Market Basket Analysis

R Packages required for Market Basket Analysis

- R 3.2.3 or higher version should be installed
- Following Libraries are installed. Check by running the below command; If Library is not installed then run the install.packages command
  ## it is okay if you get Warning Message, but you should not get Error Message

```
## install.packages("arules")

## install.packages("arulesViz")


library(arules) ## requires R 3.2.3 or above

library(arulesViz)
```

Market Basket Analysis - Overview

Market Basket Analysis

Market basket analysis is the study of items that are purchased (or otherwise grouped) together in a single transaction or multiple, sequential transactions.

Market Basket Analysis is a modelling technique based upon the theory that if you buy a certain group of items, you are more (or less) likely to buy another group of items.

e.g.

- n MBA the objective is to find rules of association
- Examples:
  - {Noodles, Chips} => {Soda} … .Retail
  - {Mobile Handset} => {Scratch Guard} … ..Electronics
  - {Formal Shirts} => {Formal Trousers} … ..Apparel
  - {Munnar Hill Station} => {Thekkady Hill Station} … . Travel & Tourism
  - {Rameshwaram Temple} => {Madurai Temple} … . Travel & Tourism
  - {Writing slate} => {Slate Pencil} … . Retail Stationary
  - {Comprehensive Motor Insurance} => {Health Insurance}

Applications

- Product recommendation – like Amazon's "customers who bought that, also bought this"

- Grouping products that co-occur in the design of a store's layout to increase the chance of cross-selling

Challenge

major difficulty is that a large number of the rules found may be trivial for anyone familiar with the business

http://www.select-statistics.co.uk/article/blog-post/market-basket-analysis-understanding-customer-behaviour

http://www.statsoft.com/Solutions/Marketing/Market-Basket-Analysis

Terminology

- Items are the objects that we are identifying association between
- Association Rules a relation of the form X -> Y
  - If you have the item / items in the items set on the LHS then customer will be interested in the item Y on the RHS
- Support is the fraction of transactions in the dataset that contain the item or item set
- Confidence is the proportion of times the customer has taken the item Y given she has also taken X
- Lift is ratio of Confidence of the Rule divided by support of Product Y alone

MBA Calculations

- Let us assume you have the Transactions for a Retail Outlet
- **Transaction Summary**

  # Invoices = 10000

  # Invoices has Product A in the item set = 900

  # Invoices has Product B in the item set = 500

  # Invoice has both Products A & B in the item set = 350
- **Support Computation**

  Support of Product A = 900 / 10000 = 9%

  Support of Product B = 500 / 10000 = 5%
- **Rule A -> B (Customer who buy A also buys B)**

  Support of Product A & B = 350 / 10000 = 3.5%

  Confidence of Rule A -> B = 350 / 900 = 38.9% (%of customers who bought B from those who bought A)

  Lift = Confidence / Support of Product B = 38.9 / 5 = 7.77

  (Likelihood of customer purchasing product B is 7.77 times higher if the customer has purchased A)

Perform Market Basket Analysis in R

Data Import

greatlearning

```
## Author: Rajesh Jakhotia
## Company Name: K2 Analytics Finishing School Pvt. Ltd
## Email : ar.jakhotia@k2analytics.co.in
## Website : k2analytics.co.in

setwd("D:/K2Analytics/MarketBasketAnalysis")
getwd()

## Let us import the data that we need to perform the Market Basket Analysis
RTxn <- read.table("datafiles/Market_Basket_Analysis.csv", sep = ",", header = T)
nrow(RTxn)
```

```
[1] 3867
```

View the Data

Let us view and eye-ball the data

View(RTxn)

str(RTxn)

**RTxn$Invoice**_No <- as.factor(RTxn$Invoice_No)

Store_ID	Invoice_No	Till_No	Item_No	Txn_Date	SKU_Code	Item_Desc	Qty	Unit	Unit_Price	Price	Cust_ID	Emp_ID
1	100012	1	1	1-Jan-16	SKU032	Breakfast Cereals	0.25	Kg	55	13.75	23464	EMP001
1	100012	1	2	1-Jan-16	SKU076	Fruit Juices	0.50	Litre	67	33.50	23464	EMP001
1	100012	1	3	1-Jan-16	SKU208	Noodles	1.00	Pack	55	55.00	23464	EMP001
1	100012	1	4	1-Jan-16	SKU048	Cut Vegetables	0.25	Kg	67	16.75	23464	EMP001
1	100017	1	1	1-Jan-16	SKU004	Apple	0.25	Kg	220	55.00	23469	EMP001
1	100017	1	2	1-Jan-16	SKU283	Sauces & Salad Dressing	1.00	Pack	33	33.00	23469	EMP001
1	100018	1	1	1-Jan-16	SKU032	Breakfast Cereals	0.25	Kg	55	13.75	23470	EMP001
1	100018	1	2	1-Jan-16	SKU037	Buns	12.00	Unit	10	120.00	23470	EMP001
1	100018	1	3	1-Jan-16	SKU038	Butter	0.25	Kg	300	75.00	23470	EMP001
1	100018	1	4	1-Jan-16	SKU039	Cakes	0.25	Kg	650	162.50	23470	EMP001
1	100018	1	5	1-Jan-16	SKU040	Candles	12.00	Unit	10	120.00	23470	EMP001
1	100018	1	6	1-Jan-16	SKU041	Canned Food	1.00	Pack	35	35.00	23470	EMP001

Structure of Data

Understanding the data structure and data type of various columns

```
str(RTxn)
RTxn$Invoice_No <- as.factor(RTxn$Invoice_No)
```

```
'data.frame':	3867 obs. of  13 variables:
 $ Store_ID  : int  1 1 1 1 1 1 1 1 1 1 ...
 $ Invoice_No: int  100012 100012 100012 100012 100017 100017 100018 100018 100018 100018 ...
 $ Till_No   : int  1 1 1 1 1 1 1 1 1 1 ...
 $ Item_No   : int  1 2 3 4 1 2 1 2 3 4 ...
 $ Txn_Date  : Factor w/ 1 level "1-Jan-16": 1 1 1 1 1 1 1 1 1 1 ...
 $ SKU_Code  : Factor w/ 301 levels "SKU001","SKU002",..: 32 76 208 48 4 283 32 37 38 39 ...
 $ Item_Desc : Factor w/ 301 levels "Aerated Drinks",..: 33 80 205 51 5 279 33 39 40 41 ...
 $ Qty       : num  0.25 0.5 1 0.25 0.25 1 0.25 12 0.25 0.25 ...
 $ Unit      : Factor w/ 5 levels "Can","Kg","Litre",..: 2 3 4 2 2 4 2 5 2 2 ...
 $ Unit_Price: int  55 67 55 67 220 33 55 10 300 650 ...
 $ Price     : num  13.8 33.5 55 16.8 55 ...
 $ Cust_ID   : int  23464 23464 23464 23464 23469 23469 23470 23470 23470 23470 ...
 $ Emp_ID    : Factor w/ 9 levels "EMP001","EMP002",..: 1 1 1 1 1 1 1 1 1 1 ...
```

From structure we can see that Txn_Date should be casted to Date Format

Aggregating data at Transaction Level

Aggregating the Invoices at Transaction Level
We want one row per transaction.
The one row should have details of all the products purchased in that transaction

```
?split
Agg.RTxn <- split(RTxn$Item_Desc,RTxn$Invoice_No)
class(Agg.RTxn)
Agg.RTxn
## To see specific row number transaction
Agg.RTxn [105]
```

```
$`100352`
[1] Apple
301 Levels: Aerated Drinks Agarbatties Antiseptic Liquid Appalams Apple Atta Auto Accessories ... Wheat Vermicelli

$`100353`
[1] Agarbatties       Antiseptic Liquid
301 Levels: Aerated Drinks Agarbatties Antiseptic Liquid Appalams Apple Atta Auto Accessories ... Wheat Vermicelli

$`100355`
[1] Bandage      Bread        Butter       Moisturisers Rawa Sooji
301 Levels: Aerated Drinks Agarbatties Antiseptic Liquid Appalams Apple Atta Auto Accessories ... Wheat Vermicelli
```

Removing duplicates

```
##install.packages("arules")
library(arules)

## logic to remove duplicate items from the list
Agg.RTxn_DD <- list()
for (i in 1:length(Agg.RTxn)) {
  Agg.RTxn_DD[[i]] <- as.character(Agg.RTxn[[i]][!duplicated(Agg.RTxn[[i]])])
}

## converting transaction items from list format to transaction format
Txns <- as(Agg.RTxn_DD, "transactions")
```

Summarizing the Transactions

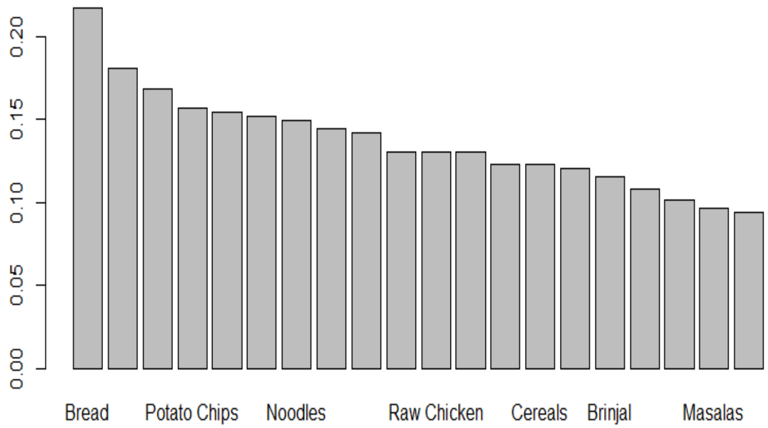
summary(Txns)

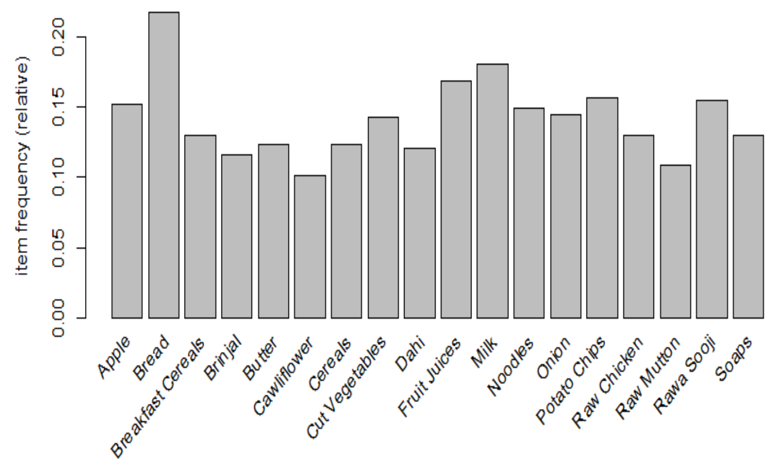
```
transactions as itemMatrix in sparse format with
 415 rows (elements/itemsets/transactions) and
 301 columns (items) and a density of 0.02783493

most frequent items:
       Bread          Milk Fruit Juices Potato Chips   Rawa Sooji      (Other)
          90            75           70           65           64         3113

element (itemset/transaction) length distribution:
sizes
 1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 27 28 29 31
79 67 36 25 23 21 18 18 16  8 10  9  6  6  4  7  5  4  4  5  3  4  4  3  2  3  1  3  2
32 33 35 36 37 38 40 41 44 46 47 49 50 52 53 65
 2  2  1  1  1  1  1  1  1  1  1  1  2  1  1  1

   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  1.000   2.000   5.000   8.378  10.500  65.000

includes extended item information - examples:
             labels
1    Aerated Drinks
2       Agarbatties
3 Antiseptic Liquid
```

inspect(Txns[10]) ## inspect specific transaction

Item Frequency Plot

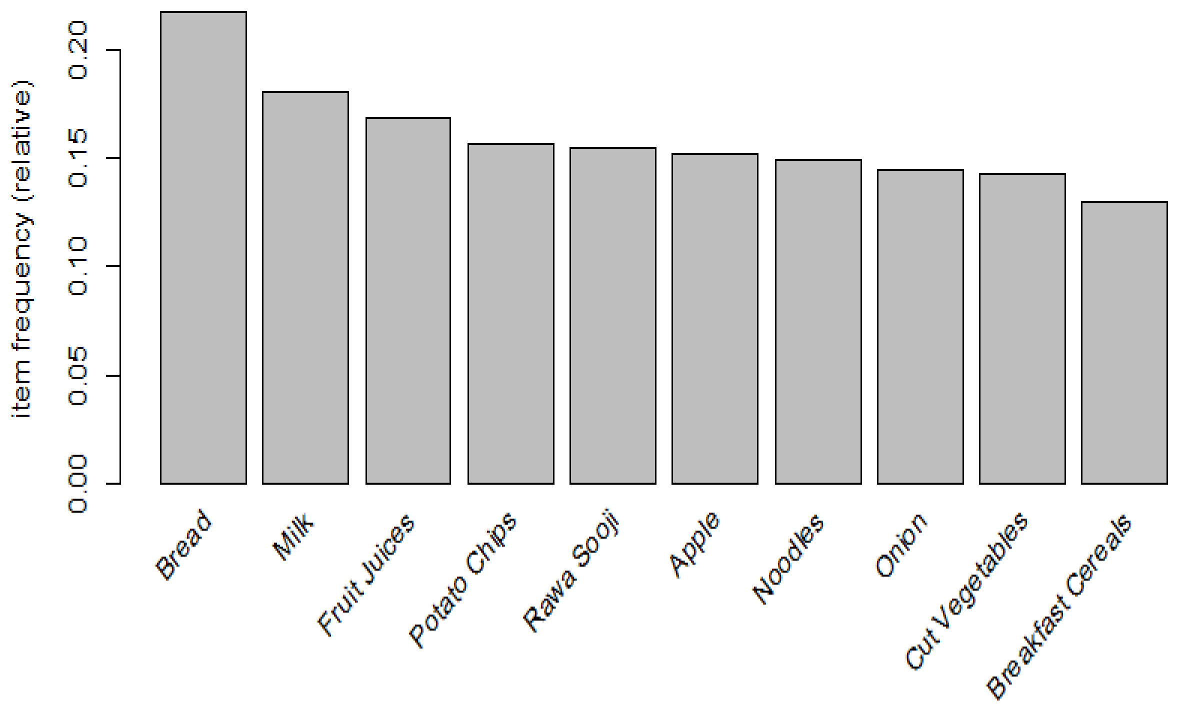
```
## Let us see the support
freq <- itemFrequency(Txns)
freq <- freq[order(-freq)]

freq["Bread"]

barplot(freq[1:20])

?itemFrequencyPlot

itemFrequencyPlot (
                Txns, support = 0.10)
```





Item Frequency Plot

itemFrequencyPlot (Txns, topN = 10)



Execute MBA

```
## install.packages("arulesViz")
library("arulesViz")
?apriori
arules1 <- apriori(data = Txns)
summary(arules1)
```

```
set of 4 rules

rule length distribution (lhs + rhs):sizes
2
4

   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
      2       2       2       2       2       2

summary of quality measures:
    support          confidence          lift
 Min.   :0.1036   Min.   :0.8148   Min.   :3.757
 1st Qu.:0.1054   1st Qu.:0.8266   1st Qu.:3.855
 Median :0.1120   Median :0.8368   Median :5.391
 Mean   :0.1114   Mean   :0.8671   Mean   :5.358
 3rd Qu.:0.1181   3rd Qu.:0.8774   3rd Qu.:6.893
 Max.   :0.1181   Max.   :0.9800   Max.   :6.893

mining info:
 data ntransactions support confidence
 Txns           415     0.1        0.8
```

Inspect the rules

greatlearning

See the Association Rules

inspect(arules1)

```
  lhs                    rhs              support   confidence lift
1 {Butter}            => {Bread}          0.1036145 0.8431373  3.887800
2 {Breakfast Cereals} => {Bread}          0.1060241 0.8148148  3.757202
3 {Dahi}              => {Cut Vegetables} 0.1180723 0.9800000  6.893220
4 {Cut Vegetables}    => {Dahi}           0.1180723 0.8305085  6.893220
```

inspect(sort(arule

```
  lhs                    rhs              support   confidence lift
3 {Dahi}              => {Cut Vegetables} 0.1180723 0.9800000  6.893220
4 {Cut Vegetables}    => {Dahi}           0.1180723 0.8305085  6.893220
1 {Butter}            => {Bread}          0.1036145 0.8431373  3.887800
2 {Breakfast Cereals} => {Bread}          0.1060241 0.8148148  3.757202
```

Execute MBA with parameters

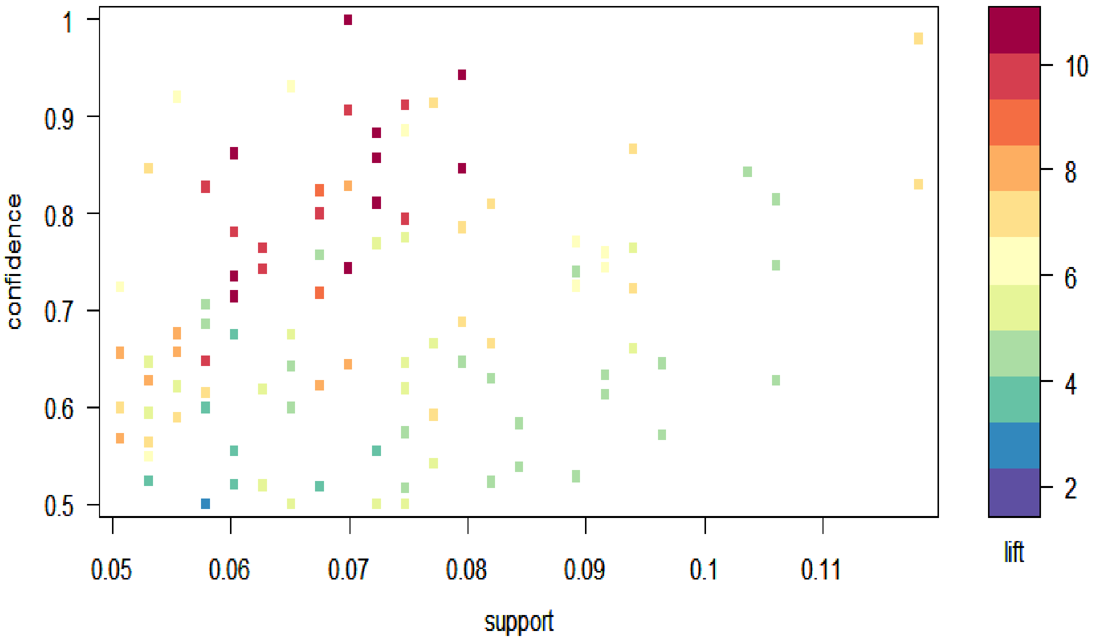
greatlearning

```
arules2 <- apriori(
        data = Txns, parameter = list(
        support = 0.05, confidence = 0.5, maxlen = 2
        )
)
```

```
Apriori

Parameter specification:
 confidence minval smax arem  aval originalSupport support minlen maxlen target   ext
        0.5    0.1    1 none FALSE            TRUE    0.05      1      2  rules FALSE

Algorithmic control:
 filter tree heap memopt load sort verbose
    0.1 TRUE TRUE  FALSE TRUE    2    TRUE

Absolute minimum support count: 20

set item appearances ...[0 item(s)] done [0.00s].
set transactions ...[301 item(s), 415 transaction(s)] done [0.00s].
sorting and recoding items ... [45 item(s)] done [0.00s].
creating transaction tree ... done [0.00s].
checking subsets of size 1 2 done [0.01s].
writing ... [152 rule(s)] done [0.00s].
creating S4 object  ... done [0.00s].
```

Graphically seeing the rules

```
plot ( arules2,control=list(
                    col = brewer.pal(11,"Spectral")
    ),
    main="Association Rules Plot"
)
```

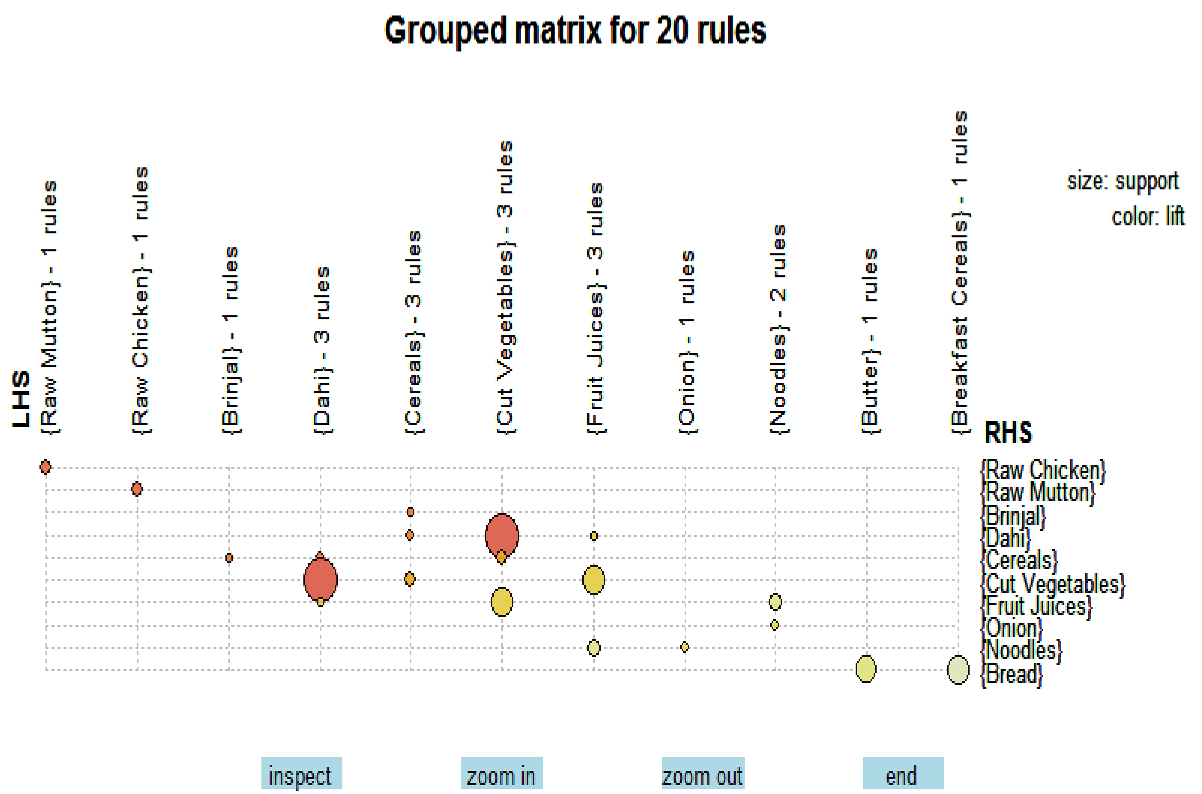


#Rules with high lift typically have low support.

Interactive Plot

Plot Interactivee Graphs

```
subrules2 <- head(sort(arules2, by="support"), 20)
plot(subrules2, method="grouped" , interactive=TRUE )
```



Exporting Rules to Excel for easy interpretation

greatlearning

```
rules_df <- as(arules2,"data.frame")
rules_df$lhs_suuport <- rules_df$support / rules_df$confidence;
rules_df$rhs_support <- rules_df$confidence / rules_df$lift;
View(rules_df)
write.table(rules_df, file = "output/mba_output.csv", sep = "," , append = F, row.names = F)
unlink("mba_output.csv")
```

	rules	support	confidence	lift	lhs_suuport	rhs_support
1	{Butter} => {Bread}	0.10361446	0.8431373	3.887800	0.12289157	0.21686747
2	{Banana} => {Apple}	0.05542169	0.9200000	6.060317	0.06024096	0.15180723
3	{Regular Eggs} => {Raw Chicken}	0.05301205	0.8461538	6.502849	0.06265060	0.13012048
4	{Other Cereals} => {Others}	0.06024096	0.8620690	10.522312	0.06987952	0.08192771
5	{Others} => {Other Cereals}	0.06024096	0.7352941	10.522312	0.08192771	0.06987952
6	{Other Cereals} => {Other Flours}	0.06024096	0.8620690	10.221675	0.06987952	0.08433735
7	{Other Flours} => {Other Cereals}	0.06024096	0.7142857	10.221675	0.08433735	0.06987952
8	{Other Cereals} => {Other Dals}	0.06987952	1.0000000	10.641026	0.06987952	0.09397590
9	{Other Dals} => {Other Cereals}	0.06987952	0.7435897	10.641026	0.09397590	0.06987952
10	{Other Cereals} => {Potato Chips}	0.05060241	0.7241379	4.623342	0.06987952	0.15662651

Thank you