MVA Midterm

```
#installing all the required packages
install.packages("knitr")
```

```
## Installing package into 'C:/Users/vidhi/Documents/R/win-library/3.6'
## (as 'lib' is unspecified)
```

```
## Error in contrib.url(repos, "source"): trying to use CRAN without setting a mirror
```

```
library(knitr)
install.packages("rmarkdown")
```

```
## Installing package into 'C:/Users/vidhi/Documents/R/win-library/3.6'
## (as 'lib' is unspecified)
```

```
## Error in contrib.url(repos, "source"): trying to use CRAN without setting a mirror
```

```
library(rmarkdown)
```

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

```
install.packages("ggplot2")
```

```
## Installing package into 'C:/Users/vidhi/Documents/R/win-library/3.6'
## (as 'lib' is unspecified)
```

```
## Error in contrib.url(repos, "source"): trying to use CRAN without setting a mirror
```

```
library(ggplot2)
```

```
## Warning: package 'ggplot2' was built under R version 3.6.3
```

```
install.packages("factoextra")
```

```
## Installing package into 'C:/Users/vidhi/Documents/R/win-library/3.6'
## (as 'lib' is unspecified)
```

```
## Error in contrib.url(repos, "source"): trying to use CRAN without setting a mirror
```

```
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://goo.gl/ve3WBa
```

```
install.packages("dplyr")
```

```
## Installing package into 'C:/Users/vidhi/Documents/R/win-library/3.6'
## (as 'lib' is unspecified)
```

```
## Error in contrib.url(repos, "source"): trying to use CRAN without setting a mirror
```

```
library(dplyr)
```

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

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

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

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

```
install.packages("GGally")
```

```
## Installing package into 'C:/Users/vidhi/Documents/R/win-library/3.6'
## (as 'lib' is unspecified)
```

```
## Error in contrib.url(repos, "source"): trying to use CRAN without setting a mirror
```

```r
library(GGally)
```

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

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

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

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

```
## Warning in install.packages("cluster", lib = "/Library/Frameworks/R.framework/
## Versions/3.5/Resources/library"): 'lib = "/Library/Frameworks/R.framework/
## Versions/3.5/Resources/library"' is not writable
```

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

```r
library(cluster)
```

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

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

```
## Warning in install.packages("psych", lib = "/Library/Frameworks/R.framework/
## Versions/3.5/Resources/library"): 'lib = "/Library/Frameworks/R.framework/
## Versions/3.5/Resources/library"' is not writable
```

```
## Error in install.packages("psych", lib = "/Library/Frameworks/R.framework/Versions/3.5/Resources/library"): unable to install packages
```

```r
library(psych)
```

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

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

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

```r
############# reading the Protein Consumption dataset and attaching it #############
data<-read.csv("Protein_Consumption.csv", fill = TRUE)
#fill = True is added so that if there are rows which have unequal lengths or there is some missing data then it will fill implicitly
attach(data)

##### Check for the dimensions of the dataset attached ###########
dim(data)
```

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

```r
#Ans- There are 25 observations and 11 variables
head(data)
```

```
##       ï..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
```

```r
tail(data)
```

```
##        ï..Country Red.Meat White.Meat Egg Milk Fish Cereals Starchy.Foods
## 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
## 23           USSR        9          5   2   17    3      44             6
## 24   West Germany       11         13   4   19    3      19             5
## 25     Yugoslavia        4          5   1   10    1      56             3
##    Pulses.Nuts.and.Oilseeds Fruits.and.Vegetables Total
## 20                        1                     2    82
## 21                        2                     5    88
## 22                        3                     3    88
## 23                        3                     3    92
## 24                        2                     4    80
## 25                        6                     3    89
```

```
######################################################################
```

```
# Q1) Use principal components analysis to investigate the relationships between the countries on the basis of these variables

############## Performing PCA ##############################
View(data)
cor(data[-1])
```

```
##                            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
## Total                    0.37369919  0.10308602  0.18970028  0.4603542
##                                Fish     Cereals Starchy.Foods
## Red.Meat                 0.06491072 -0.50970337    0.15383673
## White.Meat              -0.19719960 -0.43941908    0.33456770
## Egg                      0.04780844 -0.70131040    0.41266333
## Milk                     0.16246239 -0.59249246    0.21449173
## Fish                     1.00000000 -0.51714759    0.43868411
## Cereals                 -0.51714759  1.00000000   -0.57813449
## Starchy.Foods            0.43868411 -0.57813449    1.00000000
## Pulses.Nuts.and.Oilseeds -0.12226043  0.63605948   -0.49518800
## Fruits.and.Vegetables     0.22948842  0.04229293    0.06835670
## Total                   -0.09089592  0.18587578   -0.04418245
##                         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
## Total                                 -0.0812251            0.07201466
##                               Total
## Red.Meat                 0.37369919
## White.Meat               0.10308602
## Egg                      0.18970028
## Milk                     0.46035417
## Fish                    -0.09089592
## Cereals                  0.18587578
## Starchy.Foods           -0.04418245
## Pulses.Nuts.and.Oilseeds -0.08122510
## Fruits.and.Vegetables     0.07201466
## Total                    1.00000000
```

```
# Removing the first variable from the dataset as it is a categorical variable
#while the correlation requires quantitative(numerical values)

data_pca<- prcomp(data[,-1],scale=TRUE)
# scale=TRUE:- the variable means are set to 0, and variances are set to 1
data_pca #the components for all the variables are displayed here
```

```
## Standard deviations (1, .., p=10):
##  [1] 2.032257e+00 1.319067e+00 1.144237e+00 1.021544e+00 8.360847e-01
##  [6] 6.531975e-01 5.841454e-01 4.366348e-01 3.458098e-01 6.618503e-16
##
## Rotation (n x k) = (10 x 10):
##                                PC1         PC2         PC3          PC4
## Red.Meat                -0.3180769 -0.17809245 -0.38142753 -0.039766137
## White.Meat              -0.3140588 -0.11783853  0.36420271  0.538507972
## Egg                     -0.4202281 -0.08236350  0.02047575  0.155623651
## Milk                    -0.3870300 -0.23356182 -0.19997405 -0.320360929
## Fish                    -0.1271598  0.57388821 -0.33003267 -0.304161366
## Cereals                  0.4177240 -0.31321549 -0.02354236  0.104798477
## Starchy.Foods           -0.2880798  0.41038324  0.05768490  0.150709175
## Pulses.Nuts.and.Oilseeds  0.4177658  0.04145202 -0.24796403  0.008042093
## Fruits.and.Vegetables     0.1197680  0.34858202 -0.41210384  0.643455476
## Total                   -0.1062294 -0.41709540 -0.58081103  0.203145847
##                                 PC5          PC6         PC7         PC8
```

```
## Red.Meat                  0.53138781 -0.393811788  0.42940825 -0.1592276
## White.Meat               -0.09760147  0.309417061  0.09254681 -0.2919567
## Egg                       0.26932734 -0.059357751 -0.63995627 -0.2652806
## Milk                     -0.15848975  0.307976584 -0.17405921  0.5444724
## Fish                     -0.20323386  0.303075844  0.06315829 -0.5200308
## Cereals                  -0.29201244 -0.196460437  0.06971238 -0.2001491
## Starchy.Foods            -0.42198545 -0.680457657 -0.11769041  0.1889672
## Pulses.Nuts.and.Oilseeds  0.22507285 -0.087921207 -0.57816932 -0.0829400
## Fruits.and.Vegetables     0.16834367  0.222568384  0.08684392  0.3701826
## Total                    -0.47623561 -0.007702046 -0.05178373 -0.1801923
##                                  PC9        PC10
## Red.Meat                 -0.17150487  0.20838019
## White.Meat               -0.46186736  0.22903415
## Egg                       0.48098579  0.06827056
## Milk                     -0.13218960  0.43456461
## Fish                      0.01789764  0.21247753
## Cereals                   0.30436394  0.67412235
## Starchy.Foods            -0.14706957  0.10134794
## Pulses.Nuts.and.Oilseeds -0.58938418  0.12362100
## Fruits.and.Vegetables     0.20995988  0.11723988
## Total                    -0.04898111 -0.41440004
```

```
summary(data_pca)
```

```
## Importance of components:
##                          PC1   PC2    PC3    PC4    PC5     PC6     PC7     PC8
## Standard deviation     2.032 1.319 1.1442 1.0215 0.8361 0.65320 0.58415 0.43663
## Proportion of Variance 0.413 0.174 0.1309 0.1044 0.0699 0.04267 0.03412 0.01906
## Cumulative Proportion  0.413 0.587 0.7179 0.8223 0.8922 0.93485 0.96898 0.98804
##                            PC9      PC10
## Standard deviation     0.34581 6.619e-16
## Proportion of Variance 0.01196 0.000e+00
## Cumulative Proportion  1.00000 1.000e+00
```

```
# PC1 is able to restore 41% of the total variance, PC2 has 17% of total variance restored, PC3 has 13%,
#PC4 has 10%, PC5 has almost 7%, PC6 has 4%, PC7 has 3%, PC8 has almost 2% while PC9 has 1% of the total variance restored

#Conclusion: When I add the variances of 7 Principal Components 95% of the total variance has been restored
#i.e. 95% of the estimated protein consumption comes from these 7 components
#To have a clear idea of choosing the principal components lets' draw the scree plot and then take a final call

# sample scores stored in data_pca$x
# singular values (square roots of eigenvalues) stored in data_pca$sdev
# loadings (eigenvectors) are stored in data_pca$rotation
# variable means stored in data_pca$center
# variable standard deviations stored in data_pca$scale
# A table containing eigenvalues and %'s accounted, follows

# Eigenvalues are sdev^2
eigen_data<-data_pca$sdev^2
eigen_data
```

```
##  [1] 4.130067e+00 1.739939e+00 1.309278e+00 1.043551e+00 6.990377e-01
##  [6] 4.266669e-01 3.412258e-01 1.906500e-01 1.195844e-01 4.380459e-31
```

```
#the eigen values have no names to it so we will now assign the names to it
names(eigen_data) <- paste("PC",1:10,sep="")
eigen_data
```

```
##          PC1          PC2          PC3          PC4          PC5          PC6
## 4.130067e+00 1.739939e+00 1.309278e+00 1.043551e+00 6.990377e-01 4.266669e-01
##          PC7          PC8          PC9         PC10
## 3.412258e-01 1.906500e-01 1.195844e-01 4.380459e-31
```

```
sumlambdas<-sum(eigen_data)
sumlambdas
```

```
## [1] 10
```

```
#Calculating the sum of all the eigen values

propvar <- eigen_data/sumlambdas
propvar
```

```
##          PC1          PC2          PC3          PC4          PC5          PC6
## 4.130067e-01 1.739939e-01 1.309278e-01 1.043551e-01 6.990377e-02 4.266669e-02
##          PC7          PC8          PC9         PC10
## 3.412258e-02 1.906500e-02 1.195844e-02 4.380459e-32
```

```
cumvar_data <- cumsum(propvar)
cumvar_data
```

```
##       PC1       PC2       PC3       PC4       PC5       PC6       PC7       PC8
## 0.4130067 0.5870006 0.7179284 0.8222835 0.8921873 0.9348540 0.9689766 0.9880416
##       PC9      PC10
## 1.0000000 1.0000000
```

```
#Calculating the cumulative sum of proportion of the percentage of total variance

matlambdas <- rbind(eigen_data,propvar,cumvar_data)
matlambdas
```

```
##                   PC1       PC2       PC3       PC4        PC5        PC6
## eigen_data  4.1300672 1.7399386 1.3092782 1.0435513 0.69903765 0.42666693
## propvar     0.4130067 0.1739939 0.1309278 0.1043551 0.06990377 0.04266669
## cumvar_data 0.4130067 0.5870006 0.7179284 0.8222835 0.89218729 0.93485398
##                    PC7       PC8        PC9         PC10
## eigen_data  0.34122581 0.1906500 0.11958440 4.380459e-31
## propvar     0.03412258 0.0190650 0.01195844 4.380459e-32
## cumvar_data 0.96897656 0.9880416 1.00000000 1.000000e+00
```

```
#Putting all these values in a matrix format using the row-wise distribution

# Giving apt names to these variables
rownames(matlambdas)<- c("Eigenvalues","Prop. variance","Cum. prop. variance")
matlambdas
```

```
##                           PC1       PC2       PC3       PC4        PC5
## Eigenvalues         4.1300672 1.7399386 1.3092782 1.0435513 0.69903765
## Prop. variance      0.4130067 0.1739939 0.1309278 0.1043551 0.06990377
## Cum. prop. variance 0.4130067 0.5870006 0.7179284 0.8222835 0.89218729
##                            PC6        PC7       PC8        PC9         PC10
## Eigenvalues         0.42666693 0.34122581 0.1906500 0.11958440 4.380459e-31
## Prop. variance      0.04266669 0.03412258 0.0190650 0.01195844 4.380459e-32
## Cum. prop. variance 0.93485398 0.96897656 0.9880416 1.00000000 1.000000e+00
```

```
#very big values are displayed in each of these components so I am rounding these values till 4 decimal places
round(matlambdas,4)
```

```
##                        PC1    PC2    PC3    PC4    PC5    PC6    PC7    PC8
## Eigenvalues         4.1301 1.7399 1.3093 1.0436 0.6990 0.4267 0.3412 0.1906
## Prop. variance      0.4130 0.1740 0.1309 0.1044 0.0699 0.0427 0.0341 0.0191
## Cum. prop. variance 0.4130 0.5870 0.7179 0.8223 0.8922 0.9349 0.9690 0.9880
##                        PC9 PC10
## Eigenvalues         0.1196    0
## Prop. variance      0.0120    0
## Cum. prop. variance 1.0000    1
```

```
summary(data_pca)
```

```
## Importance of components:
##                          PC1   PC2    PC3    PC4    PC5     PC6     PC7     PC8
## Standard deviation     2.032 1.319 1.1442 1.0215 0.8361 0.65320 0.58415 0.43663
## Proportion of Variance 0.413 0.174 0.1309 0.1044 0.0699 0.04267 0.03412 0.01906
## Cumulative Proportion  0.413 0.587 0.7179 0.8223 0.8922 0.93485 0.96898 0.98804
##                            PC9      PC10
## Standard deviation     0.34581 6.619e-16
## Proportion of Variance 0.01196 0.000e+00
## Cumulative Proportion  1.00000 1.000e+00
```

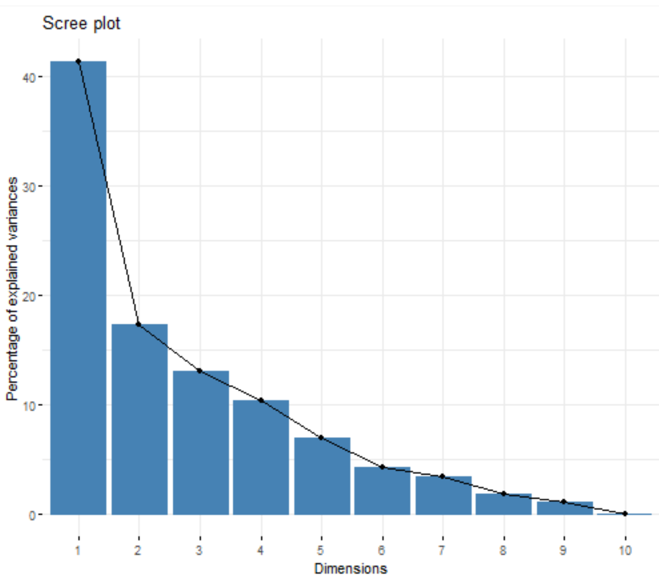
```
print(data_pca$rotation)
```

```
##                                 PC1         PC2         PC3          PC4
## Red.Meat                 -0.3180769 -0.17809245 -0.38142753 -0.039766137
## White.Meat               -0.3140588 -0.11783853  0.36420271  0.538507972
## Egg                      -0.4202281 -0.08236350  0.02047575  0.155623651
## Milk                     -0.3870300 -0.23356182 -0.19997405 -0.320360929
## Fish                     -0.1271598  0.57388821 -0.33003267 -0.304161366
## Cereals                   0.4177240 -0.31321549 -0.02354236  0.104798477
## Starchy.Foods            -0.2880798  0.41038324  0.05768490  0.150709175
## Pulses.Nuts.and.Oilseeds  0.4177658  0.04145202 -0.24796403  0.008042093
## Fruits.and.Vegetables     0.1197680  0.34858202 -0.41210384  0.643455476
## Total                    -0.1062294 -0.41709540 -0.58081103  0.203145847
##                                  PC5          PC6         PC7        PC8
## Red.Meat                  0.53138781 -0.393811788  0.42940825 -0.1592276
## White.Meat               -0.09760147  0.309417061  0.09254681 -0.2919567
## Egg                       0.26932734 -0.059357751 -0.63995627 -0.2652806
## Milk                     -0.15848975  0.307976584 -0.17405921  0.5444724
## Fish                     -0.20323386  0.303075844  0.06315829 -0.5200308
## Cereals                  -0.29201244 -0.196460437  0.06971238 -0.2001491
## Starchy.Foods            -0.42198545 -0.680457657 -0.11769041  0.1889672
## Pulses.Nuts.and.Oilseeds  0.22507285 -0.087921207 -0.57816932 -0.0829400
## Fruits.and.Vegetables     0.16834367  0.222568384  0.08684392  0.3701826
## Total                    -0.47623561 -0.007702046 -0.05178373 -0.1801923
##                                  PC9        PC10
## Red.Meat                 -0.17150487  0.20838019
## White.Meat               -0.46186736  0.22903415
## Egg                       0.48098579  0.06827056
## Milk                     -0.13218960  0.43456461
## Fish                      0.01789764  0.21247753
## Cereals                   0.30436394  0.67412235
## Starchy.Foods            -0.14706957  0.10134794
## Pulses.Nuts.and.Oilseeds -0.58938418  0.12362100
## Fruits.and.Vegetables     0.20995988  0.11723988
## Total                    -0.04898111 -0.41440004
```

```r
# Sample scores stored in data_pca$x
data_pca$x
```

```
##              PC1         PC2        PC3         PC4          PC5         PC6
##  [1,]  3.5978397 -0.64061101  1.1118946 -1.91119245  1.884437106 -0.37593345
##  [2,] -1.3862854 -0.70991905  1.1613381  0.93107494 -0.009121937  0.75816906
##  [3,] -1.6608482  0.10781730 -0.4231894  0.24680766  0.188016546 -0.91001548
##  [4,]  2.9881523 -1.84361307 -0.0730564  0.30616165 -0.134812297  0.29005421
##  [5,] -0.3686147 -0.10141825  1.2155042  0.72202089 -0.062918010 -0.37091750
##  [6,] -2.4923551  0.18474749 -0.2075253 -0.93906831 -0.822177041  0.65204948
##  [7,] -1.2387459  1.58140979  1.9302394  0.77259151 -0.139755937 -0.58954056
##  [8,] -1.7732789 -0.75352175 -0.3644876 -2.28429396 -1.224019848  0.17828822
##  [9,] -1.6448018 -0.30606640 -2.4846910  1.25325810  0.230223125 -0.33223855
## [10,]  2.0943234 -0.61997417 -3.0846378  0.31332068  0.270784604  0.64981699
## [11,]  1.4808993 -0.43978564  1.6090270  1.21709297 -0.143865961  0.11534733
## [12,] -2.6714332 -1.03848419 -0.2833724  0.15763312  0.181076517 -0.86151844
## [13,]  1.5660043 -0.01064018 -0.5907111  0.54266246  1.069631810  0.77586008
## [14,] -1.7006997 -0.50438298  0.7596605  0.64321026  0.292062273  0.92348043
## [15,] -0.8828201  1.28521025 -0.1832152 -1.71931314 -0.439007528  0.41757899
## [16,] -0.2286613  0.19642466 -0.4058046  1.67696384 -1.334150980  0.08818598
## [17,]  2.0912590  4.41252506 -0.6718598 -0.03434506 -0.291193444  0.33278906
## [18,]  2.6049767 -1.05771521  0.5868844 -0.14252039 -0.533268313 -0.20083289
## [19,]  1.5709389  2.67472726 -0.2892457  0.23912301  0.594881631 -0.60647031
## [20,] -1.8343339  0.36443676  0.5444138 -1.56417414  0.158327086  0.80195706
## [21,] -0.9293183 -0.96269089 -0.3476755  0.27836268  0.755554148  0.70844461
## [22,] -1.9728952 -0.55508144 -0.8727628 -0.60997694  1.396218668 -1.20971357
## [23,]  0.7660628 -0.48463412 -0.2720099 -0.40950179 -1.470304012 -1.24044252
## [24,] -1.6857673  0.30943116  1.2190705  0.55052071  0.810416131  0.20076819
## [25,]  3.7104025 -1.08819138  0.4162119 -0.23641829 -1.227034337 -0.19516642
##                 PC7          PC8          PC9          PC10
##  [1,]  0.6467777066  0.308209567 -0.344610598 -7.771561e-16
##  [2,]  0.0005093868 -0.012933034  0.124176638 -9.471590e-16
##  [3,]  0.1534640851 -0.334041295  0.023323758 -3.330669e-16
##  [4,]  0.5999541449 -0.762640350  0.674235551  1.665335e-16
##  [5,]  0.7878924305 -0.039689570  0.241927022 -6.661338e-16
##  [6,] -0.0364433564 -0.984127670 -0.168254146 -5.551115e-17
##  [7,] -0.0632650200 -0.313388346  0.320254182 -8.881784e-16
##  [8,] -0.0506617637  0.792618282  0.004268287 -6.661338e-16
##  [9,]  1.3629405718 -0.176345585 -0.392094989 -4.440892e-16
## [10,] -1.1867279230 -0.252605939 -0.185325024 -4.440892e-16
## [11,] -0.8173673169 -0.201792286 -0.496946360 -9.714451e-16
## [12,] -0.7338089555  0.194588527 -0.047542669 -5.551115e-16
## [13,]  0.0085984337  0.435335074  0.815121519 -8.326673e-16
## [14,] -0.2530352518  0.088559649 -0.434700410 -1.051242e-15
## [15,]  0.0122896156  0.009259812  0.182509788 -4.718448e-16
## [16,] -0.0295375727  0.839590880  0.341088667 -7.771561e-16
## [17,]  0.6466024099 -0.205548666 -0.304794550 -1.110223e-15
## [18,] -0.2135771460 -0.211277632 -0.024663621 -3.677614e-16
## [19,] -0.9520057576  0.408309790  0.166895175 -1.221245e-15
## [20,] -0.1459371778 -0.241698086  0.340921291 -6.106227e-16
## [21,]  0.6841927749  0.678069260 -0.252544924 -1.040834e-15
## [22,] -0.4798955917 -0.365578790  0.224480622 -2.636780e-16
## [23,]  0.3126133335  0.287576242 -0.038575860 -1.110223e-16
## [24,] -0.0977006735  0.141407912 -0.415483582 -1.110223e-15
## [25,] -0.1558713867 -0.081857747 -0.353665768 -4.718448e-16
```

```r
fviz_eig(data_pca)
```

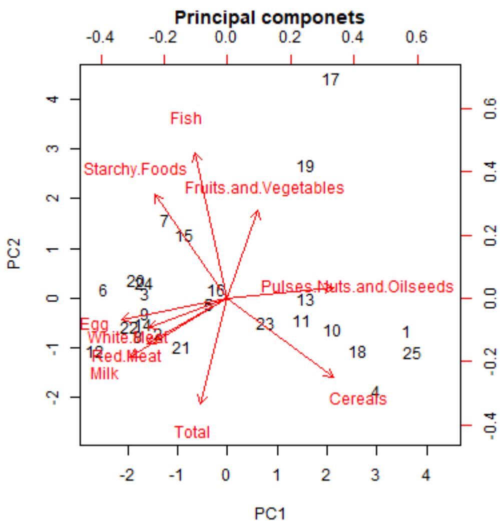



```r
summary(data_pca)
```

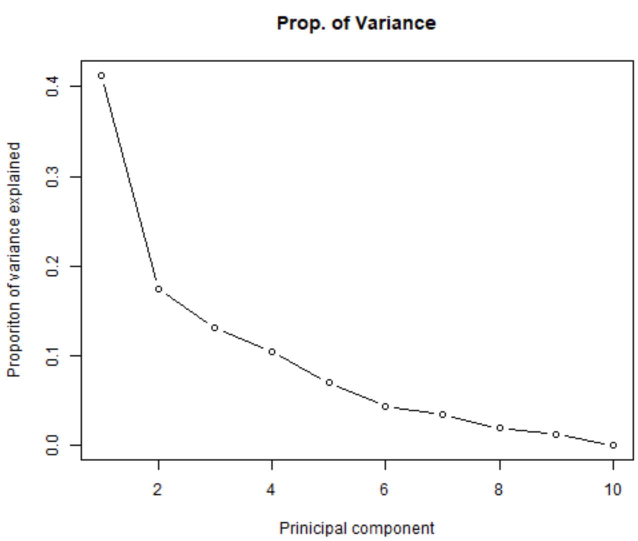
```
## Importance of components:
##                          PC1   PC2    PC3    PC4    PC5     PC6     PC7     PC8
## Standard deviation     2.032 1.319 1.1442 1.0215 0.8361 0.65320 0.58415 0.43663
## Proportion of Variance 0.413 0.174 0.1309 0.1044 0.0699 0.04267 0.03412 0.01906
```

```
## Cumulative Proportion  0.413 0.587 0.7179 0.8223 0.8922 0.93485 0.96898 0.98804
##                            PC9      PC10
## Standard deviation     0.34581 6.619e-16
## Proportion of Variance 0.01196 0.000e+00
## Cumulative Proportion  1.00000 1.000e+00
```

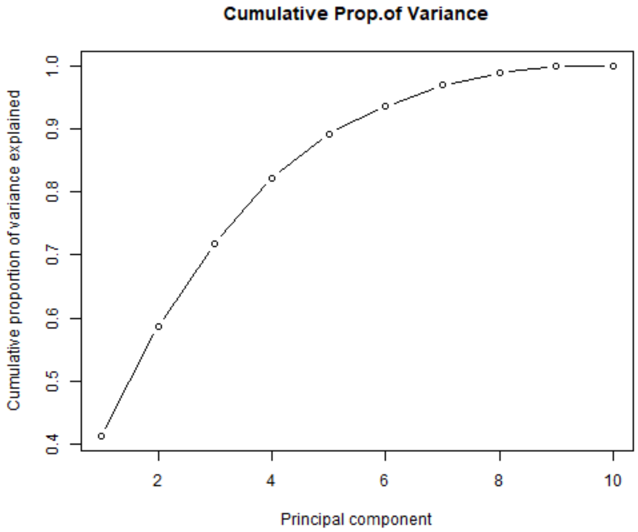
```
## Plot a biplot to view components on n-dimensional plane
biplot(data_pca, scale = 0, main = 'Principal componets')
```



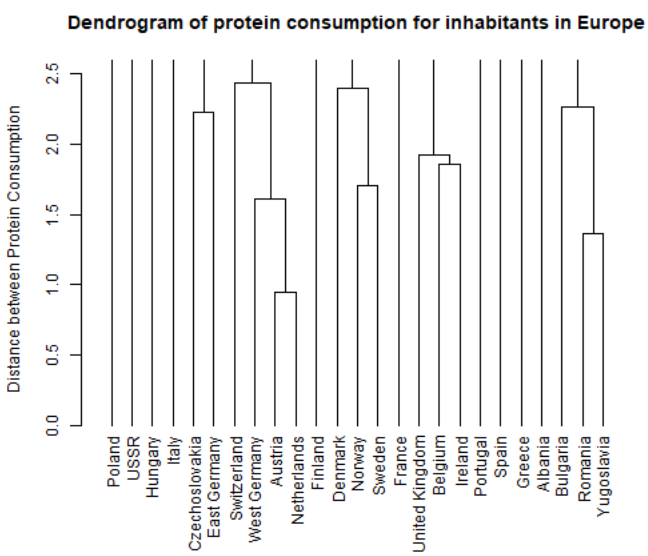

```
plot(propvar, xlab = 'Prinicipal component',ylab = 'Proporiton of variance explained',type = 'b', main = 'Prop. of Variance')
```




```
#The optimum number of components are ~ 6 i.e PC1 : PC6

# cumulative scree plot
plot(cumvar_data,xlab = 'Principal component',ylab = 'Cumulative proportion of variance explained',type = 'b', main = 'Cumulative Prop.of Variance')
```

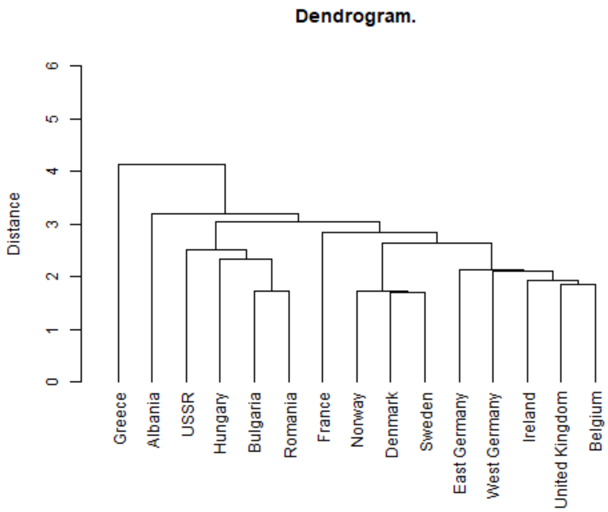



```
#Approx: ~ 92% of the variance is explained by 6 components i.e PC1 to PC6
############## End of PCA ########################
# Conclusion: PCA is a dimension -reduction tool that can be used to reduce a large set
# of variables to a small set that still contains most of the information in the large set.
# From the above scree plot I am concluding that I would want to consider
# the first 6 principal components as they help is restoring 92% of the total variance
# i.e the first 6 components will contribute to maximum average protein consumption (in grams per person per day)
##################################################

#Q2) Carry out cluster analysis to study relation between countries on their diet
########## Clustering Analysis ###################

data1<- read.csv("Protein_Consumption.csv", row.names=1, fill = TRUE)
matstd.prot<-scale(data1)

# Creating a (Euclidean) distance matrix of the standardized data
dist.prot <- dist(matstd.prot, method="euclidean")

# Invoking hclust command (cluster analysis by single linkage method)
clusprot.nn <- hclust(dist.prot)

# Plotting vertical dendrogram
# create extra margin room in the dendrogram, on the bottom (Protein consumption' labels)
par(mar=c(6, 4, 4, 2) + 0.1)
plot(as.dendrogram(clusprot.nn),ylab="Distance between Protein Consumption",ylim=c(0,2.5),main="Dendrogram of protein consumption for inhabitants in Eur
```




```
######################################################################

# take a random sample of size 15 from a dataset of data1
# sample without replacement
mysample <- data1[sample(1:nrow(data1),15,replace=FALSE),]

# Standardizing the data with scale()
matstd.loan<- scale(mysample)
# Creating a (Euclidean) distance matrix of the standardized data
dist.employ <- dist(matstd.loan, method="euclidean")
# Invoking hclust command (cluster analysis by single linkage method)
clusemploy.nn <- hclust(dist.employ, method = "single")
```

```
#Plotting

# Create extra margin room in the dendrogram, on the bottom (loan labels)
par(mar=c(8, 4, 4, 2) + 0.1)
# Object "clusemploy.nn" is converted into a object of class "dendrogram"
# in order to allow better flexibility in the (vertical) dendrogram plotting.
plot(as.dendrogram(clusemploy.nn),ylab="Distance",ylim=c(0,6),
     main="Dendrogram.")
```



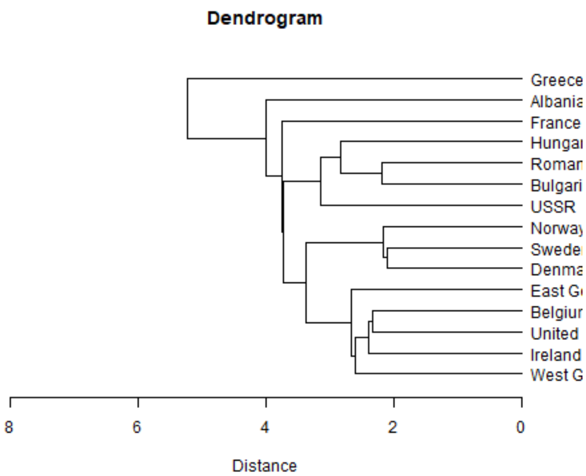

```
################### Agne Function ############################

# We will use agnes function as it allows us to select option for data standardization, the distance measure and clustering algorithm in one single func

(agn.employ <- agnes(mysample, metric="euclidean", stand=TRUE, method = "single"))
```

```
## Call:         agnes(x = mysample, metric = "euclidean", stand = TRUE, method = "single")
## Agglomerative coefficient:  0.4640585
## Order of objects:
##  [1] West Germany   Ireland        United Kingdom Belgium        East Germany
##  [6] Denmark        Sweden         Norway         USSR           Bulgaria
## [11] Romania        Hungary        France         Albania        Greece
## Height (summary):
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   2.103   2.352   2.739   3.034   3.640   5.226
##
## Available components:
## [1] "order"     "height"    "ac"        "merge"     "diss"      "call"
## [7] "method"    "order.lab" "data"
```

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

```
##       [,1] [,2]
##  [1,]   -2   -4
##  [2,]    1  -14
##  [3,]  -10  -11
##  [4,]   -7   -8
##  [5,]   -6    4
##  [6,]   -1    5
##  [7,]    6  -13
##  [8,]    3  -15
##  [9,]   -5    8
## [10,]    7    2
## [11,]   10    9
## [12,]   11   -9
## [13,]   12  -12
## [14,]   13   -3
```

```
#Dendogram
plot(as.dendrogram(agn.employ), xlab= "Distance",xlim=c(8,0),
     horiz = TRUE,main="Dendrogram")
```




```
#Interactive Plots
plot(agn.employ,ask=TRUE)
```

```
## Error in menu(tmenu, title = "\nMake a plot selection (or 0 to exit):\n"): menu() cannot be used non-interactively
```

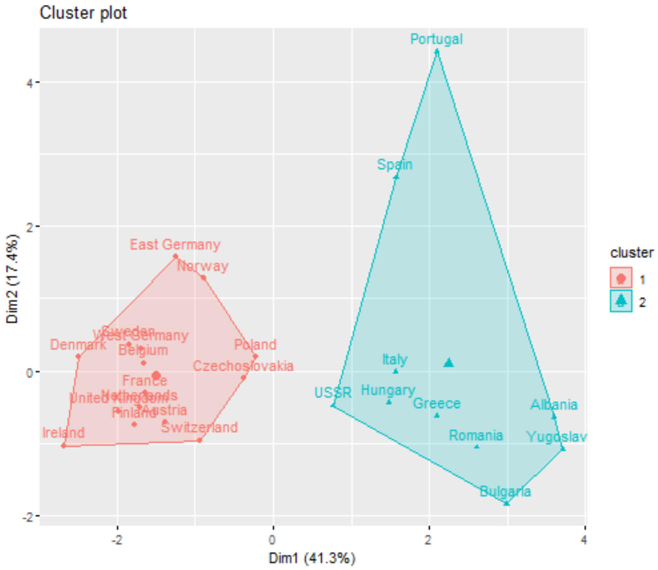
```
#################################################################

######################## K-means ################################
# Standardizing the data with scale()
matstd.employ <- scale(data1)

# K-means, k=2, 3, 4, 5
# Centers (k's) are numbers thus, 10 random sets are chosen

(kmeans2.employ <- kmeans(matstd.employ,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      Total
## 1     0.3866381               -0.6999903            -0.2088932  0.1300177
## 2    -0.5799572                1.0499854             0.3133398 -0.1950266
##
## 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] 72.91145 82.27613
##  (between_SS / total_SS =  35.3 %)
##
## Available components:
##
## [1] "cluster"      "centers"      "totss"        "withinss"     "tot.withinss"
## [6] "betweenss"    "size"         "iter"         "ifault"
```

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

```
## Perc. 2 clus
##         64.7
```

```
fviz_cluster(kmeans2.employ,data=matstd.employ)
```



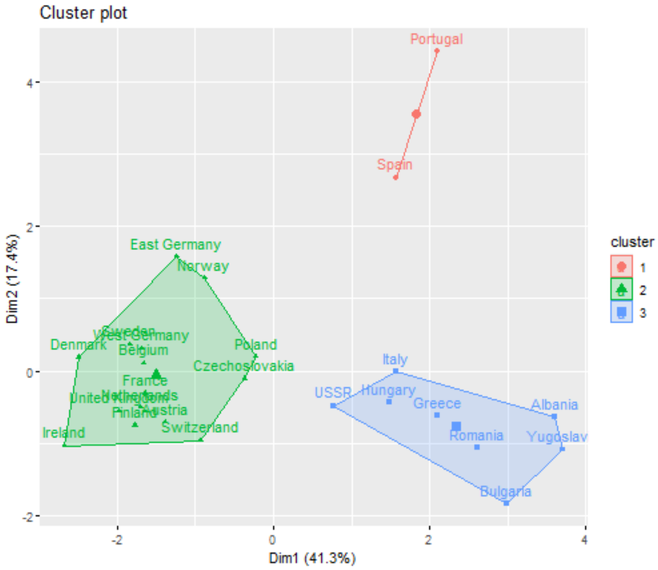

```
# Conclusion: Only 2 clusters are formed but the % of variance is 65%

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

```
## K-means clustering with 3 clusters of sizes 2, 15, 8
##
## Cluster means:
##     Red.Meat White.Meat        Egg       Milk       Fish    Cereals
## 1 -0.9696102 -1.1815761 -0.9685677 -1.4483663  1.7923261 -0.3923599
## 2  0.4701140  0.5203925  0.5859237  0.5804736  0.1306304 -0.6103377
## 3 -0.6390612 -0.6803419 -0.8564649 -0.7262965 -0.6930136  1.2424732
##   Starchy.Foods Pulses.Nuts.and.Oilseeds Fruits.and.Vegetables      Total
## 1     0.9907602                1.1985682           1.72336879 -1.4508795
## 2     0.3866381               -0.6999903          -0.20889319  0.1300177
## 3    -0.9726366                1.0128397          -0.03916747  0.1189367
##
## Clustering vector:
##        Albania        Austria        Belgium       Bulgaria Czechoslovakia
##              3              2              2              3              2
##        Denmark   East Germany        Finland         France         Greece
##              2              2              2              2              3
##        Hungary        Ireland          Italy    Netherlands         Norway
##              3              2              3              2              2
##         Poland       Portugal        Romania          Spain         Sweden
##              2              1              3              1              2
##    Switzerland United Kingdom           USSR   West Germany     Yugoslavia
##              2              2              3              2              3
##
## Within cluster sum of squares by cluster:
## [1]  4.167026 72.911454 47.382187
##  (between_SS / total_SS =  48.1 %)
##
## Available components:
##
## [1] "cluster"      "centers"      "totss"        "withinss"     "tot.withinss"
## [6] "betweenss"    "size"         "iter"         "ifault"
```

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

```
## Perc. 3 clus
##         51.9
```

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



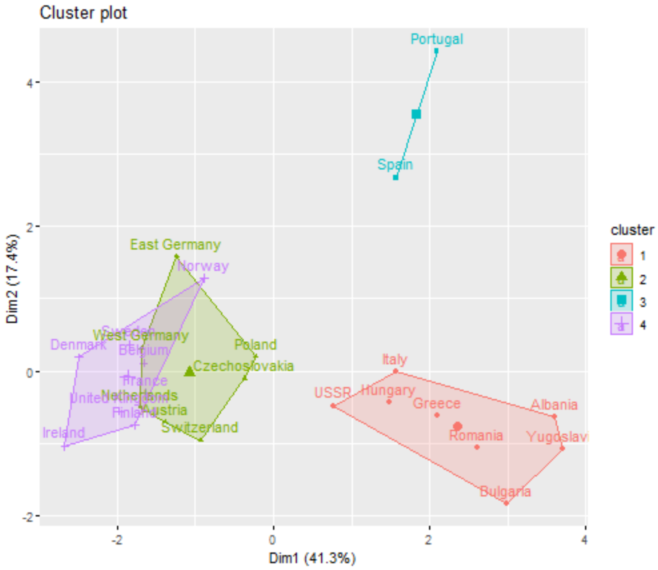

```
#Conclusion: Three clusters with 52%of the variance is restored and there are three separate groups which are visible

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

```
## K-means clustering with 4 clusters of sizes 8, 7, 2, 8
##
## Cluster means:
##      Red.Meat  White.Meat        Egg       Milk       Fish    Cereals
## 1 -0.63906125 -0.68034187 -0.8564649 -0.7262965 -0.6930136  1.2424732
## 2 -0.02518468  1.09068558  0.4407239  0.1618241 -0.4100039 -0.4702091
## 3 -0.96961017 -1.18157605 -0.9685677 -1.4483663  1.7923261 -0.3923599
## 4  0.90350039  0.02138599  0.7129734  0.9467920  0.6036854 -0.7329502
##   Starchy.Foods Pulses.Nuts.and.Oilseeds Fruits.and.Vegetables      Total
## 1    -0.9726366                1.0128397           -0.03916747  0.1189367
## 2     0.3003350               -0.7471594            0.19397225 -0.2372401
## 3     0.9907602                1.1985682            1.72336879 -1.4508795
## 4     0.4621534               -0.6587173           -0.56140044  0.4513683
##
## Clustering vector:
##        Albania        Austria        Belgium       Bulgaria Czechoslovakia
##              1              2              4              1              2
##        Denmark   East Germany        Finland         France         Greece
##              4              2              4              4              1
##        Hungary        Ireland          Italy    Netherlands         Norway
##              1              4              1              2              4
##         Poland       Portugal        Romania          Spain         Sweden
##              2              3              1              3              4
##    Switzerland United Kingdom           USSR   West Germany     Yugoslavia
##              2              4              1              2              1
##
## Within cluster sum of squares by cluster:
## [1] 47.382187 20.027050  4.167026 34.697369
##  (between_SS / total_SS =  55.7 %)
##
## Available components:
##
## [1] "cluster"      "centers"      "totss"        "withinss"     "tot.withinss"
## [6] "betweenss"    "size"         "iter"         "ifault"
```

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

```
## Perc. 4 clus
##         44.3
```

```
fviz_cluster(kmeans4.employ,data=matstd.employ)
```



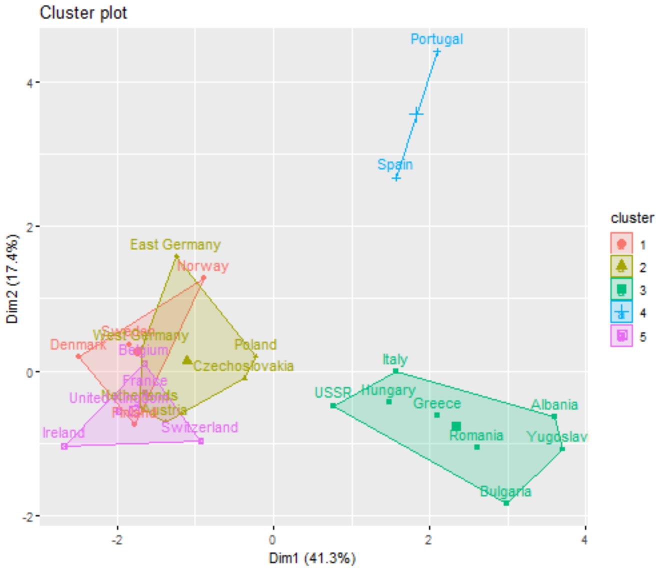

```
# Conclusion: 4 clusters and 44% of the variance is stored in these clusters and there is an overlap

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

```
## K-means clustering with 5 clusters of sizes 4, 6, 8, 2, 5
##
## Cluster means:
##      Red.Meat White.Meat        Egg        Milk       Fish    Cereals
## 1  0.05876425 -0.1791077  0.3766652  1.33424402  1.2160155 -0.8691863
## 2 -0.18608680  1.1797939  0.5261355  0.03099617 -0.3688388 -0.4529093
## 3 -0.63906125 -0.6803419 -0.8564649 -0.72629651 -0.6930136  1.2424732
## 4 -0.96961017 -1.1815761 -0.9685677 -1.44836627  1.7923261 -0.3923599
## 5  1.58663483  0.2887109  0.8250762  0.63683030 -0.1383146 -0.5921729
##   Starchy.Foods Pulses.Nuts.and.Oilseeds Fruits.and.Vegetables       Total
## 1     0.2356076               -0.9063553           -1.14891253  0.06353138
## 2     0.4873252               -0.7825363            0.15666989 -0.31814941
## 3    -0.9726366                1.0128397           -0.03916747  0.11893666
## 4     0.9907602                1.1985682            1.72336879 -1.45087949
## 5     0.3866381               -0.4358430            0.10444659  0.72100732
##
## Clustering vector:
##        Albania        Austria        Belgium       Bulgaria Czechoslovakia
##              3              2              5              3              2
##        Denmark   East Germany        Finland         France         Greece
##              1              2              1              5              3
##        Hungary        Ireland          Italy    Netherlands         Norway
##              3              5              3              2              1
##         Poland       Portugal        Romania          Spain         Sweden
##              2              4              3              4              1
##    Switzerland United Kingdom           USSR   West Germany     Yugoslavia
##              5              5              3              2              3
##
## Within cluster sum of squares by cluster:
## [1]  7.849041 15.642697 47.382187  4.167026 15.011394
##  (between_SS / total_SS =  62.5 %)
##
## Available components:
##
## [1] "cluster"      "centers"      "totss"        "withinss"     "tot.withinss"
## [6] "betweenss"    "size"         "iter"         "ifault"
```

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

```
## Perc. 5 clus
##         37.5
```

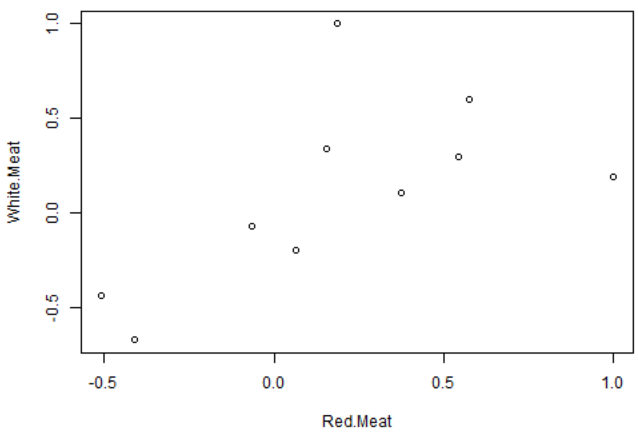
```
fviz_cluster(kmeans5.employ,data=matstd.employ)
```




```
#Conclusion: 5 clusters with 37.5% of the total variance is restored and the clusters are overlapping

############# END of Clustering ##########################
# Conclusion: Clustering is an exploratory data analysis technique which helps in identifying
# subgroups within the dataset. When I select 4 clusters there is an overlap between the clusters and
#the percentage of variance restored is also only 44% but when I cluster for 3 there are clearly
# three separate groups formed and 52% of the total variance is restored.
# Hence, I will chose 3 clusters in our problem statement where those countries in Europe who have
# similar consumption of protein are placed in the same groups.

#####################################################################
# Q3) Identify the important factors underlying the observed variables
# and examine the relationships between the countries with respect to these factors

########### Factor Analysis ######################################
#calculating the correlation matrix for all the numeric data in our dataset

corrm.emp = cor(data1)
corrm.emp
```

```
##                            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
## Total                    0.37369919  0.10308602  0.18970028  0.4603542
##                                Fish     Cereals Starchy.Foods
## Red.Meat                 0.06491072 -0.50970337    0.15383673
## White.Meat              -0.19719960 -0.43941908    0.33456770
## Egg                      0.04780844 -0.70131040    0.41266333
## Milk                     0.16246239 -0.59249246    0.21449173
## Fish                     1.00000000 -0.51714759    0.43868411
## Cereals                 -0.51714759  1.00000000   -0.57813449
## Starchy.Foods            0.43868411 -0.57813449    1.00000000
## Pulses.Nuts.and.Oilseeds -0.12226043  0.63605948   -0.49518800
## Fruits.and.Vegetables    0.22948842  0.04229293    0.06835670
## Total                   -0.09089592  0.18587578   -0.04418245
##                          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
## Total                                  -0.0812251            0.07201466
##                                Total
## Red.Meat                 0.37369919
## White.Meat               0.10308602
## Egg                      0.18970028
## Milk                     0.46035417
## Fish                    -0.09089592
## Cereals                  0.18587578
## Starchy.Foods           -0.04418245
## Pulses.Nuts.and.Oilseeds -0.08122510
## Fruits.and.Vegetables    0.07201466
## Total                    1.00000000
```

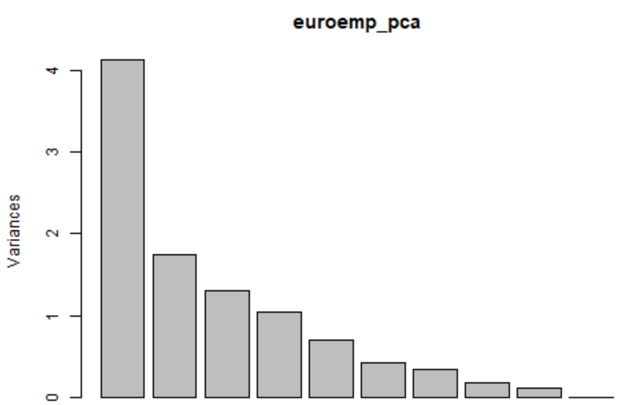
```
plot(corrm.emp)
```




```
#this is the correlation plot

#calculating the PCA and plotting these variances
euroemp_pca <- prcomp(data1, scale=TRUE)
summary(euroemp_pca)
```

```
## Importance of components:
##                          PC1   PC2    PC3    PC4    PC5     PC6     PC7     PC8
## Standard deviation     2.032 1.319 1.1442 1.0215 0.8361 0.65320 0.58415 0.43663
## Proportion of Variance 0.413 0.174 0.1309 0.1044 0.0699 0.04267 0.03412 0.01906
## Cumulative Proportion  0.413 0.587 0.7179 0.8223 0.8922 0.93485 0.96898 0.98804
##                            PC9      PC10
## Standard deviation     0.34581 6.619e-16
## Proportion of Variance 0.01196 0.000e+00
## Cumulative Proportion  1.00000 1.000e+00
```

```
plot(euroemp_pca)
```




```
#looks like Pc1, pc2, pc3,pc4,pc5 restores maximum of variance

# A table containing eigenvalues and %'s accounted, follows.
# Eigenvalues are the sdev^2
(eigen_euroemp <- round(euroemp_pca$sdev^2,2))
```

```
##  [1] 4.13 1.74 1.31 1.04 0.70 0.43 0.34 0.19 0.12 0.00
```

```
names(eigen_euroemp) <- paste("PC",1:10,sep="")
eigen_euroemp
```

```
##  PC1  PC2  PC3  PC4  PC5  PC6  PC7  PC8  PC9 PC10
## 4.13 1.74 1.31 1.04 0.70 0.43 0.34 0.19 0.12 0.00
```

```
sumlambdas <- sum(eigen_euroemp)
sumlambdas
```

```
## [1] 10
```

```
propvar <- round(eigen_euroemp/sumlambdas,2)
propvar
```

```
##  PC1  PC2  PC3  PC4  PC5  PC6  PC7  PC8  PC9 PC10
## 0.41 0.17 0.13 0.10 0.07 0.04 0.03 0.02 0.01 0.00
```

```
cumvar_euroemp <- cumsum(propvar)
cumvar_euroemp
```

```
##  PC1  PC2  PC3  PC4  PC5  PC6  PC7  PC8  PC9 PC10
## 0.41 0.58 0.71 0.81 0.88 0.92 0.95 0.97 0.98 0.98
```

```
matlambdas <- rbind(eigen_euroemp,propvar,cumvar_euroemp)
matlambdas
```

```
##                 PC1  PC2  PC3  PC4  PC5  PC6  PC7  PC8  PC9 PC10
## eigen_euroemp  4.13 1.74 1.31 1.04 0.70 0.43 0.34 0.19 0.12 0.00
## propvar        0.41 0.17 0.13 0.10 0.07 0.04 0.03 0.02 0.01 0.00
## cumvar_euroemp 0.41 0.58 0.71 0.81 0.88 0.92 0.95 0.97 0.98 0.98
```

```
rownames(matlambdas) <- c("Eigenvalues","Prop. variance","Cum. prop. variance")
rownames(matlambdas)
```

```
## [1] "Eigenvalues"         "Prop. variance"      "Cum. prop. variance"
```

```
eigvec.emp <- euroemp_pca$rotation
print(euroemp_pca)
```

```
## Standard deviations (1, .., p=10):
##  [1] 2.032257e+00 1.319067e+00 1.144237e+00 1.021544e+00 8.360847e-01
##  [6] 6.531975e-01 5.841454e-01 4.366348e-01 3.458098e-01 6.618503e-16
## 
## Rotation (n x k) = (10 x 10):
##                               PC1         PC2         PC3          PC4
## Red.Meat                 -0.3180769 -0.17809245 -0.38142753 -0.039766137
## White.Meat               -0.3140588 -0.11783853  0.36420271  0.538507972
## Egg                      -0.4202281 -0.08236350  0.02047575  0.155623651
## Milk                     -0.3870300 -0.23356182 -0.19997405 -0.320360929
## Fish                     -0.1271598  0.57388821 -0.33003267 -0.304161366
## Cereals                   0.4177240 -0.31321549 -0.02354236  0.104798477
## Starchy.Foods            -0.2880798  0.41038324  0.05768490  0.150709175
## Pulses.Nuts.and.Oilseeds  0.4177658  0.04145202 -0.24796403  0.008042093
## Fruits.and.Vegetables     0.1197680  0.34858202 -0.41210384  0.643455476
## Total                    -0.1062294 -0.41709540 -0.58081103  0.203145847
##                                 PC5          PC6         PC7        PC8
## Red.Meat                  0.53138781 -0.393811788  0.42940825 -0.1592276
## White.Meat               -0.09760147  0.309417061  0.09254681 -0.2919567
## Egg                       0.26932734 -0.059357751 -0.63995627 -0.2652806
## Milk                     -0.15848975  0.307976584 -0.17405921  0.5444724
## Fish                     -0.20323386  0.303075844  0.06315829 -0.5200308
## Cereals                  -0.29201244 -0.196460437  0.06971238 -0.2001491
## Starchy.Foods            -0.42198545 -0.680457657 -0.11769041  0.1889672
## Pulses.Nuts.and.Oilseeds  0.22507285 -0.087921207 -0.57816932 -0.0829400
## Fruits.and.Vegetables     0.16834367  0.222568384  0.08684392  0.3701826
## Total                    -0.47623561 -0.007702046 -0.05178373 -0.1801923
##                                 PC9        PC10
## Red.Meat                 -0.17150487  0.20838019
## White.Meat               -0.46186736  0.22903415
## Egg                       0.48098579  0.06827056
## Milk                     -0.13218960  0.43456461
## Fish                      0.01789764  0.21247753
## Cereals                   0.30436394  0.67412235
## Starchy.Foods            -0.14706957  0.10134794
## Pulses.Nuts.and.Oilseeds -0.58938418  0.12362100
## Fruits.and.Vegetables     0.20995988  0.11723988
## Total                    -0.04898111 -0.41440004
```

```
#Taking the first five PCs to generate linear combinations for all the variables
pcafactors.emp <- eigvec.emp[,1:5]
pcafactors.emp
```

```
##                               PC1         PC2         PC3          PC4
## Red.Meat                 -0.3180769 -0.17809245 -0.38142753 -0.039766137
## White.Meat               -0.3140588 -0.11783853  0.36420271  0.538507972
## Egg                      -0.4202281 -0.08236350  0.02047575  0.155623651
## Milk                     -0.3870300 -0.23356182 -0.19997405 -0.320360929
## Fish                     -0.1271598  0.57388821 -0.33003267 -0.304161366
## Cereals                   0.4177240 -0.31321549 -0.02354236  0.104798477
## Starchy.Foods            -0.2880798  0.41038324  0.05768490  0.150709175
## Pulses.Nuts.and.Oilseeds  0.4177658  0.04145202 -0.24796403  0.008042093
```

```
## Fruits.and.Vegetables     0.1197680  0.34858202 -0.41210384  0.643455476
## Total                    -0.1062294 -0.41709540 -0.58081103  0.203145847
##                                  PC5
## Red.Meat                  0.53138781
## White.Meat               -0.09760147
## Egg                       0.26932734
## Milk                     -0.15848975
## Fish                     -0.20323386
## Cereals                  -0.29201244
## Starchy.Foods            -0.42198545
## Pulses.Nuts.and.Oilseeds  0.22507285
## Fruits.and.Vegetables     0.16834367
## Total                    -0.47623561
```

```
# Multiplying each column of the eigenvector's matrix by the square-root of the corresponding eigenvalue in order to get the factor loadings
unrot.fact.emp <- sweep(pcafactors.emp,MARGIN=2,euroemp_pca$sdev[1:5],`*`)
unrot.fact.emp
```

```
##                                 PC1        PC2         PC3          PC4
## Red.Meat                 -0.6464140 -0.2349159 -0.43644348 -0.040622842
## White.Meat               -0.6382482 -0.1554370  0.41673420  0.550109362
## Egg                      -0.8540114 -0.1086430  0.02342911  0.158976342
## Milk                     -0.7865443 -0.3080838 -0.22881770 -0.327262651
## Fish                     -0.2584213  0.7569972 -0.37763557 -0.310714091
## Cereals                   0.8489223 -0.4131523 -0.02693804  0.107056211
## Starchy.Foods            -0.5854521  0.5413231  0.06600519  0.153955990
## Pulses.Nuts.and.Oilseeds  0.8490074  0.0546780 -0.28372960  0.008215348
## Fruits.and.Vegetables     0.2433992  0.4598032 -0.47154445  0.657317811
## Total                    -0.2158855 -0.5501769 -0.66458545  0.207522336
##                                  PC5
## Red.Meat                  0.44428523
## White.Meat               -0.08160309
## Egg                       0.22518048
## Milk                     -0.13251086
## Fish                     -0.16992073
## Cereals                  -0.24414713
## Starchy.Foods            -0.35281559
## Pulses.Nuts.and.Oilseeds  0.18817997
## Fruits.and.Vegetables     0.14074957
## Total                    -0.39817332
```

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

```
##                 Red.Meat               White.Meat                      Egg
##                0.8625590                0.9144681                0.8176674
##                     Milk                     Fish                  Cereals
##                0.8905850                0.9078512                0.9631584
##            Starchy.Foods Pulses.Nuts.and.Oilseeds    Fruits.and.Vegetables
##                0.7883228                0.8397850                0.9448934
##                    Total
##                0.9925825
```

```
# Performing the varimax rotation. The default in the varimax function is norm=TRUE thus, Kaiser normalization is carried out
rot.fact.emp <- varimax(unrot.fact.emp)
View(unrot.fact.emp)
rot.fact.emp
```

```
## $loadings
##
## Loadings:
##                          PC1    PC2    PC3    PC4    PC5
## Red.Meat                               -0.228         0.897
## White.Meat               -0.936 -0.124                0.150
## Egg                      -0.588               -0.103  0.671
## Milk                     -0.233  0.243 -0.427 -0.521  0.568
## Fish                      0.180  0.923                0.112
## Cereals                   0.419 -0.559 -0.252        -0.634
## Starchy.Foods            -0.550  0.695
## Pulses.Nuts.and.Oilseeds  0.709 -0.259         0.412 -0.309
## Fruits.and.Vegetables           0.156         0.955
## Total                                  -0.977         0.168
##
##                  PC1   PC2   PC3   PC4   PC5
## SS loadings    2.299 1.830 1.268 1.386 2.139
## Proportion Var 0.230 0.183 0.127 0.139 0.214
## Cumulative Var 0.230 0.413 0.540 0.678 0.892
##
## $rotmat
##            [,1]       [,2]       [,3]       [,4]       [,5]
## [1,]  0.6255823 -0.3589826  0.1399026  0.2682405 -0.6230992
## [2,]  0.0252904  0.7722244  0.4833311  0.3851547 -0.1451784
## [3,] -0.4909586 -0.2604259  0.6394138 -0.3857373 -0.3653693
## [4,] -0.5631966 -0.2793097 -0.1587512  0.7523791 -0.1162730
## [5,]  0.2231063 -0.3591174  0.5592552  0.2546280  0.6660754
```

```
#The print method of varimax omits loadings less than abs(0.1). In order to display all the loadings, it is necessary to ask explicitly the contents of
fact.load.emp <- rot.fact.emp$loadings[1:5,1:5]
fact.load.emp
```

```
##                   PC1         PC2         PC3         PC4       PC5
## Red.Meat   -0.07404907  0.01610055 -0.22812738 -0.01295720 0.8969986
## White.Meat -0.93583294 -0.12378641 -0.03092196  0.00129115 0.1496794
## Egg        -0.58780050  0.09130706 -0.05631238 -0.10301458 0.6708487
## Milk       -0.23275046  0.24303057 -0.42740930 -0.52134596 0.5682148
## Fish        0.17996718  0.92349352  0.04255822  0.09086870 0.1120465
```

```
#Computing the rotated factor scores for the 25 European Countries.
scale.emp <- scale(data1)
scale.emp
```

```
##                  Red.Meat  White.Meat         Egg        Milk        Fish
## Albania         0.05876425 -1.84988830 -1.86538958 -1.16658295 -1.23330478
## Austria        -0.23505701  1.62533538  0.82507616  0.38322532 -0.65699414
## Belgium         1.23404931  0.28871089  0.82507616  0.10144200  0.20747183
## Bulgaria       -0.52887828 -0.51326380 -0.96856767 -1.30747461 -0.94514946
## Czechoslovakia  0.05876425  0.82336069 -0.07174575 -0.60301630 -0.65699414
## Denmark         0.35258552  0.82336069  0.82507616  1.08768362  1.64824845
## East Germany   -0.52887828  1.09068558  0.82507616 -0.88479963  0.20747183
## Finland         0.05876425 -0.78058870 -0.07174575  2.35570856  0.49562716
## France          2.40933437  0.55603579 -0.07174575  0.38322532  0.49562716
## Greece          0.05876425 -1.31523850 -0.07174575  0.10144200  0.49562716
## Hungary        -1.41034207  1.09068558 -0.07174575 -1.02569129 -1.23330478
## Ireland         1.23404931  0.55603579  1.72189807  1.22857528 -0.65699414
## Italy          -0.23505701 -0.78058870 -0.07174575 -0.46212464 -0.36883881
## Netherlands     0.05876425  1.62533538  0.82507616  0.80590030 -0.36883881
## Norway         -0.23505701 -0.78058870 -0.07174575  0.80590030  1.64824845
## Poland         -0.82269954  0.55603579 -0.07174575  0.24233366 -0.36883881
## Portugal       -1.11652080 -1.04791360 -1.86538958 -1.73014959  2.80086974
## Romania        -1.11652080 -0.51326380 -0.96856767 -0.88479963 -0.94514946
## Spain          -0.82269954 -1.31523850 -0.07174575 -1.16658295  0.78378248
## Sweden          0.05876425  0.02138599  0.82507616  1.08768362  1.07193780
## Switzerland     0.94022805  0.55603579 -0.07174575  0.94679196 -0.65699414
## United Kingdom  2.11551310 -0.51326380  1.72189807  0.52411698 -0.08068349
## USSR           -0.23505701 -0.78058870 -0.96856767 -0.03944966 -0.36883881
## West Germany    0.35258552  1.35801048  0.82507616  0.24233366 -0.36883881
## Yugoslavia     -1.70416333 -0.78058870 -1.86538958 -1.02569129 -0.94514946
##                   Cereals Starchy.Foods Pulses.Nuts.and.Oilseeds
## Albania         0.8791769    -2.0298502               1.44620630
## Austria        -0.3923599    -0.2174840              -1.03017435
## Belgium        -0.4831840     0.9907602              -0.53489822
## Bulgaria        2.2415378    -2.0298502               0.45565404
## Czechoslovakia  0.1525844     0.3866381              -1.03017435
## Denmark        -0.9373043     0.3866381              -1.03017435
## East Germany   -0.6648321     1.5948823              -1.03017435
## Finland        -0.5740081     0.3866381              -1.03017435
## France         -0.3923599     0.3866381              -0.53489822
## Greece          0.8791769    -1.4257281               2.43675857
## Hungary         0.6975288    -0.2174840               0.95093017
## Ireland        -0.7556562     0.9907602              -0.53489822
## Italy           0.4250566    -1.4257281               0.45565404
## Netherlands    -0.9373043    -0.2174840              -0.53489822
## Norway         -0.8464803     0.3866381              -0.53489822
## Poland          0.3342325     0.9907602              -0.53489822
## Portugal       -0.4831840     0.9907602               0.95093017
## Romania         1.6057694    -0.8216060               0.95093017
## Spain          -0.3015359     0.9907602               1.44620630
## Sweden         -1.1189524    -0.2174840              -1.03017435
## Switzerland    -0.5740081    -0.8216060              -0.53489822
## United Kingdom -0.7556562     0.3866381              -0.03962209
## USSR            1.0608250     0.9907602              -0.03962209
## West Germany   -1.2097765     0.3866381              -0.53489822
## Yugoslavia      2.1507138    -0.8216060               1.44620630
##                Fruits.and.Vegetables       Total
## Albania                   -1.1489125 -2.11574280
## Austria                   -0.1044466 -0.04727917
## Belgium                   -0.1044466  0.39596304
## Bulgaria                  -0.1044466  0.69145784
## Czechoslovakia            -0.1044466 -0.49052138
## Denmark                   -1.1489125  0.69145784
## East Germany              -0.1044466 -1.37700579
## Finland                   -1.6711455  0.69145784
## France                     1.4622523  1.87343706
## Greece                     1.4622523  1.87343706
## Hungary                   -0.1044466 -0.49052138
## Ireland                   -0.6266796  0.83920525
## Italy                      1.4622523 -0.34277397
## Netherlands               -0.1044466 -0.04727917
## Norway                    -0.6266796 -0.49052138
## Poland                     1.4622523  0.98695265
## Portugal                   1.9844853 -1.52475319
## Romania                   -0.6266796  0.10046823
## Spain                      1.4622523 -1.37700579
## Sweden                    -1.1489125 -0.63826878
## Switzerland                0.4177864  0.24821564
## United Kingdom            -0.6266796  0.24821564
## USSR                      -0.6266796  0.83920525
## West Germany              -0.1044466 -0.93376358
## Yugoslavia                -0.6266796  0.39596304
## attr(,"scaled: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
##                  Total
##                  86.32
## attr(,"scaled: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
##                  Total
##               6.768309
```

```
#as.matrix(scale.emp)%*%fact.load.emp%*%solve(t(fact.load.emp)%*%fact.load.emp)

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

```
## Warning in install.packages("psych", lib = "/Library/Frameworks/R.framework/
## Versions/3.5/Resources/library"): 'lib = "/Library/Frameworks/R.framework/
## Versions/3.5/Resources/library"' is not writable
```

```
## Error in install.packages("psych", lib = "/Library/Frameworks/R.framework/Versions/3.5/Resources/library"): unable to install packages
```

```
fit.pc <- principal(data1, nfactors=5, rotate="varimax")
```

```
## Warning in cor.smooth(r): Matrix was not positive definite, smoothing was done
```

```
## In factor.stats, I could not find the RMSEA upper bound . Sorry about that
```

```
## Warning in principal(data1, nfactors = 5, rotate = "varimax"): The matrix is not
## positive semi-definite, scores found from Structure loadings
```

```
fit.pc
```

```
## Principal Components Analysis
## Call: principal(r = data1, nfactors = 5, rotate = "varimax")
## Standardized loadings (pattern matrix) based upon correlation matrix
##                            RC1   RC5   RC2   RC4   RC3   h2     u2 com
## Red.Meat                  0.07  0.90  0.02 -0.01  0.23 0.86 0.1374 1.1
## White.Meat                0.94  0.15 -0.12  0.00  0.03 0.91 0.0855 1.1
## Egg                       0.59  0.67  0.09 -0.10  0.06 0.82 0.1823 2.1
## Milk                      0.23  0.57  0.24 -0.52  0.43 0.89 0.1094 3.6
## Fish                     -0.18  0.11  0.92  0.09 -0.04 0.91 0.0921 1.1
## Cereals                  -0.42 -0.63 -0.56  0.10  0.25 0.96 0.0368 3.2
## Starchy.Foods             0.55  0.01  0.69  0.05  0.00 0.79 0.2117 1.9
## Pulses.Nuts.and.Oilseeds -0.71 -0.31 -0.26  0.41 -0.07 0.84 0.1602 2.4
## Fruits.and.Vegetables    -0.06 -0.03  0.16  0.95  0.07 0.94 0.0551 1.1
## Total                     0.03  0.17 -0.09  0.04  0.98 0.99 0.0074 1.1
##
##                        RC1  RC5  RC2  RC4  RC3
## SS loadings           2.30 2.14 1.83 1.39 1.27
## Proportion Var        0.23 0.21 0.18 0.14 0.13
## Cumulative Var        0.23 0.44 0.63 0.77 0.89
## Proportion Explained  0.26 0.24 0.21 0.16 0.14
## Cumulative Proportion 0.26 0.50 0.70 0.86 1.00
##
## Mean item complexity =  1.9
## Test of the hypothesis that 5 components are sufficient.
##
## The root mean square of the residuals (RMSR) is  0.05
##  with the empirical chi square  4.86  with prob <  0.43
##
## Fit based upon off diagonal values = 0.99
```

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

```
##  [1] 4.130 1.740 1.309 1.044 0.699 0.427 0.341 0.191 0.120 0.000
```

```
fit.pc$loadings
```

```
##
## Loadings:
##                          RC1    RC5    RC2    RC4    RC3
## Red.Meat                         0.897                0.228
## White.Meat                0.936  0.150 -0.124
## Egg                       0.588  0.671        -0.103
## Milk                      0.233  0.568  0.243 -0.521  0.427
## Fish                     -0.180  0.112  0.923
## Cereals                  -0.419 -0.634 -0.559         0.252
## Starchy.Foods             0.550         0.695
## Pulses.Nuts.and.Oilseeds -0.709 -0.309 -0.259  0.412
## Fruits.and.Vegetables                   0.156  0.955
## Total                            0.168                0.977
##
##                 RC1   RC5   RC2   RC4   RC3
```

```
## SS loadings    2.299 2.139 1.830 1.386 1.268
## Proportion Var 0.230 0.214 0.183 0.139 0.127
## Cumulative Var 0.230 0.444 0.627 0.765 0.892
```

```
# Loadings with more digits
for (i in c(5,1)) { print(fit.pc$loadings[[1,i]])}
```

```
## [1] 0.2281274
## [1] 0.07404907
```

```
# Communalities
fit.pc$communality
```

```
##                Red.Meat               White.Meat                     Egg
##               0.8625590                0.9144681               0.8176674
##                    Milk                     Fish                 Cereals
##               0.8905850                0.9078512               0.9631584
##           Starchy.Foods Pulses.Nuts.and.Oilseeds    Fruits.and.Vegetables
##               0.7883228                0.8397850               0.9448934
##                   Total
##               0.9925825
```

```
#Cereals is able to restore 96% of the total variance

# Rotated factor scores
fit.pc$scores
```

```
##                        RC1        RC5        RC2         RC4        RC3
## Albania        -5.37916900 -3.6216649 -3.6291464  0.07734096 -2.6190935
## Austria         2.97591312  1.2902085 -0.3207655 -0.91529580  0.1523285
## Belgium         1.97709986  2.3347936  1.3207201 -0.43410258  0.6652047
## Bulgaria       -3.57721608 -3.5117365 -4.0830333  0.93116104  0.4919409
## Czechoslovakia  1.58199341 -0.1427662 -0.3837129 -0.24742031 -0.5733394
## Denmark         2.65898620  2.8611177  2.5831142 -2.07000038  0.9748179
## East Germany    3.07703727  0.2892108  1.7609008 -0.15976200 -1.8776263
## Finland         0.98892592  2.1318518  1.6540365 -3.20488435  1.4531404
## France          1.38357845  3.1595434  1.2018894  1.05679941  2.5762733
## Greece         -4.24503719 -1.1301364 -1.4109331  2.48665475  1.9808523
## Hungary        -0.23708684 -2.6884066 -2.3119008  0.77848839 -1.0532456
## Ireland         3.32867621  3.7822855  0.8762098 -1.69910669  1.5698400
## Italy          -2.20973495 -1.2028148 -1.4532792  1.75379920 -0.4184105
## Netherlands     2.92133471  2.0188394  0.2293759 -0.96211172  0.2170547
## Norway          0.06928655  0.9081268  2.6301684 -1.16234778 -0.5078814
## Poland          1.26985863 -0.5205768  0.4099243  1.15911974  1.1324699
## Portugal       -3.14878421 -3.3703562  3.2664394  3.58992895 -2.7856233
## Romania        -2.92936249 -3.6215274 -2.9519579  0.40059015 -0.2694526
## Spain          -2.22231411 -2.0776776  1.4177987  2.65335072 -2.1796632
## Sweden          1.69629854  2.2993811  1.9459837 -2.22359646 -0.4368219
## Switzerland     1.03749592  1.8942025 -0.5053111 -0.46744107  0.8242122
## United Kingdom  1.42506807  3.8194452  0.8887990 -1.14386138  0.8015974
## USSR           -1.07190408 -1.5337029 -0.4131728 -0.33630301  0.9112671
## West Germany    2.98326743  1.9516984  0.7813568 -0.70417689 -0.8995641
## Yugoslavia     -4.35421133 -5.3193385 -3.5035040  0.84317713 -0.1302773
```

```
# Play with FA utilities

fa.parallel(data1) # See factor recommendation
```

```
## Warning in cor.smooth(R): Matrix was not positive definite, smoothing was done
```

```
## In smc, smcs < 0 were set to .0
```

```
## Warning in cor.smooth(R): Matrix was not positive definite, smoothing was done
```

```
## In smc, smcs < 0 were set to .0
```

```
## Warning in cor.smooth(R): Matrix was not positive definite, smoothing was done
```

```
## In smc, smcs < 0 were set to .0
```

```
## Warning in cor.smooth(r): Matrix was not positive definite, smoothing was done
```

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

```
## In factor.scores, the correlation matrix is singular, an approximation is used
```

```
## Warning in cor.smooth(r): Matrix was not positive definite, smoothing was done

## Warning in cor.smooth(r): 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.obs, :
## 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.obs, :
## The estimated weights for the factor scores are probably incorrect. Try a
## different factor score estimation method.
```

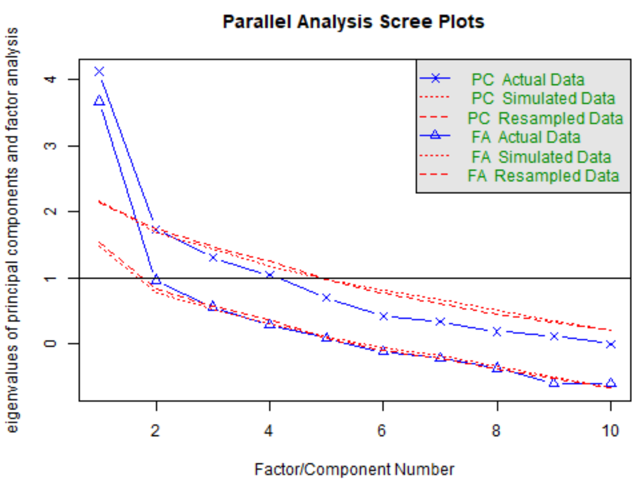
```
## Warning in fac(r = r, nfactors = nfactors, n.obs = n.obs, rotate = rotate, : An
## ultra-Heywood case was detected. Examine the results carefully
```

```
## Warning in fa.stats(r = r, f = f, phi = phi, n.obs = n.obs, np.obs = np.obs, :
## 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.obs, :
## 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
```

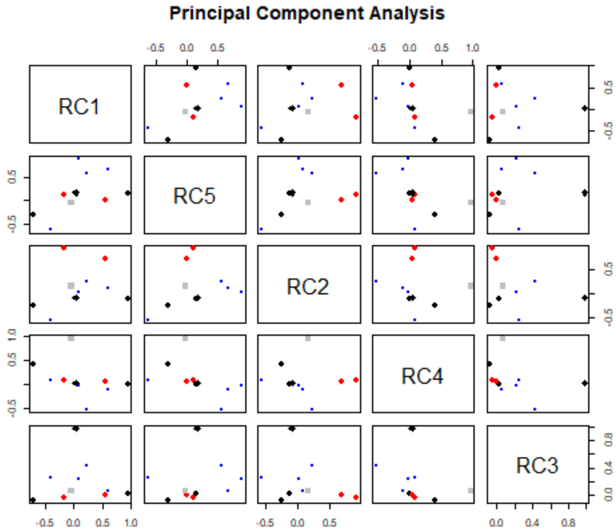
```
## Warning in fa.stats(r = r, f = f, phi = phi, n.obs = n.obs, np.obs = np.obs, :
## The estimated weights for the factor scores are probably incorrect. Try a
## different factor score estimation method.
```

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

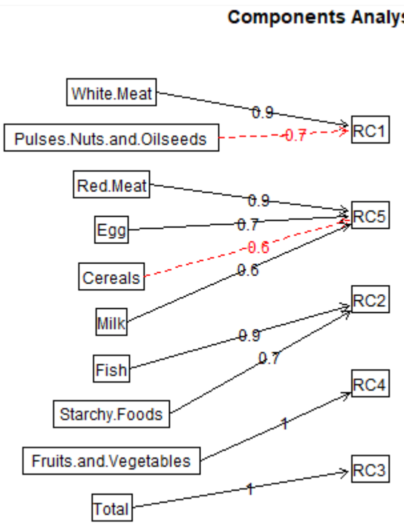



```
## Parallel analysis suggests that the number of factors =  2  and the number of components =  1
```

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




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




```r
vss(data1) # See Factor recommendations for a simple structure
```

```
## Warning in sqrt(e$values): NaNs produced
```

```
## Warning in cor.smooth(R): Matrix was not positive definite, smoothing was done
```

```
## In smc, smcs < 0 were set to .0
```

```
## Warning in cor.smooth(R): Matrix was not positive definite, smoothing was done
```

```
## In smc, smcs < 0 were set to .0
```

```
## Warning in cor.smooth(R): Matrix was not positive definite, smoothing was done
```

```
## In smc, smcs < 0 were set to .0
```

```
## Warning in cor.smooth(r): Matrix was not positive definite, smoothing was done
```

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

```
## Warning in cor.smooth(R): Matrix was not positive definite, smoothing was done
```

```
## In smc, smcs < 0 were set to .0
```

```
## Warning in cor.smooth(R): Matrix was not positive definite, smoothing was done
```

```
## In smc, smcs < 0 were set to .0
```

```
## Warning in cor.smooth(R): Matrix was not positive definite, smoothing was done
```

```
## In smc, smcs < 0 were set to .0
```

```
## Warning in cor.smooth(r): Matrix was not positive definite, smoothing was done
```

```
## Warning in fa.stats(r = r, f = f, phi = phi, n.obs = n.obs, np.obs = np.obs, :
## 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 cor.smooth(R): Matrix was not positive definite, smoothing was done
```

```
## In smc, smcs < 0 were set to .0
```

```
## Warning in cor.smooth(R): Matrix was not positive definite, smoothing was done
```

```
## In smc, smcs < 0 were set to .0
```

```
## Warning in cor.smooth(R): Matrix was not positive definite, smoothing was done
```

```
## In smc, smcs < 0 were set to .0
```

```
## Warning in cor.smooth(r): Matrix was not positive definite, smoothing was done
```

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

```
## Warning in cor.smooth(R): Matrix was not positive definite, smoothing was done
```

```
## In smc, smcs < 0 were set to .0
```

```
## Warning in cor.smooth(R): Matrix was not positive definite, smoothing was done
```

```
## In smc, smcs < 0 were set to .0
```

```
## Warning in cor.smooth(R): Matrix was not positive definite, smoothing was done
```

```
## In smc, smcs < 0 were set to .0
```

```
## Warning in cor.smooth(r): Matrix was not positive definite, smoothing was done
```

```
## In factor.stats, I could not find the RMSEA upper bound . Sorry about that
```

```
## Warning in fa.stats(r = r, f = f, phi = phi, n.obs = n.obs, np.obs = np.obs, :
## 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 cor.smooth(R): Matrix was not positive definite, smoothing was done
```

```
## In smc, smcs < 0 were set to .0
```

```
## Warning in cor.smooth(R): Matrix was not positive definite, smoothing was done
```

```
## In smc, smcs < 0 were set to .0
```

```
## Warning in cor.smooth(R): Matrix was not positive definite, smoothing was done
```

```
## In smc, smcs < 0 were set to .0
```

```
## Warning in cor.smooth(r): Matrix was not positive definite, smoothing was done
```

```
## In factor.stats, I could not find the RMSEA upper bound . Sorry about that
```

```
## Warning in fa.stats(r = r, f = f, phi = phi, n.obs = n.obs, np.obs = np.obs, :
## 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 cor.smooth(R): Matrix was not positive definite, smoothing was done
```

```
## In smc, smcs < 0 were set to .0
```

```
## Warning in cor.smooth(R): Matrix was not positive definite, smoothing was done
```

```
## In smc, smcs < 0 were set to .0
```

```
## Warning in cor.smooth(R): Matrix was not positive definite, smoothing was done
```

```
## In smc, smcs < 0 were set to .0
```

```
## Warning in cor.smooth(r): Matrix was not positive definite, smoothing was done
```

```
## Warning in fa.stats(r = r, f = f, phi = phi, n.obs = n.obs, np.obs = np.obs, :
## 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 cor.smooth(R): Matrix was not positive definite, smoothing was done
```

```
## In smc, smcs < 0 were set to .0
```

```
## Warning in cor.smooth(R): Matrix was not positive definite, smoothing was done
```

```
## In smc, smcs < 0 were set to .0
```

```
## Warning in cor.smooth(R): Matrix was not positive definite, smoothing was done
```

```
## In smc, smcs < 0 were set to .0
```

```
## Warning in cor.smooth(r): Matrix was not positive definite, smoothing was done
```

```
## Warning in fa.stats(r = r, f = f, phi = phi, n.obs = n.obs, np.obs = np.obs, :
## 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 cor.smooth(R): Matrix was not positive definite, smoothing was done
```

```
## In smc, smcs < 0 were set to .0
```

```
## Warning in cor.smooth(R): Matrix was not positive definite, smoothing was done
```

```
## In smc, smcs < 0 were set to .0
```

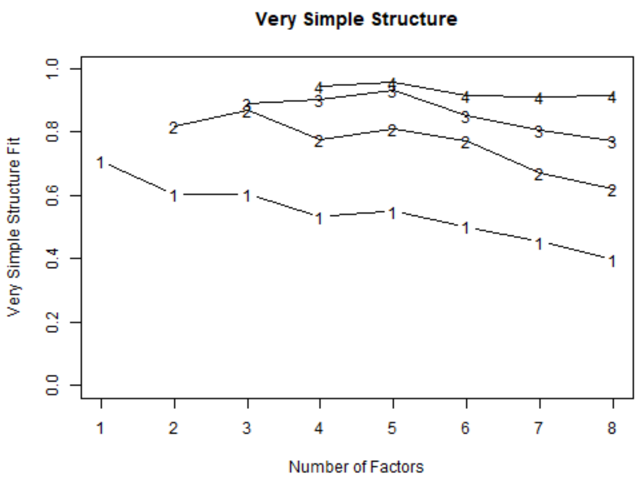
```
## Warning in cor.smooth(R): Matrix was not positive definite, smoothing was done
```

```
## In smc, smcs < 0 were set to .0
```

```
## Warning in cor.smooth(r): Matrix was not positive definite, smoothing was done
```

```
## Warning in fa.stats(r = r, f = f, phi = phi, n.obs = n.obs, np.obs = np.obs, :
## 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
```




```
##
## Very Simple Structure
## Call: vss(x = data1)
## VSS complexity 1 achieves a maximimum of 0.71  with  1  factors
## VSS complexity 2 achieves a maximimum of 0.87  with  3  factors
##
## The Velicer MAP achieves a minimum of 0.09  with  1  factors
## BIC achieves a minimum of  NA  with  5  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 0.71 0.00 0.087  35   453 3.6e-74    6.94 0.71  0.69 341   449     1.0
## 2 0.60 0.82 0.106  26   419 2.0e-72    4.25 0.82  0.78 335   416     1.3
## 3 0.60 0.87 0.145  18   390 1.0e-71    2.60 0.89  0.91 332   388     1.6
## 4 0.53 0.78 0.173  11   356 1.1e-69    1.28 0.95  1.12 321   355     1.7
## 5 0.55 0.81 0.209   5   323 1.4e-67    0.70 0.97  1.59 306   322     1.8
## 6 0.50 0.77 0.314   0   294      NA    0.61 0.97    NA  NA    NA     2.1
## 7 0.45 0.67 0.477  -4   277      NA    0.40 0.98    NA  NA    NA     2.2
## 8 0.40 0.62 1.000  -7   250      NA    0.12 0.99    NA  NA    NA     2.3
##    eChisq   SRMR eCRMS eBIC
## 1 56.2527 0.1581 0.179  -56
## 2 27.4318 0.1104 0.145  -56
## 3 12.8827 0.0757 0.120  -45
## 4  4.6981 0.0457 0.092  -31
## 5  0.9579 0.0206 0.062  -15
## 6  0.3033 0.0116    NA   NA
## 7  0.0199 0.0030    NA   NA
## 8  0.0031 0.0012    NA   NA
```

```
########### END of FCA ###########################################################################
# Conclusion: The goal of FCA is to identify groups items when considered together and explains as much of the observed co-variance as possible.
# I can see that out of 5 factors taken into consideration we have reduced the factors to two which contains most of the information in the dataset.
# When I consider 5 factors then I will be able to
# restore 87% of the total variance where RC1 contributes for most of the variance followed by RC5 which restores the second highest total variance
# White meat, pulses, nuts, oilseeds, red meat, eggs, cereals and milk are the most important protein consumed products (in grams per person per day) in
##################################################################################################
```