LAU9-By-Shantal-Cruz

```r
# HIERARCHICAL CLUSTERING
# This script performs a hierarchical clustering analysis

# Installing the package
# install.packages("dplyr")

# Loading package
library(dplyr)
```

```
##
## 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
```

```r
# Summary of dataset in package
head(mtcars)
```

```
##                    mpg cyl disp  hp drat    wt  qsec vs am gear carb
## Mazda RX4         21.0   6  160 110 3.90 2.620 16.46  0  1    4    4
## Mazda RX4 Wag     21.0   6  160 110 3.90 2.875 17.02  0  1    4    4
## Datsun 710        22.8   4  108  93 3.85 2.320 18.61  1  1    4    1
## Hornet 4 Drive    21.4   6  258 110 3.08 3.215 19.44  1  0    3    1
## Hornet Sportabout 18.7   8  360 175 3.15 3.440 17.02  0  0    3    2
## Valiant           18.1   6  225 105 2.76 3.460 20.22  1  0    3    1
```

```r
# Finding distance matrix
distance_mat <- dist(mtcars, method = 'euclidean')
distance_mat
```

```
##                       Mazda RX4 Mazda RX4 Wag  Datsun 710 Hornet 4 Drive
## Mazda RX4 Wag         0.6153251
## Datsun 710           54.9086059    54.8915169
## Hornet 4 Drive       98.1125212    98.0958939 150.9935191
## Hornet Sportabout   210.3374396   210.3358546 265.0831615    121.0297564
## Valiant              65.4717710    65.4392224 117.7547018     33.5508692
## Duster 360          241.4076490   241.4088680 294.4790230    169.4299647
## Merc 240D            50.1532711    50.1146059  49.6584796    121.2739722
## Merc 230             25.4683117    25.3284509  33.1803843    118.2433145
## Merc 280             15.3641921    15.2956865  66.9363534     91.4224033
## Merc 280C            15.6724727    15.5837744  67.0261397     91.4612914
## Merc 450SE          135.4307018   135.4254826 189.1954941     72.4964325
## Merc 450SL          135.4014424   135.3960351 189.1631745     72.4313532
## Merc 450SLC         135.4794674   135.4723157 189.2345426     72.5718466
```

```
## Cadillac Fleetwood  326.3395903   326.3355070 381.0926242    234.4403876
## Lincoln Continental 318.0469808   318.0429333 372.8012090    227.9726091
## Chrysler Imperial   304.7203408   304.7169175 359.3014906    218.1548299
## Fiat 128             93.2679950    93.2530993  40.9933763    184.9689734
## Honda Civic         102.8307567   102.8238713  52.7704607    191.5518700
## Toyota Corolla      100.6040368   100.5887588  47.6535017    192.6714187
## Toyota Corona        42.3075233    42.2659224  12.9654743    138.5304725
## Dodge Challenger    163.1150750   163.1134210 217.7795805     72.4403915
## AMC Javelin         149.6047203   149.6014522 204.3188913     61.3601899
## Camaro Z28          233.2228758   233.2248748 286.0049209    163.6632641
## Pontiac Firebird    248.6780270   248.6762035 303.3583889    156.2240346
## Fiat X1-9            92.5048389    92.4940020  39.8815148    184.4471198
## Porsche 914-2        44.4033659    44.4073589  13.1357109    139.1579524
## Lotus Europa         65.7328377    65.7362635  25.0948550    163.2367437
## Ford Pantera L      245.4247064   245.4293785 297.2940489    180.1140339
## Ferrari Dino         66.7661029    66.7764167  90.2415509    130.5523007
## Maserati Bora       265.6454248   265.6491465 309.7718171    229.3419352
## Volvo 142E           39.1894029    39.1626037  20.6939436    137.0363299
##                     Hornet Sportabout     Valiant  Duster 360   Merc 240D
## Mazda RX4 Wag
## Datsun 710
## Hornet 4 Drive
## Hornet Sportabout
## Valiant                   152.1241352
## Duster 360                 70.1767262 194.6094525
## Merc 240D                 241.5069657  89.5911056 281.2962502
## Merc 230                  233.4924012  85.0079649 265.8823313  33.6873047
## Merc 280                  199.3344960  60.2909811 227.8998521  64.7754228
## Merc 280C                 199.3406564  60.2655656 227.8813169  64.8898713
## Merc 450SE                 84.3888482  90.6970264 106.4084264 175.1620073
## Merc 450SL                 84.3683999  90.6769728 106.4320572 175.1189767
## Merc 450SLC                84.4332423  90.7092989 106.4010305 175.2118218
## Cadillac Fleetwood        116.2804201 266.6280942 119.0239068 355.6627498
## Lincoln Continental       108.0624299 259.6304391 104.5112999 348.9901277
## Chrysler Imperial          97.2049146 248.7713290  81.4297699 338.1959373
## Fiat 128                  302.0377212 152.1153263 333.9792070  68.6105903
## Honda Civic               310.0324645 158.9615769 344.0518316  72.0014488
## Toyota Corolla            309.5581776 159.8302995 341.0218232  76.2806458
## Toyota Corona             252.3331988 105.2876428 282.0508820  44.0850975
## Dodge Challenger           48.9838851 103.4310693 103.9023864 192.8617917
## AMC Javelin                61.4274240  91.0444349 110.3084921 180.5479760
## Camaro Z28                 70.9665308 187.8463771  10.0761203 273.8367985
## Pontiac Firebird           40.0052475 188.5272116  80.8057339 277.4606884
## Fiat X1-9                 301.5669483 151.4379425 333.4843231  67.9163981
## Porsche 914-2             254.1452553 106.0585767 285.1986201  39.4469276
## Lotus Europa              272.3582423 130.8248192 296.4572287  72.8971106
## Ford Pantera L             89.5934049 203.0177926  21.2655990 287.5238795
## Ferrari Dino              215.0673853 106.5694802 226.2036333 113.3023005
## Maserati Bora             170.7094473 242.4393015 107.7224977 313.8633093
## Volvo 142E                248.0063378 104.1863681 275.1353516  53.6823481
##                        Merc 230    Merc 280   Merc 280C  Merc 450SE  Merc 450SL
## Mazda RX4 Wag
## Datsun 710
## Hornet 4 Drive
```

```
## Hornet Sportabout
## Valiant
## Duster 360
## Merc 240D
## Merc 230
## Merc 280             39.2994160
## Merc 280C            39.3868519   1.5231546
## Merc 450SE          159.8179555 122.3642489 122.3461050
## Merc 450SL          159.7760899 122.3443771 122.3355492   0.9826495
## Merc 450SLC         159.8495837 122.3934970 122.3586862   1.3726252   2.1383405
## Cadillac Fleetwood  349.2832611 315.3904859 315.3557081 197.8842803 197.9154476
## Lincoln Continental 341.3154316 306.6760719 306.6406187 187.5997191 187.6330806
## Chrysler Imperial   328.4335161 292.7146896 292.6989332 171.6600758 171.6743028
## Fiat 128             69.3127910 106.5053149 106.6829794 228.3247948 228.2592340
## Honda Civic          78.5387212 116.7280991 116.8711475 238.0141824 237.9588183
## Toyota Corolla       76.7731674 113.6290721 113.8118009 235.5183809 235.4481971
## Toyota Corona        21.0962017  54.3641713  54.4258314 176.6020527 176.5727477
## Dodge Challenger    185.8331870 152.8929263 152.8722437  51.8008639  51.8242520
## AMC Javelin         172.5312555 139.1457974 139.1181977  41.2080044  41.2411618
## Camaro Z28          257.7469734 219.5520854 219.5276434  98.7203049  98.7566899
## Pontiac Firebird    271.3871978 238.1726099 238.1806292 124.3368538 124.3204160
## Fiat X1-9            68.5564864 105.7412910 105.8560373 227.7627676 227.7173075
## Porsche 914-2        22.1180967  57.6458160  57.8473863 179.5034108 179.4550855
## Lotus Europa         50.1094030  74.1443580  74.3824296 193.3074449 193.2407697
## Ford Pantera L      269.9772035 231.4081306 231.4024263 112.8181834 112.8296774
## Ferrari Dino         80.6550953  56.8365103  56.8987601 131.0272205 131.0077635
## Maserati Bora       288.8755628 250.5874125 250.5774357 157.1633256 157.1768956
## Volvo 142E           24.6913548  48.8053450  48.8884618 170.4500681 170.4225164
##                     Merc 450SLC Cadillac Fleetwood Lincoln Continental
## Mazda RX4 Wag
## Datsun 710
## Hornet 4 Drive
## Hornet Sportabout
## Valiant
## Duster 360
## Merc 240D
## Merc 230
## Merc 280
## Merc 280C
## Merc 450SE
## Merc 450SL
## Merc 450SLC
## Cadillac Fleetwood  197.8526242
## Lincoln Continental 187.5671081         15.6224446
## Chrysler Imperial   171.6557637         40.8399636          25.3714237
## Fiat 128            228.4051825        417.7687579         410.0206984
## Honda Civic         238.0828999        425.3271621         417.9679574
## Toyota Corolla      235.6024098        425.3446517         417.5429986
## Toyota Corona       176.6305359        368.3195488         360.0267515
## Dodge Challenger     51.8012606        163.6314881         156.2805020
## AMC Javelin          41.1929050        176.8610896         169.0925457
## Camaro Z28           98.7035830        128.4587210         114.0932078
## Pontiac Firebird    124.3726128         78.5385347          72.6947903
## Fiat X1-9           227.8176554        417.2490481         409.4998363
```

```
## Porsche 914-2      179.5720446       370.0956775        362.0145494
## Lotus Europa       193.3969216       388.5350012        379.4716659
## Ford Pantera L     112.8332602       134.8119464        119.7236456
## Ferrari Dino       131.0704490       328.5441628        317.7063117
## Maserati Bora      157.1683970       214.9366858        199.3420611
## Volvo 142E         170.4843735       364.1000930        355.4009443
##                    Chrysler Imperial    Fiat 128 Honda Civic Toyota Corolla
## Mazda RX4 Wag
## Datsun 710
## Hornet 4 Drive
## Hornet Sportabout
## Valiant
## Duster 360
## Merc 240D
## Merc 230
## Merc 280
## Merc 280C
## Merc 450SE
## Merc 450SL
## Merc 450SLC
## Cadillac Fleetwood
## Lincoln Continental
## Chrysler Imperial
## Fiat 128                 397.2276375
## Honda Civic              405.8152201  14.5590942
## Toyota Corolla           404.6335386   7.8324789  14.3480626
## Toyota Corona            346.5724649  52.8798281  63.8985563     59.8451285
## Dodge Challenger         145.9194779 254.2367888 261.8498815    261.8345312
## AMC Javelin              157.8097554 241.1203621 248.9636504    248.6917065
## Camaro Z28                91.2880886 325.6636235 335.8883188    332.6589699
## Pontiac Firebird          68.2030747 339.5857659 347.0655360    347.1667643
## Fiat X1-9                396.7597522   5.1473415  14.7807070     10.3922856
## Porsche 914-2            348.8466861  49.0644372  59.4588768     56.3243031
## Lotus Europa             364.5994326  49.9112509  64.0495153     53.8846563
## Ford Pantera L            95.3805385 337.1639236 347.8337714    343.9920962
## Ferrari Dino             300.1640703 128.3950054 141.7044478    133.4707617
## Maserati Bora            174.2936864 349.5338830 362.1620777    355.2601619
## Volvo 142E               341.2896659  61.3301247  73.3766041     67.7189421
##                    Toyota Corona Dodge Challenger AMC Javelin  Camaro Z28
## Mazda RX4 Wag
## Datsun 710
## Hornet 4 Drive
## Hornet Sportabout
## Valiant
## Duster 360
## Merc 240D
## Merc 230
## Merc 280
## Merc 280C
## Merc 450SE
## Merc 450SL
## Merc 450SLC
## Cadillac Fleetwood
## Lincoln Continental
```

```
## Chrysler Imperial
## Fiat 128
## Honda Civic
## Toyota Corolla
## Toyota Corona
## Dodge Challenger      205.0347927
## AMC Javelin           191.5580526       14.0154995
## Camaro Z28            273.6316895      100.3046106 105.6062618
## Pontiac Firebird      290.6240706       85.8075196  99.2836114  86.2665759
## Fiat X1-9              51.8411748      253.6624046 240.5266823 325.1490914
## Porsche 914-2           8.6535903      206.6452569 193.3080584 276.8924414
## Lotus Europa           31.2536926      226.5004836 212.7568765 287.6179004
## Ford Pantera L        285.1287911      118.7516779 123.3832044  19.3589023
## Ferrari Dino           82.2355734      174.9280395 161.1060307 216.7489910
## Maserati Bora         299.1865216      185.9059273 185.1553411 102.5946154
## Volvo 142E             12.2505275      201.3682522 187.6978440 266.5277736
##                     Pontiac Firebird   Fiat X1-9 Porsche 914-2 Lotus Europa
## Mazda RX4 Wag
## Datsun 710
## Hornet 4 Drive
## Hornet Sportabout
## Valiant
## Duster 360
## Merc 240D
## Merc 230
## Merc 280
## Merc 280C
## Merc 450SE
## Merc 450SL
## Merc 450SLC
## Cadillac Fleetwood
## Lincoln Continental
## Chrysler Imperial
## Fiat 128
## Honda Civic
## Toyota Corolla
## Toyota Corona
## Dodge Challenger
## AMC Javelin
## Camaro Z28
## Pontiac Firebird
## Fiat X1-9                 339.1396182
## Porsche 914-2             292.1646488  48.3775209
## Lotus Europa              311.3862342  49.8406880    33.7678653
## Ford Pantera L            101.7389686 336.7018783   288.5852993  297.5376920
## Ferrari Dino              255.0570519 127.8210813    87.9105966   80.4553451
## Maserati Bora             188.3240020 349.1199576   303.9222549  303.2796468
## Volvo 142E                286.7497823  60.4120429    18.7555858   27.8104457
##                     Ford Pantera L Ferrari Dino Maserati Bora
## Mazda RX4 Wag
## Datsun 710
## Hornet 4 Drive
## Hornet Sportabout
## Valiant
```

```
## Duster 360
## Merc 240D
## Merc 230
## Merc 280
## Merc 280C
## Merc 450SE
## Merc 450SL
## Merc 450SLC
## Cadillac Fleetwood
## Lincoln Continental
## Chrysler Imperial
## Fiat 128
## Honda Civic
## Toyota Corolla
## Toyota Corona
## Dodge Challenger
## AMC Javelin
## Camaro Z28
## Pontiac Firebird
## Fiat X1-9
## Porsche 914-2
## Lotus Europa
## Ford Pantera L
## Ferrari Dino           224.4587490
## Maserati Bora           86.9383253  223.5342175
## Volvo 142E             277.4803312   70.4751034   289.1157363
```

```
# Fitting Hierarchical clustering Model
# to training dataset
set.seed(240)  # Setting seed
Hierar_cl <- hclust(distance_mat, method = "average")
Hierar_cl
```

```
##
## Call:
## hclust(d = distance_mat, method = "average")
##
## Cluster method   : average
## Distance         : euclidean
## Number of objects: 32
```

```
# Plotting dendrogram
plot(Hierar_cl)

# Choosing no. of clusters
# Cutting tree by height
abline(h = 110, col = "green")

# Cutting tree by no. of clusters
fit <- cutree(Hierar_cl, k = 3 )
fit
```

```
##           Mazda RX4       Mazda RX4 Wag          Datsun 710      Hornet 4 Drive
##                   1                   1                   1                   2
##   Hornet Sportabout             Valiant          Duster 360           Merc 240D
##                   2                   2                   2                   1
```

```
##             Merc 230            Merc 280           Merc 280C          Merc 450SE 
##                   1                   1                   1                   2 
##          Merc 450SL         Merc 450SLC  Cadillac Fleetwood Lincoln Continental 
##                   2                   2                   2                   2 
##   Chrysler Imperial            Fiat 128         Honda Civic      Toyota Corolla 
##                   2                   1                   1                   1 
##       Toyota Corona    Dodge Challenger         AMC Javelin          Camaro Z28 
##                   1                   2                   2                   2 
##    Pontiac Firebird           Fiat X1-9       Porsche 914-2        Lotus Europa 
##                   2                   1                   1                   1 
##      Ford Pantera L        Ferrari Dino       Maserati Bora          Volvo 142E 
##                   2                   1                   3                   1
```

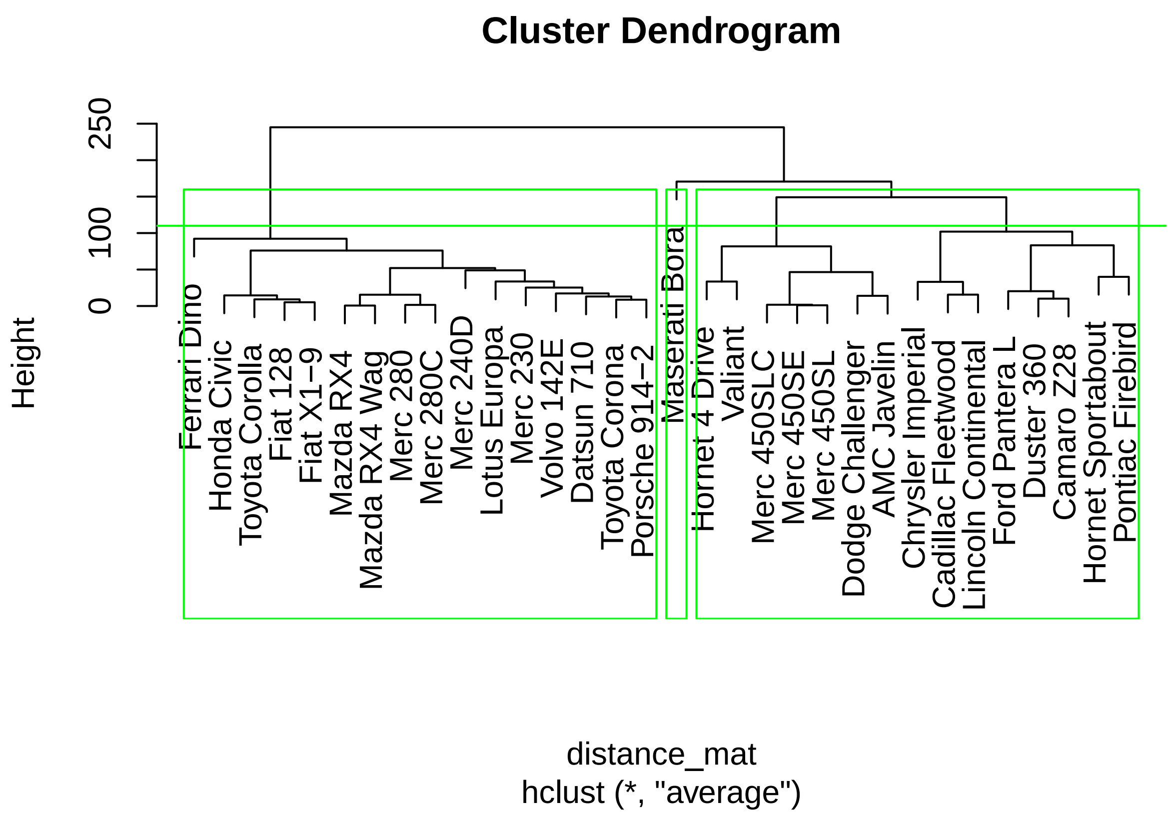
```r
table(fit)
```

```
## fit
##  1  2  3 
## 16 15  1
```

```r
rect.hclust(Hierar_cl, k = 3, border = "green")

# K-MEANS CLUSTERING

# Loading data
data(iris)

# Structure
str(iris)
```

```
## 'data.frame':    150 obs. of  5 variables:
##  $ Sepal.Length: num  5.1 4.9 4.7 4.6 5 5.4 4.6 5 4.4 4.9 ...
##  $ Sepal.Width : num  3.5 3 3.2 3.1 3.6 3.9 3.4 3.4 2.9 3.1 ...
##  $ Petal.Length: num  1.4 1.4 1.3 1.5 1.4 1.7 1.4 1.5 1.4 1.5 ...
##  $ Petal.Width : num  0.2 0.2 0.2 0.2 0.2 0.4 0.3 0.2 0.2 0.1 ...
##  $ Species     : Factor w/ 3 levels "setosa","versicolor",..: 1 1 1 1 1 1 1 1 1 1 ...
```

```r
# Installing Packages
# install.packages("ClusterR")
# install.packages("cluster")

# Loading package
library(ClusterR)
```




```r
library(cluster)

# Removing initial label of
# Species from original dataset
iris_1 <- iris[, -5]

# Fitting K-Means clustering Model
# to training dataset
set.seed(240) # Setting seed
kmeans.re <- kmeans(iris_1, centers = 3, nstart = 20)
kmeans.re
```

```
## K-means clustering with 3 clusters of sizes 50, 62, 38
##
## Cluster means:
##   Sepal.Length Sepal.Width Petal.Length Petal.Width
## 1     5.006000    3.428000     1.462000    0.246000
## 2     5.901613    2.748387     4.393548    1.433871
## 3     6.850000    3.073684     5.742105    2.071053
##
## Clustering vector:
##   [1] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
##  [38] 1 1 1 1 1 1 1 1 1 1 1 1 1 2 2 3 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
##  [75] 2 2 2 3 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 3 2 3 3 3 3 2 3 3 3 3
## [112] 3 3 2 2 3 3 3 3 2 3 2 3 2 3 3 2 2 3 3 3 3 3 2 3 3 3 3 2 3 3 3 2 3 3 3 2 3
## [149] 3 2
##
## Within cluster sum of squares by cluster:
## [1] 15.15100 39.82097 23.87947
```

```
##  (between_SS / total_SS =  88.4 %)
##
## Available components:
##
## [1] "cluster"      "centers"      "totss"        "withinss"     "tot.withinss"
## [6] "betweenss"    "size"         "iter"         "ifault"
```

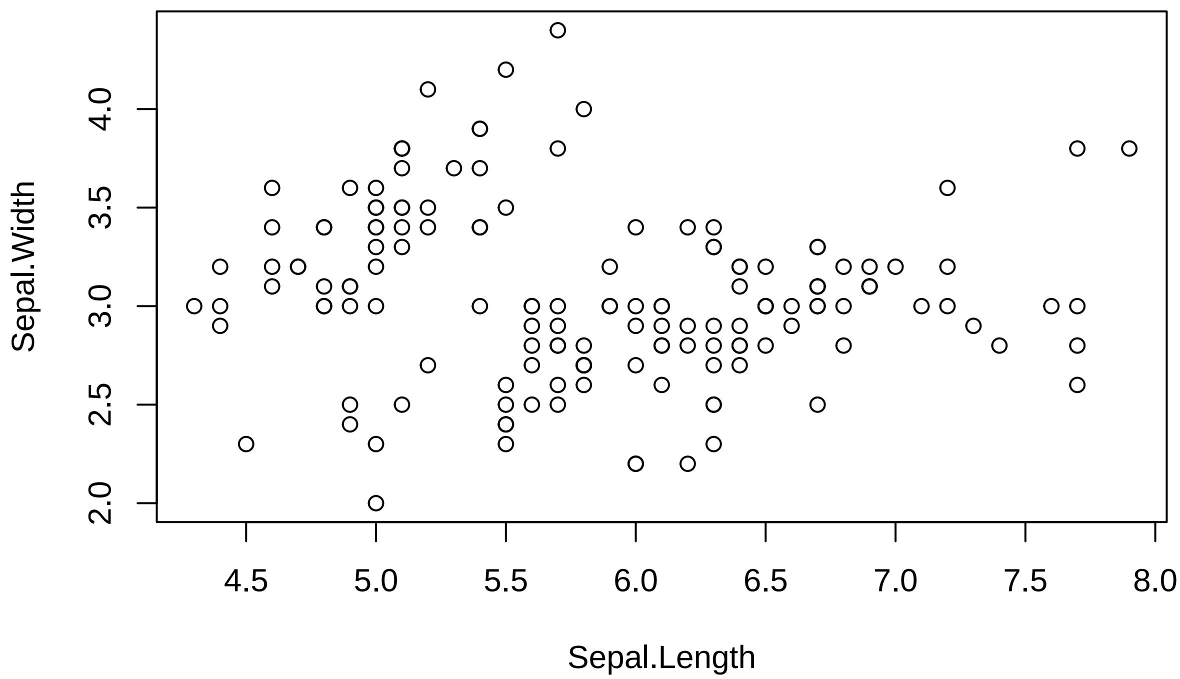
```
# Cluster identification for
# each observation
kmeans.re$cluster
```

```
##   [1] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
##  [38] 1 1 1 1 1 1 1 1 1 1 1 1 1 2 2 3 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
##  [75] 2 2 2 3 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 3 2 3 3 3 3 2 3 3 3 3
## [112] 3 3 2 2 3 3 3 3 2 3 2 3 2 3 3 2 2 3 3 3 3 3 2 3 3 3 3 2 3 3 3 2 3 3 3 2 3
## [149] 3 2
```

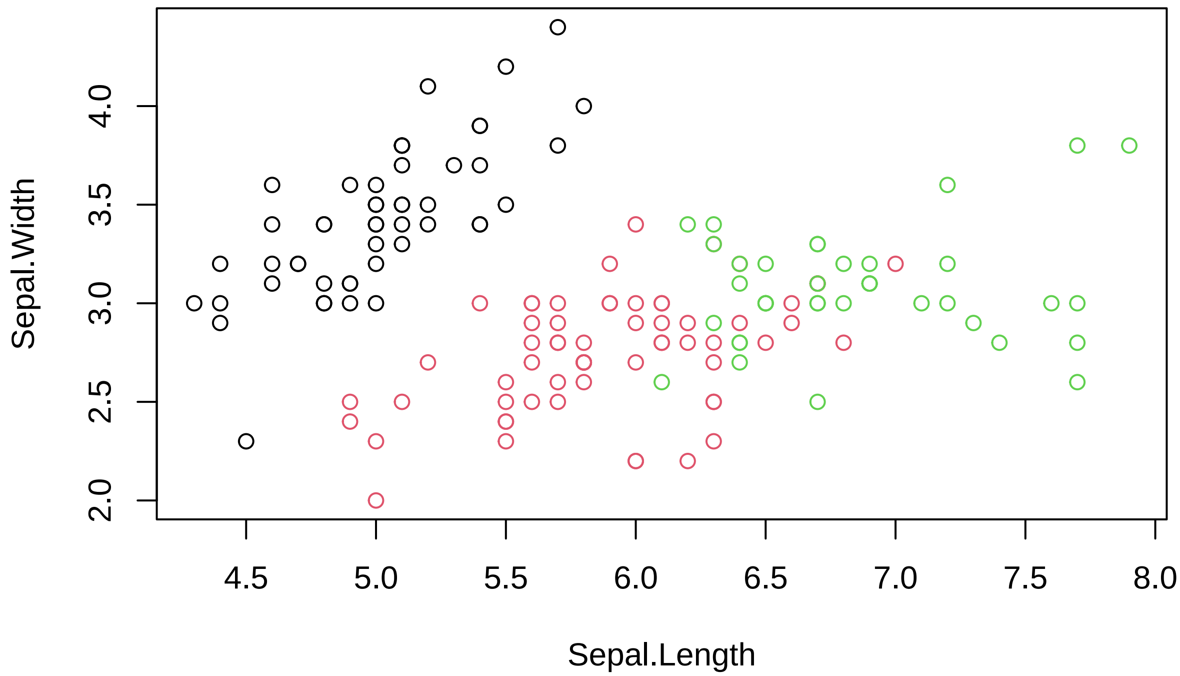
```
# Confusion Matrix
cm <- table(iris$Species, kmeans.re$cluster)
cm
```

```
##
##               1  2  3
##   setosa     50  0  0
##   versicolor  0 48  2
##   virginica   0 14 36
```

```
# Model Evaluation and visualization
plot(iris_1[c("Sepal.Length", "Sepal.Width")])
```




```
plot(iris_1[c("Sepal.Length", "Sepal.Width")],
     col = kmeans.re$cluster)
```



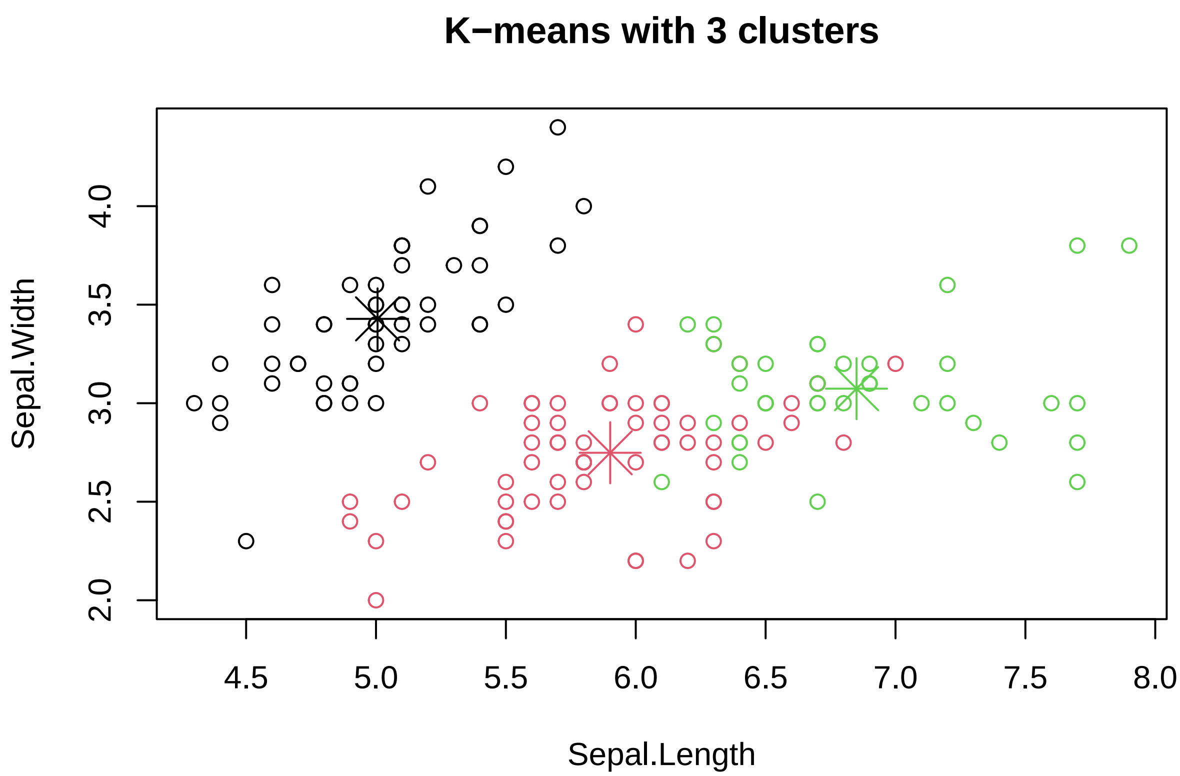

```r
plot(iris_1[c("Sepal.Length", "Sepal.Width")],
    col = kmeans.re$cluster,
    main = "K-means with 3 clusters")

## Plotiing cluster centers
kmeans.re$centers
```

```
##   Sepal.Length Sepal.Width Petal.Length Petal.Width
## 1     5.006000    3.428000     1.462000    0.246000
## 2     5.901613    2.748387     4.393548    1.433871
## 3     6.850000    3.073684     5.742105    2.071053
```

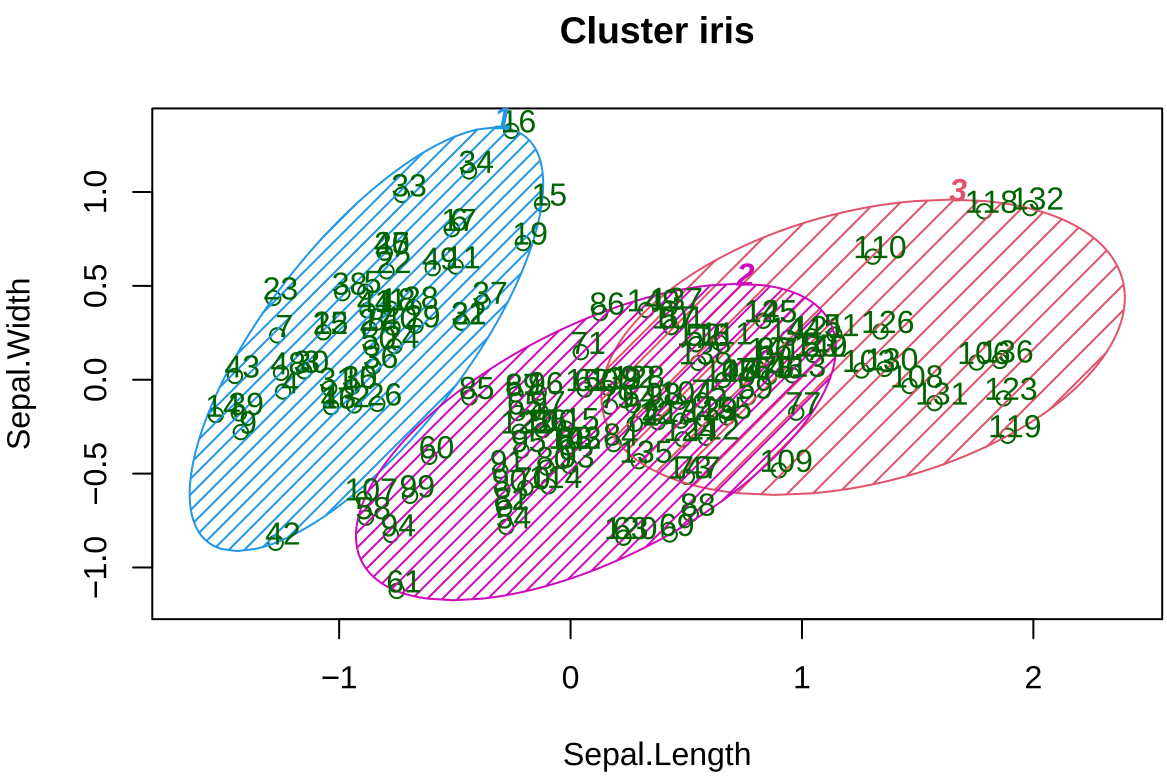
```r
kmeans.re$centers[, c("Sepal.Length", "Sepal.Width")]
```

```
##   Sepal.Length Sepal.Width
## 1     5.006000    3.428000
## 2     5.901613    2.748387
## 3     6.850000    3.073684
```

```r
# cex is font size, pch is symbol
points(kmeans.re$centers[, c("Sepal.Length", "Sepal.Width")],
    col = 1:3, pch = 8, cex = 3)
```




```r
## Visualizing clusters
y_kmeans <- kmeans.re$cluster
clusplot(iris_1[, c("Sepal.Length", "Sepal.Width")],
        y_kmeans,
        lines = 0,
        shade = TRUE,
        color = TRUE,
        labels = 2,
        plotchar = FALSE,
        span = TRUE,
        main = paste("Cluster iris"),
        xlab = 'Sepal.Length',
        ylab = 'Sepal.Width')
```

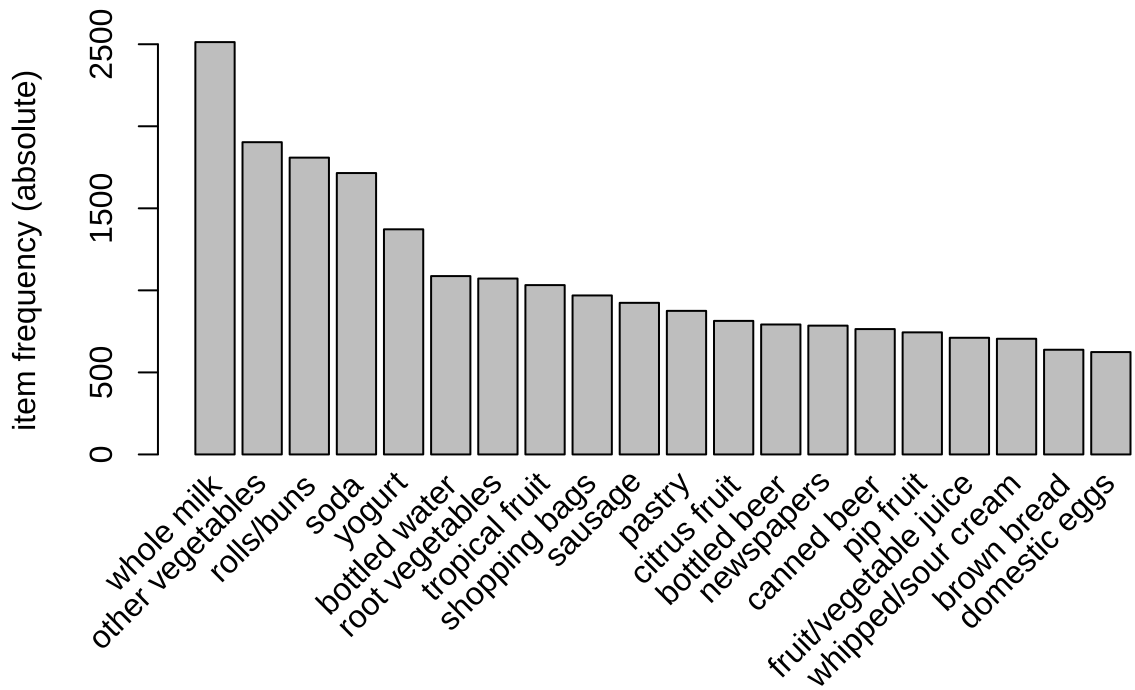


These two components explain 100 % of the point variability.

```r
# MARKET BASKET ANALYSIS
# install.packages("arules")
# install.packages("arulesViz")
# install.packages("datasets")

# Load the libraries
library(arules)
```

```
## Loading required package: Matrix

##
## Attaching package: 'arules'

## The following object is masked from 'package:dplyr':
##
##     recode

## The following objects are masked from 'package:base':
##
##     abbreviate, write
```

```r
library(arulesViz)
library(datasets)

# Load the data set
data(Groceries)

# Create an item frequency plot for the top 20 items
itemFrequencyPlot(Groceries,topN=20,type="absolute")
```




```r
# Get the rules
rules <- apriori(Groceries, parameter = list(supp = 0.001, conf = 0.8))
```

```
## Apriori
##
## Parameter specification:
##  confidence minval smax arem  aval originalSupport maxtime support minlen
##         0.8    0.1    1 none FALSE            TRUE       5   0.001      1
##  maxlen target  ext
##      10  rules TRUE
##
## Algorithmic control:
##  filter tree heap memopt load sort verbose
##     0.1 TRUE TRUE  FALSE TRUE    2    TRUE
##
## Absolute minimum support count: 9
##
## set item appearances ...[0 item(s)] done [0.00s].
## set transactions ...[169 item(s), 9835 transaction(s)] done [0.00s].
## sorting and recoding items ... [157 item(s)] done [0.00s].
## creating transaction tree ... done [0.00s].
## checking subsets of size 1 2 3 4 5 6 done [0.01s].
## writing ... [410 rule(s)] done [0.00s].
## creating S4 object  ... done [0.00s].
```

```r
# Show the top 5 rules, but only 2 digits
options(digits=2)
inspect(rules[1:5])
```

```
##     lhs                         rhs             support confidence coverage lift
## [1] {liquor, red/blush wine} => {bottled beer} 0.0019  0.90       0.0021   11.2
## [2] {curd, cereals}          => {whole milk}   0.0010  0.91       0.0011    3.6
## [3] {yogurt, cereals}        => {whole milk}   0.0017  0.81       0.0021    3.2
```

```
## [4] {butter, jam}            => {whole milk}   0.0010  0.83       0.0012    3.3
## [5] {soups, bottled beer}    => {whole milk}   0.0011  0.92       0.0012    3.6
##     count
## [1] 19
## [2] 10
## [3] 17
## [4] 10
## [5] 11
```

```r
rules<-sort(rules, by="confidence", decreasing=TRUE)

rules <- apriori(Groceries, parameter = list(supp = 0.001, conf = 0.8,maxlen=3))
```

```
## Apriori
##
## Parameter specification:
##  confidence minval smax arem  aval originalSupport maxtime support minlen
##         0.8    0.1    1 none FALSE            TRUE       5   0.001      1
##  maxlen target  ext
##       3  rules TRUE
##
## Algorithmic control:
##  filter tree heap memopt load sort verbose
##     0.1 TRUE TRUE  FALSE TRUE    2    TRUE
##
## Absolute minimum support count: 9
##
## set item appearances ...[0 item(s)] done [0.00s].
## set transactions ...[169 item(s), 9835 transaction(s)] done [0.00s].
## sorting and recoding items ... [157 item(s)] done [0.00s].
## creating transaction tree ... done [0.00s].
## checking subsets of size 1 2 3

## Warning in apriori(Groceries, parameter = list(supp = 0.001, conf = 0.8, :
## Mining stopped (maxlen reached). Only patterns up to a length of 3 returned!

##  done [0.00s].
## writing ... [29 rule(s)] done [0.00s].
## creating S4 object  ... done [0.00s].
```

```r
subset.matrix <- is.subset(rules, rules)
subset.matrix[lower.tri(subset.matrix, diag=T)] <- NA
```

```
## Warning in `[<-`(`*tmp*`, as.vector(i), value = NA): x[.] <- val: x is
## "ngTMatrix", val not in {TRUE, FALSE} is coerced; NA |--> TRUE.
```

```r
redundant <- colSums(subset.matrix, na.rm=T) >= 1
rules.pruned <- rules[!redundant]
rules<-rules.pruned

rules<-apriori(data=Groceries, parameter=list(supp=0.001,conf = 0.08),
               appearance = list(default="lhs",rhs="whole milk"),
               control = list(verbose=F))
rules<-sort(rules, decreasing=TRUE,by="confidence")
inspect(rules[1:5])
```

```
##     lhs                        rhs          support confidence coverage lift count
```

```
## [1] {rice,
##      sugar}                => {whole milk}  0.0012          1   0.0012  3.9    12
## [2] {canned fish,
##      hygiene articles}     => {whole milk}  0.0011          1   0.0011  3.9    11
## [3] {root vegetables,
##      butter,
##      rice}                 => {whole milk}  0.0010          1   0.0010  3.9    10
## [4] {root vegetables,
##      whipped/sour cream,
##      flour}                => {whole milk}  0.0017          1   0.0017  3.9    17
## [5] {butter,
##      soft cheese,
##      domestic eggs}        => {whole milk}  0.0010          1   0.0010  3.9    10
```

```r
rules<-apriori(data=Groceries, parameter=list(supp=0.001,conf = 0.15,minlen=2),
               appearance = list(default="rhs",lhs="whole milk"),
               control = list(verbose=F))
rules<-sort(rules, decreasing=TRUE,by="confidence")
inspect(rules[1:5])
```

```
##     lhs             rhs                  support confidence coverage lift count
## [1] {whole milk} => {other vegetables} 0.075   0.29       0.26     1.5  736
## [2] {whole milk} => {rolls/buns}       0.057   0.22       0.26     1.2  557
## [3] {whole milk} => {yogurt}           0.056   0.22       0.26     1.6  551
## [4] {whole milk} => {root vegetables}  0.049   0.19       0.26     1.8  481
## [5] {whole milk} => {tropical fruit}   0.042   0.17       0.26     1.6  416
```

```r
library(arulesViz)
plot(rules,method="graph")
```