PCA Algorithm

LAEB

30-10-2019

Contents

1 Introduction

Let n be the number of observations in the sample data. Let m be the number of initial features/attributes used to describe the data. Let $\vec{x}_1, ..., \vec{x}_n$ be the sample data of vectors in $\mathbb{R}^m$. Let X be the $m \times n$ matrix where each feature/attribute corresponds to a row and each observation corresponds to a column of X

$$X = \left(\vec{x}_1 ... \vec{x}_n\right)$$

1.1 Pre processing of the data

Centering the data

Suppose the data has been centered i.e. $\sum\limits_i^n \vec{x}_i = \vec{0}$. If the data is not centered, we center it by replacing x_i by $x_i - \vec{\mu}$ where $\vec{\mu}$ in $\mathbb{R}^m$ is defined by

$$\vec{\mu} = \frac{1}{n} \sum_{i=1}^{n} \vec{x}_i$$

Rescaling the data

Rescaling the data is not necessary but should be done if attributes/features are expressed in very different units. If the data is expressed in different units, we compute

$$s^j = \frac{1}{n-1} \sum_{i=1}^{n} (x_i^j - \mu^j)^2$$

and we replace the original x_i^j by x_i^j / s^j

If we want to center and rescale the data, the original x_i^j should be replaced by

$$\frac{x_i^j - \mu^j}{s^j}$$

1.2 Goal

We'd like to find the q unit vectors $\vec{u}_1, .., \vec{u}_q \in \mathbb{R}^m$ with $q < \min(n, m)$ that transform the sample data vectors as

$$\vec{y}_i = U^T \vec{x}_i = \begin{pmatrix} \vec{u}_1^T \vec{x}_i \\ . \\ . \\ \vec{u}_q^T \vec{x}_i \end{pmatrix} = \begin{pmatrix} y_i^1 \\ . \\ . \\ y_i^q \end{pmatrix}$$

where U is the $m \times q$ matrix

$$U = (\vec{u}_1 ... \vec{u}_q)$$

such that

- Decorrelation of the new coordinates

the new coordinates are decorrelated i.e. $\mathrm{cov}(y^i, y^j) = 0$ for all $i, j \in [1, q]$ and $i \neq j$.

- Maximization of the variance of the new coordinates

the variance of the sample data is maximized after being projected onto the new axes vectors.

1.3 Mathematical Derivation

Let us assume first $q = 1$ i.e. the m components of the vectors x_i will be reduced to a single output component

$$y_i^1 = \vec{u}_1^T \vec{x}_i$$

where $\vec{u}_1$ is such that it is solution to

$$\max_{\vec{u}_1} \left(\frac{1}{2} \sum_{i=1}^{n} (y_i^1)^2 \right) = \max_{\vec{u}_1} \left(\frac{1}{2} ||\vec{u}_1^T X||^2 \right)$$

subject to

$$\vec{u}_1^T \vec{u}_1 = 1$$

The Lagrangian of the problem is

$$L(\vec{u}_1, \lambda_1) = \frac{1}{2} \vec{u}_1^T X X^T \vec{u}_1 - \frac{\lambda_1}{2} (\vec{u}_1^T \vec{u}_1 - 1)$$

The vector $\vec{u}_1$ is solution of

$$\frac{\partial L}{\partial \vec{u}_1} = 0$$

which is equivalent to

$$X X^T \vec{u}_1 = \lambda_1 \vec{u}_1$$

i.e. $\vec{u}_1$ is an eigenvector of the matrix XX^T. Since XX^T is symmetric positive definite, it is diagonalized with positive eigenvalues. Since we want $\frac{1}{2}||\vec{u}_1^T X||^2 = \frac{1}{2}\vec{u}_1^T X X^T \vec{u}_1 = \frac{1}{2}\lambda_1 \vec{u}_1^T \vec{u}_1 = \frac{1}{2}\lambda_1$ to be maximal for $\vec{u}_1$, we take $\vec{u}_1$ to be the eigenvector of XX^T with the highest eigenvalue.

Now, we look for $\vec{u}_2$ which is such that the new coordinates y^1 and y^2 are decorrelated i.e.

$$\mathrm{Cov}(y^1, y^2) = 0$$

Since one can re-write the covarienace between y^1 and y^2 as

$$\sum_{i=1}^{n} y_i^1 y_i^2 = \sum_{i=1}^{n} (\vec{u}_1^T \vec{x}_i)(\vec{u}_2^T \vec{x}_i) = \sum_{i=1}^{n} (\vec{u}_1^T \vec{x}_i)(\vec{u}_2^T \vec{x}_i)^T = \vec{u}_1^T X X^T \vec{u}_2 = \vec{u}_2^T X X^T \vec{u}_1 = \lambda_1 \vec{u}_2^T \vec{u}_1$$

we are looking for the vector $\vec{u}_2$ that is solution to

$$\max_{\vec{u}_2} \left(\frac{1}{2} \sum_{i=1}^{n} (y_i^2)^2 \right) = \max_{\vec{u}_2} \left(\frac{1}{2} ||\vec{u}_2^T X||^2 \right)$$

subject to

$$\vec{u}_2^T \vec{u}_2 = 1, \qquad \vec{u}_1^T \vec{u}_2 = 0$$

The Lagrangian is now

$$L(\vec{u}_2, \lambda_2, \delta_2) = \frac{1}{2} \vec{u}_2^T X X^T \vec{u}_2 - \frac{\lambda_2}{2} (\vec{u}_2^T \vec{u}_2 - 1) - \delta_2 \vec{u}_1^T \vec{u}_2$$

By taking the partial derivative with respect to $\vec{u}_2$ we obtain

$$X X^T \vec{u}_2 - \lambda_2 \vec{u}_2 - \delta_2 \vec{u}_1 = 0$$

By multiplying by $\vec{u}_1^T$, one get

$$\vec{u}_1^T X X^T \vec{u}_2 - \delta_2 = 0$$

and since $\vec{u}_1^T X X^T \vec{u}_2 = 0$ from requiring y^1, y^2 to be decorrelated, this means that $\delta_2 = 0$ and thus

$$X X^T \vec{u}_2 = \lambda_2 \vec{u}_2$$

i.e. $\vec{u}_2$ is eigenvector of XX^T with eigenvalue $lambda_2$ with λ_2 being the largest from the remaining eigenvalue of XX^T.

By induction, PCA is the solution to the following optimization

$$\max_{\vec{u}_1,\ldots,\vec{u}_q}\left(\frac{1}{2}\sum_i ||\vec{u}_i^T X||^2\right)$$

subject to

$$\vec{u}_i^T\vec{u}_i = 1, \qquad\qquad \vec{u}_i^T\vec{u}_j = 0, \qquad i \neq j = 1, ..., q$$

2 PCA Implementation in R

There are two ways of performing a PCA

- Spectral decompostion which examines the covariance between features

- Singular Value decomposition which examines covariance between individuals

There are several functions from different packages for performing a PCA in R

- The functions prcomp() and princomp() from the built-in R stats package
- PCA() from FactoMineR package.
- dudi.pca() from ade4 package.

In the following, we will be using the prcomp() and princomp() functions from the built-in R stats package. The function princomp() uses the spectral decomposition approach whereas the function prcomp() uses the singular value decomposition approach. Since the singular value decomposition has a better numerical accuracy compared to the spectral decomposition, it will be the preferred approach.

The package factoextra will be loaded for visualizing the PCA results. The latter requires the ggplot2 package to be loaded too.

```
## Load library
library(ggplot2) # for being able to load factoextra
library(factoextra) # for visualizing the PCA's reuslts
```

```
## Welcome! Related Books: `Practical Guide To Cluster Analysis in R` at https://goo.gl/13EFCZ
```

```
library(dplyr)
```

```
##
## Attaching package: 'dplyr'

## The following objects are masked from 'package:stats':
##
##     filter, lag

## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
library(kernlab) # for kernel pca
```

```
##
## Attaching package: 'kernlab'

## The following object is masked from 'package:ggplot2':
##
##     alpha
```

The dataset to be used for PCA is imported. Detailed documentation about the dataset to be used can be found here: http://jse.amstat.org/datasets/04cars.txt It contains data about 428 cars or trucks. The dataset has one row per car i.e. a total of 428 rows and 19 variables. The variables specified include the vehicle name (variables sports car, sport utility vehicle, wagon, minivan, pickup, all-wheel drive, rear wheel drive), the retail price, dealer cost, engine size, number of cylinders, horsepower, city miles per gallon, highway miles per gallon, weight, wheelbase, length and width.

```
## Import dataset
cars04  <- readRDS(file =
'~/birwe_data/Data/playground/prepared-zone/methods-and-libraries/ml-reading-course/cars04.RData')
```

```r
## Look at dataset
str(cars04)
```

```
## 'data.frame':    387 obs. of  18 variables:
##  $ Sports    : int  0 0 0 1 0 0 0 0 0 0 ...
##  $ SUV       : int  0 0 1 0 0 0 0 0 0 0 ...
##  $ Wagon     : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ Minivan   : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ Pickup    : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ AWD       : int  0 0 1 0 0 0 0 0 0 0 ...
##  $ RWD       : int  0 0 0 1 0 0 0 0 0 0 ...
##  $ Retail    : int  43755 46100 36945 89765 23820 33195 26990 25940 35940 42490 ...
##  $ Dealer    : int  39014 41100 33337 79978 21761 30299 24647 23508 32506 38325 ...
##  $ Engine    : num  3.5 3.5 3.5 3.2 2 3.2 2.4 1.8 1.8 3 ...
##  $ Cylinders : int  6 6 6 6 4 6 4 4 4 6 ...
##  $ Horsepower: int  225 225 265 290 200 270 200 170 170 220 ...
##  $ CityMPG   : int  18 18 17 17 24 20 22 22 23 20 ...
##  $ HighwayMPG: int  24 24 23 24 31 28 29 31 30 27 ...
##  $ Weight    : int  3880 3893 4451 3153 2778 3575 3230 3252 3638 3814 ...
##  $ Wheelbase : int  115 115 106 100 101 108 105 104 105 105 ...
##  $ Length    : int  197 197 189 174 172 186 183 179 180 180 ...
##  $ Width     : int  72 72 77 71 68 72 69 70 70 70 ...
```

```r
summary(cars04)
```

```
##      Sports            SUV             Wagon             Minivan
##  Min.   :0.0000   Min.   :0.0000   Min.   :0.00000   Min.   :0.00000
##  1st Qu.:0.0000   1st Qu.:0.0000   1st Qu.:0.00000   1st Qu.:0.00000
##  Median :0.0000   Median :0.0000   Median :0.00000   Median :0.00000
##  Mean   :0.1163   Mean   :0.1525   Mean   :0.07235   Mean   :0.05426
##  3rd Qu.:0.0000   3rd Qu.:0.0000   3rd Qu.:0.00000   3rd Qu.:0.00000
##  Max.   :1.0000   Max.   :1.0000   Max.   :1.00000   Max.   :1.00000
##      Pickup       AWD              RWD              Retail
##  Min.   :0   Min.   :0.0000   Min.   :0.0000   Min.   : 10280
##  1st Qu.:0   1st Qu.:0.0000   1st Qu.:0.0000   1st Qu.: 20997
##  Median :0   Median :0.0000   Median :0.0000   Median : 28495
##  Mean   :0   Mean   :0.2016   Mean   :0.2429   Mean   : 33231
##  3rd Qu.:0   3rd Qu.:0.0000   3rd Qu.:0.0000   3rd Qu.: 39552
##  Max.   :0   Max.   :1.0000   Max.   :1.0000   Max.   :192465
##      Dealer           Engine         Cylinders        Horsepower
##  Min.   :  9875   Min.   :1.400   Min.   : 3.000   Min.   : 73.0
##  1st Qu.: 19575   1st Qu.:2.300   1st Qu.: 4.000   1st Qu.:165.0
##  Median : 26155   Median :3.000   Median : 6.000   Median :210.0
##  Mean   : 30441   Mean   :3.127   Mean   : 5.757   Mean   :214.4
##  3rd Qu.: 36124   3rd Qu.:3.800   3rd Qu.: 6.000   3rd Qu.:250.0
##  Max.   :173560   Max.   :6.000   Max.   :12.000   Max.   :493.0
##     CityMPG        HighwayMPG        Weight        Wheelbase
##  Min.   :10.00   Min.   :12.00   Min.   :1850   Min.   : 89.0
##  1st Qu.:18.00   1st Qu.:24.00   1st Qu.:3107   1st Qu.:103.0
##  Median :19.00   Median :27.00   Median :3469   Median :107.0
##  Mean   :20.31   Mean   :27.26   Mean   :3532   Mean   :107.2
##  3rd Qu.:21.50   3rd Qu.:30.00   3rd Qu.:3922   3rd Qu.:112.0
##  Max.   :60.00   Max.   :66.00   Max.   :6400   Max.   :130.0
##      Length        Width
##  Min.   :143   Min.   :64.00
```

```
##   1st Qu.:177   1st Qu.:69.00  
##   Median :186   Median :71.00  
##   Mean   :185   Mean   :71.28  
##   3rd Qu.:193   3rd Qu.:73.00  
##   Max.   :221   Max.   :81.00
```

```r
head(cars04)
```

```
##                           Sports SUV Wagon Minivan Pickup AWD RWD Retail
## Acura 3.5 RL                   0   0     0       0      0   0   0  43755
## Acura 3.5 RL Navigation        0   0     0       0      0   0   0  46100
## Acura MDX                      0   1     0       0      0   1   0  36945
## Acura NSX S                    1   0     0       0      0   0   1  89765
## Acura RSX                      0   0     0       0      0   0   0  23820
## Acura TL                       0   0     0       0      0   0   0  33195
##                           Dealer Engine Cylinders Horsepower CityMPG
## Acura 3.5 RL               39014    3.5         6        225      18
## Acura 3.5 RL Navigation    41100    3.5         6        225      18
## Acura MDX                  33337    3.5         6        265      17
## Acura NSX S                79978    3.2         6        290      17
## Acura RSX                  21761    2.0         4        200      24
## Acura TL                   30299    3.2         6        270      20
##                           HighwayMPG Weight Wheelbase Length Width
## Acura 3.5 RL                      24   3880       115    197    72
## Acura 3.5 RL Navigation           24   3893       115    197    72
## Acura MDX                         23   4451       106    189    77
## Acura NSX S                       24   3153       100    174    71
## Acura RSX                         31   2778       101    172    68
## Acura TL                          28   3575       108    186    72
```

A PCA will be performed using the function prcomp(). The prcomp() function can be used by default as follow: prcomp(x, center = TRUE, scale. = FALSE, retx = TRUE,…) where

- x is a numeric or complex matrix/data frame containing only the initial features rotate
- center is a logical value indicating whether the initial variables/features should be shifted to be zero centered
- scale. is a logical value indicating whether the initial variables should be scaled to have unit variance before the analysis is performed. Rescaling, i.e. setting scale. = TRUE, should be done if the initial variables have different units.
- retx is a logical value indicating whether the rotated variables should be returned

By looking at the PCA object generated, one can understand what are the values returned by the PCA function

- sdev: standard deviation of the principal components i.e. square roots of the covariance matrix
- rotation: loading matrix i.e. matrix whose columns are the eigenvectors of the covariance matrix
- center, scale: the centering and scaling used if center and scale. were set to TRUE in the arguments of the function
- x: the matrix of the rotated data (centered and scaled data if requested multiplied by the rotation matrix)

```r
## Perform a pca using prcomp
cars04.pca = prcomp(cars04[,8:18], center = TRUE, scale.=TRUE, retx = TRUE)
## PCA object
str(cars04.pca)
```

```
## List of 5
##  $ sdev    : num [1:11] 2.665 1.373 0.922 0.598 0.525 ...
```

```
##  $ rotation: num [1:11, 1:11] -0.264 -0.262 -0.347 -0.334 -0.319 ...
##   ..- attr(*, "dimnames")=List of 2
##   .. ..$ : chr [1:11] "Retail" "Dealer" "Engine" "Cylinders" ...
##   .. ..$ : chr [1:11] "PC1" "PC2" "PC3" "PC4" ...
##  $ center  : Named num [1:11] 33231.18 30440.65 3.13 5.76 214.44 ...
##   ..- attr(*, "names")= chr [1:11] "Retail" "Dealer" "Engine" "Cylinders" ...
##  $ scale   : Named num [1:11] 19724.63 17901.18 1.01 1.49 70.26 ...
##   ..- attr(*, "names")= chr [1:11] "Retail" "Dealer" "Engine" "Cylinders" ...
##  $ x       : num [1:387, 1:11] -1.57 -1.63 -1.9 -1.59 2.65 ...
##   ..- attr(*, "dimnames")=List of 2
##   .. ..$ : chr [1:387] "Acura 3.5 RL" "Acura 3.5 RL Navigation" "Acura MDX" "Acura NSX S" ...
##   .. ..$ : chr [1:11] "PC1" "PC2" "PC3" "PC4" ...
##  - attr(*, "class")= chr "prcomp"
```

2.1 Variances of the principal components

The variances of the principal components are the eigenvalues of the covariance matrix i.e. the squared standard deviation of the principal components and can thus be obtained as follow

```r
# Standard deviation of each PC
cars04.pca$sdev # i.e. square root of the eigenvalue of the covariance matrix
```

```
##  [1] 2.66545276 1.37256139 0.92180708 0.59750773 0.52481958 0.44490866
##  [7] 0.37485892 0.29434472 0.25765865 0.19229499 0.02811325
```

```r
# Eigenvalues of each PC i.e. variance retained
eig <- (cars04.pca$sdev)^2
eig
```

```
##  [1] 7.1046384308 1.8839247679 0.8497282852 0.3570154894 0.2754355932
##  [6] 0.1979437155 0.1405192086 0.0866388119 0.0663879807 0.0369773622
## [11] 0.0007903547
```

```r
# Variance in percentage i.e. proportion of variance (also given in summary)
variance <- eig*100/sum(eig)
variance
```

```
##  [1] 64.587622098 17.126588799  7.724802592  3.245595359  2.503959939
##  [6]  1.799488322  1.277447350  0.787625563  0.603527097  0.336157838
## [11]  0.007185043
```

```r
# Cumulative variances in percentage (also given in the summary)
cumvar <- cumsum(variance)
cumvar
```

```
##  [1]  64.58762  81.71421  89.43901  92.68461  95.18857  96.98806  98.26550
##  [8]  99.05313  99.65666  99.99281 100.00000
```

```r
# Data frame with eigenvalues, variance, and cumulative variance
eig.cars04 <- data.frame(eig = eig, variance = variance,
                                    cumvariance = cumvar)
eig.cars04
```

```
##             eig     variance cumvariance
## 1  7.1046384308 64.587622098    64.58762
## 2  1.8839247679 17.126588799    81.71421
## 3  0.8497282852  7.724802592    89.43901
## 4  0.3570154894  3.245595359    92.68461
## 5  0.2754355932  2.503959939    95.18857
## 6  0.1979437155  1.799488322    96.98806
## 7  0.1405192086  1.277447350    98.26550
## 8  0.0866388119  0.787625563    99.05313
## 9  0.0663879807  0.603527097    99.65666
## 10 0.0369773622  0.336157838    99.99281
## 11 0.0007903547  0.007185043   100.00000
```

There are 11 principal components PC1-11 each of which explains a certain percentage of the total variance in the dataset. PC1 explains nearly 65% of it, PC2 explains 17% of it etc. By knowing PC1 and PC2, 82% of the variance of the dataset is explained. Note that the standard deviation, variance and cumulative variance of the principal components can also be accessed from the summary of the PCA object or using the factoextra package as follow

```r
# Summary
summary(cars04.pca)
```

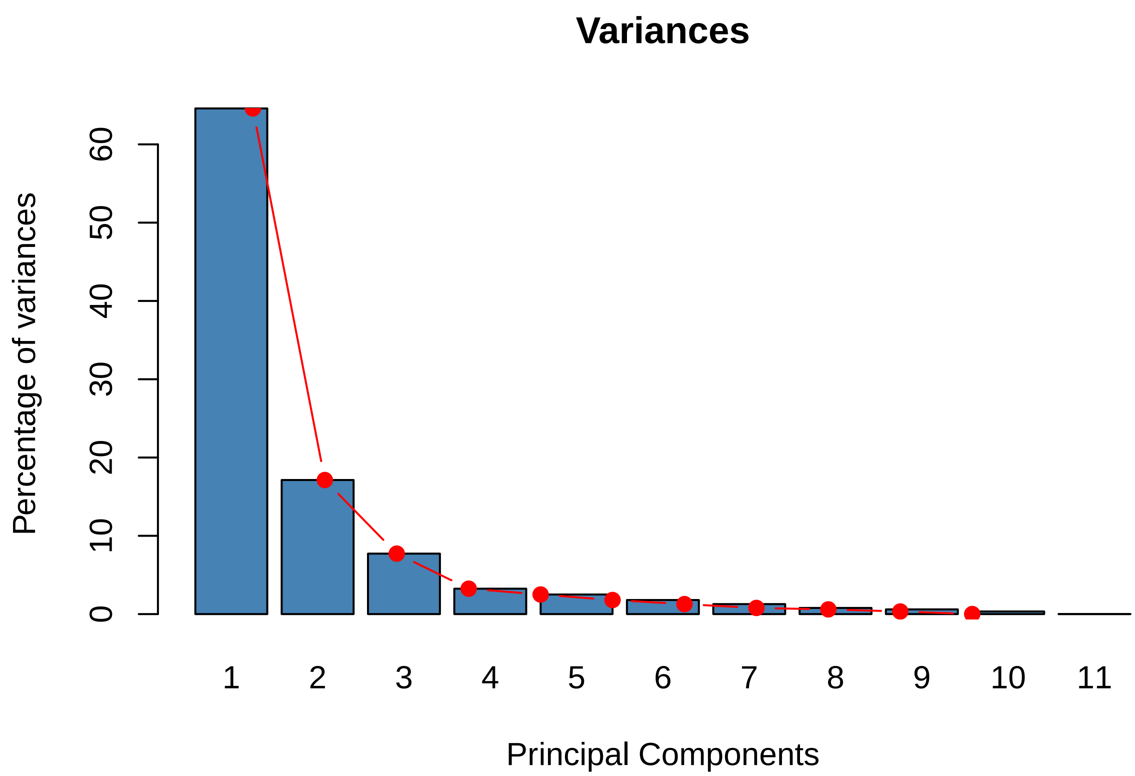
```
## Importance of components:
##                           PC1    PC2     PC3     PC4     PC5     PC6
## Standard deviation     2.6655 1.3726 0.92181 0.59751 0.52482 0.44491
## Proportion of Variance 0.6459 0.1713 0.07725 0.03246 0.02504 0.01799
## Cumulative Proportion  0.6459 0.8171 0.89439 0.92685 0.95189 0.96988
##                            PC7     PC8     PC9    PC10    PC11
## Standard deviation     0.37486 0.29434 0.25766 0.19229 0.02811
## Proportion of Variance 0.01277 0.00788 0.00604 0.00336 0.00007
## Cumulative Proportion  0.98266 0.99053 0.99657 0.99993 1.00000
```

```r
# or can be accessed using the factoextra package
eig.val <- get_eigenvalue(cars04.pca)
head(eig.val)
```

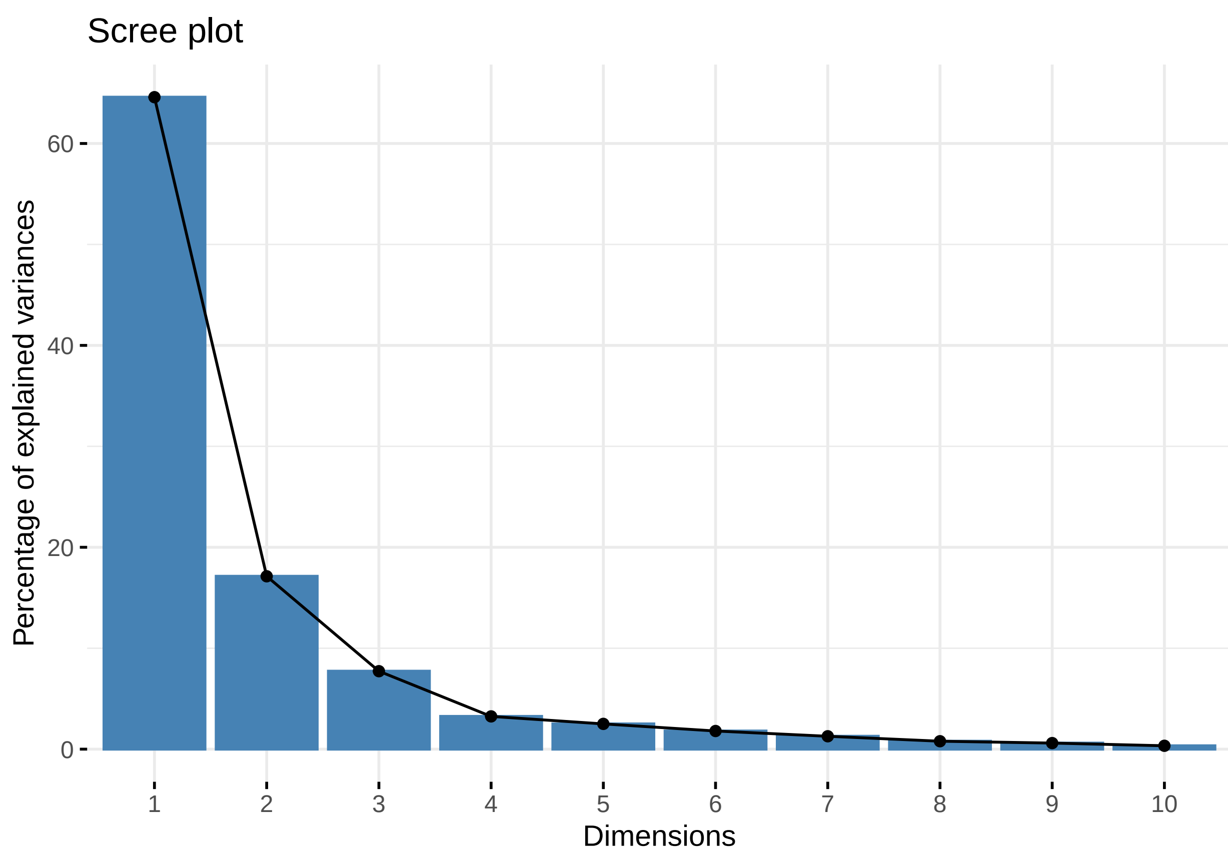
```
##       eigenvalue variance.percent cumulative.variance.percent
## Dim.1  7.1046384        64.587622                    64.58762
## Dim.2  1.8839248        17.126589                    81.71421
## Dim.3  0.8497283         7.724803                    89.43901
## Dim.4  0.3570155         3.245595                    92.68461
## Dim.5  0.2754356         2.503960                    95.18857
## Dim.6  0.1979437         1.799488                    96.98806
```

The importance of PC can be visualized using a screeplot i.e. bar plot of the variance for each dimension/PC with added connecting lines. The scree plot can be made using base R or using the factoextra package

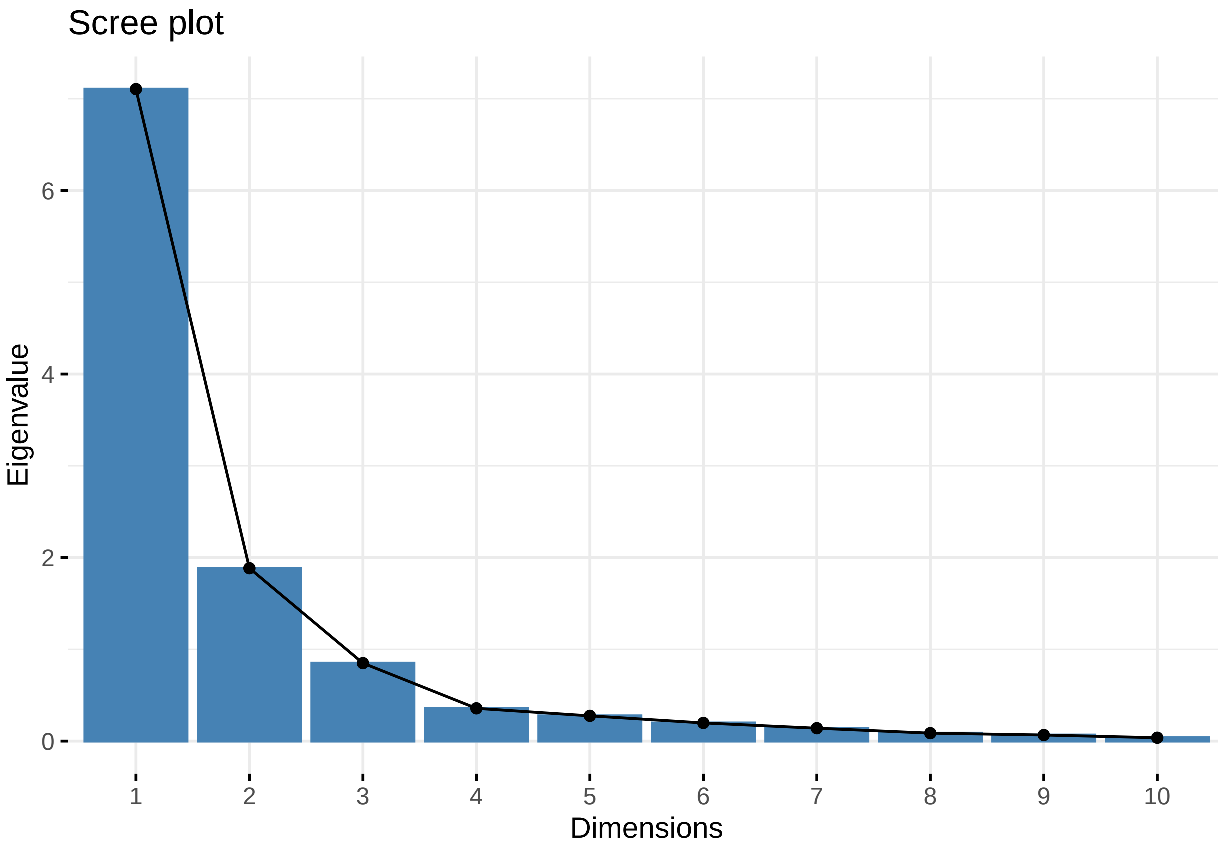
```r
## Scree Plot
# Using Base R
barplot(eig.cars04[, 2], names.arg=1:nrow(eig.cars04),
        main = "Variances",
        xlab = "Principal Components",
        ylab = "Percentage of variances",
        col ="steelblue")
# Add connected line segments to the plot
lines(x = 1:nrow(eig.cars04),
      eig.cars04[, 2],
      type="b", pch=19, col = "red")
```




```r
# Using factoextra package
fviz_eig(cars04.pca)
```



```r
# Also possible to visualize the eigenvalues instead of the explained variance using factoextra
fviz_screeplot(cars04.pca, ncp=10, choice="eigenvalue")
```



To determine the number of dimensions/PC to retain, there exist several criteria

- Kaiser Criterion: Keep PC with eigenvalue > 1 as this indicates that PCs account for more variance than accounted by one of the original variables in standardized data. This is commonly used as a cutoff point for which PCs are retained.
- Elbow shape Criterion of the scree plot
- Criterion on the cumulative variance: limit the number of component to that number that accounts for a certain fraction of the total variance e.g. 80%

In our case using Kaiser Criterion, the two first PC will be retained as these are the only principal components having eigenvalues above 1.

2.2 Graph of variables: the correlation circle

The loading matrix i.e. matrix of the eigenvectors of the covariance matrix expressed in terms of initial features can be generated as follow

```
## Loading or weight matrix
round(cars04.pca$rotation[,],2)
```

```
##              PC1   PC2   PC3   PC4   PC5   PC6   PC7   PC8   PC9  PC10
## Retail     -0.26 -0.47 -0.25  0.28 -0.05 -0.03 -0.22  0.05  0.09 -0.02
## Dealer     -0.26 -0.47 -0.26  0.29 -0.04 -0.05 -0.22  0.07  0.09 -0.03
## Engine     -0.35  0.02 -0.05 -0.53 -0.05 -0.01 -0.05 -0.36  0.68  0.01
## Cylinders  -0.33 -0.08 -0.08 -0.64  0.13 -0.09 -0.24  0.42 -0.45  0.00
## Horsepower -0.32 -0.29 -0.08 -0.06  0.12  0.21  0.81 -0.18 -0.23 -0.01
## CityMPG     0.31  0.00 -0.54 -0.19 -0.33 -0.25  0.09 -0.17 -0.12 -0.60
## HighwayMPG  0.31  0.01 -0.60 -0.13 -0.04  0.08  0.06  0.00  0.01  0.72
## Weight     -0.34  0.17  0.11  0.12 -0.40 -0.54 -0.04 -0.42 -0.34  0.30
## Wheelbase  -0.27  0.42 -0.26  0.22  0.22 -0.45  0.30  0.46  0.28 -0.04
## Length     -0.26  0.41 -0.34  0.17  0.46  0.31 -0.28 -0.41 -0.23 -0.14
## Width      -0.30  0.31 -0.09  0.09 -0.66  0.53  0.02  0.28  0.01 -0.04
##             PC11
## Retail     -0.71
## Dealer      0.70
## Engine      0.01
## Cylinders   0.00
## Horsepower  0.00
## CityMPG     0.00
## HighwayMPG  0.00
## Weight      0.00
## Wheelbase  -0.01
## Length      0.00
## Width       0.01
```

From the loading matrix, one can see that all initial features have a negative projection on PC1 except gas mileage. The first principal component tells us about whether we are getting a big, expensive gas-guzzling car with a powerful engine, or whether we are getting a small, cheap, fuel-efficient car with a wimpy engine. PC1 ~ Engine size and gas. size vs. fuel efficiency One can also notice that mileage hardly project on to PC2 at all. Instead we have a contrast between the physical size of the car (positive projection) and the price and horsepower. Basically, this axis separates mini-vans, trucks and SUVs (big, not so expensive, not so much horse-power) from sports-cars (small, expensive, lots of horse-power). PC2 ~ sporty vs boxy.

The correlation between variables and principal components can be visualized using the correlation circle. Each principal component is represented as an arrow whose coordinates are its correlations with PC1 and PC2. Correlation between variables and principal components are calculated as loadings multiplied by the principal components' standard deviations.

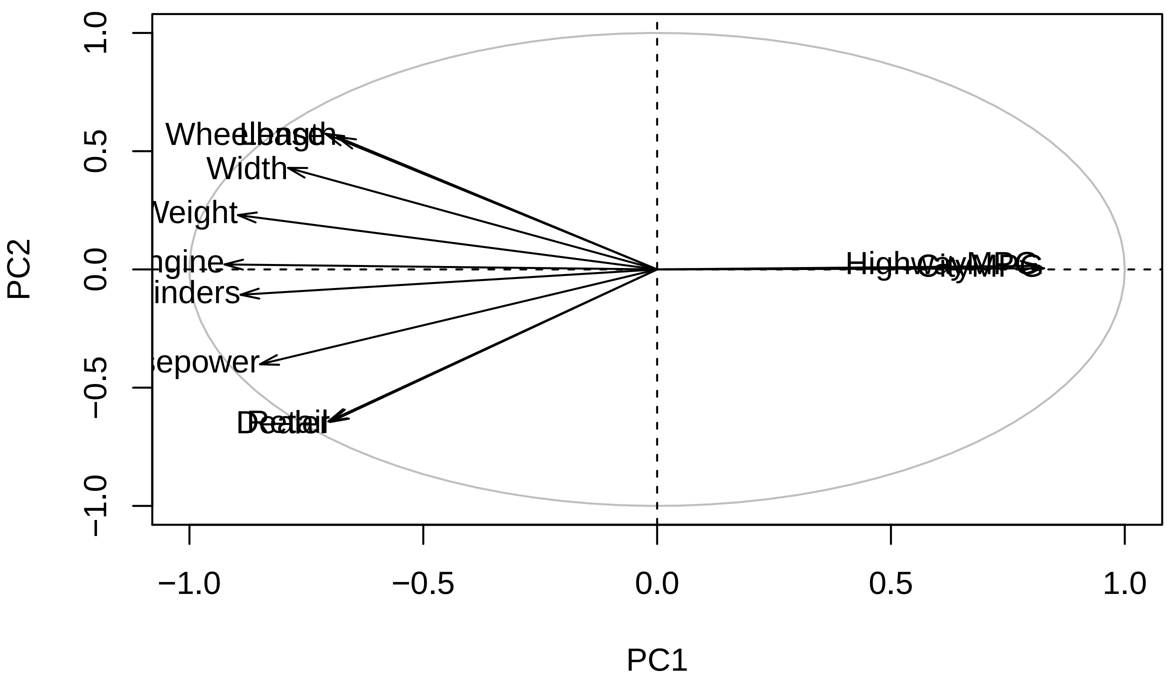
The correlation circle visualization can be made using base R as follow

```
## Graph of variables: the correlation Circle
# Correlation between variables and principal components
var_cor_func <- function(var.loadings, comp.sdev){
  var.loadings*comp.sdev
}
# Variable correlation/coordinates
loadings <- cars04.pca$rotation
sdev <- cars04.pca$sdev
var.coord <- var.cor <- t(apply(loadings, 1, var_cor_func, sdev))
```

```r
head(var.coord[, 1:4])
```

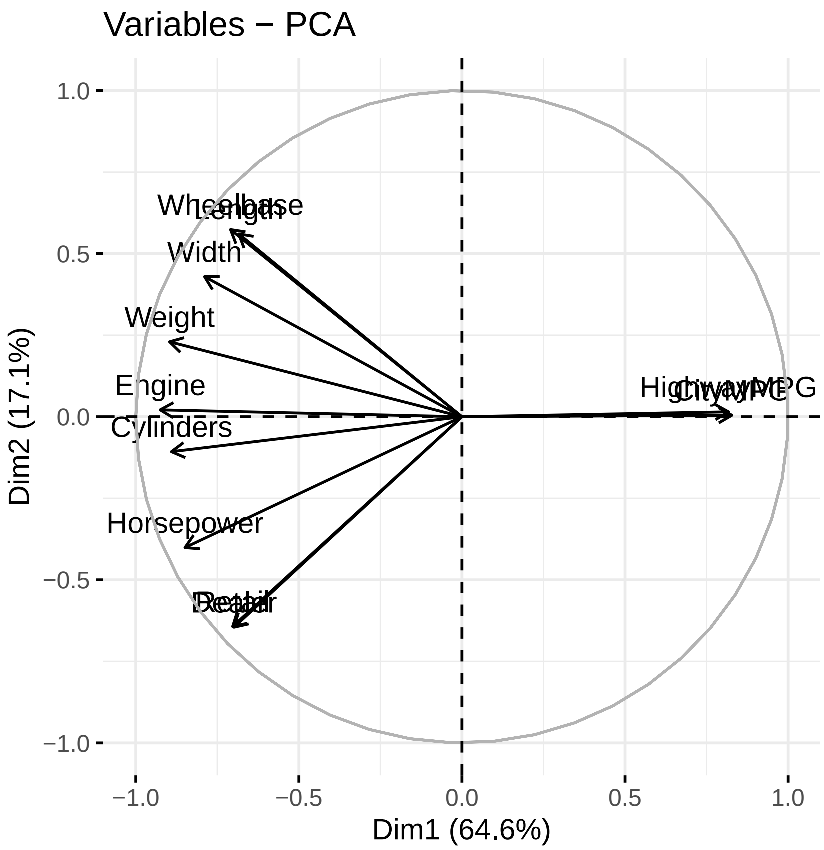
```
##                  PC1          PC2         PC3         PC4
## Retail     -0.7030143 -0.643056949 -0.23503697  0.16723268
## Dealer     -0.6991979 -0.645305050 -0.23713523  0.17191468
## Engine     -0.9251267  0.021064956 -0.04350397 -0.31391314
## Cylinders  -0.8907643 -0.107103725 -0.07508051 -0.38228518
## Horsepower -0.8492193 -0.401080935 -0.07040843 -0.03486045
## CityMPG     0.8275744  0.004619953 -0.49322482 -0.11130109
```

```r
# Graph of variables using Base R
# Plot the correlation circle
a <- seq(0, 2*pi, length = 100)
plot( cos(a), sin(a), type = 'l', col="gray",
      xlab = "PC1",  ylab = "PC2")
# Plot axis
abline(h = 0, v = 0, lty = 2)
# Add active variables
arrows(0, 0, var.coord[, 1], var.coord[, 2],
       length = 0.1, angle = 15, code = 2)
# Add labels
text(var.coord, labels=rownames(var.coord), cex = 1, adj=1)
```



or using the factoextra package

```r
# Using factoextra package
fviz_pca_var(cars04.pca)
```



The graph of variables, i.e. correlation circle, shows the relationships between all variables :

- Positively correlated variables are grouped together.
- Negatively correlated variables are positioned on opposite sides of the plot origin (opposed quadrants).
- The distance between variables and the origine measures the quality of the variables on the factor map. Variables that are away from the origin are well represented on the factor map.

The two mileage variables are positioned on opposite quadrants compared to the other variables when looking at PC1. This means that the two mileage variables are negatively correlated to the other variables. This confirms our intuition about PC1 indicating whether a car is fuel efficient, small, cheap vs big, gazz guzzling, and expensive. The two milage variables are along the PC2 axis. The variables indicating the size of a car (wheelbase, width, length, weight, engine) are in opposite quadrant to the variables indicating the power (horsepower, cylinders) and price of the car (dealer and retail) meaning that these variables are negatively correlated. The PC2 thus represents whether a car is a sport car i.e. powerful and expensive, small vs. bulky i.e. not powerful, cheap and big.

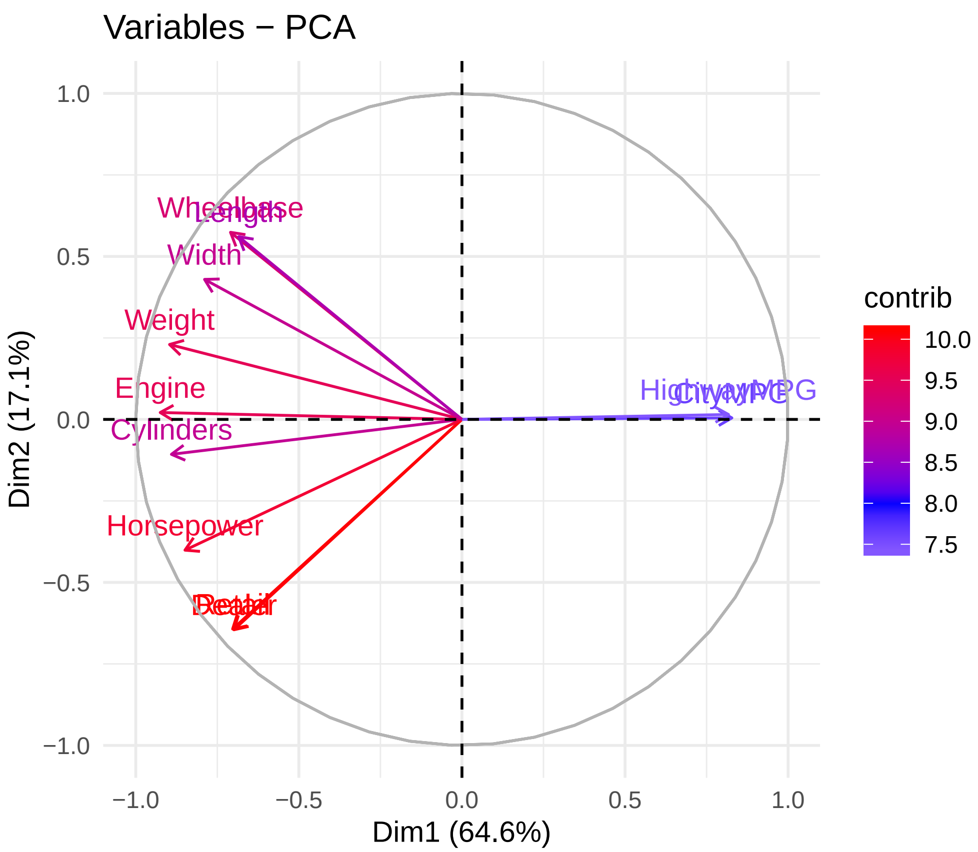
The variable arrows can be colored according to the quality of representation (cos2) of the variables on the factor map. The quality of representation correspond to the squared coordinates (loading multiplied by standard deviation) of the variables. Using factoextra package, the color of variables on the correlation circle can be automatically controlled by the value of their cos2. On the plot below, the higher (resp. lower) the cos2 of the variable is, the more "red" (resp. "blue") the arrow is.

```
## Quality of representation of the variables on the factor map
var.cos2 <- var.coord^2
head(var.cos2[, 1:4])
```

```
##                  PC1          PC2         PC3         PC4
## Retail     0.4942292 4.135222e-01 0.055242376 0.027966771
## Dealer     0.4888778 4.164186e-01 0.056233118 0.029554656
```

```
## Engine     0.8558593 4.437324e-04 0.001892595 0.098541461
## Cylinders  0.7934611 1.147121e-02 0.005637083 0.146141962
## Horsepower 0.7211734 1.608659e-01 0.004957347 0.001215251
## CityMPG    0.6848793 2.134397e-05 0.243270722 0.012387933
```

```
# Using factoextra package
fviz_pca_var(cars04.pca, col.var="contrib")+
scale_color_gradient2(low="white", mid="blue",
                      high="red", midpoint=8) + theme_minimal()
```



The contribution of a variable to a principal component can also be used instead of the quality of representation to color the arrows of the variables on the correlation circle. The contribution of a variable to a principal component is in percentage: (var.cos2 * 100) / (total cos2 of the component). Below, the correlation circle is given again with the variable arrows colored according to their contribution to PC2.

```
## Contribution of variables to PC2
comp.cos2 <- apply(var.cos2, 2, sum)
comp.cos2
```

```
##          PC1          PC2          PC3          PC4          PC5
## 7.1046384308 1.8839247679 0.8497282852 0.3570154894 0.2754355932
##          PC6          PC7          PC8          PC9         PC10
## 0.1979437155 0.1405192086 0.0866388119 0.0663879807 0.0369773622
##         PC11
## 0.0007903547
```

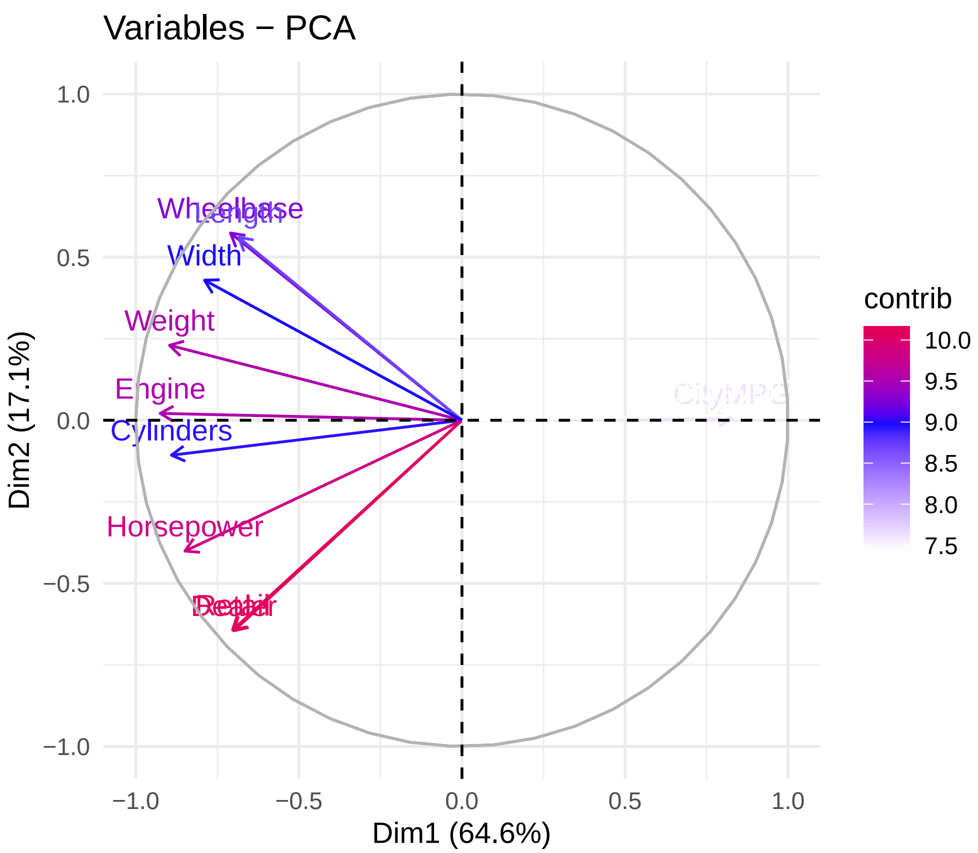
```
contrib <- function(var.cos2, comp.cos2){var.cos2*100/comp.cos2}

var.contrib <- t(apply(var.cos2,1, contrib, comp.cos2))
```

```r
head(var.contrib[, 1:4])
```

```
##                  PC1          PC2        PC3        PC4
## Retail      6.956430 21.950039964  6.5011813  7.8334894
## Dealer      6.881107 22.103781152  6.6177765  8.2782559
## Engine     12.046487  0.023553613  0.2227294 27.6014525
## Cylinders  11.168213  0.608899472  0.6633983 40.9343478
## Horsepower 10.150740  8.538871564  0.5834038  0.3403916
## CityMPG     9.639890  0.001132952 28.6292367  3.4698587
```

```r
# Highlight the most contributing variables to a PC2
fviz_pca_var(cars04.pca, col.var="contrib") +
scale_color_gradient2(low="white", mid="blue",
                      high="red", midpoint=9) + theme_minimal()
```



2.3 Graph of Individuals

The coordinates of the individuals on the principal components can be obtained easily from the PCA object as follow

```
## Graph of individuals
# Coordinates of individuals on the PC
ind.coord <- cars04.pca$x
head(ind.coord[, 1:4])
```

```
##                                 PC1         PC2        PC3        PC4
## Acura 3.5 RL             -1.5654166  0.44669421 -0.2870171  0.6098609
## Acura 3.5 RL Navigation -1.6335337  0.33929273 -0.3452409  0.6788642
## Acura MDX                -1.9041537  0.41060707  0.5519813  0.2956175
## Acura NSX S              -1.5881428 -3.85758573 -0.3563574  1.1480124
## Acura RSX                 2.6513314 -0.65360524  0.1731496  0.2890318
## Acura TL                 -0.4406721 -0.08100219 -0.1895334 -0.1317345
```

The quality of representation for the individuals on the principal components can be calculated in 2 steps

- Calculate the square distance between each individual and the PCA center of gravity

$$d2 = [(var1_{indi} - mean_{var1})/sd_{var1}]^2 + \ldots + [(var10_{indi} - mean_{var10})/sd_{var10}]^2 + \ldots + ..$$

- Calculate cos2

$$cos2 = ind.coord^2/d2$$

The contribution of individuals (in percentage) to the principal components can then be computed as follow

$$100 * (1/\text{number of individuals}) * (ind.coord^2/sdev_{PC}^2)$$

```
## Quality of representation for individuals on the principal components
center <- cars04.pca$center
scale<- cars04.pca$scale
# Compute d2
getdistance <- function(ind_row, center, scale){
  return(sum(((ind_row-center)/scale)^2))
}
d2 <- apply(eig.cars04,1,getdistance, center, scale)
# Compute the cos2
cos2 <- function(ind.coord, d2){return(ind.coord^2/d2)}
ind.cos2 <- apply(ind.coord, 2, cos2, d2)
head(ind.cos2[1:5, ])
```

```
##                                   PC1          PC2          PC3
## Acura 3.5 RL            0.0006109644 4.974812e-05 2.053864e-05
## Acura 3.5 RL Navigation 0.0004002424 1.726696e-05 1.787768e-05
## Acura MDX               0.0004508980 2.096656e-05 3.788987e-05
## Acura NSX S             0.0002910073 1.716943e-03 1.465197e-05
## Acura RSX               0.0007698675 4.678634e-05 3.283453e-06
##                                   PC4          PC5          PC6
## Acura 3.5 RL            9.272942e-05 5.406123e-05 2.652080e-05
## Acura 3.5 RL Navigation 6.912460e-05 3.012231e-05 1.791927e-05
## Acura MDX               1.086762e-05 1.895058e-04 4.232962e-05
```

```
## Acura NSX S             1.520610e-04 5.063666e-06 3.303265e-05
## Acura RSX               9.149120e-06 1.030272e-06 2.184723e-06
##                                  PC7          PC8          PC9
## Acura 3.5 RL            7.294706e-06 2.017725e-07 1.209972e-05
## Acura 3.5 RL Navigation 7.509884e-06 1.807889e-07 8.282946e-06
## Acura MDX               4.761373e-06 1.502004e-05 2.323675e-05
## Acura NSX S             3.620795e-05 1.466631e-05 1.665256e-05
## Acura RSX               2.955138e-05 4.000214e-07 2.116204e-08
##                                 PC10         PC11
## Acura 3.5 RL            1.030761e-05 5.799640e-07
## Acura 3.5 RL Navigation 6.236102e-06 3.811190e-07
## Acura MDX               1.879836e-06 1.341338e-09
## Acura NSX S             4.628129e-06 6.299065e-07
## Acura RSX               3.284588e-07 1.130455e-08
```

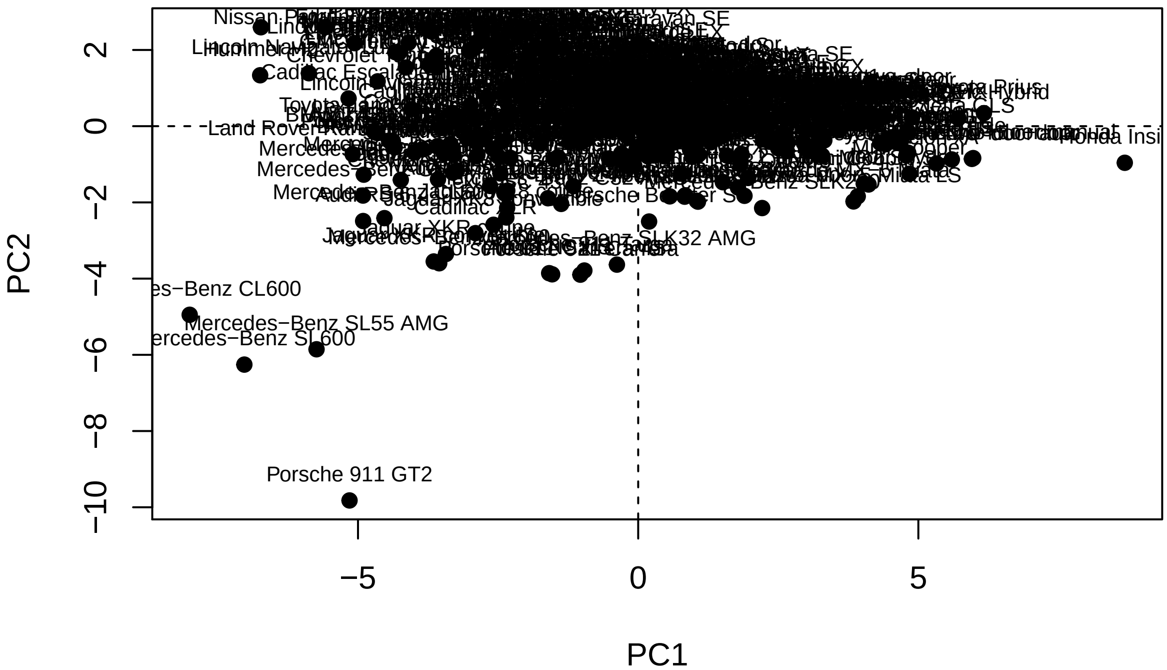
```
# Contributions of individuals to the principal components in percentage
contrib <- function(ind.coord, comp.sdev, n.ind){
  100*(1/n.ind)*ind.coord^2/comp.sdev^2
}

ind.contrib <- t(apply(ind.coord,1, contrib,
                       cars04.pca$sdev, nrow(ind.coord)))
head(ind.contrib[, 1:4])
```

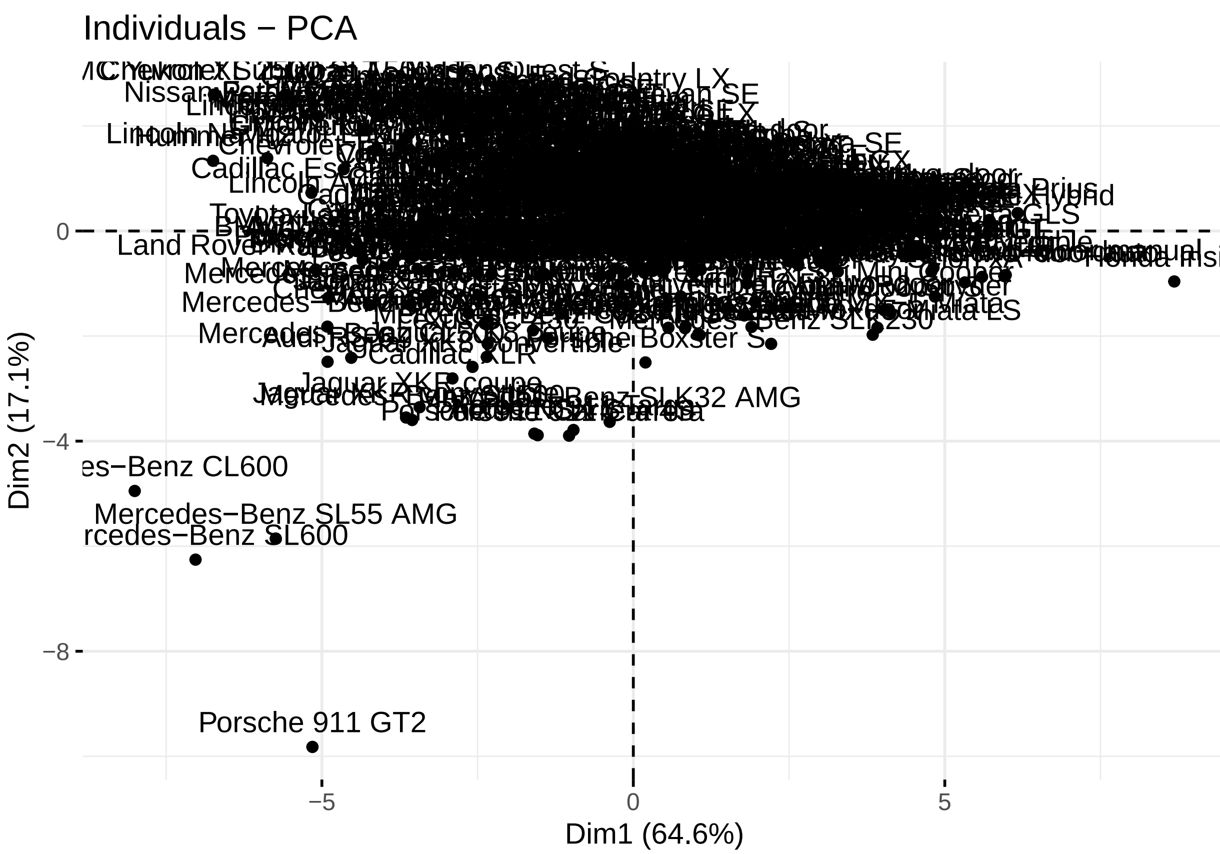
```
##                                PC1          PC2         PC3        PC4
## Acura 3.5 RL            0.08912652 0.0273681935 0.025050969 0.26919283
## Acura 3.5 RL Navigation 0.09705173 0.0157897260 0.036245442 0.33355524
## Acura MDX               0.13187153 0.0231248185 0.092652613 0.06325024
## Acura NSX S             0.09173312 2.0410641397 0.038617167 0.95388311
## Acura RSX               0.25566724 0.0585944568 0.009116997 0.06046351
## Acura TL                0.00706282 0.0008999514 0.010923974 0.01256034
```

The individual graph can be made either using base R or the factoextra package. Each individual is represented as a text (e.g. name of the individual car) centered around a point whose coordinates are the individual coordinates on PC1 and PC2. The individuals (i.e. texts) can then be colored according to the individual represnetation (cos2) or the contribution of the individuals to the principal components using the factoextra package in a similar way as for the variable graph

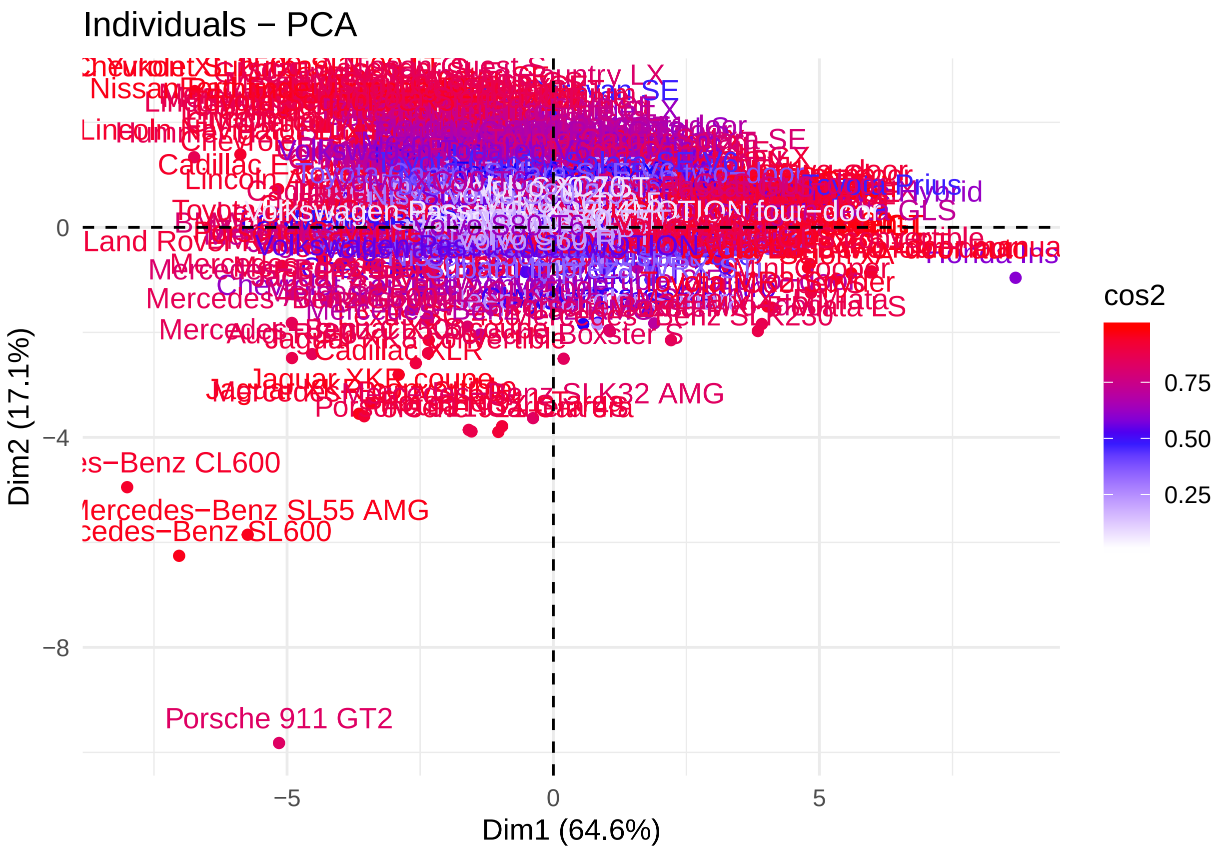
```
## Individual Graph using Base R
plot(ind.coord[,1], ind.coord[,2], pch = 19,
     xlab="PC1",ylab="PC2")
abline(h=0, v=0, lty = 2)
text(ind.coord[,1], ind.coord[,2], labels=rownames(ind.coord),
     cex=0.7, pos = 3)
```




```r
## Using factoextra package
fviz_pca_ind(cars04.pca)
```

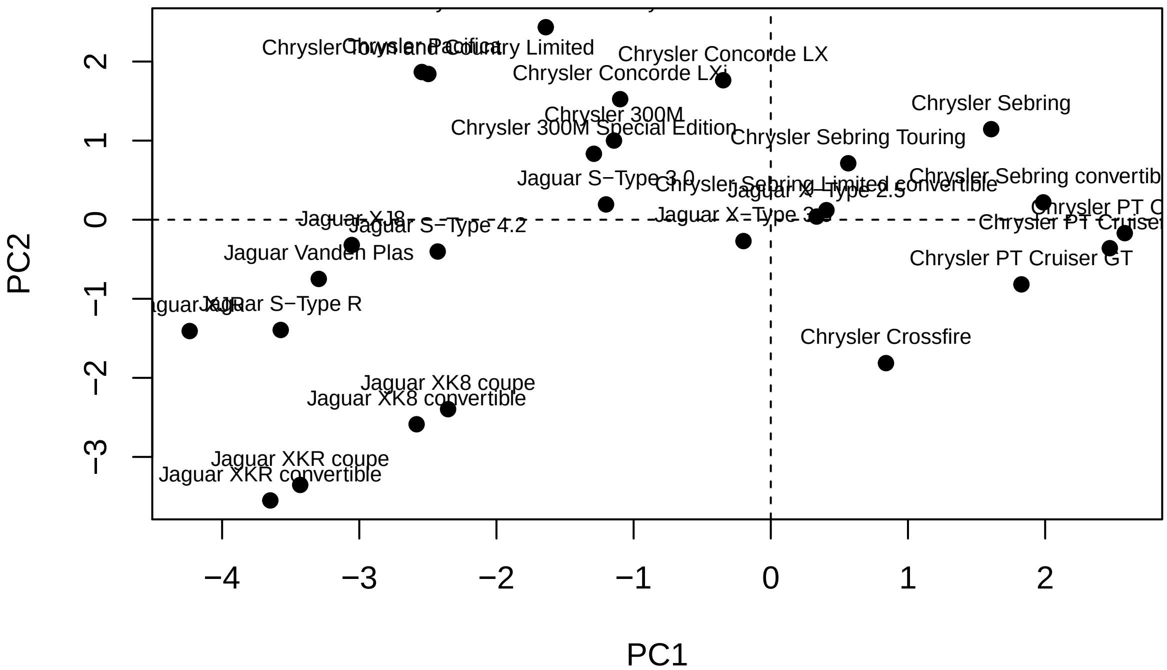



```r
fviz_pca_ind(cars04.pca, col.ind="cos2") +
  scale_color_gradient2(low="white", mid="blue",
                        high="red", midpoint=0.50) + theme_minimal()
```

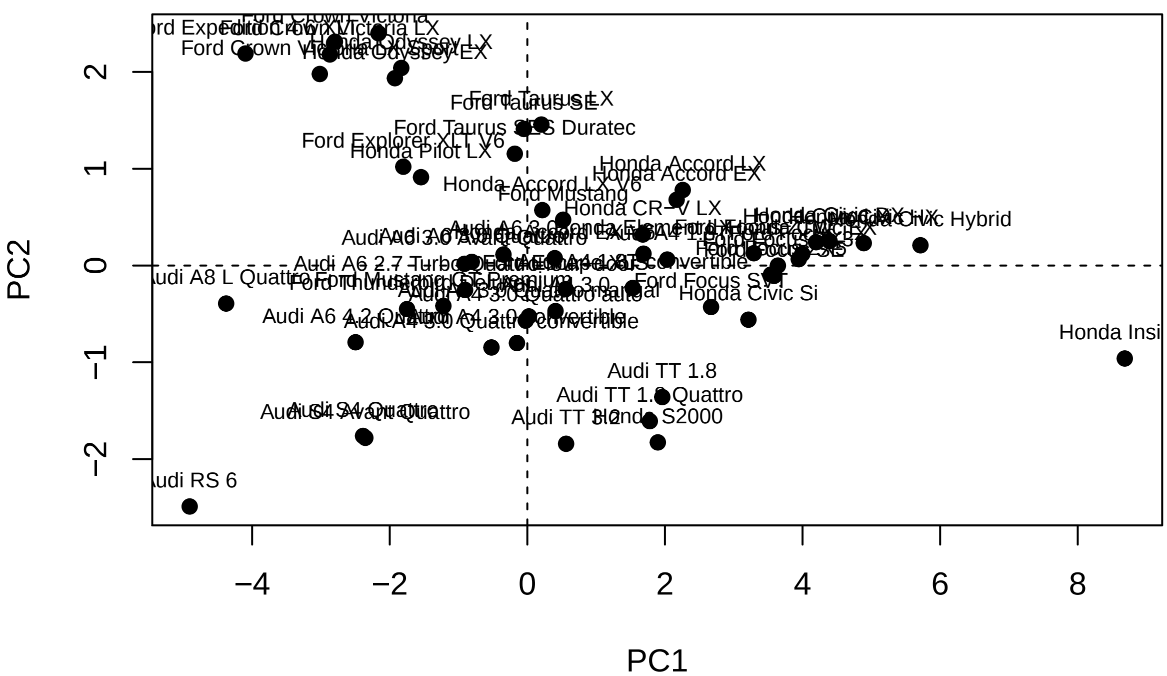



```
## Subset of the data
ind.coord.sub1 <-ind.coord[grep("Chrysler", row.names(ind.coord)),]
ind.coord.sub3 <-ind.coord[grep("Jaguar", row.names(ind.coord)),]
ind.coord.sub2 <-ind.coord[grep("Honda", row.names(ind.coord)),]
ind.coord.sub4 <-ind.coord[grep("Audi", row.names(ind.coord)),]
ind.coord.sub5 <-ind.coord[grep("Ford", row.names(ind.coord)),]

ind.coord.sub <- rbind(ind.coord.sub1,ind.coord.sub3)
ind.coord.sub_2 <- rbind(ind.coord.sub2,ind.coord.sub4, ind.coord.sub5)
## Individual Graph using Base R on a subset
plot(ind.coord.sub[,1], ind.coord.sub[,2], pch = 19,
     xlab="PC1",ylab="PC2")
abline(h=0, v=0, lty = 2)
text(ind.coord.sub[,1], ind.coord.sub[,2], labels=rownames(ind.coord.sub),
     cex=0.7, pos = 3)
```



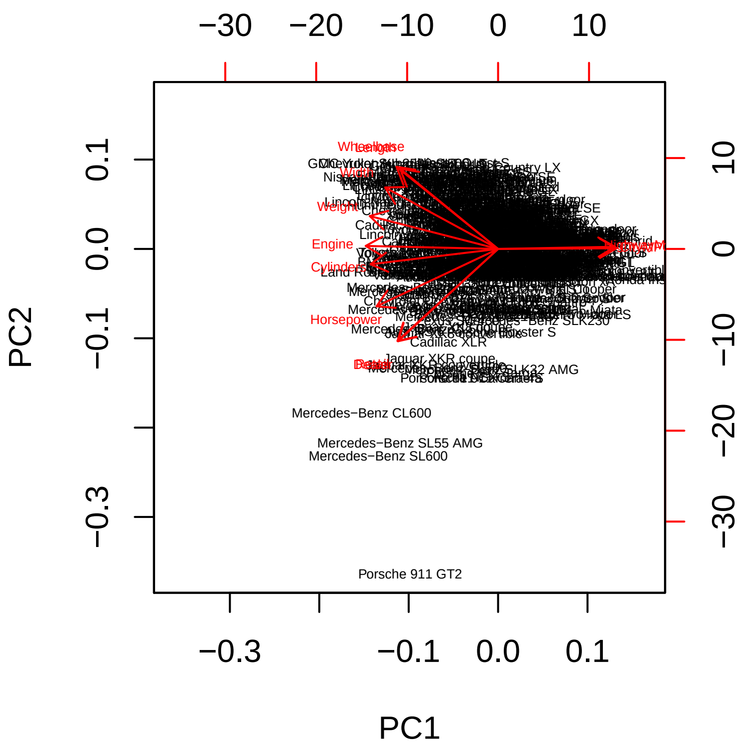

```r
## Individual Graph using Base R on a subset
plot(ind.coord.sub_2[,1], ind.coord.sub_2[,2], pch = 19,
     xlab="PC1",ylab="PC2")
abline(h=0, v=0, lty = 2)
text(ind.coord.sub_2[,1], ind.coord.sub_2[,2], labels=rownames(ind.coord.sub_2),
     cex=0.7, pos = 3)
```



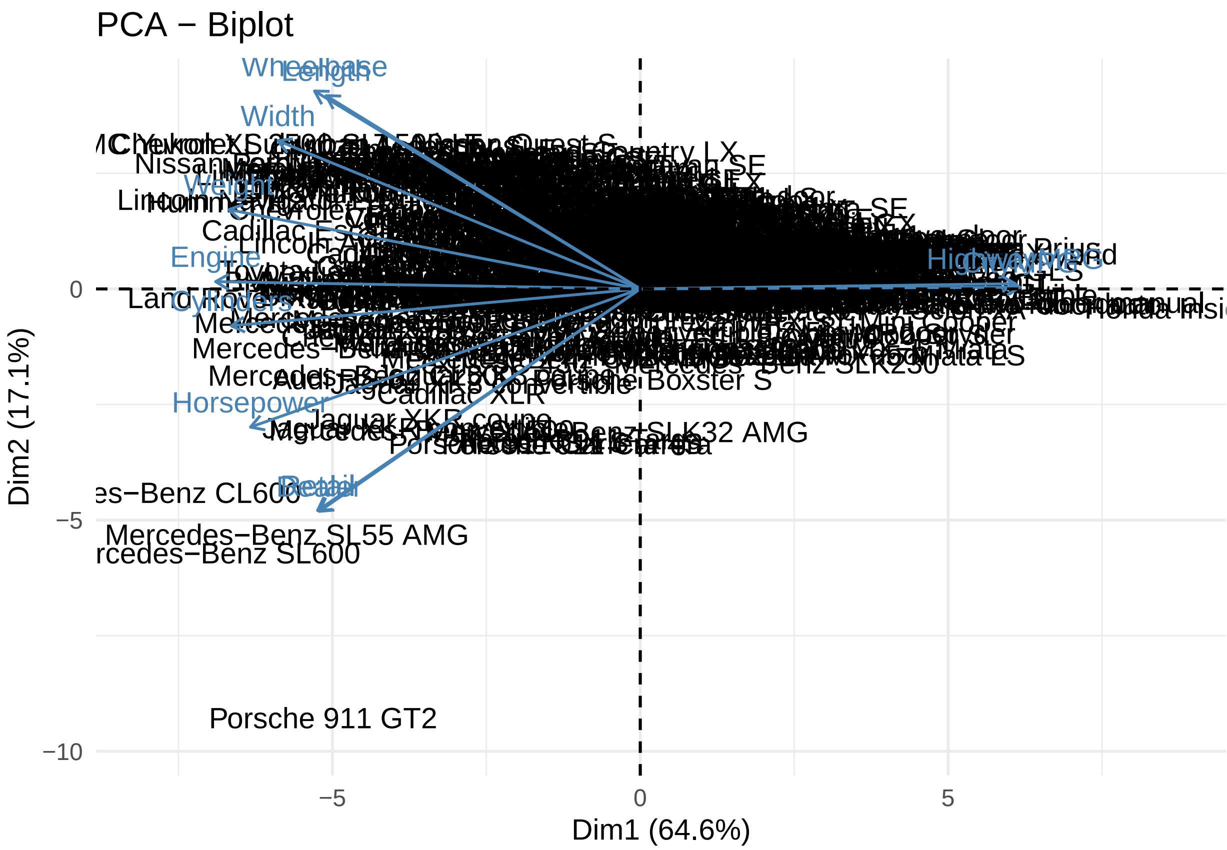
2.4 Biplots

The variable and individual graphs can be combined in one plot called the biplot. The biplot can be visualized using Base R or the factoextra package.

```r
## Biplot of individuals and variables using Base R
biplot(cars04.pca,cex=0.4)
```




```r
## Biplot using the factoextra package
fviz_pca_biplot(cars04.pca,  geom = "text") +
  theme_minimal()
```


PCA − Biplot
Wheelbase
Length
Width
Weight
Engine
Cylinders
Horsepower
Jaguar XKR coupe
Cadillac XLR
Mercedes−Benz SL55 AMG
Porsche 911 GT2
Dim1 (64.6%)
Dim2 (17.1%)
−5
0
5
−10

2.5 Predicting using principal components

The variables not used for PCA, that we will call supplementary variables, are saved in another dataset and correlations between these supplementary variables and the principal components are derived and visualized on the correlation circle plot.

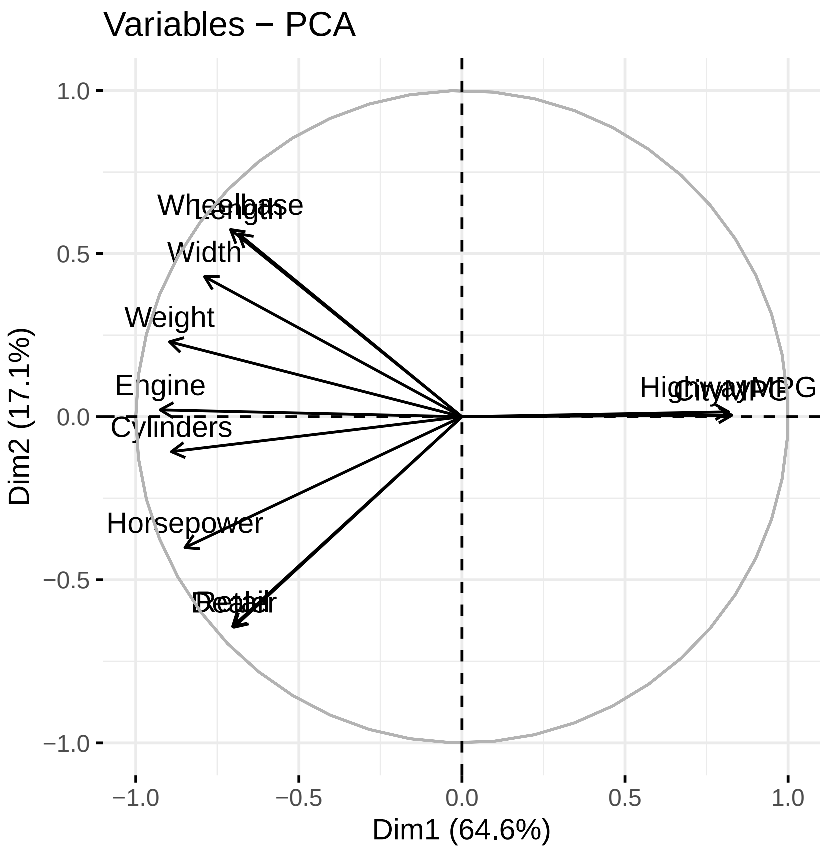
```
# Supplementary variables not used in PCA
ind.supp <- cars04[, 1:7, drop = FALSE]
head(ind.supp)
```

```
##                            Sports SUV Wagon Minivan Pickup AWD RWD
## Acura 3.5 RL                    0   0     0       0      0   0   0
## Acura 3.5 RL Navigation         0   0     0       0      0   0   0
## Acura MDX                       0   1     0       0      0   1   0
## Acura NSX S                     1   0     0       0      0   0   1
## Acura RSX                       0   0     0       0      0   0   0
## Acura TL                        0   0     0       0      0   0   0
```

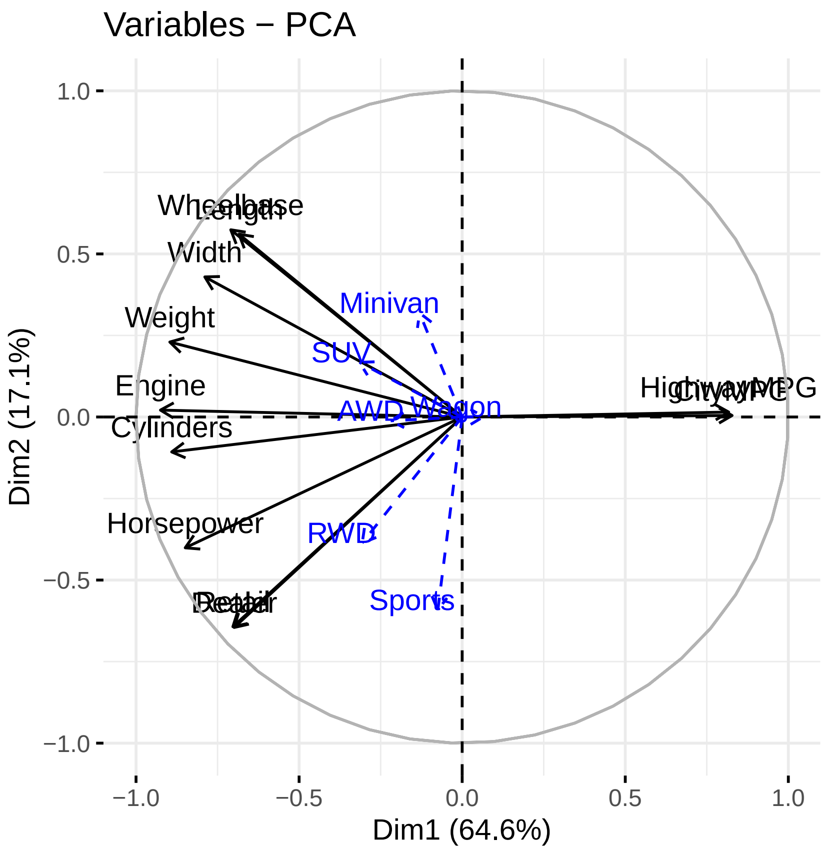
```
# Calculate the correlations between supplementary variables
# and the principal components
ind.coord <- cars04.pca$x
quanti.coord <- cor(ind.supp, ind.coord)
head(quanti.coord[, 1:4])
```

```
##                 PC1          PC2         PC3         PC4
## Sports  -0.07467828 -0.591925261  0.10031386 -0.08372176
## SUV     -0.31491236  0.167532811  0.39898337 -0.02551178
## Wagon    0.06582938  0.001107486  0.05875224  0.04492995
## Minivan -0.13084579  0.319065806 -0.02108833  0.18117014
## Pickup           NA           NA          NA          NA
## AWD     -0.21925943 -0.011662688  0.34366628  0.10953423
```

```
# Plot the correlation circle using factoextra
# Plot of active variables
p <- fviz_pca_var(cars04.pca)
p
```




```
# Add supplementary active variables
fviz_add(p, quanti.coord, color ="blue", geom="arrow")
```




```r
# get the cos2 of the supplementary quantitative variables
(quanti.coord^2)[, 1:4]
```

```
##                 PC1          PC2          PC3          PC4
## Sports  0.005576845 3.503755e-01 0.0100628702 0.0070093335
## SUV     0.099169796 2.806724e-02 0.1591877260 0.0006508511
## Wagon   0.004333508 1.226524e-06 0.0034518255 0.0020187007
## Minivan 0.017120620 1.018030e-01 0.0004447179 0.0328226198
## Pickup           NA           NA           NA           NA
## AWD     0.048074699 1.360183e-04 0.1181065119 0.0119977468
## RWD     0.093916577 1.491823e-01 0.0145306280 0.0201005053
```

We now make two logistic regression models to predict whether a car is a sports car or not. The first model uses all features available about the cars, the second model only uses PC1 and PC2. We compare the performances of these models.

```r
## Baseline Model
# we know sports car are less common
table(cars04$Sports)
```

```
##
##   0   1
## 342  45
```

```r
# Accuracy of the baseline model
342/387
```

```
## [1] 0.8837209
```

```r
## Splitting the data into a train and test set
# Load package
library(caret) # training/test sets
```

```
## Loading required package: lattice
```

```r
# Set seed
set.seed(123)
# Do the split
cars04.sub <- cars04  %>%
  select(-c("SUV","Wagon","Minivan","Pickup","AWD","RWD"))
training.sample <- cars04.sub$Sports %>% createDataPartition(p = 0.8, list = FALSE)
cars04.train.data  <- cars04.sub[training.sample, ]
cars04.test.data <- cars04.sub[-training.sample, ]
# Check rows of the train and test datasets
nrow(cars04.train.data)
```

```
## [1] 310
```

```r
nrow(cars04.test.data)
```

```
## [1] 77
```

```r
# First rows of the train and test datasets
head(cars04.train.data)
```

```
##                             Sports Retail Dealer Engine Cylinders Horsepower
## Acura 3.5 RL Navigation          0  46100  41100    3.5         6        225
## Acura NSX S                      1  89765  79978    3.2         6        290
## Acura RSX                        0  23820  21761    2.0         4        200
## Acura TSX                        0  26990  24647    2.4         4        200
## Audi A4 1.8T                     0  25940  23508    1.8         4        170
## Audi A4 1.8T convertible         0  35940  32506    1.8         4        170
##                             CityMPG HighwayMPG Weight Wheelbase Length Width
## Acura 3.5 RL Navigation          18         24   3893       115    197    72
## Acura NSX S                      17         24   3153       100    174    71
## Acura RSX                        24         31   2778       101    172    68
## Acura TSX                        22         29   3230       105    183    69
## Audi A4 1.8T                     22         31   3252       104    179    70
## Audi A4 1.8T convertible         23         30   3638       105    180    70
```

```r
head(cars04.test.data)
```

```
##                                        Sports Retail Dealer Engine Cylinders
## Acura 3.5 RL                                0  43755  39014    3.5         6
## Acura MDX                                   0  36945  33337    3.5         6
## Acura TL                                    0  33195  30299    3.2         6
## Audi A6 2.7 Turbo Quattro four-door         0  42840  38840    2.7         6
## Audi A6 3.0 Avant Quattro                   0  40840  37060    3.0         6
## Audi A6 3.0 Quattro                         0  39640  35992    3.0         6
##                                        Horsepower CityMPG HighwayMPG Weight
## Acura 3.5 RL                                  225      18         24   3880
## Acura MDX                                     265      17         23   4451
## Acura TL                                      270      20         28   3575
## Audi A6 2.7 Turbo Quattro four-door           250      18         25   3836
## Audi A6 3.0 Avant Quattro                     220      18         25   4035
## Audi A6 3.0 Quattro                           220      18         25   3880
```

```
##                                       Wheelbase Length Width
## Acura 3.5 RL                                115    197    72
## Acura MDX                                   106    189    77
## Acura TL                                    108    186    72
## Audi A6 2.7 Turbo Quattro four-door         109    192    71
## Audi A6 3.0 Avant Quattro                   109    192    71
## Audi A6 3.0 Quattro                         109    192    71
```

```
## Perform a pca using prcomp only on train data set
cars04.train.data.pca = prcomp(cars04.train.data[,2:12], center = TRUE, scale.=TRUE, retx = TRUE)
ind.sub.coord <- cars04.train.data.pca$x
## Add ind.sub.coord to dataset
cars04.train.data.new <- cbind(cars04.train.data,ind.sub.coord)
cars04.train.data <- cars04.train.data.new
```

```
## Logistic Regression using initial features
model1 = glm(Sports ~ Retail + Dealer + Engine + Cylinders +
               Horsepower + CityMPG + HighwayMPG + Weight +
               Wheelbase + Length + Width,
             data=cars04.train.data, family=binomial)
summary(model1)
```

```
##
## Call:
## glm(formula = Sports ~ Retail + Dealer + Engine + Cylinders +
##     Horsepower + CityMPG + HighwayMPG + Weight + Wheelbase +
##     Length + Width, family = binomial, data = cars04.train.data)
##
## Deviance Residuals:
##      Min        1Q    Median        3Q       Max
## -1.91398  -0.09687  -0.02637  -0.00421   2.46775
##
## Coefficients:
##               Estimate Std. Error z value Pr(>|z|)
## (Intercept) 15.8183073 19.5663548   0.808  0.41883
## Retail       0.0003144  0.0005798   0.542  0.58763
## Dealer      -0.0003365  0.0006084  -0.553  0.58020
## Engine      -0.3508380  1.4379171  -0.244  0.80724
## Cylinders    0.1250719  0.7581266   0.165  0.86896
## Horsepower   0.0485009  0.0170842   2.839  0.00453 **
## CityMPG     -1.1027857  0.4633950  -2.380  0.01732 *
## HighwayMPG   0.4954568  0.3287005   1.507  0.13173
## Weight      -0.0085153  0.0034991  -2.434  0.01495 *
## Wheelbase   -0.8496562  0.2632397  -3.228  0.00125 **
## Length      -0.0009849  0.0699568  -0.014  0.98877
## Width        1.3716417  0.4979260   2.755  0.00587 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 214.421  on 309  degrees of freedom
## Residual deviance:  39.805  on 298  degrees of freedom
## AIC: 63.805
##
```

```
## Number of Fisher Scoring iterations: 9
```

```r
## Making Predictions on the train set
# Make predictions
predictTrain = predict(model1, type="response")
# Summary
summary(predictTrain)
```

```
##      Min.   1st Qu.    Median      Mean   3rd Qu.      Max.
## 0.0000000 0.0000567 0.0011270 0.1096774 0.0145552 1.0000000
```

```r
tapply(predictTrain, cars04.train.data$Sports, mean)
```

```
##          0          1
## 0.02013292 0.83656804
```

```r
# Confusion matrix for threshold of 0.5
tab1 <- table(cars04.train.data$Sports, predictTrain > 0.5)
tab1
```

```
##
##     FALSE TRUE
##   0   274    2
##   1     5   29
```

```r
# Sensitivity TP/(TP + FN)
tab1[2,2]/(tab1[2,2] + tab1[2,1])
```

```
## [1] 0.8529412
```

```r
# Specificity  TN/(TN + FP)
tab1[1,1]/(tab1[1,1] + tab1[1,2])
```

```
## [1] 0.9927536
```

```r
# Precision TP/predicted yes.
tab1[2,2]/(tab1[2,2] + tab1[1,2])
```

```
## [1] 0.9354839
```

```r
# Accuracy (TP + TN)/(All)
(tab1[2,2] + tab1[1,1])/(tab1[2,2] + tab1[1,1] + tab1[1,2] + tab1[2,1])
```

```
## [1] 0.9774194
```

```r
## Logistic Regression using PC1 and PC2
model2 = glm(Sports ~ PC1 + PC2 ,data=cars04.train.data, family=binomial)
summary(model2)
```

```
##
## Call:
## glm(formula = Sports ~ PC1 + PC2, family = binomial, data = cars04.train.data)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -3.5214  -0.3197  -0.1646  -0.0482   2.4826
##
## Coefficients:
##              Estimate Std. Error z value Pr(>|z|)
## (Intercept)   -3.7322     0.4490  -8.313  < 2e-16 ***
```

```
## PC1           0.1462     0.0991   1.475     0.14
## PC2          -2.1676     0.3438  -6.304 2.89e-10 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 214.42  on 309  degrees of freedom
## Residual deviance: 106.77  on 307  degrees of freedom
## AIC: 112.77
##
## Number of Fisher Scoring iterations: 7
```

```r
## Making Predictions on the train set
# Make predictions
predictTrain2 = predict(model2, type="response")
# Summary
summary(predictTrain2)
```

```
##      Min.   1st Qu.    Median      Mean   3rd Qu.      Max.
## 0.0000364 0.0043083 0.0224259 0.1096774 0.0828054 1.0000000
```

```r
tapply(predictTrain2, cars04.train.data$Sports, mean)
```

```
##          0          1
## 0.05333776 0.56702290
```

```r
# Confusion matrix for threshold of 0.5
tab2 <- table(cars04.train.data$Sports, predictTrain2 > 0.5)
tab2
```

```
##
##     FALSE TRUE
##   0   272    4
##   1    13   21
```

```r
# Sensitivity TP/(TP + FN)
tab2[2,2]/(tab2[2,2] + tab2[2,1])
```

```
## [1] 0.6176471
```

```r
# Specificity  TN/(TN + FP)
tab2[1,1]/(tab2[1,1] + tab2[1,2])
```

```
## [1] 0.9855072
```

```r
# Precision TP/predicted yes.
tab2[2,2]/(tab2[2,2] + tab2[1,2])
```

```
## [1] 0.84
```

```r
# Accuracy (TP + TN)/(All)
(tab2[2,2] + tab2[1,1])/(tab2[2,2] + tab2[1,1] + tab2[1,2] + tab2[2,1])
```

```
## [1] 0.9451613
```

```r
## For fun: Logistic Regression using PC6 and PC7
# How bad can it be?
model3 = glm(Sports ~ PC6 + PC7 ,data=cars04.train.data, family=binomial)
summary(model3)
```

```
##
## Call:
## glm(formula = Sports ~ PC6 + PC7, family = binomial, data = cars04.train.data)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.4536  -0.4410  -0.2798  -0.1555   2.7725
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept)  -2.8503     0.2966  -9.609  < 2e-16 ***
## PC6          -3.3181     0.5672  -5.850 4.92e-09 ***
## PC7          -1.4001     0.5189  -2.698  0.00698 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 214.42  on 309  degrees of freedom
## Residual deviance: 162.74  on 307  degrees of freedom
## AIC: 168.74
##
## Number of Fisher Scoring iterations: 6
```

```r
## Making Predictions on the train set
# Make predictions
predictTrain3 = predict(model3, type="response")
# Summary
summary(predictTrain3)
```

```
##      Min.   1st Qu.    Median      Mean   3rd Qu.      Max.
## 0.0008009 0.0214304 0.0544925 0.1096774 0.1359915 0.9224831
```

```r
tapply(predictTrain3, cars04.train.data$Sports, mean)
```

```
##          0          1
## 0.08505337 0.30956673
```

```r
# Confusion matrix for threshold of 0.5
tab3 <- table(cars04.train.data$Sports, predictTrain3 > 0.5)
tab3
```

```
##
##     FALSE TRUE
##   0   275    1
##   1    26    8
```

```r
# Sensitivity TP/(TP + FN)
tab3[2,2]/(tab3[2,2] + tab3[2,1])
```

```
## [1] 0.2352941
```

```r
# Specificity  TN/(TN + FP)
tab3[1,1]/(tab3[1,1] + tab3[1,2])
```

```
## [1] 0.9963768
```

```r
# Precision TP/predicted yes.
tab3[2,2]/(tab3[2,2] + tab3[1,2])
```

```
## [1] 0.8888889
```

```r
# Accuracy (TP + TN)/(All)
(tab3[2,2] + tab3[1,1])/(tab3[2,2] + tab3[1,1] + tab3[1,2] + tab3[2,1])
```

```
## [1] 0.9129032
```

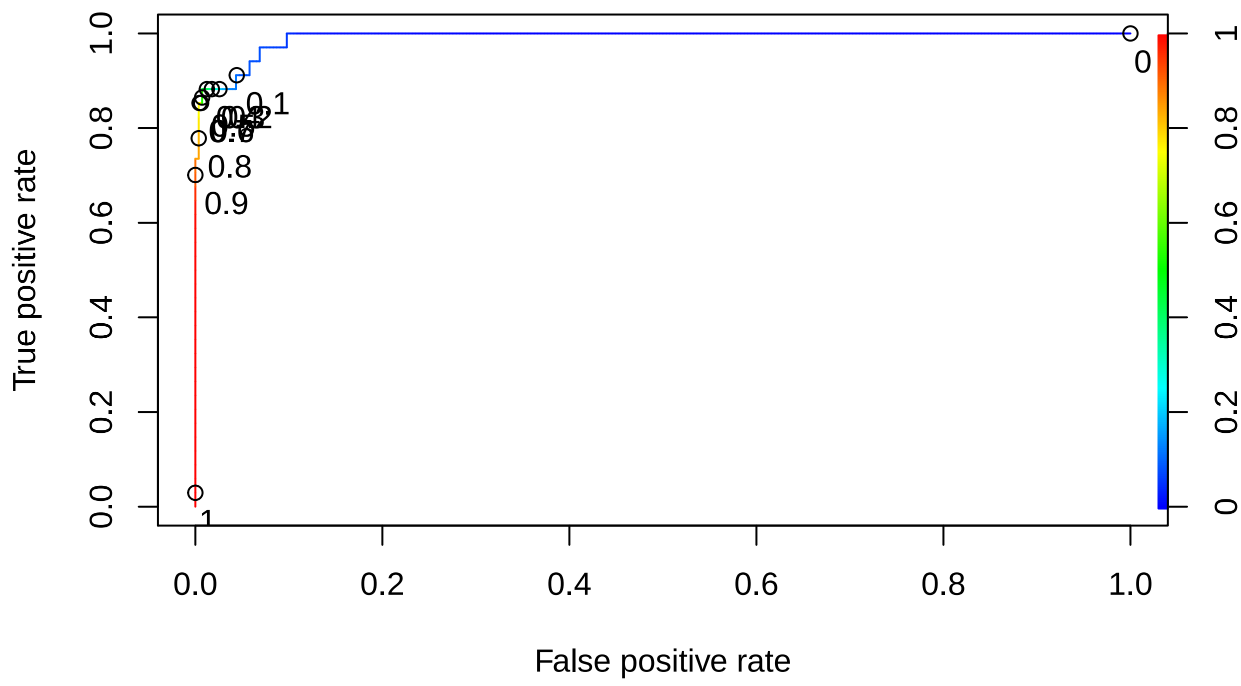
```r
# Install and load ROCR package
library(ROCR)
```

```
## Loading required package: gplots
```

```
##
## Attaching package: 'gplots'
```

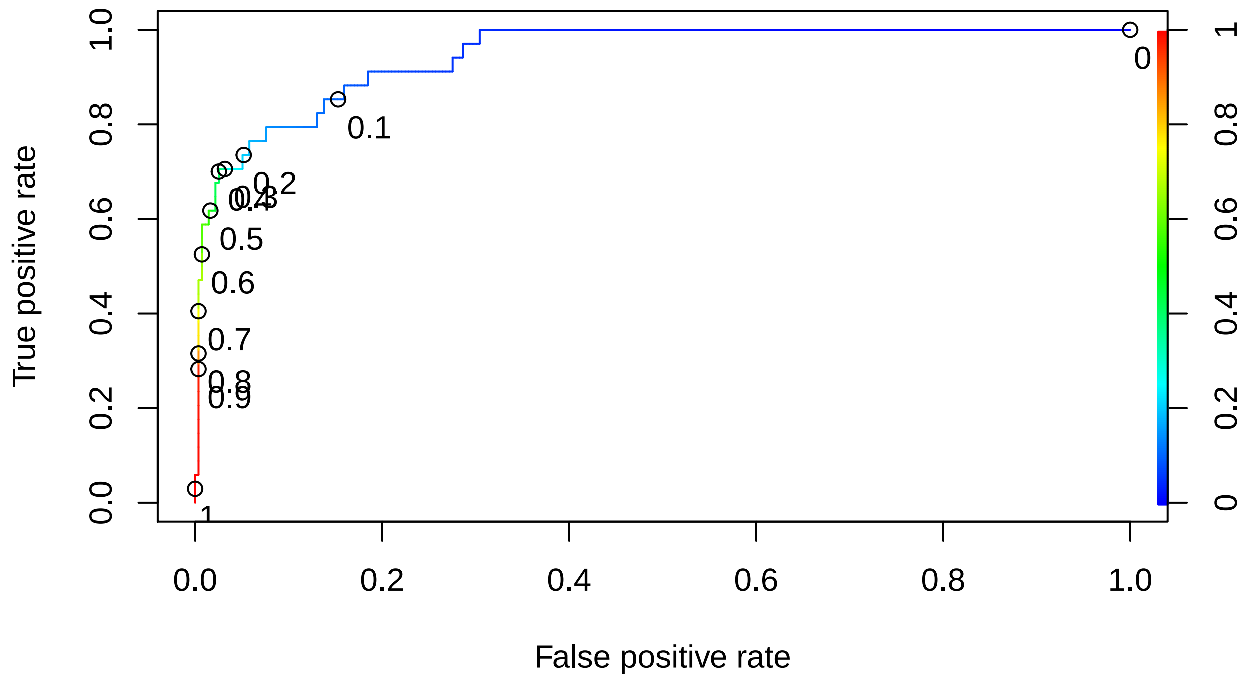
```
## The following object is masked from 'package:stats':
##
##     lowess
```

```r
# Model 1 using all initial features
# ROC predictions
ROCRpred1 = prediction(predictTrain, cars04.train.data$Sports)
# Performance function
ROCRperf1 = performance(ROCRpred1, "tpr", "fpr")
# Plot ROC curve
plot(ROCRperf1, colorize=TRUE, print.cutoffs.at=seq(0,1,by=0.1), text.adj=c(-0.2,1.7))
```




```r
# Model 2 using only PC1 and PC2
# ROC predictions
ROCRpred2 = prediction(predictTrain2, cars04.train.data$Sports)
# Performance function
ROCRperf2 = performance(ROCRpred2, "tpr", "fpr")
```

```r
# Plot ROC curve
plot(ROCRperf2, colorize=TRUE, print.cutoffs.at=seq(0,1,by=0.1), text.adj=c(-0.2,1.7))
```




```r
## Logistic Regression using PC1 and PC2
# Picking a better threshold
# Confusion matrix for threshold of 0.2
tab2new <- table(cars04.train.data$Sports, predictTrain2 > 0.2)
tab2new
```

```
##
##     FALSE TRUE
##   0   262   14
##   1     9   25
```

```r
# Sensitivity TP/(TP + FN)
tab2new[2,2]/(tab2new[2,2] + tab2new[2,1])
```

```
## [1] 0.7352941
```

```r
# Specificity  TN/(TN + FP)
tab2new[1,1]/(tab2new[1,1] + tab2new[1,2])
```

```
## [1] 0.9492754
```

```r
# Precision TP/predicted yes.
tab2new[2,2]/(tab2new[2,2] + tab2new[1,2])
```

```
## [1] 0.6410256
```

```r
# Accuracy (TP + TN)/(All)
(tab2new[2,2] + tab2new[1,1])/(tab2new[2,2] + tab2new[1,1] + tab2new[1,2] + tab2new[2,1])
```

```
## [1] 0.9258065
```

3 Kernel PCA implementation in R

```
# Kernel pca
cars04.kpca <- kpca(as.matrix(cars04[,8:18]), kernel = "rbfdot", kpar = list(sigma = 0.1),
    features = 0, th = 1e-4)
# Object
str(cars04.kpca)
```

```
## Formal class 'kpca' [package "kernlab"] with 9 slots
##   ..@ rotated : num [1:387, 1:386] -4.08e-17 1.15e-01 -1.12e-01 -2.37e-01 1.55e-01 ...
##   .. ..- attr(*, "dimnames")=List of 2
##   .. .. ..$ : chr [1:387] "Acura 3.5 RL" "Acura 3.5 RL Navigation" "Acura MDX" "Acura NSX S" ...
##   .. .. ..$ : NULL
##   ..@ pcv     : num [1:387, 1:386] 0 0.115 -0.112 -0.237 0.155 ...
##   ..@ eig     : Named num [1:386] 0.00258 0.00258 0.00258 0.00258 0.00258 ...
##   .. ..- attr(*, "names")= chr [1:386] "Comp.1" "Comp.2" "Comp.3" "Comp.4" ...
##   ..@ kernelf :Formal class 'rbfkernel' [package "kernlab"] with 2 slots
##   .. .. ..@ .Data:function (x, y = NULL)
##   .. .. ..@ kpar :List of 1
##   .. .. .. ..$ sigma: num 0.1
##   ..@ kpar    : list()
##   ..@ xmatrix : num [1:387, 1:11] 43755 46100 36945 89765 23820 ...
##   .. ..- attr(*, "dimnames")=List of 2
##   .. .. ..$ : chr [1:387] "Acura 3.5 RL" "Acura 3.5 RL Navigation" "Acura MDX" "Acura NSX S" ...
##   .. .. ..$ : chr [1:11] "Retail" "Dealer" "Engine" "Cylinders" ...
##   ..@ kcall   : language .local(x = x, kernel = "rbfdot", kpar = ..2, features = 0, th = 1e-04)
##   ..@ terms   : NULL
##   ..@ n.action: NULL
```

4 Application in Lundbeck

4.1 Already existing applications

Here are two examples were PCA was used in Lu, Biometrics.

- Example 1

PCA together with factor analysis was used by Anne (HEE) to identify a separate "brightening" dimension based on placebo-controlled trials, primarily in schizophrenia via PANSS, but also in adjunct MDD using the IDS-SR scale. The lists of the believed "brightening" items of the PANSS and the IDS-SR were provided to Anne. A PCA was performed on the PANSS item data, using Kaiser criterion, 7 principal components were retained. The brightening items were found to cluster when plotting the loadings of the items on the two first principal components. Similar findings were found in MDD using the IDS-SR scale.

- Example 2

PCA was used in the trial complexity model of the SiteIQ tool. A term frequency inverse document frequency (TFIDF) was first used on the protocol documents. The TFIDF produces a table with the 100,000 words most used in the protocol document with their frequency in the document reweighted by their prevalence. A PCA was then applied to identify colinearities between the words and reduce the dimension to 20 topics (linear combination of the initial words). A random forest was then applied to predict the average number of enrolled patients per site per month for a given trial given its vector of trial features (principal components) derived from the protocol synopsis. The model was trained on the Informa/Citeline data and used for prediction on Lundbeck Data

4.2 New applications?