Cluster Analysis for Customer Churn

Clustering is the task of dividing data points into set of groups such that data points in the same groups are more similar to other data points in the same group and dissimilar to the data points in other groups. It is basically a collection of objects on the basis of similarity and dissimilarity between them.

For ex– The data points in the graph below clustered together can be classified into one single group.

There are different types of clustering techniques:

1. Hierarchical.
2. Non-hierarchical.

```
library(cluster)
```

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

```
library(dplyr)
```

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

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

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

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

```
library(ggplot2)
```

```
## Warning: package 'ggplot2' was built under R version 3.6.2
```

```
library(readr)
```

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

```
library(Rtsne)
```

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

```
custc <-
read.csv("C:/Users/admin/Desktop/MVA/PROJECT/TelEco_Customer_Churn.csv")
dim(custc)
```

```
## [1] 7043   21
```

```
#structure of dataset
str(custc)
```

```
## 'data.frame':    7043 obs. of  21 variables:
##  $ customerID      : Factor w/ 7043 levels "0002-ORFBO","0003-MKNFE",..:
5376 3963 2565 5536 6512 6552 1003 4771 5605 4535 ...
##  $ gender          : Factor w/ 2 levels "Female","Male": 1 2 2 2 1 1 2 1 1
2 ...
##  $ SeniorCitizen   : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ Partner         : Factor w/ 2 levels "No","Yes": 2 1 1 1 1 1 1 1 2 1
...
##  $ Dependents      : Factor w/ 2 levels "No","Yes": 1 1 1 1 1 1 2 1 1 2
...
##  $ tenure          : int  1 34 2 45 2 8 22 10 28 62 ...
##  $ PhoneService    : Factor w/ 2 levels "No","Yes": 1 2 2 1 2 2 2 1 2 2
...
##  $ MultipleLines   : Factor w/ 3 levels "No","No phone service",..: 2 1 1
2 1 3 3 2 3 1 ...
##  $ InternetService : Factor w/ 3 levels "DSL","Fiber optic",..: 1 1 1 1 2
2 2 1 2 1 ...
##  $ OnlineSecurity  : Factor w/ 3 levels "No","No internet service",..: 1 3
3 3 1 1 1 3 1 3 ...
##  $ OnlineBackup    : Factor w/ 3 levels "No","No internet service",..: 3 1
3 1 1 1 3 1 1 3 ...
##  $ DeviceProtection: Factor w/ 3 levels "No","No internet service",..: 1 3
1 3 1 3 1 1 3 1 ...
##  $ TechSupport     : Factor w/ 3 levels "No","No internet service",..: 1 1
1 3 1 1 1 1 3 1 ...
##  $ StreamingTV     : Factor w/ 3 levels "No","No internet service",..: 1 1
1 1 1 3 3 1 3 1 ...
##  $ StreamingMovies : Factor w/ 3 levels "No","No internet service",..: 1 1
1 1 1 3 1 1 3 1 ...
##  $ Contract        : Factor w/ 3 levels "Month-to-month",..: 1 2 1 2 1 1 1
1 1 2 ...
##  $ PaperlessBilling: Factor w/ 2 levels "No","Yes": 2 1 2 1 2 2 2 1 2 1
...
##  $ PaymentMethod   : Factor w/ 4 levels "Bank transfer (automatic)",..: 3
4 4 1 3 3 2 4 3 1 ...
##  $ MonthlyCharges  : num  29.9 57 53.9 42.3 70.7 ...
##  $ TotalCharges    : num  29.9 1889.5 108.2 1840.8 151.7 ...
##  $ Churn           : Factor w/ 2 levels "No","Yes": 1 1 2 1 2 2 1 1 2 1
...
```

```
sapply(custc, function(x) sum(is.na(x)))
```

```
##       customerID           gender    SeniorCitizen          Partner
##                0                0                0                0
##       Dependents           tenure     PhoneService    MultipleLines
##                0                0                0                0
```

```
##  InternetService   OnlineSecurity     OnlineBackup DeviceProtection
##                0                0                0                0
##      TechSupport      StreamingTV  StreamingMovies         Contract
##                0                0                0                0
## PaperlessBilling    PaymentMethod   MonthlyCharges     TotalCharges
##                0                0                0               11
##            Churn
##                0
```

```
#
custc <- custc[complete.cases(custc),]  ## to remove which has null values
sapply(custc, function(x) sum(is.na(x)))
```

```
##       customerID           gender    SeniorCitizen          Partner
##                0                0                0                0
##       Dependents           tenure     PhoneService    MultipleLines
##                0                0                0                0
##  InternetService   OnlineSecurity     OnlineBackup DeviceProtection
##                0                0                0                0
##      TechSupport      StreamingTV  StreamingMovies         Contract
##                0                0                0                0
## PaperlessBilling    PaymentMethod   MonthlyCharges     TotalCharges
##                0                0                0                0
##            Churn
##                0
```

```
dim(custc)
```

```
## [1] 7032   21
```

Hierarchical clustering - Also popularly called as unsupervised clustering is a method in which we draw references from datasets consisting of input data without labelled responses. Generally, it is used as a process to find meaningful structure, explanatory underlying processes, generative features, and groupings inherent in a set of examples.

Here, we are taking samples from our dataset and performing an unsupervised clustering technique where the algorithm determines the number of clusters to be formed.

```
# Hirerarchical cluster analysis, Nearest-neighbor

#1:20 rows for clustering
quant_var_df1<-
data.frame(custc$tenure,custc$MonthlyCharges,custc$TotalCharges)
quant_var_df5 <- quant_var_df1[c(1:20),]
quant_var_df5
```

```
##    custc.tenure custc.MonthlyCharges custc.TotalCharges
## 1             1                29.85              29.85
## 2            34                56.95            1889.50
```

```
## 3              2                53.85             108.15
## 4             45                42.30            1840.75
## 5              2                70.70             151.65
## 6              8                99.65             820.50
## 7             22                89.10            1949.40
## 8             10                29.75             301.90
## 9             28               104.80            3046.05
## 10            62                56.15            3487.95
## 11            13                49.95             587.45
## 12            16                18.95             326.80
## 13            58               100.35            5681.10
## 14            49               103.70            5036.30
## 15            25               105.50            2686.05
## 16            69               113.25            7895.15
## 17            52                20.65            1022.95
## 18            71               106.70            7382.25
## 19            10                55.20             528.35
## 20            21                90.05            1862.90
```

```
attach(quant_var_df5)
```

Scaling – Scaling is the method to scale down various dimensions (variables) to a same plane so as to perform various techniques on data for analysis.

```
# Standardizing the data with scale()
matstd.custc1<- scale(quant_var_df5[,])
matstd.custc <- matstd.custc1[c(1:20),]
matstd.custc
```

```
##    custc.tenure custc.MonthlyCharges custc.TotalCharges
## 1    -1.2324299          -1.24359683         -0.9515272
## 2     0.1748430          -0.40148104         -0.1828111
## 3    -1.1897852          -0.49781163         -0.9191607
## 4     0.6439339          -0.85672076         -0.2029627
## 5    -1.1897852           0.02579174         -0.9011792
## 6    -0.9339174           0.92539514         -0.6246994
## 7    -0.3368926           0.59756040         -0.1580505
## 8    -0.8486282          -1.24670427         -0.8390710
## 9    -0.0810248           1.08542822          0.2952673
## 10    1.3688927          -0.42634054          0.4779338
## 11   -0.7206943          -0.61900172         -0.7210343
## 12   -0.5927604          -1.58230761         -0.8287782
## 13    1.1983142           0.94714721          1.3845075
## 14    0.8145125           1.05124640          1.1179690
## 15   -0.2089587           1.10718029          0.1464556
## 16    1.6674051           1.34800676          2.2997205
## 17    0.9424464          -1.52948116         -0.5410134
## 18    1.7526944           1.14446955          2.0877050
```

```
## 19   -0.8486282          -0.45586121         -0.7454643
## 20   -0.3795372           0.62708106         -0.1938067

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

##             1          2          3          4          5          6
7
## 2  1.81121517
## 3  0.74770431 1.55360885
## 4  2.05688190 0.65398441 2.00107005
## 5  1.27110221 1.60025780 0.52391203 2.15147873
## 6  2.21369652 1.78471716 1.47570094 2.41731544 0.97529264
## 7  2.19578056 1.12275117 1.58321017 1.75469983 1.26751158 0.82563641
## 8  0.39994974 1.48073380 0.82682688 1.66867091 1.31889790 2.18431804
2.03149561
## 9  2.88176821 1.58269595 2.28272502 2.13207435 1.94516416 1.26466408
0.71342862
## 10 3.07865974 1.36490165 2.91612994 1.08370308 2.94163376 2.88893127
2.08867245
## 11 0.83971384 1.06723311 0.52343806 1.47889106 0.81746995 1.56201991
1.39437367
## 12 0.73414533 1.54946436 1.24126517 1.56445971 1.71687627 2.53901741
2.29503126
## 13 4.02056390 2.30711485 3.61908989 2.46603702 3.43165826 2.92981176
2.20421319
## 14 3.70662829 2.05222394 3.25064418 2.32686407 3.02419021 2.47178863
1.77757996
## 15 2.78912397 1.59115659 2.16184128 2.16942995 1.79693089 1.07391387
0.60729191
## 16 5.06912851 3.38399356 4.68313764 3.48880206 4.48971585 3.93671326
3.25899022
## 17 2.23166712 1.41064104 2.39869736 0.80993489 2.66364504 3.09098078
2.51150843
## 18 4.88372406 3.16777870 4.51625013 3.23747913 4.34085887 3.82400505
3.11591100
## 19 0.90016282 1.16920016 0.38512130 1.63790660 0.61043006 1.38914633
1.31020049
## 20 2.19111892 1.16850260 1.56461620 1.80256620 1.23224457 0.76288745
0.06299633
##             8          9         10         11         12         13
14
## 2
## 3
## 4
## 5
## 6
## 7
## 8
```

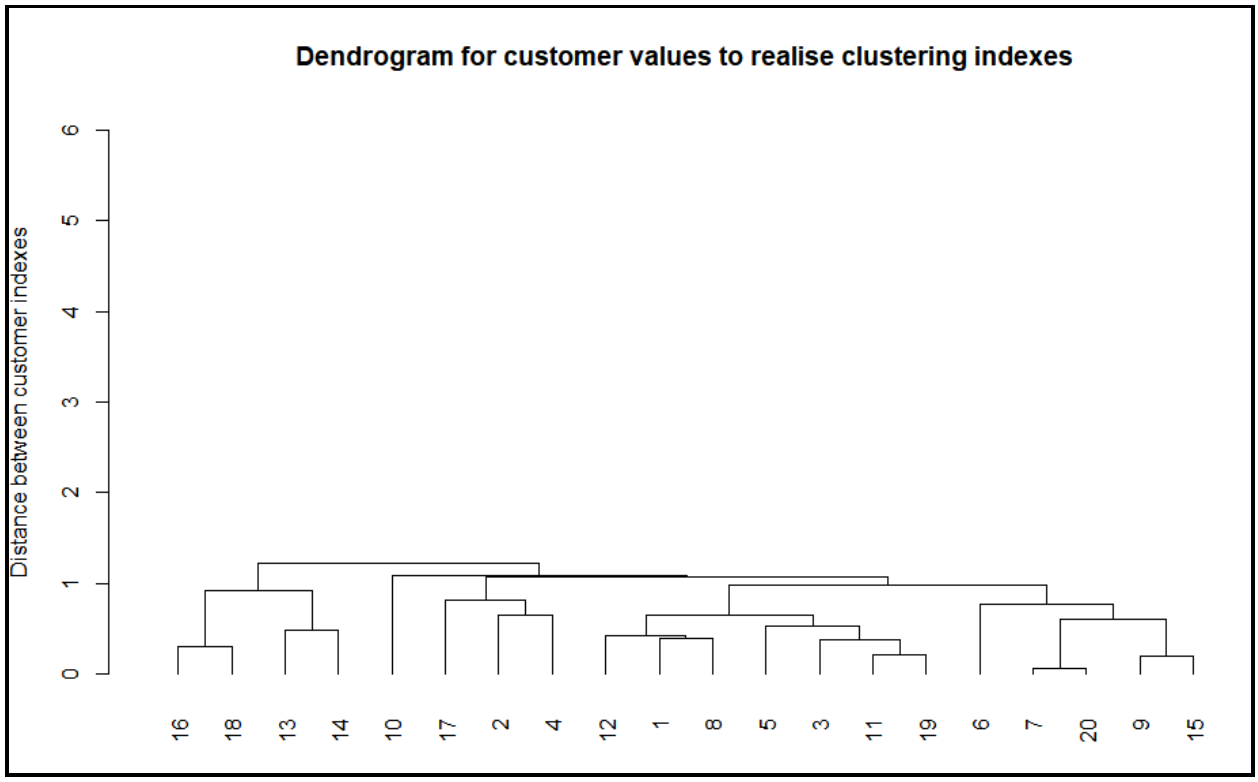
```
## 9  2.70458506
## 10 2.70645472 2.10263468
## 11 0.65139099 2.08497664 2.41681953
## 12 0.42214201 2.93975613 2.62523121 0.97771880
## 13 3.73460285 1.68590446 1.65452152 3.25095712 3.80850668
## 14 3.44624722 1.21654984 1.70301079 2.92036334 3.56452640 0.47873072
## 15 2.63081968 0.19744664 2.22512449 1.99852856 2.88647359 1.88116984
1.41225362
## 16 4.78695737 2.67278639 2.56053213 4.32401443 4.84588427 1.10378874
1.48728915
## 17 1.83759331 2.92995049 1.56109869 1.90457796 1.56283684 3.14750490
3.07062611
## 18 4.58809431 2.56492590 2.28169051 4.13720913 4.63058811 0.91692987
1.35250278
## 19 0.79636359 2.01194193 2.53277973 0.20875519 1.15814120 3.27034940
2.91715511
## 20 2.03653722 0.73374734 2.14893916 1.39537789 2.30869038 2.25458041
1.82384935
##            15         16         17         18         19
## 2
## 3
## 4
## 5
## 6
## 7
## 8
## 9
## 10
## 11
## 12
## 13
## 14
## 15
## 16 2.86623242
## 17 2.95809582 4.10795218
## 18 2.76005848 0.30602646 3.83623189
## 19 1.90992062 4.34252385 2.09819161 4.16590759
## 20 0.61267503 3.30565945 2.55322420 3.16534330 1.30274188
```

```
# Invoking hclust command (cluster analysis by single linkage method)
cluscustc.nn <- hclust(dist.custc, method = "single")
cluscustc.nn
```

```
##
## Call:
## hclust(d = dist.custc, method = "single")
##
## Cluster method   : single
## Distance         : euclidean
## Number of objects: 20
```

```r
#Plotting

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




```r
dev.new()
par(mar=c(5, 4, 4, 7) +0.1)
plot(as.dendrogram(cluscustc.nn), xlab= "Distance between customer indexess",
xlim=c(6,0),
     horiz = TRUE,main="Dendrogram for customer values to realise clustering
indexes")



#??agnes
(agn.quant_var_df5 <- agnes(quant_var_df5, metric="euclidean", stand=TRUE,
method = "single"))

## Call:     agnes(x = quant_var_df5, metric = "euclidean", stand = TRUE,
method = "single")
## Agglomerative coefficient:  0.6583063
```

```
## Order of objects:
##  [1] 1  8  12 3  11 19 5  6  7  20 9  15 2  4  17 10 13 14 16 18
## Height (summary):
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
## 0.07541 0.46476 0.71081 0.73062 1.02858 1.48492
##
## Available components:
## [1] "order"     "height"    "ac"        "merge"     "diss"      "call"
## [7] "method"    "order.lab" "data"
```

```
#View(agn.quant_var_df5)

#  Description of cluster merging
agn.quant_var_df5$merge
```

```
##       [,1] [,2]
##  [1,]   -7  -20
##  [2,]  -11  -19
##  [3,]   -9  -15
##  [4,]  -16  -18
##  [5,]   -3    2
##  [6,]   -1   -8
##  [7,]    6  -12
##  [8,]  -13  -14
##  [9,]    5   -5
## [10,]    1    3
## [11,]    7    9
## [12,]   -2   -4
## [13,]   -6   10
## [14,]   12  -17
## [15,]   11   13
## [16,]    8    4
## [17,]   15   14
## [18,]   17  -10
## [19,]   18   16
```

```
#Dendogram
plot(as.dendrogram(agn.quant_var_df5), xlab= "Distance between customer
indexes",xlim=c(8,0),
     horiz = TRUE,main="")

#Interactive Plots
#plot(agn.quant_var_df5,ask=TRUE)
#plot(agn.quant_var_df5, which.plots=2)

#100:120 rows for clustering

quant_var_df1<-
data.frame(custc$tenure,custc$MonthlyCharges,custc$TotalCharges)
```

```
quant_var_df6 <- quant_var_df1[c(100:120),]
quant_var_df6
```

```
##     custc.tenure custc.MonthlyCharges custc.TotalCharges
## 100           25                98.50            2514.50
## 101            1                20.20              20.20
## 102            1                19.45              19.45
## 103           38                95.00            3605.60
## 104           66                45.55            3027.25
## 105           68               110.00            7611.85
## 106            5                24.30             100.20
## 107           72               104.15            7303.05
## 108           32                30.15             927.65
## 109           43                94.35            3921.30
## 110           72                19.40            1363.25
## 111           55                96.75            5238.90
## 112           52                57.95            3042.25
## 113           43                91.65            3954.10
## 114           37                76.50            2868.15
## 115           64                54.60            3423.50
## 116            3                89.85             248.40
## 117           36                31.05            1126.35
## 118           10               100.25            1064.65
## 119           41                20.65             835.15
## 120           27                85.20            2151.60
```

```
attach(quant_var_df6)
```

```
## The following objects are masked from quant_var_df5:
##
##     custc.MonthlyCharges, custc.tenure, custc.TotalCharges
```

```
matstd.custc1<- scale(quant_var_df6[,])
matstd.custc <- matstd.custc1[,]
matstd.custc
```

```
##     custc.tenure custc.MonthlyCharges custc.TotalCharges
## 100  -0.53099874            0.9818807         -0.0337601
## 101  -1.53710161           -1.3147145         -1.1652597
## 102  -1.53710161           -1.3367125         -1.1656000
## 103   0.01397365            0.8792232          0.4612001
## 104   1.18776033           -0.5711808          0.1988408
## 105   1.27160224            1.3191839          2.2785719
## 106  -1.36941780           -1.1944585         -1.1289690
## 107   1.43928605            1.1475992          2.1384897
## 108  -0.23755207           -1.0228738         -0.7536094
## 109   0.22357842            0.8601582          0.6044124
## 110   1.43928605           -1.3381791         -0.5560064
## 111   0.72662985            0.9305519          1.2021207
## 112   0.60086699           -0.2074799          0.2056453
## 113   0.22357842            0.7809653          0.6192916
```

```
## 114  -0.02794730            0.3366049          0.1266676
## 115   1.10391843           -0.3057378          0.3785933
## 116  -1.45325970            0.7281700         -1.0617404
## 117  -0.06986825           -0.9964762         -0.6634723
## 118  -1.15981303            1.0332094         -0.6914616
## 119   0.13973651           -1.3015157         -0.7955706
## 120  -0.44715683            0.5917821         -0.1983839
```

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

```
##            100        101        102        103        104        105
## 101 2.75079693
## 102 2.76932824 0.02200067
## 103 0.74331627 3.14079106 3.15637195
## 104 2.32814001 3.13663597 3.14207097 1.88421847
## 105 2.95127480 5.16587014 5.17734709 2.25344964 2.81172360
## 106 2.57660270 0.20951460 0.22292573 2.95678297 2.94800170 4.99046292
## 107 2.93737192 5.08297475 5.09388816 2.21739518 2.60378718 0.27781684
## 108 2.15019411 1.39407888 1.39894958 2.27090301 1.77276719 4.11787626
## 109 0.99572445 3.31086165 3.32553438 0.25457296 1.77281314 2.02777427
## 110 3.08827469 3.03819415 3.03817214 2.82543752 1.10514516 3.88902686
## 111 1.76399143 3.97117007 3.98385109 1.02930894 1.86397778 1.26758699
## 112 1.65922078 2.77060621 2.77963840 1.26121989 0.69048420 2.66037831
## 113 1.01795394 3.26749005 3.28182743 0.28032427 1.71310380 2.03500474
## 114 0.83377373 2.58330770 2.59759377 0.63883014 1.51895652 2.69907068
## 115 2.12154445 3.22125799 3.22837863 1.61212159 0.33136146 2.50567496
## 116 1.40416450 2.04722315 2.06919211 2.12012703 3.20193134 4.35107531
## 117 2.12675216 1.58298450 1.58766074 2.18864472 1.58306342 3.97711567
## 118 0.91137987 2.42478414 2.44615698 1.65230622 2.97956491 3.84898171
## 119 2.49882770 1.71715763 1.71754086 2.52010134 1.61882580 4.19518037
## 120 0.43163334 2.39949116 2.41714255 0.85458456 2.04529324 3.10137972
##            106        107        108        109        110        111
## 101
## 102
## 103
## 104
## 105
## 106
## 107 4.90411440
## 108 1.20476394 3.98584704
## 109 3.12469167 1.97837317 2.36699690
## 110 2.87014965 3.66597910 1.71762930 2.76716526
## 111 3.78722387 1.19656806 2.92752448 0.78439230 2.95736880
## 112 2.57630440 2.50500963 1.51260817 1.20050531 1.60048131 1.51785903
## 113 3.08161505 1.97998188 2.31329508 0.08057861 2.71109630 0.78430015
## 114 2.39171939 2.61876182 1.63309654 0.75207269 2.32888790 1.44179027
## 115 3.02984501 2.30692329 1.89623582 1.47827815 1.43243954 1.53263054
## 116 1.92562966 4.33407595 2.15384406 2.36754561 3.59059691 3.14927679
```

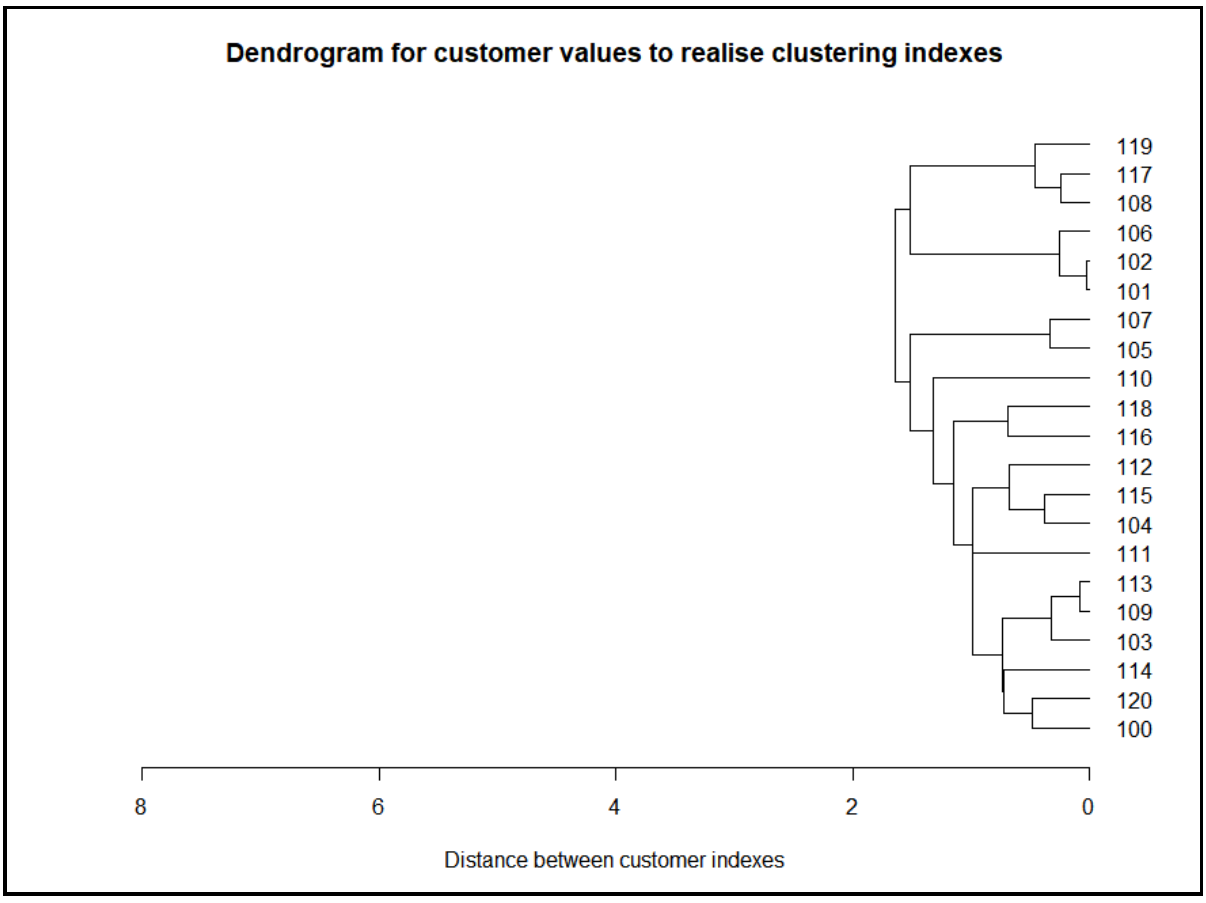
```
## 117 1.39452973 3.83739458 0.19219624 2.26731867 1.55108236 2.79790710
## 118 2.27987972 3.84408961 2.25430834 1.90342007 3.52095832 2.67485687
## 119 1.54924577 4.03679357 0.47090197 2.57678400 1.32195469 3.05243185
## 120 2.21522226 3.05427201 1.72026843 1.07999696 2.72237228 1.85848150
##            112        113        114        115        116        117
## 101
## 102
## 103
## 104
## 105
## 106
## 107
## 108
## 109
## 110
## 111
## 112
## 113 1.13599022
## 114 0.83526832 0.70950667
## 115 0.54094952 1.41910457 1.32559072
## 116 2.58865675 2.37496149 1.89661601 3.11170156
## 117 1.35194912 2.21153622 1.55022054 1.71487479 2.24650702
## 118 2.33326058 1.92236127 1.56067718 2.83941496 0.56237573 2.30399366
## 119 1.55305790 2.51904572 1.88734738 1.81655752 2.59385950 0.39297987
## 120 1.37855482 1.07436926 0.58865147 1.88262563 1.33275244 1.69741523
##            118        119
## 101
## 102
## 103
## 104
## 105
## 106
## 107
## 108
## 109
## 110
## 111
## 112
## 113
## 114
## 115
## 116
## 117
## 118
## 119 2.67406226
## 120 0.97255461 2.07018170

# Invoking hclust command (cluster analysis by single linkage method)
cluscustc.nn <- hclust(dist.custc, method = "single")
cluscustc.nn
```

```
##
## Call:
## hclust(d = dist.custc, method = "single")
##
## Cluster method   : single
## Distance         : euclidean
## Number of objects: 21
```

```
#Plotting

# Create extra margin room in the dendrogram, on the bottom (Customer labels)
par(mar=c(8, 4, 4, 2) + 0.1)
# Object "clusquant_var_df.nn" is converted into a object of class
"dendrogram"
# in order to allow better flexibility in the (vertical) dendrogram plotting.
plot(as.dendrogram(cluscustc.nn), xlab= "Distance between customer
indexes",xlim=c(0,6),
     horiz = TRUE,main="Dendrogram for customer values to realise clustering
indexes")
```




```
dev.new()
par(mar=c(5, 4, 4, 7) +0.1)
plot(as.dendrogram(cluscustc.nn), xlab= "Distance between customer indexes",
xlim=c(6,0),
```

```
     horiz = TRUE,main="Dendrogram for customer values to realise clustering
indexes")

#??agnes
(agn.quant_var_df6 <- agnes(quant_var_df6, metric="euclidean", stand=TRUE,
method = "single"))

## Call:     agnes(x = quant_var_df6, metric = "euclidean", stand = TRUE,
method = "single")
## Agglomerative coefficient:  0.7317389
## Order of objects:
##  [1] 100 120 114 103 109 113 111 104 115 112 116 118 110 105 107 101 102
106 108
## [20] 117 119
## Height (summary):
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
## 0.02409 0.33083 0.68182 0.72672 1.03380 1.63972
##
## Available components:
## [1] "order"     "height"    "ac"        "merge"     "diss"      "call"
## [7] "method"    "order.lab" "data"

#View(agn.quant_var_df6)

#  Description of cluster merging
agn.quant_var_df6$merge

##       [,1] [,2]
##  [1,]   -2   -3
##  [2,]  -10  -14
##  [3,]   -9  -18
##  [4,]    1   -7
##  [5,]   -4    2
##  [6,]   -6   -8
##  [7,]   -5  -16
##  [8,]    3  -20
##  [9,]   -1  -21
## [10,]    7  -13
## [11,]  -17  -19
## [12,]    9  -15
## [13,]   12    5
## [14,]   13  -12
## [15,]   14   10
## [16,]   15   11
## [17,]   16  -11
## [18,]    4    8
## [19,]   17    6
## [20,]   19   18

#Dendogram
plot(as.dendrogram(agn.quant_var_df6), xlab= "Distance between customer
```

```
indexes",xlim=c(8,0),
     horiz = TRUE,main="Dendrogram for customer values to realise clustering
indexes")

#Interactive Plots
#plot(agn.quant_var_df6,ask=TRUE)
#plot(agn.quant_var_df6, which.plots=2)

#1000:1020 rows for sampling

quant_var_df1<-
data.frame(custc$tenure,custc$MonthlyCharges,custc$TotalCharges)
quant_var_df7 <- quant_var_df1[c(1000:1020),]
quant_var_df7

##      custc.tenure custc.MonthlyCharges custc.TotalCharges
## 1000            4                69.55              284.90
## 1001           37                19.85              784.25
## 1002           21                20.00              417.70
## 1003           53                95.85             5016.25
## 1004           18                90.10             1612.75
## 1005            2                68.95              119.75
## 1006           32                99.55             3204.65
## 1007           23                20.75              485.20
## 1008            3                50.15              160.85
## 1009           71                58.65             4145.25
## 1010            9                95.90              827.45
## 1011            1                49.50               49.50
## 1012           18                57.45              990.85
## 1013           12                53.65              696.35
## 1014           71                80.10             5585.40
## 1015           64                24.40             1601.20
## 1016            4                40.05              162.45
## 1017           23                19.50              470.20
## 1018           39                51.05             2066.00
## 1019           28                54.35             1426.45
## 1020            5                84.70              392.50

attach(quant_var_df7)

## The following objects are masked from quant_var_df6:
##
##     custc.MonthlyCharges, custc.tenure, custc.TotalCharges

## The following objects are masked from quant_var_df5:
##
##     custc.MonthlyCharges, custc.tenure, custc.TotalCharges

matstd.custc1<- scale(quant_var_df7[,])
matstd.custc <- matstd.custc1[,]
matstd.custc
```

```
##      custc.tenure custc.MonthlyCharges custc.TotalCharges
## 1000   -0.9505770          0.453522014        -0.70604200
## 1001    0.5004139         -1.391861713        -0.40405548
## 1002   -0.2030969         -1.386292145        -0.62572997
## 1003    1.2039246          1.430053041         2.15528557
## 1004   -0.3350051          1.216552911         0.09698754
## 1005   -1.0385159          0.431243740        -0.80591798
## 1006    0.2805668          1.567435733         1.05970375
## 1007   -0.1151580         -1.358444302        -0.58490872
## 1008   -0.9945464         -0.266808857        -0.78106238
## 1009    1.9953742          0.048800030         1.62854028
## 1010   -0.7307299          1.431909564        -0.37792988
## 1011   -1.0824853         -0.290943655        -0.84840232
## 1012   -0.3350051          0.004243481        -0.27911222
## 1013   -0.5988216         -0.136852257        -0.45721381
## 1014    1.9953742          0.845248339         2.49948428
## 1015    1.6875883         -1.222918133         0.09000257
## 1016   -0.9505770         -0.641826476        -0.78009477
## 1017   -0.1151580         -1.404857373        -0.59398011
## 1018    0.5883527         -0.233391446         0.37109466
## 1019    0.1046891         -0.110860937        -0.01567910
## 1020   -0.9066076          1.016048442        -0.64096991
```

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

```
##           1000       1001       1002       1003       1004       1005
## 1001 2.36685690
## 1002 1.98748403 0.73762996
## 1003 3.71250396 3.87406206 4.20066151
## 1004 1.26728109 2.78438429 2.70453744 2.57885204
## 1005 0.13492507 2.41940318 2.00843952 3.84641207 1.38811968
## 1006 2.42371139 3.30883140 3.43498555 1.43936209 1.19535351 2.55175542
## 1007 1.99895476 0.64245886 0.10087178 4.12605880 2.67281236 2.02593635
## 1008 0.72556043 1.90860800 1.37976934 4.04162772 1.84562416 0.69987752
## 1009 3.78056913 2.90548759 3.46041890 1.67681631 3.02323807 3.90862370
## 1010 1.05509834 3.08059764 2.87785726 3.18748681 0.65461680 1.13103408
## 1011 0.76934737 1.97864460 1.42221493 4.14869146 1.93003484 0.72477090
## 1012 0.87352676 1.63176145 1.43914314 3.21364750 1.26930894 0.97712890
## 1013 0.73088240 1.66918941 1.32139920 3.53980781 1.48608339 0.79853532
## 1014 4.37121000 3.95854909 4.42493074 1.04252661 3.36756967 4.50572668
## 1015 3.22553167 1.29692648 2.02821457 3.39670140 3.16890387 3.31218490
## 1016 1.09784886 1.67610650 1.06620083 4.18939372 2.14517507 1.07697714
## 1017 2.04060205 0.64433611 0.09532031 4.16355111 2.71984576 2.06610132
## 1018 2.00009456 1.39665470 1.71733276 2.51581270 1.74070763 2.11513595
## 1019 1.38156274 1.39585003 1.44693435 2.88024691 1.40287287 1.49173474
## 1020 0.56798210 2.79890500 2.50327783 3.52771750 0.95473170 0.62177504
##           1006       1007       1008       1009       1010       1011
## 1001
```

```
## 1002
## 1003
## 1004
## 1005
## 1006
## 1007 3.37966304
## 1008 2.89461343 1.41543923
## 1009 2.36016820 3.36660651 3.85297540
## 1010 1.76291770 2.86493345 1.76571777 3.65657739
## 1011 2.99204840 1.46448235 0.11335959 3.96533599 1.82024706
## 1012 2.14823811 1.41377588 0.87201954 3.01194143 1.48478735 0.98486205
## 1013 2.44519129 1.32004683 0.52760281 3.33386998 1.57629292 0.64086123
## 1014 2.35267659 4.33867350 4.57583687 1.18020056 4.00690995 4.68748291
## 1015 3.27201842 1.92970638 2.97770812 2.01967759 3.62150986 3.06960708
## 1016 3.12752508 1.11783881 0.37758769 3.86744313 2.12378212 0.38103079
## 1017 3.42429478 0.04729126 1.45033727 3.39220622 2.91081657 1.49708266
## 1018 1.95240703 1.63542464 1.95809917 1.90801231 2.25260832 2.06934405
## 1019 2.00101498 1.38881941 1.34849919 2.51070493 1.79144951 1.46124630
## 1020 2.14608989 2.50354770 1.29347671 3.80890331 0.52255477 1.33498671
##           1012       1013       1014       1015       1016       1017
## 1001
## 1002
## 1003
## 1004
## 1005
## 1006
## 1007
## 1008
## 1009
## 1010
## 1011
## 1012
## 1013 0.34817717
## 1014 3.72270809 4.05418763
## 1015 2.39437995 2.58972105 3.19024244
## 1016 1.02338588 0.69496972 4.65248953 2.83807204
## 1017 1.46049298 1.36399139 4.36884927 1.93670553 1.14663845
## 1018 1.15404899 1.45079211 2.77005697 1.50548815 1.96478015 1.67291037
## 1019 0.52533517 0.83099688 3.28859879 1.93737183 1.40706868 1.43429132
## 1020 1.21713584 1.20734358 4.27938355 3.50387560 1.66428311 2.54742722
##           1018       1019
## 1001
## 1002
## 1003
## 1004
## 1005
## 1006
## 1007
## 1008
## 1009
```

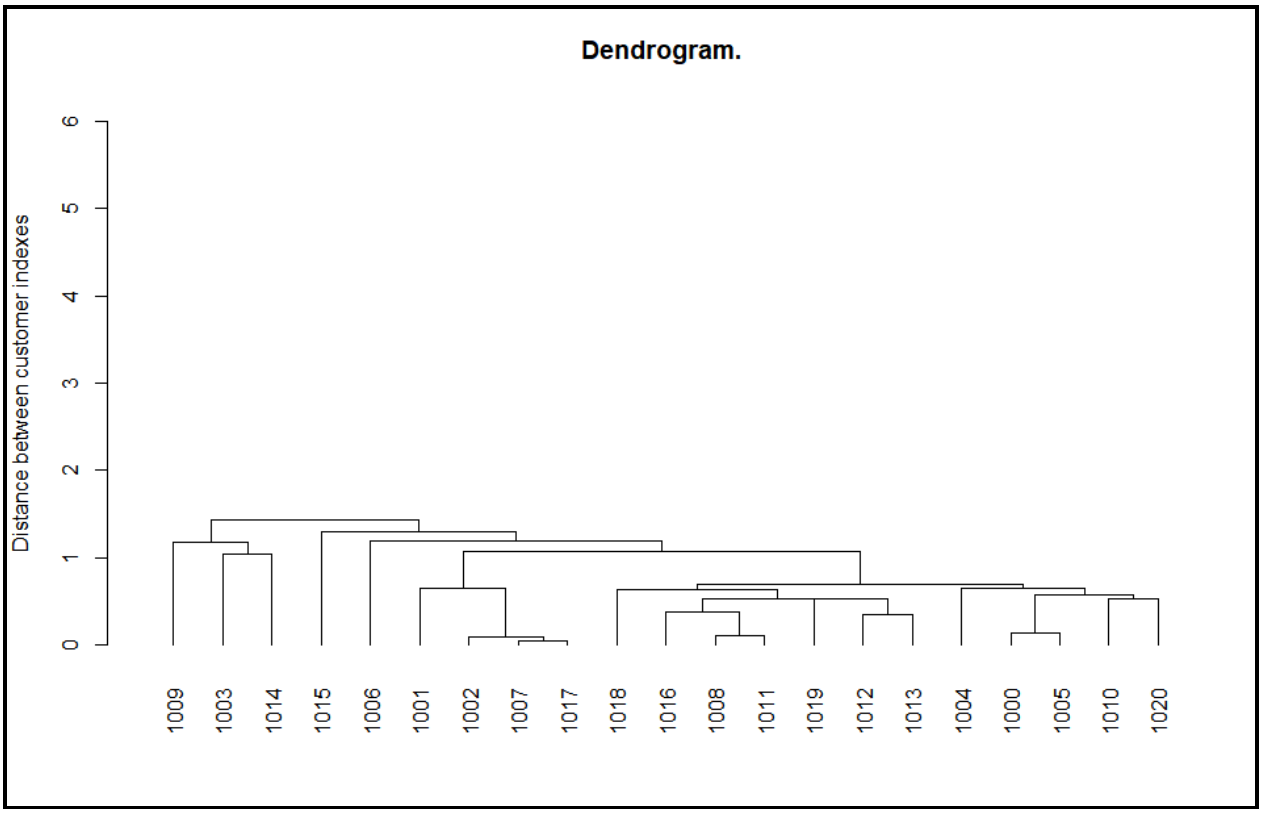
```
## 1010
## 1011
## 1012
## 1013
## 1014
## 1015
## 1016
## 1017
## 1018
## 1019 0.63129880
## 1020 2.19551386 1.63818019
```

```
# Invoking hclust command (cluster analysis by single linkage method)
cluscustc.nn <- hclust(dist.custc, method = "single")
cluscustc.nn
```

```
##
## Call:
## hclust(d = dist.custc, method = "single")
##
## Cluster method   : single
## Distance         : euclidean
## Number of objects: 21
```

```
#Plotting

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




```
dev.new()
par(mar=c(5, 4, 4, 7) +0.1)
plot(as.dendrogram(cluscustc.nn), xlab= "Distance between customer indexes",
xlim=c(6,0),
     horiz = TRUE,main="Dendrogram for customer values to realise clustering
indexes")



#??agnes
(agn.quant_var_df7 <- agnes(quant_var_df7, metric="euclidean", stand=TRUE,
method = "single"))

## Call:     agnes(x = quant_var_df7, metric = "euclidean", stand = TRUE,
method = "single")
## Agglomerative coefficient:  0.6421378
## Order of objects:
##  [1] 1000 1005 1010 1020 1004 1008 1011 1016 1012 1013 1019 1018 1001 1002
1007
## [16] 1017 1006 1015 1003 1014 1009
## Height (summary):
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
## 0.05895 0.46302 0.75865 0.83546 1.31923 1.86952
##
## Available components:
```

```
## [1] "order"     "height"    "ac"        "merge"     "diss"      "call"
## [7] "method"    "order.lab" "data"
```

```
#View(agn.quant_var_df7)

#  Description of cluster merging
agn.quant_var_df7$merge
```

```
##       [,1] [,2]
##  [1,]   -8  -18
##  [2,]   -3    1
##  [3,]   -9  -12
##  [4,]   -1   -6
##  [5,]  -13  -14
##  [6,]    3  -17
##  [7,]  -11  -21
##  [8,]    5  -20
##  [9,]    6    8
## [10,]    4    7
## [11,]    9  -19
## [12,]   -2    2
## [13,]   10   -5
## [14,]   13   11
## [15,]   -4  -15
## [16,]   14   12
## [17,]   15  -10
## [18,]   16   -7
## [19,]   18  -16
## [20,]   19   17
```

```
#Dendogram
plot(as.dendrogram(agn.quant_var_df7), xlab= "Distance between customer
indexes",xlim=c(8,0),
     horiz = TRUE,main="Dendrogram for customer values to realise clustering
indexes
     .0")

#Interactive Plots
#plot(agn.quant_var_df7,ask=TRUE)
#plot(agn.quant_var_df7, which.plots=2)

#6000:6020 rows for clustering

quant_var_df1<-
data.frame(custc$tenure,custc$MonthlyCharges,custc$TotalCharges)
quant_var_df8 <- quant_var_df1[c(6000:6020),]
quant_var_df8
```

```
##      custc.tenure custc.MonthlyCharges custc.TotalCharges
## 6000           15                19.50             239.75
## 6001           53               103.85            5485.50
```

```
## 6002           24                24.20             609.05
## 6003           37                19.35             683.75
## 6004            5                83.60             404.20
## 6005           50               100.65            5189.75
## 6006           54                94.10            5060.90
## 6007            3                74.55             233.65
## 6008           68               108.45            7176.55
## 6009            5                56.15             291.45
## 6010           33                20.35             689.75
## 6011           41                80.55            3263.90
## 6012           34                61.25            1993.20
## 6013           13                20.45             254.50
## 6014           20                18.90             347.65
## 6015           51                19.60             967.90
## 6016            3                91.50             242.95
## 6017           41                45.20            1841.90
## 6018           13                19.45             232.10
## 6019           35                25.45             809.25
## 6020           12                80.85             866.45
```

```
attach(quant_var_df8)
```

```
## The following objects are masked from quant_var_df7:
##
##     custc.MonthlyCharges, custc.tenure, custc.TotalCharges

## The following objects are masked from quant_var_df6:
##
##     custc.MonthlyCharges, custc.tenure, custc.TotalCharges

## The following objects are masked from quant_var_df5:
##
##     custc.MonthlyCharges, custc.tenure, custc.TotalCharges
```

```
matstd.custc1<- scale(quant_var_df8[,])
matstd.custc <- matstd.custc1[,]
matstd.custc
```

```
##      custc.tenure custc.MonthlyCharges custc.TotalCharges
## 6000   -0.7167268          -1.05575980        -0.70832498
## 6001    1.2220799           1.40995332         1.74163431
## 6002   -0.2575357          -0.91836973        -0.53584822
## 6003    0.4057403          -1.06014459        -0.50096055
## 6004   -1.2269391           0.81800678        -0.63152074
## 6005    1.0690163           1.31641115         1.60350812
## 6006    1.2731012           1.12494203         1.54333040
## 6007   -1.3289816           0.55345783        -0.71117390
## 6008    1.9873984           1.54442019         2.53141721
## 6009   -1.2269391           0.01559036        -0.68417916
## 6010    0.2016553          -1.03091266        -0.49815833
## 6011    0.6098252           0.72884940         0.70406493
```

```
## 6012    0.2526766           0.16467319         0.11060103
## 6013   -0.8187693          -1.02798946        -0.70143618
## 6014   -0.4616207          -1.07329895        -0.65793169
## 6015    1.1200375          -1.05283660        -0.36825199
## 6016   -1.3289816           1.04893901        -0.70683046
## 6017    0.6098252          -0.30449925         0.03993833
## 6018   -0.8187693          -1.05722139        -0.71189781
## 6019    0.3036978          -0.88182982        -0.44234741
## 6020   -0.8697905           0.73761898        -0.41563290
```

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

```
##            6000       6001       6002       6003       6004       6005
## 6001 3.98007705
## 6002 0.50939248 3.57732826
## 6003 1.14146903 3.43467988 0.67915558
## 6004 1.94350628 3.46120787 1.99095443 2.49201534
## 6005 3.76306332 0.22640114 3.36613002 3.24295245 3.24271059
## 6006 3.71279245 0.35094023 3.29256071 3.11545245 3.32782034 0.28623843
## 6007 1.72175649 3.64112174 1.82893731 2.37847898 0.29452238 3.41909991
## 6008 4.95672861 1.10794887 4.52914790 4.29892022 4.56769291 1.32530342
## 6009 1.18688280 3.71840200 1.35426232 1.96377612 0.80414241 3.49242879
## 6010 0.94245062 3.46637623 0.47428143 0.20618684 2.34033560 3.26791155
## 6011 2.63427702 1.38394677 2.23674551 2.16661869 2.27276070 1.16836779
## 6012 1.76063608 2.26952309 1.36058635 1.37753935 1.77956513 2.05467330
## 6013 0.10597788 4.00964138 0.59533095 1.24122843 1.89187551 3.79113659
## 6014 0.26062666 3.84178429 0.28382757 0.88154860 2.04045271 3.62872894
## 6015 1.86798320 3.24459043 1.39423018 0.72655729 3.01291629 3.08281949
## 6016 2.19194325 3.55432496 2.24667194 2.73859013 0.26346520 3.34059864
## 6017 1.69824338 2.49198168 1.20858681 0.95143162 2.25490136 2.29846850
## 6018 0.10211545 4.03383644 0.60436439 1.24254845 1.92081852 3.81562187
## 6019 1.06876662 3.29628153 0.57014090 0.21364531 2.29522992 3.09895153
## 6020 1.82354177 3.07924621 1.76963449 2.20594954 0.42499987 2.85847904
##            6006       6007       6008       6009       6010       6011
## 6001
## 6002
## 6003
## 6004
## 6005
## 6006
## 6007 3.49001697
## 6008 1.28937892 4.74286618
## 6009 3.52741569 0.54812663 4.79680584
## 6010 3.15648251 2.21324532 4.35884681 1.78063315
## 6011 1.14069757 2.40678931 2.42941969 2.41032429 2.16995528
## 6012 2.00401997 1.82431083 3.28227247 1.68616838 1.34261560 0.89333607
## 6013 3.74833469 1.66174215 4.99431772 1.12069589 1.04047912 2.66510756
## 6014 3.56189248 1.84431216 4.79813445 1.33119541 0.68356358 2.50015393
```

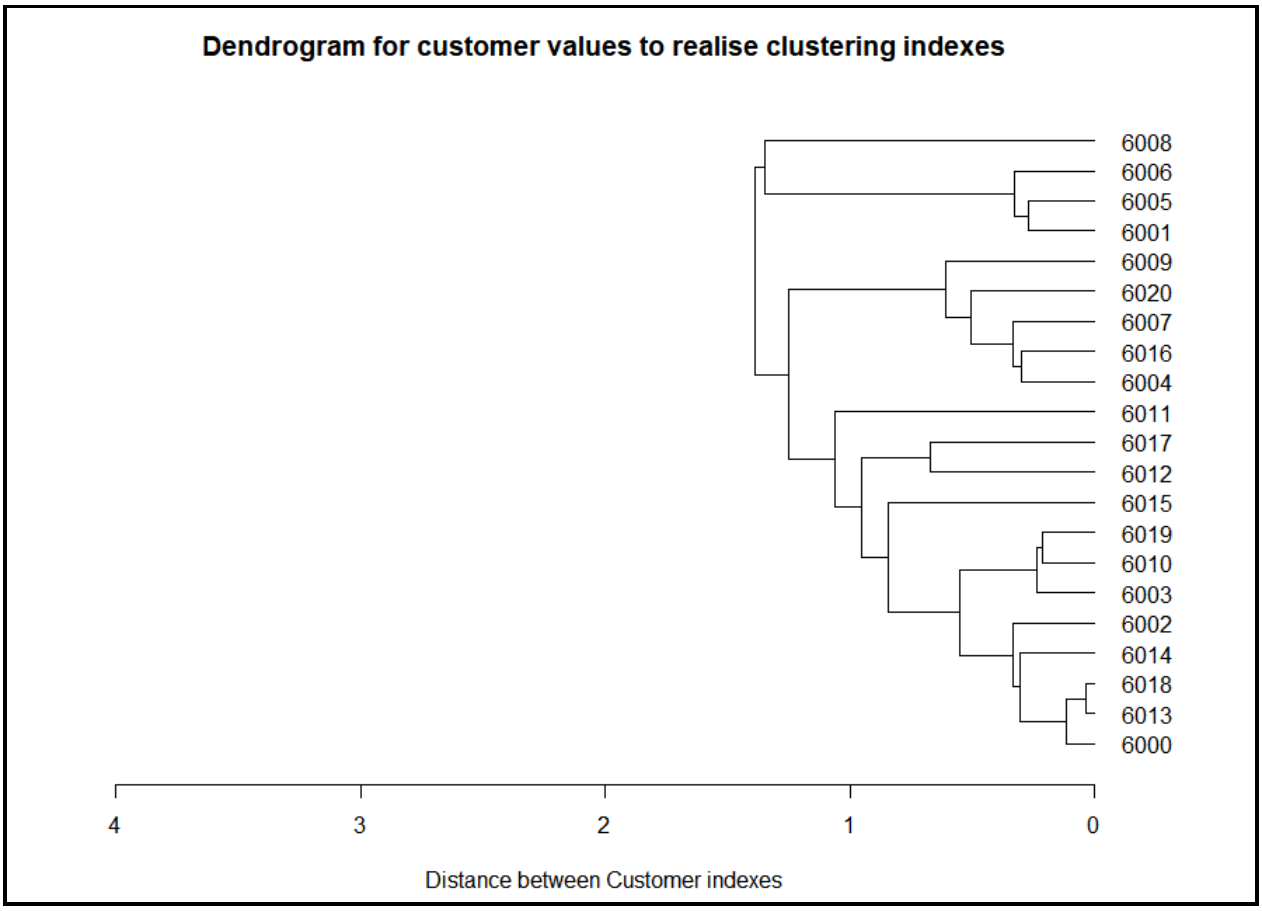
```
## 6015 2.90177454 2.94880850 3.98825015 2.59800795 0.92778342 2.14116445
## 6016 3.44090613 0.49550022 4.66155829 1.03862177 2.59078683 2.41910206
## 6017 2.17794057 2.24927357 3.39465433 2.00012664 0.99188058 1.22836215
## 6018 3.77144713 1.68955754 5.01619459 1.14817072 1.04290129 2.68995938
## 6019 2.98492756 2.19042468 4.19103959 1.79072470 0.18908520 2.00056562
## 6020 2.92908536 0.57629529 4.18324765 0.84911575 2.06942281 1.85554916
##            6012       6013       6014       6015       6016       6017
## 6001
## 6002
## 6003
## 6004
## 6005
## 6006
## 6007
## 6008
## 6009
## 6010
## 6011
## 6012
## 6013 1.79717691
## 6014 1.62278714 0.36263028
## 6015 1.56969592 1.96738423 1.60809686
## 6016 1.98790411 2.13868595 2.29316376 3.24496386
## 6017 0.59386120 1.76464058 1.49200259 0.99345092 2.47960287
## 6018 1.82140795 0.03104756 0.36156045 1.96903106 2.16708420 1.78121597
## 6019 1.18470406 1.16121572 0.81783210 0.83734338 2.54233387 0.81217246
## 6020 1.36569445 1.78931827 1.87209377 2.67719845 0.62655616 1.86623099
##            6018       6019
## 6001
## 6002
## 6003
## 6004
## 6005
## 6006
## 6007
## 6008
## 6009
## 6010
## 6011
## 6012
## 6013
## 6014
## 6015
## 6016
## 6017
## 6018
## 6019 1.16762663
## 6020 1.81984286 2.00010072
```

```r
# Invoking hclust command (cluster analysis by single linkage method)
cluscustc.nn <- hclust(dist.custc, method = "single")
cluscustc.nn

##
## Call:
## hclust(d = dist.custc, method = "single")
##
## Cluster method   : single
## Distance         : euclidean
## Number of objects: 21

#Plotting

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



```r
dev.new()
par(mar=c(5, 4, 4, 7) +0.1)
plot(as.dendrogram(cluscustc.nn), xlab= "Distance between customer indexes",
xlim=c(6,0),
     horiz = TRUE,main="Dendrogram for customer values to realise clustering
indexes")


#??agnes
(agn.quant_var_df8 <- agnes(quant_var_df8, metric="euclidean", stand=TRUE,
method = "single"))
```

```
## Call:     agnes(x = quant_var_df8, metric = "euclidean", stand = TRUE,
method = "single")
## Agglomerative coefficient:  0.6906219
## Order of objects:
##  [1] 6000 6013 6018 6014 6002 6003 6010 6019 6015 6012 6017 6011 6004 6016
6007
## [16] 6020 6009 6001 6005 6006 6008
## Height (summary):
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
## 0.03505 0.29103 0.41837 0.58249 0.87256 1.38525
##
## Available components:
## [1] "order"     "height"    "ac"        "merge"     "diss"      "call"
## [7] "method"    "order.lab" "data"
```

```r
#View(agn.quant_var_df8)

#  Description of cluster merging
agn.quant_var_df8$merge
```

```
##       [,1] [,2]
##  [1,]  -14  -19
##  [2,]   -1    1
##  [3,]  -11  -20
##  [4,]   -4    3
##  [5,]   -2   -6
##  [6,]   -5  -17
##  [7,]    2  -15
##  [8,]    5   -7
##  [9,]    7   -3
## [10,]    6   -8
## [11,]   10  -21
## [12,]    9    4
## [13,]   11  -10
## [14,]  -13  -18
## [15,]   12  -16
## [16,]   15   14
## [17,]   16  -12
```

```
## [18,]   17   13
## [19,]    8   -9
## [20,]   18   19
```

```
#Dendogram
plot(as.dendrogram(agn.quant_var_df8), xlab= "Distance between Customer
indexes",xlim=c(4,0),
     horiz = TRUE,main="Dendrogram for customer values to realise clustering
indexes")

#Interactive Plots
#plot(agn.quant_var_df8,ask=TRUE)
#plot(agn.quant_var_df8, which.plots=2)
```

Non-Hierarchical clustering – Also popularly knowns as Supervised or K-means clustering is a method in which the user or we define the number of clusters required in a data to divide det points on terms of their similarities and dissimilarities.

#sample 2 – Hierarchical clustering for rows 100-120

#sample 3 – Hierarchical clustering for rows 1000-1020

#sample 4 – Hierarchical clustering for rows 6000-6020

#sample 1 – Non-Hierarchical clustering for 1-20 row with k = 4.

#sample 1 – Non-Hierarchical clustering for 100-120 row with k = 5.

```
#K-Means Clustering
#Sample 1 -Hierarchical clustering for rows 1-20
quant_var_df1<-
data.frame(custc$tenure,custc$MonthlyCharges,custc$TotalCharges)
quant_var_df5 <- quant_var_df1[c(1:20),]
quant_var_df5
```

```
##    custc.tenure custc.MonthlyCharges custc.TotalCharges
## 1             1                29.85              29.85
## 2            34                56.95            1889.50
## 3             2                53.85             108.15
## 4            45                42.30            1840.75
## 5             2                70.70             151.65
## 6             8                99.65             820.50
## 7            22                89.10            1949.40
## 8            10                29.75             301.90
## 9            28               104.80            3046.05
## 10           62                56.15            3487.95
## 11           13                49.95             587.45
```

```
## 12           16                 18.95             326.80
## 13           58                100.35            5681.10
## 14           49                103.70            5036.30
## 15           25                105.50            2686.05
## 16           69                113.25            7895.15
## 17           52                 20.65            1022.95
## 18           71                106.70            7382.25
## 19           10                 55.20             528.35
## 20           21                 90.05            1862.90
```

```
attach(quant_var_df5)
```

```
## The following objects are masked from quant_var_df8:
##
##     custc.MonthlyCharges, custc.tenure, custc.TotalCharges

## The following objects are masked from quant_var_df7:
##
##     custc.MonthlyCharges, custc.tenure, custc.TotalCharges

## The following objects are masked from quant_var_df6:
##
##     custc.MonthlyCharges, custc.tenure, custc.TotalCharges

## The following objects are masked from quant_var_df5 (pos = 6):
##
##     custc.MonthlyCharges, custc.tenure, custc.TotalCharges
```

```
# Standardizing the data with scale()
quant_var_df5[,]
```

```
##    custc.tenure custc.MonthlyCharges custc.TotalCharges
## 1             1                 29.85              29.85
## 2            34                 56.95            1889.50
## 3             2                 53.85             108.15
## 4            45                 42.30            1840.75
## 5             2                 70.70             151.65
## 6             8                 99.65             820.50
## 7            22                 89.10            1949.40
## 8            10                 29.75             301.90
## 9            28                104.80            3046.05
## 10           62                 56.15            3487.95
## 11           13                 49.95             587.45
## 12           16                 18.95             326.80
## 13           58                100.35            5681.10
## 14           49                103.70            5036.30
## 15           25                105.50            2686.05
## 16           69                113.25            7895.15
## 17           52                 20.65            1022.95
## 18           71                106.70            7382.25
```

```
## 19           10                55.20             528.35
## 20           21                90.05            1862.90

matstd.quant_var_df5 <- scale(quant_var_df5[,])

# K-means, k=2, 3, 4, 5, 6
# Centers (k's) are numbers thus, 10 random sets are chosen

(kmeans2.quant_var_df5 <- kmeans(matstd.quant_var_df5,2,nstart = 10))

## K-means clustering with 2 clusters of sizes 7, 13
##
## Cluster means:
##   custc.tenure custc.MonthlyCharges custc.TotalCharges
## 1    0.9302622            0.8938768          1.1156512
## 2   -0.5009104           -0.4813183         -0.6007353
##
## Clustering vector:
##  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20
##  2  2  2  2  2  2  2  2  1  1  2  2  1  1  1  1  2  1  2  2
##
## Within cluster sum of squares by cluster:
## [1] 10.37800 15.29344
##  (between_SS / total_SS =  55.0 %)
##
## Available components:
##
## [1] "cluster"      "centers"      "totss"        "withinss"
"tot.withinss"
## [6] "betweenss"    "size"         "iter"         "ifault"

# Computing the percentage of variation accounted for. Two clusters
perc.var.2 <- round(100*(1 -
kmeans2.quant_var_df5$betweenss/kmeans2.quant_var_df5$totss),1)
names(perc.var.2) <- "Perc. 2 clus"
perc.var.2

## Perc. 2 clus
##           45

# Computing the percentage of variation accounted for. Three clusters
(kmeans3.quant_var_df5 <- kmeans(matstd.quant_var_df5,3,nstart = 10))

## K-means clustering with 3 clusters of sizes 5, 5, 10
##
## Cluster means:
##   custc.tenure custc.MonthlyCharges custc.TotalCharges
## 1   -0.3880662            0.8685290         -0.1069667
## 2    1.3603638            0.8129059          1.4735672
## 3   -0.4861488           -0.8407174         -0.6833002
##
```

```
## Clustering vector:
##  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20
##  3  3  3  3  3  1  1  3  1  2  3  3  2  2  1  2  3  2  3  1
##
## Within cluster sum of squares by cluster:
## [1] 1.170166 4.764672 8.968776
##  (between_SS / total_SS =  73.9 %)
##
## Available components:
##
## [1] "cluster"      "centers"      "totss"        "withinss"
"tot.withinss"
## [6] "betweenss"    "size"         "iter"         "ifault"
```

```
perc.var.3 <- round(100*(1 -
kmeans3.quant_var_df5$betweenss/kmeans3.quant_var_df5$totss),1)
names(perc.var.3) <- "Perc. 3 clus"
perc.var.3
```

```
## Perc. 3 clus
##         26.1
```

```
# Computing the percentage of variation accounted for. Four clusters
(kmeans4.quant_var_df5 <- kmeans(matstd.quant_var_df5,4,nstart = 10))
```

```
## K-means clustering with 4 clusters of sizes 4, 5, 4, 7
##
## Cluster means:
##   custc.tenure custc.MonthlyCharges custc.TotalCharges
## 1    0.7825290           -0.8035059         -0.1122134
## 2   -0.3880662            0.8685290         -0.1069667
## 3    1.3582315            1.1227175          1.7224755
## 4   -0.9461016           -0.8027845         -0.8437450
##
## Clustering vector:
##  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20
##  4  1  4  1  4  2  2  4  2  1  4  4  3  3  2  3  1  3  4  2
##
## Within cluster sum of squares by cluster:
## [1] 2.136999 1.170166 1.605810 2.373689
##  (between_SS / total_SS =  87.2 %)
##
## Available components:
##
## [1] "cluster"      "centers"      "totss"        "withinss"
"tot.withinss"
## [6] "betweenss"    "size"         "iter"         "ifault"
```

```
perc.var.4 <- round(100*(1 -
kmeans4.quant_var_df5$betweenss/kmeans4.quant_var_df5$totss),1)
```

```r
names(perc.var.4) <- "Perc. 4 clus"
perc.var.4

## Perc. 4 clus
##         12.8

# Computing the percentage of variation accounted for. Five clusters
(kmeans5.quant_var_df5 <- kmeans(matstd.quant_var_df5,5,nstart = 10))

## K-means clustering with 5 clusters of sizes 3, 4, 5, 4, 4
##
## Cluster means:
##   custc.tenure custc.MonthlyCharges custc.TotalCharges
## 1   -0.8912728           -1.3575362         -0.8731255
## 2    1.3582315            1.1227175          1.7224755
## 3   -0.3880662            0.8685290         -0.1069667
## 4   -0.9872232           -0.3867207         -0.8217096
## 5    0.7825290           -0.8035059         -0.1122134
##
## Clustering vector:
##  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20
##  1  5  4  5  4  3  3  1  3  5  4  1  2  2  3  2  5  2  4  3
##
## Within cluster sum of squares by cluster:
## [1] 0.2923777 1.6058095 1.1701665 0.4453126 2.1369991
##  (between_SS / total_SS =  90.1 %)
##
## Available components:
##
## [1] "cluster"      "centers"      "totss"        "withinss"
"tot.withinss"
## [6] "betweenss"    "size"         "iter"         "ifault"

perc.var.5 <- round(100*(1 -
kmeans5.quant_var_df5$betweenss/kmeans5.quant_var_df5$totss),1)
names(perc.var.5) <- "Perc. 5 clus"
perc.var.5

## Perc. 5 clus
##          9.9

(kmeans6.quant_var_df5 <- kmeans(matstd.quant_var_df5,6,nstart = 10))

## K-means clustering with 6 clusters of sizes 5, 1, 4, 3, 4, 3
##
## Cluster means:
##   custc.tenure custc.MonthlyCharges custc.TotalCharges
## 1   -0.3880662            0.8685290         -0.1069667
## 2    1.3688927           -0.4263405          0.4779338
## 3    1.3582315            1.1227175          1.7224755
## 4   -0.8912728           -1.3575362         -0.8731255
```

```
## 5   -0.9872232           -0.3867207         -0.8217096
## 6    0.5870744           -0.9292277         -0.3089291
##
## Clustering vector:
##  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20
##  4  6  5  6  5  1  1  4  1  2  5  4  3  3  1  3  6  3  5  1
##
## Within cluster sum of squares by cluster:
## [1] 1.1701665 0.0000000 1.6058095 0.2923777 0.4453126 1.0245328
##  (between_SS / total_SS =  92.0 %)
##
## Available components:
##
## [1] "cluster"      "centers"      "totss"        "withinss"
"tot.withinss"
## [6] "betweenss"    "size"         "iter"         "ifault"
```

```
# Computing the percentage of variation accounted for. Six clusters
perc.var.6 <- round(100*(1 -
kmeans6.quant_var_df5$betweenss/kmeans6.quant_var_df5$totss),1)
names(perc.var.6) <- "Perc. 6 clus"
perc.var.6
```

```
## Perc. 6 clus
##            8
```

```
#
kmeans4.quant_var_df5$cluster == 1
```

```
##     1     2     3     4     5     6     7     8     9    10    11    12
13
## FALSE  TRUE FALSE  TRUE FALSE FALSE FALSE FALSE FALSE  TRUE FALSE FALSE
FALSE
##    14    15    16    17    18    19    20
## FALSE FALSE FALSE  TRUE FALSE FALSE FALSE
```

```
# Saving four k-means clusters in a list
clus1 <-
matrix(names(kmeans4.quant_var_df5$cluster[kmeans4.quant_var_df5$cluster ==
1]),
                ncol=1,
nrow=length(kmeans4.quant_var_df5$cluster[kmeans4.quant_var_df5$cluster ==
1]))
colnames(clus1) <- "Cluster 1"
clus2 <-
matrix(names(kmeans4.quant_var_df5$cluster[kmeans4.quant_var_df5$cluster ==
2]),
                ncol=1,
nrow=length(kmeans4.quant_var_df5$cluster[kmeans4.quant_var_df5$cluster ==
2]))
colnames(clus2) <- "Cluster 2"
```

```r
clus3 <-
matrix(names(kmeans4.quant_var_df5$cluster[kmeans4.quant_var_df5$cluster ==
3]),
                ncol=1,
nrow=length(kmeans4.quant_var_df5$cluster[kmeans4.quant_var_df5$cluster ==
3]))
colnames(clus3) <- "Cluster 3"
clus4 <-
matrix(names(kmeans4.quant_var_df5$cluster[kmeans4.quant_var_df5$cluster ==
4]),
                ncol=1,
nrow=length(kmeans4.quant_var_df5$cluster[kmeans4.quant_var_df5$cluster ==
4]))
colnames(clus4) <- "Cluster 4"
list(clus1,clus2,clus3,clus4)
```

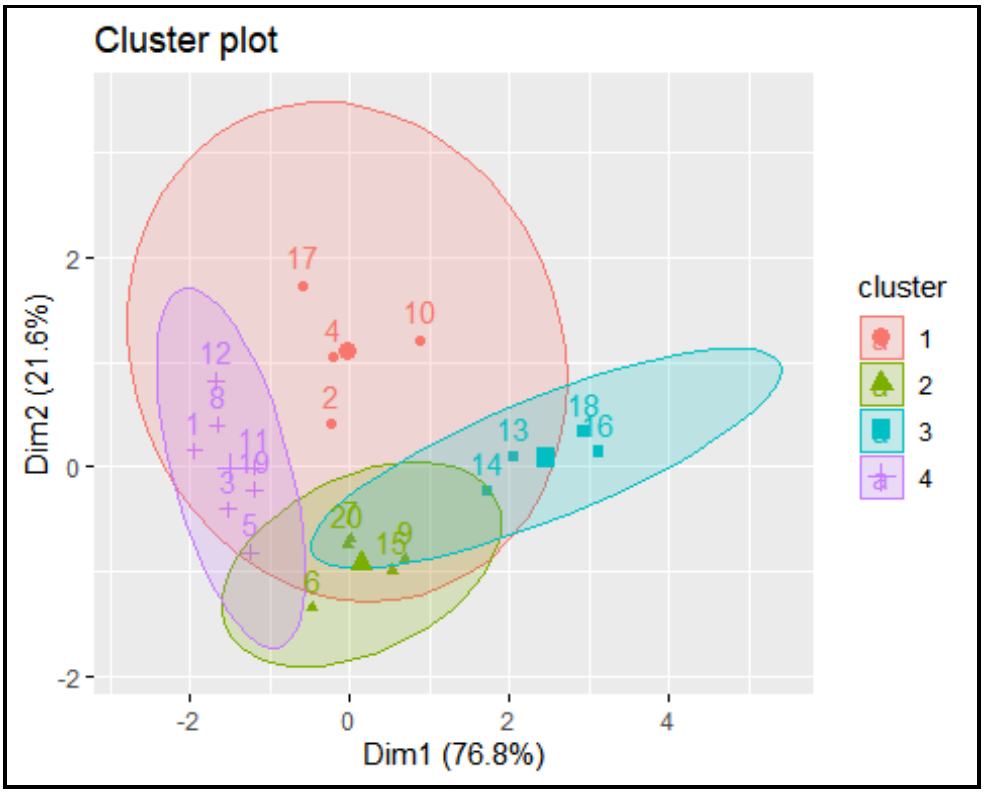
```
## [[1]]
##      Cluster 1
## [1,] "2"
## [2,] "4"
## [3,] "10"
## [4,] "17"
##
## [[2]]
##      Cluster 2
## [1,] "6"
## [2,] "7"
## [3,] "9"
## [4,] "15"
## [5,] "20"
##
## [[3]]
##      Cluster 3
## [1,] "13"
## [2,] "14"
## [3,] "16"
## [4,] "18"
##
## [[4]]
##      Cluster 4
## [1,] "1"
## [2,] "3"
## [3,] "5"
## [4,] "8"
## [5,] "11"
## [6,] "12"
## [7,] "19"
```

```r
library(factoextra)
```

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

## Welcome! Want to learn more? See two factoextra-related books at
https://goo.gl/ve3WBa

fviz_cluster(kmeans4.quant_var_df5, quant_var_df5[,], ellipse.type = "norm")
```




```
#Sample 2- Hierarchical clustering for rows 100-120
quant_var_df1<-
data.frame(custc$tenure,custc$MonthlyCharges,custc$TotalCharges)
quant_var_df6 <- quant_var_df1[c(100:120),]
quant_var_df6

##     custc.tenure custc.MonthlyCharges custc.TotalCharges
## 100           25                98.50            2514.50
## 101            1                20.20              20.20
## 102            1                19.45              19.45
## 103           38                95.00            3605.60
## 104           66                45.55            3027.25
## 105           68               110.00            7611.85
## 106            5                24.30             100.20
## 107           72               104.15            7303.05
## 108           32                30.15             927.65
## 109           43                94.35            3921.30
## 110           72                19.40            1363.25
## 111           55                96.75            5238.90
## 112           52                57.95            3042.25
```

```
## 113           43                91.65            3954.10
## 114           37                76.50            2868.15
## 115           64                54.60            3423.50
## 116            3                89.85             248.40
## 117           36                31.05            1126.35
## 118           10               100.25            1064.65
## 119           41                20.65             835.15
## 120           27                85.20            2151.60
```

```
attach(quant_var_df6)
```

```
## The following objects are masked from quant_var_df5 (pos = 4):
##
##     custc.MonthlyCharges, custc.tenure, custc.TotalCharges

## The following objects are masked from quant_var_df8:
##
##     custc.MonthlyCharges, custc.tenure, custc.TotalCharges

## The following objects are masked from quant_var_df7:
##
##     custc.MonthlyCharges, custc.tenure, custc.TotalCharges

## The following objects are masked from quant_var_df6 (pos = 7):
##
##     custc.MonthlyCharges, custc.tenure, custc.TotalCharges

## The following objects are masked from quant_var_df5 (pos = 8):
##
##     custc.MonthlyCharges, custc.tenure, custc.TotalCharges
```

```
# Standardizing the data with scale()
quant_var_df6[,]
```

```
##     custc.tenure custc.MonthlyCharges custc.TotalCharges
## 100           25                98.50            2514.50
## 101            1                20.20              20.20
## 102            1                19.45              19.45
## 103           38                95.00            3605.60
## 104           66                45.55            3027.25
## 105           68               110.00            7611.85
## 106            5                24.30             100.20
## 107           72               104.15            7303.05
## 108           32                30.15             927.65
## 109           43                94.35            3921.30
## 110           72                19.40            1363.25
## 111           55                96.75            5238.90
## 112           52                57.95            3042.25
## 113           43                91.65            3954.10
## 114           37                76.50            2868.15
## 115           64                54.60            3423.50
## 116            3                89.85             248.40
```

```
## 117           36                31.05            1126.35
## 118           10               100.25            1064.65
## 119           41                20.65             835.15
## 120           27                85.20            2151.60
```

```
matstd.quant_var_df6 <- scale(quant_var_df6[,])
(kmeans2.quant_var_df6 <- kmeans(matstd.quant_var_df6,2,nstart = 10))
```

```
## K-means clustering with 2 clusters of sizes 9, 12
##
## Cluster means:
##   custc.tenure custc.MonthlyCharges custc.TotalCharges
## 1   -0.6427879           -0.7492834         -0.8868544
## 2    0.4820910            0.5619626          0.6651408
##
## Clustering vector:
## 100 101 102 103 104 105 106 107 108 109 110 111 112 113 114 115 116 117
118 119
##   2   1   1   2   2   2   1   2   1   2   1   2   2   2   2   2   1   1
1   1
## 120
##   2
##
## Within cluster sum of squares by cluster:
## [1] 15.97046 16.29201
##  (between_SS / total_SS =  46.2 %)
##
## Available components:
##
## [1] "cluster"      "centers"      "totss"        "withinss"
"tot.withinss"
## [6] "betweenss"    "size"         "iter"         "ifault"
```

```
# Computing the percentage of variation accounted for. Two clusters
perc.var.2 <- round(100*(1 -
kmeans2.quant_var_df6$betweenss/kmeans2.quant_var_df6$totss),1)
names(perc.var.2) <- "Perc. 2 clus"
perc.var.2
```

```
## Perc. 2 clus
##         53.8
```

```
# Computing the percentage of variation accounted for. Three clusters
(kmeans3.quant_var_df6 <- kmeans(matstd.quant_var_df6,3,nstart = 10))
```

```
## K-means clustering with 3 clusters of sizes 9, 5, 7
##
## Cluster means:
##   custc.tenure custc.MonthlyCharges custc.TotalCharges
## 1    0.7545772            0.5370315          0.8985740
## 2   -0.7238351            0.7343294         -0.3717357
```

```
## 3   -0.4531455           -1.2149900         -0.8897839
##
## Clustering vector:
## 100 101 102 103 104 105 106 107 108 109 110 111 112 113 114 115 116 117
118 119
##   2   3   3   1   1   1   3   1   3   1   3   1   1   1   2   1   2   3
2   3
## 120
##   2
##
## Within cluster sum of squares by cluster:
## [1] 11.225672  2.620302  7.847382
##  (between_SS / total_SS =  63.8 %)
##
## Available components:
##
## [1] "cluster"      "centers"      "totss"        "withinss"
"tot.withinss"
## [6] "betweenss"    "size"         "iter"         "ifault"
```

```
perc.var.3 <- round(100*(1 -
kmeans3.quant_var_df6$betweenss/kmeans3.quant_var_df6$totss),1)
names(perc.var.3) <- "Perc. 3 clus"
perc.var.3
```

```
## Perc. 3 clus
##         36.2
```

```
# Computing the percentage of variation accounted for. Four clusters
(kmeans4.quant_var_df6 <- kmeans(matstd.quant_var_df6,4,nstart = 10))
```

```
## K-means clustering with 4 clusters of sizes 6, 3, 4, 8
##
## Cluster means:
##   custc.tenure custc.MonthlyCharges custc.TotalCharges
## 1   -0.7685508           -1.1944585        -0.94541351
## 2    1.1458394            1.1324450         1.87306076
## 3    1.0829580           -0.6056444         0.05676826
## 4   -0.3947556            0.7739992        -0.02172178
##
## Clustering vector:
## 100 101 102 103 104 105 106 107 108 109 110 111 112 113 114 115 116 117
118 119
##   4   1   1   4   3   2   1   2   1   4   3   2   3   4   4   3   4   1
4   1
## 120
##   4
##
## Within cluster sum of squares by cluster:
## [1] 3.521521 1.038578 1.678491 5.768619
##  (between_SS / total_SS =  80.0 %)
```

```
##
## Available components:
##
## [1] "cluster"      "centers"      "totss"        "withinss"
"tot.withinss"
## [6] "betweenss"    "size"         "iter"         "ifault"

perc.var.4 <- round(100*(1 -
kmeans4.quant_var_df6$betweenss/kmeans4.quant_var_df6$totss),1)
names(perc.var.4) <- "Perc. 4 clus"
perc.var.4

## Perc. 4 clus
##           20

# Computing the percentage of variation accounted for. Five clusters
(kmeans5.quant_var_df6 <- kmeans(matstd.quant_var_df6,5,nstart = 10))

## K-means clustering with 5 clusters of sizes 6, 3, 4, 2, 6
##
## Cluster means:
##   custc.tenure custc.MonthlyCharges custc.TotalCharges
## 1  -0.09082873            0.7384357         0.26323795
## 2   1.14583938            1.1324450         1.87306076
## 3   1.08295795           -0.6056444         0.05676826
## 4  -1.30653637            0.8806897        -0.87660099
## 5  -0.76855080           -1.1944585        -0.94541351
##
## Clustering vector:
## 100 101 102 103 104 105 106 107 108 109 110 111 112 113 114 115 116 117
118 119
##   1   5   5   1   3   2   5   2   5   1   3   2   3   1   1   3   4   5
4   5
## 120
##   1
##
## Within cluster sum of squares by cluster:
## [1] 1.4143649 1.0385780 1.6784911 0.1581332 3.5215214
##  (between_SS / total_SS =  87.0 %)
##
## Available components:
##
## [1] "cluster"      "centers"      "totss"        "withinss"
"tot.withinss"
## [6] "betweenss"    "size"         "iter"         "ifault"

perc.var.5 <- round(100*(1 -
kmeans5.quant_var_df6$betweenss/kmeans5.quant_var_df6$totss),1)
names(perc.var.5) <- "Perc. 5 clus"
perc.var.5
```

```
## Perc. 5 clus
##           13
```

```
(kmeans6.quant_var_df6 <- kmeans(matstd.quant_var_df6,6,nstart = 10))
```

```
## K-means clustering with 6 clusters of sizes 2, 3, 3, 6, 3, 4
##
## Cluster means:
##   custc.tenure custc.MonthlyCharges custc.TotalCharges
## 1  -1.30653637            0.8806897        -0.87660099
## 2  -1.48120700           -1.2819619        -1.15327623
## 3   1.14583938            1.1324450         1.87306076
## 4  -0.09082873            0.7384357         0.26323795
## 5  -0.05589460           -1.1069552        -0.73755079
## 6   1.08295795           -0.6056444         0.05676826
##
## Clustering vector:
## 100 101 102 103 104 105 106 107 108 109 110 111 112 113 114 115 116 117
118 119
##   4   2   2   4   6   3   2   3   5   4   6   3   6   4   4   6   1   5
1   5
## 120
##   4
##
## Within cluster sum of squares by cluster:
## [1] 0.15813323 0.03135876 1.03857803 1.41436492 0.13770708 1.67849113
##  (between_SS / total_SS =  92.6 %)
##
## Available components:
##
## [1] "cluster"      "centers"      "totss"        "withinss"
"tot.withinss"
## [6] "betweenss"    "size"         "iter"         "ifault"
```

```
# Computing the percentage of variation accounted for. Six clusters
perc.var.6 <- round(100*(1 -
kmeans6.quant_var_df6$betweenss/kmeans6.quant_var_df6$totss),1)
names(perc.var.6) <- "Perc. 6 clus"
perc.var.6
```

```
## Perc. 6 clus
##          7.4
```

```
#
kmeans5.quant_var_df6$cluster == 1
```

```
##   100   101   102   103   104   105   106   107   108   109   110   111
112
##  TRUE FALSE FALSE  TRUE FALSE FALSE FALSE FALSE FALSE  TRUE FALSE FALSE
FALSE
```

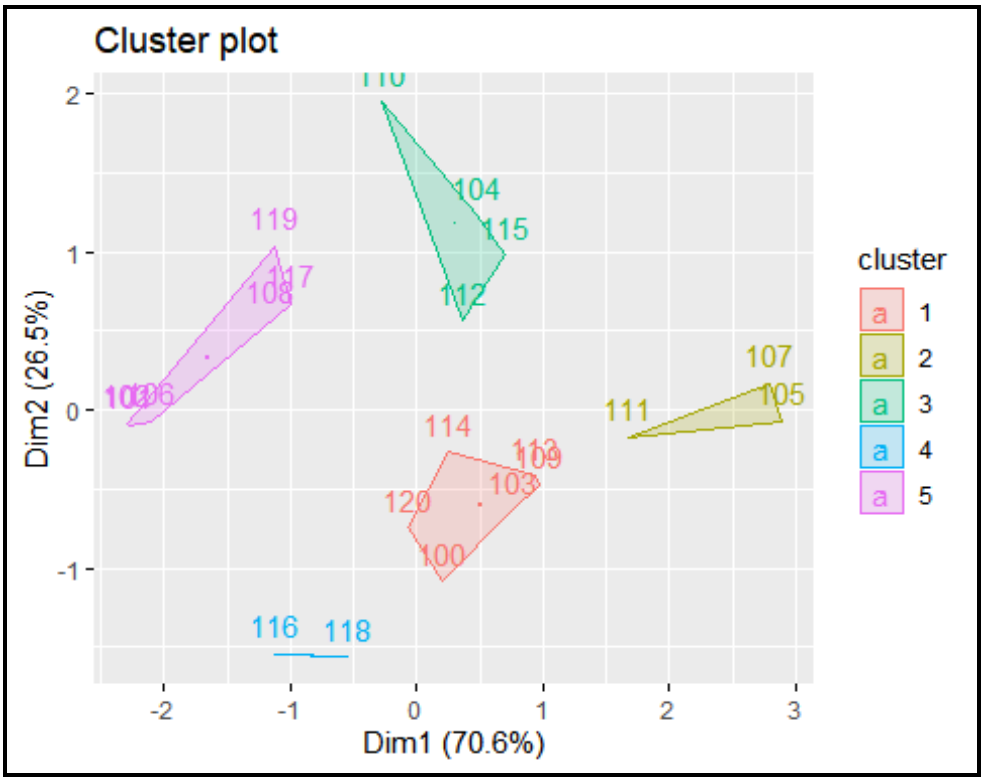
```
##   113   114   115   116   117   118   119   120
##  TRUE  TRUE FALSE FALSE FALSE FALSE FALSE  TRUE

# Saving five k-means clusters in a list
clus1 <-
matrix(names(kmeans5.quant_var_df6$cluster[kmeans5.quant_var_df6$cluster ==
1]),
                ncol=1,
nrow=length(kmeans5.quant_var_df6$cluster[kmeans5.quant_var_df6$cluster ==
1]))
colnames(clus1) <- "Cluster 1"
clus2 <-
matrix(names(kmeans5.quant_var_df6$cluster[kmeans5.quant_var_df6$cluster ==
2]),
                ncol=1,
nrow=length(kmeans5.quant_var_df6$cluster[kmeans5.quant_var_df6$cluster ==
2]))
colnames(clus2) <- "Cluster 2"
clus3 <-
matrix(names(kmeans5.quant_var_df6$cluster[kmeans5.quant_var_df6$cluster ==
3]),
                ncol=1,
nrow=length(kmeans5.quant_var_df6$cluster[kmeans5.quant_var_df6$cluster ==
3]))
colnames(clus3) <- "Cluster 3"
clus4 <-
matrix(names(kmeans5.quant_var_df6$cluster[kmeans5.quant_var_df6$cluster ==
4]),
                ncol=1,
nrow=length(kmeans5.quant_var_df6$cluster[kmeans5.quant_var_df6$cluster ==
4]))
colnames(clus4) <- "Cluster 4"
clus5 <-
matrix(names(kmeans5.quant_var_df6$cluster[kmeans5.quant_var_df6$cluster ==
5]),
                ncol=1,
nrow=length(kmeans5.quant_var_df6$cluster[kmeans5.quant_var_df6$cluster ==
5]))
colnames(clus4) <- "Cluster 5"
list(clus1,clus2,clus3,clus4,clus5)

## [[1]]
##      Cluster 1
## [1,] "100"
## [2,] "103"
## [3,] "109"
## [4,] "113"
## [5,] "114"
## [6,] "120"
##
```

```
## [[2]]
##      Cluster 2
## [1,] "105"
## [2,] "107"
## [3,] "111"
##
## [[3]]
##      Cluster 3
## [1,] "104"
## [2,] "110"
## [3,] "112"
## [4,] "115"
##
## [[4]]
##      Cluster 5
## [1,] "116"
## [2,] "118"
##
## [[5]]
##      [,1]
## [1,] "101"
## [2,] "102"
## [3,] "106"
## [4,] "108"
## [5,] "117"
## [6,] "119"
```

```
library(factoextra)
fviz_cluster(kmeans5.quant_var_df6, quant_var_df6[,], geom="text")
```


Cluster plot
Dim2 (26.5%)
Dim1 (70.6%)
2
1
0
-1
-2
-1
0
1
2
3
110
104
115
112
119
117
108
107
105
111
114
109
103
120
100
116
118
cluster
a 1
a 2
a 3
a 4
a 5