**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: | Thomas Traianos |

# 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= 1.233 | Cluster for data row 9 = 1 | Dist(1,2)= 3.493 | | Perimeter of cluster 2 prototype= -0.244 | Cluster for data row 55 = 2 | Dist(1,3)= 5.124 | | Perimeter of cluster 3 prototype= -1.011 | Cluster for data row 189 = 3 | Dist(2,3)= 2.638 |   #import seeds data and run set.seed as stated in topic  set.seed(123)  seeds\_orig = read.csv('seeds\_dataset.csv', header=T)  #find the column numbers of seeID,seedType to remove them  rm\_cols = which(colnames(seeds\_orig) %in% c('seeID', 'seedType'))  print(rm\_cols)  [1] 1 9  seeds = seeds\_orig[,-rm\_cols] #seeds without seeID and seedType columns  scseeds = scale(seeds) #scale seeds  print(class(scseeds)) #check the type of scseeds is a matrix  [1] "matrix" "array"  scseeds = as.data.frame(scseeds) #convert it to data frame  print(sapply(scseeds, mean)) #indeed the mean is 0, (very close to zero due to rounding errors) for every attribute  area perimeter compactness  2.851041e-16 1.142919e-16 1.232810e-15  lengthOfKernel widthOfKernel asymmetryCoefficient  -9.485633e-17 -3.082893e-16 -7.657359e-17  lengthOfKernelGroove  -1.130477e-16  print(sapply(scseeds, sd)) #indeed the standard deviation is 1 for every attribute  print(sapply(scseeds, sd))  area perimeter compactness  1 1 1  lengthOfKernel widthOfKernel asymmetryCoefficient  1 1 1  lengthOfKernelGroove  1  c = kmeans(scseeds, 3) #apply kmeans algorithm to scseeds with 3 centers  #print centroids perimeter, rounded  print(round(c$centers[,'perimeter'],3))  1 2 3  1.233 -0.244 -1.011  #print the clusters which belong the data of rows 9,55,189 respectively  print(c$cluster[c(9,55,189)])  [1] 1 2 3  #print the euclidean distance of the centroids, rounded  print(round(dist(c$centers, method='euclidean'),3))  1 2  2 3.493  3 5.124 2.638 |

**(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:

|  |  |  |  |
| --- | --- | --- | --- |
|  | cluster | | |
| seedType | 1 | 2 | 3 |
| 1 | 65 | 2 | 3 |
| 2 | 3 | 67 | 0 |
| 3 | 7 | 0 | 63 |

Precision rate for cluster 1 = 0.867

Recall rate for cluster 1 = 0.929

print(c$cluster)

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

[38] 1 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 3 2 2 2 2 2 2 2 2 3 1 1 1 1

[75] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1

[112] 1 1 1 1 1 1 1 1 1 1 1 1 1 2 1 1 1 1 1 1 1 1 1 1 2 1 1 2 1 3 3 3 3 3 3 3 3

[149] 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 2 3 3 3 3 3 3 3 3 3 3 3 3 3 2 3 3 3 3 3

[186] 3 3 3 3 3 3 3 3 3 3 2 3 2 3 2 3 2 3 3 3 2 3 3 3 3

#I can see c$cluster vector has stored the assignments of cluster 2 first.

#I need to rearrange this in a new vector.

#create a new vector to store integers(cluster assignment) of length

#equal to the length of c$cluster

#then rearrange as stated.

rearranged\_vector = vector(mode='integer', length=length(c$cluster))

rearranged\_vector = vector(mode='integer', length=length(c$cluster))

rearranged\_vector[c$cluster==2] =1

rearranged\_vector[c$cluster==1] =2

rearranged\_vector[c$cluster==3] =3

#create the confusion matrix and print it

library('caret')

seedTypeFactor = factor(seeds\_orig$seedType)

clusterFactor = factor(rearranged\_vector)

conf = confusionMatrix(data=clusterFactor, reference=seedTypeFactor)

print(conf$table)

Reference

Prediction 1 2 3

1 65 3 7

2 2 67 0

3 3 0 63

#The reference is seedType. I will fill the given table with conf table transposed.

**(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) = 0.404

Average Silhouette (4 clusters) = 0.341

Comment on whether the clusters are well separated:

library(cluster) #must be installed

#silhouette coefficients of 3 clusters

sil3 = silhouette(c$cluster, dist(scseeds))

#print the average silhouette of 3 clusters, rounded

print(round(mean(sil3[,3]),3))

[1] 0.404

**Plot (k=3):**

#create clustering with 4 centroids:

set.seed(123)

c4 = kmeans(scseeds,4)

#silhouette of 4 clusters:

sil4 = silhouette(c4$cluster, dist(scseeds))

#print the average silhouette of 4 clusters, rounded

print(round(mean(sil4[,3]),3))

[1] 0.341

**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:

#import europe dataset as europe\_orig. The first row contains headers.

europe\_orig = read.csv('europe\_diet.csv',header=T)

#Inspect and confirm the first column of the dataset contains the names of the countries.

print(head(europe\_orig))

X Other Alcoholic.Beverages Sugar Oils Meat Dairy.Eggs

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 Herzegovina 131 232 178 228 180 300

6 Bulgaria 42 191 271 489 242 292

Fruit.Vegetables Starchy.Roots Pulses Cereals.Grains

1 372 78 50 1144

2 261 110 7 887

3 193 336 0 837

4 203 172 23 905

5 271 139 68 1427

6 130 48 22 1102

#Since I want to manipulate the data, I will store the dataset to variable europe\_orig

europe = europe\_orig

#rename the rownames as the name of the countries.

rownames(europe) = europe[,1]

#delete the first column

europe = europe[,-1]

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 | 270.344 | 315.044 | 493.828 |   #calculate euclidean distance matrix of Greece, Belgium, Spain and Finland, rounded to 3 decimals  distMatrix = round(dist(europe[c('Greece','Spain','Belgium','Finland'),], method='euclidean'),3)  print(distMatrix)  Greece Spain Belgium  Spain 270.344  Belgium 315.044 362.893  Finland 493.828 489.669 667.253 |

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:

library(cluster) #must be installed. Contains agnes() function

#distance matrix of europe with euclidean metric

distMatrixFull = dist(europe, method='euclidean')

#since the distance matrix has been created with euclidean metric,

#I don't have to define the metric in agnes() function

#agglomarative cluster using distance matrix, with complete clustering method

cluster\_complete = agnes(distMatrixFull, method='complete')

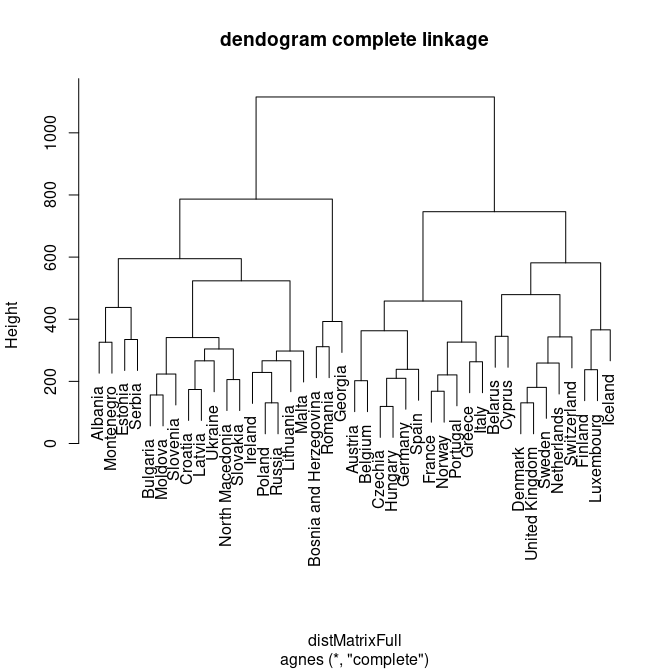
#agglomarative cluster using distance matrix, with single clustering method

cluster\_single = agnes(distMatrixFull, method='single')

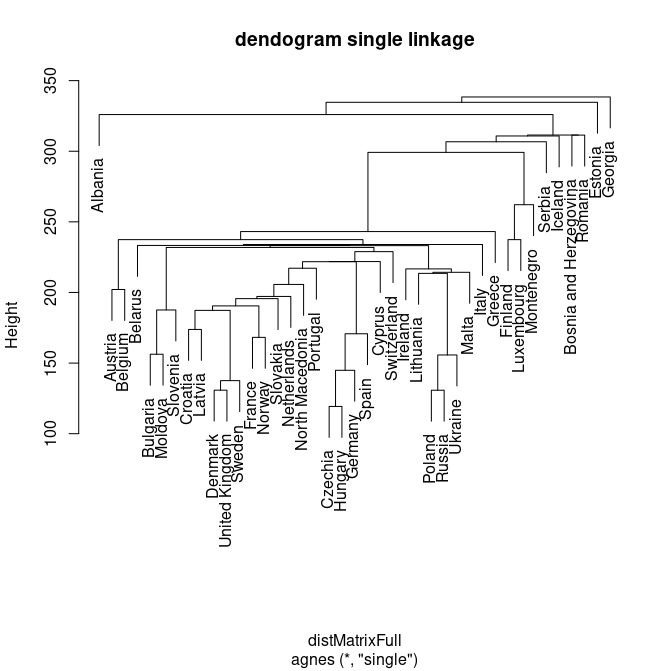
pltree(cluster\_complete, main='dendogram complete linkage') #complete linkage dendogram

pltree(cluster\_single, main='dendogram single linkage') #single linkage dendogram

**Complete 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:

#create the vector with the 7 group memberships as g, from the

#previous complete linkage hierarchical clustering

assignments\_7 = cutree(cluster\_complete, k=7)

#extract the countries names from distMatrixFull

countries = rownames(as.matrix(distMatrixFull))

#find the index numbers of the countries

index\_switz = which(countries == 'Switzerland')

index\_norway = which(countries == 'Norway')

#find in which cluster the countries has been assigned

cluster\_switz = assignments\_7[index\_switz]

cluster\_norway = assignments\_7[index\_norway]

#find the indeces of countries with the same group as Switzerland and Norway

index\_group\_switz = which(assignments\_7==cluster\_switz)

index\_group\_norway = which(assignments\_7==cluster\_norway)

#Group with Switzerland

group\_switz = countries[index\_group\_switz]

print(group\_switz)

[1] "Belarus" "Cyprus" "Denmark" "Netherlands"

[5] "Sweden" "Switzerland" "United Kingdom"

#Group with Norway

group\_norway = countries[index\_group\_norway]

print(group\_norway)

[1] "Austria" "Belgium" "Czechia" "France" "Germany" "Greece"

[7] "Hungary" "Italy" "Norway" "Portugal" "Spain"

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*

Greece\_and\_Cyprus\_max\_cluster = function(){

max\_cluster\_number = 1

while (max\_cluster\_number < length(countries)){

assignments = cutree(cluster\_complete, k=max\_cluster\_number)

index\_greece = which(countries == 'Greece')

cluster\_greece = assignments[index\_greece]

index\_group\_greece = which(assignments==cluster\_greece)

group\_greece = countries[index\_group\_greece]

flag= "Cyprus" %in% group\_greece

if (flag == FALSE){

return (max\_cluster\_number-1) #return the previous max\_cluster\_number, where flag was TRUE

}

else{

max\_cluster\_number = max\_cluster\_number+1

}

}

}

Greece\_and\_Cyprus\_max\_cluster()

[1] 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:

library(arules) #must be installed

appstrans = read.transactions('application\_data.csv', format='basket',sep=',',rm.duplicates=FALSE)

print([appstrans@itemInfo](mailto:appstrans@itemInfo))

labels

1 Amazon

2 Amazon Prime

3 Discord

4 Facebook

5 Hotstar

6 Instagram

7 Meet

8 Netflix

9 Pinterest

10 SnapChat

11 Spotify

12 Twitter

13 Whatsapp

14 Wynk

15 Youtube

16 Zoom

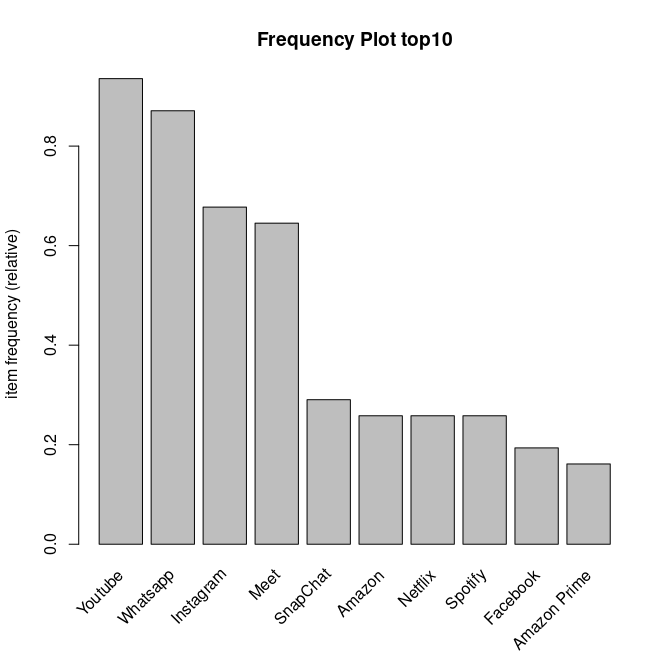
1. (**3 points**) Fill in the table below with the number of students who have installed a specific number of applications

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Answer:   |  |  | | --- | --- | | Number of applications/student | Number of students | | 3 | 2 | | 5 | 19 | | 7 | 1 |   print(summary(appstrans))  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:

itemFrequencyPlot(appstrans, topN=10)



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 | | 9 | 2 | | 7 | 3 | | 2 | 4 | |  |  | |  |  |   #run apriori with the given parameters.  itemsets = apriori(appstrans, parameter=list(support=0.25, confidence=0.8, minlen=2))    #sort by support  itemsets\_by\_sup = sort(itemsets, by = 'support')  inspect(head(itemsets\_by\_sup,4))  lhs rhs support confidence coverage lift count  [1] {Whatsapp} => {Youtube} 0.8709677 1.0000000 0.8709677 1.0689655 27  [2] {Youtube} => {Whatsapp} 0.8709677 0.9310345 0.9354839 1.0689655 27  [3] {Meet} => {Youtube} 0.6129032 0.9500000 0.6451613 1.0155172 19  [4] {Instagram} => {Youtube} 0.6129032 0.9047619 0.6774194 0.9671593 19  print(summary(itemsets))  set of 18 rules  **rule length distribution (lhs + rhs):sizes**  **2 3 4**  **9 7 2**  Min. 1st Qu. Median Mean 3rd Qu. Max.  2.000 2.000 2.500 2.611 3.000 4.000  summary of quality measures:  support confidence coverage lift  Min. :0.2581 Min. :0.8095 Min. :0.2581 Min. :0.9295  1st Qu.:0.2984 1st Qu.:0.8904 1st Qu.:0.2984 1st Qu.:0.9694  Median :0.5484 Median :0.9271 Median :0.5484 Median :1.0273  Mean :0.4785 Mean :0.9316 Mean :0.5161 Mean :1.0640  3rd Qu.:0.5484 3rd Qu.:1.0000 3rd Qu.:0.6452 3rd Qu.:1.0690  Max. :0.8710 Max. :1.0000 Max. :0.9355 Max. :1.4762  count  Min. : 8.00  1st Qu.: 9.25  Median :17.00  Mean :14.83  3rd Qu.:17.00  Max. :27.00  mining info:  data ntransactions support confidence  appstrans 31 0.25 0.8  call  apriori(data = appstrans, parameter = list(support = 0.25, confidence = 0.8, minlen = 2)) |

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 |   #create vector where true is for the rules where "Instagram" in on rhs  insta\_vector = rhs(itemsets) %in% 'Instagram'  inspect(lhs(itemsets)[insta\_vector])  items  [1] {SnapChat}  [2] {SnapChat, Youtube} |

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:  library(arulesViz)  plot(itemsets, method='paracoord') |