library(arules)

library(arulesViz)

library(datasets)

data(Groceries)

head(Groceries)

str(Groceries)

itemFrequencyPlot(Groceries,topN=10,type="absolute")

itemFrequencyPlot(Groceries,topN=15)

#######

rules = apriori(Groceries, parameter = list(supp = 0.001, conf = 0.9, maxlen=4))

rules

options(digits=2)

rules = sort(rules, by="lift", decreasing=TRUE)

inspect(rules[1:5])

rules = sort(rules, by="confidence", decreasing=TRUE)

inspect(rules[1:5])

#subset.matrix = is.subset(rules, rules)

#subset.matrix[lower.tri(subset.matrix, diag=TRUE)] = NA

#redundant = colSums(subset.matrix, na.rm=TRUE) >= 1

#rules.pruned = rules[!redundant]

#rules.pruned

#e = eclat(Groceries, parameter=list(support=0.05))

#e = sort(e, by="support", decreasing=TRUE)

#inspect(e)

table = crossTable(Groceries)

table[1:3, 1:3]

table["bottled beer","bottled beer"]

table["bottled beer","canned beer"]

###

beer.rules=apriori(data=Groceries, parameter=list(support=0.0015,confidence=0.3), appearance = list(default="lhs",rhs="bottled beer"))

beer.rules

beer.rules=sort(beer.rules, decreasing=TRUE,by="lift")

inspect(beer.rules)

cTable["bottled beer", "red/blush wine"]

cTable["red/blush wine", "red/blush wine"]

48/189

cTable["white wine", "white wine"]

cTable["bottled beer", "white wine"]

22/187

#rules=apriori(data=Groceries, parameter=list(supp=0.001,conf = 0.2),

#appearance = list(default="rhs",lhs="bottled beer"),

# control = list(verbose=FALSE))

#rules<-sort(rules, decreasing=TRUE,by="confidence")

#inspect(rules)

plot(beer.rules, method="graph", measure="lift", shading="confidence")

#######

ratings = c(3,5,5,5,1,1,5,2,5,1,1,5,3,5,1,5,4,2,4,3,4,2,1,4)

ratingMat = matrix(ratings, nrow=6)

rownames(ratingMat) = c("Homer","Marge","Bart","Lisa","Flanders","Me")

colnames(ratingMat) = c("Avengers","American Sniper","Les Miserable","Mad Max")

ratingMat

svd=svd(ratingMat)

svd

sum(svd$d)

var = sum(svd$d[1:2])

var

var/sum(svd$d)

f1 = function(x) {

score = 0

for(i in 1:n )

score <- score + svd$u[,i] %\*% t(svd$v[,i]) \* svd$d[i]

return(score)

}

n=4

f1(svd)

n=2

f1(svd)

library(psych)

pca = principal(ratingMat, nfactors=2, rotate="none")

pca

library(recommenderlab)

data(Jester5k)

Jester5k

as(Jester5k[10,], "list")

rowMeans(Jester5k[10,])

colMeans(Jester5k[,1])

hist(getRatings(Jester5k), breaks=100)

hist(getRatings(normalize(Jester5k)), breaks=100)

hist(rowCounts(Jester5k), breaks=50)

#############

set.seed(123)

e = evaluationScheme(Jester5k, method="split", train=0.8, given=15, goodRating=5)

recommenderRegistry$get\_entries(dataType = "realRatingMatrix")

ubcf = Recommender(getData(e, "train"), "UBCF")

ibcf = Recommender(getData(e, "train"), "IBCF")

svd = Recommender(getData(e, "train"), "SVD", param=list(treat\_na="0"))

popular = Recommender(getData(e, "train"), "POPULAR")

pca = Recommender(getData(e, "train"), "PCA")

random = Recommender(getData(e, "train"), "RANDOM")

ubcf\_pred = predict(ubcf, getData(e, "known"), type="ratings")

ibcf\_pred = predict(ibcf, getData(e, "known"), type="ratings")

svd\_pred = predict(svd, getData(e, "known"), type="ratings")

pop\_pred = predict(popular, getData(e, "known"), type="ratings")

pca\_pred = predict(pca, getData(e, "known"), type="ratings")

ran\_pred = predict(random, getData(e, "known"), type="ratings")

P1 = calcPredictionAccuracy(user\_pred, getData(e, "unknown"))

P1

P2 = calcPredictionAccuracy(item\_pred, getData(e, "unknown"))

P3 = calcPredictionAccuracy(svd\_pred, getData(e, "unknown"))

P4 = calcPredictionAccuracy(pop\_pred, getData(e, "unknown"))

P5 = calcPredictionAccuracy(pca\_pred, getData(e, "unknown"))

P6 = calcPredictionAccuracy(ran\_pred, getData(e, "unknown"))

error = rbind(P1,P2,P3,P4,P5,P6)

rownames(error) = c("UBCF", "IBCF", "SVD", "Popular", "PCA", "Random")

error

algorithms = list(

POPULAR = list(name = "POPULAR"),

UBCF = list(name = "UBCF"),

IBCF = list(name = "IBCF"))

evlist = evaluate(e, algorithms,n=c(5,10,15))

avg(evlist)

plot(evlist, legend="topleft", annotate=TRUE)

plot(evlist, "prec", legend="bottomright", annotate=TRUE)

R1 = Recommender(Jester5k, method="UBCF")

R1

recommend = predict(R1, Jester5k[1:2], n=5)

as(recommend, "list")

rating = predict(R1, Jester5k[300:309], type="ratings")

rating

as(rating, "matrix")[,71:73]

Jester.bin = binarize(Jester5k, minRating=5)

Jester.bin = Jester.bin[rowCounts(Jester.bin)>10]

Jester.bin

set.seed(456)

e.bin = evaluationScheme(Jester.bin, method="cross-validation", k=5, given=10)

algorithms.bin = list("random" = list(name="RANDOM", param=NULL),"popular" = list(name="POPULAR", param=NULL),"UBCF" = list(name="UBCF"))

results.bin = evaluate(e.bin, algorithms.bin, n=c(5,10,15))

plot(results.bin, legend="topleft")

plot(results.bin, "prec", legend="bottomright")