STA 546 - Homework 1

Abbas Rizvi

February 24, 2016

Problem 1

The bodyfat dataset was considered. The bodyfat dataset consists of bodyfat estimates by underwater weighing and various body circumference measurements for 252 men.

```
setwd("/Users/aarizvi/Google Drive/STA546/hw1/")
load("bodyfat.rdata")
```

The task for this problem was to pre-process the data, such that outliers, unusual distributions, and scaling issues are addressed. To begin exploration of this dataset, the analysis began with looking at the columns in order to address any scaling issues.

```
head(bodyfat)
```

```
##   density bodyfat age weight height neck chest abdomen   hip thigh knee
## 1  1.0708    12.3  23 154.25  67.75 36.2  93.1    85.2  94.5  59.0 37.3
## 2  1.0853     6.1  22 173.25  72.25 38.5  93.6    83.0  98.7  58.7 37.3
## 3  1.0414    25.3  22 154.00  66.25 34.0  95.8    87.9  99.2  59.6 38.9
## 4  1.0751    10.4  26 184.75  72.25 37.4 101.8    86.4 101.2  60.1 37.3
## 5  1.0340    28.7  24 184.25  71.25 34.4  97.3   100.0 101.9  63.2 42.2
## 6  1.0502    20.9  24 210.25  74.75 39.0 104.5    94.4 107.8  66.0 42.0
##   ankle biceps forearm wrist
## 1  21.9   32.0    27.4  17.1
## 2  23.4   30.5    28.9  18.2
## 3  24.0   28.8    25.2  16.6
## 4  22.8   32.4    29.4  18.2
## 5  24.0   32.2    27.7  17.7
## 6  25.6   35.7    30.6  18.8
```

It was quickly noticed that the units for columns weight (`bodyfat$weight`) and height (`bodyfat$height`) are measured in U.S. customary units and the remaining columns were measured using SI units. This could be problematic when computing simple calculations using the data, therefore `bodyfat$weight` and `bodyfat$height` were converted into SI units.

```
bodyfat$weight <- round((bodyfat$weight / 2.2)*1000, 2) #convert weight from lbs into grams
bodyfat$height <- round(bodyfat$height * 2.54, 2) #convert height from inches into cm
```

Subsequently, the data was visualized using boxplots and histograms to observe the distrubtions of the different column measurements and identify any outliers (Figure 1).

```
#visualize boxplots for all columns
par(mfrow = c(3,5),
          oma = c(5,4,0,0) + 0.1,
          mar = c(0,0,1,2) + 0.1)
```

```
colnames <- dimnames(bodyfat)[[2]]
for(i in 1:ncol(bodyfat)){
        boxplot(bodyfat[,i], main=colnames[i])
}
```

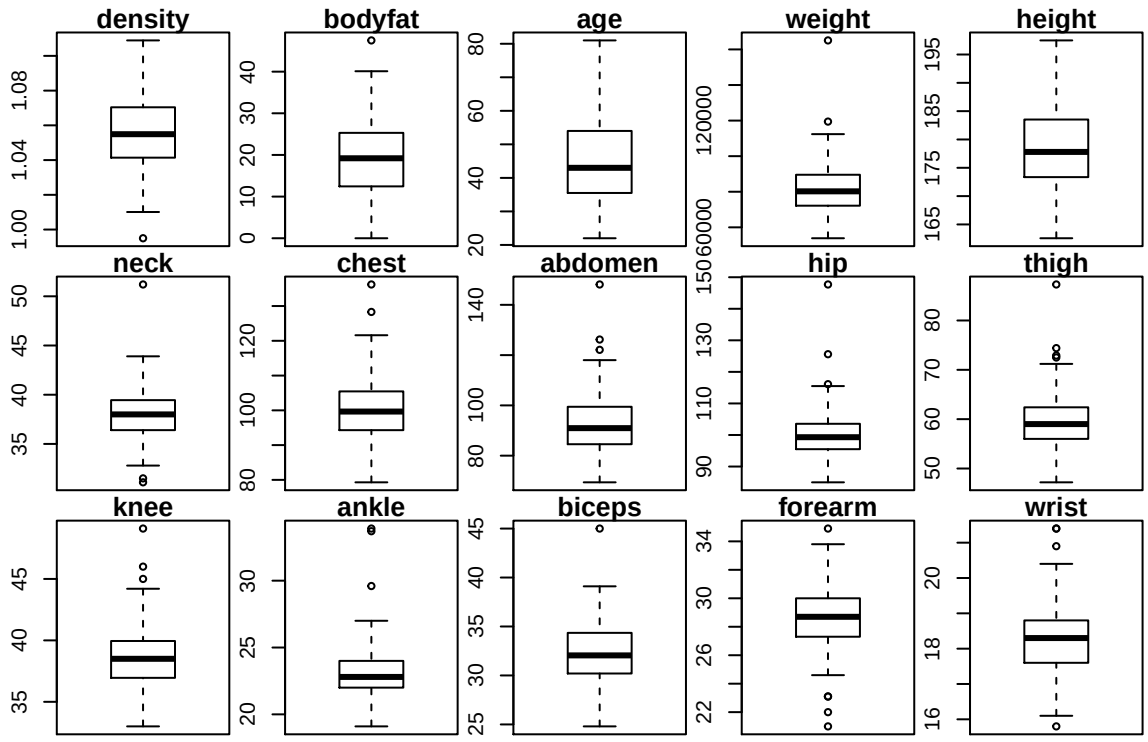

Figure 1: Boxplots of bodyfat measurements

Outliers are noticeable in the most columns (Figure 1). Typically, outliers are not removed, but for this exercise we shall remove outliers in order to keep our dataset as simple as possible. But before removing let's visualize this another way using histograms with overlayed density plots of the different measurement groups (Figure 2).

```
#visualize histograms for all columns
library(MASS)
par(mfrow = c(3,5),
          oma = c(5,4,0,0) + 0.1,
          mar = c(3,3,1,1) + 0.1)
for(i in 1:ncol(bodyfat)){
        truehist(bodyfat[,i], main=colnames[i], col="gray", border="white")
             d <- density(bodyfat[,i])
             lines(d, col="red")
}
```

The distributions all look approximately normal, however, there are some with some noticeable tails, and that is due to the outliers. We will now remove the outliers.

```
#now its time to remove outliers
indices <- list()
for(i in 1:ncol(bodyfat)){
```

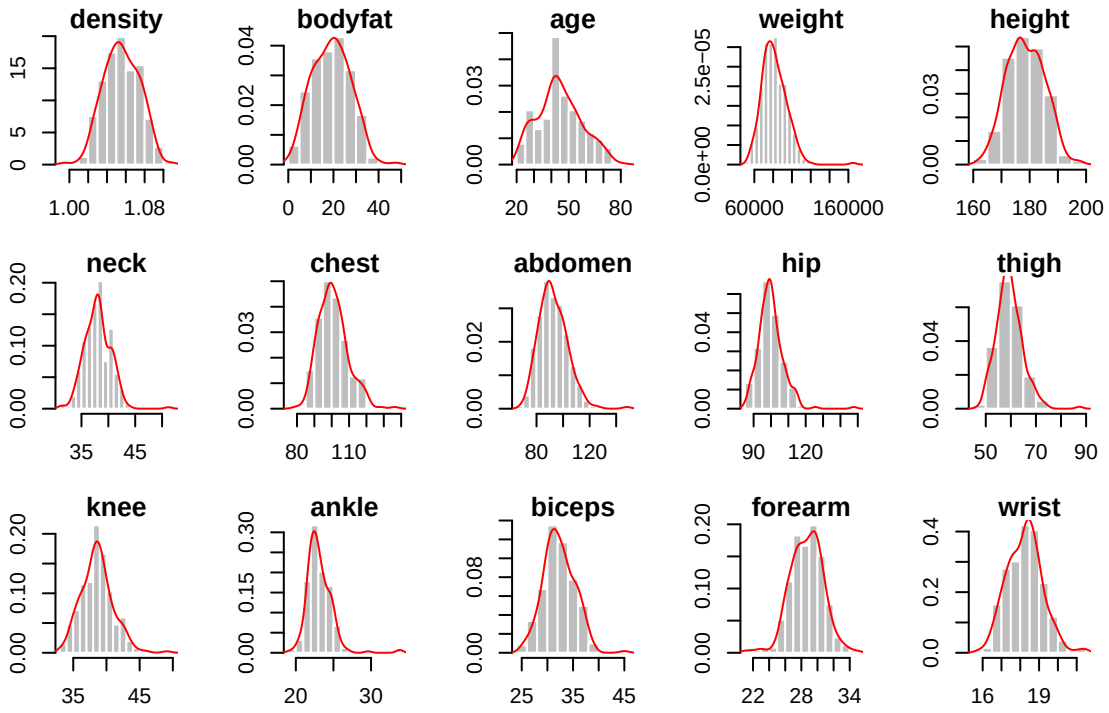

Figure 2: Histograms of bodyfat measurements

```
        indices[[i]] <- which(!bodyfat[,i] %in% boxplot.stats(bodyfat[,i])$out == 'FALSE')
}
outlier.indices <- unique(unlist(indices))
bodyfat.clean <- bodyfat[-c(outlier.indices),]
```

After dropping the row indices with outliers, the dataset has been reduced to 235 observations from 252 observations. This data set has now been pre-processed and saved as `clean_data.RData` and can be found in the attachment with the report.

The data was visualized again to see how the distributions changed after removal of outliers (Figure 3).

```
#take a look at histograms/boxplots again
par(mfrow = c(3,5),
          oma = c(5,4,0,0) + 0.1,
          mar = c(3,3,1,1) + 0.1)
for(i in 1:ncol(bodyfat.clean)){
        truehist(bodyfat.clean[,i], main=colnames[i], col="gray", border="white")
        d <- density(bodyfat.clean[,i])
        lines(d, col="red")
}
```

The histograms reveal that the outlier excluded data distributions appear to be closer to 'normal'.

Problem 2

This problem was adopted from Chapter 9 - Exercise 9.3.1. We want to measure the similarity of users or items from their rows or columns in a utility matrix. In order to measure the similarity of two documents (or

Figure 3: Histograms with outliers removed

in this case, user's reviewing movies), jaccard and cosine distances can be used.

The jaccard index is defined as the size of the intersections divided by the size of the union between samples and the jaccard distance (which measures the dissimilarity) is defined as 1-jaccard index. The following function was written to represent the jaccard distance:

```
jaccard <- function(a, b){
        int <- sum(a[a==b])
        union <- sum(a+b)-int
        round(1-(int/union), 4)
}
```

The cosine similarity is a measure of similarity between two vectors of an inner product space such that it measures the cosine of the angle between the two vectors. Cosine simlarity is reprsented by:

$$similarity = cos(\theta) = \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} A_i^2}\sqrt{\sum_{i=1}^{n} B_i^2}}$$

, where A_i and B_i are componets of vector A and B

Cosine distance is 1-cosine similarity.

The following function was written for cosine dissimilarity:

```
cosine <- function(a, b){
        AB <- a*b
        AB <- sum(AB)
        AA <- a^2
        AA <- sum(AA)
        BB <- b^2
```

```
        BB <- sum(BB)
        1-round(AB/(sqrt(AA)*sqrt(BB)), 4)
}
```

2.1

The utility matrix (`mat`) in Table 1 was considered.

```
require(xtable)
mat <- data.frame(rbind(c(4, 5, 0, 5, 1, 0, 3, 2),
                  c(0, 3, 4, 3, 1, 2, 1, 0),
                  c(2, 0, 1, 3, 0, 4, 5, 3)))
colnames(mat) <- c(letters[1:8])
rownames(mat) <- c("1", "2", "3")
xmat <- xtable(mat,
               caption="Utility Matrix Adopted from 'Ch9 Recommendation Systems'",
               digits=1)
print.xtable(xmat, comment=FALSE)
```

	a	b	c	d	e	f	g	h
1	4.0	5.0	0.0	5.0	1.0	0.0	3.0	2.0
2	0.0	3.0	4.0	3.0	1.0	2.0	1.0	0.0
3	2.0	0.0	1.0	3.0	0.0	4.0	5.0	3.0

Table 1: Utility Matrix Adopted from 'Ch9 Recommendation Systems'

The rows in table 1 are individual users and the columns are variables for movies. The utility matrix (Table 1) was converted as Boolean under the conditions that any values > 0 were converted as `TRUE` and any values < 0 were converted to `FALSE`.

```
## treat the utility matrix as boolean
boolean.mat <- rbind(c(TRUE, TRUE, FALSE, TRUE, TRUE, FALSE, TRUE, TRUE),
                     c(FALSE, TRUE, TRUE, TRUE, TRUE, TRUE, TRUE, FALSE),
                     c(TRUE, FALSE, TRUE, TRUE, FALSE, TRUE, TRUE, TRUE))
```

The Jaccard distance between users was computed for the Boolean-converted utility matrix. This Boolean matrix essentially means that if there was a rating reported by a user, they are considered `TRUE` and if they didn't have a rating for the movies, they were considered `FALSE`.

```
## compute Jaccard distance between users
User1vUser2.bool <- jaccard(boolean.mat[1,], boolean.mat[2,])
User1vUser3.bool <- jaccard(boolean.mat[1,], boolean.mat[3,])
User2vUser3.bool <- jaccard(boolean.mat[2,], boolean.mat[3,])
message(paste0("Jaccard Distance between User 1 and 2 is: ", User1vUser2.bool, "\n",
               "Jaccard Distance between User 1 and 3 is: ", User1vUser3.bool, "\n",
               "Jaccard Distance between User 2 and 3 is: ", User2vUser3.bool))
```

```
## Jaccard Distance between User 1 and 2 is: 0.5
## Jaccard Distance between User 1 and 3 is: 0.5
## Jaccard Distance between User 2 and 3 is: 0.5
```

This tells us that all 3 users are equal in distance from one another.

The cosine distance between users was computed for the Boolean-converted utility matrix.

```
## compute cosine distance between users
User1vUser2.bool.cos <- cosine(boolean.mat[1,], boolean.mat[2,])
User1vUser3.bool.cos <- cosine(boolean.mat[1,], boolean.mat[3,])
User2vUser3.bool.cos <- cosine(boolean.mat[2,], boolean.mat[3,])
message(paste0("Cosine Distance between User 1 and 2 is: ", User1vUser2.bool.cos, "\n",
               "Cosine Distance between User 1 and 3 is: ", User1vUser3.bool.cos, "\n",
               "Cosine Distance between User 2 and 3 is: ", User2vUser3.bool.cos))
```

```
## Cosine Distance between User 1 and 2 is: 0.6667
## Cosine Distance between User 1 and 3 is: 0.6667
## Cosine Distance between User 2 and 3 is: 0.6667
```

Similar to our result for Jaccard distance, the cosine distances are the same. Intuitively this makes sense, as each user rated 6 movies, with two movies overlapping between the 3, and all the remaining movies represented by at least a pair of users.

2.2

The utility matrix (Table 1) was considered under a different discretization such that ratings of 3, 4, 5 were treated as 1, and ratings 1, 2, and blanks, were treated as 0.

```
## part b -- treat ratings 3,4,5 as 1, and ratings 1,2, and blank as 0

util.mat.b <- rbind(c(1, 1, 0, 1, 0, 0, 1, 0),
                    c(0, 1, 1, 1, 0, 0, 0, 0),
                    c(0, 0, 0, 0, 1, 0, 1, 1))
```

We can already observe that `User 1` has 4 movies that are reported as 1 and 4 movies that reported as 0. The other two users have 3 movies reported a 1 and 5 movies reported as 0.

The Jaccard distance was computed between users for the discretized utility matrix:

```
## compute jaccard
User1vUser2utilb <- jaccard(util.mat.b[1,], util.mat.b[2,])
User1vUser3utilb <- jaccard(util.mat.b[1,], util.mat.b[3,])
User2vUser3utilb <- jaccard(util.mat.b[2,], util.mat.b[3,])
message(paste0("Jaccard Distance between User 1 and 2 is: ", User1vUser2utilb, "\n",
               "Jaccard Distance between User 1 and 3 is: ", User1vUser3utilb, "\n",
               "Jaccard Distance between User 2 and 3 is: ", User2vUser3utilb))
```

```
## Jaccard Distance between User 1 and 2 is: 0.6
## Jaccard Distance between User 1 and 3 is: 0.8333
## Jaccard Distance between User 2 and 3 is: 1
```

The cosine distance was computed between users for the discretized utility matrix:

```
## compute cosine
User1vUser2utilb.cos <- cosine(util.mat.b[1,], util.mat.b[2,])
User1vUser3utilb.cos <- cosine(util.mat.b[1,], util.mat.b[3,])
User2vUser3utilb.cos <- cosine(util.mat.b[2,], util.mat.b[3,])
message(paste0("Cosine Distance between User 1 and 2 is: ", User1vUser2utilb.cos, "\n",
               "Cosine Distance between User 1 and 3 is: ", User1vUser3utilb.cos, "\n",
               "Cosine Distance between User 2 and 3 is: ", User2vUser3utilb.cos))
```

```
## Cosine Distance between User 1 and 2 is: 0.4226
## Cosine Distance between User 1 and 3 is: 0.7113
## Cosine Distance between User 2 and 3 is: 1
```

The distance measures both reported user 2 and 3 were similiar. The distance measure between 1 and 3 were further than 1 and 2. This makes sense as well from looking at the matrix.

2.3

The utility matrix (Table 1) was normalized by subtracting mean user rating by each nonblank entry. The cosine distances were computed between each pair of users.

```
org.mat <- rbind(c(4, 5, 0, 5, 1, 0, 3, 2),
                 c(0, 3, 4, 3, 1, 2, 1, 0),
                 c(2, 0, 1, 3, 0, 4, 5, 3))
org.mat[org.mat==0] <- NA #replaces zeroes as NAs
#norm function subtracts non-NA values in row by mean of that row
norm <- function(x) {sweep(x, 1, rowSums(x,na.rm=T)/ncol(x), "-")}
norm.org <- norm(org.mat) #apply normalization
norm.org[is.na(norm.org)] <- 0 #replace NAs back to 0s
norm.org
```

```
##       [,1] [,2]  [,3] [,4]  [,5] [,6]  [,7]  [,8]
## [1,]  1.50 2.50  0.00 2.50 -1.50 0.00  0.50 -0.50
## [2,]  0.00 1.25  2.25 1.25 -0.75 0.25 -0.75  0.00
## [3,] -0.25 0.00 -1.25 0.75  0.00 1.75  2.75  0.75
```

```
User1vUser2utilb.norm <- cosine(norm.org[1,], norm.org[2,])
User1vUser3utilb.norm <- cosine(norm.org[1,], norm.org[3,])
User2vUser3utilb.norm <- cosine(norm.org[2,], norm.org[3,])
message(paste0("Cosine Distance between User 1 and 2 is: ", User1vUser2utilb.norm, "\n",
               "Cosine Distance between User 1 and 3 is: ", User1vUser3utilb.norm, "\n",
               "Cosine Distance between User 2 and 3 is: ", User2vUser3utilb.norm))
```

```
## Cosine Distance between User 1 and 2 is: 0.4535
## Cosine Distance between User 1 and 3 is: 0.8366
## Cosine Distance between User 2 and 3 is: 1.3126
```

The cosine distance between each pair of users was calculated. The normalized results show that user 2 and 3 are very far apart, while user 1 and 2 are the closet. This makes sense looking at the original data.

Problem 3

The Boston housing data from `MASS` was considered.

```
require(MASS)
data(Boston)
head(Boston)
```

```
##      crim zn indus chas   nox    rm  age    dis rad tax ptratio  black
## 1 0.00632 18  2.31    0 0.538 6.575 65.2 4.0900   1 296    15.3 396.90
## 2 0.02731  0  7.07    0 0.469 6.421 78.9 4.9671   2 242    17.8 396.90
## 3 0.02729  0  7.07    0 0.469 7.185 61.1 4.9671   2 242    17.8 392.83
## 4 0.03237  0  2.18    0 0.458 6.998 45.8 6.0622   3 222    18.7 394.63
## 5 0.06905  0  2.18    0 0.458 7.147 54.2 6.0622   3 222    18.7 396.90
## 6 0.02985  0  2.18    0 0.458 6.430 58.7 6.0622   3 222    18.7 394.12
##   lstat medv
## 1  4.98 24.0
## 2  9.14 21.6
## 3  4.03 34.7
## 4  2.94 33.4
## 5  5.33 36.2
## 6  5.21 28.7
```

3.1

Histograms of the different clinical variables can be seen in Figure 4.

```
boston <- Boston #keep the raw data was Boston, use boston for EDA
par(mfrow = c(3,5),
          oma = c(5,4,0,0) + 0.1,
          mar = c(3,3,1,1) + 0.1)
for(i in 1:ncol(Boston)){
        truehist(Boston[,i], main=colnames(Boston)[i], col="gray", border="white")
        d <- density(Boston[,i])
        lines(d, col="red")
}
```

After looking at the histogram and the documentation provided as to what variables are important for predictive purposes (as well as what aligns well to the questions assigned in this write-up), the columns `rad`, `zn`, `chas`, and `black` were discarded from the dataset.

```
drops <- c("rad", "zn", "chas", "black")
boston <- boston[,!(names(boston) %in% drops)]
```

The next goal was to transform the data into a binary incidence matrix. However, association rule data mining can only use data that is discrete. Hence, the data in each column variable must be discretized before being stored in a binary incidence matrix. Each variable was heuristically decoded into small subgroup. For example, a heuristic method could include looking at `summary()` or observing the distribution of the variable and making decisions on how to split the data into smaller categories. Each variable was carefully investigated and an informed decision went into the discretization process.

The following R chunk is an example of the discretization process. The code for the other variables can be found with the attached `R` files.

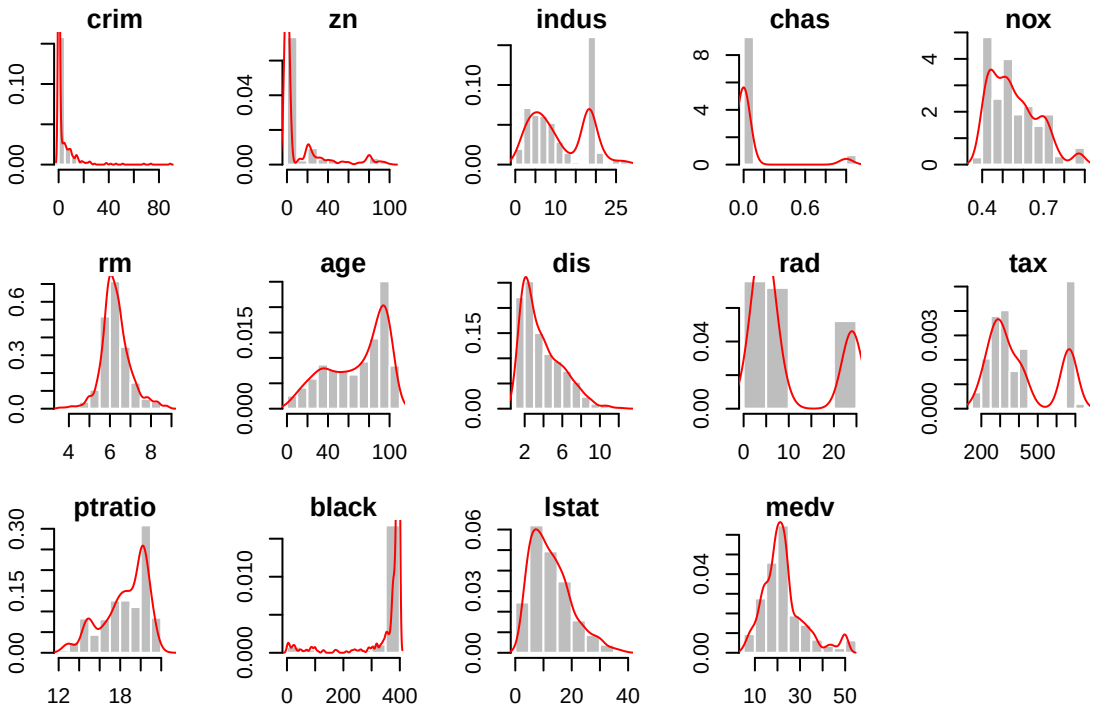

Figure 4: Histograms of different variables in Boston dataset

```
#discretize columns - could have used discretize()
summary(boston$ptratio)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   12.60   17.40   19.05   18.46   20.20   22.00
```

```
boston$ptratio <- ordered(cut(boston$ptratio, c(12.6,17.4,20.2,23),
                              labels=c("Below Average", "Average", "Above Average")))
```

3.2

After all columns were discretized, the data could be stored into a binary incidence matrix.

```
require(arules)
load("boston_dis.Rda")
Bos <- as(boston, "transactions")
Bos
```

```
## transactions in sparse format with
##  506 transactions (rows) and
##  31 items (columns)
```

`Bos` contains the binary incidence matrix. The data was visualized using `arules::itemFrequencyPlot` (Figure 5).

```
itemFrequencyPlot(Bos, support = 0.01, cex.names = 0.8)
```

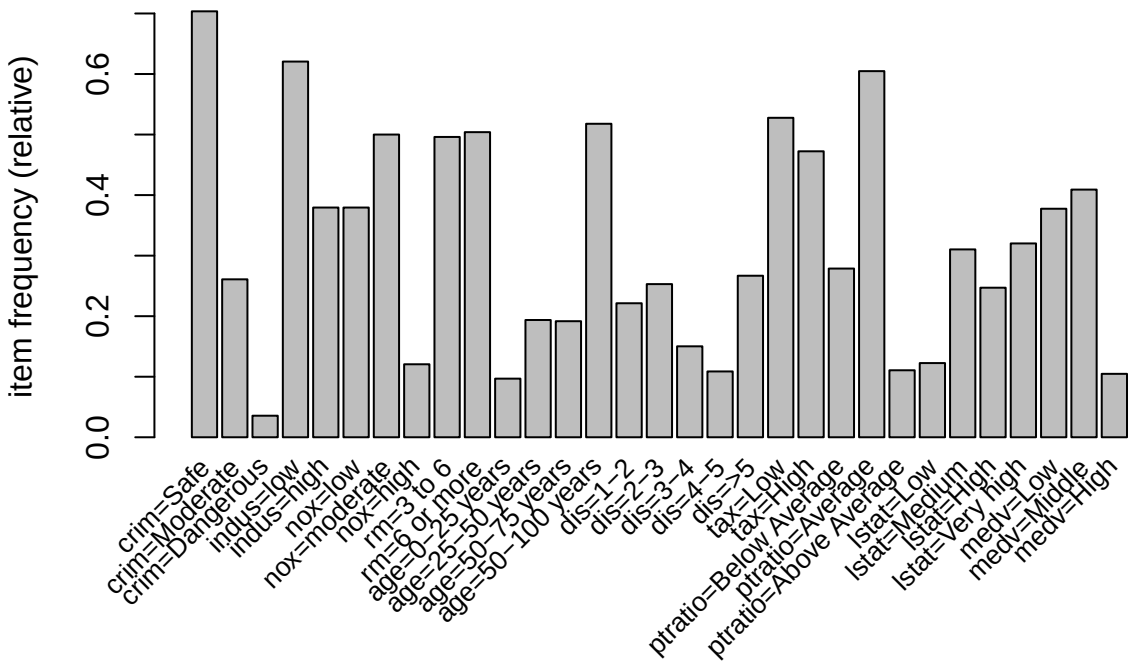

Figure 5: itemFrequencyPlot of Boston Dataset

Now we would like to mine the items using the apriori algorithm. Apriori is an algorithm for frequent item set mining and association rules over transaction databases (Wikipedia). The algorithm employs a level-wise search for individual items in the entire database. The support indicates how frequently the items appear in the dataset. The confidence indicates the number of times IF/THEN statement are TRUE in the data. The `support = 0.01` was chosen after visually observing which variables would be included with other parts of the exercise being considered (i.e. investigating crime and pupil-teacher ratios). The default `confidence` is `0.8`, however, a `confidence` of `0.6` was chosen so more vague associations were investigated.

```
# Apply the apriori algorithm
rules  <- apriori(Bos, parameter = list(support = 0.05, confidence = 0.6))
```

```
## Apriori
##
## Parameter specification:
##  confidence minval smax arem  aval originalSupport support minlen maxlen
##         0.6    0.1    1 none FALSE            TRUE    0.05      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: 25
##
## set item appearances ...[0 item(s)] done [0.00s].
## set transactions ...[31 item(s), 506 transaction(s)] done [0.00s].
```

```
## sorting and recoding items ... [30 item(s)] done [0.00s].
## creating transaction tree ... done [0.00s].
## checking subsets of size 1 2 3 4 5 6 7 8 done [0.01s].
## writing ... [9924 rule(s)] done [0.00s].
## creating S4 object  ... done [0.00s].
```

```
length(rules) #number of rules computed
```

```
## [1] 9924
```

3.3

A hypothetical scenario was considered in which a family was interested in moving to the area and would like to move as close to the city as possible conditional upon the area having low crime. This hypothetical scenario can be subsetted using the `rules` that were just created from the apriori algorithm in section 3.2. The distance measurements `boston$dis` and categorized as distance away from the city center (1 being the closest), the categories were `1-2`, `2-3`, `3-4`, `4-5`, and `>5`. The crime was measured in `boston$crim` and discretized into categories ranging from `safe`, `moderate`, and `dangerous`.

```
rulesLowCrimeDisLow <- subset(rules, subset = rhs %in% "crim=Safe" & lift>1)
```

A `lift >1` was chosen for this subset due to the fact that a lift greater than 1 is considered good association, and an increasing lift value from there on is considered stronger association. The subset under these conditions was investigated using the `arules::inspect` function.

As such, unfortunately for this hypothetical scenario, these results reveal that it is quite difficult to find a home located close to the city center when `boston$crim == 'safe'`. Nonetheless, the best results we could find suggest that with `support = 0.1304348`, `confidence = 0.8684211`, and `lift = 1.234329`, that the family could move within 3-4 weighted distances from the city center and live in a `crim == safe` neighbor. Moving any closer than that seems unlikely when considering the neighbor safety preferences

3.4

A hypothetical situation was considered in which a family was moving to the area and would only consider schools with a low pupil-teacher ratios (`boston$ptratio`). `boston$ptratio` was categorized into `below average`, `average`, and `above average`. The `rules` from section 3.2 were subsetted under the aforementioned preferences.

```
rulesLowPTRatio <- subset(rules, subset = rhs %in% "ptratio=Below Average" & lift > 1)
```

The parameters chosen for low pupil-teacher ratios subset from `rules` are based upon the same principles that were discussed in section 3.3. These rules were investigated using `arules::inspect` function. The results suggest that with `support = 0.06719368`, `confidence = 0.75555556`, and `lift = 2.711426`, this family could move into the area and find neighbors with low crime, low taxes, and houses with many rooms that fulfill their 'low' pupil-to-teacher ratio.