Midterm
Computer Science
Fall 2015
B565

Ganesh Nagarajan
gnagaraj@indiana.edu
Sunday, October 25, 9:00 p.m.

October 25, 2015

Solutions

All the work herein is solely mine.
Inorder to solve the given problem, following were the libraries and tools that contributed significantly.

Tool/Library	*Version*	*Acknowledgement to*
R	3.2.2	CRAN
R Studio	0.99.446	RStudio, Inc
MySql Database	5.6.26	Oracle, Inc
Talend Open Studio for Data Integration	6.0.0	Talend, Inc
RMySql	0.10.6	Jeroen Ooms et.al
cluster	2.03	Martin Maechler et.al
corrplot	0.73	Taiyun Wei
arules	1.2-1	Michael Hahsler et.al
arulesViz	1.0-4	Michael Hahsler et.al
e1071	1.6-7	David Meyer et.al
ggplot2	1.0.1	Hadley Wickham

Application: Solutions

Problem Statement

An interview was done with Filmflix.com to understand their business requirements for using data to their competitive advantage. There were key discussing about using data mining algorithms to understand their given data. A database schema of their operational systems were given for analysis.

0.1 Data Preparation

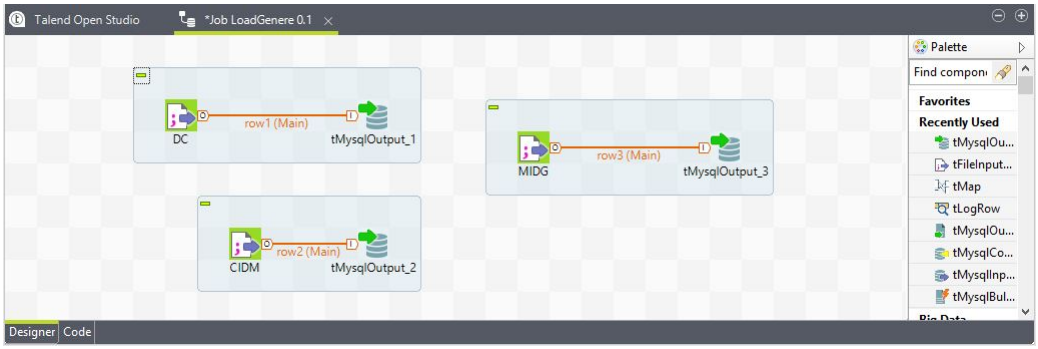
The Given data is in its operational or OLTP form, and hence cannot be used for any analysis. This data must be transformed to a suitable form for analysis. For this purpose, MySql database is used for data persistence and Talend ETL tool is used for data integration. Following were the data cleaning activities done to process the data.

1. Data Creation
   (a) The given data was reproduced in R and is converted to data frames.

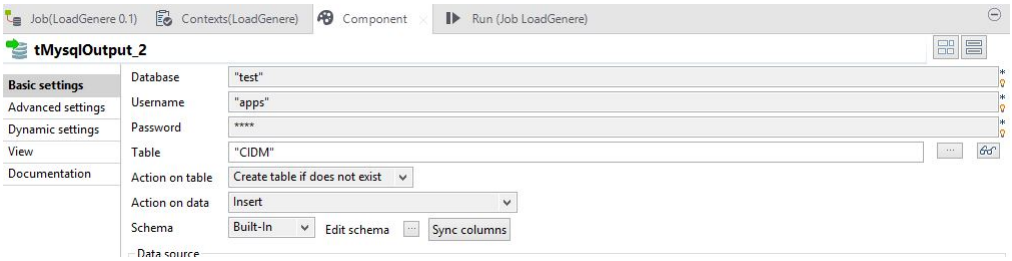
(b) Once the suitable column names are given, this data frame is exported to CSV.

```
#This program creates the CSV files for data ingestion.
#The data frame is converted to an CSV file, which is loaded
#to database using ETL
Genere<-c("Romance","Science Fiction","Horror","Comedy","Drama","Action",
          "Documentary","Classic")
Code<-c("r","s","h","c","d","a","o","l")
DC<-as.data.frame(cbind(Genere,Code))

mv1<-c("1","r","s",NA,NA)
mv2<-c("2","o","l","a",NA)
mv3<-c("3","c","d","h",NA)
mv4<-c("4","s","l","o","a")
mv5<-c("5","a","d","r",NA)
mv6<-c("6","d","h","c",NA)
mv7<-c("7","a","d","c","o")
mv8<-c("8","h","l","r",NA)
mv9<-c("9","s","d",NA,NA)
mv10<-c("10","c","r",NA,NA)
MIDG<-as.data.frame(rbind(mv1,mv2,mv3,mv4,mv5,mv6,mv7,mv8,mv9,mv10))
colnames(MIDG)<-c("Mv_ID","G1","G2","G3","G4")

C1<-c("CID1","CID2","CID3","CID4","CID5","CID6","CID7","CID8","CID9","CID10",
      "CID11","CID12","CID13","CID14")
C2<-c(1,4,7,2,4,3,1,5,10,2,1,3,8,5)
C3<-c(3,1,8,NA,8,9,2,4,1,4,10,5,1,2)
C4<-c(5,2,1,NA,10,10,3,9,2,3,8,1,7,8)
C5<-c(5,3,NA,NA,NA,1,NA,5,23,7,NA,2,NA,NA)
C6<-c(10,NA,NA,NA,NA,NA,NA,NA,NA,9,NA,NA,NA,NA)
c7<-c(8,NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,NA)
CIDM<-as.data.frame(cbind(C1,C2,C3,C4,C5,C6,c7))

write.csv(CIDM,"CIDM.csv")
write.csv(DC,"DC.csv")
write.csv(MIDG,"MIDG.csv")
```

(c) Once the data is available as CSVs, workflow jobs were created to copy this data to the MySQL database. Since Talend is an metadata based orchestration tool, we don't have to work with any data, rather only with the metadata. This metadata has been used to define the table schemas in the target database. Following is the screen shot of the workflows created.



A nifty feature of the tool is the option *create table if it doesn't exist*. Hence once the tool finds there is no table, it creates the table with the metadata that was designed in the tool's repository.



An export of this schema is attached with this homework if its needs to be reproduced. The schema name is test, no password, i.e blank password. In all connections, localhost is used to query the database.

2. Data Transformation

   (a) Since we have the data available for querying, now we have to transform the data for queries. Data Mining algorithms require specific format for data input. Eg, Clustering algorithms require features as columns.
   Hence using SQL, several temporary tables were created, several aggregations were performed to make the final data ie customers as rows and Genere / Movies seen as columns.
   A sample code is given here. However, the entire script is attached with this homework.

```
/* Select tables Created through ETL */
select * from DC;
select * from CIDM;
select * from MIDG;

/* Drop tables if they exist */
 drop table if exists cust_film;
 drop table if exists movie_genere;
 drop table if exists cluster_genere;
 drop table if exists cust_movie;

/* Convert tables from wide to long form */
/* Join Genere and Film Tables */
create table movie_genere as select ab.Mv_ID,ab.Genere from(
select MIDG.Mv_ID,DC.Genere from DC inner join MIDG on DC.Code=MIDG.G1
UNION all
select MIDG.Mv_ID,DC.Genere from DC inner join MIDG on DC.Code=MIDG.G2
UNION all
select MIDG.Mv_ID,DC.Genere from DC inner join MIDG on DC.Code=MIDG.G3
UNION all
select MIDG.Mv_ID,DC.Genere from DC inner join MIDG on DC.Code=MIDG.G4) as ab
order by MV_ID;

select * from movie_genere;

/* Convert tables from wide to long form */
/* Join Genere and Film Tables and Customer table */
create table cust_film as select cust_id,movie_id,Genere 'genere' from(
select cidm.C1 'cust_id',movie_genere.Mv_ID 'movie_id',movie_genere.Genere from
cidm inner join movie_genere where cidm.C2=movie_genere.Mv_ID
UNION all
select cidm.C1 'cust_id',movie_genere.Mv_ID 'movie_id',movie_genere.Genere from
cidm inner join movie_genere where cidm.C3=movie_genere.Mv_ID
```

```
UNION all
select cidm.C1 'cust_id',movie_genere.Mv_ID 'movie_id',movie_genere.Genere from
cidm inner join movie_genere where cidm.C4=movie_genere.Mv_ID
UNION all
select cidm.C1 'cust_id',movie_genere.Mv_ID 'movie_id',movie_genere.Genere from
cidm inner join movie_genere where cidm.C5=movie_genere.Mv_ID
UNION all
select cidm.C1 'cust_id',movie_genere.Mv_ID 'movie_id',movie_genere.Genere from
cidm inner join movie_genere where cidm.C6=movie_genere.Mv_ID
UNION all
select cidm.C1 'cust_id',movie_genere.Mv_ID 'movie_id',movie_genere.Genere from
 cidm inner join movie_genere where cidm.C7=movie_genere.Mv_ID)as temp
order by cust_id,movie_id;

select * from cust_film;

/* Convert tables from long to wide form */
/* has customer ID as rows and Genere as Columns*/
create table cluster_genere as select   cust_id,
sum(case when genere='Romance' then 1 else 0 end) 'Romance',
```

A screenshot of a sample table is given below.
This table aggregates the views in genere by customer. Similarly other table, aggregation the views of film by customer is also created with other supporting tables.

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

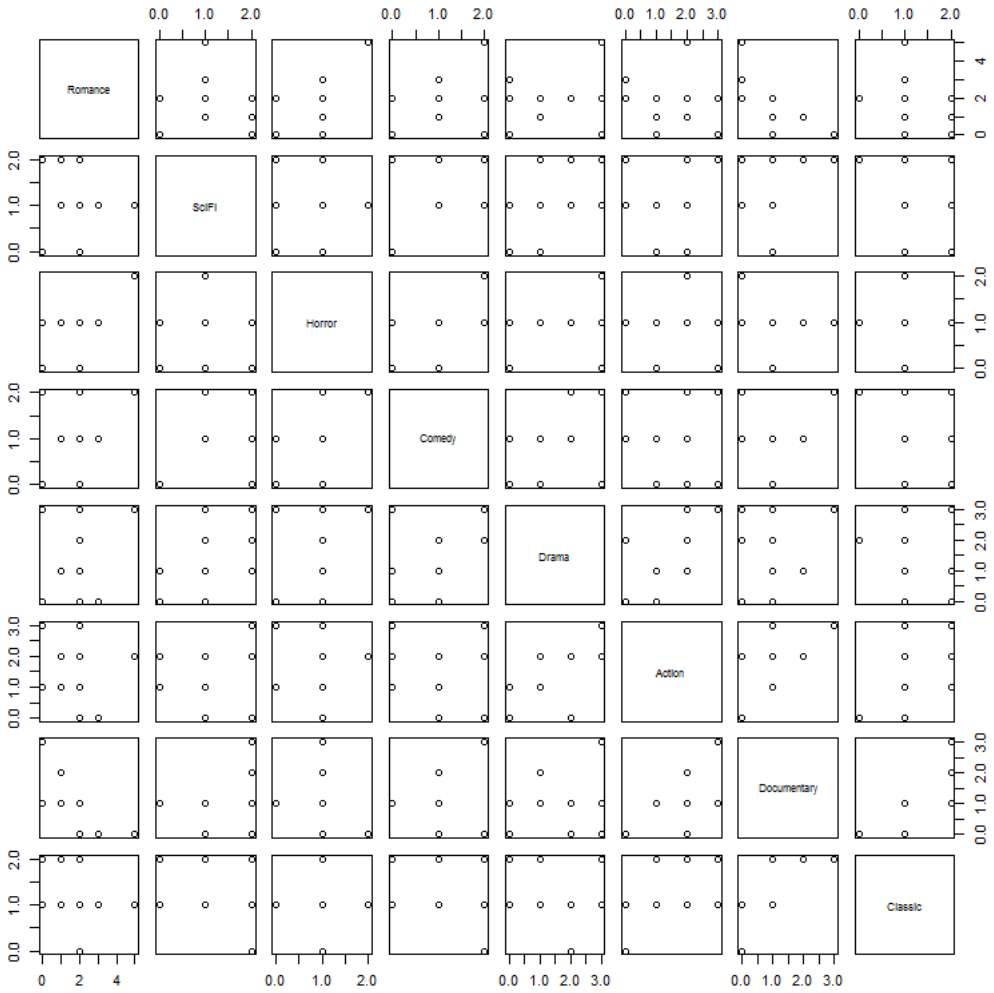
cust_id	Romance	SciFi	Horror	Comedy	Drama	Action	Documentary	Classic
CID1	5	1	2	2	3	2	0	1
CID10	0	2	1	2	3	3	3	2
CID11	3	1	1	1	0	0	0	1
CID12	2	1	1	1	2	2	1	1
CID13	2	1	1	1	1	1	1	1
CID14	2	0	1	0	1	2	1	2
CID2	1	2	1	1	1	2	2	2
CID3	2	1	1	1	1	1	1	1
CID4	0	0	0	0	0	1	1	1
CID5	2	1	1	1	0	1	1	2
CID6	2	2	1	2	2	0	0	0
CID7	1	1	1	1	1	1	1	1
CID8	2	2	0	0	3	3	1	1
CID9	2	1	0	1	0	1	1	1

Exploratory Data Analysis

Since we have the data now, in a format that can be used for analysis, visualizing data will help to understand more about the data. This helps the feature engineering process of data mining if its required. RSQL

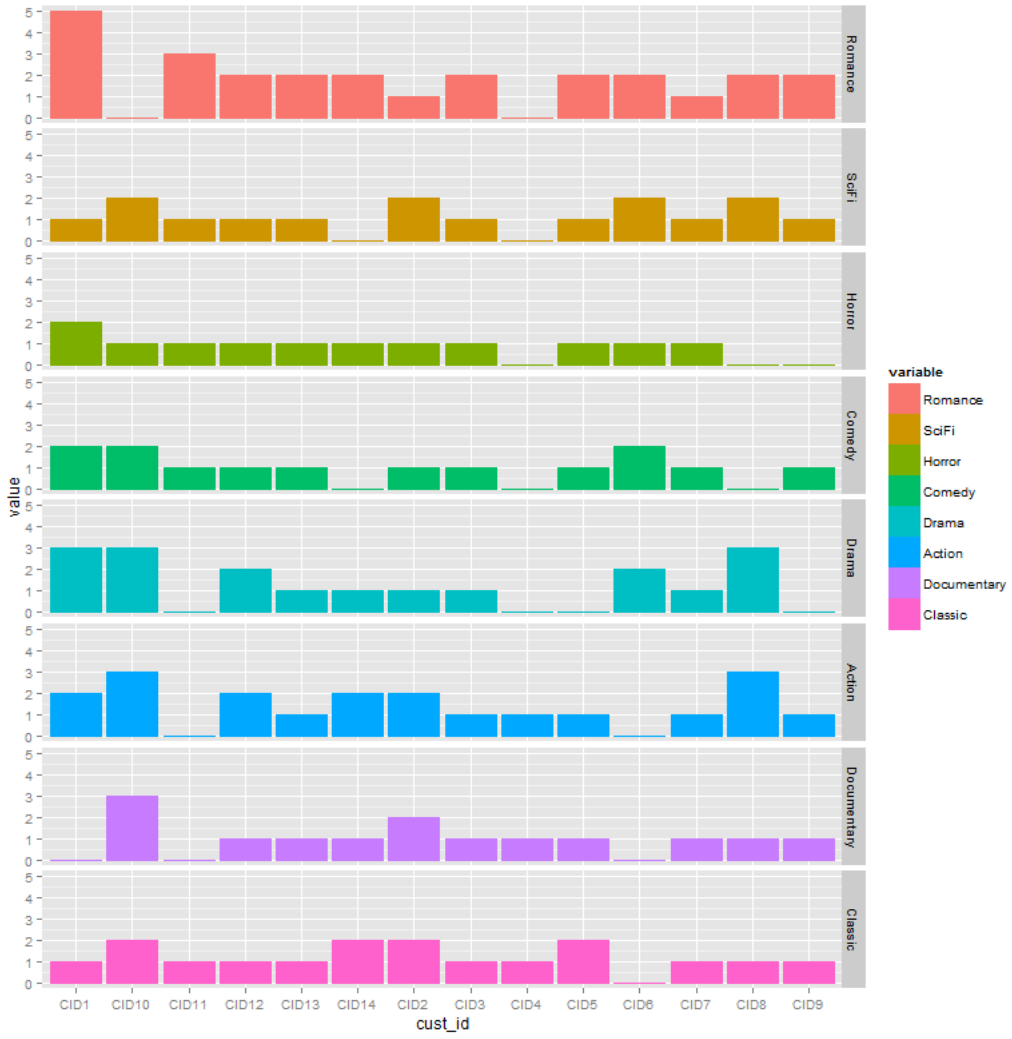
package is used to connect R to the MySQL database and the visualizations are plotted through ggplot2.

1. Plot of the entire genre vs customers are as follows.

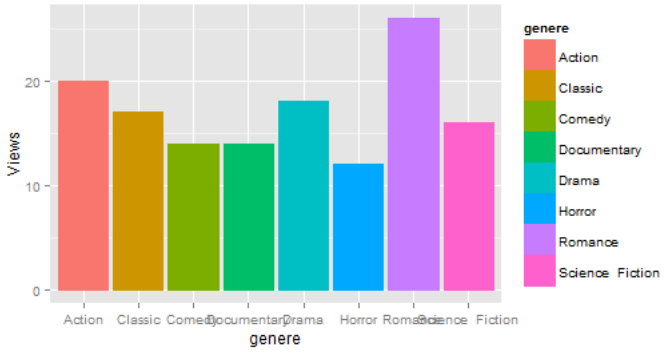


   It can be visually seen that there are some repeating patterns in the dataset. Hence this can be studied further for dimensional reduction perspective.

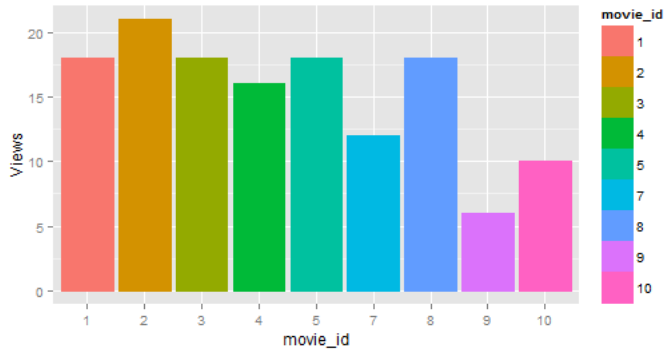
2. Since this is a small dataset, we can visualize the entire dataset. Following graphic segregates the views of all customers by genere. Eg, Customer 1 has seen the highest movies in Romance Genre!



3. When taking about promotions, it is important to understand what movies and generes are popular. Following visuals capture the popular generes and films by number of views.



It seems the most watched genre is Romance!

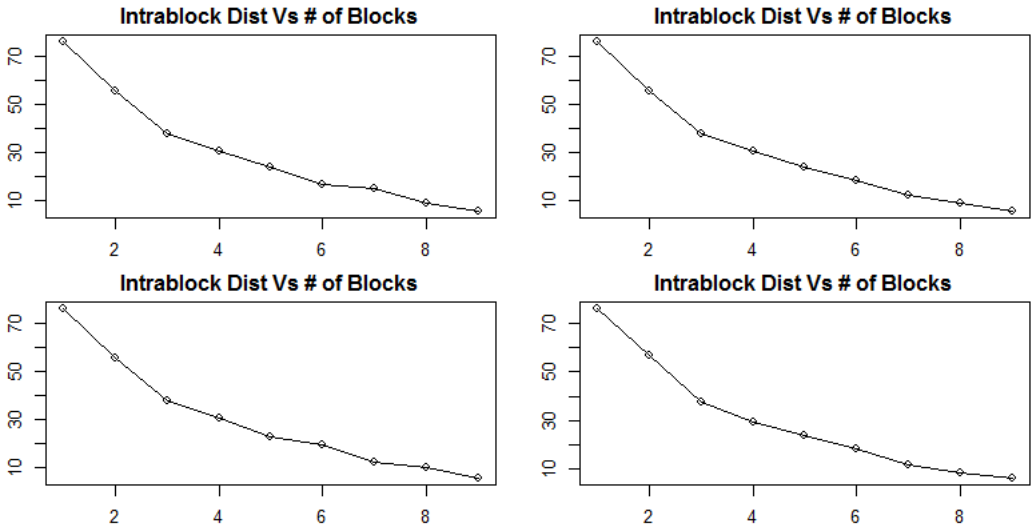


And movie 2 is the most watched movie of all times!
These popular items will help us in promotion of the films.

0.2 K-Means for Segmentation

Businesses would like to do segmentation of its customers by purchase power, buying patterns etc, FilmFlix, being an internet company, requires to segment the group both by genre aswell as the common films seen.

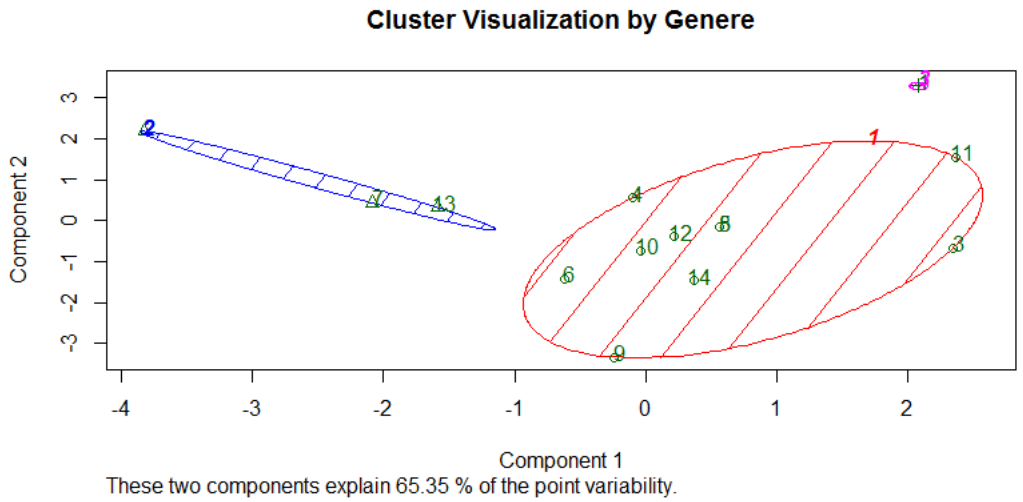
1. Determination of K for k-Means
   Optimal k ensures the interpretability of that cluster. Inorder to identify k, a function was written in R, which calculates the sum of inter block distances between the blocs. A balance has to be attained between the number of k and the minimal inter block distance. As the number of block approaches number of elements, it is indeed that the intra block distance reduces, however now each block is an element, hence would not explain anything. Hence a key is to choose a k, which is in the knee point of the curve.
   A four iteration trial was made to training dataset, and it is seen that in all 4 curves, three remains to be the most optimal point. Hence 3 is chosen as k.



2. Implementation for Genre
   K-Means was run to this setting, with input as a dataframe consisting customers as rows and Genre as columns.
   Following is the code,

```
#Program : 3
#This program does a K-Means for the customer vs film & Genere.
#This retrieves the data frames from data bases and does k-Means

#--Libraries--
library(RMySQL)
library(corrplot)

##--Diagnostoc functions : To predict the best k--------
plotMinIntra<-function(dataframe,startIndex,endIndex,clusterNo,iterationNo){
  par(mfrow=c(floor(iterationNo/2),floor(iterationNo/2)),mar=c(2,2,2,2))
  for(j in 1:iterationNo){
    nos<-1:clusterNo
    sum_v=c()
    for(i in 1:clusterNo){
      sum_v<-c(sum_v,sum(kmeans(dataframe[startIndex:endIndex],i)$withinss))
    }
    plot(nos,sum_v,main="Intrablock Dist Vs # of Blocks",xlab = "# cluster",
         ylab = "Sum of Intrablock Distance",type="o")
  }
}
#----------------------

sqldb=dbConnect(MySQL(),user='apps',password='',dbname='test',
                host='localhost')
rs = dbSendQuery(sqldb, "select * from cluster_genere")
genere_df = fetch(rs, n=-1)
rs = dbSendQuery(sqldb, "select * from cluster_movie")
movie_df = fetch(rs, n=-1)

#Determine K
plotMinIntra(genere_df,2,9,9,4)

#Call K Means for genere with cluster size 3
clust_genere<-kmeans(genere_df[2:9],3,iter.max = 10)
clusplot(genere_df[2:9],clust_genere$cluster,color=TRUE,shade=TRUE,
         labels=2,lines=0,main="Cluster Visualization by Genere")
clust_genere$cluster

#plot Corrplots
#No one has seen movie #7
movie_df<-movie_df[,-7]
corrplot(cor(movie_df[-1]),method=c("number"))
corrplot(cor(movie_df[-1]),method=c("square"))

#column 8 and 9 have same corelation. Remove Col9
tocl_movie_df<-movie_df[,-8]
#Remove Labels
tocl_movie_df<-tocl_movie_df[,-1]
#Retry finding best clusters
plotMinIntra(tocl_movie_df,1,8,5,4)
clust_movie<-kmeans(tocl_movie_df,3)
clusplot(tocl_movie_df,clust_movie$cluster,color=TRUE,shade=TRUE,
```

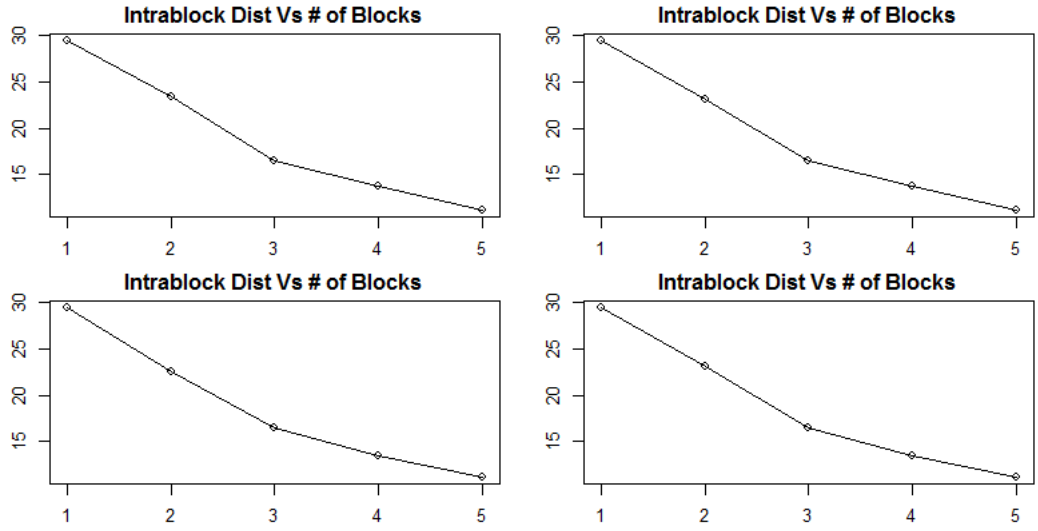
labels=2,**lines**=0,main="Cluster Visualization by Film")



Cluster package was used to visualize the cluster. This was run several times and the most consistent cluster is shown above.
When this cluster was analyzed,
Cluster 1 : Represents Drama, Action, Documentary and Classics. Most of the customers in this segment see these above said genre.
Cluster 3 : Represents customers who have seen maximum of Romance films.
Cluster 2 : Represents a wide spread of Genere. Most of these customers have seen atleast a film in every Genre.

3. Implementation of K-Means to customers who have seen similar films.

   (a) Determination of K
   As described above, a inter block distance vs clusters was plotted for this dataset. This hints that there could be three clusters in this dataset also.

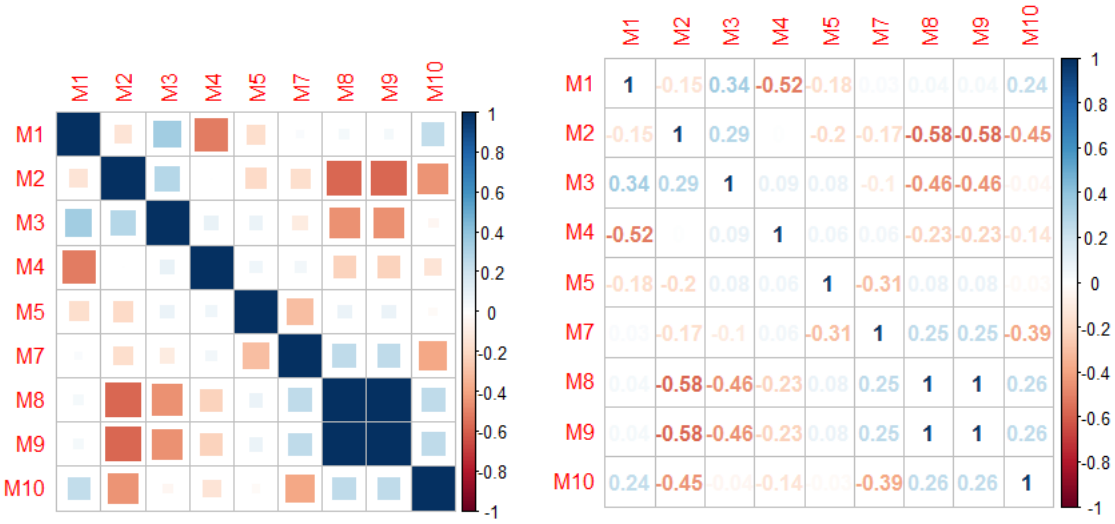


   (b) The structure of this dataset is, customers in rows and Movies in columns. The cross numbers represent the views of the movie.

(c) In this dataset, there were few observations:

i. No one has seen movie 6. Hence can be removed from the dataset.

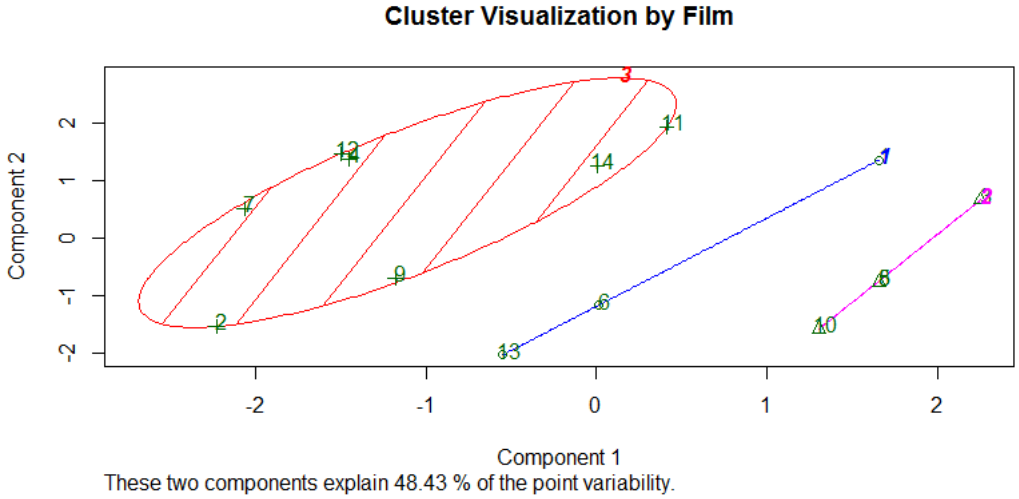
ii. A correlation matrix of the dataset brings that correlation of movie 8 and 9 are 1, ie everyone who has seen movie 8 have seen movie 9!.
This is an interesting data, An offer of 20 percent can be given to anyone who are seeing both the movies!



Also from computational perspective, we can either of the column, because these two represent the same features.

iii. There were few erroneous data, E.g there is no movie with id 23. Hence this data was removed.

(d) K-Means for Movie Dataset
Now since the data is already prepared, and k=3 is determined as optimal number of clusters, K-Means was done. Following is the visualization of that cluster.



This figure is self explanatory,
Cluster 1 : 13,6,1 customers have similar patterns.
Cluster 2 : 10,8,3 customers have similar patterns.
Cluster 3 : Rest all customers have similar patterns.

Association Rules

1. Association rules in current application context is used to find what genre/movies are bring watched together.
   The library used for this is Apriori algorithm and AprioriViz library is used for visualization of these rules.

   (a) Apriori by Genre

      i. : Data Preparation
      Since Apriori algorithm does not worry about features as such, however is interested in transactions, following is a sample datastructure that was used for apriori.

```
genere_df
    cust_id movie_id           genere
1      CID1        1          Romance
2      CID1        1 Science  Fiction
3      CID1        3           Horror
4      CID1        3            Drama
5      CID1        3           Comedy
6      CID1        5           Action
7      CID1        5          Romance
.....
17    CID10        2          Classic
18    CID10        2           Action
```

      i.e, every customer is considered as an transaction. This data is queried from the database and is written to a CSV files.
      These CSV files are then manually opened and the quotes are deleted. This processed file is then directly imported as transactions deleting the duplicate entries.

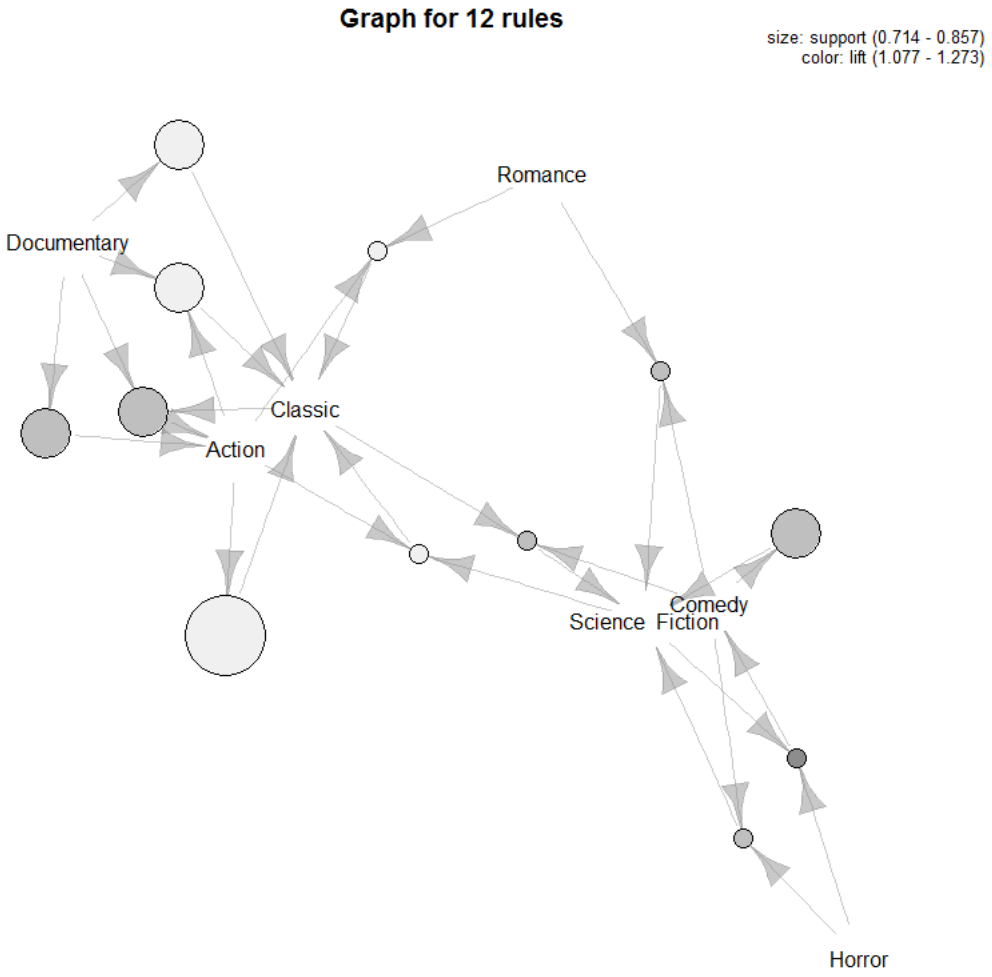
      ii. The complete code is as follows,
      The parameters used are, supp=0.7,conf=1

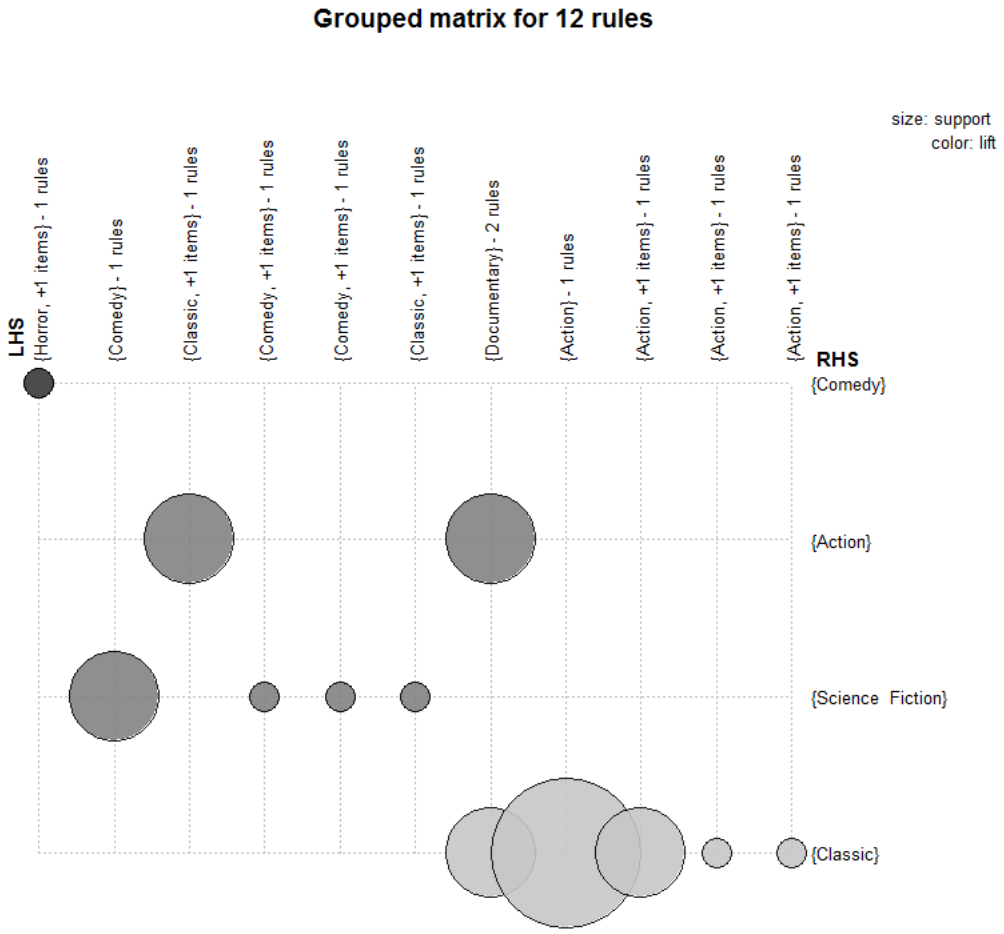
```
#Program4: Association Rules
library(arules)
library(arulesViz)
library(RMySQL)

#Retrieve data from database
sqldb=dbConnect(MySQL(),user='apps',password='',dbname='test',host='localhost')
rs = dbSendQuery(sqldb, "select * from cust_film")
genere_df = fetch(rs, n=-1)
rs = dbSendQuery(sqldb, "select * from cust_movie")
movie_df = fetch(rs, n=-1)

#Remove the movie id and convert the rest to factors.
genere_df<-genere_df[-2]
genere_df[,1]<-as.factor(genere_df[,1])
genere_df[,2]<-as.factor(genere_df[,2])

#Convert the data frame to transcation
write.table(genere_df,"transactions_genere.csv",row.names = FALSE,
            col.names = FALSE,sep = ",")
#Manually replaced quotes
txn_genere<-read.transactions(file="transactions_genere.csv",
            format = "single",sep=",",rm.duplicates=TRUE,cols=c(1,2))
```

```
inspect(txn_genere)
ru_genere<-apriori(txn_genere,parameter = list(supp=0.7,conf=1,target="rules"))
inspect(ru_genere)
plot(ru_genere,method="graph",control=list(type="items"))
plot(ru_genere,method="grouped")

#For association by film:
movie_df[,1]<-as.factor(movie_df[,1])
movie_df[,2]<-as.factor(movie_df[,2])
write.table(movie_df,"transactions_film.csv",row.names = FALSE,
            col.names = FALSE,sep = ",")
#Manually replaced quotes
txn_film<-read.transactions(file="transactions_film.csv",format = "single",
                            sep=",",rm.duplicates=TRUE,cols=c(1,2))
inspect(txn_film)
ru_film<-apriori(txn_film,parameter = list(supp=0.2,conf=0.5,target="rules"))
inspect(ru_film)
plot(ru_film,method="graph",control=list(type="items"))
plot(ru_film,method="grouped")
```

```
inspect(ru_genere)
   lhs                          rhs                support   confidence lift
1  {Documentary}             => {Action}           0.7857143 1          1.166667
2  {Documentary}             => {Classic}          0.7857143 1          1.076923
3  {Comedy}                  => {Science  Fiction} 0.7857143 1          1.166667
4  {Action}                  => {Classic}          0.8571429 1          1.076923
5  {Action,Documentary}      => {Classic}          0.7857143 1          1.076923
6  {Classic,Documentary}     => {Action}           0.7857143 1          1.166667
7  {Comedy,Horror}           => {Science  Fiction} 0.7142857 1          1.166667
8  {Horror,Science  Fiction} => {Comedy}           0.7142857 1          1.272727
9  {Comedy,Romance}          => {Science  Fiction} 0.7142857 1          1.166667
10 {Classic,Comedy}          => {Science  Fiction} 0.7142857 1          1.166667
11 {Action,Romance}          => {Classic}          0.7142857 1          1.076923
12 {Action,Science  Fiction} => {Classic}          0.7142857 1          1.076923
```

iii. Visualization and Interpretation:
The below graphics are beautiful to interpret the association rules in graphical form, especially the graph chart.
It can be seen that people who watch Action,Documentary, always see Classic.
Also people who see Classic,Documentary always see Action.
Hence there can be an discount of 25 percent when all movies of one in three genre are bought together!


Graph for 12 rules
size: support (0.714 - 0.857)
color: lift (1.077 - 1.273)
Romance
Documentary
Classic
Action
Comedy
Science Fiction
Horror



2. Visualizing for Films:

   (a) Similar to the data processed for above method, the transnational nature of the data presented for film assocation is as follows.

```
movie_df
   cust_id movie_id
1     CID1        3
2     CID1        5
3     CID1        5
4     CID1       10
....
7    CID10        3
8    CID10        7
9    CID10        9
```

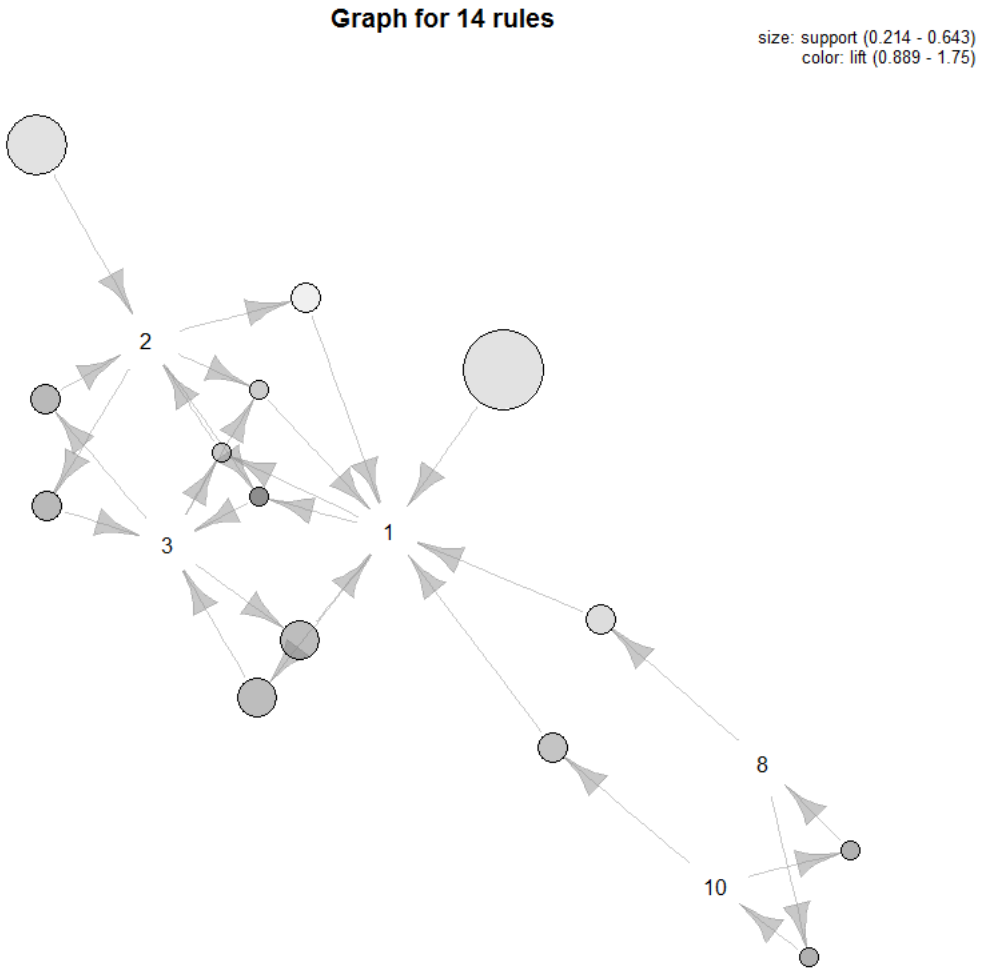
   Same as above method, Customer id is the transaction number here.

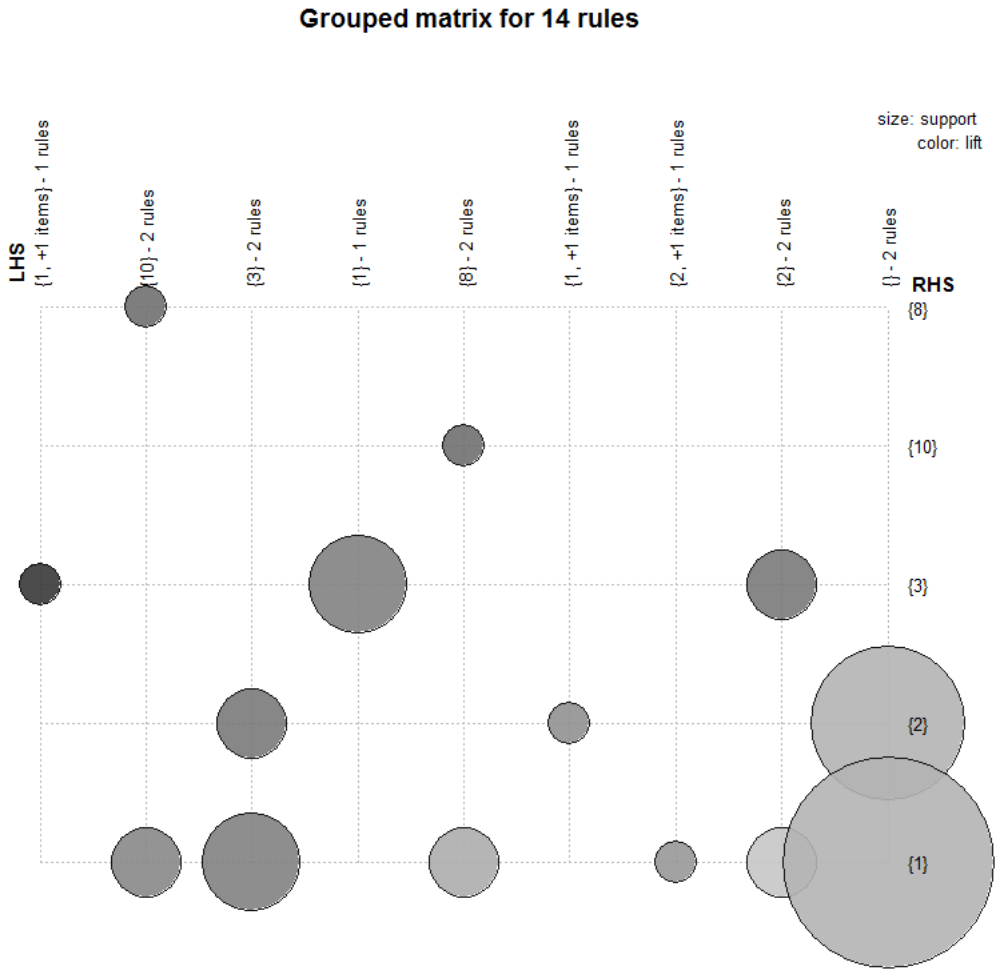
   (b) Implementation:
   Apriori algorithm was run with supp=0.2,conf=0.5 parameters and following is the output.

```
lhs      rhs  support   confidence lift
1  {}    => {2}  0.5000000 0.5000000  1.0000000
2  {}    => {1}  0.6428571 0.6428571  1.0000000
3  {10}  => {8}  0.2142857 0.6000000  1.4000000
4  {8}   => {10} 0.2142857 0.5000000  1.4000000
5  {10}  => {1}  0.2857143 0.8000000  1.2444444
6  {8}   => {1}  0.2857143 0.6666667  1.0370370
7  {2}   => {3}  0.2857143 0.5714286  1.3333333
8  {3}   => {2}  0.2857143 0.6666667  1.3333333
9  {2}   => {1}  0.2857143 0.5714286  0.8888889
10 {3}   => {1}  0.3571429 0.8333333  1.2962963
11 {1}   => {3}  0.3571429 0.5555556  1.2962963
12 {2,3} => {1}  0.2142857 0.7500000  1.1666667
13 {1,2} => {3}  0.2142857 0.7500000  1.7500000
14 {1,3} => {2}  0.2142857 0.6000000  1.2000000
```

The algorithm suffers from the fact that there is not sufficient data to build and report inferences with confidence. However with the current data following are the rules that can be made.
People who have seen movie 1,2 have seen 3, people who have seen 2,3 have seen 1 and people who have seen 1,3 have seen 2. Thus marketing plan of all three movies bundled together with 25 percent discount is a great initiative.
Also people who hav seen 8 have also seen 10 and vice versa. Hence a coupon for 10 percent discount for either of these movies would also be a great marketing initiative.

(c) as seen above, these rules can also be visualized as follows,


Graph for 14 rules
size: support (0.214 - 0.643)
color: lift (0.889 - 1.75)
2
3
1
8
10



0.3 Knn Neighbors

K-Nearest neibhour is used for selection problem. It selects the most probable item which is similar to the test dataset.
Here the input dataset/train dataset is the dataframe of customers as columns and films as rows. package class is used for KNN neibhour algorithm.

1. Program:
   The code of KNN is fairly simple. The entire dataframe other than the labels form training dataset. The data to be predicted for is the nearest neighbor of films 2,5,8,9 which is made as training set. The k nearest elements were chosen to be 3.

```
#Program 4: K-Nearest Neibhour
library(class)
library(e1071)
library(RMySQL)
library(cluster)
```

```
#Retrieve Dataframes from Database
sqldb=dbConnect(MySQL(),user='apps',password='',dbname='test',host='localhost')
rs = dbSendQuery(sqldb, "select * from cluster_genere")
genere_df = fetch(rs, n=-1)
rs = dbSendQuery(sqldb, "select * from cluster_movie")
movie_df = fetch(rs, n=-1)

#Create Test Dataset
movie_test<-as.data.frame(as.matrix(t(c(0,0,1,0,0,1,0,0,1,1,0))))
movie_test[1]=as.factor(c("NA"))

#Refactor customer name as factor
movie_df[,1]<-as.factor(movie_df[,1])
colnames(movie_test)<-colnames(movie_df)

#Run K-nn algorithm
ru_knn<-knn(movie_df[-1],movie_test[,-1],movie_df[,1],k=3)
#Print Output
ru_knn
```

This algorithm was run multiple times. There were different outputs every time.
The most consistent result seems to be 14 out of few tests.

```
[1] CID5
Levels: CID1 CID10 CID11 CID12 CID13 CID14 CID2 CID3 CID4 CID5 CID6 CID7 CID8 CID9
> ru_knn<-knn(movie_df[-1],movie_test[,-1],movie_df[,1],k=3)
> ru_knn
[1] CID14
Levels: CID1 CID10 CID11 CID12 CID13 CID14 CID2 CID3 CID4 CID5 CID6 CID7 CID8 CID9
> ru_knn<-knn(movie_df[-1],movie_test[,-1],movie_df[,1],k=3)
> ru_knn
[1] CID14
Levels: CID1 CID10 CID11 CID12 CID13 CID14 CID2 CID3 CID4 CID5 CID6 CID7 CID8 CID9
```

Hence the most nearest similar customer who has seen 2,5,8,7 is customer 14.

0.4 Naive Bayes

Following is the code written for naive bayes implementation.

```
#bayes with test and train dataset
library(RMySQL)

#Retrieve Dataset
sqldb=dbConnect(MySQL(),user='apps',password='',dbname='test',host='localhost')
rs = dbSendQuery(sqldb, "select movie_id,genere_from_cust_film;")
genere_df = fetch(rs, n=-1)

#Make Factors
genere_df=as.data.frame(lapply(genere_df,as.factor))

#Train a Bayesian Classifier
fit<-naiveBayes(movie_id~.,data=genere_df,laplace = 3)
```

```
#Predict the test dataset
predict(fit ,as.factor(c("Action","Drama")))
```

The output is given was movie 2.

0.5 Agglomerative Hierarchial Clustering

Inorder to find how the genre are related to teach other, we take up the transpose of the customer vs genere dataset.
The distance is calculated between these items in this dataset and is passed as parameter to the hclust algorithm.
The Datase Structure is as follows,

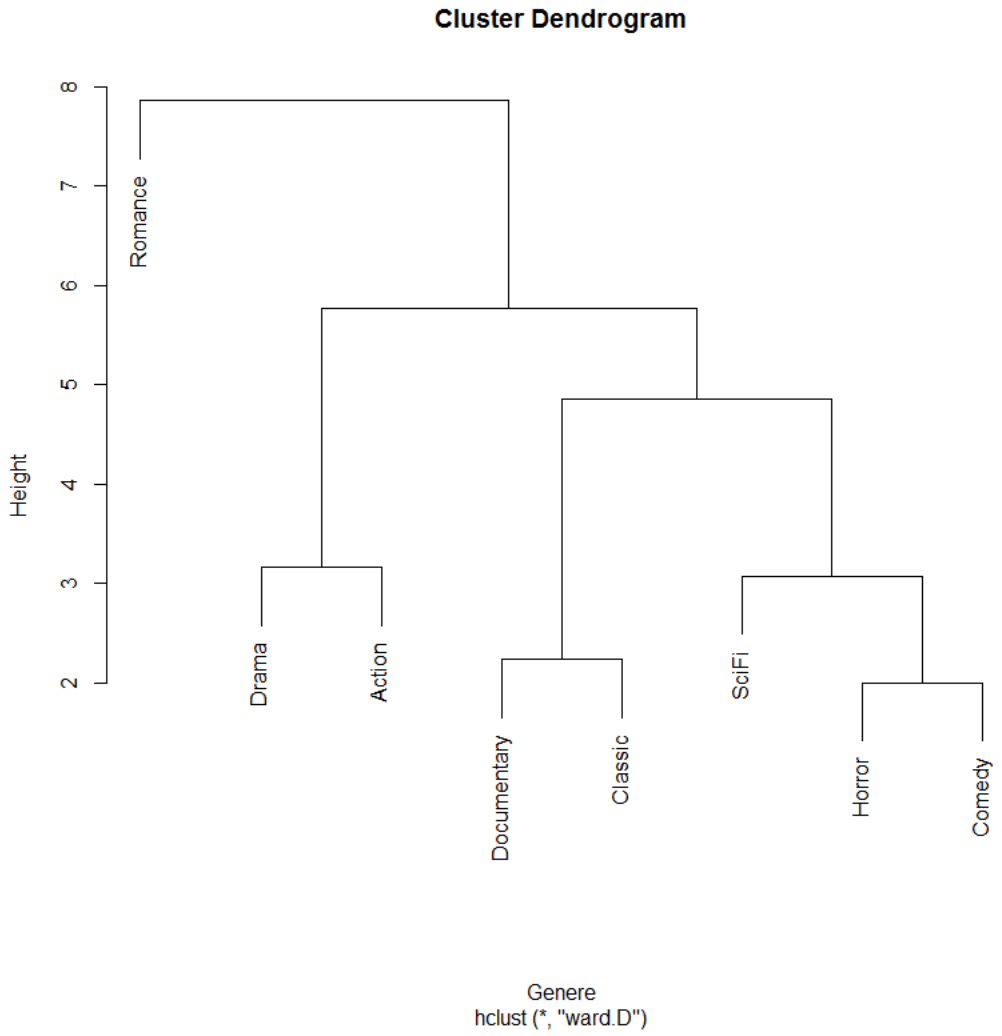
```
           V1 V2 V3 V4 V5 V6 V7 V8 V9 V10 V11 V12 V13 V14
Romance     5  0  3  2  2  2  1  2  0   2   2   1   2   2
SciFi       1  2  1  1  1  0  2  1  0   1   2   1   2   1
Horror      2  1  1  1  1  1  1  1  0   1   1   1   0   0
Comedy      2  2  1  1  1  0  1  1  0   1   2   1   0   1
Drama       3  3  0  2  1  1  1  1  0   0   2   1   3   0
Action      2  3  0  2  1  2  2  1  1   1   0   1   3   1
Documentary 0  3  0  1  1  1  2  1  1   1   0   1   1   1
Classic     1  2  1  1  1  2  2  1  1   2   0   1   1   1

>d
             Romance    SciFi   Horror   Comedy    Drama   Action Documentary
SciFi       5.830952
Horror      5.291503 3.162278
Comedy      5.477226 2.449490 2.000000
Drama       5.830952 3.464102 4.242641 4.000000
Action      6.164414 3.741657 4.472136 4.690416 3.162278
Documentary 7.483315 3.162278 3.741657 3.741657 4.690416 3.162278
Classic     5.916080 3.316625 3.000000 3.605551 4.795832 3.000000    2.236068
```

This data is then plotted with method average and ward.D method.
Code:

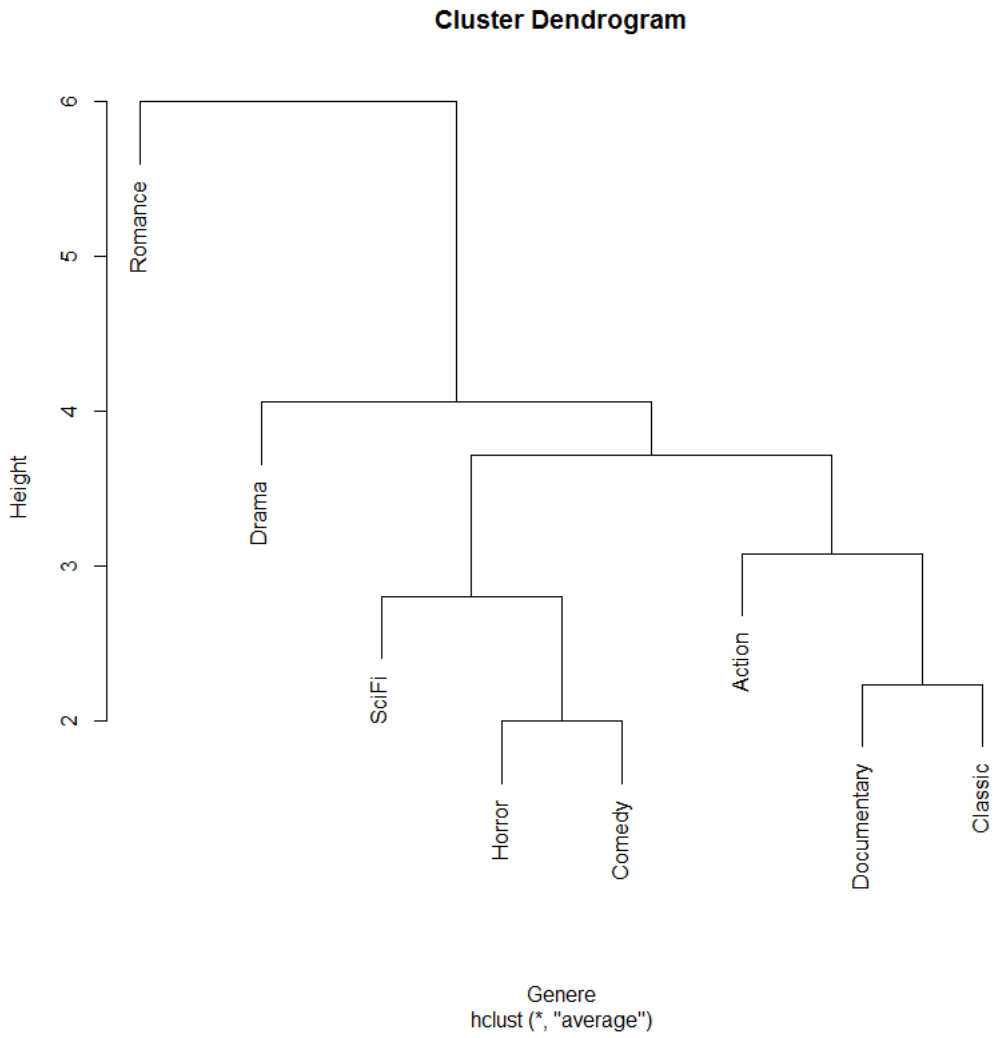
```
#
library(RMySQL)

#Retieve data from the database
sqldb=dbConnect(MySQL(),user='apps',password='',dbname='test',host='localhost')
rs = dbSendQuery(sqldb, "select * from cluster_genere")
genere_df = fetch(rs, n=-1)

#Transpose the dataframe maiking genere as rows
data<-as.data.frame(t(genere_df))
data<-data[-1,]

#Find the distance matrix
d <- dist(data, method = "euclidean")

#Fit this to the agglomerative clustering algorithm
#Both Ward.D mehod and un-weighed average method was used.
```

```
fit <- hclust(d, method="ward.D")
plot(fit ,xlab="Genere")
fit <- hclust(d, method="average")
plot(fit ,xlab="Genere")
```

1. Ward.D :



2. Un Weighted Average:



Both clusters represent similar things.
With respect to average cluster method,
People usually see Documentary and Classic together.
People who see Documentary and classic also see Action movies.
Hence rest of the plot can be interpreted as such.

Equivocation of k-l means algorithm

Algorithm

1: **ALGORITHM** `k,l-means`
2: **INPUT** (data Δ, distance $d : \Delta^2 \to \mathbb{R}_{\geq 0}$, centoid number k, Closest Matches ℓ, threshold τ)
3: **OUTPUT** (Set of centoids $c_1, c_2, \ldots, c_k$)
4: Assume centroid is structure $c = (v \in DOM(\Delta), B \subseteq \Delta)$
5: $c.v$ is the centroid value and $c.B$ is the set of nearest points.
6: τ is a percentage change from previous centroids
7: For example, $\{c_1, c_2, \ldots, c_k\}$ is previous and $\{d_1, d_2, \ldots, d_k\}$ is current
8: Total difference is $\Sigma_i \Sigma_j d(c_i, d_j)$

9: $Dom(\Delta)$ denotes domain of object.
10: $i = 0$ ▷ Initialize iterate where superscript is iteration
11: **for** $j = 1, k$ **do** ▷ Initialize Centroids
12: $c_j^i.v \leftarrow random(Dom(\Delta))$
13: $c_j^i.B \leftarrow \emptyset$
14: **end for**
15: $f_i = \Sigma_{j=1}^{k} \Sigma_{\ell=1}^{k} d(c_j^i.v, random(Dom(\Delta))$ ▷ Bootstrap difference between past centroids and current
16: **repeat**
17: $i \leftarrow i + 1$
18: **for** $\delta \in \Delta$ **do**
19: **for** $k = 0; k < l; k++$ **do**
20: $c_j^i.B \leftarrow c.B \cup \{\delta\}$, where $\min_{c_j^i}\{d(\delta, c_j^i.v)\}$
21: ▷ Associate a data point δ with the nearest l $c_j^i.v$
22: Replace $\min_{c_j^i}\{d(\delta, c_j^i.v)\}$ with $\max_{c_j^i}\{d(\delta, c_j^i.v)\} + 1$
23: ▷ This step is done so that the same minimum distance is not considered during other iteration
24: **end for**
25: **end for**
26: **for** $j = 1, k$ **do**
27: $c_j^i.v \leftarrow ave(c_j^i.B)$ ▷ Update centroid to be *best* representative of nearest data
28: $c_j^i.B \leftarrow \emptyset$ ▷ *ave* is easiest representative
29: **end for**
30: **until** $(|f_i - f_{i-1}| < \tau(f_{i-1}))$
31: **return** $(c_1^i, c_2^i, \ldots, c_3^i)$

Illustration

1. Suppose [1 2 3 4 5] is a row in distance matrix corresponding to the datum n.
   From the distance matrix, it can be inferred that there are 5 centroids. Consider l=2,

```
Iteration 1: [1 2 3 4 5] - Datum n is associated with centroid 1
Replace the distance : [6 2 3 4 5]
Iteration 2: [6 2 3 4 5] - Datum n is associated with centroid 2
Replace the distance : [6 7 3 4 5]
Iterations Reached l.
Datum point n is now associated with centroid 1 and 2
# Average and update the new representative.
```

2. Java Implementation

   (a)
```
for(int j=0;j<l;j++){
int noP=kCentroids_l.get(tuple_l.indexOf(Collections.min(tuple_l))).noPoints;
kCentroids_l.get(tuple_l.indexOf(Collections.min(tuple_l))).intIndex[noP]=i;
kCentroids_l.get(tuple_l.indexOf(Collections.min(tuple_l))).currPoints[noP]=(int)dataSet.get(i
kCentroids_l.get(tuple_l.indexOf(Collections.min(tuple_l))).noPoints++;
tuple_l.set(tuple_l.indexOf(Collections.min(tuple_l)),Collections.max(tuple_l)+1);
}
```

   (b) DataStructure Framework: EJML, Efficient Java Matrix Library is an open source library for extending matrix data structures to java. This supports loading of CSV files directly as matrix, perform advanced operations like matrix, SVD, and calculate distances. However for the purpose of this assignment, they are just used as data structures and this assignment uses no advanced operation required of these data structures.

(c) Data munging for EJML : EJML reads data only as matrix and requires space separation instead of comma separated values. Hence these steps are done manually to replace comma with a space and replace first line with number of rows, columns and the data type. Following is a sneak peak of the file used for this assignment.

```
683 11 real
1000025 5 1 1 1 2 1 3 1 1 2
1002945 5 4 4 5 7 10 3 2 1 2
1015425 3 1 1 1 2 2 3 1 1 2
1016277 6 8 8 1 3 4 3 7 1 2
1017023 4 1 1 3 2 1 3 1 1 2
1017122 8 10 10 8 7 10 9 7 1 4
```

(d) Program Invocation: The program takes 7 parameters, input file, distance function, number of clusters, number of iterations, tolerance, featurset and l. Feature set is an comma separated string which contains index of strings that are to be considered for the clustering. This is useful when we can select only few features as predictor values rather than the entire attribute set. Following is the signature used to execute the data analysis. Currently only Eucledian distance is supported. This rather an provision for further improvement.

```
"d:\\bcdata.csv" eucledian 2 5 0.1 1,2,3,4,5,6,7,8,9 2
```

(e) Interpreting the Results: Following is a snapshot of the output.

```
------------------------Iteration 2 ------------------------
Cluster Id : 0
Cluster Centers : Type = dense real , numRows = 1 , numCols = 11
1324572.000   7.167   6.744   6.709   5.684   5.449   7.850   6.056   6.017   2.560

Previous Centers : Type = dense real , numRows = 1 , numCols = 11
1324572.000   7.075   6.427   6.416   5.459   5.290   7.427   5.859   5.792   2.514

Cluster Points[1002945, 1016277, 1017122]
Cluster Internal Indexes[1, 3, 5, -1]

Cluster Id : 1
Cluster Centers : Type = dense real , numRows = 1 , numCols = 11
1347749.000   3.022   1.278   1.394   1.343   2.080   1.301   2.085   1.229   1.105

Previous Centers : Type = dense real , numRows = 1 , numCols = 11
1347749.000   2.874   1.199   1.308   1.264   2.009   1.231   2.007   1.129   1.061

Cluster Points[1000025, 1015425, 1017023, 841769]
Cluster Internal Indexes[0, 2, 4, 6, 7, -1]

------------------------Has Converged : Tolerance------------------------
```

The above is a snapshot of the output. There is an iteration number associated with these centroids. There are current centers and previous centers for every centroid. There is also references to data points associated to that centroid and reference to index of the data points. Also it gives details about if the convergence was by tolerance or iteration.

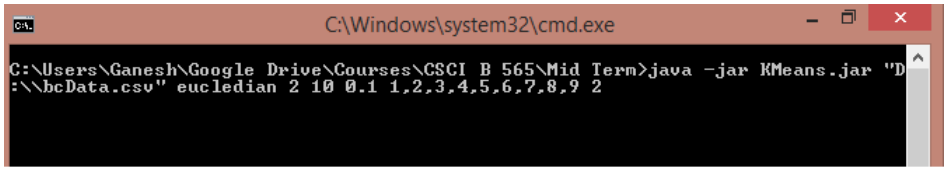
(f) Application Design: The application mainly consists of two classes, Centroid and kMeans. Centroid class is mainly used as a integrated data structure encompassing multiple array vectors and matrices for holding statuses, values and to holding centroids. the driver class consists of modular functions for processing these centroid objects. These centroids are passes as array of objects

using Java Collection framework. It has modular functions for reassigning centroid, calculating distances etc.

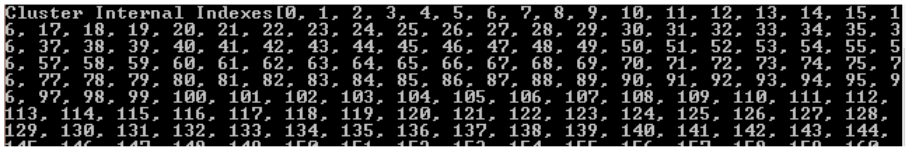
3. Running the Program for l=2

```
Features Considered are[1, 2, 3, 4, 5, 6, 7, 8, 9]

------------------------Iteration 1 ------------------------
Cluster Id : 0
Cluster Centers : Type = dense real , numRows = 1 , numCols = 11
1148278.000   4.442   3.151   3.215   2.830   3.234   3.545   3.445   2.870   1.603   4.000

Previous Centers : Type = dense real , numRows = 1 , numCols = 11
1148278.000   4.442   3.151   3.215   2.830   3.234   3.545   3.445   2.870   1.603   4.000

Cluster Points[1000025, 1002945, ...897471]
Cluster Internal Indexes[0, 1, ... -1]

Cluster Id : 1
Cluster Centers : Type = dense real , numRows = 1 , numCols = 11
1336798.000   4.442   3.151   3.215   2.830   3.234   3.545   3.445   2.870   1.603   2.000

Previous Centers : Type = dense real , numRows = 1 , numCols = 11
1336798.000   4.442   3.151   3.215   2.830   3.234   3.545   3.445   2.870   1.603   2.000

Cluster Points[1000025, 1002945, ... 897471]
Cluster Internal Indexes[0, 1, .. -1]

------------------------Has Converged : Tolerance------------------------
```

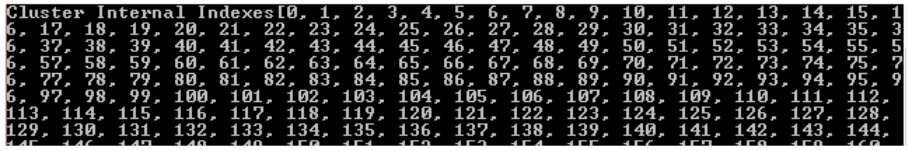
When the program was run for the breast cancer dataset, with centroids =2 and l=2, it was seen that all centroids had all points, which is expected since there are only two centroids and all points are associated with two centroids.



Cluster 1



Cluster 2



Statistical Dependency and Correlation

0.6 Problem Statement

Given two columns which can take only following values (Yes, No), (No, Yes) and (Maybe, No).

1. Statistical Dependency:
   (a) Consider X and Y be the random variables over the probability distribution of columns Do you own a house? and Do you own a car. These random variables are said to be dependent only when X is a linear/non-linear function of Y.
   (b) It is given that instead of all possible values, X and Y can only take few set of discrete values, ie (Yes, No), (No, Yes) and (Maybe, No), ie when X is Yes, Y must be NO. The similar inference can be done for other two cases.
   (c) It can be also noted that there cannot be any value which is (Yes, Yes) or (No,No) , ie no probabilistic independence, hence Random variables X and Y are dependent.
   (d) Hence from the above discussion, Both the columns are statistically dependent.
2. Statistical Correlation:
   (a) Assuming when two random variables X and Y are dependent, statistical correlation gives the expectation of a Random variable Y given X.
   (b) E.g, from the given data, converting the categorical values to nominal values, Yes = 1, No = 0, MayBe = 0.5. Also considering that all these three are in same proportion, following R code gives the correlation.

```
> cor(cbind(c(1,0,0.5),c(0,1,0)))
           [,1]       [,2]
[1,]  1.0000000 -0.8660254
[2,] -0.8660254  1.0000000
> cor(cbind(c(1,0,0.5,1,0,0.5),c(0,1,0,0,1,0)))
           [,1]       [,2]
[1,]  1.0000000 -0.8660254
[2,] -0.8660254  1.0000000
```

   (c) Since when there is X=1, Y=0, and Y=1 and X=0, There is a negative corrleation as suggested by the

References and Acknowlegements

Following links were referred for this assignment

1. http://stackoverflow.com/questions/16515370/r-basket-analysis-using-arules-package-with-unique-order-number-but-duplicate-or
2. https://prdeepakbabu.wordpress.com/2010/11/13/market-basket-analysisassociation-rule-mining-using-r-package-arules/
3. http://stats.stackexchange.com/questions/31083/how-to-produce-a-pretty-plot-of-the-results-of-k-means-cluster-analysis
4. http://stackoverflow.com/questions/12911234/clustering-large-data-matrix-using-r
5. http://www.rdatamining.com/examples/association-rules
6. http://www.rdatamining.com/docs/data-clustering-with-r

7. http://stackoverflow.com/questions/6750546/export-csv-without-col-names

8. http://stackoverflow.com/questions/32449280/how-to-create-a-decision-boundary-graph-for-knn-models-in-the-caret-package

9. http://blog.datacamp.com/machine-learning-in-r/

10. http://www.statmethods.net/advstats/cluster.html

11. http://stackoverflow.com/questions/32677141/how-to-plot-using-ggplot2-a-numeric-variable-against-categorical-variable-after

12. Peter Abeles, and Institute for Human Machine Cognition for EJML library