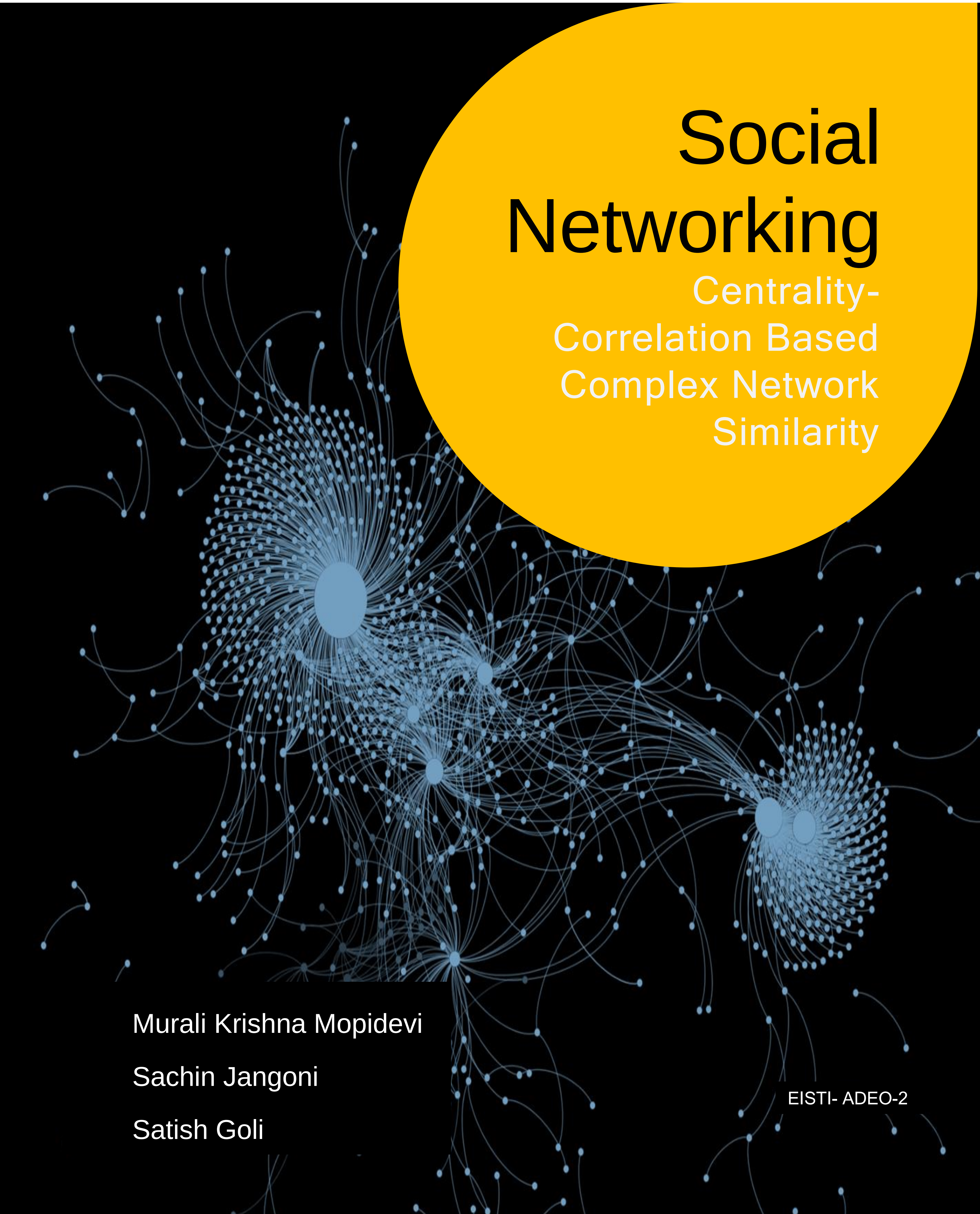
Social Networking

Centrality-Correlation Based Complex Network Similarity



Murali Krishna Mopidevi

Sachin Jangoni

Satish Goli

EISTI- ADEO-2

Table of Contents

Introduction

❖ Complex Networks

Complex network analysis is a collection of quantitative methods for studying the structure and dynamics of complex networked systems. Then, we focus on fundamental network analysis measures and algorithms related to node connectivity, distance, centrality, similarity and clustering.

❖ Centralities

Subgraph Centrality in Complex Networks. We introduce a new centrality measure that characterizes the participation of each node in all subgraphs in a network.This measure is better able to discriminate the nodes of a network than alternate measures such as degree, closeness, betweenness and eigenvector centralities.

❖ Community Detection

Community Detection in Complex Networks by Detecting and Expanding Core Nodes Through Extended Local Similarity of Nodes. First, a community's central node (core node) which has a high level of embeddedness is detected based on the similarity between graph's nodes.

❖ Correlation

A complex network is said to be degree correlated if the degrees of nodes at the end of links occur together in a nonrandom manner. In a social network the nodes represent individuals, and the links between them conceptualize friendships or other social associations.

❖ Similarity

Similarity of nodes is a basic structure quantification in complex networks. Then relative entropy is used to measure the difference between each pair of nodes' structural information. At last the value of relative entropy between each pair of nodes is used to measure nodes' structure similarity in complex networks.

Project-2

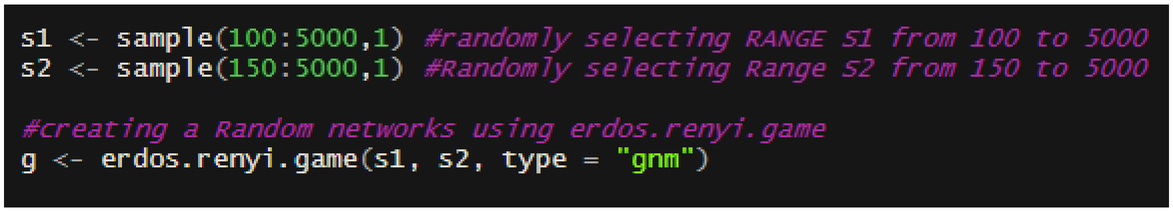
Centrality-Correlation Based Complex Network Similarity

The Proposed Approach is the following

Given a complex network of size n, and a set of k centrality measures C = {C_1,. . .,C_k}, We compute for each centrality C_i the induced ranking vectors of the nodes σ^n_i. For each couple of centrality measures C_i, C_j we can compute the raking correlation factor cor(σ^n_i, σ^n_j). A network N_i can then be represented as a vector in the k*(k−1)/ 2 dimensional space. Similarity between two networks N_i, N_j can then be measured as the inverse of the distance separating both networks is the new representation space. The goal of this project is to implement the proposed network similarity measure and to evaluate this approach by using this similarity measure to cluster networks generated using different network generators.

Creating a Random Networks

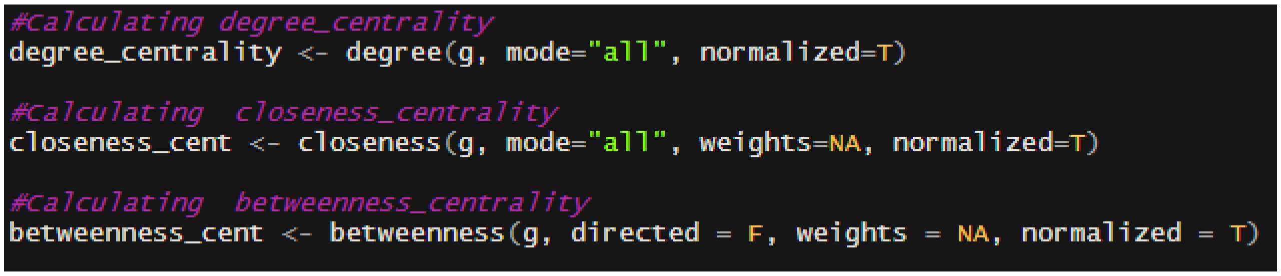
Generating random graph networks by using a function called **erdos.renyi.game** from **igraph package**.



```
s1 <- sample(100:5000,1) #randomly selecting RANGE S1 from 100 to 5000
s2 <- sample(150:5000,1) #Randomly selecting Range S2 from 150 to 5000

#creating a Random networks using erdos.renyi.game
g <- erdos.renyi.game(s1, s2, type = "gnm")
```

Calculating Centralities

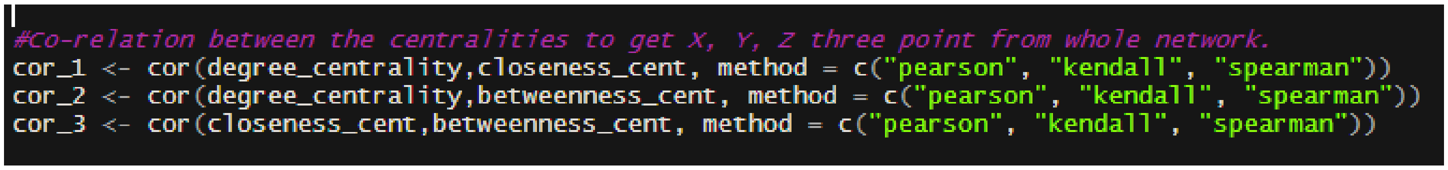
Calculating Centralities (**Degree, Closeness, and Betweenness**) for each randomly generated graph by using **igraph package**.



```
#Calculating degree_centrality
degree_centrality <- degree(g, mode="all", normalized=T)

#Calculating  closeness_centrality
closeness_cent <- closeness(g, mode="all", weights=NA, normalized=T)

#Calculating  betweenness_centrality
betweenness_cent <- betweenness(g, directed = F, weights = NA, normalized = T)
```

Calculating Centrality Correlation

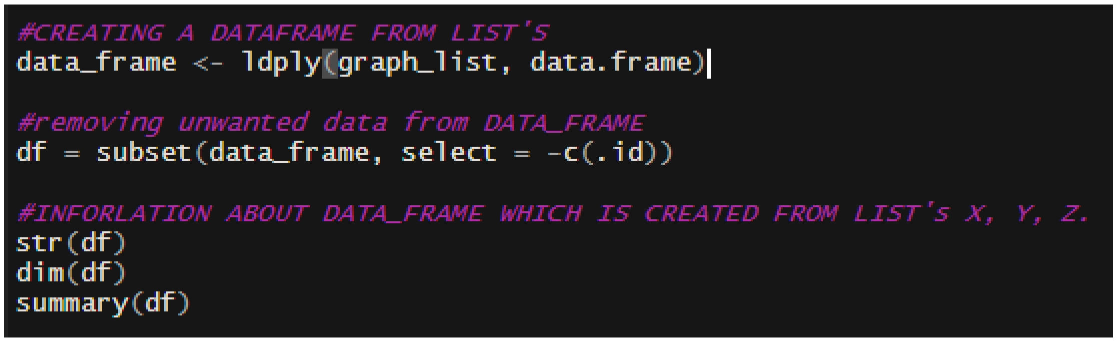
Calculating Centrality Correlation between the (**Degree, closeness, and Betweenness**) centralities using the function cor().



```
#Co-relation between the centralities to get X, Y, Z three point from whole network.
cor_1 <- cor(degree_centrality,closeness_cent, method = c("pearson", "kendall", "spearman"))
cor_2 <- cor(degree_centrality,betweenness_cent, method = c("pearson", "kendall", "spearman"))
cor_3 <- cor(closeness_cent,betweenness_cent, method = c("pearson", "kendall", "spearman"))
```

Creating a DataFrame

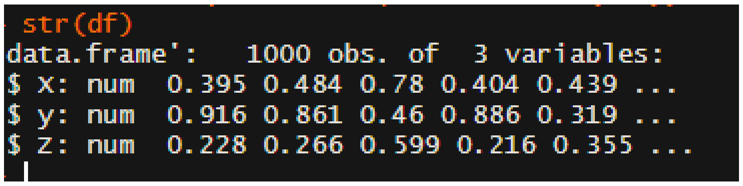
Creating a DataFrame from the Points occurred from **cor_1, cor_2 and cor_3**. And removing unwanted columns.



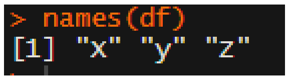
```
#CREATING A DATAFRAME FROM LIST'S
data_frame <- ldply(graph_list, data.frame)

#removing unwanted data from DATA_FRAME
df = subset(data_frame, select = -c(.id))

#INFORLATION ABOUT DATA_FRAME WHICH IS CREATED FROM LIST's X, Y, Z.
str(df)
dim(df)
summary(df)
```

Results of DataFrame(df)

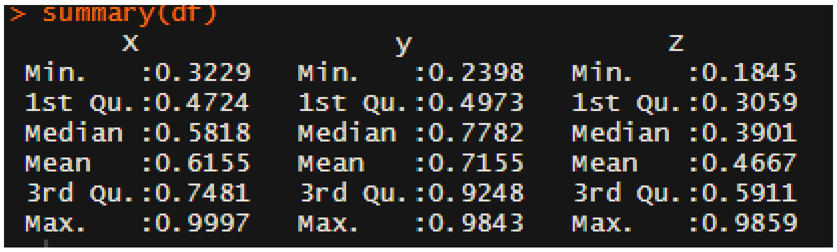
Here some of the results like Structure, Summary, Dimensions and Names in DataFrame.



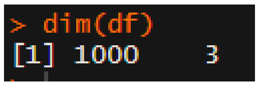
```
 str(df)
data.frame':   1000 obs. of  3 variables:
$ x: num  0.395 0.484 0.78 0.404 0.439 ...
$ y: num  0.916 0.861 0.46 0.886 0.319 ...
$ z: num  0.228 0.266 0.599 0.216 0.355 ...
```



```
> names(df)
[1] "x" "y" "z"
```



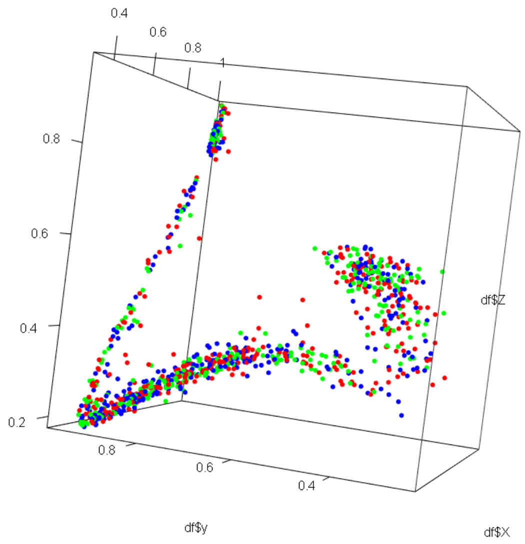
```
> summary(df)
       x                y                z         
 Min.   :0.3229   Min.   :0.2398   Min.   :0.1845  
 1st Qu.:0.4724   1st Qu.:0.4973   1st Qu.:0.3059  
 Median :0.5818   Median :0.7782   Median :0.3901  
 Mean   :0.6155   Mean   :0.7155   Mean   :0.4667  
 3rd Qu.:0.7481   3rd Qu.:0.9248   3rd Qu.:0.5911  
 Max.   :0.9997   Max.   :0.9843   Max.   :0.9859  
```



```
> dim(df)
[1] 1000    3
```

Plotting the DataFrame

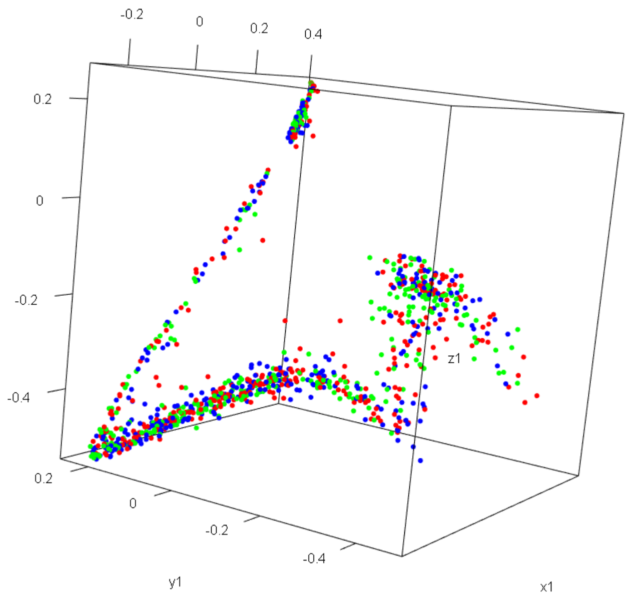
```
#plotting from dataframe in 3D
plot3d(df$x,df$y,df$z, col = c('red','green','blue'), size = 6)
```



Standardization, where the dimensions now have a mean of zero

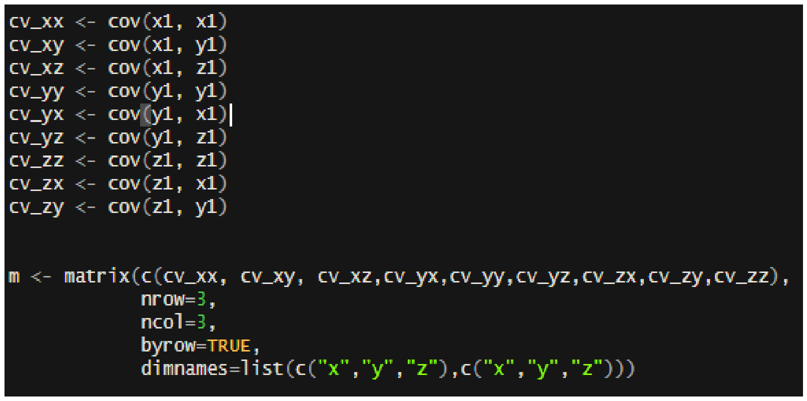
```
#mean of X
m_x <- mean(df$x)

#mean of Y
m_y <- mean(df$y)

#mean of Z
m_z <- mean(df$z)

x1 <- df$x - m_x
x1

#summary of X
summary(x1)

y1 <- df$y - m_y
y1

#summary of Y
summary(y1)

z1 <- df$z - m_y
z1

#summary of z
summary(z1)
```

```
#plotting from dataframe
plot3d(x1,y1,z1, col = c('red','green','blue'), size = 6)
```

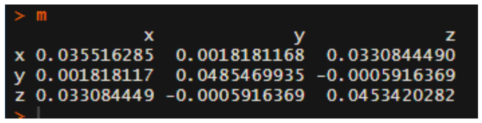


Covariance matrix

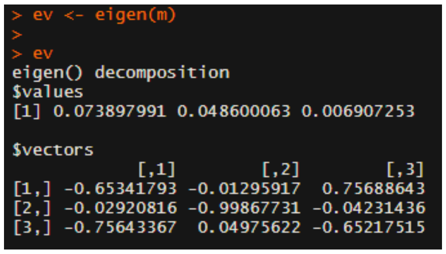
Covariance measures how dimensions vary with respect to each other and the covariance matrix contains all covariance measures between all dimensions and Names in DataFrame and eigenvalues of the covariance matrix. An eigenvector is a direction and an eigenvalue is a number that indicates how much variance is in the data in that direction.



```
cv_xx <- cov(x1, x1)
cv_xy <- cov(x1, y1)
cv_xz <- cov(x1, z1)
cv_yy <- cov(y1, y1)
cv_yx <- cov(y1, x1)
cv_yz <- cov(y1, z1)
cv_zz <- cov(z1, z1)
cv_zx <- cov(z1, x1)
cv_zy <- cov(z1, y1)


m <- matrix(c(cv_xx, cv_xy, cv_xz,cv_yx,cv_yy,cv_yz,cv_zx,cv_zy,cv_zz),
            nrow=3,
            ncol=3,
            byrow=TRUE,
            dimnames=list(c("x","y","z"),c("x","y","z")))
```



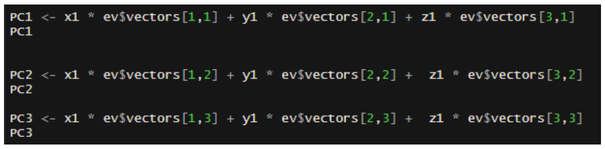
```
> m
            x            y             z
x 0.035516285  0.0018181168  0.0330844490
y 0.001818117  0.0485469935 -0.0005916369
z 0.033084449 -0.0005916369  0.0453420282
>
```



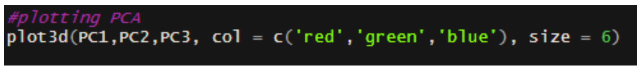
```
> ev <- eigen(m)
>
> ev
eigen() decomposition
$values
[1] 0.073897991 0.048600063 0.006907253

$vectors
            [,1]        [,2]        [,3]
[1,] -0.65341793 -0.01295917  0.75688643
[2,] -0.02920816 -0.99867731 -0.04231436
[3,] -0.75643367  0.04975622 -0.65217515
```

Principle Component Analysis

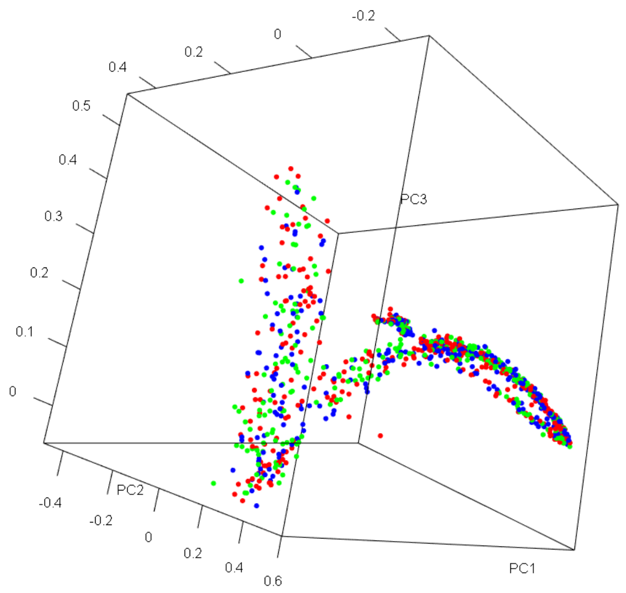
The largest eigenvalue is the first principal component. We multiply the standardized values to the first eigenvector, which is stored in ev$vectors[,1].

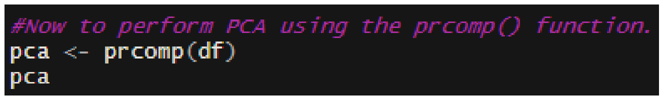


```
PC1 <- x1 * ev$vectors[1,1] + y1 * ev$vectors[2,1] + z1 * ev$vectors[3,1]
PC1


PC2 <- x1 * ev$vectors[1,2] + y1 * ev$vectors[2,2] +  z1 * ev$vectors[3,2]
PC2

PC3 <- x1 * ev$vectors[1,3] + y1 * ev$vectors[2,3] +  z1 * ev$vectors[3,3]
PC3
```

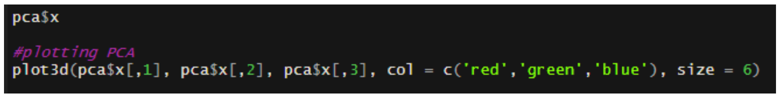


```
#plotting PCA
plot3d(PC1,PC2,PC3, col = c('red','green','blue'), size = 6)
```

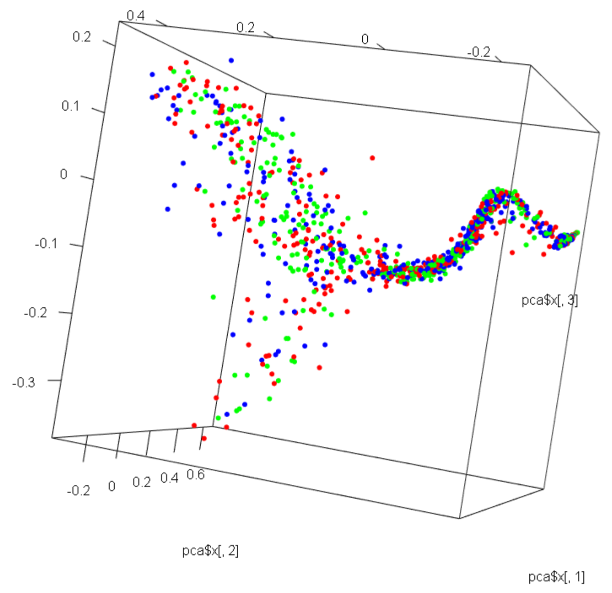


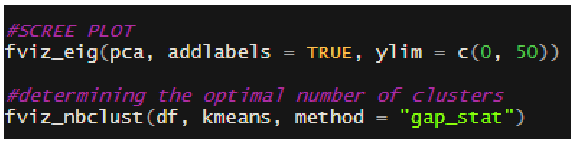


```
#Now to perform PCA using the prcomp() function.
pca <- prcomp(df)
pca
```

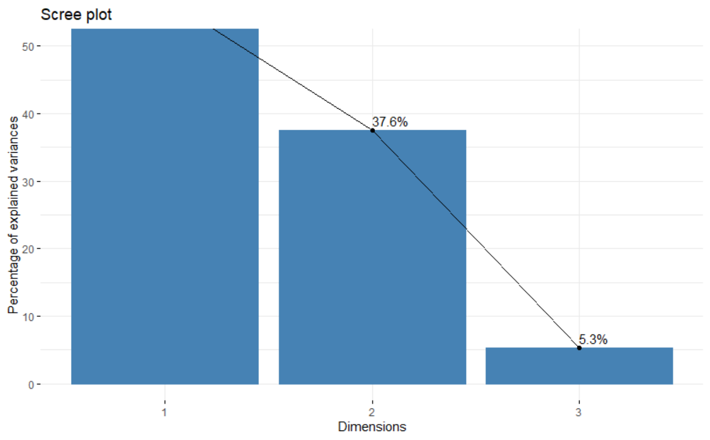


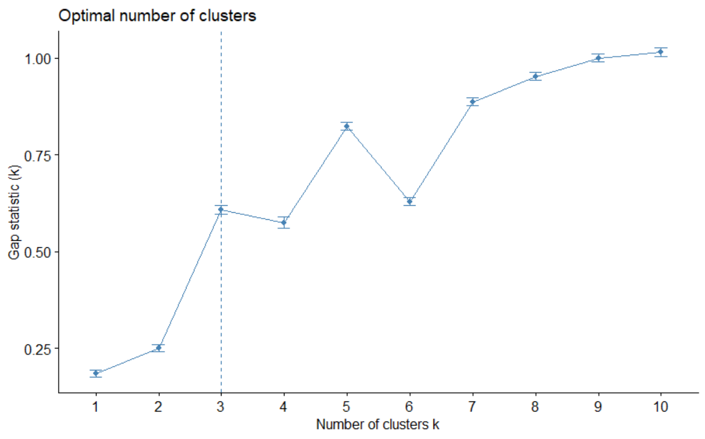
```
pca$x

#plotting PCA
plot3d(pca$x[,1], pca$x[,2], pca$x[,3], col = c('red','green','blue'), size = 6)
```





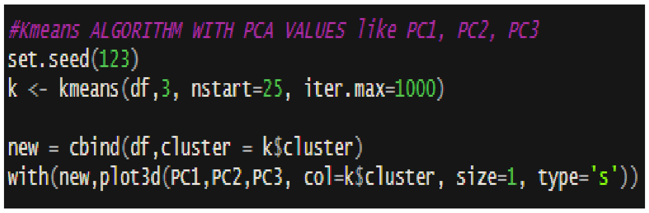
```
#SCREE PLOT
fviz_eig(pca, addlabels = TRUE, ylim = c(0, 50))

#determining the optimal number of clusters
fviz_nbclust(df, kmeans, method = "gap_stat")
```



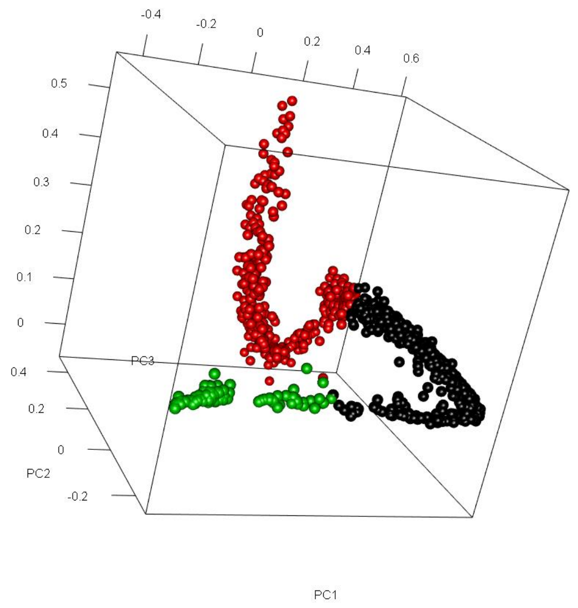


Clustering using K-Means Algorithm

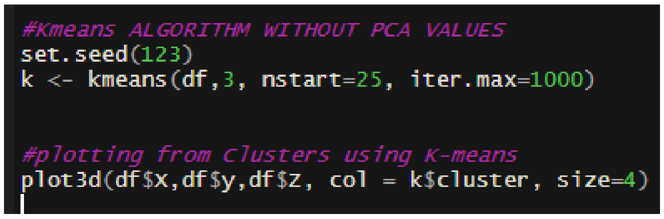
K-means with PCA values



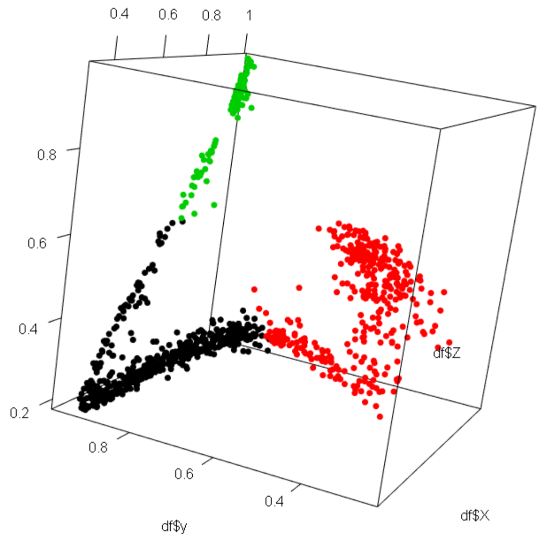
```
#Kmeans ALGORITHM WITH PCA VALUES like PC1, PC2, PC3
set.seed(123)
k <- kmeans(df,3, nstart=25, iter.max=1000)

new = cbind(df,cluster = k$cluster)
with(new,plot3d(PC1,PC2,PC3, col=k$cluster, size=1, type='s'))
```

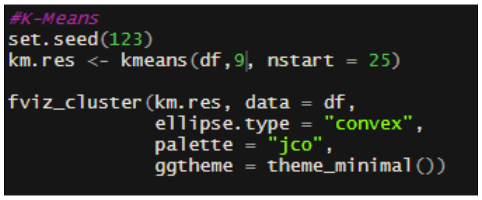


K-means without PCA values

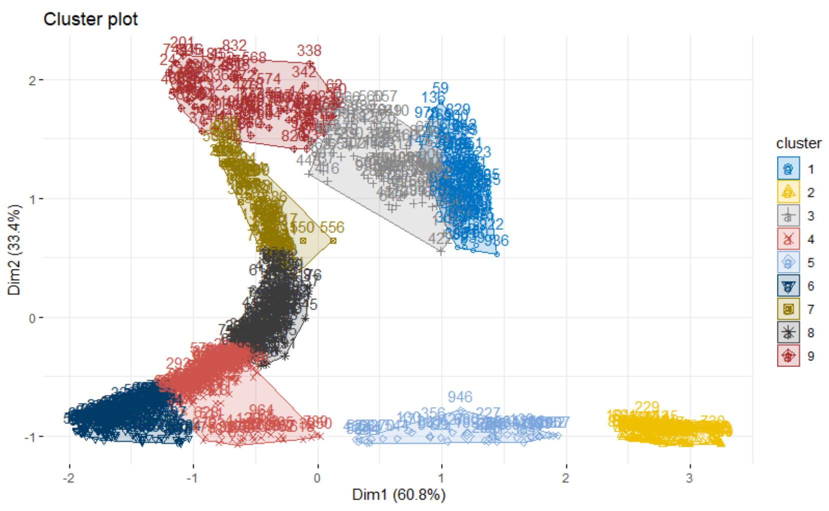


```
#Kmeans ALGORITHM WITHOUT PCA VALUES
set.seed(123)
k <- kmeans(df,3, nstart=25, iter.max=1000)

#plotting from clusters using K-means
plot3d(df$x,df$y,df$z, col = k$cluster, size=4)
```

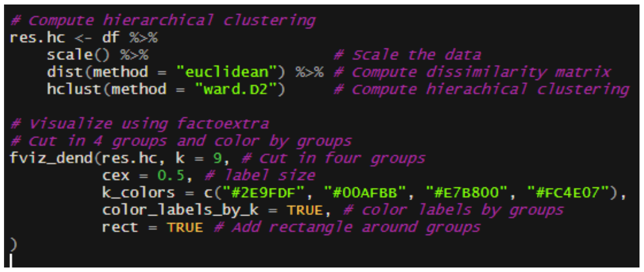




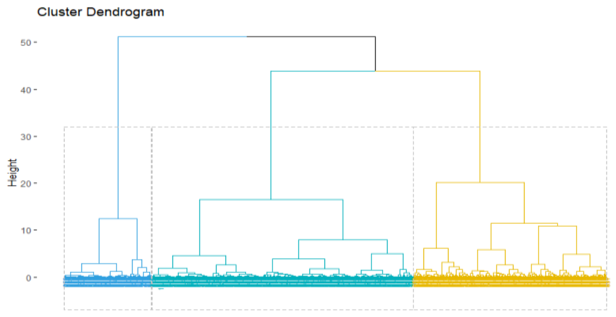
```
#K-Means
set.seed(123)
km.res <- kmeans(df,9, nstart = 25)

fviz_cluster(km.res, data = df,
             ellipse.type = "convex",
             palette = "jco",
             ggtheme = theme_minimal())
```

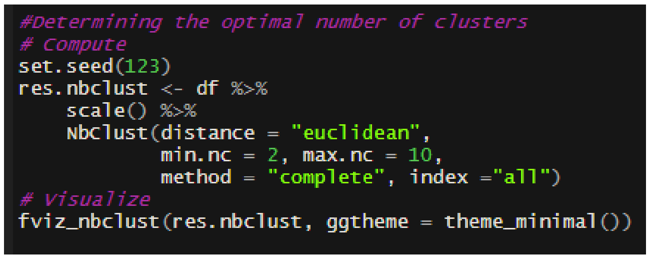


Clustering using Hierarchical Algorithm

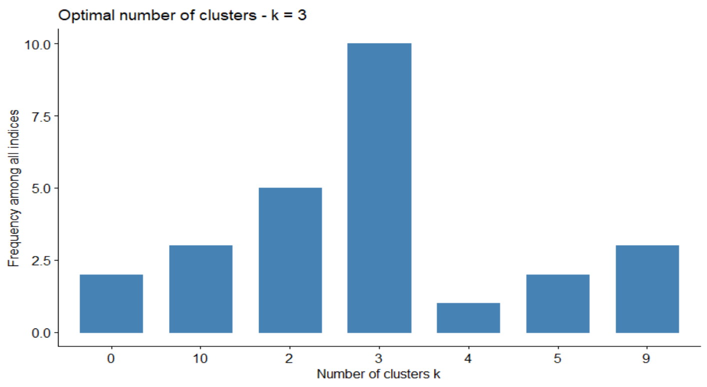


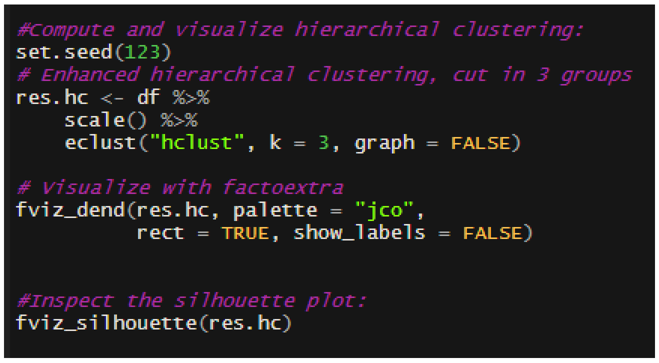
```
# Compute hierarchical clustering
res.hc <- df %>%
    scale() %>%                          # Scale the data
    dist(method = "euclidean") %>% # Compute dissimilarity matrix
    hclust(method = "ward.D2")          # Compute hierachical clustering

# Visualize using factoextra
# Cut in 4 groups and color by groups
fviz_dend(res.hc, k = 9, # Cut in four groups
          cex = 0.5, # label size
          k_colors = c("#2E9FDF", "#00AFBB", "#E7B800", "#FC4E07"),
          color_labels_by_k = TRUE, # color labels by groups
          rect = TRUE # Add rectangle around groups
)
```



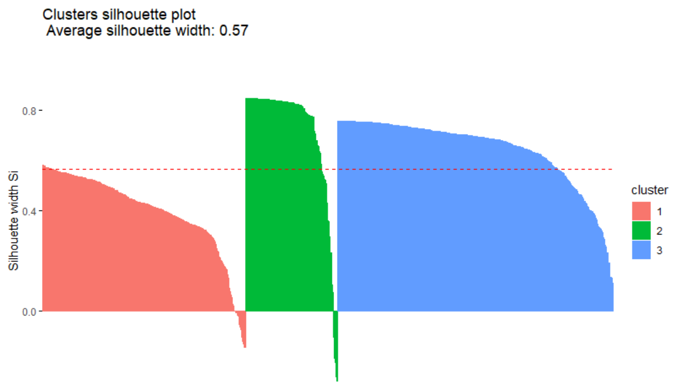


```
#Determining the optimal number of clusters
# Compute
set.seed(123)
res.nbclust <- df %>%
    scale() %>%
    NbClust(distance = "euclidean",
            min.nc = 2, max.nc = 10,
            method = "complete", index ="all")
# Visualize
fviz_nbclust(res.nbclust, ggtheme = theme_minimal())
```





```
#Compute and visualize hierarchical clustering:
set.seed(123)
# Enhanced hierarchical clustering, cut in 3 groups
res.hc <- df %>%
    scale() %>%
    eclust("hclust", k = 3, graph = FALSE)

# Visualize with factoextra
fviz_dend(res.hc, palette = "jco",
          rect = TRUE, show_labels = FALSE)


#Inspect the silhouette plot:
fviz_silhouette(res.hc)
```


Clusters silhouette plot
Average silhouette width: 0.57
Silhouette width Si
cluster