Sosyal Bilimlerde Araştırma Yöntemleri Özel Sayısı

Cilt: 22 Sayı: Özel Sayı 2 Yıl: 2022
e-ISSN: 2667-8683

AÜSBD
Anadolu Üniversitesi Sosyal Bilimler Dergisi
Anadolu University Journal of Social Sciences

R Programlama Dili ile Kümeleme Analizi [1]

Cem GÜRLER [2]

Başvuru Tarihi: 12.11.2022 **Kabul Tarihi:** 29.12.2022 **Makale Türü:** Araştırma Makalesi

Öz

Kümeleme analizi sıklıkla kullanılan, temelde, birbirine benzeyen gözlemleri bir araya gruplamayı amaçlayan çok değişkenli bir istatistik yöntemidir. Kümeleme analizi, hiyerarşik ve hiyerarşik olmayan algoritmalar şeklinde iki ana başlık altında toplanabilir. Bu iki başlık arasındaki farklardan biri, hiyerarşik olmayan algoritmaların, analiz öncesinde küme sayısına ihtiyaç duymasıdır. Ayrıca, hiyerarşik algoritmalarla oluşan küme üyelikleri nihaidir ve değişmezler. Hiyerarşik olmayan algoritmalarda ise, küme üyelikleri, sabit kalana kadar değişmektedir. İstatistiksel yöntemlerde, özellikle son yıllarda açık kaynak kodlu programların ve programlama dillerinin kullanımı yaygınlaşmıştır. Mevcut çalışmada, R programlama dili kullanılarak, hiyerarşik ve hiyerarşik olmayan kümeleme algoritmalarına yönelik uygulamaların gösterilmesi amaçlanmıştır. Ayrıca, kümeleme analizi öncesinde küme sayısının nasıl belirlenebileceği de R programlamayla gösterilmiştir. Küme sayısının belirlenmesi için literatürde sıklıkla kullanılan Elbow, ortalama Silhouette ve GAP istatistiği yöntemleri kullanılmıştır. Çalışmada analizler için factoextra() ve cluster() paketleri kullanılmıştır. Ayrıca çalışmada kullanılan kodların ve görsellerin gösterimi RMarkdown'da üretilmiştir. Kümeleme sonuçlarının nasıl yorumlandığının gösterimi için k-ortalamalar sonucunda oluşan kümeler yorumlanmıştır.

Anahtar Kelimeler: *Hiyerarşik Kümeleme, Hiyerarşik Olmayan Kümeleme, K-Ortalamalar, R Programlama*

Atıf: Güler, C. (2022). R programlama dili ile kümeleme analizi. *Anadolu Üniversitesi Sosyal Bilimler Dergisi, 22*(Özel Sayı 2), 341-366.

[1] Bu çalışma etik kurul izin belgesi gerektirmemektedir.

[2] Yalova Üniversitesi İktisadi ve İdari Bilimler Fakültesi İşletme Bölümü, **cem.gurler@yalova.edu.tr,** ORCID: 0000-0001-5127-6726

Sosyal Bilimlerde Araştırma Yöntemleri Özel Sayısı

Cilt: 22 Sayı: Özel Sayı 2 Yıl: 2022
e-ISSN: 2667-8683

AÜSBD
Anadolu Üniversitesi Sosyal Bilimler Dergisi
Anadolu University Journal of Social Sciences

Cluster Analysis with R Programming

Cem GÜRLER [3]

Submitted by: 12.11.2022 **Accepted by:** 29.12.2022 **Article Type:** Research Article

Abstract

Cluster analysis is a frequently used multivariate statistical method that basically aims to group similar observations together. Cluster analysis can be categorized under two main headings: hierarchical and non-hierarchical algorithms. One of the differences between these two categories is that non-hierarchical algorithms require the number of clusters before the analysis. Also, the cluster memberships formed by hierarchical algorithms are stable. In non-hierarchical algorithms, on the other hand, cluster memberships change until they remain constant. In statistical methods, especially in recent years, the use of open source programs and programming languages has become widespread. The present study aims to demonstrate applications for hierarchical and non-hierarchical clustering algorithms using the R programming language. In addition, how to determine the number of clusters before clustering analysis is also demonstrated with R programming. Elbow, mean Silhouette and GAP statistic methods, which are frequently used in the literature, were used to determine the number of clusters. Factoextra() and cluster() packages were used in the study. In addition, the representation of the codes and visuals used in the study were generated in RMarkdown.

Keywords: *Hierarchical Clustering, Non-Hierarchical Clustering, K-Means, R Programming*

[3] Yalova University Faculty of Economics and Administrative Science Department of Business Administration, **cem.gurler@yalova.edu.tr,** ORCID: 0000-0001-5127-6726

Giriş

Kümeleme analizi pek çok farklı alanda sıklıkla kullanılan çok değişkenli istatistiksel tekniklerden biridir. Yine çok değişkenli bir istatistiksel teknik olan faktör analizine benzer şekilde, kümeleme analizinde bağımlı bağımsız değişken ayrımı yapmamaktadır (Kalaycı, 2016). Faktör analizinden farkı ise, birbirlerine benzeyen gözlem değerlerini gruplamasıdır. Faktör analizi ise boyut azaltmak için değişkenleri gruplayan çok değişkenli bir istatistik tekniğidir. Kümeleme analizi 3 adımda özetlenebilir (Hair, Black, Babin ve Anderson, 2010):

1. Küme sayısının belirlenmesi
2. Kümeleme analizinin gerçekleştirilmesi
3. Oluşan kümelerin profillerinin oluşturulması

Kümeleme analizinin varsayımlar açısından, diğer istatistiksel yöntemlere kıyasla daha esnek olduğu ifade edilebilir. Bir başka ifadeyle, normallik, doğrusallık ve eş varyanslılık varsayımlarına karşı tutumu, diğer istatistiksel yöntemler kadar katı değildir. Bunun yanında kümeleme analizinde kullanılan örneklem, evreni iyi bir şekilde temsil etmelidir. Ayrıca veride çoklu doğrusal bağlantı sorunu olmamalıdır. Kümeleme analizinde önemli sorunlardan birisi de standardizasyondur. Değişkenlerin birimleri arasındaki farklılıklar, kümeleme algoritmalarında kullanılan hesaplamaları etkileyebilmektedir. Örneğin bir değişkenin aralığı 0-1 arasındayken bir başka değişken 1-1000 aralığında değer alabilmektedir. Değişkenlerin birimleri arasındaki bu farklılıklar, kümeleme sonuçlarını önemli ölçüde etkilemektedir. Bu bağlamda, kümeleme analizine geçilmeden önce, veri setinde yer alan değişkenlerin özelliklerine göre standardizasyon yapılması, değişkenlerin birimlerden arındırılması gerekebilmektedir (Wu, Milton, Hammond ve Spear, 1999; Mohamad ve Usman, 2013). Özellikle, Değişkenlerin birimleri arasındaki farklılıklardan ve büyüklükten oldukça etkilenen ve kümeleme analizinde sıklıkla kullanılan Öklid uzaklığı gibi uzaklık ölçütleri için standardizasyon gereklidir. Standardizasyondaki temel amaç, girdi değişkenlerinin her biri için eşit ağırlık elde etmektir (Milligan ve Cooper, 1988). Ancak, bazı durumlarda (değişkenlerin birimlerinin aynı olması gibi) veri setine standardizasyon uygulanmasına gerek yoktur. Ayrıca büyük katkıları olan değişkenlerin standardize edilmesi, bu büyük etkileri azaltarak kümeleme performansını zayıflatmaktadır (Kaufman ve Rousseeuw, 2009).

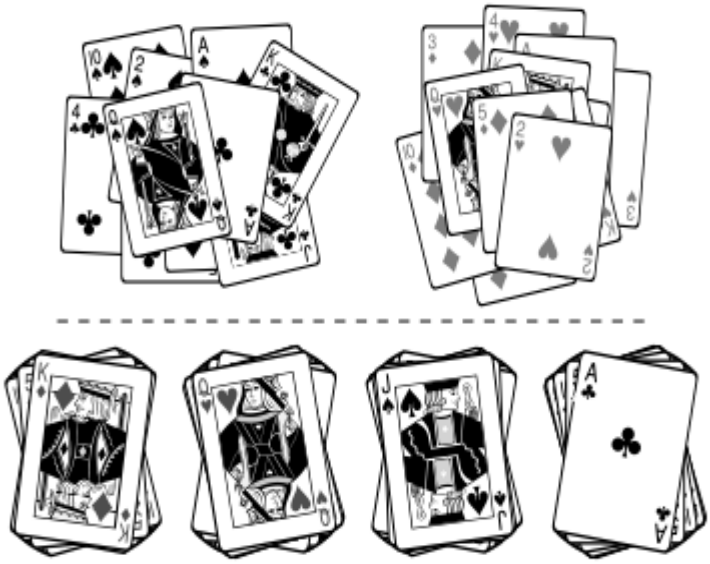
Kümeleme algoritmaları, farklı sınıflandırmalar olmasına rağmen, hiyerarşik ve hiyerarşik olmayan olmak üzere iki başlık altında toplanabilir. Bu iki başlık arasında temel fark, hiyerarşik olmayan kümeleme algoritmalarında, küme sayısının önceden belirlenmesine ihtiyaç duyulmasıdır. Çalışmanın ikinci bölümünde küme sayısının nasıl belirlenebileceği açıklanmış ve sonrasında hiyerarşik ve hiyerarşik olmayan yöntemler ve uygulamaları paylaşılmıştır. Son olarak ise kümeleme analizi sonuçlarının geçerliği incelenmiştir. Mevcut çalışmada yapılacak uygulamalar, R'da bulunan USArrests veri seti kullanılarak gerçekleştirilmiştir. Bu veri seti, 1973 yılında Amerika Birleşik Devletleri'nin 50 eyaletinin her birinde 100.000 kişi başına düşen saldırı, cinayet ve tecavüz nedeniyle tutuklanma istatistiklerini içerir. Ayrıca, kentsel alanlarda yaşayan nüfusun yüzdesi de yer almaktadır. Buradaki amaç, benzer suçların yaşandığı eyaletleri kümelemektir. Böylelikle eyalet yöneticilerinin, aynı kümede yer aldıkları eyaletlerle ortak projeler yürütebilmesine, sorunların çözümüne yönelik birlikte hareket edilmesine olanak sağlanmaktadır.

Küme Sayısının Belirlenmesi

Kümeleme analizinde en önemli sorunlardan biri olarak küme sayısına karar vermek ifade edilebilir. Bu sorunun üstesinden gelebilmek için Elbow (Dirsek), ortalama Silhouette ve GAP istatistiği gibi yöntemler mevcuttur. Elbow yöntemi kümeler tarafından açıklanan varyans yüzdesinin, ilgili küme sayısına karşı çizilmesiyle oluşan grafik şeklinde tanımlanabilir (Bholowalia ve Kumar, 2014). Ortalama Silhouette yönteminde, her bir veri için Silhouette değeri hesaplanmakta ve sonrasında ortalama alınarak, optimal küme sayısı belirlenmektedir (Nanjundan vd., 2019). GAP istatistiği, alternatif modelleri (küme sayısı en az 2),

uniform dağılıma uygun rastgele oluşturulmuş ve küme sayısının 1 olduğu modelle karşılaştırarak küme sayısını belirlemektedir (Önder, 2020). Ortalama Silhouette ve GAP istatistiği yöntemleriyle, küme sayısı net bir şekilde belirlenmektedir. Bir başka ifadeyle bu iki yöntem küme sayısının kaç olması gerektiğini tespit etmektedir. Elbow yöntemi ise küme sayısının kararını, karar vericiye bırakmaktadır. Küme sayısının belirlenmesi için hangi yöntemin kullanılacağına yönelik bir fikir birliği yoktur.

Küme sayısını belirlemek, kümeleme analizinin uygulanabilmesi ve sonuçlarının yorumlanabilmesi açısından oldukça önemlidir. Şekil 1, küme sayısının önemini göstermektedir. Küme sayısının 2 olarak belirlendiği durumda kartlar renklerine göre kümelenirken küme sayısı 4 olduğu durumda kartlar türlerine göre ayrılmaktadır.



Kaynak: Berry, M. J., & Linoff, G. S. (2004). *Data mining techniques: for marketing, sales, and customer relationship management*. John Wiley & Sons.

Şekil 1. Küme Sayısının Önemini Gösteren İki Farklı Kümeleme

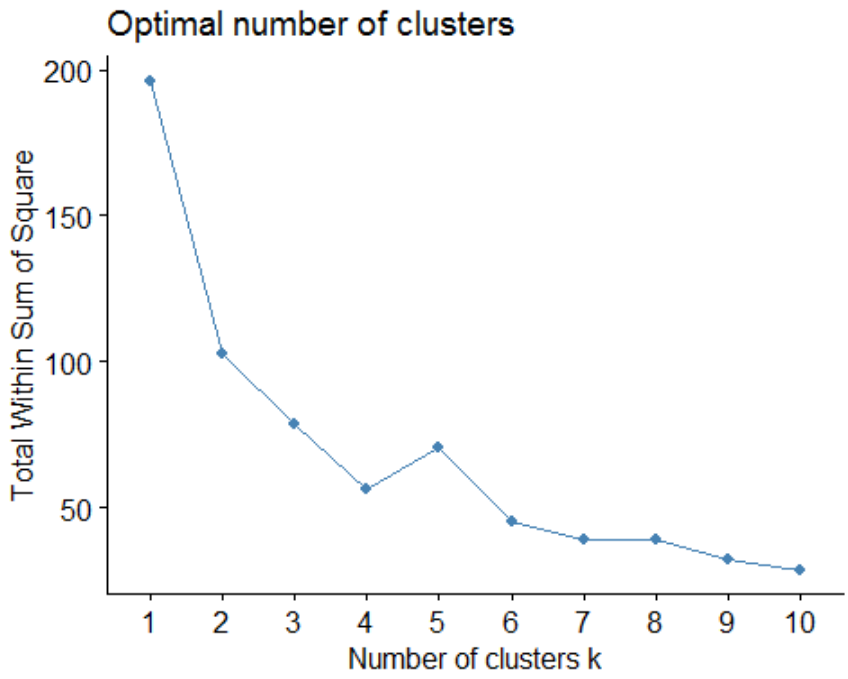
R Programlama Dili ile Küme Sayısının Belirlenmesi

Küme sayısının belirlenmesi için *factoextra* (Kassambara ve Mundt, 2017) paketinde yer alan *fviz_nbclust()* fonksiyonu kullanılmıştır. Kullanılan fonksiyonda 3 temel parametre kullanılarak küme sayısı belirlenebilmektedir:

1. Veri seti (çalışmada örnek olarak gösterilmek için, veri setine *scale()* fonksiyonu kullanılarak z-dönüşümü gerçekleştirilmiştir.)
2. Kümelemede kullanılacak yöntem (örnekte kmeans kullanılmıştır. Alternatif olarak k-medoids için "pam" ya da hiyerarşik kümeleme için "hcut" kullanılabilir.)
3. Küme sayısının belirlenmesi için kullanılacak yöntem (elbow yöntemi için "wss", Silhouette yöntemi için "silhouette" ve GAP istatistiği için "gap_stat")

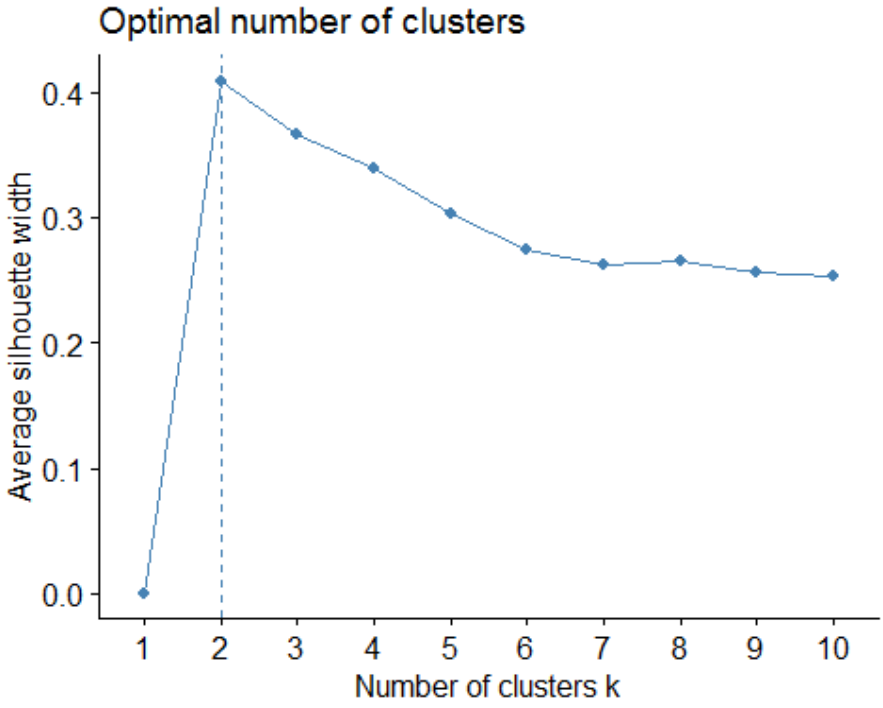
Aşağıda Elbow, ortalama Silhouette ve GAP istatistiği yöntemleri için hesaplanmış küme sayıları görülmektedir.

```
library(factoextra) #factoextra pakedini kullanmak için library fonksiyonu ile çağırıyoruz.

## Warning: package 'factoextra' was built under R version 4.1.3

## Zorunlu paket yükleniyor: ggplot2

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

fviz_nbclust(scale(USArrests), kmeans, method = "wss")
```



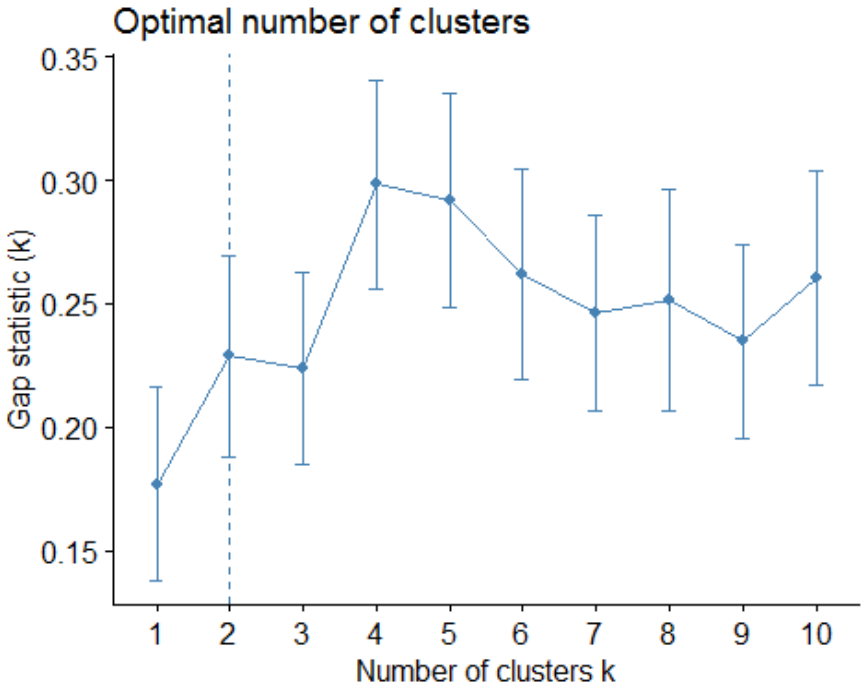
Şekil 2. Elbow Yönteminin Grafiği

```
fviz_nbclust(scale(USArrests), kmeans, method = "silhouette")
```



Şekil 3. Ortalama Silhouette Yönteminin Grafiği

```
fviz_nbclust(scale(USArrests), kmeans, method = "gap_stat")
```

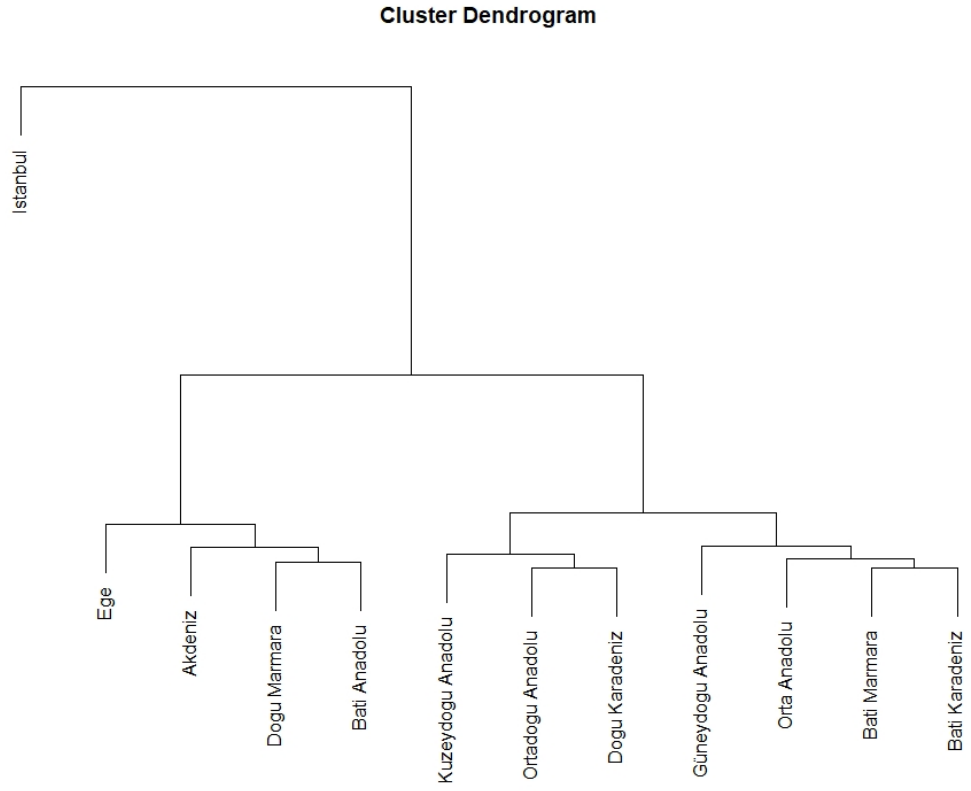


Şekil 4. GAP Istatistiği Yönteminin Grafiği

Ortalama Silhouette ve GAP istatistiği yöntemleri küme sayısını 2 olarak belirlerken, Elbow yöntemine göre küme sayısının 4 olduğu ifade edilebilir. Ortalama Silhouette ve GAP istatistiği yöntemleri karar vericiye bir seçenek sunmadan küme sayısının 2 olduğunu tespit etmiştir. Elbow yönteminde ise küme sayısı 4 olduktan sonra elde edilen marjinal kazanç oldukça azalmıştır. Bu nedenle Elbow yönteminde küme sayısı 4 olarak belirlenmiştir. Bu üç yöntem küme sayısının belirlenmesinde sıklıkla kullanılırken, küme sayısının kaç olacağının kararı karar vericiye göre değişmektedir. Araştırmacılar her üç yöntemle de küme sayısını belirledikten sonra, eğer en az iki yöntem aynı küme sayısını gösteriyorsa, karar verici buradan hareketle küme sayısını belirleyebilir. Ayrıca küme sayısına, ilerleyen başlıklarda bahsedilen geçerlik yöntemleri kullanılarak da karar verilebilir.

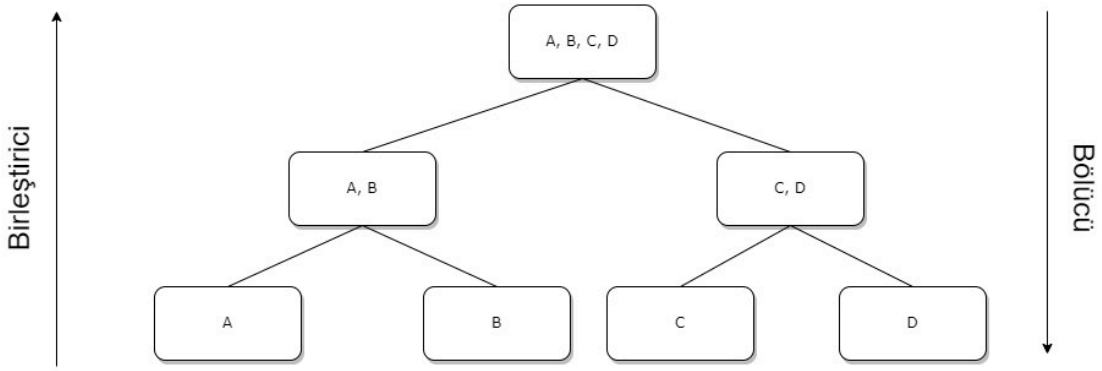
Hiyerarşik Kümeleme Algoritmaları

Hiyerarşik kümeleme yöntemlerinde veri seti, önceden belirlenmiş sayıda kümeye bölünmemektedir. Hiyerarşik kümeleme yöntemleri, veri setinde bulunan her bir gözlemi ayrı bir küme olarak kabul ederek başlamaktadır. Sonrasında tüm veri tek bir küme ile gösterilene kadar, her adımda kümelerden ikisi birleşmektedir. Böylelikle algoritmanın sonunda, veri setini temsil eden tek bir küme oluşmaktadır (Milligan ve Cooper, 1987). Hiyerarşik kümeleme algoritmalarında, birleşen kümeler nihaidir. Bir başka ifadeyle kümeyi oluşturan gözlemler, sonrasında başka bir kümede yer alamamaktadır. Bu algoritmaların tercih edilmesinde, bu sabitliğin etkisi büyüktür fakat aynı zamanda hatalı kümelemelerin düzeltilmesine de engel olmaktadır (Han, Pei ve Kamber, 2011). Hiyerarşik kümeleme algoritmalarının sonucunda, dendrogram adı verilen ağaç biçiminde bir görsel oluşmaktadır. Dendrogram, her yaprağın (leaf) bir veri noktasını temsil ettiği ve her bir dahili düğümün (node), alt yapraklarını içeren bir kümeyi temsil ettiği köklü (root) bir ağaçla temsil edilir (Cohen-Addad vd., 2019). Şekil 5'te dendrogram örneği görülmektedir.



Şekil 5. Dendrogram Örneği

Hiyerarşik kümeleme algoritmaları birleştirici (agglomerative) ve bölücü (divisive) olmak üzere iki başlık altında toplanabilir. Birleştirici ve bölücü yöntemler arasındaki fark Şekil 6'da görülmektedir. Kısaca, birleştirici yöntemler n sayıda gözlemi tek kümede birleştirirken, bölücü yöntemler tek bir kümeyi bölerek n sayıda gözleme ayırırlar.



Şekil 6. Birleştirici ve Bölücü Yöntemler

Birleştirici Algoritmalar

Literatürde ve uygulamada sıklıkla kullanılan algoritmalar, birleştirici algoritmalardır. Birleştirici algoritmalar, kümelenecek n sayıda nesneyi, art arda n kümeden daha az kümeye gruplandırır ve sonunda n sayıda nesneyi içeren tek bir küme oluştururlar (Day ve Edelsbrunner, 1984). Sıklıkla kullanılan birleştirici hiyerarşik kümeleme yöntemleri şunlardır: tek bağlantı (single linkage) / en yakın komşu yöntemi (nearest neighbor), tam bağlantı (complete linkage) / en uzak komşu yöntemi (furthest neighbor), ortalama bağlantı (average linkage), ağırlıklandırılmış ortalama bağlantı (weighted average linkage), ward yöntemi (ward's method), medyan bağlantı (median linkage) ve merkezi yöntem (centroid linkage) (Everitt vd., 2011). Bu yöntemlerin özellikleri Tablo 1'de yer almaktadır.

Tablo 1
Hiyerarşik kümeleme yöntemleri ve özellikleri

Yöntem	Genellikle benzerlik mi uzaklık mı?	Kümeler arası uzaklık	Açıklama
Tek bağlantı	Her ikisi de	İki gözlem arasındaki minimum mesafe	Özellikle büyük veri kümelerinde, dengesiz ve dağınık kümeler oluşturma eğilimindedir. Küme yapısını dikkate almaz.
Tam bağlantı	Her ikisi de	İki gözlem arasındaki maksimum mesafe	Eşit çaplarda (nesneler arasındaki maksimum mesafe) kümeler bulma eğilimindedir. Küme yapısını dikkate almaz.
Ortalama bağlantı	Her ikisi de	İki gözlem arasındaki ortalama mesafe	Kümeleri küçük varyanslarla birleştirme eğilimindedir. Küme yapısını dikkate alır. Robust olarak değerlendirilebilir.
Merkezi yöntem	Uzaklık (işlenmemiş veriye gerekli)	Ortalama vektörler arasındaki kare Öklid mesafesi	Noktaların öklid uzayında temsil edilebileceğini varsayar. Kümelenmiş iki gruptan daha fazla sayıda olan, birleştirilmiş kümede daha baskındır.
Ağırlıklandırılmış ortalama bağlantı	Her ikisi de	İki gözlem arasındaki ortalama mesafe	Ortalama bağlantıdan farklı olarak küçük kümelerdeki noktalar, büyük kümelerdeki noktalara göre daha yüksek ağırlıklıdır (küme boyutlarının eşit olmadığı durumlarda daha iyi sonuç verir).
Medyan bağlantı	Uzaklık (işlenmemiş veriye gerekli)	Ağırlıklı merkezler arasındaki kare öklid mesafesi	Noktaların öklid uzayında temsil edilebileceğini varsayar. Oluşan yeni grup, birleştirilmiş grupların ortasında konumlanmaktadır.
Ward yöntemi	Uzaklık (işlenmemiş veriye gerekli)	Tüm değişkenler üzerinden toplanan kümeler içindeki kareler toplamındaki artış	Noktaların öklid uzayında temsil edilebileceğini varsayar. Aynı boyutta, kümeler bulma eğilimindedir. Aykırı değerlere karşı hassastır.

Kaynak: Everitt, B. S., Landau, S., Leese, M., & Stahl, D. (2011). Cluster Analysis (5th edition). Chichester, UK: John Wiley & Sons, Ltd.

R Programlama Dili ile Hiyerarşik Kümeleme Uygulaması

Hiyerarşik kümeleme analizi gerçekleştirilmesi için ilk olarak uzaklık ölçütü kullanılarak uzaklıklar hesaplanmalıdır. Bu bağlamda ilk olarak *dist()* fonksiyonunu kullanarak uzaklıklar hesaplanmalıdır. Gerekli kod aşağıda yer almaktadır:

```
uzaklık <- dist(x = USArrests, method = 'euclidean') # Burada dist fonksiyonu ile uzaklık hesaplanarak, uzaklık ismiyle kaydedilmiştir.
```

Bu örnekte uzaklık ölçütü olarak Öklid uzaklığı kullanılmıştır. Alternatif uzaklıklar için method kısmına şu ölçütlerden birisini yazabilirsiniz: "euclidean", "maximum", "manhattan", "canberra", "binary" ya da "minkowski"

Uzaklık ölçütlerinin hesaplanmasından sonra, hiyerarşik kümeleme analizine geçebiliriz.

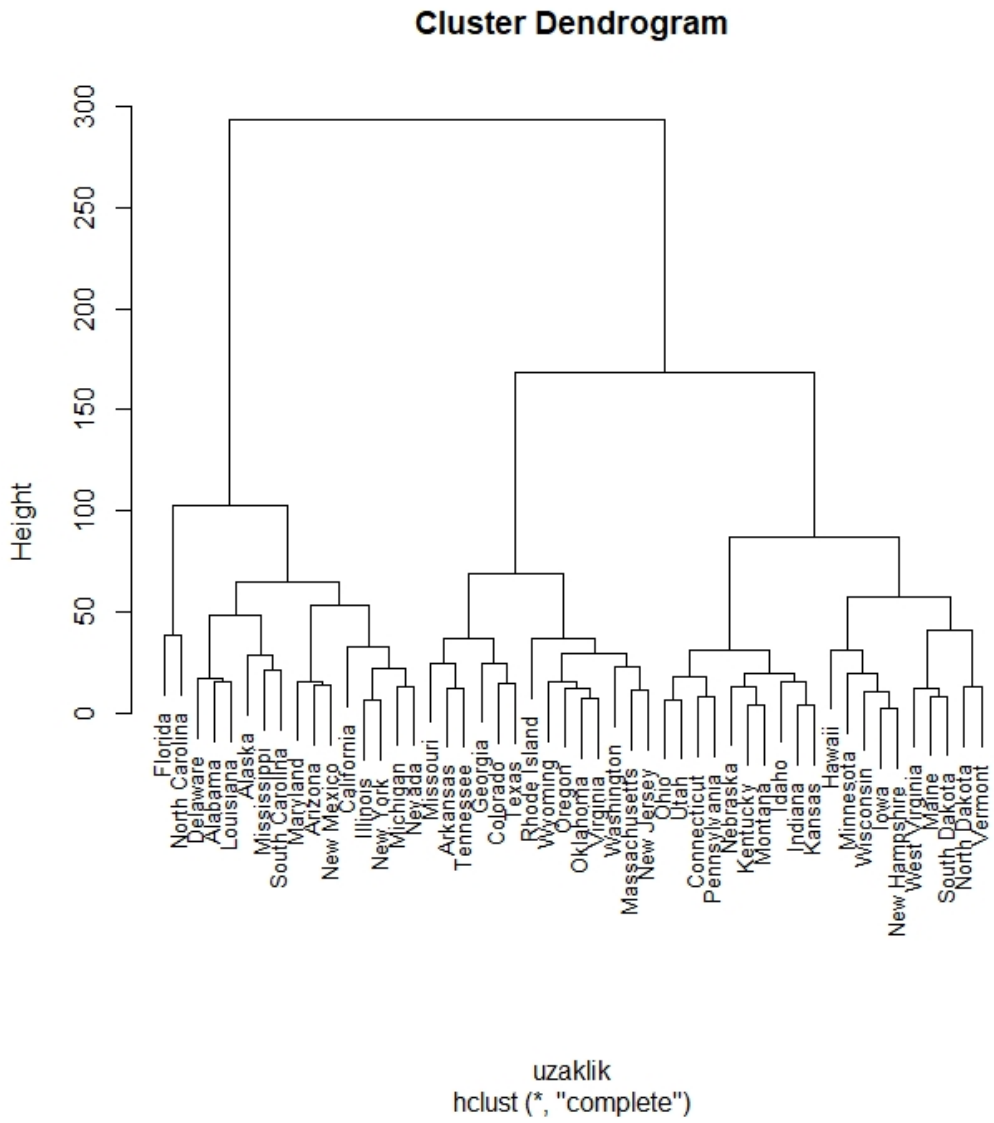
```
hiyerarsik <- hclust(uzaklık, method = 'complete') # hclust fonksiyonu ile kümeleme analizi yapılmış ve hiyerarsik ismiyle kaydedilmiştir.
```

Hiyerarşik kümeleme analizi, tam bağlantı (complete linkage) algoritması ile gerçekleştirilmiştir. Alternatif algoritmalar için method kısmına şu algoritmalardan birisini yazabilirsiniz: "ward.D", "ward.D2", "single", "complete", "average" (= UPGMA), "mcquitty" (= WPGMA), "median" (= WPGMC) ya da "centroid" (= UPGMC).

Kümeleme analizi yukarda yapılmış ve hiyerarşik ismi ile kaydedilmiştir. Şimdi plot() fonksiyonu kullanılarak kümeleme sonuçlarını dendrogram şeklinde gösterelim.

```
plot(hiyerarşik)
```

Kodun çalıştırılmasıyla birlikte elde edilecek olan dendrogram aşağıda yer almaktadır.

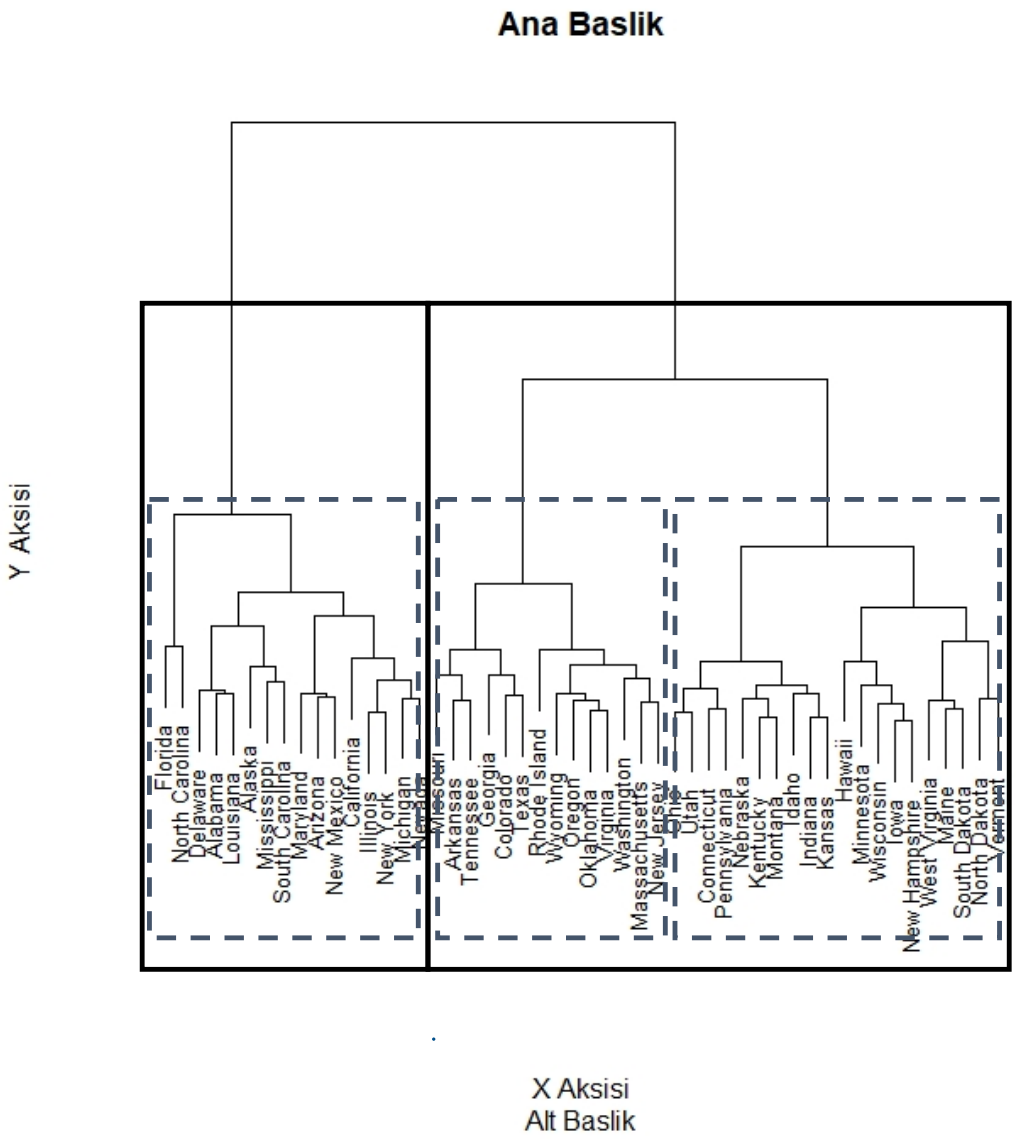


Şekil 7. Hiyerarşik kümeleme sonucu oluşan dendrogram

Dendrogramda aksis isimleri, aksisler, dendrogramın ismi, eyaletlerin yazı fontu gibi elementler kolaylıkla manipüle edilebilir. Aşağıdaki kodu kullanarak, Şekil X'teki dendrogramı manipüle edelim.

```
plot(hiyerarsik, cex = 0.8, main= "", xlab="", ylab="", sub="", axes=FALSE)
```

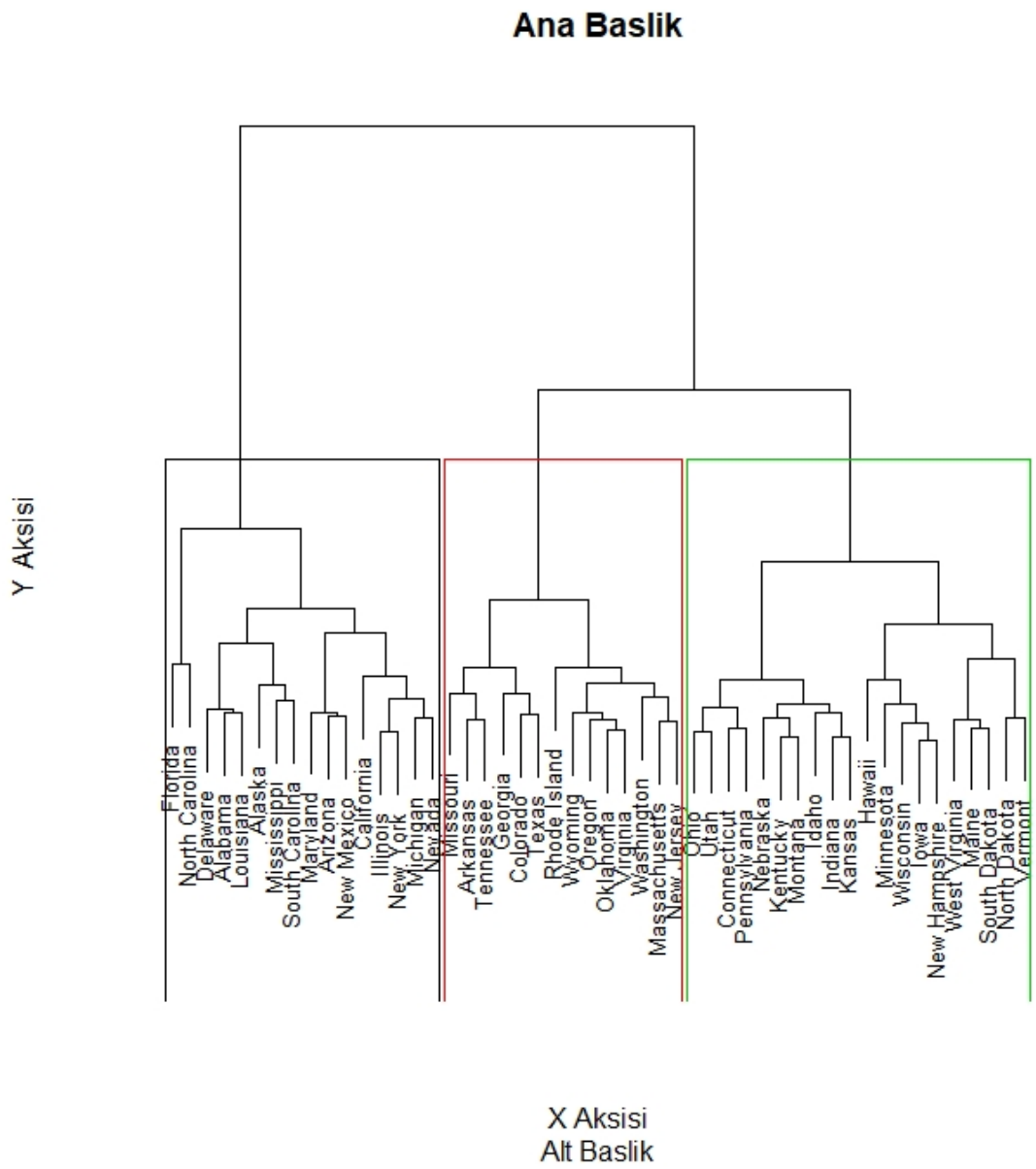
Bu kodda cex ile eyalet isimlerinin yazı fontu, main ile şeklin başlığı, xlab ile x aksisinin adı, ylab ile y aksisinin adı, sub ile alt başlık (hclust(*, "complete") yazan yer değiştirilmekte, axes = FALSE komutu ile aksislerin yer almaması sağlanmaktadır. Yukarıdaki kod çalıştırıldığında elde edilecek dendrogram aşağıda yer almaktadır.



Şekil 8. Hiyerarşik Kümeleme Sonucu Oluşan Dendrogramın Düzenlenmiş Hali

Dendrograma bakılırsa, 50 eyaletin 2 ya da 3 kümede gruplanabileceği söylenebilir. Oluşan dendrogram üzerinde, 2 veya 3 küme şeklinde bir görselleştirme yapılmak istenirse, aşağıdaki kod kullanılabilir. Kodun hemen altında ise, çalıştırılan kodun ürettiği dendrogram görülmektedir. Burada dendrograma bakılarak küme sayısı belirlenebileceği gibi, Silhouette endeksi, GAP istatistiği gibi yöntemler kullanılarak da küme sayısına karar verilebilir.

```
hiyerarsik3lu <- cutree(hiyerarsik, k = 3) # hiyerarşik, 3 kümeye ayrılmıştır.
hiyerarsik3lu # Her bir eyaletin, hangi kümede yer aldığını görebilmek için bu kod çalıştırılmalıdır.
##    Alabama     Alaska    Arizona   Arkansas California
##          1          1          1          2          1
##   Colorado Connecticut   Delaware    Florida    Georgia
##          2          3          1          1          2
##     Hawaii      Idaho   Illinois    Indiana       Iowa
##          3          3          1          3          3
##     Kansas   Kentucky  Louisiana      Maine   Maryland
##          3          3          1          3          1
## Massachusetts   Michigan  Minnesota Mississippi   Missouri
##          2          1          3          1          2
##    Montana   Nebraska     Nevada New Hampshire New Jersey
##          3          3          1          3          2
##  New Mexico   New York North Carolina North Dakota       Ohio
##          1          1          1          3          3
##   Oklahoma     Oregon Pennsylvania Rhode Island South Carolina
##          2          2          3          2          1
##  South Dakota  Tennessee      Texas       Utah    Vermont
##          3          2          2          3          3
##    Virginia Washington West Virginia  Wisconsin    Wyoming
##           2          2          3          3          2
rect.hclust(hiyerarsik, k = 3, border = 1:3) # Border 1:3 ile, 3 küme farklı renklendirilmiştir. Tek bir renk istenirse, istediğiniz rengin İngilizcesini yazabilirsiniz. Örneğin, border = "blue" gibi.
```



Şekil 9. Hiyerarşik Kümeleme Sonucu Oluşan Dendrogramın Düzenlenmiş ve Üç Kümeye Ayrılmış Hali

Hiyerarşik Olmayan Kümeleme Analizi

Hiyerarşik olmayan kümeleme analizinin, hiyerarşik yöntemlerden temel farkı, küme sayısının belirlenmesi gerekliliğidir. Bir başka ifadeyle, hiyerarşik olmayan yöntemlerin uygulanabilmesi için küme sayısı analiz öncesinde belirlenmelidir. Hiyerarşik olmayan kümeleme yöntemleri, n sayıda gözlemi, önceden belirlenmiş k sayıda kümeye ayırmaktadır. Bu bölümde k-ortalamalar ve k-medoids yöntemleri ve R uygulamaları paylaşılacaktır.

K-ortalamalar

K-ortalamalar, basit ve hızlı çalıştığı için literatürde sıklıkla kullanılan kümeleme algoritmalarındandır. K-ortalama algoritması veri setini k sayıda kümeye bölmekte ve her bir kümeyi centroid (küme merkezi) ile temsil etmektedir. Algoritma, veri ve centroidler arasındaki kare mesafeleri kullanarak, verileri kendilerine en yakın centroide atamaktadır (Žalik, 2008). K-ortalamalar algoritması 4 adımda özetlenebilir (Dehariya, Shrivastava ve Jain, 2010):

1. k sayıda gözlem, centroid olarak belirlenir.
2. Geri kalan gözlemler, kendilerine en yakın centroidin olduğu kümeye atanır.
3. Tüm gözlemler atandıktan sonra, k sayıda centroid tekrar hesaplanır.
4. Centroidler sabit kalana kadar, 2. ve 3. adımlar tekrar edilir.

K-ortalamalar algoritmasının dezavantajları ise şöyle sıralanabilir (Celebi, Kingravi ve Vela, 2013):

1. Yalnızca iyi ayrılmış, kompakt kümeleri ayırmada iyi olduğu söylenebilir.
2. Gürültülü veriden ve aykırı değerlerden aşırı etkilenmektedir.
3. İlk adımda belirlenen centroidlere karşı da oldukça duyarlıdır.

R Programlama Dili ile K-ortalamalar uygulaması

K-ortalamalar için ilk olarak *kmeans()* fonksiyonu kullanılarak kümeleme analizi gerçekleştirilecektir. *kmeans()* fonksiyonun çalışması için, mevcut örnekte 3 parametre kullanılmıştır:

1. Veri seti (çalışmada örnek olarak gösterilmek için, veri setine scale() fonksiyonu kullanılarak z-dönüşümü gerçekleştirilmiştir. Z-dönüşümü fonksiyon içinde yapılmayıp, yeni bir veri seti olarak da kaydedilebilmektedir.)
2. Küme sayısının girilmesi gerekmektedir. Mevcut örnekte Elbow yöntemi kullanılarak küme sayısı 4 olarak belirlenmiştir.
3. nstart çalışmada 20 olarak belirlenmiştir. Algoritma 20 farklı başlangıç centroidi belirlemekte ve içlerinde en iyisini seçmektedir. nstart'ın kaç olacağına dair genel bir sayı mevcut değildir. Paket içeriğinde default olarak belirlenmiş değer 1'dir. Fakat sadece 1 başlangıçla algoritmanın çalıştırılması tavsiye edilmemektedir. Çünkü algoritma farklı denemeler yapmadan, ilk yaptığı atamanın en iyi sonucu verdiğini varsayarak kümelemeyi gerçekleştirmektedir. En iyi sonuç için farklı nstart değerleri denenmelidir. Literatürde genellikle uygulanan ve tavsiye edilen nstart değerleri 20 ve 25'tir.

```
kortalama <- kmeans(scale(USArrests), centers = 4, nstart = 20)
```

kmeans() fonksiyonu çalıştırılmış ve sonuçlar "kortalama" adıyla kaydedilmiştir. Sonrasında *print(kortalama)* fonksiyonuyla sonuçlar görüntülenmiştir. Küme sayılarının 8, 13, 16 ve 13 olduğu görülmektedir. Küme sayılarından sonra, veri setindeki değişkenlerin her bir kümedeki ortalama değerleri yer almaktadır. Sonrasında ise küme iyelikleri görülmektedir. Küme iyeliklerinden sonra yer alan değerler ne kadar küçükse, o kümenin o kadar kompakt olduğu ifade edilebilir. Bir başka ifadeyle, within cluster sum of squares (WCSS) değeri arttıkça küme içi değişkenliğin de arttığı söylenebilir. WCSS değeri, küme büyüklüğünden önemli ölçüde etkilenmektedir. Son olarak "Available components" başlığı altında, görüntülenebilecek küme özellikleri bulunmaktadır (*kortalama$cluster* kodu çalıştırılarak örneğin küme üyelikleri yazdırılabilir).

```
print(kortalama)
```

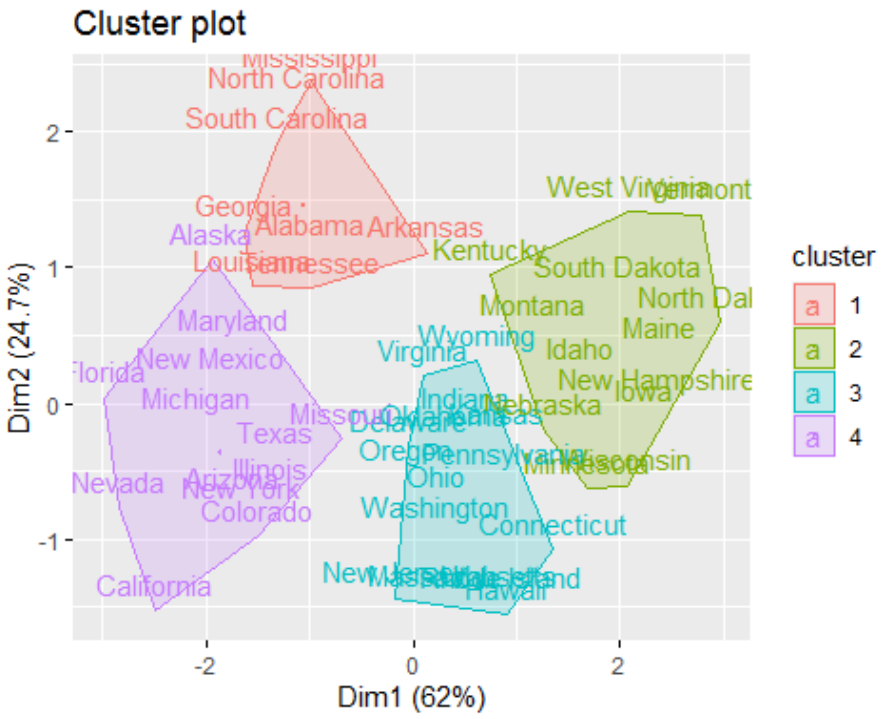
```
## K-means clustering with 4 clusters of sizes 8, 13, 16, 13
##
## Cluster means:
##   Murder  Assault  UrbanPop    Rape
## 1  1.4118898  0.8743346 -0.8145211  0.01927104
## 2 -0.9615407 -1.1066010 -0.9301069 -0.96676331
## 3 -0.4894375 -0.3826001  0.5758298 -0.26165379
## 4  0.6950701  1.0394414  0.7226370  1.27693964
##
## Clustering vector:
##    Alabama     Alaska    Arizona   Arkansas California
##          1          4          4          1          4
##   Colorado Connecticut   Delaware    Florida    Georgia
##          4          3          3          4          1
##     Hawaii      Idaho   Illinois    Indiana       Iowa
##          3          2          4          3          2
##     Kansas   Kentucky  Louisiana      Maine   Maryland
##          3          2          1          2          4
## Massachusetts   Michigan  Minnesota Mississippi   Missouri
##          3          4          2          1          4
##    Montana   Nebraska     Nevada New Hampshire New Jersey
##          2          2          4          2          3
## New Mexico   New York North Carolina North Dakota       Ohio
##          4          4          1          2          3
##   Oklahoma     Oregon Pennsylvania Rhode Island South Carolina
##          3          3          3          3          1
## South Dakota  Tennessee      Texas       Utah    Vermont
##          2          1          4          3          2
##   Virginia Washington West Virginia  Wisconsin    Wyoming
##          3          3          2          2          3
##
## Within cluster sum of squares by cluster:
## [1]  8.316061 11.952463 16.212213 19.922437
##  (between_SS / total_SS =  71.2 %)
##
## Available components:
##
## [1] "cluster"   "centers"   "totss"   "withinss"   "tot.withinss"
## [6] "betweenss"   "size"   "iter"   "ifault"
```

Kümeleme sonuçları yazdırıldıktan sonra, oluşan kümelerin görselleştirilmesi için *fviz_cluster()* fonksiyonu kullanılmıştır:

1. Kümeleme analizinin sonuçlarının yer aldığı veri
2. Kümeleme analizinde kullanılan veri

3. Kümelerde yer alan gözlemlerin gösterim şekli (örnekte “text” kullanılarak her bir gözlemin adı görselde yer almaktadır. “point” kullanılırsa eğer nokta olarak da gösterilebilir).

```
fviz_cluster(kortalama, USArrests, geom = "text")
```



Şekil 10. K-Ortalamalar Sonucu Oluşan Kümeler

K-ortalamalar sonucunda oluşan kümeleri yorumlayabilmek için “Cluster Means” başlıklı tablodan yararlanılabilir. Küme 1, kişi başına düşen saldırı değişkeninde en yüksek ortalamaya sahip olan kümedir. Kişi başına düşen saldırı, tecavüz ve kentsel alanlarda yaşayan nüfusun yüzdesi ortalamalarında en yüksek ortalamaya Küme 4 sahiptir. Böylelikle, Küme 4’ün, diğer kümelere göre daha tehlikeli olduğu sonucuna ulaşılabilir. Ayrıca, aynı kümenin kentsel alanlarda yaşayan nüfusun yüzdesi değişkeninde de en yüksek ortalamaya sahip olması nedeniyle, bu bölgede yaşayan nüfusun, suç istatistiklerini yükselttiği yorumu da getirilebilir. Küme 2, suç türlerinde en düşük ortalamaya sahip olan kümedir. Yüksek suç istatistiklerine sahip eyaletler, Küme 2’de yer alan eyaletlerle iş birliğine giderek, yeni önlemler alma konusunda destek alabilirler.

K-Medoids

Daha önce belirtildiği gibi k-ortalamalar aykırı değerlere karşı hassastır. Bu durumla karşılaşıldığında, genellikle, centroid yerine medoid kullanan, k-medoids yöntemi kullanılmaktadır. Çünkü k-medoids yöntemi, kümenin merkezinde yer alan ve medoid olarak isimlendirilen gözlemi temel almaktadır (Park ve Jun, 2009). K-medoids algoritması 4 adımda özetlenebilir (Arora, Deepali ve Varshney, 2016):

1. k sayıda gözlem, medoid olarak atanır.
2. Geri kalan gözlemler, kendilerine en yakın medoid’in olduğu kümeye atanır.
3. Tüm gözlemler atandıktan sonra, medoidler tekrar belirlenir.
4. Medoidler sabit kalana kadar, 2. ve 3. adımlar tekrar edilir.

R Programlama Dili ile K-Medoids Uygulaması

K-medoids uygulaması için *cluster()* paketi kullanılacaktır. Bu bağlamda *library ()* fonksiyonu kullanılarak daha önce yüklenen paket çağrılmaktadır.

```
library(cluster)

kmedoids <- pam(scale(USArrests), 4)
```

pam() fonksiyonu çalıştırılmış ve sonuçlar "kmedoids" adıyla kaydedilmiştir. Sonrasında *print(kmedoids)* fonksiyonuyla sonuçlar görüntülenmiştir. *pam()* fonksiyonun ilk çıktısı medoidlerdir. Örneğin Küme 1'in merkezinde Alabama, Küme 2'nin merkezinde Michigan, Küme 3'ün merkezinde Oklohama ve Küme 4'ün merkezinde New Hampshire yer almaktadır. Medoidlerden sonra küme iyelikleri görülmektedir. Küme iyeliklerinden sonra yer alan değerler ne kadar küçükse, o kümenin o kadar kompakt olduğu ifade edilebilir. Bir başka ifadeyle, within cluster sum of squares değeri arttıkça küme içi değişkenliğin de arttığı söylenebilir. Son olarak "Available components" başlığı altında, görüntülenebilecek küme özellikleri bulunmaktadır (*kmedoids$clusinfo* kodu çalıştırılarak küme büyüklükleri görülebilir).

```
print(kmedoids)
```

```
##                                                                  Medoids:
##                          ID     Murder    Assault   UrbanPop       Rape
##  Alabama                  1  1.2425641  0.7828393 -0.5209066 -0.003416473
##  Michigan                22  0.9900104  1.0108275  0.5844655  1.480613993
##  Oklahoma                36 -0.2727580 -0.2371077  0.1699510 -0.131534211
##  New Hampshire           29 -1.3059321 -1.3650491 -0.6590781 -1.252564419
##                                                  Clustering vector:
##       Alabama        Alaska       Arizona      Arkansas    California
##             1             2             2             1             2
##      Colorado   Connecticut      Delaware       Florida       Georgia
##             2             3             3             2             1
##        Hawaii         Idaho      Illinois       Indiana          Iowa
##             3             4             2             3             4
##        Kansas      Kentucky     Louisiana         Maine      Maryland
##             3             3             1             4             2
## Massachusetts      Michigan     Minnesota   Mississippi      Missouri
##             3             2             4             1             3
##       Montana      Nebraska        Nevada New Hampshire    New Jersey
##             3             3             2             4             3
##    New Mexico      New York North Carolina  North Dakota          Ohio
##             2             2             1             4             3
##      Oklahoma        Oregon  Pennsylvania  Rhode Island South Carolina
##             3             3             3             3             1
##  South Dakota     Tennessee         Texas          Utah       Vermont
##             4             1             2             3             4
##      Virginia    Washington West Virginia     Wisconsin       Wyoming
##             3             3             4             4             3
```

```
##                                     Objective                                     function:
##                                         build                                          swap
##                                      1.035116                                      1.027102
##
##                                     Available                                    components:
##          [1]     "medoids"                  "id.med"                  "clustering"     "objective"          "isolation"
##  [6] "clusinfo"   "silinfo"    "diss"       "call"       "data"
```

Küme büyüklüklerini görebilmek için aşağıda yer alan kodun çalıştırılması gerekmektedir. Kodun sonucunda küme büyüklüklerinin sırasıyla 8, 12, 20 ve 10 olduğu görülmektedir.

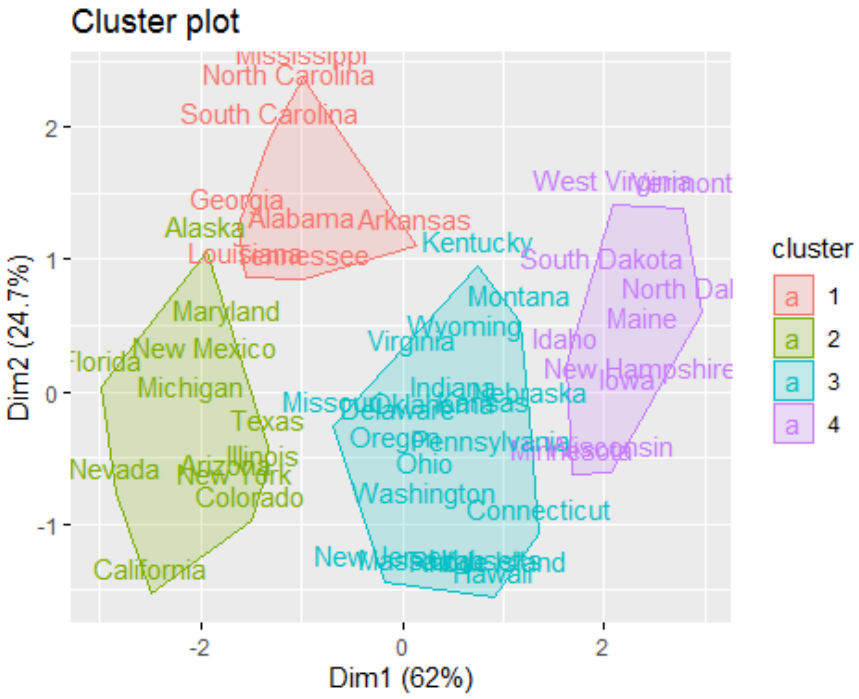
```
kmedoids$clusinfo
```

```
##                                size       max_diss                av_diss       diameter       separation
##          [1,]                     8       1.604366       0.9796963       2.337465                0.9912639
##          [2,]                12       2.115494       1.1908396       3.290377                1.1654171
##          [3,]                20       1.974670       1.0453201       3.045782                0.6083415
## [4,]   10 1.671613 0.8321050 2.401992  0.6083415
```

plot(kmedoids) kullanılarak da küme görselleri oluşturulabilir. Fakat k-ortalamalar küme görseline benzemesi için yine *fviz_cluster()* fonksiyonu kullanılmıştır:

1. Kümeleme analizinin sonuçlarının yer aldığı veri
2. Kümeleme analizinde kullanılan veri
3. Kümelerde yer alan gözlemlerin gösterim şekli (örnekte "text" kullanılarak her bir gözlemin adı görselde yer almaktadır. "point" kullanılırsa eğer nokta olarak da gösterilebilir).

```
fviz_cluster(kmedoids, USArrests, geom = "text")
```



Şekil 11. K-Medoids Sonucu Oluşan Kümeler

R Programlama Dili ile Kümeleme Analizinin Geçerliğinin İncelenmesi

Yapılan kümeleme analizinin geçerliğinin incelenmesi için Brock, Pihur, Datta ve Datta (2008) tarafından geliştirilmiş *clValid* paketinden faydalanılmıştır. İlk olarak, içsel geçerliğin tespiti için *clValid()* fonksiyonu kullanılmıştır ve "içsel" ismiyle kaydedilmiştir. İlgili fonksiyonun ilk parametresi, kümeleme analizinde kullanılmış olan veri setidir. Burada dikkat edilmesi gereken husus, eğer veri setine bir standardizasyon uygulandıysa, standardize veri seti kullanılmalıdır. Sonrasında, kümeleme analizinde kullanılan küme sayısı *nClust* parametresine, *clMethods* parametresine ise kullanılan kümeleme yöntemi yazılmıştır. Son olarak, *"internal"* ifadesiyle içsel geçerliğin inceleneceği fonksiyona girilmiştir. *Validation* ile incelenebilecek üç farklı geçerlik mevcuttur; "internal", "stability", "biological". *clMethods* için kullanılabilecek kümeleme algoritmaları ise "hierarchical", "kmeans", "diana", "fanny", "som", "model", "sota", "pam", "clara", and "agnes" şeklindedir. Bu algoritmalardan bir kaçı bir arada kullanılarak da fonksiyon çalıştırılabilir.

clValid paketi içsel geçerlikte "connectivity", "dunn" ve "silhouette" olmak üzere üç farklı değer hesaplamaktadır. Connectivity değeri 0 ile sonsuz arasında değişirken, bu değer minimize edilmeye çalışılmaktadır. Silhouette değeri -1 ile 1 arasında değişirken, 1'e yakın olması kümeleme analizinin içsel geçerliği için iyi bir göstergedir. Dunn Index de Connectivity'e benzer değerler almakta ve bu değerde hesaplanan yüksek değer iyi bir kümeleme performansı olduğunu göstermektedir (Brock vd., 2008).

İçsel geçerlikten sonra kümeleme analizinin ne ölçüde istikrarlı olduğunun tespiti için *clValid()* fonksiyonunda *validation* parametresine "stability" yazılmış ve bu fonksiyon da stability ismiyle kaydedilmiştir. Sonrasında *summary()* fonksiyonu kullanılarak sonuçlar raporlanmıştır. Sonuçta görülen APN (Average proportion of non-overlap) değeri 0-1 arasında değişirken, 0'a yakın değerler kümeleme analizi sonuçlarının tutarlı olduğunu göstermektedir. AD (Average distance), ADM (Average distance between means) ve FOM (Figure of merit) değerleri 0 ile sonsuz arasında değişirken, küçük değerler tercih edilmektedir.

```
library(clValid) #clValid paketini kullanmak için library fonksiyonu ile çağırıyoruz.

icsel <- clValid(USArrests, nClust = 4, clMethods = "kmeans", validation = "internal")
summary(icsel)

##
##                                  Clustering                                  Methods:
##                                                                               kmeans
##
##                                     Cluster                                    sizes:
##                                                                                   4
##
##                                  Validation                                  Measures:
##                                                                                   4
##
##                  kmeans                  Connectivity                            13.4306
##                                  Dunn                                         0.1591
##                                                  Silhouette                   0.4774
##
##                                    Optimal                                   Scores:
##
##                                                          Score        Method  Clusters
##                  Connectivity                          13.4306        kmeans         4
##       Dunn                                              0.1591        kmeans         4
## Silhouette    0.4774 kmeans 4

stability <- clValid(USArrests, nClust = 4, clMethods = "kmeans", validation = "stability")
summary(stability)

##
##                                  Clustering                                  Methods:
##                                                                               kmeans
##
##                                     Cluster                                    sizes:
##                                                                                   4
##
##                                  Validation                                  Measures:
##                                                                                   4
##
##                  kmeans                  APN                                  0.1606
##                                                                  AD               47.5375
##                                                                      ADM          16.3199
##                                                                      FOM          25.8884
##
##                                    Optimal                                   Scores:
##
##                                     Score                        Method  Clusters
##                  APN                0.1606                        kmeans         4
##                  AD                47.5375                        kmeans         4
##                     ADM            16.3199                        kmeans         4
## FOM 25.8884 kmeans 4
```

Her iki geçerlikte yöntemlerinde de hesaplanan değerler kullanılarak veri seti için en uygun yöntem ve küme sayısına karar verilebilir. Aşağıdaki ilk olarak hiyerarşik, k-medoids ve k-ortalamalara, 2, 3, 4, 5 ve 6 küme sayıları için içsel geçerlik değerleri hesaplanmıştır. “Optimal Scores” başlığı altında, ilgili geçerlik yöntemine göre optimal yöntem ve küme sayısı görülmektedir. Buna göre Connectivity’e göre yöntem hiyerarşik, küme sayısı 2, Dunn’a göre yöntem hiyearşik ve küme sayısı 5, Silhouette’ye göre ise yöntem pam ve küme sayısı 2 olmalıdır. İçsel geçerlikten sonra küme istikrarına göre optimal yöntem ve küme sayısı incelenmiştir. Sonuca göre APN’ye göre yöntem hiyerarşik ve küme sayısı 2, AD, ADM ve FOM’a göre optimal yöntem k-medoids ve küme sayısı 6 olarak belirlenmiştir.

```
icsel <- clValid(USArrests, nClust = 2:6, clMethods = c("hierarchical", "pam", "kmeans"), validation = "interna
l")
summary(icsel)
```

```
##
## Clustering Methods:
## hierarchical pam kmeans
##
## Cluster sizes:
## 2 3 4 5 6
##
## Validation Measures:
## 2 3 4 5 6
##
## hierarchical Connectivity 2.1913 4.0202 8.8877 12.7202 18.3778
## Dunn 0.1533 0.2503 0.2948 0.3438 0.3031
## Silhouette 0.5763 0.5319 0.5000 0.4713 0.4561
## pam Connectivity 6.0190 4.0202 11.4405 22.8016 24.8714
## Dunn 0.1033 0.2503 0.1002 0.1499 0.3022
## Silhouette 0.5927 0.5319 0.4892 0.4313 0.4494
## kmeans Connectivity 6.0190 4.0202 13.4306 17.2631 22.9206
## Dunn 0.1033 0.2503 0.1591 0.2021 0.2412
## Silhouette 0.5927 0.5319 0.4774 0.4493 0.4499
##
## Optimal Scores:
##
## Score Method Clusters
## Connectivity 2.1913 hierarchical 2
## Dunn 0.3438 hierarchical 5
## Silhouette 0.5927 pam 2
```

```
stability <- clValid(USArrests, nClust = 2:6, clMethods = c("hierarchical", "pam", "kmeans"), validation = "stab
ility")
summary(stability)
```

```
##
## Clustering Methods:
## hierarchical pam kmeans
##
```

```
## Cluster sizes:
## 2 3 4 5 6
##
## Validation Measures:
## 2 3 4 5 6
##
## hierarchical APN 0.0094 0.0912 0.1455 0.1497 0.1769
## AD 66.0661 51.2068 47.4030 41.0013 38.2303
## ADM 15.7949 15.7571 15.7669 15.3740 15.8156
## FOM 27.2575 26.2302 25.5205 24.5283 24.5359
## pam APN 0.1144 0.1598 0.1845 0.2189 0.2066
## AD 63.5303 51.7462 44.5428 40.2396 35.5343
## ADM 16.3949 17.1158 16.2841 16.7754 14.6340
## FOM 25.7070 26.2274 24.1998 23.1526 22.1243
## kmeans APN 0.1199 0.1519 0.1606 0.1642 0.1718
## AD 63.6132 51.4241 47.5375 40.1437 38.3087
## ADM 16.5751 16.2333 16.3199 14.6597 15.9141
## FOM 25.7667 26.3023 25.8884 24.1288 24.9231
##
## Optimal Scores:
##
## Score Method Clusters
## APN 0.0094 hierarchical 2
## AD 35.5343 pam 6
## ADM 14.6340 pam 6
## FOM 22.1243 pam 6
```

Sonuç

Açık kaynak kodlu programların istatistiksel analiz amaçlı kullanımı gittikçe yaygınlaşmaktadır. Yaygınlaşmayla birlikte alternatifler artmakta ve bu programların kullanımı daha kullanıcı dostu bir hale gelmektedir. Açık kaynak kodlu programların en önemli avantajı olarak, bu programların kullanımının ücretsiz olması gösterilebilir. Böylelikle veri işleme, analiz, görselleştirme gibi adımların tamamı ücretsiz bir şekilde yapılabilmektedir. Bu bağlamda mevcut çalışmada, R programlama dili ile kümeleme analizinin nasıl gerçekleştirileceğinin, adım adım gösterilmesi amaçlanmıştır. Çalışmada, Brock vd. (2008) tarafından geliştirilen *clValid*, Kassambara ve Mundt (2017) tarafından geliştirilen *factoextra* ve Maechler, Rousseeuw, Struyf, Hubert ve Hornik (2013) tarafından geliştirilen *cluster* paketleri kullanılmıştır. Literatürde R programlama dili kullanılarak kümeleme analizinin uygulama adımlarını, gösteren bir çalışmaya ulaşılamamıştır. Bu durum, çalışmanın özgün değeri olarak gösterilebilir.

Çalışmada kümeleme analizinin temel adımları gösterilmiştir. Bundan sonraki çalışmalarda farklı uzaklık ölçütleri kullanılarak kümeleme analizleri gerçekleştirilebilir.

Kaynakça

Arora, P., Deepali, D. ve Varshney, S. (2016). Analysis of k-means and k-medoids algorithm for big data. *Procedia Computer Science*, *78*, 507-512. doi: 10.1016/j.procs.2016.02.095

Berry, M. J. ve Linoff, G. S. (2004). *Data mining techniques: for marketing, sales, and customer relationship management*. New York: John Wiley & Sons.

Bholowalia, P.ve Kumar, A. (2014). EBK-means: A clustering technique based on elbow method and k-means in WSN. *International Journal of Computer Applications*, *105*(9), 17-24. Erişim adresi: https://research.ijcaonline.org/volume105/number9/pxc3899674.pdf

Brock, G., Pihur, V., Datta, S. ve Datta, S. (2008). clValid: An R package for cluster validation. *Journal of Statistical Software*, 25, 1-22. Erişim adresi: https://www.jstatsoft.org/article/view/v025i04

Celebi, M. E., Kingravi, H. A. ve Vela, P. A. (2013). A comparative study of efficient initialization methods for the k-means clustering algorithm. *Expert systems with applications*, *40*(1), 200-210. doi:10.1016/j.eswa.2012.07.021

Cohen-Addad, V., Kanade, V., Mallmann-Trenn, F. ve Mathieu, C. (2019). Hierarchical clustering: Objective functions and algorithms. *Journal of the ACM (JACM)*, *66*(4), 1-42. doi:10.1145/3321386

Day, W. H. ve Edelsbrunner, H. (1984). Efficient algorithms for agglomerative hierarchical clustering methods. *Journal of classification*, *1*(1), 7-24. Erişim adresi: https://link.springer.com/article/10.1007/bf01890115

Dehariya, V. K., Shrivastava, S. K.ve Jain, R. C. (2010, November). Clustering of image data set using k-means and fuzzy k-means algorithms. In *2010 International conference on computational intelligence and communication networks* (pp. 386-391). IEEE. doi: 10.1109/CICN.2010.80

Everitt, B. S., Landau, S., Leese, M. ve Stahl, D. (2011). Cluster Analysis (5th edition). Chichester, UK: John Wiley & Sons.

Hair, J. F. ve Black, W. C. (2000). Cluster Analysis. L. G. Grimm & P. R. Yarnold (Eds.), In *Reading and understanding MORE multivariate statistics.* (pp. 147–206). American Psychological Association. Erişim adresi: https://psycnet.apa.org/record/2000-00427-000

Hair, J. F., Black, W. C., Babin, B. J. ve Anderson, R. E. (2010). *Multivariate data analysis (Vol. 7)*. London: Pearson.

Han, J., Pei, J. ve Kamber, M. (2011). *Data mining: concepts and techniques*. Elsevier.

Kassambara, A. ve Mundt, F. (2017). Package 'factoextra'. *Extract and visualize the results of multivariate data analyses*, [Veri seti ve kodlama çizelgesi]. Erişim adresi: https://cran.microsoft.com/snapshot/2016-11-30/web/packages/factoextra/factoextra.pdf

Kaufman, L. ve Rousseeuw, P. J. (2009). *Finding groups in data: an introduction to cluster analysis*. New York: John Wiley & Sons.

Maechler, M., Rousseeuw, P., Struyf, A., Hubert, M., Hornik, K. ve Studer, M. (2013). Package 'cluster'. [Veri seti ve kodlama çizelgesi]. Erişim adresi: https://cran.microsoft.com/snapshot/2014-10-10/web/packages/cluster/cluster.pdf

Milligan, G. W. ve Cooper, M. C. (1987). Methodology review: Clustering methods. *Applied psychological measurement*, *11*(4), 329-354. doi: 10.1177/014662168701100401

Milligan, G. W. ve Cooper, M. C. (1988). A study of standardization of variables in cluster analysis. *Journal of classification*, *5*(2), 181-204. Erişim adresi: https://link.springer.com/article/10.1007/BF01897163

Mohamad, I. B.ve Usman, D. (2013). Standardization and its effects on K-means clustering algorithm. *Research Journal of Applied Sciences, Engineering and Technology*, 6(17), 3299-3303. doi: 10.19026/rjaset.6.3638

Nanjundan, S., Sankaran, S., Arjun, C. R. ve Anand, G. P. (basım aşamasında). Identifying the number of clusters for K-Means: A hypersphere density based approach. *arXiv preprint arXiv:1912.00643*. doi: 10.48550/arXiv.1912.00643

Önder, E. (2020). *Sağlıkta gelişmekte olan teknolojiler yapay zekâ & R programlama dili ile makine öğrenimi uygulamaları.* Bursa: Dora Yayınevi.

Park, H. S. ve Jun, C. H. (2009). A simple and fast algorithm for K-medoids clustering. *Expert systems with applications*, *36*(2), 3336-3341. doi: 10.1016/j.eswa.2008.01.039

Wu, J. D., Milton, D. K., Hammond, S. K. ve Spear, R. C. (1999). Hierarchical cluster analysis applied to workers exposures in fiberglass insulation manufacturing. *Annals of Occupational Hygiene*, *43*(1), 43-55. doi: 10.1093/annhyg/43.1.43

Žalik, K. R. (2008). An efficient k′-means clustering algorithm. *Pattern Recognition Letters*, *29*(9), 1385-1391. doi: 10.1016/j.patrec.2008.02.014

Extended Abstract

Purpose

It is aimed to create a resource for researchers who want to do cluster analysis. Researchers who wish can perform cluster analysis by making very minor revisions in the codes used in the study.

Design and Methodology

One of the most important problems in cluster analysis can be stated as deciding on the number of clusters. There are methods such as Elbow, average Silhouette and GAP statistics methods to overcome this problem. Cluster analysis can be categorized under two main headings: hierarchical and non-hierarchical algorithms. One of the differences between these two categories is that non-hierarchical algorithms require the number of clusters before the analysis. Also, the cluster memberships formed by hierarchical algorithms are stable. In non-hierarchical algorithms, on the other hand, cluster memberships change until they remain constant.

The use of open source programs for statistical analysis is becoming more and more widespread. With the widespread use, alternatives are increasing, and the use of these programs is becoming more user-friendly. The most important advantage of open source programs is that they are free to use. Thus, all steps such as data processing, analysis and visualization can be done free of charge. In this context, in the present study, it is aimed to show step by step how to perform clustering analysis with R programming. factoextra() and cluster() packages were used in the study to perform both hierarchical and non-hierarchical clustering.

Research Limitations

The basic steps of clustering analysis were demonstrated in the study. In future studies, clustering analyses can be performed using different distance criteria and cluster validity can be examined. One of the limitations of the study is that only R Programming is used. In addition, analyzes were carried out on only one data set.

Originality/Value

There is no study found in the literature showing the application steps of clustering analysis using R programming. This situation can be shown as the original value of the study

Araştırmacı Katkısı: Cem GÜRLER (%100).