Practice 2: Principal Components Analysis

Danae Carreras Garcia

28 de agosto de 2019

Principal Components Basics

The principal components analysis is a statistical technique to reduce dimensionality. The objective is to replace a large number of correlated variables to each other by a smaller number uncorrelated, while retaining as much information as possible from the original model. This is done in practice by following these steps

1. Center and scale the data.
2. Calculate the covariance matrix.
3. Calculate the eigenvectors and eigenvalues of the covariance matrix.
4. Choose the principal components.
5. Compute the new dataset.

Exercise 1

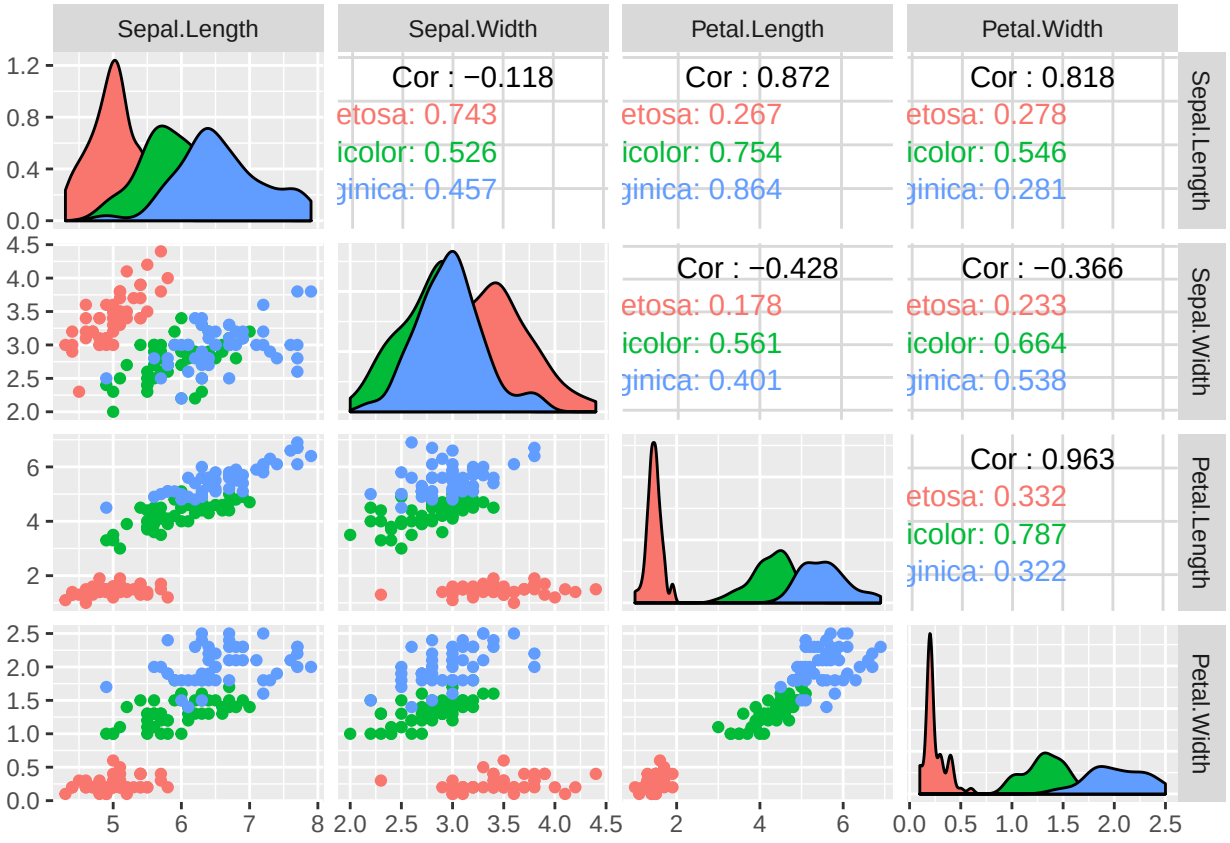
The *Iris* data set consists of 50 samples from each of three species of Iris (Iris setosa, Iris virginica and Iris versicolor). Four features were measured from each sample: the length and the width of the sepals and petals, in centimeters. Based on the combination of these four features, Fisher developed a linear discriminant model to distinguish the species from each other. Compute and visualize the principal components.

```
data = iris[,1:4]
head(data)
```

```
##   Sepal.Length Sepal.Width Petal.Length Petal.Width
## 1          5.1         3.5          1.4         0.2
## 2          4.9         3.0          1.4         0.2
## 3          4.7         3.2          1.3         0.2
## 4          4.6         3.1          1.5         0.2
## 5          5.0         3.6          1.4         0.2
## 6          5.4         3.9          1.7         0.4
```

It can be checked that the variables are highly correlated, then applying PCA can filter out the noise from the existing correlation.

```
library(ggplot2)
library(GGally)
ggpairs(iris, columns = 1:4, aes(color = Species))
```



1. Center and scale the data

One problem with principal components is that they are not scale-invariant because if we change the units of the variables, the covariance matrix of the transformed variables will also change. Additionally, if there are large difference between the variances of the original variables, then those whose variances are largest will tend to dominate the early components. In these circumstances, it is better to standardize the variables.

```
data.scaled = scale(data, center = T, scale = T)
```

2. Calculate the covariance/correlation matrix

```
Sigma = cov(data.scaled)
round(Sigma,2)
```

```
##              Sepal.Length Sepal.Width Petal.Length Petal.Width
## Sepal.Length         1.00       -0.12         0.87        0.82
## Sepal.Width         -0.12        1.00        -0.43       -0.37
## Petal.Length         0.87       -0.43         1.00        0.96
## Petal.Width          0.82       -0.37         0.96        1.00
```

Also works

```
Sigma_ = t(data.scaled)%*%data.scaled/(nrow(data.scaled)-1)
round(Sigma_,2)
```

```
##              Sepal.Length Sepal.Width Petal.Length Petal.Width
## Sepal.Length         1.00       -0.12         0.87        0.82
## Sepal.Width         -0.12        1.00        -0.43       -0.37
```

```
## Petal.Length         0.87       -0.43         1.00        0.96
## Petal.Width          0.82       -0.37         0.96        1.00
```

Remarks:

- Correlation and covariance matrix match.
- Diagonal elements are the variances of each variable.
- Off diagonal are the covariances between variables.

3. Calculate the eigenvectors and eigenvalues of the covariance matrix

```
Eigen = eigen(Sigma)
Eigenvalues = Eigen$values
Eigenvalues
```

```
## [1] 2.91849782 0.91403047 0.14675688 0.02071484
```

The sum of the variance of the original variables is equal to the sum of the variances of the principal components.

The proportion of the variance explained by the i-th principal component is given by

```
Prop.Var = Eigenvalues/sum(Eigenvalues)
Prop.Var
```

```
## [1] 0.729624454 0.228507618 0.036689219 0.005178709
```

The proportion explained by the first k principal components is given by

```
Cum.Var = cumsum(Eigenvalues)/sum(Eigenvalues)
Cum.Var
```

```
## [1] 0.7296245 0.9581321 0.9948213 1.0000000
```

```
Eigenvectors = Eigen$vectors
colnames(Eigenvectors) = c("PC1","PC2", "PC3","PC4")
rownames(Eigenvectors) = colnames(data)
Eigenvectors
```

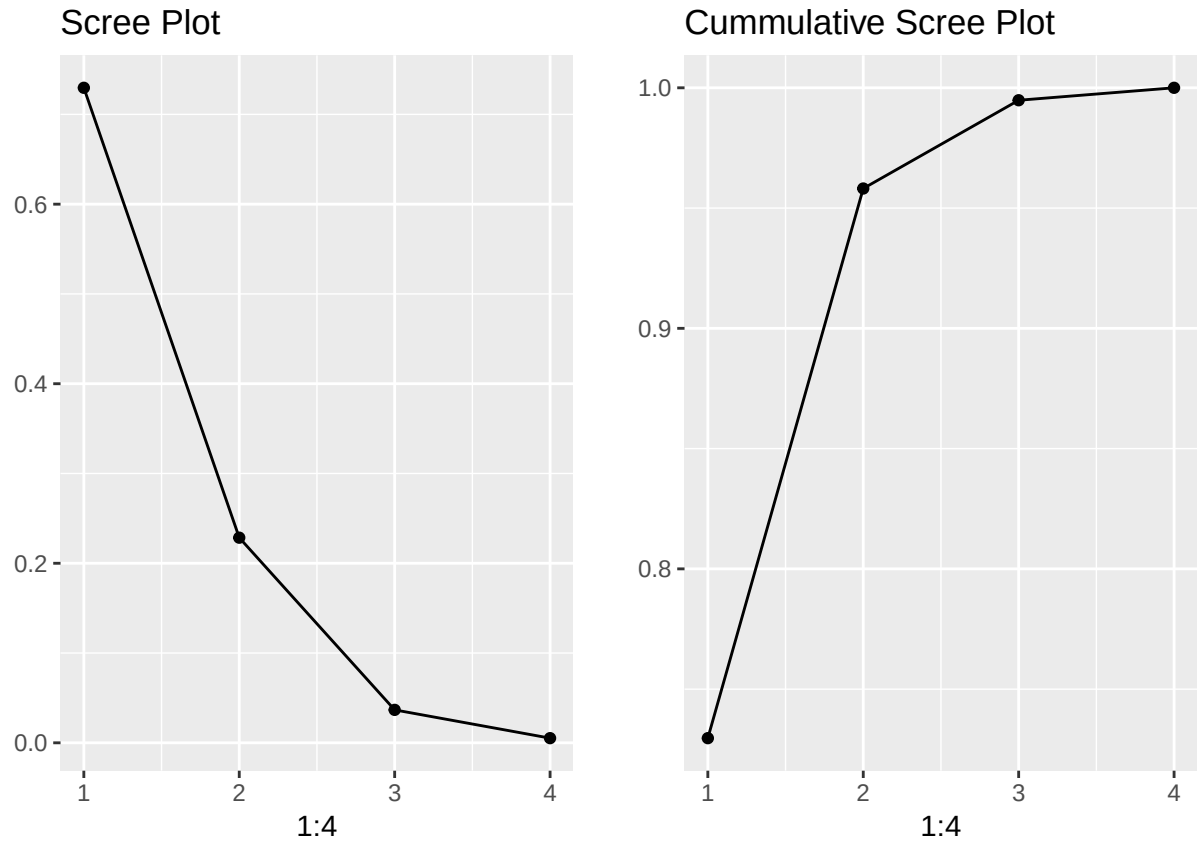
```
##                     PC1         PC2        PC3        PC4
## Sepal.Length  0.5210659 -0.37741762  0.7195664  0.2612863
## Sepal.Width  -0.2693474 -0.92329566 -0.2443818 -0.1235096
## Petal.Length  0.5804131 -0.02449161 -0.1421264 -0.8014492
## Petal.Width   0.5648565 -0.06694199 -0.6342727  0.5235971
```

4.Choose the principal components

For a general $n \times p$ data matrix, there are up to $\min(n-1, p)$ principal components that can be calculated. However, bacause the point of PCA is to significantly reduce the number of variables, we want to use the smallest number of principal components possible to explain most of the variability. There is no robust method for determining how many components to use. As the number of observations, the number of variables, and the application vary, a different level of accuracy and variable reduction are desirable. The most common technique for determining how many principal components to kep is eyeballinng the scree plot. To determine the number of components, we look for the "elbow point", where the proportion of the variance explained significantly drops off.

```
library(cowplot)
scree = ggplot(as.data.frame(Prop.Var),aes(x=1:4,y=Prop.Var))+
  geom_line(stat = "identity")+geom_point()+labs(title = "Scree Plot")+ ylab("")
```

```
cum.scree = ggplot(as.data.frame(Cum.Var),aes(x=1:4,y=Cum.Var))+
  geom_line(stat = "identity")+geom_point()+labs(title = "Cummulative Scree Plot")+ ylab("")
plot_grid(scree, cum.scree)
```



The first two principal components explain more than 95 $% of the variance.

5.Compute the new dataset

```
projection = as.data.frame(t(t(Eigenvectors)%*%t(data.scaled)))
head(projection)
```

```
##         PC1        PC2         PC3          PC4
## 1 -2.257141 -0.4784238  0.12727962  0.024087508
## 2 -2.074013  0.6718827  0.23382552  0.102662845
## 3 -2.356335  0.3407664 -0.04405390  0.028282305
## 4 -2.291707  0.5953999 -0.09098530 -0.065735340
## 5 -2.381863 -0.6446757 -0.01568565 -0.035802870
## 6 -2.068701 -1.4842053 -0.02687825  0.006586116
```

As it was decided that only the first two main components will be used, the new database includes only the first two columns.

In practice it is calculated taking into account only the chosen principal components.

```
data.new = as.data.frame(t(t(Eigenvectors[,1:2])%*%t(data.scaled)))
head(data.new)
```

```
##         PC1        PC2
```

```
## 1 -2.257141 -0.4784238
## 2 -2.074013  0.6718827
## 3 -2.356335  0.3407664
## 4 -2.291707  0.5953999
## 5 -2.381863 -0.6446757
## 6 -2.068701 -1.4842053
```

Remark: In the new data the variables are uncorrelated.

```
cov(data.new)
```

```
##              PC1          PC2
## PC1 2.918498e+00 1.609297e-15
## PC2 1.609297e-15 9.140305e-01
```

Now we can plot the first two principal components

```
ggplot(data.new, aes(x=PC1, y=PC2, color=iris$Species))+
  geom_point(size=2, alpha=0.5)+labs(color="Species")
```