Grocery Management System

Sushil Tiwari Chiranjibi Ghimire

April 30, 2018

**Reading data**

test\_data <- read.csv("/Users/chirnjibi/Desktop/test\_data\_project.csv")

**Framing Data**

head(test\_data)

## item Month Date storeName description merchandise.category  
## 1 1 March 3/30/2018 Columbia store Red onions Produce  
## 2 2 March 3/30/2018 Columbia store Tomato Produce  
## 3 3 March 3/30/2018 Columbia store Red Potato Produce  
## 4 4 March 3/30/2018 Columbia store Green chili Produce  
## 5 5 March 3/30/2018 Columbia store Garlic Produce  
## 6 6 March 3/30/2018 Columbia store Spinich Produce  
## quantity lbs amount classification  
## 1 1 2 1.99 Need  
## 2 1 3.07 4.57 Need  
## 3 1 2.06 2.04 Need  
## 4 1 0.23 2.49 Need  
## 5 1 null 0.85 Need  
## 6 1 null 2.49 Need

Since the data was prepared by ourselves, we don’t have to scale the data. Our data are in range. This data collection is based on real shopping activity over three months of period.

# Data Visualization

install.packages("ggplot2", dependencies = TRUE, repos = "http://cran.uk.r-project.org/src/contrib/ggplot2\_0.8.9.tar.gz")

## Installing package into 'C:/Users/chirnjibi/Documents/R/win-library/3.4'  
## (as 'lib' is unspecified)

## Warning: unable to access index for repository http://cran.uk.r-project.org/src/contrib/ggplot2\_0.8.9.tar.gz/src/contrib:  
## cannot open URL 'http://cran.uk.r-project.org/src/contrib/ggplot2\_0.8.9.tar.gz/src/contrib/PACKAGES'

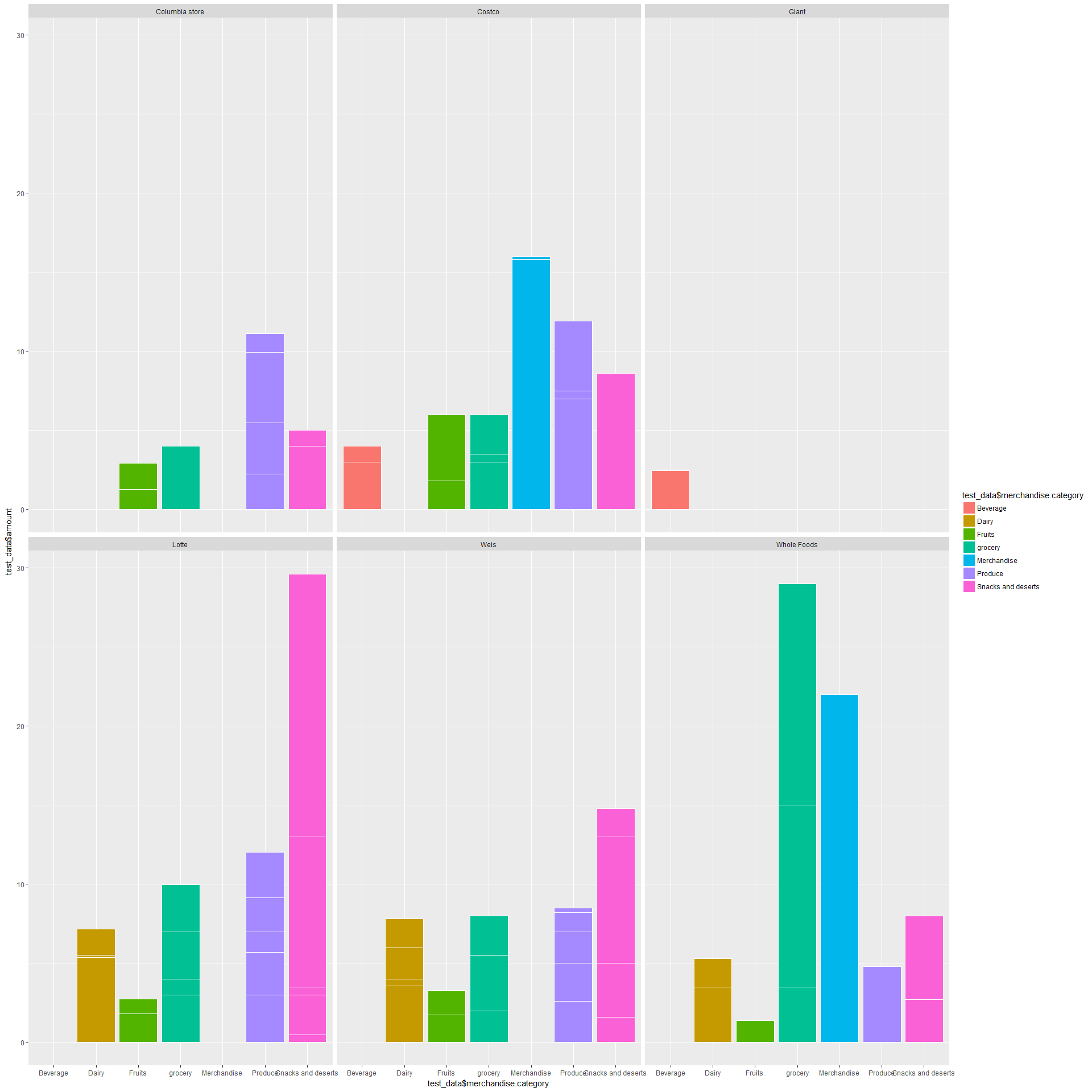
## Warning: package 'ggplot2' is not available (for R version 3.4.3)

## Warning: unable to access index for repository http://cran.uk.r-project.org/src/contrib/ggplot2\_0.8.9.tar.gz/bin/windows/contrib/3.4:  
## cannot open URL 'http://cran.uk.r-project.org/src/contrib/ggplot2\_0.8.9.tar.gz/bin/windows/contrib/3.4/PACKAGES'

library("ggplot2")

## Warning: package 'ggplot2' was built under R version 3.4.4

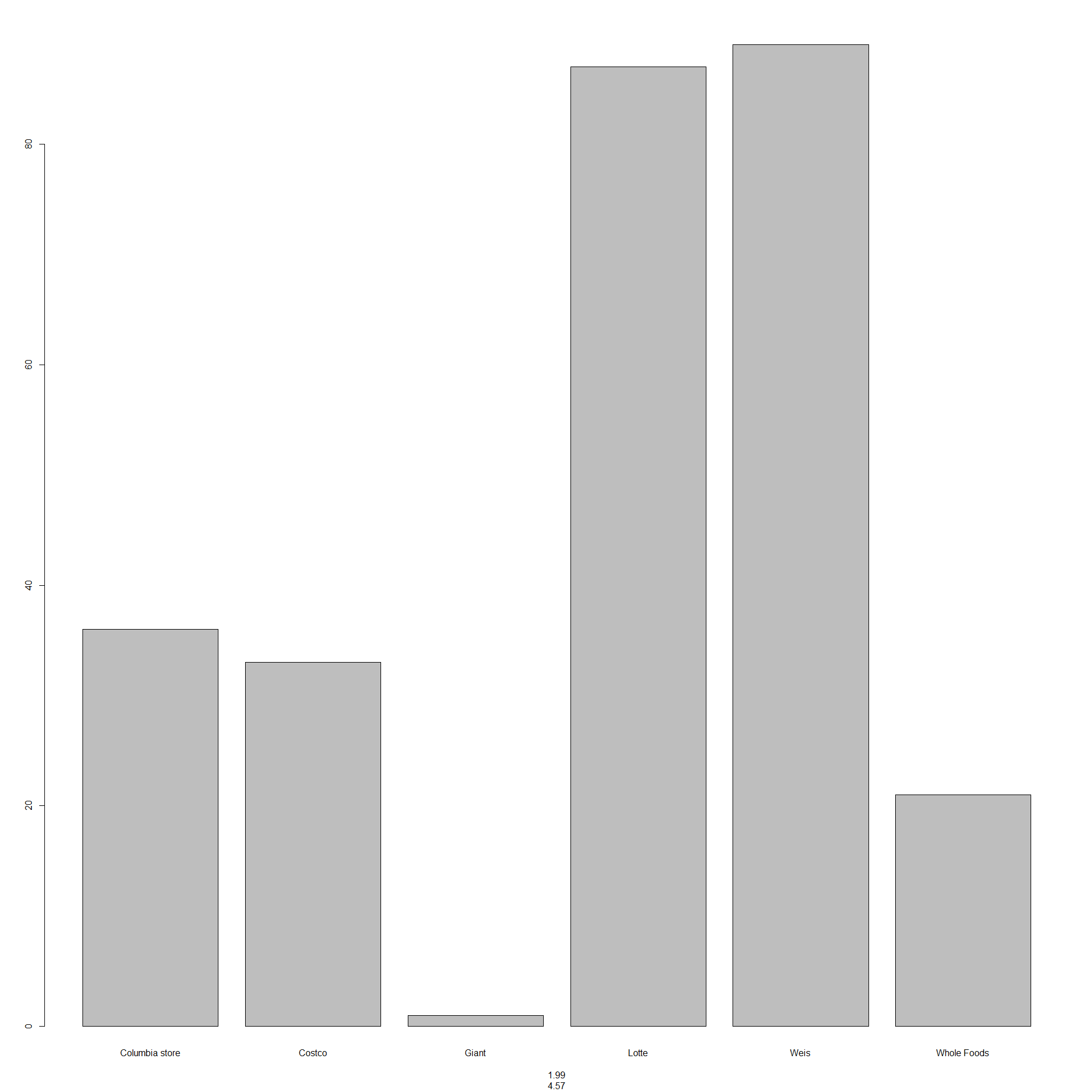
ggplot(test\_data, aes(x = test\_data$merchandise.category, y = test\_data$amount))+ geom\_bar(aes(fill = test\_data$merchandise.category), stat = "identity" , color = "white", position = position\_dodge(0.9)) + facet\_wrap(~ test\_data$storeName)



attach(test\_data)

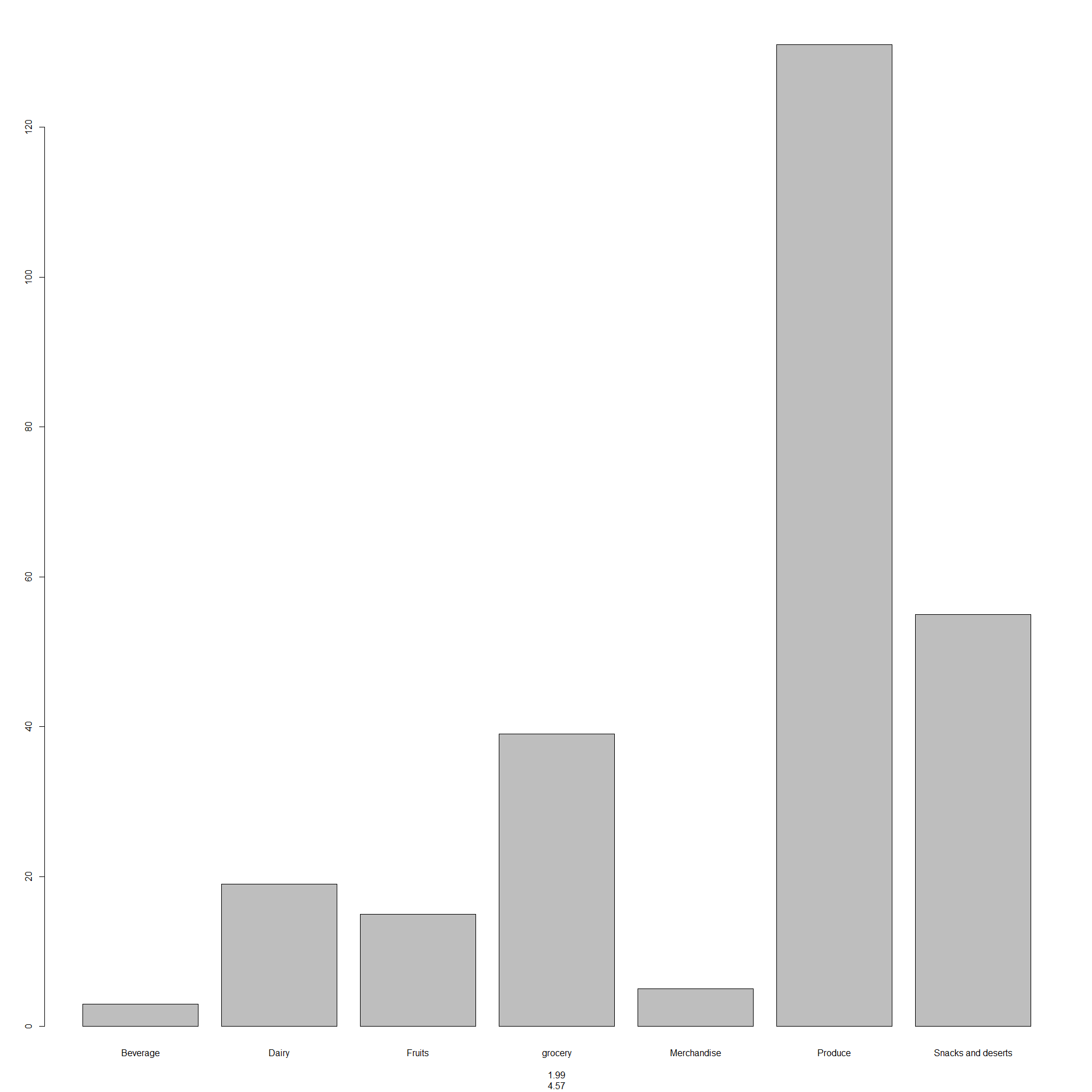
# Barplot by storename

store\_count <- table(storeName)  
barplot(store\_count, xlab = amount)



# Barplot by Merchandise category

merchandise\_category\_count <- table(merchandise.category)  
barplot(merchandise\_category\_count, xlab = amount)



From the above diagrams, the money being spends much in produce category. Similarly, the more money is spending in Lotte and Weis store.

# Clustering

newdataProject <- test\_data[,2:7]  
head(newdataProject)

## Month Date storeName description merchandise.category quantity  
## 1 March 3/30/2018 Columbia store Red onions Produce 1  
## 2 March 3/30/2018 Columbia store Tomato Produce 1  
## 3 March 3/30/2018 Columbia store Red Potato Produce 1  
## 4 March 3/30/2018 Columbia store Green chili Produce 1  
## 5 March 3/30/2018 Columbia store Garlic Produce 1  
## 6 March 3/30/2018 Columbia store Spinich Produce 1

Adding amount to the dataset

newdataProject$amount <- test\_data$amount  
head(newdataProject)

## Month Date storeName description merchandise.category quantity  
## 1 March 3/30/2018 Columbia store Red onions Produce 1  
## 2 March 3/30/2018 Columbia store Tomato Produce 1  
## 3 March 3/30/2018 Columbia store Red Potato Produce 1  
## 4 March 3/30/2018 Columbia store Green chili Produce 1  
## 5 March 3/30/2018 Columbia store Garlic Produce 1  
## 6 March 3/30/2018 Columbia store Spinich Produce 1  
## amount  
## 1 1.99  
## 2 4.57  
## 3 2.04  
## 4 2.49  
## 5 0.85  
## 6 2.49

Taking numeric column in an object for clustering

testnumdata <- newdataProject[,6:7]  
 head(testnumdata)

## quantity amount  
## 1 1 1.99  
## 2 1 4.57  
## 3 1 2.04  
## 4 1 2.49  
## 5 1 0.85  
## 6 1 2.49

Creating a cluster object

numeric\_cluster <- **kmeans**(testnumdata, center=5)

Size of each cluster

numeric\_cluster**$**size

## [1] 106   9  12  49  91

We can see, cluster 1 has 106 item, cluster 2 has 9, 3 has 12 , 4 with 49 and 5 with 91 items in it.

Getting the data in a new object so that it can be modified

testdata\_one <- newdataProject

Adding cluster\_id column to the newly created data object.

testdata\_one**$**cluster\_id <- numeric\_cluster**$**cluster  
**head**(testdata\_one)

##   Month      Date      storeName description merchandise.category quantity  
## 1 March 3/30/2018 Columbia store  Red onions              Produce        1  
## 2 March 3/30/2018 Columbia store      Tomato              Produce        1  
## 3 March 3/30/2018 Columbia store  Red Potato              Produce        1  
## 4 March 3/30/2018 Columbia store Green chili              Produce        1  
## 5 March 3/30/2018 Columbia store      Garlic              Produce        1  
## 6 March 3/30/2018 Columbia store     Spinich              Produce        1  
##   amount cluster\_id  
## 1   1.99          5  
## 2   4.57          1  
## 3   2.04          5  
## 4   2.49          5  
## 5   0.85          5  
## 6   2.49          5

Checking which fell under what

Which merchandise category fell under which cluster

**table**(testdata\_one**$**merchandise.category, testdata\_one**$**cluster\_id)

##                       
##                       1  2  3  4  5  
##   Beverage            2  0  0  0  1  
##   Dairy              14  0  0  4  1  
##   Fruits              3  0  0  1 11  
##   grocery            19  3  0  9  8  
##   Merchandise         0  4  0  1  0  
##   Produce            42  0 12 25 52  
##   Snacks and deserts 26  2  0  9 18

As we can see, Beverage fell on category 1 and 5 while almost all dairy product fell under cluster 1 except for one. Fruits as we can see fell mostly on category 5. Grocery on 1, Merchandise on 2, Produce is the one which has been divided but majority of produce were categorized under cluster 1 and 5. Similarly majority of snacks and deserts were under 1 and 5

Which store fell under which cluster

**table**(testdata\_one**$**storeName, testdata\_one**$**cluster\_id)

##                   
##                   1  2  3  4  5  
##   Columbia store 17  0  0  3 16  
##   Costco          4  9  1 14  5  
##   Giant           0  0  0  1  0  
##   Lotte          37  0  2 11 37  
##   Weis           38  0  9 12 30  
##   Whole Foods    10  0  0  8  3

If we check the cluster based on the store that we usually shop, we can see that Columbia store falls under category 1 and 5. Majority of items from Costco was categorized under cluster 4.  Items from Lotte were equally divided into cluster 1 and 5. Whole foods are categorized under 1 cluster.

**table**(testdata\_one**$**description, testdata\_one**$**cluster\_id)

##                     
##                     1  2  3  4  5  
##   Allo tikki roll   0  0  0  0  1  
##   Avocado           0  0  5  2  1  
##   Bamboo shoot      0  0  0  1  0  
##   Banana            1  0  0  0 10  
##   Bath Tissue       0  2  0  0  0  
##   Bean burger       0  2  0  0  0  
##   Beans             1  0  0  0  0  
##   Beet Root         1  0  0  0  1  
##   Bitter melon      0  0  0  0  1  
##   Blue berries      1  0  0  0  0  
##   Broccoli          1  0  0  0  2  
##   Butter            1  0  0  0  0  
##   Cabbage           1  0  0  0  2  
##   Cardamom          0  0  0  1  0  
##   cauliflower       5  0  0  0  1  
##   Cereal            3  0  0  0  0  
##   Chilli Sauce      2  0  0  0  0  
##   Chocolate Milk    2  0  0  0  0  
##   Cilantro          0  0  1  0  6  
##   coffee            1  0  0  0  0  
##   Cookies           0  0  0  0  1  
##   Corns             3  0  0  0  0  
##   Cracker           0  0  0  2  2  
##   Cucumber          0  0  1  0  3  
##   Cumin powder      1  0  0  0  0  
##   Dish washer       0  0  0  1  0  
##   Dumpling Wrapper  0  0  0  1  3  
##   Egg Plant         0  0  0  0  4  
##   Eggs              2  0  0  0  0  
##   Flour             0  0  0  1  0  
##   Garlic            0  0  0  1  1  
##   Ginger            1  0  0  0  4  
##   Grapes            1  0  0  2  0  
##   Green Cabbage     0  0  0  0  1  
##   Green chili       0  0  0  0  1  
##   Green chilli      0  0  0  0  2  
##   Green Peas        0  0  0  3  0  
##   Green Peppers     0  0  0  0  2  
##   Gulab jamun       1  0  0  0  0  
##   Half and half     1  0  0  0  0  
##   Ham Rolls         0  0  0  0  2  
##   Hot dog roll      0  0  0  0  1  
##   Hummus            0  0  0  1  0  
##   Jalapeno          1  0  0  0  4  
##   Juice             3  0  0  0  5  
##   Kaju katli        1  0  0  0  0  
##   Kimchi            0  0  0  1  0  
##   Lemon Juice       0  0  0  0  1  
##   Lentle            1  0  0  2  0  
##   lettuce           0  0  0  0  2  
##   Limes             0  0  5  1  1  
##   loki              1  0  0  0  0  
##   Loki              1  0  0  0  0  
##   Masala noodles    4  0  0  2  0  
##   Milk              9  0  0  0  1  
##   Mixed Veggies     0  0  0  1  0  
##   Mushrooms         4  0  0  2  2  
##   Nice Biscuits     3  0  0  0  0  
##   Okra              0  0  0  0  4  
##   Orange            0  0  0  1  0  
##   Pancake mix       0  0  0  0  1  
##   Paneer            0  0  0  0  1  
##   Pani poori        1  0  0  0  0  
##   Papaya            1  0  0  0  1  
##   Paper Towels      0  2  0  0  0  
##   Paratha           0  0  0  1  0  
##   Pasta             0  0  0  0  1  
##   Peanut butter     1  0  0  1  0  
##   Peas              0  0  0  1  0  
##   pickle            1  0  0  0  0  
##   Pomegranate       0  0  0  1  0  
##   potato roll       0  0  0  0  1  
##   potato sandwic    1  0  0  0  0  
##   Puffed Rice       0  0  0  0  2  
##   pumpkin           0  0  0  0  2  
##   Quinoa            0  1  0  0  0  
##   Red onions        5  0  0  1  2  
##   Red Potato        6  0  0  0  2  
##   Rice              0  1  0  0  0  
##   Roasted Chana     1  0  0  0  0  
##   Salsa             1  0  0  1  2  
##   Salt              0  0  0  0  1  
##   Sandwich cooki    1  0  0  0  0  
##   Sandwich cookie   1  0  0  0  0  
##   Soy Milk          1  0  0  3  0  
##   Spinich           1  0  0  0  1  
##   Squash            1  0  0  2  0  
##   Sugar             2  0  0  0  0  
##   Syrup             1  0  0  0  0  
##   Tea               1  0  0  1  0  
##   Tilapia           0  1  0  0  0  
##   Tofu              0  0  0  0  1  
##   Tomato            9  0  0  7  0  
##   Tumeric powder    1  0  0  0  0  
##   Vegan beef        1  0  0  0  0  
##   Vegan burger      0  0  0  1  0  
##   Vegan chicken     1  0  0  0  0  
##   Vegan Sausage     0  0  0  2  0  
##   Waffle            1  0  0  0  0  
##   wheat bread       7  0  0  0  1  
##   Yoghurt           1  0  0  1  0

Similarly if we go by the item itself we can see that we can see that Aloo tikki roll falls under cluster 5 , If we look to at banana, we can see that almost all the bananas purchase except for 1 fell onto cluster 5 whereas 1 is on cluster 1

**table**(testdata\_one**$**amount, testdata\_one**$**cluster\_id)

##          
##          1  2  3  4  5  
##   0.16   0  0  0  0  1  
##   0.4    0  0  0  0  1  
##   0.49   0  0  0  0  2  
##   0.64   0  0  0  0  1  
##   0.66   0  0  0  0  2  
##   0.79   0  0  0  0  4  
##   0.84   0  0  0  0  1  
##   0.85   0  0  0  0  2  
##   0.87   0  0  0  0  1  
##   0.88   0  0  0  0  1  
##   0.98   0  0  0  0  1  
##   0.99   0  0  1  0  0  
##   1.09   0  0  0  0  2  
##   1.11   0  0  0  0  1  
##   1.17   0  0  0  0  1  
##   1.21   0  0  0  0  1  
##   1.26   0  0  0  0  1  
##   1.29   0  0  0  0  3  
##   1.31   0  0  0  0  1  
##   1.34   0  0  0  0  1  
##   1.39   0  0  0  0  3  
##   1.48   0  0  0  0  1  
##   1.49   0  0  0  0  8  
##   1.59   0  0  0  0  3  
##   1.64   0  0  0  0  1  
##   1.69   0  0  0  0  1  
##   1.72   0  0  0  0  1  
##   1.77   0  0  0  0  2  
##   1.79   0  0  0  0  1  
##   1.8    0  0  0  0  1  
##   1.82   0  0  0  0  1  
##   1.9    0  0  0  0  1  
##   1.91   0  0  0  0  1  
##   1.99   0  0  1  0 10  
##   2      0  0  0  0  3  
##   2.02   0  0  0  0  1  
##   2.04   0  0  0  0  1  
##   2.07   0  0  0  0  1  
##   2.17   0  0  0  0  1  
##   2.24   0  0  0  0  1  
##   2.29   0  0  0  0  1  
##   2.31   0  0  0  0  1  
##   2.39   0  0  0  0  1  
##   2.44   0  0  0  0  1  
##   2.47   0  0  0  0  1  
##   2.49   0  0  0  0  4  
##   2.5    0  0  0  0  4  
##   2.58   0  0  0  0  2  
##   2.59   0  0  0  0  2  
##   2.65   0  0  0  0  1  
##   2.69   0  0  0  0  2  
##   2.74   1  0  0  0  0  
##   2.78   1  0  0  0  0  
##   2.79   3  0  0  0  0  
##   2.82   1  0  0  0  0  
##   2.91   1  0  0  0  0  
##   2.93   1  0  0  0  0  
##   2.99  24  0  0  0  0  
##   3      1  0  0  0  0  
##   3.09   0  0  1  0  0  
##   3.16   0  0  1  0  0  
##   3.28   1  0  0  0  0  
##   3.29   1  0  0  0  0  
##   3.42   1  0  0  0  0  
##   3.49  15  0  1  0  0  
##   3.51   1  0  0  0  0  
##   3.55   1  0  0  0  0  
##   3.58   1  0  0  0  0  
##   3.69   3  0  1  0  0  
##   3.75   0  0  1  0  0  
##   3.79   1  0  0  0  0  
##   3.87   1  0  0  0  0  
##   3.89   1  0  0  0  0  
##   3.99  14  0  0  0  0  
##   4      3  0  0  0  0  
##   4.29   1  0  0  0  0  
##   4.4    0  0  1  0  0  
##   4.56   1  0  0  0  0  
##   4.57   1  0  0  0  0  
##   4.78   1  0  0  0  0  
##   4.81   2  0  0  0  0  
##   4.9    1  0  0  0  0  
##   4.96   1  0  0  0  0  
##   4.98   2  0  0  0  0  
##   4.99   5  0  0  0  0  
##   5      3  0  1  0  0  
##   5.21   2  0  0  0  0  
##   5.29   1  0  0  0  0  
##   5.37   1  0  0  0  0  
##   5.49   4  0  0  0  0  
##   5.64   1  0  0  0  0  
##   5.67   2  0  0  0  0  
##   5.79   0  0  0  1  0  
##   5.98   0  0  0  2  0  
##   5.99   0  0  2  7  0  
##   6.14   0  0  0  1  0  
##   6.18   0  0  0  1  0  
##   6.19   0  0  0  2  0  
##   6.45   0  0  1  0  0  
##   6.47   0  0  0  1  0  
##   6.49   0  0  0  2  0  
##   6.59   0  0  0  1  0  
##   6.87   0  0  0  2  0  
##   6.9    0  0  0  1  0  
##   6.98   0  0  0  1  0  
##   6.99   0  0  0  7  0  
##   7.15   0  0  0  1  0  
##   7.49   0  0  0  1  0  
##   7.69   0  0  0  1  0  
##   7.8    0  0  0  1  0  
##   7.96   0  0  0  1  0  
##   7.99   0  0  0  3  0  
##   8.22   0  0  0  1  0  
##   8.49   0  0  0  1  0  
##   8.59   0  0  0  1  0  
##   8.79   0  0  0  1  0  
##   9.14   0  0  0  1  0  
##   9.95   0  0  0  1  0  
##   9.98   0  0  0  1  0  
##   11.12  0  0  0  1  0  
##   11.92  0  0  0  1  0  
##   12.01  0  0  0  1  0  
##   12.99  0  0  0  2  0  
##   14.79  0  1  0  0  0  
##   14.99  0  1  0  0  0  
##   15.79  0  1  0  0  0  
##   15.99  0  2  0  0  0  
##   21.99  0  2  0  0  0  
##   28.99  0  1  0  0  0  
##   29.58  0  1  0  0  0

If we look deeply into the classification of amount we can see that, majority of items costing less that 2.69 fell under category 5.  Item costing more than 14 dollars falls under category 1.

# Need to change all the categorical columns value to a dummy variable

We can have something with numbers representing the categories. We can have multiple clustering one with stores and other with the types of product.

Association analysis for particular date of receipt.

Data prepared using java public static ArrayList myDataList() throws Exception{

ArrayList<String> mydataList = new ArrayList<String>();  
 Set<String> myRecieptList = new HashSet<String>();  
 File file = new File("C:\\Users\\t-Stiwari\\Desktop\\testdata.xlsx");  
 FileInputStream filestream = new FileInputStream(file);  
 // FileInputStream fis = ConfigurationManager.loadFileInputStream("C:\\Users\\t-Stiwari\\Desktop\\testdata.xlsx");  
  
 XSSFWorkbook xssfWorkbook = new XSSFWorkbook(filestream);  
 XSSFSheet mySheet = xssfWorkbook.getSheetAt(0);  
 int startRowIndex = mySheet.getFirstRowNum();  
 int endRow = mySheet.getLastRowNum();  
   
   
   
 String date=null;  
 String description = null;  
 try{  
 for(int m=startRowIndex ; m< endRow; m++){  
 XSSFRow row = mySheet.getRow(m);  
 myRecieptList.add(row.getCell(2).getStringCellValue());  
 System.out.println(row.getCell(2).getStringCellValue());  
 System.out.println(row.getCell(4).getStringCellValue());  
 }  
   
 File csvfile = new File("C:\\Users\\t-Stiwari\\Desktop\\testdata.csv");  
 FileWriter fw = new FileWriter(csvfile);  
 BufferedWriter bw = new BufferedWriter(fw);  
 for(String s: myRecieptList){  
 for(int m=startRowIndex ; m< endRow; m++){  
 XSSFRow row = mySheet.getRow(m);  
   
 if(!csvfile.exists()){  
 csvfile.createNewFile();  
 }  
   
   
 if(s.equalsIgnoreCase(row.getCell(2).getStringCellValue())){  
 bw.write(row.getCell(4).getStringCellValue()+",");  
 }  
   
 }  
 bw.newLine();  
 }  
 bw.flush();  
 bw.close();  
   
   
   
 }catch(Exception e){  
 throw e;  
 }  
   
 return null;  
}

# Association Analysis

installed.packages("arules")

## Package LibPath Version Priority Depends Imports LinkingTo Suggests  
## Enhances License License\_is\_FOSS License\_restricts\_use OS\_type Archs  
## MD5sum NeedsCompilation Built

library(arules)

## Warning: package 'arules' was built under R version 3.4.4

## Loading required package: Matrix

##   
## Attaching package: 'arules'

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

Read the data

mydataTest= read.transactions("/Users/chirnjibi/Desktop/AssociationAnalysisData.csv")

inspect(head(mydataTest,3))

## items   
## [1] {bread,Tomato,Avocado,Jalapeno,Mushrooms,Corns,,,,,,,,,,,,,,,,,,,,,,,,,,,   
## wheat}   
## [2] {bread,Tomato,Cilantro,Green,   
## Cereal,Cereal,Salsa,Salsa,Pasta,wheat,   
## Milk,,,,,,,,,,,,,   
## Peppers,Broccoli,Pomegranate,Squash,Banana,Avocado,Limes,Grapes,Cucumber,cauliflower,Squash,Soy}  
## [3] {Milk,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,   
## Soy}

Find out most frequent items

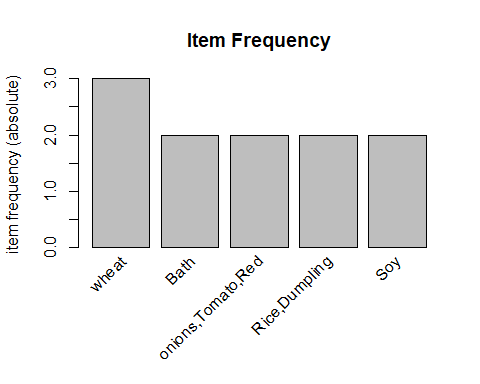
frequentItems = eclat (mydataTest, parameter = list(supp = 0.1, maxlen = 10))

## Eclat  
##   
## parameter specification:  
## tidLists support minlen maxlen target ext  
## FALSE 0.1 1 10 frequent itemsets FALSE  
##   
## algorithmic control:  
## sparse sort verbose  
## 7 -2 TRUE  
##   
## Absolute minimum support count: 2   
##   
## create itemset ...   
## set transactions ...[113 item(s), 21 transaction(s)] done [0.00s].  
## sorting and recoding items ... [1 item(s)] done [0.00s].  
## creating bit matrix ... [1 row(s), 21 column(s)] done [0.02s].  
## writing ... [1 set(s)] done [0.00s].  
## Creating S4 object ... done [0.00s].

inspect(frequentItems)

## items support count  
## [1] {wheat} 0.1428571 3

itemFrequencyPlot(mydataTest, topN=5, type="absolute", main="Item Frequency")



Create association rules

rules = apriori (mydataTest, parameter = list(supp = 0.01, conf = 0.05))

## Apriori  
##   
## Parameter specification:  
## confidence minval smax arem aval originalSupport maxtime support minlen  
## 0.05 0.1 1 none FALSE TRUE 5 0.01 1  
## maxlen target ext  
## 10 rules FALSE  
##   
## Algorithmic control:  
## filter tree heap memopt load sort verbose  
## 0.1 TRUE TRUE FALSE TRUE 2 TRUE  
##   
## Absolute minimum support count: 0   
##   
## set item appearances ...[0 item(s)] done [0.00s].  
## set transactions ...[113 item(s), 21 transaction(s)] done [0.00s].  
## sorting and recoding items ... [113 item(s)] done [0.00s].  
## creating transaction tree ... done [0.00s].  
## checking subsets of size 1 2 3 4 5 6 7 8 9 10

## Warning in apriori(mydataTest, parameter = list(supp = 0.01, conf =  
## 0.05)): Mining stopped (maxlen reached). Only patterns up to a length of 10  
## returned!

## done [0.05s].  
## writing ... [983554 rule(s)] done [0.29s].  
## creating S4 object ... done [1.02s].

High confidence rule

rules\_conf = sort(rules, by="confidence", decreasing=TRUE)  
inspect(head(rules\_conf))

## lhs rhs support confidence lift count  
## [1] {bread,Tomato,Avocado,Jalapeno,Mushrooms,Corns,,,,,,,,,,,,,,,,,,,,,,,,,,} => {wheat} 0.04761905 1 7.0 1  
## [2] {Potato,Spinich,Okra,lettuce,cauliflower,loki,,,,,,,,,,,,,,,,,,,,,,,,} => {Tomato,Milk,Red} 0.04761905 1 21.0 1  
## [3] {Tomato,Milk,Red} => {Potato,Spinich,Okra,lettuce,cauliflower,loki,,,,,,,,,,,,,,,,,,,,,,,,} 0.04761905 1 21.0 1  
## [4] {Milk,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,} => {Soy} 0.04761905 1 10.5 1  
## [5] {bread,Ginger,Tomato,Milk,,,,,,,,,,,,,,,,,,,,,,,,,,,} => {Juice,wheat} 0.04761905 1 21.0 1  
## [6] {Juice,wheat} => {bread,Ginger,Tomato,Milk,,,,,,,,,,,,,,,,,,,,,,,,,,,} 0.04761905 1 21.0 1

The rules with confidence of 1 (see above) imply that, whenever the LHS item was purchased, the RHS item was also purchased 100% of the time.

10 rules with highest lift: to get strong rules

rules.sortedLift = head(sort(rules, by ="lift"), 10)  
inspect(head(rules.sortedLift))

## lhs rhs support confidence lift count  
## [1] {Potato,Spinich,Okra,lettuce,cauliflower,loki,,,,,,,,,,,,,,,,,,,,,,,,} => {Tomato,Milk,Red} 0.04761905 1 21 1  
## [2] {Tomato,Milk,Red} => {Potato,Spinich,Okra,lettuce,cauliflower,loki,,,,,,,,,,,,,,,,,,,,,,,,} 0.04761905 1 21 1  
## [3] {bread,Ginger,Tomato,Milk,,,,,,,,,,,,,,,,,,,,,,,,,,,} => {Juice,wheat} 0.04761905 1 21 1  
## [4] {Juice,wheat} => {bread,Ginger,Tomato,Milk,,,,,,,,,,,,,,,,,,,,,,,,,,,} 0.04761905 1 21 1  
## [5] {onions,Milk,Juice,,,,,,,,,,,,,,,,,,,,} => {bread,Cilantro,Banana,Limes,Tomato,Jalapeno,Squash,Avocado,Tomato,Red} 0.04761905 1 21 1  
## [6] {bread,Cilantro,Banana,Limes,Tomato,Jalapeno,Squash,Avocado,Tomato,Red} => {onions,Milk,Juice,,,,,,,,,,,,,,,,,,,,} 0.04761905 1 21 1

A rule with a lift of 21 (see rules.sortedLife above) implies that, the items in LHS and RHS are 21 times more likely to be purchased together compared to the purchases when they are assumed to be unrelated.

# Next find the rules related to given items (products)

To find out what we had purchased before buying ‘Wheat’. This will help us understand the patterns that led to the purchase of ‘wheat’.

rules <- apriori (data=mydataTest, parameter=list (supp=0.01,conf = 0.05), appearance = list (default="lhs",rhs="wheat"), control = list (verbose=F))

## Warning in apriori(data = mydataTest, parameter = list(supp = 0.01, conf =  
## 0.05), : Mining stopped (maxlen reached). Only patterns up to a length of  
## 10 returned!

rules\_conf <- sort (rules, by="confidence", decreasing=TRUE)  
inspect(head(rules\_conf))

## lhs rhs support confidence lift count  
## [1] {bread,Tomato,Avocado,Jalapeno,Mushrooms,Corns,,,,,,,,,,,,,,,,,,,,,,,,,,} => {wheat} 0.04761905 1 7 1  
## [2] {onions,Milk,Juice,,,,,,,,,,,,,,,,,,,,} => {wheat} 0.04761905 1 7 1  
## [3] {bread,Cilantro,Banana,Limes,Tomato,Jalapeno,Squash,Avocado,Tomato,Red} => {wheat} 0.04761905 1 7 1  
## [4] {bread,Puffed} => {wheat} 0.04761905 1 7 1  
## [5] {Cabbage,Cumin} => {wheat} 0.04761905 1 7 1  
## [6] {chilli,Sugar,Bamboo} => {wheat} 0.04761905 1 7 1

To find out what products were purchased after/along with product ‘wheat’. This is a case to find out when we bought ‘Wheat’ also bought…

rules <- apriori (data=mydataTest, parameter=list (supp=0.01,conf = 0.05,minlen=2), appearance = list(default="rhs",lhs="wheat"), control = list(verbose=F))  
rules\_conf <- sort (rules, by="confidence", decreasing=TRUE)  
inspect(head(rules\_conf))

## lhs rhs support confidence lift count  
## [1] {wheat} => {bread,Tomato,Avocado,Jalapeno,Mushrooms,Corns,,,,,,,,,,,,,,,,,,,,,,,,,,} 0.04761905 0.3333333 7 1  
## [2] {wheat} => {onions,Milk,Juice,,,,,,,,,,,,,,,,,,,,} 0.04761905 0.3333333 7 1  
## [3] {wheat} => {bread,Cilantro,Banana,Limes,Tomato,Jalapeno,Squash,Avocado,Tomato,Red} 0.04761905 0.3333333 7 1  
## [4] {wheat} => {bread,Puffed} 0.04761905 0.3333333 7 1  
## [5] {wheat} => {Cabbage,Cumin} 0.04761905 0.3333333 7 1  
## [6] {wheat} => {chilli,Sugar,Bamboo} 0.04761905 0.3333333 7 1

# Neural Network

library("nnet")  
library("caret")

## Warning: package 'caret' was built under R version 3.4.4

## Loading required package: lattice

Splitting data to have less weight, will all the columns are throws weight error

data\_ <- test\_data  
split\_data <- data\_[,4:10]  
head(split\_data)

## storeName description merchandise.category quantity lbs amount  
## 1 Columbia store Red onions Produce 1 2 1.99  
## 2 Columbia store Tomato Produce 1 3.07 4.57  
## 3 Columbia store Red Potato Produce 1 2.06 2.04  
## 4 Columbia store Green chili Produce 1 0.23 2.49  
## 5 Columbia store Garlic Produce 1 null 0.85  
## 6 Columbia store Spinich Produce 1 null 2.49  
## classification  
## 1 Need  
## 2 Need  
## 3 Need  
## 4 Need  
## 5 Need  
## 6 Need

Preparing data for training and testing set

set.seed(50005)  
test\_nnet\_sample <- createDataPartition(split\_data$classification, p=.70, list = FALSE)  
project\_traindata <- split\_data[test\_nnet\_sample,]  
project\_test\_data<- split\_data  
  
project\_test\_data<- split\_data[-test\_nnet\_sample,]

Running neural Network

neuralnet\_p <- nnet(classification ~ ., data = project\_traindata, size = 6)

## # weights: 895  
## initial value 194.689116   
## iter 10 value 6.663517  
## iter 20 value 0.009421  
## final value 0.000072   
## converged

Analyzing the model

summary(neuralnet\_p)

## a 147-6-1 network with 895 weights  
## options were - entropy fitting   
## b->h1 i1->h1 i2->h1 i3->h1 i4->h1 i5->h1 i6->h1 i7->h1   
## -0.90 0.38 -0.23 -0.39 -0.48 0.35 -0.13 0.08   
## i8->h1 i9->h1 i10->h1 i11->h1 i12->h1 i13->h1 i14->h1 i15->h1   
## -0.59 0.21 0.54 -0.51 -0.42 0.19 0.38 -0.58   
## i16->h1 i17->h1 i18->h1 i19->h1 i20->h1 i21->h1 i22->h1 i23->h1   
## -0.37 0.08 0.18 0.20 0.68 0.60 -0.72 -0.15   
## i24->h1 i25->h1 i26->h1 i27->h1 i28->h1 i29->h1 i30->h1 i31->h1   
## 0.64 0.60 0.23 -0.47 0.23 -0.52 0.63 -0.13   
## i32->h1 i33->h1 i34->h1 i35->h1 i36->h1 i37->h1 i38->h1 i39->h1   
## 0.10 0.16 0.31 0.30 -0.33 0.27 -0.60 -0.30   
## i40->h1 i41->h1 i42->h1 i43->h1 i44->h1 i45->h1 i46->h1 i47->h1   
## -0.27 0.01 -0.57 -0.05 0.19 0.64 0.45 -0.57   
## i48->h1 i49->h1 i50->h1 i51->h1 i52->h1 i53->h1 i54->h1 i55->h1   
## -0.39 -0.66 0.64 -0.03 0.45 0.59 0.51 -0.62   
## i56->h1 i57->h1 i58->h1 i59->h1 i60->h1 i61->h1 i62->h1 i63->h1   
## -0.15 0.19 -0.08 -0.09 0.38 -0.19 -0.52 -0.41   
## i64->h1 i65->h1 i66->h1 i67->h1 i68->h1 i69->h1 i70->h1 i71->h1   
## -0.65 0.23 0.05 0.52 -0.24 -0.44 0.54 0.26   
## i72->h1 i73->h1 i74->h1 i75->h1 i76->h1 i77->h1 i78->h1 i79->h1   
## -0.55 0.04 0.66 0.44 -0.73 -0.51 -0.16 -0.43   
## i80->h1 i81->h1 i82->h1 i83->h1 i84->h1 i85->h1 i86->h1 i87->h1   
## 0.57 -0.33 0.43 -0.58 0.07 -0.35 0.62 -0.20   
## i88->h1 i89->h1 i90->h1 i91->h1 i92->h1 i93->h1 i94->h1 i95->h1   
## 0.47 0.23 0.70 0.03 -0.23 0.29 -0.69 -0.04   
## i96->h1 i97->h1 i98->h1 i99->h1 i100->h1 i101->h1 i102->h1 i103->h1   
## -0.26 0.66 -0.16 -0.04 0.03 -0.35 0.49 0.14   
## i104->h1 i105->h1 i106->h1 i107->h1 i108->h1 i109->h1 i110->h1 i111->h1   
## 0.11 -0.07 -0.41 -0.04 -0.29 0.65 -0.14 -1.06   
## i112->h1 i113->h1 i114->h1 i115->h1 i116->h1 i117->h1 i118->h1 i119->h1   
## -0.31 0.42 -0.43 -0.52 -0.02 -0.42 -0.63 -0.20   
## i120->h1 i121->h1 i122->h1 i123->h1 i124->h1 i125->h1 i126->h1 i127->h1   
## 0.41 -0.06 -0.37 -0.32 0.44 0.59 0.24 -0.53   
## i128->h1 i129->h1 i130->h1 i131->h1 i132->h1 i133->h1 i134->h1 i135->h1   
## 0.63 -0.12 -0.05 -0.59 0.52 -0.47 0.70 0.55   
## i136->h1 i137->h1 i138->h1 i139->h1 i140->h1 i141->h1 i142->h1 i143->h1   
## 0.29 0.46 -0.02 0.12 0.28 0.15 -0.05 -0.21   
## i144->h1 i145->h1 i146->h1 i147->h1   
## 0.58 -0.24 -1.75 -3.99   
## b->h2 i1->h2 i2->h2 i3->h2 i4->h2 i5->h2 i6->h2 i7->h2   
## 1.14 0.54 0.01 -6.12 1.75 1.71 0.55 0.53   
## i8->h2 i9->h2 i10->h2 i11->h2 i12->h2 i13->h2 i14->h2 i15->h2   
## 2.25 0.14 0.31 -0.25 -0.09 -0.45 -0.36 0.79   
## i16->h2 i17->h2 i18->h2 i19->h2 i20->h2 i21->h2 i22->h2 i23->h2   
## 1.21 -0.14 -1.67 0.36 7.20 -0.28 -6.63 -0.05   
## i24->h2 i25->h2 i26->h2 i27->h2 i28->h2 i29->h2 i30->h2 i31->h2   
## 0.03 -0.24 0.48 -0.65 0.40 1.60 0.22 -2.60   
## i32->h2 i33->h2 i34->h2 i35->h2 i36->h2 i37->h2 i38->h2 i39->h2   
## 0.65 0.68 -1.35 -0.42 0.43 -0.13 -0.06 -0.65   
## i40->h2 i41->h2 i42->h2 i43->h2 i44->h2 i45->h2 i46->h2 i47->h2   
## 1.01 0.66 0.54 -0.19 -3.48 -0.66 -0.17 -0.47   
## i48->h2 i49->h2 i50->h2 i51->h2 i52->h2 i53->h2 i54->h2 i55->h2   
## -0.13 -0.53 -0.14 0.04 -1.12 1.39 -0.56 0.45   
## i56->h2 i57->h2 i58->h2 i59->h2 i60->h2 i61->h2 i62->h2 i63->h2   
## -0.70 -0.48 -0.38 1.77 0.47 -0.09 0.22 -0.01   
## i64->h2 i65->h2 i66->h2 i67->h2 i68->h2 i69->h2 i70->h2 i71->h2   
## -0.03 0.10 -0.06 0.11 0.34 0.82 -0.55 -0.69   
## i72->h2 i73->h2 i74->h2 i75->h2 i76->h2 i77->h2 i78->h2 i79->h2   
## 2.98 -0.40 0.41 0.61 -2.59 -0.70 0.21 -0.11   
## i80->h2 i81->h2 i82->h2 i83->h2 i84->h2 i85->h2 i86->h2 i87->h2   
## 0.36 0.69 0.63 0.00 0.60 -0.73 0.16 -0.67   
## i88->h2 i89->h2 i90->h2 i91->h2 i92->h2 i93->h2 i94->h2 i95->h2   
## -1.06 3.54 -0.47 0.15 0.18 -0.43 2.05 0.07   
## i96->h2 i97->h2 i98->h2 i99->h2 i100->h2 i101->h2 i102->h2 i103->h2   
## 2.52 0.51 0.59 -1.44 -1.17 -1.34 -3.32 -0.56   
## i104->h2 i105->h2 i106->h2 i107->h2 i108->h2 i109->h2 i110->h2 i111->h2   
## 6.46 -2.11 -4.98 2.20 -2.22 0.61 3.95 1.69   
## i112->h2 i113->h2 i114->h2 i115->h2 i116->h2 i117->h2 i118->h2 i119->h2   
## -7.70 0.43 0.50 0.43 -0.33 0.46 0.48 -0.23   
## i120->h2 i121->h2 i122->h2 i123->h2 i124->h2 i125->h2 i126->h2 i127->h2   
## -0.51 -0.50 -0.25 0.67 -0.05 -0.12 0.13 0.03   
## i128->h2 i129->h2 i130->h2 i131->h2 i132->h2 i133->h2 i134->h2 i135->h2   
## -0.59 0.16 -0.57 -0.53 -0.45 0.60 -0.24 -0.42   
## i136->h2 i137->h2 i138->h2 i139->h2 i140->h2 i141->h2 i142->h2 i143->h2   
## -0.13 0.19 0.52 0.47 -0.31 0.26 -0.15 0.03   
## i144->h2 i145->h2 i146->h2 i147->h2   
## 0.55 0.10 -1.26 -3.48   
## b->h3 i1->h3 i2->h3 i3->h3 i4->h3 i5->h3 i6->h3 i7->h3   
## -2.96 -0.40 -1.19 1.81 1.05 2.12 0.03 -0.58   
## i8->h3 i9->h3 i10->h3 i11->h3 i12->h3 i13->h3 i14->h3 i15->h3   
## -4.18 -1.46 4.01 0.66 -0.08 0.42 -0.03 -0.10   
## i16->h3 i17->h3 i18->h3 i19->h3 i20->h3 i21->h3 i22->h3 i23->h3   
## -6.43 -0.24 10.57 0.38 -12.46 0.91 11.79 -0.74   
## i24->h3 i25->h3 i26->h3 i27->h3 i28->h3 i29->h3 i30->h3 i31->h3   
## -3.64 0.27 0.36 2.21 0.35 -6.58 -0.86 7.76   
## i32->h3 i33->h3 i34->h3 i35->h3 i36->h3 i37->h3 i38->h3 i39->h3   
## -0.41 0.33 6.17 0.02 -0.06 -0.37 0.20 -0.24   
## i40->h3 i41->h3 i42->h3 i43->h3 i44->h3 i45->h3 i46->h3 i47->h3   
## 0.15 -0.71 0.50 0.43 10.44 0.80 0.82 1.65   
## i48->h3 i49->h3 i50->h3 i51->h3 i52->h3 i53->h3 i54->h3 i55->h3   
## 0.13 1.71 0.88 -0.66 4.92 -8.84 0.11 -1.52   
## i56->h3 i57->h3 i58->h3 i59->h3 i60->h3 i61->h3 i62->h3 i63->h3   
## -0.20 0.18 2.74 -6.28 0.61 0.13 -0.37 0.35   
## i64->h3 i65->h3 i66->h3 i67->h3 i68->h3 i69->h3 i70->h3 i71->h3   
## -0.17 0.98 -0.27 1.71 -0.53 -0.91 1.02 0.47   
## i72->h3 i73->h3 i74->h3 i75->h3 i76->h3 i77->h3 i78->h3 i79->h3   
## -10.31 -0.27 -0.09 -0.11 6.51 1.02 0.54 -0.71   
## i80->h3 i81->h3 i82->h3 i83->h3 i84->h3 i85->h3 i86->h3 i87->h3   
## 0.63 0.19 -0.50 -5.09 0.22 1.09 0.08 1.73   
## i88->h3 i89->h3 i90->h3 i91->h3 i92->h3 i93->h3 i94->h3 i95->h3   
## 1.33 -9.39 -0.40 0.01 -1.74 0.85 -11.01 0.00   
## i96->h3 i97->h3 i98->h3 i99->h3 i100->h3 i101->h3 i102->h3 i103->h3   
## -8.78 0.25 -0.57 6.41 5.73 5.84 4.67 0.47   
## i104->h3 i105->h3 i106->h3 i107->h3 i108->h3 i109->h3 i110->h3 i111->h3   
## -12.75 4.78 3.20 -4.52 -0.66 -3.09 -6.04 7.47   
## i112->h3 i113->h3 i114->h3 i115->h3 i116->h3 i117->h3 i118->h3 i119->h3   
## -0.79 -1.89 -0.39 -0.09 0.62 -0.05 -0.01 0.12   
## i120->h3 i121->h3 i122->h3 i123->h3 i124->h3 i125->h3 i126->h3 i127->h3   
## -0.68 0.21 -0.59 -0.34 0.40 -0.41 -0.16 -0.48   
## i128->h3 i129->h3 i130->h3 i131->h3 i132->h3 i133->h3 i134->h3 i135->h3   
## -0.12 0.28 -0.14 -0.50 0.37 -0.03 -0.28 -0.65   
## i136->h3 i137->h3 i138->h3 i139->h3 i140->h3 i141->h3 i142->h3 i143->h3   
## -0.70 -0.48 0.05 -2.40 -0.24 -0.24 -0.18 0.48   
## i144->h3 i145->h3 i146->h3 i147->h3   
## -0.52 -0.28 1.69 0.25   
## b->h4 i1->h4 i2->h4 i3->h4 i4->h4 i5->h4 i6->h4 i7->h4   
## -0.75 -0.70 0.55 0.04 -0.42 0.03 -0.67 0.10   
## i8->h4 i9->h4 i10->h4 i11->h4 i12->h4 i13->h4 i14->h4 i15->h4   
## -0.65 -0.04 0.37 0.22 -0.66 0.26 -0.23 0.29   
## i16->h4 i17->h4 i18->h4 i19->h4 i20->h4 i21->h4 i22->h4 i23->h4   
## -0.60 -0.20 0.46 -0.65 0.15 -0.26 -0.12 -0.49   
## i24->h4 i25->h4 i26->h4 i27->h4 i28->h4 i29->h4 i30->h4 i31->h4   
## 0.54 0.01 -0.67 0.64 0.41 -0.34 -0.24 0.91   
## i32->h4 i33->h4 i34->h4 i35->h4 i36->h4 i37->h4 i38->h4 i39->h4   
## -0.25 0.58 -0.52 0.10 -0.19 0.11 -0.67 0.67   
## i40->h4 i41->h4 i42->h4 i43->h4 i44->h4 i45->h4 i46->h4 i47->h4   
## -0.44 0.15 -0.65 -0.28 -0.52 0.04 -0.60 0.40   
## i48->h4 i49->h4 i50->h4 i51->h4 i52->h4 i53->h4 i54->h4 i55->h4   
## 0.14 -0.04 0.25 -0.51 -0.08 -0.05 0.61 -0.05   
## i56->h4 i57->h4 i58->h4 i59->h4 i60->h4 i61->h4 i62->h4 i63->h4   
## -0.17 -0.67 0.63 -0.60 -0.29 0.18 -0.57 0.33   
## i64->h4 i65->h4 i66->h4 i67->h4 i68->h4 i69->h4 i70->h4 i71->h4   
## -0.39 -0.39 -0.43 0.30 -0.06 0.29 -0.47 0.11   
## i72->h4 i73->h4 i74->h4 i75->h4 i76->h4 i77->h4 i78->h4 i79->h4   
## -0.06 0.30 -0.29 -0.65 -0.48 0.50 0.17 0.39   
## i80->h4 i81->h4 i82->h4 i83->h4 i84->h4 i85->h4 i86->h4 i87->h4   
## -0.27 -0.68 0.04 0.47 -0.18 0.55 -0.51 -0.25   
## i88->h4 i89->h4 i90->h4 i91->h4 i92->h4 i93->h4 i94->h4 i95->h4   
## -0.54 0.42 -0.53 -0.18 -0.27 -0.45 -0.40 -0.18   
## i96->h4 i97->h4 i98->h4 i99->h4 i100->h4 i101->h4 i102->h4 i103->h4   
## 0.36 0.27 -0.51 0.15 -0.68 0.12 0.09 0.40   
## i104->h4 i105->h4 i106->h4 i107->h4 i108->h4 i109->h4 i110->h4 i111->h4   
## -0.75 0.40 -0.40 -0.54 -0.24 0.03 -1.73 0.34   
## i112->h4 i113->h4 i114->h4 i115->h4 i116->h4 i117->h4 i118->h4 i119->h4   
## -0.53 0.21 0.34 0.33 0.16 0.35 -0.07 -0.04   
## i120->h4 i121->h4 i122->h4 i123->h4 i124->h4 i125->h4 i126->h4 i127->h4   
## 0.66 -0.41 0.57 -0.05 -0.10 -0.61 0.01 0.39   
## i128->h4 i129->h4 i130->h4 i131->h4 i132->h4 i133->h4 i134->h4 i135->h4   
## 0.41 -0.29 -0.15 0.62 0.42 0.33 -0.29 0.15   
## i136->h4 i137->h4 i138->h4 i139->h4 i140->h4 i141->h4 i142->h4 i143->h4   
## 0.36 0.44 -0.28 0.04 0.12 0.22 0.45 0.27   
## i144->h4 i145->h4 i146->h4 i147->h4   
## 0.10 0.43 0.48 -1.03   
## b->h5 i1->h5 i2->h5 i3->h5 i4->h5 i5->h5 i6->h5 i7->h5   
## 1.66 0.61 -0.33 0.82 0.96 0.23 0.29 0.57   
## i8->h5 i9->h5 i10->h5 i11->h5 i12->h5 i13->h5 i14->h5 i15->h5   
## -1.85 0.63 -0.15 -0.25 0.34 0.36 -0.27 -0.12   
## i16->h5 i17->h5 i18->h5 i19->h5 i20->h5 i21->h5 i22->h5 i23->h5   
## -1.84 -0.48 0.36 0.65 -2.40 1.02 -0.46 -1.06   
## i24->h5 i25->h5 i26->h5 i27->h5 i28->h5 i29->h5 i30->h5 i31->h5   
## 0.45 0.61 0.42 -0.12 -0.83 -1.09 0.45 3.50   
## i32->h5 i33->h5 i34->h5 i35->h5 i36->h5 i37->h5 i38->h5 i39->h5   
## -0.46 -0.31 0.13 0.31 -0.09 -0.37 -0.42 0.41   
## i40->h5 i41->h5 i42->h5 i43->h5 i44->h5 i45->h5 i46->h5 i47->h5   
## -0.10 -0.27 -0.70 -0.30 0.48 0.71 0.30 0.80   
## i48->h5 i49->h5 i50->h5 i51->h5 i52->h5 i53->h5 i54->h5 i55->h5   
## -0.46 2.11 0.44 -0.56 1.63 -0.59 0.34 0.18   
## i56->h5 i57->h5 i58->h5 i59->h5 i60->h5 i61->h5 i62->h5 i63->h5   
## -0.47 -0.45 0.12 0.76 -0.45 0.11 -0.52 0.54   
## i64->h5 i65->h5 i66->h5 i67->h5 i68->h5 i69->h5 i70->h5 i71->h5   
## 0.52 0.28 -0.02 1.11 0.16 0.65 0.46 0.57   
## i72->h5 i73->h5 i74->h5 i75->h5 i76->h5 i77->h5 i78->h5 i79->h5   
## -1.14 -0.28 0.64 -0.61 2.61 0.08 -0.23 -0.09   
## i80->h5 i81->h5 i82->h5 i83->h5 i84->h5 i85->h5 i86->h5 i87->h5   
## -0.14 -0.04 0.34 -0.21 -0.34 1.28 -0.30 0.24   
## i88->h5 i89->h5 i90->h5 i91->h5 i92->h5 i93->h5 i94->h5 i95->h5   
## 0.31 -0.55 -0.41 0.57 -0.98 0.10 -2.99 -0.12   
## i96->h5 i97->h5 i98->h5 i99->h5 i100->h5 i101->h5 i102->h5 i103->h5   
## -2.64 -0.08 -0.73 0.79 -0.30 1.10 -0.62 0.43   
## i104->h5 i105->h5 i106->h5 i107->h5 i108->h5 i109->h5 i110->h5 i111->h5   
## -2.58 -0.22 -1.85 -0.75 3.69 0.19 -3.61 5.19   
## i112->h5 i113->h5 i114->h5 i115->h5 i116->h5 i117->h5 i118->h5 i119->h5   
## -3.01 -0.73 -0.62 -0.46 -0.72 -0.17 -0.38 0.58   
## i120->h5 i121->h5 i122->h5 i123->h5 i124->h5 i125->h5 i126->h5 i127->h5   
## 0.44 -0.09 0.22 0.01 0.40 -0.43 -0.04 0.38   
## i128->h5 i129->h5 i130->h5 i131->h5 i132->h5 i133->h5 i134->h5 i135->h5   
## 0.27 -0.50 -0.12 -0.36 0.03 0.41 0.15 -0.66   
## i136->h5 i137->h5 i138->h5 i139->h5 i140->h5 i141->h5 i142->h5 i143->h5   
## 0.51 0.29 -0.29 -0.44 0.42 0.35 0.68 0.09   
## i144->h5 i145->h5 i146->h5 i147->h5   
## -0.41 0.68 2.83 -2.38   
## b->h6 i1->h6 i2->h6 i3->h6 i4->h6 i5->h6 i6->h6 i7->h6   
## 1.00 0.37 0.63 -0.25 0.54 0.59 -0.19 0.61   
## i8->h6 i9->h6 i10->h6 i11->h6 i12->h6 i13->h6 i14->h6 i15->h6   
## -0.19 0.27 0.11 0.59 -0.06 0.12 0.04 -0.13   
## i16->h6 i17->h6 i18->h6 i19->h6 i20->h6 i21->h6 i22->h6 i23->h6   
## -0.63 -0.17 -0.57 0.58 0.05 0.51 0.37 -0.12   
## i24->h6 i25->h6 i26->h6 i27->h6 i28->h6 i29->h6 i30->h6 i31->h6   
## 0.09 0.40 -0.07 -0.63 -0.12 0.45 0.17 -1.42   
## i32->h6 i33->h6 i34->h6 i35->h6 i36->h6 i37->h6 i38->h6 i39->h6   
## 0.77 -0.36 -0.41 -0.45 0.20 -0.65 -0.34 -0.05   
## i40->h6 i41->h6 i42->h6 i43->h6 i44->h6 i45->h6 i46->h6 i47->h6   
## -0.06 0.48 -0.57 0.66 0.33 0.63 0.64 -0.28   
## i48->h6 i49->h6 i50->h6 i51->h6 i52->h6 i53->h6 i54->h6 i55->h6   
## 0.60 -0.83 0.44 0.60 0.15 0.00 -0.35 -0.24   
## i56->h6 i57->h6 i58->h6 i59->h6 i60->h6 i61->h6 i62->h6 i63->h6   
## -0.21 0.12 -0.12 0.66 -0.05 -0.40 -0.13 0.49   
## i64->h6 i65->h6 i66->h6 i67->h6 i68->h6 i69->h6 i70->h6 i71->h6   
## 0.31 0.28 -0.20 -0.01 0.03 -0.22 -0.01 0.19   
## i72->h6 i73->h6 i74->h6 i75->h6 i76->h6 i77->h6 i78->h6 i79->h6   
## 0.52 0.08 0.57 -0.30 0.03 -0.30 -0.10 0.19   
## i80->h6 i81->h6 i82->h6 i83->h6 i84->h6 i85->h6 i86->h6 i87->h6   
## -0.14 0.67 0.44 0.42 0.02 -0.49 -0.41 -0.49   
## i88->h6 i89->h6 i90->h6 i91->h6 i92->h6 i93->h6 i94->h6 i95->h6   
## -0.49 0.44 0.59 0.17 -0.55 0.11 0.40 -0.63   
## i96->h6 i97->h6 i98->h6 i99->h6 i100->h6 i101->h6 i102->h6 i103->h6   
## 0.59 0.40 -0.48 -0.68 0.67 -0.77 0.23 -0.44   
## i104->h6 i105->h6 i106->h6 i107->h6 i108->h6 i109->h6 i110->h6 i111->h6   
## -0.35 0.11 -0.46 0.47 -1.56 0.24 2.66 -1.49   
## i112->h6 i113->h6 i114->h6 i115->h6 i116->h6 i117->h6 i118->h6 i119->h6   
## 1.85 0.43 -0.35 0.52 0.21 0.48 0.40 -0.18   
## i120->h6 i121->h6 i122->h6 i123->h6 i124->h6 i125->h6 i126->h6 i127->h6   
## -0.35 -0.50 -0.13 -0.52 0.34 0.61 0.58 -0.62   
## i128->h6 i129->h6 i130->h6 i131->h6 i132->h6 i133->h6 i134->h6 i135->h6   
## -0.17 -0.34 0.18 -0.39 0.45 -0.47 -0.18 -0.68   
## i136->h6 i137->h6 i138->h6 i139->h6 i140->h6 i141->h6 i142->h6 i143->h6   
## -0.60 0.00 0.57 0.37 0.61 0.04 -0.30 0.09   
## i144->h6 i145->h6 i146->h6 i147->h6   
## -0.37 0.40 -0.21 1.49   
## b->o h1->o h2->o h3->o h4->o h5->o h6->o   
## -9.62 3.33 -6.87 32.00 0.50 9.15 -9.70

Now adding column to each training and test data to see if the model is fit

project\_traindata$predict <- predict(neuralnet\_p, project\_traindata, type = "class")  
project\_test\_data$predict <- predict(neuralnet\_p, project\_test\_data, type = "class")

Checking the model to see how close we are

testsample <- table(split\_data$classification[-test\_nnet\_sample], project\_test\_data$predict)  
testsample

##   
## Need Want  
## Need 55 4  
## Want 1 20

As we can see, out of 59 Needs 55 were predicted correctly and Out of 21 want 20 were predicted correctly

\*\*Creating a dummy data to see what our model predicts it to be

dummydata <- data.frame("Columbia store", "Red onions","Produce",2,"null",2.00,NA)  
names(dummydata) <- c("storeName", "description","merchandise.category","quantity","lbs","amount","classification")  
classification <- predict(neuralnet\_p, dummydata , type = "class")  
classification

## [1] "Need"

Now we can use the neural net model to predict our spending and optimize our budget

loading data for April month

TestingDataSetForApril <- **read.csv**("/Users/chirnjibi/Desktop/Project\_TestingDataSetForApril.csv")  
**head**(TestingDataSetForApril)

##   item Month     Date      storeName description merchandise.category  
## 1    1 April 4/1/2018 Columbia store  Red onions              Produce  
## 2    2 April 4/1/2018 Columbia store      Tomato              Produce  
## 3    3 April 4/1/2018 Columbia store  Red Potato              Produce  
## 4    4 April 4/1/2018 Columbia store Green chili              Produce  
## 5    5 April 4/1/2018 Columbia store      Garlic              Produce  
## 6    6 April 4/1/2018 Columbia store     Spinich              Produce  
##   quantity  lbs amount  
## 1        1    2   1.99  
## 2        1 3.07   4.57  
## 3        1 2.06   2.04  
## 4        1 0.23   2.49  
## 5        1 null   0.85  
## 6        1 null   2.49

AprilMonthClassificationOfData <-  **predict**(neuralnet\_p, TestingDataSetForApril , type = "class")

Checking the classification

TestingDataSetForApril**$**classification <- AprilMonthClassificationOfData  
**head**(TestingDataSetForApril)

##   item Month     Date      storeName description merchandise.category  
## 1    1 April 4/1/2018 Columbia store  Red onions              Produce  
## 2    2 April 4/1/2018 Columbia store      Tomato              Produce  
## 3    3 April 4/1/2018 Columbia store  Red Potato              Produce  
## 4    4 April 4/1/2018 Columbia store Green chili              Produce  
## 5    5 April 4/1/2018 Columbia store      Garlic              Produce  
## 6    6 April 4/1/2018 Columbia store     Spinich              Produce  
##   quantity  lbs amount classification  
## 1        1    2   1.99           Need  
## 2        1 3.07   4.57           Need  
## 3        1 2.06   2.04           Need  
## 4        1 0.23   2.49           Need  
## 5        1 null   0.85           Need  
## 6        1 null   2.49           Need

As we can see when we purchase Red onions from Columbia Store it is classified as our Needs, meaning the Red onion is something we can’t avoid buying

But Sandwich cookie that we bought from Weis is not necessary for us. It’s something we can avoid buying and save some money or spend it on our needs. So by categorizing things to need and wants we can filter or avoid buying unnecessary and unplanned groceries and cut down on our expenses.

References

* “Data Visualization with ggplot2.” [Online]. Available: <https://www.rstudio.com/wp-content/uploads/2015/03/ggplot2-cheatsheet.pdf>.
* Top 50 ggplot2 Visualizations.” [Online]. Available: http://r-statistics.co/Top50-Ggplot2-Visualizations-MasterList-R-Code.html. [Accessed: 08-May-2018].