multivariate2 project

Mehrab Atighi

12/23/2021

Contents

PCA Method On Data

Now we are importing our data from Excel and csv format to R.

```
rm(list=ls())
Data<-read.csv("F:/lessons/Multi countios Variate2/project/edite
#View(Data)
new.data.y<-data.frame(y1=Data$How.much.time.did.you.spend.exerc
                       y2=Data$How.much.time.did.you.spend.last.
                       y3=Data$How.much.time.did.you.spend.on.ar
                       y4=Data$How.much.time.did.you.spend.on.au
                       y5=Data$How.much.time.did.you.spend.on.vi
                       y6=Data$How.much.time.did.you.spend.study

#View(new.data.y)
head(new.data.y,5)
```

```
##    y1 y2 y3 y4 y5 y6
## 1   2 10  0  7 10  2
## 2   5  0  6  6  1  4
## 3   3  5  2 10 20  0
## 4   2  7  0  6 20  0
## 5 13  0  5 15 20 10
```

PCA Method On Data

Now We want to see the dimantiom of our data and get Correlation and Variance Covarince matrix of our variables.

```
dim(new.data.y)
```

```
## [1] 95  6
```

```
cor(new.data.y)
```

```
##           y1           y2          y3          y4           y5           y6
## y1 1.0000000  0.229320331  0.22077799  0.20190028  0.139483494  0.468624239
## y2 0.2293203  1.000000000  0.03552107 -0.13350996 -0.004623956 -0.003322887
## y3 0.2207780  0.035521065  1.00000000 -0.09152112  0.138033595  0.372941913
## y4 0.2019003 -0.133509959 -0.09152112  1.00000000  0.373966882  0.184092013
## y5 0.1394835 -0.004623956  0.13803359  0.37396688  1.000000000  0.122167726
## y6 0.4686242 -0.003322887  0.37294191  0.18409201  0.122167726  1.000000000
```

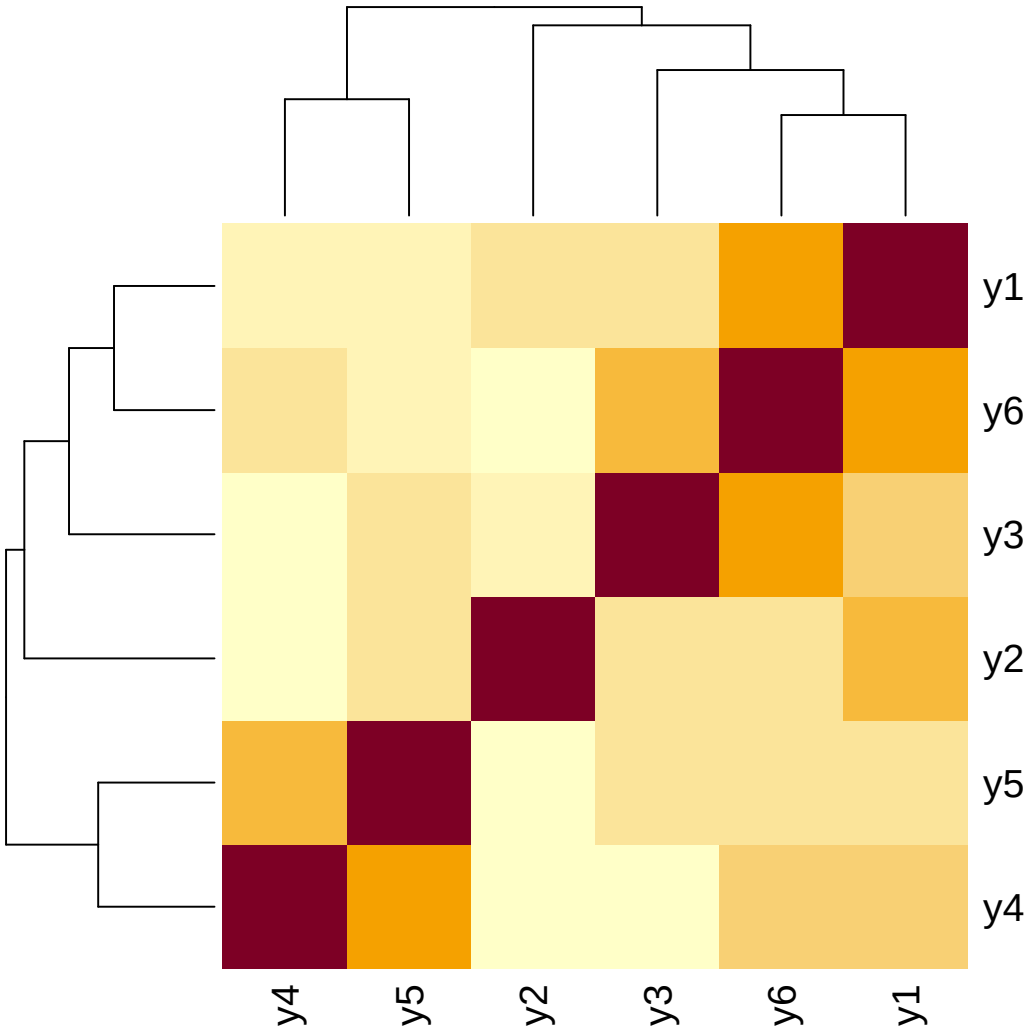
```
cov(new.data.y)
```

```
##           y1          y2          y3        y4          y5          y6
## y1 29.113868  6.31297695  5.8580041  7.913185  5.3514978 12.32806831
## y2  6.312977 26.03055201  0.8911927 -4.947887 -0.1677482 -0.08265662
## y3  5.858004  0.89119267 24.1817582 -3.269111  4.8264889  8.94139869
## y4  7.913185 -4.94788712 -3.2691111 52.762925 19.3152419  6.51958131
## y5  5.351498 -0.16774823  4.8264889 19.315242 50.5597773  4.23525336
## y6 12.328068 -0.08265662  8.9413987  6.519581  4.2352534 23.77062803
```

PCA Method On Data

Now we want to see the Correlation between variables in heatmap

```
heatmap(cor(new.data.y))
```



PCA Method On Data

Now its time to see the eigen values of correlation matrix. and use principal components method on our dataset with two matrix(Correlation and Variance Covarianve matrix.)

```
eigen(cor(new.data.y))
```

```
## eigen() decomposition
## $values
## [1] 1.8996804 1.3135788 1.0222821 0.8376152 0.4786501 0.44819
##
## $vectors
##             [,1]       [,2]       [,3]       [,4]        [,5]
## [1,] -0.5407557  0.1940342  0.2546516 -0.3614315 -0.46856970
## [2,] -0.1005109  0.4553460  0.7620038  0.2320411  0.33420840
## [3,] -0.3891328  0.3495745 -0.4731513  0.4910754  0.35464057
## [4,] -0.3288327 -0.6310244  0.1853864 -0.2225726  0.60762439
## [5,] -0.3555008 -0.4601811  0.1605149  0.6511339 -0.41360272
## [6,] -0.5581620  0.1511621 -0.2655136 -0.3175772  0.05198702
```

```
pc.r<-princomp(new.data.y , cor = TRUE , scores = TRUE)
pc.c<-princomp(new.data.y , cor = FALSE , scores = TRUE)
```

PCA Method On Data

To see the Standard deviation and proprtion of Varince for each new components we use the summary function.

```
summary(pc.r)
```

```
## Importance of components:
##                             Comp.1    Comp.2    Comp.3    Comp.
## Standard deviation     1.3782889 1.1461146 1.0110797 0.915213
## Proportion of Variance 0.3166134 0.2189298 0.1703804 0.139602
## Cumulative Proportion  0.3166134 0.5355432 0.7059235 0.845526
##                             Comp.6
## Standard deviation     0.6694725
## Proportion of Variance 0.0746989
## Cumulative Proportion  1.0000000
```

According to above outputs we should select 4 components until good cumulative proportion of variance. we can see that the first Components just have 31% of variance and the second 21% and 4 components have 84% cumulative proprtion of variance.

Pca Method On Data

Now we want to see the Cofficient of each y in each components:

```
pc.r$loadings
```

```
##
## Loadings:
##    Comp.1 Comp.2 Comp.3 Comp.4 Comp.5 Comp.6
## y1  0.541  0.194  0.255  0.361  0.469  0.505
## y2  0.101  0.455  0.762 -0.232 -0.334 -0.191
## y3  0.389  0.350 -0.473 -0.491 -0.355  0.368
## y4  0.329 -0.631  0.185  0.223 -0.608  0.201
## y5  0.356 -0.460  0.161 -0.651  0.414 -0.203
## y6  0.558  0.151 -0.266  0.318        -0.701
##
##                Comp.1 Comp.2 Comp.3 Comp.4 Comp.5 Comp.6
## SS loadings     1.000  1.000  1.000  1.000  1.000  1.000
## Proportion Var  0.167  0.167  0.167  0.167  0.167  0.167
## Cumulative Var  0.167  0.333  0.500  0.667  0.833  1.000
```

Pca Method On Data

Now we want to see the values of each observation in new dimantions, so we have values for each Componetes(all of them). Attention that here we will see just 10 observation.

```
head(pc.r$scores, 10)
```

```
##              Comp.1      Comp.2      Comp.3     Comp.4      Comp.
##  [1,] -0.8312907  0.78800045  1.12424599 -0.2291562 -0.391168
##  [2,] -0.5174568  1.16325204 -1.15464474  0.7564151 -0.369371
##  [3,] -0.2614935 -0.45647905  0.64091222 -1.0931362  0.234310
##  [4,] -0.6637798 -0.10678415  0.98458600 -1.1743495  0.496684
##  [5,]  2.2640996 -0.45415577 -0.34473645  0.3165415  0.691420
##  [6,] -0.4421232 -0.23444266  0.79595248 -1.2860104  1.146610
##  [7,] -0.3442112  0.05856331  0.06441728 -0.7962395 -0.268195
##  [8,] -1.9204466  0.78481683 -0.70643484  0.5211554  0.242360
##  [9,]  0.4892685 -1.97192890  0.59807666 -0.3625204  0.392263
## [10,]  2.0631809 -1.16656444  0.80883491  1.3869356  0.867239
```

Pca Method On Data

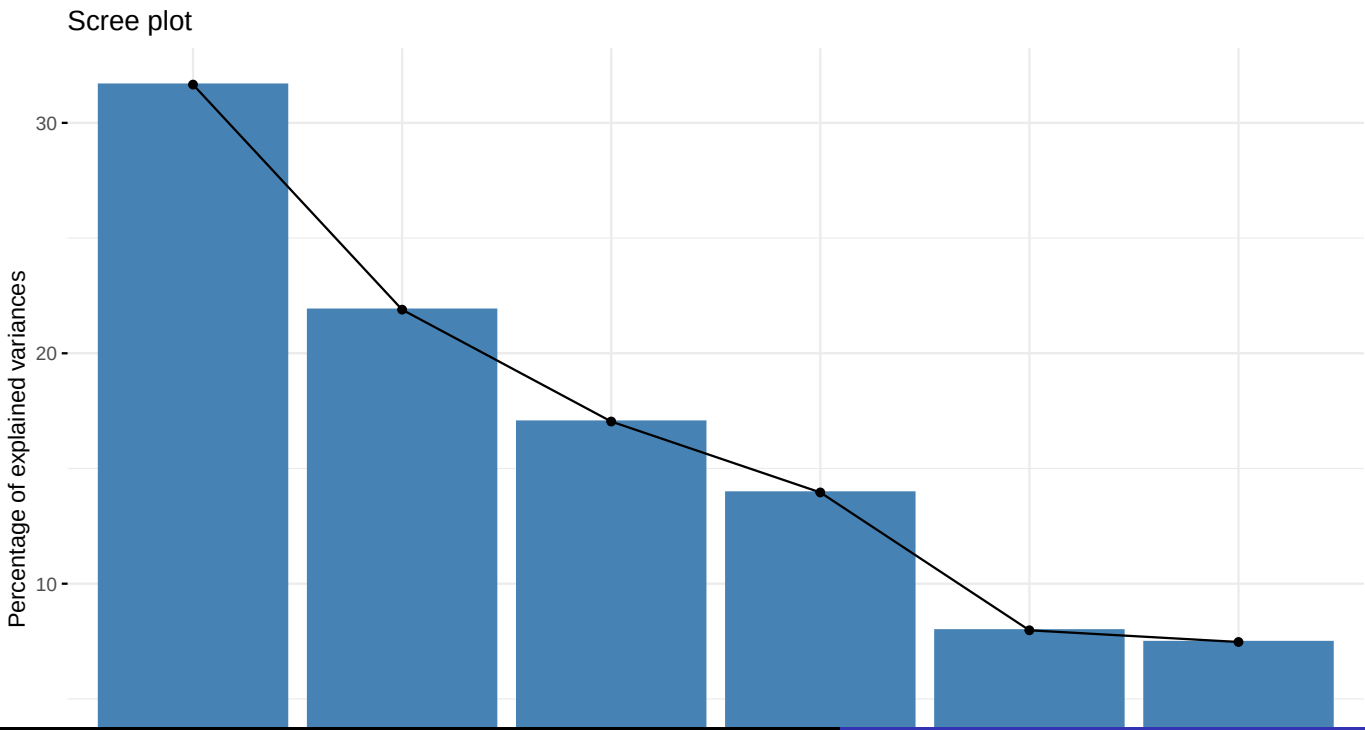
Now we want to visualazing Pca components.

```
library(factoextra)
```

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

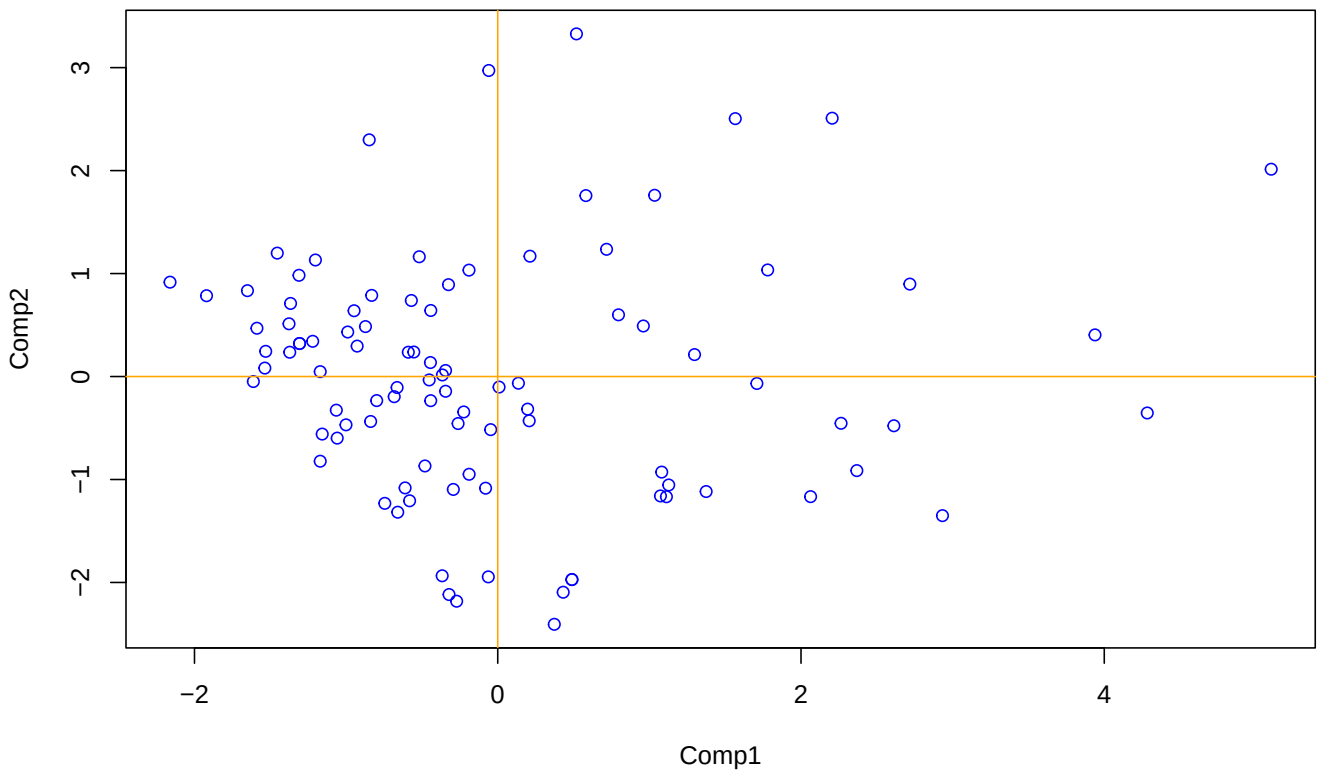
```
## Welcome! Want to learn more? See two factoextra-related
```

```
fviz_eig(pc.r)
```



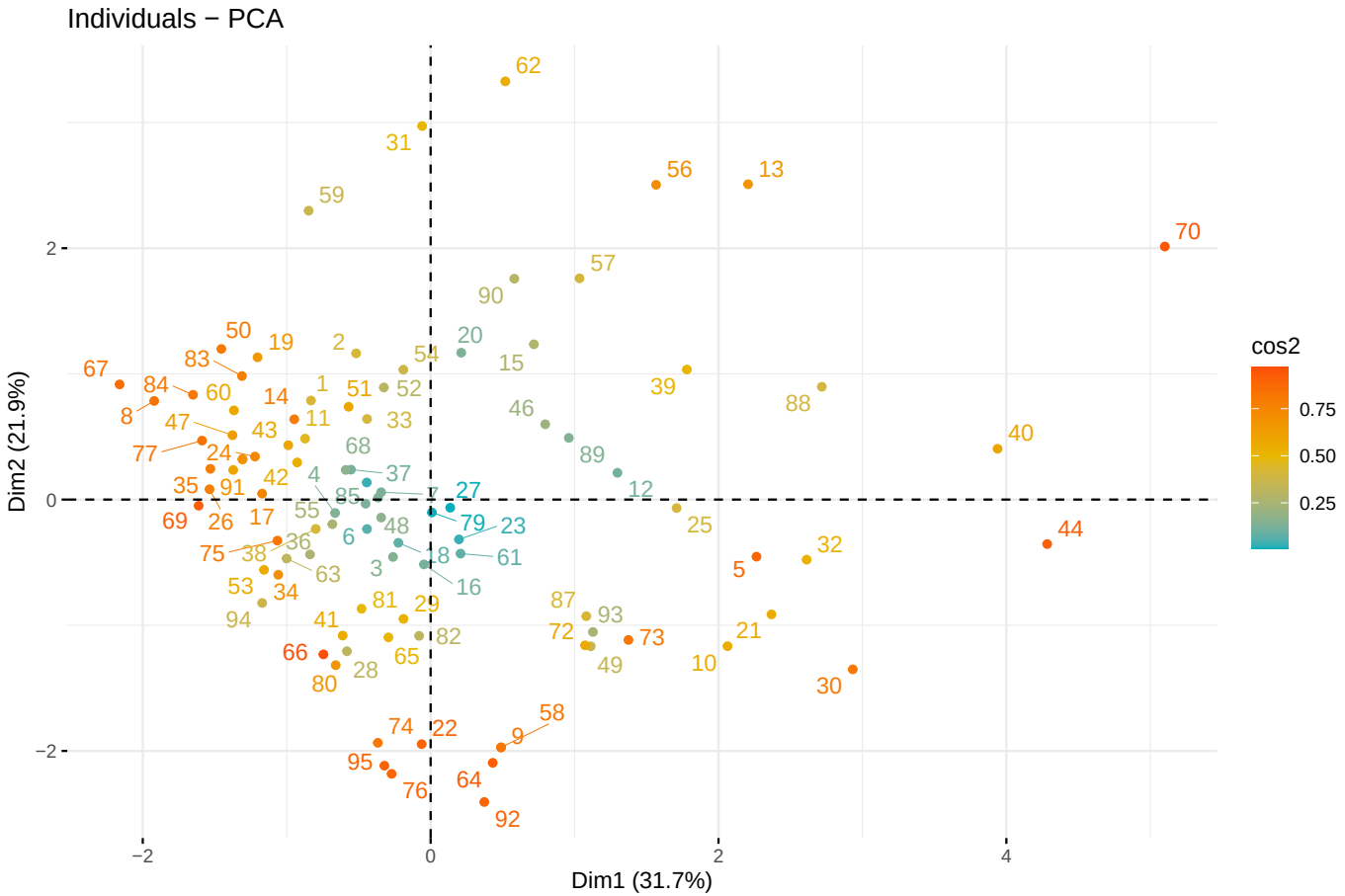
Pca Method On Data

```
plot(pc.r$scores[,1],pc.r$scores[,2]
     ,xlab = "Comp1" , ylab="Comp2" ,col="Blue")
abline(h=0 , col="orange")
abline(v=0 , col="orange")
```



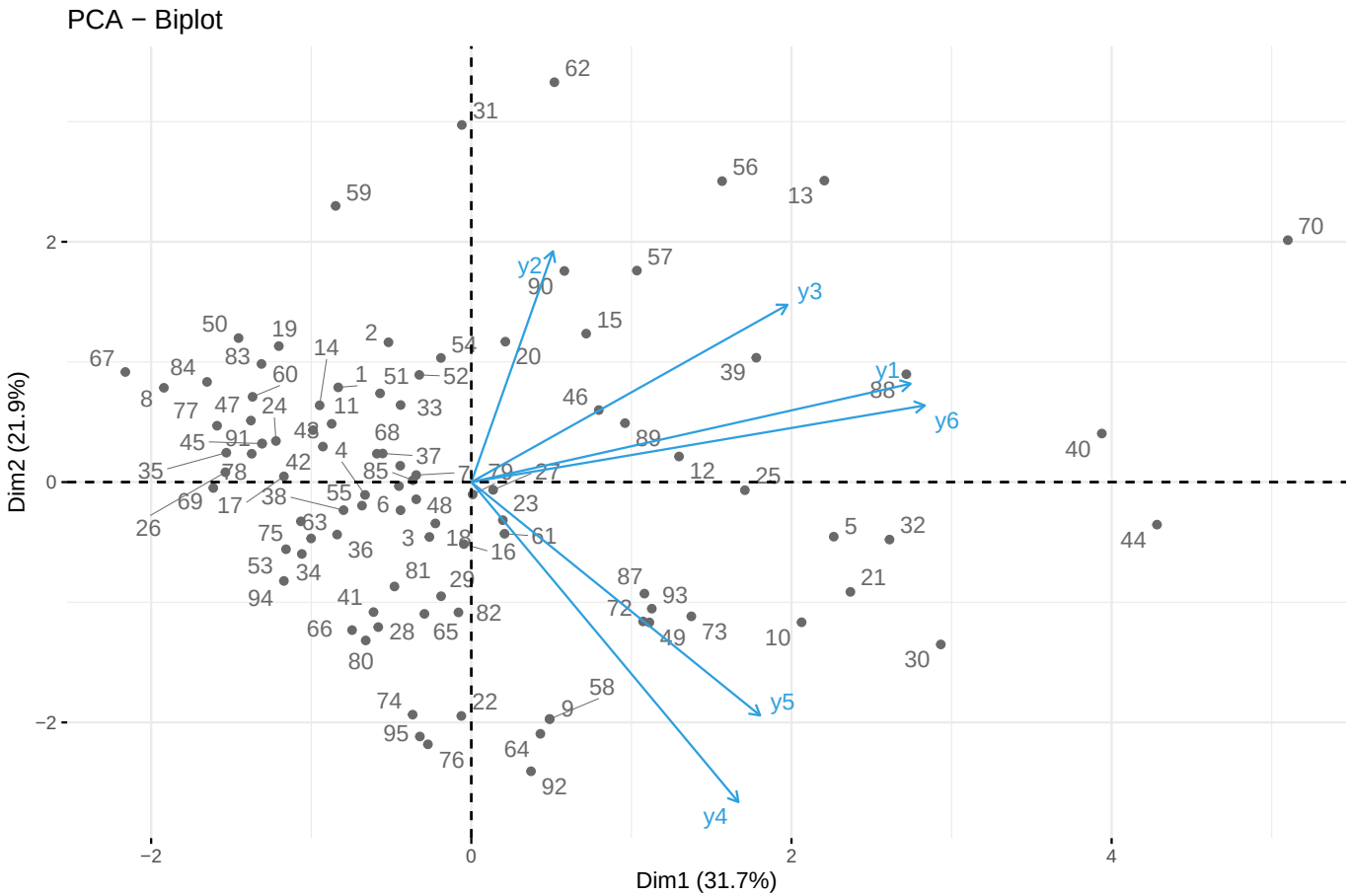
Pca Method On Data

```
fviz_pca_ind(pc.r,
             col.ind = "cos2", # Color by the quality of r
             gradient.cols = c("#00AFBB", "#E7B800", "#FC4
             repel = TRUE) #Avoid text overlapping
```



Pca Method On Data

```
fviz_pca_biplot(pc.r, repel = TRUE,
                col.var = "#2E9FDF", # Variables color
                col.ind = "#696969" # Individuals color
                )
```



Factor Analysis on Data

Now we want to do Factor Analysis on our Data.
at the first we start with factor analysis method with correlation matrix that calculated with pearson method.

```
#with Correlation Matrix
fa1<-factanal(new.data.y  , 3 ,scores = "regression" ,
              rotation = "none", cor="pearson")
fa2<-factanal(new.data.y  , 3 ,scores = "Bartlett",
              cor="pearson")
fa3<-factanal(new.data.y  , 3 ,scores = "regression" ,
              rotation = "varimax", cor="pearson")
```

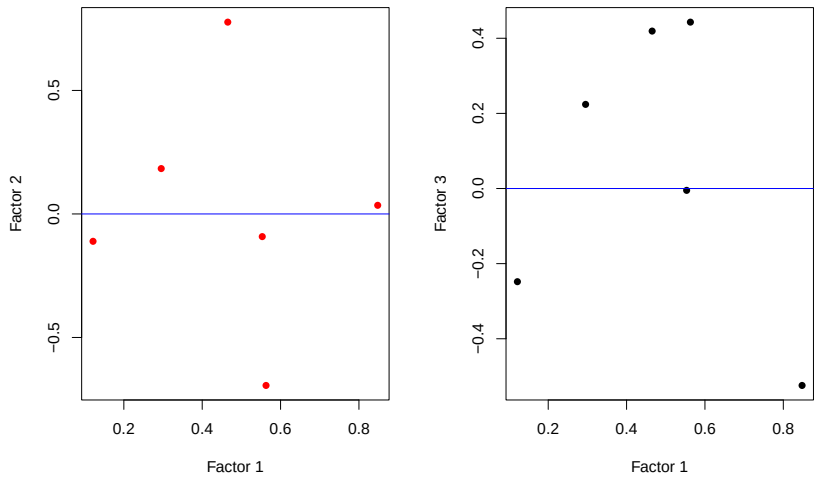
Factor Analysis on Data

in last slide we made 3 factor analysis with diffrent rotation type **none, variamx** and type of scores **regression , Bartlett**.
in next sildes we want to ploting these factor analysis points with new axes.

Factor Analysis on Data

ploting for fa1:

```
#windows(10,10)
par(mfrow=c(1,2))
plot(loadings(fa1)[,1],loadings(fa1)[,2],pch=16,xlab="Factor 1",
     ylab="Factor 2",col="red")
abline(h=0 , col="blue")
abline(v=0 , col="blue")
plot(loadings(fa1)[,1],loadings(fa1)[,3],pch=16,xlab="Factor 1",
     ylab="Factor 3",col="black")
abline(h=0 , col="blue")
abline(v=0 , col="blue")
```

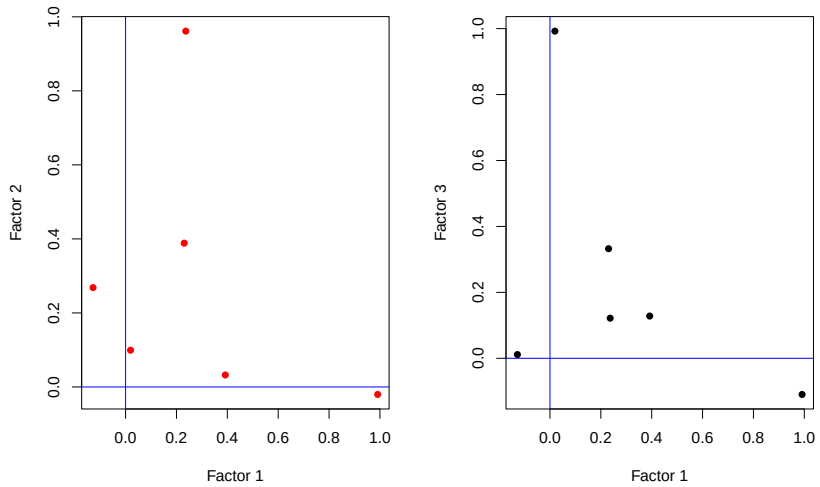


Factor Analysis on Data ploting

Factor Analysis on Data

ploting for fa3:

```
#windows(10,10)
par(mfrow=c(1,2))
plot(loadings(fa3)[,1],loadings(fa3)[,2],pch=16,xlab="Factor 1",
     ylab="Factor 2",col="red")
abline(h=0 , col="blue")
abline(v=0 , col="blue")
plot(loadings(fa3)[,1],loadings(fa3)[,3],pch=16,xlab="Factor 1",
     ylab="Factor 3",col="black")
abline(h=0 , col="blue")
abline(v=0 , col="blue")
```



Factor Analysis on Data

```
fa1
```

```
##
## Call:
## factanal(x = new.data.y, factors = 3, scores = "regression",     rotation = "none", cor = "pearson")
##
## Uniquenesses:
##    y1    y2    y3    y4    y5    y6
## 0.005 0.911 0.005 0.005 0.829 0.685
##
## Loadings:
##    Factor1 Factor2 Factor3
## y1  0.848           -0.524
## y2  0.121  -0.111   -0.248
## y3  0.563  -0.694    0.443
## y4  0.465   0.776    0.419
## y5  0.295   0.184    0.224
## y6  0.553
##
##                Factor1 Factor2 Factor3
## SS loadings      1.660   1.140   0.758
## Proportion Var   0.277   0.190   0.126
## Cumulative Var   0.277   0.467   0.593
##
## The degrees of freedom for the model is 0 and the fit was 0.0132
```

Factor Analysis on Data

```
fa2
```

```
##
## Call:
## factanal(x = new.data.y, factors = 3, scores = "Bartlett", cor = "pearson")
##
## Uniquenesses:
##    y1    y2    y3    y4    y5    y6
## 0.005 0.911 0.005 0.005 0.829 0.685
##
## Loadings:
##    Factor1 Factor2 Factor3
## y1  0.237   0.961   0.122
## y2 -0.128   0.268
## y3                  0.992
## y4  0.991          -0.110
## y5  0.392           0.128
## y6  0.230   0.388   0.332
##
##                Factor1 Factor2 Factor3
## SS loadings      1.262   1.158   1.139
## Proportion Var   0.210   0.193   0.190
## Cumulative Var   0.210   0.403   0.593
##
## The degrees of freedom for the model is 0 and the fit was 0.0132
```

Factor Analysis on Data

```
fa3
```

```
##
## Call:
## factanal(x = new.data.y, factors = 3, scores = "regression",     rotation = "varimax", cor = "pearson")
##
## Uniquenesses:
##    y1    y2    y3    y4    y5    y6
## 0.005 0.911 0.005 0.005 0.829 0.685
##
## Loadings:
##    Factor1 Factor2 Factor3
## y1  0.237   0.961   0.122
## y2 -0.128   0.268
## y3                  0.992
## y4  0.991          -0.110
## y5  0.392           0.128
## y6  0.230   0.388   0.332
##
##                Factor1 Factor2 Factor3
## SS loadings      1.262   1.158   1.139
## Proportion Var   0.210   0.193   0.190
## Cumulative Var   0.210   0.403   0.593
##
## The degrees of freedom for the model is 0 and the fit was 0.0132
```

Factor Analysis on Data

We can you and have Factor Analysis method without Correlation matrix and using covariance matrix in this algorithm.

```
#install.packages("psych")
library(psych)
```

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

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

```
library(ggplot2)
```

Factor Analysis on Data

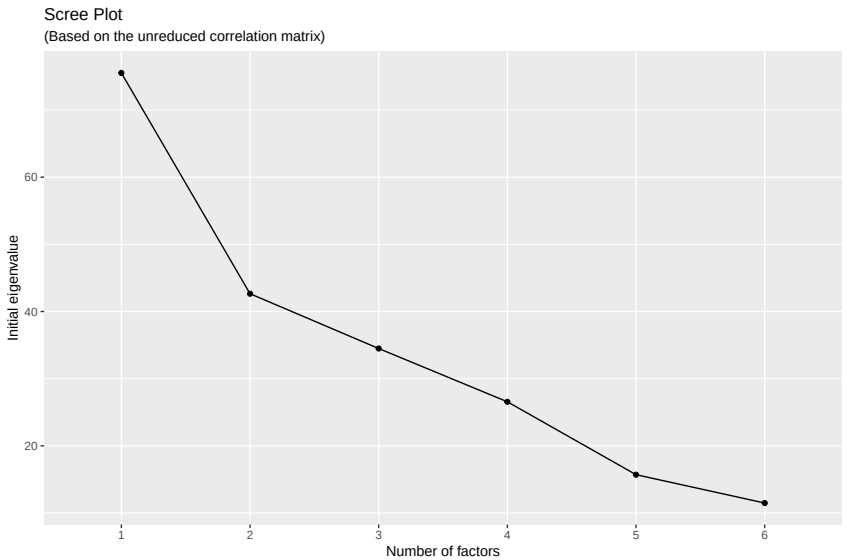
using factor analysis with covariance matrix and varimax method:

```
fa4 <- fa(new.data.y,nfactors = 6, rotate = "varimax" ,
          scores = "regression" ,covar = TRUE )
fa4
```

```
## Factor Analysis using method =  minres
## Call: fa(r = new.data.y, nfactors = 6, rotate = "varimax", scores = "regression",
##     covar = TRUE)
## Unstandardized loadings (pattern matrix) based upon covariance matrix
##     MR3   MR1  MR2   MR5   MR4 MR6 h2 u2   H2   U2
## y1 0.38  0.64 3.82  0.63  1.28   0 17 12 0.59 0.41
## y2 0.01 -0.44 0.52 -0.02  3.62   0 14 12 0.52 0.48
## y3 0.46 -0.51 1.02  3.23  0.06   0 12 12 0.49 0.51
## y4 1.80  5.81 1.33 -0.76 -0.87   0 40 13 0.76 0.24
## y5 5.91  1.50 0.42  0.75  0.06   0 38 13 0.75 0.25
## y6 0.11  0.62 2.91  1.93 -0.35   0 13 11 0.54 0.46
##
##                        MR3   MR1   MR2   MR5   MR4  MR6
## SS loadings          38.54 37.30 26.34 15.71 15.62 0.00
## Proportion Var        0.19  0.18  0.13  0.08  0.08 0.00
## Cumulative Var        0.19  0.37  0.50  0.57  0.65 0.65
## Proportion Explained  0.29  0.28  0.20  0.12  0.12 0.00
## Cumulative Proportion 0.29  0.57  0.77  0.88  1.00 1.00
##
##  Standardized loadings (pattern matrix)
##    item  MR3   MR1  MR2   MR5   MR4 MR6   h2   u2
## y1    1 0.07  0.12 0.71  0.12  0.24   0 0.59 0.41
## y2    2 0.00 -0.09 0.10  0.00  0.71   0 0.52 0.48
## y3    3 0.09 -0.10 0.21  0.66  0.01   0 0.49 0.51
## y4    4 0.25  0.80 0.18 -0.10 -0.12   0 0.76 0.24
## y5    5 0.83  0.21 0.06  0.11  0.01   0 0.75 0.25
## y6    6 0.02  0.13 0.60  0.40 -0.07   0 0.54 0.46
##
##                  MR3  MR1  MR2  MR5  MR4  MR6
```

Factor Analysis on Data

```
n_factors <- length(fa4$e.values)
scree     <- data.frame(
  Factor_n =  as.factor(1:n_factors),
  Eigenvalue = fa4$e.values)
ggplot(scree, aes(x = Factor_n, y = Eigenvalue, group = 1)) +
  geom_point() + geom_line() +
  xlab("Number of factors") +
  ylab("Initial eigenvalue") +
  labs( title = "Scree Plot",
        subtitle = "(Based on the unreduced correlation matrix)")
```



Factor Analysis on Data

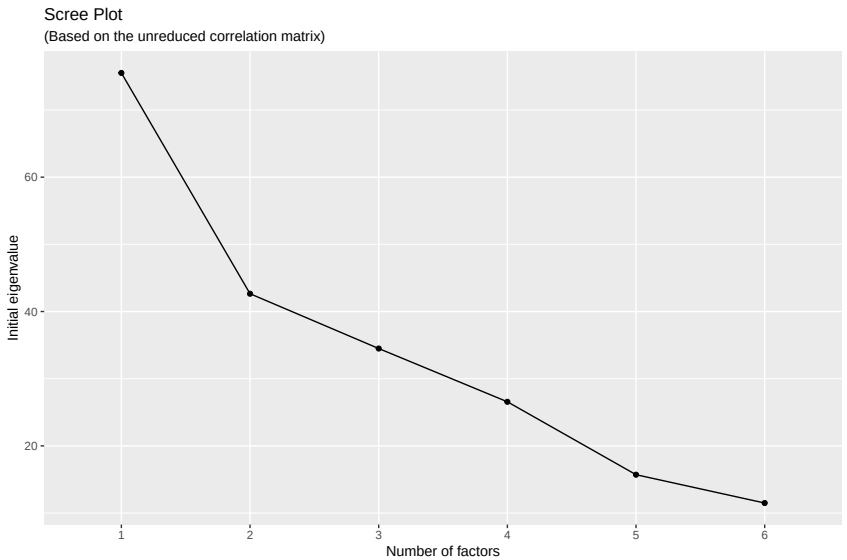
using factor analysis with covariance matrix and without varimax method:

```
fa5 <- fa(new.data.y,nfactors = 6, rotate = "none" ,
          scores = "regression" ,covar = TRUE )
fa5
```

```
## Factor Analysis using method =  minres
## Call: fa(r = new.data.y, nfactors = 6, rotate = "none", scores = "regression",
##     covar = TRUE)
## Unstandardized loadings (pattern matrix) based upon covariance matrix
##      MR1  MR2   MR3   MR4   MR5 MR6 h2 u2   H2   U2
## y1  2.05  3.1  1.29  0.85 -0.94   0 17 12 0.59 0.41
## y2 -0.29  1.8 -0.52  3.03  0.78   0 14 12 0.52 0.48
## y3  0.67  2.6 -0.81 -1.66  1.19   0 12 12 0.49 0.51
## y4  5.45 -2.0  2.48  0.34  0.63   0 40 13 0.76 0.24
## y5  5.06 -0.3 -3.48  0.11 -0.35   0 38 13 0.75 0.25
## y6  1.75  2.6  1.08 -1.23 -0.18   0 13 11 0.54 0.46
##
##                        MR1   MR2   MR3   MR4  MR5  MR6
## SS loadings          63.02 30.70 22.03 14.31 3.47 0.00
## Proportion Var        0.31  0.15  0.11  0.07 0.02 0.00
## Cumulative Var        0.31  0.45  0.56  0.63 0.65 0.65
## Proportion Explained  0.47  0.23  0.16  0.11 0.03 0.00
## Cumulative Proportion 0.47  0.70  0.87  0.97 1.00 1.00
##
##  Standardized loadings (pattern matrix)
##    item   MR1   MR2   MR3   MR4   MR5 MR6   h2   u2
## y1    1  0.38  0.58  0.24  0.16 -0.17   0 0.59 0.41
## y2    2 -0.06  0.36 -0.10  0.59  0.15   0 0.52 0.48
## y3    3  0.14  0.53 -0.17 -0.34  0.24   0 0.49 0.51
## y4    4  0.75 -0.27  0.34  0.05  0.09   0 0.76 0.24
## y5    5  0.71 -0.04 -0.49  0.02 -0.05   0 0.75 0.25
## y6    6  0.36  0.54  0.22 -0.25 -0.04   0 0.54 0.46
##
##                 MR1  MR2  MR3  MR4  MR5  MR6
```

Factor Analysis on Data

```
n_factors <- length(fa5$e.values)
scree     <- data.frame(
  Factor_n =  as.factor(1:n_factors),
  Eigenvalue = fa5$e.values)
ggplot(scree, aes(x = Factor_n, y = Eigenvalue, group = 1)) +
  geom_point() + geom_line() +
  xlab("Number of factors") +
  ylab("Initial eigenvalue") +
  labs( title = "Scree Plot",
        subtitle = "(Based on the unreduced correlation matrix)")
```



Factor Analysis on Data

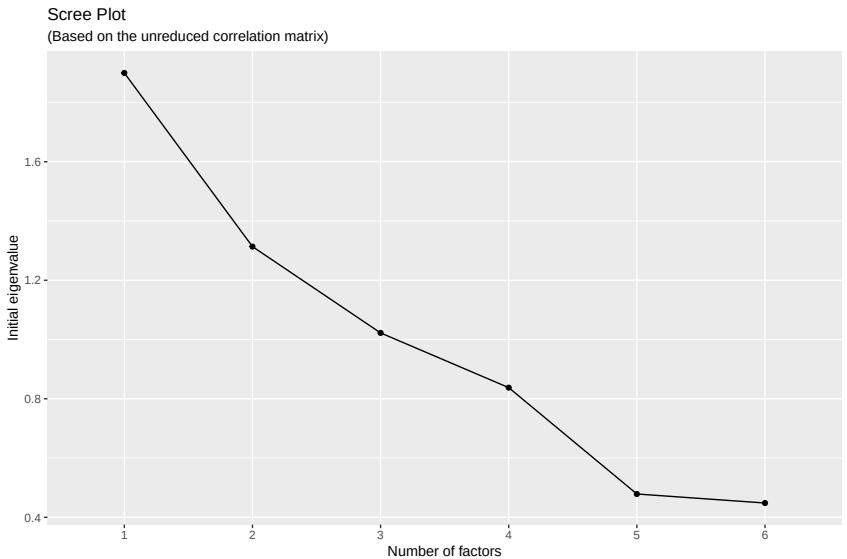
using factor analysis with correlation matrix and varimax rotate :

```
fa6 <- fa(new.data.y,nfactors = 6, rotate = "varimax",
          scores = "regression" )
fa6
```

```
## Factor Analysis using method =  minres
## Call: fa(r = new.data.y, nfactors = 6, rotate = "varimax", scores = "regression")
## Standardized loadings (pattern matrix) based upon correlation matrix
##     MR1   MR2   MR4   MR3   MR5 MR6   h2   u2 com
## y1 0.67  0.15  0.09  0.31  0.07   0 0.58 0.42 1.6
## y2 0.07 -0.05  0.01  0.60  0.00   0 0.37 0.63 1.0
## y3 0.23  0.03  0.66  0.02  0.00   0 0.49 0.51 1.2
## y4 0.29  0.64 -0.26 -0.20  0.00   0 0.59 0.41 2.0
## y5 0.02  0.66  0.17  0.04  0.00   0 0.47 0.53 1.1
## y6 0.69  0.09  0.33 -0.08 -0.07   0 0.60 0.40 1.5
##
##                        MR1  MR2  MR4  MR3  MR5  MR6
## SS loadings           1.05 0.87 0.65 0.51 0.01 0.00
## Proportion Var        0.18 0.15 0.11 0.09 0.00 0.00
## Cumulative Var        0.18 0.32 0.43 0.52 0.52 0.52
## Proportion Explained  0.34 0.28 0.21 0.16 0.00 0.00
## Cumulative Proportion 0.34 0.62 0.83 1.00 1.00 1.00
##
## Mean item complexity =  1.4
## Test of the hypothesis that 6 factors are sufficient.
##
## The degrees of freedom for the null model are  15  and the objective function was  0.78 with Chi Square
## The degrees of freedom for the model are -6  and the objective function was  0
##
## The root mean square of the residuals (RMSR) is  0
## The df corrected root mean square of the residuals is  NA
##
## The harmonic number of observations is  95 with the empirical chi square  0  with prob <  NA
## The total number of observations was  95  with Likelihood Chi Square =  0  with prob <  NA
```

Factor Analysis on Data

```
n_factors <- length(fa6$e.values)
scree     <- data.frame(
  Factor_n =  as.factor(1:n_factors),
  Eigenvalue = fa6$e.values)
ggplot(scree, aes(x = Factor_n, y = Eigenvalue, group = 1)) +
  geom_point() + geom_line() +
  xlab("Number of factors") +
  ylab("Initial eigenvalue") +
  labs( title = "Scree Plot",
        subtitle = "(Based on the unreduced correlation matrix)")
```



Factor Analysis on Data

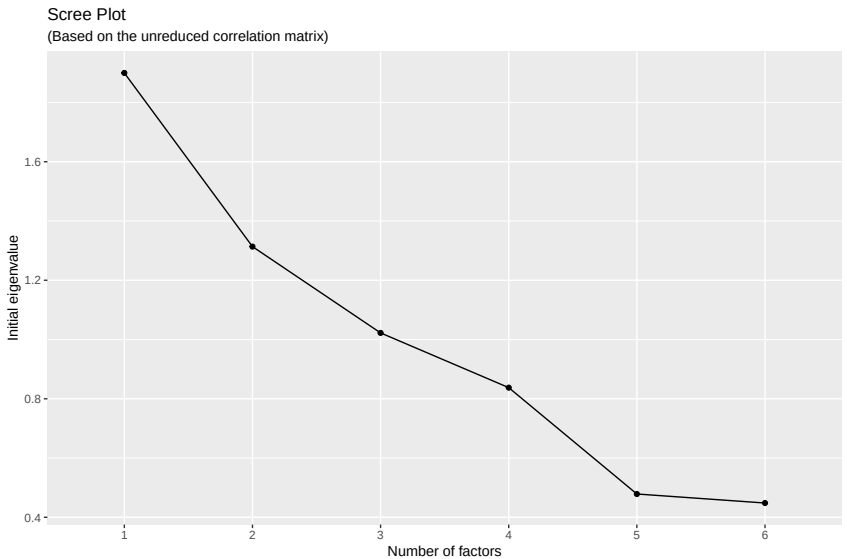
using factor analysis with correlation matrix and without varimax rotate :

```
fa7 <- fa(new.data.y,nfactors = 6, rotate = "none" ,
          scores = "regression" )
fa7
```

```
## Factor Analysis using method =  minres
## Call: fa(r = new.data.y, nfactors = 6, rotate = "none", scores = "regression")
## Standardized loadings (pattern matrix) based upon correlation matrix
##     MR1   MR2   MR3   MR4   MR5 MR6   h2   u2 com
## y1 0.67 -0.17  0.30 -0.13 -0.05   0 0.58 0.42 1.6
## y2 0.10 -0.30  0.48  0.21  0.04   0 0.37 0.63 2.2
## y3 0.44 -0.35 -0.34  0.25 -0.02   0 0.49 0.51 3.5
## y4 0.41  0.64  0.08 -0.11  0.00   0 0.59 0.41 1.8
## y5 0.39  0.40 -0.02  0.39  0.00   0 0.47 0.53 3.0
## y6 0.70 -0.18 -0.17 -0.22  0.05   0 0.60 0.40 1.5
##
##                        MR1  MR2  MR3  MR4  MR5  MR6
## SS loadings           1.46 0.83 0.47 0.33 0.01 0.00
## Proportion Var        0.24 0.14 0.08 0.06 0.00 0.00
## Cumulative Var        0.24 0.38 0.46 0.52 0.52 0.52
## Proportion Explained  0.47 0.27 0.15 0.11 0.00 0.00
## Cumulative Proportion 0.47 0.74 0.89 1.00 1.00 1.00
##
## Mean item complexity =  2.3
## Test of the hypothesis that 6 factors are sufficient.
##
## The degrees of freedom for the null model are  15  and the objective function was  0.78 with Chi Square
## The degrees of freedom for the model are -6  and the objective function was  0
##
## The root mean square of the residuals (RMSR) is  0
## The df corrected root mean square of the residuals is  NA
##
## The harmonic number of observations is  95 with the empirical chi square  0  with prob <  NA
## The total number of observations was  95  with Likelihood Chi Square =  0  with prob <  NA
```

Factor Analysis on Data

```
n_factors <- length(fa7$e.values)
scree     <- data.frame(
  Factor_n =  as.factor(1:n_factors),
  Eigenvalue = fa7$e.values)
ggplot(scree, aes(x = Factor_n, y = Eigenvalue, group = 1)) +
  geom_point() + geom_line() +
  xlab("Number of factors") +
  ylab("Initial eigenvalue") +
  labs( title = "Scree Plot",
        subtitle = "(Based on the unreduced correlation matrix)")
```



discriminats analysis on Data

Now we want to do discriminats analysis on Data. we can use lda function to do it. after that we will see the outputs and we will plotting new data.
so at the first we add our faculty column to new.data.y data frame.

```
new.data.y$group = Data$Faculty
new.data.y$group[which(new.data.y$group == "Humanities,lite
new.data.y$group[which(new.data.y$group== "Agriculture and
library(MASS)
m1 = lda (group~. , data = new.data.y)
```

discriminats analysis on Data

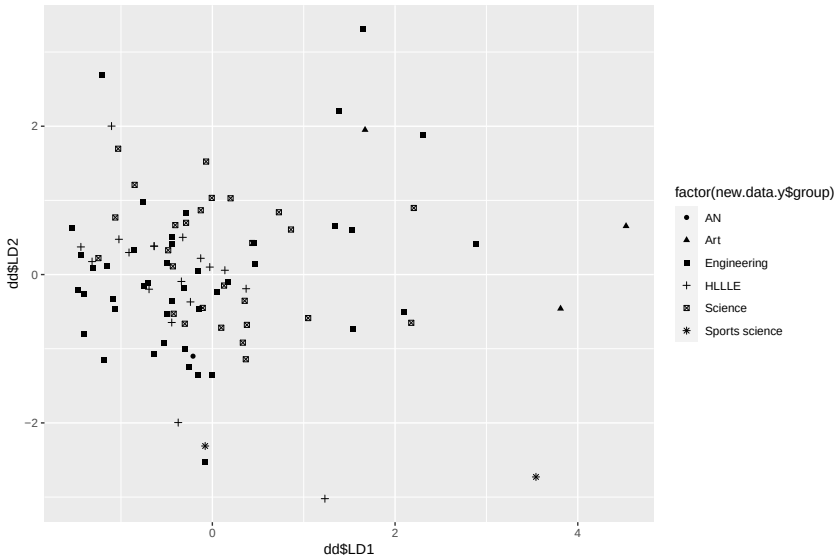
```
m1
```

```
## Call:
## lda(group ~ ., data = new.data.y)
##
## Prior probabilities of groups:
##             AN            Art    Engineering          HLLLE        Science
##     0.01052632     0.03157895     0.46315789     0.18947368     0.28421053
## Sports science
##     0.02105263
##
## Group means:
##                        y1        y2        y3        y4        y5        y6
## AN              3.000000  0.000000  2.000000 20.000000 10.000000 1.0000000
## Art            10.916667 11.666667 13.527778  8.000000 10.027778 9.3333333
## Engineering     4.868561  2.613636  2.250000 12.270833 14.094318 3.2897727
## HLLLE           4.189815  1.805556  1.000000 10.277778 12.500000 2.3611111
## Science         4.744444  2.742593  2.964815  7.835185 11.209259 3.8703704
## Sports science 13.916667  2.666667  6.525000 11.516667  6.016667 0.5833333
##
## Coefficients of linear discriminants:
##            LD1         LD2          LD3         LD4          LD5
## y1  0.06147218 -0.15924621 -0.071575448  0.12056779  0.019331489
## y2  0.10073898  0.09038684  0.077818772  0.02000412 -0.142986791
## y3  0.18434682 -0.03183761  0.048786539 -0.05112177  0.064492610
## y4  0.01191409 -0.06189764  0.130981833 -0.06147554 -0.004667816
## y5 -0.06673860  0.05544194  0.027365325  0.11468410  0.028063577
## y6 -0.01122268  0.19732705 -0.003674073 -0.03334621  0.077890124
##
## Proportion of trace:
##    LD1    LD2    LD3    LD4    LD5
## 0.6214 0.2365 0.1197 0.0205 0.0018
```

discriminats analysis on Data

Now we want to plot it

```
#plot(m1)
library(ggplot2)
pp = predict(m1)
dd= data.frame(LD1 = pp $ x [,1] ,LD2 = pp $ x[ , 2] )
ggplot(data = dd, aes(x = dd$LD1 , y = dd$LD2 ))+
  geom_point(aes (shape = factor(new.data.y$group)) , data
```



discriminats analysis on Data

Now we want to prediction and calculate the accuracy of model.

```
p1 <- predict(m1)$class
tab <- table(Predicted = p1, Actual = new.data.y$group)
tab
```

```
##                 Actual
## Predicted        AN Art Engineering HLLLE Science Sports science
##   AN              0   0           0     0       0               0
##   Art             0   2           2     0       0               0
##   Engineering     1   0          34    12      17               1
##   HLLLE           0   0           0     0       0               0
##   Science         0   1           8     5      10               0
##   Sports science  0   0           0     1       0               1
```

```
sum(diag(tab))/sum(tab)
```

```
## [1] 0.4947368
```

Clustering On Data With euclidean Distance

Now we want to use Clustering methods On our data with continues variables and columns in Dataset.

```
data = Data[,c(3,7,9,12,15,18,21,24)]
colnames(data) = c("Age" , "vorodi" , "Exercise_time" , "Study_time" , "Art_time" , "audio_visual_time" ,
head(data,4)
```

```
##   Age vorodi Exercise_time Study_time Art_time audio_visual_time social_time
## 1  20     97             2          2        0                 7          10
## 2  18     99             5          4        6                 6           0
## 3  21     97             3          0        2                10           5
## 4  21     97             2          0        0                 6           7
##   phone_time
## 1         10
## 2          1
## 3         20
## 4         20
```

Clustering On Data With euclidean Distance

we should make distance matrix with euclidean method to calculate distance values :

```
(Dist = dist(data[1:5,] , method = "euclidean",
             diag = TRUE , upper = TRUE))
```

```
##          1        2        3        4        5
## 1  0.00000 15.45962 12.00000 10.72381 21.79449
## 2 15.45962  0.00000 21.23676 22.00000 23.57965
## 3 12.00000 21.23676  0.00000  5.00000 16.09348
## 4 10.72381 22.00000  5.00000  0.00000 19.39072
## 5 21.79449 23.57965 16.09348 19.39072  0.00000
```

```
Dist1 = dist(data , method = "euclidean",
             diag = TRUE , upper = TRUE)
```

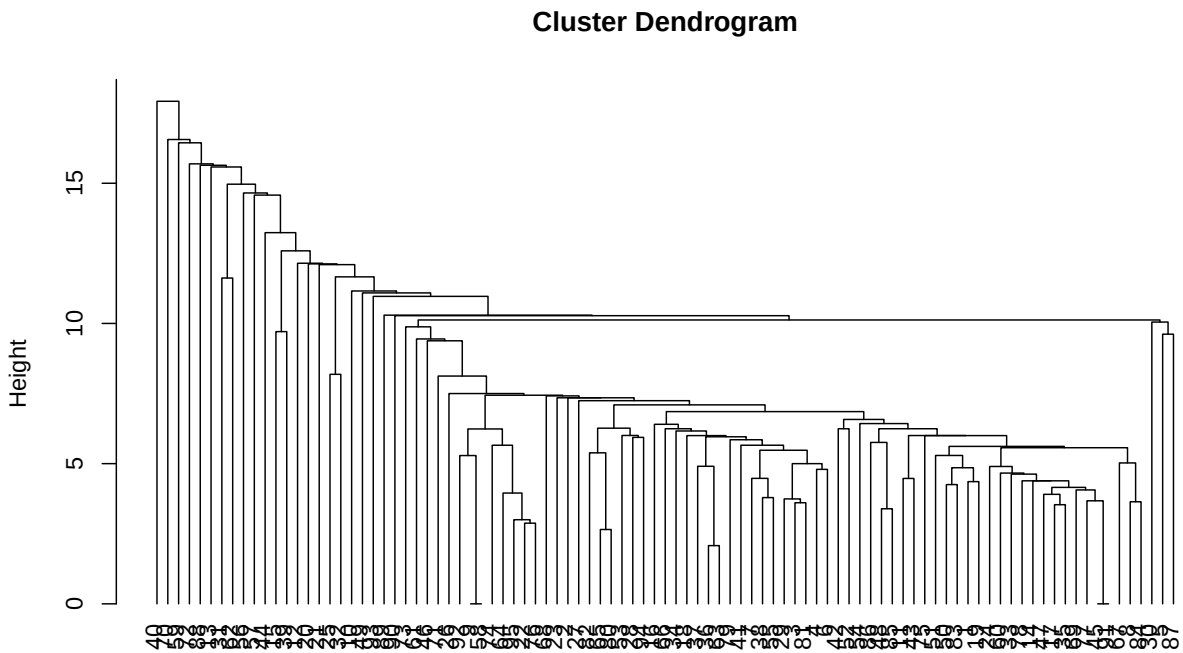
Clustering On Data With euclidean Distance

To use Single method for clustering we have:

```
#single method
model1 = hclust(Dist1 , method = "single")
model1
```

```
##
## Call:
## hclust(d = Dist1, method = "single")
##
## Cluster method   : single
## Distance         : euclidean
## Number of objects: 95
```

```
plot( model1 , hang = -1  )
```



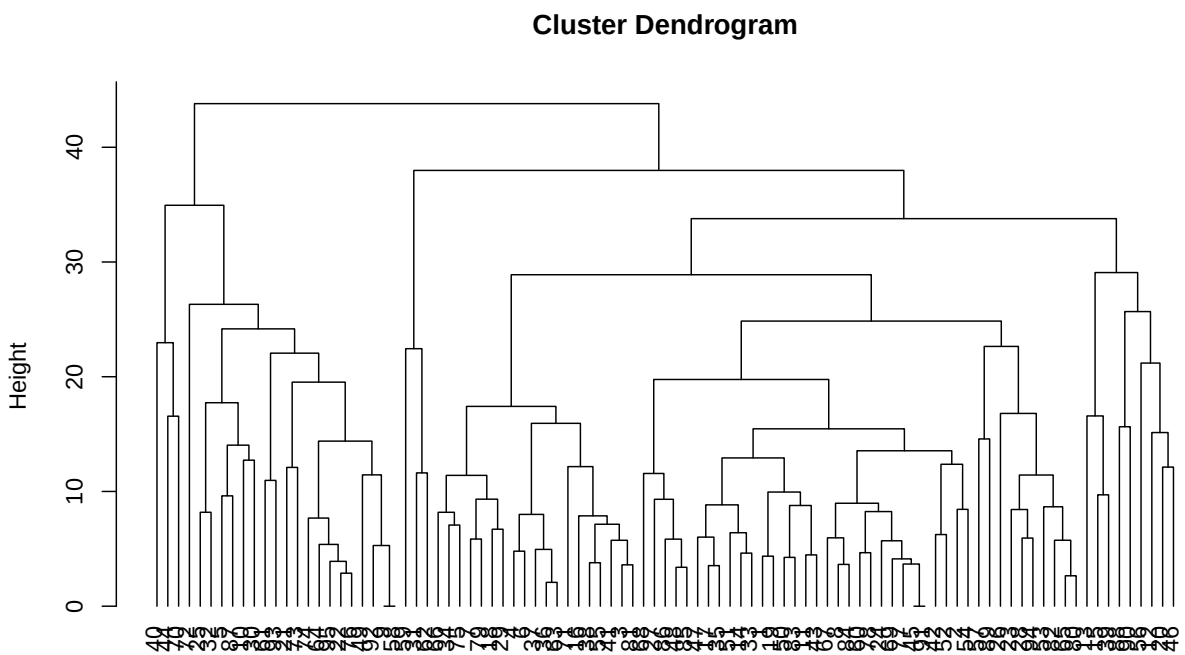
Clustering On Data With euclidean Distance

To use Complete method for Clustering we have:

```
model2 = hclust(Dist1 , method = "complete")
model2
```

```
##
## Call:
## hclust(d = Dist1, method = "complete")
##
## Cluster method   : complete
## Distance         : euclidean
## Number of objects: 95
```

```
plot( model2 , hang = -1  )
```



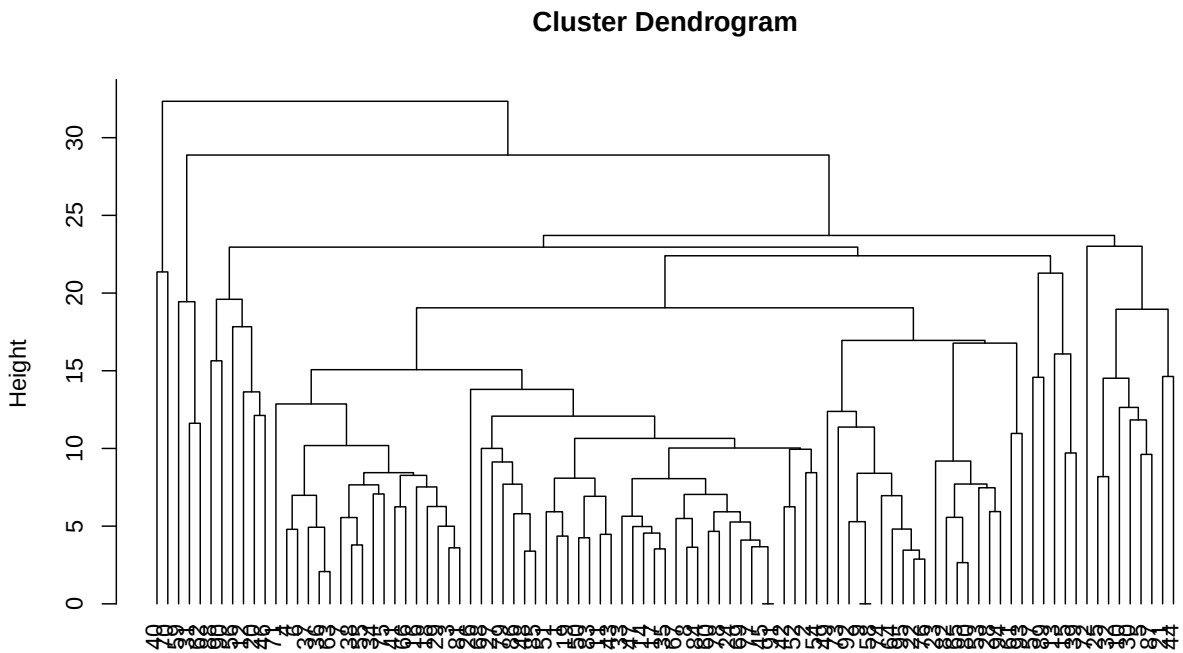
Clustering On Data With euclidean Distance

To use Average method for Clustering we have:

```
#average method:
model3 = hclust(Dist1 , method = "average")
model3
```

```
##
## Call:
## hclust(d = Dist1, method = "average")
##
## Cluster method   : average
## Distance         : euclidean
## Number of objects: 95
```

```
plot( model3 , hang = -1  )
```



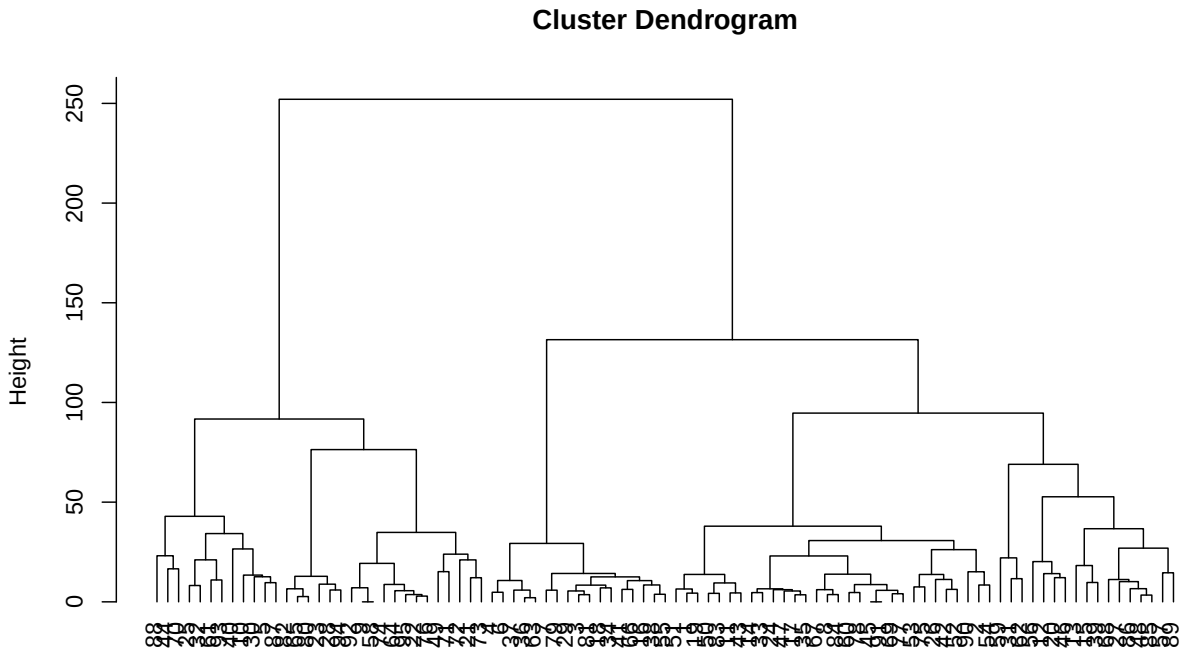
Clustering On Data With euclidean Distance

To use Ward method for Clustering we have:

```
#ward method:
model4 = hclust(Dist1 , method = "ward.D")
model4
```

```
##
## Call:
## hclust(d = Dist1, method = "ward.D")
##
## Cluster method   : ward.D
## Distance         : euclidean
## Number of objects: 95
```

```
plot( model4 , hang = -1  )
```



Clustering On Data With manhattan Distance

we should make distance matrix with manhattan method to calculate distance values :

```
(Dist = dist(data[1:5,] , method = "manhattan",
             diag = TRUE , upper = TRUE))
```

```
##    1  2  3  4  5
## 1  0 35 24 17 53
## 2 35  0 43 44 48
## 3 24 43  0  9 33
## 4 17 44  9  0 42
## 5 53 48 33 42  0
```

```
Dist1 = dist(data , method = "euclidean",
             diag = TRUE , upper = TRUE)
```

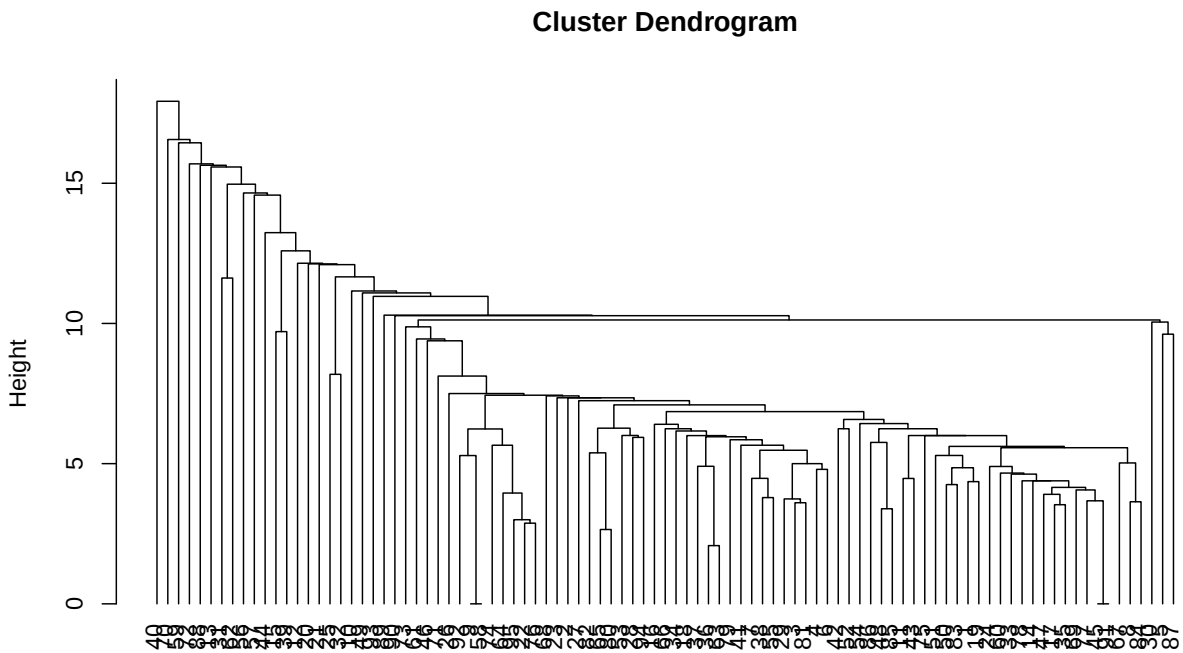
Clustering On Data With manhattan Distance

To use Single method for clustering we have:

```
#single method
model1 = hclust(Dist1 , method = "single")
model1
```

```
##
## Call:
## hclust(d = Dist1, method = "single")
##
## Cluster method   : single
## Distance         : euclidean
## Number of objects: 95
```

```
plot( model1 , hang = -1  )
```



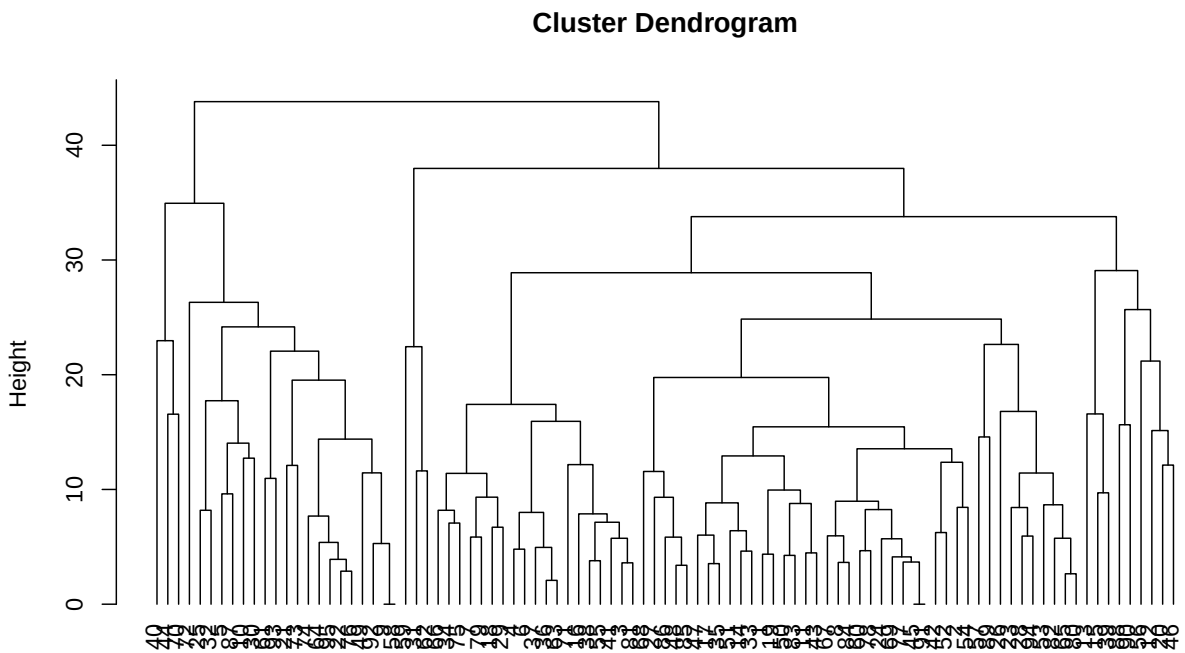
Clustering On Data With manhattan Distance

To use Complete method for Clustering we have:

```
model2 = hclust(Dist1 , method = "complete")
model2
```

```
##
## Call:
## hclust(d = Dist1, method = "complete")
##
## Cluster method   : complete
## Distance         : euclidean
## Number of objects: 95
```

```
plot( model2 , hang = -1  )
```



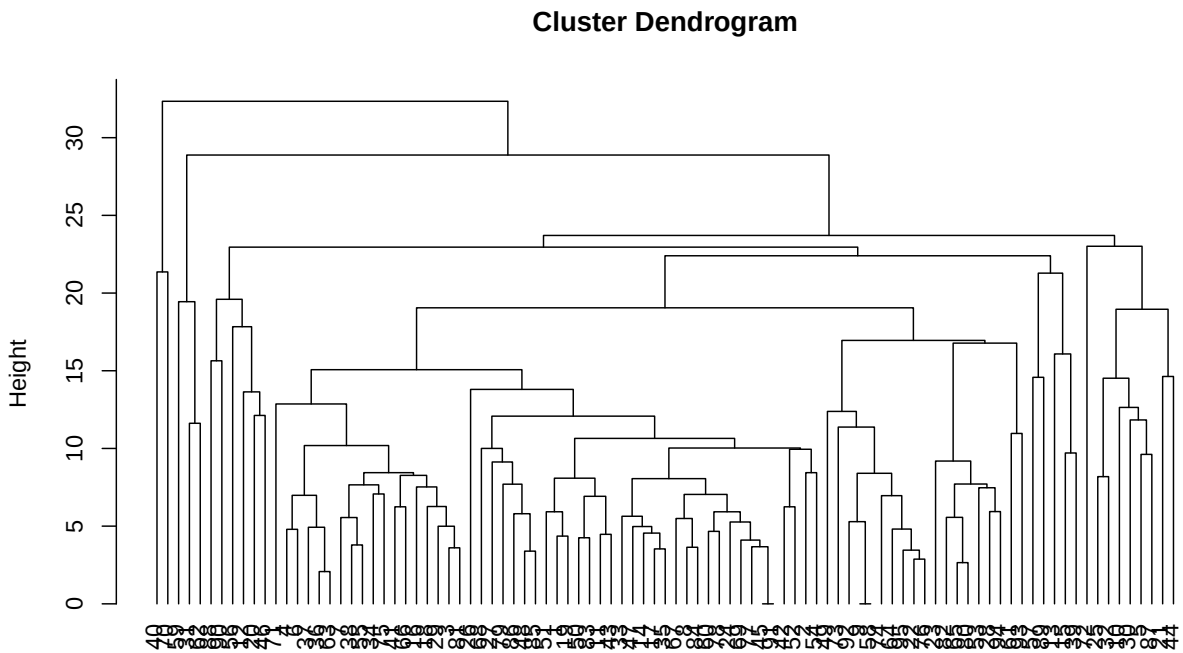
Clustering On Data With manhattan Distance

To use Average method for Clustering we have:

```
#average method:
model3 = hclust(Dist1 , method = "average")
model3
```

```
##
## Call:
## hclust(d = Dist1, method = "average")
##
## Cluster method   : average
## Distance         : euclidean
## Number of objects: 95
```

```
plot( model3 , hang = -1  )
```



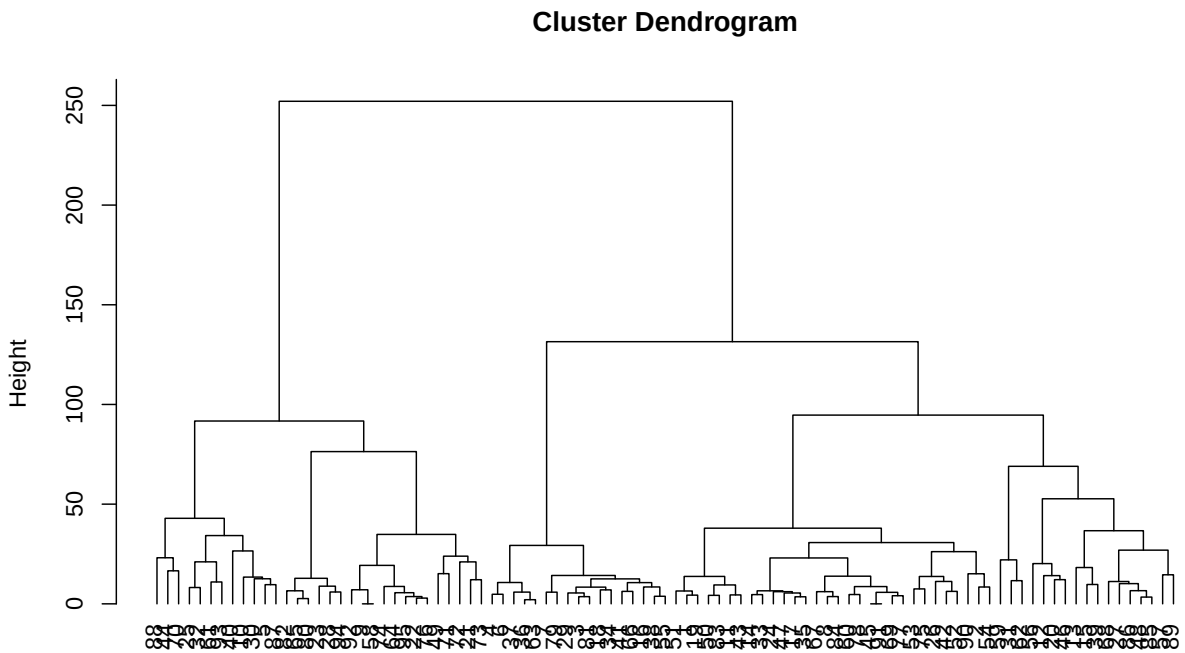
Clustering On Data With manhattan Distance

To use Ward method for Clustering we have:

```
#ward method:
model4 = hclust(Dist1 , method = "ward.D")
model4
```

```
##
## Call:
## hclust(d = Dist1, method = "ward.D")
##
## Cluster method   : ward.D
## Distance         : euclidean
## Number of objects: 95
```

```
plot( model4 , hang = -1  )
```



K-Means Clustering

in this sigment we want to use K-Means Method With four way

K-Means Way:

- Hartigan-Wong
- Lloyd
- Forgy
- MacQueen

Determining Optimal Clusters

- Elbow method
- Silhouette method
- Gap statistic

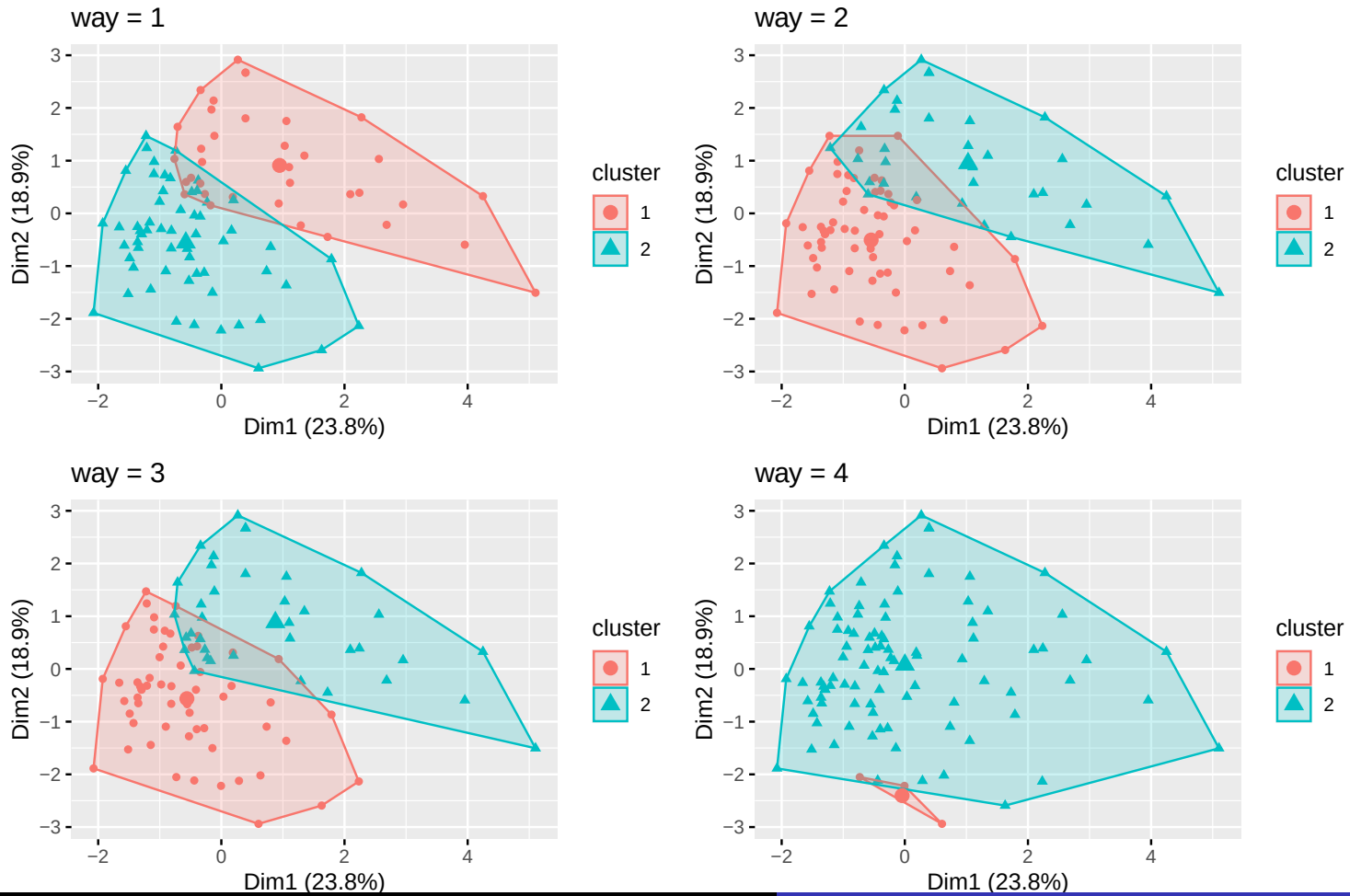
K-Means Method With 2 cluster

Now we want to clustering our data with 2 cluster and K-means method and four way.

```
library(factoextra)
library(gridExtra)
model1 = kmeans(data , 2  , algorithm = "Hartigan-Wong")
model2 = kmeans(data , 2  , algorithm = "Lloyd")
model3 = kmeans(data , 2  , algorithm = "Forgy")
model4 = kmeans(data , 2  , algorithm = "MacQueen")
```

K-Means Method With 2 cluster

```
# plots to compare with 2 cluster
p1 <- fviz_cluster(model1, geom = "point", data = data) + ggtitle("way = 1")
p2 <- fviz_cluster(model2, geom = "point",  data = data) + ggtitle("way = 2")
p3 <- fviz_cluster(model3, geom = "point",  data = data) + ggtitle("way = 3")
p4 <- fviz_cluster(model4, geom = "point",  data = data) + ggtitle("way = 4")
grid.arrange(p1, p2, p3, p4, nrow = 2)
```



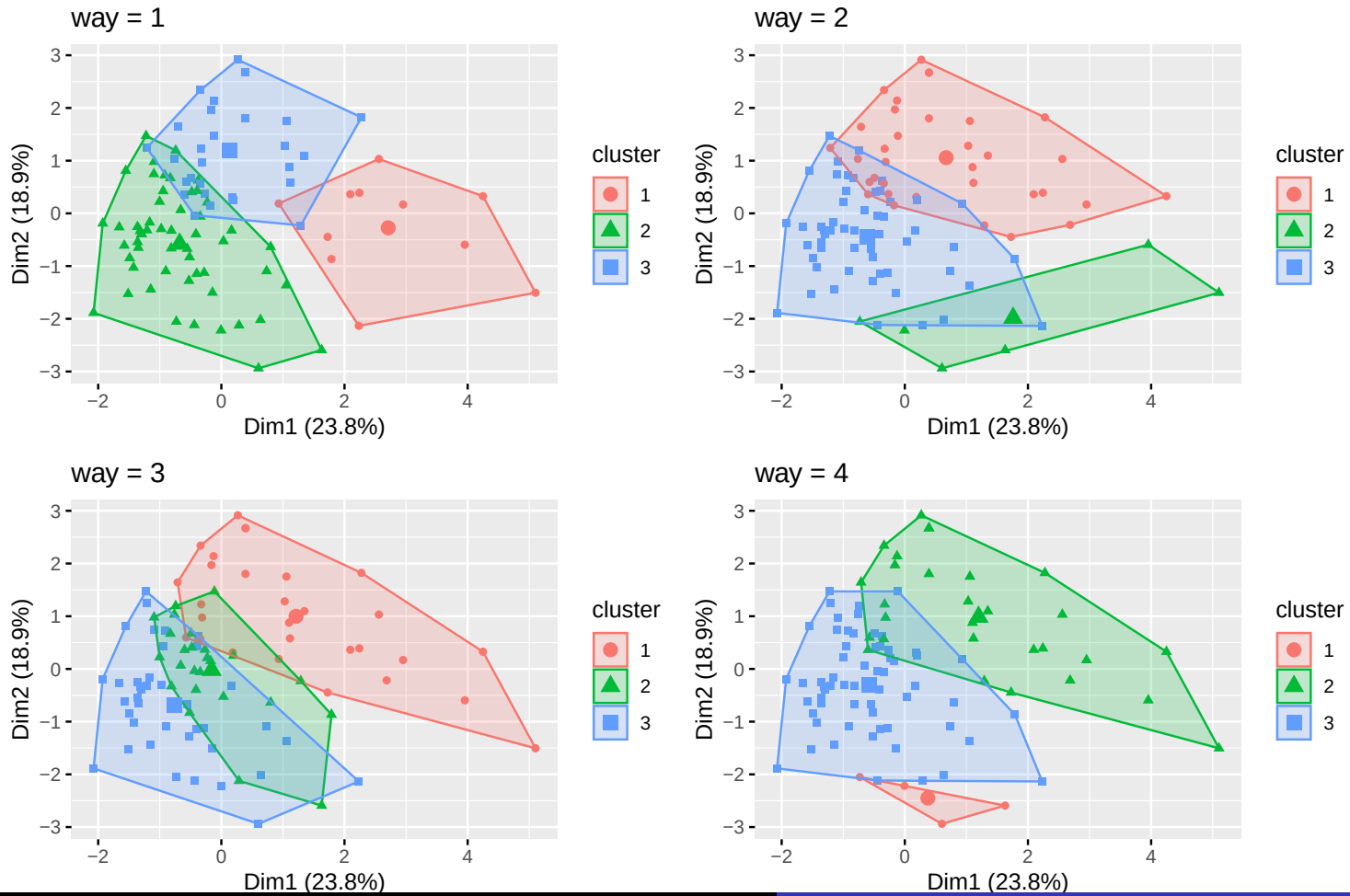
K-Means Method With 3 Cluster

Now we want to clustering our data with 3 cluster and K-means method and four way.

```
model1 = kmeans(data , 3   , algorithm = "Hartigan-Wong")
model2 = kmeans(data , 3   , algorithm = "Lloyd")
model3 = kmeans(data , 3   , algorithm = "Forgy")
model4 = kmeans(data , 3   , algorithm = "MacQueen")
```

K-Means Method With 3 cluster

```
# plots to compare with 3 cluster
p1 <- fviz_cluster(model1, geom = "point", data = data) + ggtitle("way = 1")
p2 <- fviz_cluster(model2, geom = "point",  data = data) + ggtitle("way = 2")
p3 <- fviz_cluster(model3, geom = "point",  data = data) + ggtitle("way = 3")
p4 <- fviz_cluster(model4, geom = "point",  data = data) + ggtitle("way = 4")
grid.arrange(p1, p2, p3, p4, nrow = 2)
```



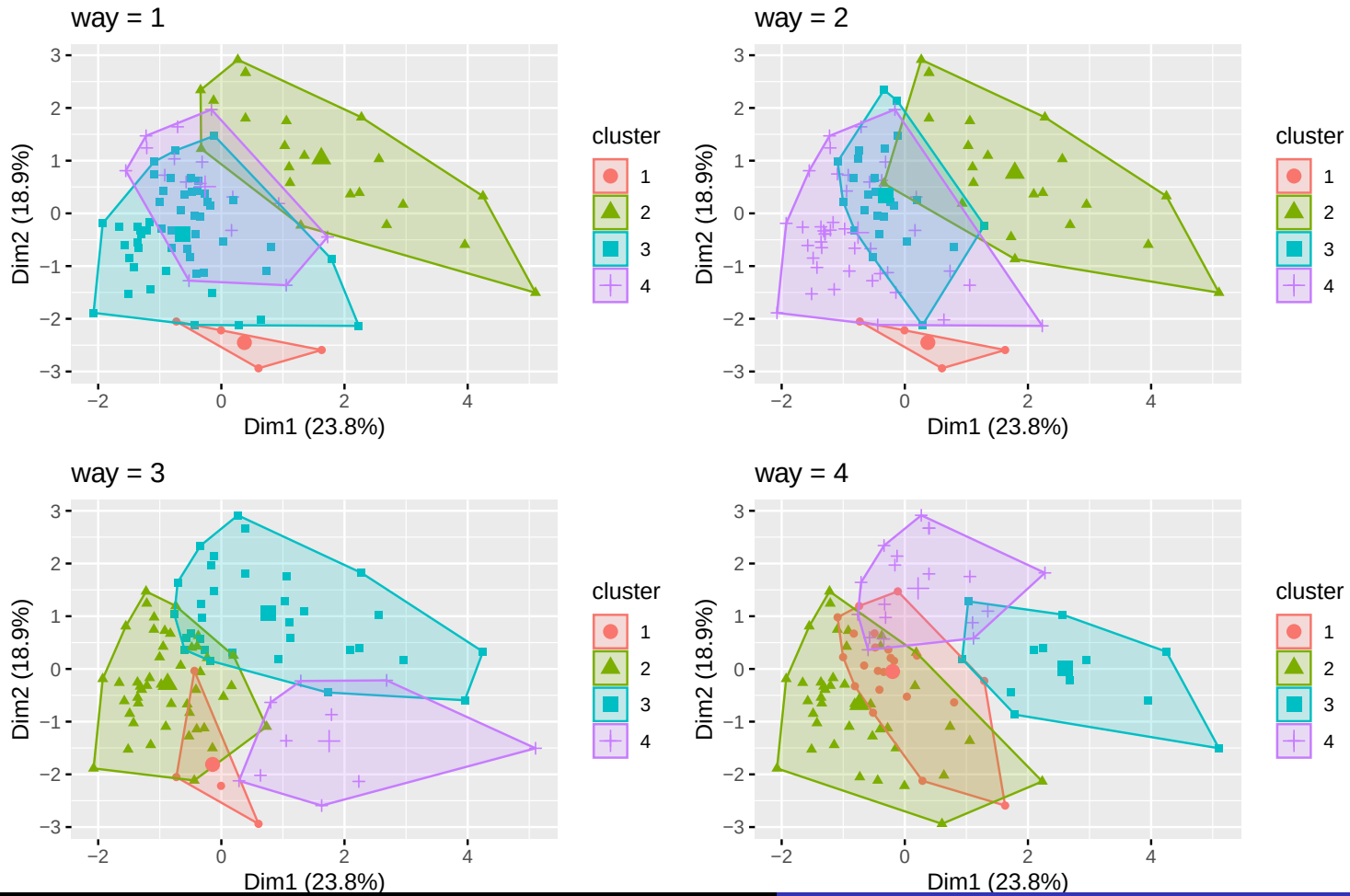
K-Means method With 4 Cluster

Now we want to clustering our data with 4 cluster and K-means method and four way.

```
#library(factoextra)
#library(gridExtra)
model1 = kmeans(data , 4   , algorithm = "Hartigan-Wong")
model2 = kmeans(data , 4   , algorithm = "Lloyd")
model3 = kmeans(data , 4   , algorithm = "Forgy")
model4 = kmeans(data , 4   , algorithm = "MacQueen")
```

K-Means Method With 4 cluster

```
# plots to compare with 4 cluster
p1 <- fviz_cluster(model1, geom = "point", data = data) + ggtitle("way = 1")
p2 <- fviz_cluster(model2, geom = "point",  data = data) + ggtitle("way = 2")
p3 <- fviz_cluster(model3, geom = "point",  data = data) + ggtitle("way = 3")
p4 <- fviz_cluster(model4, geom = "point",  data = data) + ggtitle("way = 4")
grid.arrange(p1, p2, p3, p4, nrow = 2)
```



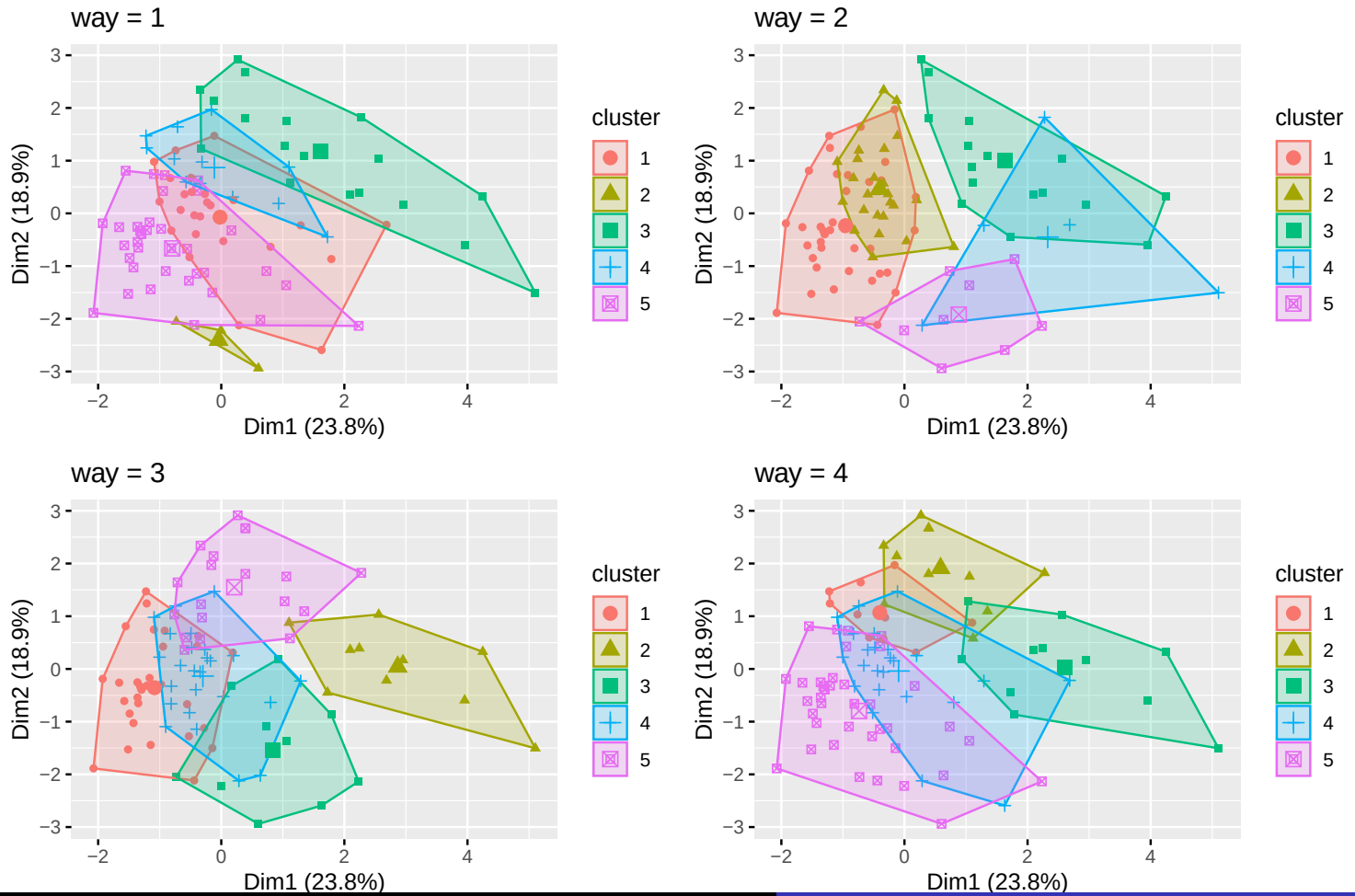
K-Means method With 5 Cluster

Now we want to clustering our data with 5 cluster and K-means method and four way.

```
#library(factoextra)
#library(gridExtra)
model1 = kmeans(data , 5   , algorithm = "Hartigan-Wong")
model2 = kmeans(data , 5   , algorithm = "Lloyd")
model3 = kmeans(data , 5   , algorithm = "Forgy")
model4 = kmeans(data , 5   , algorithm = "MacQueen")
```

K-Means Method With 5 cluster

```
# plots to compare with 5 cluster
p1 <- fviz_cluster(model1, geom = "point", data = data) + ggtitle("way = 1")
p2 <- fviz_cluster(model2, geom = "point",  data = data) + ggtitle("way = 2")
p3 <- fviz_cluster(model3, geom = "point",  data = data) + ggtitle("way = 3")
p4 <- fviz_cluster(model4, geom = "point",  data = data) + ggtitle("way = 4")
grid.arrange(p1, p2, p3, p4, nrow = 2)
```



K-Means Determining Optimal Clusters

Now we want to Determining optimal Clusters with Elbow Method, for $K = 1, 2, \ldots, 50$.

```
# Elbow Method :
# function to compute total within-cluster sum of square
wss <- function(k) {
  kmeans(data, k )$tot.withinss
}

# Compute and plot wss for k = 1 to k = 50
k.values <- 1:50

# extract wss for 2-50 clusters
library(tidyverse)
```

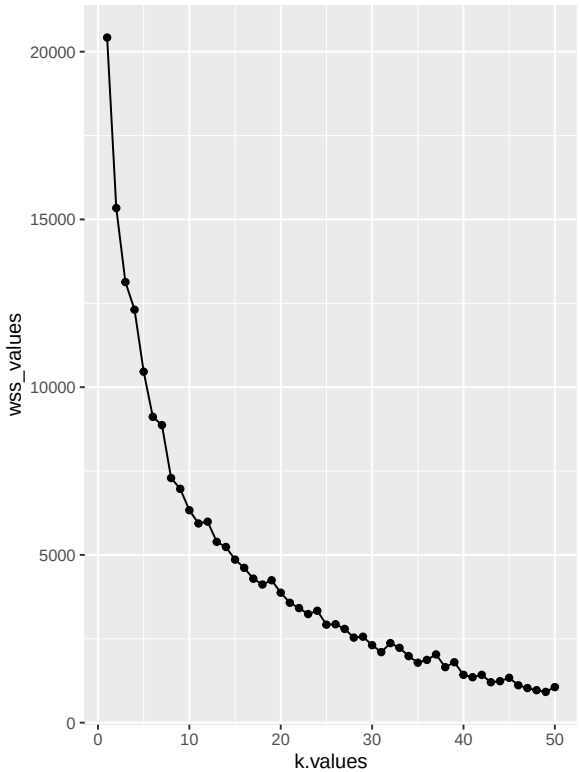
```
## -- Attaching packages --------------------------------------- tidyverse 1.3.1 --

## v tibble  3.1.5     v dplyr   1.0.7
## v tidyr   1.1.4     v stringr 1.4.0
## v readr   2.1.1     v forcats 0.5.1
## v purrr   0.3.4

## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x psych::%+%()     masks ggplot2::%+%()
## x psych::alpha()   masks ggplot2::alpha()
## x dplyr::combine() masks gridExtra::combine()
## x dplyr::filter()  masks stats::filter()
## x dplyr::lag()     masks stats::lag()
## x dplyr::select()  masks MASS::select()
```

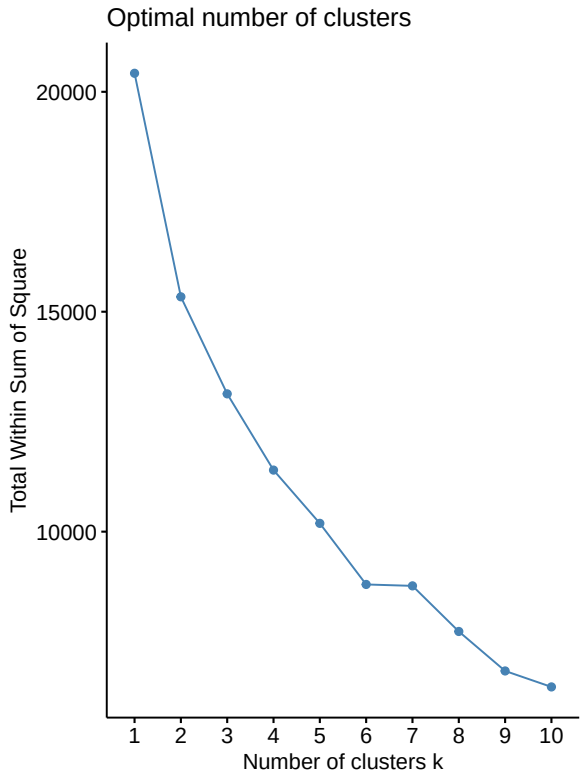
```
wss_values <- map_dbl(k.values, wss)
```

K-Means Determining Optimal Clusters

Now we should ploting last slide outputs too show the best number of clusters.

```
p1= ggplot(mapping =aes(k.values, wss_values),
     type="b", pch = 19, frame = FALSE,
     xlab="Number of clusters K",
     ylab="Total within-clusters sum of squares")+
  geom_point()+geom_line()
p2= fviz_nbclust(data, kmeans, method = "wss")
grid.arrange(p1, p2, nrow = 1)
```





K-Means Determining Optimal Clusters

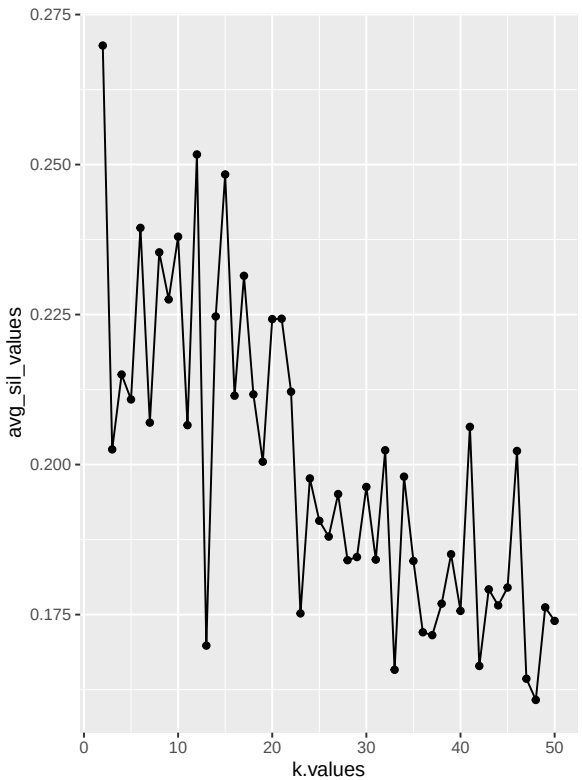
Now we want to compute Average silhouette for K Clusters, $k = 1, 2, ..., 50$.

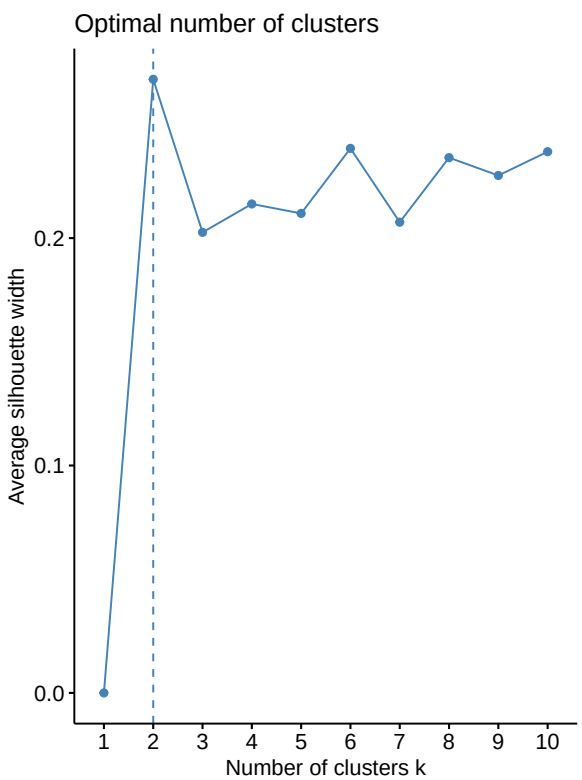
```
# function to compute average silhouette for k clusters
library(cluster)
avg_sil <- function(k) {
  km.res <- kmeans(data, centers = k)
  ss <- silhouette(km.res$cluster, dist(data))
  mean(ss[, 3])
}
# Compute and plot wss for k = 2 to k = 50
k.values <- 2:50
# extract avg silhouette for 2-50 clusters
avg_sil_values <- map_dbl(k.values, avg_sil)
```

K-Means Determining Optimal Clusters

Now we want to ploting them.

```
p1 = ggplot(mapping= aes(k.values, avg_sil_values),
     type = "b", pch = 19, frame = FALSE,
     xlab = "Number of clusters K",
     ylab = "Average Silhouettes")+
  geom_point()+geom_line()

p2 = fviz_nbclust(data, kmeans, method = "silhouette" )
grid.arrange(p1, p2, nrow = 1)
```





K-Means Determining Optimal Clusters

Now we want to determining optimal clusters with gap statistics way. # compute gap statistic

```
gap_stat <- clusGap(data, FUN = kmeans,
                    K.max = 50, B = 50)
# Print the result
print(gap_stat, method = "firstmax")
```

```
## Clustering Gap statistic ["clusGap"] from call:
## clusGap(x = data, FUNcluster = kmeans, K.max = 50, B = 50)
## B=50 simulated reference sets, k = 1..50; spaceH0="scaledPCA"
##  --> Number of clusters (method 'firstmax'): 6
##           logW   E.logW       gap     SE.sim
##  [1,] 6.125936 6.535018 0.4090825 0.01990354
##  [2,] 5.970122 6.421421 0.4512990 0.02076530
##  [3,] 5.872233 6.336881 0.4646482 0.02156149
##  [4,] 5.794430 6.271843 0.4774127 0.02140878
##  [5,] 5.731945 6.220528 0.4885830 0.02189745
##  [6,] 5.664811 6.175409 0.5105975 0.02192438
##  [7,] 5.646449 6.131333 0.4848848 0.02018628
##  [8,] 5.577513 6.096186 0.5186729 0.01980907
##  [9,] 5.510662 6.058957 0.5482947 0.02179173
## [10,] 5.483701 6.023846 0.5401450 0.02154658
## [11,] 5.439049 5.994496 0.5554473 0.02308755
## [12,] 5.412060 5.964868 0.5528080 0.02227443
## [13,] 5.383017 5.935192 0.5521746 0.02007622
```

K-Means Determining Optimal Clusters

Now we want to ploting Gap statistics with sd .

```
fviz_gap_stat(gap_stat)
```