Multivariate Analysis Final

Richard Noel Britto

12/5/2020

Multivariate Analysis Final

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

Please answer the questions below

1. Using suitable graphical method, compare the distribution of the nine variables for the prehistoric and modern Thai dog.

a. Create a Draftsman plot for the 9 variables showing each species as a different color

2. Create a distance matrix between the 5 canine groups

3. Use principal components analysis to investigate the relationships between the species on the basis of these variables

4. Carry out cluster analysis to study relation between different specifies.

a. Who is Indian Wolf related to?

5. Identify the important factors underlying the Skull measurement

a. Is there a relationships between the species with respect to these factors?

6. Carry out a discriminant function analysis to see how well it is possible to separate the groups using the measurements.

7. investigate each canine group separately to see whether logistic regression shows a significant difference between males and females for the measurements. Note that in view of the small sample sizes available for each group, it is unreasonable to expect to fit a logistic function involving all nine variables with good estimates of parameters. Therefore, consideration should be given to fitting functions using only a subset of the variables.

8. Show ROC containing both your discriminant and logistic function for gender classification for the Prehistoric Thai Dog

9. Predict the Gender for the Prehistoric Thai Dog

a. Explain the reason for choosing the MVA technique for prediction

b. What is the Hit Ratio (Accuracy) of your classification technique?

10. Create a model to predict length of the Mandible length for Prehistoric Thai Dog.

a. What is the accuracy of your model

Loading required libraries:

```r
library(cluster)
library(data.table)
library(magrittr)
library(stringr)
library(ggplot2)
library(knitr)
library(corrplot)
```

```
## corrplot 0.84 loaded
```

```r
library(tidyverse)
```

```
## -- Attaching packages ------------------------------------------ tidyverse 1.3.0 --

## v tibble  3.0.3     v purrr   0.3.4
## v tidyr   1.1.2     v dplyr   1.0.2
## v readr   1.3.1     v forcats 0.5.0

## -- Conflicts --------------------------------------------- tidyverse_conflicts() --
## x dplyr::between()   masks data.table::between()
## x tidyr::extract()   masks magrittr::extract()
## x dplyr::filter()    masks stats::filter()
## x dplyr::first()     masks data.table::first()
## x dplyr::lag()       masks stats::lag()
## x dplyr::last()      masks data.table::last()
## x purrr::set_names() masks magrittr::set_names()
## x purrr::transpose() masks data.table::transpose()
```

```r
library(factoextra)
```

```
## Welcome! Want to learn more? See two factoextra-related books at https://goo.gl/ve3WBa
```

```r
library(psych)
```

```
##
## Attaching package: 'psych'

## The following objects are masked from 'package:ggplot2':
##
##     %+%, alpha
```

```r
library(FactoMineR)
library(nFactors)
```

```
## Loading required package: lattice

##
## Attaching package: 'nFactors'

## The following object is masked from 'package:lattice':
##
##     parallel
```

```r
library(GGally)
```

```
## Registered S3 method overwritten by 'GGally':
##   method from
##   +.gg   ggplot2
```

```r
library(MASS)
```

```
##
## Attaching package: 'MASS'

## The following object is masked from 'package:dplyr':
##
##     select
```

```r
library(gvlma)
library(leaps)
library(relaimpo)
```

```
## Loading required package: boot

##
## Attaching package: 'boot'

## The following object is masked from 'package:lattice':
##
##     melanoma

## The following object is masked from 'package:psych':
##
##     logit

## Loading required package: survey

## Loading required package: grid

## Loading required package: Matrix

##
## Attaching package: 'Matrix'

## The following objects are masked from 'package:tidyr':
##
##     expand, pack, unpack

## Loading required package: survival

##
## Attaching package: 'survival'

## The following object is masked from 'package:boot':
##
##     aml
```

```
##
## Attaching package: 'survey'

## The following object is masked from 'package:graphics':
##
##     dotchart

## Loading required package: mitools

## This is the global version of package relaimpo.

## If you are a non-US user, a version with the interesting additional metric pmvd is available

## from Ulrike Groempings web site at prof.beuth-hochschule.de/groemping.
```

```r
library(cowplot)
library(regclass)
```

```
## Loading required package: bestglm

## Loading required package: VGAM

## Loading required package: stats4

## Loading required package: splines

##
## Attaching package: 'VGAM'

## The following object is masked from 'package:survey':
##
##     calibrate

## The following objects are masked from 'package:boot':
##
##     logit, simplex

## The following objects are masked from 'package:psych':
##
##     fisherz, logistic, logit

## The following object is masked from 'package:tidyr':
##
##     fill

## Loading required package: rpart

## Loading required package: randomForest
## randomForest 4.6-14

## Type rfNews() to see new features/changes/bug fixes.

##
## Attaching package: 'randomForest'

## The following object is masked from 'package:psych':
##
##     outlier

## The following object is masked from 'package:dplyr':
##
##     combine

## The following object is masked from 'package:ggplot2':
##
##     margin

## Important regclass change from 1.3:
## All functions that had a . in the name now have an _
## all.correlations -> all_correlations, cor.demo -> cor_demo, etc.

##
## Attaching package: 'regclass'

## The following object is masked from 'package:lattice':
##
##     qq
```

```r
library(e1071)
library(caret)
```

```
##
## Attaching package: 'caret'

## The following object is masked from 'package:VGAM':
##
##     predictors

## The following object is masked from 'package:survival':
##
##     cluster

## The following object is masked from 'package:purrr':
##
##     lift
```

```r
library(pROC)
```

```
## Type 'citation("pROC")' for a citation.

##
## Attaching package: 'pROC'

## The following objects are masked from 'package:stats':
##
##     cov, smooth, var
```

```r
library(ROCR)
library(klaR)
library(readxl)
library(rmarkdown)
library(car)
```

```
## Loading required package: carData

##
## Attaching package: 'car'

## The following object is masked from 'package:VGAM':
##
##     logit

## The following object is masked from 'package:boot':
##
##     logit

## The following object is masked from 'package:psych':
##
##     logit

## The following object is masked from 'package:dplyr':
##
##     recode

## The following object is masked from 'package:purrr':
##
##     some
```

```r
library(SciViews)
library(psych)
library(memisc)
```

```
##
## Attaching package: 'memisc'
## The following object is masked from 'package:car':
##
##     recode

## The following object is masked from 'package:VGAM':
##
##     Max

## The following object is masked from 'package:Matrix':
##
##     as.array

## The following objects are masked from 'package:dplyr':
##
##     collect, recode, rename, syms

## The following object is masked from 'package:purrr':
##
##     %@%

## The following object is masked from 'package:tibble':
##
##     view

## The following object is masked from 'package:ggplot2':
##
##     syms

## The following objects are masked from 'package:stats':
##
##     contr.sum, contr.treatment, contrasts

## The following object is masked from 'package:base':
##
##     as.array
```

```
#library(FFally)
```

Loading the excel file:

```
df <- read_xlsx("/Users/richardbritto/Desktop/MVA/Final_Data.xlsx")
head(df)
```

```
## # 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
```

```r
attach(df)
```

Exploring the data:

```r
dim(df)
```

```
## [1] 77 11
```

```r
head(df)
```

```
## # 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
```

```r
tail(df)
```

```
## # 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 IndianWolves   131  11.8    20    24    23   8.8    38    40   6.5 Female
## 2 IndianWolves   163  10.8    27    24    24   9.2    39    48   7   Female
## 3 IndianWolves   164  10.7    24    23    26   9.5    43    47   7.6 Female
## 4 IndianWolves   141  10.4    20    23    23   8.9    38    43   6   Female
## 5 IndianWolves   148  10.6    26    21    24   8.9    39    40   7   Female
## 6 IndianWolves   158  10.7    25    25    24   9.8    41    45   7.4 Female
```

```r
str(df)
```

```
## tibble [77 x 11] (S3: tbl_df/tbl/data.frame)
##  $ CanineGroup: chr [1:77] "ModernDog" "ModernDog" "ModernDog" "ModernDog" ...
##  $ X1         : num [1:77] 123 137 121 130 149 125 126 125 121 122 ...
##  $ X2         : num [1:77] 10.1 9.6 10.2 10.7 12 9.5 9.1 9.7 9.6 8.9 ...
##  $ X3         : num [1:77] 23 19 18 24 25 23 20 19 22 20 ...
##  $ X4         : num [1:77] 23 22 21 22 25 20 22 19 20 20 ...
##  $ X5         : num [1:77] 19 19 21 20 21 20 19 19 18 19 ...
##  $ X6         : num [1:77] 7.8 7.8 7.9 7.9 8.4 7.8 7.5 7.5 7.6 7.6 ...
##  $ X7         : num [1:77] 32 32 35 32 35 33 32 32 31 31 ...
##  $ X8         : num [1:77] 33 40 38 37 43 37 35 37 35 35 ...
##  $ X9         : num [1:77] 5.6 5.8 6.2 5.9 6.6 6.3 5.5 6.2 5.3 5.7 ...
##  $ Gender     : chr [1:77] "Male" "Male" "Male" "Male" ...
```

Cleaning the data:

```r
df$CanineGroup <- as.factor(df$CanineGroup)
str(df)
```

```
## tibble [77 x 11] (S3: tbl_df/tbl/data.frame)
##  $ CanineGroup: Factor w/ 5 levels "Cuons","GoldenJackal",..: 4 4 4 4 4 4 4 4 4 4 ...
##  $ X1         : num [1:77] 123 137 121 130 149 125 126 125 121 122 ...
##  $ X2         : num [1:77] 10.1 9.6 10.2 10.7 12 9.5 9.1 9.7 9.6 8.9 ...
##  $ X3         : num [1:77] 23 19 18 24 25 23 20 19 22 20 ...
##  $ X4         : num [1:77] 23 22 21 22 25 20 22 19 20 20 ...
##  $ X5         : num [1:77] 19 19 21 20 21 20 19 19 18 19 ...
##  $ X6         : num [1:77] 7.8 7.8 7.9 7.9 8.4 7.8 7.5 7.5 7.6 7.6 ...
##  $ X7         : num [1:77] 32 32 35 32 35 33 32 32 31 31 ...
##  $ X8         : num [1:77] 33 40 38 37 43 37 35 37 35 35 ...
##  $ X9         : num [1:77] 5.6 5.8 6.2 5.9 6.6 6.3 5.5 6.2 5.3 5.7 ...
##  $ Gender     : chr [1:77] "Male" "Male" "Male" "Male" ...
```

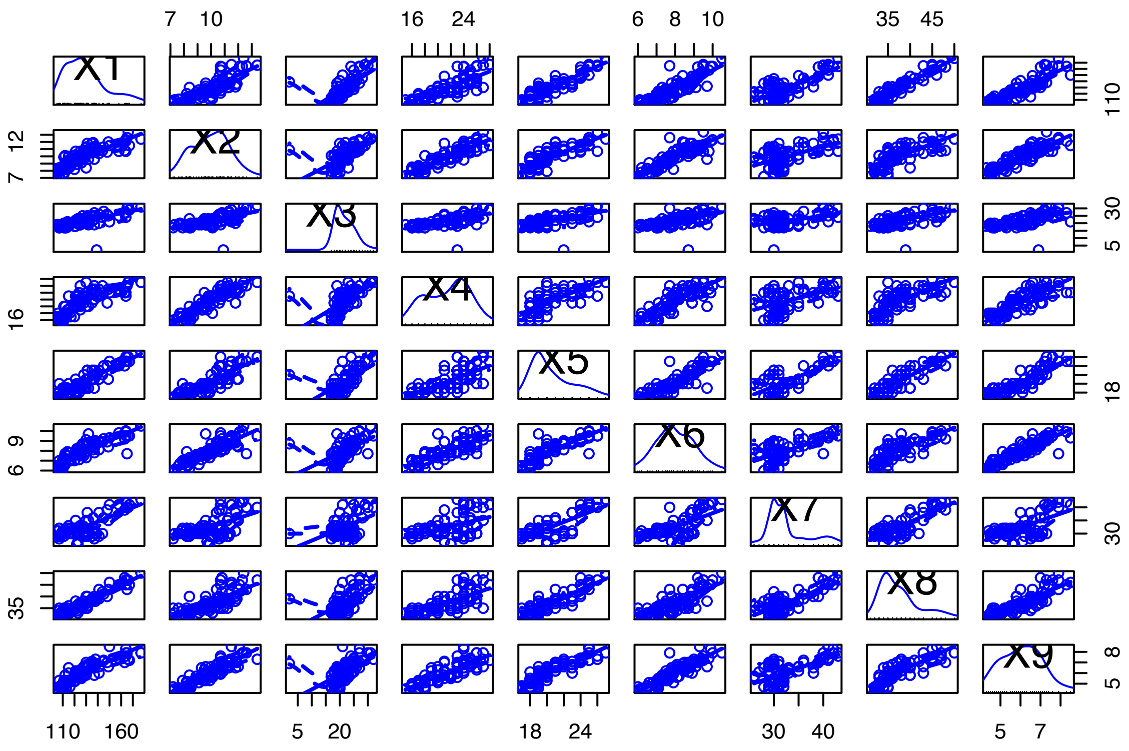
```r
df$Gender <- as.factor(df$Gender)
str(df)
```

```
## tibble [77 x 11] (S3: tbl_df/tbl/data.frame)
##  $ CanineGroup: Factor w/ 5 levels "Cuons","GoldenJackal",..: 4 4 4 4 4 4 4 4 4 4 ...
##  $ X1         : num [1:77] 123 137 121 130 149 125 126 125 121 122 ...
##  $ X2         : num [1:77] 10.1 9.6 10.2 10.7 12 9.5 9.1 9.7 9.6 8.9 ...
##  $ X3         : num [1:77] 23 19 18 24 25 23 20 19 22 20 ...
##  $ X4         : num [1:77] 23 22 21 22 25 20 22 19 20 20 ...
##  $ X5         : num [1:77] 19 19 21 20 21 20 19 19 18 19 ...
##  $ X6         : num [1:77] 7.8 7.8 7.9 7.9 8.4 7.8 7.5 7.5 7.6 7.6 ...
##  $ X7         : num [1:77] 32 32 35 32 35 33 32 32 31 31 ...
##  $ X8         : num [1:77] 33 40 38 37 43 37 35 37 35 35 ...
##  $ X9         : num [1:77] 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 ...
```

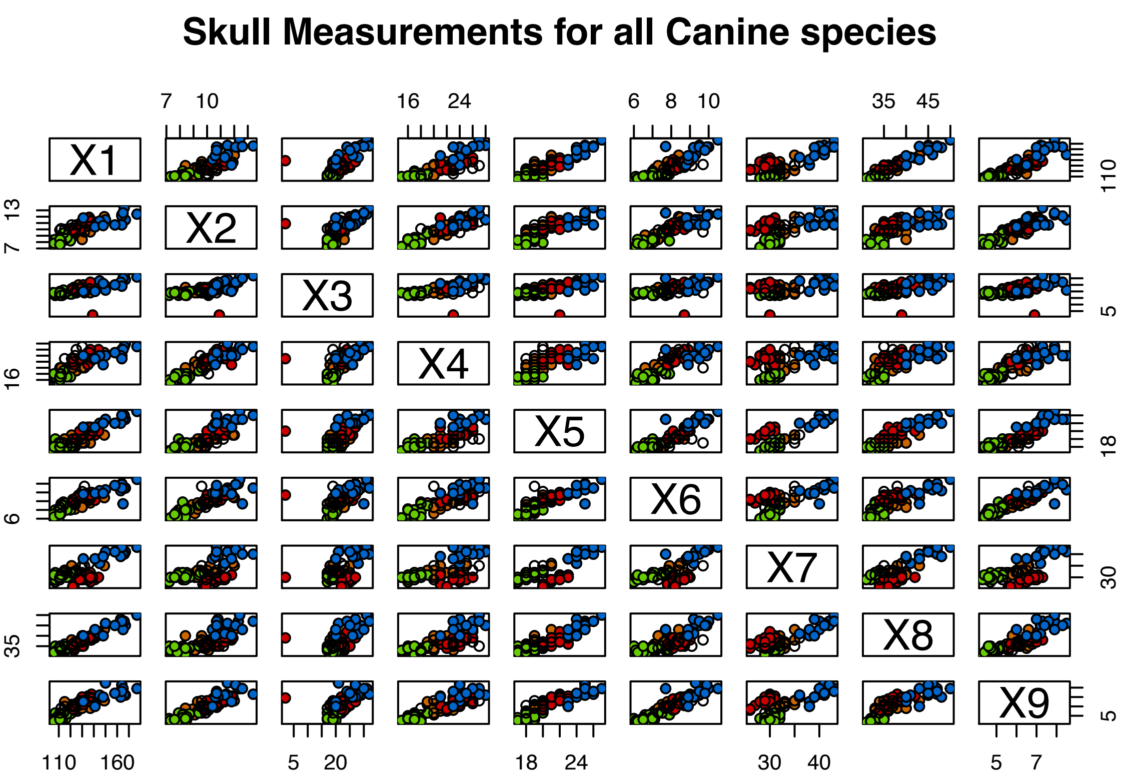
1. Using suitable graphical method, compare the distribution of the nine variables for the prehistoric and modern Thai dog.

a. Create a Draftsman plot for the 9 variables showing each species as a different color

```r
scatterplotMatrix(df[, 2:10])
```

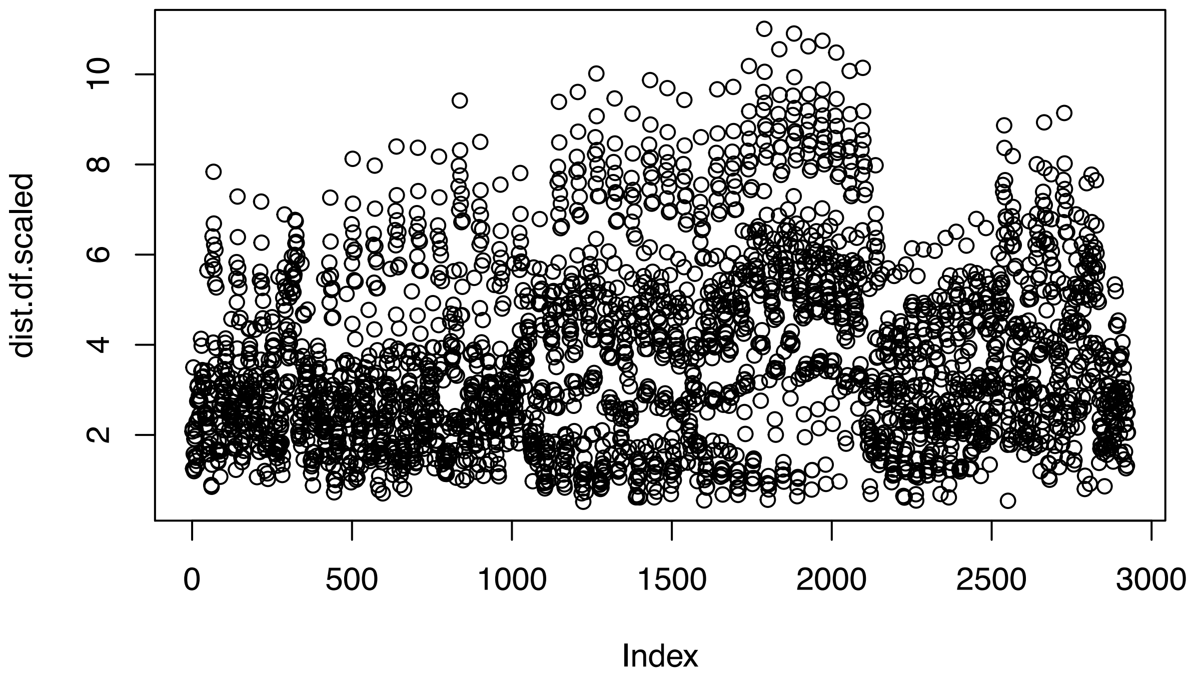


```r
pairs(df[2:10],
      main = "Skull Measurements for all Canine species",
      pch = 21,
      bg = c('#CC0000', '#66CC00', '#0066CC', '#CC6600')
      [unclass(df$CanineGroup)])
```



2. Create a distance matrix between the 5 canine groups:

```
mat.std = scale(df[c(-1, -11)])
dist.df.scaled <- dist(mat.std, method = "euclidean")
plot(dist.df.scaled)
```



3. Use principal components analysis to investigate the relationships between the species on the basis of these variables

```r
cor(df[c(-1, -11)])
```

```
##           X1        X2        X3        X4        X5        X6        X7
## X1 1.0000000 0.8259623 0.6841756 0.7976348 0.9066471 0.8515578 0.7589012
## X2 0.8259623 1.0000000 0.6184360 0.8897336 0.8213389 0.8457847 0.5597767
## X3 0.6841756 0.6184360 1.0000000 0.6200059 0.6166557 0.5608910 0.4516023
## X4 0.7976348 0.8897336 0.6200059 1.0000000 0.7402734 0.8085781 0.4707245
## X5 0.9066471 0.8213389 0.6166557 0.7402734 1.0000000 0.8537794 0.7424201
## X6 0.8515578 0.8457847 0.5608910 0.8085781 0.8537794 1.0000000 0.6456683
## X7 0.7589012 0.5597767 0.4516023 0.4707245 0.7424201 0.6456683 1.0000000
## X8 0.9494620 0.7460676 0.5906419 0.7151408 0.8777774 0.7984086 0.7867110
## X9 0.8833714 0.8874866 0.5750451 0.8229495 0.8826925 0.8942284 0.6478342
##           X8        X9
## X1 0.9494620 0.8833714
## X2 0.7460676 0.8874866
## X3 0.5906419 0.5750451
## X4 0.7151408 0.8229495
## X5 0.8777774 0.8826925
## X6 0.7984086 0.8942284
## X7 0.7867110 0.6478342
## X8 1.0000000 0.8380353
## X9 0.8380353 1.0000000
```

```r
data_pca <- prcomp(df[, -c(1, 11)], scale = TRUE)
```

```r
temp = as.matrix(df[, 1])
class(temp)
```

```
## [1] "matrix"
```

```r
temp
```

```
##       CanineGroup
##  [1,] "ModernDog"
##  [2,] "ModernDog"
##  [3,] "ModernDog"
##  [4,] "ModernDog"
##  [5,] "ModernDog"
##  [6,] "ModernDog"
##  [7,] "ModernDog"
##  [8,] "ModernDog"
##  [9,] "ModernDog"
## [10,] "ModernDog"
## [11,] "ModernDog"
## [12,] "ModernDog"
## [13,] "ModernDog"
## [14,] "ModernDog"
## [15,] "ModernDog"
## [16,] "ModernDog"
## [17,] "GoldenJackal"
## [18,] "GoldenJackal"
## [19,] "GoldenJackal"
## [20,] "GoldenJackal"
## [21,] "GoldenJackal"
## [22,] "GoldenJackal"
## [23,] "GoldenJackal"
## [24,] "GoldenJackal"
## [25,] "GoldenJackal"
## [26,] "GoldenJackal"
## [27,] "GoldenJackal"
## [28,] "GoldenJackal"
## [29,] "GoldenJackal"
## [30,] "GoldenJackal"
## [31,] "GoldenJackal"
## [32,] "GoldenJackal"
## [33,] "GoldenJackal"
## [34,] "GoldenJackal"
## [35,] "GoldenJackal"
## [36,] "GoldenJackal"
## [37,] "Cuons"
## [38,] "Cuons"
## [39,] "Cuons"
## [40,] "Cuons"
## [41,] "Cuons"
## [42,] "Cuons"
## [43,] "Cuons"
## [44,] "Cuons"
## [45,] "Cuons"
## [46,] "Cuons"
## [47,] "Cuons"
## [48,] "Cuons"
## [49,] "Cuons"
## [50,] "Cuons"
## [51,] "Cuons"
## [52,] "Cuons"
## [53,] "Cuons"
## [54,] "ThaiDogs"
## [55,] "ThaiDogs"
## [56,] "ThaiDogs"
## [57,] "ThaiDogs"
## [58,] "ThaiDogs"
## [59,] "ThaiDogs"
## [60,] "ThaiDogs"
## [61,] "ThaiDogs"
## [62,] "ThaiDogs"
## [63,] "ThaiDogs"
## [64,] "IndianWolves"
## [65,] "IndianWolves"
## [66,] "IndianWolves"
## [67,] "IndianWolves"
## [68,] "IndianWolves"
## [69,] "IndianWolves"
## [70,] "IndianWolves"
## [71,] "IndianWolves"
## [72,] "IndianWolves"
## [73,] "IndianWolves"
## [74,] "IndianWolves"
## [75,] "IndianWolves"
## [76,] "IndianWolves"
## [77,] "IndianWolves"
```

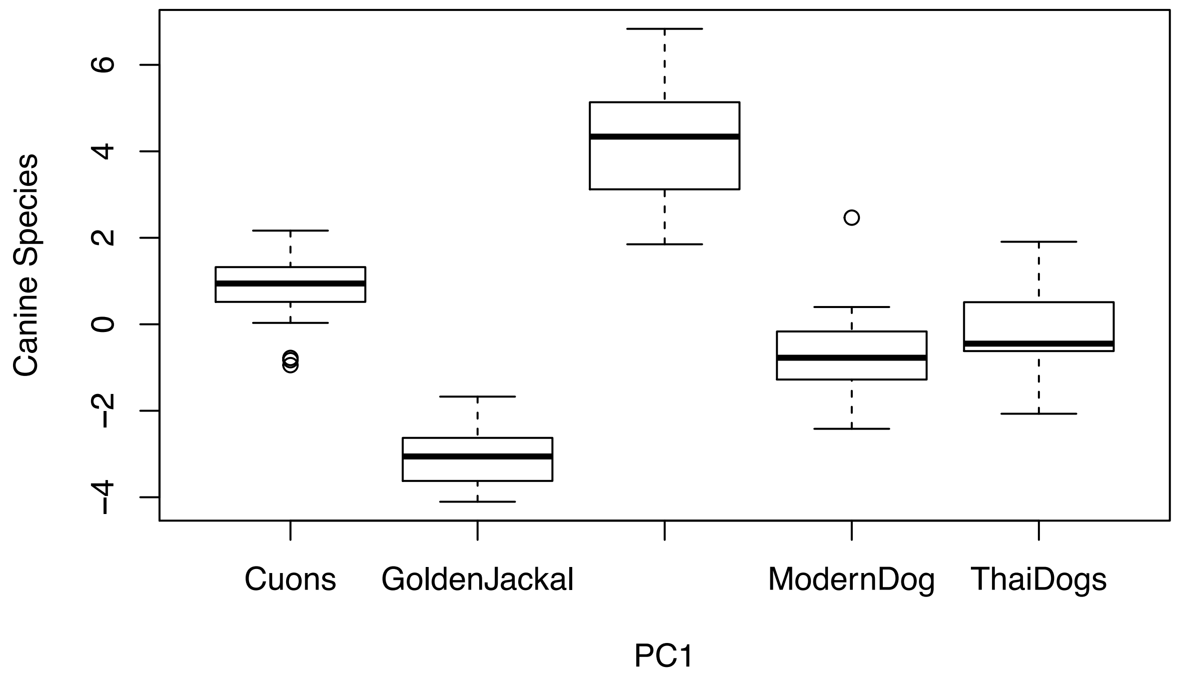
```
data_pca
```

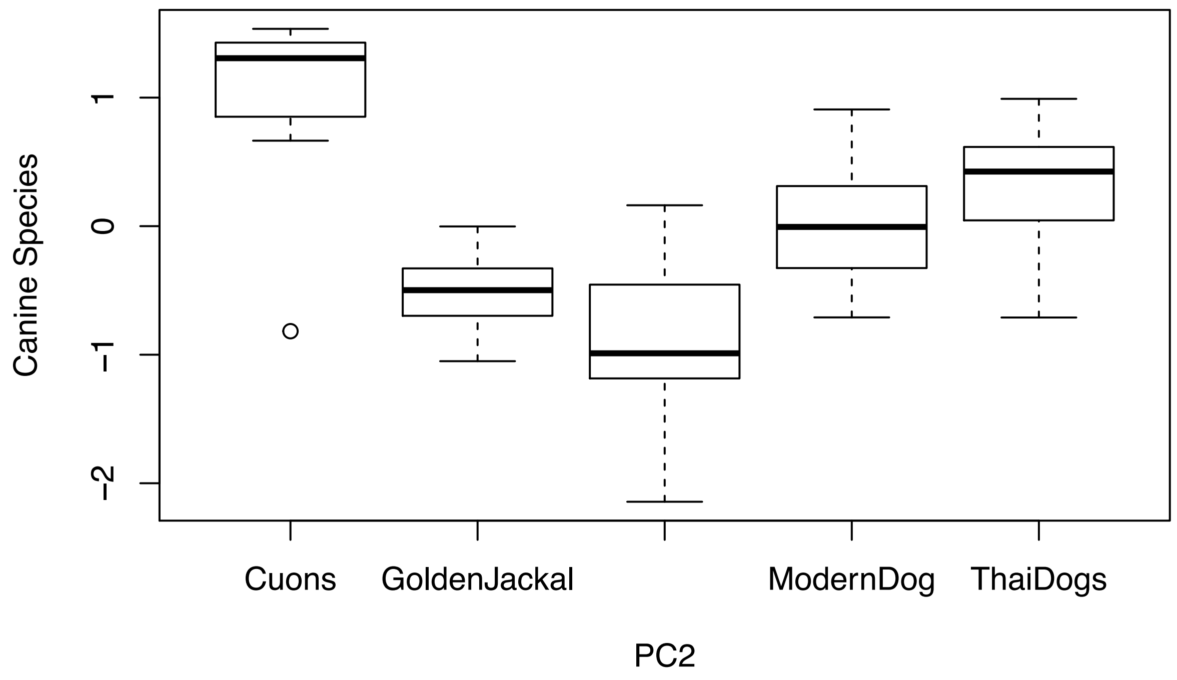
```
## Standard deviations (1, .., p=9):
## [1] 2.6555963 0.8391652 0.7365758 0.4390554 0.4241988 0.3627806 0.3031519
## [8] 0.2652189 0.1857418
##
## Rotation (n x k) = (9 x 9):
##          PC1         PC2         PC3         PC4         PC5         PC6
## X1 0.3636408 -0.11451510  0.08210471 -0.30326354  0.24950692 -0.07899550
## X2 0.3424554  0.31490128 -0.19979188  0.33605928  0.01517931  0.49451257
## X3 0.2665621  0.32018675  0.87894338  0.04161625 -0.18169514 -0.04568559
## X4 0.3265349  0.44638084 -0.16540131  0.26534253  0.54545187 -0.21526217
## X5 0.3539586 -0.14160855 -0.03861441 -0.26352534 -0.33012092  0.43239890
## X6 0.3459444  0.06792334 -0.26250857  0.05378069 -0.51974026 -0.68294862
## X7 0.2859405 -0.68736531  0.13651981  0.64014932  0.05443187 -0.01170970
## X8 0.3470802 -0.28877388  0.03666665 -0.47256682  0.40753260 -0.10978603
## X9 0.3544268  0.07362113 -0.25111557 -0.13231892 -0.24254817  0.18765482
##            PC7         PC8          PC9
## X1  0.05543869  0.16005914  0.811637429
## X2  0.13657790  0.60411640 -0.048224206
## X3  0.08257828 -0.03476461 -0.094992855
## X4 -0.30849700 -0.39244126 -0.057608736
## X5 -0.67024916 -0.19081879 -0.046144083
## X6 -0.08734948  0.23986950 -0.046794522
## X7  0.03463451 -0.11415807  0.002559605
## X8  0.12889717  0.23397418 -0.567438948
## X9  0.63372357 -0.54081471 -0.016253801
```

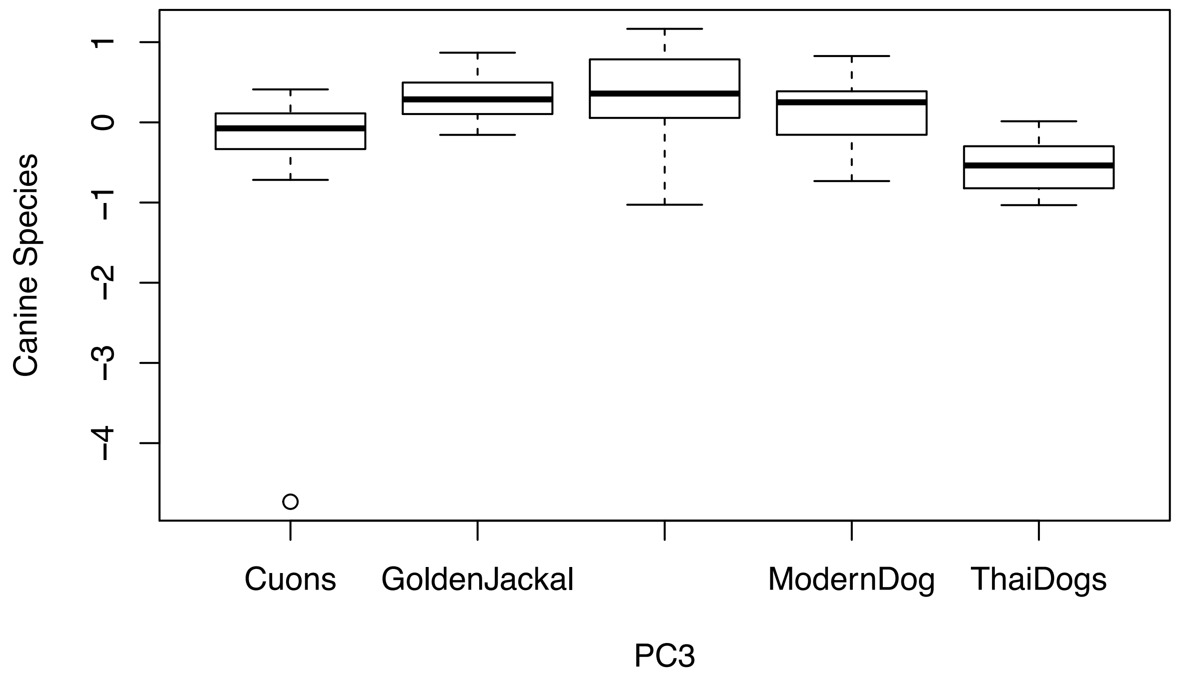
```
predict(data_pca)[, 1]
```

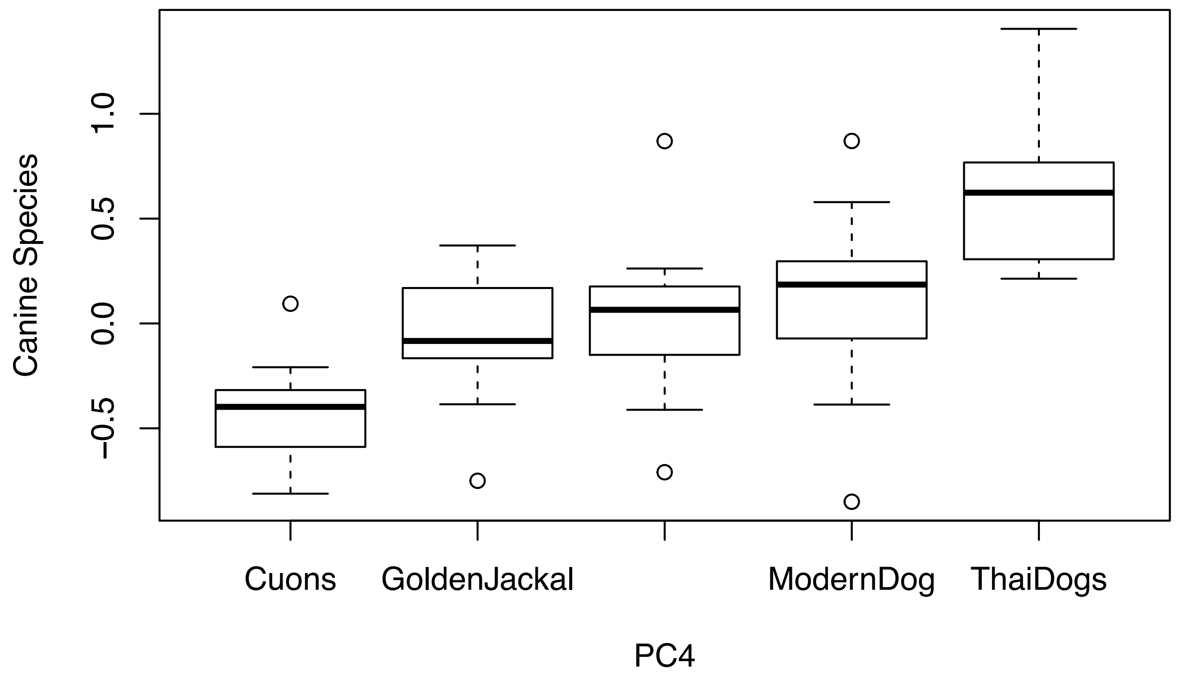
```
##  [1] -0.68594144 -0.24567287 -0.08646546  0.16794155  2.46605602 -0.31165997
##  [7] -1.13265883 -0.95932707 -1.42772439 -1.42312985 -1.08479346 -2.41795380
## [13] -1.49439221 -0.59855768 -0.86184015  0.39898643 -2.50221460 -2.66854242
## [19] -3.13382930 -2.58652273 -2.21301819 -2.97976934 -2.78947778 -2.53185477
## [25] -1.67354577 -2.78123568 -2.82501243 -3.30622428 -4.10431266 -3.66153536
## [31] -4.04399515 -3.72261526 -3.82745453 -3.58415662 -3.18995154 -3.28698703
## [37] -0.83046450  1.69865955  1.94265070  1.24244782  1.32210362  1.18642807
## [43]  0.94267205  0.83611277  1.18696596  0.58012526  0.87381349  2.16615489
## [49]  0.03166471 -0.94467005  1.96212575  0.51806252 -0.77814254 -1.81012401
## [55] -0.60672494  1.90795558 -2.06926137  1.40688395  0.51098957  0.42526384
## [61] -0.37712553 -0.62016146 -0.51807155  4.97863584  5.43014384  4.19582141
## [67]  3.12145386  6.83250997  5.78067966  5.13564203  4.48437886  1.98423294
## [73]  3.99779940  4.49728950  1.85001891  2.55231795  4.08213029
```

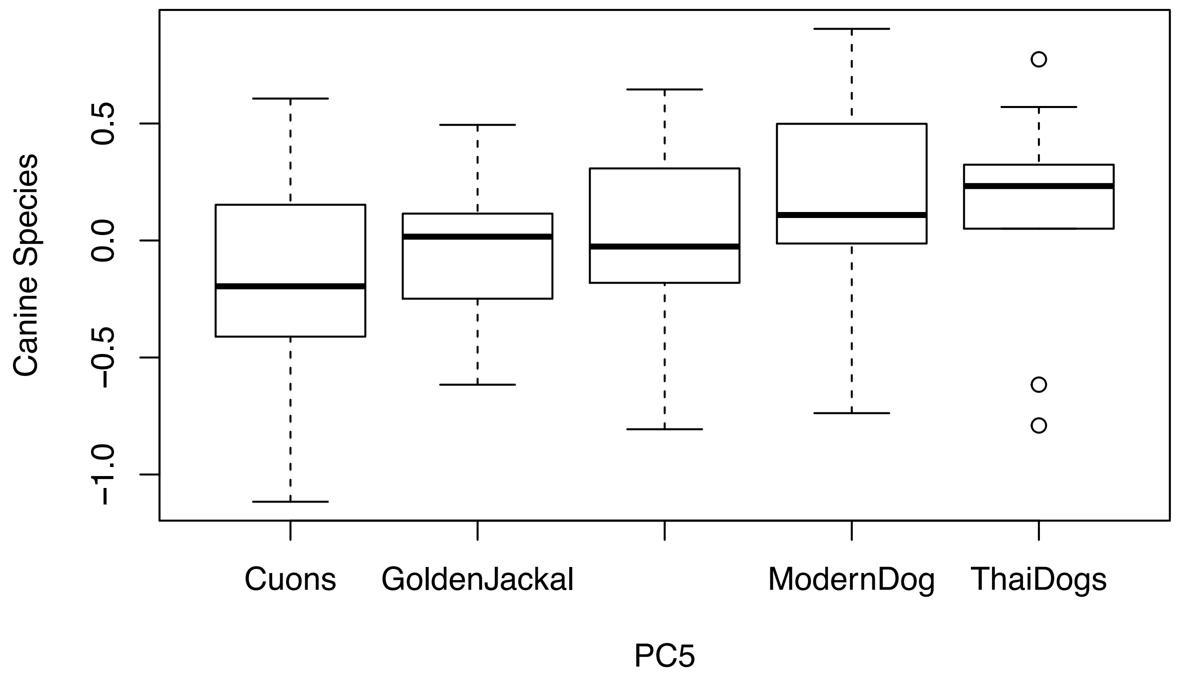
```
out <- sapply(1:5, function(i){plot(df$CanineGroup,
                            data_pca$x[, i],
                            xlab=paste("PC", i, sep = ""),
                            ylab="Canine Species")})
```


Canine Species
6
4
2
0
−2
−4
Cuons
GoldenJackal
ModernDog
ThaiDogs
PC1


Canine Species
1
0
−1
−2
Cuons
GoldenJackal
ModernDog
ThaiDogs
PC2


Canine Species
1
0
−1
−2
−3
−4
Cuons
GoldenJackal
ModernDog
ThaiDogs
PC3

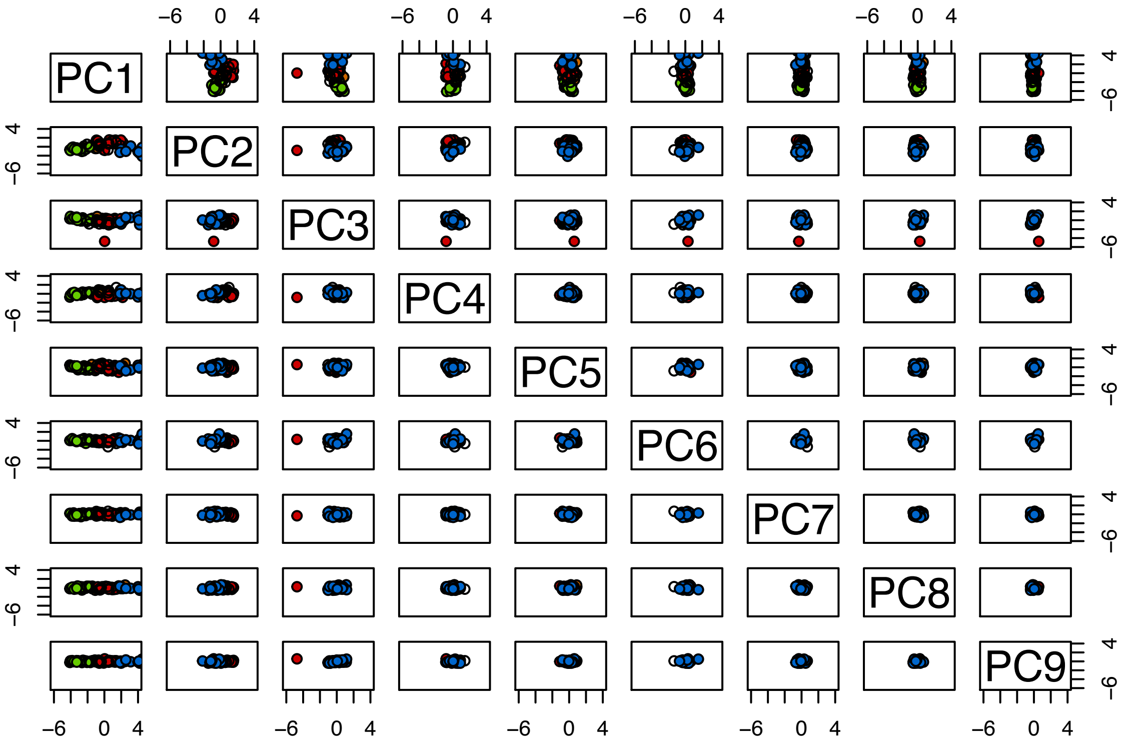

Canine Species
1.0
0.5
0.0
−0.5
Cuons
GoldenJackal
ModernDog
ThaiDogs
PC4



Below we can see the relation between each variables for each species:

```
clr <- character(nrow(df))
clr[] <- "black"
clr[df$CanineGroup == "ModernDog"] <- "#CC0000"
clr[df$CanineGroup == "GoldenJackal"] <- "#66CC00"
clr[df$CanineGroup == "Cuons"] <- "#0066CC"
clr[df$CanineGroup == "ThaiDogs"] <- "#CC6600"
clr[df$CanineGroup == "IndianWolves"] <- "#6600CC"
```

```
pairs(data_pca$x[, 1:9],
      ylim = c(-6, 4),
      xlim = c(-6, 4),
      pch = 21,
      bg = c('#CC0000', '#66CC00', '#0066CC', '#CC6600')
      [unclass(df$CanineGroup)])
```



```r
summary(data_pca)
```

```
## Importance of components:
##                           PC1     PC2     PC3     PC4     PC5     PC6     PC7
## Standard deviation     2.6556 0.83917 0.73658 0.43906 0.42420 0.36278 0.30315
## Proportion of Variance 0.7836 0.07824 0.06028 0.02142 0.01999 0.01462 0.01021
## Cumulative Proportion  0.7836 0.86182 0.92210 0.94352 0.96352 0.97814 0.98835
##                            PC8     PC9
## Standard deviation     0.26522 0.18574
## Proportion of Variance 0.00782 0.00383
## Cumulative Proportion  0.99617 1.00000
```

loadings (eigenvectors) are stored in data_pca$rotation:

```r
data_pca$rotation
```

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

Eigenvalues are sdev^2:

```
eigen_data<-data_pca$sdev^2
```

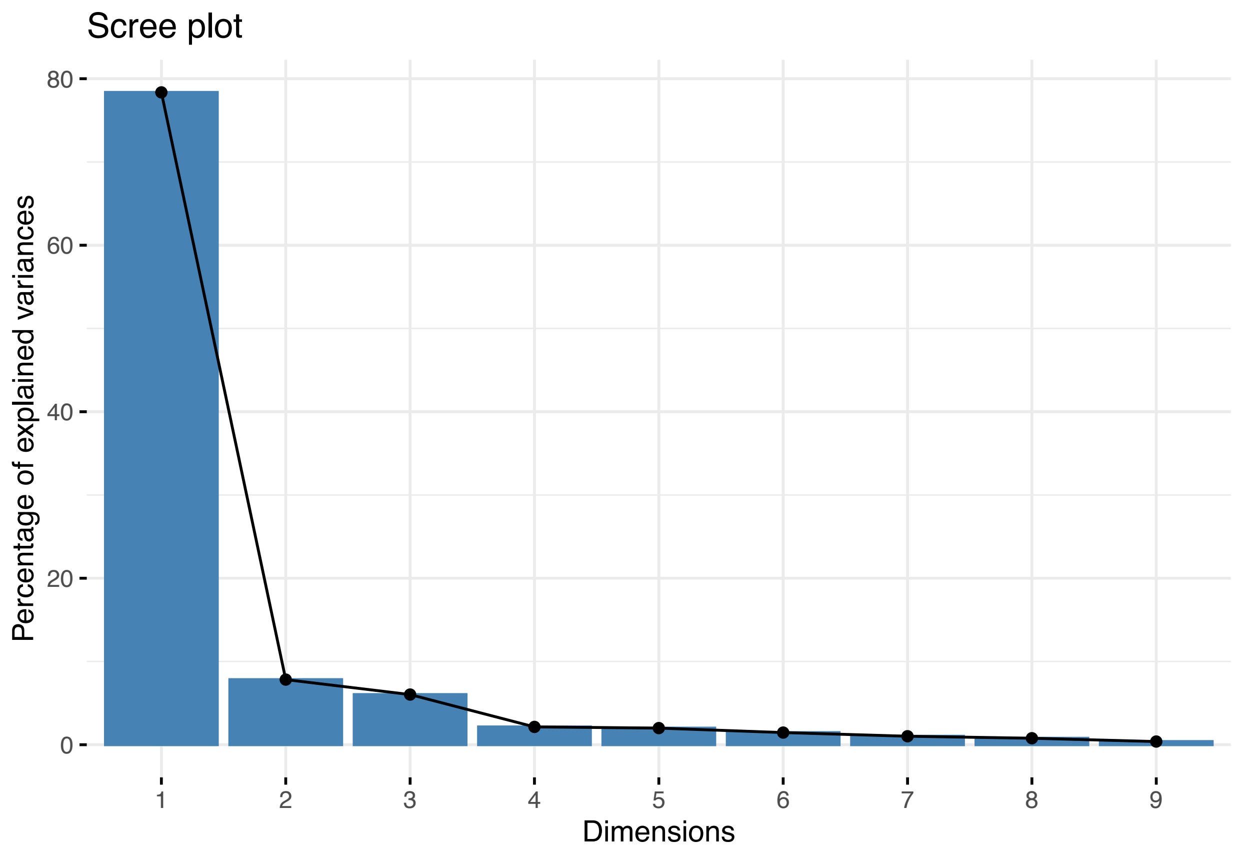
Sample scores stored in data_pca$x:

```
head(data_pca$x)
```

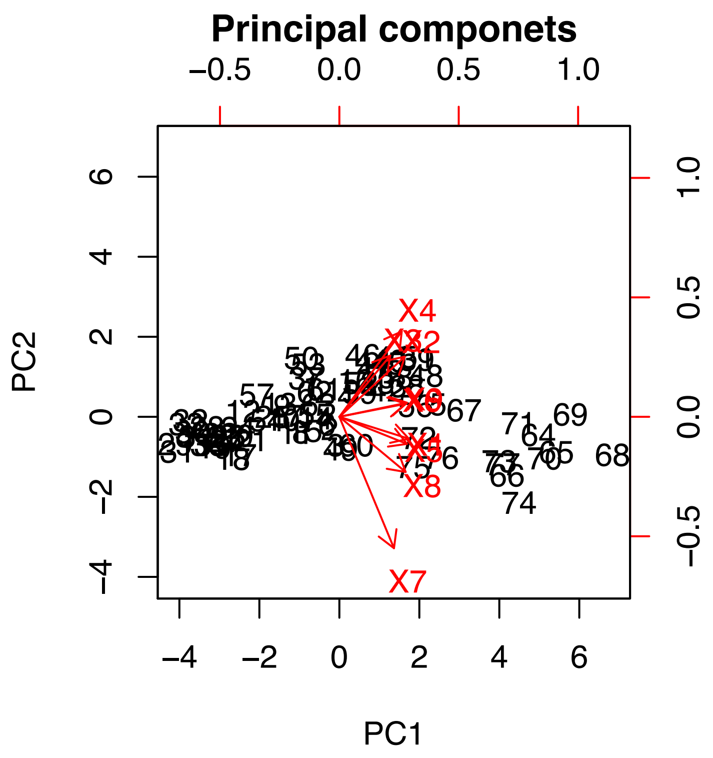
```
##              PC1        PC2        PC3         PC4          PC5         PC6
## [1,] -0.68594144  0.7845059  0.3006808  0.87047017  0.099293422 -0.13703394
## [2,] -0.24567287 -0.2998432 -0.3386984 -0.38688425  0.904531523 -0.40703561
## [3,] -0.08646546 -0.7099032 -0.7324535  0.36215960  0.007597324  0.34684409
## [4,]  0.16794155  0.5263915  0.4238685  0.25556873  0.116376021  0.15817439
## [5,]  2.46605602  0.3055451  0.2215356  0.12938866  0.883927889  0.13952874
## [6,] -0.31165997 -0.1929752  0.4205179 -0.01594082 -0.279041559  0.03359648
##              PC7         PC8         PC9
## [1,] -0.12574305 -0.08103187  0.27299561
## [2,]  0.21201705  0.24663975  0.14167354
## [3,] -0.04042272 -0.04720248 -0.36986352
## [4,]  0.09155686  0.34942670  0.02832178
## [5,]  0.34760207  0.63208273 -0.03260669
## [6,]  0.38738491 -0.23534059 -0.10684702
```

Viewing the scree plot to see how many PCs are optimal for our data and we can conclude that maximum variance is covered by the top 3 PCs i.e the PC1, PC2, PC3:

```
fviz_eig(data_pca)
```

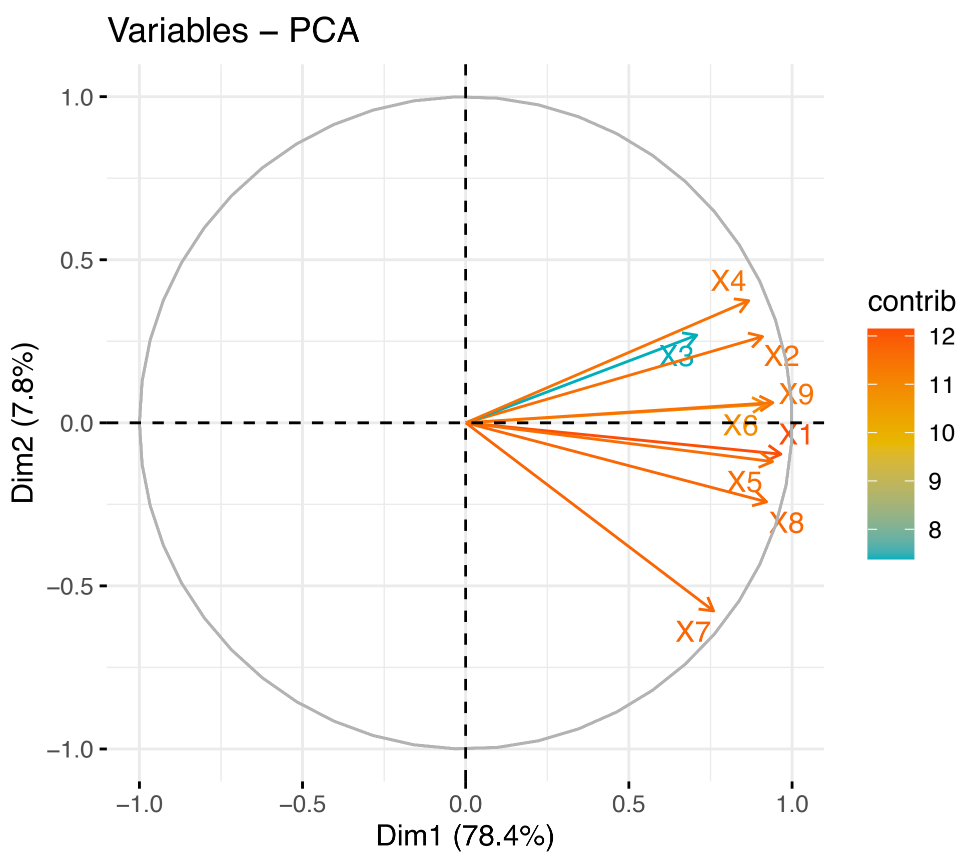


```r
biplot(data_pca, scale = 0, main = 'Principal componets')
```



We can see the contributions of each variables to the top 2 dimensions and this gives us some good insights on how do the variables contributre to all the data points in our data:

```
fviz_pca_var(data_pca,
             col.var = "contrib",
             gradient.cols = c("#00AFBB", "#E7B800", "#FC4E07"),
             repel = TRUE)
```



4. Carry out cluster analysis to study relation between different specifies.

a. Who is Indian Wolf related to?

Hirerarchic cluster analysis, Nearest-neighbor Standardizing the data with scale()

```
temp_values = c('X1', 'X2', 'X3', 'X4', 'X5', 'X6', 'X7', 'X8', 'X9')
df_species = df[, -11]
df_species
```

```
## # A tibble: 77 x 10
##    CanineGroup    X1    X2    X3    X4    X5    X6    X7    X8    X9
##    <fct>       <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
##  1 ModernDog     123  10.1    23    23    19   7.8    32    33   5.6
##  2 ModernDog     137   9.6    19    22    19   7.8    32    40   5.8
##  3 ModernDog     121  10.2    18    21    21   7.9    35    38   6.2
##  4 ModernDog     130  10.7    24    22    20   7.9    32    37   5.9
##  5 ModernDog     149  12      25    25    21   8.4    35    43   6.6
##  6 ModernDog     125   9.5    23    20    20   7.8    33    37   6.3
##  7 ModernDog     126   9.1    20    22    19   7.5    32    35   5.5
##  8 ModernDog     125   9.7    19    19    19   7.5    32    37   6.2
##  9 ModernDog     121   9.6    22    20    18   7.6    31    35   5.3
## 10 ModernDog     122   8.9    20    20    19   7.6    31    35   5.7
## # ... with 67 more rows
```

```
df_species = df_species %>%
  group_by(CanineGroup)  %>%
  summarise_at(temp_values, mean, na.rm = TRUE)
```

```
setDT(df_species)

df_species = df_species %>%
  remove_rownames %>%
  column_to_rownames(var = "CanineGroup")

mat.std_species = scale(df_species)
```

Creating a (Euclidean) distance matrix of the standardized data :

```
dist_species <- dist(mat.std_species, method = "euclidean")
```

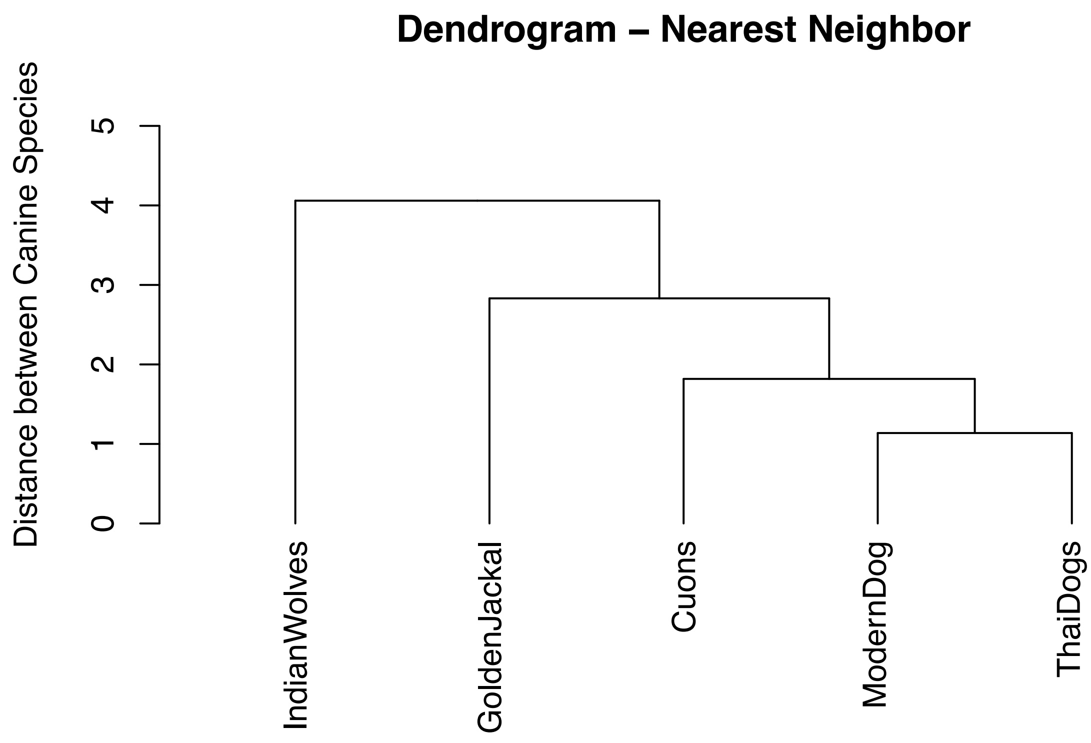
```
clusspecies_nn <- hclust(dist_species, method = "single")
```

```
dist_species
```

```
##                 Cuons GoldenJackal IndianWolves ModernDog
## GoldenJackal 4.680179
## IndianWolves 4.059156     7.806678
## ModernDog    2.062237     2.831293     5.171149
## ThaiDogs     1.818116     3.524634     4.964169  1.137355
```

As we see above that Indian Wolf is closed to Cuans with a distance of 4.059156. Futher, we can visualize this using dendrogram or a kmeans plot.

```
# For Nearest Neighbor - Single Linkage:
par(mar = c(8, 4, 2, 1) + 0.1)
plot(as.dendrogram(clusspecies_nn),
     ylab = "Distance between Canine Species",
     ylim = c(0, 5.5),
     main = "Dendrogram - Nearest Neighbor")
```

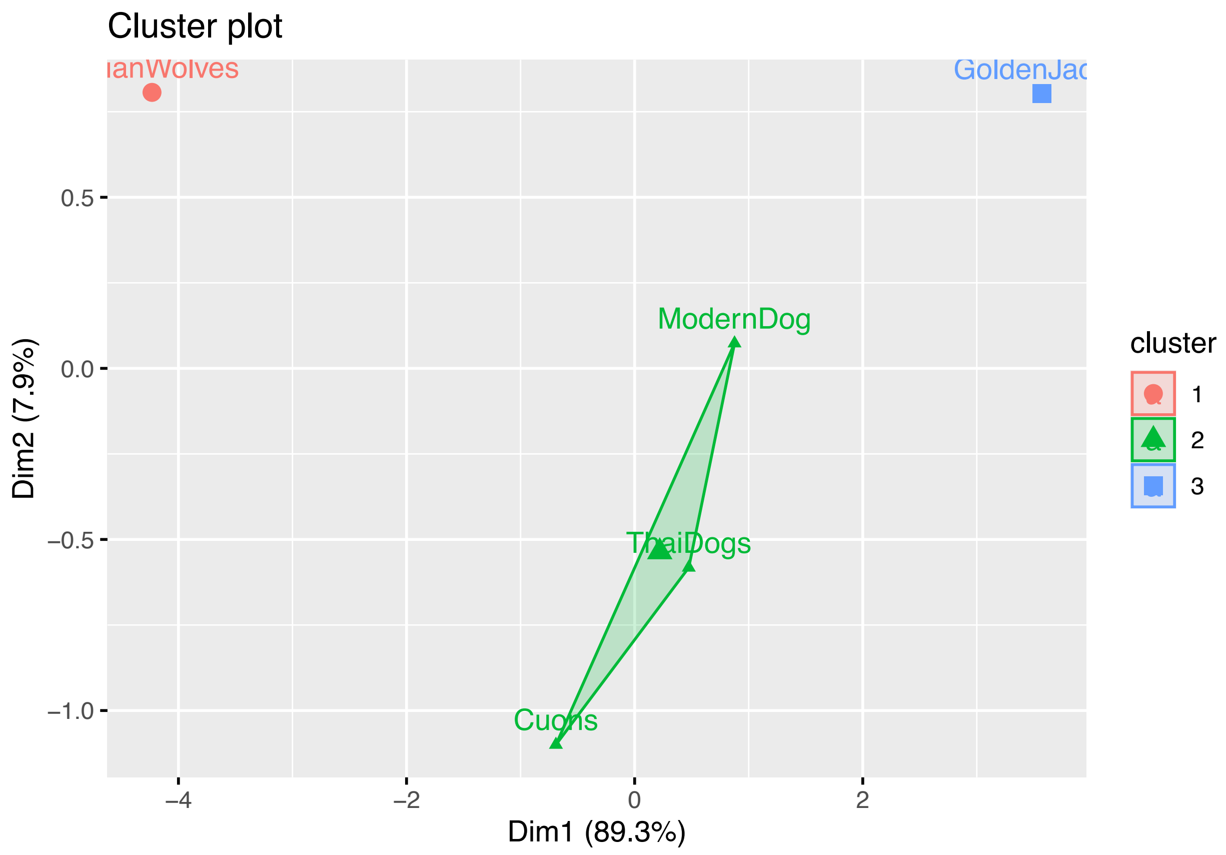


IndianWolves is closest to Cuons

Kmeans

```
kmeans_3_species <- kmeans(mat.std_species, 3, nstart = 10)

fviz_cluster(kmeans_3_species, data = df_species)
```



5. Identify the important factors underlying the Skull measurement

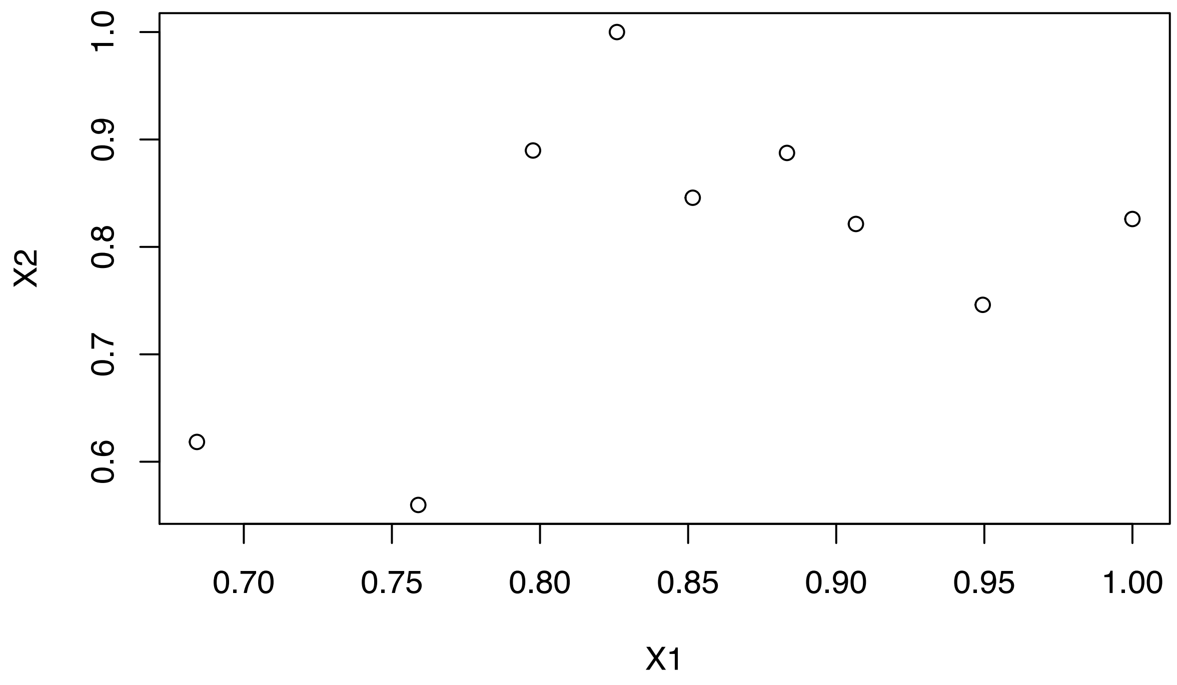
a. Is there a relationships between the species with respect to these factors?

Calculating the correlation matrix for all the numeric data in our dataset

```
corr.mat = cor(df[, 2:10])
corr.mat
```

```
##           X1        X2        X3        X4        X5        X6        X7
## X1 1.0000000 0.8259623 0.6841756 0.7976348 0.9066471 0.8515578 0.7589012
## X2 0.8259623 1.0000000 0.6184360 0.8897336 0.8213389 0.8457847 0.5597767
## X3 0.6841756 0.6184360 1.0000000 0.6200059 0.6166557 0.5608910 0.4516023
## X4 0.7976348 0.8897336 0.6200059 1.0000000 0.7402734 0.8085781 0.4707245
## X5 0.9066471 0.8213389 0.6166557 0.7402734 1.0000000 0.8537794 0.7424201
## X6 0.8515578 0.8457847 0.5608910 0.8085781 0.8537794 1.0000000 0.6456683
## X7 0.7589012 0.5597767 0.4516023 0.4707245 0.7424201 0.6456683 1.0000000
## X8 0.9494620 0.7460676 0.5906419 0.7151408 0.8777774 0.7984086 0.7867110
## X9 0.8833714 0.8874866 0.5750451 0.8229495 0.8826925 0.8942284 0.6478342
##           X8        X9
## X1 0.9494620 0.8833714
## X2 0.7460676 0.8874866
## X3 0.5906419 0.5750451
## X4 0.7151408 0.8229495
## X5 0.8777774 0.8826925
## X6 0.7984086 0.8942284
## X7 0.7867110 0.6478342
## X8 1.0000000 0.8380353
## X9 0.8380353 1.0000000
```

```
plot(corr.mat)
```



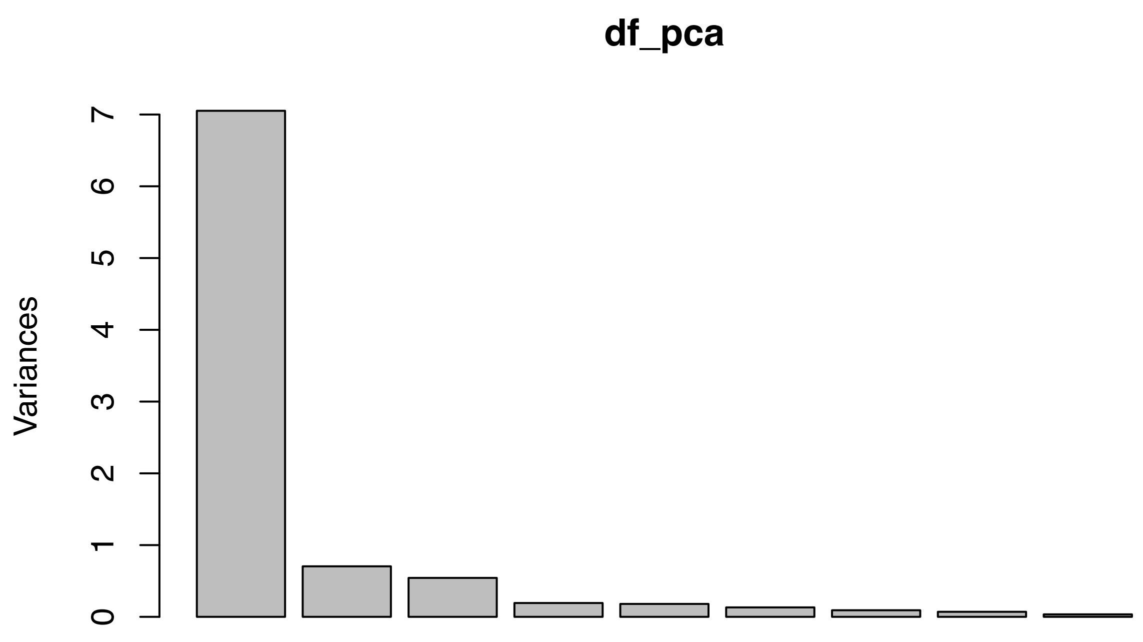
This is the correlation plot

Calculating the PCA and plotting these variances:

```
df_pca <- prcomp(df[2:10], scale = TRUE)
summary(df_pca)
```

```
## Importance of components:
##                           PC1     PC2     PC3     PC4     PC5     PC6     PC7
## Standard deviation     2.6556 0.83917 0.73658 0.43906 0.42420 0.36278 0.30315
## Proportion of Variance 0.7836 0.07824 0.06028 0.02142 0.01999 0.01462 0.01021
## Cumulative Proportion  0.7836 0.86182 0.92210 0.94352 0.96352 0.97814 0.98835
##                            PC8     PC9
## Standard deviation     0.26522 0.18574
## Proportion of Variance 0.00782 0.00383
## Cumulative Proportion  0.99617 1.00000
```

```r
plot(df_pca)
```



As from Q3 we can see that the top 3 PCs restore maximum variance

A table containing eigenvalues and %'s accounted, follows. Eigenvalues are the sdev^2:

```r
eigen_values <- round(df_pca$sdev^2, 2)
head(eigen_values)
```

```
## [1] 7.05 0.70 0.54 0.19 0.18 0.13
```

Eigen vectors as follows:

```r
eigen_vectors <- df_pca$rotation
```

Taking the 3 PCs to generate linear combinations for all the variables:

```r
pcafactors <- eigen_vectors[, 1:3]
pcafactors
```

```
##          PC1         PC2         PC3
## X1 0.3636408 -0.11451510  0.08210471
## X2 0.3424554  0.31490128 -0.19979188
## X3 0.2665621  0.32018675  0.87894338
## X4 0.3265349  0.44638084 -0.16540131
## X5 0.3539586 -0.14160855 -0.03861441
## X6 0.3459444  0.06792334 -0.26250857
## X7 0.2859405 -0.68736531  0.13651981
## X8 0.3470802 -0.28877388  0.03666665
## X9 0.3544268  0.07362113 -0.25111557
```

Multiplying each column of the eigenvector's matrix by the square-root of the corresponding eigenvalue in order to get the factor loadings:

```
unrot.fact <- sweep(pcafactors, MARGIN = 2, df_pca$sdev[1:3], `*`)
unrot.fact
```

```
##          PC1         PC2         PC3
## X1 0.9656832 -0.09609709  0.06047634
## X2 0.9094234  0.26425421 -0.14716186
## X3 0.7078814  0.26868959  0.64740843
## X4 0.8671448  0.37458728 -0.12183061
## X5 0.9399711 -0.11883297 -0.02844244
## X6 0.9186886  0.05699890 -0.19335746
## X7 0.7593427 -0.57681308  0.10055719
## X8 0.9217049 -0.24232900  0.02700777
## X9 0.9412145  0.06178029 -0.18496566
```

Computing communalities:

```
communalities <- rowSums(unrot.fact^2)
communalities
```

```
##        X1        X2        X3        X4        X5        X6        X7        X8
## 0.9454361 0.9185378 0.9924279 0.9070985 0.8984760 0.8846248 0.9194263 0.9089926
##        X9
## 0.9239138
```

Performing the varimax rotation. The default in the varimax function is norm=TRUE thus, Kaiser normalization is carried out:

```
rot.fact <- varimax(unrot.fact)
rot.fact
```

```
## $loadings
##
## Loadings:
##    PC1    PC2    PC3
## 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
##
##                  PC1   PC2   PC3
## SS loadings    3.985 2.919 1.395
## Proportion Var 0.443 0.324 0.155
## Cumulative Var 0.443 0.767 0.922
##
## $rotmat
##             [,1]        [,2]      [,3]
## [1,]  0.7231780 -0.59076663 0.3577826
## [2,]  0.4847295  0.80315318 0.3463846
## [3,] -0.4919867 -0.07706991 0.8671847
```

The print method of varimax omits loadings less than abs(0.1).

```
fact.load <- rot.fact$loadings[1:3, 1:3]
fact.load
```

```
##          PC1        PC2       PC3
## X1 0.6220262 -0.6523350 0.3646623
## X2 0.8581685 -0.3136786 0.2892930
## X3 0.3236497 -0.2522895 0.9077603
```

Computing the rotated factor scores for the 25 European Countries.

```
scale <- scale(df[, 2:10])
head(scale)
```

```
##               X1          X2         X3         X4         X5          X6
## [1,] -0.34133663  0.09904431  0.3235696  0.4459670 -0.5997769 -0.19537598
## [2,]  0.45857835 -0.25733007 -0.6255679  0.1499372 -0.5997769 -0.19537598
## [3,] -0.45561020  0.17031919 -0.8628523 -0.1460926  0.2034026 -0.09768799
## [4,]  0.05862086  0.52669357  0.5608540  0.1499372 -0.1981871 -0.09768799
## [5,]  1.14421975  1.45326697  0.7981384  1.0380266  0.2034026  0.39075196
## [6,] -0.22706306 -0.32860495  0.3235696 -0.4421225 -0.1981871 -0.19537598
##              X7          X8         X9
## [1,] -0.1244973 -0.99951775 -0.4661440
## [2,] -0.1244973  0.58968599 -0.2700069
## [3,]  0.5944746  0.13562778  0.1222673
## [4,] -0.1244973 -0.09140133 -0.1719384
## [5,]  0.5944746  1.27077331  0.5145414
## [6,]  0.1151600 -0.09140133  0.2203358
```

```
fit.pc <- principal(df[, 2:10], nfactors = 3, rotate = "varimax")
fit.pc
```

```
## Principal Components Analysis
## Call: principal(r = df[, 2:10], 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.pc$values, 3)
```

```
## [1] 7.052 0.704 0.543 0.193 0.180 0.132 0.092 0.070 0.035
```

```
fit.pc$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
```

Loadings with more digits

```
for (i in c(3, 1)) { print(fit.pc$loadings[[1, i]])}
```

```
## [1] 0.3646623
## [1] 0.6220262
```

Communalities

```
fit.pc$communality
```

```
##        X1        X2        X3        X4        X5        X6        X7        X8 
## 0.9454361 0.9185378 0.9924279 0.9070985 0.8984760 0.8846248 0.9194263 0.9089926 
##        X9 
## 0.9239138
```

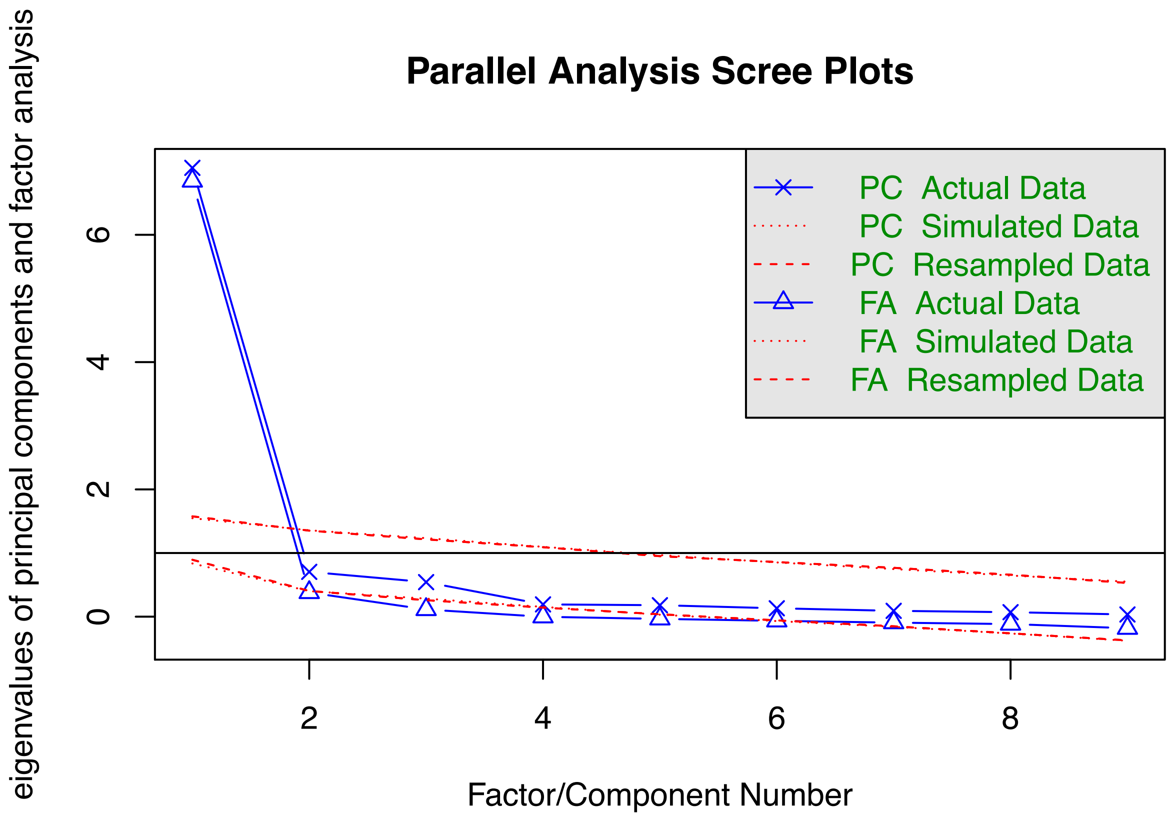
Mandible is able to restore 95% of the total variance Rotated factor scores

```
head(fit.pc$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
```

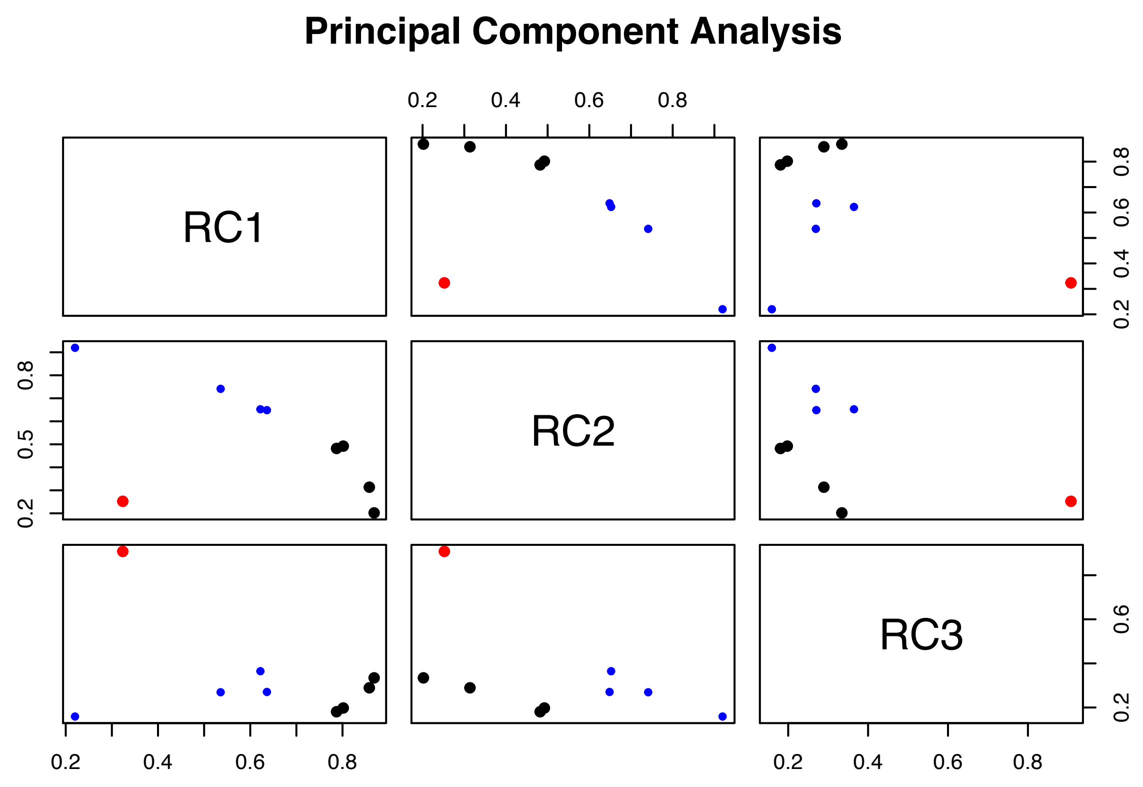
FA utilities

```
fa.parallel(df[, 2:10]) # See factor recommendation
```



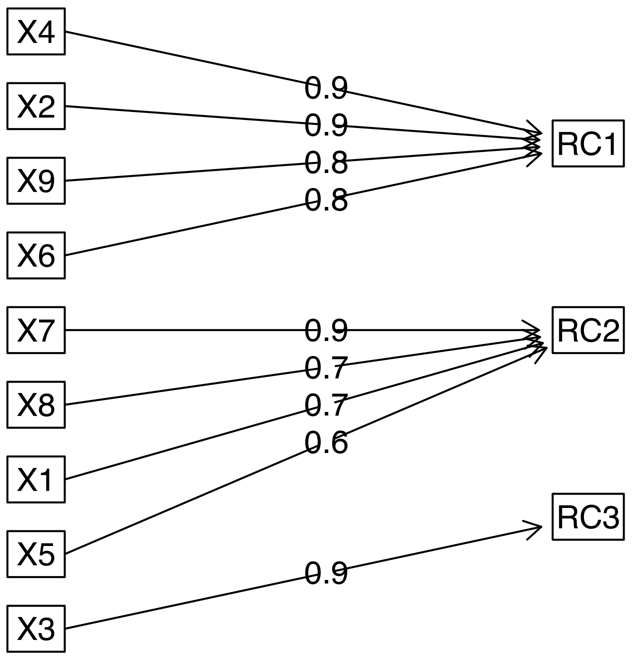
```
## Parallel analysis suggests that the number of factors =  1  and the number of components =  1
```

```
fa.plot(fit.pc) # See Correlations within Factors
```

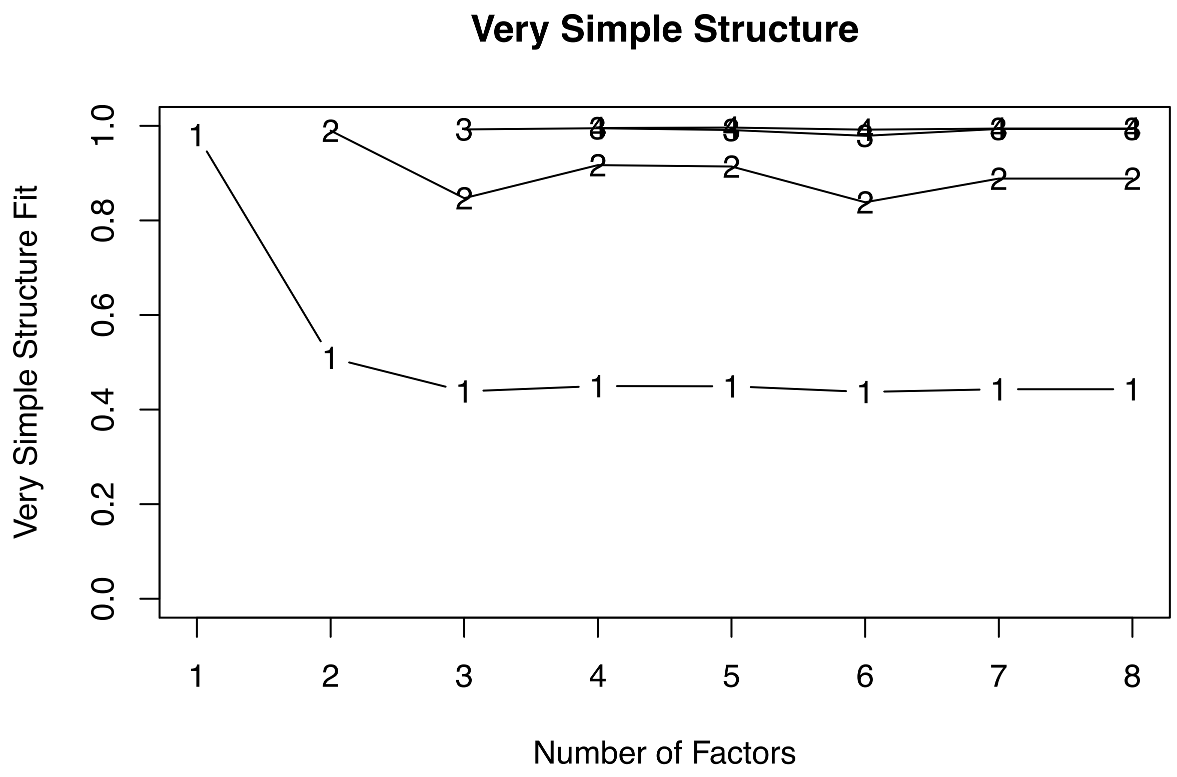


```
fa.diagram(fit.pc) # Visualize the relationship
```

Components Analysis



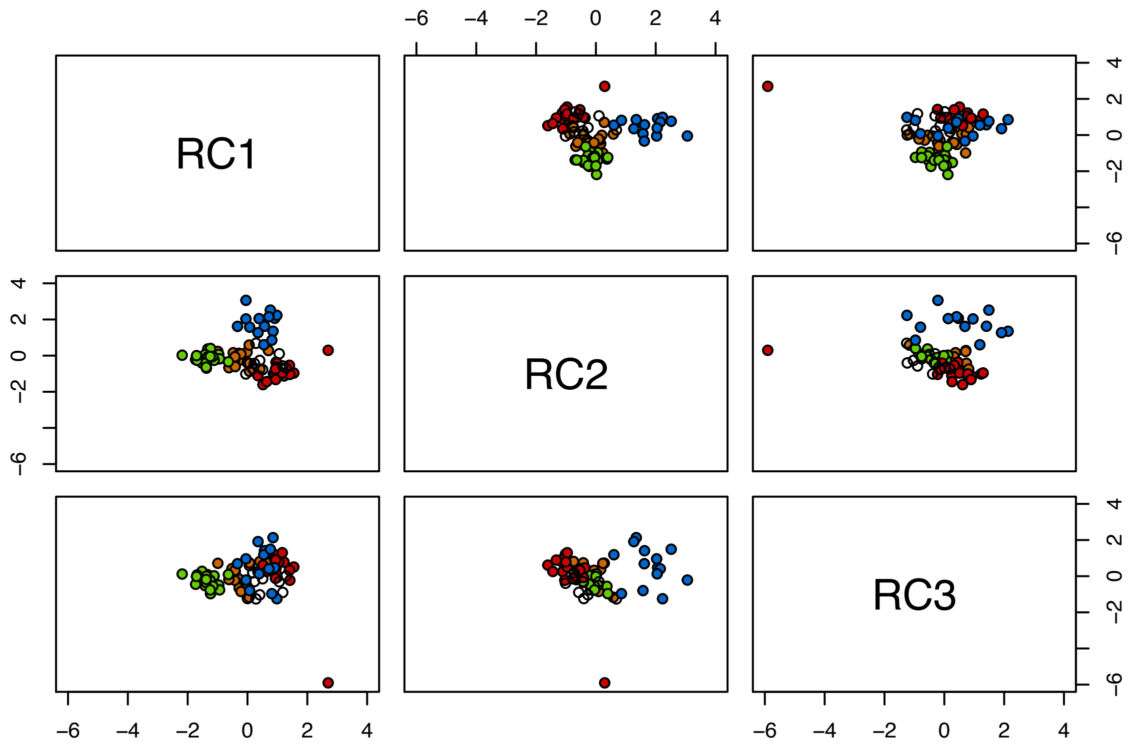
```r
vss(df[, 2:10]) # See Factor recommendations for a simple structure
```



```
##
## Very Simple Structure
## Call: vss(x = df[, 2:10])
## VSS complexity 1 achieves a maximimum of 0.98  with  1  factors
## VSS complexity 2 achieves a maximimum of 0.99  with  2  factors
##
## The Velicer MAP achieves a minimum of 0.08  with  2  factors
## BIC achieves a minimum of  NA  with  2  factors
## Sample Size adjusted BIC achieves a minimum of  NA  with  4  factors
##
## Statistics by number of factors
##   vss1 vss2   map dof   chisq    prob sqresid  fit RMSEA BIC  SABIC complex
## 1 0.98 0.00 0.094  27 1.4e+02 2.6e-17    0.98 0.98 0.233  23 108.37     1.0
## 2 0.51 0.99 0.080  19 4.4e+01 8.6e-04    0.51 0.99 0.131 -38  21.67     1.7
## 3 0.44 0.85 0.121  12 1.5e+01 2.2e-01    0.38 0.99 0.059 -37   1.12     2.1
## 4 0.45 0.92 0.165   6 4.1e+00 6.6e-01    0.23 1.00 0.000 -22  -3.05     2.0
## 5 0.45 0.91 0.174   1 3.0e-01 5.8e-01    0.16 1.00 0.000  -4  -0.89     2.0
## 6 0.44 0.84 0.288  -3 4.7e-08      NA    0.13 1.00    NA  NA     NA     2.2
## 7 0.44 0.89 0.550  -6 1.4e-06      NA    0.28 0.99    NA  NA     NA     2.2
## 8 0.44 0.89 1.000  -8 0.0e+00      NA    0.28 0.99    NA  NA     NA     2.2
##    eChisq    SRMR  eCRMS   eBIC
## 1 1.7e+01 5.5e-02 0.0635 -100.5
## 2 2.1e+00 2.0e-02 0.0269  -80.4
## 3 4.1e-01 8.6e-03 0.0150  -51.7
## 4 9.5e-02 4.1e-03 0.0101  -26.0
## 5 5.3e-03 9.8e-04 0.0059   -4.3
## 6 8.6e-10 3.9e-07     NA     NA
## 7 1.4e-08 1.6e-06     NA     NA
## 8 9.9e-18 4.2e-11     NA     NA
```

```
clr <- character(nrow(df))
clr[] <- "black"
clr[df$CanineGroup == "ModernDog"] <- "#CC0000"
clr[df$CanineGroup == "GoldenJackal"] <- "#66CC00"
clr[df$CanineGroup == "Cuons"] <- "#0066CC"
clr[df$CanineGroup == "ThaiDogs"] <- "#CC6600"
clr[df$CanineGroup == "IndianWolves"] <- "#6600CC"

pairs(fit.pc$scores[, 1:3],
      ylim = c(-6, 4),
      xlim = c(-6, 4),
      pch =21,
      bg=c('#CC0000','#66CC00','#0066CC','#CC6600')
      [unclass(df$CanineGroup)])
```



We can see ther relationship between species and the factors using these plots. From fa.parallel() function the number of factors recommended are 1 the same number of factors can be verified from the VSS diagram which tells us that there is only 1 factor which helps in identifying the skull measurements. The variables X4, X2, X9 and X6 contribute to factor 1 i.e. RC1

6. Carry out a discriminant function analysis to see how well it is possible to separate the groups using the measurements.

Lets cut the data into two parts 75 % and 25%

```
sample_size <- floor(0.75 * nrow(df))
```

```
train <- sample(nrow(df), size = sample_size)
```

```
train.df <- as.data.frame(df[train, ])
```

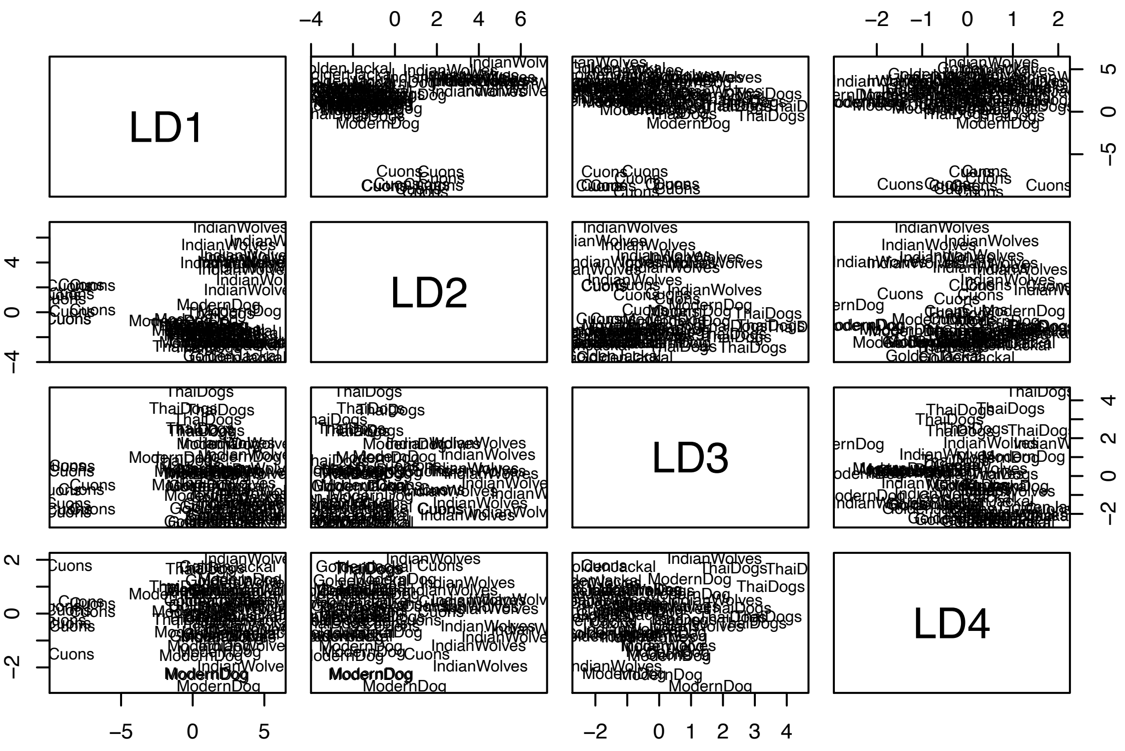
```
test.df <- as.data.frame(df[-train, ])
```

We now have a training and a test set. Training is 75% and test is 25%

```
lda <- lda(formula = train.df$CanineGroup ~
             X1 + X2 + X3 + X4 + X5 + X6 + X7 + X8 + X9,
           data = train.df)
summary(lda)
```

```
##         Length Class  Mode
## prior    5     -none- numeric
## counts   5     -none- numeric
## means   45     -none- numeric
## scaling 36     -none- numeric
## lev      5     -none- character
## svd      4     -none- numeric
## N        1     -none- numeric
## call     3     -none- call
## terms    3     terms  call
## xlevels  0     -none- list
```

```
plot(lda)
```

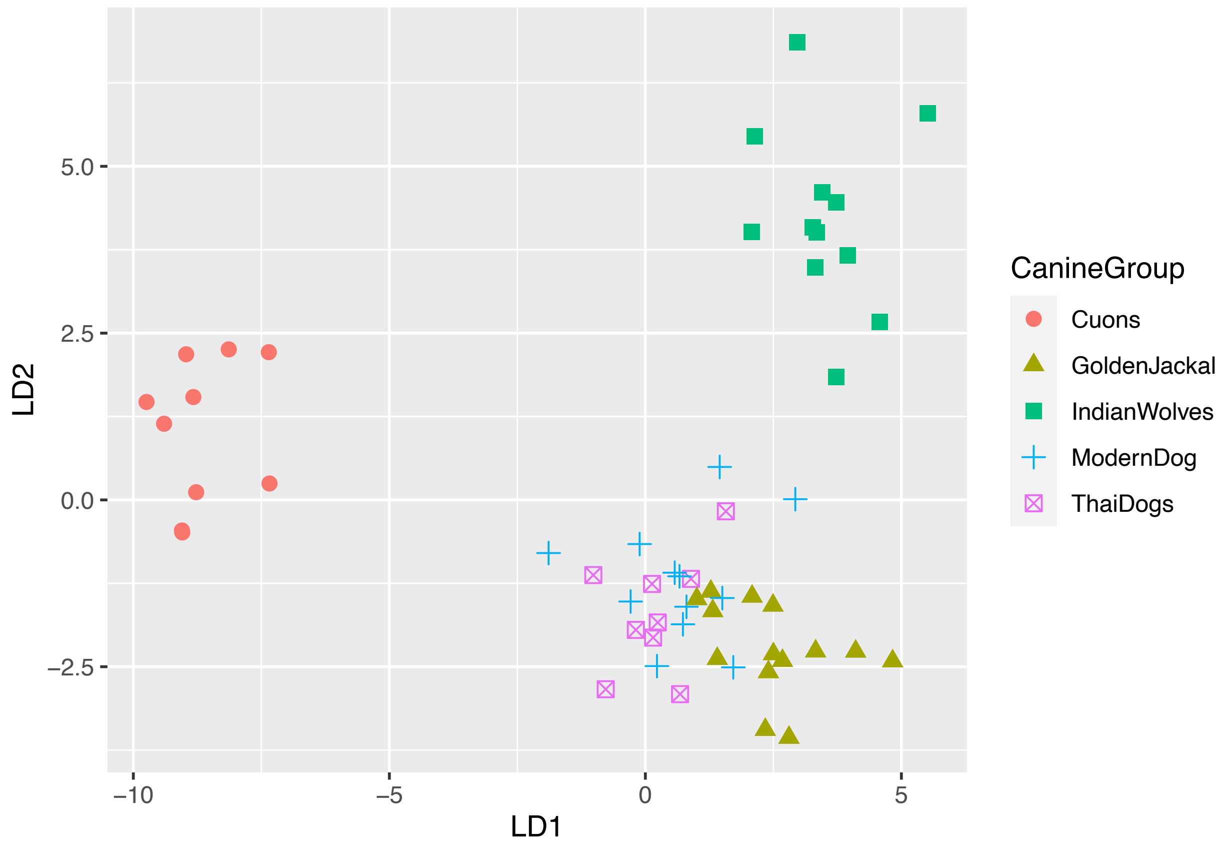


```r
lda.predict <- predict(lda, newdata = test.df)
```

The above prediction has three items called class, posterior and x

```r
lda.data <- cbind(train.df, predict(lda)$x)
```

```r
ggplot(lda.data) + geom_point(aes(LD1,
                                  LD2,
                                  colour = CanineGroup,
                                  shape = CanineGroup),
                              size = 2.5)
```



```
mean(lda.predict$class == test.df$CanineGroup)
```

```
## [1] 0.85
```

7. investigate each canine group separately to see whether logistic regression shows a significant difference between males and females for the measurements. Note that in view of the small sample sizes available for each group, it is unreasonable to expect to fit a logistic function involving all nine variables with good estimates of parameters. Therefore, consideration should be given to fitting functions using only a subset of the variables.

For Modern Dog

```
xtabs(~ CanineGroup + Gender, df[1:16, ])
```

```
##                Gender
## CanineGroup     Female Male Unknown
##   Cuons              0    0       0
##   GoldenJackal       0    0       0
##   IndianWolves       0    0       0
##   ModernDog          8    8       0
##   ThaiDogs           0    0       0
```

```
logistic <- glm(Gender ~ X2 + X5 + X7 + X9,
                data = df[1:16, ],
                family = "binomial")
```

```
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
```

```
summary(logistic)
```

```
##
## Call:
## glm(formula = Gender ~ X2 + X5 + X7 + X9, family = "binomial",
##     data = df[1:16, ])
##
## Deviance Residuals:
##      Min        1Q    Median        3Q       Max
## -1.66110  -0.18145  -0.00007   0.52009   1.55767
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept) -248.690    197.870  -1.257    0.209
## X2             5.860      5.392   1.087    0.277
## X5            -5.682      5.839  -0.973    0.331
## X7            10.570      8.598   1.229    0.219
## X9            -6.340      5.225  -1.213    0.225
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 22.181  on 15  degrees of freedom
## Residual deviance: 10.377  on 11  degrees of freedom
## AIC: 20.377
##
## Number of Fisher Scoring iterations: 9
```

```
confusion_matrix(logistic)
```

```
##                 Predicted Female Predicted Male Total
## Actual Female                  7              1     8
## Actual Male                    1              7     8
## Total                          8              8    16
```

For Golden Jackals

```
xtabs(~ CanineGroup + Gender, data=df[17:36, ])
```

```
##                 Gender
## CanineGroup      Female Male Unknown
##   Cuons               0    0       0
##   GoldenJackal       10   10       0
##   IndianWolves        0    0       0
##   ModernDog           0    0       0
##   ThaiDogs            0    0       0
```

```
logistic <- glm(Gender ~ X2 + X5 + X7 + X9,
                data = df[17:36, ],
                family = "binomial")
summary(logistic)
```

```
##
## Call:
## glm(formula = Gender ~ X2 + X5 + X7 + X9, family = "binomial",
##     data = df[17:36, ])
##
## Deviance Residuals:
##      Min        1Q    Median        3Q       Max
## -2.35880  -0.26570  -0.00231   0.41771   1.04750
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept) -94.7912    46.7563  -2.027   0.0426 *
## X2            2.3168     3.3165   0.699   0.4848
## X5            1.3503     1.4360   0.940   0.3470
## X7            0.8786     0.9973   0.881   0.3783
## X9            5.0866     3.9038   1.303   0.1926
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 27.726  on 19  degrees of freedom
## Residual deviance: 10.307  on 15  degrees of freedom
## AIC: 20.307
##
## Number of Fisher Scoring iterations: 6
```

```
confusion_matrix(logistic)
```

```
##                Predicted Female Predicted Male Total
## Actual Female                 9              1    10
## Actual Male                   0             10    10
## Total                         9             11    20
```

For Cuons

```
xtabs(~ CanineGroup + Gender, data = df[37:53, ])
```

```
##                Gender
## CanineGroup     Female Male Unknown
##   Cuons              8    9       0
##   GoldenJackal       0    0       0
##   IndianWolves       0    0       0
##   ModernDog          0    0       0
##   ThaiDogs           0    0       0
```

```
logistic <- glm(Gender ~ X2 + X5 + X7 + X9,
                data = df[37:53, ],
                family = "binomial")
summary(logistic)
```

```
##
```

```
## Call:
## glm(formula = Gender ~ X2 + X5 + X7 + X9, family = "binomial",
##     data = df[37:53, ])
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.6647  -1.0491   0.7121   1.0022   1.2719
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept)  -3.3022    12.7101  -0.260    0.795
## X2           -0.6355     1.4351  -0.443    0.658
## X5           -0.9823     1.1581  -0.848    0.396
## X7            0.6569     1.2160   0.540    0.589
## X9            1.8545     2.3458   0.791    0.429
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 23.508  on 16  degrees of freedom
## Residual deviance: 20.665  on 12  degrees of freedom
## AIC: 30.665
##
## Number of Fisher Scoring iterations: 4
```

```
confusion_matrix(logistic)
```

```
##               Predicted Female Predicted Male Total
## Actual Female                4              4     8
## Actual Male                  3              6     9
## Total                        7             10    17
```

For Thai Dogs we cannot find and predict the gender since it is unknown in the dataset

For Indian Wolves

```
xtabs(~ CanineGroup + Gender, data = df[64:77, ])
```

```
##               Gender
## CanineGroup    Female Male Unknown
##   Cuons             0    0       0
##   GoldenJackal      0    0       0
##   IndianWolves      6    8       0
##   ModernDog         0    0       0
##   ThaiDogs          0    0       0
```

```
logistic <- glm(Gender ~ X2 + X5 + X7 + X9,
                data = df[64:77, ],
                family = "binomial")
```

```
## Warning: glm.fit: algorithm did not converge
```

```
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
```

```
summary(logistic)
```

```
##
## Call:
## glm(formula = Gender ~ X2 + X5 + X7 + X9, family = "binomial",
##     data = df[64:77, ])
##
## Deviance Residuals:
##        Min          1Q      Median          3Q         Max
## -1.679e-05  -2.110e-08   2.110e-08   2.110e-08   1.991e-05
##
## Coefficients:
##               Estimate Std. Error z value Pr(>|z|)
## (Intercept) -1.526e+03  1.212e+07       0        1
## X2           7.368e+01  6.098e+05       0        1
## X5           9.270e+01  1.093e+06       0        1
## X7          -3.977e+01  7.878e+05       0        1
## X9           1.933e+00  1.211e+05       0        1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 1.9121e+01  on 13  degrees of freedom
## Residual deviance: 9.2083e-10  on  9  degrees of freedom
## AIC: 10
##
## Number of Fisher Scoring iterations: 25
```

```
confusion_matrix(logistic)
```

```
##               Predicted Female Predicted Male Total
## Actual Female                6              0     6
## Actual Male                  0              8     8
## Total                        6              8    14
```

8. Show ROC containing both your discriminant and logistic function for gender classification for the Prehistoric Thai Dog

```
xtabs(~ CanineGroup + Gender, data=df[54:63, ])
```

```
##               Gender
## CanineGroup     Female Male Unknown
##   Cuons              0    0       0
##   GoldenJackal       0    0       0
##   IndianWolves       0    0       0
##   ModernDog          0    0       0
##   ThaiDogs           0    0      10
```

Randomly assigning the values of Male and Female which is 50% Female and 50% Male

```r
df$Gender[54:63] <- c("Female",
                      "Male",
                      "Male",
                      "Female",
                      "Female",
                      "Female",
                      "Male",
                      "Female",
                      "Male",
                      "Male")
```

```r
logistic_simple <- glm(Gender ~., data = df[c(-1)], family = "binomial")
summary(logistic_simple)
```

```
##
## Call:
## glm(formula = Gender ~ ., family = "binomial", data = df[c(-1)])
##
## Deviance Residuals:
##    Min      1Q  Median      3Q     Max
## -1.842  -1.036   0.411   1.050   1.844
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept) -0.96963    2.81347  -0.345   0.7304
## X1           0.03666    0.06954   0.527   0.5981
## X2           0.23846    0.49937   0.478   0.6330
## X3           0.10227    0.10040   1.019   0.3084
## X4          -0.17269    0.17928  -0.963   0.3354
## X5          -0.22164    0.27341  -0.811   0.4176
## X6          -0.65196    0.73024  -0.893   0.3720
## X7           0.07159    0.10588   0.676   0.4989
## X8          -0.14762    0.21479  -0.687   0.4919
## X9           1.38744    0.82018   1.692   0.0907 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 106.628  on 76  degrees of freedom
## Residual deviance:  96.677  on 67  degrees of freedom
## AIC: 116.68
##
## Number of Fisher Scoring iterations: 4
```

```r
pdata <- predict(logistic_simple, newdata = df, type = "response")
pdata
```

```
##         1         2         3         4         5         6         7         8
## 0.5549875 0.4067477 0.4516714 0.6069229 0.6851925 0.7109144 0.4304144 0.7146370
##         9        10        11        12        13        14        15        16
## 0.5032641 0.5028978 0.7250236 0.4094083 0.5401111 0.5653697 0.5221524 0.5640158
##        17        18        19        20        21        22        23        24
## 0.5188810 0.3248237 0.3205044 0.3938034 0.2959413 0.4753977 0.1825277 0.5729241
##        25        26        27        28        29        30        31        32
## 0.4067892 0.4155722 0.3466740 0.3062581 0.3372643 0.4704012 0.4828238 0.3673306
##        33        34        35        36        37        38        39        40
## 0.3307675 0.3056470 0.3045916 0.2799278 0.5099554 0.7830929 0.7251121 0.6586111
##        41        42        43        44        45        46        47        48
## 0.5764570 0.6187301 0.5476574 0.5842145 0.6186942 0.5397957 0.6831842 0.4922782
##        49        50        51        52        53        54        55        56
## 0.1643165 0.3379583 0.4152510 0.4033627 0.2671415 0.5838307 0.3439986 0.7544320
##        57        58        59        60        61        62        63        64
## 0.4040157 0.5712119 0.3429020 0.5413810 0.5696953 0.6376788 0.4328645 0.6393246
##        65        66        67        68        69        70        71        72
## 0.7712961 0.8331748 0.7224211 0.6925156 0.7347038 0.9189919 0.9728126 0.3885048
##        73        74        75        76        77
## 0.5741460 0.6910565 0.1900079 0.8166125 0.6140271
```

```
pdataF <- as.factor(ifelse(test = as.numeric(pdata > 0.5) == 1,
                           yes = "Female", no = "Male" ))
pdataF
```

```
##  [1] Female Male   Male   Female Female Female Male   Female Female Female
## [11] Female Male   Female Female Female Female Female Male   Male   Male
## [21] Male   Male   Male   Female Male   Male   Male   Male   Male   Male
## [31] Male   Male   Male   Male   Male   Male   Female Female Female Female
## [41] Female Female Female Female Female Female Female Male   Male   Male
## [51] Male   Male   Male   Female Male   Female Male   Female Male   Female
## [61] Female Female Male   Female Female Female Female Female Female Female
## [71] Female Male   Female Female Male   Female Female
## Levels: Female Male
```

Here, the actual values will be displayed for the earlier classified Male and Females

```
confusionMatrix(pdataF, as.factor(df$Gender))
```

```
## Warning in levels(reference) != levels(data): longer object length is not a
## multiple of shorter object length
```

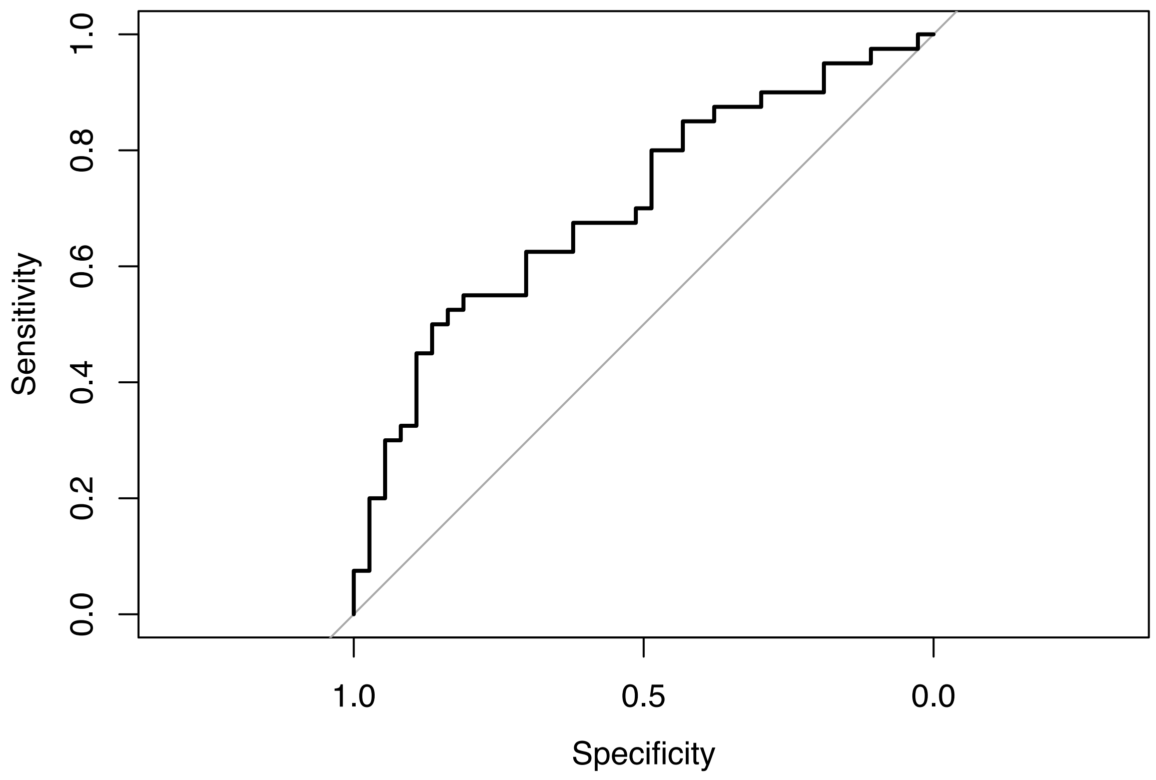
```
## Warning in confusionMatrix.default(pdataF, as.factor(df$Gender)): Levels are not
## in the same order for reference and data. Refactoring data to match.
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction Female Male Unknown
##    Female      16   27       0
##    Male        21   13       0
##    Unknown      0    0       0
##
## Overall Statistics
##
##                Accuracy : 0.3766
##                   95% CI : (0.2687, 0.4944)
##     No Information Rate : 0.5195
##     P-Value [Acc > NIR] : 0.9957
##
##                    Kappa : -0.2411
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                         Class: Female Class: Male Class: Unknown
## Sensitivity                    0.4324      0.3250             NA
## Specificity                    0.3250      0.4324              1
## Pos Pred Value                 0.3721      0.3824             NA
## Neg Pred Value                 0.3824      0.3721             NA
## Prevalence                     0.4805      0.5195              0
## Detection Rate                 0.2078      0.1688              0
## Detection Prevalence           0.5584      0.4416              0
## Balanced Accuracy              0.3787      0.3787             NA
```

```r
roc(df$Gender, logistic_simple$fitted.values, plot = TRUE)
```

```
## Setting levels: control = Female, case = Male
```

```
## Setting direction: controls < cases
```

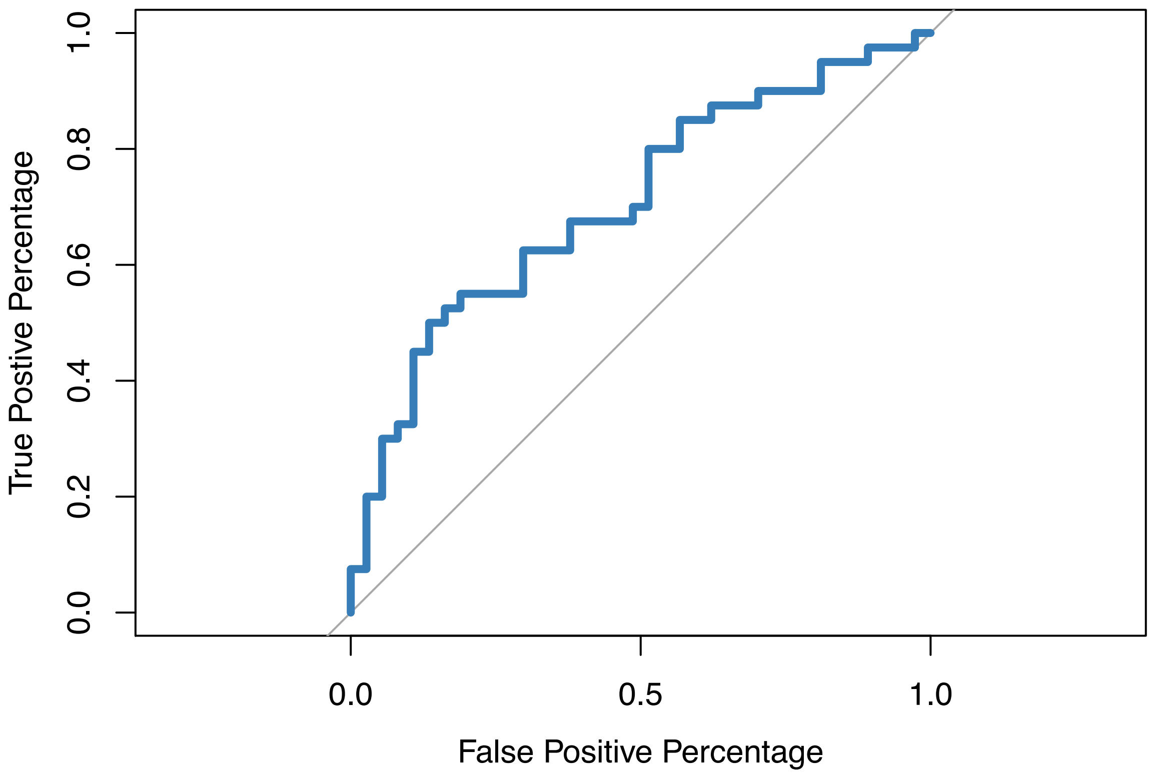


```
##
## Call:
## roc.default(response = df$Gender, predictor = logistic_simple$fitted.values,     plot = TRUE)
##
## Data: logistic_simple$fitted.values in 37 controls (df$Gender Female) < 40 cases (df$Gender Male).
## Area under the curve: 0.7068
```

```r
roc(df$Gender,
    logistic_simple$fitted.values,
    plot = TRUE,
    legacy.axes=TRUE,
    xlab="False Positive Percentage",
    ylab="True Postive Percentage",
    col="#377eb8",
    lwd = 4)
```

```
## Setting levels: control = Female, case = Male

## Setting direction: controls < cases
```



```
##
## Call:
## roc.default(response = df$Gender, predictor = logistic_simple$fitted.values,     plot = TRUE, legacy
##
## Data: logistic_simple$fitted.values in 37 controls (df$Gender Female) < 40 cases (df$Gender Male).
## Area under the curve: 0.7068
```

```r
roc.info <- roc(df$Gender, logistic_simple$fitted.values, legacy.axes = TRUE)
```

```
## Setting levels: control = Female, case = Male
```

```
## Setting direction: controls < cases
```

```r
str(roc.info)
```

```
## List of 15
##  $ percent           : logi FALSE
##  $ sensitivities     : num [1:78] 1 1 0.975 0.975 0.975 0.975 0.95 0.95 0.95 0.95 ...
##  $ specificities     : num [1:78] 0 0.027 0.027 0.0541 0.0811 ...
##  $ thresholds        : num [1:78] -Inf 0.173 0.186 0.229 0.274 ...
##  $ direction         : chr "<"
##  $ cases             : Named num [1:40] 0.555 0.407 0.452 0.607 0.685 ...
##   ..- attr(*, "names")= chr [1:40] "1" "2" "3" "4" ...
##  $ controls          : Named num [1:37] 0.503 0.503 0.725 0.409 0.54 ...
##   ..- attr(*, "names")= chr [1:37] "9" "10" "11" "12" ...
##  $ fun.sesp          :function (thresholds, controls, cases, direction)
##  $ auc               : 'auc' num 0.707
##   ..- attr(*, "partial.auc")= logi FALSE
##   ..- attr(*, "percent")= logi FALSE
##   ..- attr(*, "roc")=List of 15
##   .. ..$ percent           : logi FALSE
##   .. ..$ sensitivities     : num [1:78] 1 1 0.975 0.975 0.975 0.975 0.95 0.95 0.95 0.95 ...
##   .. ..$ specificities     : num [1:78] 0 0.027 0.027 0.0541 0.0811 ...
##   .. ..$ thresholds        : num [1:78] -Inf 0.173 0.186 0.229 0.274 ...
##   .. ..$ direction         : chr "<"
##   .. ..$ cases             : Named num [1:40] 0.555 0.407 0.452 0.607 0.685 ...
##   .. .. ..- attr(*, "names")= chr [1:40] "1" "2" "3" "4" ...
##   .. ..$ controls          : Named num [1:37] 0.503 0.503 0.725 0.409 0.54 ...
##   .. .. ..- attr(*, "names")= chr [1:37] "9" "10" "11" "12" ...
##   .. ..$ fun.sesp          :function (thresholds, controls, cases, direction)
##   .. ..$ auc               : 'auc' num 0.707
##   .. .. ..- attr(*, "partial.auc")= logi FALSE
##   .. .. ..- attr(*, "percent")= logi FALSE
##   .. .. ..- attr(*, "roc")=List of 8
##   .. .. .. ..$ percent      : logi FALSE
##   .. .. .. ..$ sensitivities: num [1:78] 1 1 0.975 0.975 0.975 0.975 0.95 0.95 0.95 0.95 ...
##   .. .. .. ..$ specificities: num [1:78] 0 0.027 0.027 0.0541 0.0811 ...
##   .. .. .. ..$ thresholds   : num [1:78] -Inf 0.173 0.186 0.229 0.274 ...
##   .. .. .. ..$ direction    : chr "<"
##   .. .. .. ..$ cases        : Named num [1:40] 0.555 0.407 0.452 0.607 0.685 ...
##   .. .. .. .. ..- attr(*, "names")= chr [1:40] "1" "2" "3" "4" ...
##   .. .. .. ..$ controls     : Named num [1:37] 0.503 0.503 0.725 0.409 0.54 ...
##   .. .. .. .. ..- attr(*, "names")= chr [1:37] "9" "10" "11" "12" ...
##   .. .. .. ..$ fun.sesp     :function (thresholds, controls, cases, direction)
##   .. .. .. ..- attr(*, "class")= chr "roc"
##   .. ..$ call              : language roc.default(response = df$Gender, predictor = logistic_simple$
##   .. ..$ original.predictor: Named num [1:77] 0.555 0.407 0.452 0.607 0.685 ...
##   .. .. ..- attr(*, "names")= chr [1:77] "1" "2" "3" "4" ...
##   .. ..$ original.response : Factor w/ 3 levels "Female","Male",..: 2 2 2 2 2 2 2 2 1 1 ...
##   .. ..$ predictor          : Named num [1:77] 0.555 0.407 0.452 0.607 0.685 ...
##   .. .. ..- attr(*, "names")= chr [1:77] "1" "2" "3" "4" ...
##   .. ..$ response           : Factor w/ 3 levels "Female","Male",..: 2 2 2 2 2 2 2 2 1 1 ...
##   .. ..$ levels             : chr [1:2] "Female" "Male"
##   .. ..- attr(*, "class")= chr "roc"
##  $ call                : language roc.default(response = df$Gender, predictor = logistic_simple$fitted
##  $ original.predictor: Named num [1:77] 0.555 0.407 0.452 0.607 0.685 ...
##   ..- attr(*, "names")= chr [1:77] "1" "2" "3" "4" ...
##  $ original.response : Factor w/ 3 levels "Female","Male",..: 2 2 2 2 2 2 2 2 1 1 ...
##  $ predictor         : Named num [1:77] 0.555 0.407 0.452 0.607 0.685 ...
##   ..- attr(*, "names")= chr [1:77] "1" "2" "3" "4" ...
##  $ response          : Factor w/ 3 levels "Female","Male",..: 2 2 2 2 2 2 2 2 1 1 ...
##  $ levels            : chr [1:2] "Female" "Male"
##  - attr(*, "class")= chr "roc"
```

```
roc.df <- data.frame(tpp = roc.info$sensitivities * 100,
                     fpp = (1 - roc.info$specificities) * 100,
                     thresholds = roc.info$thresholds)
```

```
head(roc.df)
```

```
##     tpp       fpp thresholds
## 1 100.0 100.00000       -Inf
## 2 100.0  97.29730  0.1734221
## 3  97.5  97.29730  0.1862678
## 4  97.5  94.59459  0.2285747
## 5  97.5  91.89189  0.2735347
## 6  97.5  89.18919  0.2879346
```

```
roc.df[roc.df$tpp > 60 & roc.df$tpp < 80, ]
```

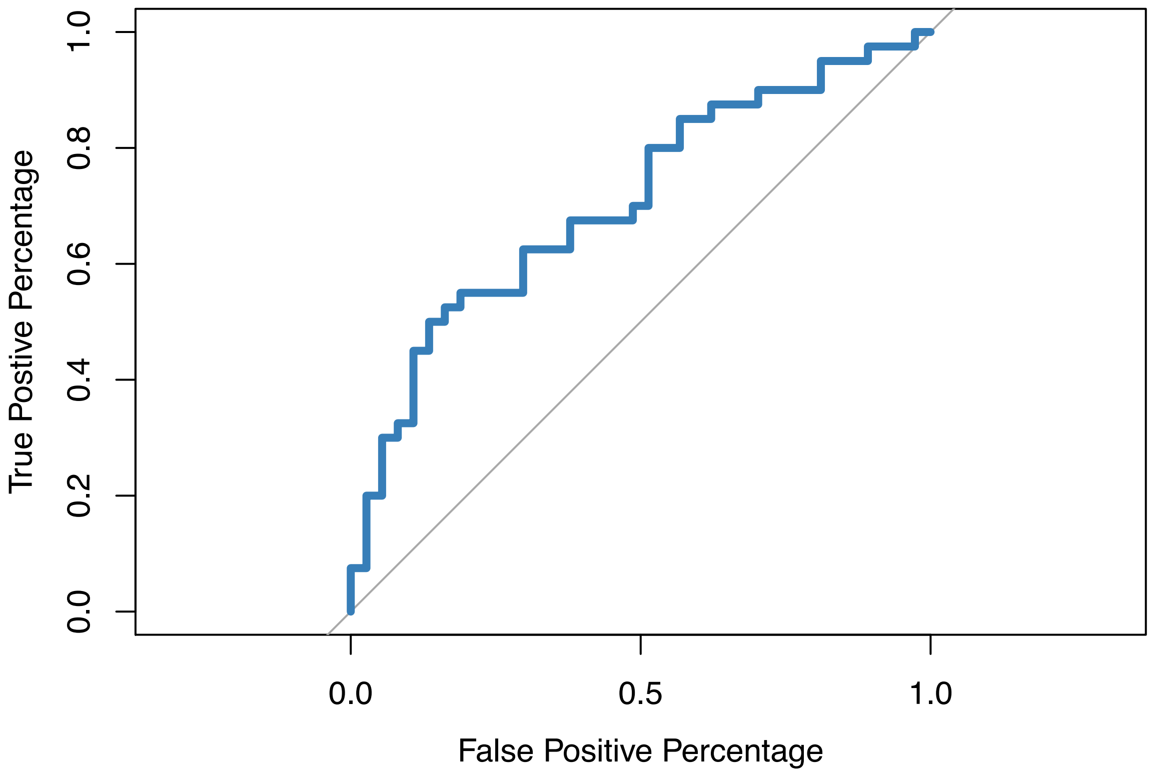
```
##     tpp      fpp thresholds
## 28 77.5 51.35135  0.4229933
## 29 75.0 51.35135  0.4316394
## 30 72.5 51.35135  0.4422679
## 31 70.0 51.35135  0.4610363
## 32 70.0 48.64865  0.4728995
## 33 67.5 48.64865  0.4791108
## 34 67.5 45.94595  0.4875510
## 35 67.5 43.24324  0.4975880
## 36 67.5 40.54054  0.5030810
## 37 67.5 37.83784  0.5066097
## 38 65.0 37.83784  0.5144182
## 39 62.5 37.83784  0.5205167
## 40 62.5 35.13514  0.5309741
## 41 62.5 32.43243  0.5399534
## 42 62.5 29.72973  0.5407460
```

```
roc(df$Gender,
    logistic_simple$fitted.values,
    plot = TRUE,
    legacy.axes = TRUE,
    xlab = "False Positive Percentage",
    ylab = "True Postive Percentage",
    col = "#377eb8",
    lwd = 4)
```

```
## Warning in roc.default(df$Gender, logistic_simple$fitted.values, plot = TRUE, :
## 'response' has more than two levels. Consider setting 'levels' explicitly or
## using 'multiclass.roc' instead

## Setting levels: control = Female, case = Male

## Setting direction: controls < cases
```



```
##
## Call:
## roc.default(response = df$Gender, predictor = logistic_simple$fitted.values,     plot = TRUE, legacy
##
## Data: logistic_simple$fitted.values in 37 controls (df$Gender Female) < 40 cases (df$Gender Male).
## Area under the curve: 0.7068
```

```
par(pty = "s")
```

Conclusion: The Area under the curve is 70% so that means When predicting the gender of thai dogs we are correct 70% of the times

9. Predict the Gender for the Prehistoric Thai Dog

a. Explain the reason for choosing the MVA technique for prediction

b. What is the Hit Ratio (Accuracy) of your classification technique?

Let's use Logistic Regression: The gender for the dog will have only two distinct class i.e. either Male or Female, so its very obvious and apt if I use Logistic regression.

```
df_not_thai = df[-c(54:63), -1]
```

```
df_not_thai$Gender = as.factor(df_not_thai$Gender)
```

```
logistic_simple = glm(Gender~., data = df_not_thai, family = "binomial")
summary(logistic_simple)
```

```
##
## Call:
## glm(formula = Gender ~ ., family = "binomial", data = df_not_thai)
##
## 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
```

Encoded dependant variable

```
df_not_thai_encoded = df_not_thai
```

```
setDT(df_not_thai_encoded)
```

```r
df_not_thai_encoded[Gender == 'Male', 'Gender_flag' := 0]
df_not_thai_encoded[Gender == 'Female', 'Gender_flag' := 1]
str(df_not_thai_encoded)
```

```
## Classes 'data.table' and 'data.frame':   67 obs. of  11 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 ...
##  $ Gender_flag: num  0 0 0 0 0 0 0 0 1 1 ...
##  - attr(*, ".internal.selfref")=<externalptr>
```

```r
df_not_thai_encoded_logistic = df_not_thai_encoded[, -c('Gender')]

logistic_simple_encoded = glm(Gender_flag~.,
                              data = df_not_thai_encoded_logistic,
                              family = "binomial")
summary(logistic_simple_encoded)
```

```
##
## Call:
## glm(formula = Gender_flag ~ ., family = "binomial", data = df_not_thai_encoded_logistic)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.7916  -1.0086  -0.3837   1.1021   1.9718
##
## 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
```

```r
confusion_matrix(logistic_simple)
```

```
##                Predicted Female Predicted Male Total
## Actual Female                23              9    32
## Actual Male                  12             23    35
## Total                        35             32    67
```

```r
confusion_matrix(logistic_simple_encoded)
```

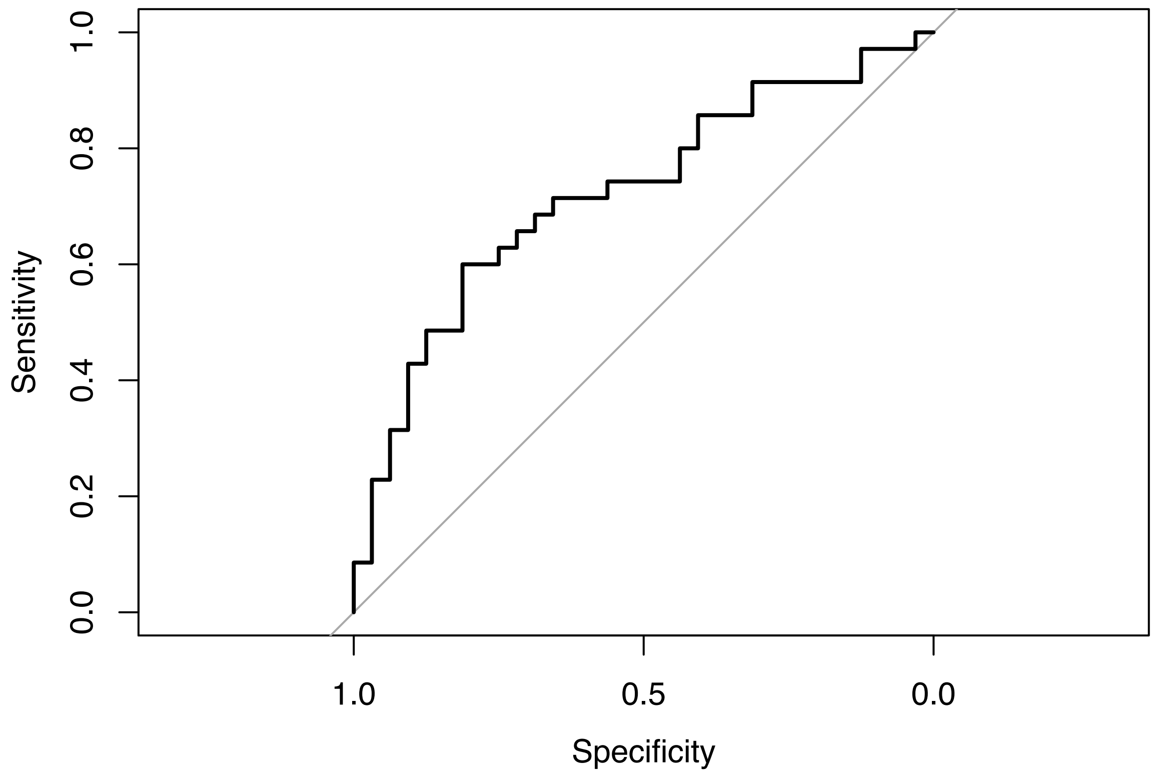
```
##           Predicted 0 Predicted 1 Total
## Actual 0           23          12    35
## Actual 1            9          23    32
## Total              32          35    67
```

```r
roc(df_not_thai$Gender,
    logistic_simple$fitted.values,
    plot = TRUE)
```

```
## Warning in roc.default(df_not_thai$Gender, logistic_simple$fitted.values, :
## 'response' has more than two levels. Consider setting 'levels' explicitly or
## using 'multiclass.roc' instead

## Setting levels: control = Female, case = Male

## Setting direction: controls < cases
```



```
##
## Call:
## roc.default(response = df_not_thai$Gender, predictor = logistic_simple$fitted.values,     plot = TRUI
##
## Data: logistic_simple$fitted.values in 32 controls (df_not_thai$Gender Female) < 35 cases (df_not_tha
## Area under the curve: 0.7214
```

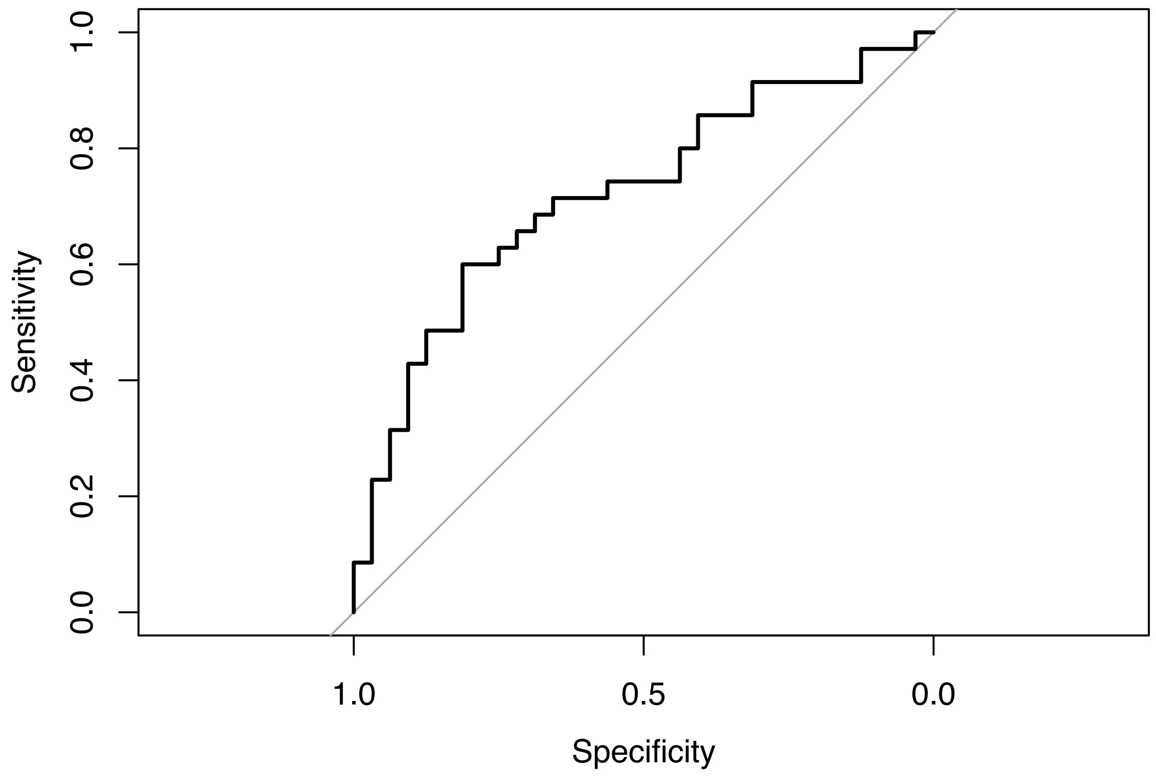
```
par(pty = "s")
```

```
roc(df_not_thai$Gender,
    logistic_simple$fitted.values,
    plot = TRUE)
```

```
## Warning in roc.default(df_not_thai$Gender, logistic_simple$fitted.values, :
## 'response' has more than two levels. Consider setting 'levels' explicitly or
## using 'multiclass.roc' instead
```

```
## Setting levels: control = Female, case = Male
```

```
## Setting direction: controls < cases
```



```
##
## Call:
```

```
## roc.default(response = df_not_thai$Gender, predictor = logistic_simple$fitted.values,     plot = TRUI
##
## Data: logistic_simple$fitted.values in 32 controls (df_not_thai$Gender Female) < 35 cases (df_not_tha
## Area under the curve: 0.7214
```

Conclusion: From the ROC curve we can conclude with a good AUC of 72.14%

10. Create a model to predict length of the Mandible length for Prehistoric Thai Dog.

a. What is the accuracy of your model

```
ThaiDog<-df[c(54:63), ]
```

X1 is the madible length

```
head(ThaiDog)
```

```
## # A tibble: 6 x 11
##   CanineGroup    X1    X2    X3    X4    X5    X6    X7    X8    X9 Gender
##   <fct>       <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <fct>
## 1 ThaiDogs      112  10.1    17    18    19   7.7    31    33   5.8 Female
## 2 ThaiDogs      115  10      18    23    20   7.8    33    36   6   Male
## 3 ThaiDogs      136  11.9    22    25    21   8.5    36    39   7   Male
## 4 ThaiDogs      111   9.9    19    20    18   7.3    29    34   5.3 Female
## 5 ThaiDogs      130  11.2    23    27    20   9.1    35    35   6.6 Female
## 6 ThaiDogs      125  10.7    19    26    20   8.4    33    37   6.3 Female
```

```
fit<-lm(X1~., data = ThaiDog[c(-1, -11)])
summary(fit)
```

```
##
## Call:
## lm(formula = X1 ~ ., data = ThaiDog[c(-1, -11)])
##
## Residuals:
##        1        2        3        4        5        6        7        8
##  0.07584 -0.21856 -0.25971 -0.54318 -0.16928 -0.04839  0.08665  1.13774
##        9       10
## -0.41248  0.35137
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) -59.9544    34.2225  -1.752    0.330
## X2           -6.9316     3.6587  -1.895    0.309
## X3            4.0239     1.5351   2.621    0.232
## X4           -1.0622     0.6274  -1.693    0.340
## X5           10.1737     5.3594   1.898    0.309
## X6            8.2435     2.7720   2.974    0.207
## X7           -8.2872     3.3399  -2.481    0.244
## X8            1.5138     0.4795   3.157    0.195
## X9           24.6157     7.8538   3.134    0.197
##
## Residual standard error: 1.429 on 1 degrees of freedom
## Multiple R-squared:  0.9968, Adjusted R-squared:  0.9712
## F-statistic: 38.89 on 8 and 1 DF,  p-value: 0.1234
```

Predicted Values for the variable X1

```
fitted(fit)
```

```
##        1        2        3        4        5        6        7        8
## 111.9242 115.2186 136.2597 111.5432 130.1693 125.0484 131.9133 119.8623
##        9       10
## 122.4125 123.6486
```

```
residuals(fit)
```

```
##           1           2           3           4           5           6
##  0.07584371 -0.21856152 -0.25971481 -0.54317831 -0.16928212 -0.04838838
##           7           8           9          10
##  0.08665477  1.13773883 -0.41248334  0.35137118
```

Anova Table

```
anova(fit) # sum of squares from diff variables
```

```
## Analysis of Variance Table
##
## Response: X1
##           Df  Sum Sq Mean Sq F value  Pr(>F)
## X2         1 168.189 168.189 82.3414 0.06988 .
## X3         1 113.895 113.895 55.7604 0.08475 .
## X4         1  11.908  11.908  5.8297 0.24997
## X5         1 168.423 168.423 82.4563 0.06983 .
## X6         1  90.833  90.833 44.4697 0.09476 .
## X7         1   9.957   9.957  4.8748 0.27074
## X8         1  52.288  52.288 25.5990 0.12422
## X9         1  20.065  20.065  9.8234 0.19662
## Residuals  1   2.043   2.043
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
gvmodel <- gvlma(fit)
summary(gvmodel)
```

```
##
## Call:
## lm(formula = X1 ~ ., data = ThaiDog[c(-1, -11)])
##
## Residuals:
##        1        2        3        4        5        6        7        8
##  0.07584 -0.21856 -0.25971 -0.54318 -0.16928 -0.04839  0.08665  1.13774
##        9       10
## -0.41248  0.35137
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) -59.9544    34.2225  -1.752    0.330
## X2           -6.9316     3.6587  -1.895    0.309
## X3            4.0239     1.5351   2.621    0.232
## X4           -1.0622     0.6274  -1.693    0.340
## X5           10.1737     5.3594   1.898    0.309
## X6            8.2435     2.7720   2.974    0.207
## X7           -8.2872     3.3399  -2.481    0.244
## X8            1.5138     0.4795   3.157    0.195
## X9           24.6157     7.8538   3.134    0.197
##
## Residual standard error: 1.429 on 1 degrees of freedom
## Multiple R-squared:  0.9968, Adjusted R-squared:  0.9712
## F-statistic: 38.89 on 8 and 1 DF,  p-value: 0.1234
##
##
## ASSESSMENT OF THE LINEAR MODEL ASSUMPTIONS
## USING THE GLOBAL TEST ON 4 DEGREES-OF-FREEDOM:
## Level of Significance =  0.05
##
## Call:
##  gvlma(x = fit)
##
##                      Value  p-value                   Decision
## Global Stat        15.6962 0.003455 Assumptions NOT satisfied!
## Skewness            3.0747 0.079519    Assumptions acceptable.
## Kurtosis            0.7586 0.383763    Assumptions acceptable.
## Link Function      10.0000 0.001565 Assumptions NOT satisfied!
## Heteroscedasticity  1.8628 0.172297    Assumptions acceptable.
```

```r
fit1 <- fit
fit2 <- lm(X1 ~ X4 + X6 + X7 + X9, data = ThaiDog)
fit3 <- lm(X1 ~ X2 + X3 + X5 + X6, data = ThaiDog)
```

```r
# compare models
anova(fit1, fit2, fit3)
```

```
## Analysis of Variance Table
##
## Model 1: X1 ~ X2 + X3 + X4 + X5 + X6 + X7 + X8 + X9
## Model 2: X1 ~ X4 + X6 + X7 + X9
## Model 3: X1 ~ X2 + X3 + X5 + X6
##   Res.Df    RSS Df Sum of Sq      F Pr(>F)
## 1      1  2.043
## 2      5 50.777 -4   -48.735 5.9649 0.2968
## 3      5 88.091  0   -37.314
```

```
step <- stepAIC(fit, direction = "both")# step1 take corelation. step2 take highest corellation
```

```
## Start:  AIC=2.12
## X1 ~ X2 + X3 + X4 + X5 + X6 + X7 + X8 + X9
##
##        Df Sum of Sq     RSS     AIC
## <none>               2.0426  2.1163
## - X4    1    5.8552  7.8977 13.6399
## - X2    1    7.3315  9.3741 15.3536
## - X5    1    7.3604  9.4030 15.3845
## - X7    1   12.5753 14.6179 19.7966
## - X3    1   14.0345 16.0770 20.7481
## - X6    1   18.0639 20.1064 22.9845
## - X9    1   20.0650 22.1075 23.9333
## - X8    1   20.3561 22.3987 24.0642
```

```
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 ~ X2 + X3 + X4 + X5 + X6 + X7 + X8 + X9
##
##
##   Step Df Deviance Resid. Df Resid. Dev     AIC
## 1                          1   2.042577 2.11627
```

```
predict.lm(fit1, data.frame(X2 = 10.1,
                            X3 = 17,
                            X4 = 18,
                            X5 = 19,
                            X6 = 7.7,
                            X7 = 31,
                            X8 = 33,
                            X9 = 5.8))
```

```
##        1
## 111.9242
```

```
predict.lm(fit2, data.frame(X2 = 10.1,
                            X3 = 17,
                            X4 = 18,
                            X5 = 19,
                            X6 = 7.7,
                            X7 = 31,
                            X8 = 33,
                            X9 = 5.8))
```

```
##        1
## 115.5767
```

```r
predict.lm(fit3, data.frame(X2 = 10.1,
                            X3 = 17,
                            X4 = 18,
                            X5 = 19,
                            X6 = 7.7,
                            X7 = 31,
                            X8 = 33,
                            X9 = 5.8))
```

```
##        1
## 112.4628
```

The predicted value for X1 is the best when predicted with fit 1 we cn compare it to the first index since we have tried to predidct the first index which had an actual value of 120