Module 6 CT Option 1

Scott Miner

3/21/2020

# Exploratory Data Analysis (EDA)

The assignment asks the student to download the **CourseTopics.csv** file, which contains eight courses in which a student could enroll. Each data row provides the courses taken (1 in the attribute) by a single student.

The student will:

* Create and interpret 3-5 association rules.
* Follow the process described in section 14.1 of the text.
* Use the R code example in Table 14.4 of the text.
* Use 0.01 for minimum support.
* Run the minimum confidence of 0.1 and 0.5 in the apriori() function.

We begin by reading the CSV file into a data frame, getting the dimensions of the data frame, printing the column names, descriptive statistics, and a sample of the first 20 records.

## Read the CSV File

# read the CSV file into a dataframe  
course.topics.df <- read.csv(params$file)

## Get Data Frame Dimensions

Get the dimensions of the data frame.

# get data frame dimensions  
autofit(set\_header\_labels(flextable(data.frame(cbind(  
 c("rows","columns"),  
 dim(course.topics.df)))),  
 X1 = "",  
 X2 = "count"))

|  | count |
| --- | --- |
| rows | 365 |
| columns | 8 |

We see the data frame has 365 rows and 8 columns.

## Print column names

Print the names of the columns.

# get the names of the columns  
autofit(set\_header\_labels(flextable(  
 data.frame(names(course.topics.df))),  
 names.course.topics.df. = "Course Offerings"))

| Course Offerings |
| --- |
| Intro |
| DataMining |
| Survey |
| Cat.Data |
| Regression |
| Forecast |
| DOE |
| SW |

## Generate Descriptive Statistics

The following code generates descriptive statistics for the data frame.

# generate descriptive statistics  
descRowNames <- colnames(course.topics.df)  
  
# get summary statistics  
autofit(set\_header\_labels(flextable(cbind(  
 descRowNames,  
 data.frame(mean = sapply(Filter(  
 is.numeric, course.topics.df), mean, na.rm = TRUE),  
 sd = sapply(Filter(is.numeric, course.topics.df),  
 sd, na.rm = TRUE),  
 min = sapply(Filter(is.numeric, course.topics.df),  
 min, na.rm = TRUE),  
 max = sapply(Filter(is.numeric, course.topics.df),  
 max, na.rm = TRUE),  
 median = sapply(Filter(is.numeric, course.topics.df),  
 median, na.rm = TRUE),  
 length = sapply(Filter(  
 is.numeric, course.topics.df),  
 length),  
 mis.val = sapply(Filter(  
 is.numeric, course.topics.df),  
 function(x) sum(is.na(x))),  
 cume.ratio = percent(cumsum(  
 sapply(Filter(is.numeric,course.topics.df),  
 function(x) mean(is.na(x)))))))),  
 descRowNames = "Column Header"))

| Column Header | mean | sd | min | max | median | length | mis.val | cume.ratio |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Intro | 0.3945205 | 0.4894184 | 0 | 1 | 0 | 365 | 0 | 0% |
| DataMining | 0.1780822 | 0.3831071 | 0 | 1 | 0 | 365 | 0 | 0% |
| Survey | 0.1863014 | 0.3898841 | 0 | 1 | 0 | 365 | 0 | 0% |
| Cat.Data | 0.2082192 | 0.4065918 | 0 | 1 | 0 | 365 | 0 | 0% |
| Regression | 0.2082192 | 0.4065918 | 0 | 1 | 0 | 365 | 0 | 0% |
| Forecast | 0.1397260 | 0.3471785 | 0 | 1 | 0 | 365 | 0 | 0% |
| DOE | 0.1726027 | 0.3784222 | 0 | 1 | 0 | 365 | 0 | 0% |
| SW | 0.2219178 | 0.4161066 | 0 | 1 | 0 | 365 | 0 | 0% |

## Print the first 20 records

Print first 20 records of the data frame.

# print the first 20 records of the data frame.  
autofit(flextable(head(course.topics.df, n = 20)))

| Intro | DataMining | Survey | Cat.Data | Regression | Forecast | DOE | SW |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |
| 0 | 1 | 0 | 1 | 1 | 0 | 0 | 1 |
| 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 1 | 0 | 1 | 1 | 1 |
| 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 |
| 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 1 | 1 | 0 | 0 | 1 | 0 | 0 |
| 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 |
| 1 | 0 | 1 | 1 | 0 | 1 | 1 | 1 |
| 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 |
| 1 | 0 | 0 | 0 | 0 | 1 | 0 | 0 |
| 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 |

# Create Association Rules

We will now begin the process of creating association rules. We will first convert the data frame to a matrix. Next, we convert the matrix to a transaction database. We then generate rules using the apriori() algorithm, with a support level of 0.01 and confidence levels of 0.1 and 0.5, respectively.

## Convert the data frame to a matrix

Convert the data frame to a matrix, examine the first 20 records of the matrix.

# converts the data frame to a matrix  
course.topics.mat <- as.matrix(course.topics.df)  
  
# examine first 20 rows of the matrix  
autofit(flextable(head(as.data.frame(course.topics.mat), n = 20)))

| Intro | DataMining | Survey | Cat.Data | Regression | Forecast | DOE | SW |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |
| 0 | 1 | 0 | 1 | 1 | 0 | 0 | 1 |
| 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 1 | 0 | 1 | 1 | 1 |
| 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 |
| 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 1 | 1 | 0 | 0 | 1 | 0 | 0 |
| 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 |
| 1 | 0 | 1 | 1 | 0 | 1 | 1 | 1 |
| 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 |
| 1 | 0 | 0 | 0 | 0 | 1 | 0 | 0 |
| 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 |

## Convert the binary incidence matrix into a transaction database

Next, we convert the binary incidence matrix into a transaction database.

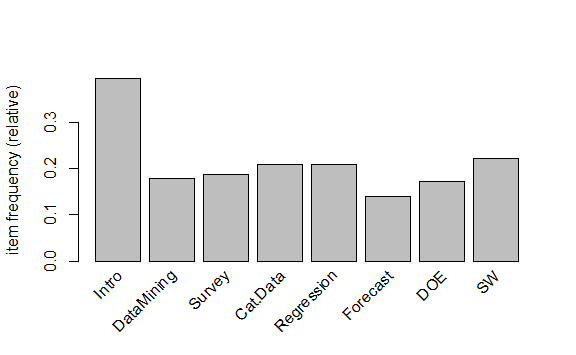
# converts the binary incidence matrix to a transaction database  
# requires package library(arules)  
course.topics.trans <- as(course.topics.mat, "transactions")  
  
# convert the transaction database to a data frame for flextable output  
df <- data.frame(  
 Items = labels(items(course.topics.trans))  
)  
  
# Console output, first 20 records  
# inspect(head(course.topics.trans, n = 20))  
  
# Flextable output, first 20 records  
autofit(flextable(head(df, 20)))

| Items |
| --- |
| {Intro,DataMining} |
| {Survey} |
| {DataMining,Cat.Data,Regression,SW} |
| {Intro} |
| {Intro,DataMining} |
| {DataMining} |
| {Intro} |
| {Cat.Data,Forecast,DOE,SW} |
| {Intro} |
| {Cat.Data} |
| {Intro} |
| {DataMining} |
| {DataMining} |
| {DataMining,Survey,Forecast} |
| {DataMining,Survey} |
| {Intro,Survey,Cat.Data,Forecast,DOE,SW} |
| {Intro,Regression} |
| {Intro,Forecast} |
| {Intro} |
| {Regression} |

## Plot the data

Now, we plot the relative item frequency of the data. The plot tells us that the Intro course occurs most often, and the Forecast class occurs least often.

itemFrequencyPlot(course.topics.trans)



## Generate Rules (Apriori algorithm)

Now, we will generate the rules from our dataset.

### Confidence = 0.1

We run the apriori() algorithm and set the minimum confidence to 0.1. We order the association rules by confidence levels from low to high.

# use 0.01 for minimum support, confidence = 0.1  
rules <- apriori(course.topics.trans,   
 parameter = list(supp = 0.01,  
 conf = 0.1, target = "rules"))

# convert to a dataframe to output as a flextable  
ruledf <- data.frame (  
 lhs = labels(lhs(rules)),  
 rhs = labels(rhs(rules)),  
 rules@quality  
)  
  
# output using the console  
# inspect the first six rules, sorted by their lift  
# inspect(head(sort(rules, by = "lift"), n = 6))  
  
# output as a flextable  
  
autofit(flextable(head(ruledf[order(ruledf$confidence),], 20)))

| lhs | rhs | support | confidence | lift | count |
| --- | --- | --- | --- | --- | --- |
| {Intro} | {DOE} | 0.04657534 | 0.1180556 | 0.6839727 | 17 |
| {Regression} | {Survey} | 0.02465753 | 0.1184211 | 0.6356424 | 9 |
| {Intro} | {Forecast} | 0.05205479 | 0.1319444 | 0.9443083 | 19 |
| {Survey} | {Regression} | 0.02465753 | 0.1323529 | 0.6356424 | 9 |
| {DataMining} | {DOE} | 0.02465753 | 0.1384615 | 0.8021978 | 9 |
| {Intro} | {DataMining} | 0.05479452 | 0.1388889 | 0.7799145 | 20 |
| {} | {Forecast} | 0.13972603 | 0.1397260 | 1.0000000 | 51 |
| {DOE} | {DataMining} | 0.02465753 | 0.1428571 | 0.8021978 | 9 |
| {Regression} | {DOE} | 0.03013699 | 0.1447368 | 0.8385547 | 11 |
| {Intro} | {Survey} | 0.06027397 | 0.1527778 | 0.8200572 | 22 |
| {Intro,Regression} | {Survey} | 0.01095890 | 0.1538462 | 0.8257919 | 4 |
| {DOE} | {Forecast} | 0.02739726 | 0.1587302 | 1.1360100 | 10 |
| {SW} | {Forecast} | 0.03561644 | 0.1604938 | 1.1486323 | 13 |
| {Survey} | {DataMining} | 0.03013699 | 0.1617647 | 0.9083710 | 11 |
| {DataMining} | {Survey} | 0.03013699 | 0.1692308 | 0.9083710 | 11 |
| {Intro,SW} | {Forecast} | 0.01643836 | 0.1714286 | 1.2268908 | 6 |
| {Intro,SW} | {DataMining} | 0.01643836 | 0.1714286 | 0.9626374 | 6 |
| {} | {DOE} | 0.17260274 | 0.1726027 | 1.0000000 | 63 |
| {SW} | {DataMining} | 0.03835616 | 0.1728395 | 0.9705603 | 14 |
| {DOE} | {Regression} | 0.03013699 | 0.1746032 | 0.8385547 | 11 |

### Confidence = 0.5

Next, we rerun the algorithm, this time with a minimum confidence of 0.5. We again rank the rules by confidence level, from low to high.

# use 0.01 for minimum support, confidence = 0.5  
rules <- apriori(course.topics.trans,  
 parameter = list(supp = 0.01,   
 conf = 0.5, target = "rules"))

# convert to a dataframe to output as a flextable  
ruledf <- data.frame (  
 lhs = labels(lhs(rules)),  
 rhs = labels(rhs(rules)),  
 rules@quality  
)  
  
# output using the console  
# inspect the first six rules, sorted by their lift  
# inspect(head(sort(rules, by = "lift"), n = 6))  
  
# output as a flextable  
autofit(flextable(head(ruledf[order(ruledf$confidence),], 20)))

| lhs | rhs | support | confidence | lift | count |
| --- | --- | --- | --- | --- | --- |
| {Forecast,DOE} | {Cat.Data} | 0.01369863 | 0.5000000 | 2.401316 | 5 |
| {Regression,Forecast} | {DataMining} | 0.01917808 | 0.5000000 | 2.807692 | 7 |
| {Cat.Data,Forecast} | {Survey} | 0.02191781 | 0.5000000 | 2.683824 | 8 |
| {Regression,Forecast} | {Intro} | 0.01917808 | 0.5000000 | 1.267361 | 7 |
| {Cat.Data,Forecast} | {Intro} | 0.02191781 | 0.5000000 | 1.267361 | 8 |
| {Survey,DOE} | {SW} | 0.01643836 | 0.5000000 | 2.253086 | 6 |
| {Cat.Data,Regression} | {DataMining} | 0.02739726 | 0.5000000 | 2.807692 | 10 |
| {Intro,DataMining} | {Regression} | 0.02739726 | 0.5000000 | 2.401316 | 10 |
| {Intro,DataMining,Regression} | {Forecast} | 0.01369863 | 0.5000000 | 3.578431 | 5 |
| {Intro,Survey,Cat.Data} | {Forecast} | 0.01369863 | 0.5000000 | 3.578431 | 5 |
| {Intro,Cat.Data,DOE} | {Survey} | 0.01095890 | 0.5000000 | 2.683824 | 4 |
| {Intro,Cat.Data,DOE} | {Regression} | 0.01095890 | 0.5000000 | 2.401316 | 4 |
| {Intro,Regression,SW} | {DOE} | 0.01917808 | 0.5000000 | 2.896825 | 7 |
| {Intro,Cat.Data,Regression} | {DataMining} | 0.01643836 | 0.5000000 | 2.807692 | 6 |
| {Intro,Survey,Cat.Data} | {SW} | 0.01369863 | 0.5000000 | 2.253086 | 5 |
| {DOE,SW} | {Intro} | 0.03013699 | 0.5238095 | 1.327712 | 11 |
| {Intro,DOE} | {Regression} | 0.02465753 | 0.5294118 | 2.542570 | 9 |
| {Cat.Data,DOE} | {SW} | 0.02465753 | 0.5294118 | 2.385621 | 9 |
| {Intro,Regression} | {SW} | 0.03835616 | 0.5384615 | 2.426401 | 14 |
| {DataMining,Survey} | {Cat.Data} | 0.01643836 | 0.5454545 | 2.619617 | 6 |

### Frequent Itemsets

To generate strong rules, we need to find all frequent itemsets. Frequent itemsets are those with high support.

# negate the in search function to exclude rules that have Intro as the consequent  
`%nin%` = Negate(`%in%`)  
  
# minimum support = 0.01, confidence = 0.5, minimum rule length = 2  
rules <- apriori(course.topics.trans,  
 parameter = list(supp = 0.01,   
 conf = 0.5, minlen=2, target = "rules"))  
  
# exclude rules that have Intro as the consequent  
rules <- subset(rules, rhs %nin% c("Intro"))

# convert to a dataframe to output as a flextable  
ruledf <- data.frame (  
 lhs = labels(lhs(rules)),  
 rhs = labels(rhs(rules)),  
 rules@quality  
)  
  
# set variables for highlighting  
vars <- c("ruledf$support", "ruledf$confidence")  
  
# set display and highlighting parameters  
autofit(set\_header\_labels(bg(flextable(  
 cbind(c(1:10),head(ruledf[order(-ruledf$count,  
 -ruledf$lift),], 10))),  
 i = ~ lift >= 1.7 & support > 0.027,  
 j = names(ruledf), bg = "light blue"),  
 'c(1:10)' = 'id'))

| id | lhs | rhs | support | confidence | lift | count |
| --- | --- | --- | --- | --- | --- | --- |
| 1 | {Intro,Regression} | {SW} | 0.03835616 | 0.5384615 | 2.426401 | 14 |
| 2 | {Intro,Survey} | {SW} | 0.03287671 | 0.5454545 | 2.457912 | 12 |
| 3 | {Intro,DOE} | {SW} | 0.03013699 | 0.6470588 | 2.915759 | 11 |
| 4 | {DataMining,Regression} | {Cat.Data} | 0.02739726 | 0.6250000 | 3.001645 | 10 |
| 5 | {Cat.Data,Regression} | {DataMining} | 0.02739726 | 0.5000000 | 2.807692 | 10 |
| 6 | {DataMining,Cat.Data} | {Regression} | 0.02739726 | 0.5555556 | 2.668129 | 10 |
| 7 | {Intro,DataMining} | {Regression} | 0.02739726 | 0.5000000 | 2.401316 | 10 |
| 8 | {Intro,DOE} | {Regression} | 0.02465753 | 0.5294118 | 2.542570 | 9 |
| 9 | {Cat.Data,DOE} | {SW} | 0.02465753 | 0.5294118 | 2.385621 | 9 |
| 10 | {Survey,Forecast} | {Cat.Data} | 0.02191781 | 0.5714286 | 2.744361 | 8 |

We filter for associations with a minimal confidence = 0.5 and sort the results by the count, which can be thought of as the support before we compute the ratio. Rules that meet a threshold (support > 0.027 and lift >= 1.7) are highlighted in blue.

* Rules 1 - 3 include the Intro course in the antecedent and the SW course in the consequent.
  + Highest support (3.8%): Rule 1: {Intro,Regression} => {SW}
  + Highest confidence (65%): Rule 3: {Intro,DOE} => {SW}
  + Highest lift (2.92): Rule 3: {Intro,DOE} => {SW}
* Rules 4 - 6 involve the same trio of courses with different antecedents and consequents and reduces the number of itemsets to be considered from a business perspective.
  + Of these rules, Rule 4: {Intro,DOE} => {SW} has the highest confidence (63%) and lift ratio (3.0).
  + The support amongst these itemsets are all equals (2.7%)

**Option #1: Course Selection Association Rules**

The assignment asks the student to create and interpret 3-5 association rules from the ***CourseTopics.csv*** file (*Module 6: Critical Thinking*, n.d.).The file contains 365 rows and eight columns. The titles of the columns are the classes for which students have registered: {Intro}, {DataMining}, {Survey}, {Cat.Data}, {Regression}, {Forecast}, {DOE}, and {SW}. The {Intro} itemset occurs most frequently, the {Forecast} itemset least. Itemsets with the highest *support* are those which occur most often. Support is equal to the probability of randomly selecting a transaction that contains both the *antecedent* and *consequent* portions of a rule. The antecedent refers to the IF portion of the rule, and the consequent the THEN portion. Itemset {Intro,Regression,SW} has the highest support (support = 0.038, count = 14). In analyzing the results, we obtain the following association rules, each with support greater than 0.027 (10 occurrences) and *lift* above 2.4: (a) “if Intro and Regression then SW,” (b) “if Intro and Survey then SW,” (c) “if Intro and DOE then SW,” (d) “if DataMining and Regression then Cat.Data,” and (e) “if Intro and DataMining then Regression.” Lift tells us how much more likely the consequent is given the antecedent in comparison to if the consequent and antecedent were independent. For instance, we can say a student is three times as more likely to be enrolled in Cat.Data when enrolled in DataMining and Regression than if Cat.Data was not associated with itemset {DataMining,Regression}. We can translate each rule into a sentence that provides information about performance. We translate rule (a) as follows, “If a student is enrolled in Intro and Regression, then with 53% confidence, he/she will be enrolled in SW.” Of the three rules with antecedent {Intro} and consequent {SW}, rule (c) has the highest confidence (65%) and lift ratio (2.92). These results tell us the student is 2.92 times as more likely to be enrolled in SW given itemset {Intro,DOE} in comparison to random chance (Shmueli et al., n.d.).

References

Module 6: Critical Thinking. (n.d.). Retrieved March 21, 2020, from https://csuglobal.instructure.com/courses/18808/assignments/382122

Shmueli, G., Bruce, P. C., Yahav, I., Patel, N. R., Lichtendahl, K. C., & Jr. (n.d.). Data Mining for Business Analytics. Retrieved from https://platform.virdocs.com/r/s/0/doc/503437/sp/21743577/mi/74416521?cfi=%2F4%2F2%2F6%2F18%2F14&menu=search&q=%7Bred%2Cwhite%7D