Protein Consumption Analysis

Saurabh Sankhe

March 29, 2019

Loading the dataset

Principal components analysis

```
#Applying PCA function on the dataset
protein_pca <- prcomp(protein_consumption, scale=TRUE)

#Printing the results of pca to console
protein_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
```

```
#Printing the summary of the pca to console
summary(protein_pca)
```

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

We get from summary the std deviation, Proportion of Variance and the cummulative variance.

In order to find the eigen values we need to square the std deviations. Which is done as below

```
#Storing and Printing the eigen values on the console
eigen_protien <- protein_pca$sdev^2
eigen_protien
```

```
##  [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
```

```
#Assigning names of PC to the values of PCA
names(eigen_protien) <- paste("PC",1:10,sep="")

#Printing the sum of eigen values to console
sum(eigen_protien)
```

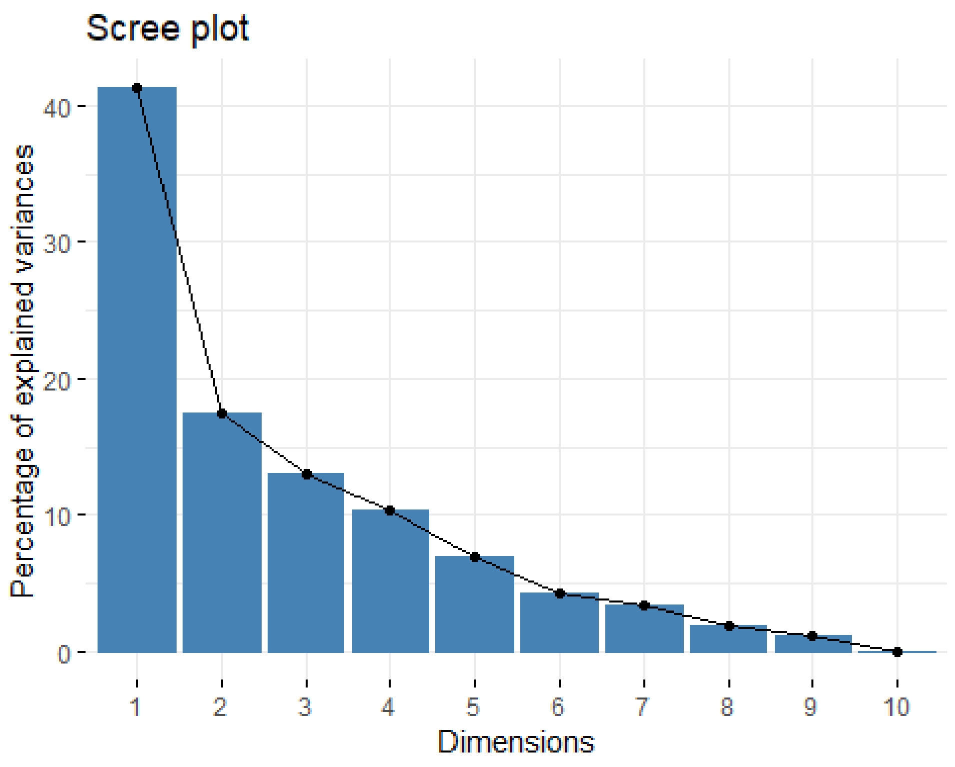
```
## [1] 10
```

Visualizing the results of PCA

```
#Importing the required libraries
library(factoextra)
```

```
## Warning: package 'factoextra' was built under R version 3.5.3

## Loading required package: ggplot2

## Welcome! Related Books: `Practical Guide To Cluster Analysis in R` at
https://goo.gl/13EFCZ
```

```
#Printing the Scree plot of PCA to console
fviz_eig(protein_pca)
```



In order to retain the values of Principal Components we have 2 approaches take in account the values whose eigen values are greater than 0.7 or we can take components who account for 70-90% variablility.

Here , in our case if we take 0.7 as threshold for eigen values we get first four components

```
#Printing the names of the eigenvalues who crosses 0.7 threshold value
names(eigen_protien[eigen_protien>0.7])

## [1] "PC1" "PC2" "PC3" "PC4"

#Printing the summary of PCA to concole
summary(protein_pca)

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

If we look at the summary we know that first four principal components accounts for roughly 82% of the total variance. Thus, we can consider our first four components as our Principal components.

Now we predict the value of these components using predict function

```
# Constructing the new dataframe with 4 Principal components and output
variable
new_protien <- predict(protein_pca)[,1:4]

#Changing the row names of the
row.names(new_protien) <- row.names(protein_consumption)

#Printing the head of the data
head(new_protien)
```

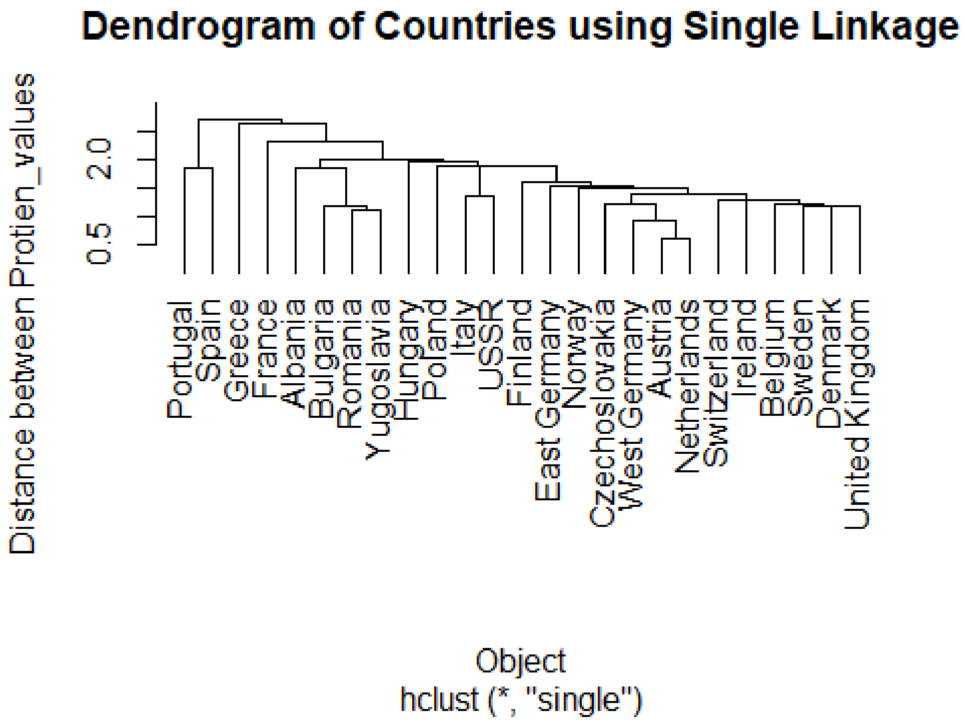
```
##                       PC1        PC2        PC3        PC4
## Albania         3.5978397 -0.6406110  1.1118946 -1.9111924
## Austria        -1.3862854 -0.7099190  1.1613381  0.9310749
## Belgium        -1.6608482  0.1078173 -0.4231894  0.2468077
## Bulgaria        2.9881523 -1.8436131 -0.0730564  0.3061617
## Czechoslovakia -0.3686147 -0.1014183  1.2155042  0.7220209
## Denmark        -2.4923551  0.1847475 -0.2075253 -0.9390683
```

Cluster Analysis (Using Principal Component Analysis):

Agglomerative Clustering Using Single, Complete and average Linkage

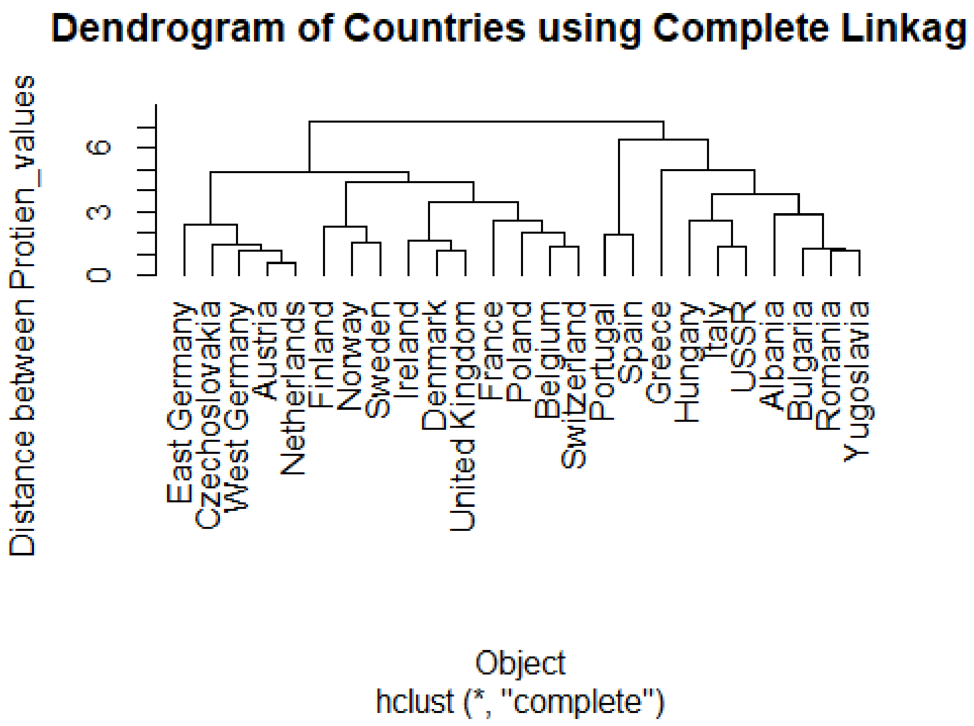
1.Single Linkage

```
#Calculating distance Matrix for the PRincipal components
dist.protien <- dist(new_protien,method='euclidean')

#Calculating Agglomerative  Clustering using single linkage
clustprotien.nn <- hclust(dist.protien, method = "single")

#Plotting the Agglomeratuve Clustering
plot(clustprotien.nn,hang=-1,xlab="Object",ylab="Distance between
Protien_values",main="Dendrogram of Countries using Single Linkage")
```



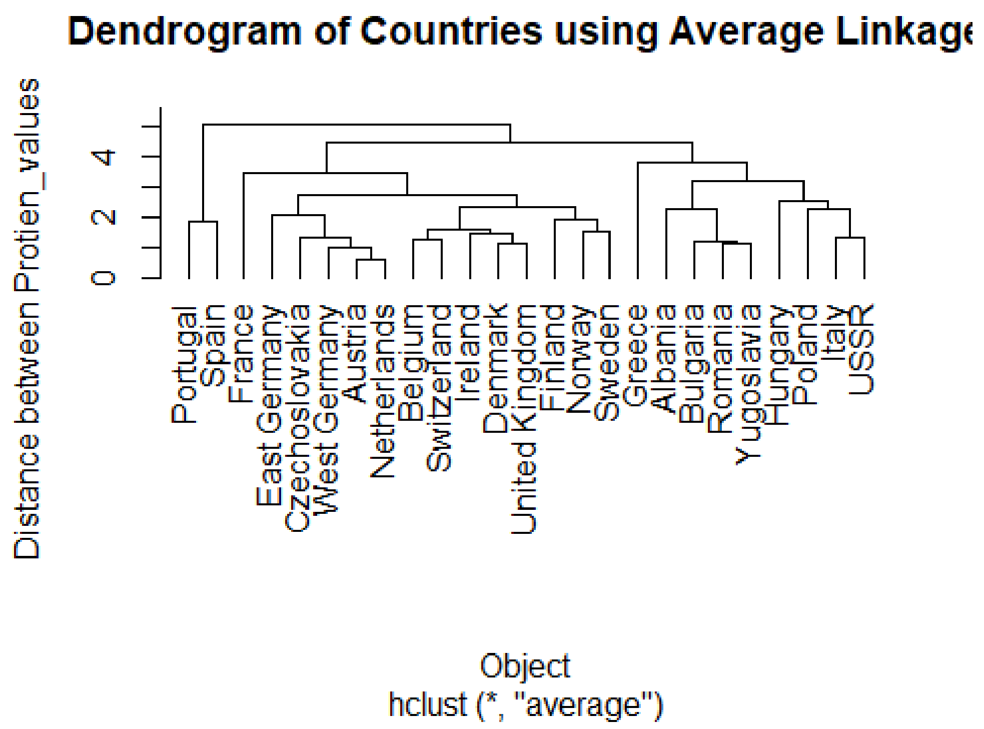
2.Complete Linkage

```
#Calculating Agglomerative  Clustering using single linkage
clustprotien.nn <- hclust(dist.protien, method = "complete")

#Plotting the Agglomeratuve Clustering
plot(clustprotien.nn,hang=-1,xlab="Object",ylab="Distance between
Protien_values",main="Dendrogram of Countries using Complete Linkage")
```



3. Average Linkage

```
#Calculating Agglomerative  Clustering using Average linkage i.e calculating
the average distance
clustprotien.nn <- hclust(dist.protien, method = "average")

#Plotting the Agglomeratuve Clustering
plot(clustprotien.nn,hang=-1,xlab="Object",ylab="Distance between
Protien_values",main="Dendrogram of Countries using Average Linkage")
```



We get the dendograms for single linkage, complete linkage and average linkage as above. We can Thus group the protien consumption by countries as shown above. If the data is large it is not advised to use agglomerative clustering instead we use k-means clustering.

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

Ans:

In order to solve this we need to use the original data

```
#Printing the head of data to console
head(protein_consumption)

##                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
```

```
#Loading the required library
library(psych)
```

```
## Warning: package 'psych' was built under R version 3.5.3
```

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

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

```
#Applying Factor Analysis on the data with 4 factors
fit.pc <- principal(protein_consumption,nfactors = 4, 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(protein_consumption, nfactors = 4, rotate =
## "varimax"): The matrix is not positive semi-definite, scores found from
## Structure loadings
```

```
#Printing the results of Factor Analysis
fit.pc
```

```
## Principal Components Analysis
## Call: principal(r = protein_consumption, nfactors = 4, rotate = "varimax")
## Standardized loadings (pattern matrix) based upon correlation matrix
##                            RC1   RC2   RC3   RC4   h2    u2 com
## Red.Meat                  0.26  0.21  0.74 -0.11 0.67 0.335 1.5
## White.Meat                0.94 -0.16  0.05  0.03 0.91 0.092 1.1
## Egg                       0.72  0.21  0.43 -0.15 0.77 0.233 1.9
## Milk                      0.32  0.26  0.68 -0.49 0.87 0.127 2.6
## Fish                     -0.15  0.92  0.03  0.11 0.88 0.121 1.1
## Cereals                  -0.58 -0.71 -0.17  0.18 0.90 0.096 2.2
## Starchy.Foods             0.53  0.60 -0.05  0.13 0.66 0.336 2.1
## Pulses.Nuts.and.Oilseeds -0.75 -0.24 -0.21  0.37 0.80 0.196 1.9
## Fruits.and.Vegetables    -0.07  0.15  0.01  0.95 0.93 0.075 1.1
## Total                    -0.03 -0.25  0.86  0.18 0.83 0.166 1.3
##
##                        RC1  RC2  RC3  RC4
## SS loadings           2.77 2.04 2.00 1.41
## Proportion Var        0.28 0.20 0.20 0.14
## Cumulative Var        0.28 0.48 0.68 0.82
## Proportion Explained  0.34 0.25 0.24 0.17
## Cumulative Proportion 0.34 0.59 0.83 1.00
##
```

```
## Mean item complexity =  1.7
## Test of the hypothesis that 4 components are sufficient.
##
## The root mean square of the residuals (RMSR) is  0.07
##  with the empirical chi square  10.96  with prob <  0.45
##
## Fit based upon off diagonal values = 0.97
```

```
#rounding the values to 3 decimal places
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
```

```
#Printing the loading data to console for the
fit.pc$loadings
```

```
##
## Loadings:
##                          RC1    RC2    RC3    RC4
## Red.Meat                  0.259  0.213  0.735 -0.111
## White.Meat                0.937 -0.161
## Egg                       0.720  0.208  0.426 -0.154
## Milk                      0.316  0.261  0.684 -0.488
## Fish                     -0.148  0.918         0.113
## Cereals                  -0.579 -0.714 -0.167  0.176
## Starchy.Foods             0.531  0.602         0.131
## Pulses.Nuts.and.Oilseeds -0.752 -0.240 -0.205  0.373
## Fruits.and.Vegetables            0.153         0.947
## Total                           -0.246  0.860  0.183
##
##                  RC1   RC2   RC3   RC4
## SS loadings    2.773 2.040 2.005 1.405
## Proportion Var 0.277 0.204 0.200 0.141
## Cumulative Var 0.277 0.481 0.682 0.822
```

Now we look at the cummunality

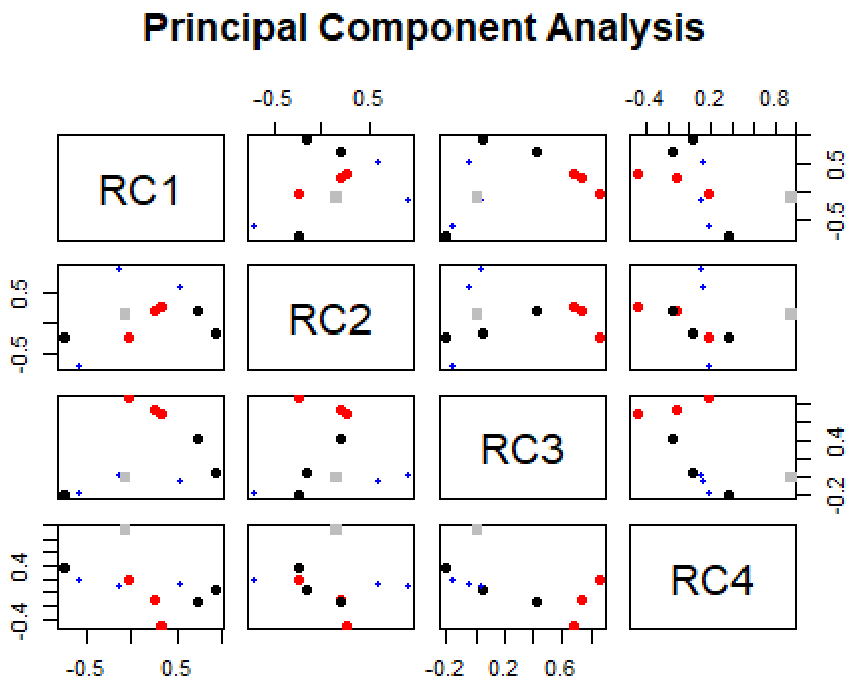
```
fit.pc$communality
```

```
##                 Red.Meat               White.Meat                      Egg
##                0.6651696                0.9078091                0.7669612
##                     Milk                     Fish                  Cereals
##                0.8730259                0.8789782                0.9035506
##            Starchy.Foods Pulses.Nuts.and.Oilseeds    Fruits.and.Vegetables
##                0.6638440                0.8043733                0.9250830
##                    Total
##                0.8340405
```

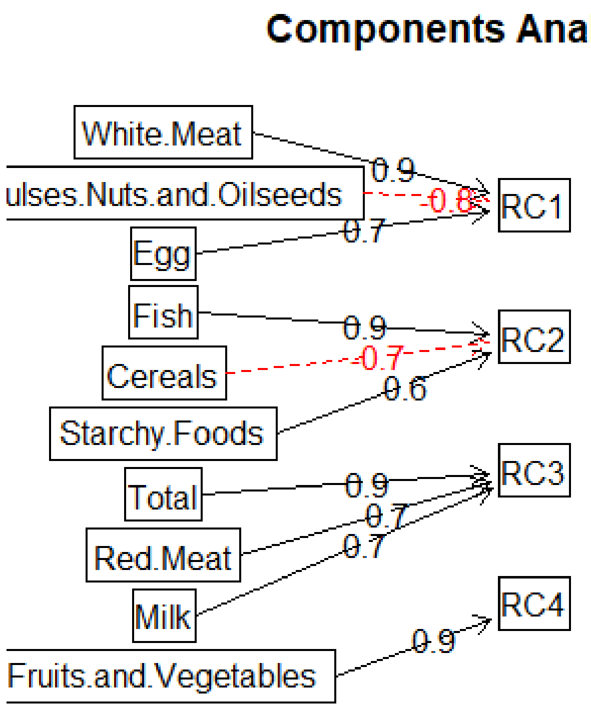
```
#Printing the scores
fit.pc$scores
```

```
##                       RC1         RC2        RC3         RC4
## Albania        -5.7605734 -3.36700491 -3.8633784 -0.39909495
## Austria         3.1694665 -0.25158518  0.7561406 -0.89662602
## Belgium         2.3868189  1.56149347  1.8337942 -0.46124833
## Bulgaria       -4.3208782 -4.55896299 -1.5315590  1.04713255
## Czechoslovakia  1.5578630 -0.41964941 -0.6288392 -0.24291402
## Denmark         3.1399401  2.71500531  2.3853571 -1.94188415
## East Germany    3.2301617  1.84901437 -1.5130562 -0.21860020
## Finland         1.3146712  1.65752382  2.4667399 -3.00560097
## France          1.8007806  1.36671635  3.8271586  1.14045289
## Greece         -4.5810717 -1.61817397  1.0655364  2.56985982
## Hungary        -0.7455549 -2.64356726 -2.4704373  0.82593410
## Ireland         3.9476155  1.21073039  3.3976235 -1.69957990
## Italy          -2.3770340 -1.36214582 -0.9545821  1.57348514
## Netherlands     3.2792689  0.45581557  1.2593824 -1.01402637
## Norway          0.3278365  2.77552853  0.1615539 -1.17661290
## Poland          0.9782808 -0.08782629  0.4161197  1.51507925
## Portugal       -3.5783940  3.05623951 -4.2281851  3.53676835
## Romania        -3.6436378 -3.44529927 -2.2388510  0.53509931
## Spain          -2.4289034  1.46500473 -2.9477324  2.46464265
## Sweden          2.2286387  2.34569499  1.0237541 -2.35625096
## Switzerland     1.3670756 -0.21504380  1.7800778 -0.57852957
## United Kingdom  2.1943713  1.57880111  2.9414916 -1.42721666
## USSR           -1.4884026 -0.92805896 -0.1658179 -0.01163555
## West Germany    3.4360982  1.19111164  0.3305165 -0.91145006
## Yugoslavia     -5.4344376 -4.33136193 -3.1028077  1.13281655
```

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



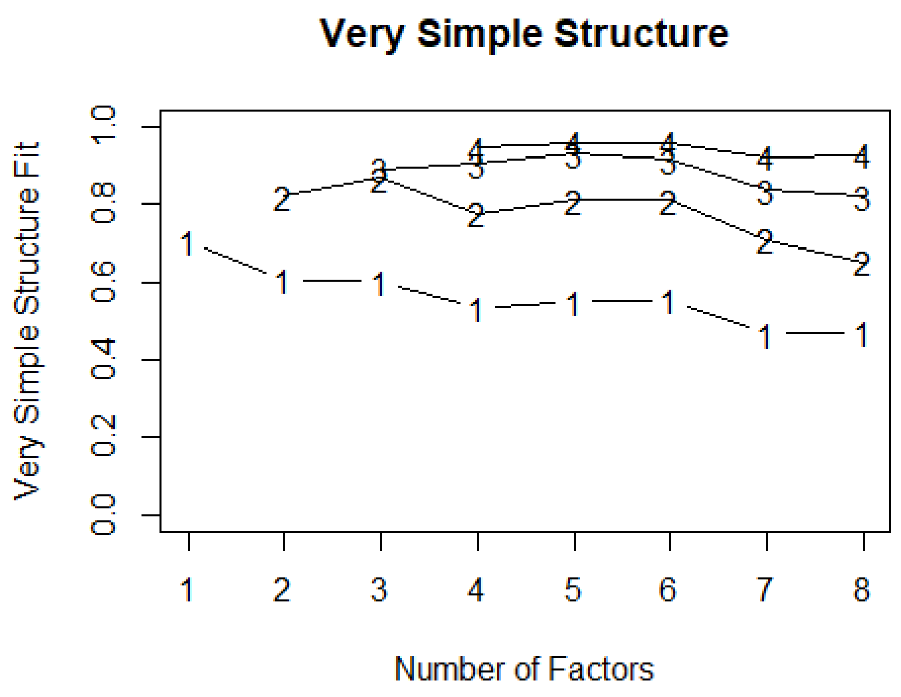
```r
#Visualize the relationship
fa.diagram(fit.pc)
```



```
#Visualizing the data
vss(protein_consumption)

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

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

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

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

## 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 extraction method.

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

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

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

## 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 extraction 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

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

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

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


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


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

## 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 extraction 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


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


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

## 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 extraction 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

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

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

## 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 extraction 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

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

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

## 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 extraction 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

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

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

## 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 extraction 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 = protein_consumption)
## 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   6.941 0.71  0.80 341   449     1.0
## 2 0.60 0.82 0.106  26   419 2.0e-72   4.245 0.82  0.91 335   416     1.3
## 3 0.60 0.87 0.145  18   390 1.0e-71   2.603 0.89  1.08 332   388     1.6
## 4 0.53 0.78 0.173  11   356 1.2e-69   1.284 0.95  1.36 321   355     1.7
## 5 0.55 0.81 0.209   5   323 1.4e-67   0.699 0.97  1.97 307   322     1.8
## 6 0.55 0.81 0.314   0   305      NA   0.411 0.98    NA  NA    NA     1.9
## 7 0.47 0.71 0.477  -4   278      NA   0.286 0.99    NA  NA    NA     2.3
## 8 0.47 0.65 1.000  -7   250      NA   0.078 1.00    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  0.9578 0.0206 0.062  -15
## 6  0.3915 0.0132    NA   NA
## 7  0.0359 0.0040    NA   NA
## 8  0.0026 0.0011    NA   NA
```