Week14 IP

Siiren

4/10/2021

# PART 1: Dimensionality Reduction

#load library  
library(tidyverse)

## -- Attaching packages --------------------------------------- tidyverse 1.3.0 --

## v ggplot2 3.3.3 v purrr 0.3.4  
## v tibble 3.1.0 v dplyr 1.0.5  
## v tidyr 1.1.3 v stringr 1.4.0  
## v readr 1.4.0 v forcats 0.5.1

## -- Conflicts ------------------------------------------ tidyverse\_conflicts() --  
## x dplyr::filter() masks stats::filter()  
## x dplyr::lag() masks stats::lag()

library(dplyr)  
library(data.table)

##   
## Attaching package: 'data.table'

## The following objects are masked from 'package:dplyr':  
##   
## between, first, last

## The following object is masked from 'package:purrr':  
##   
## transpose

# Loading our dataset  
# ---  
#   
df <-read.csv("http://bit.ly/CarreFourDataset")  
head(df)

## Invoice.ID Branch Customer.type Gender Product.line Unit.price  
## 1 750-67-8428 A Member Female Health and beauty 74.69  
## 2 226-31-3081 C Normal Female Electronic accessories 15.28  
## 3 631-41-3108 A Normal Male Home and lifestyle 46.33  
## 4 123-19-1176 A Member Male Health and beauty 58.22  
## 5 373-73-7910 A Normal Male Sports and travel 86.31  
## 6 699-14-3026 C Normal Male Electronic accessories 85.39  
## Quantity Tax Date Time Payment cogs gross.margin.percentage  
## 1 7 26.1415 1/5/2019 13:08 Ewallet 522.83 4.761905  
## 2 5 3.8200 3/8/2019 10:29 Cash 76.40 4.761905  
## 3 7 16.2155 3/3/2019 13:23 Credit card 324.31 4.761905  
## 4 8 23.2880 1/27/2019 20:33 Ewallet 465.76 4.761905  
## 5 7 30.2085 2/8/2019 10:37 Ewallet 604.17 4.761905  
## 6 7 29.8865 3/25/2019 18:30 Ewallet 597.73 4.761905  
## gross.income Rating Total  
## 1 26.1415 9.1 548.9715  
## 2 3.8200 9.6 80.2200  
## 3 16.2155 7.4 340.5255  
## 4 23.2880 8.4 489.0480  
## 5 30.2085 5.3 634.3785  
## 6 29.8865 4.1 627.6165

nums <- select\_if(df, is.numeric)  
head(nums)

## Unit.price Quantity Tax cogs gross.margin.percentage gross.income  
## 1 74.69 7 26.1415 522.83 4.761905 26.1415  
## 2 15.28 5 3.8200 76.40 4.761905 3.8200  
## 3 46.33 7 16.2155 324.31 4.761905 16.2155  
## 4 58.22 8 23.2880 465.76 4.761905 23.2880  
## 5 86.31 7 30.2085 604.17 4.761905 30.2085  
## 6 85.39 7 29.8865 597.73 4.761905 29.8865  
## Rating Total  
## 1 9.1 548.9715  
## 2 9.6 80.2200  
## 3 7.4 340.5255  
## 4 8.4 489.0480  
## 5 5.3 634.3785  
## 6 4.1 627.6165

# Changing column names to lower case, and replacing spaces with underscores  
colnames(df) = tolower(str\_replace\_all(colnames(df), c(' ' = '\_')))  
  
# Checking column names.  
colnames(df)

## [1] "invoice.id" "branch"   
## [3] "customer.type" "gender"   
## [5] "product.line" "unit.price"   
## [7] "quantity" "tax"   
## [9] "date" "time"   
## [11] "payment" "cogs"   
## [13] "gross.margin.percentage" "gross.income"   
## [15] "rating" "total"

# Dropping unnecessary columns  
df$invoice.id <- NULL  
df$date <- NULL  
df$time <- NULL  
  
head(df)

## branch customer.type gender product.line unit.price quantity  
## 1 A Member Female Health and beauty 74.69 7  
## 2 C Normal Female Electronic accessories 15.28 5  
## 3 A Normal Male Home and lifestyle 46.33 7  
## 4 A Member Male Health and beauty 58.22 8  
## 5 A Normal Male Sports and travel 86.31 7  
## 6 C Normal Male Electronic accessories 85.39 7  
## tax payment cogs gross.margin.percentage gross.income rating  
## 1 26.1415 Ewallet 522.83 4.761905 26.1415 9.1  
## 2 3.8200 Cash 76.40 4.761905 3.8200 9.6  
## 3 16.2155 Credit card 324.31 4.761905 16.2155 7.4  
## 4 23.2880 Ewallet 465.76 4.761905 23.2880 8.4  
## 5 30.2085 Ewallet 604.17 4.761905 30.2085 5.3  
## 6 29.8865 Ewallet 597.73 4.761905 29.8865 4.1  
## total  
## 1 548.9715  
## 2 80.2200  
## 3 340.5255  
## 4 489.0480  
## 5 634.3785  
## 6 627.6165

df1<-df  
df1<-df1%>%  
 mutate(branch = replace(branch, branch == "A", "1"))  
df1<-df1%>%  
 mutate(branch = replace(branch, branch == "B", "2"))  
df1<-df1%>%  
 mutate(branch = replace(branch, branch == "C", "3"))  
  
head(df1)

## branch customer.type gender product.line unit.price quantity  
## 1 1 Member Female Health and beauty 74.69 7  
## 2 3 Normal Female Electronic accessories 15.28 5  
## 3 1 Normal Male Home and lifestyle 46.33 7  
## 4 1 Member Male Health and beauty 58.22 8  
## 5 1 Normal Male Sports and travel 86.31 7  
## 6 3 Normal Male Electronic accessories 85.39 7  
## tax payment cogs gross.margin.percentage gross.income rating  
## 1 26.1415 Ewallet 522.83 4.761905 26.1415 9.1  
## 2 3.8200 Cash 76.40 4.761905 3.8200 9.6  
## 3 16.2155 Credit card 324.31 4.761905 16.2155 7.4  
## 4 23.2880 Ewallet 465.76 4.761905 23.2880 8.4  
## 5 30.2085 Ewallet 604.17 4.761905 30.2085 5.3  
## 6 29.8865 Ewallet 597.73 4.761905 29.8865 4.1  
## total  
## 1 548.9715  
## 2 80.2200  
## 3 340.5255  
## 4 489.0480  
## 5 634.3785  
## 6 627.6165

# Dropping unnecessary columns  
df2<-df1  
df2$customer.type <- NULL  
df2$gender <- NULL  
df2$product.line <- NULL  
df2$payment <- NULL  
  
head(df2)

## branch unit.price quantity tax cogs gross.margin.percentage  
## 1 1 74.69 7 26.1415 522.83 4.761905  
## 2 3 15.28 5 3.8200 76.40 4.761905  
## 3 1 46.33 7 16.2155 324.31 4.761905  
## 4 1 58.22 8 23.2880 465.76 4.761905  
## 5 1 86.31 7 30.2085 604.17 4.761905  
## 6 3 85.39 7 29.8865 597.73 4.761905  
## gross.income rating total  
## 1 26.1415 9.1 548.9715  
## 2 3.8200 9.6 80.2200  
## 3 16.2155 7.4 340.5255  
## 4 23.2880 8.4 489.0480  
## 5 30.2085 5.3 634.3785  
## 6 29.8865 4.1 627.6165

#onehotencoding categorical columns  
library(caret)

## Loading required package: lattice

##   
## Attaching package: 'caret'

## The following object is masked from 'package:purrr':  
##   
## lift

library(dplyr)  
  
df1$branch<-as.numeric(df1$branch)  
  
dmy <- dummyVars(" ~ .", data = df1, fullRank = T)  
df\_transformed <- data.frame(predict(dmy, newdata = df1))  
  
head(df\_transformed)

## branch customer.typeNormal genderMale product.lineFashion.accessories  
## 1 1 0 0 0  
## 2 3 1 0 0  
## 3 1 1 1 0  
## 4 1 0 1 0  
## 5 1 1 1 0  
## 6 3 1 1 0  
## product.lineFood.and.beverages product.lineHealth.and.beauty  
## 1 0 1  
## 2 0 0  
## 3 0 0  
## 4 0 1  
## 5 0 0  
## 6 0 0  
## product.lineHome.and.lifestyle product.lineSports.and.travel unit.price  
## 1 0 0 74.69  
## 2 0 0 15.28  
## 3 1 0 46.33  
## 4 0 0 58.22  
## 5 0 1 86.31  
## 6 0 0 85.39  
## quantity tax paymentCredit.card paymentEwallet cogs  
## 1 7 26.1415 0 1 522.83  
## 2 5 3.8200 0 0 76.40  
## 3 7 16.2155 1 0 324.31  
## 4 8 23.2880 0 1 465.76  
## 5 7 30.2085 0 1 604.17  
## 6 7 29.8865 0 1 597.73  
## gross.margin.percentage gross.income rating total  
## 1 4.761905 26.1415 9.1 548.9715  
## 2 4.761905 3.8200 9.6 80.2200  
## 3 4.761905 16.2155 7.4 340.5255  
## 4 4.761905 23.2880 8.4 489.0480  
## 5 4.761905 30.2085 5.3 634.3785  
## 6 4.761905 29.8865 4.1 627.6165

df\_transformed <- lapply(df\_transformed,as.numeric)  
str(df\_transformed)

## List of 18  
## $ branch : num [1:1000] 1 3 1 1 1 3 1 3 1 2 ...  
## $ customer.typeNormal : num [1:1000] 0 1 1 0 1 1 0 1 0 0 ...  
## $ genderMale : num [1:1000] 0 0 1 1 1 1 0 0 0 0 ...  
## $ product.lineFashion.accessories: num [1:1000] 0 0 0 0 0 0 0 0 0 0 ...  
## $ product.lineFood.and.beverages : num [1:1000] 0 0 0 0 0 0 0 0 0 1 ...  
## $ product.lineHealth.and.beauty : num [1:1000] 1 0 0 1 0 0 0 0 1 0 ...  
## $ product.lineHome.and.lifestyle : num [1:1000] 0 0 1 0 0 0 0 1 0 0 ...  
## $ product.lineSports.and.travel : num [1:1000] 0 0 0 0 1 0 0 0 0 0 ...  
## $ unit.price : num [1:1000] 74.7 15.3 46.3 58.2 86.3 ...  
## $ quantity : num [1:1000] 7 5 7 8 7 7 6 10 2 3 ...  
## $ tax : num [1:1000] 26.14 3.82 16.22 23.29 30.21 ...  
## $ paymentCredit.card : num [1:1000] 0 0 1 0 0 0 0 0 1 1 ...  
## $ paymentEwallet : num [1:1000] 1 0 0 1 1 1 1 1 0 0 ...  
## $ cogs : num [1:1000] 522.8 76.4 324.3 465.8 604.2 ...  
## $ gross.margin.percentage : num [1:1000] 4.76 4.76 4.76 4.76 4.76 ...  
## $ gross.income : num [1:1000] 26.14 3.82 16.22 23.29 30.21 ...  
## $ rating : num [1:1000] 9.1 9.6 7.4 8.4 5.3 4.1 5.8 8 7.2 5.9 ...  
## $ total : num [1:1000] 549 80.2 340.5 489 634.4 ...

# Loading our tnse library  
#   
library(Rtsne)

## Warning: package 'Rtsne' was built under R version 4.0.5

# Curating the database for analysis   
#   
Labels<-df2$total  
df2$total<-as.factor(df2$total)  
  
# For plotting  
#  
colors = rainbow(length(unique(df2$total)))  
names(colors) = unique(df2$total)

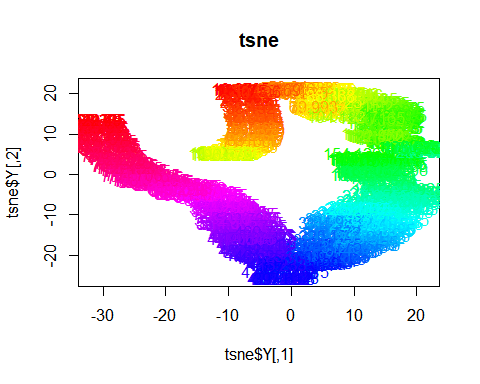
# Executing the algorithm on curated data  
#   
tsne <- Rtsne(df2[,-1], dims = 2, perplexity=30, verbose=TRUE, max\_iter = 500)

## Performing PCA  
## Read the 1000 x 50 data matrix successfully!  
## OpenMP is working. 1 threads.  
## Using no\_dims = 2, perplexity = 30.000000, and theta = 0.500000  
## Computing input similarities...  
## Building tree...  
## Done in 0.77 seconds (sparsity = 0.102662)!  
## Learning embedding...  
## Iteration 50: error is 59.778631 (50 iterations in 0.36 seconds)  
## Iteration 100: error is 52.754254 (50 iterations in 0.52 seconds)  
## Iteration 150: error is 51.492204 (50 iterations in 0.42 seconds)  
## Iteration 200: error is 51.010504 (50 iterations in 1.20 seconds)  
## Iteration 250: error is 50.790130 (50 iterations in 0.50 seconds)  
## Iteration 300: error is 0.594888 (50 iterations in 0.30 seconds)  
## Iteration 350: error is 0.423318 (50 iterations in 0.38 seconds)  
## Iteration 400: error is 0.381682 (50 iterations in 0.44 seconds)  
## Iteration 450: error is 0.366366 (50 iterations in 0.31 seconds)  
## Iteration 500: error is 0.355265 (50 iterations in 0.42 seconds)  
## Fitting performed in 4.86 seconds.

# Getting the duration of execution  
#   
exeTimeTsne <- system.time(Rtsne(df2[,-1], dims = 2, perplexity=30, verbose=TRUE, max\_iter = 500))

## Performing PCA  
## Read the 1000 x 50 data matrix successfully!  
## OpenMP is working. 1 threads.  
## Using no\_dims = 2, perplexity = 30.000000, and theta = 0.500000  
## Computing input similarities...  
## Building tree...  
## Done in 0.55 seconds (sparsity = 0.102662)!  
## Learning embedding...  
## Iteration 50: error is 58.913165 (50 iterations in 1.09 seconds)  
## Iteration 100: error is 52.459310 (50 iterations in 0.40 seconds)  
## Iteration 150: error is 51.357079 (50 iterations in 0.39 seconds)  
## Iteration 200: error is 50.834691 (50 iterations in 0.33 seconds)  
## Iteration 250: error is 50.446448 (50 iterations in 0.81 seconds)  
## Iteration 300: error is 0.589327 (50 iterations in 0.37 seconds)  
## Iteration 350: error is 0.413265 (50 iterations in 0.40 seconds)  
## Iteration 400: error is 0.367018 (50 iterations in 0.40 seconds)  
## Iteration 450: error is 0.355894 (50 iterations in 0.57 seconds)  
## Iteration 500: error is 0.348110 (50 iterations in 0.60 seconds)  
## Fitting performed in 5.34 seconds.

# Plotting our graph and closely examining the graph  
#   
plot(tsne$Y, t='n', main="tsne")  
text(tsne$Y, labels=df2$total, col=colors[df2$total])



# PART 2: Feature Selection

library(wskm)

## Warning: package 'wskm' was built under R version 4.0.5

## Loading required package: latticeExtra

## Warning: package 'latticeExtra' was built under R version 4.0.5

##   
## Attaching package: 'latticeExtra'

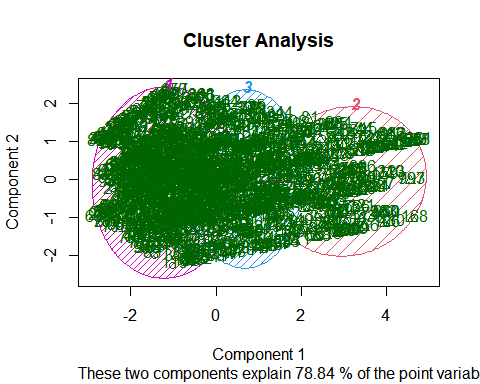
## The following object is masked from 'package:ggplot2':  
##   
## layer

## Loading required package: fpc

## Warning: package 'fpc' was built under R version 4.0.5

set.seed(2)  
model <- ewkm(df2[-1], 3, lambda=2, maxiter=1000)

library("cluster")  
  
# Cluster Plot against 1st 2 principal components  
# ---  
#  
clusplot(df2[1:5], model$cluster, color=TRUE, shade=TRUE,  
 labels=2, lines=1,main='Cluster Analysis')



round(model$weights\*100,2)

## unit.price quantity tax cogs gross.margin.percentage gross.income rating  
## 1 0 0 0 0 99.99 0 0  
## 2 0 0 0 0 99.99 0 0  
## 3 0 0 0 0 99.99 0 0  
## total  
## 1 0  
## 2 0  
## 3 0

# PART 3: Association Rules

# Loading the arules library  
#  
library(arules)

## Warning: package 'arules' was built under R version 4.0.5

## Loading required package: Matrix

##   
## Attaching package: 'Matrix'

## The following objects are masked from 'package:tidyr':  
##   
## expand, pack, unpack

##   
## Attaching package: 'arules'

## The following object is masked from 'package:dplyr':  
##   
## recode

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

#load data  
super <- read.csv("http://bit.ly/SupermarketDatasetII", header = TRUE)  
head(super)

## shrimp almonds avocado vegetables.mix green.grapes  
## 1 burgers meatballs eggs   
## 2 chutney   
## 3 turkey avocado   
## 4 mineral water milk energy bar whole wheat rice green tea  
## 5 low fat yogurt   
## 6 whole wheat pasta french fries   
## whole.weat.flour yams cottage.cheese energy.drink tomato.juice low.fat.yogurt  
## 1   
## 2   
## 3   
## 4   
## 5   
## 6   
## green.tea honey salad mineral.water salmon antioxydant.juice frozen.smoothie  
## 1   
## 2   
## 3   
## 4   
## 5   
## 6   
## spinach olive.oil  
## 1 NA  
## 2 NA  
## 3 NA  
## 4 NA  
## 5 NA  
## 6 NA

# Loading our transactions dataset from our csv file  
path <-"http://bit.ly/SupermarketDatasetII"  
  
Transactions<-read.transactions(path, sep = ",")

## Warning in asMethod(object): removing duplicated items in transactions

Transactions

## transactions in sparse format with  
## 7501 transactions (rows) and  
## 119 items (columns)

# Verifying the object's class  
#   
class(Transactions)

## [1] "transactions"  
## attr(,"package")  
## [1] "arules"

# Previewing our first 5 transactions  
#  
inspect(Transactions[1:10])

## items   
## [1] {almonds,   
## antioxydant juice,  
## avocado,   
## cottage cheese,   
## energy drink,   
## frozen smoothie,   
## green grapes,   
## green tea,   
## honey,   
## low fat yogurt,   
## mineral water,   
## olive oil,   
## salad,   
## salmon,   
## shrimp,   
## spinach,   
## tomato juice,   
## vegetables mix,   
## whole weat flour,   
## yams}   
## [2] {burgers,   
## eggs,   
## meatballs}   
## [3] {chutney}   
## [4] {avocado,   
## turkey}   
## [5] {energy bar,   
## green tea,   
## milk,   
## mineral water,   
## whole wheat rice}   
## [6] {low fat yogurt}   
## [7] {french fries,   
## whole wheat pasta}  
## [8] {light cream,   
## shallot,   
## soup}   
## [9] {frozen vegetables,  
## green tea,   
## spaghetti}   
## [10] {french fries}

# alternatively way to preview the items that make up our dataset,  
#   
items<-as.data.frame(itemLabels(Transactions))  
colnames(items) <- "Item"  
head(items, 10)

## Item  
## 1 almonds  
## 2 antioxydant juice  
## 3 asparagus  
## 4 avocado  
## 5 babies food  
## 6 bacon  
## 7 barbecue sauce  
## 8 black tea  
## 9 blueberries  
## 10 body spray

# Generating a summary of the transaction dataset  
   
summary(Transactions)

## transactions as itemMatrix in sparse format with  
## 7501 rows (elements/itemsets/transactions) and  
## 119 columns (items) and a density of 0.03288973   
##   
## most frequent items:  
## mineral water eggs spaghetti french fries chocolate   
## 1788 1348 1306 1282 1229   
## (Other)   
## 22405   
##   
## element (itemset/transaction) length distribution:  
## sizes  
## 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16   
## 1754 1358 1044 816 667 493 391 324 259 139 102 67 40 22 17 4   
## 18 19 20   
## 1 2 1   
##   
## Min. 1st Qu. Median Mean 3rd Qu. Max.   
## 1.000 2.000 3.000 3.914 5.000 20.000   
##   
## includes extended item information - examples:  
## labels  
## 1 almonds  
## 2 antioxydant juice  
## 3 asparagus

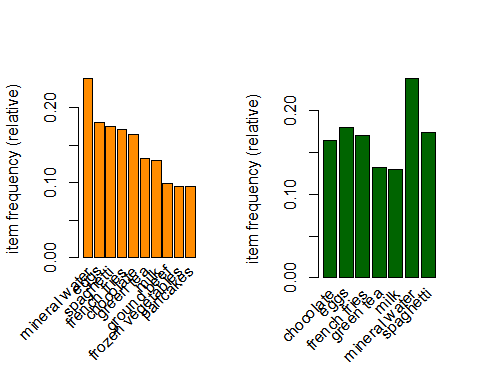
# Exploring the frequency of some articles   
  
itemFrequency(Transactions[, 1:10],type = "absolute")

## almonds antioxydant juice asparagus avocado   
## 153 67 36 250   
## babies food bacon barbecue sauce black tea   
## 34 65 81 107   
## blueberries body spray   
## 69 86

round(itemFrequency(Transactions[, 1:10],type = "relative")\*100,2)

## almonds antioxydant juice asparagus avocado   
## 2.04 0.89 0.48 3.33   
## babies food bacon barbecue sauce black tea   
## 0.45 0.87 1.08 1.43   
## blueberries body spray   
## 0.92 1.15

# Producing a chart of frequencies and fitering   
#   
par(mfrow = c(1, 2))  
  
# plot the frequency of items  
itemFrequencyPlot(Transactions, topN = 10,col="darkorange")  
itemFrequencyPlot(Transactions, support = 0.1,col="darkgreen")



# Building a model based on association rules   
#   
rules <- apriori (Transactions, parameter = list(supp = 0.001, conf = 0.8))

## Apriori  
##   
## Parameter specification:  
## confidence minval smax arem aval originalSupport maxtime support minlen  
## 0.8 0.1 1 none FALSE TRUE 5 0.001 1  
## maxlen target ext  
## 10 rules TRUE  
##   
## Algorithmic control:  
## filter tree heap memopt load sort verbose  
## 0.1 TRUE TRUE FALSE TRUE 2 TRUE  
##   
## Absolute minimum support count: 7   
##   
## set item appearances ...[0 item(s)] done [0.00s].  
## set transactions ...[119 item(s), 7501 transaction(s)] done [0.00s].  
## sorting and recoding items ... [116 item(s)] done [0.00s].  
## creating transaction tree ... done [0.00s].  
## checking subsets of size 1 2 3 4 5 6 done [0.06s].  
## writing ... [74 rule(s)] done [0.00s].  
## creating S4 object ... done [0.00s].

rules

## set of 74 rules

# We use measures of significance and interest on the rules,   
  
# Building a apriori model with Min Support as 0.002 and confidence as 0.8.  
rules2 <- apriori (Transactions,parameter = list(supp = 0.002, conf = 0.8))

## Apriori  
##   
## Parameter specification:  
## confidence minval smax arem aval originalSupport maxtime support minlen  
## 0.8 0.1 1 none FALSE TRUE 5 0.002 1  
## maxlen target ext  
## 10 rules TRUE  
##   
## Algorithmic control:  
## filter tree heap memopt load sort verbose  
## 0.1 TRUE TRUE FALSE TRUE 2 TRUE  
##   
## Absolute minimum support count: 15   
##   
## set item appearances ...[0 item(s)] done [0.00s].  
## set transactions ...[119 item(s), 7501 transaction(s)] done [0.01s].  
## sorting and recoding items ... [115 item(s)] done [0.00s].  
## creating transaction tree ... done [0.00s].  
## checking subsets of size 1 2 3 4 5 done [0.01s].  
## writing ... [2 rule(s)] done [0.00s].  
## creating S4 object ... done [0.00s].

# Building apriori model with Min Support as 0.002 and confidence as 0.6.  
rules3 <- apriori (Transactions, parameter = list(supp = 0.001, conf = 0.6))

## Apriori  
##   
## Parameter specification:  
## confidence minval smax arem aval originalSupport maxtime support minlen  
## 0.6 0.1 1 none FALSE TRUE 5 0.001 1  
## maxlen target ext  
## 10 rules TRUE  
##   
## Algorithmic control:  
## filter tree heap memopt load sort verbose  
## 0.1 TRUE TRUE FALSE TRUE 2 TRUE  
##   
## Absolute minimum support count: 7   
##   
## set item appearances ...[0 item(s)] done [0.00s].  
## set transactions ...[119 item(s), 7501 transaction(s)] done [0.00s].  
## sorting and recoding items ... [116 item(s)] done [0.00s].  
## creating transaction tree ... done [0.00s].  
## checking subsets of size 1 2 3 4 5 6 done [0.02s].  
## writing ... [545 rule(s)] done [0.01s].  
## creating S4 object ... done [0.00s].

rules2

## set of 2 rules

rules3

## set of 545 rules

# We can perform an exploration of our model   
# through the use of the summary function as shown  
#   
summary(rules)

## set of 74 rules  
##   
## rule length distribution (lhs + rhs):sizes  
## 3 4 5 6   
## 15 42 16 1   
##   
## Min. 1st Qu. Median Mean 3rd Qu. Max.   
## 3.000 4.000 4.000 4.041 4.000 6.000   
##   
## summary of quality measures:  
## support confidence coverage lift   
## Min. :0.001067 Min. :0.8000 Min. :0.001067 Min. : 3.356   
## 1st Qu.:0.001067 1st Qu.:0.8000 1st Qu.:0.001333 1st Qu.: 3.432   
## Median :0.001133 Median :0.8333 Median :0.001333 Median : 3.795   
## Mean :0.001256 Mean :0.8504 Mean :0.001479 Mean : 4.823   
## 3rd Qu.:0.001333 3rd Qu.:0.8889 3rd Qu.:0.001600 3rd Qu.: 4.877   
## Max. :0.002533 Max. :1.0000 Max. :0.002666 Max. :12.722   
## count   
## Min. : 8.000   
## 1st Qu.: 8.000   
## Median : 8.500   
## Mean : 9.419   
## 3rd Qu.:10.000   
## Max. :19.000   
##   
## mining info:  
## data ntransactions support confidence  
## Transactions 7501 0.001 0.8

# Observing rules built in our model i.e. first 10 model rules  
# ---  
#   
inspect(rules[1:10])

## lhs rhs support confidence  
## [1] {frozen smoothie,spinach} => {mineral water} 0.001066524 0.8888889   
## [2] {bacon,pancakes} => {spaghetti} 0.001733102 0.8125000   
## [3] {nonfat milk,turkey} => {mineral water} 0.001199840 0.8181818   
## [4] {ground beef,nonfat milk} => {mineral water} 0.001599787 0.8571429   
## [5] {mushroom cream sauce,pasta} => {escalope} 0.002532996 0.9500000   
## [6] {milk,pasta} => {shrimp} 0.001599787 0.8571429   
## [7] {cooking oil,fromage blanc} => {mineral water} 0.001199840 0.8181818   
## [8] {black tea,salmon} => {mineral water} 0.001066524 0.8000000   
## [9] {black tea,frozen smoothie} => {milk} 0.001199840 0.8181818   
## [10] {red wine,tomato sauce} => {chocolate} 0.001066524 0.8000000   
## coverage lift count  
## [1] 0.001199840 3.729058 8   
## [2] 0.002133049 4.666587 13   
## [3] 0.001466471 3.432428 9   
## [4] 0.001866418 3.595877 12   
## [5] 0.002666311 11.976387 19   
## [6] 0.001866418 11.995203 12   
## [7] 0.001466471 3.432428 9   
## [8] 0.001333156 3.356152 8   
## [9] 0.001466471 6.313973 9   
## [10] 0.001333156 4.882669 8

# Ordering these rules by a criteria such as the level of confidence  
#   
rules<-sort(rules, by="confidence", decreasing=TRUE)  
inspect(rules[1:5])

## lhs rhs support   
## [1] {french fries,mushroom cream sauce,pasta} => {escalope} 0.001066524  
## [2] {ground beef,light cream,olive oil} => {mineral water} 0.001199840  
## [3] {cake,meatballs,mineral water} => {milk} 0.001066524  
## [4] {cake,olive oil,shrimp} => {mineral water} 0.001199840  
## [5] {mushroom cream sauce,pasta} => {escalope} 0.002532996  
## confidence coverage lift count  
## [1] 1.00 0.001066524 12.606723 8   
## [2] 1.00 0.001199840 4.195190 9   
## [3] 1.00 0.001066524 7.717078 8   
## [4] 1.00 0.001199840 4.195190 9   
## [5] 0.95 0.002666311 11.976387 19

# Interpretation  
# ---  
# The given five rules have a confidence of 95  
# ---

# PART 4: Anomaly Detection

# Load tidyverse and anomalize  
# ---  
#   
library(tidyverse)  
library(anomalize)

## Warning: package 'anomalize' was built under R version 4.0.5

## == Use anomalize to improve your Forecasts by 50%! =============================  
## Business Science offers a 1-hour course - Lab #18: Time Series Anomaly Detection!  
## </> Learn more at: https://university.business-science.io/p/learning-labs-pro </>

library(dplyr)  
library(tibbletime)

## Warning: package 'tibbletime' was built under R version 4.0.5

##   
## Attaching package: 'tibbletime'

## The following object is masked from 'package:stats':  
##   
## filter

#load data  
anom<-read.csv("http://bit.ly/CarreFourSalesDataset")  
head(anom)

## Date Sales  
## 1 1/5/2019 548.9715  
## 2 3/8/2019 80.2200  
## 3 3/3/2019 340.5255  
## 4 1/27/2019 489.0480  
## 5 2/8/2019 634.3785  
## 6 3/25/2019 627.6165

anom$Date <- as.Date(anom$Date, format = "%m/%d/%Y")  
head(anom)

## Date Sales  
## 1 2019-01-05 548.9715  
## 2 2019-03-08 80.2200  
## 3 2019-03-03 340.5255  
## 4 2019-01-27 489.0480  
## 5 2019-02-08 634.3785  
## 6 2019-03-25 627.6165

anom\_df <- anom%>%  
 group\_by(Date)%>%  
 summarise(Grouped\_sales = sum(Sales))  
  
head(anom\_df)

## # A tibble: 6 x 2  
## Date Grouped\_sales  
## <date> <dbl>  
## 1 2019-01-01 4745.  
## 2 2019-01-02 1946.  
## 3 2019-01-03 2078.  
## 4 2019-01-04 1624.  
## 5 2019-01-05 3537.  
## 6 2019-01-06 3614.

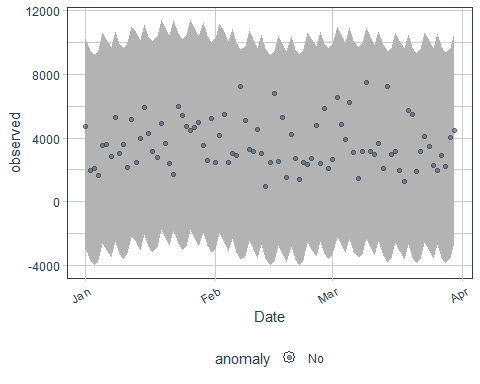
anom\_df %>%  
 time\_decompose(Grouped\_sales) %>%  
 anomalize(remainder) %>%  
 time\_recompose() %>%  
 plot\_anomalies(time\_recomposed = TRUE, ncol = 3, alpha\_dots = 0.5)

## Converting from tbl\_df to tbl\_time.  
## Auto-index message: index = Date

## frequency = 7 days

## trend = 30 days

## Registered S3 method overwritten by 'quantmod':  
## method from  
## as.zoo.data.frame zoo



The data had no anomalies this means the company sales are consistent