Homework 2

Naga Kartheek Peddisetty, 50538422

03/08/2024

Question 1) (Modified Exercise 14.4 in ESL) Cluster the demographic data (>data(marketing in ESL package)) of Table 14.1 using “generalized association rules” with a classification tree. (This data can also be found on the ESL website). Specifically, generate a reference sample the same size as the training set, by either (a) randomly permuting the columns independently, or by (b) sampling from a uniform distribution the values within each feature. Build a classification tree to the training sample (class 1) and the reference sample (class 0).

```
library(arules)
```

```
## Loading required package: Matrix
```

```
##
## Attaching package: 'arules'
```

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

```
library(rpart)
library(rpart.plot)

load("E:/Buffalo/files/marketing.RData")
head(marketing)
```

```
##   Income Sex Marital Age Edu Occupation Lived Dual_Income Household
## 1      9   2       1   5   4          5     5           3         3
## 2      9   1       1   5   5          5     5           3         5
## 3      9   2       1   3   5          1     5           2         3
## 4      1   2       5   1   2          6     5           1         4
## 5      1   2       5   1   2          6     3           1         4
## 6      8   1       1   6   4          8     5           3         2
##   Householdu18 Status Home_Type Ethnic Language
## 1            0      1         1      7       NA
## 2            2      1         1      7        1
## 3            1      2         3      7        1
## 4            2      3         1      7        1
## 5            2      3         1      7        1
## 6            0      1         1      7        1
```

```
dim(marketing)
```

```
## [1] 8993   14
```

```
str(marketing)
```

```
## 'data.frame':    8993 obs. of  14 variables:
##  $ Income      : int  9 9 9 1 1 8 1 6 2 4 ...
##  $ Sex         : int  2 1 2 2 2 1 1 1 1 1 ...
##  $ Marital     : int  1 1 1 5 5 1 5 3 1 1 ...
##  $ Age         : int  5 5 3 1 1 6 2 3 6 7 ...
##  $ Edu         : int  4 5 5 2 2 4 3 4 3 4 ...
##  $ Occupation  : int  5 5 1 6 6 8 9 3 8 8 ...
##  $ Lived       : int  5 5 5 5 3 5 4 5 5 4 ...
##  $ Dual_Income : int  3 3 2 1 1 3 1 1 3 3 ...
##  $ Household   : int  3 5 3 4 4 2 3 1 3 2 ...
##  $ Householdu18: int  0 2 1 2 2 0 1 0 0 0 ...
##  $ Status      : int  1 1 2 3 3 1 2 2 2 2 ...
##  $ Home_Type   : int  1 1 3 1 1 1 3 3 3 3 ...
##  $ Ethnic      : int  7 7 7 7 7 7 7 7 7 7 ...
##  $ Language    : int  NA 1 1 1 1 1 1 1 1 1 ...
```

```
summary(marketing)
```

```
##      Income           Sex           Marital           Age       
##  Min.   :1.000   Min.   :1.000   Min.   :1.000   Min.   :1.000  
##  1st Qu.:2.000   1st Qu.:1.000   1st Qu.:1.000   1st Qu.:2.000  
##  Median :5.000   Median :2.000   Median :3.000   Median :3.000  
##  Mean   :4.895   Mean   :1.547   Mean   :3.031   Mean   :3.415  
##  3rd Qu.:7.000   3rd Qu.:2.000   3rd Qu.:5.000   3rd Qu.:4.000  
##  Max.   :9.000   Max.   :2.000   Max.   :5.000   Max.   :7.000  
##                                  NA's   :160                    
##       Edu          Occupation        Lived        Dual_Income   
##  Min.   :1.000   Min.   :1.000   Min.   :1.000   Min.   :1.000  
##  1st Qu.:3.000   1st Qu.:1.000   1st Qu.:4.000   1st Qu.:1.000  
##  Median :4.000   Median :4.000   Median :5.000   Median :1.000  
##  Mean   :3.835   Mean   :3.788   Mean   :4.198   Mean   :1.545  
##  3rd Qu.:5.000   3rd Qu.:6.000   3rd Qu.:5.000   3rd Qu.:2.000  
##  Max.   :6.000   Max.   :9.000   Max.   :5.000   Max.   :3.000  
##  NA's   :86      NA's   :136     NA's   :913                    
##    Household      Householdu18        Status        Home_Type    
##  Min.   :1.000   Min.   :0.0000   Min.   :1.000   Min.   :1.000  
##  1st Qu.:2.000   1st Qu.:0.0000   1st Qu.:1.000   1st Qu.:1.000  
##  Median :3.000   Median :0.0000   Median :2.000   Median :1.000  
##  Mean   :2.852   Mean   :0.6669   Mean   :1.837   Mean   :1.856  
##  3rd Qu.:4.000   3rd Qu.:1.0000   3rd Qu.:2.000   3rd Qu.:3.000  
##  Max.   :9.000   Max.   :9.0000   Max.   :3.000   Max.   :5.000  
##  NA's   :375                      NA's   :240     NA's   :357    
##      Ethnic         Language    
##  Min.   :1.000   Min.   :1.000  
##  1st Qu.:5.000   1st Qu.:1.000  
##  Median :7.000   Median :1.000  
##  Mean   :5.956   Mean   :1.127  
##  3rd Qu.:7.000   3rd Qu.:1.000  
##  Max.   :8.000   Max.   :3.000  
##  NA's   :68      NA's   :359
```

```
colSums(marketing)
```

```
##       Income          Sex      Marital          Age          Edu   Occupation 
##        44021        13911           NA        30713           NA           NA 
##        Lived  Dual_Income    Household Householdu18       Status    Home_Type 
##           NA        13892           NA         5997           NA           NA 
##       Ethnic     Language 
##           NA           NA
```

```
set.seed(123)

market <- marketing

## Removing rows with NA values
market <- na.omit(market)
colSums(market)
```

```
##       Income          Sex      Marital          Age          Edu   Occupation
##        34738        10685        20616        23433        26770        25023
##        Lived  Dual_Income    Household Householdu18       Status    Home_Type
##        28972        10658        19675         4723        12578        12517
##       Ethnic     Language
##        41347         7677
```

```
dim(market)
```

```
## [1] 6876   14
```

```
## Creating reference data of the same size as market data with random uniform distribution "run if" function for each feature by taking min and max values.
reference_mar <- data.frame(lapply(market, function(x) round(runif(length(x), min(x), max(x)))))

dim(reference_mar)
```

```
## [1] 6876   14
```

```
head(reference_mar)
```

```
##   Income Sex Marital Age Edu Occupation Lived Dual_Income Household
## 1      3   1       5   7   1          2     3           3         2
## 2      7   2       3   5   3          9     3           2         6
## 3      4   2       4   7   2          2     3           3         7
## 4      8   1       5   4   3          2     3           2         4
## 5      9   1       4   7   2          7     4           2         8
## 6      1   1       5   3   2          5     4           1         6
##   Householdu18 Status Home_Type Ethnic Language
## 1            7      2         3      4        1
## 2            2      1         1      5        2
## 3            5      2         2      7        1
## 4            5      2         3      8        2
## 5            5      2         3      2        1
## 6            8      2         4      3        2
```

```
## Creating a new column Y and assigning value 1 to market data and 0 to reference data
market$Y <- 1
reference_mar$Y <- 0
head(market)
```

```
##   Income Sex Marital Age Edu Occupation Lived Dual_Income Household
## 2      9   1       1   5   5          5     5           3         5
## 3      9   2       1   3   5          1     5           2         3
## 4      1   2       5   1   2          6     5           1         4
## 5      1   2       5   1   2          6     3           1         4
## 6      8   1       1   6   4          8     5           3         2
## 7      1   1       5   2   3          9     4           1         3
##   Householdu18 Status Home_Type Ethnic Language Y
## 2            2      1         1      7        1 1
## 3            1      2         3      7        1 1
## 4            2      3         1      7        1 1
## 5            2      3         1      7        1 1
## 6            0      1         1      7        1 1
## 7            1      2         3      7        1 1
```

```
dim(market)
```

```
## [1] 6876   15
```

```
dim(reference_mar)
```

```
## [1] 6876   15
```

```
head(reference_mar)
```

```
##   Income Sex Marital Age Edu Occupation Lived Dual_Income Household
## 1      3   1       5   7   1          2     3           3         2
## 2      7   2       3   5   3          9     3           2         6
## 3      4   2       4   7   2          2     3           3         7
## 4      8   1       5   4   3          2     3           2         4
## 5      9   1       4   7   2          7     4           2         8
## 6      1   1       5   3   2          5     4           1         6
##   Householdu18 Status Home_Type Ethnic Language Y
## 1            7      2         3      4        1 0
## 2            2      1         1      5        2 0
## 3            5      2         2      7        1 0
## 4            5      2         3      8        2 0
## 5            5      2         3      2        1 0
## 6            8      2         4      3        2 0
```

```
## Combine market and reference data.
combined_data <- rbind(market,reference_mar)
dim(combined_data)
```

```
## [1] 13752    15
```

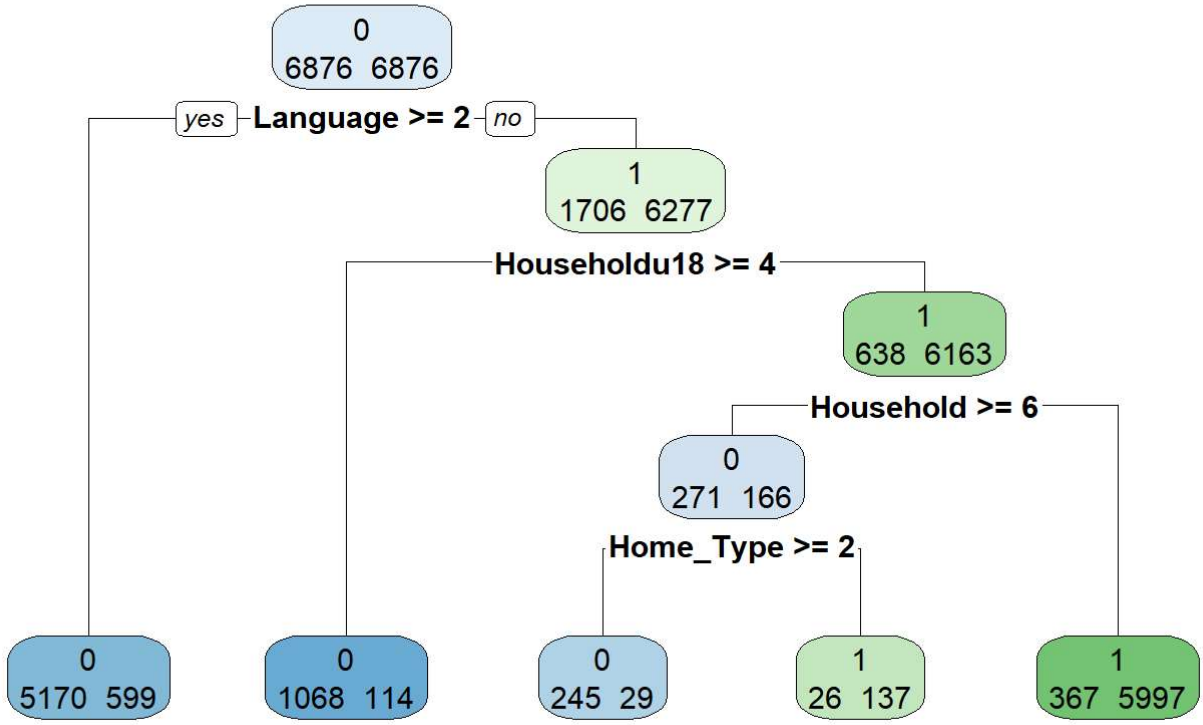
```
head(combined_data)
```

```
##   Income Sex Marital Age Edu Occupation Lived Dual_Income Household
## 2      9   1       1   5   5          5     5           3         5
## 3      9   2       1   3   5          1     5           2         3
## 4      1   2       5   1   2          6     5           1         4
## 5      1   2       5   1   2          6     3           1         4
## 6      8   1       1   6   4          8     5           3         2
## 7      1   1       5   2   3          9     4           1         3
##   Householdu18 Status Home_Type Ethnic Language Y
## 2            2      1         1      7        1 1
## 3            1      2         3      7        1 1
## 4            2      3         1      7        1 1
## 5            2      3         1      7        1 1
## 6            0      1         1      7        1 1
## 7            1      2         3      7        1 1
```

```
unique(combined_data$Y)
```

```
## [1] 1 0
```

```
## classification tree
tree <- rpart(Y ~ ., data = combined_data, method = "class")

## Visualize the tree
rpart.plot(tree, digits = 3, extra = 1, main = "Classification Tree")
```




```
## Extract the rules from classification tree
cat("Rules extracted from classification tree:")
```

```
## Rules extracted from classification tree:
```

```
print(rpart.rules(tree))
```

```
##     Y
##  0.10 when Language <  2 & Householdu18 >= 4
##  0.10 when Language >= 2
##  0.11 when Language <  2 & Householdu18 <  4 & Household >= 6 & Home_Type >= 2
##  0.84 when Language <  2 & Householdu18 <  4 & Household >= 6 & Home_Type <  2
##  0.94 when Language <  2 & Householdu18 <  4 & Household <  6
```

```
tree_rules <- c("   lhs                 =>                      rhs    ",
                "{Language < 2, Householdu18 >= 4} => {0}",
                "{Language >= 2} => {0}",
                "{Language < 2, Householdu18 < 4, Household >= 6, Home_Type >= 2} => {0}",
                "{Language < 2, Householdu18 < 4, Household >= 6, Home_Type < 2} => {1}",
                "{Language < 2, Householdu18 < 4, Household < 6} => {1}" )

print(tree_rules)
```

```
## [1] "   lhs                 =>                      rhs    "
## [2] "{Language < 2, Householdu18 >= 4} => {0}"
## [3] "{Language >= 2} => {0}"
## [4] "{Language < 2, Householdu18 < 4, Household >= 6, Home_Type >= 2} => {0}"
## [5] "{Language < 2, Householdu18 < 4, Household >= 6, Home_Type < 2} => {1}"
## [6] "{Language < 2, Householdu18 < 4, Household < 6} => {1}"
```

Describe the terminal nodes having highest estimated class 1 probability.

```
tree
```

```
## n= 13752
##
## node), split, n, loss, yval, (yprob)
##       * denotes terminal node
##
##  1) root 13752 6876 0 (0.50000000 0.50000000)
##    2) Language>=1.5 5769  599 0 (0.89616918 0.10383082) *
##    3) Language< 1.5 7983 1706 1 (0.21370412 0.78629588)
##      6) Householdu18>=3.5 1182  114 0 (0.90355330 0.09644670) *
##      7) Householdu18< 3.5 6801  638 1 (0.09380973 0.90619027)
##       14) Household>=5.5 437  166 0 (0.62013730 0.37986270)
##         28) Home_Type>=1.5 274   29 0 (0.89416058 0.10583942) *
##         29) Home_Type< 1.5 163   26 1 (0.15950920 0.84049080) *
##       15) Household< 5.5 6364  367 1 (0.05766813 0.94233187) *
```

As we can see above the terminal nodes are node 2, node 6, node 28, node 29 and node 15.

node 2, node 6 and node 28 are belong to class 0 according to the “yval”.

node 29 and node 15 are belong to class 1 according to the “yval”.

For example, Language>=1.5 5769 599 0 (0.89616918,0.10383082) in this the values inside the parenthesis are estimated probability for class 0 (89.61 %) and estimated probability for class 1 (10.38 %).

We need the terminal nodes which are having highest estimated class 1 probabilities:

node 2: 10.38%

node 6: 9.64%

node 28: 10.58%

node 29: 84.04%

node 15: 94.23%

Therefore, the terminal nodes having highest estimated class 1 probability are node 15 and node 29.

Question 2) Consider the California Housing Data from KAGGLE. You are going to apply association rules to this data. (https://www.kaggle.com/camnugent/california-housing-prices (https://www.kaggle.com/camnugent/california-housing-prices)).

```
library("arules")

dat <- read.csv("housing.csv")

head(dat)
```

```
##   longitude latitude housing_median_age total_rooms total_bedrooms population
## 1   -122.23    37.88                41         880            129        322
## 2   -122.22    37.86                21        7099           1106       2401
## 3   -122.24    37.85                52        1467            190        496
## 4   -122.25    37.85                52        1274            235        558
## 5   -122.25    37.85                52        1627            280        565
## 6   -122.25    37.85                52         919            213        413
##   households median_income median_house_value ocean_proximity
## 1        126        8.3252             452600        NEAR BAY
## 2       1138        8.3014             358500        NEAR BAY
## 3        177        7.2574             352100        NEAR BAY
## 4        219        5.6431             341300        NEAR BAY
## 5        259        3.8462             342200        NEAR BAY
## 6        193        4.0368             269700        NEAR BAY
```

```
dim(dat)
```

```
## [1] 20640    10
```

```
str(dat)
```

```
## 'data.frame':    20640 obs. of  10 variables:
##  $ longitude         : num  -122 -122 -122 -122 -122 ...
##  $ latitude          : num  37.9 37.9 37.9 37.9 37.9 ...
##  $ housing_median_age: num  41 21 52 52 52 52 52 52 42 52 ...
##  $ total_rooms       : num  880 7099 1467 1274 1627 ...
##  $ total_bedrooms    : num  129 1106 190 235 280 ...
##  $ population        : num  322 2401 496 558 565 ...
##  $ households        : num  126 1138 177 219 259 ...
##  $ median_income     : num  8.33 8.3 7.26 5.64 3.85 ...
##  $ median_house_value: num  452600 358500 352100 341300 342200 ...
##  $ ocean_proximity   : chr  "NEAR BAY" "NEAR BAY" "NEAR BAY" "NEAR BAY" ...
```

```
summary(dat)
```

```
##    longitude         latitude     housing_median_age  total_rooms
##  Min.   :-124.3   Min.   :32.54   Min.   : 1.00      Min.   :    2
##  1st Qu.:-121.8   1st Qu.:33.93   1st Qu.:18.00      1st Qu.: 1448
##  Median :-118.5   Median :34.26   Median :29.00      Median : 2127
##  Mean   :-119.6   Mean   :35.63   Mean   :28.64      Mean   : 2636
##  3rd Qu.:-118.0   3rd Qu.:37.71   3rd Qu.:37.00      3rd Qu.: 3148
##  Max.   :-114.3   Max.   :41.95   Max.   :52.00      Max.   :39320
##
##  total_bedrooms     population      households     median_income
##  Min.   :   1.0   Min.   :    3   Min.   :   1.0   Min.   : 0.4999
##  1st Qu.: 296.0   1st Qu.:  787   1st Qu.: 280.0   1st Qu.: 2.5634
##  Median : 435.0   Median : 1166   Median : 409.0   Median : 3.5348
##  Mean   : 537.9   Mean   : 1425   Mean   : 499.5   Mean   : 3.8707
##  3rd Qu.: 647.0   3rd Qu.: 1725   3rd Qu.: 605.0   3rd Qu.: 4.7432
##  Max.   :6445.0   Max.   :35682   Max.   :6082.0   Max.   :15.0001
##  NA's   :207
##  median_house_value ocean_proximity
##  Min.   : 14999     Length:20640
##  1st Qu.:119600     Class :character
##  Median :179700     Mode  :character
##  Mean   :206856
##  3rd Qu.:264725
##  Max.   :500001
##
```

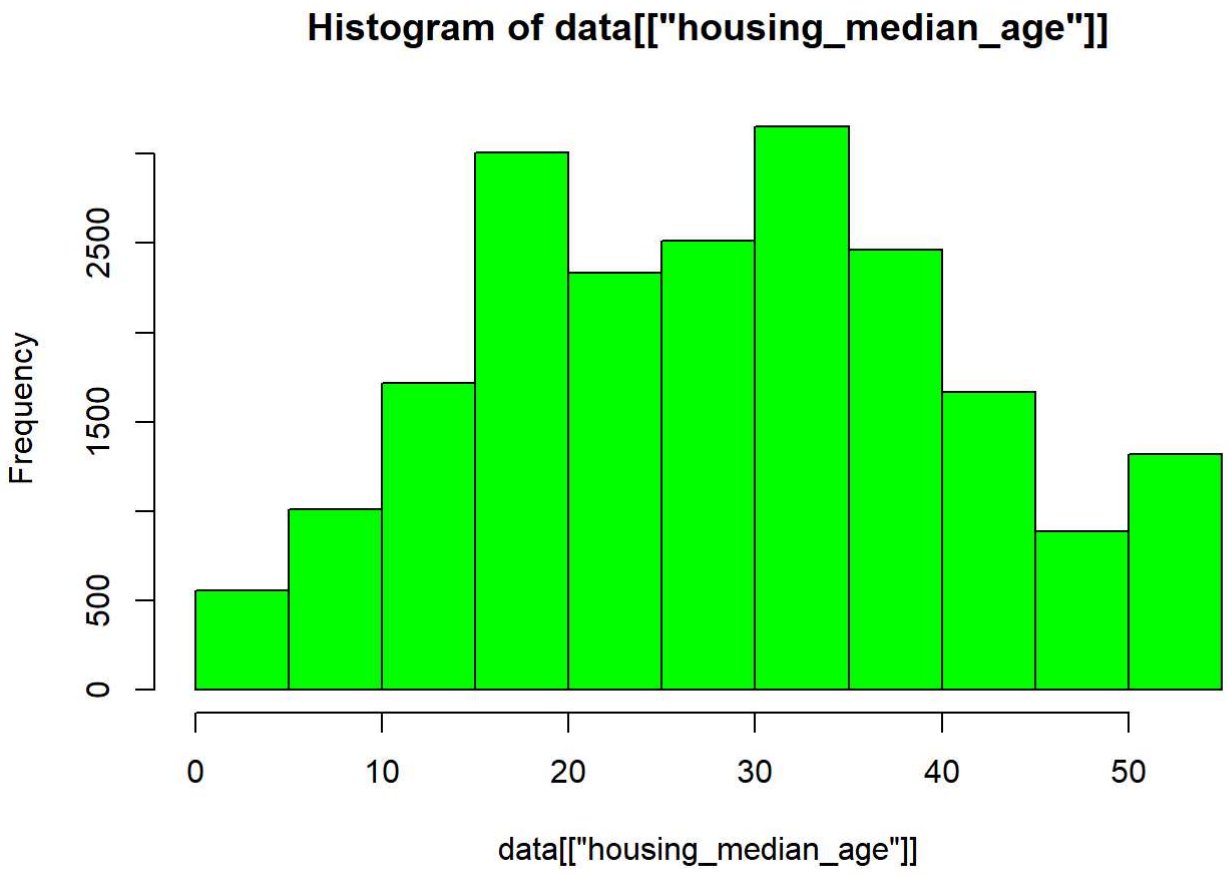
```
dat$ocean_proximity <- as.factor(dat$ocean_proximity)

### Variable selection

data <- dat[,c("housing_median_age","population","households","median_income", "median_house_val
ue", "ocean_proximity")]
head(data)
```

```
##   housing_median_age population households median_income median_house_value
## 1                 41        322        126        8.3252             452600
## 2                 21       2401       1138        8.3014             358500
## 3                 52        496        177        7.2574             352100
## 4                 52        558        219        5.6431             341300
## 5                 52        565        259        3.8462             342200
## 6                 52        413        193        4.0368             269700
##   ocean_proximity
## 1        NEAR BAY
## 2        NEAR BAY
## 3        NEAR BAY
## 4        NEAR BAY
## 5        NEAR BAY
## 6        NEAR BAY
```

```
dim(data)
```

```
## [1] 20640     6
```

```
hist(data[["housing_median_age"]], col = 'green')
```



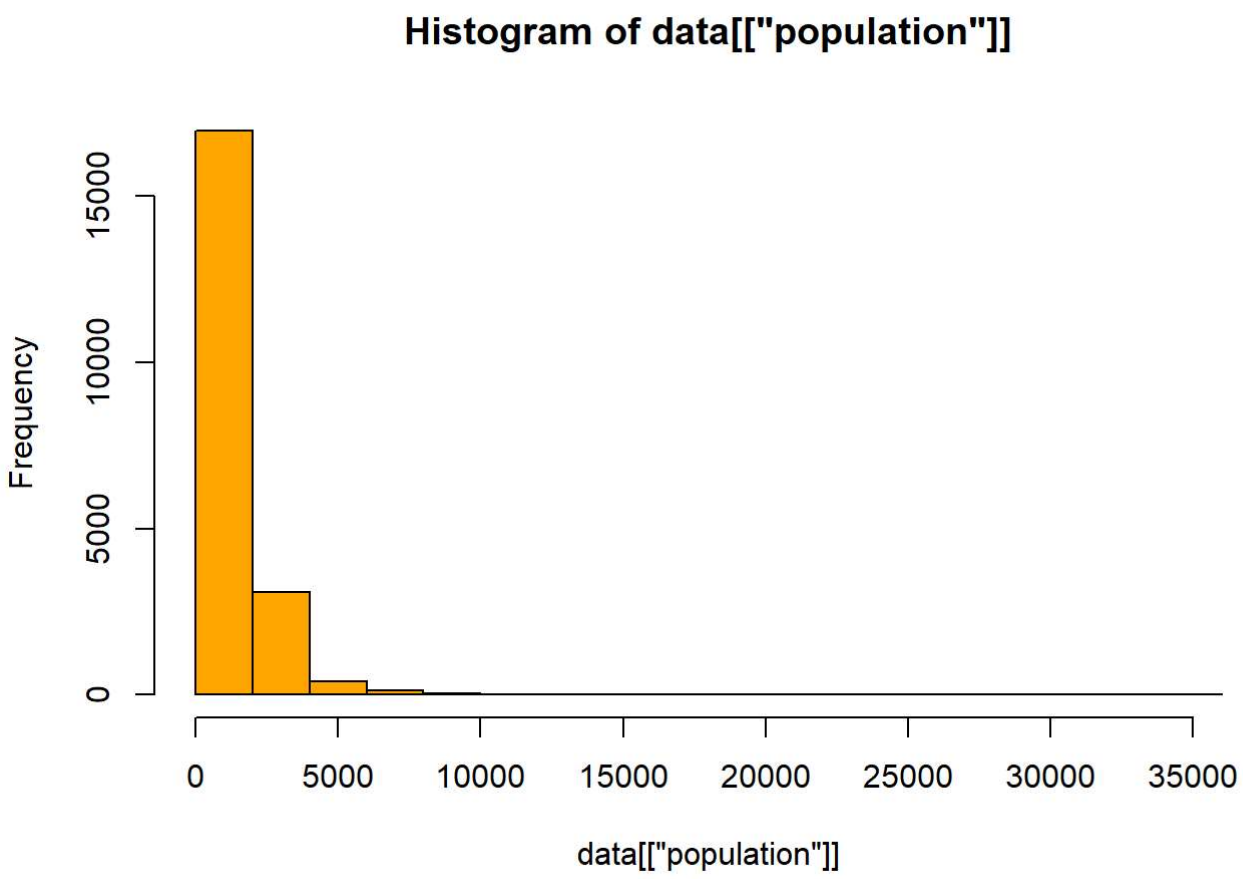

```
min(data[["housing_median_age"]])
```

```
## [1] 1
```

```
max(data[["housing_median_age"]])
```

```
## [1] 52
```

```
hist(data[["population"]], col = 'orange')
```



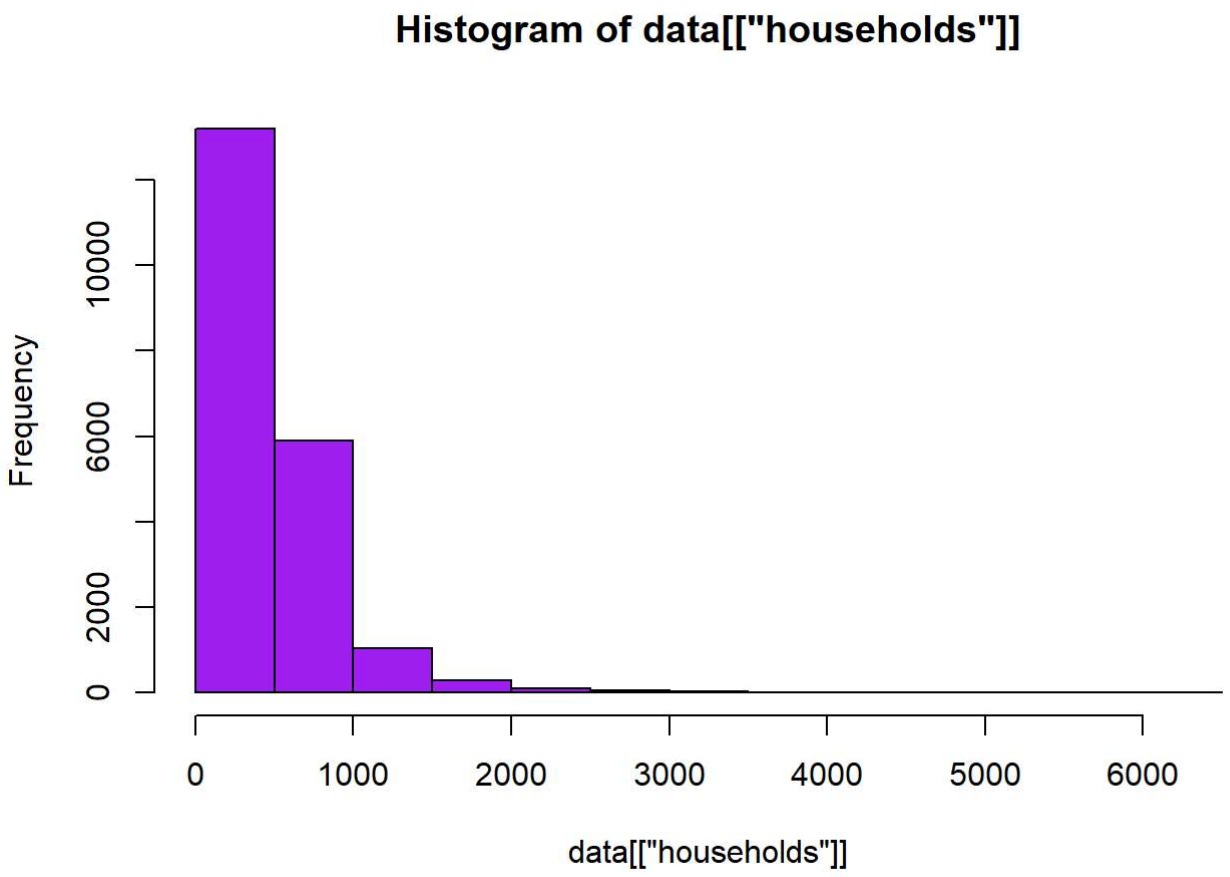

```
min(data[["population"]])
```

```
## [1] 3
```

```
max(data[["population"]])
```

```
## [1] 35682
```

```
hist(data[["households"]], col = 'purple')
```



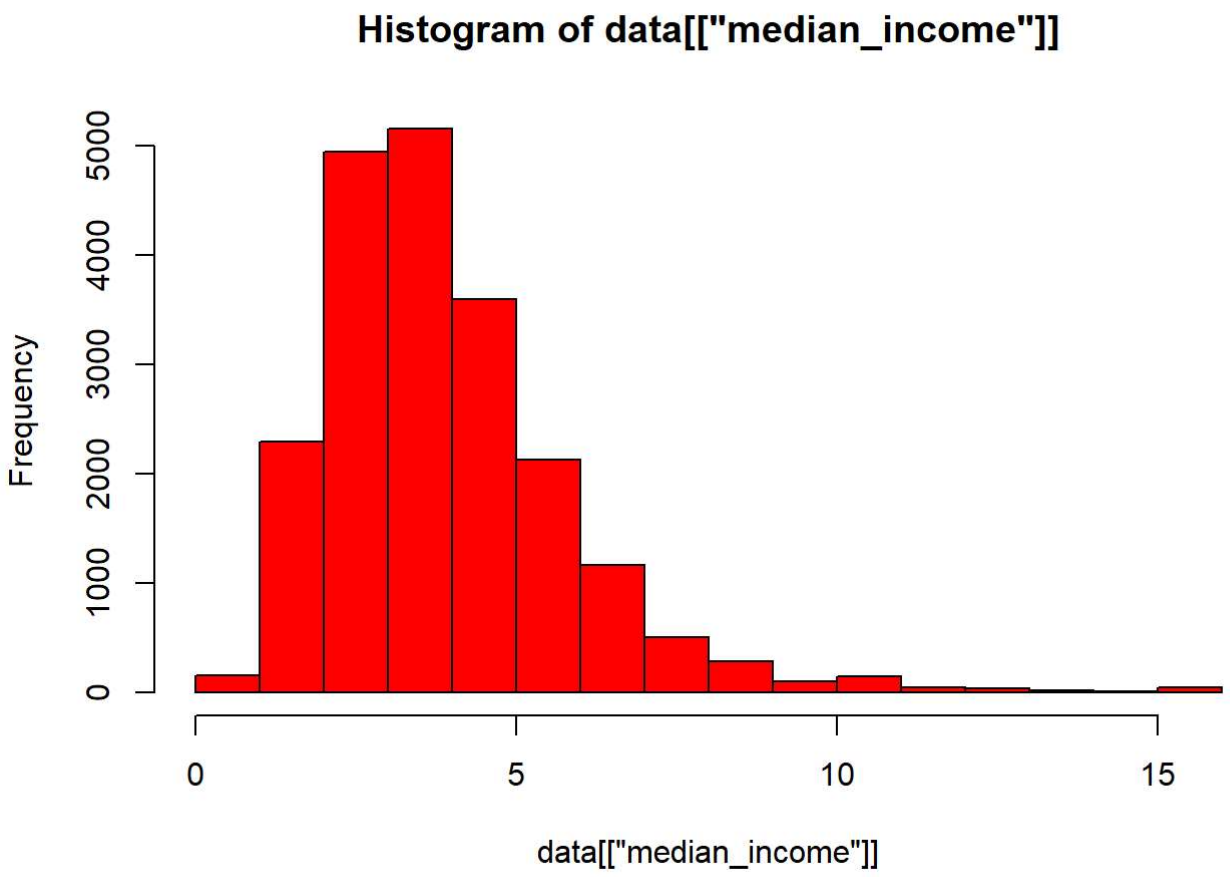

```
min(data[["households"]])
```

```
## [1] 1
```

```
max(data[["households"]])
```

```
## [1] 6082
```

```
hist(data[["median_income"]], col = 'red')
```



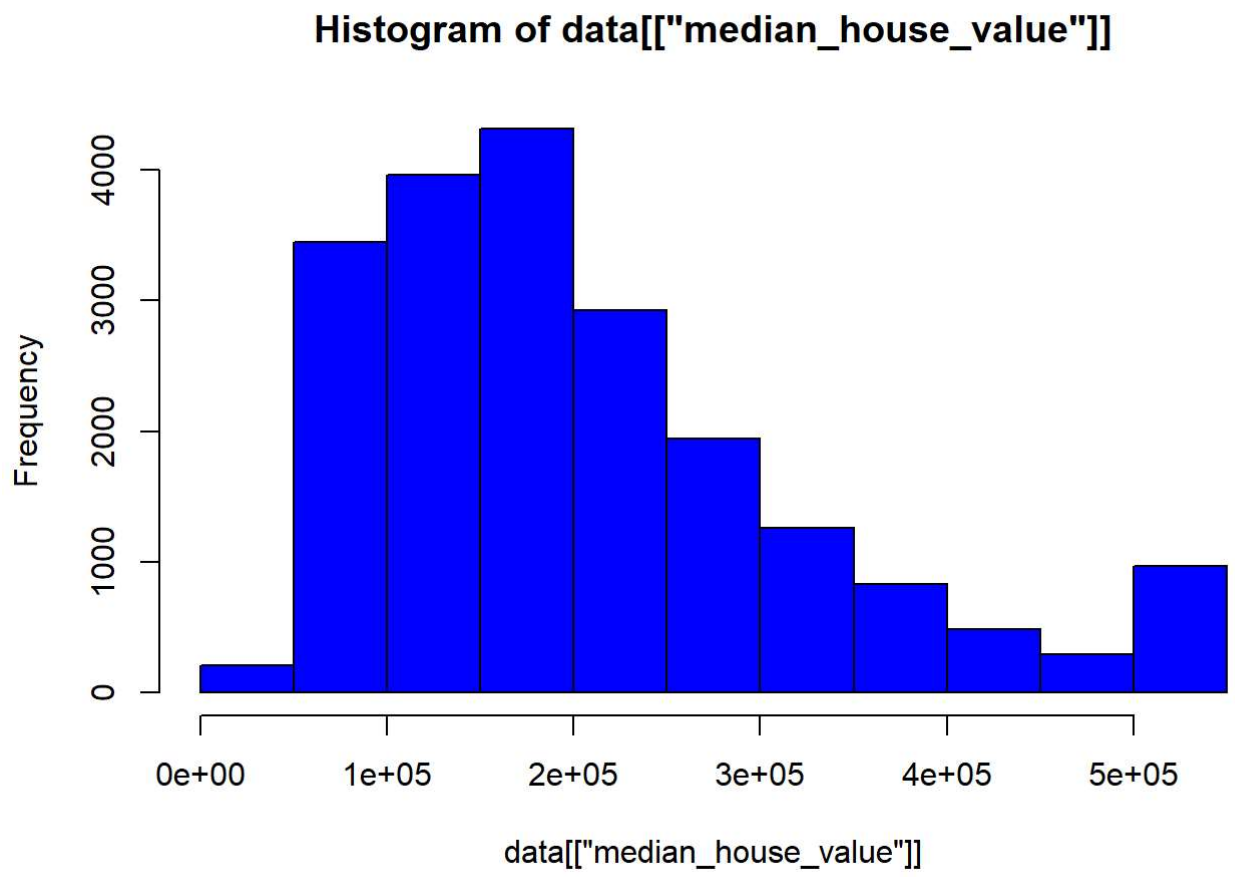

```
min(data[["median_income"]])
```

```
## [1] 0.4999
```

```
max(data[["median_income"]])
```

```
## [1] 15.0001
```

```
hist(data[["median_house_value"]], col = 'blue')
```




```
min(data[["median_house_value"]])
```

```
## [1] 14999
```

```
max(data[["median_house_value"]])
```

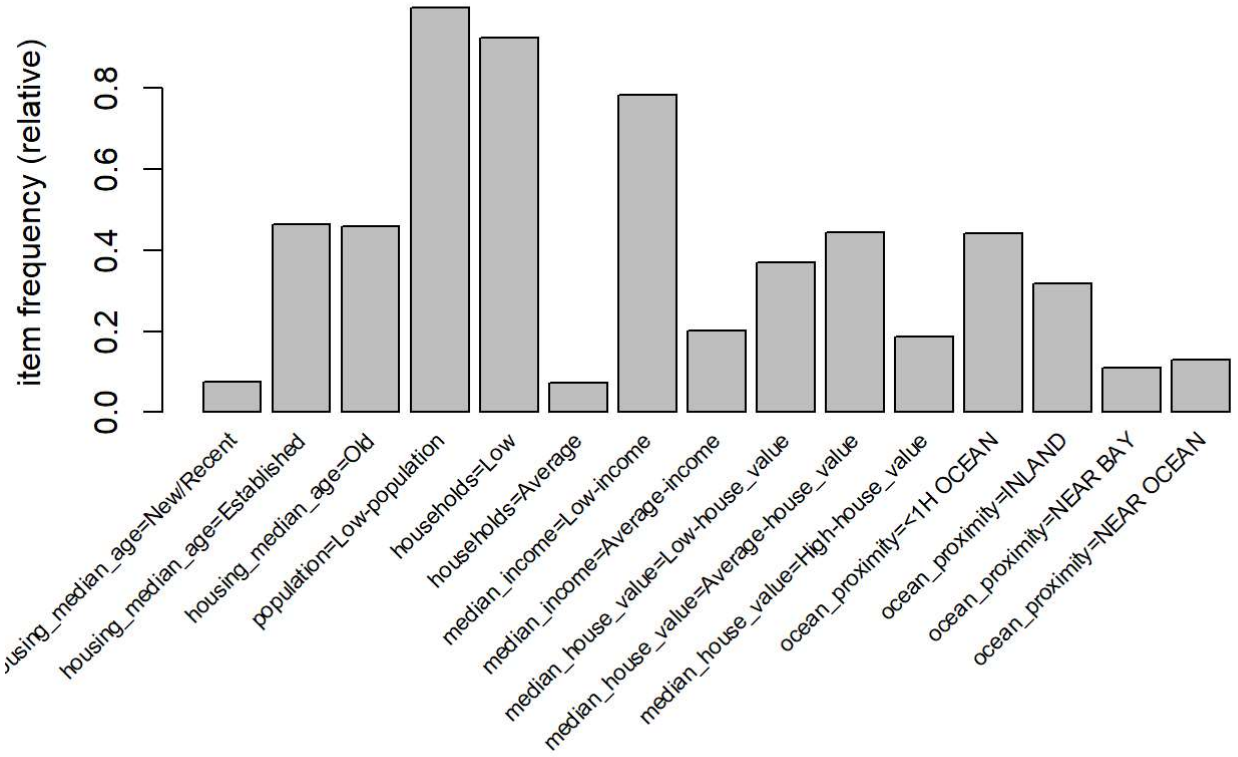
```
## [1] 500001
```

a) Read in the data and transfer to a binary incidence matrix. Visualize this matrix.

```
### Dealing with ordered variables

data[["housing_median_age"]] <- ordered(cut(data[["housing_median_age"]], c(1, 10, 30, 52)), lab
els = c("New/Recent", "Established", "Old"))

data[["population"]] <- ordered(cut(data[["population"]], c(1, 10000, 20000, 35700)), labels = c
("Low-population", "Average-population", "High-population"))

data[["households"]] <- ordered(cut(data[["households"]], c(0, 1000, 3000, 6100)), labels = c("L
ow", "Average", "High"))

data[["median_income"]] <- ordered(cut(data[["median_income"]], c(0, 5, 10, 15)), labels = c("Lo
w-income", "Average-income", "High-income"))

data[["median_house_value"]] <- ordered(cut(data[["median_house_value"]], c(0, 150000, 300000, 5
00001)), labels = c("Low-house_value", "Average-house_value", "High-house_value"))

#Converting to a binary incidence matrix
data1 <- as(data, "transactions")
summary(data1)
```

```
## transactions as itemMatrix in sparse format with
##  20640 rows (elements/itemsets/transactions) and
##  20 columns (items) and a density of 0.2998716
##
## most frequent items:
##      population=Low-population                 households=Low
##                          20617                          19098
##       median_income=Low-income housing_median_age=Established
##                          16151                           9576
##         housing_median_age=Old                         (Other)
##                           9495                          48850
##
## element (itemset/transaction) length distribution:
## sizes
##     5     6
##    53 20587
##
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   5.000   6.000   6.000   5.997   6.000   6.000
##
## includes extended item information - examples:
##                           labels          variables      levels
## 1  housing_median_age=New/Recent housing_median_age  New/Recent
## 2 housing_median_age=Established housing_median_age Established
## 3         housing_median_age=Old housing_median_age         Old
##
## includes extended transaction information - examples:
##   transactionID
## 1             1
## 2             2
## 3             3
```

```
# Visualize the binary incidence matrix using item frequency plot
itemFrequencyPlot(data1, support = 0.05, cex.names = 0.7)
```




```
# Apply the apriori algorithm
rules  <- apriori(data1, parameter = list(support = 0.001 , confidence = 0.6, minlen = 2, maxlen
= 5))
```

```
## Apriori
##
## Parameter specification:
##  confidence minval smax arem  aval originalSupport maxtime support minlen
##         0.6    0.1    1 none FALSE            TRUE       5   0.001      2
##  maxlen target  ext
##       5  rules TRUE
##
## Algorithmic control:
##  filter tree heap memopt load sort verbose
##     0.1 TRUE TRUE  FALSE TRUE    2    TRUE
##
## Absolute minimum support count: 20
##
## set item appearances ...[0 item(s)] done [0.00s].
## set transactions ...[20 item(s), 20640 transaction(s)] done [0.00s].
## sorting and recoding items ... [18 item(s)] done [0.00s].
## creating transaction tree ... done [0.00s].
## checking subsets of size 1 2 3 4 5
```

```
## Warning in apriori(data1, parameter = list(support = 0.001, confidence = 0.6, :
## Mining stopped (maxlen reached). Only patterns up to a length of 5 returned!
```

```
##  done [0.00s].
## writing ... [1654 rule(s)] done [0.00s].
## creating S4 object  ... done [0.00s].
```

```
summary(rules)
```

```
## set of 1654 rules
##
## rule length distribution (lhs + rhs):sizes
##   2   3   4   5
##  48 272 660 674
##
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   2.000   4.000   4.000   4.185   5.000   5.000
##
## summary of quality measures:
##     support           confidence        coverage             lift
##  Min.   :0.001017   Min.   :0.6000   Min.   :0.001017   Min.   :0.6478
##  1st Qu.:0.004651   1st Qu.:0.7531   1st Qu.:0.005475   1st Qu.:1.0002
##  Median :0.017660   Median :0.9353   Median :0.021536   Median :1.0011
##  Mean   :0.057294   Mean   :0.8758   Mean   :0.065566   Mean   :1.3516
##  3rd Qu.:0.059787   3rd Qu.:1.0000   3rd Qu.:0.071221   3rd Qu.:1.3356
##  Max.   :0.925291   Max.   :1.0000   Max.   :0.998886   Max.   :9.0121
##      count
##  Min.   :   21.0
##  1st Qu.:   96.0
##  Median :  364.5
##  Mean   : 1182.6
##  3rd Qu.: 1234.0
##  Max.   :19098.0
##
## mining info:
##   data ntransactions support confidence
##  data1         20640   0.001        0.6
##
call
##  apriori(data = data1, parameter = list(support = 0.001, confidence = 0.6, minlen = 2, maxlen
= 5))
```

b) What are the top three high lift rules?

```
inspect(head(sort(rules, by = "lift"), n = 3))
```

```
##     lhs                                          rhs                                 support
confidence    coverage     lift count
## [1] {households=Average,
##      median_income=Average-income,
##      median_house_value=Average-house_value,
##      ocean_proximity=INLAND}                  => {housing_median_age=New/Recent} 0.001986434
0.6833333 0.002906977 9.012141    41
## [2] {households=Average,
##      median_income=Average-income,
##      ocean_proximity=INLAND}                  => {housing_median_age=New/Recent} 0.002470930
0.6455696 0.003827519 8.514094    51
## [3] {population=Low-population,
##      households=Average,
##      median_income=Average-income,
##      ocean_proximity=INLAND}                  => {housing_median_age=New/Recent} 0.002470930
0.6455696 0.003827519 8.514094    51
```

c) What are the top 4 rules according to confidence?

```
inspect(head(sort(rules, by = "confidence"), n = 4))
```

```
##     lhs                            rhs                            support confidence   covera
ge     lift count
## [1] {median_income=High-income} => {population=Low-population} 0.01254845          1 0.012548
45 1.001116   259
## [2] {ocean_proximity=NEAR BAY}  => {population=Low-population} 0.11094961          1 0.110949
61 1.001116  2290
## [3] {housing_median_age=Old}    => {population=Low-population} 0.46002907          1 0.460029
07 1.001116  9495
## [4] {households=Low}            => {population=Low-population} 0.92529070          1 0.925290
70 1.001116 19098
```

d) A person comes to you and wants to purchase an average priced home as close to the ocean as possible. What can you recommend and/or what can the expect? Use association rules to guide you. -2 marks Incorrect code and no explanation.

```
rules_close_ocean <- subset(rules, subset = lhs %in% "ocean_proximity=<1H OCEAN" & rhs %in% "med
ian_house_value=Average-house_value")

cat("Top five association rules sorted by lift for purchasing an average-priced home close to th
e ocean:")
```

```
## Top five association rules sorted by lift for purchasing an average-priced home close to the
ocean:
```

```
inspect(head(sort(rules_close_ocean, by = "lift"), n = 5))
```

```
##     lhs                                  rhs                                                 support
confidence    coverage     lift count
## [1] {housing_median_age=New/Recent,
##      households=Average,
##      median_income=Low-income,
##      ocean_proximity=<1H OCEAN}       => {median_house_value=Average-house_value} 0.002519380
0.8125000 0.003100775 1.826002    52
## [2] {housing_median_age=New/Recent,
##      median_income=Low-income,
##      ocean_proximity=<1H OCEAN}       => {median_house_value=Average-house_value} 0.009738372
0.7153025 0.013614341 1.607561   201
## [3] {housing_median_age=New/Recent,
##      population=Low-population,
##      median_income=Low-income,
##      ocean_proximity=<1H OCEAN}       => {median_house_value=Average-house_value} 0.009689922
0.7142857 0.013565891 1.605276   200
## [4] {housing_median_age=Established,
##      households=Low,
##      median_income=Low-income,
##      ocean_proximity=<1H OCEAN}       => {median_house_value=Average-house_value} 0.075823643
0.7002237 0.108284884 1.573674  1565
## [5] {housing_median_age=Established,
##      population=Low-population,
##      median_income=Low-income,
##      ocean_proximity=<1H OCEAN}       => {median_house_value=Average-house_value} 0.089486434
0.7001516 0.127810078 1.573512  1847
```

e) What characteristics in the data associate with low population areas?

```
low_pop_rule <- subset(rules, subset = rhs %in% "population=Low-population")

cat("Top association rules for low population areas:")
```

```
## Top association rules for low population areas:
```

```
inspect(head(low_pop_rule, n = 3))
```

```
##     lhs                                rhs                             support confidence   co
verage      lift count
## [1] {median_income=High-income}     => {population=Low-population} 0.01254845  1.0000000 0.01
254845 1.0011156   259
## [2] {households=Average}            => {population=Low-population} 0.07223837  0.9979920 0.07
238372 0.9991053  1491
## [3] {housing_median_age=New/Recent} => {population=Low-population} 0.07509690  0.9904153 0.07
582364 0.9915202  1550
```

```
cat("Top association rules for low population areas sorted by lift:")
```

```
## Top association rules for low population areas sorted by lift:
```

```
inspect(head(sort(low_pop_rule, by = "lift"), n = 3))
```

```
##     lhs                             rhs                             support confidence   covera
ge     lift count
## [1] {median_income=High-income} => {population=Low-population} 0.01254845          1 0.012548
45 1.001116   259
## [2] {ocean_proximity=NEAR BAY}  => {population=Low-population} 0.11094961          1 0.110949
61 1.001116  2290
## [3] {housing_median_age=Old}    => {population=Low-population} 0.46002907          1 0.460029
07 1.001116  9495
```

The association rules above shows that high-income neighborhoods, areas with an average number of households, and newly established or old housing areas are highly associated with low population densities. This shows that exclusivity, stable housing markets, and recent/old constructions help to create low-population zones.

Question 3) Consider the MovieLense data that is available in the recommenderlab package data (MovieLense) ? MovieLense The data was collected through the MovieLens web site during a seven month, and contains about 100,000 ratings (1-5) from 943 users on 1664 movies. See the help file on the data to understand how to best manipulate the object.

```
library(recommenderlab)
```

```
## Loading required package: proxy
```

```
##
## Attaching package: 'proxy'
```

```
## The following object is masked from 'package:Matrix':
##
##     as.matrix
```

```
## The following objects are masked from 'package:stats':
##
##     as.dist, dist
```

```
## The following object is masked from 'package:base':
##
##     as.matrix
```

```
## Registered S3 methods overwritten by 'registry':
##   method               from
##   print.registry_field proxy
##   print.registry_entry proxy
```

```
data("MovieLense")

movie_lense <- MovieLense

dim(movie_lense)
```

```
## [1]  943 1664
```

```
paste("Number of users = ", dim(movie_lense)[1])
```

```
## [1] "Number of users =  943"
```

```
paste("Number of movies = ", dim(movie_lense)[2])
```

```
## [1] "Number of movies =  1664"
```

```
m_lense_df <- data.frame(as(movie_lense, "matrix"))
#head(m_lense_df)

# To get NaN count of every user
# Number of movies that user "i" did not give any rating
rowSums(is.na(m_lense_df))
```

```
##    1    2    3    4    5    6    7    8    9   10   11   12   13   14   15   16
## 1393 1603 1613 1641 1489 1456 1264 1605 1642 1480 1484 1613 1034 1566 1561 1524
##   17   18   19   20   21   22   23   24   25   26   27   28   29   30   31   32
## 1636 1387 1645 1616 1487 1537 1513 1596 1586 1557 1639 1586 1632 1621 1630 1624
##   33   34   35   36   37   38   39   40   41   42   43   44   45   46   47   48
## 1641 1644 1641 1645 1607 1543 1642 1631 1612 1481 1443 1513 1616 1637 1641 1599
##   49   50   51   52   53   54   55   56   57   58   59   60   61   62   63   64
## 1451 1641 1641 1608 1636 1600 1643 1477 1558 1511 1283 1456 1643 1432 1572 1464
##   65   66   67   68   69   70   71   72   73   74   75   76   77   78   79   80
## 1584 1626 1634 1630 1600 1533 1626 1527 1599 1626 1585 1582 1593 1643 1611 1636
##   81   82   83   84   85   86   87   88   89   90   91   92   93   94   95   96
## 1606 1496 1509 1596 1377 1642 1454 1644 1589 1367 1566 1278 1644 1266 1386 1608
##   97   98   99  100  101  102  103  104  105  106  107  108  109  110  111  112
## 1602 1637 1530 1608 1597 1448 1635 1554 1642 1600 1643 1631 1430 1531 1641 1619
##  113  114  115  116  117  118  119  120  121  122  123  124  125  126  127  128
## 1615 1616 1572 1524 1579 1593 1485 1638 1590 1603 1610 1640 1482 1621 1642 1481
##  129  130  131  132  133  134  135  136  137  138  139  140  141  142  143  144
## 1636 1313 1634 1642 1638 1639 1609 1630 1618 1613 1642 1645 1557 1632 1644 1460
##  145  146  147  148  149  150  151  152  153  154  155  156  157  158  159  160
## 1351 1635 1644 1599 1630 1633 1357 1558 1641 1613 1642 1627 1614 1491 1558 1544
##  161  162  163  164  165  166  167  168  169  170  171  172  173  174  175  176
## 1606 1622 1641 1601 1635 1644 1595 1595 1627 1644 1638 1637 1624 1488 1626 1604
##  177  178  179  180  181  182  183  184  185  186  187  188  189  190  191  192
## 1550 1394 1623 1601 1233 1636 1611 1413 1616 1573 1608 1552 1479 1605 1637 1629
##  193  194  195  196  197  198  199  200  201  202  203  204  205  206  207  208
## 1544 1359 1569 1625 1546 1483 1625 1448 1281 1644 1621 1624 1643 1600 1434 1631
##  209  210  211  212  213  214  215  216  217  218  219  220  221  222  223  224
## 1631 1532 1628 1640 1538 1537 1571 1533 1588 1611 1637 1644 1519 1279 1559 1523
##  225  226  227  228  229  230  231  232  233  234  235  236  237  238  239  240
## 1637 1614 1606 1643 1635 1532 1643 1572 1554 1187 1571 1540 1616 1635 1506 1640
##  241  242  243  244  245  246  247  248  249  250  251  252  253  254  255  256
## 1642 1645 1584 1427 1642 1469 1638 1608 1503 1542 1587 1644 1567 1505 1581 1457
##  257  258  259  260  261  262  263  264  265  266  267  268  269  270  271  272
## 1607 1641 1618 1641 1637 1504 1541 1544 1618 1642 1479 1337 1342 1527 1386 1613
##  273  274  275  276  277  278  279  280  281  282  283  284  285  286  287  288
## 1644 1593 1569 1149 1610 1641 1230 1407 1638 1643 1611 1619 1632 1379 1603 1590
##  289  290  291  292  293  294  295  296  297  298  299  300  301  302  303  304
## 1637 1514 1369 1540 1277 1516 1468 1519 1473 1537 1385 1645 1389 1645 1182 1638
##  305  306  307  308  309  310  311  312  313  314  315  316  317  318  319  320
## 1444 1633 1552 1267 1645 1643 1371 1441 1403 1420 1578 1589 1643 1491 1642 1510
##  321  322  323  324  325  326  327  328  329  330  331  332  333  334  335  336
## 1539 1614 1576 1599 1524 1479 1382 1381 1600 1517 1595 1481 1638 1335 1642 1534
##  337  338  339  340  341  342  343  344  345  346  347  348  349  350  351  352
## 1630 1589 1410 1620 1644 1463 1428 1476 1434 1471 1466 1606 1623 1614 1620 1622
##  353  354  355  356  357  358  359  360  361  362  363  364  365  366  367  368
## 1639 1434 1638 1640 1588 1622 1638 1563 1543 1640 1353 1645 1607 1631 1608 1620
##  369  370  371  372  373  374  375  376  377  378  379  380  381  382  383  384
## 1643 1589 1613 1598 1431 1390 1636 1635 1632 1290 1468 1503 1539 1611 1594 1642
##  385  386  387  388  389  390  391  392  393  394  395  396  397  398  399  400
## 1385 1641 1361 1613 1393 1633 1543 1555 1218 1515 1607 1610 1564 1492 1346 1642
##  401  402  403  404  405  406  407  408  409  410  411  412  413  414  415  416
## 1511 1596 1614 1621  929 1323 1438 1637 1463 1637 1606 1613 1615 1639 1640 1177
```

```
##  417  418  419  420  421  422  423  424  425  426  427  428  429  430  431  432
## 1300 1644 1632 1628 1602 1568 1601 1620 1461 1563 1637 1611 1251 1600 1644 1602
##  433  434  435  436  437  438  439  440  441  442  443  444  445  446  447  448
## 1625 1620 1286 1520 1401 1631 1634 1613 1644 1522 1640 1640 1531 1630 1525 1628
##  449  450  451  452  453  454  455  456  457  458  459  460  461  462  463  464
## 1590 1126 1570 1458 1509 1429 1475 1448 1388 1481 1546 1597 1641 1631 1533 1611
##  465  466  467  468  469  470  471  472  473  474  475  476  477  478  479  480
## 1584 1565 1621 1521 1621 1608 1633 1401 1631 1337 1645 1582 1629 1551 1465 1604
##  481  482  483  484  485  486  487  488  489  490  491  492  493  494  495  496
## 1609 1639 1605 1525 1639 1477 1433 1511 1559 1607 1631 1609 1530 1617 1451 1536
##  497  498  499  500  501  502  503  504  505  506  507  508  509  510  511  512
## 1386 1516 1562 1440 1591 1628 1506 1413 1552 1422 1606 1577 1633 1635 1640 1643
##  513  514  515  516  517  518  519  520  521  522  523  524  525  526  527  528
## 1642 1472 1624 1643 1627 1591 1619 1641 1515 1634 1564 1359 1610 1596 1527 1611
##  529  530  531  532  533  534  535  536  537  538  539  540  541  542  543  544
## 1622 1619 1634 1394 1405 1584 1447 1502 1177 1583 1609 1601 1532 1529 1468 1633
##  545  546  547  548  549  550  551  552  553  554  555  556  557  558  559  560
## 1503 1605 1642 1509 1639 1624 1331 1580 1564 1550 1612 1621 1613 1644 1588 1564
##  561  562  563  564  565  566  567  568  569  570  571  572  573  574  575  576
## 1308 1592 1634 1630 1629 1515 1511 1584 1595 1644 1644 1644 1613 1620 1638 1628
##  577  578  579  580  581  582  583  584  585  586  587  588  589  590  591  592
## 1476 1641 1592 1618 1636 1603 1638 1640 1584 1498 1571 1440 1622 1616 1580 1308
##  593  594  595  596  597  598  599  600  601  602  603  604  605  606  607  608
## 1508 1639 1569 1644 1623 1639 1617 1575 1516 1635 1617 1638 1574 1428 1627 1504
##  609  610  611  612  613  614  615  616  617  618  619  620  621  622  623  624
## 1636 1589 1625 1637 1637 1625 1563 1623 1557 1448 1575 1555 1495 1436 1619 1525
##  625  626  627  628  629  630  631  632  633  634  635  636  637  638  639  640
## 1558 1634 1483 1637 1545 1556 1644 1546 1606 1529 1631 1644 1561 1597 1517 1555
##  641  642  643  644  645  646  647  648  649  650  651  652  653  654  655  656
## 1627 1346 1459 1622 1543 1626 1606 1369 1640 1354 1644 1642 1382 1518  987 1641
##  657  658  659  660  661  662  663  664  665  666  667  668  669  670  671  672
## 1635 1593 1474 1441 1543 1642 1507 1499 1522 1419 1618 1618 1566 1618 1540 1635
##  673  674  675  676  677  678  679  680  681  682  683  684  685  686  687  688
## 1631 1623 1631 1589 1616 1636 1603 1626 1642 1267 1591 1578 1644 1593 1644 1640
##  689  690  691  692  693  694  695  696  697  698  699  700  701  702  703  704
## 1628 1549 1633 1624 1510 1507 1627 1635 1561 1537 1516 1643 1632 1633 1617 1574
##  705  706  707  708  709  710  711  712  713  714  715  716  717  718  719  720
## 1550 1635 1431 1560 1526 1580 1443 1502 1634 1620 1498 1395 1573 1625 1597 1635
##  721  722  723  724  725  726  727  728  729  730  731  732  733  734  735  736
## 1500 1622 1642 1582 1640 1640 1344 1638 1643 1627 1581 1644 1558 1600 1611 1640
##  737  738  739  740  741  742  743  744  745  746  747  748  749  750  751  752
## 1631 1516 1629 1644 1560 1638 1631 1637 1604 1597 1376 1556 1359 1633 1497 1593
##  753  754  755  756  757  758  759  760  761  762  763  764  765  766  767  768
## 1607 1630 1627 1555 1498 1309 1632 1623 1598 1643 1535 1555 1641 1489 1627 1600
##  769  770  771  772  773  774  775  776  777  778  779  780  781  782  783  784
## 1635 1602 1594 1631 1504 1440 1637 1560 1628 1600 1627 1609 1623 1438 1635 1627
##  785  786  787  788  789  790  791  792  793  794  795  796  797  798  799  800
## 1637 1547 1609 1416 1631 1434 1639 1617 1609 1625 1505 1307 1638 1426 1640 1636
##  801  802  803  804  805  806  807  808  809  810  811  812  813  814  815  816
## 1640 1590 1632 1333 1391 1524 1460 1641 1644 1639 1643 1645 1636 1629 1481 1639
##  817  818  819  820  821  822  823  824  825  826  827  828  829  830  831  832
## 1628 1644 1638 1643 1602 1639 1479 1645 1519 1549 1637 1561 1601 1558 1591 1639
```

```
##  833  834  835  836  837  838  839  840  841  842  843  844  845  846  847  848
## 1398 1611 1562 1615 1618 1571 1607 1468 1633 1639 1460 1583 1635 1260 1518 1514
##  849  850  851  852  853  854  855  856  857  858  859  860  861  862  863  864
## 1641 1613 1458 1615 1623 1449 1641 1638 1643 1643 1625 1595 1621 1499 1562 1370
##  865  866  867  868  869  870  871  872  873  874  875  876  877  878  879  880
## 1599 1645 1569 1457 1617 1396 1549 1594 1645 1630 1576 1643 1583 1530 1635 1299
##  881  882  883  884  885  886  887  888  889  890  891  892  893  894  895  896
## 1405 1527 1396 1622 1562 1425 1493 1644 1341 1545 1617 1439 1605 1422 1644 1302
##  897  898  899  900  901  902  903  904  905  906  907  908  909  910  911  912
## 1480 1634 1529 1619 1540 1616 1528 1617 1624 1623 1518 1590 1638 1604 1566 1612
##  913  914  915  916  917  918  919  920  921  922  923  924  925  926  927  928
## 1533 1641 1639 1348 1630 1561 1449 1639 1554 1537 1590 1582 1632 1645 1544 1633
##  929  930  931  932  933  934  935  936  937  938  939  940  941  942  943
## 1615 1601 1604 1423 1480 1491 1625 1523 1626 1556 1616 1557 1642 1587 1496
```

```
# Average ratings of first 20 users of all movies (1-5 Scale)
user_average <- rowMeans(m_lense_df, na.rm = TRUE)
head(user_average, n=20)
```

```
##        1        2        3        4        5        6        7        8
## 3.605166 3.704918 2.764706 4.304348 2.874286 3.639423 3.965000 3.796610
##        9       10       11       12       13       14       15       16
## 4.272727 4.206522 3.455556 4.392157 3.095238 4.091837 2.873786 4.328571
##       17       18       19       20
## 3.035714 3.880866 3.631579 3.104167
```

```
# Maximum and minimum ratings given by each user
rating_max <- apply(m_lense_df, 1, max, na.rm=TRUE)
rating_min <- apply(m_lense_df, 1, min, na.rm=TRUE)

# max and min ratings given by first 20 users
head(data.frame(max_rating = rating_max, min_rating = rating_min), n=20)
```

```
##    max_rating min_rating
## 1           5          1
## 2           5          1
## 3           5          1
## 4           5          2
## 5           5          1
## 6           5          1
## 7           5          1
## 8           5          1
## 9           5          1
## 10          5          3
## 11          5          1
## 12          5          1
## 13          5          1
## 14          5          1
## 15          5          1
## 16          5          1
## 17          5          1
## 18          5          2
## 19          5          2
## 20          5          1
```

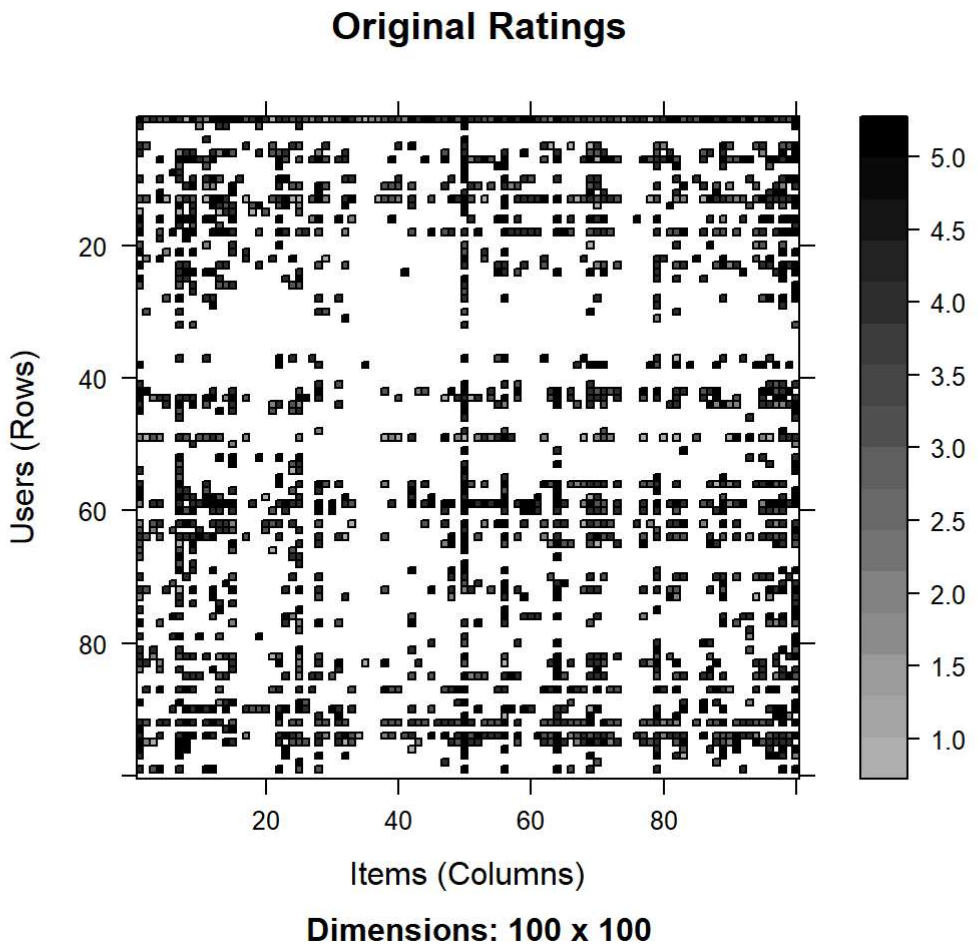
```
# max and min ratings given by last 20 users
tail(data.frame(max_rating = rating_max, min_rating = rating_min), n=20)
```

```
##     max_rating min_rating
## 924          5          2
## 925          5          1
## 926          5          1
## 927          5          1
## 928          5          3
## 929          5          1
## 930          5          1
## 931          5          1
## 932          5          1
## 933          5          1
## 934          5          1
## 935          5          1
## 936          5          1
## 937          5          1
## 938          5          1
## 939          5          2
## 940          5          1
## 941          5          2
## 942          5          2
## 943          5          1
```

```
# normalize the ratings matrix
m_norm <- normalize(movie_lense)
m_norm
```

```
## 943 x 1664 rating matrix of class 'realRatingMatrix' with 99392 ratings.
## Normalized using center on rows.
```

```
# visualizing using image
image(movie_lense[1:100, 1:100], main="Original Ratings")
```




```
cat("Rating matrix of 10 rows and 10 columns:")
```

```
## Rating matrix of 10 rows and 10 columns:
```

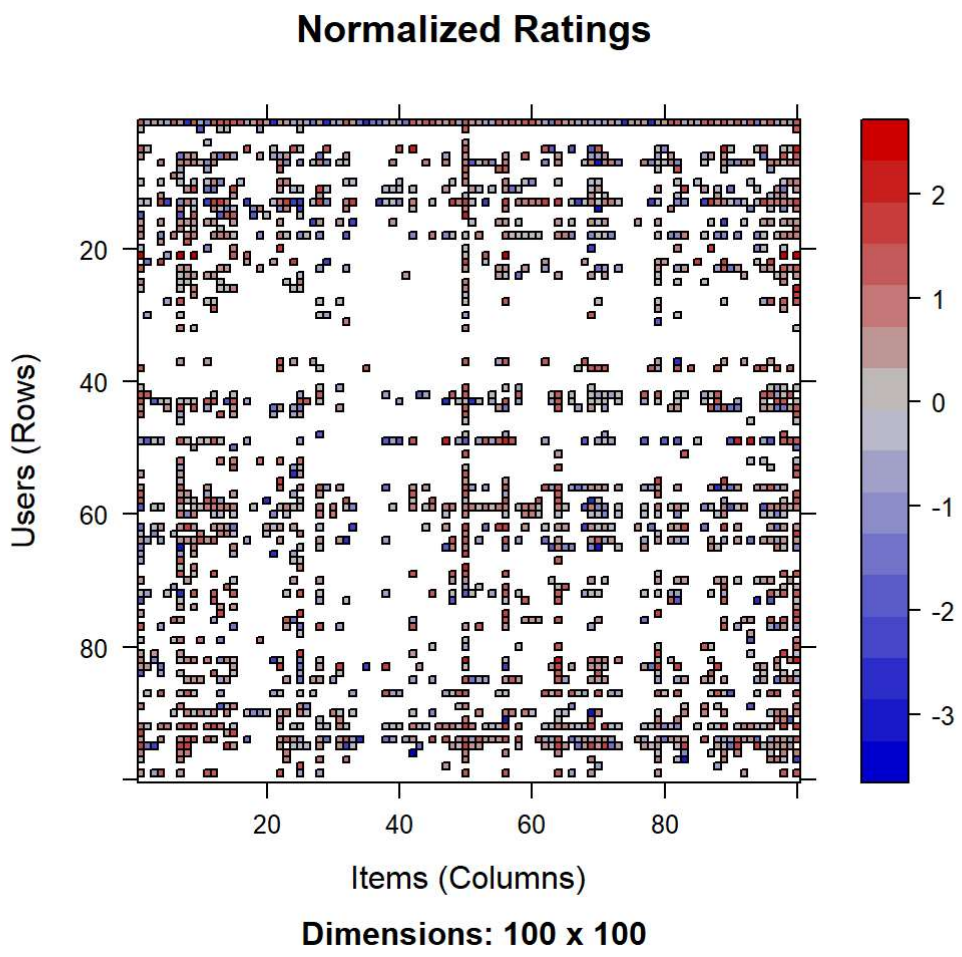
```
getRatingMatrix(movie_lense)[1:10, 1:10]
```

```
## 10 x 10 sparse Matrix of class "dgCMatrix"
```

```
##   [[ suppressing 10 column names 'Toy Story (1995)', 'GoldenEye (1995)', 'Four Rooms (1995)'
... ]]
```

```
##
## 1  5 3 4 3 3 5 4 1 5 3
## 2  4 . . . . . . . . 2
## 3  . . . . . . . . . .
## 4  . . . . . . . . . .
## 5  4 3 . . . . . . . .
## 6  4 . . . . . 2 4 4 .
## 7  . . . 5 . . 5 5 5 4
## 8  . . . . . . 3 . . .
## 9  . . . . . 5 4 . . .
## 10 4 . . 4 . . 4 . 4 .
```

```
image(m_norm[1:100, 1:100], main="Normalized Ratings")
```




```
cat("Rating matrix of 10 rows and 10 columns for the normalized data:")
```

```
## Rating matrix of 10 rows and 10 columns for the normalized data:
```

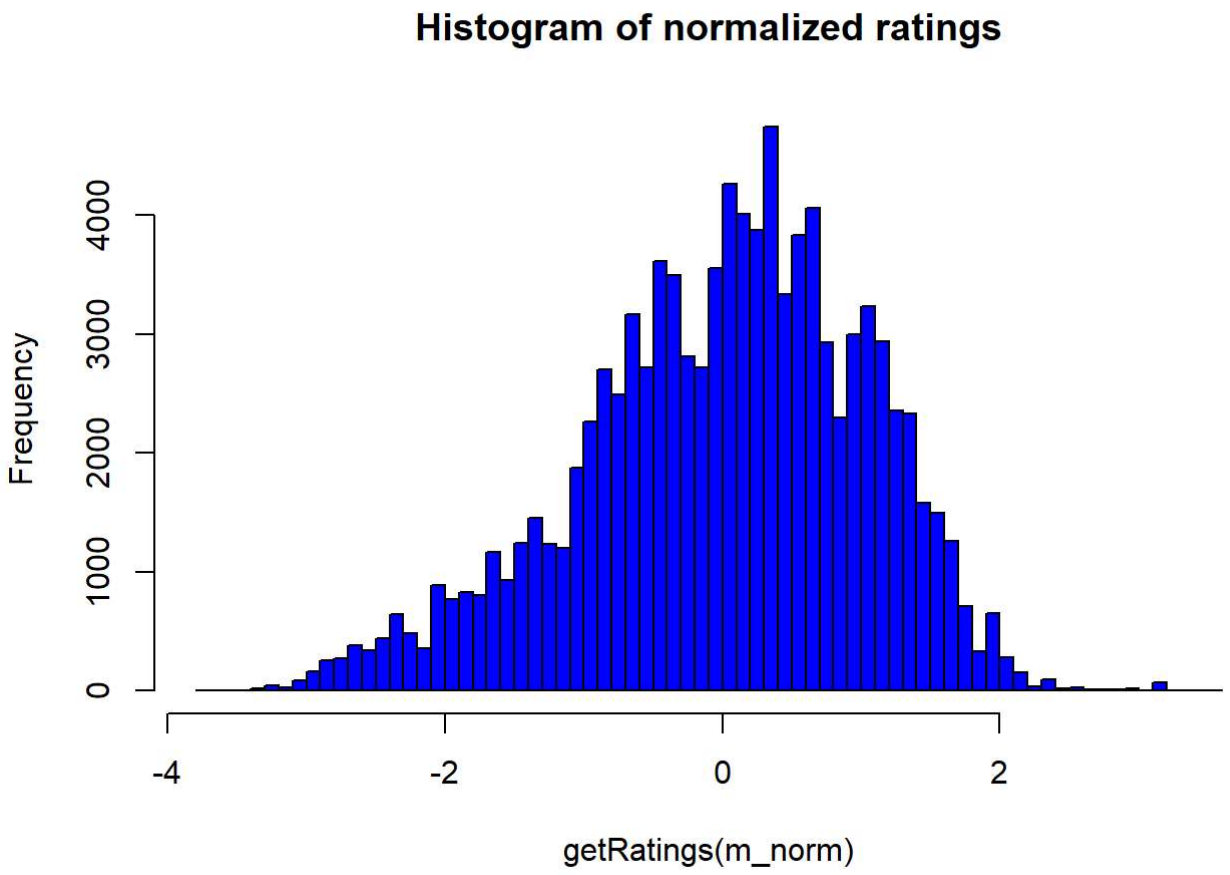
```
getRatingMatrix(m_norm)[1:10, 1:10]
```

```
## 10 x 10 sparse Matrix of class "dgCMatrix"
```

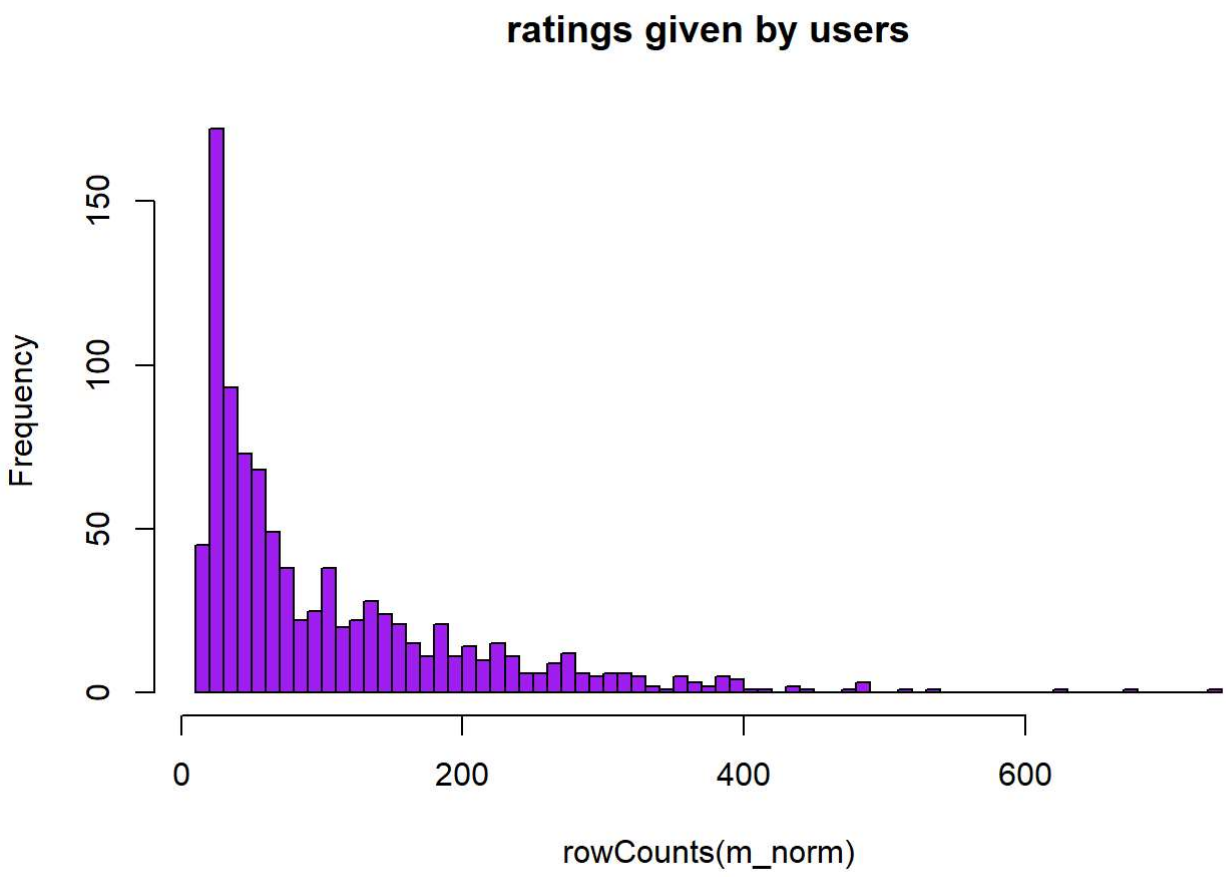
```
##   [[ suppressing 10 column names 'Toy Story (1995)', 'GoldenEye (1995)', 'Four Rooms (1995)'
... ]]
```

```
##
## 1   1.3948339 -0.6051661 0.3948339 -0.6051661 -0.6051661 1.3948339  0.3948339
## 2   0.2950820  .         .          .          .         .          .
## 3   .          .         .          .          .         .          .
## 4   .          .         .          .          .         .          .
## 5   1.1257143  0.1257143 .          .          .         .          .
## 6   0.3605769  .         .          .          .         .         -1.6394231
## 7   .          .         .          1.0350000  .         .          1.0350000
## 8   .          .         .          .          .         .         -0.7966102
## 9   .          .         .          .          .         0.7272727 -0.2727273
## 10 -0.2065217  .         .         -0.2065217  .         .         -0.2065217
##
## 1  -2.6051661  1.3948339 -0.6051661
## 2   .          .         -1.7049180
## 3   .          .          .
## 4   .          .          .
## 5   .          .          .
## 6   0.3605769  0.3605769  .
## 7   1.0350000  1.0350000  0.0350000
## 8   .          .          .
## 9   .          .          .
## 10  .         -0.2065217  .
```

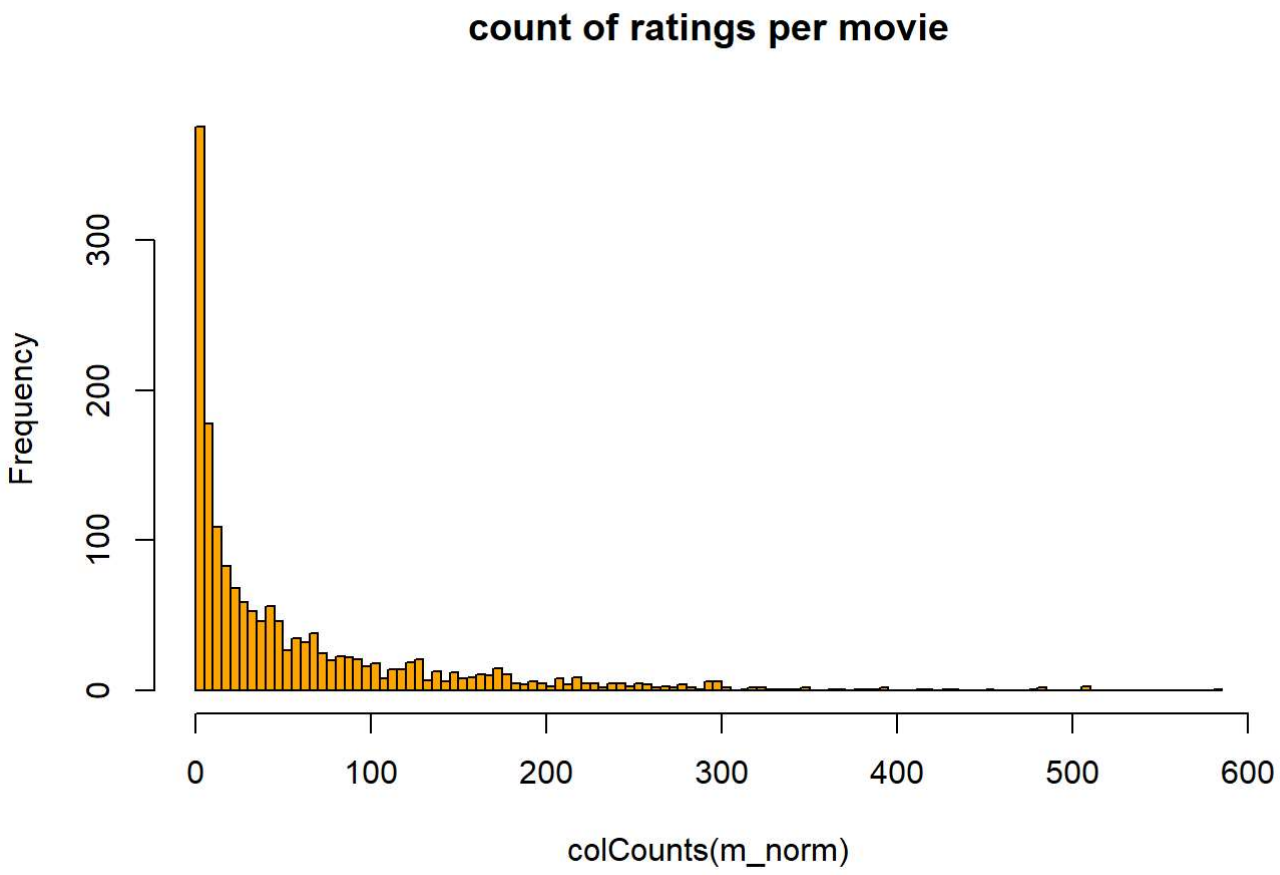
```
hist(getRatings(m_norm), breaks = 100, col = 'blue', main = "Histogram of normalized ratings")
```




```
hist(rowCounts(m_norm), breaks = 100, col = 'purple', main = "ratings given by users")
```




```
hist(colCounts(m_norm), breaks = 100, col = 'orange', main = "count of ratings per movie")
```




```
# Create a evaluation scheme (train, test)
eval_scheme <- evaluationScheme(m_norm, method="split", train=0.8, k=10, given=25, goodRating=3)
```

```
## Warning in .local(data, ...): Dropping these users from the evaluation since they have fewer
rating than specified in given!
## These users are 4, 9, 19, 33, 34, 35, 36, 39, 47, 50, 51, 55, 61, 78, 86, 88, 93, 105, 107, 1
11, 124, 127, 132, 139, 140, 143, 147, 153, 155, 163, 166, 170, 202, 205, 212, 220, 228, 231, 24
0, 241, 242, 245, 252, 258, 260, 266, 273, 278, 282, 300, 302, 309, 310, 317, 319, 335, 341, 35
6, 362, 364, 369, 384, 386, 400, 415, 418, 431, 441, 443, 444, 461, 475, 511, 512, 513, 516, 52
0, 547, 558, 570, 571, 572, 578, 584, 596, 631, 636, 649, 651, 652, 656, 662, 681, 685, 687, 68
8, 700, 723, 725, 726, 729, 732, 736, 740, 762, 765, 799, 801, 808, 809, 811, 812, 818, 820, 82
4, 849, 855, 857, 858, 866, 873, 876, 888, 895, 914, 926, 941
```

```
eval_scheme
```

```
## Evaluation scheme with 25 items given
## Method: 'split' with 10 run(s).
## Training set proportion: 0.800
## Good ratings: >=3.000000
## Data set: 816 x 1664 rating matrix of class 'realRatingMatrix' with 96667 ratings.
## Normalized using center on rows.
```

a) Develop a user-based recommender system. Create the system so that outputs a user's top ten recommendations. Demo it on five users.

```
# Create a recommender system using UBCF = User Based Collaborative Filtering
userbased_model <- Recommender(getData(eval_scheme,"train"), "UBCF")
```

```
## Warning in .local(x, ...): x was already normalized by row!
```

```
userbased_model
```

```
## Recommender of type 'UBCF' for 'realRatingMatrix'
## learned using 652 users.
```

```
# predicted ratings using model developed by UBCF
pred_ratings <- predict(userbased_model, getData(eval_scheme, "known"), type="ratings")
pred_ratings
```

```
## 164 x 1664 rating matrix of class 'realRatingMatrix' with 96617 ratings.
```

```
# Calculate the error
calcPredictionAccuracy(pred_ratings, getData(eval_scheme, "unknown"))
```

```
##      RMSE       MSE       MAE
## 1.2321418 1.5181735 0.9623162
```

```
# Create a recommended system
# UBCF = User Based Collaborative Filtering
recommender_engine <- Recommender(movie_lense, method="UBCF")
recommender_engine
```

```
## Recommender of type 'UBCF' for 'realRatingMatrix'
## learned using 943 users.
```

```
# Create the system that outputs a user's top ten recommendations
recommended_pred <- predict(recommender_engine, movie_lense, n=10)

# Convert the output to a list
recommended_list <- as(recommended_pred, "list")

# Get the top 10 movie recommendations of first 5 users
recommended_list[1:5]
```

```
## $`0`
##  [1] "Boot, Das (1981)"
##  [2] "Matilda (1996)"
##  [3] "Winter Guest, The (1997)"
##  [4] "She's the One (1996)"
##  [5] "Manchurian Candidate, The (1962)"
##  [6] "City of Lost Children, The (1995)"
##  [7] "Double vie de Veronique, La (Double Life of Veronique, The) (1991)"
##  [8] "187 (1997)"
##  [9] "Leaving Las Vegas (1995)"
## [10] "Wag the Dog (1997)"
##
## $`1`
##  [1] "Con Air (1997)"
##  [2] "Paradise Lost: The Child Murders at Robin Hood Hills (1996)"
##  [3] "Forbidden Planet (1956)"
##  [4] "Great Dictator, The (1940)"
##  [5] "Rob Roy (1995)"
##  [6] "M (1931)"
##  [7] "Game, The (1997)"
##  [8] "Local Hero (1983)"
##  [9] "C'est arrive pres de chez vous (1992)"
## [10] "In the Name of the Father (1993)"
##
## $`2`
##  [1] "Heavy Metal (1981)"
##  [2] "Mystery Science Theater 3000: The Movie (1996)"
##  [3] "Fear of a Black Hat (1993)"
##  [4] "Serial Mom (1994)"
##  [5] "Brady Bunch Movie, The (1995)"
##  [6] "Forbidden Planet (1956)"
##  [7] "39 Steps, The (1935)"
##  [8] "Crying Game, The (1992)"
##  [9] "Paths of Glory (1957)"
## [10] "Adventures of Priscilla, Queen of the Desert, The (1994)"
##
## $`3`
##  [1] "It Happened One Night (1934)"      "Jungle Book, The (1994)"
##  [3] "Deer Hunter, The (1978)"           "Man Who Would Be King, The (1975)"
##  [5] "39 Steps, The (1935)"              "Innocents, The (1961)"
##  [7] "Old Man and the Sea, The (1958)"   "Delta of Venus (1994)"
##  [9] "Big Lebowski, The (1998)"          "Richard III (1995)"
##
## $`4`
##  [1] "Paradise Lost: The Child Murders at Robin Hood Hills (1996)"
##  [2] "Leaving Las Vegas (1995)"
##  [3] "Primal Fear (1996)"
##  [4] "Anna Karenina (1997)"
##  [5] "Wallace & Gromit: The Best of Aardman Animation (1996)"
##  [6] "Wizard of Oz, The (1939)"
##  [7] "Clockwork Orange, A (1971)"
##  [8] "Field of Dreams (1989)"
```

```
##  [9] "Sling Blade (1996)"
## [10] "Night of the Living Dead (1968)"
```

```
# Creating recommender engine (with method = "POPULAR")
recomm_eng_popular <- Recommender(movie_lense, method = "POPULAR")

# Create the system so that outputs a user's top ten recommendations (with method = "POPULAR")
recommended_pred_popular <- predict(recomm_eng_popular, movie_lense, n=10)

recommended_list_popular <- as(recommended_pred_popular, "list")

recommended_list_popular[1:5]
```

```
## $`1`
##  [1] "Titanic (1997)"
##  [2] "Schindler's List (1993)"
##  [3] "L.A. Confidential (1997)"
##  [4] "Casablanca (1942)"
##  [5] "One Flew Over the Cuckoo's Nest (1975)"
##  [6] "Rear Window (1954)"
##  [7] "To Kill a Mockingbird (1962)"
##  [8] "Boot, Das (1981)"
##  [9] "Dr. Strangelove or: How I Learned to Stop Worrying and Love the Bomb (1963)"
## [10] "North by Northwest (1959)"
##
## $`2`
##  [1] "Raiders of the Lost Ark (1981)"   "Silence of the Lambs, The (1991)"
##  [3] "Schindler's List (1993)"          "Shawshank Redemption, The (1994)"
##  [5] "Empire Strikes Back, The (1980)"  "Return of the Jedi (1983)"
##  [7] "Usual Suspects, The (1995)"       "Casablanca (1942)"
##  [9] "Pulp Fiction (1994)"              "Princess Bride, The (1987)"
##
## $`3`
##  [1] "Star Wars (1977)"                 "Godfather, The (1972)"
##  [3] "Fargo (1996)"                     "Raiders of the Lost Ark (1981)"
##  [5] "Silence of the Lambs, The (1991)" "Titanic (1997)"
##  [7] "Shawshank Redemption, The (1994)" "Empire Strikes Back, The (1980)"
##  [9] "Usual Suspects, The (1995)"       "Casablanca (1942)"
##
## $`4`
##  [1] "Godfather, The (1972)"            "Fargo (1996)"
##  [3] "Raiders of the Lost Ark (1981)"   "Silence of the Lambs, The (1991)"
##  [5] "Titanic (1997)"                   "Schindler's List (1993)"
##  [7] "Shawshank Redemption, The (1994)" "Empire Strikes Back, The (1980)"
##  [9] "Return of the Jedi (1983)"        "Usual Suspects, The (1995)"
##
## $`5`
##  [1] "Godfather, The (1972)"
##  [2] "Titanic (1997)"
##  [3] "Schindler's List (1993)"
##  [4] "Shawshank Redemption, The (1994)"
##  [5] "Usual Suspects, The (1995)"
##  [6] "L.A. Confidential (1997)"
##  [7] "Casablanca (1942)"
##  [8] "Pulp Fiction (1994)"
##  [9] "One Flew Over the Cuckoo's Nest (1975)"
## [10] "Braveheart (1995)"
```

b) For the same users in part A, predict their top 10 recommendations based on an item-based recommender system. How do they compare?

```
# Create a recommender system
# IBCF = Item Based Collaborative Filtering
recommender_sys_IBCF <- Recommender(movie_lense, method="IBCF")
recommender_sys_IBCF
```

```
## Recommender of type 'IBCF' for 'realRatingMatrix'
## learned using 943 users.
```

```
# Create the system that outputs a user’s top ten recommendations
recommended_pred1 <- predict(recommender_sys_IBCF, movie_lense, n=10)

# Convert the output to a list
recommended_list1 <- as(recommended_pred1, "list")

recommended_list1[1:5]
```

```
## $`0`
## [1] "Entertaining Angels: The Dorothy Day Story (1996)"
## [2] "Big Bang Theory, The (1994)"
## [3] "They Made Me a Criminal (1939)"
## [4] "Cyclo (1995)"
## [5] "King of New York (1990)"
## [6] "Very Natural Thing, A (1974)"
## [7] "Walk in the Sun, A (1945)"
## [8] "Hush (1998)"
##
## $`1`
##  [1] "They Made Me a Criminal (1939)"
##  [2] "Entertaining Angels: The Dorothy Day Story (1996)"
##  [3] "Big Bang Theory, The (1994)"
##  [4] "Other Voices, Other Rooms (1997)"
##  [5] "Further Gesture, A (1996)"
##  [6] "Cyclo (1995)"
##  [7] "Very Natural Thing, A (1974)"
##  [8] "Walk in the Sun, A (1945)"
##  [9] "Visitors, The (Visiteurs, Les) (1993)"
## [10] "Hush (1998)"
##
## $`2`
##  [1] "Entertaining Angels: The Dorothy Day Story (1996)"
##  [2] "Legal Deceit (1997)"
##  [3] "Other Voices, Other Rooms (1997)"
##  [4] "Hush (1998)"
##  [5] "Big Bang Theory, The (1994)"
##  [6] "Visitors, The (Visiteurs, Les) (1993)"
##  [7] "Mirage (1995)"
##  [8] "Further Gesture, A (1996)"
##  [9] "Cyclo (1995)"
## [10] "Tokyo Fist (1995)"
##
## $`3`
##  [1] "Men of Means (1998)"
##  [2] "Entertaining Angels: The Dorothy Day Story (1996)"
##  [3] "Mirage (1995)"
##  [4] "Further Gesture, A (1996)"
##  [5] "Big Bang Theory, The (1994)"
##  [6] "Other Voices, Other Rooms (1997)"
##  [7] "Star Kid (1997)"
##  [8] "Very Natural Thing, A (1974)"
##  [9] "Walk in the Sun, A (1945)"
## [10] "Tokyo Fist (1995)"
##
## $`4`
##  [1] "They Made Me a Criminal (1939)"
##  [2] "Coldblooded (1995)"
##  [3] "Visitors, The (Visiteurs, Les) (1993)"
##  [4] "Entertaining Angels: The Dorothy Day Story (1996)"
##  [5] "King of New York (1990)"
```

```
##  [6] "Shopping (1994)"
##  [7] "Nemesis 2: Nebula (1995)"
##  [8] "Very Natural Thing, A (1974)"
##  [9] "Walk in the Sun, A (1945)"
## [10] "Cyclo (1995)"
```

After comparison on User based (Part A) and Item based (Part B) collaborative filtering for both, the top ten recommendations for first 5 users are completely different.

c) Consider the first user part A. Are the goals of a recommender system met – relevance, novelty, serendipity, diversity? Comment on each of the best you can given the information you have.

```
recommended_list[1]  ### 10 recommended movies for 1st user in part A.
```

```
## $`0`
##  [1] "Boot, Das (1981)"
##  [2] "Matilda (1996)"
##  [3] "Winter Guest, The (1997)"
##  [4] "She's the One (1996)"
##  [5] "Manchurian Candidate, The (1962)"
##  [6] "City of Lost Children, The (1995)"
##  [7] "Double vie de Veronique, La (Double Life of Veronique, The) (1991)"
##  [8] "187 (1997)"
##  [9] "Leaving Las Vegas (1995)"
## [10] "Wag the Dog (1997)"
```

```
### Based on the list of 10 recommended movies from part A for the first user, below is my asses
sment of how well the goals of relevance, novelty, serendipity, and diversity are met:

### Relevance: The list seems to contain critically acclaimed dramatic and art-house style film
s. If matched to a user inclined to these genres, it could be relevant. If mismatched to the use
r's preferences, relevance would be low.

### Novelty: There is a mix of classic, older films along with 1990s releases, so some temporal
diversity. But lacks odd titles that would bring more novelty. Overall novelty seems moderate.

### Serendipity: Some less mainstream picks like "City of Lost Children" and "Double Life of Ver
onique" provide potential for delightful unexpected discoveries. But serendipity could be higher
with more surprising and uncommon picks.

### Diversity: There is decent diversity of genres beyond just drama/art-house - e.g. political
thriller, comedy, etc. But the recommended movies still seems to have a narrower "critical darli
ngs" style, lacking broader taste differences.

### In summary, particularly in better matching to unique user interests to enhance relevance, a
s well as providing more novel, unexpected finds and catering to a wider spread of tastes to inc
rease diversity. Some serendipitous picks show promise. More personalization could help hit the
recommender goals more efficiently.
```