**Data Science and Machine Learning (MSc)**

DAMA51: Foundations in Computer Science

Academic Year: 2022–2023

|  |  |
| --- | --- |
| #4 Written Assignment | |
| Submission Deadline | Wed, 26 April 2023, 11:59 PM |
| Student Name: | TRANTEROU VASILEIA |

# Remarks

The deadline is definitive.

An indicative solution will be posted online along with the returning of the graded assignments.

The assignment is due via the STUDY submission system. **You are expected to turn in a document (.DOC, .ODT, .PDF) and a compressed (.ZIP, .RAR) file containing all your work:**

* **1 document file (this document) with the answers to all the questions, along with the R code and the results of the execution of the code**
* **1 compressed file with 3 R scripts with the code that answers to each one of the problems to the Topics 3 and 5.**

**You should not make any changes in the written assignment file other than providing your own answers.** You should also type all of your answers into Word and not attach any handwritten notes as pictures into your work otherwise a 5% reduction of your final grade will be applied. Make sure to name all the files (ZIP file, DOC file and R script files) with **your last name first followed by a dash symbol and the names of each component at the end**. For example, for the student with the last name Aggelou the files should be named as follows: Aggelou-HW4.zip, Aggelou-HW4.doc, Aggelou-Topic3.R, Aggelou-Topic4.R, and Aggelou-Topic5.R. The R script files should automatically run with the **source** command and generate the correct results. Also, please include comments before each command to explain the functionality of the command that follows. In the computations, use three decimal places.

|  |  |  |
| --- | --- | --- |
| Topic | Points | Grades |
| 1. **Online Quiz** | 40 |  |
| 1. **Article review** | 5 |  |
| 1. **Prototype-based Clustering** | 20 |  |
| 1. **Hierarchcal based Clustering using R** | 20 |  |
| 1. **Itemset Mining and Association Rules using R** | 20 |  |
| **TOTAL** | **105 (max 100)** | **/100** |

# Topic 1: Quiz

: Online QUIZ

**(40 points)** Complete the corresponding online quiz available at:

<https://study.eap.gr/mod/quiz/view.php?id=24568>

You have one effort and unlimited time to complete the quiz, up to the submission deadline.

# Topic 2: Article Review

The article “The planning and care of data” (https://dl.acm.org/doi/10.1145/3532633) makes a point about what drives complexity in software systems compared to data-oriented projects. What is this comparison? Why does the author believe that modern start-ups are less inclined to treat data engineering properly?

Note: You should write up your answer to a maximum of 100 words. Any text in excess of 100 words will not be taken into consideration.

**(5 points)**

|  |
| --- |
|  |

# Topic 3: Prototype-based and k-means Clustering

**(20 total points)** This topic will use the ***seeds*** dataset, which contains data about the physical properties of the internal kernel structure of various wheats. The wheats come from three different varieties.

Read the data using a command like the one below:

seeds <- read.csv("seeds\_dataset", header = TRUE)

Note about reproducibility for the k-means algorithm: Since k-means will pseudo-randomly initialize its state, make sure that exactly before using the k-means algorithm, you call set.seed(123).

All the topics are expected to be answered using R unless explicitly stated otherwise.

**(a) (6 points)** Perform a cluster analysis with the k-means algorithm. The desired number of clusters is 3. For your analysis use all the features of the dataset except columns seeID and seedType. Ensure that the dataset is scaled; if not, scale it so that the mean is0 and the standard deviation is 1.

Provide the scaled values of the attribute perimeter for each cluster prototype (centroid).

Find the cluster prototype that the data instances of rows 9, 55 and 189 belong to.

(Fill all values)

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Answer:**   |  |  |  | | --- | --- | --- | | *Value of attribute “perimeter” for each cluster prototype:* | *Cluster prototype of data instances:* | *Euclidean distance between centroids:* | | Perimeter of cluster 1 prototype= -0.07835561 | Cluster for data row 9 = 2 | dist(1,2)= 3.522711 | | Perimeter of cluster 2 prototype= 0.63354296 | Cluster for data row 55 = 2 | dist(1,3)= 3.292530 | | Perimeter of cluster 3 prototype= -0.96468107 | Cluster for data row 189 = 3 | dist(2,3)= 2.991381 |   **R code**: new\_dataset<-select(seeds\_dataset, -seeID, -seedType)  > data\_scaled <- scale(new\_dataset)  > set.seed(123) # for reproducibility  > kmeans\_model <- kmeans(data\_scaled, centers = 3)  > kmeans\_model  K-means clustering with 3 clusters of sizes 23, 114, 73  Cluster means:  area perimeter compactness lengthOfKernel widthOfKernel  1 -0.1044938 -0.07835561 -0.2165082 0.03674955 0.05617052  2 0.6444335 0.63354296 0.5466212 0.27631389 0.34978572  3 -0.9734529 -0.96468107 -0.7854128 -0.44308251 -0.56393828  asymmetryCoefficient lengthOfKernelGroove  1 0.2816133 -2.7419179  2 -0.3246355 0.4593783  3 0.4182376 0.1465066  Clustering vector:  [1] 1 2 2 2 2 2 2 1 2 2 2 2 2 2 2 2 2 2 2 3 2 3 2 3 2 2 3 3 2 3 2 2 2 3  [35] 1 2 2 2 2 2 2 2 2 2 1 2 2 2 2 2 2 2 3 2 2 2 2 2 2 3 3 3 2 3 3 3 1 2  [69] 1 3 1 2 2 2 2 2 2 2 2 2 1 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2  [103] 2 2 2 2 1 2 1 2 2 2 2 2 2 2 1 2 2 1 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2  [137] 2 2 2 2 3 1 3 3 3 3 3 3 3 3 3 1 1 3 3 3 3 1 1 1 3 3 3 3 3 3 1 3 3 3  [171] 3 3 1 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 1 3 2 3 1  [205] 3 1 3 3 3 3  Within cluster sum of squares by cluster:  [1] 136.9448 426.4480 303.3806  (between\_SS / total\_SS = 40.8 %)  Available components:  [1] "cluster" "centers" "totss" "withinss"  [5] "tot.withinss" "betweenss" "size" "iter"  [9] "ifault"  > kmeans\_model$centers  area perimeter compactness lengthOfKernel widthOfKernel  1 -0.1044938 -0.07835561 -0.2165082 0.03674955 0.05617052  2 0.6444335 0.63354296 0.5466212 0.27631389 0.34978572  3 -0.9734529 -0.96468107 -0.7854128 -0.44308251 -0.56393828  asymmetryCoefficient lengthOfKernelGroove  1 0.2816133 -2.7419179  2 -0.3246355 0.4593783  3 0.4182376 0.1465066  > centroids <- kmeans\_model$centers  > centroid\_distance <- dist(centroids)  > print(centroid\_distance)  1 2  2 3.522711  3 3.292530 2.991381 |

**(b) (8 points)** Count how many wheats are assigned to each cluster.

For achieving this, first create a new vector to hold all the assignments (i.e., the vector of integers indicating the cluster to which each point is allocated) and, in this vector, rename cluster 1 to cluster 2, and cluster 2 to cluster 1. **(2 points)**

Then, using a confusion matrix such as the one below, make a comparison of this vector with the attribute seedType. Compare the values of the diagonal elements against the other elements. **(2 points)**

Count how many wheats have been falsely assigned to an incorrect cluster and calculate the accuracy of clustering. **(2 points)**

Then, using **pen and paper**, calculate the precision and recall rates for cluster 1. **(2 points)**

(Fill all values)

Answer:

**Rcode**: cluster\_assignments <- kmeans\_model$cluster

> cluster\_assignments

[1] 1 2 2 2 2 2 2 1 2 2 2 2 2 2 2 2 2 2 2 3 2 3 2 3 2 2 3 3 2 3 2 2 2 3

[35] 1 2 2 2 2 2 2 2 2 2 1 2 2 2 2 2 2 2 3 2 2 2 2 2 2 3 3 3 2 3 3 3 1 2

[69] 1 3 1 2 2 2 2 2 2 2 2 2 1 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2

[103] 2 2 2 2 1 2 1 2 2 2 2 2 2 2 1 2 2 1 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2

[137] 2 2 2 2 3 1 3 3 3 3 3 3 3 3 3 1 1 3 3 3 3 1 1 1 3 3 3 3 3 3 1 3 3 3

[171] 3 3 1 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 1 3 2 3 1

[205] 3 1 3 3 3 3

> renamed\_assignments <- ifelse(cluster\_assignments == 1, 2, ifelse(cluster\_assignments == 2, 1, cluster\_assignments))

> renamed\_assignments

[1] 2 1 1 1 1 1 1 2 1 1 1 1 1 1 1 1 1 1 1 3 1 3 1 3 1 1 3 3 1 3 1 1 1 3

[35] 2 1 1 1 1 1 1 1 1 1 2 1 1 1 1 1 1 1 3 1 1 1 1 1 1 3 3 3 1 3 3 3 2 1

[69] 2 3 2 1 1 1 1 1 1 1 1 1 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1

[103] 1 1 1 1 2 1 2 1 1 1 1 1 1 1 2 1 1 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1

[137] 1 1 1 1 3 2 3 3 3 3 3 3 3 3 3 2 2 3 3 3 3 2 2 2 3 3 3 3 3 3 2 3 3 3

[171] 3 3 2 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 2 3 1 3 2

[205] 3 2 3 3 3 3

> confusion\_matrix <- table(renamed\_assignments, seeds\_dataset$seedType)

> confusion\_matrix

renamed\_assignments 1 2 3

1 49 64 1

2 6 6 11

3 15 0 58

|  |  |  |  |
| --- | --- | --- | --- |
|  | cluster | | |
| seedType | 1 | 2 | 3 |
| 1 | 49 | 6 | 15 |
| 2 | 64 | 6 | 0 |
| 3 | 1 | 11 | 58 |

Precision rate for cluster 1 = 0.423

Recall rate for cluster 1 = 0.7

Accuracy = (49+6+58)/ 210 = 0.53809524 or 0.538

Recall rate for cluster 1= 49/(49+6+15) = 49/70 = 0.7

Precision rate for cluster 1 = 49/ (49+64+1) = 49/114 = 0.42982456

**(c) (6 points)**

Calculate the average silhouette for the k-means clustering that has been performed (i.e., with k=3) (note that you first need to have the cluster package installed). Repeat the calculation for a clustering with 4 clusters (i.e., k=4) and confirm that the average silhouette is lower. **(3 points)**

Using the function fviz\_cluster() of the factoextra R package (you will need to have it installed first), visualize the k-means clusters for k=3 as well as for k=4. Based on the plots, comment whether the clusters are well separated. **(3 points)**

Answer:

Average Silhouette (3 clusters) = 0.3086675

Average Silhouette (4 clusters) = 0.2623978

Comment on whether the clusters are well separated: The clusters are well separated because they have clear visual separation and minimal overlap between them.

**R code:** avg\_sil<- function(k){

+ km<-kmeans(data\_scaled, centers = k)

+ ss<-silhouette(km$cluster, dist(data\_scaled))

+ mean(ss[,3])}

> #k=3

> avg\_sil(3)

[1] 0.3086675

> #k=4

> avg\_sil(4)

[1] 0.2623978

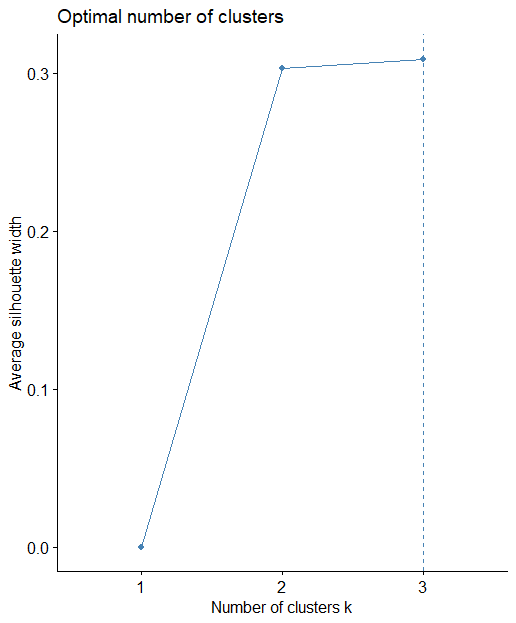
install.packages("factoextra")

library(factoextra)

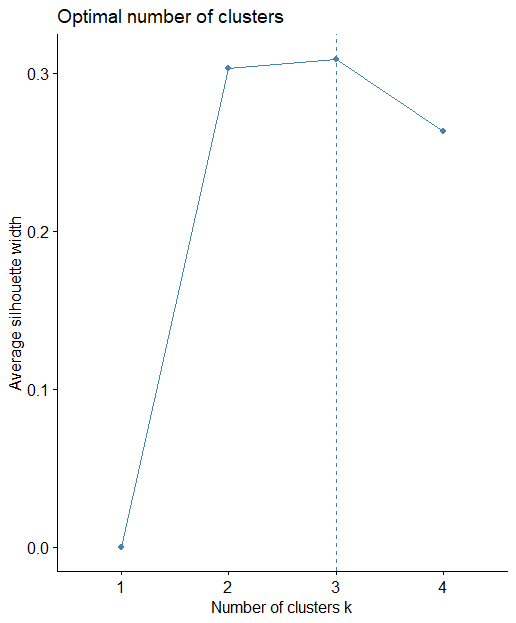
fviz\_nbclust(data\_scaled,kmeans,method= "silhouette",k.max=3)

fviz\_nbclust(data\_scaled,kmeans,method= "silhouette",k.max=4)

**Plot (k=3):**



**Plot (k=4):**



# Topic 4: Hierarchical based Clustering using R

**(20 points)** For this topic, you will work on the ***europe\_diet*** dataset which can be found here. This dataset includes records on the kilocalories received daily per person from different food categories in several European countries. For all questions, you are requested to provide your R code and the result of its execution in every answer box. All the topics are expected to be answered using R unless explicitly stated otherwise.

1. (**2 points**) Inspect the dataset, set the row names according to the values of the corresponding country column and then, remove this column.

Answer: **Rcode:**

head(europe\_diet)

# A tibble: 6 × 11

Country Other Alcoholic.Beverages Sugar Oils Meat Dairy.Eggs

<chr> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>

1 Albania 33 73 197 322 345 579

2 Austria 37 243 436 1045 416 326

3 Belarus 68 214 307 620 438 237

4 Belgium 34 171 496 988 313 428

5 Bosnia and Herz… 131 232 178 228 180 300

6 Bulgaria 42 191 271 489 242 292

# ℹ 4 more variables: Fruit.Vegetables <dbl>, Starchy.Roots <dbl>,

# Pulses <dbl>, Cereals.Grains <dbl>

> summary(europe\_diet)

Country Other Alcoholic.Beverages Sugar

Length:41 Min. : 18.00 Min. : 53 Min. :178.0

Class :character 1st Qu.: 37.00 1st Qu.:130 1st Qu.:302.0

Mode :character Median : 60.00 Median :162 Median :371.0

Mean : 70.37 Mean :170 Mean :358.7

3rd Qu.: 94.00 3rd Qu.:191 3rd Qu.:428.0

Max. :190.00 Max. :396 Max. :582.0

Oils Meat Dairy.Eggs Fruit.Vegetables

Min. : 226.0 Min. :141.0 Min. :228.0 Min. : 96.0

1st Qu.: 437.0 1st Qu.:306.0 1st Qu.:293.0 1st Qu.:158.0

Median : 583.0 Median :390.0 Median :349.0 Median :194.0

Mean : 571.6 Mean :381.4 Mean :367.3 Mean :197.3

3rd Qu.: 734.0 3rd Qu.:461.0 3rd Qu.:424.0 3rd Qu.:219.0

Max. :1045.0 Max. :710.0 Max. :579.0 Max. :372.0

Starchy.Roots Pulses Cereals.Grains

Min. : 48.0 Min. : 0.00 Min. : 671.0

1st Qu.: 93.0 1st Qu.:11.00 1st Qu.: 872.0

Median :109.0 Median :18.00 Median : 975.0

Mean :128.3 Mean :26.27 Mean : 995.3

3rd Qu.:169.0 3rd Qu.:37.00 3rd Qu.:1099.0

Max. :336.0 Max. :82.00 Max. :1562.0

> df<- data.frame(europe\_diet)

> rownames(df)<-europe\_diet$Country

> df[,1] <- NULL

> df

Other Alcoholic.Beverages Sugar Oils Meat

Albania 33 73 197 322 345

Austria 37 243 436 1045 416

Belarus 68 214 307 620 438

Belgium 34 171 496 988 313

Bosnia and Herzegovina 131 232 178 228 180

Bulgaria 42 191 271 489 242

Croatia 93 159 401 531 329

Cyprus 97 95 312 532 358

Czechia 60 253 389 786 306

Denmark 101 164 483 621 441

Estonia 111 396 268 289 276

Finland 83 164 304 437 599

France 54 130 374 758 527

Georgia 67 66 337 256 141

Germany 33 255 469 801 390

Greece 65 100 279 800 328

Hungary 38 183 343 804 314

Iceland 120 131 434 457 710

Ireland 48 305 395 551 439

Italy 30 101 308 875 415

Latvia 94 184 371 595 321

Lithuania 36 287 407 363 485

Luxembourg 190 306 320 401 634

Malta 95 118 548 438 401

Moldova 41 108 208 431 208

Montenegro 143 83 330 323 495

Netherlands 22 133 428 526 461

North Macedonia 87 53 375 608 179

Norway 181 119 314 679 500

Poland 18 184 430 583 410

Portugal 70 214 257 747 461

Romania 60 159 244 419 219

Russia 36 165 439 482 372

Serbia 73 164 270 226 355

Slovakia 54 151 428 606 239

Slovenia 104 133 234 590 295

Spain 44 158 302 797 458

Sweden 45 140 413 651 416

Switzerland 32 171 582 734 477

Ukraine 22 162 430 438 270

United Kingdom 93 153 394 609 476

Dairy.Eggs Fruit.Vegetables Starchy.Roots Pulses

Albania 579 372 78 50

Austria 326 261 110 7

Belarus 237 193 336 0

Belgium 428 203 172 23

Bosnia and Herzegovina 300 271 139 68

Bulgaria 292 130 48 22

Croatia 349 198 75 7

Cyprus 263 153 53 30

Czechia 323 125 135 18

Denmark 392 197 109 10

Estonia 459 174 154 82

Finland 545 158 119 11

France 392 175 95 17

Georgia 277 96 103 0

Germany 378 179 115 7

Greece 418 285 133 45

Hungary 265 131 84 29

Iceland 573 196 54 7

Ireland 393 211 154 31

Italy 324 261 66 52

Latvia 314 140 214 0

Lithuania 340 128 174 31

Luxembourg 424 236 93 11

Malta 356 219 76 37

Moldova 370 142 99 8

Montenegro 558 301 218 65

Netherlands 514 254 169 16

North Macedonia 228 264 106 45

Norway 381 189 98 45

Poland 278 146 191 17

Portugal 286 254 124 36

Romania 481 219 180 18

Russia 328 162 207 17

Serbia 264 209 73 56

Slovakia 242 110 93 12

Slovenia 319 204 98 19

Spain 293 174 105 48

Sweden 444 174 104 17

Switzerland 438 194 75 17

Ukraine 305 188 249 14

United Kingdom 384 212 181 32

Cereals.Grains

Albania 1144

Austria 887

Belarus 837

Belgium 905

Bosnia and Herzegovina 1427

Bulgaria 1102

Croatia 917

Cyprus 756

Czechia 861

Denmark 849

Estonia 1044

Finland 948

France 960

Georgia 1562

Germany 872

Greece 947

Hungary 846

Iceland 698

Ireland 1073

Italy 1147

Latvia 941

Lithuania 1166

Luxembourg 924

Malta 1090

Moldova 1099

Montenegro 975

Netherlands 705

North Macedonia 1004

Norway 979

Poland 1194

Portugal 1028

Romania 1359

Russia 1153

Serbia 1038

Slovakia 1009

Slovenia 1172

Spain 795

Sweden 775

Switzerland 671

Ukraine 1060

United Kingdom 890

1. **(4 points**) Calculate the dissimilarity distance matrices of the dataset using the Euclidean distance method. Then fill in the following table with the distances of Spain, Belgium and Finland to Greece.

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Answer:     |  |  |  |  | | --- | --- | --- | --- | | Euclidian distance | Spain | Belgium | Finland | | Greece | 270.3442 | 315.0444 | 493.8279 |   **Rcode**: dist\_matrix <- dist(df, method = "euclidean")  > dist\_df <- as.data.frame(as.matrix(dist\_matrix))  > distances <- dist\_df[c(37, 4, 12), 16] # 37=Spain, 4=Belgium, 12=Finland, 16=Greece  > #distances <- dist\_matrix[c("Spain", "Belgium", "Finland"), "Greece"]  > #I tried also in this way but i had an error of incorrect number of dimensions  > print(distances)  [1] 270.3442 315.0444 493.8279 |

1. (**6 points**) Now, perform agglomerative hierarchical clustering using the **Euclidian** dissimilarity distance matrix and for both complete **(2 points)** and single **(2 points)** linkage. Provide the dendrograms of both analyses **(2 points)**.

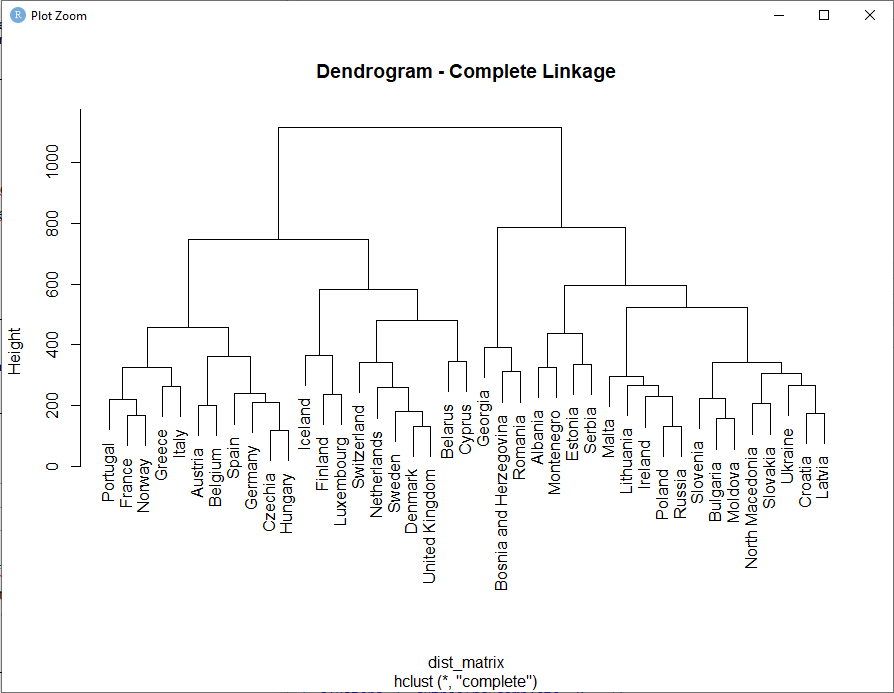
Answer:

**Rcode:**

hc\_complete <- hclust(dist\_matrix, method = "complete")

plot(hc\_complete, main = "Dendrogram - Complete Linkage")

**Complete linkage:**

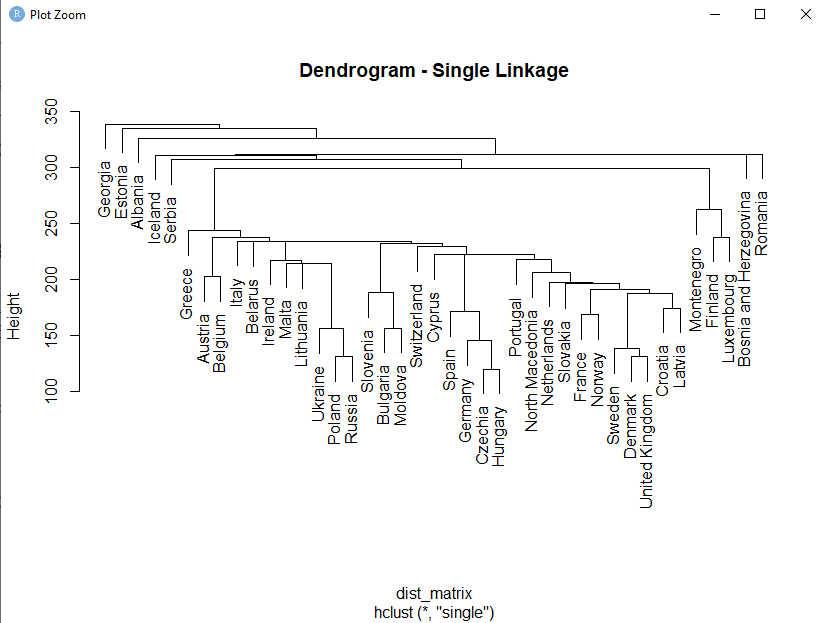


**Rcode:**

hc\_single <- hclust(dist\_matrix, method = "single")

plot(hc\_single, main = "Dendrogram - Single Linkage")

**Single linkage:**



1. (**5 points**) Now, based on the complete linkage hierarchical clustering of the previous question, cluster the European countries to 7 different groups **(2 points)**. Using R, identify the countries that have been assigned to the same cluster as i) Switzerland and ii) Norway **(3 points)**.

Answer: **Rcode:**

hc\_complete <- hclust(dist\_matrix, method = "complete")

> clusters <- cutree(hc\_complete, k = 7)

> table(clusters)

clusters

1 2 3 4 5 6 7

4 11 7 3 8 3 5

> switzerland <- which(rownames(dist\_df) == "Switzerland")

> switzerland

[1] 39

> switzerland\_cluster <- which(clusters == clusters[switzerland])

> print(switzerland\_cluster)

Belarus Cyprus Denmark Netherlands

3 8 10 27

Sweden Switzerland United Kingdom

38 39 41

> norway <- which(rownames(dist\_df) == "Norway")

> norway

[1] 29

> norway\_cluster <- which(clusters == clusters[norway])

> print(norway\_cluster)

Austria Belgium Czechia France Germany Greece Hungary Italy

2 4 9 13 15 16 17 20

Norway Portugal Spain

29 31 37

1. (**3 points**) Using R, identify the maximum number of clusters k for which Greece and Cyprus belong to the same cluster.

Answer:

*Requested maximum number of clusters: 3*

**R code:**

k <- 2

> clusters <- cutree(hc\_complete, k = k)

> while (clusters[16] == clusters[8]) {

+

+ k <- k + 1

+ clusters <- cutree(hc\_complete, k = k)

+ }

> print(paste0("The maximum number of clusters for which Greece and Cyprus belong to the same cluster is ", k - 1))

[1] "The maximum number of clusters for which Greece and Cyprus belong to the same cluster is 3"

# Topic 5: Itemset Mining and Association Rules using R

**(20 points)** For this topic, you will work on the ***application\_data*** dataset which can be found here. This dataset includes records on the applications that university students have installed on their smartphones. Each record (transaction) includes the set of applications installed by each student. You are requested to provide your R code and the result of its execution in every answer box. All the topics are expected to be answered using R unless explicitly stated otherwise.

Please read the dataset using the following command:

appstrans<-read.transactions(“path/application\_data.csv", format = "basket", sep=",",rm.duplicates=FALSE)

1. (**3 points**) Provide the names of all different applications installed.

Answer: **Rcode:**

unique\_apps <- itemLabels(appstrans)

unique\_apps

[1] "Amazon" "Amazon Prime" "Discord" "Facebook"

[5] "Hotstar" "Instagram" "Meet" "Netflix"

[9] "Pinterest" "SnapChat" "Spotify" "Twitter"

[13] "Whatsapp" "Wynk" "Youtube" "Zoom”

1. (**3 points**) Fill in the table below with the number of students who have installed a specific number of applications

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Answer:   |  |  | | --- | --- | | Number of applications/student | Number of students | | 3 | 2 | | 5 | 19 | | 7 | 1 |   **Rcode:**  summary(appstrans) #from the sizes i can see the number of applications and the number of students  transactions as itemMatrix in sparse format with  31 rows (elements/itemsets/transactions) and  16 columns (items) and a density of 0.3145161  most frequent items:  Youtube Whatsapp Instagram Meet SnapChat (Other)  29 27 21 20 9 50  element (itemset/transaction) length distribution:  sizes  3 4 5 6 7 10  2 5 19 3 1 1  Min. 1st Qu. Median Mean 3rd Qu. Max.  3.000 5.000 5.000 5.032 5.000 10.000  includes extended item information - examples:  labels  1 Amazon  2 Amazon Prime  3 Discord |

1. (**2 points**) Now, create an item frequency plot with the top10 applications in terms of relative frequency.

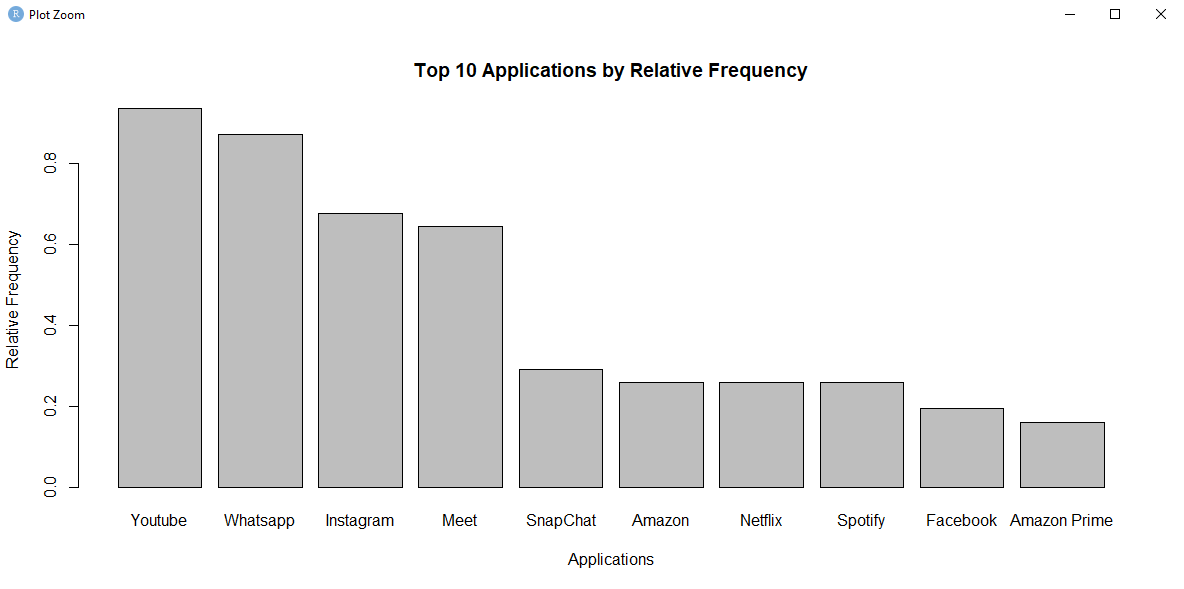
Answer:

**Rcode:**

freq <- itemFrequency(appstrans, type = "relative")

top10 <- head(sort(freq, decreasing = TRUE), 10)

barplot(top10, main = "Top 10 Applications by Relative Frequency", xlab = "Applications", ylab = "Relative Frequency")



1. (**6 points**) Run the apriori algorithm for a minimum support threshold of 0.25, a minimum confidence threshold of 0.8 and minimum of 2 items involved in a rule **(2 points)**. Sort the rules generated according to decreasing value of “*support*” and list only the Top4 of them **(2 points)**. Identify the rule length distribution, i.e. the number of rules with 2 items, 3 items, etc . **(2 points)**

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Answer:   |  |  | | --- | --- | | Number of rules | Number of items | | 18 | 2 | | 9 | 3 | | 2 | 4 | | 0 | 5 | | 0 | 6 |   **Rcode:**  rules <- apriori(appstrans, parameter = list(supp = 0.25, conf = 0.8, minlen = 2))  inspect(rules)  top4 <- head(sort(rules, by = "support", decreasing = TRUE), 4)  inspect(top4)  rules2 <- apriori(appstrans, parameter = list(supp = 0.25, conf = 0.8, minlen = 3))  inspect(rules2)  rules3 <- apriori(appstrans, parameter = list(supp = 0.25, conf = 0.8, minlen = 4))  inspect(rules3)  rules4 <- apriori(appstrans, parameter = list(supp = 0.25, conf = 0.8, minlen = 5))  inspect(rules4)  rules5 <- apriori(appstrans, parameter = list(supp = 0.25, conf = 0.8, minlen = 6))  inspect(rules5)  **Output:**  rules <- apriori(appstrans, parameter = list(supp = 0.25, conf = 0.8, minlen = 2))  Apriori  Parameter specification:  confidence minval smax arem aval originalSupport maxtime support  0.8 0.1 1 none FALSE TRUE 5 0.25  minlen maxlen target ext  2 10 rules TRUE  Algorithmic control:  filter tree heap memopt load sort verbose  0.1 TRUE TRUE FALSE TRUE 2 TRUE  Absolute minimum support count: 7  set item appearances ...[0 item(s)] done [0.00s].  set transactions ...[16 item(s), 31 transaction(s)] done [0.00s].  sorting and recoding items ... [8 item(s)] done [0.00s].  creating transaction tree ... done [0.00s].  checking subsets of size 1 2 3 4 done [0.00s].  writing ... [18 rule(s)] done [0.00s].  creating S4 object ... done [0.00s].  > inspect(rules)  lhs rhs support confidence  [1] {Amazon} => {Youtube} 0.2580645 1.0000000  [2] {SnapChat} => {Instagram} 0.2903226 1.0000000  [3] {SnapChat} => {Youtube} 0.2580645 0.8888889  [4] {Meet} => {Whatsapp} 0.5483871 0.8500000  [5] {Meet} => {Youtube} 0.6129032 0.9500000  [6] {Instagram} => {Whatsapp} 0.5483871 0.8095238  [7] {Instagram} => {Youtube} 0.6129032 0.9047619  [8] {Whatsapp} => {Youtube} 0.8709677 1.0000000  [9] {Youtube} => {Whatsapp} 0.8709677 0.9310345  [10] {Instagram, SnapChat} => {Youtube} 0.2580645 0.8888889  [11] {SnapChat, Youtube} => {Instagram} 0.2580645 1.0000000  [12] {Instagram, Meet} => {Youtube} 0.3870968 0.9230769  [13] {Meet, Whatsapp} => {Youtube} 0.5483871 1.0000000  [14] {Meet, Youtube} => {Whatsapp} 0.5483871 0.8947368  [15] {Instagram, Whatsapp} => {Youtube} 0.5483871 1.0000000  [16] {Instagram, Youtube} => {Whatsapp} 0.5483871 0.8947368  [17] {Instagram, Meet, Whatsapp} => {Youtube} 0.3225806 1.0000000  [18] {Instagram, Meet, Youtube} => {Whatsapp} 0.3225806 0.8333333  coverage lift count  [1] 0.2580645 1.0689655 8  [2] 0.2903226 1.4761905 9  [3] 0.2903226 0.9501916 8  [4] 0.6451613 0.9759259 17  [5] 0.6451613 1.0155172 19  [6] 0.6774194 0.9294533 17  [7] 0.6774194 0.9671593 19  [8] 0.8709677 1.0689655 27  [9] 0.9354839 1.0689655 27  [10] 0.2903226 0.9501916 8  [11] 0.2580645 1.4761905 8  [12] 0.4193548 0.9867374 12  [13] 0.5483871 1.0689655 17  [14] 0.6129032 1.0272904 17  [15] 0.5483871 1.0689655 17  [16] 0.6129032 1.0272904 17  [17] 0.3225806 1.0689655 10  [18] 0.3870968 0.9567901 10  > top4 <- head(sort(rules, by = "support", decreasing = TRUE), 4)  > inspect(top4)  lhs rhs support confidence coverage lift  [1] {Whatsapp} => {Youtube} 0.8709677 1.0000000 0.8709677 1.0689655  [2] {Youtube} => {Whatsapp} 0.8709677 0.9310345 0.9354839 1.0689655  [3] {Meet} => {Youtube} 0.6129032 0.9500000 0.6451613 1.0155172  [4] {Instagram} => {Youtube} 0.6129032 0.9047619 0.6774194 0.9671593  count  [1] 27  [2] 27  [3] 19  [4] 19  > rules2 <- apriori(appstrans, parameter = list(supp = 0.25, conf = 0.8, minlen = 3))  Apriori  Parameter specification:  confidence minval smax arem aval originalSupport maxtime support  0.8 0.1 1 none FALSE TRUE 5 0.25  minlen maxlen target ext  3 10 rules TRUE  Algorithmic control:  filter tree heap memopt load sort verbose  0.1 TRUE TRUE FALSE TRUE 2 TRUE  Absolute minimum support count: 7  set item appearances ...[0 item(s)] done [0.00s].  set transactions ...[16 item(s), 31 transaction(s)] done [0.00s].  sorting and recoding items ... [8 item(s)] done [0.00s].  creating transaction tree ... done [0.00s].  checking subsets of size 1 2 3 4 done [0.00s].  writing ... [9 rule(s)] done [0.01s].  creating S4 object ... done [0.00s].  > inspect(rules2)  lhs rhs support confidence  [1] {Instagram, SnapChat} => {Youtube} 0.2580645 0.8888889  [2] {SnapChat, Youtube} => {Instagram} 0.2580645 1.0000000  [3] {Instagram, Meet} => {Youtube} 0.3870968 0.9230769  [4] {Meet, Whatsapp} => {Youtube} 0.5483871 1.0000000  [5] {Meet, Youtube} => {Whatsapp} 0.5483871 0.8947368  [6] {Instagram, Whatsapp} => {Youtube} 0.5483871 1.0000000  [7] {Instagram, Youtube} => {Whatsapp} 0.5483871 0.8947368  [8] {Instagram, Meet, Whatsapp} => {Youtube} 0.3225806 1.0000000  [9] {Instagram, Meet, Youtube} => {Whatsapp} 0.3225806 0.8333333  coverage lift count  [1] 0.2903226 0.9501916 8  [2] 0.2580645 1.4761905 8  [3] 0.4193548 0.9867374 12  [4] 0.5483871 1.0689655 17  [5] 0.6129032 1.0272904 17  [6] 0.5483871 1.0689655 17  [7] 0.6129032 1.0272904 17  [8] 0.3225806 1.0689655 10  [9] 0.3870968 0.9567901 10  > rules3 <- apriori(appstrans, parameter = list(supp = 0.25, conf = 0.8, minlen = 4))  Apriori  Parameter specification:  confidence minval smax arem aval originalSupport maxtime support  0.8 0.1 1 none FALSE TRUE 5 0.25  minlen maxlen target ext  4 10 rules TRUE  Algorithmic control:  filter tree heap memopt load sort verbose  0.1 TRUE TRUE FALSE TRUE 2 TRUE  Absolute minimum support count: 7  set item appearances ...[0 item(s)] done [0.00s].  set transactions ...[16 item(s), 31 transaction(s)] done [0.00s].  sorting and recoding items ... [8 item(s)] done [0.00s].  creating transaction tree ... done [0.00s].  checking subsets of size 1 2 3 4 done [0.00s].  writing ... [2 rule(s)] done [0.00s].  creating S4 object ... done [0.00s].  > inspect(rules3)  lhs rhs support confidence  [1] {Instagram, Meet, Whatsapp} => {Youtube} 0.3225806 1.0000000  [2] {Instagram, Meet, Youtube} => {Whatsapp} 0.3225806 0.8333333  coverage lift count  [1] 0.3225806 1.0689655 10  [2] 0.3870968 0.9567901 10  > rules4 <- apriori(appstrans, parameter = list(supp = 0.25, conf = 0.8, minlen = 5))  Apriori  Parameter specification:  confidence minval smax arem aval originalSupport maxtime support  0.8 0.1 1 none FALSE TRUE 5 0.25  minlen maxlen target ext  5 10 rules TRUE  Algorithmic control:  filter tree heap memopt load sort verbose  0.1 TRUE TRUE FALSE TRUE 2 TRUE  Absolute minimum support count: 7  set item appearances ...[0 item(s)] done [0.00s].  set transactions ...[16 item(s), 31 transaction(s)] done [0.00s].  sorting and recoding items ... [8 item(s)] done [0.00s].  creating transaction tree ... done [0.00s].  checking subsets of size 1 2 3 4 done [0.00s].  writing ... [0 rule(s)] done [0.00s].  creating S4 object ... done [0.00s].  > inspect(rules4)  > rules5 <- apriori(appstrans, parameter = list(supp = 0.25, conf = 0.8, minlen = 6))  Apriori  Parameter specification:  confidence minval smax arem aval originalSupport maxtime support  0.8 0.1 1 none FALSE TRUE 5 0.25  minlen maxlen target ext  6 10 rules TRUE  Algorithmic control:  filter tree heap memopt load sort verbose  0.1 TRUE TRUE FALSE TRUE 2 TRUE  Absolute minimum support count: 7  set item appearances ...[0 item(s)] done [0.00s].  set transactions ...[16 item(s), 31 transaction(s)] done [0.00s].  sorting and recoding items ... [8 item(s)] done [0.00s].  creating transaction tree ... done [0.00s].  checking subsets of size 1 2 3 4 done [0.00s].  writing ... [0 rule(s)] done [0.00s].  creating S4 object ... done [0.00s].  > inspect(rules5) |

1. (**3 points**) Identify all the applications that hold the role of the antecedent in the rules where Instagram is the consequent **.**

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Answer:   |  |  | | --- | --- | | Rule | | | **Antecedent** | **Consequent** | | {SnapChat} |  Instagram | | {SnapChat, Youtube} |  Instagram |   **Rcode:**  instagram\_rules <- subset(rules, subset = rhs %in% "Instagram")  inspect(instagram\_rules)  lhs rhs support confidence coverage  [1] {SnapChat} => {Instagram} 0.2903226 1 0.2903226  [2] {SnapChat, Youtube} => {Instagram} 0.2580645 1 0.2580645  lift count  [1] 1.47619 9  [2] 1.47619 8 |

1. (**3 points**) Install the ***arulesViz*** package and load the ***arulesViz*** library. Create a Parallel Coordinates plot and highlight the arrows representing the rules considered in the previous question .

|  |
| --- |
| Answer: **Rcode:**  install.packages("arulesViz")  library(arulesViz)  instagram\_rules <- subset(rules, subset = rhs %in% "Instagram")  plot(rules, method = "paracoord")  plot(instagram\_rules, method = "paracoord",col = "red",add= TRUE  plot(rules, method = "paracoord") |