PCA Revealed

Part 8: Hacking your own PCA

Gaston **S**anchez

August 2014

Readme

License:

You are free to:

Share — copy and redistribute the material

Adapt — rebuild and transform the material

Under the following conditions:

Attribution — You must give appropriate credit, provide a link to the license, and indicate if changes were made.

NonCommercial — You may not use this work for commercial purposes.

ShareAlike — If you remix, transform, or build upon this work, you must distribute your contributions under the same license to this one.

Reminder

PCA

Principal Components Analysis (PCA) allows us to study and explore a set of quantitative variables measured on a set of objects

Core Idea

With PCA we seek to reduce the dimensionality (reduce the number of variables) of a data set while retaining as much as possible of the variation present in the data

Presentation

About

In these slides we'll briefly describe how you could implement your own PCA function in R.

Keep in mind

There are several ways in which the results of PCA can be obtained. We'll cover some of the simplest and easiet ways to do that.

Data considerations

Data Structure

Data

The analyzed data takes the form of a table (i.e. `matrix`) $\mathbf{X}$:

$$\mathbf{X}_{n,p} = \begin{bmatrix} x_{11} & x_{12} & \cdots & x_{1p} \\ x_{21} & x_{22} & \cdots & x_{2p} \\ \vdots & \vdots & \ddots & \vdots \\ x_{n1} & x_{n2} & \cdots & x_{np} \end{bmatrix}$$

- n objects in the rows
- p quantitative variables in the columns

Data Considerations

Normalization

In practice, there are two main flavors in which PCA can be performed:

- PCA on standardized data (normalized variables)
- PCA on raw data (non-normalized variables)

Unless the analyzed variables are all measured in the same scale, it is strongly recommended to normalize all the variables *(in this way we are comparing apples to apples)*

Data Considerations

Useful Arguments

To implement your own script or function in R for PCA, it's recommended to have arguments for both mean-centering and normalizing the variables

Mean Centered Variables

There are several ways in R to perform mean-centering and normalizing operations. For sake of simplicity, we'll use the function `scale()`

Toy dataset
`cars2004`

Data

Download the data in R using the package `RCurl` as follows:

```
# load package RCurl
library(RCurl)

# google docs spreadsheets url
google_docs = "https://docs.google.com/spreadsheet/"

# public key of data 'cars'
cars_key = "pub?key=0AjoVnZ9iB261dHRfQlVuWDRUSHdZQ1A4N294TEstcOE&output=csv"

# download URL of data file
cars_csv = getURL(paste(google_docs, cars_key, sep = ""))

# import data in R (through a text connection)
cars2004 = read.csv(textConnection(cars_csv), row.names = 1, header = TRUE)
```


Data

Use the function `head()` to take a peek of the data contained in `cars2004`:

```
# take a peek
head(cars2004)
```

```
##                     Cylinders Horsepower Speed Weight Width Length
## Citroen C2               1124         61   158    932  1659   3666
## Smart Fortwo              698         52   135    730  1515   2500
## Mini 1.6 170             1598        170   218   1215  1690   3625
## Nissan Micra 1.2         1240         65   154    965  1660   3715
## Renault Clio 3.0 V6      2946        255   245   1400  1810   3812
## Audi A3 1.9              1896        105   187   1295  1765   4203
```

Toy Data Example: cars2004

The data consists of 24 cars measured on the following six variables:

`Cylinders`	Number of cylinders (cm^3)
`Horsepower`	Horsepower (hp)
`Speed`	Maximum speed (km/h)
`Weight`	Weight of the car (kg)
`Width`	Width of the car (mm)
`Length`	Length of the car (mm)

PCA with SVD

PCA with SVD

PCA through SVD

One approach to program PCA in R is with the Singular Value Decomposition (SVD) using the function `svd()`

PCA with the function svd()

```r
pca_svd <- function(dataset, center = TRUE, scale = TRUE) {
  # mean-center and normalization
  X = scale(dataset, center = center, scale = scale)
  # singular value decomposition
  SVD = svd(X)
  # scores
  scores = SVD$u %*% diag(SVD$d)
  rownames(scores) = rownames(dataset)
  # loadings
  loadings = SVD$v
  rownames(loadings) = colnames(dataset)
  # results
  list(
    values = SVD$d^2 / (nrow(X) - 1),
    scores = scores,
    loadings = loadings
  )
}
```


PCA with SVD

PCA with the function `svd()`:

```
# Apply it!
pca1 = pca_svd(cars2004)

# eigen values
pca1$values

## [1] 4.41127 0.85341 0.43566 0.23587 0.05144 0.01235
```


PCA with SVD

Scores:

```
# scores
head(pca1$scores, n = 5)

##                          [,1]    [,2]    [,3]     [,4]     [,5]     [,6]
## Citroen C2           -2.541258 -0.4992 -0.1754  0.16222 -0.20277  0.03095
## Smart Fortwo         -4.062768 -1.6308  0.2686 -0.90465 -0.02865 -0.03289
## Mini 1.6 170         -1.352811 -0.7985  0.3645 -0.04997  0.45427  0.04992
## Nissan Micra 1.2     -2.460421 -0.3951 -0.1701  0.12174 -0.28329  0.04972
## Renault Clio 3.0 V6  -0.003062 -0.8963  0.3770 -0.26840  0.27979 -0.13127

tail(pca1$scores, n = 5)

##                        [,1]   [,2]     [,3]    [,4]      [,5]      [,6]
## Renault Scenic 1.9  -0.8241 0.3723 -0.24040  0.1099  0.079333  0.009328
## VW Touran 1.9 TDI   -0.7880 0.6979 -0.23130  0.1306 -0.001267  0.018130
## LandRover Defender  -1.0491 0.7356 -0.17242 -1.1576 -0.300317 -0.008357
## LandRover Discovery  0.8327 1.8799 -1.48618 -0.9787  0.304380 -0.033905
## Nissan X-Trail 2.2  -0.6015 0.7071 -0.04458  0.1142 -0.170485  0.116171
```


PCA with SVD

Loadings:

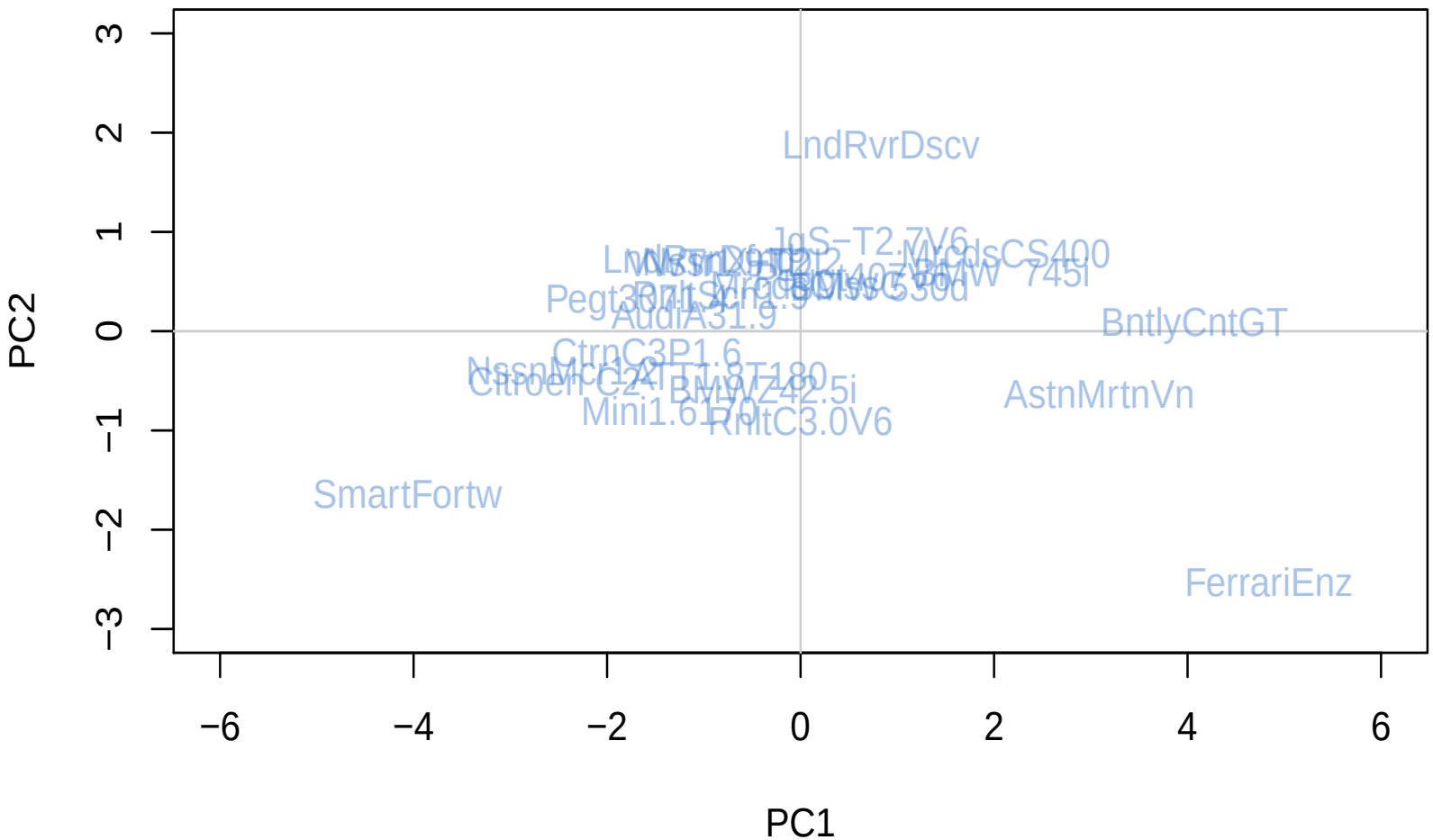
```
# loadings
head(pca1$loadings)

##                [,1]    [,2]    [,3]    [,4]     [,5]      [,6]
## Cylinders   0.4582 -0.1374  0.2144 -0.2321 -0.65250 -0.495560
## Horsepower 0.4396 -0.3817  0.1369 -0.1728 -0.09445  0.776846
## Speed       0.4219 -0.3667  0.3126  0.4100  0.56864 -0.313684
## Weight      0.3604  0.6232  0.2234 -0.5294  0.38924 -0.007511
## Width       0.3815 -0.1202 -0.8830 -0.1399  0.15302 -0.131875
## Length      0.3786  0.5460 -0.0897  0.6696 -0.25894  0.187311
```


Plot of scores

```
# plot of scores
plot(pca1$scores[,1], pca1$scores[,2], type = "n",
     xlab = "PC1", ylab = "PC2",
     xlim = c(-6, 6), ylim = c(-3, 3))
abline(h = 0, v = 0, col = "gray80")
text(pca1$scores[,1], pca1$scores[,2],
     labels = abbreviate(rownames(pca1$scores), 10),
     col = "#4380d377" , xpd = TRUE)
```

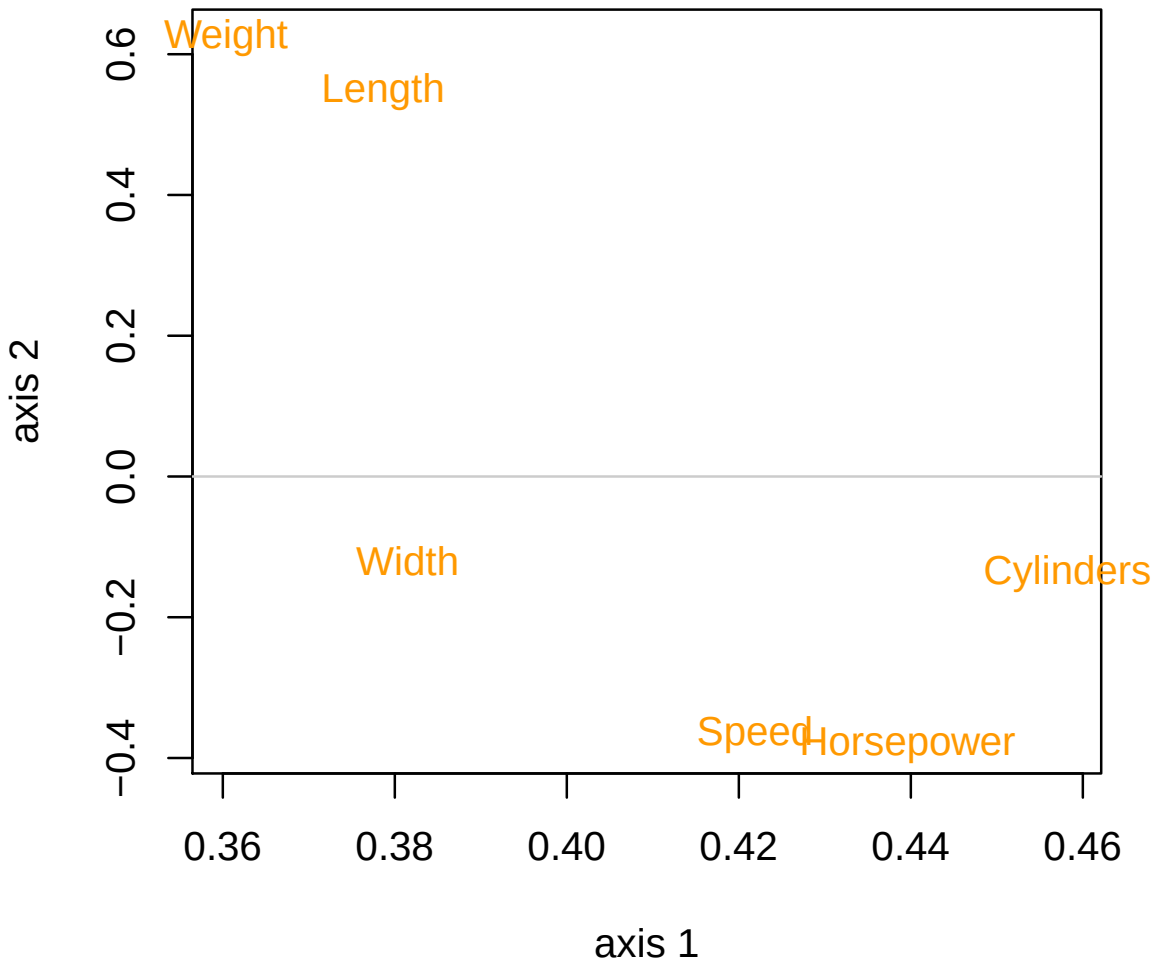

Plot of scores



Plot of loadings

```
# plot of loadings
plot(pca1$loadings[,1], pca1$loadings[,2], type = "n",
     xlab = "axis 1", ylab = "axis 2")
abline(h = 0, v = 0, col = "gray80")
text(pca1$loadings[,1], pca1$loadings[,2],
     labels = rownames(pca1$loadings, 10),
     col = "#ff9900" , xpd = TRUE)
```


Plot of loadings



PCA with EVD

PCA with EVD

PCA through EVD

Another approach to program PCA in R is with the EigenValue Decomposition (EVD) using the function `eigen()`

PCA with EVD

```r
pca_evd <- function(dataset, center = TRUE, scale = TRUE) {
  # mean-center and normalization
  X = scale(dataset, center = center, scale = scale)
  # eigenvalue decomposition
  if (nrow(X) >= ncol(X)) {
    EVD = eigen(t(X) %*% X)
  } else {
    EVD = eigen(X %*% t(X))
  }
  # scores
  scores = X %*% EVD$vectors
  rownames(scores) = rownames(dataset)
  # loadings
  loadings = EVD$vectors
  rownames(loadings) = colnames(dataset)
  # results
  list(
    values = EVD$values / (nrow(X) - 1),
    scores = scores,
    loadings = loadings
  )
}
```


PCA with EVD

PCA with the function `eigen()`:

```
# Apply it!
pca2 = pca_evd(cars2004)

# eigen values
pca2$values

## [1] 4.41127 0.85341 0.43566 0.23587 0.05144 0.01235
```


PCA with EVD

Scores:

```
# scores
head(pca2$scores, n = 5)

##                          [,1]    [,2]    [,3]     [,4]     [,5]     [,6]
## Citroen C2           2.541258 -0.4992  0.1754  0.16222  0.20277  0.03095
## Smart Fortwo         4.062768 -1.6308 -0.2686 -0.90465  0.02865 -0.03289
## Mini 1.6 170         1.352811 -0.7985 -0.3645 -0.04997 -0.45427  0.04992
## Nissan Micra 1.2     2.460421 -0.3951  0.1701  0.12174  0.28329  0.04972
## Renault Clio 3.0 V6  0.003062 -0.8963 -0.3770 -0.26840 -0.27979 -0.13127

tail(pca2$scores, n = 5)

##                          [,1]   [,2]    [,3]    [,4]      [,5]      [,6]
## Renault Scenic 1.9     0.8241 0.3723 0.24040  0.1099 -0.079333  0.009328
## VW Touran 1.9 TDI      0.7880 0.6979 0.23130  0.1306  0.001267  0.018130
## LandRover Defender     1.0491 0.7356 0.17242 -1.1576  0.300317 -0.008357
## LandRover Discovery   -0.8327 1.8799 1.48618 -0.9787 -0.304380 -0.033905
## Nissan X-Trail 2.2     0.6015 0.7071 0.04458  0.1142  0.170485  0.116171
```

PCA with EVD

Loadings:

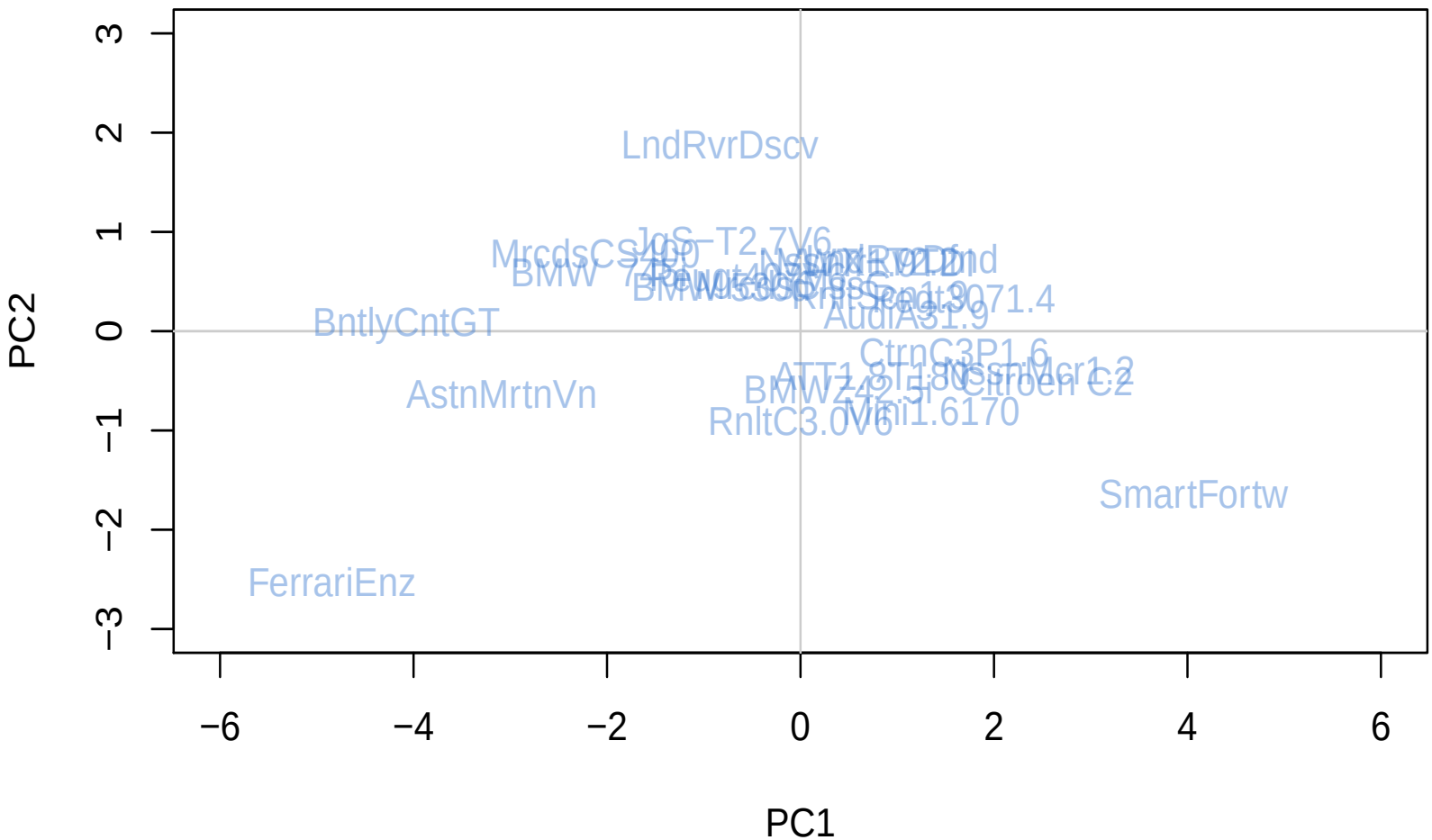
```
# scores
head(pca2$loadings, n = 5)

##                [,1]    [,2]    [,3]    [,4]     [,5]      [,6]
## Cylinders  -0.4582 -0.1374 -0.2144 -0.2321  0.65250 -0.495560
## Horsepower -0.4396 -0.3817 -0.1369 -0.1728  0.09445  0.776846
## Speed      -0.4219 -0.3667 -0.3126  0.4100 -0.56864 -0.313684
## Weight     -0.3604  0.6232 -0.2234 -0.5294 -0.38924 -0.007511
## Width      -0.3815 -0.1202  0.8830 -0.1399 -0.15302 -0.131875
```


Plot of scores

```
# plot of scores
plot(pca2$scores[,1], pca2$scores[,2], type = "n",
     xlab = "PC1", ylab = "PC2",
     xlim = c(-6, 6), ylim = c(-3, 3))
abline(h = 0, v = 0, col = "gray80")
text(pca2$scores[,1], pca2$scores[,2],
     labels = abbreviate(rownames(pca2$scores), 10),
     col = "#4380d377" , xpd = TRUE)
```


Plot of scores



Plot of loadings

```
# plot of loadings
plot(pca2$loadings[,1], pca2$loadings[,2], type = "n",
     xlab = "axis 1", ylab = "axis 2")
abline(h = 0, v = 0, col = "gray80")
text(pca2$loadings[,1], pca2$loadings[,2],
     labels = rownames(pca2$loadings, 10),
     col = "#ff9900" , xpd = TRUE)
```


Plot of loadings