Section 11 | Clustering

Mohammad Saqib Ansari

2023-12-22

Clustering

Theory of Clustering

Clustering is an unsupervised learning technique that involves grouping similar data points together based on certain characteristics or features. The primary objective is to discover inherent structures within the data, where items in the same cluster are more similar to each other compared to those in different clusters.

Basics of Clustering

Algorithms

Several algorithms are used for clustering:

1. **K-Means**: Divides data into K clusters by minimizing the sum of distances within clusters.
2. **Hierarchical Clustering**: Builds a tree of clusters by merging or splitting them based on similarity.
3. **DBSCAN**: Density-based algorithm that forms clusters based on dense regions separated by sparse areas.
4. **Mean Shift**: Identifies modes in the data density to form clusters.

Distance Measures

Common distance measures used in clustering include Euclidean distance, Manhattan distance, and Cosine similarity.

Evaluation Metrics

Metrics such as Silhouette Score, Davies-Bouldin Index, and Elbow Method aid in evaluating clustering performance.

Assumptions in Clustering

- **Homogeneity**: Items within a cluster are more similar to each other.
- **Separation**: Items from different clusters are dissimilar.
- **Cluster Shape**: Assumption on the shape of clusters (e.g., spherical for K-Means).

Types of Clustering

Partitioning Methods

- **K-Means**: Divides data into K clusters by minimizing intra-cluster variance.
- **Fuzzy C-Means**: Similar to K-Means but assigns data points to multiple clusters with varying degrees of membership.

Hierarchical Methods

- **Agglomerative**: Starts with individual data points as clusters and merges them based on similarity.
- **Divisive**: Begins with a single cluster and splits it into smaller clusters hierarchically.

Density-Based Methods

- **DBSCAN**: Forms clusters based on density-connected points.
- **OPTICS**: Orders points based on their density reachability.

Model-Based Methods

- **Gaussian Mixture Models (GMM)**: Assumes data points are generated from a mixture of Gaussian distributions.

Applications of Clustering

- **Customer Segmentation**: Grouping customers based on purchasing behavior.
- **Image Segmentation**: Partitioning images into distinct regions.
- **Anomaly Detection**: Identifying outliers in data.
- **Recommendation Systems**: Grouping users/items with similar preferences.

Disadvantages of Clustering

- **Sensitivity to Parameters**: Performance can vary based on initial parameters.
- **Scalability**: Some algorithms may struggle with large datasets.
- **Assumption of Cluster Shape**: Algorithms assuming specific cluster shapes may perform poorly with irregular-shaped clusters.
- **Difficulty in Choosing K**: Determining the optimal number of clusters can be challenging.

Hierarchical Clustering Analysis (HCA) for Individual Characterization

Hierarchical Clustering Analysis (HCA), specifically Ascending Hierarchical Clustering (AHC), constructs a hierarchy of individuals graphically represented by a dendrogram or hierarchical tree.

Step 1: Read the Data

The `decath` dataset is read using `read.csv`. The dataset presumably contains information about decathlon participants.

```
decath <- read.csv("decathlon.csv", header = TRUE, row.names = 1)
```

Step 2: Data Standardization

If necessary, the data is standardized using the `scale` function to bring all variables to a common scale, especially important when variables have different units or scales.

```
library(cluster)

# Standardize the data
scaled_decath <- scale(decath[, 1:10])
```

Step 3: Construct Ascending Hierarchical Clustering

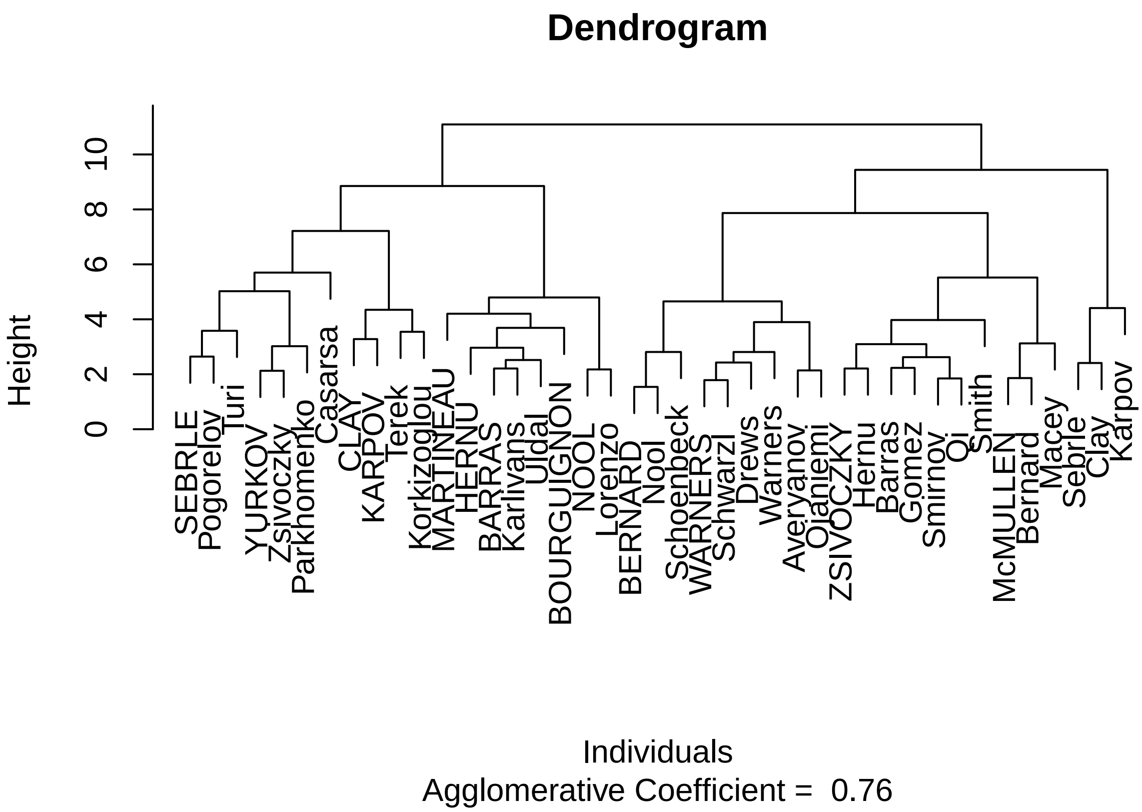
The `agnes` function from the `cluster` library performs the AHC using the Ward's linkage method, which minimizes the within-cluster variance.

```
res.ahc <- agnes(scaled_decath, method = "ward")
```

Step 4: Visualize the Dendrogram

The resulting hierarchical clustering is plotted as a dendrogram using the `plot` function. This visualizes the hierarchy of individuals based on their similarity or dissimilarity.

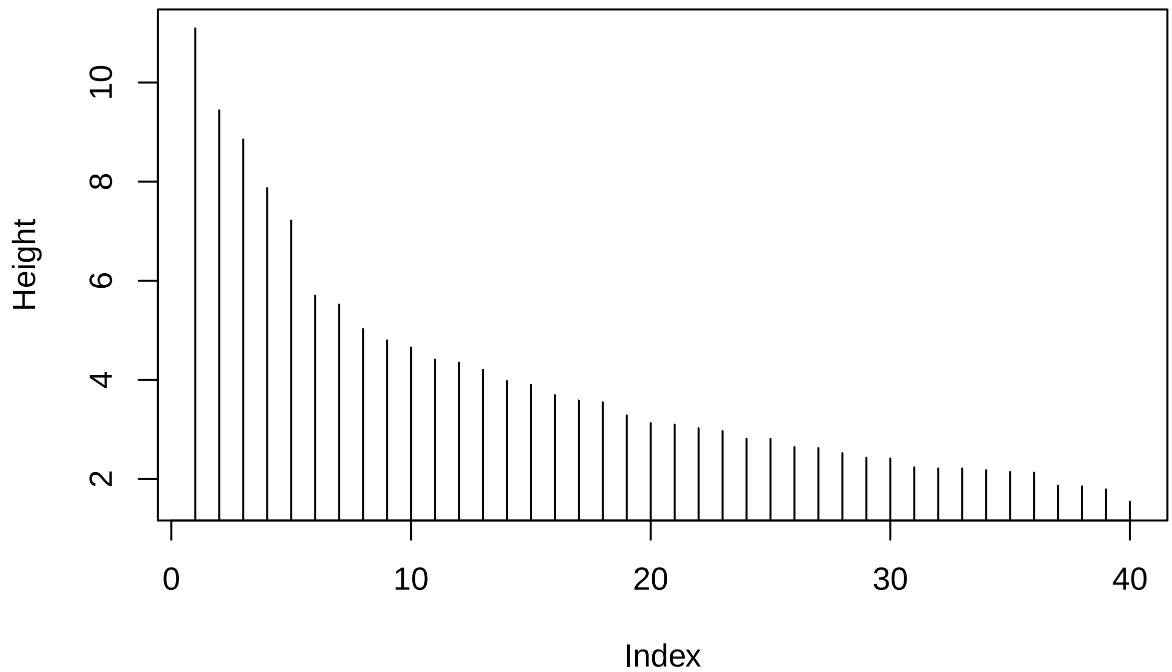
```
plot(res.ahc, which.plots = 2, main = "Dendrogram", xlab = "Individuals")
```



Step 5: Prune the Hierarchical Tree and Characterize the Clusters

The hierarchical tree can be pruned using a specific number of clusters (`k=4` in this case) using the `cutree` function. Then, the `FactoMineR` library is used to describe the clusters' characteristics.

```
res.ahc2 <- as.hclust(res.ahc)
plot(rev(res.ahc2$height), type = "h", ylab = "Height")
```




```r
clusters.hac <- cutree(res.ahc, k = 4)

# Convert cluster results to a factor
clusters.hac <- as.factor(clusters.hac)

# Combine cluster information with the dataset
decath.comp <- cbind.data.frame(decath, clusters.hac)

# Characterize clusters using FactoMineR's catdes function
library("FactoMineR")
catdes(decath.comp, num.var = 14)
```

```
##
## Link between the cluster variable and the quantitative variables
## ================================================================
##                    Eta2      P-value
## Points        0.7181466 2.834869e-10
## X100m         0.5794940 4.206021e-07
## Shot.put      0.4746713 2.358915e-05
## Long.jump     0.4405023 7.319114e-05
## X110m.hurdle 0.4254154 1.178937e-04
## X400m         0.4124012 1.759714e-04
## X1500m        0.4078248 2.021278e-04
## Discus        0.3593367 8.199096e-04
## High.jump     0.2864826 5.454320e-03
```

```
## Javeline     0.1999118 3.912122e-02
## Rank         0.1916166 4.655439e-02
##
## Description of each cluster by quantitative variables
## =====================================================
## $`1`
##              v.test Mean in category Overall mean
## X1500m     3.899044       290.763636    279.02488
## X400m      2.753420        50.435455     49.61634
## Long.jump -2.038672         7.093636      7.26000
##           sd in category Overall sd      p.value
## X1500m        12.6274652 11.5300118 9.657328e-05
## X400m          1.2725877  1.1392975 5.897622e-03
## Long.jump      0.2836218  0.3125193 4.148279e-02
##
## $`2`
##                 v.test Mean in category Overall mean
## X100m        -2.265855         10.89789     10.99805
## X110m.hurdle -2.397231         14.41579     14.60585
## X400m        -2.579590         49.11632     49.61634
## X1500m       -2.975997        273.18684    279.02488
##              sd in category Overall sd     p.value
## X100m             0.1701572  0.2597956 0.023460250
## X110m.hurdle      0.3097931  0.4660000 0.016519515
## X400m             0.5562394  1.1392975 0.009891780
## X1500m            5.6838942 11.5300118 0.002920378
##
## $`3`
##                 v.test Mean in category Overall mean
## X100m         4.053791         11.33625    10.998049
## X110m.hurdle  3.368905         15.11000    14.605854
## High.jump    -2.412939          1.90875     1.976829
## Shot.put     -3.134116         13.65750    14.477073
## Points       -3.643979       7609.62500  8005.365854
##              sd in category   Overall sd      p.value
## X100m            0.14194519   0.25979560 5.039416e-05
## X110m.hurdle     0.32939338   0.46599998 7.546748e-04
## High.jump        0.05464373   0.08785906 1.582447e-02
## Shot.put         0.60373318   0.81431175 1.723728e-03
## Points         143.94611622 338.18394159 2.684548e-04
##
## $`4`
##                 v.test Mean in category Overall mean
## Points        4.242103       8812.66667  8005.365854
## Long.jump     3.468581          7.87000     7.260000
## Discus        3.107539         50.16000    44.325610
## Shot.put      2.974272         15.84000    14.477073
## Javeline      2.586808         65.25667    58.316585
## High.jump     2.289003          2.09000     1.976829
## X110m.hurdle -2.119695         14.05000    14.605854
## Rank         -2.299627          2.00000    12.121951
## X400m        -2.333955         48.12000    49.616341
## X100m        -2.745523         10.59667    10.998049
##              sd in category   Overall sd      p.value
```

```
## Points          68.78145745 338.18394159 2.214348e-05
## Long.jump        0.06480741   0.31251927 5.232144e-04
## Discus           1.19668988   3.33639725 1.886523e-03
## Shot.put         0.46568945   0.81431175 2.936847e-03
## Javeline         6.87867397   4.76759315 9.686955e-03
## High.jump        0.02449490   0.08785906 2.207917e-02
## X110m.hurdle     0.06531973   0.46599998 3.403177e-02
## Rank             0.81649658   7.82178048 2.146935e-02
## X400m            0.98634004   1.13929751 1.959810e-02
## X100m            0.18080069   0.25979560 6.041458e-03
```

The `cutree` function is used to assign each individual to a cluster. The resulting clusters are then added to the `decath` dataset. Finally, the `catdes` function from the `FactoMineR` package characterizes the clusters in terms of their variables.

This process provides insights into how individuals are grouped based on the variables included in the decathlon dataset, helping to understand similarities and differences between groups.

K-Means Algorithm Implementation

The K-Means algorithm is utilized to partition datasets into clusters. Below are the steps followed in the code:

Part 1: K-Means on Iris Dataset

Step 1: Read the Data

The code begins by loading the 'iris' dataset and displaying its initial rows using the `head` function.

```r
data("iris")
head(iris)
```

```
##   Sepal.Length Sepal.Width Petal.Length Petal.Width Species
## 1          5.1         3.5          1.4         0.2  setosa
## 2          4.9         3.0          1.4         0.2  setosa
## 3          4.7         3.2          1.3         0.2  setosa
## 4          4.6         3.1          1.5         0.2  setosa
## 5          5.0         3.6          1.4         0.2  setosa
## 6          5.4         3.9          1.7         0.4  setosa
```

Step 2: Unsupervised Learning using K-Means

For unsupervised learning, only the 'petal width' and 'petal length' columns (`x = iris[, 3:4]`) are selected from the 'iris' dataset. Then, K-Means is applied with `n=3` clusters.

```r
x <- iris[, 3:4]  # Using petal width and length
model <- kmeans(x, 3)  # For n=3 clusters
model
```

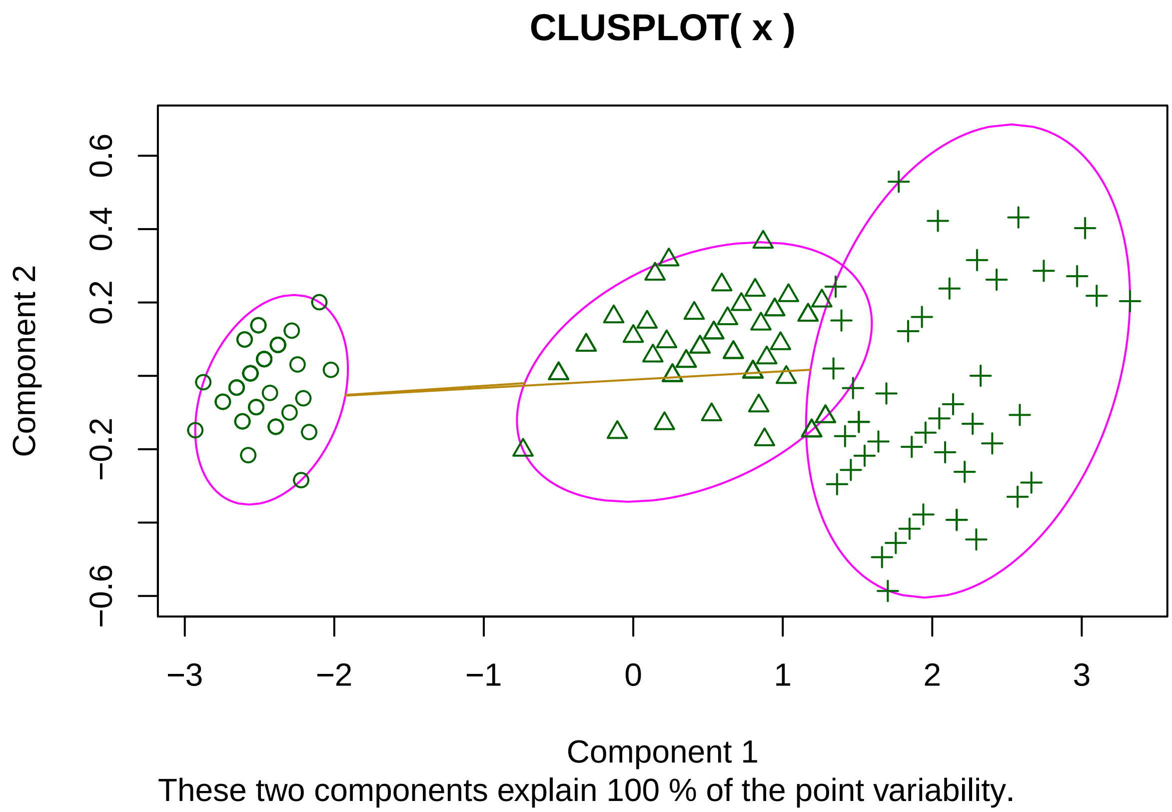
```
## K-means clustering with 3 clusters of sizes 50, 54, 46
##
## Cluster means:
##   Petal.Length Petal.Width
## 1     1.462000    0.246000
## 2     4.292593    1.359259
## 3     5.626087    2.047826
##
## Clustering vector:
##   [1] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
##  [30] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2
##  [59] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 3 2 2 2 2 2 3 2 2 2 2
##  [88] 2 2 2 2 2 2 2 2 2 2 2 2 2 3 3 3 3 3 3 2 3 3 3 3 3 3 3 3 3
## [117] 3 3 3 2 3 3 3 2 3 3 2 2 3 3 3 3 3 3 3 3 3 3 3 2 3 3 3 3 3
## [146] 3 3 3 3 3
##
## Within cluster sum of squares by cluster:
## [1]  2.02200 14.22741 15.16348
##  (between_SS / total_SS =  94.3 %)
##
## Available components:
##
## [1] "cluster"      "centers"      "totss"        "withinss"
## [5] "tot.withinss" "betweenss"    "size"         "iter"
## [9] "ifault"
```

Step 3: Visualize Clusters

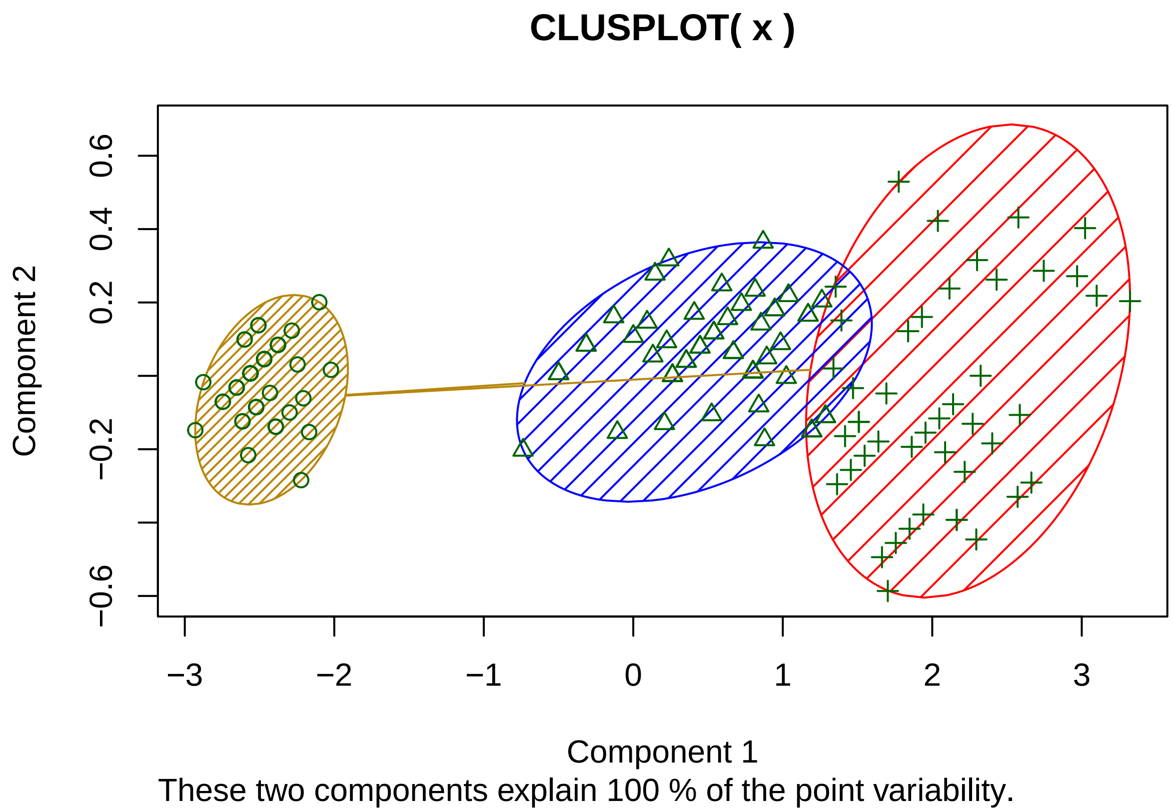
The resulting clusters are visualized using a cluster plot with `clusplot` to display the distribution of clusters based on the selected features.

```r
library(cluster)

clusplot(x, model$cluster)
```




```
clusplot(x, model$cluster, color = TRUE, shade = TRUE)
```



Part 2: K-Means on Decathlon Dataset

Step 1: Read the Data

The code reads the 'decathlon' dataset using `read.csv`.

```
decath <- read.csv("decathlon.csv", header = TRUE, row.names = 1)
```

Step 2: Apply K-Means

The K-Means algorithm is applied to the standardized version of columns 1 to 10 of the 'decathlon' dataset with `centers=4`.

```
results.kmeans <- kmeans(scale(decath[, 1:10]), centers = 4)
results.kmeans
```

```
## K-means clustering with 4 clusters of sizes 13, 14, 6, 8
##
## Cluster means:
##         X100m   Long.jump   Shot.put   High.jump      X400m
## 1 -0.3815502 -0.04376132  0.1015202  0.30372937 -0.3576185
## 2 -0.5737366  0.86463340  0.2556730  0.05973646 -0.5554005
## 3  0.4509791 -1.19047001  0.5089512  0.22301610  1.3513004
```

```
## 4  1.2858238 -0.54914380 -0.9941115 -0.76536110  0.5396056
##   X110m.hurdle       Discus  Pole.vault    Javeline     X1500m
## 1   -0.2700193 -0.002116207 -0.82227703  0.17186646 -0.7011655
## 2   -0.7058166  0.267318878  0.85453642  0.09526835  0.3302527
## 3    0.8071680  0.117744777  0.08714985  0.03420084  1.0519885
## 4    1.0685843 -0.552677783 -0.22460095 -0.47165324 -0.2275398
##
## Clustering vector:
##      SEBRLE        CLAY      KARPOV     BERNARD      YURKOV
##           2           2           2           2           3
##     WARNERS   ZSIVOCZKY     McMULLEN   MARTINEAU       HERNU
##           2           1           1           4           4
##      BARRAS        NOOL BOURGUIGNON      Sebrle        Clay
##           4           4           4           2           2
##      Karpov       Macey     Warners  Zsivoczky       Hernu
##           2           1           2           1           1
##        Nool     Bernard    Schwarzl   Pogorelov  Schoenbeck
##           2           1           2           2           2
##      Barras       Smith   Averyanov    Ojaniemi     Smirnov
##           1           1           1           1           1
##          Qi       Drews Parkhomenko       Terek       Gomez
##           1           2           3           3           1
##        Turi     Lorenzo   Karlivans Korkizoglou       Uldal
##           3           4           4           3           4
##     Casarsa
##           3
##
## Within cluster sum of squares by cluster:
## [1]  62.59352 112.04308  46.25851  39.51395
##  (between_SS / total_SS =  34.9 %)
##
## Available components:
##
## [1] "cluster"      "centers"      "totss"        "withinss"
## [5] "tot.withinss" "betweenss"    "size"         "iter"
## [9] "ifault"
```

Step 3: Characterize Clusters

The resulting clusters from K-Means are added to the 'decath' dataset. Then, the clusters are characterized using the `catdes` function from the 'FactoMineR' package.

```r
library(FactoMineR)

decath.comp <- cbind.data.frame(decath, factor(results.kmeans$cluster))
colnames(decath.comp)[14] <- "Cluster"

catdes(decath.comp, num.var = 14)
```

```
##
## Link between the cluster variable and the quantitative variables
## ================================================================
##                   Eta2      P-value
```

```
## Points       0.5682279 6.798190e-07
## Long.jump    0.5351738 2.589336e-06
## X110m.hurdle 0.5241604 3.955754e-06
## X100m        0.5237003 4.025499e-06
## Pole.vault   0.4865551 1.562107e-05
## X400m        0.4816657 1.853023e-05
## Rank         0.4204522 1.375075e-04
## X1500m       0.3743110 5.388999e-04
## Shot.put     0.2627348 9.637538e-03
##
## Description of each cluster by quantitative variables
## =====================================================
## $`1`
##               v.test Mean in category Overall mean
## X1500m     -3.059181       270.840000   279.024878
## Pole.vault -3.587590         4.533846     4.762439
##            sd in category Overall sd      p.value
## X1500m          5.6646475 11.5300118 0.0022194330
## Pole.vault      0.1749522  0.2745887 0.0003337491
##
## $`2`
##                 v.test Mean in category Overall mean
## Long.jump     3.986631         7.533571     7.260000
## Pole.vault    3.940076         5.000000     4.762439
## Points        3.525256      8267.142857  8005.365854
## X400m        -2.560828        48.975714    49.616341
## X100m        -2.645371        10.847143    10.998049
## X110m.hurdle -3.254362        14.272857    14.605854
## Rank         -3.439715         6.214286    12.121951
##              sd in category  Overall sd      p.value
## Long.jump         0.2229544   0.3125193 6.701805e-05
## Pole.vault        0.1836145   0.2745887 8.145567e-05
## Points          301.8114021 338.1839416 4.230740e-04
## X400m             0.9761273   1.1392975 1.044232e-02
## X100m             0.1973550   0.2597956 8.160134e-03
## X110m.hurdle      0.3116873   0.4660000 1.136472e-03
## Rank              5.0310262   7.8217805 5.823280e-04
##
## $`3`
##                 v.test Mean in category Overall mean
## X400m         3.582494        51.175000     49.61634
## Rank          2.804845        20.500000     12.12195
## X1500m        2.788974       291.305000    279.02488
## X110m.hurdle  2.139920        14.986667     14.60585
## Long.jump    -3.156109         6.883333      7.26000
##              sd in category Overall sd      p.value
## X400m             1.1211861  1.1392975 0.0003403298
## Rank              7.4554231  7.8217805 0.0050340788
## X1500m           13.4583899 11.5300118 0.0052875244
## X110m.hurdle      0.3708399  0.4660000 0.0323612562
## Long.jump         0.1814448  0.3125193 0.0015988911
##
## $`4`
##                 v.test Mean in category Overall mean
```

```
## X100m         4.053791         11.33625    10.998049
## X110m.hurdle  3.368905         15.11000    14.605854
## High.jump    -2.412939          1.90875     1.976829
## Shot.put     -3.134116         13.65750    14.477073
## Points       -3.643979       7609.62500  8005.365854
##              sd in category   Overall sd      p.value
## X100m            0.14194519   0.25979560 5.039416e-05
## X110m.hurdle     0.32939338   0.46599998 7.546748e-04
## High.jump        0.05464373   0.08785906 1.582447e-02
## Shot.put         0.60373318   0.81431175 1.723728e-03
## Points         143.94611622 338.18394159 2.684548e-04
```

The `catdes` function displays information about the clusters, indicating whether cluster means significantly differ from the overall mean (>2 indicates significant difference). Additionally, it interprets the positive and negative signs in the v test results, indicating if the cluster mean is superior or inferior compared to other clusters.

This analysis helps in understanding how the variables in the 'decathlon' dataset contribute to the formation of clusters and how distinct these clusters are from each other.

Conclusion

Clustering is a powerful unsupervised learning technique used for various applications, although it comes with its set of assumptions, types, challenges, and evaluation methods. Understanding these aspects is crucial for effective implementation in data analysis and machine learning tasks.