

Task 1 – Hypothesis Testing

Q1. State the null hypothesis and the alternative hypothesis, and report the key parameters you set for the testing.

Hypothesis Testing:

Main concept of hypothesis testing is to either accept null hypothesis or reject the null hypothesis and moving to alternate hypothesis.

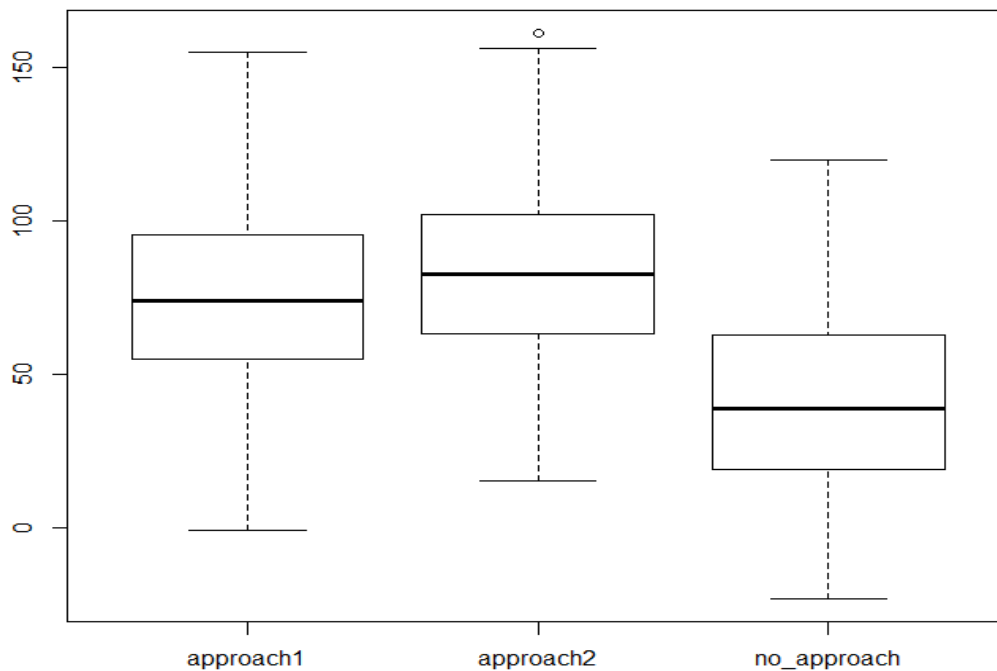
Consider a scenario of two different approaches like approach_1 and approach_2. for these two approaches we can derive two statements for hypothesis testing which belongs to Null hypothesis and alternate hypothesis.

Null hypothesis (H_0): Approach_1 and Approach_2 has no difference. (**P-value > 0.05**)

Alternative hypothesis (H_A): Approach_1 is better than approach_2. (**P-value < 0.05**)

Key parameters for selecting hypothesis testing is p-value which should be compared with alpha value (level of significance) which is equal to .05.

Data Exploration: plot(dataset)

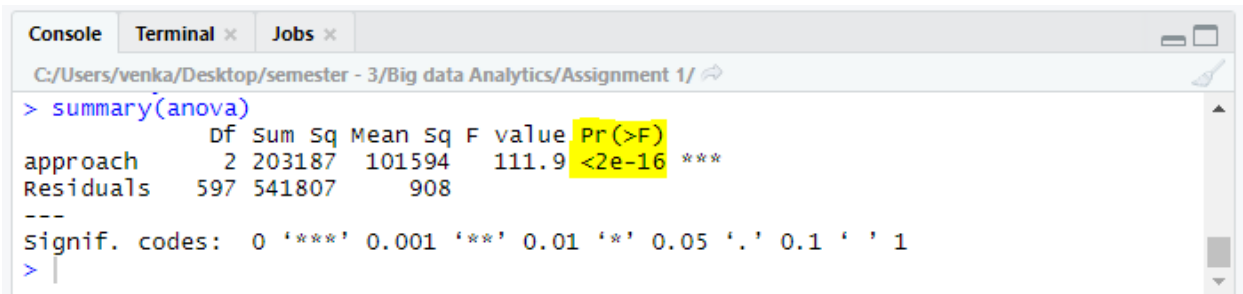


Based on the Question given for task-1, we need to select one of the approaches like t-test, Wilcoxon Rank-sum Test, Power and sample size, Anova test to get this P-value. By proper data exploration on A1_performance_test.csv, we can observe that we are dealing with **three** population data. So Anova test will be best for handling this kind of dataset, also given question

is similar to example stated in textbook (Data Science and Big Data Analytics: Discovering, Analyzing, Visualizing and Presenting Data).

Function Module used: `anova <- aov(performance ~ approach, data = performancedata)`

Output by Summary function:



```
> summary(anova)
          Df Sum Sq Mean Sq F value Pr(>F)
approach    2 203187   101594   111.9 <2e-16 ***
Residuals  597 541807     908
---
signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
> |
```

Fig -1

From figure 1 highlighted part is value of P.

$P\text{-value} = 2.2e-16 = 2.2 \times 10^{-16} = .000000000000000022$

As $P\text{-value} < 0.05$ (Alpha value), **So rejecting null hypothesis.**

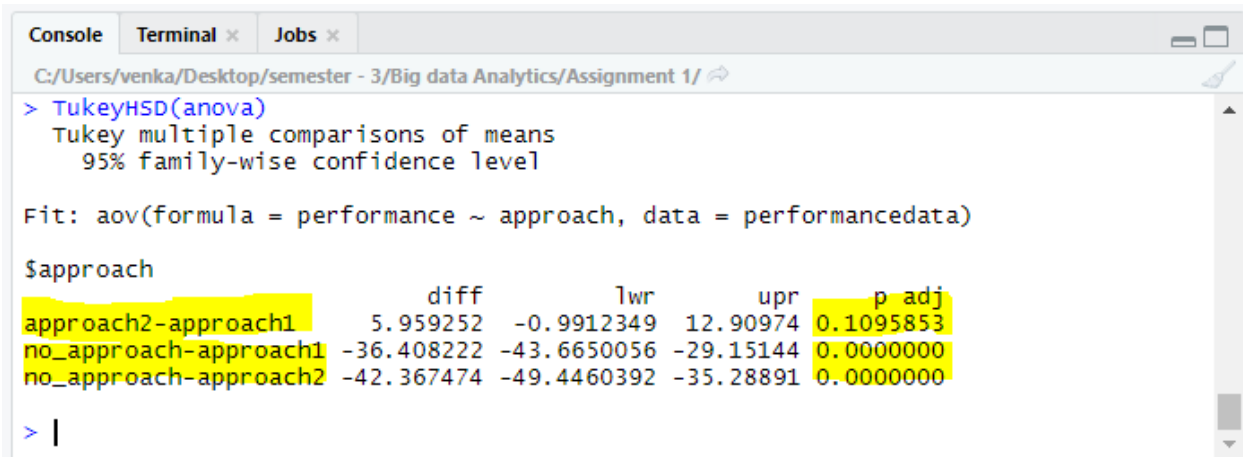
Q2. Answer the above two questions and sufficiently justify your conclusions by using the analysis results.

Q2 - 1. Whether the two new learning approaches can effectively improve student learning performance?

This question is dealing with the comparison analysis between any two approaches, which can be obtained by using function module : TukeyHSD()

Function Module used: TukeyHSD(anova)

Output:



```
> TukeyHSD(anova)
Tukey multiple comparisons of means
 95% family-wise confidence level

Fit: aov(formula = performance ~ approach, data = performancedata)

$approach
              diff            lwr            upr      p adj
approach2-approach1  5.959252 -0.9912349 12.90974 0.1095853
no_approach-approach1 -36.408222 -43.6650056 -29.15144 0.0000000
no_approach-approach2 -42.367474 -49.4460392 -35.28891 0.0000000
> |
```

From above figure result of P-value are shared among population:

“no_approach-approach1” = 0 (Rejecting Null hypothesis P-value<.05)

Final Statement: Much difference between no_approach and approach1

“no_approach-approach2” = 0 (Rejecting Null hypothesis P-value<.05)

Final Statement: Much difference between no_approach and approach2

“approach2-approach1” = 0.1095853 (Accepting null hypothesis P>.05)

Final Statement: Both approach1 and approach2 has same effect.

Hence, we can conclude that both the approach1 and approach2 effectively improve student learning with respect to noapproach.

Q2 - 2. In terms of improving student learning performance, whether the two approaches are significantly different from each other?

From fig-2 P-value for approach2 against approach1 is 0.1095853

“approach2-approach1” = 0.1095853 (Accepting null hypothesis P>.05)

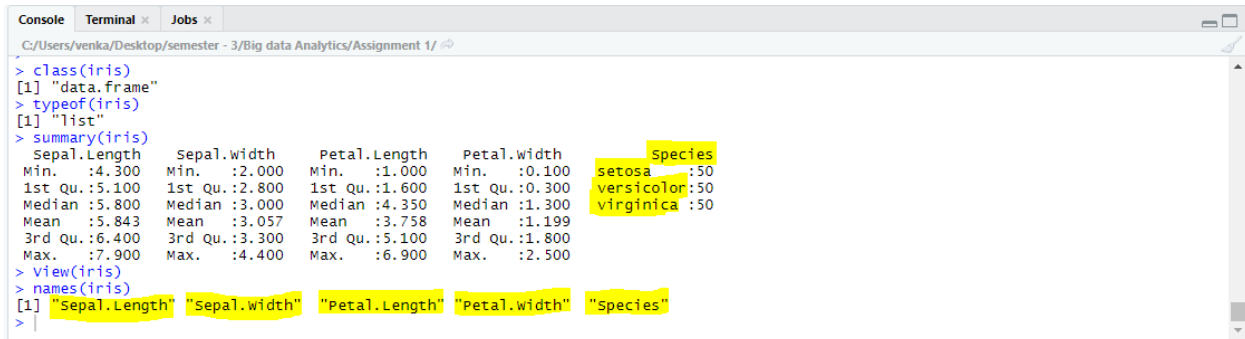
Final Statement: Both approach1 and approach2 has same effect.

Conclusion: Both approach1 and approach2 has same effect in terms of improving learning performance.

Task 2 – Clustering

Q1. Describe your observation on this dataset such as the number of examples, the number of features, and the meaning of these features. You shall also use summary() function to help you gain more understanding.

Data Exploration:



```
> class(iris)
[1] "data.frame"
> typeof(iris)
[1] "list"
> summary(iris)
  Sepal.Length   Sepal.Width   Petal.Length   Petal.Width   Species
Min.   :4.300   Min.   :2.000   Min.   :1.000   Min.   :0.100   setosa   :50
1st Qu.:5.100   1st Qu.:2.800   1st Qu.:1.600   1st Qu.:0.300   versicolor:50
Median :5.800   Median :3.000   Median :4.350   Median :1.300   virginica :50
Mean   :5.843   Mean   :3.057   Mean   :3.758   Mean   :1.199
3rd Qu.:6.400   3rd Qu.:3.300   3rd Qu.:5.100   3rd Qu.:1.800
Max.   :7.900   Max.   :4.400   Max.   :6.900   Max.   :2.500
> view(iris)
> names(iris)
[1] "Sepal.Length" "Sepal.Width" "Petal.Length" "Petal.Width" "Species"
```

From above figure using class() function module help in finding class type of “data.frame” and by typeof() we can find its data type as “list”.

By using summary(iris_dataset) we can observe that we are dealing with five features namely: "Sepal.Length", "Sepal.Width", "Petal.Length", "Petal.Width", "Species". Only three species available namely: Setosa, versicolor and Virginica.

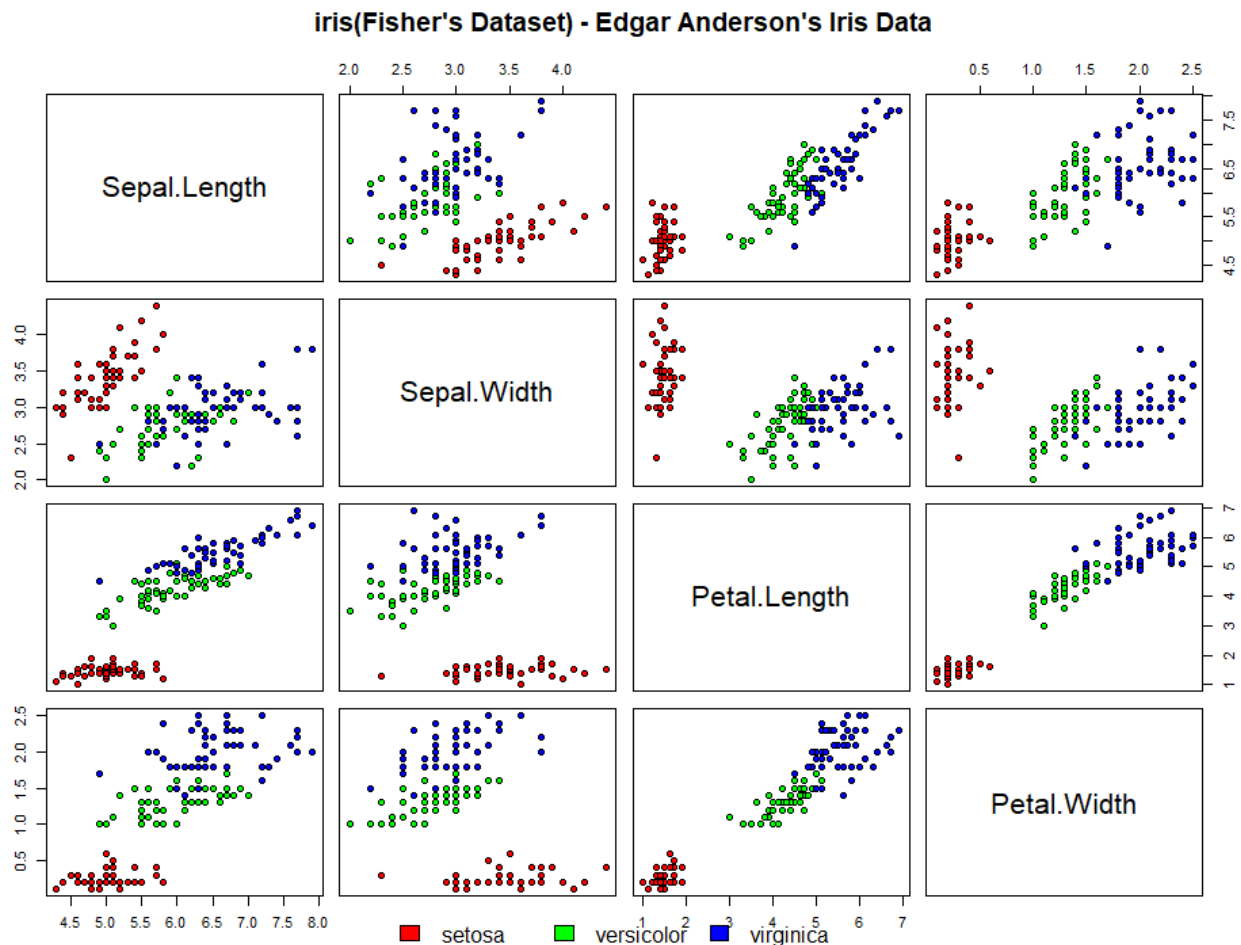
By using view() we can observe that it has 150 records. Has both numerical (Sepal.Length, Sepal.Width, Petal.Length & Petal.Width) and textual data(Species). All measurements of four features are given in centimeters for all three species.

Q2. Plot the scatterplot matrix of Iris dataset to visualize the pairwise relationships among the four attributes.

Function modules used:

- c("'", "", ",") - To set colors for species
- pairs(dataset, main="iris(Fisher's Dataset) - Edgar Anderson's Iris Data",
pch = 21, bg = colors[unclass(iris\$Species)])) – To draw plot matrix.
- Par(xpd=TRUE) – To help ledger for plotting.
- legend(0.3, 0.03, horiz = TRUE, as.vector(unique(iris\$Species)),
fill = colors, bty = "n") – To display the species names.

Output:



Observations:

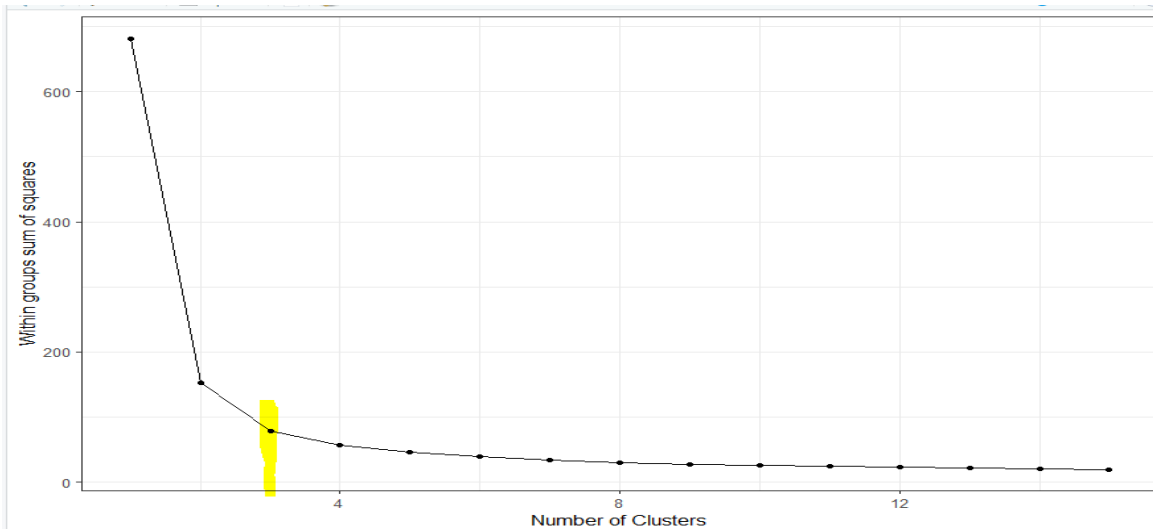
From above figure we can observe that setosa species can be easily differentiated from Versicolor and Virginica (which has lower petal length and width). But virginica and versicolor cannot be easily separated as measurement of their sepal and petal width/length are too closely equal.

Q3. Perform K-means clustering analysis on the Iris dataset and report your result. Explain how you choose the number of clusters and justify your choice.

Choosing number of clusters is the most important task for K-means clustering. I used a method called WSS (Within Sum of Squares) which was used for tutorials.

Here only numeric data must be considered for getting sum of squares for clusters count. Proper numeric data handling code is written and got below graph by using ggplot().

Output:



From the above graph K value needs to be selected from the number of clusters. Highlighted mark from the above image gives a huge change in within sum of squares, Hence K=3 is selected as best value. At K=3 the “elbow” finding is obtained from image, moving lower of K(K>3) value will leads to linear WSS values.

No of Cluster for K-Means = 3

Applying K-Mean Clustering with K=3:

Function module used: `km = kmeans(kmeansdata,3, nstart=25)`

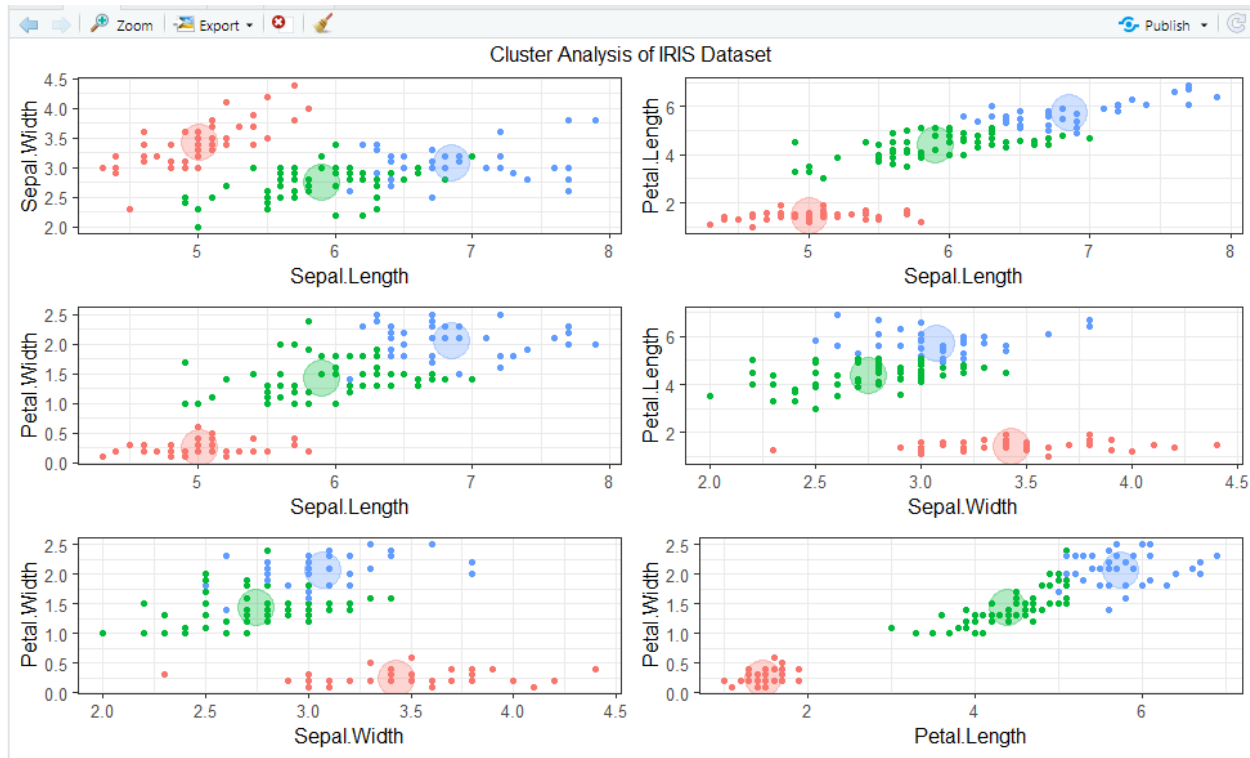
nstart is taken as 25 which means R will try 25 different random starting assignments and get the lowest within cluster variation.

Output:

[illegible]

Q4. Find ways to visualize your clustering result and perform diagnostics to answer the following questions:

By using function modules ggplot() and grid.arrange() we will display the graph below. Before plotting dataset preparation must be done with the obtained K-Means clusters.



Q4 - a. Are the clusters well separated from each other?

By observations clusters are not separated much. Means for the clusters are almost similar and identical.

Q4 - b. Do any of the clusters have only a few points?

By proper observation cluster 3 has only few points.

Q4 - c. Do any of the centroids appear to be too close to each other?

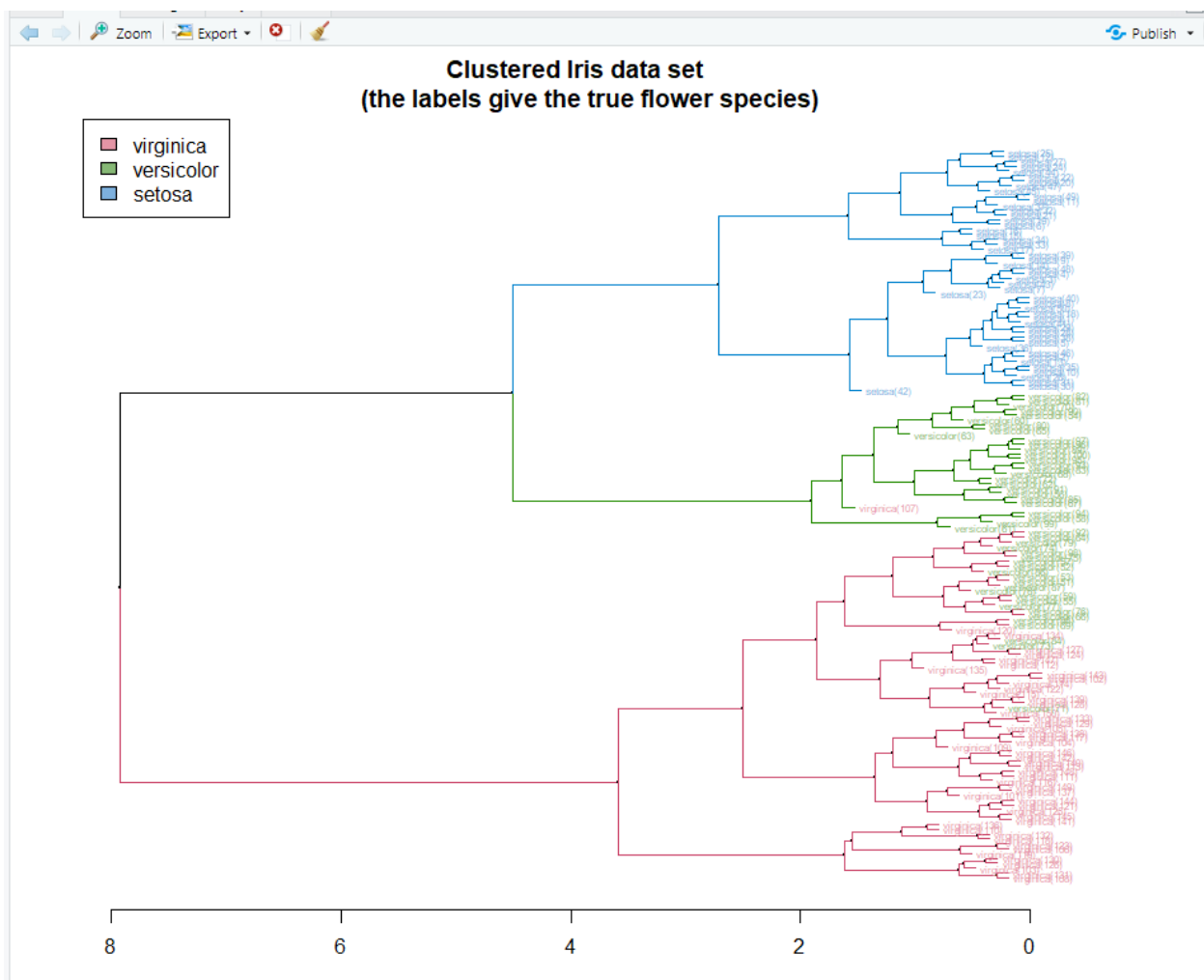
Cluster 2 and 3 seems to be nearer. petal width vs Sepal Width of cluster3 and petal length vs sepal width of cluster2

Q5. (For CSCI946 student ONLY) Learn to perform hierarchical agglomerative clustering via `hclust()` function and compare the clustering result with that obtained with K-means clustering.

Reference taken from: https://cran.r-project.org/web/packages/dendextend/vignettes/Cluster_Analysis.html

By using above web site hierarchical clustering is obtained by using `hclust()` function. External libraries used for code are library (dendextend), library (colorspace). By using function modules like `plot()` – for plotting graph and `legend()` – for adding labels, obtained below image:

Output:



Q6. Show the output of your code on this data and attach your code at the end of the report. – Code attached at last & results shared above with images.

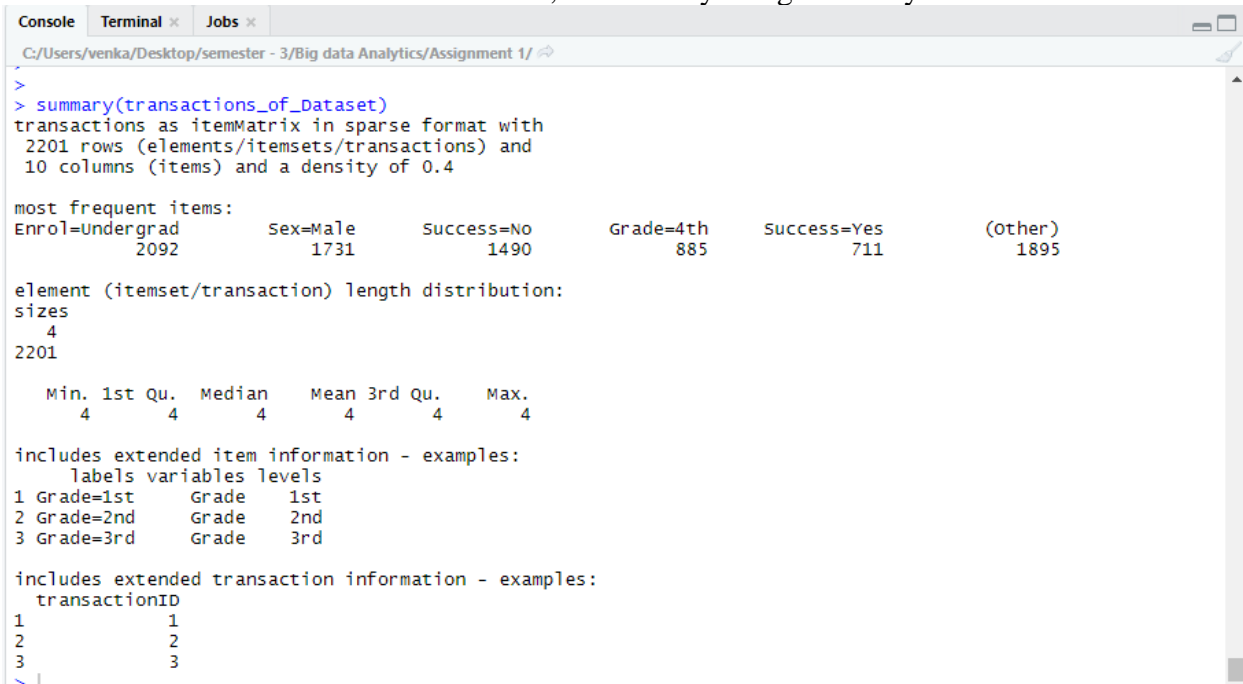
Task 3 – Association Rule

Q1. Generate frequent itemsets by applying various “support” thresholds and inspect these itemsets by displaying their support, confidence, and lift values.

Main Aim of association rule is to answer these kinds of questions like:

- 1.) which kind of products got together?
- 2.) Of those customers who are similar to this person, what products do they tend to buy?
- 3.) Of those customers who have purchased this product, what other similar products do they tend to view or purchase?

This huge dataset needs to be breakdown into transactions by using as() function module. Itemset can be derived based on the transaction-ID, obtained by using summary of transactions.



```
> summary(transactions_of_Dataset)
transactions as itemMatrix in sparse format with
2201 rows (elements/itemsets/transactions) and
10 columns (items) and a density of 0.4

most frequent items:
Enrol=Undergrad    Sex=Male    Success=No    Grade=4th    Success=Yes    (other)
      2092         1731         1490         885         711         1895

element (itemset/transaction) length distribution:
sizes
4
2201

   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
     4       4       4       4       4       4

includes extended item information - examples:
  labels variables levels
1 Grade=1st      Grade   1st
2 Grade=2nd      Grade   2nd
3 Grade=3rd      Grade   3rd

includes extended transaction information - examples:
 transactionID
1             1
2             2
3             3
> |
```

To generate the frequent itemsets we need to use **apriori()** function module with a support and confidence value as input to the function module. Apriori algorithm takes a bottom-up iterative approach to uncover the frequent itemsets by first determining all the possible items and then identifying which among them are frequent based on support, confidence, and lift values.

Support of an item set is defined as percentage of transactions that contain that item set. As example shared in text, If 80% of all transactions contains itemset{bread}, then the support is 0.8 and if 70% of all transactions contain a itemset{bread, egg} then the support of {bread, egg} is 0.7.

Confidence is like measure of certainty or trustworthiness associated with each discovered rule.

Mathematical notation: Confidence (X -> Y) = Support (X^Y) / Support (X)

For example, if{bread, eggs, milk} has a support of 0.15 and {bread, eggs} also has a support of 0.15, the confidence of rule {bread, eggs}->{milk} is 1, which means 100% of the time a

customer buys bread and eggs, milk is bought as well. Above rule is true for 100% of the transactions.

Results of itemset1 with support = 0.01 and confidence = 0.3 is shared below:

```

Console Terminal x Jobs x
C:/Users/venka/Desktop/semester - 3/Big data Analytics/Assignment 1/

> #support threshold at 0.01
> itemset1 = apriori(transactions_of_Dataset, parameter = list(support = 0.01, confidence = 0.3))
Apriori

Parameter specification:
confidence minval smax arem aval originalsupport maxtime support minlen maxlen target ext
0.3 0.1 1 none FALSE TRUE 5 0.01 1 10 rules FALSE

Algorithmic control:
filter tree heap memopt load sort verbose
0.1 TRUE TRUE FALSE TRUE 2 TRUE

Absolute minimum support count: 22

set item appearances ... [0 item(s)] done [0.00s].
set transactions ... [10 item(s), 2201 transaction(s)] done [0.00s].
sorting and recoding items ... [10 item(s)] done [0.00s].
creating transaction tree ... done [0.00s].
checking subsets of size 1 2 3 4 done [0.00s].
writing ... [165 rule(s)] done [0.00s].
creating S4 object ... done [0.00s].
> summary(itemset1@quality)
      support      confidence      lift      count
Min. :0.01045 Min. :0.3012 Min. :0.3883 Min. : 23.0
1st Qu.:0.04089 1st Qu.:0.4912 1st Qu.:0.9665 1st Qu.: 90.0
Median :0.07587 Median :0.7319 Median :1.0580 Median : 167.0
Mean :0.15954 Mean :0.6975 Mean :1.3687 Mean : 351.2
3rd Qu.:0.21627 3rd Qu.:0.9158 3rd Qu.:1.4215 3rd Qu.: 476.0
Max. :0.95048 Max. :1.0000 Max. :3.9855 Max. :2092.0
> itemset1
set of 165 rules
> #Inspect the itemsets by displaying the property support, confidence, and lift
> inspect(head(sort(itemset1, by = "support"), 10))
      lhs      rhs      support confidence lift count
[1] {} => {Enrol=Undergrad} 0.9504771 0.9504771 1.000000 2092
[2] {} => {Sex=Male} 0.7864607 0.7864607 1.000000 1731
[3] {Sex=Male} => {Enrol=Undergrad} 0.7573830 0.9630272 1.013204 1667
[4] {Enrol=Undergrad} => {Sex=Male} 0.7573830 0.7968451 1.013204 1667
[5] {} => {Success=No} 0.6769650 0.6769650 1.000000 1490
[6] {Success=No} => {Enrol=Undergrad} 0.6533394 0.9651007 1.015386 1438
[7] {Enrol=Undergrad} => {Success=No} 0.6533394 0.6873805 1.015386 1438
[8] {Success=No} => {Sex=Male} 0.6197183 0.9154362 1.163995 1364
[9] {Sex=Male} => {Success=No} 0.6197183 0.7879838 1.163995 1364
[10] {Sex=Male, Success=No} => {Enrol=Undergrad} 0.6038164 0.9743402 1.025106 1329
>

```

Results of itemset2 with support = 0.02 and confidence = 0.6 is shared below:

```

Console Terminal Jobs
C:/Users/venka/Desktop/semester - 3/Big data Analytics/Assignment 1/

> #support threshold at 0.02
> itemset2 = apriori(transactions_of_Dataset, parameter = list(support = 0.02,confidence = 0.6))
Apriori

Parameter specification:
confidence minval smax arem aval originalsupport maxtime support minlen maxlen target ext
0.6 0.1 1 none FALSE TRUE 5 0.02 1 10 rules FALSE

Algorithmic control:
filter tree heap memopt load sort verbose
0.1 TRUE TRUE FALSE TRUE 2 TRUE

Absolute minimum support count: 44

set item appearances ...[0 item(s)] done [0.00s].
set transactions ...[10 item(s), 2201 transaction(s)] done [0.00s].
sorting and recoding items ... [10 item(s)] done [0.00s].
creating transaction tree ... done [0.00s].
checking subsets of size 1 2 3 4 done [0.00s].
writing ... [95 rule(s)] done [0.00s].
creating s4 object ... done [0.00s].
> summary(itemset2@quality)
      support      confidence      lift      count
Min. :0.02181 Min. :0.6076 Min. :0.7726 Min. : 48.0
1st Qu.:0.05452 1st Qu.:0.7773 1st Qu.:0.9681 1st Qu.:120.0
Median :0.08950 Median :0.8774 Median :1.0521 Median :197.0
Mean :0.20155 Mean :0.8581 Mean :1.3489 Mean : 443.6
3rd Qu.:0.30077 3rd Qu.:0.9672 3rd Qu.:1.2336 3rd Qu.: 662.0
Max. :0.95048 Max. :1.0000 Max. :3.9855 Max. :2092.0
> itemset2
set of 95 rules
> inspect(head(sort(itemset2, by = "support"), 10))
      lhs      rhs      support confidence lift count
[1] {} => {Enrol=Undergrad} 0.9504771 0.9504771 1.000000 2092
[2] {} => {Sex=Male} 0.7864607 0.7864607 1.000000 1731
[3] {Sex=Male} => {Enrol=Undergrad} 0.7573830 0.9630272 1.013204 1667
[4] {Enrol=Undergrad} => {Sex=Male} 0.7573830 0.7968451 1.013204 1667
[5] {} => {Success=No} 0.6769650 0.6769650 1.000000 1490
[6] {Success=No} => {Enrol=Undergrad} 0.6533394 0.9651007 1.015386 1438
[7] {Enrol=Undergrad} => {Success=No} 0.6533394 0.6873805 1.015386 1438
[8] {Success=No} => {Sex=Male} 0.6197183 0.9154362 1.163995 1364
[9] {Sex=Male} => {Success=No} 0.6197183 0.7879838 1.163995 1364
[10] {Sex=Male,Success=No} => {Enrol=Undergrad} 0.6038164 0.9743402 1.025106 1329
>

```

Results of itemset2 with support = 0.6 and confidence = 0.8 is shared below:

```

Console Terminal x Jobs x
C:/Users/venka/Desktop/semester - 3/Big data Analytics/Assignment 1/
[10] {Sex=Male,Success=No} => {Enrol=Undergrad} 0.6038164 0.9743402 1.025106 1329
> #support threshold at 0.6
> itemset3 = apriori(transactions_of_Dataset, parameter = list(support = 0.6,confidence = 0.8,minlen = 1,maxlen = 8))
Apriori

Parameter specification:
confidence minval smax arem aval originalsupport maxtime support minlen maxlen target ext
0.8 0.1 1 none FALSE TRUE 5 0.6 1 8 rules FALSE

Algorithmic control:
filter tree heap memopt load sort verbose
0.1 TRUE TRUE FALSE TRUE 2 TRUE

Absolute minimum support count: 1320

set item appearances ...[0 item(s)] done [0.00s].
set transactions ...[10 item(s), 2201 transaction(s)] done [0.00s].
sorting and recoding items ... [3 item(s)] done [0.00s].
creating transaction tree ... done [0.00s].
checking subsets of size 1 2 3 done [0.00s].
writing ... [6 rule(s)] done [0.00s].
creating s4 object ... done [0.00s].
> summary(itemset3@quality)
      support      confidence      lift      count
Min. :0.6038 Min. :0.9154 Min. :1.000 Min. :1329
1st Qu.:0.6078 1st Qu.:0.9308 1st Qu.:1.014 1st Qu.:1338
Median :0.6365 Median :0.9568 Median :1.020 Median :1401
Mean :0.6981 Mean :0.9488 Mean :1.065 Mean :1536
3rd Qu.:0.7314 3rd Qu.:0.9646 3rd Qu.:1.129 3rd Qu.:1610
Max. :0.9505 Max. :0.9743 Max. :1.175 Max. :2092
> itemset3
set of 6 rules
> inspect(head(sort(itemset3, by = "support"), 10))
      lhs      rhs      support confidence lift count
[1] {}      => {Enrol=Undergrad} 0.9504771 0.9504771 1.000000 2092
[2] {Sex=Male}      => {Enrol=Undergrad} 0.7573830 0.9630272 1.013204 1667
[3] {Success=No}    => {Enrol=Undergrad} 0.6533394 0.9651007 1.015386 1438
[4] {Success=No}    => {Sex=Male} 0.6197183 0.9154362 1.163995 1364
[5] {Sex=Male,Success=No} => {Enrol=Undergrad} 0.6038164 0.9743402 1.025106 1329
[6] {Enrol=Undergrad,Success=No} => {Sex=Male} 0.6038164 0.9242003 1.175139 1329
>

```

From the above three images as support and confidence increases the rules always decreases as shared below:

itemset1 - set of 165 rules
 itemset2 - set of 95 rules
 itemset3 - set of 6 rules

Also lift, value reaches to max stating that there is no coincidence in occurring of lhs->rhs.

Q2. Set the right hand side (rhs) as the attribute “Success” to generate the frequent itemsets that can help to predict if a student can pass this test or not based on his/her grade, gender, and/or enrolment.

Minimum value for support is 0.02 and confidence is 0.6. As specified for question setting the RHS to success with parameters yes and no can be done by using the function module below:

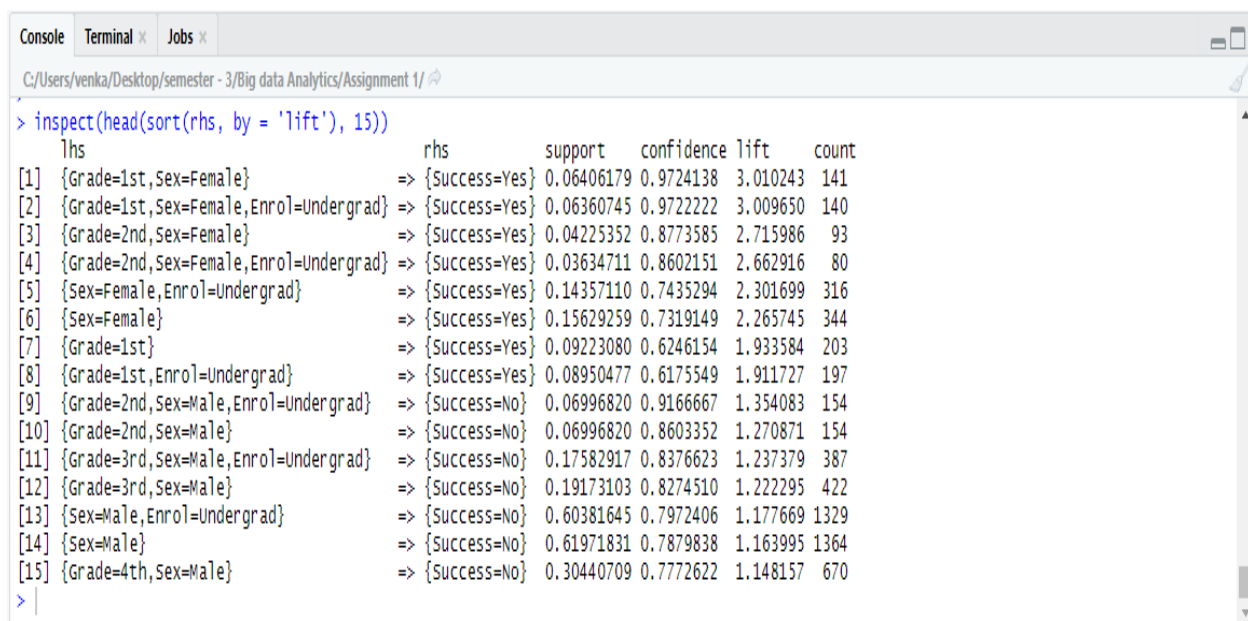
```
rhs = apriori ( transactions_of_Dataset, parameter = list(support = 0.02,confidence = 0.6),  
appearance = list(rhs = c('Success=No', 'Success=Yes'), default = 'lhs'))
```

dataset must be provided in transactions.

Inspect the results using this command:

```
inspect(head(sort(rhs, by = 'lift'), 15))
```

Which displays only 15 transactions in below figure:

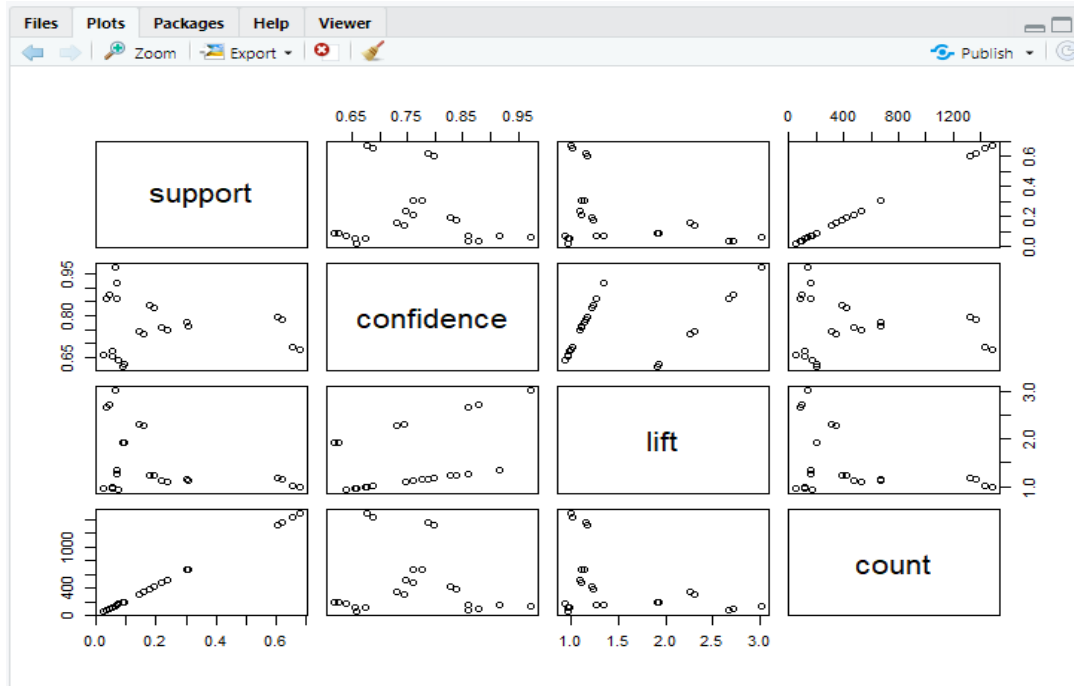


	lhs	rhs	support	confidence	lift	count
[1]	{Grade=1st,Sex=Female}	=> {Success=Yes}	0.06406179	0.9724138	3.010243	141
[2]	{Grade=1st,Sex=Female,Enrol=Undergrad}	=> {Success=Yes}	0.06360745	0.9722222	3.009650	140
[3]	{Grade=2nd,Sex=Female}	=> {Success=Yes}	0.04225352	0.8773585	2.715986	93
[4]	{Grade=2nd,Sex=Female,Enrol=Undergrad}	=> {Success=Yes}	0.03634711	0.8602151	2.662916	80
[5]	{Sex=Female,Enrol=Undergrad}	=> {Success=Yes}	0.14357110	0.7435294	2.301699	316
[6]	{Sex=Female}	=> {Success=Yes}	0.15629259	0.7319149	2.265745	344
[7]	{Grade=1st}	=> {Success=Yes}	0.09223080	0.6246154	1.933584	203
[8]	{Grade=1st,Enrol=Undergrad}	=> {Success=Yes}	0.08950477	0.6175549	1.911727	197
[9]	{Grade=2nd,Sex=Male,Enrol=Undergrad}	=> {Success=No}	0.06996820	0.9166667	1.354083	154
[10]	{Grade=2nd,Sex=Male}	=> {Success=No}	0.06996820	0.8603352	1.270871	154
[11]	{Grade=3rd,Sex=Male,Enrol=Undergrad}	=> {Success=No}	0.17582917	0.8376623	1.237379	387
[12]	{Grade=3rd,Sex=Male}	=> {Success=No}	0.19173103	0.8274510	1.222295	422
[13]	{Sex=Male,Enrol=Undergrad}	=> {Success=No}	0.60381645	0.7972406	1.177669	1329
[14]	{Sex=Male}	=> {Success=No}	0.61971831	0.7879838	1.163995	1364
[15]	{Grade=4th,Sex=Male}	=> {Success=No}	0.30440709	0.7772622	1.148157	670

Above figure shows the prediction of student can pass this test based on his/her grade, gender and/or enrolment will be from rows one to eight. In vise versa we can also predict failure based on lhs value itemsets which are from rows 8 to 15.

Q3. Visualize the rules generated in the last step by 1) showing the relationship among support, confidence, and lift and

Rules generated by RHS can be plotted to know the effects of support, confidence, and lift on these rules. for task 1, function module used is plot() to show relationship among the parameters.

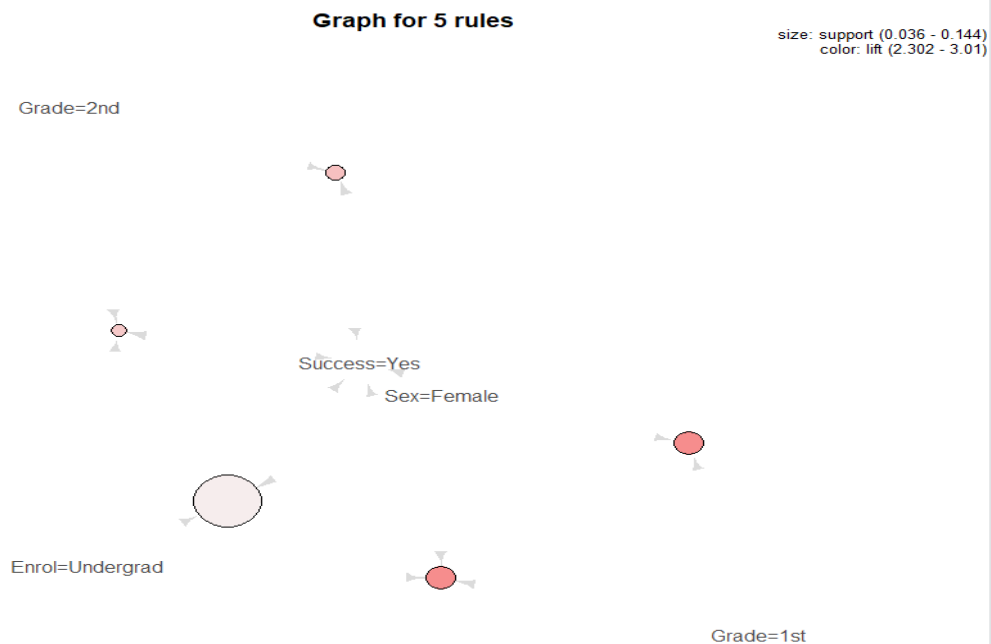


Q3 - 2) using the graph visualization based on the sorted lift value.

Function Module used:

```
lift <- head(sort(rhs, by = "lift"), 5)
```

```
plot(lift, method = "graph", control = list(type = "items"))
```



Task 1

```
# Task 1

# Stendent Number: 6228975

# student Name: Venkata Sandeep Kumar Karamsetty

# Student ID: vskk033@uowmail.edu.au


# Reading file or data

# For setting current directing to working directory.

setwd("./")


# Reading of .CSV file

performancedata <- read.csv("./A1_performance_test.csv")


# Data Exploration of performance data collected.

class(performancedata)

typeof(performancedata)

summary(performancedata)

plot(performancedata)


# Slecting anova test as number of parameters > 3

anova <- aov(performance ~ approach, data = performancedata)


#summary of Annova Results

summary(anova)

#Calculate mean to check better approach

mean(performancedata$approach == "approach1")

mean(performancedata$approach == "approach2")

mean(performancedata$approach == "no_approach")

# Function module to perform pair-wise tests for difference of means.

TukeyHSD(anova)
```

Task 2

Task 2

Student Number: 6228975

student Name: Venkata Sandeep Kumar Karamsetty

Student ID: vskk033@uowmail.edu.au

install packages, if necessary

library(ggplot2)

library(dplyr)

library(cluster)

library(lattice)

library(graphics)

library(grid)

library(gridExtra)

library(dendextend)

library(stats)

library(colorspace)

1. Describe your observation on this dataset such as the number of examples, the number of features,

and the meaning of these features. You shall also use summary() function to help you gain more understanding.

class(iris)

typeof(iris)

summary(iris)

View(iris)

names(iris)

#2. Plot the scatterplot matrix of Iris dataset to visualize the pairwise relationships between the four attributes

defining colors

colors <- c("red", "green", "blue")

drawing plot matrix


```

pairs(iris[1:4], main = "iris(Fisher's Dataset) - Edgar Anderson's Iris Data",
      pch = 21, bg = colors[unclass(iris$Species)])

# graphical parameter setting to clip plotting to the figure region
par(xpd = TRUE)

# adding legend to plot
legend(0.3, 0.03, horiz = TRUE, as.vector(unique(iris$Species)),
      fill = colors, bty = "n")

#3.Perform K-means clustering analysis on the Iris dataset and report your result.
# Explain how you choose the number of clusters and justify your choice.

Numeric_Of_IRIS <- select(iris, -Species) # make a data table with only the numeric measurements from iris
wss <- (nrow(Numeric_Of_IRIS)-1)*sum(apply(Numeric_Of_IRIS,2,var))
for (i in 2:15) wss[i] <- sum(kmeans(Numeric_Of_IRIS,
                                   nstart=25,
                                   centers=i)$withinss)

Data <- data.frame(centers=1:15, wss)
ggplot(Data, aes(x=centers, y=wss)) + geom_point() + geom_line() +
  xlab("Number of Clusters") + ylab("Within groups sum of squares")

# Kmeans Clustering

Data = as.data.frame(iris)
kmeansdata_orig = as.matrix(Data[,c("Sepal.Length", "Sepal.Width", "Petal.Length", "Petal.Width", "Species")])
kmeansdata <- kmeansdata_orig[,1:4]
kmeansdata[1:10,]

km = kmeans(kmeansdata,3, nstart=25)
km

#4. preparing iris data with kmeans clustering results for plotting

```

```

prepareiris = as.data.frame(iris)
prepareiris$cluster = factor(km$cluster)
centers=as.data.frame(km$centers)

g1=ggplot(data=prepareiris, aes(x=Sepal.Length, y=Sepal.Width, color=cluster )) +
  geom_point() + theme(legend.position="right") +
  geom_point(data=centers, aes(x=Sepal.Length,y=Sepal.Width, color=as.factor(c(1,2,3))),
    size=10, alpha=.3, show.legend=FALSE)

g2 =ggplot(data=prepareiris, aes(x=Sepal.Length, y=Petal.Length, color=cluster )) +
  geom_point() +
  geom_point(data=centers, aes(x=Sepal.Length,y=Petal.Length, color=as.factor(c(1,2,3))),
    size=10, alpha=.3, show.legend=FALSE)

g3 = ggplot(data=prepareiris, aes(x=Sepal.Length, y=Petal.Width, color=cluster )) +
  geom_point() +
  geom_point(data=centers, aes(x=Sepal.Length,y=Petal.Width, color=as.factor(c(1,2,3))),
    size=10, alpha=.3, show.legend=FALSE)

g4 = ggplot(data=prepareiris, aes(x=Sepal.Width, y=Petal.Length, color=cluster )) +
  geom_point() +
  geom_point(data=centers, aes(x=Sepal.Width,y=Petal.Length, color=as.factor(c(1,2,3))),
    size=10, alpha=.3, show.legend=FALSE)

g5 = ggplot(data=prepareiris, aes(x=Sepal.Width, y=Petal.Width, color=cluster )) +
  geom_point() +
  geom_point(data=centers, aes(x=Sepal.Width,y=Petal.Width, color=as.factor(c(1,2,3))),
    size=10, alpha=.3, show.legend=FALSE)

g6 = ggplot(data=prepareiris, aes(x=Petal.Length, y=Petal.Width, color=cluster )) +
  geom_point() +
  geom_point(data=centers, aes(x=Petal.Length,y=Petal.Width, color=as.factor(c(1,2,3))),

```

```

size=10, alpha=.3, show.legend=FALSE)

grid.arrange(arrangeGrob(g1 + theme(legend.position="none"),
  g2 + theme(legend.position="none"),
  g3 + theme(legend.position="none"),
  g4 + theme(legend.position="none"),
  g5 + theme(legend.position="none"),
  g6 + theme(legend.position="none"),
  top = "Cluster Analysis of IRIS Dataset", ncol=2, nrow = 4))

# 5. Learn to perform hierarchical agglomerative clustering via hclust()
# function and compare the clustering result with that obtained with K-means clustering.
# Reference taken from: https://cran.r-project.org/web/packages/dendextend/vignettes/Cluster\_Analysis.html

d_iris <- dist(iris) # method="man" # is a bit better
hc_iris <- hclust(d_iris, method = "complete")
iris_species <- rev(levels(iris[,5]))

dend <- as.dendrogram(hc_iris)
# order it the closest we can to the order of the observations:
dend <- rotate(dend, 1:150)

# Color the branches based on the clusters:
dend <- color_branches(dend, k=3) #, groupLabels=iris_species)

# Manually match the labels, as much as possible, to the real classification of the flowers:
labels_colors(dend) <-
  rainbow_hcl(3)[sort_levels_values(
    as.numeric(iris[,5])[order.dendrogram(dend)]
  )]

# We shall add the flower type to the labels:

```

```

labels(dend) <- paste(as.character(iris[,5])[order.dendrogram(dend)],
                      "(",labels(dend),")",
                      sep = "")

# We hang the dendrogram a bit:
dend <- hang.dendrogram(dend,hang_height=0.1)

# reduce the size of the labels:
# dend <- assign_values_to_leaves_nodePar(dend, 0.5, "lab.cex")
dend <- set(dend, "labels_cex", 0.5)

# plot
par(mar = c(3,3,3,7))
plot(dend,
      main = "Clustered Iris data set
            (the labels give the true flower species)",
      horiz = TRUE, nodePar = list(cex = .007))
legend("topleft", legend = iris_species, fill = rainbow_hcl(3))

```

Task 3

```
# Task 3

# Stendent Number: 6228975

# student Name: Venkata Sandeep Kumar Karamsetty

# Student ID: vskk033@uowmail.edu.au


#load library

# use below if not installed

#install.packages('arules')

#install.packages('arulesViz')

library('arules')

library('arulesViz')


#setting current directory to working directory

setwd('./')


#reading .csv file

DataSet <- read.csv("./A1_success_data.csv")


#Data Exploration

class(DataSet)

typeof(DataSet)

head(DataSet)

summary(DataSet)


#convert the data frame to transactions so we can use it in apriori algorithm

transactions_of_Dataset <- as(DataSet, 'transactions')

summary(transactions_of_Dataset)

transactions_of_Dataset


# 1.) Generate frequent itemsets by applying various support thresholds and inspect these itemsets by displaying
their
```

```

# support, confidence, and lift values

#support threshold at 0.01
itemset1 = apriori(transactions_of_Dataset, parameter = list(support = 0.01, confidence = 0.3))
summary(itemset1@quality)
itemset1
#Inspect the itemsets by displaying the property support, confidence, and lift
inspect(head(sort(itemset1, by = "support"), 10))

#support threshold at 0.02
itemset2 = apriori(transactions_of_Dataset, parameter = list(support = 0.02,confidence = 0.6))
summary(itemset2@quality)
itemset2
inspect(head(sort(itemset2, by = "support"), 10))

#support threshold at 0.6
itemset3 = apriori(transactions_of_Dataset, parameter = list(support = 0.6,confidence = 0.8,minlen = 1,maxlen = 8))
summary(itemset3@quality)
itemset3
inspect(head(sort(itemset3, by = "support"), 10))

rhs = apriori(transactions_of_Dataset, parameter = list(support = 0.02,confidence = 0.6),
              appearance = list(rhs = c('Success=No', 'Success=Yes'), default = 'lhs'))

inspect(head(sort(rhs, by = 'lift'), 15))
#3.) - Visualize the rules generated in the last step by
#3-1) showing the relationship among support, confidence and lift

plot(rhs@quality)
#3-2) using the graph visualization based on the sorted lift value.

lift <- head(sort(rhs, by = "lift"), 5)
plot(lift, method = "graph", control = list(type = "items"))

```

