|  |  |
| --- | --- |
| Instructions | As an extra exercise to [Chapter 2](http://guidetodatamining.com/chapter2/), the Guide To Data Minining book website has a file containing [movie ratings for 25 movies](http://guidetodatamining.com/assets/data/Movie_Ratings.csv).  Based on the [recommender.py](https://raw.githubusercontent.com/zacharski/pg2dm-python/master/ch2/recommender.py) code discussed last week and this week:   * If you choose to use Python, create a function that loads the data into your classifier.  If you choose to use R, set up your script to load the data into your data frame. * If you choose to use Python, add to the recommend method to recommend movies for a specific person.  If you choose to use R, follow the apriori instructions earlier in this slide deck and any of the distance formulas discussed in earlier weeks.   **Answer the two questions you are given in this assignment and remember to show your code and its output.** |

**Answer the following questions (5 pts each):**

1. What associations have you discovered about the movie ratings?
2. What is the most prevalent general rule for movie ratings in your dataset? (Remember the children in 2nd class on the Titanic!)

Remember to show your code and its output!

Movie Ratings Data is been categorized in to movie names and people who had rated the movie in the rating of 1 to 5, where by seeing the data we identified that all people had rated all films because of which there considered to be some NA values in the data set. For that, taking movie ratings and people in ratio idealizing that how many people rated how much and how it would be considered as best film with more number of ratings, and high ratings. Taking these in consideration identified the ratings of the film against the people and movies.   
  
Associations:  
  
R Code :  
library(readxl)

Goteti <- read\_excel("C:/DM SEM 2/week 5/Movie.xlsx")

View(Movie)

str(Movie)

na.rm=TRUE

na.NA=TRUE

summary(Movie)

hist(Movie)

is.numeric(Movie)

is.factor(Movie)

as.factor(Movie)

sort(Movie)

library(arules)

na.rm=TRUE

na.NA=TRUE

rules.all <- apriori(Movie)

Movie <- lapply( Movie, as.factor)

library(magrittr)

inspect(rules3.all)

summary(rules.all)

rules <- apriori(Movie,

control = list(verbose=F),

parameter = list(minlen=2, supp=0.8, conf=0.8),

appearance = list(rhs=c("Movie=Alien"),

lhs=c(Heather= 4)))

rules <- apriori(Movie, control = list(verbose=F),

parameter = list(minlen=3, supp=0.002, conf=0.2),

appearance = list(default="none", rhs=c("Heather= 4"),

lhs=c("Movie=Alien", "Movie=Avatar", "Movie=Blade Runner"

)))

rules.sortedbyconf <- sort(rules, by="confidence")

quality(rules) <- round(quality(rules), digits=3)

rules.sorted <- sort(rules, by="lift")

inspect(rules.sorted)  
  
Output:

|  |
| --- |
| > library(readxl)  > Goteti <- read\_excel("C:/DM SEM 2/week 5/Movie.xlsx")  > View(Movie)  > str(Movie)  List of 26  $ Movie : Factor w/ 25 levels "Alien","Avatar",..: 1 2 3 4 5 6 7 8 9 10 ...  $ Patrick C: Factor w/ 5 levels "1","2","3","4",..: NA 4 5 4 5 4 NA NA 2 4 ...  $ Heather : Factor w/ 5 levels "1","2","3","4",..: NA 5 NA NA 4 5 5 5 NA 4 ...  $ Bryan : Factor w/ 5 levels "1","2","3","4",..: 2 5 NA 5 3 4 5 4 3 3 ...  $ Patrick T: Factor w/ 5 levels "1","2","3","4",..: NA 4 NA NA 2 3 NA NA 5 2 ...  $ Thomas : Factor w/ 5 levels "1","2","3","4",..: 5 2 5 4 4 3 4 4 2 NA ...  $ aaron : Factor w/ 4 levels "1","3","4","5": 3 NA 3 3 NA NA 3 NA NA 4 ...  $ vanessa : Factor w/ 4 levels "1","2","3","4": NA 4 NA 3 4 4 4 2 NA 2 ...  $ greg : Factor w/ 4 levels "1","3","4","5": NA 2 1 3 4 4 4 2 NA 2 ...  $ brian : Factor w/ 5 levels "1","2","3","4",..: 4 NA 5 4 3 5 5 3 1 5 ...  $ ben : Factor w/ 5 levels "1","2","3","4",..: NA 3 5 NA 4 5 5 1 NA 2 ...  $ Katherine: Factor w/ 4 levels "2","3","4","5": NA 4 NA 2 4 4 2 NA 1 3 ...  $ Jonathan : Factor w/ 4 levels "2","3","4","5": NA 3 NA 3 4 4 3 2 1 3 ...  $ Zwe : Factor w/ 5 levels "1","2","3","4",..: NA 4 NA 3 3 5 3 2 2 3 ...  $ Erin : Factor w/ 5 levels "1","2","3","4",..: NA 4 NA 4 3 5 NA 3 1 4 ...  $ Chris : Factor w/ 5 levels "1","2","3","4",..: 2 1 5 2 3 3 3 2 1 3 ...  $ Zak : Factor w/ 5 levels "1","2","3","4",..: NA 5 NA 5 3 5 5 NA 1 4 ...  $ Matt : Factor w/ 3 levels "1","3","4": NA NA NA NA NA 3 NA NA NA 1 ...  $ Chris : Factor w/ 5 levels "1","2","3","4",..: 4 NA 3 4 3 5 3 5 NA 4 ...  $ Josh : Factor w/ 3 levels "3","4","5": 1 2 NA NA NA 2 2 3 NA NA ...  $ Amy : Factor w/ 5 levels "1","2","3","4",..: NA 3 3 3 4 4 NA 3 1 NA ...  $ Valerie : Factor w/ 5 levels "1","2","3","4",..: NA 2 3 4 3 5 3 2 1 5 ...  $ Gary : Factor w/ 5 levels "1","2","3","4",..: 2 1 1 5 4 4 3 4 1 3 ...  $ Stephen : Factor w/ 5 levels "1","2","3","4",..: 5 4 NA 5 3 1 4 2 NA 5 ...  $ Jessica : Factor w/ 5 levels "1","2","3","4",..: NA NA NA NA NA 4 NA 3 1 3 ...  $ Jeff : Factor w/ 5 levels "1","2","3","4",..: 4 4 5 4 3 4 4 5 1 3 ...  > na.rm=TRUE  > na.NA=TRUE  > summary(Movie)  Length Class Mode  Movie 25 factor numeric  Patrick C 25 factor numeric  Heather 25 factor numeric  Bryan 25 factor numeric  Patrick T 25 factor numeric  Thomas 25 factor numeric  aaron 25 factor numeric  vanessa 25 factor numeric  greg 25 factor numeric  brian 25 factor numeric  ben 25 factor numeric  Katherine 25 factor numeric  Jonathan 25 factor numeric  Zwe 25 factor numeric  Erin 25 factor numeric  Chris 25 factor numeric  Zak 25 factor numeric  Matt 25 factor numeric  Chris 25 factor numeric  Josh 25 factor numeric  Amy 25 factor numeric  Valerie 25 factor numeric  Gary 25 factor numeric  Stephen 25 factor numeric  Jessica 25 factor numeric  Jeff 25 factor numeric  > hist(Movie)  Error in hist.default(Movie) : 'x' must be numeric  > is.numeric(Movie)  [1] FALSE  > is.factor(Movie)  [1] FALSE  > as.factor(Movie)  Error in sort.list(y) : 'x' must be atomic for 'sort.list'  Have you called 'sort' on a list?  > sort(Movie)  Error in sort.int(x, na.last = na.last, decreasing = decreasing, ...) :  'x' must be atomic  > library(arules)  > na.rm=TRUE  > na.NA=TRUE  > rules.all <- apriori(Movie)  Apriori  Parameter specification:  confidence minval smax arem aval originalSupport maxtime support minlen maxlen target ext  0.8 0.1 1 none FALSE TRUE 5 0.1 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: 2  set item appearances ...[0 item(s)] done [0.00s].  set transactions ...[30 item(s), 26 transaction(s)] done [0.00s].  sorting and recoding items ... [5 item(s)] done [0.00s].  creating transaction tree ... done [0.00s].  checking subsets of size 1 2 3 4 5 done [0.00s].  writing ... [80 rule(s)] done [0.00s].  creating S4 object ... done [0.00s].  Warning message:  In asMethod(object) : removing duplicated items in transactions  > Movie <- lapply( Movie, as.factor)  > library(magrittr)  > inspect(rules3.all)  lhs rhs support confidence lift  [1] {} => {2} 0.8076923 0.8076923 1.000000  [2] {} => {1} 0.8461538 0.8461538 1.000000  [3] {} => {5} 0.8846154 0.8846154 1.000000  [4] {} => {4} 0.9615385 0.9615385 1.000000  [5] {} => {3} 0.9615385 0.9615385 1.000000  [6] {2} => {1} 0.7307692 0.9047619 1.069264  [7] {1} => {2} 0.7307692 0.8636364 1.069264  [8] {2} => {5} 0.7692308 0.9523810 1.076605  [9] {5} => {2} 0.7692308 0.8695652 1.076605  [10] {2} => {4} 0.8076923 1.0000000 1.040000  [11] {4} => {2} 0.8076923 0.8400000 1.040000  [12] {2} => {3} 0.8076923 1.0000000 1.040000  [13] {3} => {2} 0.8076923 0.8400000 1.040000  [14] {1} => {5} 0.7692308 0.9090909 1.027668  [15] {5} => {1} 0.7692308 0.8695652 1.027668  [16] {1} => {4} 0.8461538 1.0000000 1.040000  [17] {4} => {1} 0.8461538 0.8800000 1.040000  [18] {1} => {3} 0.8461538 1.0000000 1.040000  [19] {3} => {1} 0.8461538 0.8800000 1.040000  [20] {5} => {4} 0.8846154 1.0000000 1.040000  [21] {4} => {5} 0.8846154 0.9200000 1.040000  [22] {5} => {3} 0.8846154 1.0000000 1.040000  [23] {3} => {5} 0.8846154 0.9200000 1.040000  [24] {4} => {3} 0.9615385 1.0000000 1.040000  [25] {3} => {4} 0.9615385 1.0000000 1.040000  [26] {1,2} => {5} 0.6923077 0.9473684 1.070938  [27] {2,5} => {1} 0.6923077 0.9000000 1.063636  [28] {1,5} => {2} 0.6923077 0.9000000 1.114286  [29] {1,2} => {4} 0.7307692 1.0000000 1.040000  [30] {2,4} => {1} 0.7307692 0.9047619 1.069264  [31] {1,4} => {2} 0.7307692 0.8636364 1.069264  [32] {1,2} => {3} 0.7307692 1.0000000 1.040000  [33] {2,3} => {1} 0.7307692 0.9047619 1.069264  [34] {1,3} => {2} 0.7307692 0.8636364 1.069264  [35] {2,5} => {4} 0.7692308 1.0000000 1.040000  [36] {2,4} => {5} 0.7692308 0.9523810 1.076605  [37] {4,5} => {2} 0.7692308 0.8695652 1.076605  [38] {2,5} => {3} 0.7692308 1.0000000 1.040000  [39] {2,3} => {5} 0.7692308 0.9523810 1.076605  [40] {3,5} => {2} 0.7692308 0.8695652 1.076605  [41] {2,4} => {3} 0.8076923 1.0000000 1.040000  [42] {2,3} => {4} 0.8076923 1.0000000 1.040000  [43] {3,4} => {2} 0.8076923 0.8400000 1.040000  [44] {1,5} => {4} 0.7692308 1.0000000 1.040000  [45] {1,4} => {5} 0.7692308 0.9090909 1.027668  [46] {4,5} => {1} 0.7692308 0.8695652 1.027668  [47] {1,5} => {3} 0.7692308 1.0000000 1.040000  [48] {1,3} => {5} 0.7692308 0.9090909 1.027668  [49] {3,5} => {1} 0.7692308 0.8695652 1.027668  [50] {1,4} => {3} 0.8461538 1.0000000 1.040000  [51] {1,3} => {4} 0.8461538 1.0000000 1.040000  [52] {3,4} => {1} 0.8461538 0.8800000 1.040000  [53] {4,5} => {3} 0.8846154 1.0000000 1.040000  [54] {3,5} => {4} 0.8846154 1.0000000 1.040000  [55] {3,4} => {5} 0.8846154 0.9200000 1.040000  [56] {1,2,5} => {4} 0.6923077 1.0000000 1.040000  [57] {1,2,4} => {5} 0.6923077 0.9473684 1.070938  [58] {2,4,5} => {1} 0.6923077 0.9000000 1.063636  [59] {1,4,5} => {2} 0.6923077 0.9000000 1.114286  [60] {1,2,5} => {3} 0.6923077 1.0000000 1.040000  [61] {1,2,3} => {5} 0.6923077 0.9473684 1.070938  [62] {2,3,5} => {1} 0.6923077 0.9000000 1.063636  [63] {1,3,5} => {2} 0.6923077 0.9000000 1.114286  [64] {1,2,4} => {3} 0.7307692 1.0000000 1.040000  [65] {1,2,3} => {4} 0.7307692 1.0000000 1.040000  [66] {2,3,4} => {1} 0.7307692 0.9047619 1.069264  [67] {1,3,4} => {2} 0.7307692 0.8636364 1.069264  [68] {2,4,5} => {3} 0.7692308 1.0000000 1.040000  [69] {2,3,5} => {4} 0.7692308 1.0000000 1.040000  [70] {2,3,4} => {5} 0.7692308 0.9523810 1.076605  [71] {3,4,5} => {2} 0.7692308 0.8695652 1.076605  [72] {1,4,5} => {3} 0.7692308 1.0000000 1.040000  [73] {1,3,5} => {4} 0.7692308 1.0000000 1.040000  [74] {1,3,4} => {5} 0.7692308 0.9090909 1.027668  [75] {3,4,5} => {1} 0.7692308 0.8695652 1.027668  [76] {1,2,4,5} => {3} 0.6923077 1.0000000 1.040000  [77] {1,2,3,5} => {4} 0.6923077 1.0000000 1.040000  [78] {1,2,3,4} => {5} 0.6923077 0.9473684 1.070938  [79] {2,3,4,5} => {1} 0.6923077 0.9000000 1.063636  [80] {1,3,4,5} => {2} 0.6923077 0.9000000 1.114286  > summary(rules.all)  set of 80 rules  rule length distribution (lhs + rhs):sizes  1 2 3 4 5  5 20 30 20 5  Min. 1st Qu. Median Mean 3rd Qu. Max.  1 2 3 3 4 5  summary of quality measures:  support confidence lift  Min. :0.6923 Min. :0.8077 Min. :1.000  1st Qu.:0.7308 1st Qu.:0.8835 1st Qu.:1.040  Median :0.7692 Median :0.9337 Median :1.040  Mean :0.7808 Mean :0.9356 Mean :1.049  3rd Qu.:0.8173 3rd Qu.:1.0000 3rd Qu.:1.069  Max. :0.9615 Max. :1.0000 Max. :1.114  mining info:  data ntransactions support confidence  Movie 26 0.1 0.8  > rules <- apriori(Movie,  + control = list(verbose=F),  + parameter = list(minlen=2, supp=0.8, conf=0.8),  + appearance = list(rhs=c("Movie=Alien"),  + lhs=c(Heather= 4)))  Error in asMethod(object) : Movie=Alien is an unknown item label  In addition: Warning message:  In asMethod(object) : removing duplicated items in transactions  > rules <- apriori(Movie, control = list(verbose=F),  + parameter = list(minlen=3, supp=0.002, conf=0.2),  + appearance = list(default="none", rhs=c("Heather= 4"),  + lhs=c("Movie=Alien", "Movie=Avatar", "Movie=Blade Runner"  + )))  Error in asMethod(object) :  Movie=Alien is an unknown item label, Movie=Avatar is an unknown item label, Movie=Blade Runner is an unknown item label  In addition: Warning message:  In asMethod(object) : removing duplicated items in transactions  > rules.sortedbyconf <- sort(rules, by="confidence")  > quality(rules) <- round(quality(rules), digits=3)  > rules.sorted <- sort(rules, by="lift")  > inspect(rules.sorted)  lhs rhs support confidence lift  [1] {2} => {4} 0.808 1 1.04  [2] {1} => {4} 0.846 1 1.04  [3] {5} => {4} 0.885 1 1.04  [4] {3} => {4} 0.962 1 1.04  [5] {2,3} => {4} 0.808 1 1.04  [6] {1,3} => {4} 0.846 1 1.04  [7] {3,5} => {4} 0.885 1 1.04 |
|  |
| |  | | --- | | > | |

By using Movie <- lapply( Movie, as.factor) , got the values in factor, to ensure values to be modified to factor values and then identify the factor values which movie rating is closest and how many ratings paired up. But here very important point is that redundant ratings is effecting to identify the summary and get to a closer look, so instead of over all rule we can identify using matrix, and pruned value will bring to closer look and taking all value in constraint from rating 1, 2,3,4,5, so that ; a user who had given what all rating can be identified and can be removed using a pruned data.   
  
Example of all rating of one user and one film:   
  
rules <- apriori(Movie,

control = list(verbose=F),

parameter = list(minlen=2, supp=0.8, conf=0.8),

appearance = list(rhs=c("Movie=Alien"),

lhs=c(Heather= 4)))

rules <- apriori(Movie, control = list(verbose=F),

parameter = list(minlen=3, supp=0.002, conf=0.2),

appearance = list(default="none", rhs=c("Heather= 4"),

lhs=c("Movie=Alien", "Movie=Avatar", "Movie=Blade Runner"

)))

rules1 <- apriori(Movie, control = list(verbose=F),

parameter = list(minlen=3, supp=0.002, conf=0.2),

appearance = list(default="none", rhs=c("Heather= 1"),

lhs=c("Movie=Alien", "Movie=Avatar", "Movie=Blade Runner"

)))

rules2 <- apriori(Movie, control = list(verbose=F),

parameter = list(minlen=3, supp=0.002, conf=0.2),

appearance = list(default="none", rhs=c("Heather= 2"),

lhs=c("Movie=Alien", "Movie=Avatar", "Movie=Blade Runner"

)))

rules3 <- apriori(Movie, control = list(verbose=F),

parameter = list(minlen=3, supp=0.002, conf=0.2),

appearance = list(default="none", rhs=c("Heather= 3"),

lhs=c("Movie=Alien", "Movie=Avatar", "Movie=Blade Runner"

)))

rules4 <- apriori(Movie, control = list(verbose=F),

parameter = list(minlen=3, supp=0.002, conf=0.2),

appearance = list(default="none", rhs=c("Heather= 5"),

lhs=c("Movie=Alien", "Movie=Avatar", "Movie=Blade Runner"

)))  
  
If we now identify the prune, and matrix the data, we get a closer look of the data, and get more clear identification of how many ratings are identified with one move using matrix and one user for one film and same thing would be identified with all films.   
  
  
R Code:   
  
subset.matrix <- is.subset(rules.sorted, rules.sorted)

subset.matrix[lower.tri(subset.matrix, diag = T)] <- NA

redundant <- colSums(subset.matrix, na.rm = T) >= 1

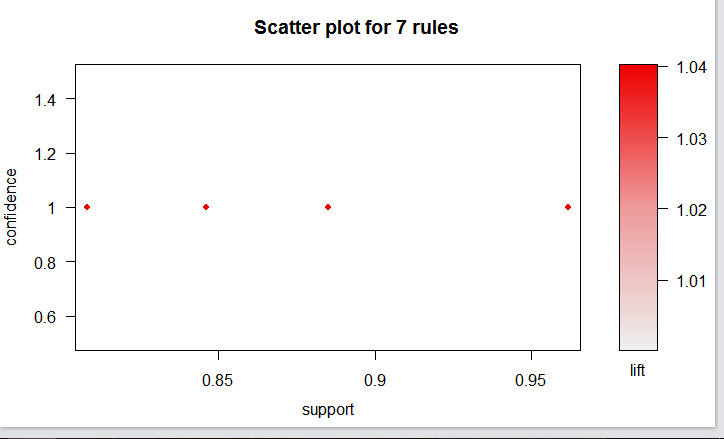
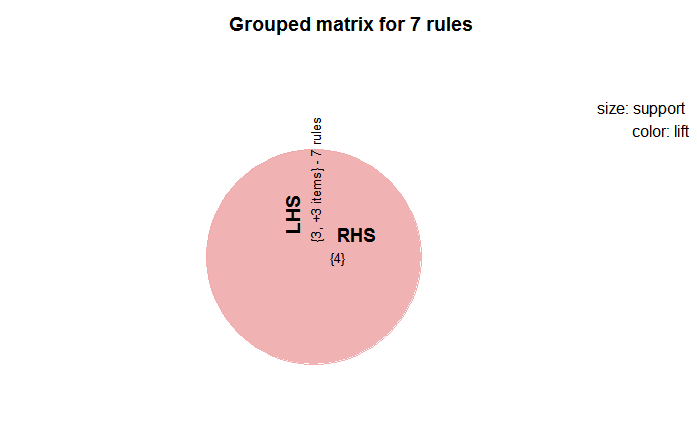
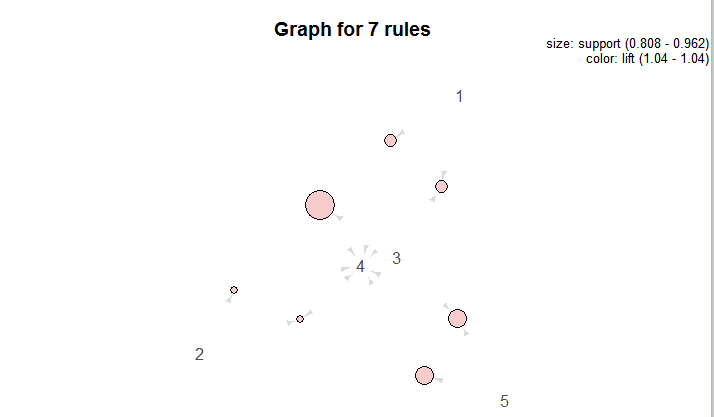
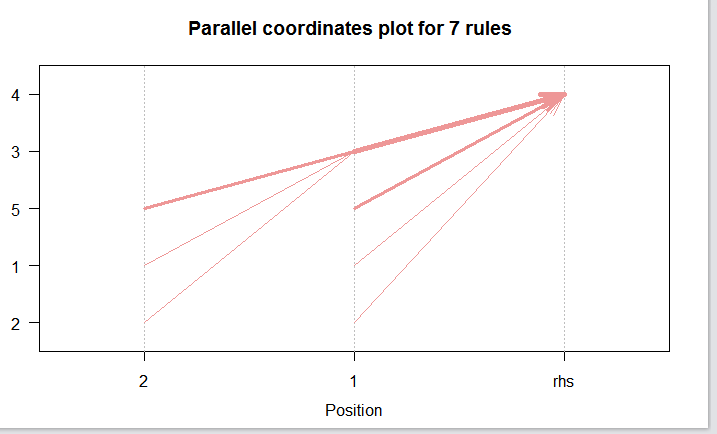
which(redundant)

rules.pruned=rules.sorted[!redundant]

inspect(rules.pruned)  
  
  
Output:

|  |
| --- |
| > inspect(rules.sorted)  lhs rhs support confidence lift  [1] {2} => {4} 0.808 1 1.04  [2] {1} => {4} 0.846 1 1.04  [3] {5} => {4} 0.885 1 1.04  [4] {3} => {4} 0.962 1 1.04  [5] {2,3} => {4} 0.808 1 1.04  [6] {1,3} => {4} 0.846 1 1.04  [7] {3,5} => {4} 0.885 1 1.04  > subset.matrix <- is.subset(rules.sorted, rules.sorted)  > subset.matrix[lower.tri(subset.matrix, diag = T)] <- NA  > redundant <- colSums(subset.matrix, na.rm = T) >= 1  > which(redundant)  {2,3,4} {1,3,4} {3,4,5}  5 6 7  > rules.pruned=rules.sorted[!redundant]  > inspect(rules.pruned)  lhs rhs support confidence lift  [1] {2} => {4} 0.808 1 1.04  [2] {1} => {4} 0.846 1 1.04  [3] {5} => {4} 0.885 1 1.04  [4] {3} => {4} 0.962 1 1.04 |
|  |
| |  | | --- | | > | |

Kept the support and confidence as 0.8 as because the values are ranging in this region.

  
  
  
  
  
  
  
  
  
inspect(rules.sorted[5:2])

lhs rhs support confidence lift

[1] {2,3} => {4} 0.808 1 1.04

[2] {3} => {4} 0.962 1 1.04

[3] {5} => {4} 0.885 1 1.04

[4] {1} => {4} 0.846 1 1.04

As on the value of support and confidence

2. What is the most prevalent general rule for movie ratings in your dataset? (Remember the children in 2nd class on the Titanic!)

In overall 80 data values which are been identified with data set observations, it would become very difficult to identify the best outcomes from it and answer the people, so for that it was been identified to make best output it would be easy to identify what rating and for which move and which person. So , For that we make a identification to have a best rules need ot be identified with 7 values are been identified with support and confidence keeping with 0.8 s a regional value, we get the best interpret the value and making the best outcomes, using the spots.   
  
  
  
R Code:   
  
rules <- apriori(Movie,

control = list(verbose=F),

parameter = list(minlen=2, supp=0.6, conf=0.6),

appearance = list(rhs=c("Movie=Alien"),

lhs=c(Heather= 4)))

rules <- apriori(Movie, control = list(verbose=F),

parameter = list(minlen=3, supp=0.002, conf=0.2),

appearance = list(default="none", rhs=c("Heather= 4"),

lhs=c("Movie=Alien", "Movie=Avatar", "Movie=Blade Runner"

)))

rules1 <- apriori(Movie, control = list(verbose=F),

parameter = list(minlen=3, supp=0.002, conf=0.2),

appearance = list(default="none", rhs=c("Heather= 1"),

lhs=c("Movie=Alien", "Movie=Avatar", "Movie=Blade Runner"

)))

rules2 <- apriori(Movie, control = list(verbose=F),

parameter = list(minlen=3, supp=0.002, conf=0.2),

appearance = list(default="none", rhs=c("Heather= 2"),

lhs=c("Movie=Alien", "Movie=Avatar", "Movie=Blade Runner"

)))

rules3 <- apriori(Movie, control = list(verbose=F),

parameter = list(minlen=3, supp=0.002, conf=0.2),

appearance = list(default="none", rhs=c("Heather= 3"),

lhs=c("Movie=Alien", "Movie=Avatar", "Movie=Blade Runner"

)))

rules4 <- apriori(Movie, control = list(verbose=F),

parameter = list(minlen=3, supp=0.002, conf=0.2),

appearance = list(default="none", rhs=c("Heather= 5"),

lhs=c("Movie=Alien", "Movie=Avatar", "Movie=Blade Runner"

)))

rules.sortedbyconf <- sort(rules, by="confidence")

quality(rules) <- round(quality(rules), digits=3)

rules.sorted <- sort(rules, by="lift")

inspect(rules.sorted)

library(arulesViz)

plot(rules.sorted)

plot(rules.sorted, method = "grouped")

plot(rules.sorted, method = "graph")

plot(rules.sorted, method = "paracoord", control = list(reorder = TRUE))

inspect(rules.sorted[5:2])

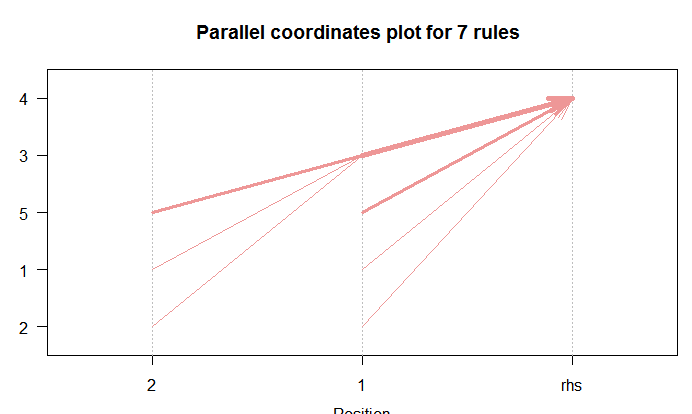
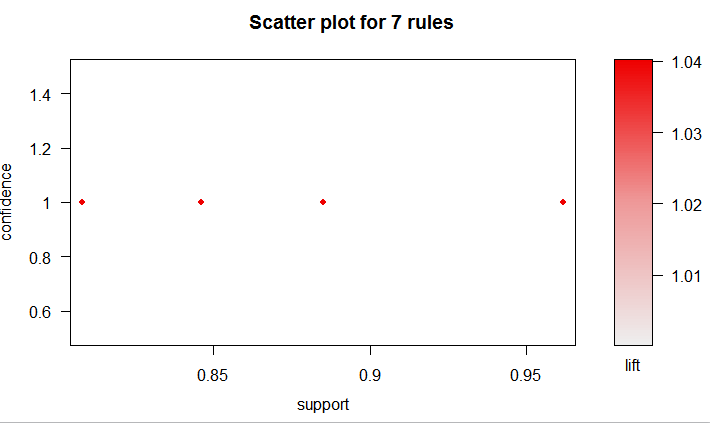
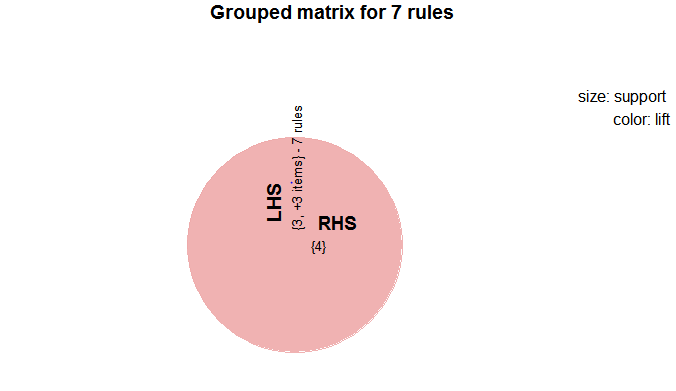
subset.matrix <- is.subset(rules.sorted, rules.sorted)

subset.matrix[lower.tri(subset.matrix, diag = T)] <- NA

redundant <- colSums(subset.matrix, na.rm = T) >= 1

which(redundant)

rules.pruned=rules.sorted[!redundant]

inspect(rules.pruned)  
  
  
Keeping the code with best 7 rules which would be nearest with pruned value, so after that it was been moved with redundant value , and pruned value with 0.8 with lhs and rhs identified with value which was identified with moving with spot of rating 4 and having moving to right side, so make identified with values of support and confidence of 0.8 and values are been decreased from 80 to 7.   
  
  
  
  
  
  
  
  
  
  
Values varies as per the confidence and support values are been deferred and change to more less value .