Canine Species Analysis

This Project has an analysis of various Canine groups and their analysis is based on the Skull measurement information obtained for each specie.

The accompanied dataset, from Higham et al. (1980), gives 9 skull measurement for different canine groups.

The variables

- X1 = length of mandible
- X2 = breadth of mandible below 1st molar
- X3 = breadth of articular condyle
- X4 = height of mandible below first molar
- X5 length of first molar, X6 = breadth of first molar
- X7 = length of first to third molar inclusive (first to second for Cuon)
- X8 = length from first to fourth premolar inclusive
- X9 = breadth of lower canine

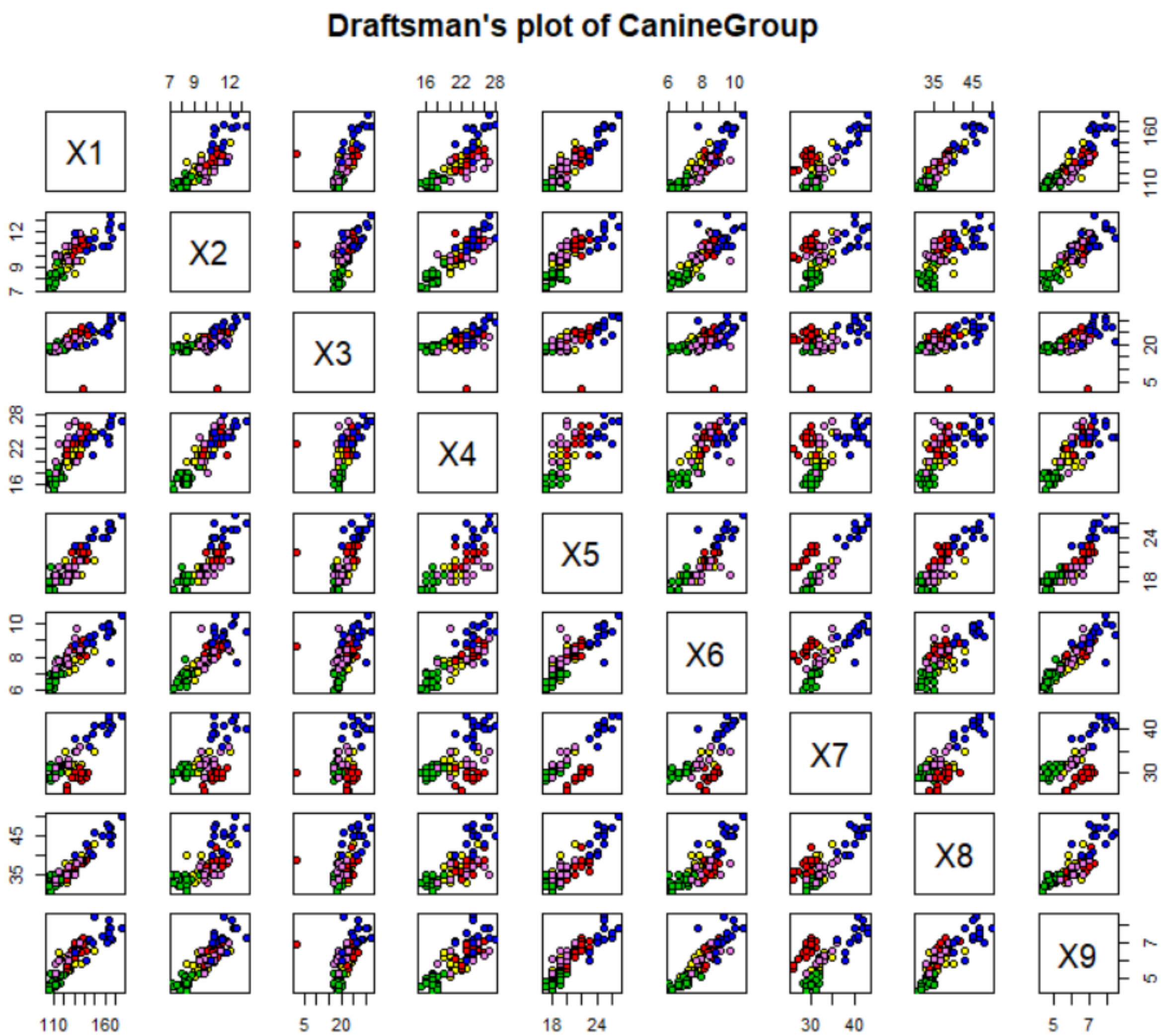
All measured in millimeters

Setting the data:

```
> library(readxl)
> library(ggplot2)
> canine_data<- read_excel("C:/Users/prera/Downloads/Final_Data.xlsx")
> dim(canine_data)
[1] 77 11
> attach(canine_data)
> head(canine_data )
# A tibble: 6 x 11
  CanineGroup    X1    X2    X3    X4    X5    X6    X7    X8    X9 Gender
  <chr>       <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <chr>
1 ModernDog     123  10.1    23    23    19   7.8    32    33   5.6 Male
2 ModernDog     137   9.6    19    22    19   7.8    32    40   5.8 Male
3 ModernDog     121  10.2    18    21    21   7.9    35    38   6.2 Male
4 ModernDog     130  10.7    24    22    20   7.9    32    37   5.9 Male
5 ModernDog     149  12      25    25    21   8.4    35    43   6.6 Male
6 ModernDog     125   9.5    23    20    20   7.8    33    37   6.3 Male
> str(canine_data)
Classes ‘tbl_df’, ‘tbl’ and 'data.frame':       77 obs. of  11 variables:
 $ CanineGroup: chr  "ModernDog" "ModernDog" "ModernDog" "ModernDog" ...
 $ X1         : num  123 137 121 130 149 125 126 125 121 122 ...
 $ X2         : num  10.1 9.6 10.2 10.7 12 9.5 9.1 9.7 9.6 8.9 ...
 $ X3         : num  23 19 18 24 25 23 20 19 22 20 ...
 $ X4         : num  23 22 21 22 25 20 22 19 20 20 ...
 $ X5         : num  19 19 21 20 21 20 19 19 18 19 ...
 $ X6         : num  7.8 7.8 7.9 7.9 8.4 7.8 7.5 7.5 7.6 7.6 ...
 $ X7         : num  32 32 35 32 35 33 32 32 31 31 ...
 $ X8         : num  33 40 38 37 43 37 35 37 35 35 ...
 $ X9         : num  5.6 5.8 6.2 5.9 6.6 6.3 5.5 6.2 5.3 5.7 ...
 $ Gender     : chr  "Male" "Male" "Male" "Male" ...
> canine_data <- data.frame(canine_data)
> canine_numb <- canine_data[2:10]
> #Converting the 2 character variables into categorical variables
> canine_data$CanineGroup <- as.factor(canine_data$CanineGroup)
> canine_data$Gender <- as.factor(canine_data$Gender)
> str(canine_data)
'data.frame':   77 obs. of  11 variables:
 $ CanineGroup: Factor w/ 5 levels "Cuons","GoldenJackal",..: 4 4 4 4 4 4 4 4 4 4 ...
 $ X1         : num  123 137 121 130 149 125 126 125 121 122 ...
 $ X2         : num  10.1 9.6 10.2 10.7 12 9.5 9.1 9.7 9.6 8.9 ...
 $ X3         : num  23 19 18 24 25 23 20 19 22 20 ...
 $ X4         : num  23 22 21 22 25 20 22 19 20 20 ...
 $ X5         : num  19 19 21 20 21 20 19 19 18 19 ...
 $ X6         : num  7.8 7.8 7.9 7.9 8.4 7.8 7.5 7.5 7.6 7.6 ...
 $ X7         : num  32 32 35 32 35 33 32 32 31 31 ...
 $ X8         : num  33 40 38 37 43 37 35 37 35 35 ...
 $ X9         : num  5.6 5.8 6.2 5.9 6.6 6.3 5.5 6.2 5.3 5.7 ...
 $ Gender     : Factor w/ 3 levels "Female","Male",..: 2 2 2 2 2 2 2 2 1 1 ...
```

Draftsman plot

```
> pairs(canine_numb, main = "Draftsman's plot of CanineGroup  ", pch = 21, bg = c("red", "green3", "blue","yellow","violet")
[unclass(canine$ï..CanineGroup)])
> #This type of image is also called a Draftsman's display - it shows the possible two-dimensional projections of
  multidimensional data (in this case, nine dimensional).
  #An actual engineer might use this to represent five dimensional physical objects of CanineGroup.
  #We can observe that  most of the variables could be used to predict the CanineGroup, although we observe there is
  poor relation between (x2,x3) and (x3,x4)
```



This type of image is also called a Draftsman's display - it shows the possible two-dimensional projections of multidimensional data (in this case, nine dimensional). An actual engineer might use this to represent five dimensional physical objects of Canine Group. We can observe that most of the variables could be used to predict the Canine Group, although we observe there is poor relation between (x2,x3) and (x3,x4)

Distance Matrix between the 5 canine groups

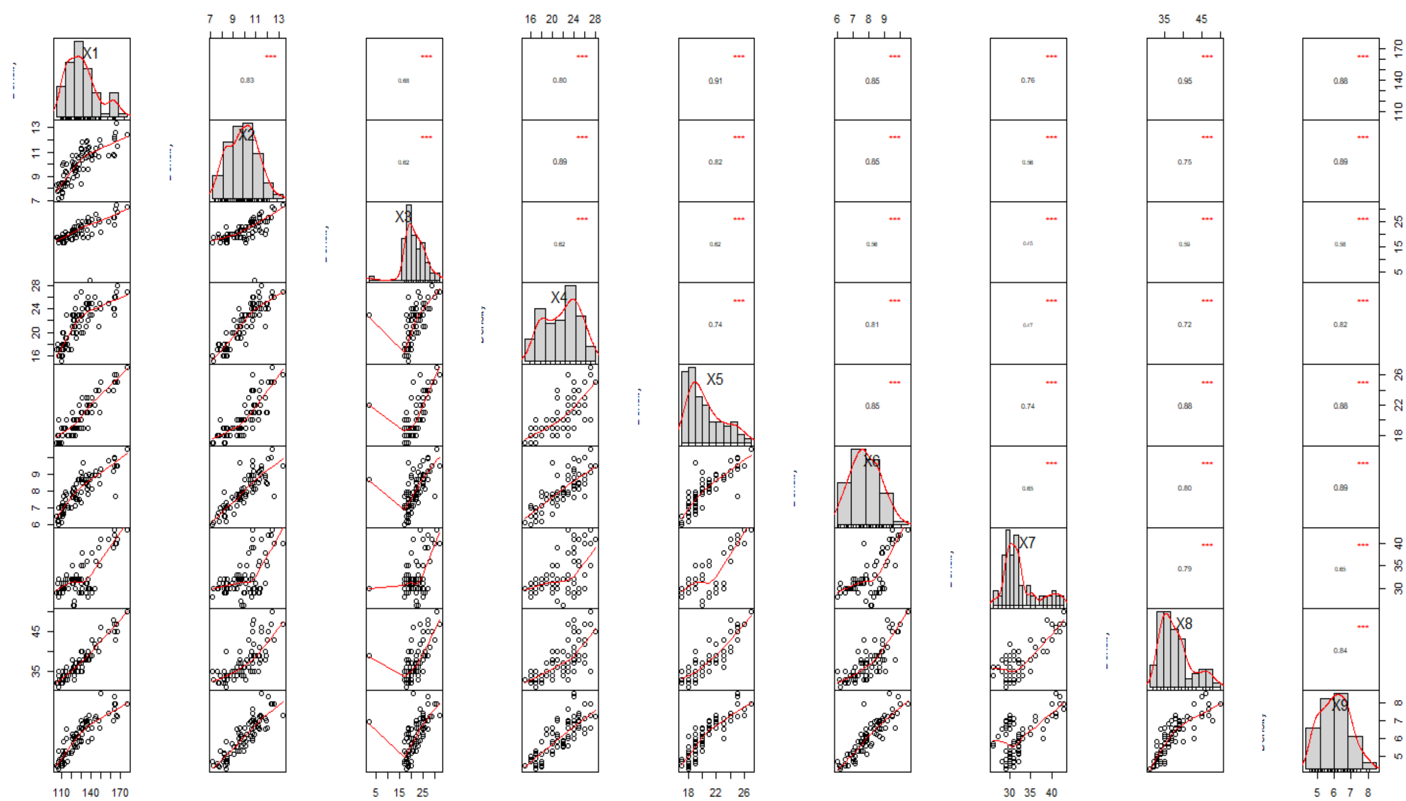
```
> means.df <- aggregate(as.matrix(canine_data[,2:10]),list(ï..CanineGroup), mean)
> # Creates a list carrying the covariance matrix for all the variables, by ï..CanineGroup
> covs.list <- by(canine_data[,2:10],ï..CanineGroup,cov)
> covs.list
ï..CanineGroup: Cuons
          X1        X2         X3        X4         X5         X6       X7         X8         X9
X1 41.316176 2.6191176 -1.3492647 7.4007353  3.5698529  1.0404412  7.06250 10.6360294  2.2783088
X2  2.619118 0.3706618  0.2963235 0.3713235  0.3694853  0.1309191  0.70625  0.5823529  0.2103309
X3 -1.349265 0.2963235 30.8676471 2.4301471 -0.1985294 -0.1294118 -0.56250 -0.5477941 -0.1955882
X4  7.400735 0.3713235  2.4301471 2.4926471  0.4264706  0.1643382  0.93750  2.0147059  0.3044118
X5  3.569853 0.3694853 -0.1985294 0.4264706  1.0147059  0.2496324  1.31250  0.8345588  0.3253676
X6  1.040441 0.1309191 -0.1294118 0.1643382  0.2496324  0.1173529  0.32500  0.2400735  0.1151471
X7  7.062500 0.7062500 -0.5625000 0.9375000  1.3125000  0.3250000  2.25000  1.7500000  0.6125000
X8 10.636029 0.5823529 -0.5477941 2.0147059  0.8345588  0.2400735  1.75000  3.7205882  0.6286765
X9  2.278309 0.2103309 -0.1955882 0.3044118  0.3253676  0.1151471  0.61250  0.6286765  0.2286029
------------------------------------------------------------------------------------------
ï..CanineGroup: GoldenJackal
           X1         X2          X3         X4          X5         X6        X7          X8         X9
X1 15.0526316 0.80000000  1.52631579 1.10526316  0.68421053 1.11578947 2.2631579  2.15789474 0.64736842
X2  0.8000000 0.25326316  0.19684211 0.23684211  0.15684211 0.15663158 0.1494737  0.02842105 0.06905263
X3  1.5263158 0.19684211  1.30526316 0.52631579 -0.07368421 0.12210526 0.2526316 -0.11578947 0.04947368
X4  1.1052632 0.23684211  0.52631579 0.94736842  0.36842105 0.21578947 0.4736842  0.05263158 0.05263158
X5  0.6842105 0.15684211 -0.07368421 0.36842105  0.69473684 0.26526316 0.6105263  0.13684211 0.13578947
X6  1.1157895 0.15663158  0.12210526 0.21578947  0.26526316 0.23081579 0.3628947  0.12078947 0.09939474
X7  2.2631579 0.14947368  0.25263158 0.47368421  0.61052632 0.36289474 1.2921053  0.13421053 0.17184211
X8  2.1578947 0.02842105 -0.11578947 0.05263158  0.13684211 0.12078947 0.1342105  1.39736842 0.21921053
X9  0.6473684 0.06905263  0.04947368 0.05263158  0.13578947 0.09939474 0.1718421  0.21921053 0.09734211
------------------------------------------------------------------------------------------
ï..CanineGroup: IndianWolves
           X1         X2         X3         X4         X5         X6         X7         X8        X9
X1 156.401099 4.84670330 37.1483516 14.6483516 11.9560440 3.80164835 18.9945055 30.5439560 4.9203297
X2   4.846703 0.80489011  2.1203297  1.1703297  0.5703297 0.05851648  0.4664835  0.7873626 0.2717033
X3  37.148352 2.12032967 14.9505495  4.6043956  2.8351648 0.52252747  1.8736264  4.5879121 0.9060440
X4  14.648352 1.17032967  4.6043956  3.6043956  1.1428571 0.36483516  1.3736264  2.9340659 0.3637363
X5  11.956044 0.57032967  2.8351648  1.1428571  1.2967033 0.37252747  1.7582418  2.5494505 0.5175824
X6   3.801648 0.05851648  0.5225275  0.3648352  0.3725275 0.44554945  0.7763736  1.3005495 0.1458791
X7  18.994505 0.46648352  1.8736264  1.3736264  1.7582418 0.77637363  4.1813187  4.5109890 0.9214286
X8  30.543956 0.78736264  4.5879121  2.9340659  2.5494505 1.30054945  4.5109890  9.2582418 0.9785714
X9   4.920330 0.27170330  0.9060440  0.3637363  0.5175824 0.14587912  0.9214286  0.9785714 0.4607143
------------------------------------------------------------------------------------------
```

```
ï..CanineGroup: ModernDog
          X1        X2         X3        X4        X5        X6        X7         X8        X9
X1 72.329167 4.3350000 10.2916667 9.8083333 2.4250000 2.1450000 5.2708333 19.9083333 1.4979167
X2  4.335000 0.7180000  0.9966667 1.0433333 0.4766667 0.2893333 0.7983333  0.9300000 0.1828333
X3 10.291667 0.9966667  5.3166667 2.2166667 0.3833333 0.5833333 0.3083333  1.6166667 0.1858333
X4  9.808333 1.0433333  2.2166667 2.6500000 0.5500000 0.4633333 0.9250000  1.6500000 0.1108333
X5  2.425000 0.4766667  0.3833333 0.5500000 0.7833333 0.2700000 1.1083333  0.9500000 0.2925000
X6  2.145000 0.2893333  0.5833333 0.4633333 0.2700000 0.2313333 0.4616667  0.4166667 0.1385000
X7  5.270833 0.7983333  0.3083333 0.9250000 1.1083333 0.4616667 2.0625000  1.9583333 0.4887500
X8 19.908333 0.9300000  1.6166667 1.6500000 0.9500000 0.4166667 1.9583333  7.4500000 0.5741667
X9  1.497917 0.1828333  0.1858333 0.1108333 0.2925000 0.1385000 0.4887500  0.5741667 0.1796250
------------------------------------------------------------------------------------------
ï..CanineGroup: ThaiDogs
          X1        X2         X3         X4        X5        X6         X7         X8        X9
X1 70.844444 3.3311111 10.3333333 13.8666667 4.5111111 4.8755556 15.8444444 12.5333333 3.7822222
X2  3.331111 0.5937778  0.8000000  1.2822222 0.5533333 0.1571111  0.9088889  0.4044444 0.2268889
X3 10.333333 0.8000000  3.7777778  3.7777778 0.3333333 0.4333333  1.8888889  0.8888889 0.4888889
X4 13.866667 1.2822222  3.7777778  8.1000000 1.5888889 0.5433333  3.2000000  2.3222222 0.7522222
X5  4.511111 0.5533333  0.3333333  1.5888889 0.9000000 0.2811111  1.5111111  1.0333333 0.3322222
X6  4.875556 0.1571111  0.4333333  0.5433333 0.2811111 0.5498889  1.2755556  0.7211111 0.2796667
X7 15.844444 0.9088889  1.8888889  3.2000000 1.5111111 1.2755556  4.4000000  2.8666667 0.9822222
X8 12.533333 0.4044444  0.8888889  2.3222222 1.0333333 0.7211111  2.8666667  3.4333333 0.6633333
X9  3.782222 0.2268889  0.4888889  0.7522222 0.3322222 0.2796667  0.9822222  0.6633333 0.2290000
> n <- nrow(canine_data)       ## Number of sampling units
> p <- ncol(canine_data) - 2   ## Number of variables
> m <- nlevels(ï..CanineGroup)     ## Number of groups
> CanineGroup.list <- table(ï..CanineGroup)  # Number of observations per ï..CanineGroup
> library(heplots)
> V.pool <- boxM(canine_data[,2:10],ï..CanineGroup)$pooled  # Pooled Covariance matrix
> V.pool
          X1        X2         X3        X4        X5        X6        X7         X8        X9
X1 65.316814 2.9877504 10.2460463 8.3578519 4.2016836 2.2683818 8.6748760 14.1621236 2.3503608
X2  2.987750 0.5183356  0.8082703 0.7339648 0.3949451 0.1609086 0.5605456  0.5233800 0.1804713
X3 10.246046 0.8082703 11.4831874 2.4442986 0.5698704 0.2735037 0.5803075  1.1240021 0.2330092
X4  8.357852 0.7339648  2.4442986 3.0192986 0.7115371 0.3237815 1.1740575  1.6253910 0.2643286
X5  4.201684 0.3949451  0.5698704 0.7115371 0.9186450 0.2841246 1.1900298  1.0089694 0.3040549
X6  2.268382 0.1609086  0.2735037 0.3237815 0.2841246 0.2843651 0.5637897  0.4969905 0.1419692
X7  8.674876 0.5605456  0.5803075 1.1740575 1.1900298 0.5637897 2.5756200  2.0051091 0.5724281
X8 14.162124 0.5233800  1.1240021 1.6253910 1.0089694 0.4969905 2.0051091  4.8484244 0.5767743
X9  2.350361 0.1804713  0.2330092 0.2643286 0.3040549 0.1419692 0.5724281  0.5767743 0.2257196
> P <- matrix(rep(0,m*m),nrow=m,ncol=m)  # Initializing Penrose's distance matrix
> for (j in 1:m)
+ { for (i in 1:m)
+ { if (i==j) {P[i,i]=0 }  else
+ { for (k in 1:p) {
+   P[i,j] <- P[i,j]+(((means.df[i,(k+1)]-means.df[j,(k+1)])^2)/(p*V.pool[k,k]))
+ }
+ }
+ }

> # Creating a random order for the Matrix
> P <- P[,c(2,3,1,4,5)]
> P <- P[c(2,3,1,4,5),]
> colnames(P) <- levels(ï..CanineGroup)[c(2,3,1,4,5)]
> rownames(P) <- levels(ï..CanineGroup)[c(2,3,1,4,5)]
> (P.Dist <- as.dist(P)) # Penrose distances
             GoldenJackal IndianWolves      Cuons  ModernDog
IndianWolves   27.0257232
Cuons           8.5203838    9.7501462
ModernDog       2.9772469   13.2001037  2.0467669
ThaiDogs        4.7471934   11.8479023  1.7034588  0.4165551
```

It quantifies dissimilarity between sample data for numerical computation. As the distance between ThaiDogs and ModernDog (0.4165551) is smaller than the distance between IndianWolves and GoldenJackal (27.025723).we conclude that ThaiDogs and ModernDog are more similar and IndianWolves and GoldenJackal are very different from one another.

Principal components analysis

```
> #3 PCA
> canine_num<-canine[,-c(1,11)]
> canine_corr<-cor(canine_numb)
> canine_corr
          X1        X2        X3        X4        X5        X6        X7        X8        X9
X1 1.0000000 0.8259623 0.6841756 0.7976348 0.9066471 0.8515578 0.7589012 0.9494620 0.8833714
X2 0.8259623 1.0000000 0.6184360 0.8897336 0.8213389 0.8457847 0.5597767 0.7460676 0.8874866
X3 0.6841756 0.6184360 1.0000000 0.6200059 0.6166557 0.5608910 0.4516023 0.5906419 0.5750451
X4 0.7976348 0.8897336 0.6200059 1.0000000 0.7402734 0.8085781 0.4707245 0.7151408 0.8229495
X5 0.9066471 0.8213389 0.6166557 0.7402734 1.0000000 0.8537794 0.7424201 0.8777774 0.8826925
X6 0.8515578 0.8457847 0.5608910 0.8085781 0.8537794 1.0000000 0.6456683 0.7984086 0.8942284
X7 0.7589012 0.5597767 0.4516023 0.4707245 0.7424201 0.6456683 1.0000000 0.7867110 0.6478342
X8 0.9494620 0.7460676 0.5906419 0.7151408 0.8777774 0.7984086 0.7867110 1.0000000 0.8380353
X9 0.8833714 0.8874866 0.5750451 0.8229495 0.8826925 0.8942284 0.6478342 0.8380353 1.0000000
> library("PerformanceAnalytics")
> #Get the Correlations between the measurements
> chart.Correlation(canine_numb, histogram=TRUE, pch=19)
```



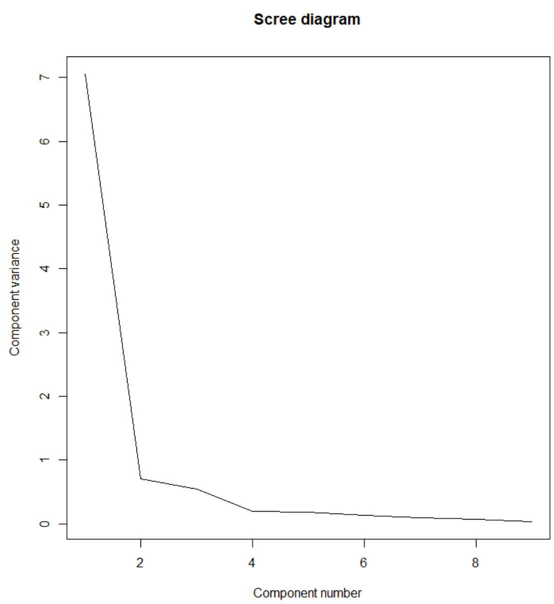
We can observe that there is good positive correlation between the following variables (x1 and x8) and (X1 and X5).now we Find the principal components of data Using prcomp to compute the principal components (eigenvalues and eigenvectors). With scale=TRUE, means are set to zero, and variances set to one

```
> canine_pca <- prcomp(canine_num,scale=TRUE)
> canine_pca
Standard deviations (1, .., p=9):
[1] 2.6555963 0.8391652 0.7365758 0.4390554 0.4241988 0.3627806 0.3031519 0.2652189 0.1857418

Rotation (n x k) = (9 x 9):
         PC1         PC2         PC3         PC4         PC5         PC6         PC7         PC8          PC9
X1 0.3636408 -0.11451510  0.08210471 -0.30326354  0.24950692 -0.07899550  0.05543869  0.16005914  0.811637429
X2 0.3424554  0.31490128 -0.19979188  0.33605928  0.01517931  0.49451257  0.13657790  0.60411640 -0.048224206
X3 0.2665621  0.32018675  0.87894338  0.04161625 -0.18169514 -0.04568559  0.08257828 -0.03476461 -0.094992855
X4 0.3265349  0.44638084 -0.16540131  0.26534253  0.54545187 -0.21526217 -0.30849700 -0.39244126 -0.057608736
X5 0.3539586 -0.14160855 -0.03861441 -0.26352534 -0.33012092  0.43239890 -0.67024916 -0.19081879 -0.046144083
X6 0.3459444  0.06792334 -0.26250857  0.05378069 -0.51974026 -0.68294862 -0.08734948  0.23986950 -0.046794522
X7 0.2859405 -0.68736531  0.13651981  0.64014932  0.05443187 -0.01170970  0.03463451 -0.11415807  0.002559605
X8 0.3470802 -0.28877388  0.03666665 -0.47256682  0.40753260 -0.10978603  0.12889717  0.23397418 -0.567438948
X9 0.3544268  0.07362113 -0.25111557 -0.13231892 -0.24254817  0.18765482  0.63372357 -0.54081471 -0.016253801
> summary(canine_pca)
Importance of components:
                          PC1     PC2     PC3     PC4     PC5     PC6     PC7     PC8     PC9
Standard deviation     2.6556 0.83917 0.73658 0.43906 0.42420 0.36278 0.30315 0.26522 0.18574
Proportion of Variance 0.7836 0.07824 0.06028 0.02142 0.01999 0.01462 0.01021 0.00782 0.00383
Cumulative Proportion  0.7836 0.86182 0.92210 0.94352 0.96352 0.97814 0.98835 0.99617 1.00000
```

According to me when we observe the SD it keeps decreasing and comparing the SD of third component and fourth component we can see more differential decrease in the fourth component. Hence i am considering to take pc1 to pc3 but however we cannot conclude this at this step. We can decide as to how many PC's to consider by doing further analysis.

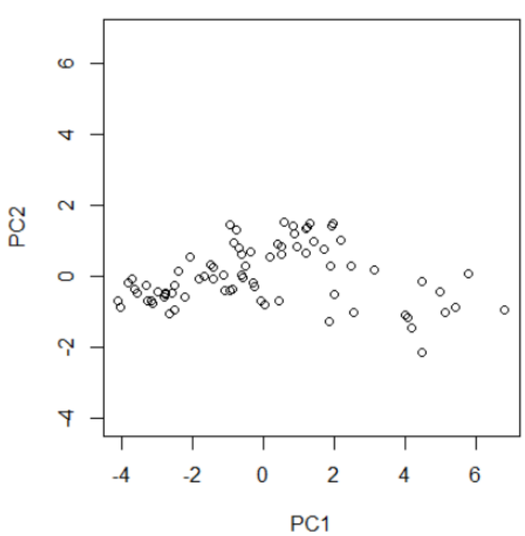
```
> #Sample scores
> canine_pca$x
              PC1          PC2          PC3          PC4          PC5          PC6          PC7          PC8           PC9
 [1,] -0.68594144  0.784505851  0.300680848  0.870470169  0.099293422 -0.137033941 -0.125743045 -0.081031870  0.2729956129
 [2,] -0.24567287 -0.299843248 -0.338698392 -0.386884246  0.904531523 -0.407035612  0.212017051  0.246639751  0.1416735360
 [3,] -0.08646546 -0.709903195 -0.732453524  0.362159600  0.007597324  0.346844095 -0.040422717 -0.047202480 -0.3698635173
 [4,]  0.16794155  0.526391452  0.423868541  0.255568728  0.116376021  0.158174388  0.091556859  0.349426697  0.0283217760
 [5,]  2.46605602  0.305545132  0.221535557  0.129388655  0.883927889  0.139528740  0.347602065  0.632082729 -0.0326066925
 [6,] -0.31165997 -0.192975153  0.420517896 -0.015940817 -0.279041559  0.033596480  0.387384912 -0.235340589 -0.1068470218
 [7,] -1.13265883  0.022117338 -0.001351901  0.253430159  0.460259705 -0.274897014 -0.159067856 -0.254279988  0.2888227418
 [8,] -0.95932707 -0.389654503 -0.308889300 -0.136456734  0.029737576  0.222079003  0.644115631  0.086676248  0.0267191553
 [9,] -1.42772439  0.268517876  0.425436242  0.306100767  0.101550728 -0.244693163  0.223639848  0.364631603  0.0453265447
[10,] -1.42312985 -0.075076506 -0.001323952 -0.056378966 -0.033258294 -0.226990915  0.098904547 -0.199911701  0.1359348236
[11,] -1.08479346 -0.397357243 -0.457414517  0.287286817 -0.737919911  0.226905194  0.385692502 -0.529169838 -0.0732135119
[12,] -2.41795380  0.145447343  0.069556048  0.241403411  0.321812064  0.508873829 -0.287284089 -0.191979914  0.0001707043
[13,] -1.49439221  0.318157266  0.279280030 -0.061989237  0.631422502 -0.068589173  0.280944711  0.012209311  0.0904772991
[14,] -0.59855768 -0.033957848  0.354006167 -0.081049692  0.452690662 -0.085214200  0.300688841  0.079224385 -0.0548985944
[15,] -0.86184015 -0.352203858  0.827012220 -0.850306862  0.537179718 -0.377924804  0.368420383 -0.224527764 -0.1755114316
[16,]  0.39898643  0.907891121  0.326817698  0.578720272 -0.448994514 -0.420413811 -0.011638914  0.086598079  0.0743710810
[17,] -2.50221460 -0.951253968  0.143968142 -0.152184938  0.101657301 -0.120008246  0.277133051  0.019273046  0.2180306195
[18,] -2.66854242 -1.050460703 -0.146821799 -0.126008083 -0.425257747  0.234390985 -0.318289295 -0.479958855 -0.2618149293
[19,] -3.13382930 -0.765330804  0.184550458  0.051337298 -0.558478957  0.046101042 -0.357483655  0.080924412  0.1455078947
[20,] -2.58652273 -0.471999686  0.531478241  0.371990091  0.006237923 -0.190501859 -0.136270365  0.161573466  0.2211361243
[21,] -2.21301819 -0.577129122  0.037091946  0.209207907 -0.616304017 -0.562841172 -0.280239562 -0.078962489  0.0997622246
[22,] -2.97976934 -0.428419991  0.258074597 -0.054107753 -0.442317831  0.031176138  0.180789986  0.252051767  0.0399036745
[23,] -2.78947778 -0.480480879 -0.156266128 -0.112624135  0.050455054 -0.008507846 -0.523207692  0.243127587  0.0026450138
[24,] -2.53185477 -0.272427223  0.402052583 -0.160236215 -0.140539836  0.078620315  0.309627989  0.192229345  0.1442393490
[25,] -1.67354577 -0.002191241  0.271619096  0.179981134 -0.200562634  0.031336373 -0.006904838  0.238858283 -0.2467850771
[26,] -2.78123568 -0.505269212  0.301645011 -0.306596708 -0.181358119  0.224472607 -0.088724900 -0.130998186 -0.0557006989
[27,] -2.82501243 -0.601166102  0.063341427  0.087204136 -0.296959874  0.201959635 -0.247776574  0.060596147 -0.0127365734
[28,] -3.30622428 -0.264797684  0.700011860  0.072398198  0.058353044 -0.189309999 -0.366013843 -0.214676412  0.1658330818
[29,] -4.10431266 -0.685413059  0.329995467 -0.750369799  0.494202102  0.073491063  0.220890191 -0.188848219 -0.2411101560
[30,] -3.66153536 -0.384750833  0.264067753 -0.137014973  0.026516965  0.142466843  0.310255014  0.173159393 -0.0178344266
[31,] -4.04399515 -0.868937592  0.868308006 -0.284587131  0.128201995  0.077954476  0.195885234 -0.050127321  0.1240428684
[32,] -3.72261526 -0.090659528  0.460905050  0.173559701  0.082024673  0.086404835  0.023048601  0.170350291 -0.0663817666
[33,] -3.82745453 -0.182930525  0.349841098  0.164631204  0.129526936  0.485377688 -0.377602490  0.145975138  0.1726660135
[34,] -3.58415662 -0.490741820  0.639320364  0.313237549  0.321961493  0.191714584 -0.402894707 -0.300579061 -0.0218718055
[35,] -3.18995154 -0.696670292 -0.036956395 -0.171421289 -0.142519719  0.004192919 -0.016782296  0.414288952 -0.0310504434
[36,] -3.28698703 -0.698090489  0.606290706 -0.385321616  0.294238242 -0.072930907 -0.137014078 -0.073856123 -0.1700624512
[37,] -0.83046450  0.960679604 -0.030247176 -0.651965865 -0.224025394  0.065189126  0.086237451 -0.058331937 -0.0726510951
[38,]  1.69865955  0.765958005 -0.074666591 -0.317633490 -1.116429070  0.628699956  0.199311596  0.425185706 -0.0329738188
[39,]  1.94265070  1.427794661 -0.291595138 -0.219515432 -0.410925436  0.018527480  0.181207300 -0.162960817  0.0618406487
[40,]  1.24244782  1.395257107  0.410773608 -0.516661632  0.417743578  0.058524804  0.083783879 -0.084648374  0.1404381072
[41,]  1.32210362  1.482079235 -0.010095969 -0.318722580  0.152772606 -0.011028610  0.112138469  0.081654992 -0.1488954218
[42,]  1.18642807  0.664925109 -0.425761400 -0.325116618  0.205555050  0.463511525 -0.027906424 -0.132509659 -0.0115849232
[43,]  0.94267205  0.851161769 -0.333315206 -0.356732912 -0.804617659  0.178416762 -0.407686760 -0.396695387  0.1115834185
[44,]  0.83611277  1.404765607  0.193189579 -0.606327224  0.056873910 -0.043452998  0.006595990 -0.008781152  0.1220724606
[45,]  1.18696596  1.332771923  0.065975881 -0.517597040  0.148398971 -0.154230922  0.176826143 -0.349259454 -0.1234519117
[46,]  0.58012526  1.534916368  0.152658482  0.094207689 -0.320122532 -0.027306038 -0.266206667  0.084782037  0.2162431554
[47,]  0.87381349  1.211079378 -0.717359562 -0.208704416 -0.468972702  0.178381526  0.340752335  0.117599024 -0.0387497591
[48,]  2.16615489  1.026720501 -0.327335755 -0.806968307  0.341134833 -0.342703982  0.032176487 -0.179247133 -0.1413451648
[49,]  0.03166471 -0.817033069 -4.730882852 -0.811969172  0.606573954  0.316172197 -0.326054707  0.247940211  0.5692006458
[50,] -0.94467005  1.470900068  0.111942625 -0.397789900 -0.336060824 -0.213636291 -0.378925698  0.116905546  0.1356105362
[51,]  1.96212575  1.495891969  0.352870519 -0.252907524 -0.048463694  0.104035427 -0.595319658 -0.017350658 -0.1641215423
[52,]  0.51806252  0.846638067 -0.458621119 -0.539356774 -0.548977290 -0.170758746 -0.309890866 -0.324932921 -0.0989702421
[53,] -0.77814254  1.306647386 -0.149169933 -0.588680141 -0.195636728 -0.261198034 -0.279811080  0.235701059 -0.1554762328
[54,] -1.81012401 -0.087546315 -0.813788086  0.424454055 -0.616037706  0.402606323  0.302997079  0.346589006 -0.0158409581
[55,] -0.60672494  0.045127712 -0.821728445  0.609585990  0.262622151  0.087709426 -0.183272172 -0.312787291 -0.3926294064
[56,]  1.90795558  0.273257118 -0.575618216  0.836979522  0.323898131  0.299268427  0.369212391  0.066333383 -0.0594482901
[57,] -2.06926137  0.548474236 -0.286703322  0.306301228  0.125281859  0.169387613  0.142118818  0.358402792 -0.2197954502
[58,]  1.40688395  0.990435862 -0.499318182  1.405235914  0.050567519 -0.603712611 -0.036969458 -0.286288447  0.1418842467
[59,]  0.51098957  0.605122186 -1.032247893  0.734808799  0.570580780 -0.282725129 -0.173322187 -0.242096000 -0.1876828266
[60,]  0.42526384 -0.710658443 -0.866968900  0.213487092 -0.790563161 -1.335966923  0.680262010  0.265867445  0.1037942927
[61,] -0.37712553  0.684703380 -0.318762771  0.767866490  0.202625397  0.062199059  0.185727352  0.093048358 -0.0638930186
[62,] -0.62016146  0.616223177  0.013529855  0.637693372  0.272817699 -0.425615800  0.452191851 -0.269986420  0.0078125854
[63,] -0.51807155  0.300880391 -0.297750137  0.266793365  0.774354723 -0.276882619  0.051622197 -0.468200141 -0.1219739770
[64,]  4.97863584 -0.454803912  0.785610331  0.235420747  0.478889683 -0.348036457 -0.528953300 -0.225017315  0.2900530114
[65,]  5.43014384 -0.883989516  0.087185285  0.176453527 -0.014214157 -0.160809944  0.142080746  0.165642852 -0.0884300972
[66,]  4.19582141 -1.464914916 -1.027294217 -0.149694082 -0.179591390  0.418060354  0.479996535 -0.533022390 -0.3901772539
[67,]  3.12145386  0.162573562  0.593731104 -0.003220108 -0.590094975  0.086736277 -0.136198230 -0.108731390 -0.0880741930
[68,]  6.83250997 -0.952533937  0.817662353 -0.260634910 -0.067185591 -0.355191335 -0.304856894  0.274317047 -0.0419962150
[69,]  5.78067966  0.057932801  1.128795919  0.172128008  0.210018562  0.502366877 -0.353846508  0.679301497 -0.1505834324
[70,]  5.13564203 -1.039798168  0.124081997  0.128863581 -0.568734521  0.086322716  0.503859779 -0.029943072  0.2083976892
[71,]  4.48437886 -0.156180195  1.165911651  0.262085497  0.645535041  1.547670358  0.295840088 -0.417486865  0.4992245855
```

```
[72,]  1.98423294 -0.525520052 -0.864042512  0.869896512 -0.003627085  0.396588690 -0.433031742  0.290842494 -0.3961524701
[73,]  3.99779940 -1.110955640  0.746491616 -0.709555926  0.438577340 -0.379596125 -0.042000621  0.244670880 -0.1103574972
[74,]  4.49728950 -2.143891251  0.055550197 -0.411812661 -0.180901653 -0.051846681 -0.203127945 -0.212060862  0.0950680774
[75,]  1.85001891 -1.263460559 -0.446342872  0.031012090  0.308041106 -0.311776220 -0.677800069  0.343626708 -0.2503114021
[76,]  2.55231795 -1.025526071  0.653272161  0.099458862 -0.806580392  0.219171629 -0.063113499 -0.117101389  0.3189186401
[77,]  4.08213029 -1.184586209  0.059285946  0.020237767 -0.037342784 -0.691839655 -0.071880511 -0.229417784  0.0431334575
> #Singular values (square roots of eigenvalues)
> canine_pca$sdev
[1] 2.6555963 0.8391652 0.7365758 0.4390554 0.4241988 0.3627806 0.3031519 0.2652189 0.1857418
> #Loadings (eigenvectors)
> canine_pca$rotation
         PC1         PC2         PC3         PC4         PC5         PC6         PC7         PC8          PC9
X1 0.3636408 -0.11451510  0.08210471 -0.30326354  0.24950692 -0.07899550  0.05543869  0.16005914  0.811637429
X2 0.3424554  0.31490128 -0.19979188  0.33605928  0.01517931  0.49451257  0.13657790  0.60411640 -0.048224206
X3 0.2665621  0.32018675  0.87894338  0.04161625 -0.18169514 -0.04568559  0.08257828 -0.03476461 -0.094992855
X4 0.3265349  0.44638084 -0.16540131  0.26534253  0.54545187 -0.21526217 -0.30849700 -0.39244126 -0.057608736
X5 0.3539586 -0.14160855 -0.03861441 -0.26352534 -0.33012092  0.43239890 -0.67024916 -0.19081879 -0.046144083
X6 0.3459444  0.06792334 -0.26250857  0.05378069 -0.51974026 -0.68294862 -0.08734948  0.23986950 -0.046794522
X7 0.2859405 -0.68736531  0.13651981  0.64014932  0.05443187 -0.01170970  0.03463451 -0.11415807  0.002559605
X8 0.3470802 -0.28877388  0.03666665 -0.47256682  0.40753260 -0.10978603  0.12889717  0.23397418 -0.567438948
X9 0.3544268  0.07362113 -0.25111557 -0.13231892 -0.24254817  0.18765482  0.63372357 -0.54081471 -0.016253801
> #Variable means
> canine_pca$center
        X1         X2         X3         X4         X5         X6         X7         X8         X9
128.974026   9.961039  21.636364  21.493506  20.493506   8.000000  32.519481  37.402597   6.075325
> #Variable standard deviations
> canine_pca$scale
       X1        X2        X3        X4        X5        X6        X7        X8        X9
17.501860  1.403019  4.214352  3.378038  2.490103  1.023667  4.172625  4.404722  1.019695
> #Extract variance against features
> eigenvalues<-canine_pca$sdev^2
> eigenvalues
[1] 7.05219159 0.70419829 0.54254392 0.19276967 0.17994462 0.13160975 0.09190108 0.07034106 0.03450002
> sum(eigenvalues)
[1] 9
> names(eigenvalues) <- paste("PC",1:9,sep="")
> eigenvalues
       PC1        PC2        PC3        PC4        PC5        PC6        PC7        PC8        PC9
7.05219159 0.70419829 0.54254392 0.19276967 0.17994462 0.13160975 0.09190108 0.07034106 0.03450002
> sumoflambdas <- sum(eigenvalues)
> sumoflambdas
[1] 9
> #Variance %
> p_cvar<- (eigenvalues/sumoflambdas)*100
> round(p_cvar,4)
    PC1     PC2     PC3     PC4     PC5     PC6     PC7     PC8     PC9
78.3577  7.8244  6.0283  2.1419  1.9994  1.4623  1.0211  0.7816  0.3833
> #Calculate cumulative of variance
> cumvar <- cumsum(p_cvar)
> cumvar
      PC1       PC2       PC3       PC4       PC5       PC6       PC7       PC8       PC9
 78.35768  86.18211  92.21038  94.35226  96.35165  97.81398  98.83510  99.61667 100.00000
> matlambdas <- rbind(eigenvalues,p_cvar,cumvar)
> round(matlambdas,4)
                PC1     PC2     PC3     PC4     PC5     PC6     PC7     PC8      PC9
eigenvalues  7.0522  0.7042  0.5425  0.1928  0.1799  0.1316  0.0919  0.0703   0.0345
p_cvar      78.3577  7.8244  6.0283  2.1419  1.9994  1.4623  1.0211  0.7816   0.3833
cumvar      78.3577 86.1821 92.2104 94.3523 96.3516 97.8140 98.8351 99.6167 100.0000
> eigenvec_Protein <- canine_pca$rotation
> eigenvec_Protein
         PC1         PC2         PC3         PC4         PC5         PC6         PC7         PC8          PC9
X1 0.3636408 -0.11451510  0.08210471 -0.30326354  0.24950692 -0.07899550  0.05543869  0.16005914  0.811637429
X2 0.3424554  0.31490128 -0.19979188  0.33605928  0.01517931  0.49451257  0.13657790  0.60411640 -0.048224206
X3 0.2665621  0.32018675  0.87894338  0.04161625 -0.18169514 -0.04568559  0.08257828 -0.03476461 -0.094992855
X4 0.3265349  0.44638084 -0.16540131  0.26534253  0.54545187 -0.21526217 -0.30849700 -0.39244126 -0.057608736
X5 0.3539586 -0.14160855 -0.03861441 -0.26352534 -0.33012092  0.43239890 -0.67024916 -0.19081879 -0.046144083
X6 0.3459444  0.06792334 -0.26250857  0.05378069 -0.51974026 -0.68294862 -0.08734948  0.23986950 -0.046794522
X7 0.2859405 -0.68736531  0.13651981  0.64014932  0.05443187 -0.01170970  0.03463451 -0.11415807  0.002559605
X8 0.3470802 -0.28877388  0.03666665 -0.47256682  0.40753260 -0.10978603  0.12889717  0.23397418 -0.567438948
X9 0.3544268  0.07362113 -0.25111557 -0.13231892 -0.24254817  0.18765482  0.63372357 -0.54081471 -0.016253801

> w<-canine_pca$x
> head(w)
             PC1        PC2        PC3         PC4          PC5         PC6         PC7         PC8         PC9
[1,] -0.68594144  0.7845059  0.3006808  0.87047017  0.099293422 -0.13703394 -0.12574305 -0.08103187  0.27299561
[2,] -0.24567287 -0.2998432 -0.3386984 -0.38688425  0.904531523 -0.40703561  0.21201705  0.24663975  0.14167354
[3,] -0.08646546 -0.7099032 -0.7324535  0.36215960  0.007597324  0.34684409 -0.04042272 -0.04720248 -0.36986352
[4,]  0.16794155  0.5263915  0.4238685  0.25556873  0.116376021  0.15817439  0.09155686  0.34942670  0.02832178
[5,]  2.46605602  0.3055451  0.2215356  0.12938866  0.883927889  0.13952874  0.34760207  0.63208273 -0.03260669
[6,] -0.31165997 -0.1929752  0.4205179 -0.01594082 -0.279041559  0.03359648  0.38738491 -0.23534059 -0.10684702
> # This tells how many PC'S can we consider for our analysis.
> plot(eigenvalues, xlab = "Component number", ylab = "Component variance", type = "l", main = "Scree diagram")
```

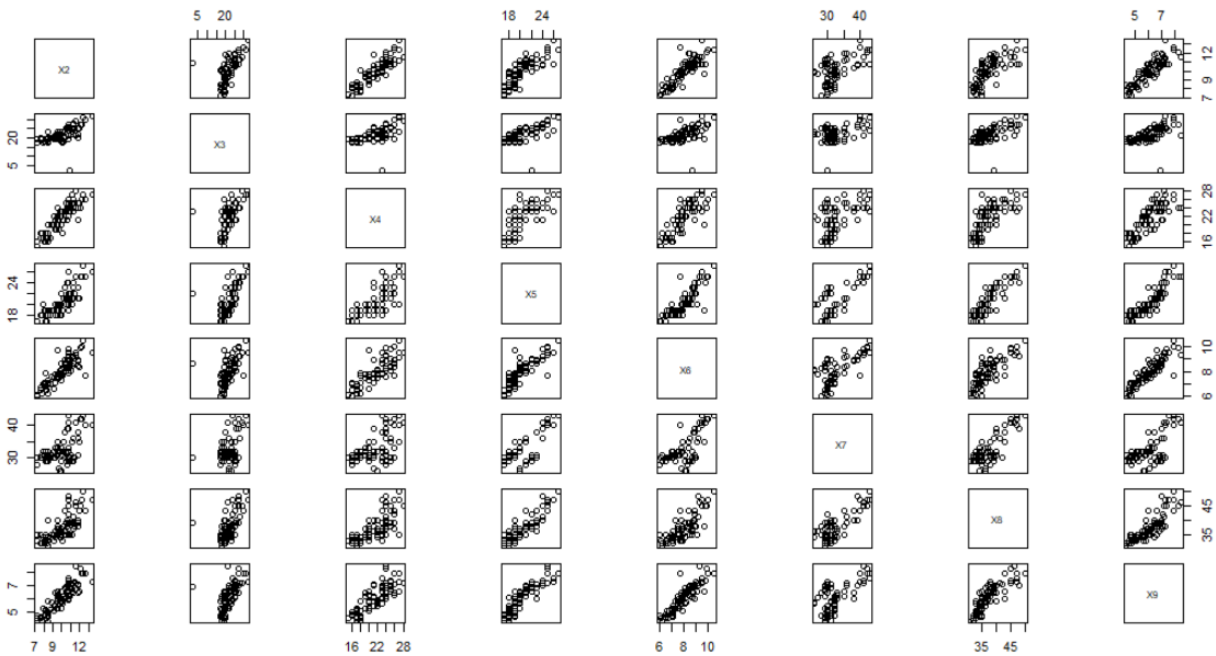


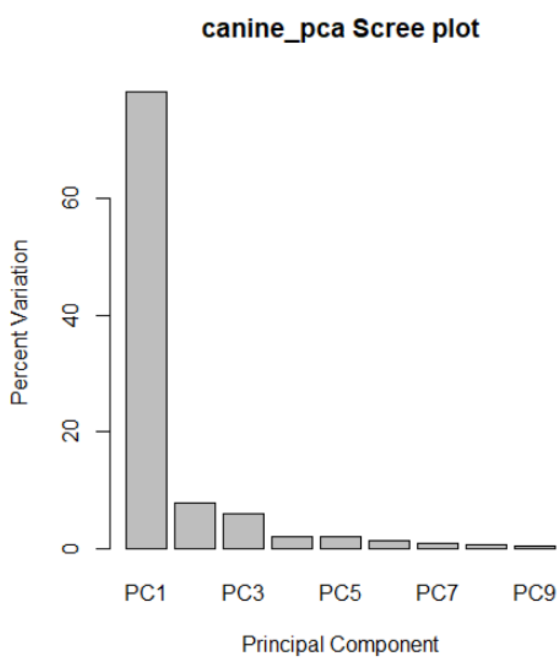
#We see that the curve from component number 3 the slop decreases with less change, taking into consideration the standard variation of the PC'S we can see that the SD variation is not much from PC1 TO PC3 But the decrease from PC3 to PC4 and further on ,so i decided to choose only Three PC's

```
> print(summary(canine_pca))
Importance of components:
                          PC1     PC2     PC3     PC4     PC5     PC6     PC7     PC8     PC9
Standard deviation     2.6556 0.83917 0.73658 0.43906 0.42420 0.36278 0.30315 0.26522 0.18574
Proportion of Variance 0.7836 0.07824 0.06028 0.02142 0.01999 0.01462 0.01021 0.00782 0.00383
Cumulative Proportion  0.7836 0.86182 0.92210 0.94352 0.96352 0.97814 0.98835 0.99617 1.00000
> View(canine_pca)
> diag(cov(canine_pca$x))
       PC1        PC2        PC3        PC4        PC5        PC6        PC7        PC8        PC9 
7.05219159 0.70419829 0.54254392 0.19276967 0.17994462 0.13160975 0.09190108 0.07034106 0.03450002 
> xlim <- range(canine_pca$x[,1])
> canine_pca$x[,1]
 [1] -0.68594144 -0.24567287 -0.08646546  0.16794155  2.46605602 -0.31165997 -1.13265883 -0.95932707 -1.42772439 -1.42312985 -1.08479346 -2.41795380 -1.49439221
[14] -0.59855768 -0.86184015  0.39898643 -2.50221460 -2.66854242 -3.13382930 -2.58652273 -2.21301819 -2.97976934 -2.78947778 -2.53185477 -1.67354577 -2.78123568
[27] -2.82501243 -3.30622428 -4.10431266 -3.66153536 -4.04399515 -3.72261526 -3.82745453 -3.58415662 -3.18995154 -3.28698703 -0.83046450  1.69865955  1.94265070
[40]  1.24244782  1.32210362  1.18642807  0.94267205  0.83611277  1.18696596  0.58012526  0.87381349  2.16615489  0.03166471 -0.94467005  1.96212575  0.51806252
[53] -0.77814254 -1.81012401 -0.60672494  1.90795558 -2.06926137  1.40688395  0.51098957  0.42526384 -0.37712553 -0.62016146 -0.51807155  4.97863584  5.43014384
[66]  4.19582141  3.12145386  6.83250997  5.78067966  5.13564203  4.48437886  1.98423294  3.99779940  4.49728950  1.85001891  2.55231795  4.08213029
> #canine_pca$x
> # This gives the plot between PC1 and PC2
> plot(canine_pca$x,xlim=xlim,ylim=xlim)
```

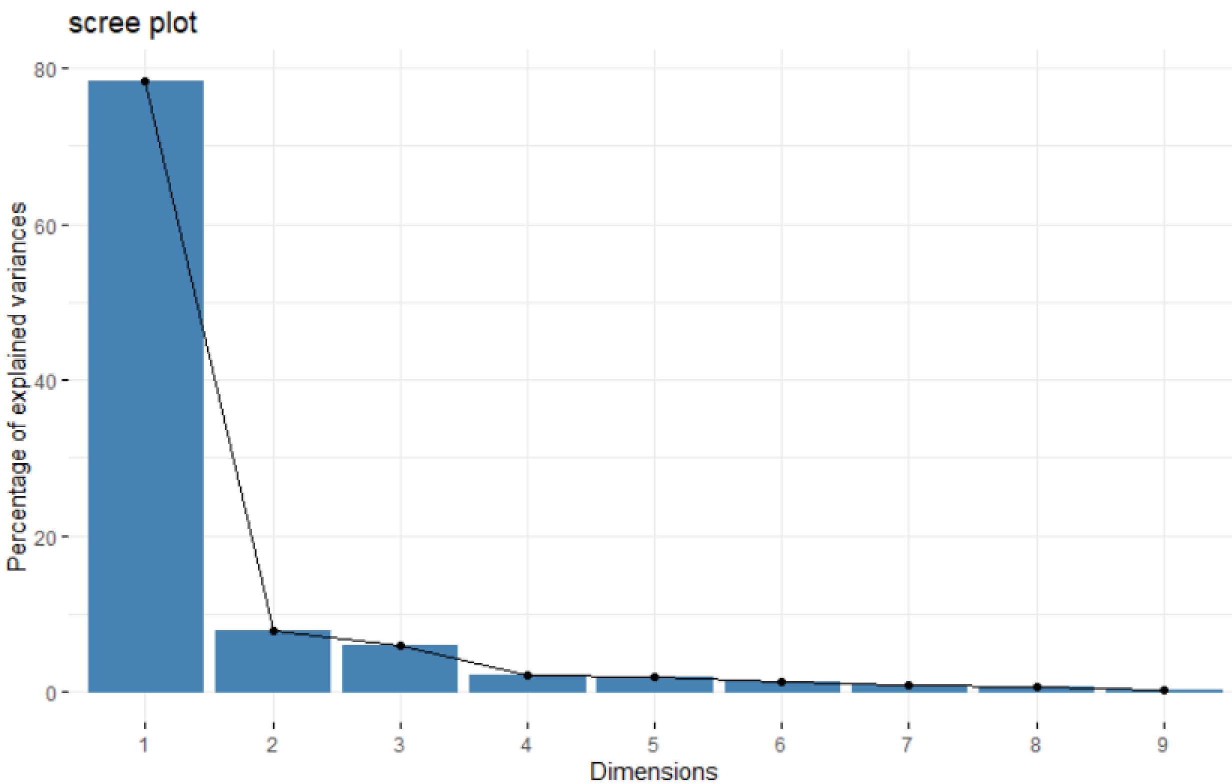


```
> canine_pca$rotation[,1]
       X1        X2        X3        X4        X5        X6        X7        X8        X9 
0.3636408 0.3424554 0.2665621 0.3265349 0.3539586 0.3459444 0.2859405 0.3470802 0.3544268 
> canine_pca$rotation
         PC1         PC2         PC3         PC4         PC5         PC6         PC7         PC8          PC9
X1 0.3636408 -0.11451510  0.08210471 -0.30326354  0.24950692 -0.07899550  0.05543869  0.16005914  0.811637429
X2 0.3424554  0.31490128 -0.19979188  0.33605928  0.01517931  0.49451257  0.13657790  0.60411640 -0.048224206
X3 0.2665621  0.32018675  0.87894338  0.04161625 -0.18169514 -0.04568559  0.08257828 -0.03476461 -0.094992855
X4 0.3265349  0.44638084 -0.16540131  0.26534253  0.54545187 -0.21526217 -0.30849700 -0.39244126 -0.057608736
X5 0.3539586 -0.14160855 -0.03861441 -0.26352534 -0.33012092  0.43239890 -0.67024916 -0.19081879 -0.046144083
X6 0.3459444  0.06792334 -0.26250857  0.05378069 -0.51974026 -0.68294862 -0.08734948  0.23986950 -0.046794522
X7 0.2859405 -0.68736531  0.13651981  0.64014932  0.05443187 -0.01170970  0.03463451 -0.11415807  0.002559605
X8 0.3470802 -0.28877388  0.03666665 -0.47256682  0.40753260 -0.10978603  0.12889717  0.23397418 -0.567438948
X9 0.3544268  0.07362113 -0.25111557 -0.13231892 -0.24254817  0.18765482  0.63372357 -0.54081471 -0.016253801
> plot(canine_num[,-1])
> barplot(p_cvar,main="canine_pca Scree plot",xlab="Principal Component",ylab="Percent Variation")
```





```r
#Visualize PCA using Scree plot
library(factoextra)
fviz_screeplot(canine_pca, type='bar',main='scree plot')
```



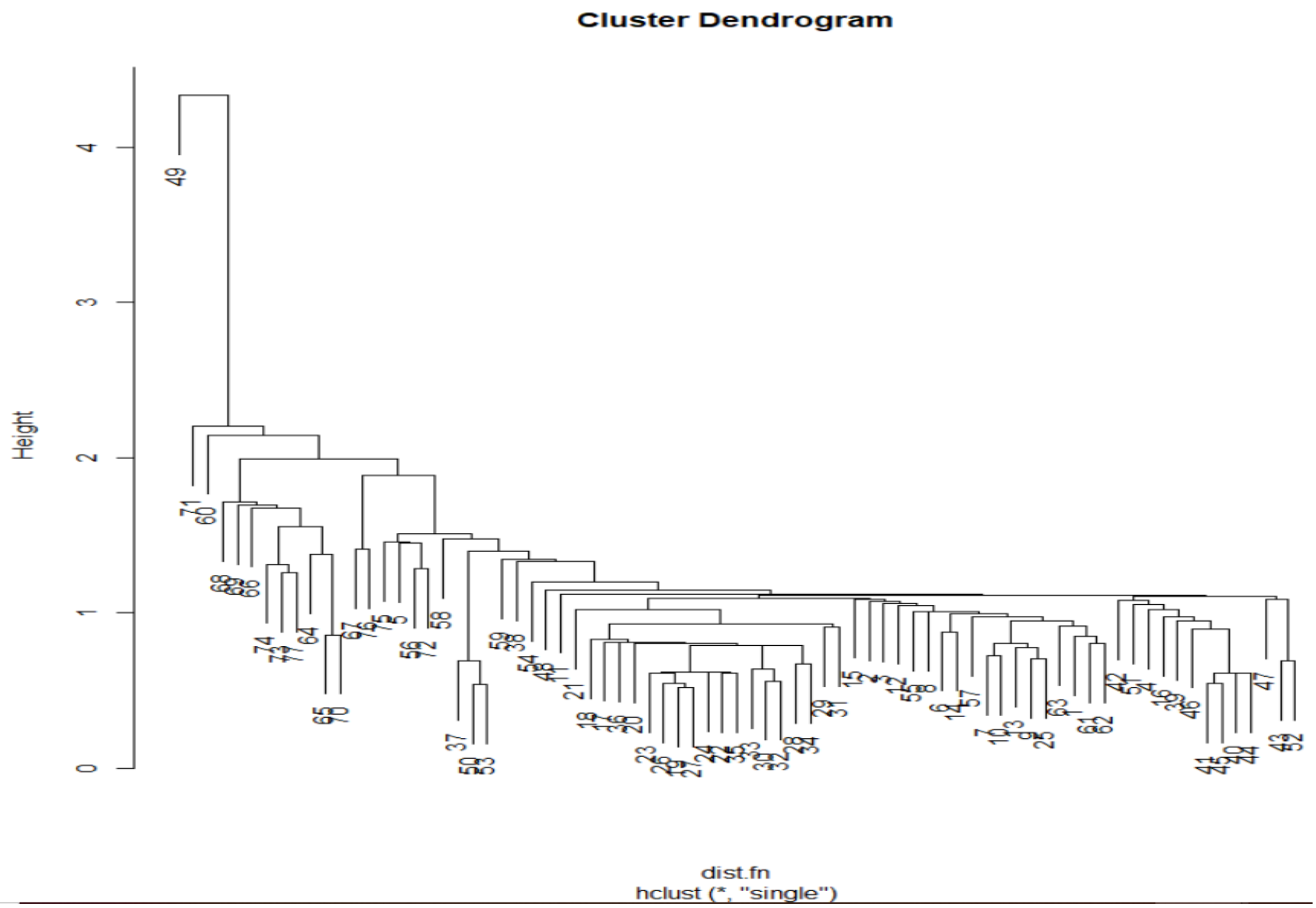
##Principal component analysis, or PCA, is a statistical procedure that allows you to summarize the information #content in large data tables by means of a smaller set of "summary indices" that can be more easily visualized and analyzed. The main idea of (PCA) is to reduce the dimensionality of a data set consisting of many #variables correlated with each other, either heavily or lightly, while retaining the variation present in the dataset up to the maximum extent. Interpretation of the principal components is based on finding which variables are most strongly correlated with #each component, i.e. Interpretation of the principal components is based on finding which variables are most strongly correlated with each component i.e., which of these numbers are large in magnitude, the farthest from zero in either direction.

After performing PCA on the canine Consumption Dataset, we obtain nine Principal components. Each of these components represent the percentage of variability present in the dataset. In other words, PC1 explains 78% of total variance, PC2 explains 7.5% and so on. We will consider the first three principal components as they sum up to 91.5% of total variance and the other six can be omitted as they contribute to only 8.5% of total variance.

cluster Analysis

Performing Hierarchical cluster analysis, Nearest-neighbor

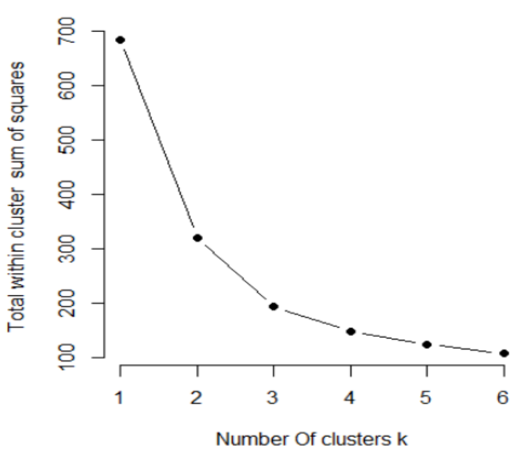
```
> # Standardizing the data with scale()
> matstd.fn <- scale(canine_data[,2:10])
> # Creating a (Euclidean) distance matrix of the standardized data
> dist.fn <- dist(matstd.fn, method="euclidean")
> # Invoking hclust command (cluster analysis by single linkage method)
> clusfn.nn <- hclust(dist.fn, method = "single")
> plot(clusfn.nn)
```



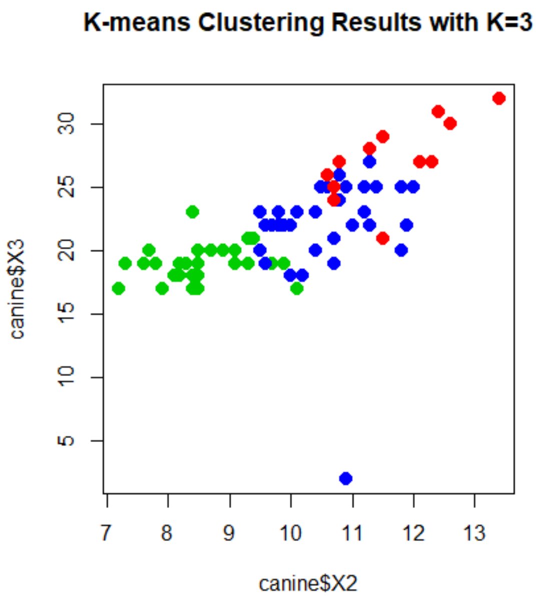

```
> (kmeans2<-kmeans(matstd.fn,2,nstart = 10))
K-means clustering with 2 clusters of sizes 33, 44

Cluster means:
          X1         X2         X3         X4         X5         X6         X7         X8         X9
1  0.8723872  0.8917073  0.7334245  0.8675852  0.8970576  0.9176751  0.5799499  0.7960761  0.9068156
2 -0.6542904 -0.6687805 -0.5500683 -0.6506889 -0.6727932 -0.6882563 -0.4349624 -0.5970571 -0.6801117

Clustering vector:
 [1] 2 2 2 2 1 2 2 2 2 2 2 2 2 2 2 1 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 1 1 1 1 1 1 1 1 1 1 1 2 2 1 1 2 2 2 1 2 1 1 1 2 2 2 1

Within cluster sum of squares by cluster:
[1] 183.3078 137.6728
 (between_SS / total_SS =  53.1 %)

Available components:

[1] "cluster"      "centers"      "totss"        "withinss"     "tot.withinss" "betweenss"    "size"         "iter"         "ifault"
```

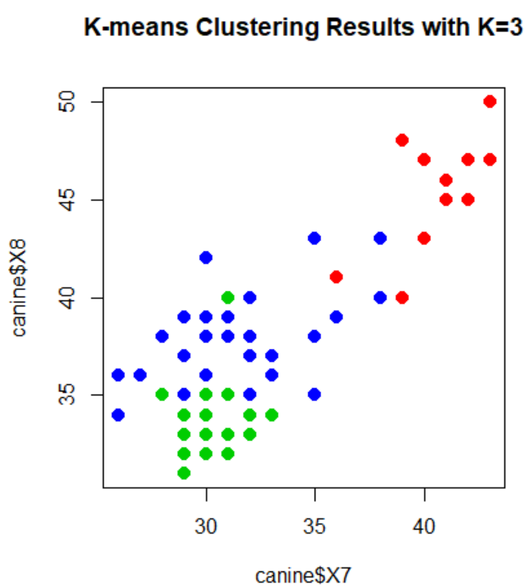
```
> # Computing the percentage of variation accounted for Two clusters
> perc.var.2 <- round(100*(1 - kmeans2$betweenss/kmeans2$totss),1)
> names(perc.var.2) <- "Perc. 2 clus"
> perc.var.2
Perc. 2 clus
        46.9
> # Computing the percentage of variation accounted for three clusters
> (kmeans3<-kmeans(matstd.fn,3,nstart = 10))
K-means clustering with 3 clusters of sizes 12, 30, 35

Cluster means:
          X1         X2          X3         X4         X5         X6         X7          X8         X9
1  1.8250998  1.2097445  1.33202825  1.0133575  1.8097616  1.3839132  1.9325611  1.80050789  1.4952269
2 -0.8593768 -0.9938371 -0.61765844 -1.0440497 -0.8407307 -0.9671111 -0.4680061 -0.78005628 -0.9630246
3  0.1108602  0.4370909  0.07272612  0.5474629  0.1001367  0.3544678 -0.2614443  0.05130268  0.3128004

Clustering vector:
 [1] 3 3 3 3 3 3 2 2 2 2 2 2 2 3 2 3 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 2 3 3 2 3 3 3 3 3 3 1

Within cluster sum of squares by cluster:
[1]  33.56015  48.65771 111.50575
 (between_SS / total_SS =  71.7 %)

Available components:

[1] "cluster"      "centers"      "totss"        "withinss"     "tot.withinss" "betweenss"    "size"         "iter"         "ifault"
> perc.var.3 <- round(100*(1 - kmeans3$betweenss/kmeans3$totss),1)
> names(perc.var.3) <- "Perc. 3 clus"
> perc.var.3
Perc. 3 clus
        28.3
> # Computing the percentage of variation accounted for four clusters
> (kmeans4<-kmeans(matstd.fn,4,nstart = 10))
K-means clustering with 4 clusters of sizes 12, 25, 20, 20

Cluster means:
          X1         X2         X3         X4         X5         X6         X7         X8         X9
1  1.8250998  1.2097445  1.3320283  1.0133575  1.8097616  1.3839132  1.9325611  1.8005079  1.4952269
2 -0.3184819 -0.1632472 -0.3977749 -0.1105691 -0.4712682 -0.1836534 -0.3162231 -0.2911869 -0.1719384
3 -1.0326917 -1.2801246 -0.7442101 -1.3154104 -0.9210487 -1.2015623 -0.5319147 -0.9427605 -1.2555957
4  0.3357343  0.7583369  0.4422118  0.8456072  0.4242770  0.6007811 -0.2323431  0.2264394  0.5733826

Clustering vector:
 [1] 2 2 2 2 4 2 2 2 2 2 2 3 2 2 2 4 3 3 3 3 3 3 3 3 2 3 3 3 3 3 3 3 3 3 3 3 2 4 4 4 4 4 4 4 4 4 4 4 2 2 4 4 2 2 2 4 2 4 4 2 2 2 2 1

Within cluster sum of squares by cluster:
[1] 33.56015 61.66440 16.21449 37.30154
 (between_SS / total_SS =  78.3 %)

Available components:

[1] "cluster"      "centers"      "totss"        "withinss"     "tot.withinss" "betweenss"    "size"         "iter"         "ifault"
> perc.var.4 <- round(100*(1 - kmeans4$betweenss/kmeans4$totss),1)
> names(perc.var.4) <- "Perc. 4 clus"
> perc.var.4
Perc. 4 clus
        21.7
```

```
> #Scree to determine the elbow (optimum number of clusters )
> k.max<-6
> wss<-sapply(1:k.max,function(k){kmeans(matstd.fn,k,nstart=50)$tot.withinss})
> plot(1:k.max,wss, type='b',pch =19,frame = FALSE, xlab ="Number Of clusters k",ylab ="Total within cluster  sum of squares")
```



#lets plot to understand further if k means really helps





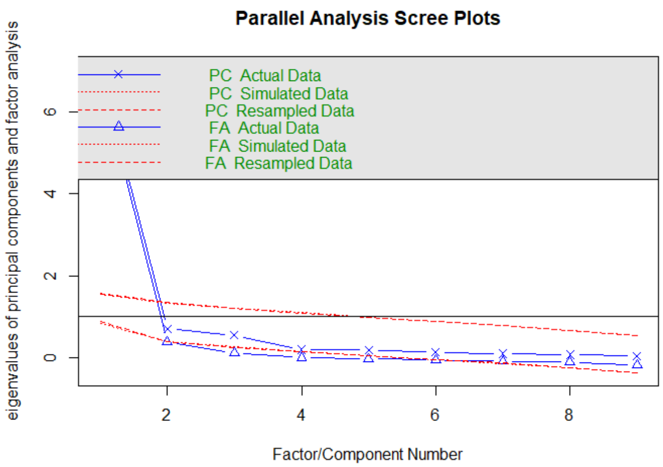
#Clustering is a method of grouping together a set of objects together in such a way that the objects in one cluster is like the objects in the same cluster than objects present in the different cluster. We form 3 clusters for the given dataset as This covers all the variance present in the dataset. we see that having with 3 clusters we are getting good clustering with efficiency of almost 78%.From the plot we see that with 3 clusters Indian wolf is being assigned to cluster 2 along with thaidogs, couns, moderndog. Hence we can tell that it is related to these species based on clustering.

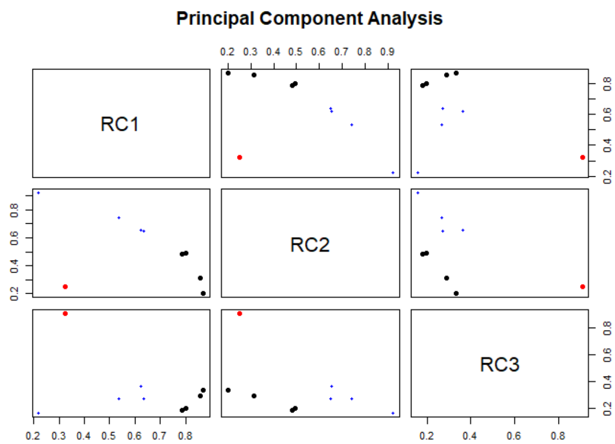
Factor Analysis

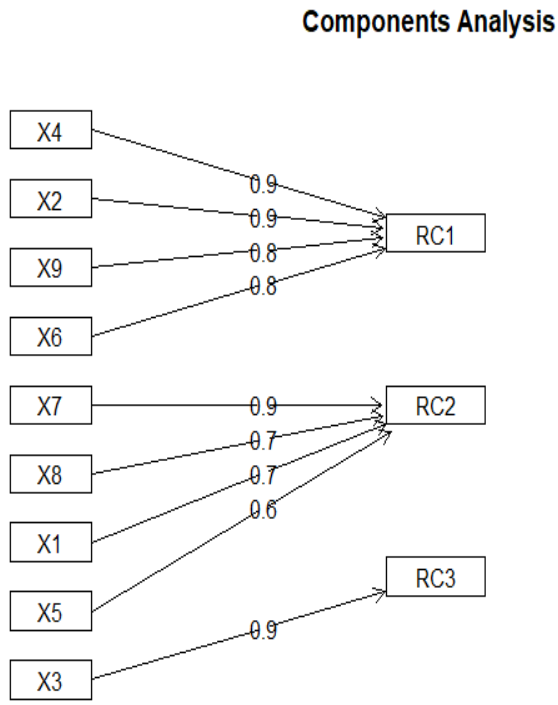
Considering 3 factors into consideration as we have observed the by the analysis made in the PCA section

```
> #Considering 3 factors into consideration as we have observed  the by the analysis made in the PCA section
> fit.pro <- principal(canine_numb, nfactors=3, rotate="varimax")
> fit.pro
Principal Components Analysis
Call: principal(r = canine_numb, nfactors = 3, rotate = "varimax")
Standardized loadings (pattern matrix) based upon correlation matrix
    RC1  RC2  RC3   h2     u2 com
X1 0.62 0.65 0.36 0.95 0.0546 2.6
X2 0.86 0.31 0.29 0.92 0.0815 1.5
X3 0.32 0.25 0.91 0.99 0.0076 1.4
X4 0.87 0.20 0.33 0.91 0.0929 1.4
X5 0.64 0.65 0.27 0.90 0.1015 2.3
X6 0.79 0.48 0.18 0.88 0.1154 1.8
X7 0.22 0.92 0.16 0.92 0.0806 1.2
X8 0.54 0.74 0.27 0.91 0.0910 2.1
X9 0.80 0.49 0.20 0.92 0.0761 1.8

                       RC1  RC2  RC3
SS loadings           3.98 2.92 1.40
Proportion Var        0.44 0.32 0.16
Cumulative Var        0.44 0.77 0.92
Proportion Explained  0.48 0.35 0.17
Cumulative Proportion 0.48 0.83 1.00

Mean item complexity =  1.8
Test of the hypothesis that 3 components are sufficient.

The root mean square of the residuals (RMSR) is  0.02
 with the empirical chi square  3  with prob <  1

Fit based upon off diagonal values = 1
> round(fit.pro$values, 4)
[1] 7.0522 0.7042 0.5425 0.1928 0.1799 0.1316 0.0919 0.0703 0.0345
> #Loadings
> fit.pro$loadings

Loadings:
   RC1   RC2   RC3  
X1 0.622 0.652 0.365
X2 0.858 0.314 0.289
X3 0.324 0.252 0.908
X4 0.869 0.202 0.334
X5 0.636 0.649 0.270
X6 0.787 0.482 0.181
X7 0.220 0.920 0.159
X8 0.536 0.741 0.269
X9 0.802 0.492 0.198

                 RC1   RC2   RC3
SS loadings    3.985 2.919 1.395
Proportion Var 0.443 0.324 0.155
Cumulative Var 0.443 0.767 0.922
```

```
> #communalities
> fit.pro$communality
       X1        X2        X3        X4        X5        X6        X7        X8        X9 
0.9454361 0.9185378 0.9924279 0.9070985 0.8984760 0.8846248 0.9194263 0.9089926 0.9239138 
> #Rotated factor scores
> head(fit.pro$scores)
             RC1        RC2        RC3
[1,]  0.06552327 -0.8719736  0.5854046
[2,] -0.01387213  0.1968842 -0.5556222
[3,]  0.05562332  0.5835645 -1.1670095
[4,]  0.06667753 -0.4220909  0.7389350
[5,]  0.70008295  0.2793482  0.7191850
[6,] -0.47722049  0.1593615  0.3734396
> round(fit.pro$values,4)
[1] 7.0522 0.7042 0.5425 0.1928 0.1799 0.1316 0.0919 0.0703 0.0345
> #Factor recommendation
> fa.parallel(canine_numb)

> #Correlations within Factors
> fa.plot(fit.pro)
> #Visualizing the relationship
> fa.diagram(fit.pro)
```







#Here we can observe that the RC1 is contributed by 0.9 of X4 AND X2 ,0.8 0f X9 AND X6 and 0.6 of x5.In the similar way other factors contributing to respective RC's are shown in the diagram. Hence showing the relationships between the species with respect to these factors

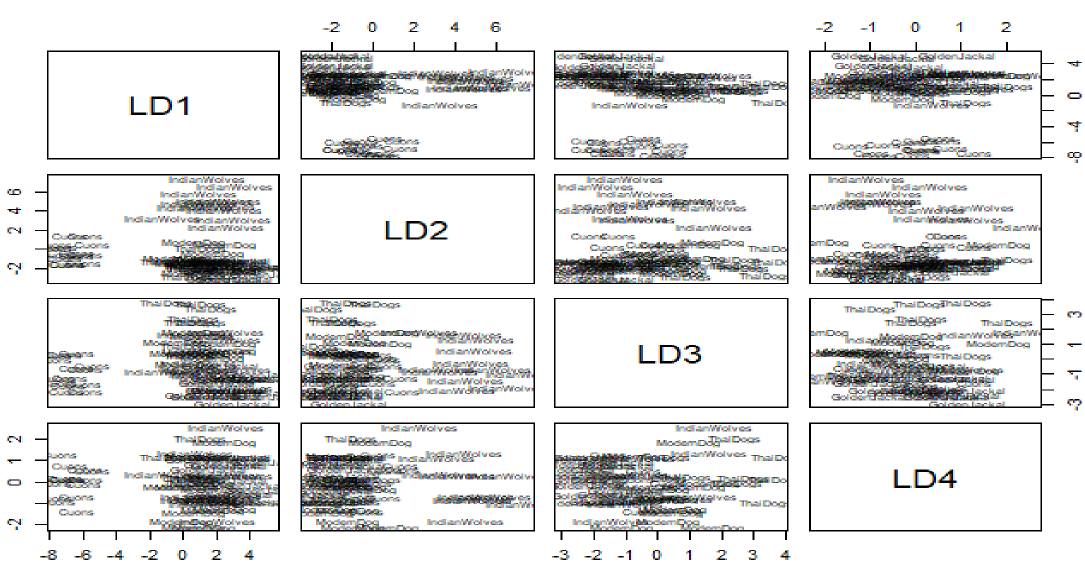
Here we can also observe that very less portions of the unique variance has been given away from each of the variables’ 0.1 unique variance of x4,x2,x7,x3 ,and 0.2 unique variance of x9 , x6 and 0.3 of x8 and 0.4 of x5 is given away.

Discriminant function Analysis

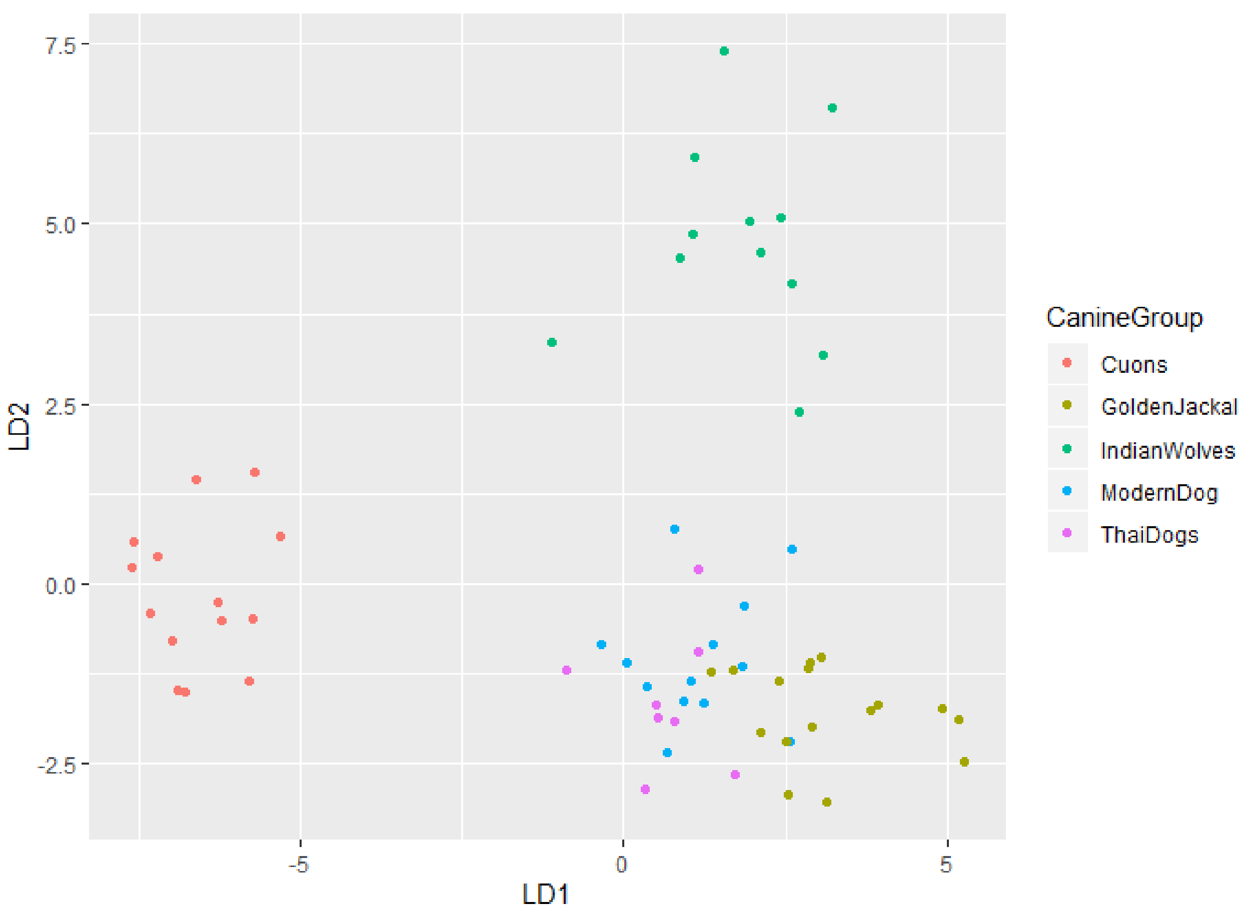
Initially we split the data into training and testing sets. Using the 80-20 rule,80% of the data for training and 20% of the data for testing.

```
> library(klaR)
> library(tidyverse)
> library(caret)
> set.seed(123)
> training_samples <- canine_data$CanineGroup %>% createDataPartition(p = 0.8, list = FALSE)
> training_data <- canine_data[training_samples, ]
> testing_data <- canine_data[-training_samples, ]
> #Estimate preprocessing parameters
> preproc.param <- train.data %>%
+   preProcess(method = c("center", "scale"))
> train_transformed <- preproc.param %>% predict(training_data)
> test_transformed <- preproc.param %>% predict(testing_data)
> train_transformed
    CanineGroup          X1           X2         X3         X4         X5         X6         X7         X8          X9 Gender
1     ModernDog -0.34973425  0.061979831  0.2928336  0.4622625 -0.5938638 -0.1609105 -0.1085148 -1.0196587 -0.45127977   Male
2     ModernDog  0.45147513 -0.286656717 -0.8085703  0.1589027 -0.5938638 -0.1609105 -0.1085148  0.5471340 -0.25787415   Male
3     ModernDog -0.46419274  0.131707140 -1.0839213 -0.1444570  0.1937871 -0.0615246  0.5987026  0.0994789  0.12893708   Male
5     ModernDog  1.13822602  1.386798714  0.8435355  1.0689820  0.1937871  0.4354048  0.5987026  1.2186165  0.51574830   Male
6     ModernDog -0.23527577 -0.356384027  0.2928336 -0.4478168 -0.2000383 -0.1609105  0.1272243 -0.1243486  0.22563988   Male
8     ModernDog -0.23527577 -0.216929408 -0.8085703 -0.7511765 -0.5938638 -0.4590681 -0.1085148 -0.1243486  0.12893708   Male
9     ModernDog -0.46419274 -0.286656717  0.0174826 -0.4478168 -0.9876892 -0.3596823 -0.3442540 -0.5720037 -0.74138819 Female
10    ModernDog -0.40696349 -0.774747885 -0.5332194 -0.4478168 -0.5938638 -0.3596823 -0.3442540 -0.5720037 -0.35457696 Female
12    ModernDog -0.97925591 -0.635293266 -0.8085703 -0.4478168 -0.5938638 -1.3535411 -0.5799931 -1.0196587 -0.93479380 Female
13    ModernDog -0.29250501 -0.495838646 -0.2578684 -0.1444570 -0.9876892 -0.8566117 -0.5799931 -0.3481762 -0.54798257 Female
14    ModernDog -0.06358805 -0.286656717  0.0174826 -0.1444570 -0.5938638 -0.4590681 -0.1085148  0.0994789 -0.25787415 Female
15    ModernDog  0.05087044 -1.123384433  0.2928336 -0.4478168 -0.5938638 -0.6578399 -0.3442540  0.5471340 -0.25787415 Female
16    ModernDog -0.12081729  0.340889069  0.8435355  0.4622625 -0.2000383  0.7335625 -0.1085148 -0.5720037  0.03223427 Female
17 GoldenJackal -0.52142198 -1.262839052 -1.0839213 -1.3578961 -0.9876892 -0.9559976 -0.1085148 -0.5720037 -0.83809099   Male
19 GoldenJackal -1.09371439 -1.332566362 -1.0839213 -1.6612558 -0.5938638 -0.8566117 -0.3442540 -1.2434863 -1.32160503   Male
20 GoldenJackal -0.75033894 -1.053657123 -0.5332194 -1.0545363 -0.9876892 -0.8566117 -0.1085148 -1.0196587 -1.32160503   Male
21 GoldenJackal -0.86479742 -1.262839052 -0.8085703 -1.0545363 -0.5938638 -0.0615246 -0.1085148 -1.0196587 -0.93479380   Male
23 GoldenJackal -0.92202667 -1.053657123 -1.3592723 -1.0545363 -0.5938638 -0.8566117 -0.5799931 -0.7958312 -1.41830784   Male
24 GoldenJackal -0.69310970 -0.914202504 -0.5332194 -1.3578961 -0.9876892 -0.9559976 -0.5799931 -0.7958312 -0.83809099   Male
25 GoldenJackal -0.86479742 -0.426111337 -0.2578684 -0.7511765 -0.5938638 -0.4590681 -0.3442540 -0.5720037 -0.74138819   Male
26 GoldenJackal -0.97925591 -1.262839052 -0.8085703 -1.3578961 -0.5938638 -1.1547694 -0.5799931 -0.7958312 -0.93479380   Male
27 GoldenJackal -1.09371439 -1.053657123 -1.0839213 -1.3578961 -0.5938638 -0.9559976 -0.3442540 -1.0196587 -1.12819941 Female
29 GoldenJackal -1.26540211 -1.960112149 -1.3592723 -1.6612558 -1.3815146 -1.9498565 -1.0514714 -0.5720037 -1.32160503 Female
30 GoldenJackal -1.20817287 -1.262839052 -1.0839213 -1.6612558 -1.3815146 -1.4529270 -0.8157323 -1.0196587 -1.22490222 Female
31 GoldenJackal -1.09371439 -1.890384839 -0.8085703 -1.9646156 -1.3815146 -1.8504706 -0.5799931 -1.0196587 -1.51501064 Female
33 GoldenJackal -1.26540211 -1.123384433 -1.0839213 -1.3578961 -0.9876892 -1.7510847 -0.8157323 -1.4673138 -1.70841626 Female
34 GoldenJackal -1.32263136 -1.541748291 -0.8085703 -1.0545363 -0.9876892 -1.7510847 -0.3442540 -1.2434863 -1.61171345 Female
35 GoldenJackal -1.03648515 -1.123384433 -1.3592723 -1.6612558 -0.9876892 -0.9559976 -0.5799931 -0.7958312 -1.32160503 Female
36 GoldenJackal -1.03648515 -1.681202910 -0.8085703 -1.3578961 -0.9876892 -1.4529270 -0.5799931 -0.5720037 -1.41830784 Female
37        Cuons -0.34973425 -0.216929408  0.0174826 -0.1444570 -0.2000383 -0.1609105 -1.2872106 -0.3481762  0.03223427   Male
38        Cuons  0.33701664  1.247344095  0.8435355 -0.1444570  0.9814380  0.9323343 -0.3442540  0.0994789  0.99926234   Male
39        Cuons  0.50870437  0.968434856  0.8435355  1.0689820  0.5876126  1.0317202 -0.5799931  0.0994789  1.19266795   Male
40        Cuons  0.68039209  0.550070998  1.1188865  1.0689820  0.1937871  0.1372472 -0.8157323  0.3233064  0.51574830   Male
41        Cuons  0.33701664  0.828980237  0.8435355  1.0689820  0.1937871  0.5347907 -0.8157323  0.3233064  0.61245111   Male
42        Cuons  0.39424588  0.689525617  0.0174826  0.7656222  0.5876126  0.1372472 -0.3442540  0.3233064  0.70915392   Male
44        Cuons  0.45147513  0.410616379  0.8435355  0.7656222  0.1937871  0.3360190 -1.0514714  0.0994789  0.41904550   Male
46        Cuons  0.10809968  0.619798308  0.8435355  0.7656222  0.1937871  0.5347907 -0.8157323 -0.5720037  0.12893708 Female
47        Cuons  0.05087044  0.898707546  0.0174826  0.4622625  0.1937871  0.7335625 -0.8157323 -0.1243486  0.90255953 Female
48        Cuons  0.85207982  0.550070998  0.5681846  1.3723418  0.5876126  0.9323343 -0.5799931  0.9947890  0.99926234 Female
50        Cuons -0.34973425 -0.147202098  0.2928336  0.1589027 -0.2000383  0.1372472 -1.5229497 -0.7958312 -0.45127977 Female
51        Cuons  0.45147513  0.898707546  1.3942375  1.3723418  0.9814380  0.7335625 -0.5799931  0.3233064  0.41904550 Female
52        Cuons -0.06358805 -0.007747479  0.0174826  0.4622625  0.5876126  0.7335625 -0.8157323 -0.1243486  0.51574830 Female
```

```
52        Cuons -0.06358805 -0.007747479  0.0174826  0.4622625  0.5876126  0.7335625 -0.8157323 -0.1243486  0.51574830  Female
53        Cuons -0.40696349 -0.077474788  0.0174826  0.1589027 -0.2000383  0.2366331 -1.5229497 -0.3481762 -0.35457696  Female
54     ThaiDogs -0.97925591  0.061979831 -1.3592723 -1.0545363 -0.5938638 -0.2602964 -0.3442540 -1.0196587 -0.25787415 Unknown
55     ThaiDogs -0.80756818 -0.007747479 -1.0839213  0.4622625 -0.2000383 -0.1609105  0.1272243 -0.3481762 -0.06446854 Unknown
56     ThaiDogs  0.39424588  1.317071404  0.0174826  1.0689820  0.1937871  0.5347907  0.8344417  0.3233064  0.90255953 Unknown
57     ThaiDogs -1.03648515 -0.077474788 -0.8085703 -0.4478168 -0.9876892 -0.6578399 -0.8157323 -0.7958312 -0.74138819 Unknown
59     ThaiDogs -0.23527577  0.480343689 -0.8085703  1.3723418 -0.2000383  0.4354048  0.1272243 -0.1243486  0.22563988 Unknown
61     ThaiDogs -0.46419274  0.480343689 -0.2578684  0.4622625 -0.5938638 -0.0615246 -0.1085148 -0.5720037 -0.06446854 Unknown
62     ThaiDogs -0.40696349 -0.147202098  0.0174826  0.4622625 -0.9876892 -0.0615246 -0.1085148 -0.5720037  0.03223427 Unknown
63     ThaiDogs -0.29250501 -0.356384027 -0.5332194  0.7656222 -0.5938638 -0.3596823 -0.1085148 -0.1243486 -0.06446854 Unknown
65 IndianWolves  1.99666464  1.595980643  1.3942375  1.3723418  1.7690889  2.0255791  2.2488766  2.1139266  1.77288479    Male
66 IndianWolves  1.19545526  1.038162166 -0.2578684  0.7656222  1.7690889  1.3298778  2.0131375  1.8900991  2.35310164    Male
67 IndianWolves  0.90930906  0.898707546  1.6695885  0.7656222  1.3752634  1.2304919  0.8344417  0.7709615  1.09596515    Male
68 IndianWolves  2.74064478  1.665707952  2.4956414  1.6757015  2.5567397  2.5225085  2.4846158  2.7854092  1.77288479    Male
69 IndianWolves  2.11112312  2.362981049  2.7709924  1.6757015  2.1629143  1.5286496  1.7773983  2.1139266  1.19266795    Male
70 IndianWolves  1.99666464  1.456526023  1.3942375  0.7656222  1.7690889  1.9261932  2.2488766  1.6662716  2.15969602    Male
71 IndianWolves  2.05389388  1.805162572  2.2202904  1.3723418  1.7690889 -0.2602964  1.7773983  1.2186165  1.77288479    Male
72 IndianWolves  0.10809968  1.247344095 -0.5332194  0.7656222  0.9814380  0.8329484  1.3059200  0.5471340  0.41904550  Female
73 IndianWolves  1.93943540  0.550070998  1.3942375  0.7656222  1.3752634  1.2304919  1.5416592  2.3377542  0.90255953  Female
74 IndianWolves  1.99666464  0.480343689  0.5681846  0.4622625  2.1629143  1.5286496  2.4846158  2.1139266  1.48277637  Female
75 IndianWolves  0.68039209  0.271161760 -0.5332194  0.4622625  0.9814380  0.9323343  1.3059200  1.2186165 -0.06446854  Female
76 IndianWolves  1.08099678  0.410616379  1.1188865 -0.1444570  1.3752634  0.9323343  1.5416592  0.5471340  0.90255953  Female
> #Fit the model
> model <- lda(CanineGroup ~ X1+X2+X3+X4+X5+X6+X7+X8+X9, data = train_transformed)
> model
Call:
lda(CanineGroup ~ X1 + X2 + X3 + X4 + X5 + X6 + X7 + X8 + X9,
    data = train_transformed)

Prior probabilities of groups:
      Cuons GoldenJackal IndianWolves    ModernDog     ThaiDogs
  0.2222222    0.2539683    0.1904762    0.2063492    0.1269841

Group means:
                    X1         X2         X3         X4         X5          X6          X7         X8           X9
Cuons        0.2143826  0.5152073  0.5485166  0.6572795  0.3344391  0.49929577 -0.84940930  0.0195405  0.474304244
GoldenJackal -1.0007169 -1.2628391 -0.9290364 -1.3578961 -0.9138469 -1.14234613 -0.49159096 -0.9077450 -1.224902222
IndianWolves  1.5674453  1.1485637  1.1418324  0.8920221  1.6706325  1.31331352  1.79704325  1.6103147  1.313546462
ModernDog    -0.1516330 -0.1954749 -0.1307833 -0.0977863 -0.4726867 -0.29852171 -0.10851484 -0.1415661 -0.213242087
ThaiDogs     -0.4785000  0.2188663 -0.6020571  0.3864225 -0.4954074 -0.07394783 -0.04958006 -0.4041330 -0.004029284

Coefficients of linear discriminants:
           LD1         LD2        LD3          LD4
X1 -1.97072716  0.31966418 -1.6475816 -1.791634507
X2  0.06807624 -0.19192502  0.6369900  0.004391232
X3  0.99095629  0.13693446 -0.1322325 -0.573740739
X4 -0.76054550 -0.28871452  1.6031046  0.501051689
X5 -2.43079790  2.21886698 -2.7044257  2.188532378
X6 -0.55809049 -0.07197145  0.2410220 -0.015939568
X7  5.26574802  0.60720401  1.5297908  0.264218219
X8  0.99632709  0.23806894  0.2776807 -0.510735191
X9 -1.46257979 -0.38760990  1.2142292 -0.165711380

Proportion of trace:
   LD1    LD2    LD3    LD4
0.6347 0.2757 0.0831 0.0064
> plot(model)
```



```
> #Predictions
>  predictions <- model %>% predict(test_transformed)
>  names(predictions)
[1] "class"     "posterior" "x"
>  #Linear discriminants
> head(predictions$x, 10)
          LD1        LD2        LD3         LD4
4    0.4356528 -0.5164550  0.5316930 -0.71066819
7    1.0183009 -1.3282084  0.7191164 -0.23634222
11   1.1679786 -0.7637476  0.9020575  1.46762903
18   2.8191202 -0.2591108 -1.8265538  2.43825960
22   2.7506776 -2.0852008 -1.4779341 -0.12556377
28   3.2301641 -1.9528649 -1.6421005  0.21878152
32   3.7201019 -3.0957270 -0.6032432 -0.07921485
43  -7.4892226  1.3040956 -1.9033387  1.91440481
45  -6.4840681 -0.4828682  0.6652580 -0.64769490
49 -12.6129402 -0.1597392 -0.3020132  3.19237196
> lda_model <- cbind(train.transformed, predict(model)$x)
>  ggplot(lda_model, aes(LD1, LD2,LD3,LD4)) +geom_point(aes(color = CanineGroup))
Warning message:
Duplicated aesthetics after name standardisation:
> mean(predictions$class==test_transformed$CanineGroup)
[1] 0.9285714
```



Model Accuracy for train and test is 92.857%. We can see that we could separate the groups of the canine with different colors as shown in the above diagram.

Logistic Regression

To investigate each canine group separately we first have to split them into groups i.e each canine group is taken and analyzed.

ModernDog has the data from 1:16 rows so we group them together. Similarly GoldenJackal has the data from 17:36 rows. Cuons has the data from 37:53 rows.IndianWolves has the data from 64:77 rows. While we ignore the ThaiDog for now as its gender is unknown.

<u>Analysis for ModernDog</u>

```
> #7
> library(car)
> library(MASS)
> canine_set1<-canine_data[1:16,]
> xtabs(~Gender + CanineGroup, data=canine_set1)
         CanineGroup
Gender    Cuons GoldenJackal IndianWolves ModernDog ThaiDogs
  Female      0            0            0         8        0
  Male        0            0            0         8        0
  Unknown     0            0            0         0        0
> #using the variables X1, X2, X3 for the logistic model
> logistic_simple_1 <- glm(Gender~X2+X3+X4 , data=canine_set1, family="binomial")
> summary(logistic_simple_1)

Call:
glm(formula = Gender ~ X2 + X3 + X4, family = "binomial", data = canine_set1)

Deviance Residuals:
     Min        1Q    Median        3Q       Max
-1.44704  -0.86076  -0.01154   0.71505   2.18570

Coefficients:
            Estimate Std. Error z value Pr(>|z|)
(Intercept) -22.2726    13.6466  -1.632    0.103
X2            2.6844     1.8691   1.436    0.151
X3           -0.7248     0.5481  -1.322    0.186
X4            0.5574     0.6326   0.881    0.378

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 22.181  on 15  degrees of freedom
Residual deviance: 14.893  on 12  degrees of freedom
AIC: 22.893

Number of Fisher Scoring iterations: 6

> #Confusion matrix
> confusion_matrix(logistic_simple_1)
              Predicted Female Predicted Male Total
Actual Female                7              1     8
Actual Male                  2              6     8
Total                        9              7    16
```

#There were total of 16 ModernDog 8 Female and 8 male.Here we can observe that the accuracy of our model for predicting the ModernDog's gender is 81.25%.out of the 8 females in ModernDog's 7 of them were predicted correctly,out of the 8 males in ModernDog's 6 of them were predicted correctly.

Analysis for GoldenJackal

```
> canine_set2<-canine_data[17:36,]
> canine_set2<-canine_data[17:36,]
> xtabs(~Gender + CanineGroup, data=canine_set2)
         CanineGroup
Gender    Cuons GoldenJackal IndianWolves ModernDog ThaiDogs
  Female      0           10            0         0        0
  Male        0           10            0         0        0
  Unknown     0            0            0         0        0
> #using the variables X4+X5+X6 for the logistic model
> logistic_simple_2 <- glm(Gender~X4+X5+X6, data=canine_set2, family="binomial")
> summary(logistic_simple_1)

Call:
glm(formula = Gender ~ X2 + X3 + X4, family = "binomial", data = canine_set1)

Deviance Residuals:
     Min        1Q    Median        3Q       Max
-1.44704  -0.86076  -0.01154   0.71505   2.18570

Coefficients:
            Estimate Std. Error z value Pr(>|z|)
(Intercept) -22.2726    13.6466  -1.632    0.103
X2            2.6844     1.8691   1.436    0.151
X3           -0.7248     0.5481  -1.322    0.186
X4            0.5574     0.6326   0.881    0.378

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 22.181  on 15  degrees of freedom
Residual deviance: 14.893  on 12  degrees of freedom
AIC: 22.893

Number of Fisher Scoring iterations: 6

> #Confusion matrix
> confusion_matrix(logistic_simple_2)
              Predicted Female Predicted Male Total
Actual Female                8              2    10
Actual Male                  1              9    10
Total                        9             11    20
```

#There were total of 20 GoldenJackal 10 Female and 10 male.Here we can observe that the accuracy of our model for predicting the GoldenJackal's gender is 85%. out of the 10 females in GoldenJackal's 8 of them were predicted correctly ,out of the 10 males in GoldenJackal's 9 of them were predicted correctly.

Analysis for Cuons

```
> canine_set3<-canine_data[37:53,]
> xtabs(~Gender + CanineGroup, data=canine_set3)
         CanineGroup
Gender    Cuons GoldenJackal IndianWolves ModernDog ThaiDogs
  Female      8            0            0         0        0
  Male        9            0            0         0        0
  Unknown     0            0            0         0        0
> #using the variables X4+X7+X8 for the logistic model
> logistic_simple_3 <- glm(Gender~X4+X7+X8, data=canine_set3, family="binomial")
> #using the variables X1, X2, X3 for the logistic model
> summary(logistic_simple_3)

Call:
glm(formula = Gender ~ X4 + X7 + X8, family = "binomial", data = canine_set3)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-1.5560  -1.2136   0.6235   1.1757   1.2839

Coefficients:
            Estimate Std. Error z value Pr(>|z|)
(Intercept) -10.0052    11.7967  -0.848    0.396
X4           -0.2573     0.4691  -0.548    0.583
X7            0.3382     0.4542   0.745    0.457
X8            0.1701     0.4158   0.409    0.682

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 23.508  on 16  degrees of freedom
Residual deviance: 22.155  on 13  degrees of freedom
AIC: 30.155

Number of Fisher Scoring iterations: 4

> #Confusion matrix
> confusion_matrix(logistic_simple_3)
              Predicted Female Predicted Male Total
Actual Female                3              5     8
Actual Male                  2              7     9
Total                        5             12    17
```

#There were total of 17 Cuons 8 Female and 9 male.Here we can observe that the accuracy of our model for predicting the Cuons's gender is 58.8% which is less.Out of the 8 females in Cuons's 3 of them were predicted correctly.Out of the 9 males in Cuons's 7 of them were predicted correctly.

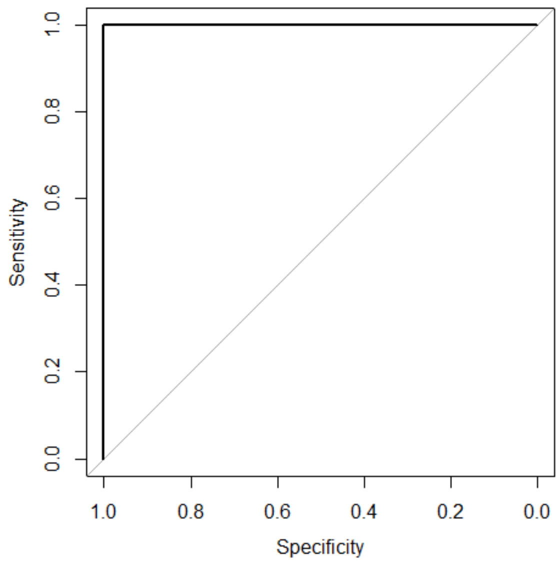
Analysis for Cuons

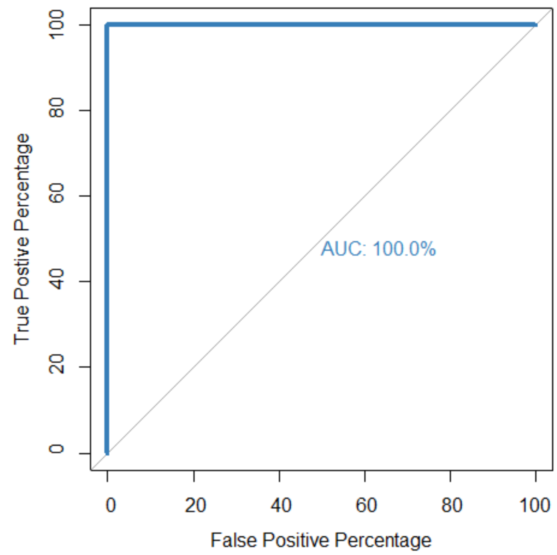
```
> canine_set4<-canine_data[64:77,]
> xtabs(~Gender + CanineGroup, data=canine_set4)
         CanineGroup
Gender    Cuons GoldenJackal IndianWolves ModernDog ThaiDogs
  Female      0            0            6         0        0
  Male        0            0            8         0        0
  Unknown     0            0            0         0        0
> #using the variables X1+X8+X9+X5 for the logistic model
> logistic_simple_4 <- glm(Gender~X1+X8+X9+X5, data=canine_set4, family="binomial")
> #using the variables X1, X2, X3 for the logistic model
> summary(logistic_simple_4)

Call:
glm(formula = Gender ~ X1 + X8 + X9 + X5, family = "binomial",
    data = canine_set4)

Deviance Residuals:
     Min        1Q    Median        3Q       Max
-1.71137  -0.13530   0.09277   0.51719   1.38815

Coefficients:
            Estimate Std. Error z value Pr(>|z|)
(Intercept) -44.1704    32.3444  -1.366    0.172
X1            0.1875     0.2562   0.732    0.464
X8           -0.9037     0.9848  -0.918    0.359
X9            4.5913     3.8812   1.183    0.237
X5            0.8542     1.5788   0.541    0.588

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 19.1214  on 13  degrees of freedom
Residual deviance:  9.5583  on  9  degrees of freedom
AIC: 19.558

Number of Fisher Scoring iterations: 7

> #Confusion matrix
> confusion_matrix(logistic_simple_4)
              Predicted Female Predicted Male Total
Actual Female                4              2     6
Actual Male                  1              7     8
Total                        5              9    14
```

#There were total of 14 IndianWolves 6 Female and 8 male.Here we can observe that the accuracy of our model for predicting the IndianWolves's gender is 78.5% .out of the 6 females in IndianWolves's 4 of them were predicted correctly,out of the 8 males in IndianWolves's 7 of them were predicted correctly.

ROC

```
> #8 ROC
> library(pROC)
> library(tidyverse)
> library(regclass)
> canine_data$Gender[53:63] <- c("Female", "Male", "Female", "Male", "Male", "Male", "Female","Female","Male","Male","Female")
> canine_data$Gender <- as.factor(canine_data$Gender)
> str(canine_data)
'data.frame':   77 obs. of  11 variables:
 $ CanineGroup: Factor w/ 5 levels "Cuons","GoldenJackal",..: 4 4 4 4 4 4 4 4 4 4 ...
 $ X1         : num  123 137 121 130 149 125 126 125 121 122 ...
 $ X2         : num  10.1 9.6 10.2 10.7 12 9.5 9.1 9.7 9.6 8.9 ...
 $ X3         : num  23 19 18 24 25 23 20 19 22 20 ...
 $ X4         : num  23 22 21 22 25 20 22 19 20 20 ...
 $ X5         : num  19 19 21 20 21 20 19 19 18 19 ...
 $ X6         : num  7.8 7.8 7.9 7.9 8.4 7.8 7.5 7.5 7.6 7.6 ...
 $ X7         : num  32 32 35 32 35 33 32 32 31 31 ...
 $ X8         : num  33 40 38 37 43 37 35 37 35 35 ...
 $ X9         : num  5.6 5.8 6.2 5.9 6.6 6.3 5.5 6.2 5.3 5.7 ...
 $ Gender     : Factor w/ 3 levels "Female","Male",..: 2 2 2 2 2 2 2 2 1 1 ...
> data_ThaiDog <- canine_data[c(53:63),]
> logistic_ThaiDog <- glm(Gender ~ .,data=data_ThaiDog, family="binomial")
> summary(logistic_a)

Call:
glm(formula = Gender ~ ., family = "binomial", data = data_a[c(-1)])

Deviance Residuals:
 [1]  0  0  0  0  0  0  0  0  0  0

Coefficients:
              Estimate Std. Error z value Pr(>|z|)
(Intercept) -1.942e+02  6.328e+06       0        1
X1          -4.115e+00  9.167e+04       0        1
X2          -7.501e+00  7.185e+05       0        1
X3           2.629e+01  3.948e+05       0        1
X4          -1.099e+01  1.131e+05       0        1
X5           4.108e+01  1.054e+06       0        1
X6           2.782e+01  7.972e+05       0        1
X7          -3.841e+01  8.190e+05       0        1
X8          -8.144e-02  1.456e+05       0        1
X9           1.215e+02  2.369e+06       0        1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 1.3460e+01  on 9  degrees of freedom
Residual deviance: 4.2867e-10  on 0  degrees of freedom
AIC: 20

Number of Fisher Scoring iterations: 23

> roc(data_ThaiDog$Gender,logistic_ThaiDog$fitted.values,plot=TRUE)
```






```
> ## If we want to find out the optimal threshold we can store the
> ## data used to make the ROC graph in a variable...
> roc.info <- roc(data_ThaiDog$Gender, logistic_ThaiDog$fitted.values, legacy.axes=TRUE)
```

```
str(roc.info)
roc.df <- data.frame(tpp=roc.info$sensitivities*100, ## tpp = true positive percentage
                     fpp=(1 - roc.info$specificities)*100, ## fpp = false positive precentage
                     thresholds=roc.info$thresholds)
roc.df
```

```
> roc.df <- data.frame(tpp=roc.info$sensitivities*100, ## tpp = true positive percentage
+                      fpp=(1 - roc.info$specificities)*100, ## fpp = false positive precentage
+                      thresholds=roc.info$thresholds)
> roc.df
  tpp fpp   thresholds
1 100 100         -Inf
2 100  80 2.143345e-11
3 100  40 2.143345e-11
4 100  20 2.143345e-11
5 100   0 5.000000e-01
6   0   0          Inf
> ## now let's look at the thresholds between TPP 60% and 80%
> roc.df[roc.df$tpp > 60 & roc.df$tpp < 80,]
[1] tpp        fpp        thresholds
<0 rows> (or 0-length row.names)
> roc(data_ThaiDog$Gender,logistic_ThaiDog$fitted.values,plot=TRUE, legacy.axes=TRUE, xlab="False Positive Percentage", ylab="True Postive Percentage", col="#377eb8", lwd=4, percent=TRUE)
```

The Area under the curve is 1 and AUC IS 100% which means it is overfitting and this is due to small sample space.

Gender Prediction for the Prehistoric Thai Dog

```
> #9
> canine_9 = canine_data[-c(54:63),-1]
> #view(data_9)
> canine_9$Gender <- as.factor(canine_9$Gender)
> str(canine_9)
'data.frame':	67 obs. of  10 variables:
 $ X1    : num  123 137 121 130 149 125 126 125 121 122 ...
 $ X2    : num  10.1 9.6 10.2 10.7 12 9.5 9.1 9.7 9.6 8.9 ...
 $ X3    : num  23 19 18 24 25 23 20 19 22 20 ...
 $ X4    : num  23 22 21 22 25 20 22 19 20 20 ...
 $ X5    : num  19 19 21 20 21 20 19 19 18 19 ...
 $ X6    : num  7.8 7.8 7.9 7.9 8.4 7.8 7.5 7.5 7.6 7.6 ...
 $ X7    : num  32 32 35 32 35 33 32 32 31 31 ...
 $ X8    : num  33 40 38 37 43 37 35 37 35 35 ...
 $ X9    : num  5.6 5.8 6.2 5.9 6.6 6.3 5.5 6.2 5.3 5.7 ...
 $ Gender: Factor w/ 3 levels "Female","Male",..: 2 2 2 2 2 2 2 2 1 1
> logistic <- glm(Gender ~ ., data=canine_9, family="binomial")
> summary(logistic)

Call:
glm(formula = Gender ~ ., family = "binomial", data = canine_9)

Deviance Residuals: 
    Min       1Q   Median       3Q      Max  
-1.9718  -1.1021   0.3837   1.0086   1.7916  

Coefficients:
            Estimate Std. Error z value Pr(>|z|)
(Intercept) -0.94892    2.92170  -0.325    0.745
X1           0.05571    0.07781   0.716    0.474
X2           0.76742    0.59927   1.281    0.200
X3           0.09570    0.09692   0.987    0.323
X4          -0.26780    0.25630  -1.045    0.296
X5          -0.16317    0.36040  -0.453    0.651
X6          -0.84409    0.97851  -0.863    0.388
X7           0.07344    0.13052   0.563    0.574
X8          -0.24511    0.23483  -1.044    0.297
X9           1.12319    0.85645   1.311    0.190

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 92.747  on 66  degrees of freedom
Residual deviance: 81.939  on 57  degrees of freedom
AIC: 101.94

Number of Fisher Scoring iterations: 4

> #Checking accuracy
> pred_9 <- predict(logistic,newdata=canine_9,type="response")
> pred_9
        1         2         3         4         5         6         7         8         9        10        11        12        13        14        15        16        17        18 
0.6373411 0.3438914 0.4611295 0.7082149 0.7660123 0.6757455 0.4001009 0.7317289 0.5517566 0.4553697 0.7044623 0.4771642 0.5273017 0.5376575 0.3340535 0.6067306 0.4914652 0.2585982 
       19        20        21        22        23        24        25        26        27        28        29        30        31        32        33        34        35        36 
0.3664417 0.4356189 0.2524704 0.5230936 0.2008980 0.6158678 0.4397013 0.4030895 0.3906903 0.2808497 0.2311434 0.4927282 0.4574246 0.4052972 0.4550128 0.2984661 0.3377446 0.2143183 
       37        38        39        40        41        42        43        44        45        46        47        48        49        50        51        52        53        64 
0.4981630 0.8878184 0.7479240 0.6699633 0.5889636 0.6629109 0.5727399 0.5973199 0.5328368 0.6441110 0.7435923 0.3722040 0.2063523 0.3773638 0.4563857 0.3414591 0.2615866 0.6053135 
       65        66        67        68        69        70        71        72        73        74        75        76        77 
0.7624328 0.7631413 0.7414619 0.6619095 0.8545097 0.9290440 0.9891649 0.4995653 0.4642971 0.6106067 0.1634855 0.8568749 0.4669179 
> pred_new <- as.factor(ifelse(test=as.numeric(pred_9>0.5) == 0, yes="Female", no="Male"))
> pred_new
 [1] Male   Female Female Male   Male   Male   Female Male   Male   Female Male   Female Male   Male   Female Male   Female Female Female Female Female Male   Female Male   Female Female
[27] Female Female Female Female Female Female Female Female Female Female Female Male   Male   Male   Male   Male   Male   Male   Male   Male   Male   Female Female Female Female Female
[53] Female Male   Male   Male   Male   Male   Male   Male   Male   Female Female Male   Female Male   Female
Levels: Female Male
```

```
> confusionMatrix(pred_new,canine_9$Gender)
Confusion Matrix and Statistics

          Reference
Prediction Female Male Unknown
   Female      23   12       0
   Male         9   23       0
   Unknown      0    0       0

Overall Statistics

               Accuracy : 0.6866
                 95% CI : (0.5616, 0.7944)
    No Information Rate : 0.5224
    P-Value [Acc > NIR] : 0.004694

                  Kappa : 0.3744

 Mcnemar's Test P-Value : NA

Statistics by Class:

                     Class: Female Class: Male Class: Unknown
Sensitivity                 0.7188      0.6571             NA
Specificity                 0.6571      0.7188              1
Pos Pred Value              0.6571      0.7188             NA
Neg Pred Value              0.7188      0.6571             NA
Prevalence                  0.4776      0.5224              0
Detection Rate              0.3433      0.3433              0
Detection Prevalence        0.5224      0.4776              0
Balanced Accuracy           0.6879      0.6879             NA
> # Hit Ratio/Accuracy is 68.66%
>
> #Predicting the gender of thai dogs
> thai_dog <- canine_data[c(54:63),-1]
> thai_dog$Gender[1] <- "Female"
> thai_dog$Gender[2] <- "Male"
> thai_dog$Gender = as.factor(thai_dog$Gender)
> #Using a threshold of 0.5 for determining the gender
> pred_t <- predict(logistic,newdata=thai_dog,type="response")
> pred_t
       54        55        56        57        58        59        60        61        62        63
0.7213970 0.3072734 0.8200314 0.4863042 0.5792303 0.3013514 0.4052520 0.6447660 0.5823274 0.3190485
> pred_tF <- as.factor(ifelse(test=as.numeric(pred_t>0.5) == 0, yes="Female", no="Male"))
> pred_tF
 [1] Male   Female Male   Female Male   Female Female Male   Male   Female
Levels: Female Male
```

a)Logistic Regression is a classification technique in which require set of class as 0 or 1 (Binary) for the values of the dependent variable. In this dataset we see that the gender takes the binary form (Female or Male) hence we could use logistic regression.

b)Hit ratio is 68.66%

Hence by the above method we have predicted the gender for ThaiDog.

Predict length of the Mandible length for Thai Dog

```
> training <- canine_data[1:54,c(-1,-11)]
> testing <- canine_data[55:63,c(-1,-11)]
> names(training)
[1] "X1" "X2" "X3" "X4" "X5" "X6" "X7" "X8" "X9"
> #Linear regression
> simple_fit <- lm(X1 ~ ., data=training)
> summary(simple_fit)

Call:
lm(formula = X1 ~ ., data = training)

Residuals:
    Min      1Q  Median      3Q     Max 
-7.9043 -1.7756  0.1646  2.2588  4.9245 

Coefficients:
            Estimate Std. Error t value Pr(>|t|)    
(Intercept)  5.45014   11.24007   0.485  0.63011    
X2           1.10337    0.93624   1.179  0.24478    
X3          -0.01864    0.13082  -0.142  0.88734    
X4           1.32883    0.35480   3.745  0.00051 ***
X5          -0.84924    0.60799  -1.397  0.16933    
X6           2.82657    1.54383   1.831  0.07375 .  
X7           0.09674    0.24424   0.396  0.69391    
X8           2.03205    0.27876   7.290  3.8e-09 ***
X9          -0.07583    1.58201  -0.048  0.96198    
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 2.982 on 45 degrees of freedom
Multiple R-squared:  0.9423,	Adjusted R-squared:  0.932 
F-statistic: 91.83 on 8 and 45 DF,  p-value: < 2.2e-16

> step <- stepAIC(simple_fit, direction="both")
Start:  AIC=126.17
X1 ~ X2 + X3 + X4 + X5 + X6 + X7 + X8 + X9

       Df Sum of Sq    RSS    AIC
- X9    1      0.02 400.31 124.17
- X3    1      0.18 400.47 124.20
- X7    1      1.40 401.68 124.36
- X2    1     12.35 412.64 125.81
<none>              400.29 126.17
- X5    1     17.35 417.64 126.46
- X6    1     29.82 430.10 128.05
- X4    1    124.78 525.06 138.82
- X8    1    472.66 872.95 166.28
```

```
Step:  AIC=124.18
X1 ~ X2 + X3 + X4 + X5 + X6 + X7 + X8

       Df Sum of Sq     RSS    AIC
- X3    1      0.18  400.48 122.20
- X7    1      1.40  401.71 122.36
- X2    1     13.09  413.40 123.91
<none>               400.31 124.17
- X5    1     20.11  420.41 124.82
+ X9    1      0.02  400.29 126.17
- X6    1     34.22  434.53 126.61
- X4    1    126.18  526.49 136.97
- X8    1    616.33 1016.63 172.50

Step:  AIC=122.2
X1 ~ X2 + X4 + X5 + X6 + X7 + X8

       Df Sum of Sq     RSS    AIC
- X7    1      1.43  401.91 120.39
- X2    1     12.99  413.47 121.92
<none>               400.48 122.20
- X5    1     19.95  420.43 122.82
+ X3    1      0.18  400.31 124.17
+ X9    1      0.02  400.47 124.20
- X6    1     34.24  434.72 124.63
- X4    1    138.35  538.83 136.22
- X8    1    622.19 1022.67 170.82

Step:  AIC=120.39
X1 ~ X2 + X4 + X5 + X6 + X8

       Df Sum of Sq     RSS    AIC
- X2    1     14.59  416.50 120.32
<none>               401.91 120.39
- X5    1     20.46  422.37 121.07
+ X7    1      1.43  400.48 122.20
+ X3    1      0.20  401.71 122.36
+ X9    1      0.02  401.89 122.39
- X6    1     34.55  436.46 122.84
- X4    1    142.09  544.00 134.74
- X8    1    692.34 1094.26 172.48


Step:  AIC=120.32
X1 ~ X4 + X5 + X6 + X8

       Df Sum of Sq     RSS    AIC
<none>               416.50 120.32
+ X2    1     14.59  401.91 120.39
- X5    1     17.54  434.04 120.54
+ X7    1      3.03  413.47 121.92
+ X9    1      0.83  415.66 122.21
+ X3    1      0.09  416.41 122.31
- X6    1     72.14  488.64 126.94
- X4    1    222.62  639.12 141.44
- X8    1    717.40 1133.90 172.40
> step$anova # display results
Stepwise Model Path
Analysis of Deviance Table

Initial Model:
X1 ~ X2 + X3 + X4 + X5 + X6 + X7 + X8 + X9

Final Model:
X1 ~ X4 + X5 + X6 + X8


  Step Df    Deviance Resid. Df Resid. Dev      AIC
1                          45   400.2856 126.1725
2 - X9  1  0.02043468      46   400.3061 124.1753
3 - X3  1  0.17619197      47   400.4823 122.1990
4 - X7  1  1.42851802      48   401.9108 120.3913
5 - X2  1 14.58821084      49   416.4990 120.3166
> fit1 <- lm(X1 ~ X2+X3+X9+X7, data=training)
> summary(fit1)

Call:
lm(formula = X1 ~ X2 + X3 + X9 + X7, data = training)

Residuals:
     Min       1Q   Median       3Q      Max
-12.3568  -2.7002  -0.2079   2.9427  13.5872

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  39.1685    13.7099   2.857  0.00626 **
X2            3.4419     1.3463   2.557  0.01372 *
X3            0.2695     0.2179   1.237  0.22210
X9            6.6323     1.8962   3.498  0.00101 **
X7            0.2379     0.4103   0.580  0.56465
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

```
Residual standard error: 5.3 on 49 degrees of freedom
Multiple R-squared:  0.8015,	Adjusted R-squared:  0.7853
F-statistic: 49.46 on 4 and 49 DF,  p-value: < 2.2e-16

> #Build the model
> Pred <- predict(fit1, testing)
> Pred
      55       56       57       58       59       60       61       62       63 
126.0845 141.0483 120.4155 136.0177 130.7530 129.4324 129.0645 126.8995 124.6646 
> actuals_preds <- data.frame(cbind(actuals=testing$X1, predicteds=Pred))  # make actuals_predicteds dataframe.
> correlation_accuracy <- cor(actuals_preds)
> min_max_accuracy <- mean(apply(actuals_preds, 1, min) / apply(actuals_preds, 1, max))
> min_max_accuracy
[1] 0.9537762
```

We have obtained the length of ThaiDog as we see the values above. We could train the data as done above and then build a model for the prediction of the length and compare the accuracy.

a)The accuracy for the model is 95.37%

Thank you 😊