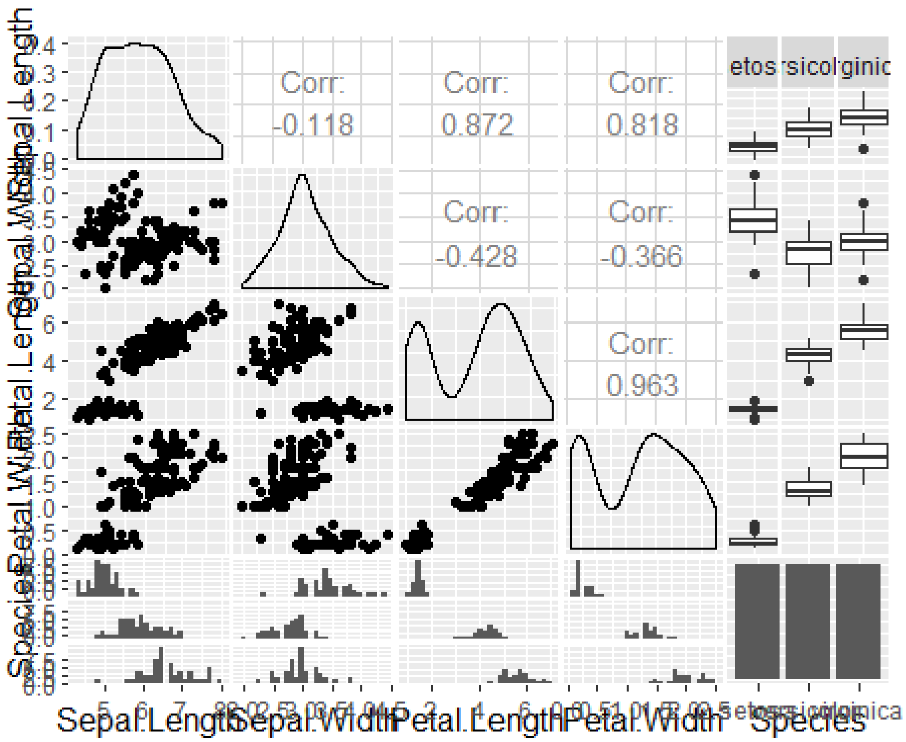
PCA

Pratik Gandhi

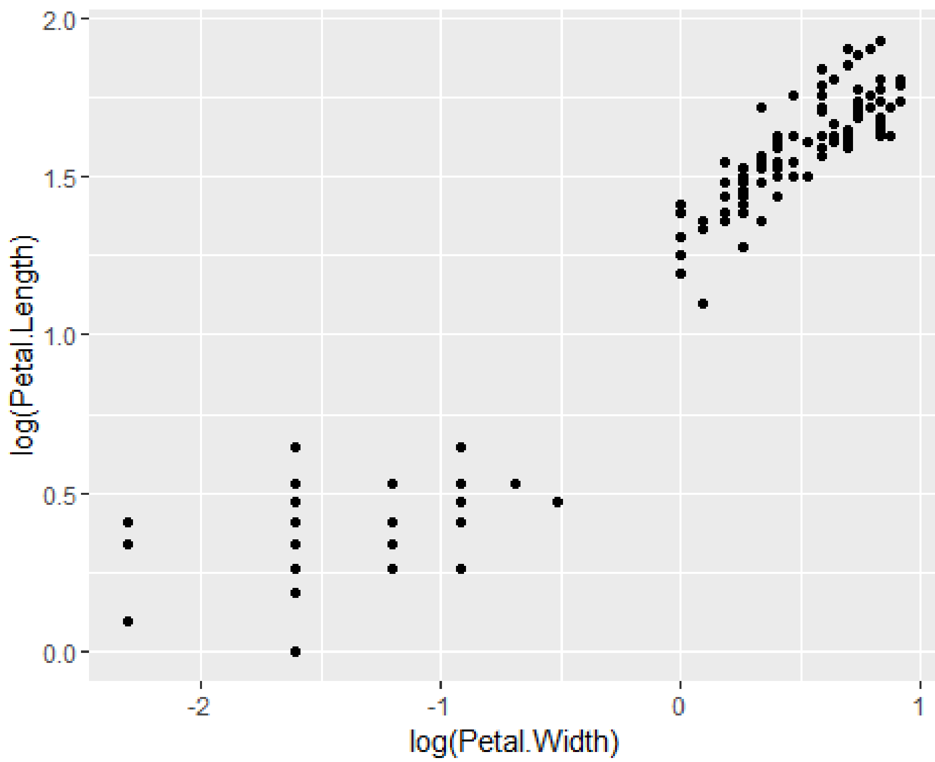
May 16, 2016

```
library(GGally)

## Warning: package 'GGally' was built under R version 3.2.5

library(ggplot2)

## Warning: package 'ggplot2' was built under R version 3.2.4

# Loading the data:
data("iris")

# Looking at first few observations of the dataset:
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

# Looking at the class type of all variables:
str(iris)

## 'data.frame':    150 obs. of  5 variables:
##  $ Sepal.Length: num  5.1 4.9 4.7 4.6 5 5.4 4.6 5 4.4 4.9 ...
##  $ Sepal.Width : num  3.5 3 3.2 3.1 3.6 3.9 3.4 3.4 2.9 3.1 ...
##  $ Petal.Length: num  1.4 1.4 1.3 1.5 1.4 1.7 1.4 1.5 1.4 1.5 ...
##  $ Petal.Width : num  0.2 0.2 0.2 0.2 0.2 0.4 0.3 0.2 0.2 0.1 ...
##  $ Species     : Factor w/ 3 levels "setosa","versicolor",..: 1 1 1 1 1 1
1 1 1 1 ...

# Tabling them by Species:
table(iris$Species)

## 
##     setosa versicolor  virginica 
##         50         50         50

# Looking at the variables in pairs to look at the correlations:
ggpairs(iris)

## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```




```
# Looking at one of the pairs with highest correlation:
qplot(log(Petal.Width), log(Petal.Length), data = iris)
```



Calculating PCA step by step:

```
# Applying log to all the continuous variables:
log.iris <- log(iris [, 1:4])
head(log.iris)
```

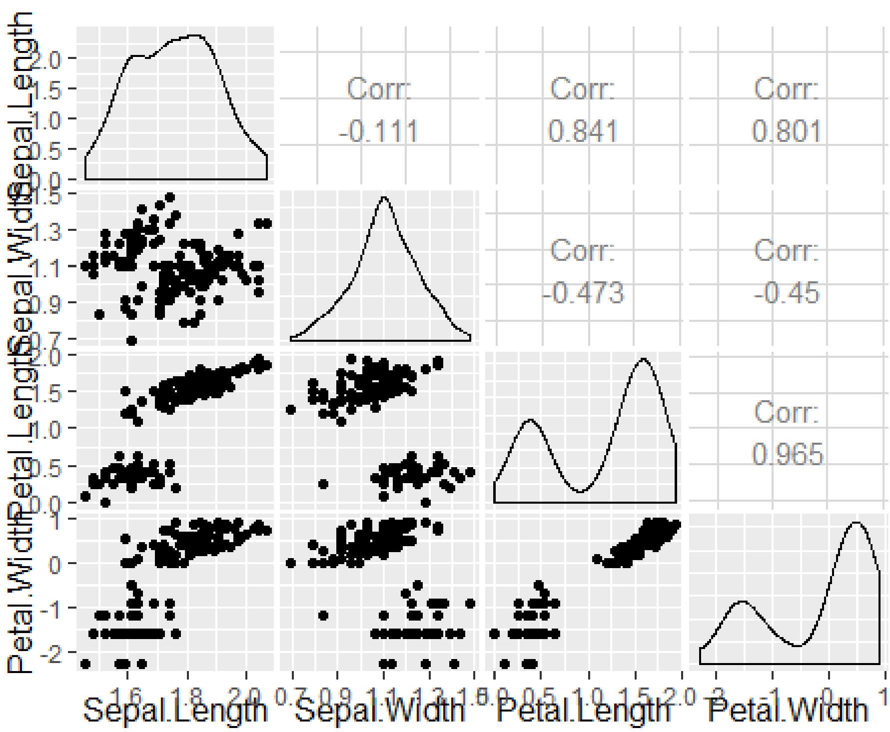
```
##   Sepal.Length Sepal.Width Petal.Length Petal.Width
## 1     1.629241    1.252763    0.3364722  -1.6094379
## 2     1.589235    1.098612    0.3364722  -1.6094379
## 3     1.547563    1.163151    0.2623643  -1.6094379
## 4     1.526056    1.131402    0.4054651  -1.6094379
## 5     1.609438    1.280934    0.3364722  -1.6094379
## 6     1.686399    1.360977    0.5306283  -0.9162907
```

```
# Storing the discrete variable in another one:
iris.species <- iris [,5]
iris.species
```

```
##   [1] setosa     setosa     setosa     setosa     setosa     setosa
##   [7] setosa     setosa     setosa     setosa     setosa     setosa
##  [13] setosa     setosa     setosa     setosa     setosa     setosa
##  [19] setosa     setosa     setosa     setosa     setosa     setosa
##  [25] setosa     setosa     setosa     setosa     setosa     setosa
##  [31] setosa     setosa     setosa     setosa     setosa     setosa
##  [37] setosa     setosa     setosa     setosa     setosa     setosa
##  [43] setosa     setosa     setosa     setosa     setosa     setosa
##  [49] setosa     setosa     versicolor versicolor versicolor versicolor
```

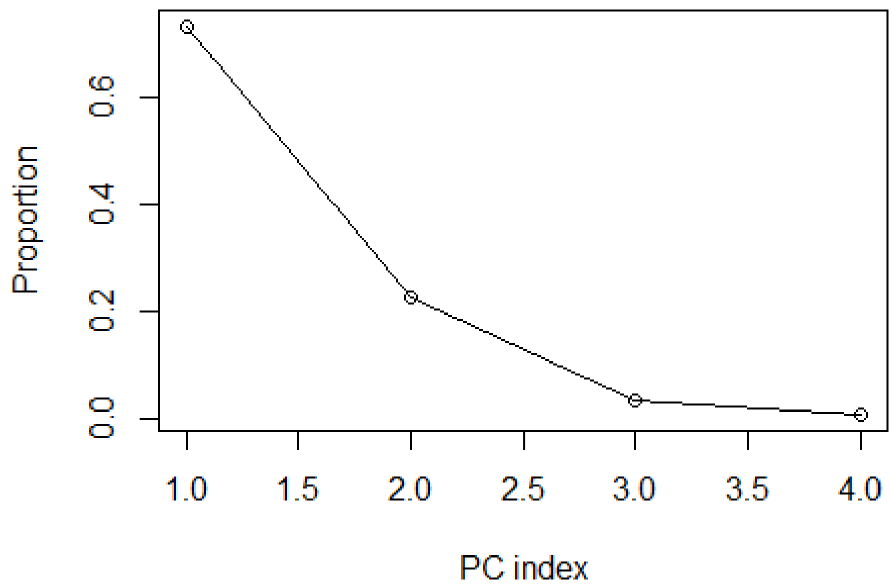
```
##  [55] versicolor versicolor versicolor versicolor versicolor versicolor
##  [61] versicolor versicolor versicolor versicolor versicolor versicolor
##  [67] versicolor versicolor versicolor versicolor versicolor versicolor
##  [73] versicolor versicolor versicolor versicolor versicolor versicolor
##  [79] versicolor versicolor versicolor versicolor versicolor versicolor
##  [85] versicolor versicolor versicolor versicolor versicolor versicolor
##  [91] versicolor versicolor versicolor versicolor versicolor versicolor
##  [97] versicolor versicolor versicolor versicolor virginica  virginica
## [103] virginica  virginica  virginica  virginica  virginica  virginica
## [109] virginica  virginica  virginica  virginica  virginica  virginica
## [115] virginica  virginica  virginica  virginica  virginica  virginica
## [121] virginica  virginica  virginica  virginica  virginica  virginica
## [127] virginica  virginica  virginica  virginica  virginica  virginica
## [133] virginica  virginica  virginica  virginica  virginica  virginica
## [139] virginica  virginica  virginica  virginica  virginica  virginica
## [145] virginica  virginica  virginica  virginica  virginica  virginica
## Levels: setosa versicolor virginica
```

```
#pairs(log.iris)
ggpairs(log.iris)
```




```
# Scaling the continuous variables:
log.iris.scaled <- scale(log.iris, center = TRUE, scale = TRUE)

## Here scale = (log.iris - colMeans(log.iris)) / apply(log.iris, 2, sd)
## In other words : (log.iris - mu) / sd
```

```r
# Performing SVD on our matrix:
log.iris.svd <- svd (log.iris.scaled)
names(log.iris.svd)

## [1] "d" "u" "v"

# SVD is performed to find the eigenvalues and eigenvectors of any data. We get three matrices when we run SVD.
# U represents the left singular vectors
# V represents the right singular vectors - Eigen Vectors
# D represents the diagonal vectors

U <- log.iris.svd$u
V <- log.iris.svd$v # PC loadings
D <- log.iris.svd$d # Strength of each component

# Multiplication of Scaled Data with the Loadings:
# Final Data (PCs) = RowFeatureVector (V) x RowZeroMeanData (log.iris.scaled)
Z <- log.iris.scaled %*% V # This are our PCs
head(Z)

##           [,1]       [,2]        [,3]         [,4]
## [1,] -2.406639 -0.3969554  0.19396467  0.004779476
## [2,] -2.223539  0.6901804  0.35000151  0.048868378
## [3,] -2.581105  0.4275418  0.01889761  0.049909545
## [4,] -2.450869  0.6860074 -0.06874595 -0.149646465
## [5,] -2.536853 -0.5082516  0.02932259 -0.040048202
## [6,] -1.841495 -1.2899381 -0.25276831  0.163890597

## We have changed our original data in terms of the eigenvectors. This will reorient the data in the direction where the data is having maximum variance.

## Variance explained by each PC:
var.exp <- D^2 / sum(D^2)
var.exp

## [1] 0.73312837 0.22675677 0.03325206 0.00686280

## Cummulative Sum of the Variation Explained:
cum.var.exp <- cumsum(var.exp)

# Plotting the PCs for both with and without Cummulative Sum:
plot(var.exp, xlab = "PC index", ylab = "Proportion")
lines(var.exp)
```




```r
plot(cum.var.exp, xlab = "PC index", ylab = "Cummulative Proportion")
lines(cum.var.exp)
```