Dimension Reduction

SVD & PCA

Dimension Reduction

- Dimension Reduction Algorithms are a part of unsupervised learning algorithms
- Goals of Dimension Reduction Algorithms are:
  - to find structure within predictors(features)
  - to aid in visualization
- We will learn in this section Singular Value Decomposition and Principal Component Analysis

Why Dimension Reduction?

- Statistical Purpose: For finding a new set of variables that are uncorrelated and explain as much as variance as possible
- Data Compression Purpose: For finding the best matrix created with fewer variables that explain the original data

Singular Value Decomposition

- SVD is a method for identifying and ordering the dimensions along which data points exhibit the most variation.
- Once we have identified where the most variation is, it's possible to find the best approximation of the original data points using fewer dimensions.
- What makes SVD practical for NLP like applications is that you can simply ignore variation below a particular threshold to massively reduce your data but be assured that the main relationships of interest have been preserved.

SVD & PCA

- If M is a matrix, then the matrix M ($m \times n$) can be factorized as
  $$M = U\Sigma V^T,$$
  where U is $m \times m$ left singular orthogonal matrix,
  Σ is $m \times n$ diagonal matrix
  V is $n \times n$ right singular orthogonal matrix
  V^T is transpose of matric V.
- Principal Components are obtained by first scaling and centering the data and then extracting the right singular values from it

Principal Component Analysis

- Finds a linear combination of variables to create principal component
- PCA maintains most of variance from the original data in Principal Components created
- Principal Components are uncorrelated (i.e. orthogonal to each other) with each other
- If our data is having n observations and p variables then we can have at most $\min(n-1, p)$ principal components

A two dimensional example

- Consider the data of two variables in which we find fit a regression line to it.
- This regression line is determined such that we have minimum residuals.
- This line can be said to be the first component of this data.
- Once the line is fitted, each point can be mapped on the line and perpendicular projection can be drawn of all the points on the line.
- This projected value can be referred to as component scores or factor scores.

Steps in PCA

1. Data Preparation: Make the data completely numerical. If, it is not numerical then dummy variables can be replaced with categorical variables
2. Centering and Scaling(if needed): Subtract each value mean of its respective column and (optional) divide values by their respective column Standard Deviations
3. Calculate Covariance Matrix / Correlation matrix
4. Calculate eigenvalues and eigenvectors of the covariance matrix / correlation matrix
5. Extract the components with maximum variation
6. Calculate the scores

PCA in R

- We can use functions *prcomp()* and *princomp()* in R to get the principal components

Syntax :

prcomp(x, …)

princomp(x, …)

Where x : numeric matrix or data frame object

- Function prcomp() uses Singular Value Decomposition approach
- Function princomp() uses Spectral Decomposition approach

Example: Milk (dataset in package flexclust)

- The data set contains the ingredients of mammal's milk of 25 animals.
- A data frame with 25 observations on the following 5 variables (all in percent)
  - water
  - protein
  - fat
  - lactose
  - ash

R Program & Output

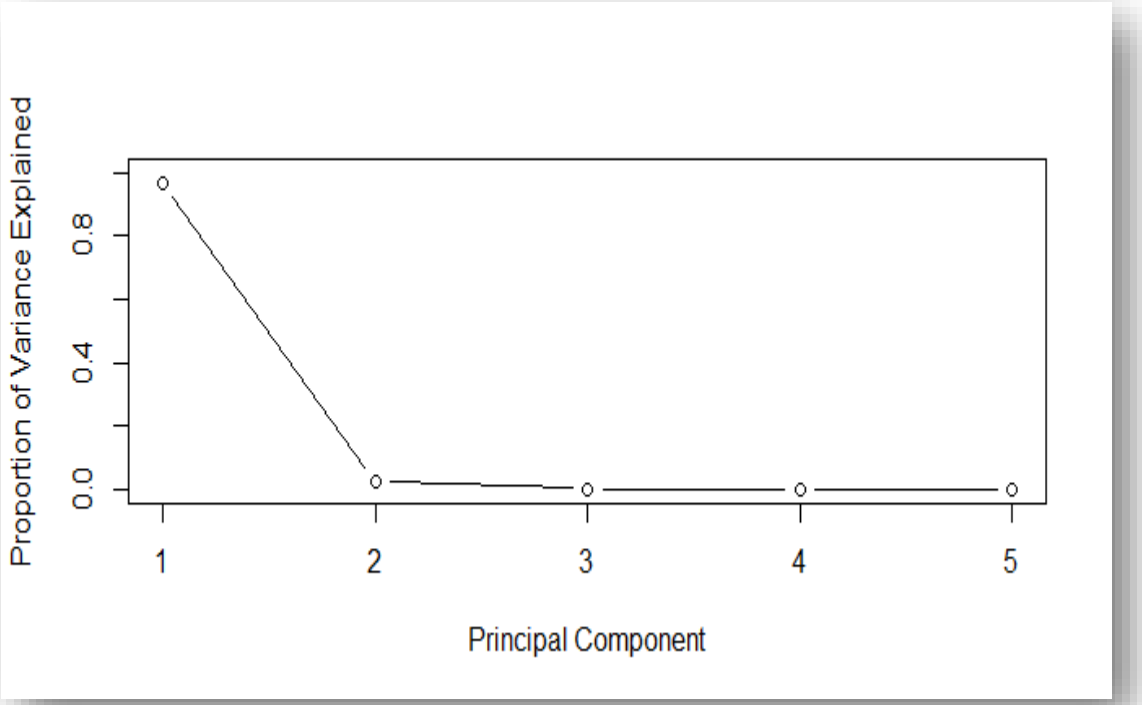
```
> data(milk,package="flexclust")
> prc <- prcomp(milk)
> names(prc)
[1] "sdev"     "rotation" "center"   "scale"    "x"
> prc$sdev
[1] 16.7978370  2.8520722  1.0970813  0.5529434  0.2602628
> prc$rotation
                PC1        PC2         PC3         PC4          PC5
water   -0.76163901  0.1560720 -0.57443881 -0.25593731 -0.007980218
protein  0.16060436 -0.8536804 -0.26796098 -0.39275281 -0.139205643
fat      0.62047078  0.4429998 -0.55244430 -0.33676547  0.012709314
lactose -0.09474321  0.1822437  0.53829678 -0.81635500  0.040150186
ash      0.01232867 -0.1319461 -0.05708616 -0.01987089  0.989335404
> prc$center
  water protein     fat lactose     ash
78.1840  6.2120 10.3080  4.1320  0.8632
> prc$scale
[1] FALSE
```

Principal Components

```
> prc$x
                PC1         PC2         PC3         PC4         PC5
HORSE     -15.699712  1.39197129  0.78431583 -0.745985682 -0.107171004
ORANGUTAN -13.038725  3.12466664  0.16567273  0.029808998 -0.040544661
MONKEY    -13.369091  2.15252958  0.66944715 -0.314736887 -0.204578654
DONKEY    -15.681458  2.23453003  0.30995318 -0.007946847  0.042963397
HIPPO     -13.843989  4.27403009 -2.11711932  0.829914816 -0.134380501
CAMEL     -12.034705  0.88207313 -0.55503648  0.413741173  0.089044149
BISON     -12.354335 -0.96671901  0.96894134 -0.058081301  0.116964048
BUFFALO    -4.581620 -0.07472475 -0.52511055 -0.530816742 -0.077929801
GUINEA PIG -4.432366 -2.07028321 -1.50604285  0.798296377 -0.305085635
CAT        -4.490963 -4.49773268 -0.63919284 -1.268082504 -0.720663411
FOX        -5.346419 -1.61967975  0.77852120 -0.170495743 -0.040371922
LLAMA     -11.255276  0.39863441  0.56309423 -0.023754469  0.161556626
MULE      -15.089414  1.97199739 -0.19988497  0.386359851  0.049827843
PIG        -6.560672 -2.45475661 -0.25849121  0.571663412 -0.009711660
ZEBRA     -10.151385  1.78744627 -0.06310054  0.114573084  0.198592076
SHEEP      -5.482741 -0.51588050  0.43396688  0.115167826  0.074179545
DOG         1.540935 -3.53888784  0.07257563  0.458893058 -0.137343266
ELEPHANT    9.663123  4.59042799  1.77412249 -0.708163934  0.344232864
RABBIT      8.182417 -5.63109875 -0.50283397  0.224097979  0.574798165
RAT         6.316605 -2.64501034  0.71973369  0.177876460  0.156213136
DEER       16.007811 -1.68177440 -0.10968560 -0.423827042  0.103966802
REINDEER   17.275552 -1.86198226  0.05651254 -0.380545631  0.074593918
WHALE      17.987185 -2.00835779 -1.04946468 -0.111977448  0.291015498
SEAL       44.823398  5.35002888 -2.40823821 -0.534540641 -0.008082254
DOLPHIN    41.615844  1.40855222  2.63734433  1.158561837 -0.492085297
```

Variance Explained

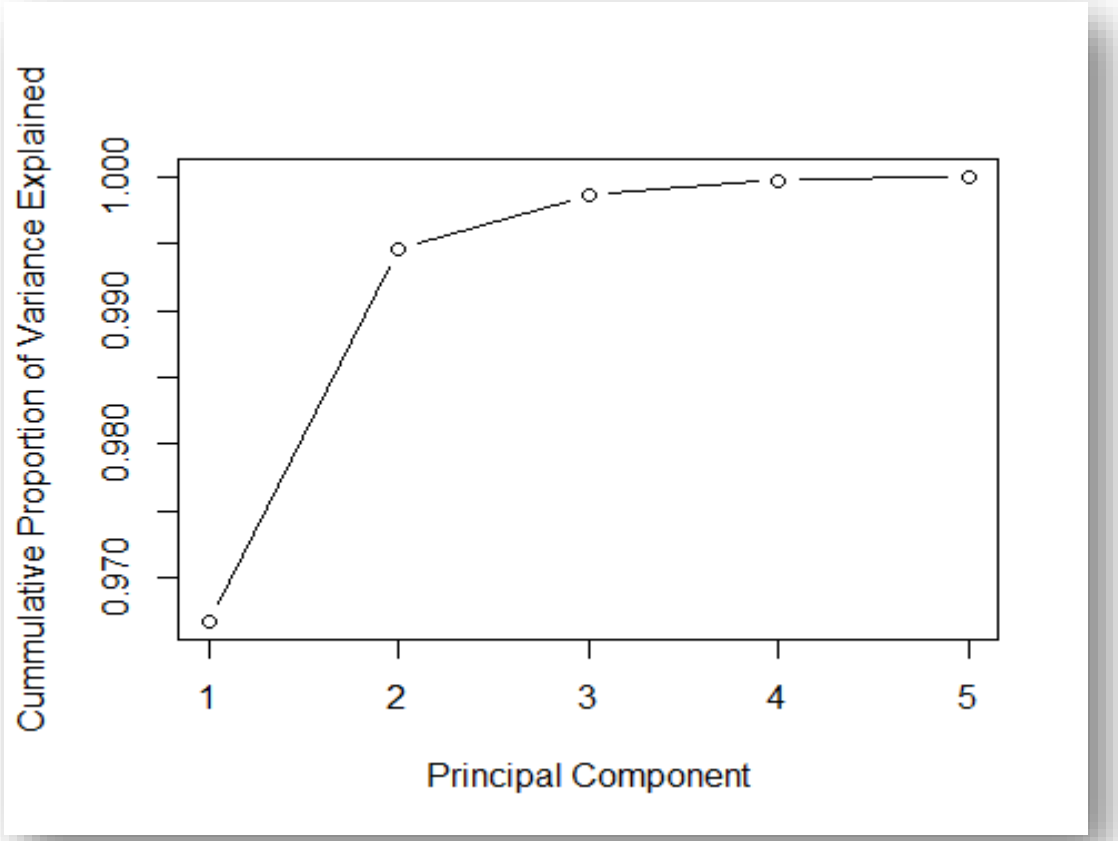
```
pr.var <- prc$sdev^2
pve <- pr.var / sum(pr.var)
# Plot variance explained for each principal component
plot(pve, xlab = "Principal Component",
     ylab = "Proportion of Variance Explained",
     ylim = c(0, 1), type = "b")
```



- We can observe that the maximum variance(96.67%) has been already explained by the first principal component

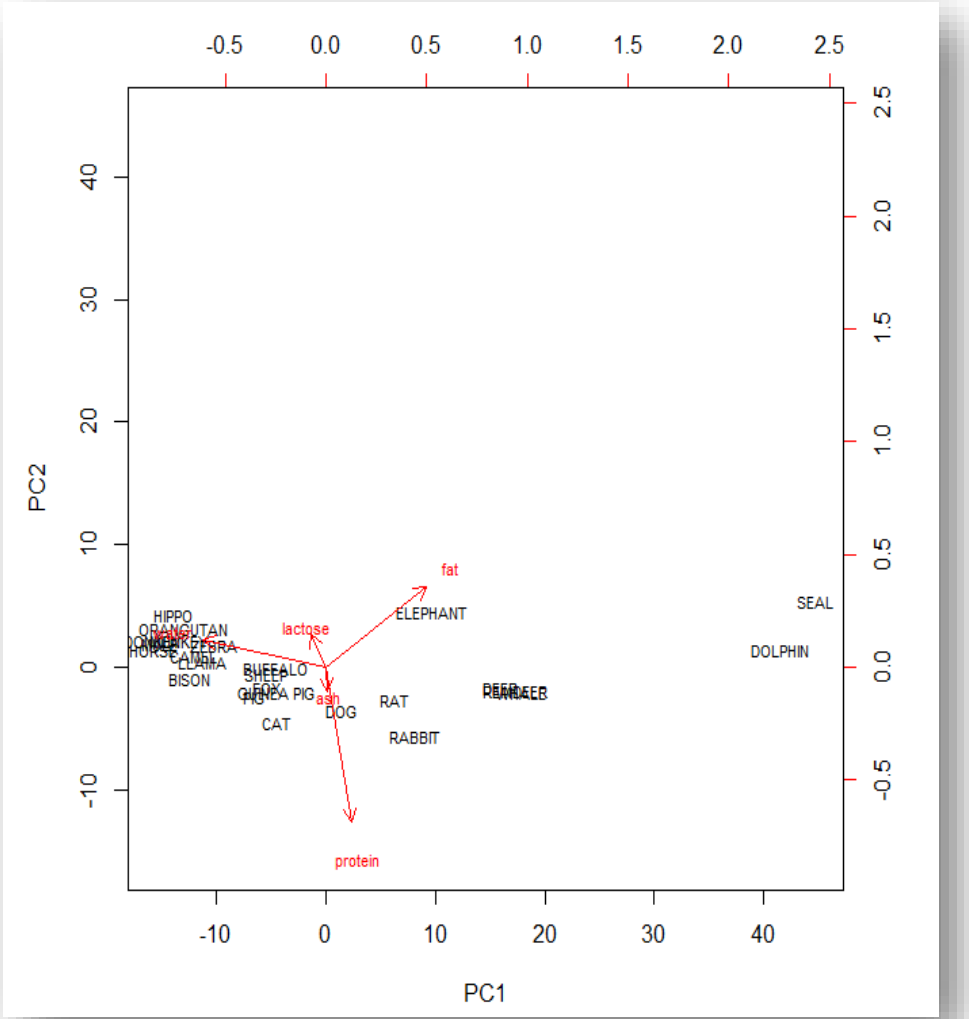
Cumulative Variation Explained

```
# Plot cumulative proportion of variance explained
plot(cumsum(pve), xlab = "Principal Component",
     ylab = "Cummulative Proportion of Variance Explained",
     type = "b")
```



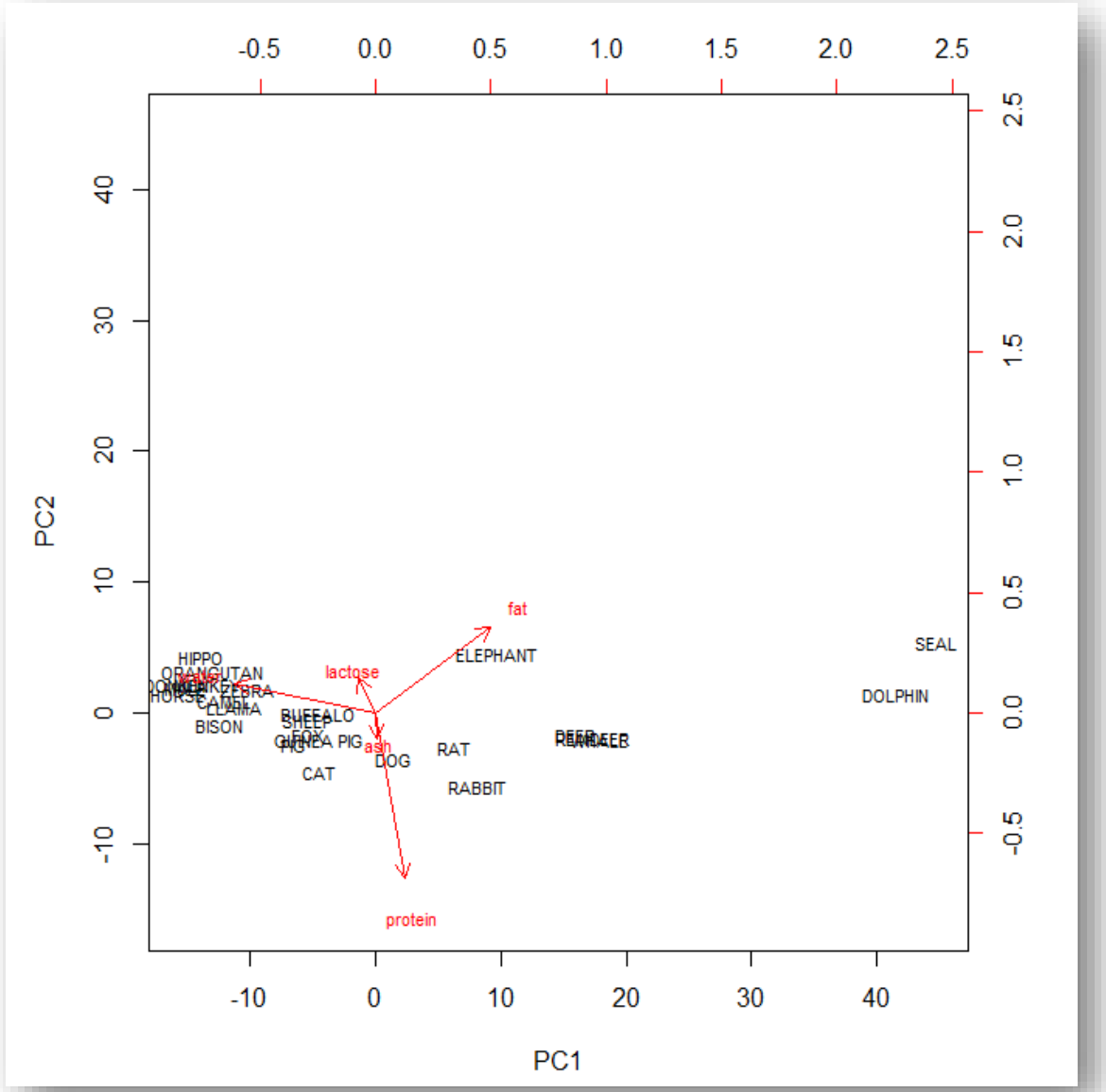
Biplot

- Biplot contains two types of elements:
  1. Principal Components Scores
  2. Loadings
- Biplot can be seen as a scatter plot with first two principal components at X-axis and Y-axis respectively
- Loadings of the components are shown by arrows



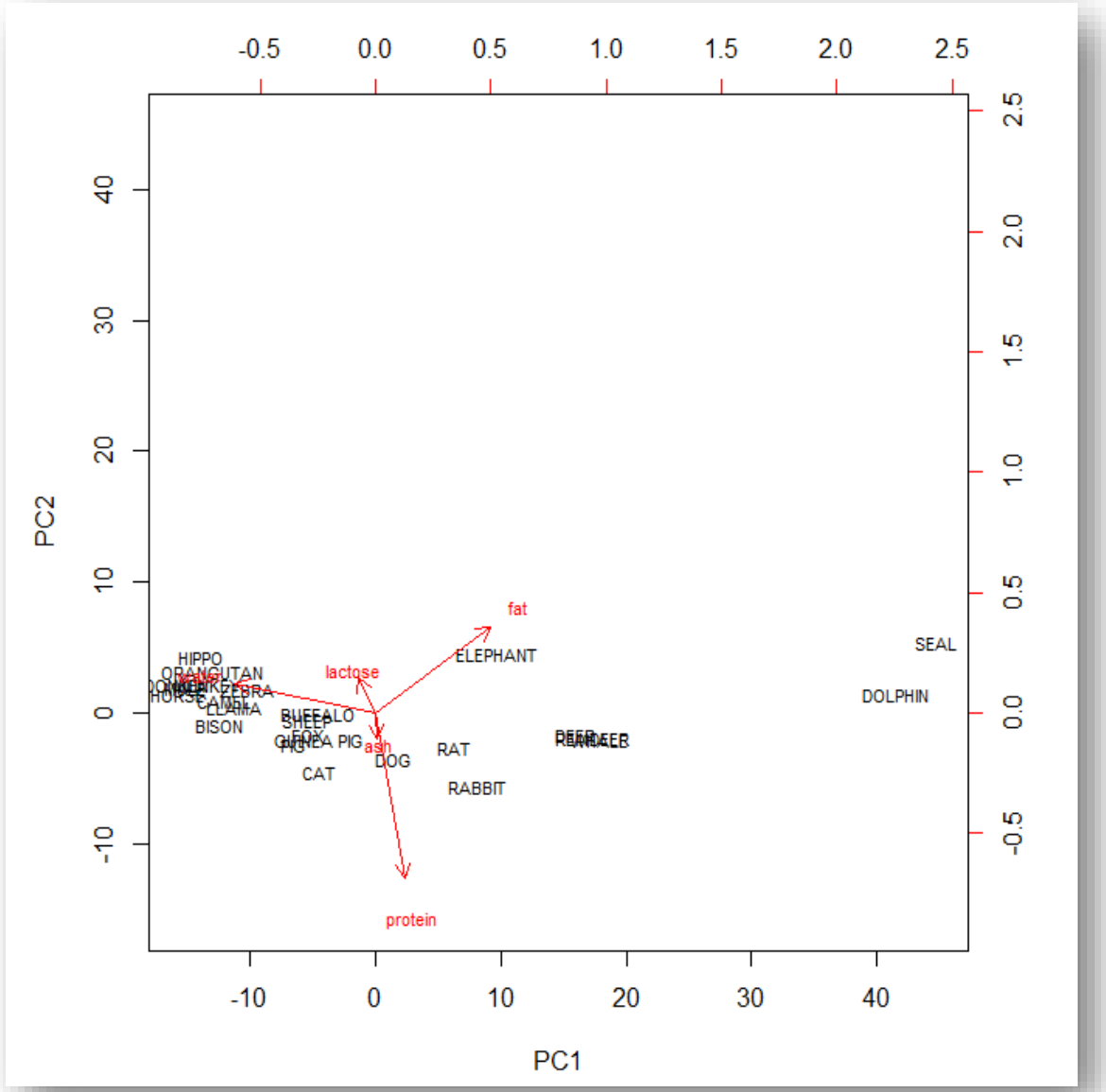
Biplot Interpretation

- Points like Deer and Reindeer are having nearly equal scores for PC1 and PC2
- There is a quite high concentration of points at the left side indicating that many of those points have water content similar
- The loadings have been indicated by red lines. e.g loadings of first two PCs for **protein** in component ($prc$$rotation) are PC1=0.16060436 and PC2=-0.8536804. Hence the red point **protein** on the graph. The scales for loadings have been shown on the top and right edges of the graph



Biplot Interpretation

- Consider the red point **fat**. In the data, Seal, Dolphin, Whale, Reindeer, Deer, Elephant and Rabbit are the highest.
- But, only elephant is seen to be nearer to red point **fat**. This is because the other animals though they have high **fat** in their milk, they don't have the similar **protein**, **lactose**, **ash** and **water** contents in their milk.
- We also can observe here that, Deer and Reindeer are having similar milk characteristics hence are very close.



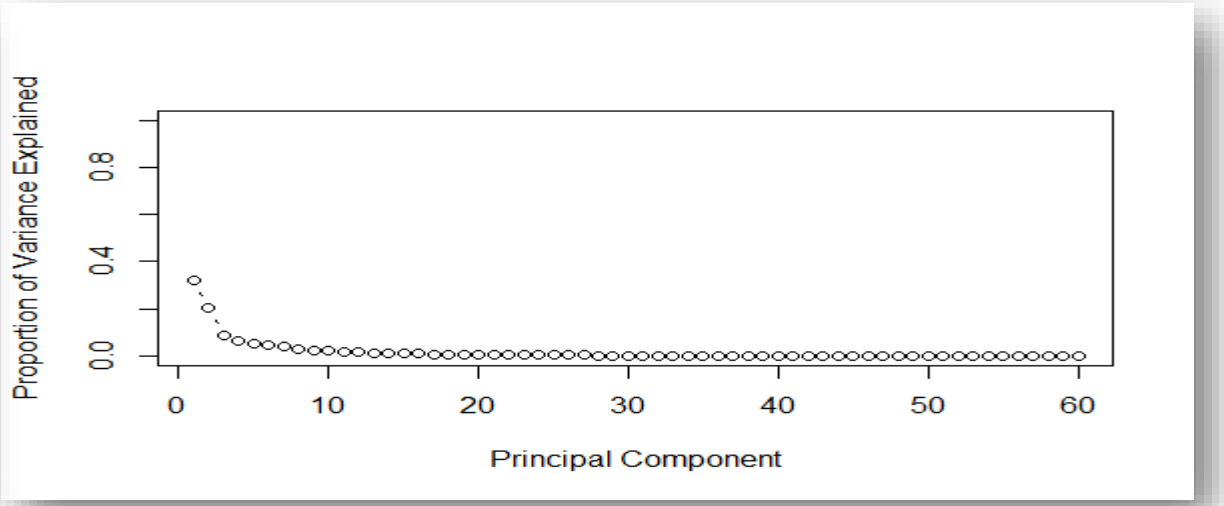
PCA for Improving Accuracy : Example

- Sometimes the PCA method can also prove to be a good algorithm for improving accuracy of a supervised learning algorithm.
- Consider dataset Sonar from package **mlbench** in which we have 60 sonar signal variables and one categorical response variable names Class.
- Our Findings here:
  - For the original data, LDA was applied and accuracy came out to be 0.6613.
  - After applying PCA to the data and then partitioning it and testing the accuracy, it came out to be 0.7097.
  - When PCA was applied separately on training data and then loading calculated on validation data separately, the accuracy came out to be 0.7258

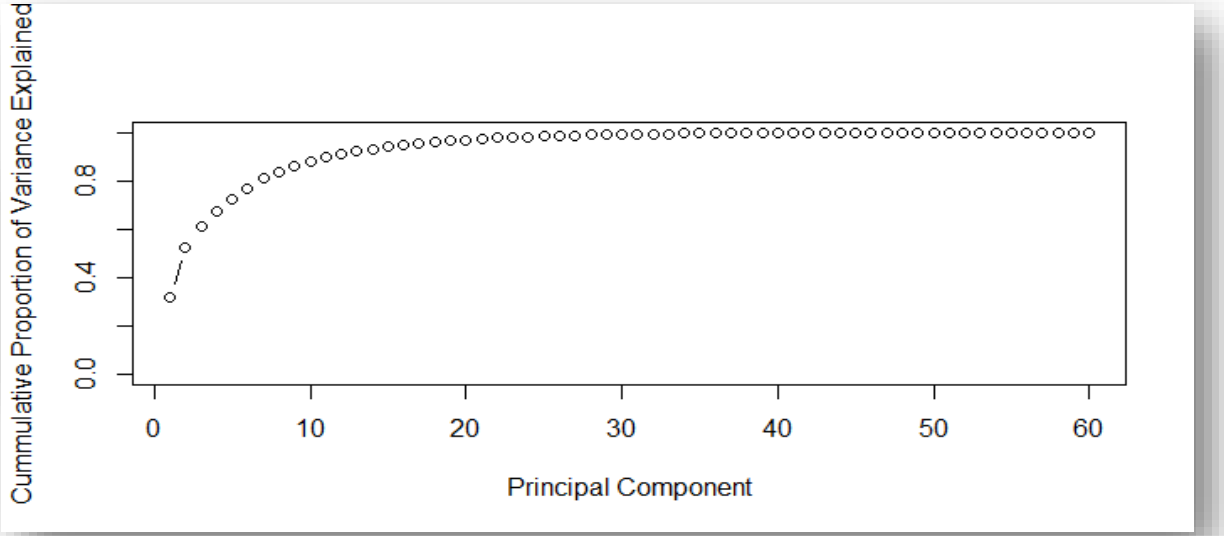
Sonar: Variation Explained

```
prc <- prcomp(Sonar[,-61])
names(prc)
pr.var <- prc$sdev^2
pve <- pr.var / sum(pr.var)

# Plot variance explained for each principal component
plot(pve, xlab = "Principal Component",
     ylab = "Proportion of Variance Explained",
     ylim = c(0, 1), type = "b")
```




```
# Plot cumulative proportion of variance explained
plot(cumsum(pve), xlab = "Principal Component",
     ylab = "Cummulative Proportion of Variance Explained",
     ylim = c(0, 1), type = "b")
```



Example: Total Variation Explained

```
> data.frame(pve,cumsum(pve)*100)
             pve cumsum.pve....100
1  3.197115e-01           31.97115
2  2.038306e-01           52.35421
3  8.555820e-02           60.91003
4  6.459322e-02           67.36935
5  5.164156e-02           72.53351
6  4.451402e-02           76.98491
7  4.207696e-02           81.19260
8  2.632652e-02           83.82526
9  2.230037e-02           86.05529
10 1.921817e-02           87.97711
```

- Total Variation explained upto first 6 components is nearly 77%. Hence we can take just first 6 PCs

Accuracy: Original Data

```
library(MASS)

model.lda <- lda(Class ~ . , data = training)

pred.lda <- predict(model.lda, newdata = validation)
confusionMatrix(pred.lda$class,validation$Class)
```

```
Confusion Matrix and Statistics

          Reference
Prediction  M  R
         M 21  9
         R 12 20

               Accuracy : 0.6613
                 95% CI : (0.5299, 0.7767)
    No Information Rate : 0.5323
    P-Value [Acc > NIR] : 0.02721

                  Kappa : 0.324
 Mcnemar's Test P-Value : 0.66252

            Sensitivity : 0.6364
            Specificity : 0.6897
         Pos Pred Value : 0.7000
         Neg Pred Value : 0.6250
             Prevalence : 0.5323
         Detection Rate : 0.3387
   Detection Prevalence : 0.4839
      Balanced Accuracy : 0.6630

       'Positive' Class : M
```

Accuracy: PCA Applied and then Partitioned

```
prc <- prcomp(Sonar[,-61])
names(prc)
```

```
trainingPCA <- cbind.data.frame(prc$x[intrain,1:6],Class=training$Class)
model.lda.PCA <- lda(Class ~ . , data = trainingPCA)

valid.PCA <- as.data.frame(prc$x[-intrain,1:6])
pred.lda.PCA <- predict(model.lda.PCA, newdata = valid.PCA)
confusionMatrix(pred.lda.PCA$class,validation$Class)
```

```
Confusion Matrix and Statistics

          Reference
Prediction  M  R
         M 22  7
         R 11 22

               Accuracy : 0.7097
                 95% CI : (0.5805, 0.818)
    No Information Rate : 0.5323
    P-Value [Acc > NIR] : 0.003325

                  Kappa : 0.4218
 Mcnemar's Test P-Value : 0.479500

            Sensitivity : 0.6667
            Specificity : 0.7586
         Pos Pred Value : 0.7586
         Neg Pred Value : 0.6667
             Prevalence : 0.5323
         Detection Rate : 0.3548
   Detection Prevalence : 0.4677
      Balanced Accuracy : 0.7126

       'Positive' Class : M
```

Accuracy: PCA on training and then scores on validation

```r
prc <- prcomp(training[,-61])
names(prc)

trainingPCA <- cbind.data.frame(prc$x[,1:6],Class=training$Class)
model.lda.PCA <- lda(Class ~ . , data = trainingPCA)

validation.PCA <- as.data.frame(predict(prc, newdata = validation))

pred.lda.PCA <- predict(model.lda.PCA, newdata = validation.PCA)
confusionMatrix(pred.lda.PCA$class,validation$Class)
```

```
Confusion Matrix and Statistics

          Reference
Prediction  M  R
         M 22  6
         R 11 23

               Accuracy : 0.7258
                 95% CI : (0.5977, 0.8315)
    No Information Rate : 0.5323
    P-Value [Acc > NIR] : 0.001435

                  Kappa : 0.455
 Mcnemar's Test P-Value : 0.331975

            Sensitivity : 0.6667
            Specificity : 0.7931
         Pos Pred Value : 0.7857
         Neg Pred Value : 0.6765
             Prevalence : 0.5323
         Detection Rate : 0.3548
   Detection Prevalence : 0.4516
      Balanced Accuracy : 0.7299

       'Positive' Class : M
```