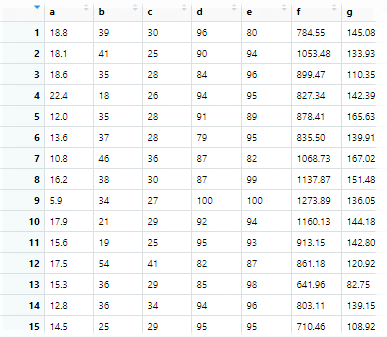
# Principal Components Analysis

## Abstract

This contribution develops an analysis on the dataset ‘bad-drivers.csv’ (taken from github.com/fivethirtyeight/data) by running the Principal Components Analysis. In particular, the study is focused on the different steps in the computation of the principal componets.  
1.How to choose Principal Components 2.Proportion Variance Explained 3.Thanks to this it is now possible to consider just three of the principal components, always keeping in mind that the goal is to lower the cross-validation risk estimate. #Description of the dataset I chose the dataset ‘bad-drivers.csv’ (taken from github.com/fivethirtyeight/data), that considers data whose sources are: National Highway Traffic Safety Administration and National Association of Insurance Commissioners. This folder contains data behind the story ‘Dear Mona, Which State Has The Worst Drivers?’, where Mona Chalabi, a British data journalist answer some questions about drivers and car accidents in the USA. She says that historic data that could indicate where America’s worst drivers are: the number of car crashes in each state (especially those where the driver was negligent in some way), how much insurance companies pay out, and how much insurance companies charge drivers. The reported variables are: . State (a specific country in the USA); . Number of drivers involved in fatal collisions per billion miles; . Percentage of drivers involved in fatal collisions who were speeding; . Percentage of drivers involved in fatal collisions who were alcohol impaired; . Percentage of drivers involved in fatal collisions who were not distracted; . Percentage of drivers involved in fatal collisions who had not been involved in any previous accidents; . Car insurance premiums; . Losses incurred by insurance companies for collisions per insured driver. I chose to compute the predictions on the variable ‘Losses incurred by insurance companies for collisions per insured driver’ (variable y).

## Analysis

The Principal Component Analysis is one of the unsupervised learning methods and, as all unsupervised methods, it is much more challenging than supervised learning: in fact, it is often performed as an exploratory data analysis, whose goal is to check if there exist a possible dimension of the dataset in which I can study my data and see the relation between the variables. My goal here is to find a low-dimension representation of the dataset by defining the principal components. First of all, I imported the dataset, renamed all the columns and chose not to consider the first column of the dataset in order to make it easier to do the whole analysis. These are the first fifteen rows as example:



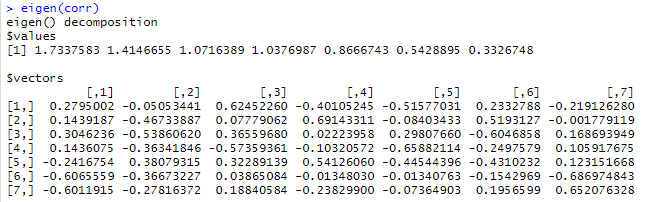
Then my analysis starts with the definition of the number of rows and columns, since they are useful for the next steps. In this case the number of rows (n) is 51 and the number of column (p) is 7. Two other elements needed are the mean and the standard deviation for each variable reported in the dataset. I built a table to summarize the means and the standard deviations; the output is the following: | | mean | std | |-|——–|——|  
|a| 15.79 | 4.12| |b| 31.73 | 9.63| |c| 30.69 | 5.13| |d| 85.92 | 15.16| |e| 88.73 | 6.96| |f| 886.96 |178.30| |g| 134.49 | 24.84|

Based on this, I decided to run the analysis by starting from the correlation matrix (the alternative is the covariance matrix) that is the following:

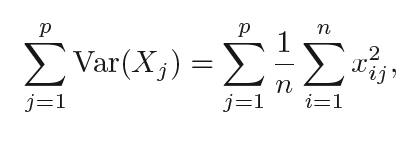
|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  | a | b | c | d | e | f | g |
| a | 1.000 | -0.029 | 0.199 | 0.010 | -0.018 | -0.200 | -0.036 |
| b | -0.029 | 1.000 | 0.286 | 0.132 | 0.014 | 0.043 | -0.061 |
| c | 0.199 | 0.286 | 1.000 | 0.043 | -0.245 | -0.017 | -0.084 |
| d | 0.010 | 0.132 | 0.043 | 1.000 | -0.195 | 0.020 | -0.058 |
| e | -0.018 | 0.014 | -0.245 | -0.195 | 1.000 | 0.076 | 0.043 |
| f | -0.200 | 0.043 | -0.017 | 0.020 | 0.076 | 1.000 | 0.623 |
| g | -0.036 | -0.061 | -0.084 | -0.058 | 0.043 | 0.623 | 1.000 |

The absolute values are between 0 and 1 and it is possible to get the following information: -if the correlation is equal to 1 then the relationship is linear -in case the value is close to 0, it is non-linear -if both values tend to increase or decrease together the coefficient is positive, and the line that represents the correlation slopes upward, otherwise the coefficient is negative. In this case, most of the values have non-linear relationship.

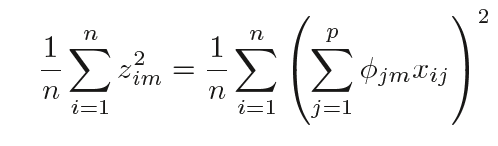
Then, there is a very important step: the choice of the principal components. It canbe done in different ways, the first is the definition of the eigenvalues and eigenvectors. The eigenvalues and eigenvectors of a correlation (or covariance) matrix represent the “core” of a PCA: the eigenvectors (principal components) represent the directions of the new feature space, whereas the eigenvalues determine their magnitude and, consequently explain the variance of the data along the new feature axes. The R function ‘eigen()’ outputs the eigenvalues and eigenvectors:



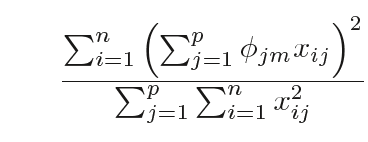
The eigenvector with the highest eigenvalues is then considered as principal components in most cases.In this case, the first four components are able to axplain at least one variable by themselves. However it is possible to base the choice of the principal components on the proportion of variance explained, which indicates how much of the variance in the data is not contained in the first few principal components. The total variance present in a dataset is defined as:



and the variance explained by m-th principal component by:

.

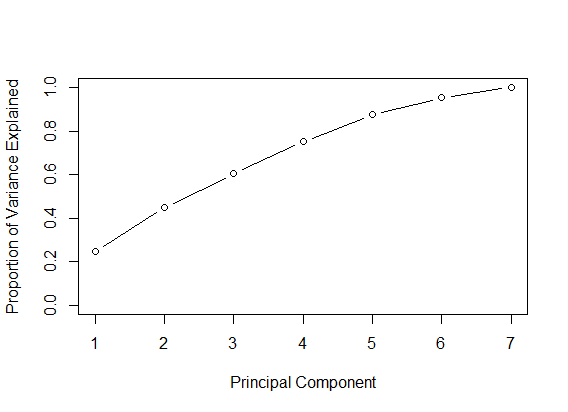
Then, the PVE of the m-th component is given by:

.

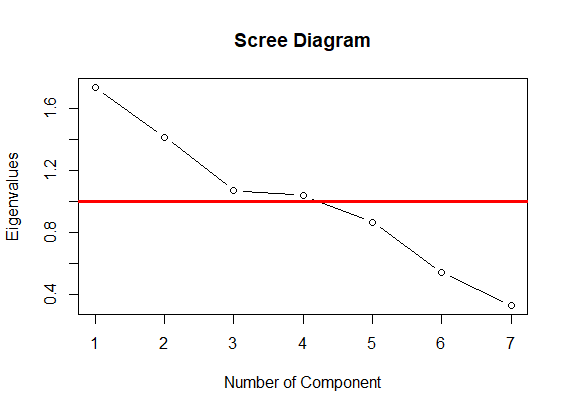
Therefore I computed the PVE, by considering that it is equal to the ratio between the eigenvalues and the number of columns, and the cumulative variance explained.The following table shows the eigenvalues, the pve and the cumulative pve:

|  |  |  |  |
| --- | --- | --- | --- |
|  | eigenvelues | % variance | % cum variance |
| [1,] | 1.73 | 24.77 | 24.77 |
| [2,] | 1.41 | 20.21 | 44.98 |
| [3,] | 1.07 | 15.31 | 60.29 |
| [4,] | 1.04 | 14.82 | 75.11 |
| [5,] | 0.87 | 12.38 | 87.49 |
| [6,] | 0.54 | 7.76 | 95.25 |
| [7,] | 0.33 | 4.75 | 100.00 |

The cumulative proportion of explained variance is represented by the following graph:



At this stage, it is possible to visually define the principal components by using the Scree Plot. In fact, I chose the components located above the red line in the following diagram, where the components are on x and the eigenvalues on y, that indicates values greater than 1:



Based on this, I chose the first four components, confirming the counclusions done on the output of the ‘eigen()’ function. I built the matrix of the four components, obtained by multiplying the eigenvector by the root of the respective eigenvalue.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Comp1 | Comp2 | Comp3 | Comp4 |
| a | -0.36802 | 0.06011 | -0.64651 | -0.63619 |
| b | -0.18950 | 0.55585 | -0.08053 | -0.07924 |
| c | -0.40110 | 0.64062 | -0.37847 | -0.37242 |
| d | -0.18909 | 0.43225 | 0.59378 | 0.58431 |
| e | 0.31822 | -0.45291 | -0.33426 | -0.32892 |
| f | 0.79867 | 0.43619 | -0.04001 | -0.03937 |
| g | 0.79160 | 0.33085 | -0.19504 | -0.19192 |

This table shows the correlations of the original variables with each of the four components. For example the variables ‘f’ and ‘g’ have a high correlation with the first component (79%), ‘b’ and ‘c’ have a relatively high correlation with the second one (respectively 55% and 64%). Another element that can be considered is the communality that tells how is the part of variance explained for each variable after the reduction. Using four components, I get the following results:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | Comp1 | Comp2 | Comp3 | Comp4 | communality |
| a | -0.36802 | 0.06011 | -0.64651 | -0.63619 | 0.9617648 |
| b | -0.18950 | 0.55585 | -0.08053 | -0.07924 | 0.3576435 |
| c | -0.40110 | 0.64062 | -0.37847 | -0.37242 | 0.8532114 |
| d | -0.18909 | 0.43225 | 0.59378 | 0.58431 | 0.9165880 |
| e | 0.31822 | -0.45291 | -0.33426 | -0.32892 | 0.5263096 |
| f | 0.79867 | 0.43619 | -0.04001 | -0.03937 | 0.8312863 |
| g | 0.79160 | 0.33085 | -0.19504 | -0.19192 | 0.8109662 |

In other words, I get: -96% of the variance of ‘a’(Number of drivers involved in fatal collisions per billion miles) explained, -35% of the variance of ‘b’ (Percentage of drivers involved in fatal collisions who were speeding) explained, -85% of the variance of ‘c’ (Percentage of drivers involved in fatal collisions who were alcohol impaired) explained, -91% of the variance of ‘d’ (Percentage of drivers involved in fatal collisions who were not distracted) explained, -52% of the variance of ‘e’ (Percentage of drivers involved in fatal collisions who had not been involved in any previous accidents) explained, -83% of the variance of ‘f’ (Car insurance premiums) explained, -81% of the variance of ‘g’ ( Losses incurred by insurance companies for collisions per insured driver) explained.

### principal component analysis ###  
bad\_drivers <- read.csv("https://raw.githubusercontent.com/fivethirtyeight/data/master/bad-drivers/bad-drivers.csv", sep = ",")  
head(bad\_drivers)  
dir.create("data")  
save(bad\_drivers, file=file.path("data","bad\_drivers.rda"))  
a <- bad\_drivers$Number.of.drivers.involved.in.fatal.collisions.per.billion.miles  
b <- bad\_drivers$Percentage.Of.Drivers.Involved.In.Fatal.Collisions.Who.Were.Speeding  
c <- bad\_drivers$Percentage.Of.Drivers.Involved.In.Fatal.Collisions.Who.Were.Alcohol.Impaired  
d <- bad\_drivers$Percentage.Of.Drivers.Involved.In.Fatal.Collisions.Who.Were.Not.Distracted  
e <- bad\_drivers$Percentage.Of.Drivers.Involved.In.Fatal.Collisions.Who.Had.Not.Been.Involved.In.Any.Previous.Accidents  
f <- bad\_drivers$Car.Insurance.Premiums....  
g <- bad\_drivers$Losses.incurred.by.insurance.companies.for.collisions.per.insured.driver....  
  
bad\_drivers <- data.frame(a,b,c,d,e,f,g)  
bad\_drivers  
  
n <- nrow(bad\_drivers) #number of rows  
p <- ncol(bad\_drivers) #number of columns  
  
# mean and std:  
mean <- colMeans(bad\_drivers)  
std <- apply(bad\_drivers, 2, sd)  
table <- round(cbind(mean, std),2)  
table  
  
## PCA starting from correlation matrix  
# calculate matrix R:  
corr <- cor(bad\_drivers)  
round(corr,3)  
  
  
# calculate eigenvalues and eigenvectors:  
eigen(corr)  
eigenvalues <- eigen(corr)$values  
eigenvectors <- eigen(corr)$vectors  
eigenvalues  
eigenvectors  
  
#proportion of variance explained  
pve = eigenvalues/p  
pvecum = cumsum(pve)  
tab <- round(cbind(eigenvalues,pve\*100,pvecum\*100),2)  
colnames(tab)<-c("eigenvelues", "% variance","% cum variance")  
tab  
plot(pvecum, xlab="Principal Component", ylab="Proportion of Variance Explained", ylim=c(0,1),type='b')  
  
# Use Scree Diagram to select the components:  
plot(eigenvalues, type="b", main="Scree Diagram", xlab="Number of Component", ylab="Eigenvalues")  
abline(h=1, lwd=3, col="red")  
  
#Select 4 components   
eigen(corr)$vectors[,1:4]  
  
  
#Matrix of the components, obtained by multiplying the eigenvector by the root of the respective eigenvalue (if necessary we can change the sign for interpretative reasons)  
comp <- round(cbind(-eigen(rho)$vectors[,1]\*sqrt(eigenvalues[1]),-eigen(rho)$vectors[,2]\*sqrt(eigenvalues[2]),-eigen(rho)$vectors[,3]\*sqrt(eigenvalues[3]),-eigen(rho)$vectors[,3]\*sqrt(eigenvalues[4])),5)  
rownames(comp)<-row.names(table)  
colnames(comp)<-c("Comp1","Comp2","Comp3","Comp4")  
comp  
  
# The sum of the squares of the values of each row of the component   
#matrix is the respective 'communality',   
# The communality is the sum of the squared component loadings   
#up to the number of components you extract.  
communality<-comp[,1]^2+comp[,2]^2+comp[,3]^2+comp[,4]^2  
comp<-cbind(comp,communality)  
comp  
  
  
# Calculate the scores for the selected components and graph them:  
bad\_drivers.scale <- scale(bad\_drivers, T, T)  
score <- bad\_drivers.scale%\*%autovec[,1:4]  
# normalized scores changed sign (non-normalized scores divided by   
#square root of the respective eigenvalue)  
## score chart  
scorez<-round(cbind(-score[,1]/sqrt(eigenvalues[1]),-score[,2]/sqrt(eigenvalues[2]),-score[,3]/sqrt(eigenvalues[3]),-score[,4]/sqrt(eigenvalues[4])),4)  
plot(scorez, main="Scores plot")  
text(scorez, rownames(bad\_drivers))  
abline(v=0,h=0,col="red")  
# Loadings plot  
plot(comp[,1:4], main="Loadings plot", xlim=range(-1,1))  
text(comp, rownames(comp))  
abline(v=0,h=0,col="red")