Course Title: Multivariate Analysis.
Course Number: MTH 514
Department: Mathematics And Statistics
Instructor's Name: Subhajit Dutta Sir

Numerical Assignment

Submitted by :

Shubham Kumar(181144)

Kaustav Khatua (181075)

Iris dataset

This a popular dataset to apply classification techniques as the data is small, balanced and no missing value is present. Here goal is to predict **Species** of the flower depending on its features (eg. sepal and petal length, sepal and petal width). All the feature variables are continuous.

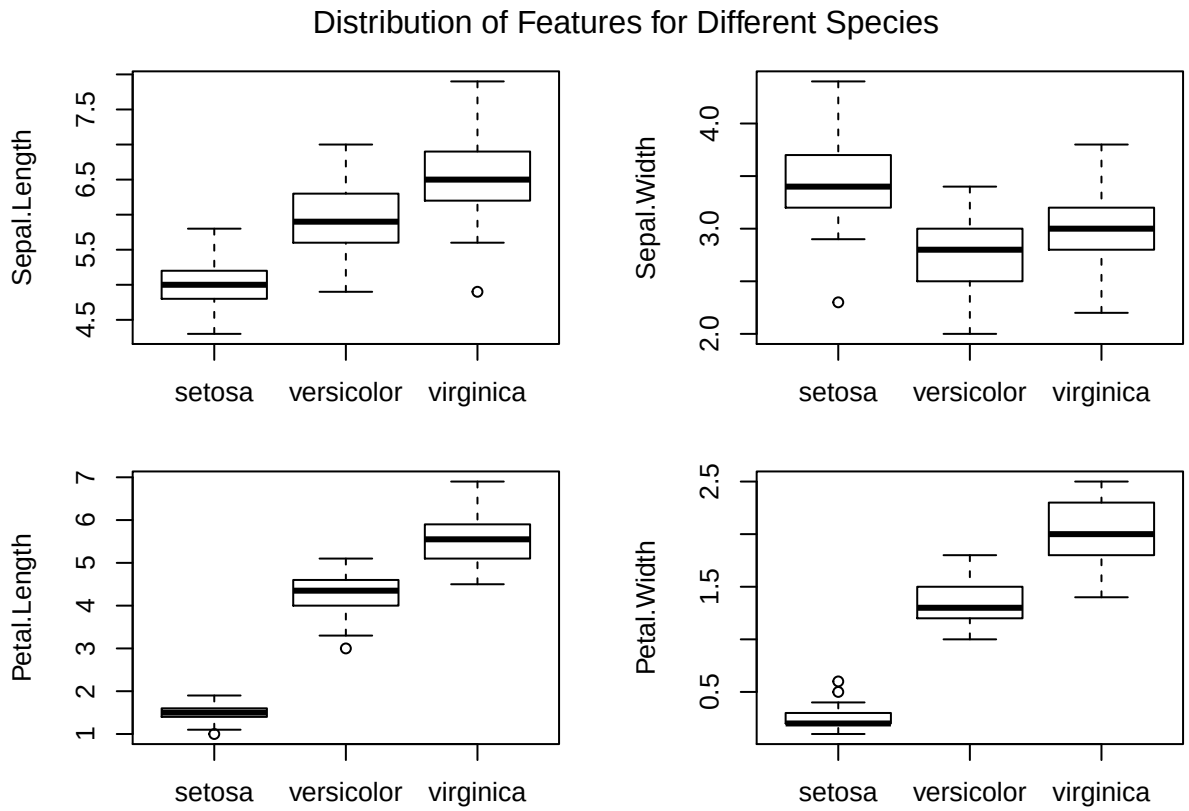
```
iris_data<-read.table("http://people.stat.sc.edu/Hitchcock/fisheriris.txt",header=TRUE)
iris_data<-iris_data[,-1]    # As first column is the index of observation, so dropping it.
```

We will use **Multinomial Logistic Regression** to classify our data.

Visualising Data

Before applying classification tachniques we have to first ensure that, we can classify data using the four features. We can answer this question by observing distribution of features across different Species. If distributions are more or less different then we can expect that we can classify data using feature variables.

```
par(mar=c(2,4.1,2,2.1), mfrow=c(2,2),oma=c(1,0,1,0))
for(i in 1:4){boxplot(iris_data[,i]~iris_data[,5],ylab=colnames(iris_data)[i])}
mtext("Distribution of Features for Different Species",outer=TRUE,line=-1)
```



From above diagram we can see that features are more or less differently distributed across different species. So, we expect that classifying Species using these feature variables will be meaningful.

It may be noted that, there is nothing to worry about the **outliers** shown in the above plots. These are not the usual outliers, they are not just following the characteristic of their cluster and falling into the range of other clusters. It is usual in classification.

Model Fitting and Performance Check

We first split the dataset into training, testing data of size 110 and 40 respectively; then fiitted model on training dataset.

```
set.seed(4)
random_numbers<-sample(1:150,size=110)
training_data<-iris_data[random_numbers,]
test_data<-iris_data[-random_numbers,]
```

There are many packages available in R to perform this technique. But we used **multinom** function from **nnet** package as it is easy to apply.

```
library(nnet)
```

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

```
logistic<-multinom(Species~.,data=training_data,maxit=1000)
```

```
summary(logistic)
```

```
## Call:
## multinom(formula = Species ~ ., data = training_data, maxit = 1000)
##
## Coefficients:
##            (Intercept) Sepal.Length Sepal.Width Petal.Length Petal.Width
## versicolor    11.62407   -0.5045477   -10.68799     9.171049   -1.511381
## virginica    -13.09618   -1.9207677   -18.59954    15.925196   12.324955
##
## Std. Errors:
##            (Intercept) Sepal.Length Sepal.Width Petal.Length Petal.Width
## versicolor    30.74340     131.7508    185.5276     70.49025    20.44192
## virginica     31.13954     131.7687    185.6250     70.59956    21.04638
##
## Residual Deviance: 9.930246
## AIC: 29.93025
```

Now we check for significance of regressors.

```
logistic_coefficients<-coefficients(logistic)    # Extracting coefficients.
logistic_sd<-summary(logistic)$standard.errors    # Extracting std. deviations of coefficients.

z_values<-logistic_coefficients/logistic_sd
p_values<-(1-pnorm(abs(z_values)))*2
p_values
```

```
##            (Intercept) Sepal.Length Sepal.Width Petal.Length Petal.Width
## versicolor   0.7053564    0.9969445   0.9540604    0.8964843   0.9410618
## virginica    0.6740733    0.9883698   0.9201859    0.8215353   0.5581381
```

All p-values are greater than 0.05 indicating each regressor is significant.

Now we observe performance of this model on test dataset.

```
# Test data has class labels in its fifth column, so we have to exclude it.
# We add one column of 1 for intercept.
x<-cbind(1,test_data[,-5])

true_labels<-test_data[,5]
predicted_labels<-rep("a",nrow(x))

for(i in 1:nrow(x)){
  denominator<-1+exp( as.numeric(x[i,]) %*% logistic_coefficients[1,] )+
               exp( as.numeric(x[i,]) %*% logistic_coefficients[2,] )

  p_setosa<-1/denominator
  p_versicolor<-exp(as.numeric(x[i,])%*%logistic_coefficients[1,])/denominator
  p_virginica<-1-p_setosa-p_versicolor

  probability_vector<-c(p_setosa,p_versicolor,p_virginica)
  names(probability_vector)<-c("setosa","versicolor","virginica")

  predicted_labels[i]<-names(which.max(probability_vector))
}
```

```
table(true_labels,predicted_labels)    # One column has one fixed label of predicted labels
```

```
##             predicted_labels
## true_labels  setosa versicolor virginica
##   setosa         14          0         0
##   versicolor      0         11         2
##   virginica       0          0        13
```

Considering the number of misclassified instances with respect to total size of the test data we can imply that Logistic Regression can classify the data well.

Warning:

Though the there is no problem in prediction; yet we have another issue. The problem is that if the data changes slightly the coefficient estimates are **changing significantly**. Sometimes the algorithm of fitting model is **not** even **converging** after 1000 steps.

```
m=multinom(Species~.,data=iris_data[c(1:30,51:80,101:130),c(-3,-4)],maxit=10000)
n=multinom(Species~.,data=iris_data[c(1:40,51:90,101:140),c(-3,-4)],maxit=10000)
```

```
coefficients(m)
```

```
##            (Intercept) Sepal.Length Sepal.Width
## versicolor   -5.109419     20.58623   -34.60508
## virginica   -13.787594     21.95120   -34.58887
```

```
coefficients(n)
```

```
##            (Intercept) Sepal.Length Sepal.Width
## versicolor   -19.31462     33.26502   -51.12993
## virginica    -30.29828     34.90743   -50.90905
```

The reason behind the problem may be that,
1)The training data is too small to fit a model with 10 coefficients to be estimated.
2)May be **Multicollinearity** is present in the data.

```
cor(iris_data[,-5])
```

```
##              Sepal.Length Sepal.Width Petal.Length Petal.Width
## Sepal.Length    1.0000000  -0.1175698    0.8717538   0.8179411
## Sepal.Width    -0.1175698   1.0000000   -0.4284401  -0.3661259
## Petal.Length    0.8717538  -0.4284401    1.0000000   0.9628654
## Petal.Width     0.8179411  -0.3661259    0.9628654   1.0000000
```

Three correlations are very high indicating multicollinearity is present.
In both the cases reducing number of variables may solve the problem. Now, **Principal Component Analysis** and **Factor Analysis** are two useful methods to reduce number of variables. We chose PCA.

Principal Component Analysis

```
pr_comp<-prcomp(iris_data[,-5])
pr_comp_var<-(pr_comp$sdev)^2  # variances of PCs/eigen values of the covariance matrix of the data.
cumsum(pr_comp_var)/sum(pr_comp_var)  # Proportion of  variation explained by Principal Components.
```

```
## [1] 0.9246187 0.9776852 0.9947878 1.0000000
```

The first principal component alone explains **92%** of total variability in the original data. So, may be the first PC will be enough for model building.

Now we check whether coefficient estimates are stable or not (problem that we were facing).

```
rotated_data<-as.data.frame(pr_comp$x)     # Extracting rotated data matrix.
rotated_data$V5<-iris_data[,5]             # Adding the class labels.
model_one=multinom(V5~PC1,data=rotated_data[c(1:30,51:80,100:130),],maxit=1000)
model_two=multinom(V5~PC1,data=rotated_data[c(1:40,51:90,100:140),],maxit=1000)
```

```
print(coefficients(model_one))
```

```
##            (Intercept)      PC1
## versicolor    16.34421 11.50189
## virginica     10.15304 16.22919
```

```
print(coefficients(model_two))
```

```
##            (Intercept)      PC1
## versicolor   16.094842 11.49974
## virginica     9.464539 16.57583
```

The variation in magnitude of coefficient estimates are smaller than before. So, we think we solved our problem to some extent.

Final Model

We will finally fit model with rotated training data and check performance of the fitted model on the test data.

```
new_index<-sample(1:150,110)
new_training_data<-rotated_data[new_index,]
new_test_data<-rotated_data[-new_index,]
new_model<-multinom(V5~PC1,data=new_training_data,maxit=1000)
new_coefficients<-coefficients(new_model)
```

```
new_x<-cbind(1,new_test_data[,-5])

new_true_labels<-new_test_data[,5]
new_predicted_labels<-rep("a",nrow(x))

for(i in 1:nrow(new_x)){
  denominator<-1+exp( as.numeric(new_x[i,1:2]) %*% new_coefficients[1,] )+
               exp( as.numeric(new_x[i,1:2]) %*% new_coefficients[2,] )

  p_setosa<-1/denominator
  p_versicolor<-exp(as.numeric(new_x[i,1:2])%*%new_coefficients[1,])/denominator
  p_virginica<-1-p_setosa-p_versicolor

  probability_vector<-c(p_setosa,p_versicolor,p_virginica)
  names(probability_vector)<-c("setosa","versicolor","virginica")

  new_predicted_labels[i]<-names(which.max(probability_vector))
}
table(new_true_labels,new_predicted_labels)
```

```
##                new_predicted_labels
## new_true_labels setosa versicolor virginica
##      setosa         14          0         0
##      versicolor      0         11         4
##      virginica       0          1        10
```

Performance on test data is good. So, **new_model** is our final model.

SAT Scores Data

The dataset has four columns, first three are scores of individuals in three different subjects and the fourth column is on his/her academic status (graduate or non graduate). Here goal is to predict whether an individual will graduate or not, using his/her scores in these subjects.

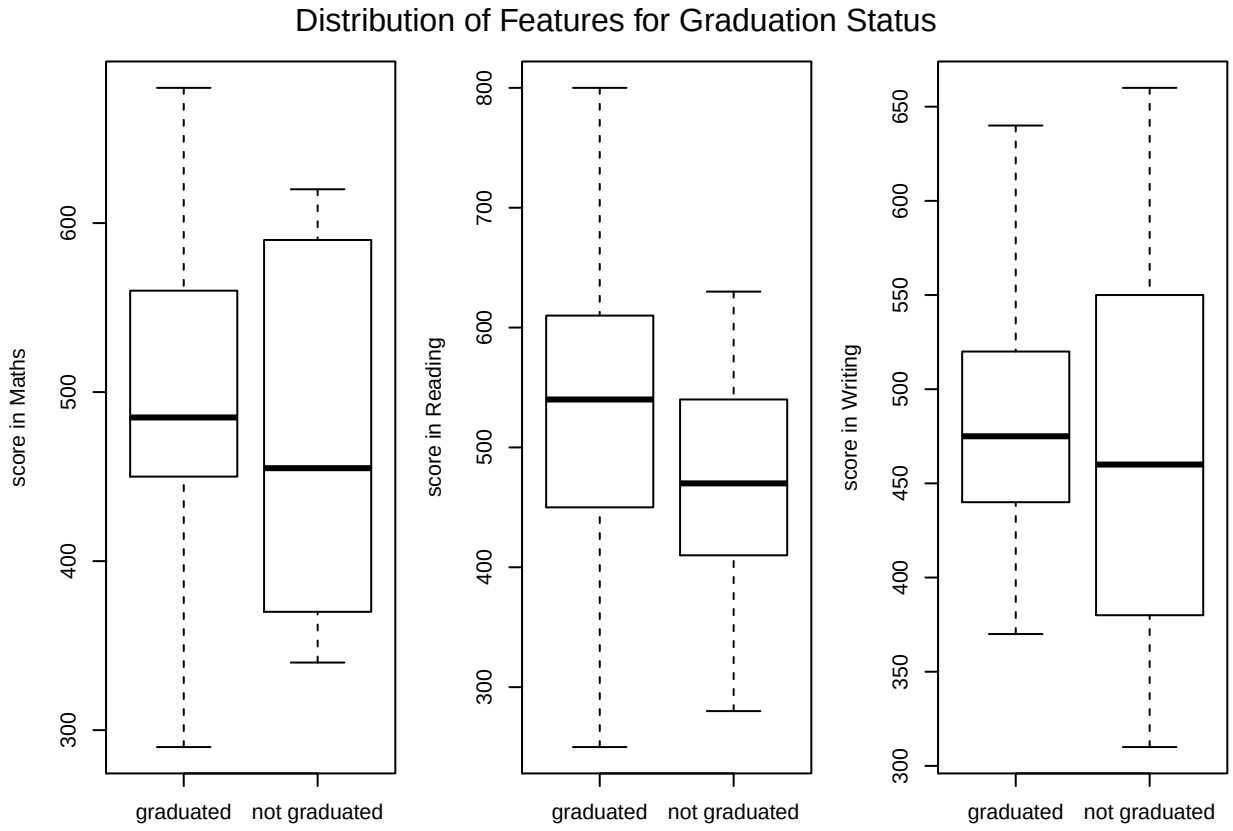
```
satgradu <- read.table("http://www.stat.sc.edu/~hitchcock/satgradu.txt", header=T)

for (i in 1:length(satgradu$gradu)){              # Re-encoding to have better readability.
  if(satgradu$gradu[i]==1){satgradu$gradu[i]="graduated"}
  else{satgradu$gradu[i]="not graduated"}
}
satgradu$gradu<-as.factor(satgradu$gradu)
```

Visualising Data

Before applying classification techniques we have to make sure that these three variables are capable of classifying the data. Boxplots serve the purpose to some extent.

```
par(mar=c(2,4.5,2,0.5),mfrow=c(1,3),oma=c(1,0,1,0))
boxplot(math~gradu,data=satgradu,ylab="score in Maths")
boxplot(reading~gradu,data=satgradu,ylab="score in Reading")
boxplot(writing~gradu,data=satgradu,ylab="score in Writing")
mtext("Distribution of Features for Graduation Status",outer=TRUE,line=-1)
```



Medians of all the variables are higher for graduated students. It also appears that candidate succesfully graduated tends to have higher marks in reading and spread of the boxplot for graduated student is less compared to the non graduated student.

Applying Classification Techniques

The dataset is too small to split into train and test set. So, we apply all he techniques on the full dataset.

Logistic Regression

```
library(reshape2)
satgradu$gradu<-relevel(satgradu$gradu,ref="not graduated")
logistic_regression<-glm(gradu~.,data=satgradu,family=binomial)
summary(logistic_regression)
```

```
##
## Call:
## glm(formula = gradu ~ ., family = binomial, data = satgradu)
##
## Deviance Residuals:
##      Min        1Q    Median        3Q       Max
## -1.95784   0.04024   0.65174   0.78397   1.09748
##
## Coefficients:
##               Estimate Std. Error z value Pr(>|z|)
## (Intercept) -1.4940606  2.5348993  -0.589    0.556
## math         0.0012308  0.0047303   0.260    0.795
## reading      0.0037698  0.0034071   1.106    0.269
## writing      0.0002277  0.0052656   0.043    0.966
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 44.987  on 39  degrees of freedom
## Residual deviance: 43.116  on 36  degrees of freedom
## AIC: 51.116
##
## Number of Fisher Scoring iterations: 4
```

No significant regressor! But we will not care about it, if the model correctly classifies most of the observations in the data.

```
glm.prob=predict(logistic_regression,satgradu,type = "response")
glm.pred=rep("not graduated",40)
glm.pred[glm.prob>0.5]="graduated"
table(glm.pred,satgradu$gradu)
```

```
##
## glm.pred    not graduated graduated
##   graduated            10        30
```

$10 \div 40 \times 100 = 25\%$ data are misclassified, which may not be too bad. But the problem is that all are classified as graduated; i.e. all the predicted probabilities are more than 0.5.

One may think of increasing the cut-off, as it will give not-graduated predictions also. But doing so is not always meaingful; because we do it only when we want to become too ceratain about some event. Here, if our purpose is to, discard all the fraud degree cases completely, then increasing cut-off makes sense. In that case setting the cut-off to 0.85 may detect all the non graduates completely (in the cost of making many graduates classified as non graduates). But, our purpose is to just classify data correctly. Problem here is that, the estimated probabilities are mixed up for the two classes and Logistic Regression can not find any linear function of the feature vector to discriminate between classes.

Now, one may be suspicious about the fact that, only 25% of the data is not graduated, so problem may lie there; i.e. data is suffering from **class imbalance** problem. We first try to address it using **over sampling**.

To apply over sampling we replicated not graduated data two times and added that to original data. On this data we again applied logistic regression and checked model performance.

```
data<-satgradu
for(i in 1:2){          # Creating balanced data.
  data<-rbind(data,satgradu[satgradu$gradu=="not graduated",])
}

data$gradu<-relevel(data$gradu,ref="not graduated")
balanced_model<-glm(gradu~.,data=data,family="binomial")

glm.prob=predict(balanced_model,data,type = "response")
glm.pred=rep("not graduated",60)
glm.pred[glm.prob>0.5]="graduated"
table(glm.pred,data$gradu)
```

```
##
## glm.pred        not graduated graduated
##   graduated                12        19
##   not graduated            18        11
```

No significant change in misclassification rate ($(12 + 11) \div 60 \approx 22\%$). So, we have to go for different classificaion technique.

Discriminant Analysis

One may try LDA next. But concepts of LDA are built on the basis of normality assumption, narrowing down its applicability. On the other hand, we don't need any assumption to hold, for applying Logistic Regression. Even if the assumptions hold, it is unlikely that LDA will produce remarkably different linear classifier. So, we skip LDA. We apply QDA next, as it produces quadratic function to classify data.

```
library(MASS)
```

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

```
qda.fit <- qda(gradu ~., data=satgradu)
qda.pred=predict(qda.fit,satgradu)
qda.class=qda.pred$class
table(qda.class,satgradu$gradu)
```

```
##
## qda.class       not graduated graduated
##   not graduated             2         0
##   graduated                 8        30
```

Results are similar to Logistic Regression. But, before switching classification technique, we apply QDA on over sampled data.

```
qda.fit_over <- qda(gradu ~., data=data)
qda.pred_over<-predict(qda.fit_over,data)
qda.class_over<-qda.pred_over$class
table(qda.class_over,data$gradu)
```

```
##
## qda.class_over  not graduated graduated
##   not graduated            24         9
##   graduated                 6        21
```

Misclassification rate remains same. But this is more acceptable than any of the results obtained before. So, we can conclude that the data is not linearly classifiable and higher order functions may yield better result, also the original data is not sufficient for classification.

SVM

Next, we try SVM both on original and over sampled data.

```
library(e1071)
fit_full=svm(gradu~.,data=satgradu)    # Using radial kernel.
prediction_full=predict(fit_full,satgradu[,-4])
table(prediction_full,satgradu$gradu)
```

```
##
## prediction_full not graduated graduated
##   not graduated             2         0
##   graduated                 8        30
```

```
fit=svm(gradu~.,data=data)
prediction=predict(fit,data[,-4])
table(prediction,data$gradu)
```

```
##
## prediction      not graduated graduated
##   not graduated            30         5
##   graduated                 0        25
```

SVM on over sampled data produces most accurate results.
But can an SVM model trained on training data produce satisfactory results on test data?

```
index<-sample(1:60,size=40)
train<-data[index,]
test<-data[-index,]
fit_train=svm(gradu~.,data=train)
prediction_test=predict(fit_train,test[,-4])
table(prediction_test,test$gradu)
```

```
##
## prediction_test not graduated graduated
##   not graduated             8         2
##   graduated                 2         8
```

So, misclassification rate on test data is 20%. So, we can conclude that SVM is classifying data well and we take it as our final model.

Note: SVM transforms data into higher dimension with its kernel trick and classifies data. May be this is the reason that we get best results from SVM in our case.

Foodstuffs Content Data

The data has 27 observations. In every row the first column is food code and other five columns are amount of nutrient ingredients in it. Based on these five features we have to seperate out similar type of foods (**ie.** we have to find clusters).

```
food <- read.table("http://people.stat.sc.edu/hitchcock/foodstuffs.txt", header=T)
food <- food[,-1]    # Food code is not required to find clusters.
```

Data Preparation

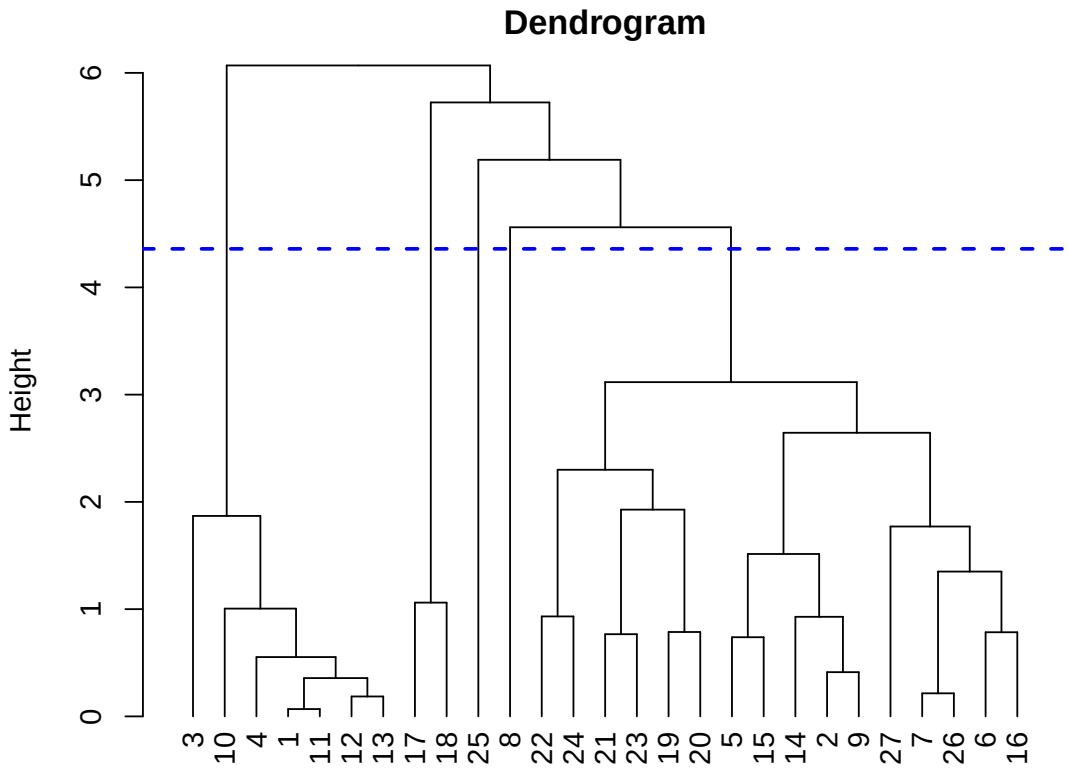
Most of the clustering techniques are distance based. So, prior to applying any clustering techniques we have to scale the data, as the measurement magnitudes are not in same scale (Calcium and Iron have very small measurements, Protein and Fat have measurements in the scale of 10, whereas Energy has measurement in scale of 100). If we don't scale data then Calcium and Iron will lose significance in forming clusters. Another advantage of scaling is that clustering techniques converge faster on scaled data. We will apply **Hierarchical Clustering** technic on scaled data.

```
scaled_data <- scale(food)
```

Forming Clusters

We will use **complete linkage** hierarchical clustering method.

```
par(mar=c(1.5,4.1,1,2.1))
clusters <- as.dendrogram(hclust(dist(scaled_data), method="complete"))
plot(clusters,main="Dendrogram",ylab="Height")+
  abline(h=heights_per_k.dendrogram(clusters)["5"]-0.2,lwd=2,lty=2,col="blue")
```



From dendrogram we can see, a horizontal line, at height just more than 4, will cut 5 vertical lines; **ie** if we set **measure of dissimilarity** (height) not to be more than 4.5 (approximately) then we will get 5 clusters. The number of clusters will decrease gradually with the increase in height and at height equal to 6 we will get one cluster.

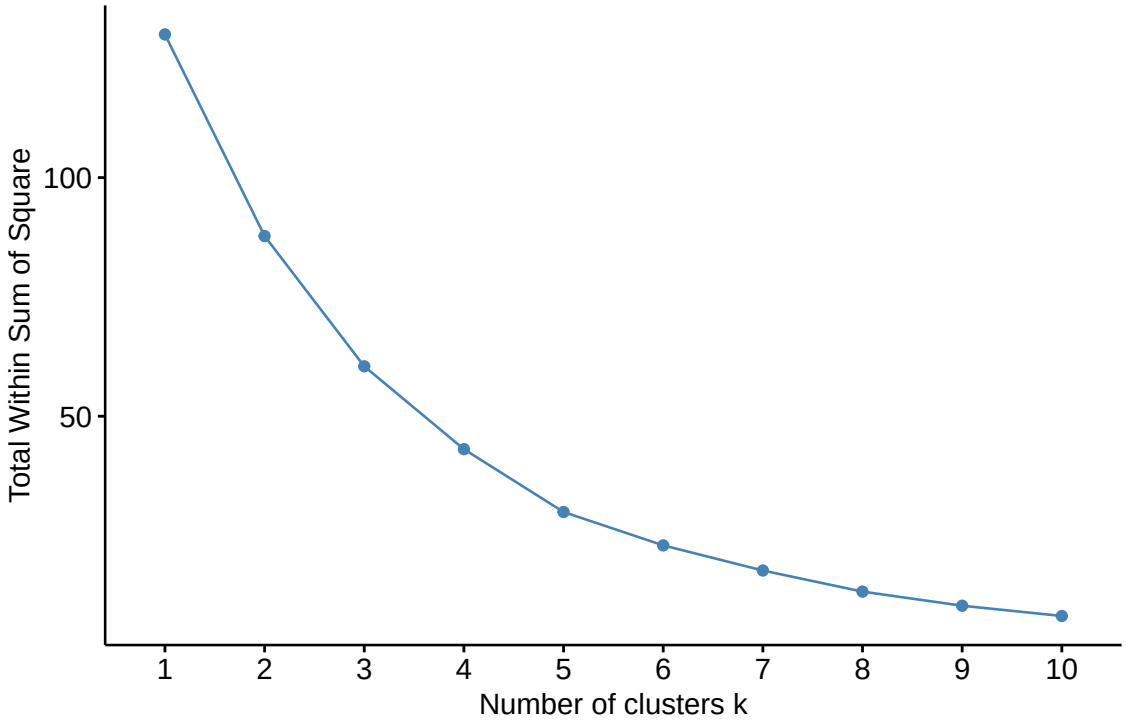
Determining Appropriate Number of Clusters

But we don't know appropriate number of clusters. We will determine it by **Elbow Method**; i.e. we will choose the number after which the total within cluster sum of squares does not change significantly.

```
library(factoextra)
```

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

```
fviz_nbclust(scaled_data, FUN = hcut, method = "wss")+ggtitle("")
```



One may pick 4 or 5 as the appropriate number of clusters. We go with 4, as change in total wss from 4 to 5 and 5 to 6 is more or less same.

So now we obtain 4 clusters.

```
clust_four<-cutree(clusters,k=4)    # clust_two contains members of each clusters.
table(clust_four)
```

```
## clust_four
##  1  2  3  4
##  7 17  2  1
```

One may be suspicious about k=4, as there was a cluster containing only one observation. But the fact is that the observation had calcium measurement very different from others. Which will be clear from following code blocks.

```
print(tail(sort(food$Calcium))); print(subset(food,clust_four==4))
```

```
## [1]  74  82  98 157 159 367
```

```
##    Energy Protein Fat Calcium Iron
## 25    180      22   9     367  2.5
```

So, may be due to the high calcium level the observation in cluster 4,can not fit into any other cluster.

Pottery Data

The dataset taken gives the chemical composition of 45 different pottery . The kiln site refers to the place at which they are found .All the features variable are continuous. Our interest centres on whether, on the basis of their chemical compositions, the pots can be divided into distinct groups, and how these groups relate to the kiln site.

Reading of Dataset

```
potteryTable63 <- read.table("http://people.stat.sc.edu/Hitchcock/potteryTable63.txt",header=TRUE)
pottery<- potteryTable63[,-1]       # As first column is the index of observation, so dropping it.
kiln_site<- pottery[,1]
```

Since there are variations among variables/feature vectors, they need to be scaled or normalised.

```
pottery_scaled<-as.data.frame(scale(pottery))
pottery_scaled[,1]<-as.factor(kiln_site)  # Reassigning value of kiln site which got
# change due to scaling.
```

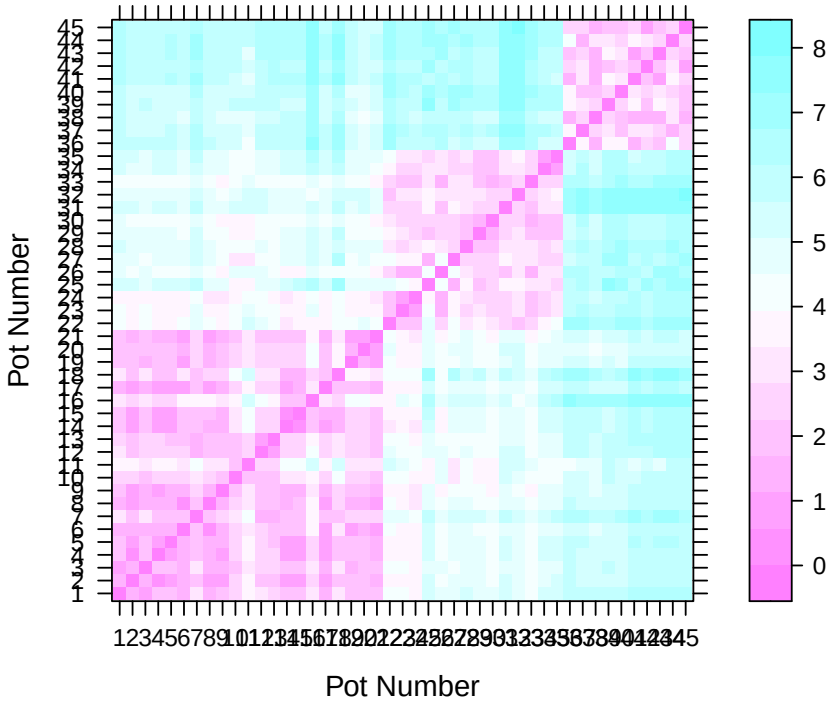
Multivariate Visualisation of the Data

We start our analysis with computing the dissimilarity matrix containing the Euclidean distance of the chemical measurements on all 45 pots.

```
library("lattice")                #Required for image plot
```

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

```
pottery_dist <- dist(pottery_scaled[, colnames(pottery_scaled) != "kiln"])
levelplot(as.matrix(pottery_dist), xlab = "Pot Number",ylab = "Pot Number")
```



This levelplot leads to the impression that there are at least three distinct groups with small inter-cluster differences (the dark rectangles) whereas much larger distances can be observed for all other cells.

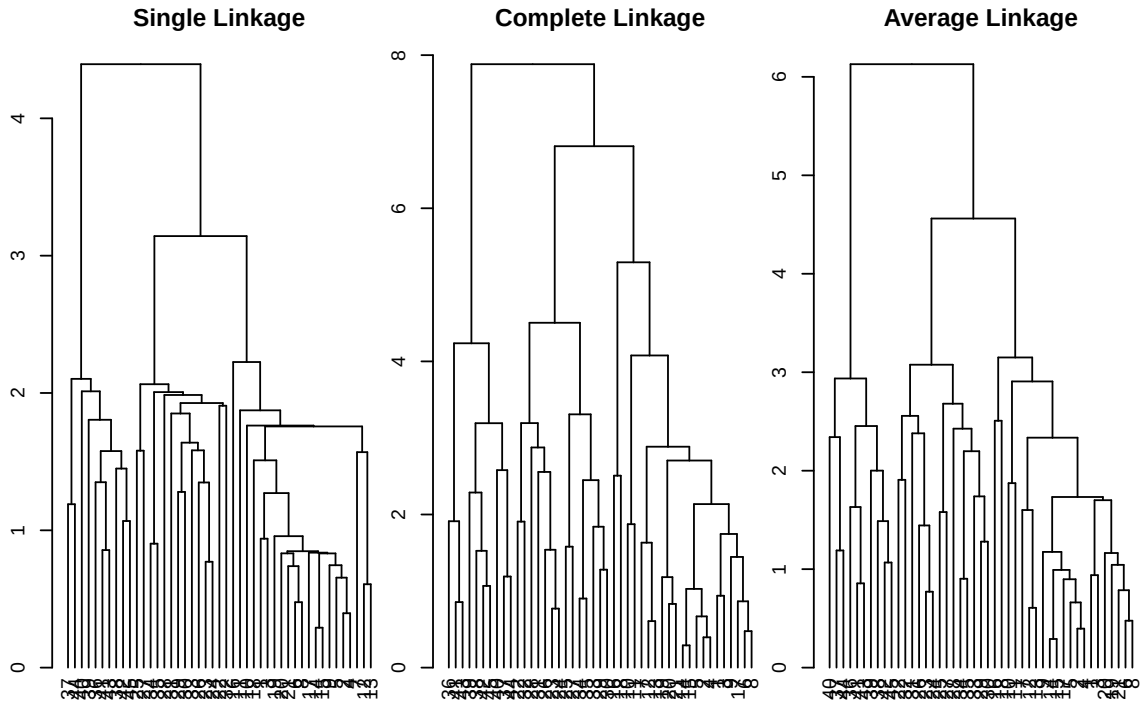
Applying Clustering Techniques

We will try out a series of clustering techniques and compare them.

1. Agglomerative Hierarchical Clustering

```
pottery_single <- hclust(pottery_dist, method = "single")     # single linkage.
pottery_complete <- hclust(pottery_dist, method = "complete")  # complete linkage.
pottery_average <- hclust(pottery_dist, method = "average")  # average linkage.
```

```
par(mar=c(2,2,2,0),mfrow=c(1,3),oma=c(1,0,1,0))
plot(as.dendrogram(pottery_single), main = "Single Linkage")
plot(as.dendrogram(pottery_complete), main = "Complete Linkage")
plot(as.dendrogram(pottery_average), main = "Average Linkage")
```



Results obtained from three methods are slightly different. Which was expected; as, Single linkage considers minimum distance between clusters; where as complete and average linkage considers maximum and average distance respectively.

Now we calculate Agglomerative coefficients. It measures the dissimilarity of an object to the first cluster it joins, divided by the dissimilarity of the final merger in the cluster analysis, averaged across all samples.

```
agglo_coeffs<-data.frame("pottery_single"=cluster::coef.hclust(pottery_single),
                         "pottery_average"=cluster::coef.hclust(pottery_average),
                         "pottery_complete"=cluster::coef.hclust(pottery_complete))
agglo_coeffs
```

```
##   pottery_single pottery_average pottery_complete
## 1      0.7389701       0.7966937        0.8368577
```

Consequently, it appears that an agglomerative approach with complete linkage provides the optimal results. But in fact average linkage method is providing more serviceable clustering.

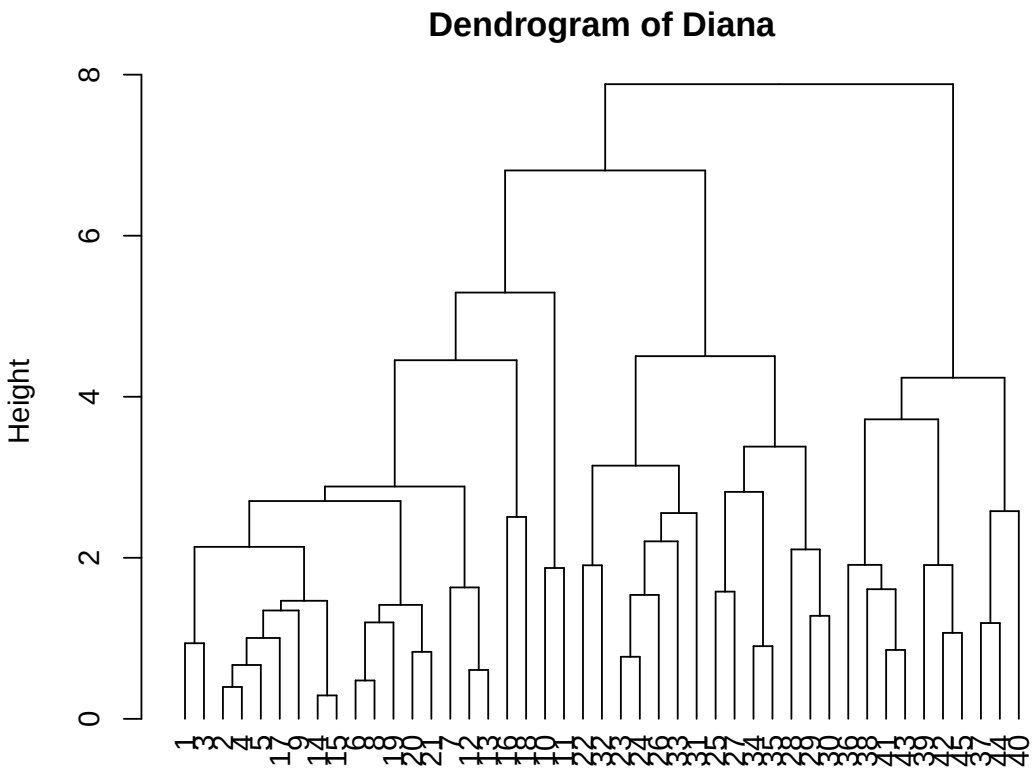
2. Divisive Hierarchical Clustering

```
dv <- cluster::diana(pottery_dist, metric = "manhattan", stand = TRUE)
dv$dc
```

```
## [1] 0.8386789
```

Divisive coefficient closer to one suggests stronger group distinctions. Here it is 0.83867,which is satisfactory.

```
par(mfrow=c(1,1),mar=c(1,4.2,2,2.1))
plot(as.dendrogram(dv),ylab="Height",main="Dendrogram of Diana")
```



3. k-Means Clustering

```
k2 <- kmeans(pottery_scaled, centers = 2)     # Assuming there are 2 clusters.
k3 <- kmeans(pottery_scaled, centers = 3)     # Assuming there are 3 clusters.
k4 <- kmeans(pottery_scaled, centers = 4)     # Assuming there are 4 clusters.
k5 <- kmeans(pottery_scaled, centers = 5)     # Assuming there are 5 clusters.
wss_mat<-data.frame("Number of Clusters"=2:5,
            "Between SS"=c(k2$betweenss,k3$betweenss,k4$betweenss,k5$betweenss),
            "Tot. Within SS"=c(k2$tot.withinss,k3$tot.withinss,k4$tot.withinss,k5$tot.withinss))
wss_mat
```

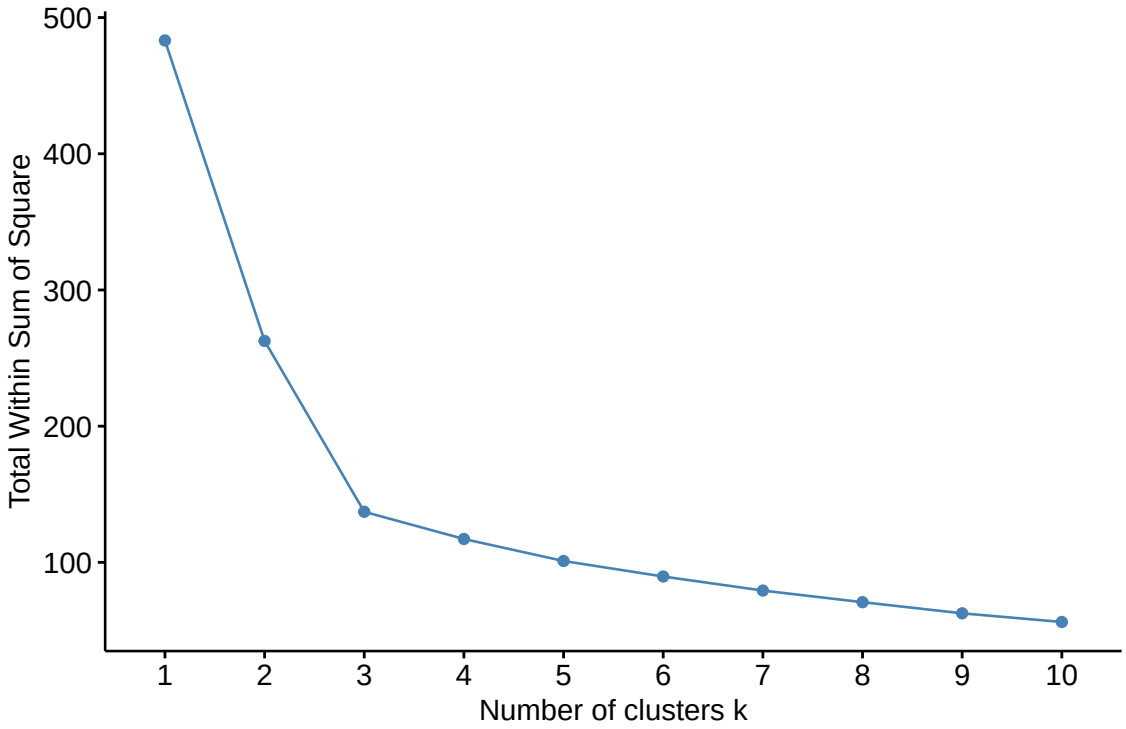
```
##   Number.of.Clusters Between.SS Tot..Within.SS
## 1                  2   220.6239       262.5761
## 2                  3   346.0521       137.1479
## 3                  4   366.0175       117.1825
## 4                  5   378.4925       104.7075
```

Total wss does not change significantly after 3 clusters. So, we take it as appropriate number of clusters for k-Means (by Elbow method).

Determining the Optimal Number of Clusters

We chose the appropriate number for k-Means. kmeans function returns total within cluster sum of squares, so applying Elbow method was easy. But, hclust function does not do so. We can calculate manually but, fviz_nbclust function from factoextra package calculates wss for different number of clusters and plot them.

```
library(factoextra)
fviz_nbclust(pottery_scaled, FUN = hcut, method = "wss")+ggtitle("")
```



From diagram it seems that 3 will be appropriate choice of number of clusters for Hierachical clustring. So, both the method gives the same number.

Visualising the Clusters

Our interest is now a comparison with the kiln sites at which the pottery was found. Doing so also help us to interpret the formed clusters.

```
pottery_cluster_Aver <- cutree(pottery_average, h = 4)
table(pottery_cluster_Aver,pottery$Kiln)
```

```
##
## pottery_cluster_Aver  1  2  3  4  5
##                    1 21  0  0  0  0
##                    2  0 12  2  0  0
##                    3  0  0  0  5  5
```

We are getting 3 clusters and the contingency table shows that cluster 1 contains all pots found at kiln site number one, cluster 2 contains all pots from kiln sites number two and three, and cluster three collects the ten pots from kiln sites four and five. So the clusters actually correspond to pots from three different regions. So, we can say that we get reasonable clusters.

Life Expectancy Data

The eight columns of the data represent life expectancy for men and women from different countries across different age groups. The prefix letter stands for gender and following number represents age. So, w75 represents life expectancy of a 75 year old woman. We will apply Factor Analysis on it.

```
life<-read.table("http://people.stat.sc.edu/Hitchcock/lifeex.txt",header=TRUE)
```

Applying Factor Analysis

First we will observe the correlation matrix of the data. If, some correlations are high then clearly we can represent some of the variables with the help of others.

```
cor(life[,-1])
```

```
           m0       m25       m50       m75        w0       w25       w50       w75
m0  1.0000000 0.7483563 0.6356374 0.2898761 0.9801778 0.8739607 0.6965615 0.3175692
m25 0.7483563 1.0000000 0.6670171 0.3911019 0.6933171 0.7247417 0.6474890 0.3930184
m50 0.6356374 0.6670171 1.0000000 0.7519835 0.5574436 0.7716579 0.8021425 0.5931704
m75 0.2898761 0.3911019 0.7519835 1.0000000 0.2472239 0.5466748 0.6869444 0.7104841
w0  0.9801778 0.6933171 0.5574436 0.2472239 1.0000000 0.8873568 0.7098933 0.3651831
w25 0.8739607 0.7247417 0.7716579 0.5466748 0.8873568 1.0000000 0.9399013 0.6843048
w50 0.6965615 0.6474890 0.8021425 0.6869444 0.7098933 0.9399013 1.0000000 0.8279568
w75 0.3175692 0.3930184 0.5931704 0.7104841 0.3651831 0.6843048 0.8279568 1.0000000
```

Correlations between m0 and w0, w25 and w50, etc. are high, indicating some variables can be well explained with the help of others. So, we expect that, we can reduce number of variables using Factor Analysis.

Now, we don't know what is the right number of factors. To determine it, we first fit model with different number of factors (upto four) and consider the one for which some evalutionary measure is satisfactory. Either we can look for the **communalities** or the **p-value** of the fit. In both the cases higher the better situation.

```
one_factor<-factanal(life[,-1],factors=1)    # Fitting one factor model.
communalities<-1-one_factor$uniquenesses
communalities
```

```
##        m0       m25       m50       m75        w0       w25       w50       w75
## 0.7624864 0.5304617 0.6014815 0.3042447 0.7832480 0.9950000 0.8829882 0.4684498
```

```
unname(one_factor$PVAL)
```

```
## [1] 1.879555e-24
```

Communalities for some variables are high but few are low as well, so one factor can replace some of the variables, but not all. The p-value also indicates that we have to extract more factors.

Now, we increase number of factors gradually.

```
two_factor<-factanal(life[,-1],factors=2)    # Fitting two factor model.
communalities<-1-two_factor$uniquenesses
communalities
```

```
##        m0       m25       m50       m75        w0       w25       w50       w75
## 0.9760108 0.5576653 0.6540319 0.5921643 0.9845974 0.9887526 0.9853247 0.8221271
```

```
unname(two_factor$PVAL)
```

```
## [1] 1.911514e-05
```

Communalities, as well as p-value has increased. But, not only the p-value is insignificant, some communalities are low also. So, we have to increase number of factors.

```
three_factor<-factanal(life[,-1],factors=3)    # Fitting three factor model.
communalities<-1-three_factor$uniquenesses
communalities
```

```
##        m0       m25       m50       m75        w0       w25       w50       w75
## 0.9950000 0.6383261 0.9337228 0.7122064 0.9950000 0.9889330 0.9798799 0.8540204
```

```
unname(three_factor$PVAL)
```

```
## [1] 0.4578204
```

All the communalities except m25 have increased. So, we can add one more factor to make all the communalities significant. But, in Factor Analysis our goal is to represent original variables using as few underlying factors possible. So, though all the communalities are not great, but we stop at three factors, considering the significant p-value.

Interpreting the Results

It may be noted that, we are using **varimax** rotation (default to factanal function) to obtain estimates accordingly, so that a set of original features is highly correlated with only one factor and nearly uncorrelated with others. In this way we get easy to interpret factors. But, what are they in our set up?

```
three_factor$loadings
```

```
##
## Loadings:
##     Factor1 Factor2 Factor3
## m0  0.964   0.122   0.226
## m25 0.646   0.169   0.438
## m50 0.430   0.354   0.790
## m75         0.525   0.656
## w0  0.970   0.217
## w25 0.764   0.556   0.310
## w50 0.536   0.729   0.401
## w75 0.156   0.867   0.280
##
##                Factor1 Factor2 Factor3
## SS loadings      3.375   2.082   1.640
## Proportion Var   0.422   0.260   0.205
## Cumulative Var   0.422   0.682   0.887
```

Life expectancy for different age grops are following this pattern. 1) younger ages (m0,m25,w0,w25) have highest loadings on Factor1, 2) middle aged and old women (w50,w75) have loadings on Factor2 and 3) middle and old aged men (m50,m75) on Factor3. So,it may be he case that, upto some early ages life expectancy pattern remains same irrespective of the gender. After that, it changes differently for men and women.

Keeping this pattern in mind we may interpret the three factors as follows:

- Factor1 is Life expectancy for younger people.
- Factor2 is Life expectancy for adult male.
- Factor3 is Life expectancy for adult female.

Also, from the last line of the output from R, we see that, Factor3 has cummulative variance 0.887; i.e. the three factors are explaining nearly 89% of the variability in the original data.

Way for Further Analysis

We know that 3 factors are enough. Now, we can obtain the data matrix of factor vectors and apply our model of interest on this dimension reduced data.

```
analysis<-factanal(life[,-1],factors=3,scores="regression")
sc<-analysis$scores
```

sc matrix is the required data matrix.