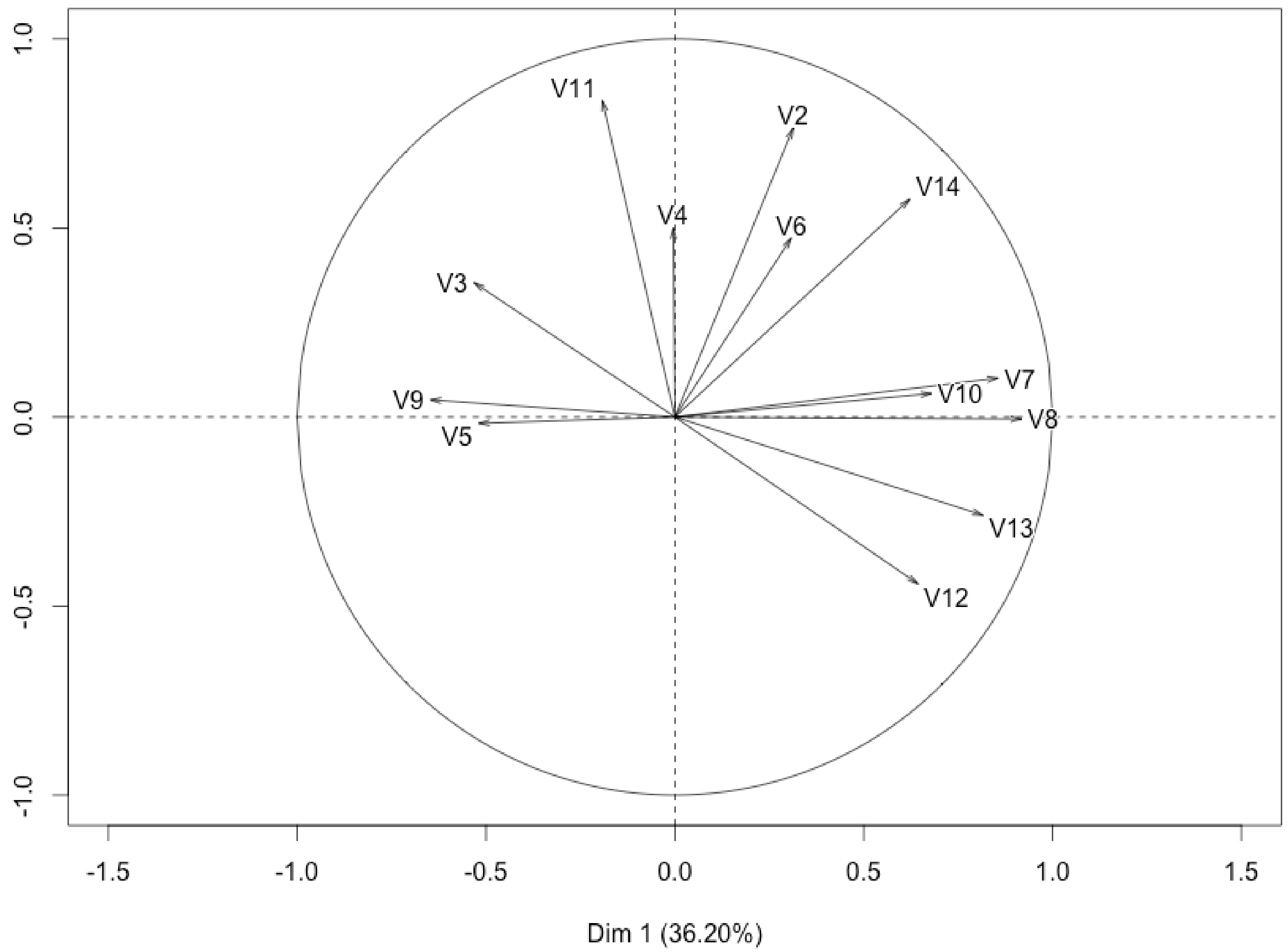




Note: Although PCA is very useful in finding the components with the highest variation, it does not always mean that the component are useful for separation in that order.

```
library(scatterplot3d)
pairs(pc$scores[,1:5], col=rainbow(3)[wine[,1]], asp=1)
scatterplot3d(pc$scores[,c(1,2,4)], color=rainbow(3)[wine[,1]])
```

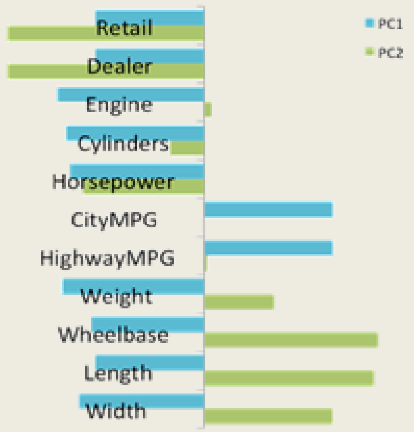
Variables factor map (PCA)

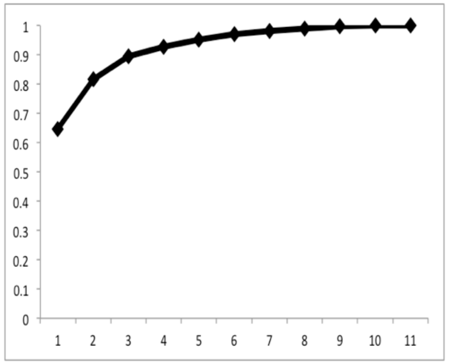


Another example: cars

```
               Sports SUV Wagon Minivan Pickup AWD RWD Retail Dealer Engine Cylinders Horsepower CityMPG HighwayMPG Weight Wheelbase Length Width
Acura 3.5 RL   0      0   0     0       0      0   0   43755  39014  3.5    6         225        18      24         3880   115       197    72
Acura MDX      0      1   0     0       0      1   0   36945  33337  3.5    6         265        17      23         4451   106       189    77
Acura NSX S    1      0   0     0       0      0   1   89765  79978  3.2    6         290        17      24         3153   100       174    71
Acura RSX      0      0   0     0       0      0   0   23820  21761  2.0    4         200        24      31         2778   101       172    68
Acura TL       0      0   0     0       0      0   0   33195  30299  3.2    6         270        20      28         3575   108       186    72
```

```
> pca = prcomp(cars[,8:18], scale.=TRUE)
> round(pca$rotation[,1:2],2)

             PC1   PC2
Retail      -0.26 -0.47
Dealer      -0.26 -0.47
Engine      -0.35  0.02
Cylinders   -0.33 -0.08
Horsepower  -0.32 -0.29
CityMPG      0.31  0.00
HighwayMPG   0.31  0.01
Weight      -0.34  0.17
Wheelbase   -0.27  0.42
Length      -0.26  0.41
Width       -0.30  0.31
```

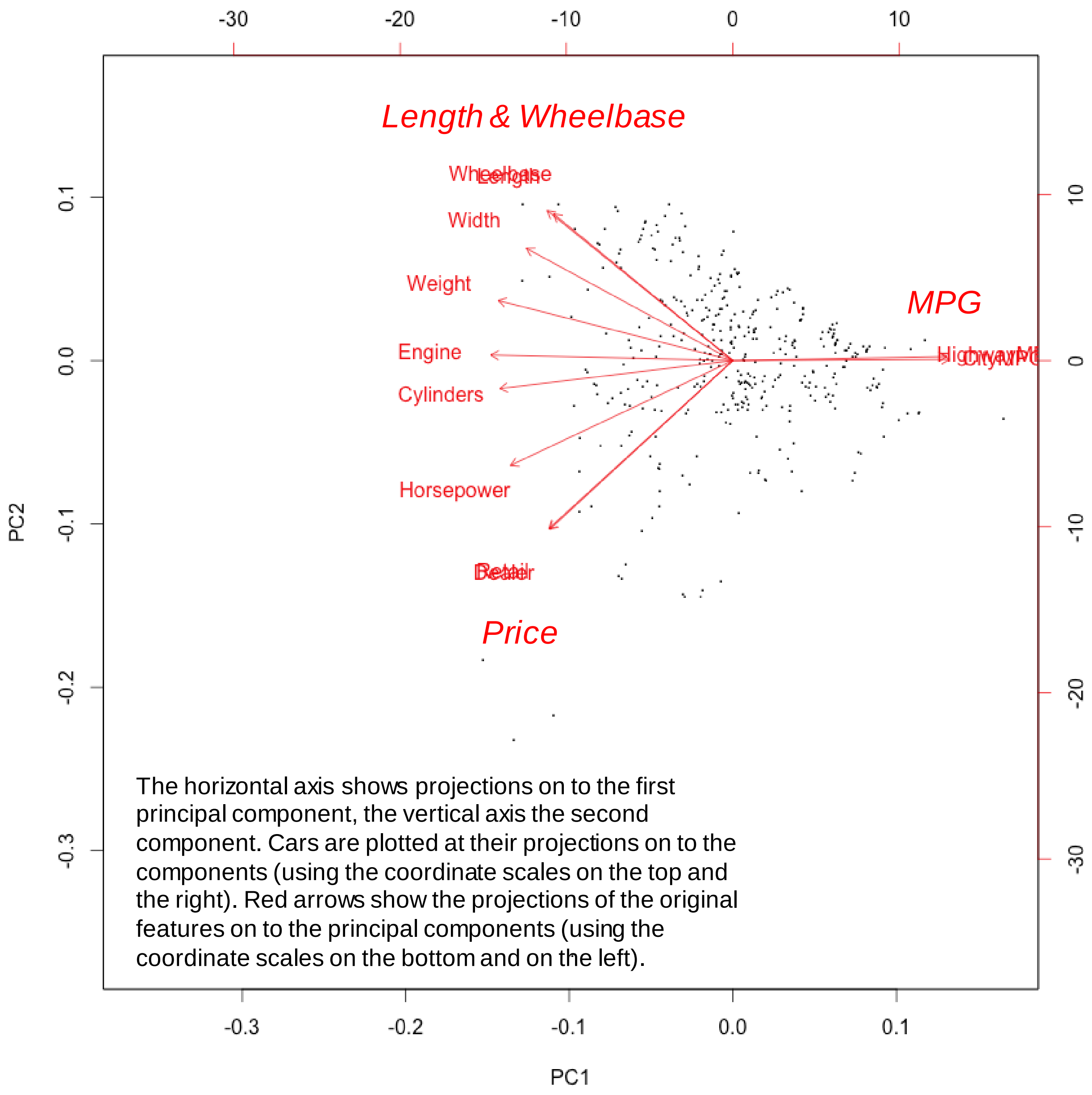




Cumulative Proportion

Recipes for interpretation

- **Coordinates**
  - Use the bi-plot to summarize what each component means.
- **Correlations**
  - For many datasets, the data clusters into groups of highly correlated attributes
- **Clusters**
  - Clusters indicate a preference for particular combinations of attribute values. Summarize each cluster by its prototypical member.
- **Funnels**
  - Funnels are wide at one end and narrow at the other. They happen when one dimension affects the variance of another, orthogonal dimension.
- **Voids**
  - Voids are areas inside the range of the data which are unusually unpopulated.



The horizontal axis shows projections on to the first principal component, the vertical axis the second component. Cars are plotted at their projections on to the components (using the coordinate scales on the top and the right). Red arrows show the projections of the original features on to the principal components (using the coordinate scales on the bottom and on the left).