Homework 6

Kenigbolo Meya Stephen

March 14, 2016

This is an R Markdown document. Markdown is a simple formatting syntax for authoring HTML, PDF, and MS Word documents. For more details on using R Markdown see <http://rmarkdown.rstudio.com>.

1. Construct an FP-tree using the same data set as last week (use the support count threshold smin = 2). Explain all the steps of the tree construction and draw a resulting tree. Based on this tree answer the questions: how many transactions contain {E,F} and {C,H} ?

library(arules)

## Warning: package 'arules' was built under R version 3.2.4

## Loading required package: Matrix

##   
## Attaching package: 'arules'

## The following objects are masked from 'package:base':  
##   
## %in%, abbreviate, write

abcset <- read.transactions("C:/Users/Kenigbolo PC/Desktop/Data Mining/abcset.csv", rm.duplicates= FALSE, format="basket", sep=",")  
  
  
inspect(abcset)

## items   
## 1 {A,B,C,F,H}  
## 2 {C,E,F,H}   
## 3 {B,D,E}   
## 4 {A,C,F,H}   
## 5 {A,E,F}   
## 6 {B,D,H}   
## 7 {B,C,D,E,F}  
## 8 {A,C,E,H}   
## 9 {A,E,G}   
## 10 {B,E,H}

{E,F} = 3 {H,C} = 4

1. Evaluate various interestingness measures for association rules. Generate randomly a broad range of various 2x2 contingency tables (f11, f10, f01, f00) for N=10,000 items. Sample the space so that each cell individually, in pairs, or triples is larger than "others". In this way sample at least 10,000 different possible contingency tables. Calculate 5 various scores based on those data (feel free to select) and report 10 top 2x2 tables that are the "best" according to that measure. Use rows to represent the 4 numbers; and if useful, also the marginal sums and N.

randomtablevalues <- sample(1:4,1000,rep=TRUE,prob=c(.2,.3,.2,.3))  
table(randomtablevalues)

## randomtablevalues  
## 1 2 3 4   
## 221 290 215 274

func = ceiling(runif(10000, 0,1000))  
randomGen = t(sapply(func, function(z) c(z, 4\*z, 3\*z, 2\*z)))  
  
colnames(randomGen) <- c("f11","f01","f10","f00")  
  
F1plus <- rowSums(randomGen[, c(1, 3)])  
F0plus <- rowSums(randomGen[, c(2, 4)])  
Fplus1 <- rowSums(randomGen[, c(1, 2)])  
Fplus0 <- rowSums(randomGen[, c(3, 4)])  
T <- rowSums(randomGen[, c(1, 2, 3, 4)])  
  
randomGen <- cbind(randomGen, F1plus)  
randomGen <- cbind(randomGen, F0plus)  
randomGen <- cbind(randomGen, Fplus1 )  
randomGen <- cbind(randomGen, Fplus0 )  
randomGen <- cbind(randomGen, T )  
  
  
oddsRatiofunction <- function(f11,f01,f10,f00) {  
 oddsratio <- ((f11/f00)/(f10/f01))  
 return(oddsratio)  
}  
  
jaccard <- function(f11,F1plus,Fplus1) {  
 jaccard <- (f11/(F1plus+Fplus1-f11))  
 return(jaccard)  
}  
  
laplace <- function(f11,F1plus) {  
 laplace <- ((f11+1)/(F1plus+2))  
 return (laplace)  
}  
  
certainityFactor <- function(f11,F1plus,Fplus1){  
 certfact <- (((f11/F1plus)-(Fplus1/10000))/(1-(Fplus1/10000)))  
 return (certfact)  
}  
  
addedValue <- function(f11,F1plus,Fplus1){  
 addedvalue <- (((f11/F1plus)-(Fplus1/10000)))  
 return (addedvalue)  
}  
  
jaccard <- jaccard(randomGen[,c(1)], randomGen[,c(5)], randomGen[,c(7)])  
oddRatio <- oddsRatiofunction(randomGen[,c(1)], randomGen[,c(2)], randomGen[,c(3)], randomGen[,c(4)])  
laplace <- laplace(randomGen[,c(1)], randomGen[,c(5)])  
certainityFactor <- certainityFactor(randomGen[,c(1)],randomGen[,c(5)], randomGen[,c(7)])  
addedValue <- addedValue(randomGen[,c(1)],randomGen[,c(5)], randomGen[,c(7)])  
  
  
randomGen <- cbind(randomGen, oddRatio)  
randomGen <- cbind(randomGen, jaccard)  
randomGen <- cbind(randomGen, laplace)  
randomGen <- cbind(randomGen, certainityFactor)  
randomGen <- cbind(randomGen, addedValue)  
  
  
top10 <- subset(randomGen, laplace < 0.27)  
head(top10, 10)

## f11 f01 f10 f00 F1plus F0plus Fplus1 Fplus0 T oddRatio  
## [1,] 90 360 270 180 360 540 450 450 900 0.6666667  
## [2,] 934 3736 2802 1868 3736 5604 4670 4670 9340 0.6666667  
## [3,] 597 2388 1791 1194 2388 3582 2985 2985 5970 0.6666667  
## [4,] 897 3588 2691 1794 3588 5382 4485 4485 8970 0.6666667  
## [5,] 301 1204 903 602 1204 1806 1505 1505 3010 0.6666667  
## [6,] 663 2652 1989 1326 2652 3978 3315 3315 6630 0.6666667  
## [7,] 193 772 579 386 772 1158 965 965 1930 0.6666667  
## [8,] 158 632 474 316 632 948 790 790 1580 0.6666667  
## [9,] 436 1744 1308 872 1744 2616 2180 2180 4360 0.6666667  
## [10,] 828 3312 2484 1656 3312 4968 4140 4140 8280 0.6666667  
## jaccard laplace certainityFactor addedValue  
## [1,] 0.125 0.2513812 0.21465969 0.2050  
## [2,] 0.125 0.2501338 -0.40712946 -0.2170  
## [3,] 0.125 0.2502092 -0.06913756 -0.0485  
## [4,] 0.125 0.2501393 -0.35992747 -0.1985  
## [5,] 0.125 0.2504146 0.11712772 0.0995  
## [6,] 0.125 0.2501884 -0.12191473 -0.0815  
## [7,] 0.125 0.2506460 0.16989485 0.1535  
## [8,] 0.125 0.2507886 0.18566775 0.1710  
## [9,] 0.125 0.2502864 0.04092072 0.0320  
## [10,] 0.125 0.2501509 -0.27986348 -0.1640

1. Compare interestingness measures starting from various fixed examples of (f11, f10, f01, f00) and experimenting with each of the four values - by increasing or decreasing it, one at a time.

newRandGen <- head(randomGen, 1)  
print(newRandGen)

## f11 f01 f10 f00 F1plus F0plus Fplus1 Fplus0 T oddRatio jaccard  
## [1,] 90 360 270 180 360 540 450 450 900 0.6666667 0.125  
## laplace certainityFactor addedValue  
## [1,] 0.2513812 0.2146597 0.205

newRandGen[1, 1] + 10

## f11   
## 100

addedValuefunc <- function(f11,F1plus,Fplus1){  
 addedvalue <- (((f11/F1plus)-(Fplus1/10000)))  
 return (addedvalue)  
}  
addedValue <- addedValuefunc(newRandGen[,c(1)],newRandGen[,c(5)], newRandGen[,c(7)])  
print(addedValue)

## f11   
## 0.205

newRandGen[1, 2] + 10

## f01   
## 370

addedValue <- addedValuefunc(newRandGen[,c(1)],newRandGen[,c(5)], newRandGen[,c(7)])  
print(addedValue)

## f11   
## 0.205

newRandGen[1, 3] + 10

## f10   
## 280

addedValue <- addedValuefunc(newRandGen[,c(1)],newRandGen[,c(5)], newRandGen[,c(7)])  
print(addedValue)

## f11   
## 0.205

newRandGen[1, 4] + 10

## f00   
## 190

addedValue <- addedValuefunc(newRandGen[,c(1)],newRandGen[,c(5)], newRandGen[,c(7)])  
print(addedValue)

## f11   
## 0.205

From the above we can see that the addedvalue measure doesn't change at all even though we increase the values one at a time.

1. Install R packages arules and arulesViz Get the Titanic survival data from <https://courses.cs.ut.ee/MTAT.03.183/2014_spring/uploads/Main/titanic.txt>

Make sure to explore all these commands, vary parameters, read the manual ... Try to vary them to provide nice interpretable outputs. See also 6. and 7.

library(arulesViz)

## Warning: package 'arulesViz' was built under R version 3.2.4

## Loading required package: grid

library(arulesViz)  
  
titanic <- read.table( "C:/Users/Kenigbolo PC/Desktop/Data Mining/titanic.txt", sep = ',' , header = TRUE)  
  
#observe the data  
##first 6 observations  
head(titanic)

## Class Sex Age Survived  
## 1 3rd Male Child No  
## 2 3rd Male Child No  
## 3 3rd Male Child No  
## 4 3rd Male Child No  
## 5 3rd Male Child No  
## 6 3rd Male Child No

#types of features  
str(titanic)

## 'data.frame': 2201 obs. of 4 variables:  
## $ Class : Factor w/ 4 levels "1st","2nd","3rd",..: 3 3 3 3 3 3 3 3 3 3 ...  
## $ Sex : Factor w/ 2 levels "Female","Male": 2 2 2 2 2 2 2 2 2 2 ...  
## $ Age : Factor w/ 2 levels "Adult","Child": 2 2 2 2 2 2 2 2 2 2 ...  
## $ Survived: Factor w/ 2 levels "No","Yes": 1 1 1 1 1 1 1 1 1 1 ...

#dimensionality of the data  
dim(titanic)

## [1] 2201 4

#load package for frequent set mining  
library(arules)  
  
  
#run apriori algorithm with default settings  
rules = apriori(titanic)

## Apriori  
##   
## Parameter specification:  
## confidence minval smax arem aval originalSupport support minlen maxlen  
## 0.8 0.1 1 none FALSE TRUE 0.1 1 10  
## target ext  
## rules FALSE  
##   
## Algorithmic control:  
## filter tree heap memopt load sort verbose  
## 0.1 TRUE TRUE FALSE TRUE 2 TRUE  
##   
## Absolute minimum support count: 220   
##   
## set item appearances ...[0 item(s)] done [0.00s].  
## set transactions ...[10 item(s), 2201 transaction(s)] done [0.00s].  
## sorting and recoding items ... [9 item(s)] done [0.00s].  
## creating transaction tree ... done [0.00s].  
## checking subsets of size 1 2 3 4 done [0.00s].  
## writing ... [27 rule(s)] done [0.00s].  
## creating S4 object ... done [0.00s].

#inspection of the result  
inspect(rules)

## lhs rhs support   
## 1 {} => {Age=Adult} 0.9504771  
## 2 {Class=2nd} => {Age=Adult} 0.1185825  
## 3 {Class=1st} => {Age=Adult} 0.1449341  
## 4 {Sex=Female} => {Age=Adult} 0.1930940  
## 5 {Class=3rd} => {Age=Adult} 0.2848705  
## 6 {Survived=Yes} => {Age=Adult} 0.2971377  
## 7 {Class=Crew} => {Sex=Male} 0.3916402  
## 8 {Class=Crew} => {Age=Adult} 0.4020900  
## 9 {Survived=No} => {Sex=Male} 0.6197183  
## 10 {Survived=No} => {Age=Adult} 0.6533394  
## 11 {Sex=Male} => {Age=Adult} 0.7573830  
## 12 {Sex=Female,Survived=Yes} => {Age=Adult} 0.1435711  
## 13 {Class=3rd,Sex=Male} => {Survived=No} 0.1917310  
## 14 {Class=3rd,Survived=No} => {Age=Adult} 0.2162653  
## 15 {Class=3rd,Sex=Male} => {Age=Adult} 0.2099046  
## 16 {Sex=Male,Survived=Yes} => {Age=Adult} 0.1535666  
## 17 {Class=Crew,Survived=No} => {Sex=Male} 0.3044071  
## 18 {Class=Crew,Survived=No} => {Age=Adult} 0.3057701  
## 19 {Class=Crew,Sex=Male} => {Age=Adult} 0.3916402  
## 20 {Class=Crew,Age=Adult} => {Sex=Male} 0.3916402  
## 21 {Sex=Male,Survived=No} => {Age=Adult} 0.6038164  
## 22 {Age=Adult,Survived=No} => {Sex=Male} 0.6038164  
## 23 {Class=3rd,Sex=Male,Survived=No} => {Age=Adult} 0.1758292  
## 24 {Class=3rd,Age=Adult,Survived=No} => {Sex=Male} 0.1758292  
## 25 {Class=3rd,Sex=Male,Age=Adult} => {Survived=No} 0.1758292  
## 26 {Class=Crew,Sex=Male,Survived=No} => {Age=Adult} 0.3044071  
## 27 {Class=Crew,Age=Adult,Survived=No} => {Sex=Male} 0.3044071  
## confidence lift   
## 1 0.9504771 1.0000000  
## 2 0.9157895 0.9635051  
## 3 0.9815385 1.0326798  
## 4 0.9042553 0.9513700  
## 5 0.8881020 0.9343750  
## 6 0.9198312 0.9677574  
## 7 0.9740113 1.2384742  
## 8 1.0000000 1.0521033  
## 9 0.9154362 1.1639949  
## 10 0.9651007 1.0153856  
## 11 0.9630272 1.0132040  
## 12 0.9186047 0.9664669  
## 13 0.8274510 1.2222950  
## 14 0.9015152 0.9484870  
## 15 0.9058824 0.9530818  
## 16 0.9209809 0.9689670  
## 17 0.9955423 1.2658514  
## 18 1.0000000 1.0521033  
## 19 1.0000000 1.0521033  
## 20 0.9740113 1.2384742  
## 21 0.9743402 1.0251065  
## 22 0.9242003 1.1751385  
## 23 0.9170616 0.9648435  
## 24 0.8130252 1.0337773  
## 25 0.8376623 1.2373791  
## 26 1.0000000 1.0521033  
## 27 0.9955423 1.2658514

#now let us assume, we want to see only those rules that have rhs as survived:  
rules = apriori(titanic,appearance = list(rhs=c("Survived=No", "Survived=Yes"),default="lhs"))

## Apriori  
##   
## Parameter specification:  
## confidence minval smax arem aval originalSupport support minlen maxlen  
## 0.8 0.1 1 none FALSE TRUE 0.1 1 10  
## target ext  
## rules FALSE  
##   
## Algorithmic control:  
## filter tree heap memopt load sort verbose  
## 0.1 TRUE TRUE FALSE TRUE 2 TRUE  
##   
## Absolute minimum support count: 220   
##   
## set item appearances ...[2 item(s)] done [0.00s].  
## set transactions ...[10 item(s), 2201 transaction(s)] done [0.00s].  
## sorting and recoding items ... [9 item(s)] done [0.00s].  
## creating transaction tree ... done [0.00s].  
## checking subsets of size 1 2 3 4 done [0.00s].  
## writing ... [2 rule(s)] done [0.00s].  
## creating S4 object ... done [0.00s].

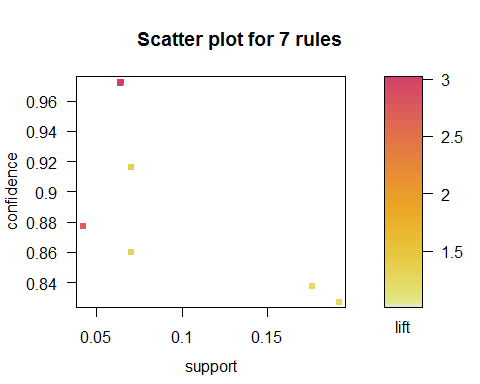
inspect(rules)

## lhs rhs support confidence  
## 1 {Class=3rd,Sex=Male} => {Survived=No} 0.1917310 0.8274510   
## 2 {Class=3rd,Sex=Male,Age=Adult} => {Survived=No} 0.1758292 0.8376623   
## lift   
## 1 1.222295  
## 2 1.237379

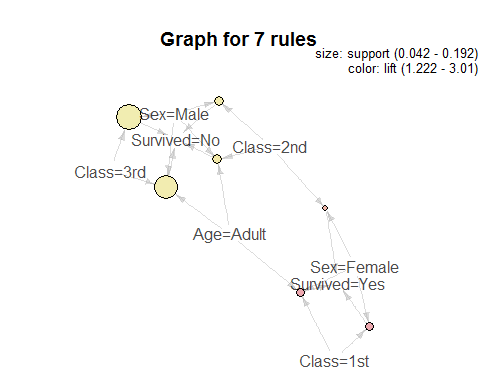
#let us relax the default settings for the rules we are looking for   
rules = apriori(titanic,parameter = list(minlen=2, supp=0.04, conf=0.8),appearance = list(rhs=c("Survived=No", "Survived=Yes"),default="lhs"))

## Apriori  
##   
## Parameter specification:  
## confidence minval smax arem aval originalSupport support minlen maxlen  
## 0.8 0.1 1 none FALSE TRUE 0.04 2 10  
## target ext  
## rules FALSE  
##   
## Algorithmic control:  
## filter tree heap memopt load sort verbose  
## 0.1 TRUE TRUE FALSE TRUE 2 TRUE  
##   
## Absolute minimum support count: 88   
##   
## set item appearances ...[2 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 ... [7 rule(s)] done [0.00s].  
## creating S4 object ... done [0.00s].

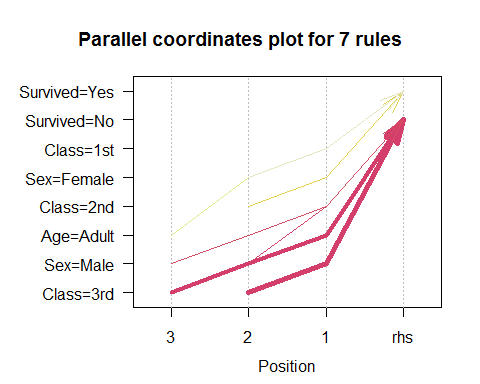
#visualization  
library(arulesViz)  
plot(rules)



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



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



1. Report clearly the most "interesting" rules discovered from Titanic data, and how you came up with those in R.

rules <- apriori(titanic, parameter = list(minlen=2, supp=0.005, conf=0.8), appearance =list(rhs=c("Survived=No", "Survived=Yes"), default="lhs"), control = list(verbose=F))  
  
#I will output interestingness based on Lift  
sortedrulesdata <- sort(rules, by="lift")  
inspect(sortedrulesdata)

## lhs rhs support   
## 1 {Class=2nd,Age=Child} => {Survived=Yes} 0.010904134  
## 7 {Class=2nd,Sex=Female,Age=Child} => {Survived=Yes} 0.005906406  
## 4 {Class=1st,Sex=Female} => {Survived=Yes} 0.064061790  
## 10 {Class=1st,Sex=Female,Age=Adult} => {Survived=Yes} 0.063607451  
## 2 {Class=2nd,Sex=Female} => {Survived=Yes} 0.042253521  
## 5 {Class=Crew,Sex=Female} => {Survived=Yes} 0.009086779  
## 11 {Class=Crew,Sex=Female,Age=Adult} => {Survived=Yes} 0.009086779  
## 8 {Class=2nd,Sex=Female,Age=Adult} => {Survived=Yes} 0.036347115  
## 9 {Class=2nd,Sex=Male,Age=Adult} => {Survived=No} 0.069968196  
## 3 {Class=2nd,Sex=Male} => {Survived=No} 0.069968196  
## 12 {Class=3rd,Sex=Male,Age=Adult} => {Survived=No} 0.175829169  
## 6 {Class=3rd,Sex=Male} => {Survived=No} 0.191731031  
## confidence lift   
## 1 1.0000000 3.095640  
## 7 1.0000000 3.095640  
## 4 0.9724138 3.010243  
## 10 0.9722222 3.009650  
## 2 0.8773585 2.715986  
## 5 0.8695652 2.691861  
## 11 0.8695652 2.691861  
## 8 0.8602151 2.662916  
## 9 0.9166667 1.354083  
## 3 0.8603352 1.270871  
## 12 0.8376623 1.237379  
## 6 0.8274510 1.222295

## I will calculate hyperconfidence and add it to the quality slot for my sorted rules  
quality(sortedrulesdata) <- cbind(quality(sortedrulesdata), hyperConfidence = interestMeasure(rules, measure = "hyperConfidence", transactions = titanic))  
  
## I will output the hyperconfidence in my "interesting" rules also  
inspect(head(sort(sortedrulesdata, by = "hyperConfidence")))

## lhs rhs support   
## 1 {Class=2nd,Age=Child} => {Survived=Yes} 0.010904134  
## 7 {Class=2nd,Sex=Female,Age=Child} => {Survived=Yes} 0.005906406  
## 4 {Class=1st,Sex=Female} => {Survived=Yes} 0.064061790  
## 10 {Class=1st,Sex=Female,Age=Adult} => {Survived=Yes} 0.063607451  
## 2 {Class=2nd,Sex=Female} => {Survived=Yes} 0.042253521  
## 5 {Class=Crew,Sex=Female} => {Survived=Yes} 0.009086779  
## confidence lift hyperConfidence  
## 1 1.0000000 3.095640 0   
## 7 1.0000000 3.095640 0   
## 4 0.9724138 3.010243 0   
## 10 0.9722222 3.009650 0   
## 2 0.8773585 2.715986 0   
## 5 0.8695652 2.691861 0

#I will output interestingness based on hyperConfidence measure for the first five values  
hyperconfidence <- sort(sortedrulesdata, by="hyperConfidence")  
inspect(hyperconfidence)

## lhs rhs support   
## 1 {Class=2nd,Age=Child} => {Survived=Yes} 0.010904134  
## 7 {Class=2nd,Sex=Female,Age=Child} => {Survived=Yes} 0.005906406  
## 4 {Class=1st,Sex=Female} => {Survived=Yes} 0.064061790  
## 10 {Class=1st,Sex=Female,Age=Adult} => {Survived=Yes} 0.063607451  
## 2 {Class=2nd,Sex=Female} => {Survived=Yes} 0.042253521  
## 5 {Class=Crew,Sex=Female} => {Survived=Yes} 0.009086779  
## 11 {Class=Crew,Sex=Female,Age=Adult} => {Survived=Yes} 0.009086779  
## 8 {Class=2nd,Sex=Female,Age=Adult} => {Survived=Yes} 0.036347115  
## 9 {Class=2nd,Sex=Male,Age=Adult} => {Survived=No} 0.069968196  
## 3 {Class=2nd,Sex=Male} => {Survived=No} 0.069968196  
## 12 {Class=3rd,Sex=Male,Age=Adult} => {Survived=No} 0.175829169  
## 6 {Class=3rd,Sex=Male} => {Survived=No} 0.191731031  
## confidence lift hyperConfidence  
## 1 1.0000000 3.095640 0   
## 7 1.0000000 3.095640 0   
## 4 0.9724138 3.010243 0   
## 10 0.9722222 3.009650 0   
## 2 0.8773585 2.715986 0   
## 5 0.8695652 2.691861 0   
## 11 0.8695652 2.691861 0   
## 8 0.8602151 2.662916 0   
## 9 0.9166667 1.354083 0   
## 3 0.8603352 1.270871 0   
## 12 0.8376623 1.237379 0   
## 6 0.8274510 1.222295 0

## Now I will calculate measures of leverage and oddsRatio   
interesting <- interestMeasure(rules, c("leverage", "oddsRatio" ), transactions = titanic)  
inspect(head(rules))

## lhs rhs support confidence  
## 1 {Class=2nd,Age=Child} => {Survived=Yes} 0.010904134 1.0000000   
## 2 {Class=2nd,Sex=Female} => {Survived=Yes} 0.042253521 0.8773585   
## 3 {Class=2nd,Sex=Male} => {Survived=No} 0.069968196 0.8603352   
## 4 {Class=1st,Sex=Female} => {Survived=Yes} 0.064061790 0.9724138   
## 5 {Class=Crew,Sex=Female} => {Survived=Yes} 0.009086779 0.8695652   
## 6 {Class=3rd,Sex=Male} => {Survived=No} 0.191731031 0.8274510   
## lift   
## 1 3.095640  
## 2 2.715986  
## 3 1.270871  
## 4 3.010243  
## 5 2.691861  
## 6 1.222295

head(interesting)

## leverage oddsRatio  
## 1 0.007381718 NA  
## 2 0.026696180 17.097461  
## 3 0.014912886 3.162994  
## 4 0.042780521 91.897368  
## 5 0.005711129 14.346358  
## 6 0.034869533 2.797348

## I will calculate all available measures for the first 5 rules and show them in a table format where the measures are rows   
t(interestMeasure(head(rules, 5), transactions = titanic))

## [,1] [,2] [,3] [,4]  
## support 0.010904134 0.042253521 0.069968196 0.064061790  
## coverage 0.010904134 0.048159927 0.081326670 0.065879146  
## confidence 1.000000000 0.877358491 0.860335196 0.972413793  
## lift 3.095639944 2.715985988 1.270870983 3.010242980  
## leverage 0.007381718 0.026696180 0.014912886 0.042780521  
## hyperLift Inf Inf Inf Inf  
## hyperConfidence 0.000000000 0.000000000 0.000000000 0.000000000  
## fishersExactTest 1.000000000 1.000000000 1.000000000 1.000000000  
## improvement NA NA NA NA  
## chiSquared 0.092412235 0.284375800 0.054446944 0.543982995  
## cosine 0.183726085 0.338762411 0.298195490 0.439137284  
## conviction NA 5.519868591 2.312930486 24.539981826  
## gini 0.010104510 0.031094131 0.005953321 0.059480021  
## oddsRatio NA 17.097460792 3.162994012 91.897368421  
## phi 0.151996903 0.266634488 0.116669345 0.368776014  
## doc 0.684428112 0.582370424 0.199603247 0.695176439  
## RLD 1.000000000 0.818836267 0.567648053 0.959250174  
## imbalance 0.966244726 0.835635359 0.865346535 0.791608392  
## kulczynski 0.516877637 0.504080089 0.481845450 0.585363015  
## collectiveStrength 0.073620683 0.318245999 0.250182280 0.541830312  
## jaccard 0.033755274 0.128453039 0.101650165 0.197202797  
## kappa -10.754928009 -10.505323981 -5.599277186 -10.372813481  
## mutualInformation -0.085706014 -0.095592349 -0.009176430 -0.146620830  
## lambda 0.033755274 0.112517581 0.000000000 0.192686357  
## jMeasure NA 0.032127586 0.007247296 0.064781335  
## laplace 0.510671410 0.533153751 0.550410317 0.554997993  
## certainty 1.000000000 0.818836267 0.567648053 0.959250174  
## addedValue 0.676965016 0.554323506 0.183370180 0.649378809  
## ralambrodrainy 0.000000000 0.005906406 0.011358473 0.001817356  
## descriptiveConfirm 0.010904134 0.036347115 0.058609723 0.062244434  
## confirmedConfidence 1.000000000 0.754716981 0.720670391 0.944827586  
## sebag NA 7.153846154 6.160000000 35.250000000  
## counterexample 1.000000000 0.860215054 0.837662338 0.971631206  
## casualSupport 0.333939119 0.359382099 0.735574739 0.385279418  
## casualConfidence 1.000000000 0.964768596 0.960889168 0.991696979  
## leastContradiction 0.033755274 0.112517581 0.086577181 0.192686357  
## centeredConfidence 0.676965016 0.554323506 0.183370180 0.649378809  
## varyingLiaison 2.095639944 1.715985988 0.270870983 2.010242980  
## yuleQ NA 0.889487259 0.519576537 0.978470865  
## yuleY NA 0.610509057 0.280182181 0.811076786  
## lerman 0.248752264 0.428066679 0.127113620 0.586512549  
## implicationIndex -0.171833845 -0.295701202 -0.184013812 -0.405152921  
## [,5]  
## support 0.009086779  
## coverage 0.010449796  
## confidence 0.869565217  
## lift 2.691860821  
## leverage 0.005711129  
## hyperLift Inf  
## hyperConfidence 0.000000000  
## fishersExactTest 1.000000000  
## improvement NA  
## chiSquared 0.057695581  
## cosine 0.156398030  
## conviction 5.190065122  
## gini 0.006308532  
## oddsRatio 14.346357935  
## phi 0.120099522  
## doc 0.552301673  
## RLD 0.807324190  
## imbalance 0.963585434  
## kulczynski 0.448847306  
## collectiveStrength 0.060584452  
## jaccard 0.028011204  
## kappa -10.760842053  
## mutualInformation -0.058799872  
## lambda 0.023909986  
## jMeasure 0.006753483  
## laplace 0.507565643  
## certainty 0.807324190  
## addedValue 0.546530233  
## ralambrodrainy 0.001363017  
## descriptiveConfirm 0.007723762  
## confirmedConfidence 0.739130435  
## sebag 6.666666667  
## counterexample 0.850000000  
## casualSupport 0.330758746  
## casualConfidence 0.966336452  
## leastContradiction 0.023909986  
## centeredConfidence 0.546530233  
## varyingLiaison 1.691860821  
## yuleQ 0.869675919  
## yuleY 0.582259273  
## lerman 0.196595379  
## implicationIndex -0.135804753

# find redundant rules  
subset.matrix <- is.subset(sortedrulesdata, sortedrulesdata)  
subset.matrix[lower.tri(subset.matrix, diag=T)] <- NA

## Warning in if (diag) row(x) >= col(x) else row(x) > col(x): the condition  
## has length > 1 and only the first element will be used

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

## {Class=2nd,Sex=Female,Age=Child,Survived=Yes}   
## 2   
## {Class=1st,Sex=Female,Age=Adult,Survived=Yes}   
## 4   
## {Class=Crew,Sex=Female,Age=Adult,Survived=Yes}   
## 7   
## {Class=2nd,Sex=Female,Age=Adult,Survived=Yes}   
## 8

# remove redundant rules  
rules.pruned <- sortedrulesdata[!redundant]  
inspect(rules.pruned)

## lhs rhs support confidence  
## 1 {Class=2nd,Age=Child} => {Survived=Yes} 0.010904134 1.0000000   
## 4 {Class=1st,Sex=Female} => {Survived=Yes} 0.064061790 0.9724138   
## 2 {Class=2nd,Sex=Female} => {Survived=Yes} 0.042253521 0.8773585   
## 5 {Class=Crew,Sex=Female} => {Survived=Yes} 0.009086779 0.8695652   
## 9 {Class=2nd,Sex=Male,Age=Adult} => {Survived=No} 0.069968196 0.9166667   
## 3 {Class=2nd,Sex=Male} => {Survived=No} 0.069968196 0.8603352   
## 12 {Class=3rd,Sex=Male,Age=Adult} => {Survived=No} 0.175829169 0.8376623   
## 6 {Class=3rd,Sex=Male} => {Survived=No} 0.191731031 0.8274510   
## lift hyperConfidence  
## 1 3.095640 0   
## 4 3.010243 0   
## 2 2.715986 0   
## 5 2.691861 0   
## 9 1.354083 0   
## 3 1.270871 0   
## 12 1.237379 0   
## 6 1.222295 0

1. Continue exploring various interestingness measures - how to describe them the best, using perhaps the scatterplots measuring the effect of each field in the 2x2 tables. (e.g. how would symmetry look like, or other properties).

interestfunc <- function(f11, F1plus, Fplus1){  
 interest <- ((10000 \* f11)/(F1plus\*Fplus1))  
 return (interest)  
}  
  
convictionfunc <- function(F1plus, Fplus0, f10){  
 conviction <- ((F1plus\*Fplus0)/(10000 \* f10))  
 return (conviction)  
}  
  
interest <- interestfunc(randomGen[,c(1)], randomGen[,c(5)], randomGen[,c(7)])  
conviction <- convictionfunc(randomGen[,c(5)], randomGen[,c(8)], randomGen[,c(3)])  
  
  
  
interest <- interestfunc(top10[,c(1)], top10[,c(5)], top10[,c(7)])  
conviction <- convictionfunc(top10[,c(5)], top10[,c(8)], top10[,c(3)])  
  
  
  
top10 <- cbind(top10, interest)  
top10 <- cbind(top10, conviction)  
randomGen <- cbind(randomGen, interest)

## Warning in cbind(randomGen, interest): number of rows of result is not a  
## multiple of vector length (arg 2)

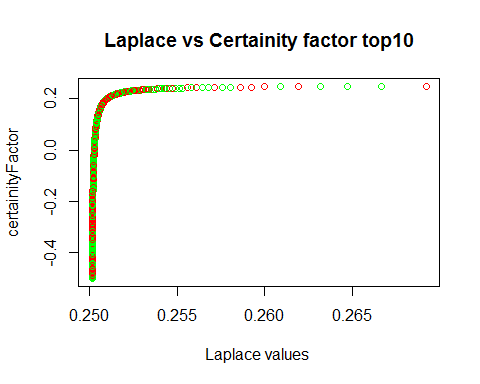
randomGen <- cbind(randomGen, conviction)

## Warning in cbind(randomGen, conviction): number of rows of result is not a  
## multiple of vector length (arg 2)

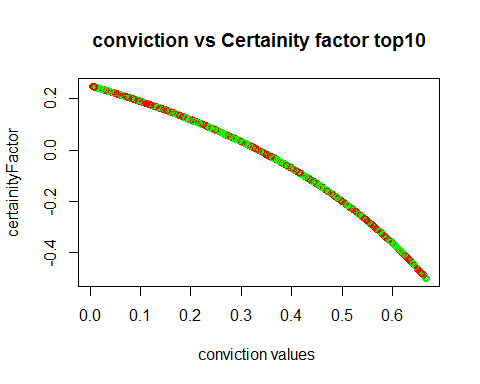
head(top10, 10)

## f11 f01 f10 f00 F1plus F0plus Fplus1 Fplus0 T oddRatio  
## [1,] 90 360 270 180 360 540 450 450 900 0.6666667  
## [2,] 934 3736 2802 1868 3736 5604 4670 4670 9340 0.6666667  
## [3,] 597 2388 1791 1194 2388 3582 2985 2985 5970 0.6666667  
## [4,] 897 3588 2691 1794 3588 5382 4485 4485 8970 0.6666667  
## [5,] 301 1204 903 602 1204 1806 1505 1505 3010 0.6666667  
## [6,] 663 2652 1989 1326 2652 3978 3315 3315 6630 0.6666667  
## [7,] 193 772 579 386 772 1158 965 965 1930 0.6666667  
## [8,] 158 632 474 316 632 948 790 790 1580 0.6666667  
## [9,] 436 1744 1308 872 1744 2616 2180 2180 4360 0.6666667  
## [10,] 828 3312 2484 1656 3312 4968 4140 4140 8280 0.6666667  
## jaccard laplace certainityFactor addedValue interest conviction  
## [1,] 0.125 0.2513812 0.21465969 0.2050 5.5555556 0.0600000  
## [2,] 0.125 0.2501338 -0.40712946 -0.2170 0.5353319 0.6226667  
## [3,] 0.125 0.2502092 -0.06913756 -0.0485 0.8375209 0.3980000  
## [4,] 0.125 0.2501393 -0.35992747 -0.1985 0.5574136 0.5980000  
## [5,] 0.125 0.2504146 0.11712772 0.0995 1.6611296 0.2006667  
## [6,] 0.125 0.2501884 -0.12191473 -0.0815 0.7541478 0.4420000  
## [7,] 0.125 0.2506460 0.16989485 0.1535 2.5906736 0.1286667  
## [8,] 0.125 0.2507886 0.18566775 0.1710 3.1645570 0.1053333  
## [9,] 0.125 0.2502864 0.04092072 0.0320 1.1467890 0.2906667  
## [10,] 0.125 0.2501509 -0.27986348 -0.1640 0.6038647 0.5520000

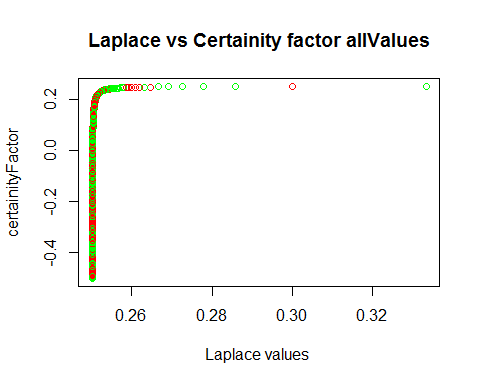
top10df <- as.data.frame(top10)  
randomGenNew <- as.data.frame(randomGen)  
  
library(lattice)  
  
plot(x = top10df$laplace, y=top10df$certainityFactor, main="Laplace vs Certainity factor top10", xlab="Laplace values ", ylab="certainityFactor", col=c('Green','Red'))



plot(x = top10df$conviction, y=top10df$certainityFactor, main="conviction vs Certainity factor top10", xlab="conviction values ", ylab="certainityFactor", col=c('Green','Red'))



plot(x = randomGenNew$laplace, y=randomGenNew$certainityFactor, main="Laplace vs Certainity factor allValues", xlab="Laplace values ", ylab="certainityFactor", col=c('Green','Red'))



plot(x = randomGenNew$conviction, y=randomGenNew$certainityFactor, main="conviction vs Certainity factor allValues", xlab="conviction values ", ylab="certainityFactor", col=c('Green','Red'))

