ProteinDataset

The accompanied dataset gives estimates of the average protein consumption (in grams per person per day) from different food sources for the inhabitants of 25 European countries.

Objective:

To-

- Use principal components analysis to investigate the relationships between the countries on the basis of these variables
- Carry out cluster analysis to study relation between countries on their diet
- Identify the important factors underlying the observed variables and examine the relationships between the countries with respect to these factors

Contents:

Note:

Please refer to the Text highlighted in Blue color for my explanation and interpretation.

Including Libraries

```
library(cluster)
```

```
## Warning: package 'cluster' was built under R version 3.6.2
```

```
library(data.table)#Data. table is an extension of data. frame package in R. It is widely used for fast aggregation of large datasets,
```

```
## Warning: package 'data.table' was built under R version 3.6.2
```

```
library(Hmisc)#data analysis funs
```

```
## Warning: package 'Hmisc' was built under R version 3.6.2
```

```
## Loading required package: lattice
```

```
## Warning: package 'lattice' was built under R version 3.6.2
```

```
## Loading required package: survival
```

```
## Warning: package 'survival' was built under R version 3.6.2
```

```
## Loading required package: Formula
```

```
## Loading required package: ggplot2
```

```
##
## Attaching package: 'Hmisc'
```

```
## The following objects are masked from 'package:base':
##
##     format.pval, units
```

```
library(dplyr)
```

```
## Warning: package 'dplyr' was built under R version 3.6.2
```

```
##
## Attaching package: 'dplyr'
```

```
## The following objects are masked from 'package:Hmisc':
##
##     src, summarize
```

```
## The following objects are masked from 'package:data.table':
##
##     between, first, last
```

```
## The following objects are masked from 'package:stats':
##
##     filter, lag
```

```
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
library(tidyverse)
```

```
## Warning: package 'tidyverse' was built under R version 3.6.2
```

```
## -- Attaching packages ---------------------------------------- tidyverse 1.
3.0 --
```

```
## v tibble  2.1.3     v purrr   0.3.3
## v tidyr   1.0.2     v stringr 1.4.0
## v readr   1.3.1     v forcats 0.4.0

## Warning: package 'tidyr' was built under R version 3.6.2

## Warning: package 'purrr' was built under R version 3.6.2

## Warning: package 'stringr' was built under R version 3.6.3

## -- Conflicts ---------------------------------------- tidyverse_conflict
s() --
## x dplyr::between()   masks data.table::between()
## 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 dplyr::src()       masks Hmisc::src()
## x dplyr::summarize() masks Hmisc::summarize()
## x purrr::transpose() masks data.table::transpose()

library(ggplot2)
library(plotly)

## Warning: package 'plotly' was built under R version 3.6.2

##
## Attaching package: 'plotly'

## The following object is masked from 'package:Hmisc':
##
##     subplot

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

## The following object is masked from 'package:stats':
##
##     filter

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

library(GGally)

## Warning: package 'GGally' was built under R version 3.6.2

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

```
##
## Attaching package: 'GGally'

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

library(ggthemes)

## Warning: package 'ggthemes' was built under R version 3.6.2

library(psych)

## Warning: package 'psych' was built under R version 3.6.2

##
## Attaching package: 'psych'

## The following object is masked from 'package:Hmisc':
##
##     describe

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

library(relaimpo)

## Warning: package 'relaimpo' was built under R version 3.6.2

## Loading required package: MASS

##
## Attaching package: 'MASS'

## The following object is masked from 'package:plotly':
##
##     select

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

## Loading required package: boot

## Warning: package 'boot' was built under R version 3.6.2

##
## Attaching package: 'boot'

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

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

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

## Loading required package: survey

## Warning: package 'survey' was built under R version 3.6.2

## Loading required package: grid

## Loading required package: Matrix

##
## Attaching package: 'Matrix'

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

##
## Attaching package: 'survey'

## The following object is masked from 'package:Hmisc':
##
##     deff

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

## Loading required package: mitools

## Warning: package 'mitools' was built under R version 3.6.2

## 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.

library(e1071)

## Warning: package 'e1071' was built under R version 3.6.2

##
## Attaching package: 'e1071'
```

```
## The following object is masked from 'package:Hmisc':
##
##     impute
```

```
library(corrplot)
```

```
## Warning: package 'corrplot' was built under R version 3.6.3
```

```
## corrplot 0.84 loaded
```

```
library(factoextra)
```

```
## Warning: package 'factoextra' was built under R version 3.6.3
```

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

```
library(fpc)
```

```
## Warning: package 'fpc' was built under R version 3.6.3
```

R Markdown

This is an R Markdown document. Markdown is a simple formatting syntax for authoring HTML, PDF, and MS Word documents. For more details on using R Markdown see http://rmarkdown.rstudio.com.

When you click the **Knit** button a document will be generated that includes both content as well as the output of any embedded R code chunks within the document. You can embed an R code chunk like this:

Loading Dataset into data frame

```
##Loading Dataset into dataframe
Protein <- read.delim("C:/Alok/OneDrive/Rutgers_MITA/Semester2/MVA/MVA_Midter
m_FinalFolder/Protein_Consumption.csv", header = TRUE,sep = ",")
Prot_DS <- Protein
#View(Prot_DS)
names(Prot_DS)
```

```
##  [1] "ï..Country"              "Red.Meat"
##  [3] "White.Meat"              "Egg"
##  [5] "Milk"                    "Fish"
##  [7] "Cereals"                 "Starchy.Foods"
##  [9] "Pulses.Nuts.and.Oilseeds" "Fruits.and.Vegetables"
## [11] "Total"
```

```
#Renaming 1st column to Country for simplicity
names(Prot_DS)[names(Prot_DS) == "ï..Country"] <- "Country"
attach(Prot_DS)
```

```
#*********************************
```

Basic Exploratory Data Analysis:

```
head(Prot_DS)
```

```
##          Country Red.Meat White.Meat Egg Milk Fish Cereals Starchy.Foods
## 1        Albania       10          1   1    9    0      42             1
## 2        Austria        9         14   4   20    2      28             4
## 3        Belgium       14          9   4   18    5      27             6
## 4       Bulgaria        8          6   2    8    1      57             1
## 5 Czechoslovakia       10         11   3   13    2      34             5
## 6        Denmark       11         11   4   25   10      22             5
##   Pulses.Nuts.and.Oilseeds Fruits.and.Vegetables Total
## 1                        6                     2    72
## 2                        1                     4    86
## 3                        2                     4    89
## 4                        4                     4    91
## 5                        1                     4    83
## 6                        1                     2    91
```

```
dim(Prot_DS)
```

```
## [1] 25 11
```

#This dataset has 25 rows and 11 columns

```
#To check data types
str(Prot_DS)
```

```
## 'data.frame':    25 obs. of  11 variables:
##  $ Country                 : Factor w/ 25 levels "Albania","Austria",..: 1
2 3 4 5 6 7 8 9 10 ...
##  $ Red.Meat                : int  10 9 14 8 10 11 8 10 18 10 ...
##  $ White.Meat              : int  1 14 9 6 11 11 12 5 10 3 ...
##  $ Egg                     : int  1 4 4 2 3 4 4 3 3 3 ...
##  $ Milk                    : int  9 20 18 8 13 25 11 34 20 18 ...
##  $ Fish                    : int  0 2 5 1 2 10 5 6 6 6 ...
##  $ Cereals                 : int  42 28 27 57 34 22 25 26 28 42 ...
##  $ Starchy.Foods           : int  1 4 6 1 5 5 7 5 5 2 ...
##  $ Pulses.Nuts.and.Oilseeds: int  6 1 2 4 1 1 1 1 2 8 ...
##  $ Fruits.and.Vegetables   : int  2 4 4 4 4 2 4 1 7 7 ...
##  $ Total                   : int  72 86 89 91 83 91 77 91 99 99 ...
```

```
glimpse(Prot_DS)
```

```
## Observations: 25
## Variables: 11
## $ Country                 <fct> Albania, Austria, Belgium, Bulgaria, Czec
h...
## $ Red.Meat                <int> 10, 9, 14, 8, 10, 11, 8, 10, 18, 10, 5, 1
4...
```

```
## $ White.Meat               <int> 1, 14, 9, 6, 11, 11, 12, 5, 10, 3, 12, 10
,...
## $ Egg                      <int> 1, 4, 4, 2, 3, 4, 4, 3, 3, 3, 3, 5, 3, 4,
...
## $ Milk                     <int> 9, 20, 18, 8, 13, 25, 11, 34, 20, 18, 10,
...
## $ Fish                     <int> 0, 2, 5, 1, 2, 10, 5, 6, 6, 6, 0, 2, 3, 3
,...
## $ Cereals                  <int> 42, 28, 27, 57, 34, 22, 25, 26, 28, 42, 4
0...
## $ Starchy.Foods            <int> 1, 4, 6, 1, 5, 5, 7, 5, 5, 2, 4, 6, 2, 4,
...
## $ Pulses.Nuts.and.Oilseeds <int> 6, 1, 2, 4, 1, 1, 1, 1, 2, 8, 5, 2, 4, 2,
...
## $ Fruits.and.Vegetables    <int> 2, 4, 4, 4, 4, 2, 4, 1, 7, 7, 4, 3, 7, 4,
...
## $ Total                    <int> 72, 86, 89, 91, 83, 91, 77, 91, 99, 99, 8
3...
```

```
summary(Prot_DS)
```

```
##           Country      Red.Meat      White.Meat         Egg
##  Albania       : 1   Min.   : 4.0   Min.   : 1.00   Min.   :1.00
##  Austria       : 1   1st Qu.: 8.0   1st Qu.: 5.00   1st Qu.:3.00
##  Belgium       : 1   Median :10.0   Median : 8.00   Median :3.00
##  Bulgaria      : 1   Mean   : 9.8   Mean   : 7.92   Mean   :3.08
##  Czechoslovakia: 1   3rd Qu.:11.0   3rd Qu.:11.00   3rd Qu.:4.00
##  Denmark       : 1   Max.   :18.0   Max.   :14.00   Max.   :5.00
##  (Other)       :19
##       Milk            Fish          Cereals      Starchy.Foods
##  Min.   : 5.00   Min.   : 0.00   Min.   :19.00   Min.   :1.00
##  1st Qu.:11.00   1st Qu.: 2.00   1st Qu.:24.00   1st Qu.:3.00
##  Median :18.00   Median : 3.00   Median :28.00   Median :5.00
##  Mean   :17.28   Mean   : 4.28   Mean   :32.32   Mean   :4.36
##  3rd Qu.:23.00   3rd Qu.: 6.00   3rd Qu.:40.00   3rd Qu.:6.00
##  Max.   :34.00   Max.   :14.00   Max.   :57.00   Max.   :7.00
##
##  Pulses.Nuts.and.Oilseeds Fruits.and.Vegetables     Total
##  Min.   :1.00             Min.   :1.0           Min.   :72.00
##  1st Qu.:2.00             1st Qu.:3.0           1st Qu.:83.00
##  Median :2.00             Median :4.0           Median :87.00
##  Mean   :3.08             Mean   :4.2           Mean   :86.32
##  3rd Qu.:5.00             3rd Qu.:5.0           3rd Qu.:91.00
##  Max.   :8.00             Max.   :8.0           Max.   :99.00
##
```

```
grep('NA',Prot_DS)
```

```
## integer(0)
```

```r
#There are no NULL values in our dataset
#We will drop column 1 as it's categorical
Prot_DS.num <- Prot_DS[,-1]
```

#I am not passing 'Total' Column for Techniques and correlation matrix etc as it is just an addition of all the values. Also, it will definitely have co rrelation with other columns and so may create multicollinearity issues.

So dropping 'Total' variable.

```r
Prot_DS.num<- Prot_DS.num[,-10]
dim(Prot_DS.num)
```

```
## [1] 25  9
```

```r
head(Prot_DS.num)
```

```
##   Red.Meat White.Meat Egg Milk Fish Cereals Starchy.Foods
## 1       10          1   1    9    0      42             1
## 2        9         14   4   20    2      28             4
## 3       14          9   4   18    5      27             6
## 4        8          6   2    8    1      57             1
## 5       10         11   3   13    2      34             5
## 6       11         11   4   25   10      22             5
##   Pulses.Nuts.and.Oilseeds Fruits.and.Vegetables
## 1                        6                     2
## 2                        1                     4
## 3                        2                     4
## 4                        4                     4
## 5                        1                     4
## 6                        1                     2
```

```r
# Computing the means of each variable in data frame
colMeans(Prot_DS.num)
```

```
##                 Red.Meat               White.Meat                      Egg
##                     9.80                     7.92                     3.08
##                     Milk                     Fish                  Cereals
##                    17.28                     4.28                    32.32
##           Starchy.Foods Pulses.Nuts.and.Oilseeds    Fruits.and.Vegetables
##                     4.36                     3.08                     4.20
```

```r
#avg found
```

```r
# Covariance matrix without total column
cov(Prot_DS.num)
```

```
##                            Red.Meat White.Meat        Egg       Milk
## Red.Meat                 11.5833333   2.400000  2.1833333  13.141667
## White.Meat                2.4000000  13.993333  2.5066667   7.898333
```

```
## Egg                          2.1833333   2.506667  1.2433333   4.851667
## Milk                        13.1416667   7.898333  4.8516667  50.376667
## Fish                         0.7666667  -2.560000  0.1850000   4.001667
## Cereals                    -19.1000000 -18.098333 -8.6100000 -46.301667
## Starchy.Foods                0.8666667   2.071667  0.7616667   2.520000
## Pulses.Nuts.and.Oilseeds    -2.8166667  -5.076667 -1.3400000  -8.940000
## Fruits.and.Vegetables       -0.4166667  -0.525000 -0.3500000  -5.433333
##                                    Fish     Cereals Starchy.Foods
## Red.Meat                      0.7666667 -19.1000000     0.8666667
## White.Meat                   -2.5600000 -18.0983333     2.0716667
## Egg                           0.1850000  -8.6100000     0.7616667
## Milk                          4.0016667 -46.3016667     2.5200000
## Fish                         12.0433333 -19.7600000     2.5200000
## Cereals                     -19.7600000 121.2266667   -10.5366667
## Starchy.Foods                2.5200000 -10.5366667     2.7400000
## Pulses.Nuts.and.Oilseeds    -0.8566667  14.1400000    -1.6550000
## Fruits.and.Vegetables        1.5250000   0.8916667     0.2166667
##                           Pulses.Nuts.and.Oilseeds Fruits.and.Vegetables
## Red.Meat                                -2.8166667            -0.4166667
## White.Meat                              -5.0766667            -0.5250000
## Egg                                     -1.3400000            -0.3500000
## Milk                                    -8.9400000            -5.4333333
## Fish                                    -0.8566667             1.5250000
## Cereals                                 14.1400000             0.8916667
## Starchy.Foods                           -1.6550000             0.2166667
## Pulses.Nuts.and.Oilseeds                 4.0766667             1.3583333
## Fruits.and.Vegetables                    1.3583333             3.6666667
```

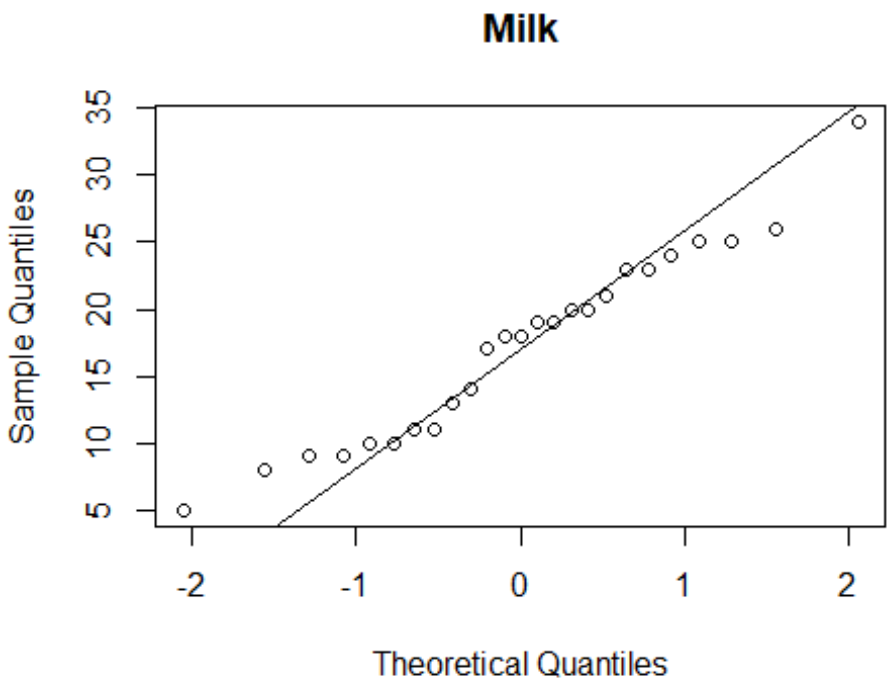
```
# Finding correlation -Correlation matrix takes units out and gives normalize
d values
cor(Prot_DS.num)
```

```
##                             Red.Meat  White.Meat         Egg       Milk
## Red.Meat                  1.00000000  0.18850977  0.57532001  0.5440251
## White.Meat                0.18850977  1.00000000  0.60095535  0.2974816
## Egg                       0.57532001  0.60095535  1.00000000  0.6130310
## Milk                      0.54402512  0.29748163  0.61303102  1.0000000
## Fish                      0.06491072 -0.19719960  0.04780844  0.1624624
## Cereals                  -0.50970337 -0.43941908 -0.70131040 -0.5924925
## Starchy.Foods             0.15383673  0.33456770  0.41266333  0.2144917
## Pulses.Nuts.and.Oilseeds -0.40988882 -0.67214885 -0.59519381 -0.6238357
## Fruits.and.Vegetables    -0.06393465 -0.07329308 -0.16392249 -0.3997753
##                                 Fish     Cereals Starchy.Foods
## Red.Meat                  0.06491072 -0.50970337     0.1538367
## White.Meat               -0.19719960 -0.43941908     0.3345677
## Egg                       0.04780844 -0.70131040     0.4126633
## Milk                      0.16246239 -0.59249246     0.2144917
## Fish                      1.00000000 -0.51714759     0.4386841
## Cereals                  -0.51714759  1.00000000    -0.5781345
## Starchy.Foods             0.43868411 -0.57813449     1.0000000
```

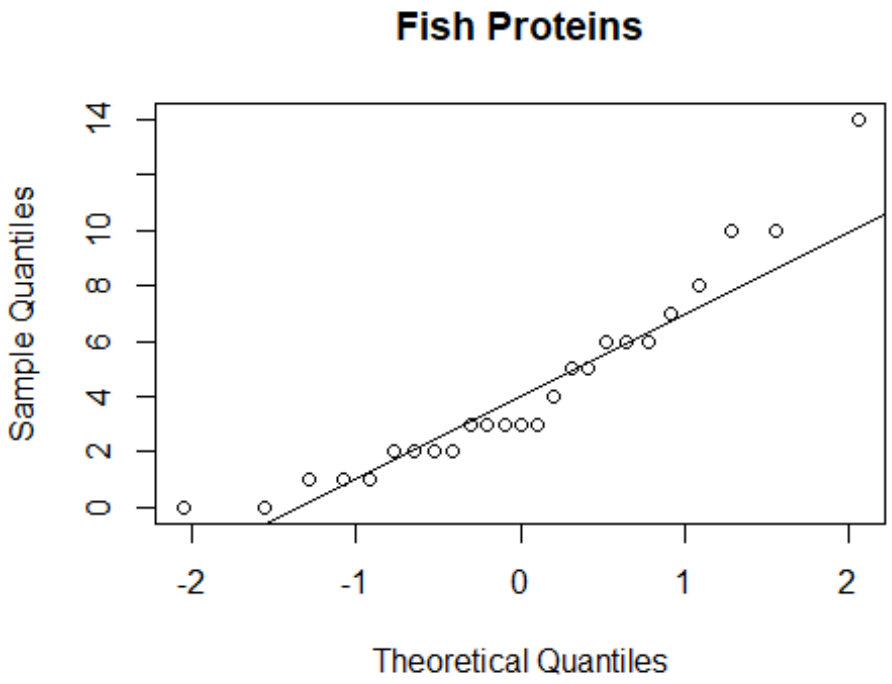
```
## Pulses.Nuts.and.Oilseeds -0.12226043  0.63605948    -0.4951880
## Fruits.and.Vegetables     0.22948842  0.04229293     0.0683567
##                          Pulses.Nuts.and.Oilseeds Fruits.and.Vegetables
## Red.Meat                               -0.4098888           -0.06393465
## White.Meat                             -0.6721488           -0.07329308
## Egg                                    -0.5951938           -0.16392249
## Milk                                   -0.6238357           -0.39977527
## Fish                                   -0.1222604            0.22948842
## Cereals                                 0.6360595            0.04229293
## Starchy.Foods                          -0.4951880            0.06835670
## Pulses.Nuts.and.Oilseeds                1.0000000            0.35133227
## Fruits.and.Vegetables                   0.3513323            1.00000000
```

To check if variables are Normal individually for Milk and Fish just to get an idea

```
#If it is normal it shud show straight line
qqnorm(Prot_DS.num[,"Milk"], main = "Milk")
qqline(Prot_DS.num[,"Milk"]) #not very bad
```




```
#Milk appears almost normal
qqnorm(Prot_DS.num[,"Fish"], main = "Fish Proteins")
qqline(Prot_DS.num[,"Fish"])
```



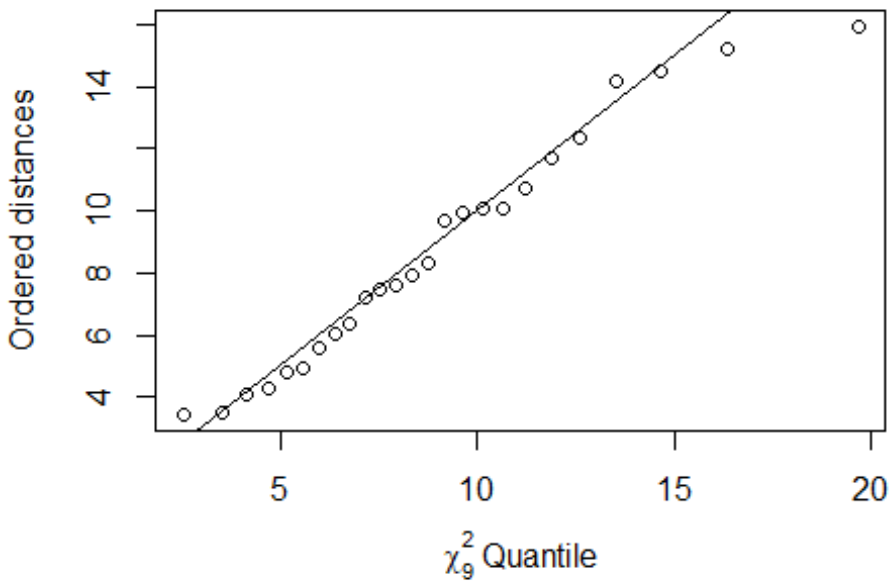

```
#Fish appears almost normal
```

#now multivariate plot to check if the variables are multivariate normal

```
names(Prot_DS.num)
```

```
## [1] "Red.Meat"                  "White.Meat"
## [3] "Egg"                       "Milk"
## [5] "Fish"                      "Cereals"
## [7] "Starchy.Foods"             "Pulses.Nuts.and.Oilseeds"
## [9] "Fruits.and.Vegetables"
```

```
x <- Prot_DS.num[, c("Red.Meat", "White.Meat", "Egg","Milk","Fish","Cereals",
                     "Starchy.Foods","Pulses.Nuts.and.Oilseeds","Fruits.and.V
egetables")]
cm <- colMeans(x)
#cm
S <- cov(x)
#S
d <- apply(x, MARGIN = 1, function(x)t(x - cm) %*% solve(S) %*% (x - cm))
#d
plot(qchisq((1:nrow(x) - 1/2) / nrow(x), df = 9), sort(d),
     xlab = expression(paste(chi[9]^2, " Quantile")),
     ylab = "Ordered distances")
abline(a = 0, b = 1)
```



This graph shows that our data is a multivariate normal and ca n be passed for our models without need of transformation.

```
#**************************
```

#Question1: PCA

```
##&&&&&&&&& PCA main start &&&&&&&&&&&&&&&&&&&&&&
```

```
#Get the Correlations between the measurements
#Finding correlation
#View(Prot_DS.num)
cor.PT<-cor(Prot_DS.num)
cor.PT
```

```
##                             Red.Meat  White.Meat         Egg       Milk
## Red.Meat                  1.00000000  0.18850977  0.57532001  0.5440251
## White.Meat                0.18850977  1.00000000  0.60095535  0.2974816
## Egg                       0.57532001  0.60095535  1.00000000  0.6130310
## Milk                      0.54402512  0.29748163  0.61303102  1.0000000
## Fish                      0.06491072 -0.19719960  0.04780844  0.1624624
## Cereals                  -0.50970337 -0.43941908 -0.70131040 -0.5924925
## Starchy.Foods             0.15383673  0.33456770  0.41266333  0.2144917
## Pulses.Nuts.and.Oilseeds -0.40988882 -0.67214885 -0.59519381 -0.6238357
## Fruits.and.Vegetables    -0.06393465 -0.07329308 -0.16392249 -0.3997753
##                                Fish     Cereals Starchy.Foods
```

```
## Red.Meat                    0.06491072 -0.50970337     0.1538367
## White.Meat                 -0.19719960 -0.43941908     0.3345677
## Egg                         0.04780844 -0.70131040     0.4126633
## Milk                        0.16246239 -0.59249246     0.2144917
## Fish                        1.00000000 -0.51714759     0.4386841
## Cereals                    -0.51714759  1.00000000    -0.5781345
## Starchy.Foods               0.43868411 -0.57813449     1.0000000
## Pulses.Nuts.and.Oilseeds   -0.12226043  0.63605948    -0.4951880
## Fruits.and.Vegetables       0.22948842  0.04229293     0.0683567
##                          Pulses.Nuts.and.Oilseeds Fruits.and.Vegetables
## Red.Meat                               -0.4098888           -0.06393465
## White.Meat                             -0.6721488           -0.07329308
## Egg                                    -0.5951938           -0.16392249
## Milk                                   -0.6238357           -0.39977527
## Fish                                   -0.1222604            0.22948842
## Cereals                                 0.6360595            0.04229293
## Starchy.Foods                          -0.4951880            0.06835670
## Pulses.Nuts.and.Oilseeds                1.0000000            0.35133227
## Fruits.and.Vegetables                   0.3513323            1.00000000
```

```
#Plotting correlation
corrplot(cor.PT,method="number")
```

#As per above graph, most of the variables are correlated with each other.
#There seems to be Negative correlation between Eggs and Cereals

#There seems to be Negative correlation between Milk & Pulses.nuts.oilseeds
#There seems to be +ve correlation between Cereals and Pulses.nuts.oilseeds
#There seems to be +ve correlation between White.Meat and Pulses.nuts.oilseeds

```
# Using prcomp to compute the principal components (eigenvalues and eigenvectors). With scale=TRUE, variable means are set to zero, and variances set to one
Protein_pca <- prcomp(Prot_DS.num,scale=TRUE)
summary(Protein_pca)
```

```
## Importance of components:
##                           PC1    PC2    PC3    PC4     PC5     PC6     PC7
## Standard deviation     2.0237 1.2747 1.0418 0.9513 0.65325 0.58902 0.51916
## Proportion of Variance 0.4551 0.1805 0.1206 0.1006 0.04742 0.03855 0.02995
## Cumulative Proportion  0.4551 0.6356 0.7562 0.8568 0.90417 0.94272 0.97266
##                           PC8     PC9
## Standard deviation     0.36677 0.33391
## Proportion of Variance 0.01495 0.01239
## Cumulative Proportion  0.98761 1.00000
```

```
Protein_pca
```

```
## Standard deviations (1, .., p=9):
## [1] 2.0237432 1.2747169 1.0417887 0.9513238 0.6532516 0.5890163 0.5191570
## [8] 0.3667732 0.3339091
##
## Rotation (n x k) = (9 x 9):
##                                 PC1         PC2         PC3         PC4
## Red.Meat                 -0.3106693 -0.06957085 -0.35546338 -0.59650142
## White.Meat               -0.3159279 -0.21457197  0.62841986 -0.03961214
## Egg                      -0.4205930 -0.09986721  0.08050675 -0.25525634
## Milk                     -0.3788776 -0.16867961 -0.40414435  0.03223542
## Fish                     -0.1341071  0.65161517 -0.29971395  0.23487897
## Cereals                   0.4298291 -0.25366332  0.06815673  0.02030764
## Starchy.Foods            -0.2959618  0.38888491  0.28085511  0.30524504
## Pulses.Nuts.and.Oilseeds  0.4218085  0.12932932 -0.14030066 -0.25125596
## Fruits.and.Vegetables     0.1223681  0.50377330  0.34041535 -0.60376932
##                                 PC5         PC6         PC7          PC8
## Red.Meat                  0.39658595 -0.37671581  0.22797808 -0.049688240
## White.Meat               -0.31059983 -0.08129384  0.14601621 -0.028186225
## Egg                       0.06707700  0.66453033  0.03595386 -0.467400341
## Milk                     -0.31800256  0.01779923 -0.71798985  0.102202763
## Fish                     -0.30432982 -0.04476482  0.23683595 -0.440552318
## Cereals                   0.18501820 -0.19398782 -0.34306417 -0.720660760
## Starchy.Foods             0.67317396  0.02444741 -0.32554187  0.082975933
## Pulses.Nuts.and.Oilseeds  0.09378094  0.58676016 -0.03105426  0.217739473
```

```
## Fruits.and.Vegetables    -0.22763119 -0.15823653 -0.35941199  0.009714519
##                                 PC9
## Red.Meat                 -0.2506754
## White.Meat               -0.5766036
## Egg                       0.2750188
## Milk                     -0.1903416
## Fish                     -0.2600351
## Cereals                  -0.1921878
## Starchy.Foods            -0.1499922
## Pulses.Nuts.and.Oilseeds -0.5666397
## Fruits.and.Vegetables     0.2114057
```

```
#9 Principal components are created as PC1, PC2, PC3, PC4, PC5, PC6, PC7, PC8 and PC9
head(Protein_pca) #op of this std deviation is in order PC1, PC2, PC3, PC4, PC5, PC6, PC7, PC8 and PC9
```

```
## $sdev
## [1] 2.0237432 1.2747169 1.0417887 0.9513238 0.6532516 0.5890163 0.5191570
## [8] 0.3667732 0.3339091
##
## $rotation
##                                 PC1         PC2         PC3         PC4
## Red.Meat                 -0.3106693 -0.06957085 -0.35546338 -0.59650142
## White.Meat               -0.3159279 -0.21457197  0.62841986 -0.03961214
## Egg                      -0.4205930 -0.09986721  0.08050675 -0.25525634
## Milk                     -0.3788776 -0.16867961 -0.40414435  0.03223542
## Fish                     -0.1341071  0.65161517 -0.29971395  0.23487897
## Cereals                   0.4298291 -0.25366332  0.06815673  0.02030764
## Starchy.Foods            -0.2959618  0.38888491  0.28085511  0.30524504
## Pulses.Nuts.and.Oilseeds  0.4218085  0.12932932 -0.14030066 -0.25125596
## Fruits.and.Vegetables     0.1223681  0.50377330  0.34041535 -0.60376932
##                                 PC5         PC6         PC7          PC8
## Red.Meat                  0.39658595 -0.37671581  0.22797808 -0.049688240
## White.Meat               -0.31059983 -0.08129384  0.14601621 -0.028186225
## Egg                       0.06707700  0.66453033  0.03595386 -0.467400341
## Milk                     -0.31800256  0.01779923 -0.71798985  0.102202763
## Fish                     -0.30432982 -0.04476482  0.23683595 -0.440552318
## Cereals                   0.18501820 -0.19398782 -0.34306417 -0.720660760
## Starchy.Foods             0.67317396  0.02444741 -0.32554187  0.082975933
## Pulses.Nuts.and.Oilseeds  0.09378094  0.58676016 -0.03105426  0.217739473
## Fruits.and.Vegetables    -0.22763119 -0.15823653 -0.35941199  0.009714519
##                                 PC9
## Red.Meat                 -0.2506754
## White.Meat               -0.5766036
## Egg                       0.2750188
## Milk                     -0.1903416
## Fish                     -0.2600351
## Cereals                  -0.1921878
## Starchy.Foods            -0.1499922
```

```
## Pulses.Nuts.and.Oilseeds -0.5666397
## Fruits.and.Vegetables     0.2114057
##
## $center
##                  Red.Meat                White.Meat                       Egg
##                      9.80                      7.92                      3.08
##                      Milk                      Fish                   Cereals
##                     17.28                      4.28                     32.32
##             Starchy.Foods Pulses.Nuts.and.Oilseeds    Fruits.and.Vegetables
##                      4.36                      3.08                      4.20
##
## $scale
##                  Red.Meat                White.Meat                       Egg
##                  3.403430                  3.740766                  1.115049
##                      Milk                      Fish                   Cereals
##                  7.097652                  3.470351                 11.010298
##             Starchy.Foods Pulses.Nuts.and.Oilseeds    Fruits.and.Vegetables
##                  1.655295                  2.019076                  1.914854
##
## $x
##               PC1         PC2          PC3         PC4        PC5
PC6
##  [1,]  3.4062175 -1.43187183 -1.596648133 -0.08434257  0.4124395 -0.266714
4820
##  [2,] -1.3961709 -1.07844406  1.234558817 -0.02919248 -0.7564630  0.023797
5418
##  [3,] -1.6271911  0.27394175 -0.009163712 -0.41608341  0.9108462 -0.126926
3837
##  [4,]  3.0996115 -1.50333675  0.082356700 -0.30660707 -0.2970873 -0.584211
9100
##  [5,] -0.4277883 -0.57418064  1.159335459  0.21991003  0.3701307 -0.726157
0266
##  [6,] -2.4422594  0.28305004 -0.676942687  1.02016258 -0.6562849  0.062718
4045
##  [7,] -1.4249913  0.60782538  1.746831101  0.87710306  0.6028516  0.213844
8106
##  [8,] -1.7006498 -0.58298031 -1.972677332  1.58071748 -0.2011453 -0.205840
6000
##  [9,] -1.4354297  0.89590251 -0.161539920 -1.95053301  0.3099538 -1.475552
7601
## [10,]  2.3291742  0.86546599 -1.227337046 -1.75741320 -0.6575195  1.009731
2103
## [11,]  1.4302687 -0.95052166  1.782611863  0.26555332 -0.1057918  0.865773
2666
## [12,] -2.5809791 -0.82037615 -0.161750192 -0.51252848  0.8610870  0.641559
5029
## [13,]  1.5501576  0.16192833 -0.053056104 -1.33599650 -0.7676190  0.031281
8001
## [14,] -1.7115591 -0.78012960  0.766301047 -0.25865817 -0.9164207  0.304055
3671
```

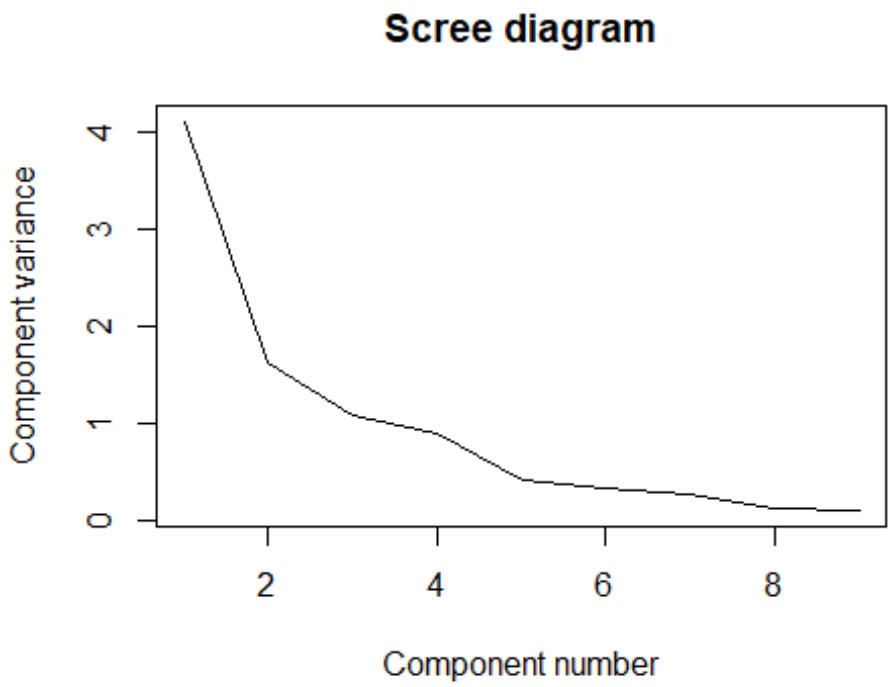
```
## [15,] -0.9571511  1.10929163 -1.319851198  1.21615923 -0.4173226  0.003856
1601
## [16,] -0.1285106  0.63184836  1.522555810 -0.03104612 -0.1228267 -0.347985
4540
## [17,]  1.8854364  4.23632323  0.235407502  0.64127627 -0.3296311 -0.528080
5539
## [18,]  2.6361730 -1.10164486  0.169166371  0.60431439  0.1965040  0.170800
0230
## [19,]  1.4042842  2.43957843  0.249276728 -0.24228673  0.6238140  1.013227
6525
## [20,] -1.9196053 -0.08881654 -1.085799797  0.90373795 -0.7886161  0.284867
8709
## [21,] -0.8862644 -0.79798276 -0.228906351 -1.06865159 -0.7103254 -0.689517
4928
## [22,] -1.9396765 -0.32877834 -1.274231236 -1.19215725  1.2311866  0.633927
4501
## [23,]  0.8607657 -0.15774231 -0.215679913  1.04275420  1.2112175 -0.581477
6989
## [24,] -1.8007758 -0.34409820  0.872728311 -0.26262846 -0.1813817  0.272694
5750
## [25,]  3.7769132 -0.96425165  0.162453908  1.07643653  0.1784042  0.000328
7263
##               PC7         PC8         PC9
##  [1,]  0.94892837  0.84693053  0.15478609
##  [2,]  0.05758584 -0.05177819  0.11624278
##  [3,]  0.22683921 -0.22319293 -0.09689498
##  [4,]  0.39976618 -0.90940273  0.25018422
##  [5,]  0.29971869 -0.06798719  0.25074519
##  [6,]  0.48030200 -0.56925372 -0.50886295
##  [7,]  0.53117349 -0.18580431  0.29526903
##  [8,] -0.97347796  0.28022893  0.12113082
##  [9,] -0.03008584 -0.06846045 -0.51649154
## [10,] -0.57538334 -0.34740216 -0.45103458
## [11,]  0.11900810  0.19668872 -0.44150330
## [12,] -0.43471746  0.03742272 -0.05217871
## [13,] -0.14708797 -0.12872601  0.85624862
## [14,]  0.06091030  0.35043459 -0.28870555
## [15,]  0.04796743 -0.05700862  0.18258443
## [16,] -1.31643147 -0.01492251  0.31505313
## [17,]  0.53140483  0.20289705 -0.20295441
## [18,] -0.04058813 -0.17580879 -0.13304725
## [19,] -0.14851022  0.27557451  0.36210459
## [20,]  0.41870881 -0.19737555  0.30259740
## [21,] -0.21158255  0.59042991  0.03956071
## [22,]  0.43367349 -0.24441516  0.13761916
## [23,] -0.72141844 -0.05214970 -0.11645720
## [24,]  0.39030488  0.53225955 -0.13919641
## [25,] -0.34700826 -0.01917849 -0.43679928
```

```
#Protein_pca
#Insights from Above PCA Output
#Below are the Protein sources that influence the Contents of Principal C
omponents:
#PC1 is dominated by Negative effect of Cereals &Egg and Positive effect
of Pulse.Nut.oilseeds
#PC2 is dominated by Positive effect of Fish and Fruits.Veg
#PC3 is dominated by Negative effect of Milk and Positive effect White.m
eat
#PC4 is dominated by Negative effect of Red.Meat and Fruits.Veg
#PC5 is dominated by Positive effect of Starchy.Foods

#From Summary of Pincipal components,
#Proportion of Variance, PC1, PC3 until PC5 explain 45%,  18%, 12%, 10
% and 4% of variance respectively.
#'Cumulative Proportion' field, 90.5% of Cummulative variance is explai
ned by PC1 until PC5
#So I will include PC1 until PC5 in my data input
#So My input variables will be reduced from 11 to 5

#$x gives the new dataset #u need to rename these columns
head(Protein_pca$x)

##             PC1        PC2          PC3         PC4        PC5         PC6
## [1,]  3.4062175 -1.4318718 -1.596648133 -0.08434257  0.4124395 -0.26671448
## [2,] -1.3961709 -1.0784441  1.234558817 -0.02919248 -0.7564630  0.02379754
## [3,] -1.6271911  0.2739418 -0.009163712 -0.41608341  0.9108462 -0.12692638
## [4,]  3.0996115 -1.5033368  0.082356700 -0.30660707 -0.2970873 -0.58421191
## [5,] -0.4277883 -0.5741806  1.159335459  0.21991003  0.3701307 -0.72615703
## [6,] -2.4422594  0.2830500 -0.676942687  1.02016258 -0.6562849  0.06271840
##             PC7         PC8         PC9
## [1,] 0.94892837  0.84693053  0.15478609
## [2,] 0.05758584 -0.05177819  0.11624278
## [3,] 0.22683921 -0.22319293 -0.09689498
## [4,] 0.39976618 -0.90940273  0.25018422
## [5,] 0.29971869 -0.06798719  0.25074519
## [6,] 0.48030200 -0.56925372 -0.50886295

(eigen_Prot <- Protein_pca$sdev^2) #singular values (square roots of eigenval
ues)

## [1] 4.0955365 1.6249031 1.0853237 0.9050170 0.4267377 0.3469402 0.2695240
## [8] 0.1345226 0.1114953

names(eigen_Prot) <- paste("PC",1:9,sep="") #Naming PC components
eigen_Prot
```

```
##        PC1        PC2        PC3        PC4        PC5        PC6        PC7
PC8
## 4.0955365 1.6249031 1.0853237 0.9050170 0.4267377 0.3469402 0.2695240 0.13
45226
##        PC9
## 0.1114953
```

#Plotting Scree diagram

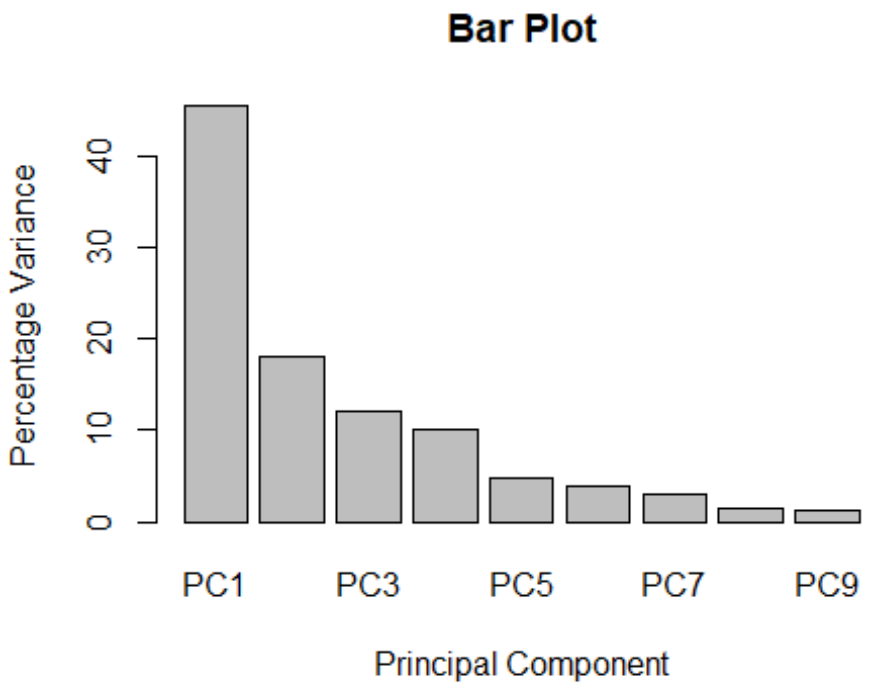
```
plot(eigen_Prot, xlab = "Component number", ylab = "Component variance", type
= "l", main = "Scree diagram")
```



#Scree plot confirms that taking 5 Principals is enough without loosing much information.

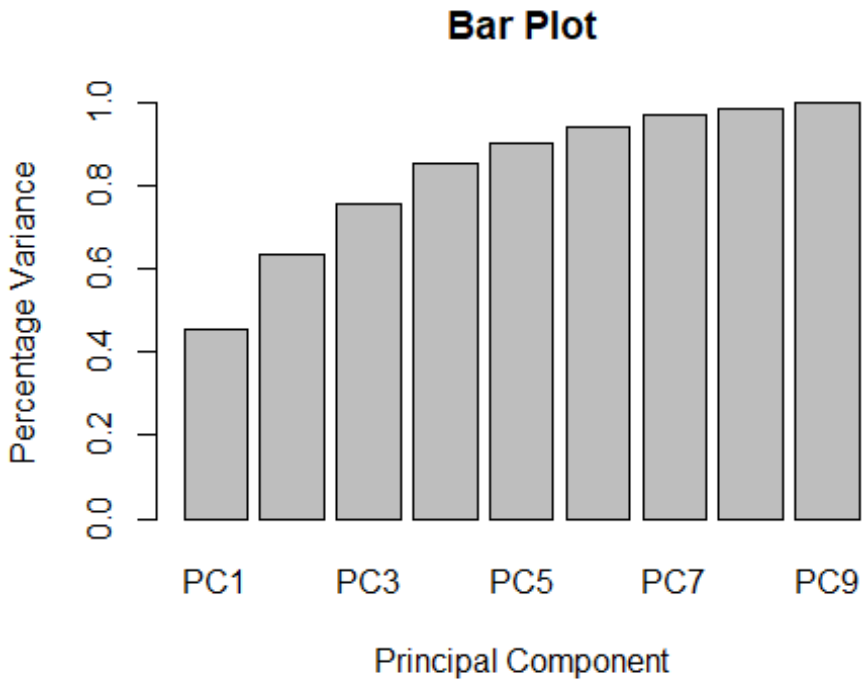
```
sumlambdas <- sum(eigen_Prot)
sumlambdas #sum of genvalues is total var of ur dataset

## [1] 9

propvar <- eigen_Prot/sumlambdas
#Printing Proper variance per PC
propvar

##        PC1        PC2        PC3        PC4        PC5        PC6        P
C7
## 0.45505961 0.18054478 0.12059152 0.10055744 0.04741530 0.03854891 0.029947
11
```

```
##         PC8         PC9
## 0.01494695 0.01238837

#Percentage of total variance
percentvar <- (eigen_Prot/sumlambdas) *100
percentvar

##       PC1       PC2       PC3       PC4       PC5       PC6       PC7
PC8
## 45.505961 18.054478 12.059152 10.055744  4.741530  3.854891  2.994711  1.4
94695
##       PC9
##  1.238837

#Bar plot of Percentage variance
barplot(percentvar, main = "Bar Plot", xlab = "Principal Component", ylab = "
Percentage Variance")
```

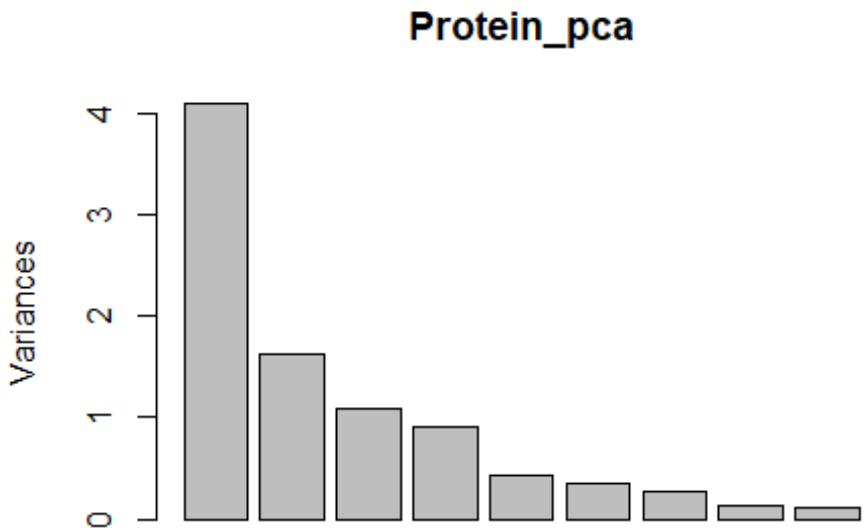



```
#As per above graph, PC1 holds 78% of ur total var, PC2 14% and so on
#Cummulative variance
cumvar_Prot <- cumsum(propvar)
cumvar_Prot

##       PC1       PC2       PC3       PC4       PC5       PC6       PC7
PC8
## 0.4550596 0.6356044 0.7561959 0.8567534 0.9041687 0.9427176 0.9726647 0.98
76116
```

```
##       PC9
## 1.0000000

#Bar plot of Cummulative Percentage variance
barplot(cumvar_Prot, main = "Bar Plot", xlab = "Principal Component", ylab =
"Percentage Variance")
```




```
#Plotting log scree diagram
#plot(log(eigen_Prot), xlab = "Component number",ylab = "log(Component varian
ce)", type="l",main = "Log(eigenvalue) diagram")
#Plotting Histogram
plot(Protein_pca)
```




```
#Printing our new Dataset after PCA
#Binding with categorical columns from the original dataset
Prottyp_pca <- cbind(data.frame(Country),Protein_pca$x)
head(Prottyp_pca)
```

```
##          Country        PC1        PC2          PC3         PC4        PC5
## 1        Albania  3.4062175 -1.4318718 -1.596648133 -0.08434257  0.4124395
## 2        Austria -1.3961709 -1.0784441  1.234558817 -0.02919248 -0.7564630
## 3        Belgium -1.6271911  0.2739418 -0.009163712 -0.41608341  0.9108462
## 4       Bulgaria  3.0996115 -1.5033368  0.082356700 -0.30660707 -0.2970873
## 5 Czechoslovakia -0.4277883 -0.5741806  1.159335459  0.21991003  0.3701307
## 6        Denmark -2.4422594  0.2830500 -0.676942687  1.02016258 -0.6562849
##            PC6        PC7         PC8         PC9
## 1 -0.26671448 0.94892837  0.84693053  0.15478609
## 2  0.02379754 0.05758584 -0.05177819  0.11624278
## 3 -0.12692638 0.22683921 -0.22319293 -0.09689498
## 4 -0.58421191 0.39976618 -0.90940273  0.25018422
## 5 -0.72615703 0.29971869 -0.06798719  0.25074519
## 6  0.06271840 0.48030200 -0.56925372 -0.50886295
```

```
#Renaming Principal components PC1 to PC5
names(Prottyp_pca) <- c("Country", "NegCerlEgg_PostivePulse", "NegativeFish_FrutVeg",
                        "NegateMilk_PostiveWhtMeat","NegateRedMet_FrutVeg","Postive_StarchFud",
                        "PC6","PC7","PC8","PC9")
```

#This is our new dataset

```
head(Prottyp_pca,5)
```

```
##          Country NegCerlEgg_PostivePulse NegativeFish_FrutVeg
## 1        Albania               3.4062175           -1.4318718
## 2        Austria              -1.3961709           -1.0784441
## 3        Belgium              -1.6271911            0.2739418
## 4       Bulgaria               3.0996115           -1.5033368
## 5 Czechoslovakia              -0.4277883           -0.5741806
##   NegateMilk_PostiveWhtMeat NegateRedMet_FrutVeg Postive_StarchFud
PC6
## 1              -1.596648133          -0.08434257         0.4124395 -0.2667
1448
## 2               1.234558817          -0.02919248        -0.7564630  0.0237
9754
## 3              -0.009163712          -0.41608341         0.9108462 -0.1269
2638
## 4               0.082356700          -0.30660707        -0.2970873 -0.5842
1191
## 5               1.159335459           0.21991003         0.3701307 -0.7261
5703
##          PC7         PC8         PC9
## 1 0.94892837  0.84693053  0.15478609
## 2 0.05758584 -0.05177819  0.11624278
## 3 0.22683921 -0.22319293 -0.09689498
## 4 0.39976618 -0.90940273  0.25018422
## 5 0.29971869 -0.06798719  0.25074519
```

#PCA Conclusion:

#Principal Component analysis is a statistical technique that uses Orthog
onal Transformation.

#It helps in reducing the number of input variables to be passed to a mo
del.

#The principal componens are Non-correlated with each other.

#After performing PCA on Protein Consumption dataset, it can be conclu
ded that:

#Based on per person protein consumption in Europian countries,

#Total 9 Principal components are created.

#Contents of Principal Components:

#PC1 is dominated by Negative effect of Cereals &Egg and Positive effect
of Pulse.Nut.oilseeds

#PC2 is dominated by Positive effect of Fish and Fruits.Veg

#PC3 is dominated by Negative effect of Milk and Positive effect White.m
eat

#PC4 is dominated by Negative effect of Red.Meat and Fruits.Veg

```
#PC5 is dominated by Positive effect of Starchy.Foods
#PC components renamed accordingly.
#From Summary of Pincipal components,
#Proportion of Variance, PC1, PC3 until PC5 explain 45%,  18%, 12%, 10 % and 4% of variance respectively.
#'Cumulative Proportion' field, 90.5% of Cummulative variance is explained by PC1 until PC5
#So I will include PC1 until PC5 in my data input
#So My input variables will be reduced from 11 to 5.

##########  PCA main end &&&&&&&&&&&&&&&&&&&&&&&&
```

Note that the `echo = FALSE` parameter was added to the code chunk to prevent printing of the R code that generated the plot.

#Question Number 3: Factor Analysis:

```
##########  Fractal main start &&&&&&&&&&&&&&&&&&&&&&&&
#Creating new input dataframe for Factor analysis With ROW NAMES.
PT_Fact_1 <- read.csv("C:/Alok/OneDrive/Rutgers_MITA/Semester2/MVA/MVA_Midterm_FinalFolder/Protein_Consumption.csv",row.names=1, fill = TRUE)
head(PT_Fact_1)
```

```
##                Red.Meat White.Meat Egg Milk Fish Cereals Starchy.Foods
## Albania              10          1   1    9    0      42             1
## Austria               9         14   4   20    2      28             4
## Belgium              14          9   4   18    5      27             6
## Bulgaria              8          6   2    8    1      57             1
## Czechoslovakia       10         11   3   13    2      34             5
## Denmark              11         11   4   25   10      22             5
##                Pulses.Nuts.and.Oilseeds Fruits.and.Vegetables Total
## Albania                               6                     2    72
## Austria                               1                     4    86
## Belgium                               2                     4    89
## Bulgaria                              4                     4    91
## Czechoslovakia                        1                     4    83
## Denmark                               1                     2    91
```

```
names(PT_Fact_1)
```

```
##  [1] "Red.Meat"                "White.Meat"
##  [3] "Egg"                     "Milk"
##  [5] "Fish"                    "Cereals"
##  [7] "Starchy.Foods"           "Pulses.Nuts.and.Oilseeds"
##  [9] "Fruits.and.Vegetables"   "Total"
```

```r
#Removing Total Column
PT_Fact_1<- PT_Fact_1[,-10]
dim(PT_Fact_1)
```

```
## [1] 25  9
```

```r
attach(PT_Fact_1)
```

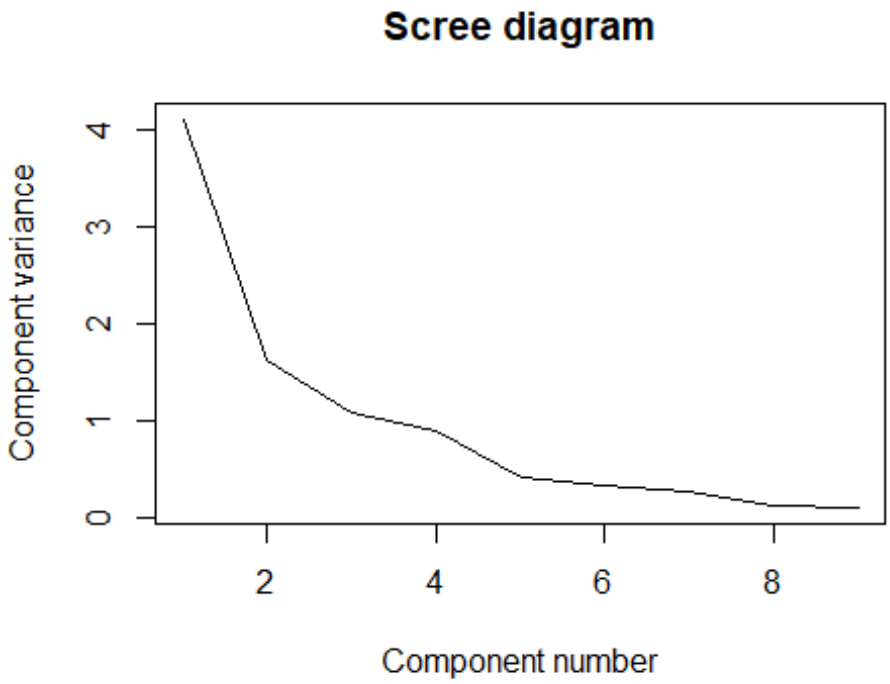
```
## The following objects are masked from Prot_DS:
##
##     Cereals, Egg, Fish, Fruits.and.Vegetables, Milk,
##     Pulses.Nuts.and.Oilseeds, Red.Meat, Starchy.Foods, White.Meat
```

```r
#Finding correlation
cor.PT<-cor(PT_Fact_1)
cor.PT
```

```
##                             Red.Meat  White.Meat         Egg       Milk
## Red.Meat                  1.00000000  0.18850977  0.57532001  0.5440251
## White.Meat                0.18850977  1.00000000  0.60095535  0.2974816
## Egg                       0.57532001  0.60095535  1.00000000  0.6130310
## Milk                      0.54402512  0.29748163  0.61303102  1.0000000
## Fish                      0.06491072 -0.19719960  0.04780844  0.1624624
## Cereals                  -0.50970337 -0.43941908 -0.70131040 -0.5924925
## Starchy.Foods             0.15383673  0.33456770  0.41266333  0.2144917
## Pulses.Nuts.and.Oilseeds -0.40988882 -0.67214885 -0.59519381 -0.6238357
## Fruits.and.Vegetables    -0.06393465 -0.07329308 -0.16392249 -0.3997753
##                                 Fish     Cereals Starchy.Foods
## Red.Meat                  0.06491072 -0.50970337     0.1538367
## White.Meat               -0.19719960 -0.43941908     0.3345677
## Egg                       0.04780844 -0.70131040     0.4126633
## Milk                      0.16246239 -0.59249246     0.2144917
## Fish                      1.00000000 -0.51714759     0.4386841
## Cereals                  -0.51714759  1.00000000    -0.5781345
## Starchy.Foods             0.43868411 -0.57813449     1.0000000
## Pulses.Nuts.and.Oilseeds -0.12226043  0.63605948    -0.4951880
## Fruits.and.Vegetables     0.22948842  0.04229293     0.0683567
##                          Pulses.Nuts.and.Oilseeds Fruits.and.Vegetables
## Red.Meat                               -0.4098888           -0.06393465
## White.Meat                             -0.6721488           -0.07329308
## Egg                                    -0.5951938           -0.16392249
## Milk                                   -0.6238357           -0.39977527
## Fish                                   -0.1222604            0.22948842
## Cereals                                 0.6360595            0.04229293
## Starchy.Foods                          -0.4951880            0.06835670
## Pulses.Nuts.and.Oilseeds                1.0000000            0.35133227
## Fruits.and.Vegetables                   0.3513323            1.00000000
```

```r
#Plotting correlation
corrplot(cor.PT,method="number")
```

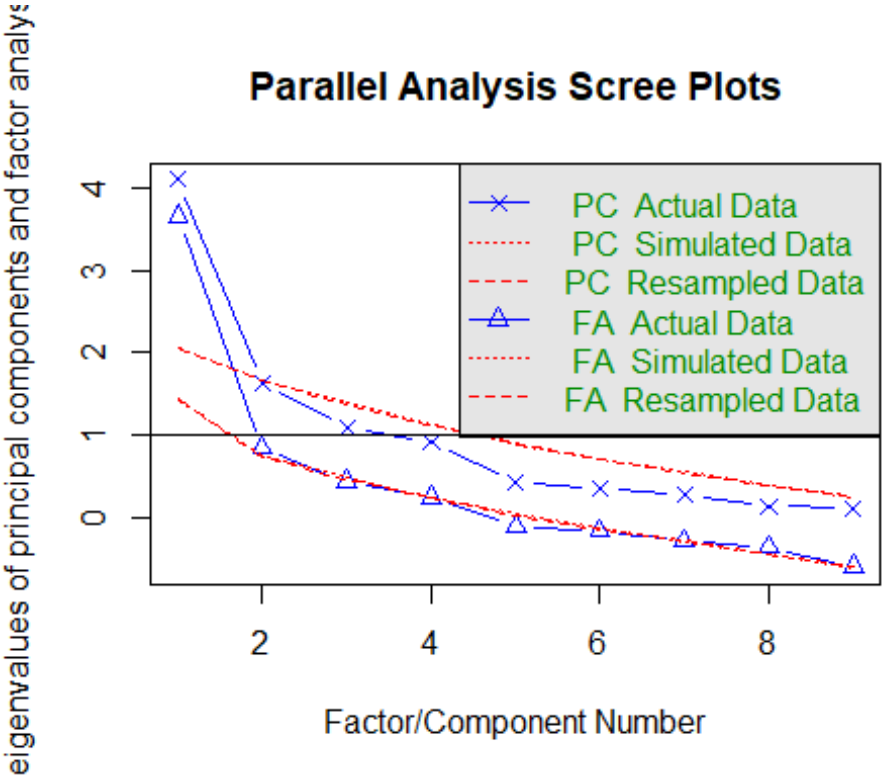

```
#As per above graph, most of the variables are correlated with each other.
#There seems to be Negative correlation between Eggs and Cereals
#There seems to be Negative correlation between Milk & Pulses.nuts.oilseeds
#There seems to be +ve correlation between Cereals and  Pulses.nuts.oilseeds
#There seems to be +ve correlation between White.Meat and  Pulses.nuts.oilseeds

#To check how many factors needed, Plotting Scree diagram
#Same scree diagram used in PCA
plot(eigen_Prot, xlab = "Component number", ylab = "Component variance", type = "l", main = "Scree diagram")
```




```
#As per scree plot, there should be 5 factors, will see what parallel analy
sis suggests
fa.parallel(PT_Fact_1)

## Warning in fa.stats(r = r, f = f, phi = phi, n.obs = n.obs, np.obs = np.ob
s, :
## The estimated weights for the factor scores are probably incorrect. Try a
## different factor score estimation method.

## Warning in fa.stats(r = r, f = f, phi = phi, n.obs = n.obs, np.obs = np.ob
s, :
## The estimated weights for the factor scores are probably incorrect. Try a
## different factor score estimation method.

## Warning in fac(r = r, nfactors = nfactors, n.obs = n.obs, rotate = rotate,
: An
## ultra-Heywood case was detected. Examine the results carefully

## Warning in fa.stats(r = r, f = f, phi = phi, n.obs = n.obs, np.obs = np.ob
s, :
## The estimated weights for the factor scores are probably incorrect. Try a
## different factor score estimation method.

## Warning in fac(r = r, nfactors = nfactors, n.obs = n.obs, rotate = rotate,
: An
## ultra-Heywood case was detected. Examine the results carefully
```

```
## Warning in fa.stats(r = r, f = f, phi = phi, n.obs = n.obs, np.obs = np.ob
s, :
## The estimated weights for the factor scores are probably incorrect. Try a
## different factor score estimation method.

## Warning in fac(r = r, nfactors = nfactors, n.obs = n.obs, rotate = rotate,
: An
## ultra-Heywood case was detected. Examine the results carefully

## Warning in fa.stats(r = r, f = f, phi = phi, n.obs = n.obs, np.obs = np.ob
s, :
## The estimated weights for the factor scores are probably incorrect. Try a
## different factor score estimation method.

## Warning in fac(r = r, nfactors = nfactors, n.obs = n.obs, rotate = rotate,
: An
## ultra-Heywood case was detected. Examine the results carefully

## Warning in fa.stats(r = r, f = f, phi = phi, n.obs = n.obs, np.obs = np.ob
s, :
## The estimated weights for the factor scores are probably incorrect. Try a
## different factor score estimation method.

## Warning in fac(r = r, nfactors = nfactors, n.obs = n.obs, rotate = rotate,
: An
## ultra-Heywood case was detected. Examine the results carefully

## Warning in fa.stats(r = r, f = f, phi = phi, n.obs = n.obs, np.obs = np.ob
s, :
## The estimated weights for the factor scores are probably incorrect. Try a
## different factor score estimation method.

## Warning in fac(r = r, nfactors = nfactors, n.obs = n.obs, rotate = rotate,
: An
## ultra-Heywood case was detected. Examine the results carefully

## Warning in fa.stats(r = r, f = f, phi = phi, n.obs = n.obs, np.obs = np.ob
s, :
## The estimated weights for the factor scores are probably incorrect. Try a
## different factor score estimation method.

## Warning in fac(r = r, nfactors = nfactors, n.obs = n.obs, rotate = rotate,
: An
## ultra-Heywood case was detected. Examine the results carefully

## Warning in fa.stats(r = r, f = f, phi = phi, n.obs = n.obs, np.obs = np.ob
s, :
## The estimated weights for the factor scores are probably incorrect. Try a
## different factor score estimation method.
```

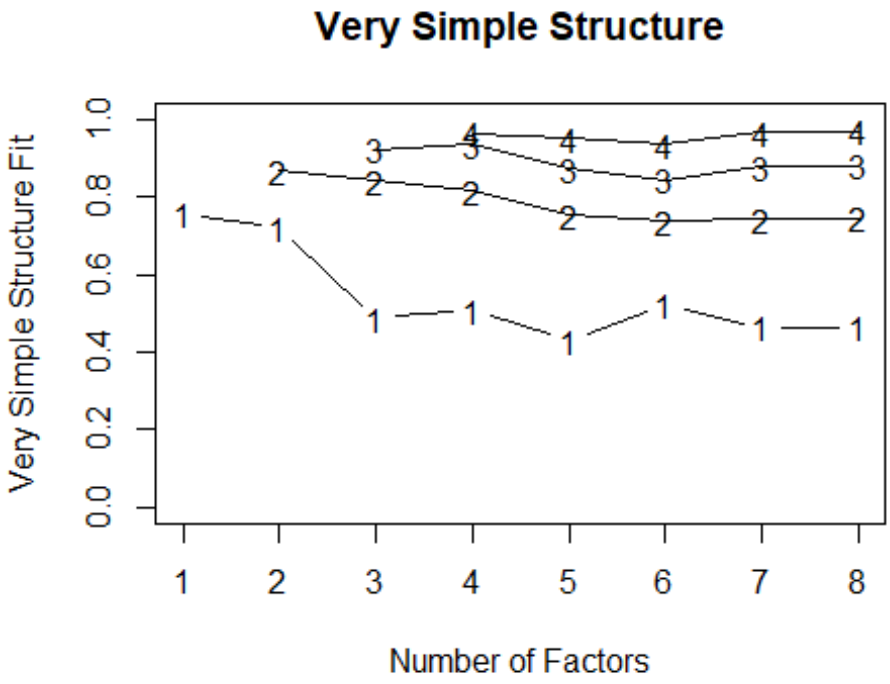
```
## Warning in fac(r = r, nfactors = nfactors, n.obs = n.obs, rotate = rotate,
: An
## ultra-Heywood case was detected. Examine the results carefully
```



#Parallel analysis Also suggests that the number of factors are 5 or 6.

```
vss(PT_Fact_1) # See Factor recommendations for a simple structure
```

```
## Warning in fa.stats(r = r, f = f, phi = phi, n.obs = n.obs, np.obs = np.ob
s, :
## The estimated weights for the factor scores are probably incorrect. Try a
## different factor score estimation method.

## Warning in fa.stats(r = r, f = f, phi = phi, n.obs = n.obs, np.obs = np.ob
s, :
## An ultra-Heywood case was detected. Examine the results carefully
```

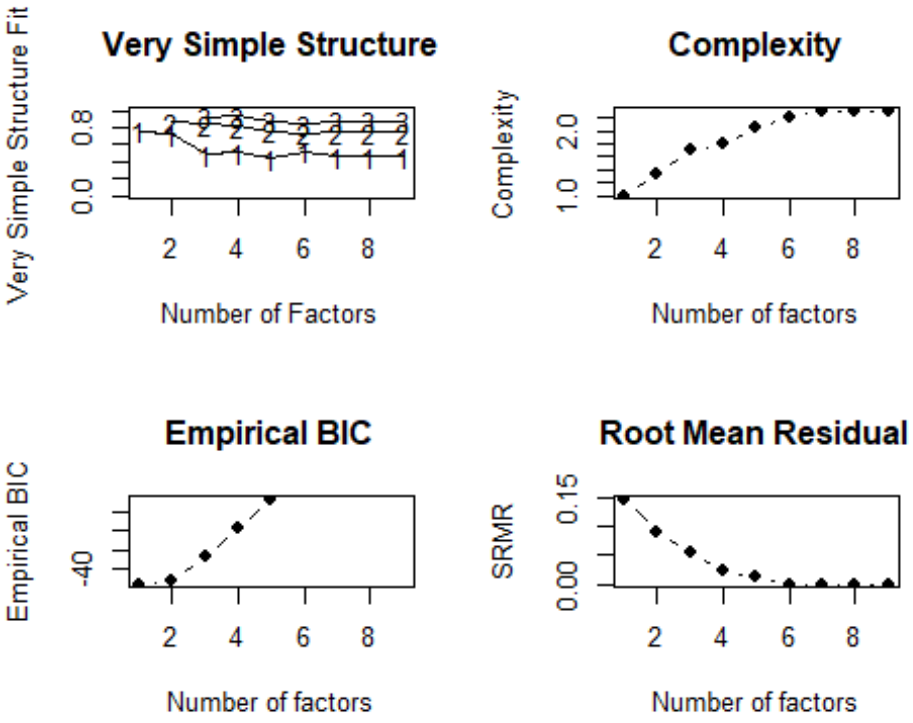



```
##
## Very Simple Structure
## Call: vss(x = PT_Fact_1)
## VSS complexity 1 achieves a maximimum of 0.76  with  1  factors
## VSS complexity 2 achieves a maximimum of 0.87  with  2  factors
##
## The Velicer MAP achieves a minimum of 0.08  with  1  factors
## BIC achieves a minimum of  NA  with  1  factors
## Sample Size adjusted BIC achieves a minimum of  NA  with  5  factors
##
## Statistics by number of factors
##   vss1 vss2   map dof   chisq prob sqresid  fit RMSEA   BIC SABIC complex
## 1 0.76 0.00 0.084  27 4.7e+01 0.01    5.31 0.76 0.167 -40.0  43.7     1.0
## 2 0.72 0.87 0.098  19 2.7e+01 0.11    2.90 0.87 0.121 -34.4  24.6     1.3
## 3 0.49 0.84 0.110  12 1.4e+01 0.27    1.71 0.92 0.081 -24.1  13.1     1.7
## 4 0.51 0.82 0.144   6 5.4e+00 0.49    0.78 0.96 0.000 -13.9   4.7     1.8
## 5 0.43 0.75 0.208   1 1.4e+00 0.23    0.67 0.97 0.123  -1.8   1.3     2.0
## 6 0.52 0.74 0.301  -3 4.9e-08   NA    0.45 0.98    NA    NA    NA     2.2
## 7 0.46 0.74 0.496  -6 4.7e-08   NA    0.42 0.98    NA    NA    NA     2.3
## 8 0.46 0.75 1.000  -8 0.0e+00   NA    0.40 0.98    NA    NA    NA     2.3
##    eChisq    SRMR eCRMS eBIC
## 1 3.9e+01 1.5e-01 0.169  -48
## 2 1.5e+01 9.1e-02 0.125  -46
## 3 5.3e+00 5.4e-02 0.094  -33
## 4 8.7e-01 2.2e-02 0.054  -18
## 5 2.4e-01 1.2e-02 0.070   -3
## 6 5.5e-09 1.8e-06    NA   NA
```

```
## 7 3.8e-09 1.5e-06    NA   NA
## 8 1.7e-19 9.8e-12    NA   NA
```

```
nfactors(PT_Fact_1)
```

```
## Warning in fa.stats(r = r, f = f, phi = phi, n.obs = n.obs, np.obs = np.ob
s, :
## The estimated weights for the factor scores are probably incorrect. Try a
## different factor score estimation method.

## Warning in fa.stats(r = r, f = f, phi = phi, n.obs = n.obs, np.obs = np.ob
s, :
## An ultra-Heywood case was detected. Examine the results carefully
```




```
##
## Number of factors
## Call: vss(x = x, n = n, rotate = rotate, diagonal = diagonal, fm = fm,
##     n.obs = n.obs, plot = FALSE, title = title, use = use, cor = cor)
## VSS complexity 1 achieves a maximimum of 0.76  with  1  factors
## VSS complexity 2 achieves a maximimum of 0.87  with  2  factors
## The Velicer MAP achieves a minimum of 0.08  with  1  factors
## Empirical BIC achieves a minimum of  -48.17  with  1  factors
## Sample Size adjusted BIC achieves a minimum of  1.3  with  5  factors
##
## Statistics by number of factors
##   vss1 vss2   map dof   chisq prob sqresid  fit RMSEA   BIC SABIC complex
## 1 0.76 0.00 0.084  27 4.7e+01 0.01    5.31 0.76 0.167 -40.0  43.7     1.0
```

```
## 2 0.72 0.87 0.098  19 2.7e+01 0.11    2.90 0.87 0.121 -34.4  24.6     1.3
## 3 0.49 0.84 0.110  12 1.4e+01 0.27    1.71 0.92 0.081 -24.1  13.1     1.7
## 4 0.51 0.82 0.144   6 5.4e+00 0.49    0.78 0.96 0.000 -13.9   4.7     1.8
## 5 0.43 0.75 0.208   1 1.4e+00 0.23    0.67 0.97 0.123  -1.8   1.3     2.0
## 6 0.52 0.74 0.301  -3 4.9e-08   NA    0.45 0.98    NA    NA    NA     2.2
## 7 0.46 0.74 0.496  -6 4.7e-08   NA    0.42 0.98    NA    NA    NA     2.3
## 8 0.46 0.75 1.000  -8 0.0e+00   NA    0.40 0.98    NA    NA    NA     2.3
## 9 0.46 0.75    NA  -9 0.0e+00   NA    0.40 0.98    NA    NA    NA     2.3
##    eChisq    SRMR eCRMS eBIC
## 1 3.9e+01 1.5e-01 0.169  -48
## 2 1.5e+01 9.1e-02 0.125  -46
## 3 5.3e+00 5.4e-02 0.094  -33
## 4 8.7e-01 2.2e-02 0.054  -18
## 5 2.4e-01 1.2e-02 0.070   -3
## 6 5.5e-09 1.8e-06    NA   NA
## 7 3.8e-09 1.5e-06    NA   NA
## 8 1.7e-19 9.8e-12    NA   NA
## 9 1.7e-19 9.8e-12    NA   NA
```

#nfactors suggests we can either go with 5 factors or 6 factors, #I would prefer 5 factors as suggested in scree plot

```
#Factoring
fit.PT2 <- principal(PT_Fact_1, nfactors=5, rotate="varimax")
fit.PT2 #2 factors RC1, RC2, RC3, RC4 and RC5 are created

## Principal Components Analysis
## Call: principal(r = PT_Fact_1, nfactors = 5, rotate = "varimax")
## Standardized loadings (pattern matrix) based upon correlation matrix
##                             RC3   RC1   RC2   RC4   RC5   h2    u2 com
## Red.Meat                   0.07  0.96  0.02 -0.02  0.06 0.93 0.071 1.0
## White.Meat                 0.95  0.08 -0.15  0.02  0.13 0.95 0.045 1.1
## Egg                        0.60  0.63  0.07 -0.09  0.20 0.81 0.191 2.3
## Milk                       0.36  0.57  0.36 -0.50 -0.12 0.86 0.144 3.6
## Fish                      -0.13  0.02  0.93  0.12  0.24 0.95 0.049 1.2
## Cereals                   -0.48 -0.48 -0.58  0.05 -0.28 0.88 0.119 3.4
## Starchy.Foods              0.27  0.10  0.30  0.01  0.89 0.97 0.032 1.5
## Pulses.Nuts.and.Oilseeds -0.70 -0.28 -0.20  0.39 -0.27 0.84 0.162 2.5
## Fruits.and.Vegetables    -0.05 -0.03  0.14  0.96  0.00 0.95 0.048 1.1
##
##                        RC3  RC1  RC2  RC4  RC5
## SS loadings           2.22 1.97 1.51 1.36 1.07
## Proportion Var        0.25 0.22 0.17 0.15 0.12
## Cumulative Var        0.25 0.47 0.63 0.78 0.90
## Proportion Explained  0.27 0.24 0.19 0.17 0.13
## Cumulative Proportion 0.27 0.52 0.70 0.87 1.00
##
## Mean item complexity =  2
## Test of the hypothesis that 5 components are sufficient.
```

```
##
## The root mean square of the residuals (RMSR) is  0.04
##  with the empirical chi square  2.82  with prob <  0.093
##
## Fit based upon off diagonal values = 0.99
```

```
round(fit.PT2$values, 3)
```

```
## [1] 4.096 1.625 1.085 0.905 0.427 0.347 0.270 0.135 0.111
```

```
#Above are factor values for all 9 Protein variables
fit.PT2$loadings
```

```
##
## Loadings:
##                            RC3    RC1    RC2    RC4    RC5
## Red.Meat                          0.959
## White.Meat                 0.954         -0.148          0.128
## Egg                        0.597  0.631                  0.197
## Milk                       0.363  0.574  0.363 -0.497 -0.124
## Fish                      -0.129          0.929  0.123  0.237
## Cereals                   -0.481 -0.480 -0.582         -0.279
## Starchy.Foods              0.274          0.303          0.890
## Pulses.Nuts.and.Oilseeds -0.702 -0.276 -0.199  0.395 -0.271
## Fruits.and.Vegetables                    0.143  0.963
##
##                  RC3   RC1   RC2   RC4   RC5
## SS loadings    2.221 1.971 1.512 1.358 1.075
## Proportion Var 0.247 0.219 0.168 0.151 0.119
## Cumulative Var 0.247 0.466 0.634 0.785 0.904
```

```
# Above are the Loadings for all 9 Protein variables
for (i in c(1,2,3,4,5)) { print(fit.PT2$loadings[[1,i]])}
```

```
## [1] 0.0720525
## [1] 0.958774
## [1] 0.01684964
## [1] -0.02340237
## [1] 0.06439545
```

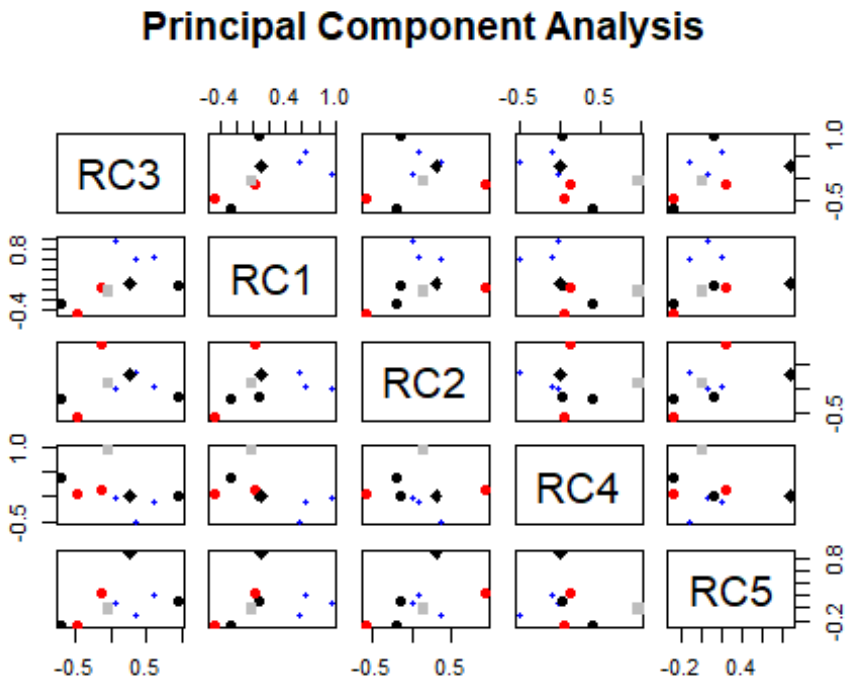
```
#Printing Communalities
fit.PT2$communality
```

```
##                 Red.Meat               White.Meat                      Egg
##                0.9294175                0.9547848                0.8086214
##                     Milk                     Fish                  Cereals
##                0.8555033                0.9505387                0.8812403
##            Starchy.Foods Pulses.Nuts.and.Oilseeds    Fruits.and.Vegetables
##                0.9677946                0.8381162                0.9515011
```

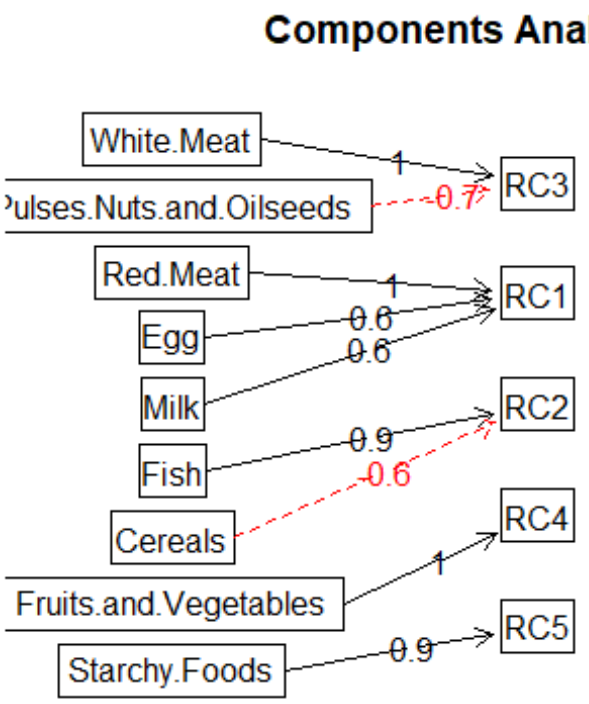
```
#Rotated factor scores
head(fit.PT2$scores)
```

```
##                         RC3         RC1         RC2         RC4         RC5
## Albania          -2.0031994  0.02995195 -1.0471354 -0.86476770 -0.9960020
## Austria           1.8321448 -0.31164548 -0.2939964  0.01354201 -0.6304227
## Belgium          -0.1204075  1.07961592 -0.3337226  0.05997267  1.2382100
## Bulgaria         -0.4714265 -0.70919176 -1.2200866  0.03280737 -1.3560240
## Czechoslovakia    0.7021694 -0.32004879 -0.8604159 -0.02106508  0.7240223
## Denmark           0.7497881  0.01692643  1.5304608 -1.07545985 -0.1605578
```

```
#Plotting
plot(fit.PT2)
```




```
# Plotting the relationship and mapping between variables and factors wit
h weights
fa.diagram(fit.PT2)
```




```
#Above, output gives weigths going in RCs
#Red line indicates negative relation

#Now lets rename these factors as per their contributing variables as pe
r above graph
colnames(fit.PT2$loadings) <- c("WhtMet_NegPulse","RedMet_Egg_Milk","Fish_Neg
Cerl"
                                ,"FrutVeg","StrchFud")
fit.PT2$loadings

##
## Loadings:
##                          WhtMet_NegPulse RedMet_Egg_Milk Fish_NegCerl Frut
Veg
## Red.Meat                                  0.959
## White.Meat                0.954                          -0.148
## Egg                       0.597           0.631
## Milk                      0.363           0.574           0.363       -0.4
97
## Fish                     -0.129                           0.929        0.1
23
## Cereals                  -0.481          -0.480          -0.582
## Starchy.Foods             0.274                           0.303
## Pulses.Nuts.and.Oilseeds -0.702          -0.276          -0.199        0.3
95
```

```
## Fruits.and.Vegetables                                    0.143        0.9
63
##                           StrchFud
## Red.Meat
## White.Meat                 0.128
## Egg                        0.197
## Milk                      -0.124
## Fish                       0.237
## Cereals                   -0.279
## Starchy.Foods              0.890
## Pulses.Nuts.and.Oilseeds -0.271
## Fruits.and.Vegetables
##
##                 WhtMet_NegPulse RedMet_Egg_Milk Fish_NegCerl FrutVeg StrchF
ud
## SS loadings               2.221           1.971        1.512   1.358    1.0
75
## Proportion Var            0.247           0.219        0.168   0.151    0.1
19
## Cumulative Var            0.247           0.466        0.634   0.785    0.9
04
```

#Factor Analysis Conclusion:

#Factor analysis is a technique used to reduce number of columns.
#Factor analysis tries to find if there is any underlying latent variable in your input columns.
#After performing Factor analysis on Protein Consumption dataset, it can be concluded that:
#Based on per personProtein consumption in Europian countries,
#Total 5 factors have been formed with common variance of different Protein sources contributing to them.
#For example, Fish and Cereals contributing to RC2 positively and Negatively respectively.
#As per Above Factors,
#For example, It did form the latent variable using the collinear variables 'RedMeat, Egg and milk;.
#The factors are Renamed accordingly.
#As per above diagram, almost all the factors have significant contribution and so
#Though RC2 and RC3 covering lesser variance, by omitting them I would loose 4 variables
#So, its better not to loose any of 5 factors
#So we will take All 5 Factors, RC1 to RC5 as inputs for our models

#Above factor analysis, we can conclude to reduce number of 9 Protien variables to 5 in our input dataset.

```
##########  Fractal main end &&&&&&&&&&&&&&&&&&&&&&&
```

#Question 2: Cluster Analysis

```
##########  Cluster main start &&&&&&&&&&&&&&&&&&&&&&&
```

#Cluster analysis is a technique that groups the observations into clusters based on similarities
#Clustering types: Hierarchical and nonhierarchical

```
#install.packages("cluster", lib="/Library/Frameworks/R.framework/Versions/3.5/Resources/library")
#library(cluster)

#Creating new input dataframe for cluster analysis
PT_Clust_1 <- read.csv("C:/Alok/OneDrive/Rutgers_MITA/Semester2/MVA/MVA_Midterm_FinalFolder/Protein_Consumption.csv",row.names=1, fill = TRUE)
head(PT_Clust_1)
```

```
##                Red.Meat White.Meat Egg Milk Fish Cereals Starchy.Foods
## Albania              10          1   1    9    0      42             1
## Austria               9         14   4   20    2      28             4
## Belgium              14          9   4   18    5      27             6
## Bulgaria              8          6   2    8    1      57             1
## Czechoslovakia       10         11   3   13    2      34             5
## Denmark              11         11   4   25   10      22             5
##                Pulses.Nuts.and.Oilseeds Fruits.and.Vegetables Total
## Albania                               6                     2    72
## Austria                               1                     4    86
## Belgium                               2                     4    89
## Bulgaria                              4                     4    91
## Czechoslovakia                        1                     4    83
## Denmark                               1                     2    91
```

```
attach(PT_Clust_1)
```

```
## The following objects are masked from PT_Fact_1:
##
##     Cereals, Egg, Fish, Fruits.and.Vegetables, Milk,
##     Pulses.Nuts.and.Oilseeds, Red.Meat, Starchy.Foods, White.Meat

## The following objects are masked from Prot_DS:
##
##     Cereals, Egg, Fish, Fruits.and.Vegetables, Milk,
##     Pulses.Nuts.and.Oilseeds, Red.Meat, Starchy.Foods, Total,
##     White.Meat
```

```r
#Removing Total Column
PT_Clust_1<- PT_Clust_1[,-10]
dim(PT_Clust_1)
```

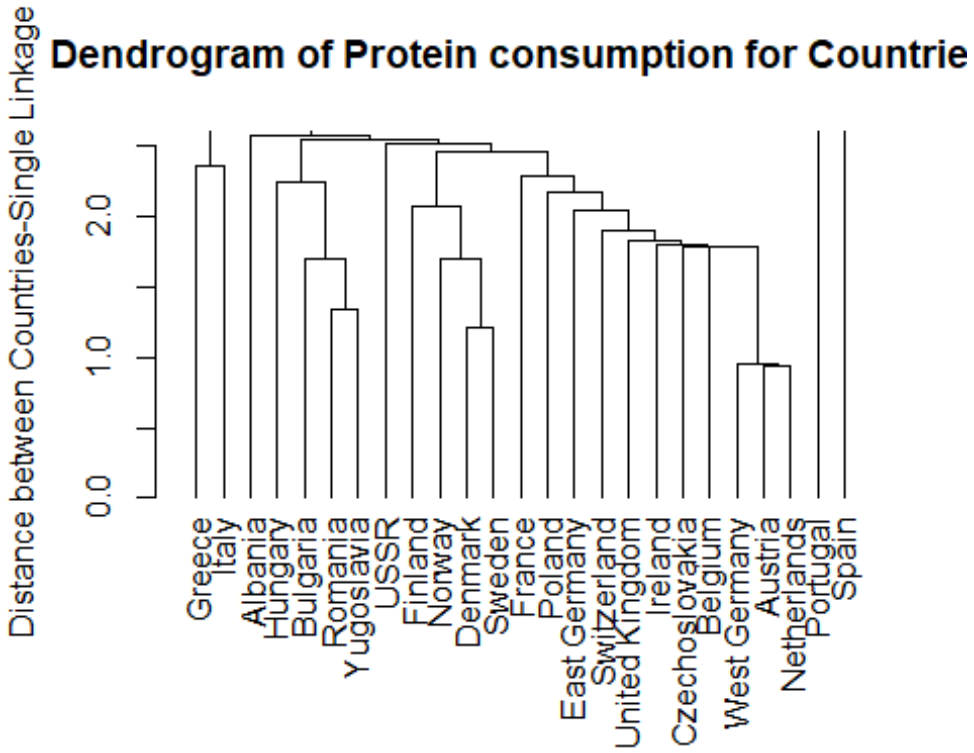
```
## [1] 25  9
```

```r
# Standardizing the data with scale()
matstd.PT <- scale(PT_Clust_1)
head(matstd.PT)
```

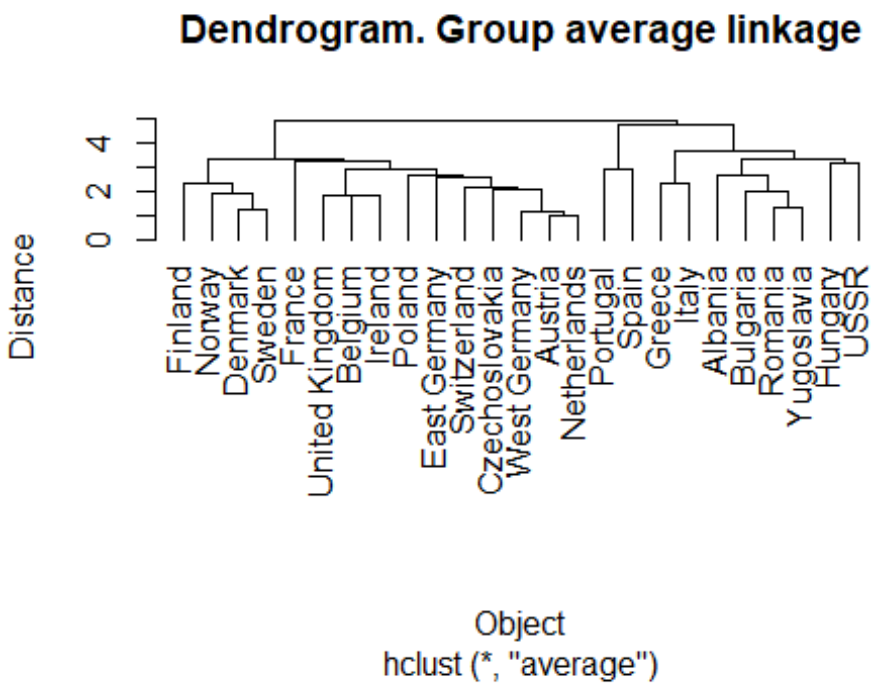
```
##                   Red.Meat White.Meat         Egg       Milk       Fish
## Albania         0.05876425 -1.8498883 -1.86538958 -1.1665829 -1.2333048
## Austria        -0.23505701  1.6253354  0.82507616  0.3832253 -0.6569941
## Belgium         1.23404931  0.2887109  0.82507616  0.1014420  0.2074718
## Bulgaria       -0.52887828 -0.5132638 -0.96856767 -1.3074746 -0.9451495
## Czechoslovakia  0.05876425  0.8233607 -0.07174575 -0.6030163 -0.6569941
## Denmark         0.35258552  0.8233607  0.82507616  1.0876836  1.6482484
##                   Cereals Starchy.Foods Pulses.Nuts.and.Oilseeds
## Albania         0.8791769    -2.0298502                1.4462063
## Austria        -0.3923599    -0.2174840               -1.0301744
## Belgium        -0.4831840     0.9907602               -0.5348982
## Bulgaria        2.2415378    -2.0298502                0.4556540
## Czechoslovakia  0.1525844     0.3866381               -1.0301744
## Denmark        -0.9373043     0.3866381               -1.0301744
##                Fruits.and.Vegetables
## Albania                   -1.1489125
## Austria                   -0.1044466
## Belgium                   -0.1044466
## Bulgaria                  -0.1044466
## Czechoslovakia            -0.1044466
## Denmark                   -1.1489125
```

Hierarchical Clustering

```r
# Creating a (Euclidean) distance matrix of the standardized data
dist.PT_Clust_1 <- dist(matstd.PT, method="euclidean")
# Invoking hclust command (cluster analysis by single linkage method)
clusPT.nn <- hclust(dist.PT_Clust_1, method = "single")
# Plotting vertical dendrogram
# create extra margin room in the dendrogram, on the bottom
par(mar=c(6, 4, 4, 2) + 0.1)
plot(as.dendrogram(clusPT.nn),ylab="Distance between Countries-Single Linkage
",ylim=c(0,2.5),main="Dendrogram of Protein consumption for Countries")
```




```r
#Average
clusPT.avl <- hclust(dist.PT_Clust_1,method="average")
plot(clusPT.avl,hang=-1,xlab="Object",ylab="Distance",
     main="Dendrogram. Group average linkage")
```



#Dendogrm shows that countries have roughly 2 main group which are s ubdivided into smalled grps.

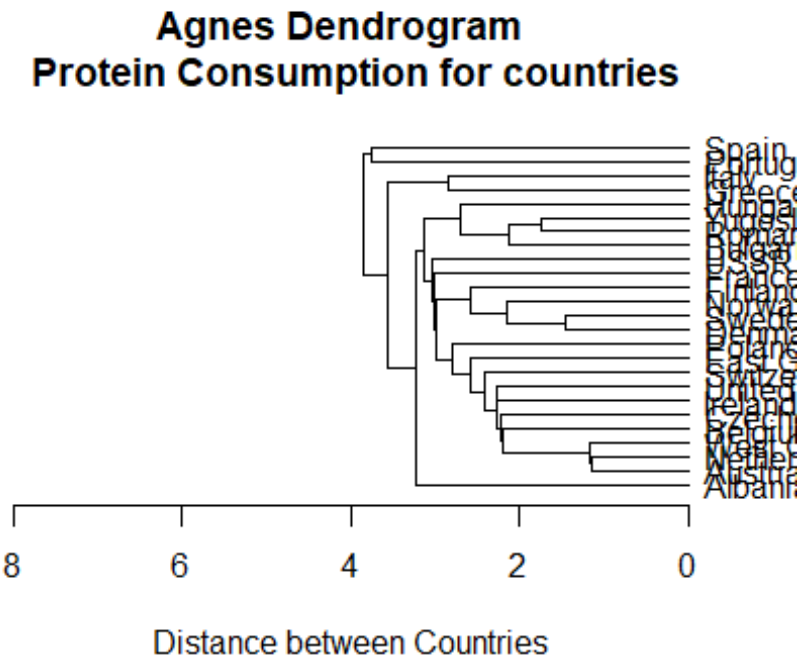
```
#Lazy option --> agnes is 1 liner command for clustering
# We will use agnes function as it allows us to select option for data standa
rdization, the distance measure and clustering algorithm in one single functi
on
(agn.PT <- agnes(PT_Clust_1, metric="euclidean", stand=TRUE, method = "single
"))

## Call:     agnes(x = PT_Clust_1, metric = "euclidean", stand = TRUE, method
= "single")
## Agglomerative coefficient:  0.3894588
## Order of objects:
##  [1] Albania        Austria        Netherlands    West Germany   Belgium
##  [6] Czechoslovakia Ireland        United Kingdom Switzerland    East Germ
any
## [11] Poland         Denmark        Sweden         Norway         Finland
## [16] France         USSR           Bulgaria       Romania        Yugoslavi
a
## [21] Hungary        Greece         Italy          Portugal       Spain
## Height (summary):
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   1.156   2.187   2.582   2.551   3.020   3.839
##
## Available components:
```

```
## [1] "order"     "height"    "ac"        "merge"     "diss"      "call"
## [7] "method"    "order.lab" "data"
```

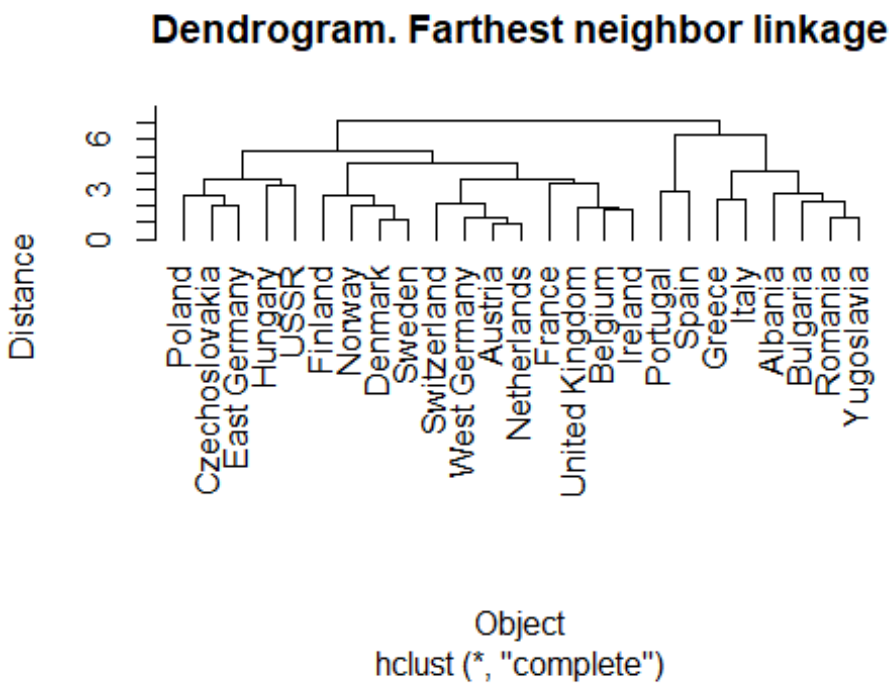
```
#  Description of cluster merging
agn.PT$merge
```

```
##       [,1] [,2]
##  [1,]   -2  -14
##  [2,]    1  -24
##  [3,]   -6  -20
##  [4,]  -18  -25
##  [5,]   -4    4
##  [6,]    3  -15
##  [7,]    2   -3
##  [8,]    7   -5
##  [9,]  -12  -22
## [10,]    8    9
## [11,]   10  -21
## [12,]    6   -8
## [13,]   11   -7
## [14,]    5  -11
## [15,]   13  -16
## [16,]  -10  -13
## [17,]   15   12
## [18,]   17   -9
## [19,]   18  -23
## [20,]   19   14
## [21,]   -1   20
## [22,]   21   16
## [23,]  -17  -19
## [24,]   22   23
```

```
#Dendogram
plot(as.dendrogram(agn.PT), xlab= "Distance between Countries",xlim=c(8,0),
     horiz = TRUE,main="Agnes Dendrogram  \n Protein Consumption for countrie
s")
```




```
#Default - Complete Linkage
clusPT.fn <- hclust(dist.PT_Clust_1)
plot(clusPT.fn,hang=-1,xlab="Object",ylab="Distance",
     main="Dendrogram. Farthest neighbor linkage")
```



#Complete Linkage Dendogram shows that countries have roughly 2 mai n group which are subdivided into smalled grps.
#If you cut at level 4 then you get around 6 Different Clusters of the coun tries.
#If you cut at level 5 then you get around 3 Different Clusters of the coun tries.

Non Hierarchical clustering -- K-Means Clustering######

```
#Creating new input dataframe for cluster analysis
head(PT_Clust_1)
```

```
##                Red.Meat White.Meat Egg Milk Fish Cereals Starchy.Foods
## Albania              10          1   1    9    0      42             1
## Austria               9         14   4   20    2      28             4
## Belgium              14          9   4   18    5      27             6
## Bulgaria              8          6   2    8    1      57             1
## Czechoslovakia       10         11   3   13    2      34             5
## Denmark              11         11   4   25   10      22             5
##                Pulses.Nuts.and.Oilseeds Fruits.and.Vegetables
## Albania                               6                     2
## Austria                               1                     4
## Belgium                               2                     4
## Bulgaria                              4                     4
```

```
## Czechoslovakia                          1                    4
## Denmark                                 1                    2
```

```
dim(PT_Clust_1)
```

```
## [1] 25  9
```

```
# Standardizing the data with scale()
matstd.PT <- scale(PT_Clust_1[,1:9])
head(matstd.PT)
```

```
##                   Red.Meat White.Meat         Egg       Milk       Fish
## Albania         0.05876425 -1.8498883 -1.86538958 -1.1665829 -1.2333048
## Austria        -0.23505701  1.6253354  0.82507616  0.3832253 -0.6569941
## Belgium         1.23404931  0.2887109  0.82507616  0.1014420  0.2074718
## Bulgaria       -0.52887828 -0.5132638 -0.96856767 -1.3074746 -0.9451495
## Czechoslovakia  0.05876425  0.8233607 -0.07174575 -0.6030163 -0.6569941
## Denmark         0.35258552  0.8233607  0.82507616  1.0876836  1.6482484
##                   Cereals Starchy.Foods Pulses.Nuts.and.Oilseeds
## Albania         0.8791769    -2.0298502                1.4462063
## Austria        -0.3923599    -0.2174840               -1.0301744
## Belgium        -0.4831840     0.9907602               -0.5348982
## Bulgaria        2.2415378    -2.0298502                0.4556540
## Czechoslovakia  0.1525844     0.3866381               -1.0301744
## Denmark        -0.9373043     0.3866381               -1.0301744
##                Fruits.and.Vegetables
## Albania                   -1.1489125
## Austria                   -0.1044466
## Belgium                   -0.1044466
## Bulgaria                  -0.1044466
## Czechoslovakia            -0.1044466
## Denmark                   -1.1489125
```

```
#matstd.PT
```

#Implementing K-Means Clustering with different values of k.

```
# K-means, k=2, 3, 4, 5, 6
# Centers (k's) are numbers thus, 10 random sets are chosen
#k=2
(kmeans2.PT <- kmeans(matstd.PT,2,nstart = 10))
```

```
## K-means clustering with 2 clusters of sizes 15, 10
##
## Cluster means:
##    Red.Meat White.Meat        Egg       Milk       Fish    Cereals
## 1  0.470114  0.5203925  0.5859237  0.5804736  0.1306304 -0.6103377
## 2 -0.705171 -0.7805887 -0.8788855 -0.8707105 -0.1959456  0.9155065
##   Starchy.Foods Pulses.Nuts.and.Oilseeds Fruits.and.Vegetables
## 1     0.3866381               -0.6999903            -0.2088932
## 2    -0.5799572                1.0499854             0.3133398
##
```

```
## Clustering vector:
##        Albania        Austria        Belgium       Bulgaria Czechoslovakia
##              2              1              1              2              1
##        Denmark   East Germany        Finland         France         Greece
##              1              1              1              1              2
##        Hungary        Ireland          Italy    Netherlands         Norway
##              2              1              2              1              1
##         Poland       Portugal        Romania          Spain         Sweden
##              1              2              2              2              1
##    Switzerland United Kingdom           USSR   West Germany     Yugoslavia
##              1              1              2              1              2
##
## Within cluster sum of squares by cluster:
## [1] 63.07954 68.74196
##  (between_SS / total_SS =  39.0 %)
##
## Available components:
##
## [1] "cluster"      "centers"      "totss"        "withinss"     "tot.withi
nss"
## [6] "betweenss"    "size"         "iter"         "ifault"
```

```
# Computing the percentage of variation accounted for. Two clusters
perc.var.2 <- round(100*(1 - kmeans2.PT$betweenss/kmeans2.PT$totss),1)
names(perc.var.2) <- "Perc. 2 clus"
perc.var.2
```

```
## Perc. 2 clus
##           61
```

```
#61% variance with k=2

# Computing the percentage of variation accounted for. Three clusters
(kmeans3.PT <- kmeans(matstd.PT,3,nstart = 10))
```

```
## K-means clustering with 3 clusters of sizes 15, 8, 2
##
## Cluster means:
##     Red.Meat White.Meat        Egg       Milk       Fish    Cereals
## 1  0.4701140  0.5203925  0.5859237  0.5804736  0.1306304 -0.6103377
## 2 -0.6390612 -0.6803419 -0.8564649 -0.7262965 -0.6930136  1.2424732
## 3 -0.9696102 -1.1815761 -0.9685677 -1.4483663  1.7923261 -0.3923599
##   Starchy.Foods Pulses.Nuts.and.Oilseeds Fruits.and.Vegetables
## 1     0.3866381               -0.6999903           -0.20889319
## 2    -0.9726366                1.0128397           -0.03916747
## 3     0.9907602                1.1985682            1.72336879
##
## Clustering vector:
##        Albania        Austria        Belgium       Bulgaria Czechoslovakia
##              2              1              1              2              1
##        Denmark   East Germany        Finland         France         Greece
```

```
##             1             1             1             1             2
##       Hungary       Ireland         Italy   Netherlands        Norway
##             2             1             2             1             1
##        Poland      Portugal       Romania         Spain        Sweden
##             1             3             2             3             1
##   Switzerland United Kingdom          USSR  West Germany    Yugoslavia
##             1             1             2             1             2
##
## Within cluster sum of squares by cluster:
## [1] 63.079540 37.801856  4.156111
##  (between_SS / total_SS =  51.4 %)
##
## Available components:
##
## [1] "cluster"      "centers"      "totss"        "withinss"     "tot.withi
nss"
## [6] "betweenss"    "size"         "iter"         "ifault"
```

```
perc.var.3 <- round(100*(1 - kmeans3.PT$betweenss/kmeans3.PT$totss),1)
names(perc.var.3) <- "Perc. 3 clus"
perc.var.3
```

```
## Perc. 3 clus
##         48.6
```

```
#49% variance with k=2

# Computing the percentage of variation accounted for. Four clusters
(kmeans4.PT <- kmeans(matstd.PT,4,nstart = 10))
```

```
## K-means clustering with 4 clusters of sizes 2, 11, 8, 4
##
## Cluster means:
##      Red.Meat White.Meat        Egg       Milk       Fish    Cereals
## 1 -0.96961017 -1.1815761 -0.9685677 -1.4483663  1.7923261 -0.3923599
## 2  0.61969576  0.7747562  0.6620176  0.3063753 -0.2640551 -0.5162109
## 3 -0.63906125 -0.6803419 -0.8564649 -0.7262965 -0.6930136  1.2424732
## 4  0.05876425 -0.1791077  0.3766652  1.3342440  1.2160155 -0.8691863
##   Starchy.Foods Pulses.Nuts.and.Oilseeds Fruits.and.Vegetables
## 1     0.9907602               1.1985682            1.72336879
## 2     0.4415583              -0.6249484            0.13293203
## 3    -0.9726366               1.0128397           -0.03916747
## 4     0.2356076              -0.9063553           -1.14891253
##
## Clustering vector:
##        Albania        Austria        Belgium       Bulgaria Czechoslovakia
##              3              2              2              3              2
##        Denmark   East Germany        Finland         France         Greece
##              4              2              4              2              3
##        Hungary        Ireland          Italy    Netherlands         Norway
##              3              2              3              2              4
```

```
##         Poland       Portugal        Romania          Spain         Sweden
##              2              1              3              1              4
##    Switzerland United Kingdom           USSR   West Germany     Yugoslavia
##              2              2              3              2              3
##
## Within cluster sum of squares by cluster:
## [1]  4.156111 37.920956 37.801856  6.260960
##  (between_SS / total_SS =  60.1 %)
##
## Available components:
##
## [1] "cluster"      "centers"      "totss"        "withinss"     "tot.withi
nss"
## [6] "betweenss"    "size"         "iter"         "ifault"
```

```
perc.var.4 <- round(100*(1 - kmeans4.PT$betweenss/kmeans4.PT$totss),1)
names(perc.var.4) <- "Perc. 4 clus"
perc.var.4
```

```
## Perc. 4 clus
##         39.9
```

```
# Computing the percentage of variation accounted for. Five clusters
(kmeans5.PT <- kmeans(matstd.PT,5,nstart = 10))
```

```
## K-means clustering with 5 clusters of sizes 8, 4, 4, 5, 4
##
## Cluster means:
##      Red.Meat White.Meat         Egg       Milk       Fish    Cereals
## 1  1.01368336  0.7565295  0.82507616  0.5769513 -0.2607806 -0.6875381
## 2  0.05876425 -0.1791077  0.37666520  1.3342440  1.2160155 -0.8691863
## 3 -0.82269954 -0.9142512 -1.41697862 -1.0961371 -1.0171883  1.7192995
## 4 -0.58764253  0.5560358 -0.07174575 -0.4621246 -0.4841009  0.3160677
## 5 -0.52887828 -1.1147448 -0.52015671 -0.8143538  0.9278601  0.1298784
##   Starchy.Foods Pulses.Nuts.and.Oilseeds Fruits.and.Vegetables
## 1     0.2356076               -0.5348982            0.02611165
## 2     0.2356076               -0.9063553           -1.14891253
## 3    -1.4257281                1.0747492           -0.62667956
## 4     0.7491114               -0.3367878            0.10444659
## 5    -0.2174840                1.3223873            1.59281055
##
## Clustering vector:
##        Albania        Austria        Belgium       Bulgaria Czechoslovakia
##              3              1              1              3              4
##        Denmark   East Germany        Finland         France         Greece
##              2              4              2              1              5
##        Hungary        Ireland          Italy    Netherlands         Norway
##              4              1              5              1              2
##         Poland       Portugal        Romania          Spain         Sweden
##              4              5              3              5              2
##    Switzerland United Kingdom           USSR   West Germany     Yugoslavia
```

```
##             1             1             4             1             3
##
## Within cluster sum of squares by cluster:
## [1] 22.498461  6.260960  7.764725 16.446716 20.196237
##  (between_SS / total_SS =  66.1 %)
##
## Available components:
##
## [1] "cluster"      "centers"      "totss"        "withinss"     "tot.withi
nss"
## [6] "betweenss"    "size"         "iter"         "ifault"
```

```
perc.var.5 <- round(100*(1 - kmeans5.PT$betweenss/kmeans5.PT$totss),1)
names(perc.var.5) <- "Perc. 5 clus"
perc.var.5
```

```
## Perc. 5 clus
##         33.9
```

```
(kmeans6.PT <- kmeans(matstd.PT,6,nstart = 10))
```

```
## K-means clustering with 6 clusters of sizes 4, 8, 2, 4, 5, 2
##
## Cluster means:
##      Red.Meat White.Meat         Egg       Milk        Fish    Cereals
## 1  0.05876425 -0.1791077  0.37666520  1.3342440  1.21601546 -0.8691863
## 2  1.01368336  0.7565295  0.82507616  0.5769513 -0.26078057 -0.6875381
## 3 -0.08814638 -1.0479136 -0.07174575 -0.1803413  0.06339417  0.6521168
## 4 -0.82269954 -0.9142512 -1.41697862 -1.0961371 -1.01718829  1.7192995
## 5 -0.58764253  0.5560358 -0.07174575 -0.4621246 -0.48410094  0.3160677
## 6 -0.96961017 -1.1815761 -0.96856767 -1.4483663  1.79232611 -0.3923599
##   Starchy.Foods Pulses.Nuts.and.Oilseeds Fruits.and.Vegetables
## 1     0.2356076               -0.9063553           -1.14891253
## 2     0.2356076               -0.5348982            0.02611165
## 3    -1.4257281                1.4462063            1.46225231
## 4    -1.4257281                1.0747492           -0.62667956
## 5     0.7491114               -0.3367878            0.10444659
## 6     0.9907602                1.1985682            1.72336879
##
## Clustering vector:
##       Albania       Austria       Belgium      Bulgaria Czechoslovakia
##             4             2             2             4             5
##       Denmark   East Germany       Finland        France        Greece
##             1             5             1             2             3
##       Hungary       Ireland         Italy   Netherlands        Norway
##             5             2             3             2             1
##        Poland      Portugal       Romania         Spain        Sweden
##             5             6             4             6             1
##   Switzerland United Kingdom          USSR  West Germany     Yugoslavia
##             2             2             5             2             4
##
```

```
## Within cluster sum of squares by cluster:
## [1]  6.260960 22.498461  2.784045  7.764725 16.446716  4.156111
##  (between_SS / total_SS =  72.3 %)
##
## Available components:
##
## [1] "cluster"      "centers"      "totss"        "withinss"     "tot.withi
nss"
## [6] "betweenss"    "size"         "iter"         "ifault"

# Computing the percentage of variation accounted for. Six clusters
perc.var.6 <- round(100*(1 - kmeans6.PT$betweenss/kmeans6.PT$totss),1)
names(perc.var.6) <- "Perc. 6 clus"
perc.var.6

## Perc. 6 clus
##         27.7

#
(kmeans7.PT <- kmeans(matstd.PT,7,nstart = 10))

## K-means clustering with 7 clusters of sizes 2, 4, 4, 4, 4, 2, 5
##
## Cluster means:
##      Red.Meat White.Meat         Egg       Milk        Fish    Cereals
## 1 -0.96961017 -1.1815761 -0.96856767 -1.4483663  1.79232611 -0.3923599
## 2  0.27913020  1.2911793  0.60087068  0.5945628 -0.51291647 -0.7783622
## 3 -0.82269954 -0.9142512 -1.41697862 -1.0961371 -1.01718829  1.7192995
## 4  1.74823652  0.2218797  1.04928164  0.5593399 -0.00864466 -0.5967141
## 5  0.05876425 -0.1791077  0.37666520  1.3342440  1.21601546 -0.8691863
## 6 -0.08814638 -1.0479136 -0.07174575 -0.1803413  0.06339417  0.6521168
## 7 -0.58764253  0.5560358 -0.07174575 -0.4621246 -0.48410094  0.3160677
##   Starchy.Foods Pulses.Nuts.and.Oilseeds Fruits.and.Vegetables
## 1     0.9907602                1.1985682             1.72336879
## 2    -0.2174840               -0.6587173             0.02611165
## 3    -1.4257281                1.0747492            -0.62667956
## 4     0.6886992               -0.4110792             0.02611165
## 5     0.2356076               -0.9063553            -1.14891253
## 6    -1.4257281                1.4462063             1.46225231
## 7     0.7491114               -0.3367878             0.10444659
##
## Clustering vector:
##        Albania        Austria        Belgium       Bulgaria Czechoslovakia
##              3              2              4              3              7
##        Denmark   East Germany        Finland         France         Greece
##              5              7              5              4              6
##        Hungary        Ireland          Italy    Netherlands         Norway
##              7              4              6              2              5
##         Poland       Portugal        Romania          Spain         Sweden
##              7              1              3              1              5
##    Switzerland United Kingdom           USSR   West Germany     Yugoslavia
```

```
##              2              4              7              2              3
##
## Within cluster sum of squares by cluster:
## [1]  4.156111  4.067913  7.764725  9.083016  6.260960  2.784045 16.446716
##  (between_SS / total_SS =  76.6 %)
##
## Available components:
##
## [1] "cluster"      "centers"      "totss"        "withinss"     "tot.withi
nss"
## [6] "betweenss"    "size"         "iter"         "ifault"
```

```
# Computing the percentage of variation accounted for. Six clusters
perc.var.7 <- round(100*(1 - kmeans7.PT$betweenss/kmeans7.PT$totss),1)
names(perc.var.7) <- "Perc. 7 clus"
perc.var.7
```

```
## Perc. 7 clus
##         23.4
```

#It can be seen that Variance goes down as K increases...

#To Identify the Best number of K Clusters, plotting Elbow Plot

```
wss=c()########## empty vector to hold wss
for(i in 2:10)#### from 2 to 10 cluster
{
  km = kmeans(matstd.PT[,1:9],i)
  wss[i-1]=km$tot.withinss
}
wss
```

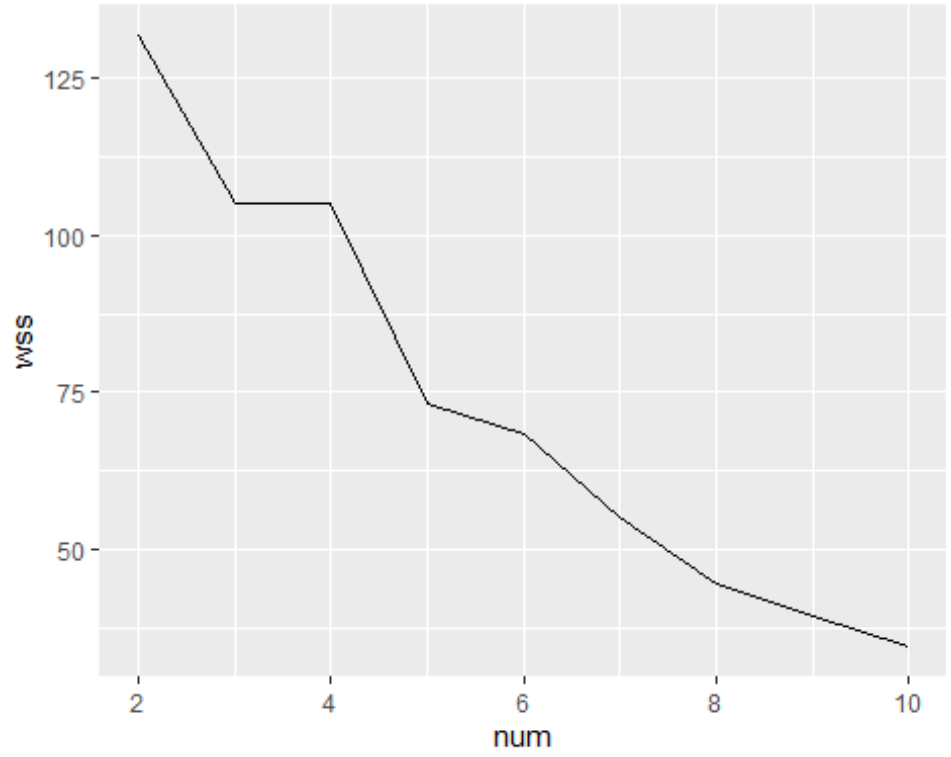
```
## [1] 131.82150 105.17225 105.21957  73.26291  68.34571  55.08963  44.49715
## [8]  39.24410  34.60564
```

```
#Creating a 'elbowdt' data table with column names num and wss with the conte
nts of wss
elbowdt = data.table(num=2:10,wss)
elbowdt
```

```
##    num       wss
## 1:   2 131.82150
## 2:   3 105.17225
## 3:   4 105.21957
## 4:   5  73.26291
## 5:   6  68.34571
## 6:   7  55.08963
## 7:   8  44.49715
```

```
## 8:   9  39.24410
## 9:  10  34.60564
```

```
#Plotting
ggplot(elbowdt,aes(x=num,y=wss)) + geom_line()
```



#For k = 3 the between sum of square/total sum of square ratio tends to change slowly
#and remain less changing as compared to others.
#Also this dataset has only 25 rows so more than 3/4 clusters would not make much sense to me.
#Therefore, k = 3 should be a good choice for the number of clu sters.
#For 3 clusters, k-means = 3

```
# Centers (k's) are numbers thus, 10 random sets are chosen
(kmeans3.PT <- kmeans(matstd.PT,3,nstart = 10))
```

```
## K-means clustering with 3 clusters of sizes 15, 2, 8
##
## Cluster means:
##     Red.Meat White.Meat        Egg       Milk       Fish    Cereals
## 1  0.4701140  0.5203925  0.5859237  0.5804736  0.1306304 -0.6103377
## 2 -0.9696102 -1.1815761 -0.9685677 -1.4483663  1.7923261 -0.3923599
## 3 -0.6390612 -0.6803419 -0.8564649 -0.7262965 -0.6930136  1.2424732
##   Starchy.Foods Pulses.Nuts.and.Oilseeds Fruits.and.Vegetables
```

```
## 1     0.3866381               -0.6999903           -0.20889319
## 2     0.9907602                1.1985682            1.72336879
## 3    -0.9726366                1.0128397           -0.03916747
##
## Clustering vector:
##        Albania        Austria        Belgium       Bulgaria Czechoslovakia
##              3              1              1              3              1
##        Denmark   East Germany        Finland         France         Greece
##              1              1              1              1              3
##        Hungary        Ireland          Italy    Netherlands         Norway
##              3              1              3              1              1
##         Poland       Portugal        Romania          Spain         Sweden
##              1              2              3              2              1
##    Switzerland United Kingdom           USSR   West Germany     Yugoslavia
##              1              1              3              1              3
##
## Within cluster sum of squares by cluster:
## [1] 63.079540  4.156111 37.801856
##  (between_SS / total_SS =  51.4 %)
##
## Available components:
##
## [1] "cluster"      "centers"      "totss"        "withinss"     "tot.withi
nss"
## [6] "betweenss"    "size"         "iter"         "ifault"
```

```
perc.var.3 <- round(100*(1 - kmeans3.PT$betweenss/kmeans3.PT$totss),1)
names(perc.var.3) <- "Perc. 3 clus"
perc.var.3
```

```
## Perc. 3 clus
##         48.6
```

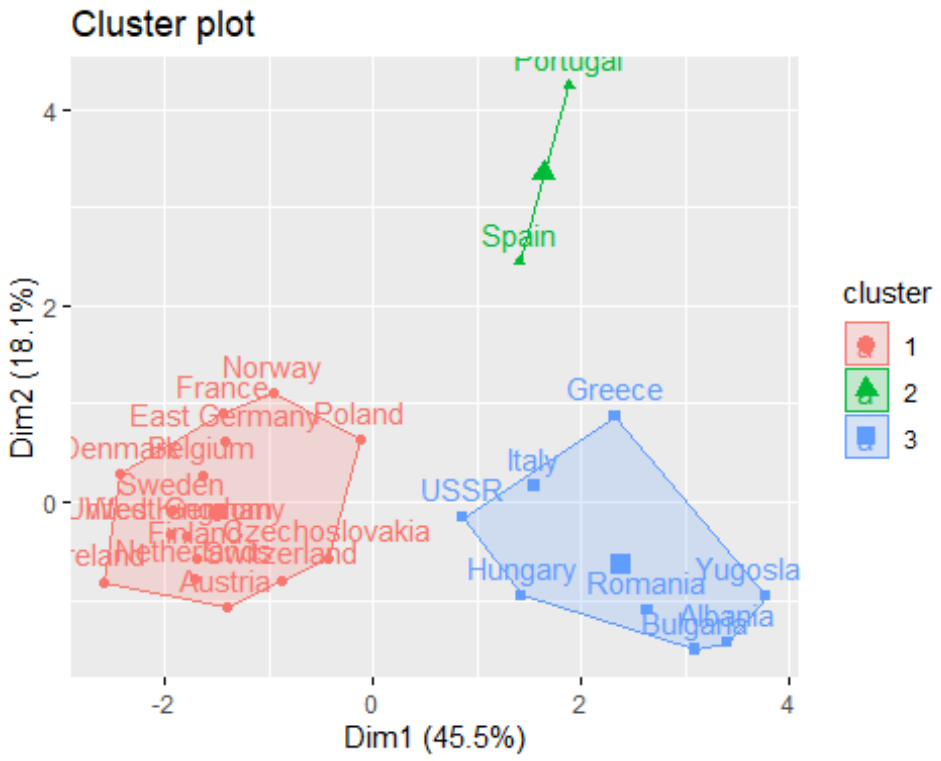
```
kmeans3.PT
```

```
## K-means clustering with 3 clusters of sizes 15, 2, 8
##
## Cluster means:
##     Red.Meat White.Meat        Egg       Milk       Fish    Cereals
## 1  0.4701140  0.5203925  0.5859237  0.5804736  0.1306304 -0.6103377
## 2 -0.9696102 -1.1815761 -0.9685677 -1.4483663  1.7923261 -0.3923599
## 3 -0.6390612 -0.6803419 -0.8564649 -0.7262965 -0.6930136  1.2424732
##   Starchy.Foods Pulses.Nuts.and.Oilseeds Fruits.and.Vegetables
## 1     0.3866381               -0.6999903           -0.20889319
## 2     0.9907602                1.1985682            1.72336879
## 3    -0.9726366                1.0128397           -0.03916747
##
## Clustering vector:
##        Albania        Austria        Belgium       Bulgaria Czechoslovakia
##              3              1              1              3              1
##        Denmark   East Germany        Finland         France         Greece
```

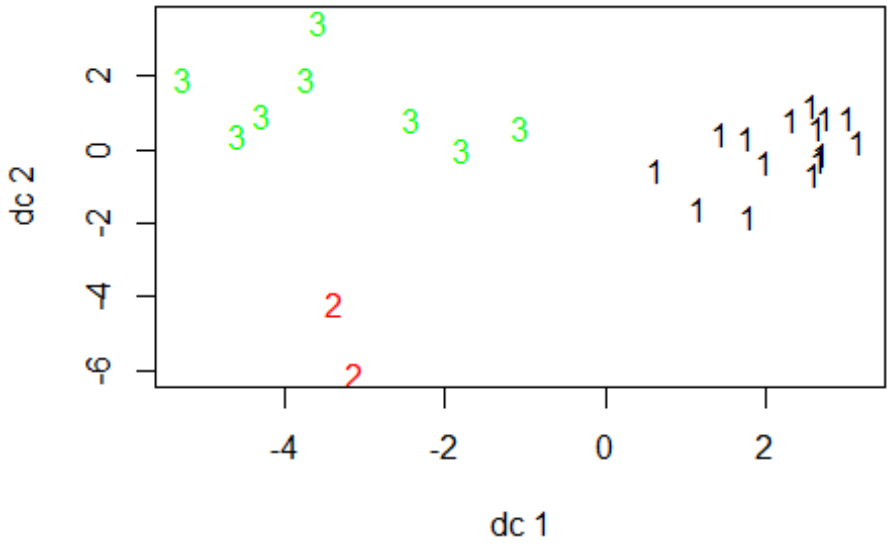
```
##              1              1              1              1              3
##        Hungary        Ireland          Italy    Netherlands         Norway
##              3              1              3              1              1
##         Poland       Portugal        Romania          Spain         Sweden
##              1              2              3              2              1
##    Switzerland United Kingdom           USSR   West Germany     Yugoslavia
##              1              1              3              1              3
##
## Within cluster sum of squares by cluster:
## [1] 63.079540  4.156111 37.801856
##  (between_SS / total_SS =  51.4 %)
##
## Available components:
##
## [1] "cluster"      "centers"      "totss"        "withinss"     "tot.withi
nss"
## [6] "betweenss"    "size"         "iter"         "ifault"

kmeans3.PT$cluster

##        Albania        Austria        Belgium       Bulgaria Czechoslovakia
##              3              1              1              3              1
##        Denmark   East Germany        Finland         France         Greece
##              1              1              1              1              3
##        Hungary        Ireland          Italy    Netherlands         Norway
##              3              1              3              1              1
##         Poland       Portugal        Romania          Spain         Sweden
##              1              2              3              2              1
##    Switzerland United Kingdom           USSR   West Germany     Yugoslavia
##              1              1              3              1              3
```

#plotting output of kmeans for 3 clusters

```
fviz_cluster(kmeans3.PT,data=matstd.PT)
```




```
#Clusters plotting in another way to see them more clearly
plotcluster(matstd.PT,kmeans3.PT$cluster)
```



#Creating separate matrices for clusters

```
clus1 <- matrix(names(kmeans3.PT$cluster[kmeans3.PT$cluster == 1]),
                ncol=1, nrow=length(kmeans3.PT$cluster[kmeans3.PT$cluster ==
1]))
colnames(clus1) <- "Cluster 1"
clus1
```

```
##       Cluster 1
##  [1,] "Austria"
##  [2,] "Belgium"
##  [3,] "Czechoslovakia"
##  [4,] "Denmark"
##  [5,] "East Germany"
##  [6,] "Finland"
##  [7,] "France"
##  [8,] "Ireland"
##  [9,] "Netherlands"
## [10,] "Norway"
## [11,] "Poland"
## [12,] "Sweden"
## [13,] "Switzerland"
## [14,] "United Kingdom"
## [15,] "West Germany"
```

```
clus2 <- matrix(names(kmeans3.PT$cluster[kmeans3.PT$cluster == 2]),
                ncol=1, nrow=length(kmeans3.PT$cluster[kmeans3.PT$cluster ==
2]))
colnames(clus2) <- "Cluster 2"
clus2
```

```
##      Cluster 2
## [1,] "Portugal"
## [2,] "Spain"
```

```
clus3 <- matrix(names(kmeans3.PT$cluster[kmeans3.PT$cluster == 3]),
                ncol=1, nrow=length(kmeans3.PT$cluster[kmeans3.PT$cluster ==
3]))
colnames(clus3) <- "Cluster 3"
clus3
```

```
##      Cluster 3
## [1,] "Albania"
## [2,] "Bulgaria"
## [3,] "Greece"
## [4,] "Hungary"
## [5,] "Italy"
## [6,] "Romania"
## [7,] "USSR"
## [8,] "Yugoslavia"
```

```
#clus4 <- matrix(names(kmeans4.PT$cluster[kmeans4.PT$cluster == 4]),
#                ncol=1, nrow=length(kmeans4.PT$cluster[kmeans4.PT$cluster ==
4]))
#colnames(clus4) <- "Cluster 4"
#clus4

#clus5 <- matrix(names(kmeans5.PT$cluster[kmeans5.PT$cluster == 5]),
#                ncol=1, nrow=length(kmeans5.PT$cluster[kmeans5.PT$cluster ==
5]))
#colnames(clus5) <- "Cluster 5"
#clus5

#clus6 <- matrix(names(kmeans6.PT$cluster[kmeans6.PT$cluster == 6]),
#                ncol=1, nrow=length(kmeans6.PT$cluster[kmeans6.PT$cluster ==
6]))
#colnames(clus6) <- "Cluster 6"
#clus6

#clus7 <- matrix(names(kmeans7.PT$cluster[kmeans7.PT$cluster == 7]),
#                ncol=1, nrow=length(kmeans7.PT$cluster[kmeans7.PT$cluster ==
7]))
#colnames(clus7) <- "Cluster 7"
#clus7
```

#Displaying all the Countries in their respective clusters

```
list(clus1,clus2,clus3)
```

```
## [[1]]
##       Cluster 1
##  [1,] "Austria"
##  [2,] "Belgium"
##  [3,] "Czechoslovakia"
##  [4,] "Denmark"
##  [5,] "East Germany"
##  [6,] "Finland"
##  [7,] "France"
##  [8,] "Ireland"
##  [9,] "Netherlands"
## [10,] "Norway"
## [11,] "Poland"
## [12,] "Sweden"
## [13,] "Switzerland"
## [14,] "United Kingdom"
## [15,] "West Germany"
##
## [[2]]
##      Cluster 2
## [1,] "Portugal"
## [2,] "Spain"
```

```
##
## [[3]]
##      Cluster 3
## [1,] "Albania"
## [2,] "Bulgaria"
## [3,] "Greece"
## [4,] "Hungary"
## [5,] "Italy"
## [6,] "Romania"
## [7,] "USSR"
## [8,] "Yugoslavia"
```

```
#Making Subsets for 3 clusters using Row filtering from the Original dataset
#(Not the scaled one)
```

#So below are the 3 cluster sets of Original entire dataset

```
#BF_Clust_1
#Using original dataframe to capture Country names as clusters have been made
based on country
names
```

```
## function (x)  .Primitive("names")
```

```
head(Prot_DS)
```

```
##          Country Red.Meat White.Meat Egg Milk Fish Cereals Starchy.Foods
## 1        Albania       10          1   1    9    0      42             1
## 2        Austria        9         14   4   20    2      28             4
## 3        Belgium       14          9   4   18    5      27             6
## 4       Bulgaria        8          6   2    8    1      57             1
## 5 Czechoslovakia       10         11   3   13    2      34             5
## 6        Denmark       11         11   4   25   10      22             5
##   Pulses.Nuts.and.Oilseeds Fruits.and.Vegetables Total
## 1                        6                     2    72
## 2                        1                     4    86
## 3                        2                     4    89
## 4                        4                     4    91
## 5                        1                     4    83
## 6                        1                     2    91
```

```
PT_Cl1_Dt<-subset(Prot_DS,Prot_DS$Country %in% clus1)
#PT_Cl1_Dt
PT_Cl2_Dt<-subset(Prot_DS,Prot_DS$Country %in% clus2)
#PT_Cl2_Dt
PT_Cl3_Dt<-subset(Prot_DS,Prot_DS$Country %in% clus3)
#PT_Cl3_Dt
#PT_Cl4_Dt<-subset(Prot_DS,Prot_DS$Country %in% clus4)
#PT_Cl4_Dt
#PT_Cl5_Dt<-subset(Prot_DS,Prot_DS$Country %in% clus5)
#PT_Cl5_Dt
#PT_Cl6_Dt<-subset(Prot_DS,Prot_DS$Country %in% clus6)
#PT_Cl6_Dt
```

```
#PT_Cl7_Dt<-subset(Prot_DS,Prot_DS$Country %in% clus7)
#PT_Cl7_Dt
```

#Printing all the columns of the Clusters formed
#Original observations after clustering with all the variables

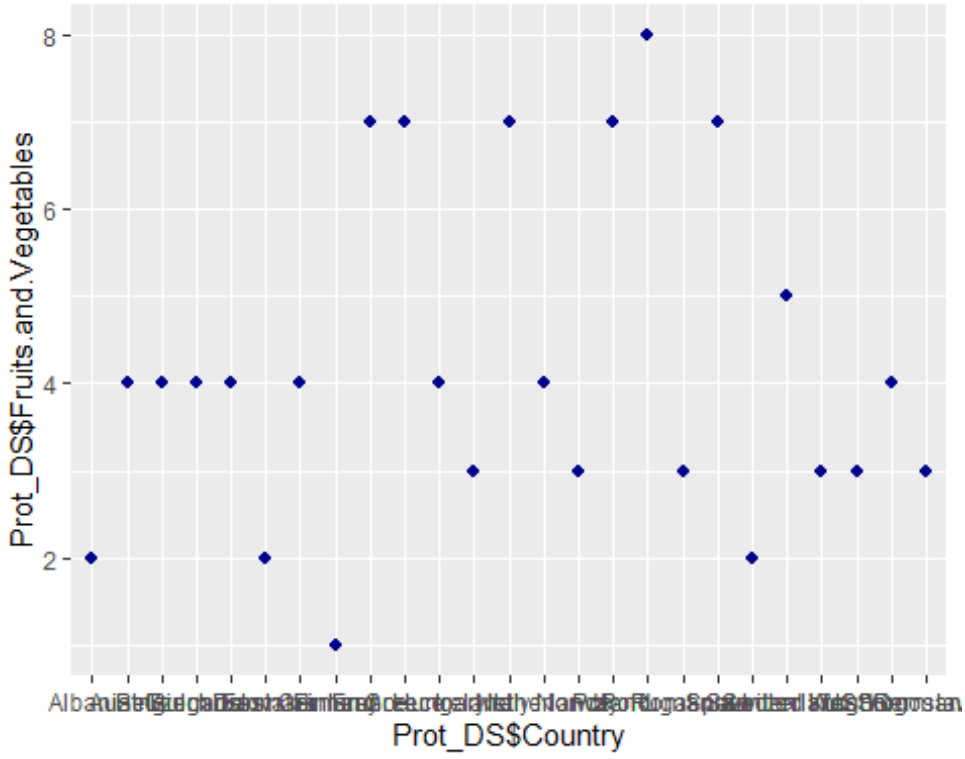
```
list(PT_Cl1_Dt,PT_Cl2_Dt,PT_Cl3_Dt)
```

```
## [[1]]
##           Country Red.Meat White.Meat Egg Milk Fish Cereals Starchy.Foods
## 2         Austria        9         14   4   20    2      28             4
## 3         Belgium       14          9   4   18    5      27             6
## 5  Czechoslovakia       10         11   3   13    2      34             5
## 6         Denmark       11         11   4   25   10      22             5
## 7    East Germany        8         12   4   11    5      25             7
## 8         Finland       10          5   3   34    6      26             5
## 9          France       18         10   3   20    6      28             5
## 12        Ireland       14         10   5   26    2      24             6
## 14    Netherlands       10         14   4   23    3      22             4
## 15         Norway        9          5   3   23   10      23             5
## 16         Poland        7         10   3   19    3      36             6
## 20         Sweden       10          8   4   25    8      20             4
## 21    Switzerland       13         10   3   24    2      26             3
## 22 United Kingdom       17          6   5   21    4      24             5
## 24   West Germany       11         13   4   19    3      19             5
##    Pulses.Nuts.and.Oilseeds Fruits.and.Vegetables Total
## 2                         1                     4    86
## 3                         2                     4    89
## 5                         1                     4    83
## 6                         1                     2    91
## 7                         1                     4    77
## 8                         1                     1    91
## 9                         2                     7    99
## 12                        2                     3    92
## 14                        2                     4    86
## 15                        2                     3    83
## 16                        2                     7    93
## 20                        1                     2    82
## 21                        2                     5    88
## 22                        3                     3    88
## 24                        2                     4    80
##
## [[2]]
##     Country Red.Meat White.Meat Egg Milk Fish Cereals Starchy.Foods
## 17 Portugal        6          4   1    5   14      27             6
## 19    Spain        7          3   3    9    7      29             6
##    Pulses.Nuts.and.Oilseeds Fruits.and.Vegetables Total
## 17                        5                     8    76
## 19                        6                     7    77
##
```

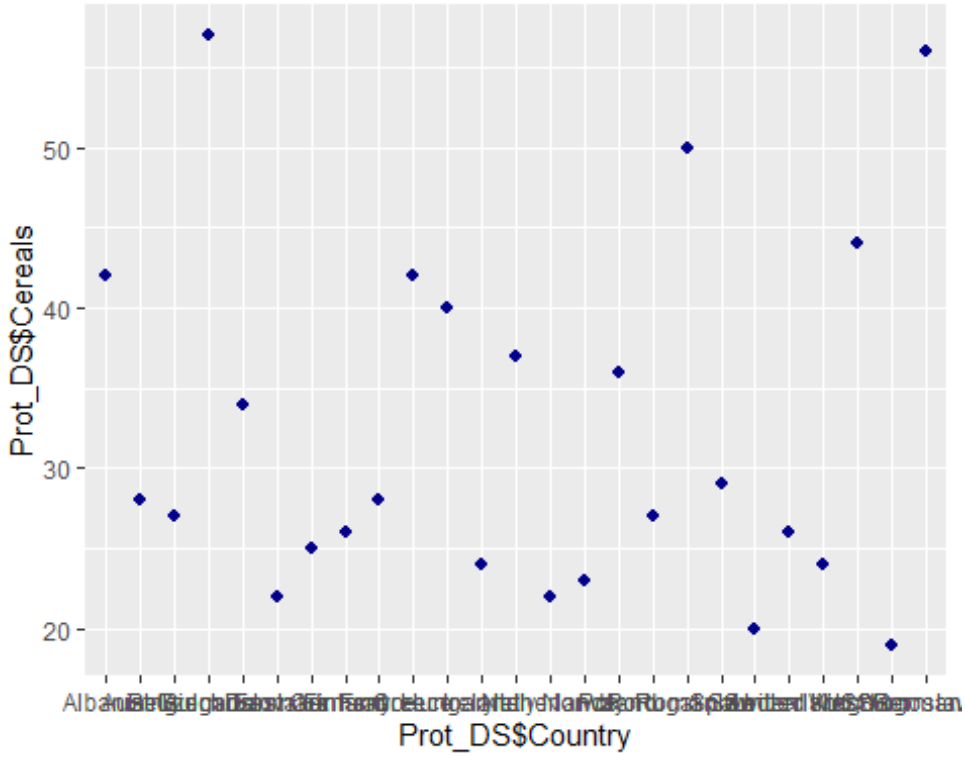
```
## [[3]]
##       Country Red.Meat White.Meat Egg Milk Fish Cereals Starchy.Foods
## 1     Albania       10          1   1    9    0      42             1
## 4    Bulgaria        8          6   2    8    1      57             1
## 10     Greece       10          3   3   18    6      42             2
## 11    Hungary        5         12   3   10    0      40             4
## 13      Italy        9          5   3   14    3      37             2
## 18    Romania        6          6   2   11    1      50             3
## 23       USSR        9          5   2   17    3      44             6
## 25 Yugoslavia        4          5   1   10    1      56             3
##    Pulses.Nuts.and.Oilseeds Fruits.and.Vegetables Total
## 1                         6                     2    72
## 4                         4                     4    91
## 10                        8                     7    99
## 11                        5                     4    83
## 13                        4                     7    84
## 18                        5                     3    87
## 23                        3                     3    92
## 25                        6                     3    89
```

#Trying to plot different variables to see if I can spot any groups within them.

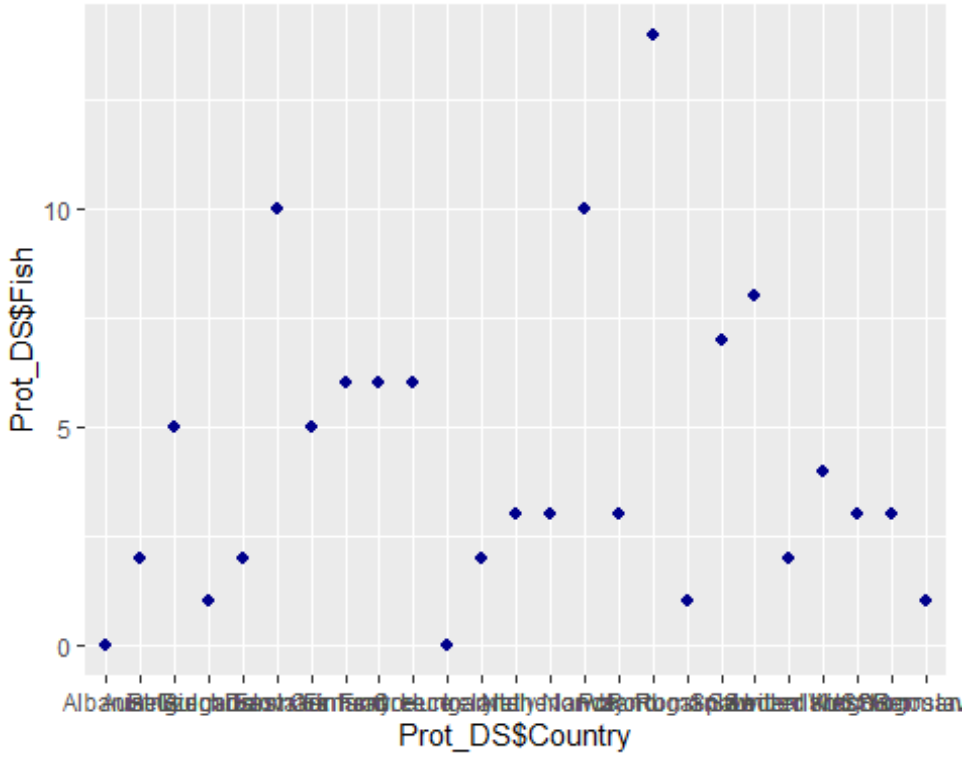
```
#Plotting observations against Fruits and vegetables
ggplot(Prot_DS,
       aes(x=Prot_DS$Country,y=Prot_DS$Fruits.and.Vegetables))+
  geom_point(size=1.8,color='dark blue')
```



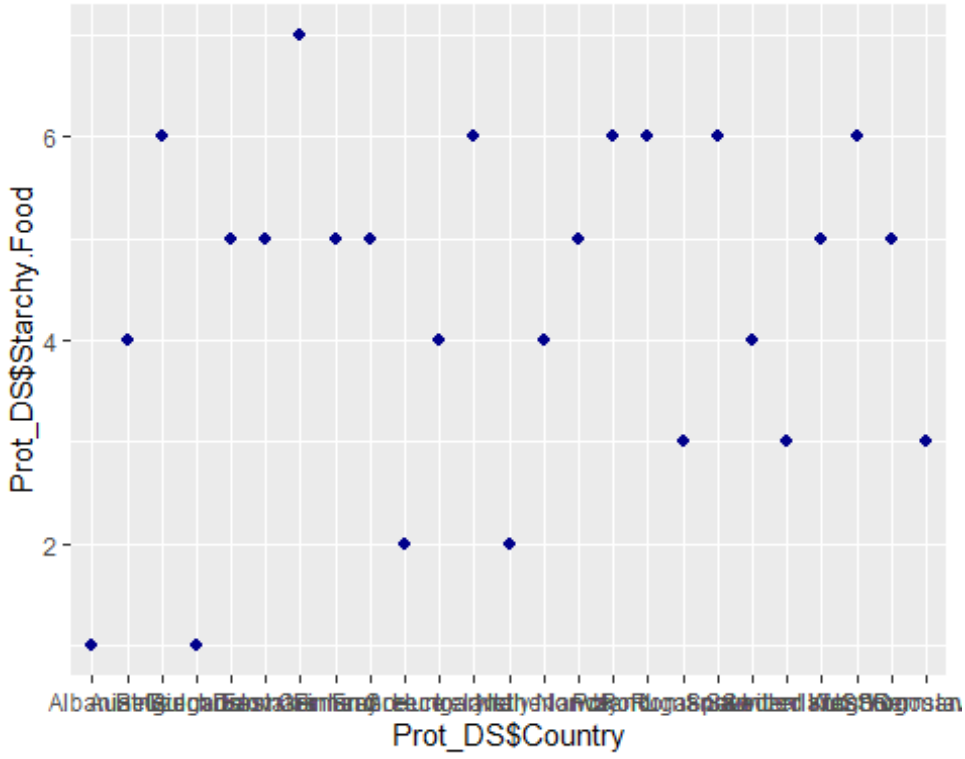

```
#Plotting observations against Cereals
ggplot(Prot_DS,
       aes(x=Prot_DS$Country,y=Prot_DS$Cereals))+
  geom_point(size=1.8,color='dark blue')
```




```
#Plotting Scatter plotss with different variables to check on what basis clus
ters may have formed.
#Plotting observations against Fish
ggplot(Prot_DS,
       aes(x=Prot_DS$Country,y=Prot_DS$Fish))+
  geom_point(size=1.8,color='dark blue')
```




```
#Plotting observations against Cereals
ggplot(Prot_DS,
       aes(x=Prot_DS$Country,y=Prot_DS$Starchy.Food))+
  geom_point(size=1.8,color='dark blue')
```



#As per above 3 clusters formed based on similarities of per pe rson Protien consumprion for Europian countries,
#I can roughly see that
#Clustering might have formed based on variables 'Fruits and Veg' , Fish and upto some extent
#based on Cereals and starchy foods.
#Because There seems to be 3 different ranges of these variables in the a llocated 3 clusters.
#So roughly, there are 3 groups formed manily based on the similarities of the per person Protein consumption
#through intake of Fruits and Veg, Fish, Cereals and Starchy foods.
#Hierarchical and non hierarchical clustering has been performed on Pr otein consumptiondataset above.

```
##&&&&&&&&&& Cluster main end &&&&&&&&&&&&&&&&&&&&&&&&
```