Principal Component Analysis in R using the Iris data set.

Principal Component Analysis is a method used for extracting features by combining input variables in a particular manner, which is constructing a small number of features as linear combinations of the original ones. By using PCA we discard the least significant variables while retaining the most crucial components of all the variables and capture the greater information of the data set.

Furthermore, after PCA, every new variable is entirely independent of the others, which is an added advantage. To preserve most of the variability among data, PCA projects data onto a lower-dimensional subspace.

Consider the scenario where one needs to capture an image of an individual's face, and seeks to determine the most informative pose. An en face photo would not provide insight into the size of the subject's nose, while a side portrait would obscure the remaining features of the face. It is posited that utilizing PCA in facial recognition would result in a 3/4 face photo being optimal.

Although physical object manipulation is a simplistic means of comprehending the workings of PCA, in reality PCA is a mathematical technique that operates on the data itself.

In PCA, the goal is to find a set of new orthogonal axes, called principal components, such that when the data is projected onto these new axes, the maximum amount of variance is retained. The first principal component is found in the direction of the greatest variance, the second principal component is in the direction of the next greatest variance, and so on.

To find these new axes, PCA performs a mathematical transformation on the original data that involves centering the data (subtracting the mean from each variable) and then calculating the covariance matrix. This covariance matrix captures the relationships between the variables and shows how much they vary together.

The next step is to find the eigenvectors of this covariance matrix. Eigenvectors are directions in which the data varies the most, and they are orthogonal to each other. These eigenvectors define the new coordinate system in which the data is represented.

The corresponding eigenvalues of these eigenvectors indicate the amount of variance in the data that is captured by each principal component.

Therefore, PCA rotates the coordinate system in the Euclidean space such that the new axes align with the directions in which the data varies the most.

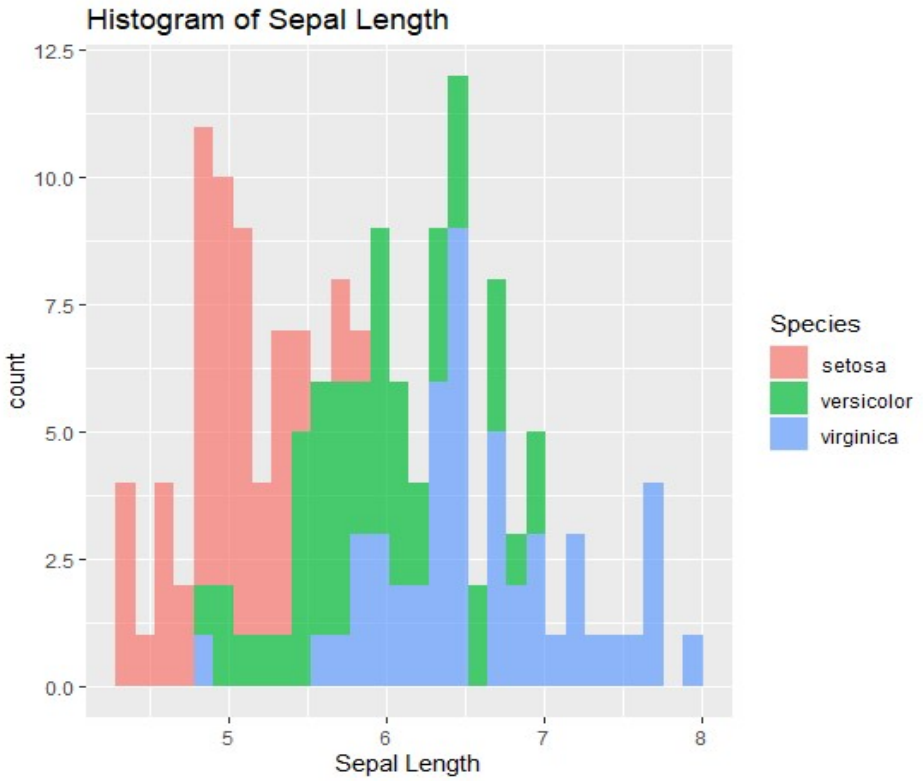
By rotating the coordinate system in this way, we can identify the underlying structure and patterns in the data, and reduce the dimensionality of the data while retaining most of the information.

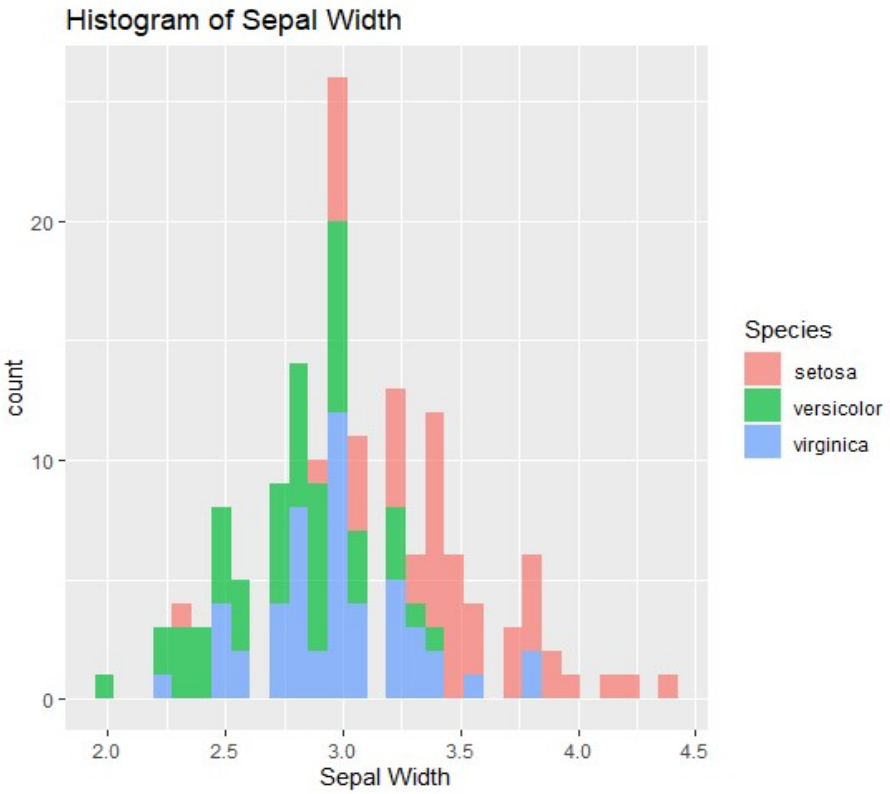
What is more we need to keep in mind a few things about the data:

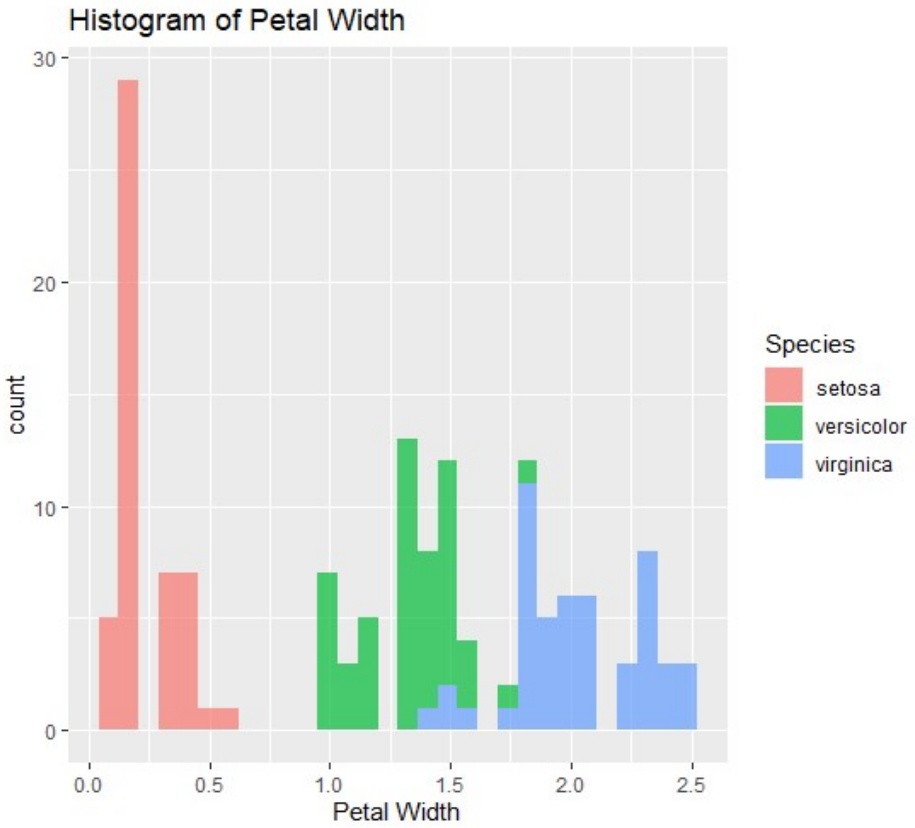
1)Scale of the variables: PCA is sensitive to the scale of the variables, so it's important to standardize or normalize the data before performing PCA.
2)Linearity: PCA assumes that the relationship between variables is linear. If there are non-linear relationships between variables, PCA may not be the best technique to use.
3)Outliers: Outliers can have a significant impact on PCA results, as they can dominate the variance and skew the principal components.
4)Correlation: PCA works best when there are correlations between variables.
5)Missing data: PCA requires complete data, so it's important to handle missing values before performing PCA.
6)Numeric values: PCA only works with numeric features, like continuous quantities or counts.

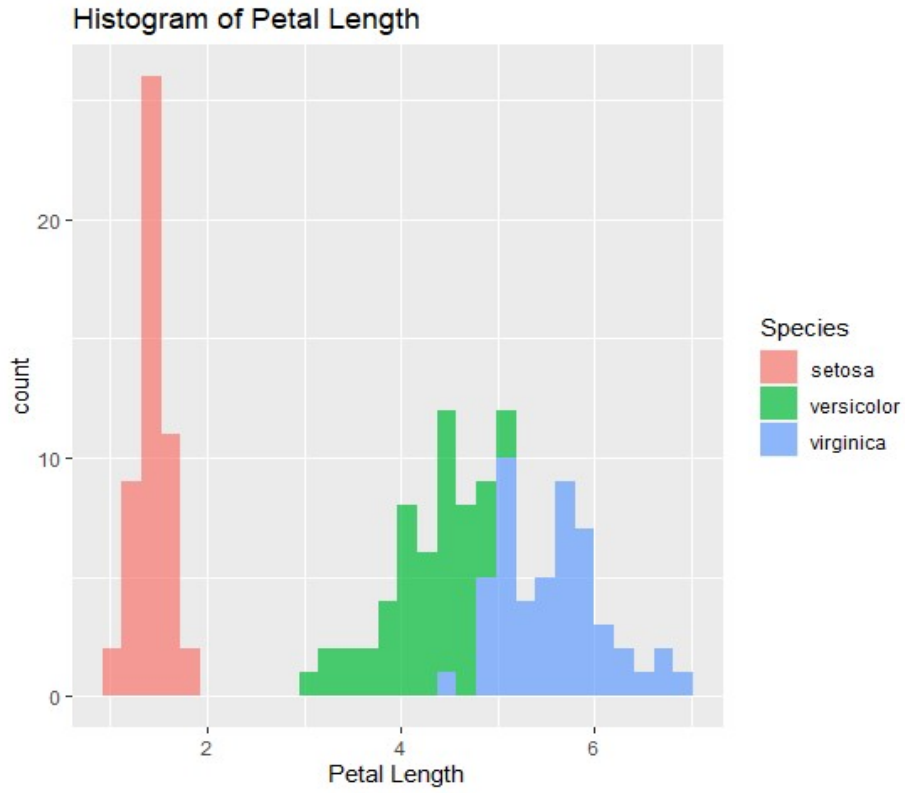
Application

```
> #iris data set
> library(datasets)
> data(iris)
> head(iris)
  Sepal.Length Sepal.Width Petal.Length Petal.Width Species
1          5.1         3.5          1.4         0.2  setosa
2          4.9         3.0          1.4         0.2  setosa
3          4.7         3.2          1.3         0.2  setosa
4          4.6         3.1          1.5         0.2  setosa
5          5.0         3.6          1.4         0.2  setosa
6          5.4         3.9          1.7         0.4  setosa

> library(ggplot2)

> #Histograms for its attribute

> ggplot(iris)
+geom_histogram(aes(x=Sepal.Length,fill=Species),alpha=0.7,bins=30)+labs(title="Histogram
of Sepal Length",x="Sepal Length")


> ggplot(iris)
+geom_histogram(aes(x=Sepal.Width,fill=Species),alpha=0.7,bins=30)+labs(title="Histogram of
Sepal Width",x="Sepal Width")


> ggplot(iris)
+geom_histogram(aes(x=Petal.Length,fill=Species),alpha=0.7,bins=30)+labs(title="Histogram
of Petal Length",x="Petal Length")


> ggplot(iris)
+geom_histogram(aes(x=Petal.Width,fill=Species),alpha=0.7,bins=30)+labs(title="Histogram of
Petal Width",x="Petal Width")
```

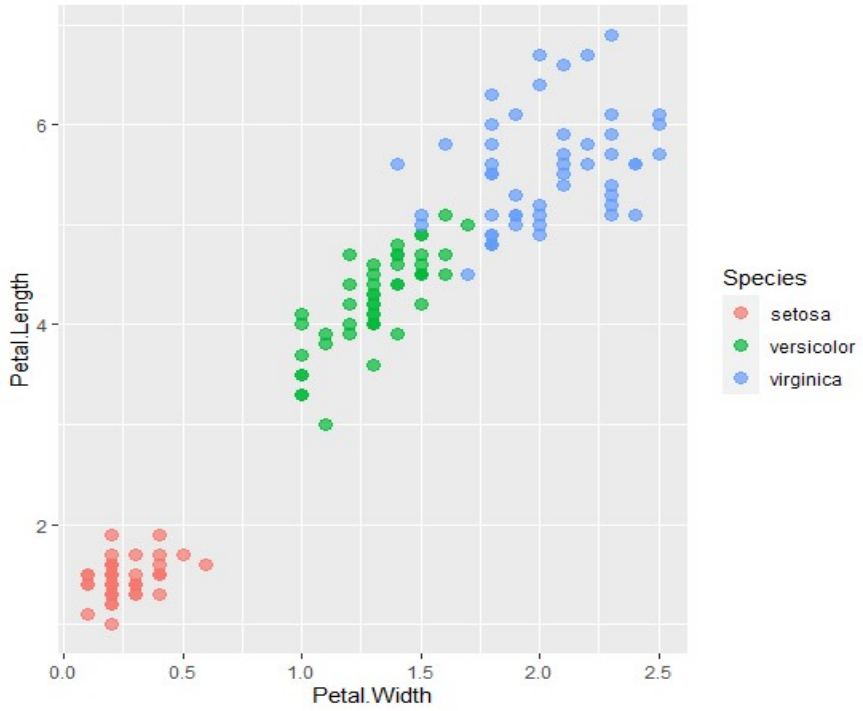

Histogram of Sepal Length
12.5
10.0
7.5
5.0
2.5
0.0
count
5
6
7
8
Sepal Length
Species
setosa
versicolor
virginica


Histogram of Sepal Width
20
10
0
count
2.0
2.5
3.0
3.5
4.0
4.5
Sepal Width
Species
setosa
versicolor
virginica

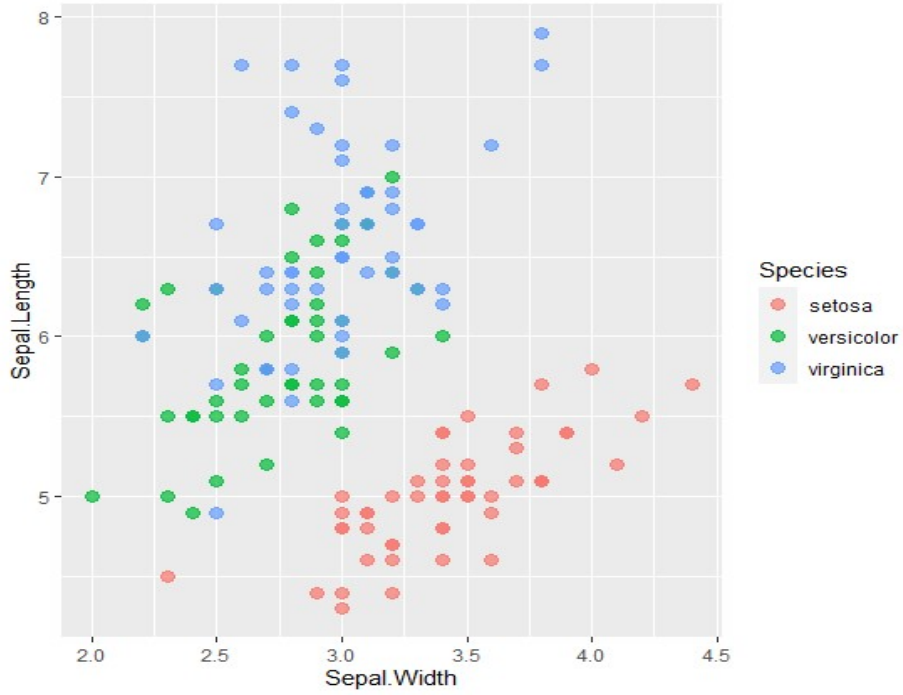

Histogram of Petal Width
30
20
10
0
count
0.0
0.5
1.0
1.5
2.0
2.5
Petal Width
Species
setosa
versicolor
virginica


Histogram of Petal Length
20
10
0
count
2
4
6
Petal Length
Species
setosa
versicolor
virginica

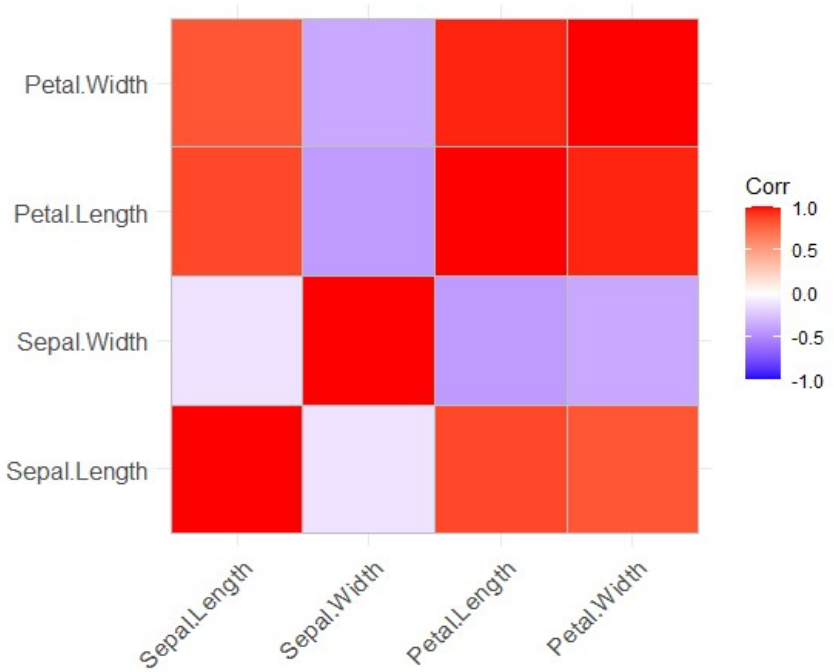
```
> #There is huge amount of overlapping in sepal length and width histograms, but in case of petal length and width things are more clear

> #Scatterplot between Petal Width and Petal Length
> ggplot(iris)
+geom_point(aes(x=Petal.Width,y=Petal.Length,colour=Species),alpha=0.7,size=3)
```



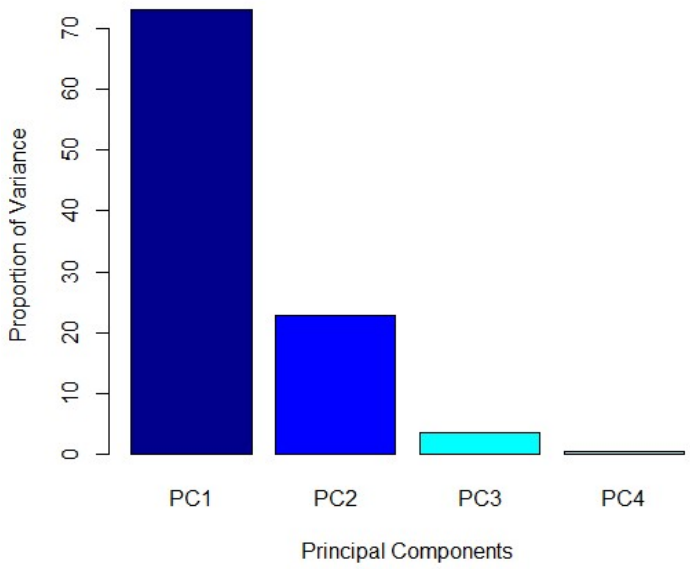

```
> #the plot shows positive linear relationship between petal length and width

> #Scatterplot between Sepal Width and Sepal Length
> ggplot(iris)
+geom_point(aes(x=Sepal.Width,y=Sepal.Length,colour=Species),alpha=0.7,size=3)
```



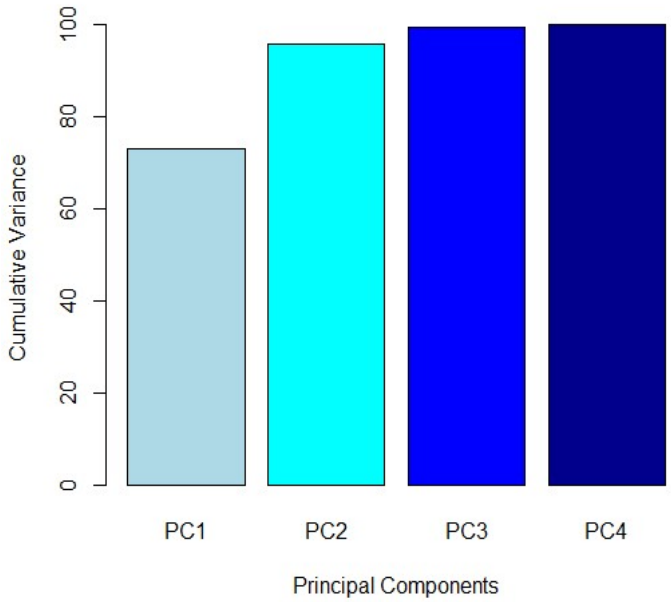
```
> #on contrary, sepal length and width show no relationship at all
> #also setosa seems to differ from the other two species

> #Normalization of the data set

> #Even though features in iris data set are initially measured in centimeters,
> #it is necessary to transform the data onto a unit scale
> #(mean=0 and variance=1) for many machine learning algorithms to function optimally.

> #First we remove the last column named "Species" since it doesn't contain numerical
values
> iris_num<-iris[,-5]

> #checking for missing values
> colSums(is.na(iris_num))
Sepal.Length  Sepal.Width Petal.Length  Petal.Width
           0            0            0            0


> #Then we scale the data set
> iris_sc<-scale(iris_num,center=T,scale=T)

> #install.packages("ggcorrplot")
> library(ggcorrplot)

> #Computation of Covariance (Correlation) Matrix
> iris_corr<-cor(iris_sc)
> iris_corr
              Sepal.Length Sepal.Width Petal.Length Petal.Width
Sepal.Length    1.0000000  -0.1175698    0.8717538   0.8179411
Sepal.Width    -0.1175698   1.0000000   -0.4284401  -0.3661259
Petal.Length    0.8717538  -0.4284401    1.0000000   0.9628654
Petal.Width     0.8179411  -0.3661259    0.9628654   1.0000000

> ggcorrplot(iris_corr)
```



```
> #The values are between -1 and 1.
> #The higher the value the most positively correlated the two variables.
> #Close to -1 shows a strong negative relationship
> #Close to 0 shows lack of linear relationship
> #Close to 1 shows a strong positive relationship

> #We can confirm that petal length and width have a strong positive relationship, since
the value between them is close to 1
> #also sepal length vs. petal length and sepal length vs. petal width have a strong
positive relationship too
> #on the other hand we have a poor relationship between sepal length and width



> #Computing the Eigenvectors and Eigenvalues
> ev<-eigen(iris_corr)
> ev
eigen() decomposition
$values
[1] 2.91849782 0.91403047 0.14675688 0.02071484

$vectors
            [,1]        [,2]       [,3]       [,4]
[1,]  0.5210659 -0.37741762  0.7195664  0.2612863
[2,] -0.2693474 -0.92329566 -0.2443818 -0.1235096
[3,]  0.5804131 -0.02449161 -0.1421264 -0.8014492
[4,]  0.5648565 -0.06694199 -0.6342727  0.5235971



> e_vectors<-ev$vectors
> e_values<-ev$values
> e_vectors
            [,1]        [,2]       [,3]       [,4]
[1,]  0.5210659 -0.37741762  0.7195664  0.2612863
[2,] -0.2693474 -0.92329566 -0.2443818 -0.1235096
[3,]  0.5804131 -0.02449161 -0.1421264 -0.8014492
[4,]  0.5648565 -0.06694199 -0.6342727  0.5235971
> e_values
[1] 2.91849782 0.91403047 0.14675688 0.02071484



> #We want to keep the eigenvectors with the highest eigenvalues and the ones with the
lowest eigenvalues will be discarded

> #we then need to check the proportion of variance for each component
> total_value<-sum(e_values)
> pr_var<-round((e_values/total_value)*100,2)
> print(paste0(pr_var,"%"))
[1] "72.96%" "22.85%" "3.67%"  "0.52%"

> barplot(pr_var,ylab="Proportion of Variance",xlab="Principal
Components",names.arg=c("PC1","PC2","PC3","PC4"),col=c("darkblue","blue","cyan","lightblue"
))
```

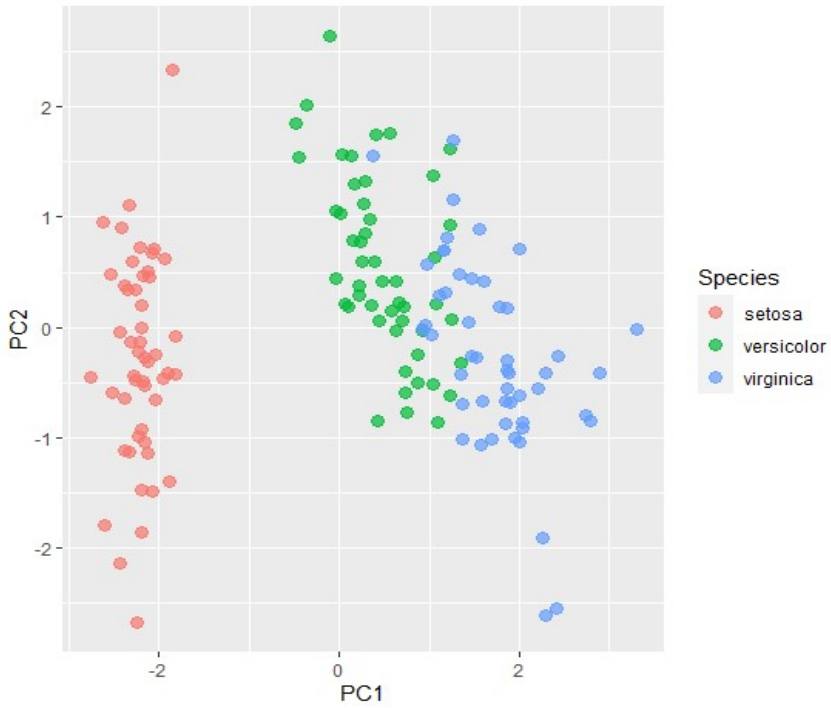



```
> #and the cumulative proportion of variance
> cumul_var<-c(pr_var[1],sum(pr_var[1:2]),sum(pr_var[1:3]),sum(pr_var))
> print(paste0(cumul_var,"%"))
[1] "72.96%" "95.81%" "99.48%" "100%"

> barplot(cumul_var,ylab="Cumulative Variance",xlab="Principal
Components",names.arg=c("PC1","PC2","PC3","PC4"),col=c("lightblue","cyan","blue","darkblue"
```

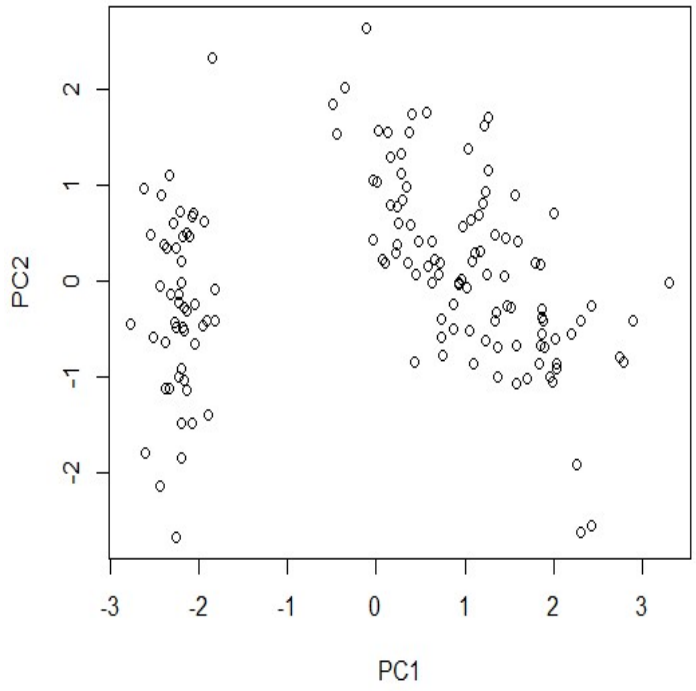



```
))

> # the first component contains 73% of the variance
> # the second one 23% and added together are 96% of the variance.
```

```
> #next step is the creation of the projection matrix which will help to transform the iris
data to a 2-dimensional subspace based on the top two principal components
> pr_matrix<-t(rbind(e_vectors[,1],e_vectors[,2]))
> pr_matrix
            [,1]        [,2]
[1,]  0.5210659 -0.37741762
[2,] -0.2693474 -0.92329566
[3,]  0.5804131 -0.02449161
[4,]  0.5648565 -0.06694199

> # new_subspace=iris_num*pr_matrix
> new_subspace<-data.frame(iris_sc%*%pr_matrix)
> # %*% produces the dot product of 150x4 and 4x2 matrices
> head(new_subspace)
          X1         X2
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

> #lets label back the new subspace with the  column 'Species'
> lbled_sub<-cbind(new_subspace,iris$Species)
> lbled_sub<-data.frame("PC1"=lbled_sub$X1,"PC2"=lbled_sub$X2,"Species"=iris$Species)
> head(lbled_sub)
        PC1        PC2 Species
1 -2.257141 -0.4784238  setosa
2 -2.074013  0.6718827  setosa
3 -2.356335  0.3407664  setosa
4 -2.291707  0.5953999  setosa
5 -2.381863 -0.6446757  setosa
6 -2.068701 -1.4842053  setosa

> #Visualization of the new 2D Subspace of iris data set
> ggplot(lbled_sub)+geom_point(aes(x=PC1,y=PC2,colour=Species),alpha=0.7,size=3)
```



```
> #Same using Prcomp()
> iris_pca<-prcomp(iris_num,center=T,scale=T)
> summary(iris_pca)
Importance of components:
                          PC1     PC2      PC3      PC4
Standard deviation     1.7084 0.9560 0.38309 0.14393
Proportion of Variance 0.7296 0.2285 0.03669 0.00518
Cumulative Proportion  0.7296 0.9581 0.99482 1.00000
> plot(predict(iris_pca))
```




```
> #We succesfully reduced the dimensionality of the data set from 4D to 2D, both ways.
```