Data Preparation_ data sources: SQL_Server/ Context Aware RS for Restaurants Project

Antoine.P _ from July to Octobre 2017

1) BUILDING A USER DATA SET

2) EXTRACTING RELEVANT EVENT INFORMATIONS

3) RESHAPING DATA AND GETTING LABELS FOR ALGORITHMS

4) FREQUENT ITEM SET_ APRIORI ALGORITHM

1) BUILDING A USER DATA SET

First, we will build a Data set aiming to link 2 tables: OrderDetail & OrderHeader. The table Product wil be also used to get product information: name, price….

Building this Data set has double objectives: the first is to create inputs , what we ’re going to do in this script, which will be fit to ML algorithms. The second is to get labels for those ML algorithms, these labels are simply ordered products we can extract from the Data set.

The following SQL code is to execute when extracting the 1st Data set from our Microsoft SQL Server as follow:

select D.OrderHeaderID, H.ID, D.ID, D.PersonID, D.ProductID, P.Name,D.ProductGroupID,D.IsSuggestion, P.Available, P.GrossPrice, P.NetPrice,‘D.NegociatedNetPrice’,‘D.Quantity’,P.WorkingOrder as PWorkingOrder, D.WorkingOrder as DWorkingOrder, H.DeviceID as HDeviceID, H.EmployeeID, H.NbDiners,H.CreationDatetime, H.LastEditionDatetime,D.WorkspaceLocation, H.ShopID, D.OrderHeader_ShopID from dbo.OrderDetail D left join dbo.OrderHeader H on H.ID= D.OrderHeaderID left join dbo.Product P on D.ProductID = P.ID where H.ShopID=4 and H.LastEditionDatetime<‘2017-08-31’ order by D.OrderHeaderID , PersonID

Importing required libraries

```
library(dplyr)
```

```
## Warning: package 'dplyr' was built under R version 3.4.3
```

```
##
## Attaching package: 'dplyr'
```

```
## The following objects are masked from 'package:stats':
##
##     filter, lag
```

```
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
library(reshape2)
```

```
## Warning: package 'reshape2' was built under R version 3.4.4
```

```
library(lubridate)
```

```
##
## Attaching package: 'lubridate'
```

```
## The following object is masked from 'package:base':
##
##     date
```

```
library(arules)
```

```
## Loading required package: Matrix
```

```
##
## Attaching package: 'arules'
```

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

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

```
library(arulesViz)
```

```
## Warning: package 'arulesViz' was built under R version 3.4.4
```

```
## Loading required package: grid
```

```
library(RColorBrewer)
```

Importing the raw data set

```
df= read.csv("C:/Users/Pham Antoine/Desktop/extractSQL2.csv", sep=";", header = TRUE, stringsAsFactors = F)
names(df)<- c('D.OrderHeaderID','H.ID', 'D.ID', 'D.PersonID', 'D.ProductID', 'P.Name','ProductGroupID','IsSugge
stion', 'P.Available', 'P.GrossPrice', 'P.NetPrice', 'D.NegociatedNetPrice','D.Quantity','P.WorkingOrder', 'D.W
orkingOrder', 'H.DeviceID', 'H.EmployeeID', 'H.NbDiners','H.CreationDatetime', 'H.LastEditionDatetime','D.Works
paceLocation', 'H.ShopID', 'D.OrderHeader_ShopID')
colnames(df)
```

```
##  [1] "D.OrderHeaderID"       "H.ID"
##  [3] "D.ID"                  "D.PersonID"
##  [5] "D.ProductID"           "P.Name"
##  [7] "ProductGroupID"        "IsSuggestion"
##  [9] "P.Available"           "P.GrossPrice"
## [11] "P.NetPrice"            "D.NegociatedNetPrice"
## [13] "D.Quantity"            "P.WorkingOrder"
## [15] "D.WorkingOrder"        "H.DeviceID"
## [17] "H.EmployeeID"          "H.NbDiners"
## [19] "H.CreationDatetime"    "H.LastEditionDatetime"
## [21] "D.WorkspaceLocation"   "H.ShopID"
## [23] "D.OrderHeader_ShopID"
```

```
str(df)
```

```
## 'data.frame':    42491 obs. of  23 variables:
##  $ D.OrderHeaderID      : int  1 1 1 1 1 1 1 1 1 1 ...
##  $ H.ID                 : int  1 1 1 1 1 1 1 1 1 1 ...
##  $ D.ID                 : int  57 58 62 63 64 68 69 74 75 76 ...
##  $ D.PersonID           : chr  "136" "136" "136" "136" ...
##  $ D.ProductID          : int  35 17 86 25 33 35 17 86 25 33 ...
##  $ P.Name               : chr  "MENU HAMBOURGEOIS" "MAXINUS" "EXPRESSO" "PATATEDOUCE" ...
##  $ ProductGroupID       : chr  "NULL" "NULL" "NULL" "NULL" ...
##  $ IsSuggestion         : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ P.Available          : int  1 1 1 1 1 1 1 1 1 1 ...
##  $ P.GrossPrice         : num  14.4 9 1.35 2.7 4.05 14.4 9 1.35 2.7 4.05 ...
##  $ P.NetPrice           : num  16 10 1.5 3 4.5 16 10 1.5 3 4.5 ...
##  $ D.NegociatedNetPrice : chr  "18.5000000" "NULL" "1.5000000" "NULL" ...
##  $ D.Quantity           : num  1 1 1 1 1 1 1 1 1 1 ...
##  $ P.WorkingOrder       : int  0 3 1 3 4 0 3 1 3 4 ...
##  $ D.WorkingOrder       : int  7 7 7 7 7 7 7 7 7 7 ...
##  $ H.DeviceID           : int  16 16 16 16 16 16 16 16 16 16 ...
##  $ H.EmployeeID         : chr  "NULL" "NULL" "NULL" "NULL" ...
##  $ H.NbDiners           : int  4 4 4 4 4 4 4 4 4 4 ...
##  $ H.CreationDatetime   : chr  "2016-02-23 12:08:06.560" "2016-02-23 12:08:06.560" "2016-02-23 12:08:06.560"
 "2016-02-23 12:08:06.560" ...
##  $ H.LastEditionDatetime: chr  "2016-02-23 20:05:44.963" "2016-02-23 20:05:44.963" "2016-02-23 20:05:44.963"
 "2016-02-23 20:05:44.963" ...
##  $ D.WorkspaceLocation  : chr  "NULL" "NULL" "NULL" "NULL" ...
##  $ H.ShopID             : int  4 4 4 4 4 4 4 4 4 4 ...
##  $ D.OrderHeader_ShopID : int  4 4 4 4 4 4 4 4 4 4 ...
```

Creating 2 functions to convertir variables between factor and numeric class

```
to.numerics<- function ( df,variables){
  for (variable in variables){
    df[[variable]]<- as.numeric(df[[variable]])

  }
  return(df)
}
```

```
to.factors<- function ( df,variables){
  for (variable in variables){
    df[[variable]]<- as.factor(df[[variable]])
  }
  return(df)
}
```

Applying these function to convert variables to required type

```
factor_vars<-c('D.PersonID','P.Name','ProductGroupID','IsSuggestion','H.DeviceID','H.EmployeeID','H.CreationDat
etime','H.LastEditionDatetime','D.WorkspaceLocation')
df<-to.factors(df,factor_vars)
```

```
df$D.NegociatedNetPrice<-as.numeric(df$D.NegociatedNetPrice,na.rm=TRUE)# change the typ of this variable before
 assigning 0 to records having null value
```

```
## Warning: NAs introduced by coercion
```

Checking NA values

```
sapply(df, function(x) sum ( is.na(x)))
```

```
##       D.OrderHeaderID                  H.ID                  D.ID
##                     0                     0                     0
##            D.PersonID           D.ProductID                P.Name
##                     0                     0                     0
##        ProductGroupID          IsSuggestion           P.Available
##                     0                     0                     0
##          P.GrossPrice            P.NetPrice  D.NegociatedNetPrice
##                     0                     0                 12132
##            D.Quantity        P.WorkingOrder        D.WorkingOrder
##                     0                     0                     0
##            H.DeviceID          H.EmployeeID            H.NbDiners
##                     0                     0                     0
##    H.CreationDatetime H.LastEditionDatetime   D.WorkspaceLocation
##                     0                     0                     0
##              H.ShopID  D.OrderHeader_ShopID
##                     0                     0
```

if Na value replace by 0

```
df[is.na(df)]<-0
```

```
str(df)
```

```
## 'data.frame':    42491 obs. of  23 variables:
##  $ D.OrderHeaderID      : int  1 1 1 1 1 1 1 1 1 1 ...
##  $ H.ID                 : int  1 1 1 1 1 1 1 1 1 1 ...
##  $ D.ID                 : int  57 58 62 63 64 68 69 74 75 76 ...
##  $ D.PersonID           : Factor w/ 9543 levels "-1","10000","10001",..: 1457 1457 1457 1457 1457 1457 1457
1457 1457 1457 ...
##  $ D.ProductID          : int  35 17 86 25 33 35 17 86 25 33 ...
##  $ P.Name               : Factor w/ 156 levels "ABATILLE","ABATILLES PLATES",..: 86 76 42 105 134 86 76 42 1
05 134 ...
##  $ ProductGroupID       : Factor w/ 25 levels "11","12","13",..: 25 25 25 25 25 25 25 25 25 25 ...
##  $ IsSuggestion         : Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 1 1 1 ...
##  $ P.Available          : int  1 1 1 1 1 1 1 1 1 1 ...
##  $ P.GrossPrice         : num  14.4 9 1.35 2.7 4.05 14.4 9 1.35 2.7 4.05 ...
##  $ P.NetPrice           : num  16 10 1.5 3 4.5 16 10 1.5 3 4.5 ...
##  $ D.NegociatedNetPrice : num  18.5 0 1.5 0 0 18.5 0 1.5 0 0 ...
##  $ D.Quantity           : num  1 1 1 1 1 1 1 1 1 1 ...
##  $ P.WorkingOrder       : int  0 3 1 3 4 0 3 1 3 4 ...
##  $ D.WorkingOrder       : int  7 7 7 7 7 7 7 7 7 7 ...
##  $ H.DeviceID           : Factor w/ 359 levels "2","3","4","5",..: 14 14 14 14 14 14 14 14 14 14 ...
##  $ H.EmployeeID         : Factor w/ 1 level "NULL": 1 1 1 1 1 1 1 1 1 1 ...
##  $ H.NbDiners           : int  4 4 4 4 4 4 4 4 4 4 ...
##  $ H.CreationDatetime   : Factor w/ 4459 levels "2016-02-23 12:08:06.560",..: 1 1 1 1 1 1 1 1 1 1 ...
##  $ H.LastEditionDatetime: Factor w/ 4459 levels "2016-02-23 20:05:44.963",..: 1 1 1 1 1 1 1 1 1 1 ...
##  $ D.WorkspaceLocation  : Factor w/ 5 levels "0","1","2","3",..: 5 5 5 5 5 5 5 5 5 5 ...
##  $ H.ShopID             : int  4 4 4 4 4 4 4 4 4 4 ...
##  $ D.OrderHeader_ShopID : int  4 4 4 4 4 4 4 4 4 4 ...
```

```
head(df,2)
```

```
##   D.OrderHeaderID H.ID D.ID D.PersonID D.ProductID            P.Name
## 1               1    1   57        136          35 MENU HAMBOURGEOIS
## 2               1    1   58        136          17           MAXINUS
##   ProductGroupID IsSuggestion P.Available P.GrossPrice P.NetPrice
## 1           NULL            0           1         14.4         16
## 2           NULL            0           1          9.0         10
##   D.NegociatedNetPrice D.Quantity P.WorkingOrder D.WorkingOrder H.DeviceID
## 1                 18.5          1              0              7         16
## 2                  0.0          1              3              7         16
##   H.EmployeeID H.NbDiners      H.CreationDatetime   H.LastEditionDatetime
## 1         NULL          4 2016-02-23 12:08:06.560 2016-02-23 20:05:44.963
## 2         NULL          4 2016-02-23 12:08:06.560 2016-02-23 20:05:44.963
##   D.WorkspaceLocation H.ShopID D.OrderHeader_ShopID
## 1                NULL        4                    4
## 2                NULL        4                    4
```

What are restaurants (Shop_ID) included in the data set?

```
unique(df$D.OrderHeader_ShopID)
```

```
## [1] 4 6
```

Excluding records related to the Shop_ID 6

```
#library(dplyr)
df1=filter(df,D.OrderHeader_ShopID==4)
```

```
unique(df1$H.ShopID)
```

```
## [1] 4
```

Removing records where PersonID == NULL.

Note that if PersonID=NULL , the record has no ID in the table ‘OrderDetail’, so not easy to link to the table Event

```
df1=subset(df1, D.PersonID !='NULL')
```

```
print(dim(df))
```

```
## [1] 42491    23
```

```
print(dim(df1))
```

```
## [1] 39316    23
```

```
print(colnames(df1))
```

```
##  [1] "D.OrderHeaderID"       "H.ID"
##  [3] "D.ID"                  "D.PersonID"
##  [5] "D.ProductID"           "P.Name"
##  [7] "ProductGroupID"        "IsSuggestion"
##  [9] "P.Available"           "P.GrossPrice"
## [11] "P.NetPrice"            "D.NegociatedNetPrice"
## [13] "D.Quantity"            "P.WorkingOrder"
## [15] "D.WorkingOrder"        "H.DeviceID"
## [17] "H.EmployeeID"          "H.NbDiners"
## [19] "H.CreationDatetime"    "H.LastEditionDatetime"
## [21] "D.WorkspaceLocation"   "H.ShopID"
## [23] "D.OrderHeader_ShopID"
```

Checking the whole information concerning a given customer

```
filter(df1,D.PersonID==178)
```

```
##   D.OrderHeaderID H.ID D.ID D.PersonID D.ProductID        P.Name
## 1               1    1   49        178         105     MENU PLAT
## 2               1    1   50        178          86      EXPRESSO
## 3               1    1   51        178           8       TARTARE
## 4               1    1   52        178          29 GATEAUCAROTTE
##   ProductGroupID IsSuggestion P.Available P.GrossPrice P.NetPrice
## 1           NULL            0           1        14.40       16.0
## 2           NULL            0           1         1.35        1.5
## 3           NULL            0           1        11.25       12.5
## 4           NULL            0           1         4.05        4.5
##   D.NegociatedNetPrice D.Quantity P.WorkingOrder D.WorkingOrder H.DeviceID
## 1                 16.0          1              0              7         16
## 2                  1.5          1              1              7         16
## 3                  0.0          1              3              7         16
## 4                  0.0          1              4              7         16
##   H.EmployeeID H.NbDiners      H.CreationDatetime   H.LastEditionDatetime
## 1         NULL          4 2016-02-23 12:08:06.560 2016-02-23 20:05:44.963
## 2         NULL          4 2016-02-23 12:08:06.560 2016-02-23 20:05:44.963
## 3         NULL          4 2016-02-23 12:08:06.560 2016-02-23 20:05:44.963
## 4         NULL          4 2016-02-23 12:08:06.560 2016-02-23 20:05:44.963
##   D.WorkspaceLocation H.ShopID D.OrderHeader_ShopID
## 1                NULL        4                    4
## 2                NULL        4                    4
## 3                NULL        4                    4
## 4                NULL        4                    4
```

We will be calculating for each customer the total of times the customer has visited the restaurant and his average ticket as well

```
# Attention: Using the package ' funModelling' may cause issue to the  function 'summarise' of 'dpyr' package
tab1=df1 %>%
  group_by(D.PersonID) %>%
    summarise(nb_visits=length(unique(H.ID)),
              avg_ticketU=sum(D.NegociatedNetPrice*D.Quantity)/length(unique(H.ID)))
```

```
# cheking with the customer ID=539
filter(tab1,D.PersonID==539)
```

```
## # A tibble: 1 x 3
##   D.PersonID nb_visits avg_ticketU
##   <fct>          <int>       <dbl>
## 1 539               13        14.9
```

Make the list of price for each product

```
It_netprice<-df1%>%group_by(P.Name)%>%summarise(NetPrice=unique(P.NetPrice))
```

```
head(It_netprice,5)
```

```
## # A tibble: 5 x 2
##   P.Name           NetPrice
##   <fct>               <dbl>
## 1 ABATILLES PLATES     3.50
## 2 ABATILLES RED        3.50
## 3 AVECESAR            12.5
## 4 BADOIT 33cl          3.00
## 5 BAILEYS              5.00
```

```
tab2=subset(df1,select=c('D.OrderHeaderID','D.PersonID','P.Name','H.NbDiners','D.Quantity'))
```

Using the ‘dcast’ function (equivalent in Python: https://stackoverflow.com/questions/36970264/pandas-equivalent-for-r-dcast (https://stackoverflow.com/questions/36970264/pandas-equivalent-for-r-dcast))

```
#library(reshape2)
It_nbdinner<-dcast(tab2, H.NbDiners~P.Name, value.var = 'D.Quantity',fun.aggregate = sum)
```

```
subset(It_nbdinner,select=c(1:2))
```

```
##    H.NbDiners ABATILLES PLATES
## 1           0                0
## 2           1               18
## 3           2               21
## 4           3                8
## 5           4               23
## 6           5                2
## 7           6                2
## 8           7                0
## 9           8                0
## 10          9                0
## 11         10                0
## 12         20                0
```

```
subset(df1,H.NbDiners==8, select=c('H.NbDiners','P.Name','D.Quantity'))[c(1:5),1:3]
```

```
##       H.NbDiners            P.Name D.Quantity
## 32302          8      PUNCH Maison          2
## 32303          8              BRIE          1
## 32304          8           NUGGETS          1
## 32305          8        MAXIFLETTE          1
## 32306          8 MENU HAMBOURGEOIS          1
```

```
df1%>%group_by(H.NbDiners)%>%summarise('count_nb'=length(unique(D.OrderHeaderID)))
```

```
## # A tibble: 12 x 2
##    H.NbDiners count_nb
##         <int>    <int>
##  1          0       10
##  2          1     1095
##  3          2     2068
##  4          3      570
##  5          4      546
##  6          5       23
##  7          6        8
##  8          7        3
##  9          8        7
## 10          9        2
## 11         10        1
## 12         20        1
```

```
It_nbdiner<-merge(x=df1%>%group_by(H.NbDiners)%>%summarise('count_nb'=length(unique(D.OrderHeaderID))),y = It_n
bdinner,by = 'H.NbDiners', all.y=T)
```

```
dim(It_nbdiner)
```

```
## [1]  12 127
```

```
subset(df1,H.NbDiners=='2'& P.Name=='ABATILLES PLATES')[c(1:3),1:4]
```

```
##      D.OrderHeaderID H.ID  D.ID D.PersonID
## 384               27   27   432        436
## 2238             268  268  4517       6478
## 8486            1010 1010 10823       9817
```

Getting all returning customers (nb_visits>1)

```
re_cust<-filter(tab1, nb_visits !='1') # to find out returning customers
unique(re_cust$nb_visits)
```

```
## [1]  2  3 10  9  4  5 13
```

```
re_cust
```

```
## # A tibble: 326 x 3
##    D.PersonID nb_visits avg_ticketU
##    <fct>          <int>       <dbl>
##  1 10077              2       24.2
##  2 10090              2       10.8
##  3 10091              2       10.4
##  4 10099              2       12.0
##  5 10109              2       36.0
##  6 10167              2       18.2
##  7 10216              2       31.5
##  8 10219              2       10.0
##  9 10223              2       15.5
## 10 10236              2        8.75
## # ... with 316 more rows
```

```
table(re_cust$nb_visits)
```

```
##
##   2   3   4   5   9  10  13
## 285  32   3   3   1   1   1
```

And all new customers

```
newcust<-filter(tab1, nb_visits=='1') # subsetting new customers
```

```
head(newcust,2)
```

```
## # A tibble: 2 x 3
##   D.PersonID nb_visits avg_ticketU
##   <fct>          <int>       <dbl>
## 1 -1                 1        21.3
## 2 10000              1        19.2
```

checking with the customer ID N°-1

```
filter(df1,D.PersonID==-1)[c(1:5),1:5]
```

```
##    D.OrderHeaderID H.ID D.ID D.PersonID D.ProductID
## 1               50   50  581         -1          47
## 2               50   50  582         -1          81
## 3               50   50  583         -1          81
## 4               50   50  584         -1          80
## NA              NA   NA   NA       <NA>          NA
```

```
print(dim(tab1))
```

```
## [1] 9541    3
```

```
print(dim(newcust))
```

```
## [1] 9215    3
```

```
print(dim(re_cust))
```

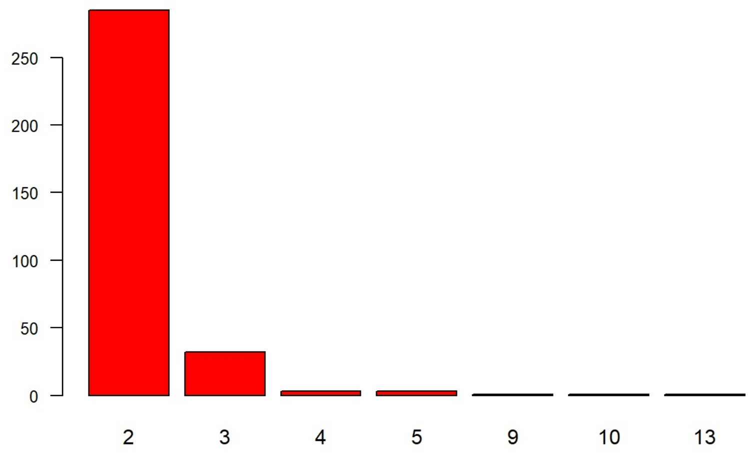
```
## [1] 326   3
```

```
print(dim(It_nbdiner))
```

```
## [1]  12 127
```

Now, plotting a bar chart to display the number of clients for each number of visite

```
barplot(table(re_cust$nb_visits),
        main= '',las=1,col = re_cust$nb_visits,cex.axis=0.8,cex.names=1)
```



Who is the customer having visited the restaurant 13 times?

```
subset(re_cust,nb_visits=='13', select = c(D.PersonID))
```

```
## # A tibble: 1 x 1
##   D.PersonID
##   <fct>
## 1 539
```

It's the Customer with Person ID =539 and his profile as follow:

```
subset(df1,D.PersonID==539)[c(1:5),1:7]
```

```
##     D.OrderHeaderID H.ID D.ID D.PersonID D.ProductID            P.Name
## 448              37   37  496        539          35 MENU HAMBOURGEOIS
## 449              37   37  497        539          16        MAXHALEINE
## 450              37   37  498        539          35 MENU HAMBOURGEOIS
## 451              37   37  499        539          19        MAXIFLETTE
## 452              37   37  500        539          26       SALADEASIAT
##     ProductGroupID
## 448           NULL
## 449           NULL
## 450           NULL
## 451           NULL
## 452           NULL
```

Looking again at the tail of data set

```
tail(df1)
```

```
##       D.OrderHeaderID H.ID   D.ID D.PersonID D.ProductID      P.Name
## 40334            4502 4502 293323      43438          24 GROSSEFRITE
## 40335            4502 4502 293314      43439          23   PLANTAMAX
## 40336            4502 4502 293315      43439          25 PATATEDOUCE
## 40338            4503 4503 293333      43440          21  MAXPARTOUT
## 40339            4503 4503 293334      43440          25 PATATEDOUCE
## 40340            4503 4503 293331      43440          21  MAXPARTOUT
##       ProductGroupID IsSuggestion P.Available P.GrossPrice P.NetPrice
## 40334              5            1           1          2.7          3
## 40335              4            0           1          9.0         10
## 40336              5            1           1          2.7          3
## 40338              5            0           1          9.0         10
## 40339              5            0           1          2.7          3
## 40340              5            0           1          9.0         10
##       D.NegociatedNetPrice D.Quantity P.WorkingOrder D.WorkingOrder
## 40334                 0.00          1              3              3
## 40335                13.00          1              3              3
## 40336                 0.00          1              3              3
## 40338                 9.88          1              3              3
## 40339                 0.00          1              3              3
## 40340                 9.88          1              3              3
##       H.DeviceID H.EmployeeID H.NbDiners      H.CreationDatetime
## 40334          3         NULL          4 2017-08-30 19:34:25.793
## 40335          3         NULL          4 2017-08-30 19:34:25.793
## 40336          3         NULL          4 2017-08-30 19:34:25.793
## 40338      20141         NULL          2 2017-08-30 19:50:26.333
## 40339      20141         NULL          2 2017-08-30 19:50:26.333
## 40340      20141         NULL          2 2017-08-30 19:50:26.333
##         H.LastEditionDatetime D.WorkspaceLocation H.ShopID
## 40334 2017-08-30 20:08:50.687                   3        4
## 40335 2017-08-30 20:08:50.687                   2        4
## 40336 2017-08-30 20:08:50.687                   2        4
## 40338 2017-08-30 20:38:25.343                   0        4
## 40339 2017-08-30 20:38:25.343                   0        4
## 40340 2017-08-30 20:38:25.343                   0        4
##       D.OrderHeader_ShopID
## 40334                    4
## 40335                    4
## 40336                    4
## 40338                    4
## 40339                    4
## 40340                    4
```

Creating a column containing this binary value: O if the customer is new , 1 otherwise

```
df1$user_visit<-ifelse(df1$D.PersonID %in% newcust$D.PersonID,0,1)
table(df1$user_visit)# to check the distribution of this new variableb
```

```
## 
##     0     1 
## 36665  2651
```

We’re going to do a check

```
table(merge(x=df1,y=tab1,by='D.PersonID',x.all=TRUE)$nb_visits) # We will create a data set by this function la
ter
```

```
## 
##     1     2     3     4     5     9    10    13 
## 36665  1938   363    91    79    82    65    33
```

It looks correct since we have obtained the same result by 2 computing methods

Checking more

```
filter(df1,D.PersonID==136)[c(1:5),1:6]
```

```
##   D.OrderHeaderID H.ID D.ID D.PersonID D.ProductID            P.Name
## 1               1    1   57        136          35 MENU HAMBOURGEOIS
## 2               1    1   58        136          17           MAXINUS
## 3               1    1   62        136          86          EXPRESSO
## 4               1    1   63        136          25       PATATEDOUCE
## 5               1    1   64        136          33        SOUPEFRUIT
```

In order to compute the sold quantity of an item for each value of number of visits of customer, we are going to create a column in the data set df1 that shows the number of visits of each customer

```
print(colnames(re_cust))
```

```
## [1] "D.PersonID"  "nb_visits"   "avg_ticketU"
```

```
print(dim(re_cust))
```

```
## [1] 326   3
```

```
#df1$nb_visits<-ifelse(df1$D.PersonID %in% re_cust$D.PersonID,re_cust$nb_visits,1)# To check if it is not wrong
```

We reuse the function dcast for computing the number of occurence of each item by each number of visits of customer

Make a data set containing the list of sold Items and merging it with nb_visits variable in the ‘x’ data set (see also the above check with function ‘merge’)

```
tab3=subset(merge(x=df1,y=tab1,by='D.PersonID',x.all=TRUE), select=c('D.OrderHeaderID','D.PersonID','P.Name','n
b_visits','D.Quantity'))
```

```
print(dim(df1))
```

```
## [1] 39316    24
```

```
print(dim(tab3))
```

```
## [1] 39316     5
```

```
print(table(tab3$nb_visits))
```

```
## 
##     1     2     3     4     5     9    10    13 
## 36665  1938   363    91    79    82    65    33
```

Apply the function dcast to make a data set containing for each value of the visit frequency the count of each sold item

```
IT_return_cust<-dcast(tab3,nb_visits~P.Name, value.var = 'D.Quantity',fun.aggregate = sum)
```

```
head(IT_return_cust,2)[1:5] # we have to add a column to show the number of customers related to each value of
nb_visits
```

```
##   nb_visits ABATILLES PLATES ABATILLES RED AVECESAR BADOIT 33cl
## 1         1               67           103      178          59
## 2         2                6             8       11           9
```

Calculate again the number of customers for each value of nb_visits

```
table(tab1$nb_visits)
```

```
## 
##    1    2    3    4    5    9   10   13 
## 9215  285   32    3    3    1    1    1
```

```
as.data.frame(table(tab1$nb_visits))
```

```
##   Var1 Freq
## 1    1 9215
## 2    2  285
## 3    3   32
## 4    4    3
## 5    5    3
## 6    9    1
## 7   10    1
## 8   13    1
```

Adding this feature to IT_return_cust

```
IT_return_cust<-merge( y =as.data.frame(table(tab1$nb_visits)),x=IT_return_cust, by.x ='nb_visits', by.y='Var1'
, all.x = TRUE )
```

```
IT_return_cust<-IT_return_cust[c(1,127,2:126)]
```

```
names(IT_return_cust)[2]<-c('count_nb')
```

```
dim(IT_return_cust)
```

```
## [1]   8 127
```

Making a check with values represented by df1 data set. It looks correct!!

```
df1%>%group_by(P.Name)%>%summarise('sold.quanti'=sum(D.Quantity))
```

```
## # A tibble: 125 x 2
##    P.Name             sold.quanti
##    <fct>                    <dbl>
##  1 ABATILLES PLATES           74.
##  2 ABATILLES RED             114.
##  3 AVECESAR                  196.
##  4 BADOIT 33cl                69.
##  5 BAILEYS                    35.
##  6 BIERE SANS GLUTEN          19.
##  7 Boisson Rouge               1.
##  8 BRIE                      248.
##  9 CAFEGOURMAND             1286.
## 10 CAFEGOURMANDMENU          370.
## # ... with 115 more rows
```

```
filter(tab3,nb_visits==13)[c(1:5),1:5]
```

```
##   D.OrderHeaderID D.PersonID            P.Name nb_visits D.Quantity
## 1              37        539      CAFEGOURMAND        13          1
## 2              40        539              DECA        13          1
## 3              40        539         THE GLACE        13          1
## 4              37        539       GROSSEFRITE        13          1
## 5              40        539 MENU HAMBOURGEOIS        13          1
```

```
filter(df1,D.PersonID==539)[c(1:5),1:5]
```

```
##   D.OrderHeaderID H.ID D.ID D.PersonID D.ProductID
## 1              37   37  496        539          35
## 2              37   37  497        539          16
## 3              37   37  498        539          35
## 4              37   37  499        539          19
## 5              37   37  500        539          26
```

```
unique(df1$D.WorkspaceLocation)
```

```
## [1] NULL 2    0    3    1   
## Levels: 0 1 2 3 NULL
```

```
table(df$D.WorkspaceLocation)
```

```
## 
##     0     1     2     3  NULL 
##   812   301   682   268 40428
```

```
table((df1%>%group_by(D.OrderHeaderID,D.PersonID)%>%summarise(tab2=unique(D.WorkspaceLocation)))$tab2) #b=unique(D.WorkspaceLocation))
```

```
## 
##    0    1    2    3 NULL 
##  187   64  158   60 9471
```

As we can see at this variable ‘D.WorkspaceLocation’, many rows which have a nulle value

So far, we’ve obtained a data frame (tab1) describing the number of visits and the average tiket of an given customer.

Looking again at this data frame

```
nb_visit<-tab1
head(nb_visit)
```

```
## # A tibble: 6 x 3
##   D.PersonID nb_visits avg_ticketU
##   <fct>          <int>       <dbl>
## 1 -1                 1        21.3
## 2 10000              1        19.2
## 3 10001              1        25.0
## 4 10002              1        22.5
## 5 10004              1        17.0
## 6 10005              1        16.0
```

```
colnames(df1)
```

```
##  [1] "D.OrderHeaderID"       "H.ID"                 
##  [3] "D.ID"                  "D.PersonID"           
##  [5] "D.ProductID"           "P.Name"               
##  [7] "ProductGroupID"        "IsSuggestion"         
##  [9] "P.Available"           "P.GrossPrice"         
## [11] "P.NetPrice"            "D.NegociatedNetPrice" 
## [13] "D.Quantity"            "P.WorkingOrder"       
## [15] "D.WorkingOrder"        "H.DeviceID"           
## [17] "H.EmployeeID"          "H.NbDiners"           
## [19] "H.CreationDatetime"    "H.LastEditionDatetime"
## [21] "D.WorkspaceLocation"   "H.ShopID"             
## [23] "D.OrderHeader_ShopID"  "user_visit"
```

```
(df1%>%group_by(P.Name)%>%summarise(P.NetPrice=unique(P.NetPrice),ProductGroupID=mode(ProductGroupID)))[c(1:3),
1:3]
```

```
## # A tibble: 3 x 3
##   P.Name           P.NetPrice ProductGroupID
##   <fct>                 <dbl> <chr>
## 1 ABATILLES PLATES       3.50 numeric
## 2 ABATILLES RED          3.50 numeric
## 3 AVECESAR              12.5  numeric
```

What we ‘re going to do now is to create a new data Set by using ’group_by’ function to create only one row for each pairH.ID+PersonID. We will be selecting by ‘summarise’ function informations which seem relevant for futur analysis

```
tab4=df1%>%group_by(D.OrderHeaderID,H.ID,D.PersonID)%>%summarise(DeviceID=unique(H.DeviceID),
                                                             H.CreationDatetime=unique(H.CreationDatetime),
                                                             H.H.LastEditionDatetime=unique(H.LastEditionDatet
ime),
                                                             D.WorkspaceLocation=unique(D.WorkspaceLocation),
                                                             user_visit=unique(user_visit),
                                                             H.NbDiners = unique(H.NbDiners))
                                                            # 0 if the customer comme the 1st time , 1 otherwi
se
tail(tab4)
```

```
## # A tibble: 6 x 9
## # Groups:   D.OrderHeaderID, H.ID [3]
##   D.OrderHeaderID  H.ID D.PersonID DeviceID H.CreationDatetime
##             <int> <int> <fct>      <fct>    <fct>
## 1            4501  4501 43435      6        2017-08-30 19:11:07.753
## 2            4502  4502 43436      3        2017-08-30 19:34:25.793
## 3            4502  4502 43437      3        2017-08-30 19:34:25.793
## 4            4502  4502 43438      3        2017-08-30 19:34:25.793
## 5            4502  4502 43439      3        2017-08-30 19:34:25.793
## 6            4503  4503 43440      20141    2017-08-30 19:50:26.333
## # ... with 4 more variables: H.H.LastEditionDatetime <fct>,
## #   D.WorkspaceLocation <fct>, user_visit <dbl>, H.NbDiners <int>
```

Merging 2 data frames (tab1/nb_visit & tab4),we will be applying a left join function to conserve all rows in (tab4). This join will be based on PersonID column, so for returning customers, values which are represented by (tab1) can be duplicated

```
tab4<- merge(x=tab4,y=nb_visit,by='D.PersonID',all.x = TRUE )# if the 1st visit , nb_visit= 1 ans so on...
tab4<-arrange(tab4,desc(H.ID))
head(tab4,2)
```

```
##   D.PersonID D.OrderHeaderID H.ID DeviceID      H.CreationDatetime
## 1      43440            4503 4503    20141 2017-08-30 19:50:26.333
## 2      43436            4502 4502        3 2017-08-30 19:34:25.793
##   H.H.LastEditionDatetime D.WorkspaceLocation user_visit H.NbDiners
## 1 2017-08-30 20:38:25.343                   0          0          2
## 2 2017-08-30 20:08:50.687                   1          0          4
##   nb_visits avg_ticketU
## 1         1       19.76
## 2         1       17.00
```

```
filter(tab4,D.PersonID==219)
```

```
##   D.PersonID D.OrderHeaderID H.ID DeviceID      H.CreationDatetime
## 1        219             158  158      110 2016-05-04 12:39:27.253
## 2        219             137  137      110 2016-04-20 17:44:05.440
## 3        219             135  135      304 2016-04-20 11:10:30.867
##   H.H.LastEditionDatetime D.WorkspaceLocation user_visit H.NbDiners
## 1 2016-05-04 12:39:27.253                NULL          1          1
## 2 2016-04-20 17:44:05.440                NULL          1          1
## 3 2016-04-20 11:10:30.867                NULL          1          1
##   nb_visits avg_ticketU
## 1         3          20
## 2         3          20
## 3         3          20
```

```
dim(tab4)
```

```
## [1] 9940   11
```

Working with date&time data

```
#library(lubridate)
tab4$Date<-date(tab4$H.CreationDatetime)
tail(tab4,3)
```

```
##      D.PersonID D.OrderHeaderID H.ID DeviceID      H.CreationDatetime
## 9938        220               1    1       16 2016-02-23 12:08:06.560
## 9939        221               1    1       16 2016-02-23 12:08:06.560
## 9940        222               1    1       16 2016-02-23 12:08:06.560
##      H.H.LastEditionDatetime D.WorkspaceLocation user_visit H.NbDiners
## 9938 2016-02-23 20:05:44.963                NULL          0          4
## 9939 2016-02-23 20:05:44.963                NULL          0          4
## 9940 2016-02-23 20:05:44.963                NULL          0          4
##      nb_visits avg_ticketU       Date
## 9938         1         7.5 2016-02-23
## 9939         1        32.0 2016-02-23
## 9940         1        36.0 2016-02-23
```

Calculating purchasing frequency for each customer

```
tab5<-tab4%>%group_by(D.PersonID)%>%summarise(as.numeric(max(Date)-min(Date))/as.numeric(unique(nb_visits)))# (
max date _min date)/nb_visite
names(tab5)<-c('D.PersonID','pch_freq')
head(tab5,2)
```

```
## # A tibble: 2 x 2
##   D.PersonID pch_freq
##   <fct>         <dbl>
## 1 -1               0.
## 2 10000            0.
```

```
filter(tab5,D.PersonID==136)
```

```
## # A tibble: 1 x 2
##   D.PersonID pch_freq
##   <fct>         <dbl>
## 1 136            19.8
```

Merging this variable describing time dimension to our data set

```
tab4<-merge(x=tab4,y=tab5,by='D.PersonID',all.x =T)
tab4<-arrange(tab4,desc(H.CreationDatetime))
```

Checking the dimension of the data set

```
dim(tab4)
```

```
## [1] 9940   13
```

Distribution of variable ‘WorkspaceLocation’

```
table(tab4$D.WorkspaceLocation)
```

```
## 
##    0    1    2    3 NULL 
##  187   64  158   60 9471
```

```
filter(tab4,D.PersonID==219)
```

```
##   D.PersonID D.OrderHeaderID H.ID DeviceID      H.CreationDatetime
## 1        219             158  158      110 2016-05-04 12:39:27.253
## 2        219             137  137      110 2016-04-20 17:44:05.440
## 3        219             135  135      304 2016-04-20 11:10:30.867
##   H.H.LastEditionDatetime D.WorkspaceLocation user_visit H.NbDiners
## 1 2016-05-04 12:39:27.253                NULL          1          1
## 2 2016-04-20 17:44:05.440                NULL          1          1
## 3 2016-04-20 11:10:30.867                NULL          1          1
##   nb_visits avg_ticketU       Date pch_freq
## 1         3          20 2016-05-04 4.666667
## 2         3          20 2016-04-20 4.666667
## 3         3          20 2016-04-20 4.666667
```

```
dim(tab4)
```

```
## [1] 9940   13
```

```
#write.csv(tab4,file="seen_data.csv",row.names = FALSE)
```

Take a look at these orders

```
filter(tab4,D.OrderHeaderID==3560| D.OrderHeaderID==3561|D.OrderHeaderID==3562|D.OrderHeaderID==3563|D.OrderHea
derID==3566|D.OrderHeaderID==3567|D.OrderHeaderID==3557|
         D.OrderHeaderID==3558|D.OrderHeaderID==4475)[c(1:5),1:6]
```

```
##   D.PersonID D.OrderHeaderID H.ID DeviceID      H.CreationDatetime
## 1      43370            4475 4475        5 2017-08-29 11:36:17.050
## 2      43371            4475 4475        5 2017-08-29 11:36:17.050
## 3      43372            4475 4475        5 2017-08-29 11:36:17.050
## 4      43373            4475 4475        5 2017-08-29 11:36:17.050
## 5         23            3567 3567        4 2017-07-04 10:37:08.580
##   H.H.LastEditionDatetime
## 1 2017-08-29 12:22:27.640
## 2 2017-08-29 12:22:27.640
## 3 2017-08-29 12:22:27.640
## 4 2017-08-29 12:22:27.640
## 5 2017-07-04 10:52:07.160
```

Check again our data sets

```
print(table(re_cust$nb_visits))
```

```
##
##   2   3   4   5   9  10  13
## 285  32   3   3   1   1   1
```

```
print(table(tab4$user_visit))
```

```
##
##    0    1
## 9215  725
```

```
print(table(newcust$nb_visits))
```

```
##
##    1
## 9215
```

```
print(table(tab1$nb_visits))
```

```
##
##    1    2    3    4    5    9   10   13
## 9215  285   32    3    3    1    1    1
```

```
length(unique(tab4$DeviceID))
```

```
## [1] 334
```

2) EXTRACTING RELEVANT EVENT INFORMATIONS

The goal of this task is to get data describing actions realized by the User for each Order.

In order to make a link between Event, Order and User, we have utilized 3 tables.

The SQL code to get this data set is as following:

select distinct(TimeStamp), E.ID,E.UserID,D.PersonID as PersonID, E.DeviceID, D.OrderHeaderID as OrderHeaderID, E.TimeStamp, H.DeviceID, H.ID, H.CreationDatetime,H.LastEditionDatetime,E.Parameter from Event E left join dbo.OrderDetail D on E.UserID=D.PersonID left join dbo.OrderHeader H on H.DeviceID=E.DeviceID and D.OrderHeaderID=H.ID where H.DeviceID is not null and H.ID is not null and cast(E.Timestamp as date)=cast(H.CreationDatetime as date) # our assumption was when theses 2 dates are the same, we can link the Event to the OrderHeaderID order by E.TimeStamp , E.UserID

```
event_df<- read.csv("C:/Users/Pham Antoine/Desktop/extractSQL_event_ID1.csv",sep=';',header = TRUE)
```

```
tail(event_df,3)
```

```
##                   ï..TimeStamp    ID UserID PersonID DeviceID
## 319291 2016-10-03 17:17:01.727 40591  10403    10403       11
## 319292 2016-10-03 17:16:56.147 40590  10404    10404       11
## 319293 2016-10-03 17:16:36.137 40589  10403    10403       11
##        OrderHeaderID               TimeStamp DeviceID.1 ID.1
## 319291          1204 2016-10-03 17:17:01.727         11 1204
## 319292          1204 2016-10-03 17:16:56.147         11 1204
## 319293          1204 2016-10-03 17:16:36.137         11 1204
##               CreationDatetime     LastEditionDatetime    Parameter Type
## 319291 2016-10-03 17:23:11.443 2016-10-03 17:23:11.443         MENU    0
## 319292 2016-10-03 17:23:11.443 2016-10-03 17:23:11.443 HAMBOURGEOIS    0
## 319293 2016-10-03 17:23:11.443 2016-10-03 17:23:11.443    MENU PLAT    1
```

```
dim(event_df)# before importing the data set having Type (318475  7)
```

```
## [1] 319293     13
```

Check out the output above a given customer

```
filter(event_df,PersonID==23)[c(1:5),1:6]
```

```
##              ï..TimeStamp     ID UserID PersonID DeviceID OrderHeaderID
## 1 2017-07-04 13:25:11.400 357889     23       23        9          3566
## 2 2017-07-04 13:25:11.393 357888     23       23        9          3566
## 3 2017-07-04 13:13:44.727 357887     23       23        9          3566
## 4 2017-07-04 13:13:11.473 357900     23       23        4          3567
## 5 2017-07-04 13:11:46.553 357886     23       23        9          3566
```

```
filter(tab4,D.PersonID==23)
```

```
##   D.PersonID D.OrderHeaderID H.ID DeviceID      H.CreationDatetime
## 1         23            3567 3567        4 2017-07-04 10:37:08.580
## 2         23            3566 3566        9 2017-07-04 10:22:30.953
## 3         23            3563 3563    15954 2017-07-03 18:58:33.417
## 4         23            3562 3562        2 2017-07-03 18:42:29.333
## 5         23            3561 3561       11 2017-07-03 18:20:06.123
## 6         23            3560 3560        5 2017-07-03 18:07:24.593
## 7         23            3559 3559        7 2017-07-03 17:46:28.447
## 8         23            3558 3558        9 2017-07-03 17:22:01.017
## 9         23            3557 3557        4 2017-07-03 17:14:23.323
##   H.H.LastEditionDatetime D.WorkspaceLocation user_visit H.NbDiners
## 1 2017-07-04 10:52:07.160                NULL          1          2
## 2 2017-07-04 11:13:01.443                NULL          1          2
## 3 2017-07-03 18:59:08.137                NULL          1          1
## 4 2017-07-03 18:42:29.333                NULL          1          1
## 5 2017-07-03 18:20:06.123                NULL          1          1
## 6 2017-07-03 18:59:07.650                NULL          1          1
## 7 2017-07-03 18:52:22.977                NULL          1          1
## 8 2017-07-03 18:12:31.680                NULL          1          1
## 9 2017-07-03 18:37:04.777                NULL          1          1
##   nb_visits avg_ticketU       Date  pch_freq
## 1         9    42.07778 2017-07-04 0.1111111
## 2         9    42.07778 2017-07-04 0.1111111
## 3         9    42.07778 2017-07-03 0.1111111
## 4         9    42.07778 2017-07-03 0.1111111
## 5         9    42.07778 2017-07-03 0.1111111
## 6         9    42.07778 2017-07-03 0.1111111
## 7         9    42.07778 2017-07-03 0.1111111
## 8         9    42.07778 2017-07-03 0.1111111
## 9         9    42.07778 2017-07-03 0.1111111
```

Check out the output

```
str(event_df)
```

```
## 'data.frame':    319293 obs. of  13 variables:
##  $ ï..TimeStamp       : Factor w/ 283837 levels "2016-10-03 17:16:36.137",..: 283837 283836 283835 283835 28
3834 283833 283832 283831 283830 283829 ...
##  $ ID                 : int  455301 455300 455298 455299 455297 455296 455295 455294 455293 455292 ...
##  $ UserID             : int  43439 43436 43437 43437 43439 43436 43436 43436 43436 43436 ...
##  $ PersonID           : int  43439 43436 43437 43437 43439 43436 43436 43436 43436 43436 ...
##  $ DeviceID           : int  3 3 3 3 3 3 3 3 3 3 ...
##  $ OrderHeaderID      : int  4502 4502 4502 4502 4502 4502 4502 4502 4502 4502 ...
##  $ TimeStamp          : Factor w/ 283837 levels "2016-10-03 17:16:36.137",..: 283837 283836 283835 283835 28
3834 283833 283832 283831 283830 283829 ...
##  $ DeviceID.1         : int  3 3 3 3 3 3 3 3 3 3 ...
##  $ ID.1               : int  4502 4502 4502 4502 4502 4502 4502 4502 4502 4502 ...
##  $ CreationDatetime   : Factor w/ 2753 levels "2016-10-03 17:23:11.443",..: 2753 2753 2753 2753 2753 2753 27
53 2753 2753 2753 ...
##  $ LastEditionDatetime: Factor w/ 2753 levels "2016-10-03 17:23:11.443",..: 2753 2753 2753 2753 2753 2753 27
53 2753 2753 2753 ...
##  $ Parameter          : Factor w/ 168 levels "20","43","44",..: 11 11 11 11 143 94 139 94 8 52 ...
##  $ Type               : int  9 9 9 9 8 0 0 0 0 0 ...
```

Converting numeric variables to factor variables

```
to_facvars<-c('UserID','PersonID','DeviceID','OrderHeaderID','DeviceID.1','ID.1')
event_df<-to.factors(event_df,to_facvars)
```

Checking Na values

```
sapply(event_df,function(x) sum ( is.na(x)))
```

```
##        ï..TimeStamp                  ID              UserID
##                   0                   0                   0
##            PersonID            DeviceID       OrderHeaderID
##                   0                   0                   0
##           TimeStamp          DeviceID.1                ID.1
##                   0                   0                   0
##    CreationDatetime LastEditionDatetime           Parameter
##                   0                   0                   0
##                Type
##                   0
```

```
colnames(event_df)
```

```
##  [1] "ï..TimeStamp"        "ID"                   "UserID"              
##  [4] "PersonID"            "DeviceID"             "OrderHeaderID"       
##  [7] "TimeStamp"           "DeviceID.1"           "ID.1"                
## [10] "CreationDatetime"    "LastEditionDatetime" "Parameter"           
## [13] "Type"
```

Dropping duplicated columns

```
event_df<- subset(event_df,select=names(event_df)%in%c('PersonID','DeviceID','OrderHeaderID','TimeStamp','Creat
ionDatetime','LastEditionDatetime','Parameter','Type'))
colnames(event_df)
```

```
## [1] "PersonID"            "DeviceID"            "OrderHeaderID"      
## [4] "TimeStamp"           "CreationDatetime"    "LastEditionDatetime"
## [7] "Parameter"           "Type"
```

Renaming columns for having the same names as ones in the 1st data set

```
names(event_df)<-c('D.PersonID', 'H.DeviceID','D.OrderHeaderID','E.TimeStamp','H.CreationDatetime','H.LastEditi
onDatetime','E.Parameter','Type')# Type is a new variable so we don't need #changing  variable names
```

```
tail(event_df,2)
```

```
##        D.PersonID H.DeviceID D.OrderHeaderID             E.TimeStamp
## 319292      10404         11            1204 2016-10-03 17:16:56.147
## 319293      10403         11            1204 2016-10-03 17:16:36.137
##             H.CreationDatetime   H.LastEditionDatetime  E.Parameter Type
## 319292 2016-10-03 17:23:11.443 2016-10-03 17:23:11.443 HAMBOURGEOIS    0
## 319293 2016-10-03 17:23:11.443 2016-10-03 17:23:11.443    MENU PLAT    1
```

Now, computing for each pair : Order/User, the duration between Max&Min of variable ‘TimeStamp’

```
tab6=event_df%>%group_by(D.OrderHeaderID,D.PersonID,H.DeviceID)%>%summarise(H.CreationDatetime=unique(H.Creatio
nDatetime),
                                                                        H.LastEditionDatetime= unique(H.LastEdit
ionDatetime),
                                                                       visit_duration=difftime(max(as.POSIXct(E.
TimeStamp)),min(as.POSIXct(E.TimeStamp)),units='min')
                                                                       #maxETimestamp=max(as.POSIXlt(E.TimeStamp
)),
                                                                       #minETimestamp=min(as.POSIXlt(E.TimeStamp
))
                                                                       )
```

```
tail(tab6,2)
```

```
## # A tibble: 2 x 6
## # Groups:   D.OrderHeaderID, D.PersonID [2]
##   D.OrderHeaderID D.PersonID H.DeviceID H.CreationDateti~ H.LastEditionDa~
##   <fct>           <fct>      <fct>      <fct>             <fct>           
## 1 4502            43438      3          2017-08-30 19:34~ 2017-08-30 20:0~
## 2 4502            43439      3          2017-08-30 19:34~ 2017-08-30 20:0~
## # ... with 1 more variable: visit_duration <time>
```

```
dim(tab6)
```

```
## [1] 6733    6
```

!!!!6733 rows : it’s good since we haved obtained the same number of rows when applying function pivot_table in Python to group by Order/Person ID all actions (Parameter): e.g, order1 person2 youtube=3,xmax=2…..

Checking with the PersonID 10407

```
filter(tab6,D.PersonID ==10407)
```

```
## # A tibble: 1 x 6
## # Groups:   D.OrderHeaderID, D.PersonID [1]
##   D.OrderHeaderID D.PersonID H.DeviceID H.CreationDateti~ H.LastEditionDa~
##   <fct>           <fct>      <fct>      <fct>             <fct>           
## 1 1205            10407      4          2016-10-03 18:13~ 2016-10-03 18:1~
## # ... with 1 more variable: visit_duration <time>
```

ATTENTION : make sure visit_duration is mesured by minutes instead of hour!!!!!!!!!!!

```
id10407=filter(event_df,D.PersonID==10407)
```

As we can see below, the Time difference of ID10407 is 1.06 hours and not minutes, that’s why we have used :difftime(max(as.POSIXct(E.TimeStamp)),min(as.POSIXct(E.TimeStamp)),units=‘min’)

```
max(as.POSIXlt(id10407$E.TimeStamp))-min(as.POSIXlt(id10407$E.TimeStamp))
```

```
## Time difference of 1.062549 hours
```

```
max(as.POSIXct(id10407$E.TimeStamp))-min(as.POSIXct(id10407$E.TimeStamp))
```

```
## Time difference of 1.062549 hours
```

```
difftime(max(as.POSIXct(id10407$E.TimeStamp)),min(as.POSIXct(id10407$E.TimeStamp)),units='min')
```

```
## Time difference of 63.75295 mins
```

Getting the event duration for this User

```
id10406=filter(event_df,D.PersonID==10406)
id10406[c(1:5),1:8]
```

```
##   D.PersonID H.DeviceID D.OrderHeaderID              E.TimeStamp
## 1      10406          4            1205 2016-10-03 19:11:34.813
## 2      10406          4            1205 2016-10-03 19:11:34.810
## 3      10406          4            1205 2016-10-03 18:52:00.377
## 4      10406          4            1205 2016-10-03 18:52:00.377
## 5      10406          4            1205 2016-10-03 18:51:58.063
##        H.CreationDatetime   H.LastEditionDatetime E.Parameter Type
## 1 2016-10-03 18:13:57.800 2016-10-03 18:13:57.800      Tetris    9
## 2 2016-10-03 18:13:57.800 2016-10-03 18:13:57.800         Ski    9
## 3 2016-10-03 18:13:57.800 2016-10-03 18:13:57.800         Ski    9
## 4 2016-10-03 18:13:57.800 2016-10-03 18:13:57.800      Tetris    9
## 5 2016-10-03 18:13:57.800 2016-10-03 18:13:57.800     Catalog    9
```

```
max(as.POSIXlt(id10406$E.TimeStamp))-min(as.POSIXlt(id10406$E.TimeStamp))
```

```
## Time difference of 1.058362 hours
```

checking it with the new data frame

```
max(as.POSIXlt(filter(event_df,D.PersonID==23&D.OrderHeaderID==3559)$E.TimeStamp))-min(as.POSIXlt(filter(event_
df,D.PersonID==23&D.OrderHeaderID==3559)$E.TimeStamp))
```

```
## Time difference of 2.413027 hours
```

```
subset(tab6,D.PersonID==23&D.OrderHeaderID==3559,select = c('visit_duration'))
```

```
## # A tibble: 1 x 1
##   visit_duration
##   <time>
## 1 144.781600002448
```

Thatlooks good!!!!

Take a look at a particular case

```
filter(event_df,D.PersonID==539&D.OrderHeaderID==2037)
```

```
##   D.PersonID H.DeviceID D.OrderHeaderID              E.TimeStamp
## 1        539         18            2037 2017-01-31 14:35:53.017
## 2        539         18            2037 2017-01-31 14:35:53.007
## 3        539         18            2037 2017-01-31 14:35:51.460
## 4        539         18            2037 2017-01-31 14:35:51.460
## 5        539         18            2037 2017-01-31 14:35:07.963
##        H.CreationDatetime   H.LastEditionDatetime E.Parameter Type
## 1 2017-01-31 13:35:59.137 2017-01-31 13:35:59.137           6   10
## 2 2017-01-31 13:35:59.137 2017-01-31 13:35:59.137     BAILEYS    1
## 3 2017-01-31 13:35:59.137 2017-01-31 13:35:59.137    BOISSONS    0
## 4 2017-01-31 13:35:59.137 2017-01-31 13:35:59.137   APERITIFS    0
## 5 2017-01-31 13:35:59.137 2017-01-31 13:35:59.137       Login    3
```

```
filter(tab6,D.PersonID==539)
```

```
## # A tibble: 2 x 6
## # Groups:   D.OrderHeaderID, D.PersonID [2]
##   D.OrderHeaderID D.PersonID H.DeviceID H.CreationDateti~ H.LastEditionDa~
##   <fct>           <fct>      <fct>      <fct>             <fct>
## 1 1720            539        16         2016-12-21 20:30~ 2016-12-21 21:0~
## 2 2037            539        18         2017-01-31 13:35~ 2017-01-31 13:3~
## # ... with 1 more variable: visit_duration <time>
```

There are more D.OrderHeaderID in the set (tab4) than in the set (tab6) since no information recorded in the table Event for somme OrderHeaderID (for example, if Device N°> 100 = no Event recorded or Date < 21/12 => no Event recorded)

We can also add the ‘visit_duration’ from tab6 to tab4; a NA’s value will be given to rows in tab4 having no value in tab6

```
User=merge(x=tab4, y=tab6, by= c('D.PersonID', 'D.OrderHeaderID'), all.x = T)
dim(User)
```

```
## [1] 9940   17
```

Check out ‘User’ data set

```
library(funModeling)
```

```
## Warning: package 'funModeling' was built under R version 3.4.4
```

```
## Loading required package: Hmisc
```

```
## Warning: package 'Hmisc' was built under R version 3.4.4
```

```
## Loading required package: lattice
```

```
## Loading required package: survival
```

```
## Loading required package: Formula
```

```
## Loading required package: ggplot2
```

```
##
## Attaching package: 'Hmisc'
```

```
## The following objects are masked from 'package:dplyr':
##
##     src, summarize
```

```
## The following objects are masked from 'package:base':
##
##     format.pval, units
```

```
## funModeling v.1.6.7 :)
## Examples and tutorials at livebook.datascienceheroes.com
```

```
df_status(User)
```

```
##                   variable q_zeros p_zeros q_na  p_na q_inf p_inf     type
## 1               D.PersonID       0    0.00    0  0.00     0     0   factor
## 2          D.OrderHeaderID       0    0.00    0  0.00     0     0  integer
## 3                     H.ID       0    0.00    0  0.00     0     0  integer
## 4                 DeviceID       0    0.00    0  0.00     0     0   factor
## 5     H.CreationDatetime.x       0    0.00    0  0.00     0     0   factor
## 6   H.H.LastEditionDatetime       0    0.00    0  0.00     0     0   factor
## 7       D.WorkspaceLocation     187    1.88    0  0.00     0     0   factor
## 8               user_visit    9215   92.71    0  0.00     0     0  numeric
## 9               H.NbDiners      10    0.10    0  0.00     0     0  integer
## 10               nb_visits       0    0.00    0  0.00     0     0  integer
## 11             avg_ticketU     182    1.83    0  0.00     0     0  numeric
## 12                    Date       0    0.00    0  0.00     0     0     Date
## 13                pch_freq    9894   99.54    0  0.00     0     0  numeric
## 14              H.DeviceID       0    0.00 3207 32.26     0     0   factor
## 15    H.CreationDatetime.y       0    0.00 3207 32.26     0     0   factor
## 16   H.LastEditionDatetime       0    0.00 3207 32.26     0     0   factor
## 17          visit_duration       2    0.02 3207 32.26     0     0 difftime
##    unique
## 1    9541
## 2    4334
## 3    4334
## 4     334
## 5    4334
## 6    4334
## 7       5
## 8       2
## 9      12
## 10      8
## 11    650
## 12    504
## 13      8
## 14     12
## 15   2753
## 16   2753
## 17   6673
```

```
subset(User,D.PersonID==539,select=c("D.OrderHeaderID","D.PersonID" ,"user_visit","nb_visits", "H.NbDiners","vi
sit_duration" ))
```

```
##      D.OrderHeaderID D.PersonID user_visit nb_visits H.NbDiners
## 7752              37        539          1        13          1
## 7753              40        539          1        13          1
## 7754             912        539          1        13          1
## 7755             913        539          1        13          1
## 7756             914        539          1        13          1
## 7757             982        539          1        13          1
## 7758            1080        539          1        13          1
## 7759            1081        539          1        13          1
## 7760            1082        539          1        13          1
## 7761            1207        539          1        13          1
## 7762            1720        539          1        13          1
## 7763            2036        539          1        13          1
## 7764            2037        539          1        13          1
##      visit_duration
## 7752        NA mins
## 7753        NA mins
## 7754        NA mins
## 7755        NA mins
## 7756        NA mins
## 7757        NA mins
## 7758        NA mins
## 7759        NA mins
## 7760        NA mins
## 7761        NA mins
## 7762  14.71333 mins
## 7763        NA mins
## 7764   0.75090 mins
```

```
subset(event_df,D.PersonID==23&D.OrderHeaderID==3559,select=c('D.PersonID','D.OrderHeaderID' ,'E.TimeStamp','E.Parameter','Type'))[c(1:5),1:5]
```

```
##       D.PersonID D.OrderHeaderID             E.TimeStamp
## 79529         23            3559 2017-07-03 20:54:58.943
## 79548         23            3559 2017-07-03 20:43:31.223
## 79549         23            3559 2017-07-03 20:43:31.027
## 79559         23            3559 2017-07-03 20:41:46.203
## 79560         23            3559 2017-07-03 20:41:44.857
##               E.Parameter Type
## 79529 animatedUserControl    9
## 79548             YouTube    8
## 79549 animatedUserControl    9
## 79559            Connect4    8
## 79560 animatedUserControl    9
```

Look at returning customers

```
filter(User, nb_visits!=1)[c(1:10),1:7]
```

```
##    D.PersonID D.OrderHeaderID H.ID DeviceID    H.CreationDatetime.x
## 1       10077            1106 1106        5 2016-09-22 17:00:08.797
## 2       10077            1110 1110        5 2016-09-22 18:29:33.890
## 3       10090            1107 1107        9 2016-09-22 18:27:40.810
## 4       10090            1112 1112        9 2016-09-22 19:02:50.263
## 5       10091            1107 1107        9 2016-09-22 18:27:40.810
## 6       10091            1112 1112        9 2016-09-22 19:02:50.263
## 7       10099            1114 1114        3 2016-09-22 20:09:00.447
## 8       10099            1115 1115        3 2016-09-22 20:37:15.167
## 9       10109            1118 1118       11 2016-09-23 10:47:24.137
## 10      10109            1119 1119       11 2016-09-23 11:33:02.603
##    H.H.LastEditionDatetime D.WorkspaceLocation
## 1  2016-09-22 17:00:08.797                NULL
## 2  2016-09-22 18:29:33.890                NULL
## 3  2016-09-22 18:27:40.810                NULL
## 4  2016-09-22 19:02:50.263                NULL
## 5  2016-09-22 18:27:40.810                NULL
## 6  2016-09-22 19:02:50.263                NULL
## 7  2016-09-22 20:09:00.447                NULL
## 8  2016-09-22 20:37:15.167                NULL
## 9  2016-09-23 10:47:24.137                NULL
## 10 2016-09-23 11:33:02.603                NULL
```

```
filter(User,D.PersonID==38331)
```

```
##   D.PersonID D.OrderHeaderID H.ID DeviceID    H.CreationDatetime.x
## 1      38331            3609 3609        2 2017-07-07 11:34:11.137
## 2      38331            3611 3611    16951 2017-07-07 16:50:31.597
##   H.H.LastEditionDatetime D.WorkspaceLocation user_visit H.NbDiners
## 1 2017-07-07 12:40:24.770                NULL          1          2
## 2 2017-07-07 16:53:16.703                NULL          1          2
##   nb_visits avg_ticketU       Date pch_freq H.DeviceID
## 1         2        8.55 2017-07-07        0          2
## 2         2        8.55 2017-07-07        0       <NA>
##      H.CreationDatetime.y    H.LastEditionDatetime visit_duration
## 1 2017-07-07 11:34:11.137 2017-07-07 12:40:24.770   31.6169 mins
## 2                    <NA>                    <NA>        NA mins
```

```
colnames(User)
```

```
##  [1] "D.PersonID"              "D.OrderHeaderID"
##  [3] "H.ID"                    "DeviceID"
##  [5] "H.CreationDatetime.x"    "H.H.LastEditionDatetime"
##  [7] "D.WorkspaceLocation"     "user_visit"
##  [9] "H.NbDiners"              "nb_visits"
## [11] "avg_ticketU"             "Date"
## [13] "pch_freq"                "H.DeviceID"
## [15] "H.CreationDatetime.y"    "H.LastEditionDatetime"
## [17] "visit_duration"
```

Drop duplicate columns

```
User<-subset(User, select = names(User)%in%c("D.PersonID","D.OrderHeaderID","H.ID","DeviceID","H.CreationDateti
me.x","H.H.LastEditionDatetime","D.WorkspaceLocation",
                                     "user_visit","H.NbDiners","nb_visits","avg_ticketU","Date","pch_freq","vis
it_duration"))
names(User)<-c("D.PersonID","D.OrderHeaderID","H.ID","DeviceID","H.CreationDatetime","H.LastEditionDatetime","D
.WorkspaceLocation",
                                     "user_visit","H.NbDiners","nb_visits","avg_ticketU","Date","pch_freq","vis
it_duration")
colnames(User)
```

```
##  [1] "D.PersonID"            "D.OrderHeaderID"
##  [3] "H.ID"                  "DeviceID"
##  [5] "H.CreationDatetime"    "H.LastEditionDatetime"
##  [7] "D.WorkspaceLocation"   "user_visit"
##  [9] "H.NbDiners"            "nb_visits"
## [11] "avg_ticketU"           "Date"
## [13] "pch_freq"              "visit_duration"
```

```
filter(tab6,D.PersonID==23)
```

```
## # A tibble: 8 x 6
## # Groups:   D.OrderHeaderID, D.PersonID [8]
##   D.OrderHeaderID D.PersonID H.DeviceID H.CreationDateti~ H.LastEditionDa~
##   <fct>           <fct>      <fct>      <fct>             <fct>
## 1 3557            23         4          2017-07-03 17:14~ 2017-07-03 18:3~
## 2 3558            23         9          2017-07-03 17:22~ 2017-07-03 18:1~
## 3 3559            23         7          2017-07-03 17:46~ 2017-07-03 18:5~
## 4 3560            23         5          2017-07-03 18:07~ 2017-07-03 18:5~
## 5 3561            23         11         2017-07-03 18:20~ 2017-07-03 18:2~
## 6 3562            23         2          2017-07-03 18:42~ 2017-07-03 18:4~
## 7 3566            23         9          2017-07-04 10:22~ 2017-07-04 11:1~
## 8 3567            23         4          2017-07-04 10:37~ 2017-07-04 10:5~
## # ... with 1 more variable: visit_duration <time>
```

How many are there mising values on each variable?

```
sapply(User, function(x) sum ( is.na(x)))
```

```
##            D.PersonID       D.OrderHeaderID                  H.ID
##                     0                     0                     0
##              DeviceID    H.CreationDatetime H.LastEditionDatetime
##                     0                     0                     0
##   D.WorkspaceLocation            user_visit            H.NbDiners
##                     0                     0                     0
##             nb_visits           avg_ticketU                  Date
##                     0                     0                     0
##              pch_freq        visit_duration
##                     0                  3207
```

```
filter(tab4,D.OrderHeaderID==1)
```

```
##   D.PersonID D.OrderHeaderID H.ID DeviceID      H.CreationDatetime
## 1        136               1    1       16 2016-02-23 12:08:06.560
## 2        178               1    1       16 2016-02-23 12:08:06.560
## 3        220               1    1       16 2016-02-23 12:08:06.560
## 4        221               1    1       16 2016-02-23 12:08:06.560
## 5        222               1    1       16 2016-02-23 12:08:06.560
##   H.H.LastEditionDatetime D.WorkspaceLocation user_visit H.NbDiners
## 1 2016-02-23 20:05:44.963                NULL          1          4
## 2 2016-02-23 20:05:44.963                NULL          0          4
## 3 2016-02-23 20:05:44.963                NULL          0          4
## 4 2016-02-23 20:05:44.963                NULL          0          4
## 5 2016-02-23 20:05:44.963                NULL          0          4
##   nb_visits avg_ticketU       Date pch_freq
## 1        10       41.25 2016-02-23     19.8
## 2         1       17.50 2016-02-23      0.0
## 3         1        7.50 2016-02-23      0.0
## 4         1       32.00 2016-02-23      0.0
## 5         1       36.00 2016-02-23      0.0
```

3) RESHAPING DATA AND GETTING LABELS FOR ALGORITHMS

For training our predictive models, we need to feed labels (outcomes) corresponding to each record to models

These labels are simply items bought by each customer (PersonID) at each transaction (OredrHeaderID)

```
unique(df1$D.WorkingOrder)
```

```
## [1] 7 3 4 1 5 2 0
```

```
table(as.factor(df1$D.WorkingOrder))
```

```
## 
##     0     1     2     3     4     5     7 
##    99  5426   416 26043  3603   448  3281
```

Subseting columns allowing to prepare an outcomes (labels) matrix

```
tab7<-subset(df1,select = c('H.ID','D.PersonID','P.Name','D.Quantity','P.NetPrice','D.NegociatedNetPrice'))
tail(tab7,5)
```

```
##       H.ID D.PersonID      P.Name D.Quantity P.NetPrice
## 40335 4502      43439   PLANTAMAX          1         10
## 40336 4502      43439 PATATEDOUCE          1          3
## 40338 4503      43440  MAXPARTOUT          1         10
## 40339 4503      43440 PATATEDOUCE          1          3
## 40340 4503      43440  MAXPARTOUT          1         10
##       D.NegociatedNetPrice
## 40335                13.00
## 40336                 0.00
## 40338                 9.88
## 40339                 0.00
## 40340                 9.88
```

Using ‘reshape2’ package to get an outcome matrix

```
dim(tab7)
```

```
## [1] 39316     6
```

The function ‘dcast’ will be utilized to group records by OrderHeaderID & PersonID. It will also create columns each of which representes an Item

```
outcomes<-dcast(tab7,H.ID+D.PersonID~P.Name, value.var = 'D.Quantity',fun.aggregate=sum)
# It's very important to add fun.aggregate.fun =sum to count the quantity of sold items and not only the number
 of item occurence
head(outcomes,3)
```

```
##   H.ID D.PersonID ABATILLES PLATES ABATILLES RED AVECESAR BADOIT 33cl
## 1    1        136                0             0        0           0
## 2    1        178                0             0        0           0
## 3    1        220                0             0        0           0
##   BAILEYS BIERE SANS GLUTEN Boisson Rouge BRIE CAFEGOURMAND
## 1       0                 0             0    0            0
## 2       0                 0             0    0            0
## 3       0                 0             0    1            0
##   CAFEGOURMANDMENU CAPPUCCINO CHAMPAGNE BOUTEILLE CHAMPAGNE COUPE CHOCOLAT
## 1                0          0                   0               0        0
## 2                0          0                   0               0        0
## 3                0          0                   0               0        0
##   COCA COCA ZERO Cocktail Saint Valentin alcoolisÃ©
## 1    0         0                                  0
## 2    0         0                                  0
## 3    0         0                                  0
##   Cocktail Saint Valentin sans alcool COMPOTEE CERISE NOIRE CrÃ¨me au bleu
## 1                                   0                     0              0
## 2                                   0                     0              0
## 3                                   0                     0              0
##   CREME POIVRE VERT CREMEBRULEE CROQUANT DECA DESPERADOS Dessert du jour
## 1                 0           0        0    0          0               0
## 2                 0           0        0    0          0               0
## 3                 0           0        0    0          0               0
##   Domaine La Colombette RosÃ© DOUBLE EXPRESSO DUOSALADE EVIAN EXPRESSO
## 1                           0               0         0     0        2
## 2                           0               0         0     0        1
## 3                           0               0         0     0        0
##   FRAICHEUR FUMAX GATEAUCAROTTE GET GIN GRIMBERGEN GROSSEFRITE
## 1         0     0             0   0   0          0           0
## 2         0     0             1   0   0          0           0
## 3         0     0             0   0   0          0           0
##   HAMBOURGEOIS DU MOMENT HAMBOURGEOIS DU MOMENT MENU HAMPE HOEGGARDEN
## 1                      0                          0     0          0
## 2                      0                          0     0          0
## 3                      0                          0     0          0
##   INFUSION JUS FRUIT MAISON KEKETTE EXTRA KEKETTE RED KIR LATTE MACCHIATTO
## 1        0                0             0           0   0                0
## 2        0                0             0           0   0                0
## 3        0                0             0           0   0                0
##   LILLET MALIBU MARTINI MAX CUVEE  MAXCHAMPETRE MAXCHAMPETRE MENU
## 1      0      0       0          0            0                 0
## 2      0      0       0          0            0                 0
## 3      0      0       0          0            0                 0
##   MAXCOCOTTE MAXCOCOTTE MENU MAXHALEINE MAXHALEINE MENU MAXIFLETTE
## 1          0               0          0               0          0
## 2          0               0          0               0          0
## 3          0               0          0               0          0
##   MAXIFLETTE MENU MAXINUS MAXINUS MENU MAXNAUDOU MAXNAUDOU MENU MAXPARTOUT
## 1               0       2            0         0              0          0
## 2               0       0            0         0              0          0
## 3               0       0            0         0              0          0
##   MAXPARTOUT MENU MAXPOUSSIN MAXYONNAISE MENU HAMBOURGEOIS MENU PLAT
## 1               0          0           0                 2         0
## 2               0          0           0                 0         1
## 3               0          0           0                 0         0
##   MINIMAX MOKACCINO MOUFLET MOUSSECHOCO NUGGETS OASIS ORANGINA
## 1       0         0       0           0       0     0        0
## 2       0         0       0           0       0     0        0
## 3       0         0       0           0       0     0        0
##   PAPOLLE BLANC MOEL PAPOLLE BLANC SEC PAPOLLE ROSE PAPOLLE ROUGE PASTIS
## 1                  0                 0            0             0      0
## 2                  0                 0            0             0      0
## 3                  0                 0            0             0      0
##   PATATEDOUCE PELFORTH PESTO PICHET PUNCH PICHET SANGRIA PINEAU
## 1           2        0     0            0              0      0
## 2           0        0     0            0              0      0
## 3           0        0     0            0              0      0
##   PINT PELFORTH PLANTAMAX PLANTAMAX MENU PLAT DU MOMENT PORTO PUNCH Maison
## 1             0         0              0              0     0            0
## 2             0         0              0              0     0            0
## 3             0         0              0              0     0            0
##   RED BULL RHUM RHUM ARRANGE RICARD SALADEASIAT SANGRIA SAUCE BARBECUE
## 1        0    0            0      0           0       0              0
## 2        0    0            0      0           0       0              0
## 3        0    0            0      0           0       0              0
##   Sauce ForestiÃ¨re SAUCE POIVRE SAUMON SCHWEPPES SIROP SOUPEFRUIT TAPAS
## 1                 0            0      0         0     0          2     0
## 2                 0            0      0         0     0          0     0
## 3                 0            0      0         0     0          0     0
##   TARTARE TEQUILA THE THE GLACE TOURISTE Verre de Boisson Rouge
## 1       0       0   0         0        0                      0
## 2       1       0   0         0        0                      0
## 3       0       0   0         0        0                      0
##   Verre de Bordeaux Rouge Agape VERRE MAX CUVEE VERRE PAPOLLE BLANC MOEL
## 1                             0               0                        0
## 2                             0               0                        0
## 3                             0               0                        0
##   VERRE PAPOLLE BLANC SEC VERRE PAPOLLE ROSE VERRE PAPOLLE ROUGE
## 1                       0                  0                   0
## 2                       0                  0                   0
## 3                       0                  0                   0
##   Verre RosÃ© la Colombette VODKA WHISKY XAMAX XAMAX MENU
## 1                         0     0      0     0          0
## 2                         0     0      0     0          0
## 3                         0     0      0     0          0
```

NOTE: D.WorkingOrder variable has 7 possible values

```
unique(df1$D.WorkingOrder)
```

```
## [1] 7 3 4 1 5 2 0
```

IMPORTANT: The function dcast applied here doesn’t retain these values , it calculates the occurence frequency of each item for each pair H.ID/PersonID Taking an example on the table ‘outcomes’:

```
subset(outcomes,D.PersonID ==271,select=c('H.ID','D.PersonID','MENU HAMBOURGEOIS'))
```

```
##    H.ID D.PersonID MENU HAMBOURGEOIS
## 20    8        271                 8
```

In df1:

```
df1%>%filter(D.PersonID==271 & P.Name=='MENU HAMBOURGEOIS')%>%select('H.ID','D.PersonID','P.Name','P.WorkingOrder','D.WorkingOrder','P.NetPrice','D.NegociatedNetPrice','D.Quantity','P.WorkingOrder')
```

```
##   H.ID D.PersonID            P.Name P.WorkingOrder D.WorkingOrder
## 1    8        271 MENU HAMBOURGEOIS              0              7
## 2    8        271 MENU HAMBOURGEOIS              0              7
## 3    8        271 MENU HAMBOURGEOIS              0              7
## 4    8        271 MENU HAMBOURGEOIS              0              7
## 5    8        271 MENU HAMBOURGEOIS              0              7
##   P.NetPrice D.NegociatedNetPrice D.Quantity
## 1         16                   48          3
## 2         16                   32          2
## 3         16                   16          1
## 4         16                   16          1
## 5         16                   16          1
```

We’ve observed that 8 Menu Hambourgeois’ in the table ‘outcomes’ corresponds to the total of times ‘MENU HAMBOURGEOIS’ has occured on the table df1.

```
head(User[ order(tab4$H.ID), ],7)
```

```
##      D.PersonID D.OrderHeaderID H.ID DeviceID      H.CreationDatetime
## 9936       9991            1078 1078       11 2016-09-18 18:58:36.733
## 9937       9993            1079 1079        4 2016-09-19 11:20:02.597
## 9938       9996            1083 1083        4 2016-09-19 19:00:14.880
## 9939       9997            1084 1084        7 2016-09-20 10:03:23.943
## 9940       9999            1085 1085        3 2016-09-20 10:37:02.813
## 9935       9990            1077 1077        8 2016-09-18 18:41:04.460
## 9933       9988            1076 1076        3 2016-09-18 17:11:17.713
##        H.LastEditionDatetime D.WorkspaceLocation user_visit H.NbDiners
## 9936 2016-09-18 18:58:36.733                NULL          0          1
## 9937 2016-09-19 11:20:02.597                NULL          0          1
## 9938 2016-09-19 19:00:14.880                NULL          0          1
## 9939 2016-09-20 10:03:23.943                NULL          0          1
## 9940 2016-09-20 10:37:02.813                NULL          0          2
## 9935 2016-09-18 18:41:04.460                NULL          0          2
## 9933 2016-09-18 17:11:17.713                NULL          0          3
##      nb_visits avg_ticketU       Date pch_freq visit_duration
## 9936         1         8.5 2016-09-18        0        NA mins
## 9937         1        15.5 2016-09-19        0        NA mins
## 9938         1        13.5 2016-09-19        0        NA mins
## 9939         1        24.2 2016-09-20        0        NA mins
## 9940         1        18.0 2016-09-20        0        NA mins
## 9935         1        17.5 2016-09-18        0        NA mins
## 9933         1        20.0 2016-09-18        0        NA mins
```

```
head(outcomes,5)[c(1:5),1:5]
```

```
##   H.ID D.PersonID ABATILLES PLATES ABATILLES RED AVECESAR
## 1    1        136                0             0        0
## 2    1        178                0             0        0
## 3    1        220                0             0        0
## 4    1        221                0             0        0
## 5    1        222                0             0        0
```

Verifying if 2 tables ‘Users’ & ‘outcomes’ have the same number of rows It looks correct!

```
print(dim(User))
```

```
## [1] 9940   14
```

```
print(dim(outcomes))
```

```
## [1] 9940  127
```

It looks good!

Converting numerics labels so that they become factor variables ***Not do this now, we need this variable retained numeric to do somme statistics outcomes<- to.factors(outcomes,c(colnames(outcomes[,c(1,3:127)]))) str(outcomes)

Taking again the example with ‘MENU HAMBOURGEOIS’

```
table( outcomes$`MENU HAMBOURGEOIS`)
```

```
## 
##    0    1    2    3    4    5    8 
## 8388 1487   57    4    1    2    1
```

```
filter(outcomes,`MENU HAMBOURGEOIS`==5)[c(1:2),1:5]
```

```
##   H.ID D.PersonID ABATILLES PLATES ABATILLES RED AVECESAR
## 1    8        274                0             0        0
## 2 4175      42550                0             0        0
```

```
table(outcomes$`Verre de Bordeaux Rouge Agape`)
```

```
## 
##    0    1    2 
## 9928   11    1
```

```
#8/9880
```

Sales volume for each Item

```
solditems=sapply(outcomes[,3:127],function(x)sum(x))
```

```
head(solditems,5)
```

```
## ABATILLES PLATES    ABATILLES RED         AVECESAR      BADOIT 33cl 
##               74              114              196               69 
##          BAILEYS 
##               35
```

4) USING FREQUENT ITEMSET BASED ASSOCIATION RULES ALGORITHM (ARules package)

```
#library(arules)
fact <- data.frame(lapply(outcomes[,3:127],as.factor))# to ignore H.ID & PersonID
```

```
head(fact,3)[c(1:2),1:5]
```

```
##   ABATILLES.PLATES ABATILLES.RED AVECESAR BADOIT.33cl BAILEYS
## 1                0             0        0           0       0
## 2                0             0        0           0       0
```

```
dim(fact)
```

```
## [1] 9940  125
```

```
colnames(fact)
```

```
##   [1] "ABATILLES.PLATES"
##   [2] "ABATILLES.RED"
##   [3] "AVECESAR"
##   [4] "BADOIT.33cl"
##   [5] "BAILEYS"
##   [6] "BIERE.SANS.GLUTEN"
##   [7] "Boisson.Rouge"
##   [8] "BRIE"
##   [9] "CAFEGOURMAND"
##  [10] "CAFEGOURMANDMENU"
##  [11] "CAPPUCCINO"
##  [12] "CHAMPAGNE.BOUTEILLE"
##  [13] "CHAMPAGNE.COUPE"
##  [14] "CHOCOLAT"
##  [15] "COCA"
##  [16] "COCA.ZERO"
##  [17] "Cocktail.Saint.Valentin.alcoolisÃ."
##  [18] "Cocktail.Saint.Valentin.sans.alcool"
##  [19] "COMPOTEE.CERISE.NOIRE"
##  [20] "CrÃ.me.au.bleu"
##  [21] "CREME.POIVRE.VERT"
##  [22] "CREMEBRULEE"
##  [23] "CROQUANT"
##  [24] "DECA"
##  [25] "DESPERADOS"
##  [26] "Dessert.du.jour"
##  [27] "Domaine.La.Colombette.RosÃ."
##  [28] "DOUBLE.EXPRESSO"
##  [29] "DUOSALADE"
##  [30] "EVIAN"
##  [31] "EXPRESSO"
##  [32] "FRAICHEUR"
##  [33] "FUMAX"
##  [34] "GATEAUCAROTTE"
##  [35] "GET"
##  [36] "GIN"
##  [37] "GRIMBERGEN"
##  [38] "GROSSEFRITE"
##  [39] "HAMBOURGEOIS.DU.MOMENT"
##  [40] "HAMBOURGEOIS.DU.MOMENT.MENU"
##  [41] "HAMPE"
##  [42] "HOEGGARDEN"
##  [43] "INFUSION"
##  [44] "JUS.FRUIT.MAISON"
##  [45] "KEKETTE.EXTRA"
##  [46] "KEKETTE.RED"
##  [47] "KIR"
##  [48] "LATTE.MACCHIATTO"
##  [49] "LILLET"
##  [50] "MALIBU"
##  [51] "MARTINI"
##  [52] "MAX.CUVEE."
##  [53] "MAXCHAMPETRE"
##  [54] "MAXCHAMPETRE.MENU"
##  [55] "MAXCOCOTTE"
##  [56] "MAXCOCOTTE.MENU"
##  [57] "MAXHALEINE"
##  [58] "MAXHALEINE.MENU"
##  [59] "MAXIFLETTE"
##  [60] "MAXIFLETTE.MENU"
##  [61] "MAXINUS"
##  [62] "MAXINUS.MENU"
##  [63] "MAXNAUDOU"
##  [64] "MAXNAUDOU.MENU"
##  [65] "MAXPARTOUT"
##  [66] "MAXPARTOUT.MENU"
##  [67] "MAXPOUSSIN"
##  [68] "MAXYONNAISE"
##  [69] "MENU.HAMBOURGEOIS"
##  [70] "MENU.PLAT"
##  [71] "MINIMAX"
##  [72] "MOKACCINO"
##  [73] "MOUFLET"
##  [74] "MOUSSECHOCO"
##  [75] "NUGGETS"
##  [76] "OASIS"
##  [77] "ORANGINA"
##  [78] "PAPOLLE.BLANC.MOEL"
##  [79] "PAPOLLE.BLANC.SEC"
##  [80] "PAPOLLE.ROSE"
##  [81] "PAPOLLE.ROUGE"
##  [82] "PASTIS"
##  [83] "PATATEDOUCE"
##  [84] "PELFORTH"
##  [85] "PESTO"
##  [86] "PICHET.PUNCH"
##  [87] "PICHET.SANGRIA"
##  [88] "PINEAU"
##  [89] "PINT.PELFORTH"
##  [90] "PLANTAMAX"
##  [91] "PLANTAMAX.MENU"
##  [92] "PLAT.DU.MOMENT"
##  [93] "PORTO"
##  [94] "PUNCH.Maison"
##  [95] "RED.BULL"
##  [96] "RHUM"
##  [97] "RHUM.ARRANGE"
##  [98] "RICARD"
##  [99] "SALADEASIAT"
## [100] "SANGRIA"
## [101] "SAUCE.BARBECUE"
## [102] "Sauce.ForestiÃ.re"
## [103] "SAUCE.POIVRE"
## [104] "SAUMON"
## [105] "SCHWEPPES"
## [106] "SIROP"
## [107] "SOUPEFRUIT"
## [108] "TAPAS"
## [109] "TARTARE"
## [110] "TEQUILA"
## [111] "THE"
## [112] "THE.GLACE"
## [113] "TOURISTE"
## [114] "Verre.de.Boisson.Rouge"
## [115] "Verre.de.Bordeaux.Rouge.Agape"
## [116] "VERRE.MAX.CUVEE"
## [117] "VERRE.PAPOLLE.BLANC.MOEL"
## [118] "VERRE.PAPOLLE.BLANC.SEC"
## [119] "VERRE.PAPOLLE.ROSE"
## [120] "VERRE.PAPOLLE.ROUGE"
## [121] "Verre.RosÃ..la.Colombette"
## [122] "VODKA"
## [123] "WHISKY"
## [124] "XAMAX"
## [125] "XAMAX.MENU"
```

Writing a function that can change a value not null by the name of Item in each column

```
to_nameItems<- function (df){
  for ( i in c(1:length(colnames(df)))){
    df[[i]]<-ifelse(df[i]!=0,colnames(df[i]),0)
  }
  return(df)
}
```

Applying this function to the matrix ‘fact’

```
test1<-as.data.frame(to_nameItems(fact))
test1 <- data.frame(lapply(test1[,1:125],unlist))# to convert variables related to products from vector ( list)
 to factor variables
```

Adding 2 ID columns and creating a matrix containing label Item names

```
itemlabels_matrix<-cbind(outcomes[,1:2],test1)#adding H.ID and PersonID to this matrix creating a new data set
called 'itemlabels_matrix'
head(itemlabels_matrix,2)
```

```
##   H.ID D.PersonID ABATILLES.PLATES ABATILLES.RED AVECESAR BADOIT.33cl
## 1    1        136                0             0        0           0
## 2    1        178                0             0        0           0
##   BAILEYS BIERE.SANS.GLUTEN Boisson.Rouge BRIE CAFEGOURMAND
## 1       0                 0             0    0            0
## 2       0                 0             0    0            0
##   CAFEGOURMANDMENU CAPPUCCINO CHAMPAGNE.BOUTEILLE CHAMPAGNE.COUPE CHOCOLAT
## 1                0          0                   0               0        0
## 2                0          0                   0               0        0
##   COCA COCA.ZERO Cocktail.Saint.Valentin.alcoolisÃ.
## 1    0         0                                  0
## 2    0         0                                  0
##   Cocktail.Saint.Valentin.sans.alcool COMPOTEE.CERISE.NOIRE CrÃ.me.au.bleu
## 1                                   0                     0              0
## 2                                   0                     0              0
##   CREME.POIVRE.VERT CREMEBRULEE CROQUANT DECA DESPERADOS Dessert.du.jour
## 1                 0           0        0    0          0               0
## 2                 0           0        0    0          0               0
##   Domaine.La.Colombette.RosÃ. DOUBLE.EXPRESSO DUOSALADE EVIAN EXPRESSO
## 1                           0               0         0     0 EXPRESSO
## 2                           0               0         0     0 EXPRESSO
##   FRAICHEUR FUMAX GATEAUCAROTTE GET GIN GRIMBERGEN GROSSEFRITE
## 1         0     0             0   0   0          0           0
## 2         0     0 GATEAUCAROTTE   0   0          0           0
##   HAMBOURGEOIS.DU.MOMENT HAMBOURGEOIS.DU.MOMENT.MENU HAMPE HOEGGARDEN
## 1                      0                           0     0          0
## 2                      0                           0     0          0
##   INFUSION JUS.FRUIT.MAISON KEKETTE.EXTRA KEKETTE.RED KIR LATTE.MACCHIATTO
## 1        0                0             0           0   0                0
## 2        0                0             0           0   0                0
##   LILLET MALIBU MARTINI MAX.CUVEE. MAXCHAMPETRE MAXCHAMPETRE.MENU
## 1      0      0       0          0            0                 0
## 2      0      0       0          0            0                 0
##   MAXCOCOTTE MAXCOCOTTE.MENU MAXHALEINE MAXHALEINE.MENU MAXIFLETTE
## 1          0               0          0               0          0
## 2          0               0          0               0          0
##   MAXIFLETTE.MENU MAXINUS MAXINUS.MENU MAXNAUDOU MAXNAUDOU.MENU MAXPARTOUT
## 1               0 MAXINUS            0         0              0          0
## 2               0       0            0         0              0          0
##   MAXPARTOUT.MENU MAXPOUSSIN MAXYONNAISE MENU.HAMBOURGEOIS MENU.PLAT
## 1               0          0           0 MENU.HAMBOURGEOIS         0
## 2               0          0           0                 0 MENU.PLAT
##   MINIMAX MOKACCINO MOUFLET MOUSSECHOCO NUGGETS OASIS ORANGINA
## 1       0         0       0           0       0     0        0
## 2       0         0       0           0       0     0        0
##   PAPOLLE.BLANC.MOEL PAPOLLE.BLANC.SEC PAPOLLE.ROSE PAPOLLE.ROUGE PASTIS
## 1                  0                 0            0             0      0
## 2                  0                 0            0             0      0
##   PATATEDOUCE PELFORTH PESTO PICHET.PUNCH PICHET.SANGRIA PINEAU
## 1 PATATEDOUCE        0     0            0              0      0
## 2           0        0     0            0              0      0
##   PINT.PELFORTH PLANTAMAX PLANTAMAX.MENU PLAT.DU.MOMENT PORTO PUNCH.Maison
## 1             0         0              0              0     0            0
## 2             0         0              0              0     0            0
##   RED.BULL RHUM RHUM.ARRANGE RICARD SALADEASIAT SANGRIA SAUCE.BARBECUE
## 1        0    0            0      0           0       0              0
## 2        0    0            0      0           0       0              0
##   Sauce.ForestiÃ.re SAUCE.POIVRE SAUMON SCHWEPPES SIROP SOUPEFRUIT TAPAS
## 1                 0            0      0         0     0 SOUPEFRUIT     0
## 2                 0            0      0         0     0          0     0
##   TARTARE TEQUILA THE THE.GLACE TOURISTE Verre.de.Boisson.Rouge
## 1       0       0   0         0        0                      0
## 2 TARTARE       0   0         0        0                      0
##   Verre.de.Bordeaux.Rouge.Agape VERRE.MAX.CUVEE VERRE.PAPOLLE.BLANC.MOEL
## 1                             0               0                        0
## 2                             0               0                        0
##   VERRE.PAPOLLE.BLANC.SEC VERRE.PAPOLLE.ROSE VERRE.PAPOLLE.ROUGE
## 1                       0                  0                   0
## 2                       0                  0                   0
##   Verre.RosÃ..la.Colombette VODKA WHISKY XAMAX XAMAX.MENU
## 1                         0     0      0     0          0
## 2                         0     0      0     0          0
```

Writing a function that can convert a occurence value to a binary value

```
to_binaryitems<- function (df){
  for ( i in c(1:length(colnames(df)))){
    df[[i]]<-ifelse(df[i]!=0,1,0)
  }
  return(df)
}
```

Applying this function to the matrix ‘fact’ and creating a matrix containing binary lablel values

```
test2<-to_binaryitems(fact)
test2 <- data.frame(lapply(test2[,1:125],unlist))# to convert variables related to products from vector ( list)
 to factor variable
test2<-data.frame(lapply(test2[,1:125],as.factor))
```

```
itembinary_matrix<-cbind(outcomes[,1:2],test2)
dim(itembinary_matrix)
```

```
## [1] 9940  127
```

Applyng Arules package onto ‘itemslabel_matrix’ https://cran.r-project.org/web/packages/arules/vignettes/arules.pdf(p.23) (https://cran.r-project.org/web/packages/arules/vignettes/arules.pdf(p.23))

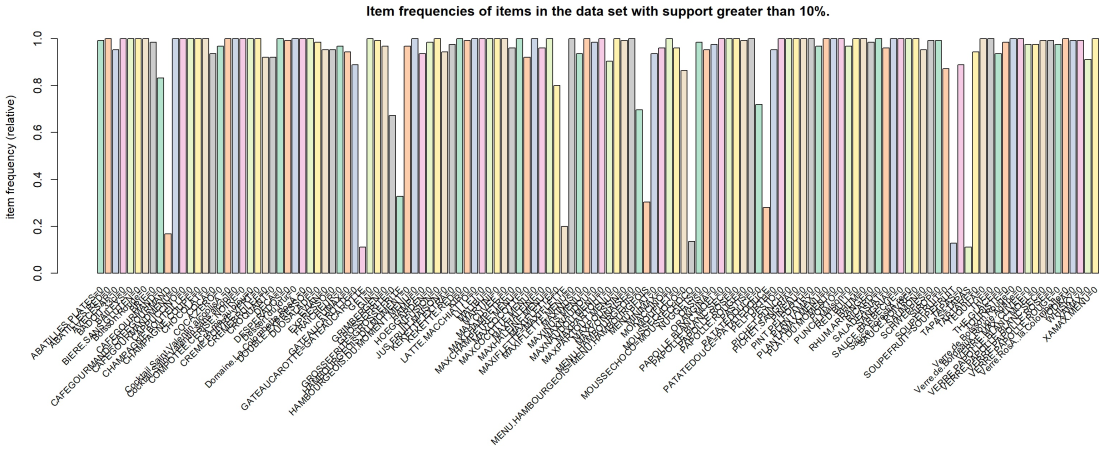
```
trans_itemlabel <- as(itemlabels_matrix[,3:127], "transactions")
```

```
summary(trans_itemlabel)
```

```
## transactions as itemMatrix in sparse format with
##  9940 rows (elements/itemsets/transactions) and
##  250 columns (items) and a density of 0.5
## 
## most frequent items:
##               Boisson.Rouge=0         CHAMPAGNE.BOUTEILLE=0
##                          9939                          9939
## Domaine.La.Colombette.RosÃ.=0                   MOKACCINO=0
##                          9936                          9935
##             CHAMPAGNE.COUPE=0                       (Other)
##                          9934                       1192817
## 
## element (itemset/transaction) length distribution:
## sizes
##  125 
## 9940 
## 
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
##     125     125     125     125     125     125 
## 
## includes extended item information - examples:
##                               labels        variables           levels
## 1                 ABATILLES.PLATES=0 ABATILLES.PLATES                0
## 2 ABATILLES.PLATES=ABATILLES.PLATES ABATILLES.PLATES ABATILLES.PLATES
## 3                    ABATILLES.RED=0    ABATILLES.RED                0
## 
## includes extended transaction information - examples:
##   transactionID
## 1             1
## 2             2
## 3             3
```

Let’s see which items are important in the data set we can use the itemFrequencyPlot(). In order to reduce the number of items, we only plot the item frequency for items with a support greater than 10% (using the parameter support).

```
#library(arulesViz)
#library(RColorBrewer)
itemFrequencyPlot(trans_itemlabel[c(seq(2,250,2))], support = 0.1, cex.names=0.8,col=brewer.pal(8,'Pastel2'),ma
in='Item frequencies of items in the data set with support greater than 10%.')
```



```
trans_itemlabel[c(1,3)]
```

```
## transactions in sparse format with
##  2 transactions (rows) and
##  250 items (columns)
```

Next, we recall the function apriori() to find all rules (the default association type for apriori()) with a minimum support of 10% and a confidence of 0.6.

```
rules <- apriori(trans_itemlabel[,126:250], parameter = list(support = 0.05, confidence = 0.,minlen=2,maxlen=4)
)
```

```
## Apriori
##
## Parameter specification:
##  confidence minval smax arem  aval originalSupport maxtime support minlen
##           0    0.1    1 none FALSE            TRUE       5    0.05      2
##  maxlen target   ext
##       4  rules FALSE
##
## Algorithmic control:
##  filter tree heap memopt load sort verbose
##     0.1 TRUE TRUE  FALSE TRUE    2    TRUE
##
## Absolute minimum support count: 497
##
## set item appearances ...[0 item(s)] done [0.00s].
## set transactions ...[125 item(s), 9940 transaction(s)] done [0.02s].
## sorting and recoding items ... [73 item(s)] done [0.01s].
## creating transaction tree ... done [0.01s].
## checking subsets of size 1 2 3 4
```

```
## Warning in apriori(trans_itemlabel[, 126:250], parameter = list(support =
## 0.05, : Mining stopped (maxlen reached). Only patterns up to a length of 4
## returned!
```

```
##  done [3.96s].
## writing ... [3777373 rule(s)] done [0.63s].
## creating S4 object  ... done [1.50s].
```

```
summary(rules)
```

```
## set of 3777373 rules
##
## rule length distribution (lhs + rhs):sizes
##       2       3       4
##    5062  168351 3603960
##
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   2.000   4.000   4.000   3.953   4.000   4.000
##
## summary of quality measures:
##     support          confidence           lift
##  Min.   :0.05000   Min.   :0.05065   Min.   :0.6022
##  1st Qu.:0.09235   1st Qu.:0.95288   1st Qu.:0.9994
##  Median :0.84406   Median :0.98640   Median :1.0000
##  Mean   :0.59200   Mean   :0.89001   Mean   :1.0012
##  3rd Qu.:0.92726   3rd Qu.:0.99708   3rd Qu.:1.0011
##  Max.   :0.99879   Max.   :1.00000   Max.   :1.7250
##
## mining info:
##                        data ntransactions support confidence
##  trans_itemlabel[, 126:250]          9940    0.05          0
```

As typical for association rule mining, the number of rules found is huge.

To analyze these rules, for example, subset() can be used to produce separate subsets of rules for each item which resulted form a given product in which the manager is interested

For instance, he want to know which are products to produce ‘Jus fruit maison’ in the right-hand-side of the rule.

The following code allows to do this requiring at the same time that the lift measure exceeds 1

```
rulesSoupfruit <- subset(rules, subset = rhs %in% "SOUPEFRUIT=SOUPEFRUIT" & lift > 1)
rulesSoupfruit
```

```
## set of 19624 rules
```

```
inspect(head(rulesSoupfruit, n = 10, by = "confidence"))
```

```
##      lhs                 rhs                        support confidence     lift
## [1]  {MAXPARTOUT=0,
##       MOUSSECHOCO=0,
##       NUGGETS=0}      => {SOUPEFRUIT=SOUPEFRUIT} 0.08772636  0.1145259 1.162806
## [2]  {MAXPARTOUT=0,
##       MOUFLET=0,
##       MOUSSECHOCO=0}  => {SOUPEFRUIT=SOUPEFRUIT} 0.08853119  0.1145237 1.162784
## [3]  {MAXPARTOUT=0,
##       MOUSSECHOCO=0,
##       SAUCE.POIVRE=0} => {SOUPEFRUIT=SOUPEFRUIT} 0.08802817  0.1129907 1.147219
## [4]  {MAXPARTOUT=0,
##       MINIMAX=0,
##       MOUSSECHOCO=0}  => {SOUPEFRUIT=SOUPEFRUIT} 0.08843058  0.1129530 1.146836
## [5]  {MAXPARTOUT=0,
##       MOUSSECHOCO=0,
##       THE=0}          => {SOUPEFRUIT=SOUPEFRUIT} 0.08993964  0.1122833 1.140037
## [6]  {MAXPARTOUT=0,
##       MOUSSECHOCO=0,
##       PELFORTH=0}     => {SOUPEFRUIT=SOUPEFRUIT} 0.08631791  0.1122155 1.139349
## [7]  {MOUSSECHOCO=0,
##       NUGGETS=0,
##       SAUCE.POIVRE=0} => {SOUPEFRUIT=SOUPEFRUIT} 0.09094567  0.1120059 1.137221
## [8]  {MAXPARTOUT=0,
##       MOUSSECHOCO=0,
##       PLANTAMAX=0}    => {SOUPEFRUIT=SOUPEFRUIT} 0.08702213  0.1118438 1.135575
## [9]  {MOUFLET=0,
##       MOUSSECHOCO=0,
##       SAUCE.POIVRE=0} => {SOUPEFRUIT=SOUPEFRUIT} 0.09164990  0.1118203 1.135336
## [10] {MOUSSECHOCO=0,
##       NUGGETS=0,
##       PELFORTH=0}     => {SOUPEFRUIT=SOUPEFRUIT} 0.08943662  0.1117677 1.134801
```

We can observe that transforming the itemlabel_matrix to transaction matrix “trans_itemlabel matrix” by function ’as" is not a good solution.

See this post:

https://stackoverflow.com/questions/44618956/convert-r-data-frame-column-to-arules-transactions (https://stackoverflow.com/questions/44618956/convert-r-data-frame-column-to-arules-transactions)

“Have a look at the examples in ? transactions. You need a list with vectors of items (item labels) and not a data.frame.”

or:

http://mhahsler.github.io/arules/reference/transactions-class.html (http://mhahsler.github.io/arules/reference/transactions-class.html): said we need transform from dat frame to matrix

or:

https://stackoverflow.com/questions/19569391/convert-character-matrix-to-logical (https://stackoverflow.com/questions/19569391/convert-character-matrix-to-logical)

4.1) Applying association rules

As following the code for converting ourbinaty item data set to a logical item matrix

```
items_matrix <- as.matrix(itembinary_matrix[,3:127])
items_matrix<-items_matrix != "0" & items_matrix != "FALSE" # https://stackoverflow.com/questions/19569391/conv
ert-character-matrix-to-logical
head(items_matrix,1)
```

```
##      ABATILLES.PLATES ABATILLES.RED AVECESAR BADOIT.33cl BAILEYS
## [1,]            FALSE         FALSE    FALSE       FALSE   FALSE
##      BIERE.SANS.GLUTEN Boisson.Rouge  BRIE CAFEGOURMAND CAFEGOURMANDMENU
## [1,]             FALSE         FALSE FALSE        FALSE            FALSE
##      CAPPUCCINO CHAMPAGNE.BOUTEILLE CHAMPAGNE.COUPE CHOCOLAT  COCA
## [1,]      FALSE               FALSE           FALSE    FALSE FALSE
##      COCA.ZERO Cocktail.Saint.Valentin.alcoolisÃ.
## [1,]     FALSE                              FALSE
##      Cocktail.Saint.Valentin.sans.alcool COMPOTEE.CERISE.NOIRE
## [1,]                               FALSE                 FALSE
##      CrÃ.me.au.bleu CREME.POIVRE.VERT CREMEBRULEE CROQUANT  DECA
## [1,]          FALSE             FALSE       FALSE    FALSE FALSE
##      DESPERADOS Dessert.du.jour Domaine.La.Colombette.RosÃ.
## [1,]      FALSE           FALSE                       FALSE
##      DOUBLE.EXPRESSO DUOSALADE EVIAN EXPRESSO FRAICHEUR FUMAX
## [1,]           FALSE     FALSE FALSE     TRUE     FALSE FALSE
##      GATEAUCAROTTE   GET   GIN GRIMBERGEN GROSSEFRITE
## [1,]         FALSE FALSE FALSE      FALSE       FALSE
##      HAMBOURGEOIS.DU.MOMENT HAMBOURGEOIS.DU.MOMENT.MENU HAMPE HOEGGARDEN
## [1,]                  FALSE                       FALSE FALSE      FALSE
##      INFUSION JUS.FRUIT.MAISON KEKETTE.EXTRA KEKETTE.RED   KIR
## [1,]    FALSE            FALSE         FALSE       FALSE FALSE
##      LATTE.MACCHIATTO LILLET MALIBU MARTINI MAX.CUVEE. MAXCHAMPETRE
## [1,]            FALSE  FALSE  FALSE   FALSE      FALSE        FALSE
##      MAXCHAMPETRE.MENU MAXCOCOTTE MAXCOCOTTE.MENU MAXHALEINE
## [1,]             FALSE      FALSE           FALSE      FALSE
##      MAXHALEINE.MENU MAXIFLETTE MAXIFLETTE.MENU MAXINUS MAXINUS.MENU
## [1,]           FALSE      FALSE           FALSE    TRUE        FALSE
##      MAXNAUDOU MAXNAUDOU.MENU MAXPARTOUT MAXPARTOUT.MENU MAXPOUSSIN
## [1,]     FALSE          FALSE      FALSE           FALSE      FALSE
##      MAXYONNAISE MENU.HAMBOURGEOIS MENU.PLAT MINIMAX MOKACCINO MOUFLET
## [1,]       FALSE              TRUE     FALSE   FALSE     FALSE   FALSE
##      MOUSSECHOCO NUGGETS OASIS ORANGINA PAPOLLE.BLANC.MOEL
## [1,]       FALSE   FALSE FALSE    FALSE              FALSE
##      PAPOLLE.BLANC.SEC PAPOLLE.ROSE PAPOLLE.ROUGE PASTIS PATATEDOUCE
## [1,]             FALSE        FALSE         FALSE  FALSE        TRUE
##      PELFORTH PESTO PICHET.PUNCH PICHET.SANGRIA PINEAU PINT.PELFORTH
## [1,]    FALSE FALSE        FALSE          FALSE  FALSE         FALSE
##      PLANTAMAX PLANTAMAX.MENU PLAT.DU.MOMENT PORTO PUNCH.Maison RED.BULL
## [1,]     FALSE          FALSE          FALSE FALSE        FALSE    FALSE
##       RHUM RHUM.ARRANGE RICARD SALADEASIAT SANGRIA SAUCE.BARBECUE
## [1,] FALSE        FALSE  FALSE       FALSE   FALSE          FALSE
##      Sauce.ForestiÃ.re SAUCE.POIVRE SAUMON SCHWEPPES SIROP SOUPEFRUIT
## [1,]             FALSE        FALSE  FALSE     FALSE FALSE       TRUE
##      TAPAS TARTARE TEQUILA   THE THE.GLACE TOURISTE Verre.de.Boisson.Rouge
## [1,] FALSE   FALSE   FALSE FALSE     FALSE    FALSE                  FALSE
##      Verre.de.Bordeaux.Rouge.Agape VERRE.MAX.CUVEE
## [1,]                         FALSE           FALSE
##      VERRE.PAPOLLE.BLANC.MOEL VERRE.PAPOLLE.BLANC.SEC VERRE.PAPOLLE.ROSE
## [1,]                    FALSE                   FALSE              FALSE
##      VERRE.PAPOLLE.ROUGE Verre.RosÃ..la.Colombette VODKA WHISKY XAMAX
## [1,]               FALSE                     FALSE FALSE  FALSE FALSE
##      XAMAX.MENU
## [1,]      FALSE
```

Coercing this matrix

```
trans_itemlabel1 <- as(items_matrix, "transactions")
trans_itemlabel1
```

```
## transactions in sparse format with
##  9940 transactions (rows) and
##  125 items (columns)
```

Let us check the most frequently purchased products using the summary function.

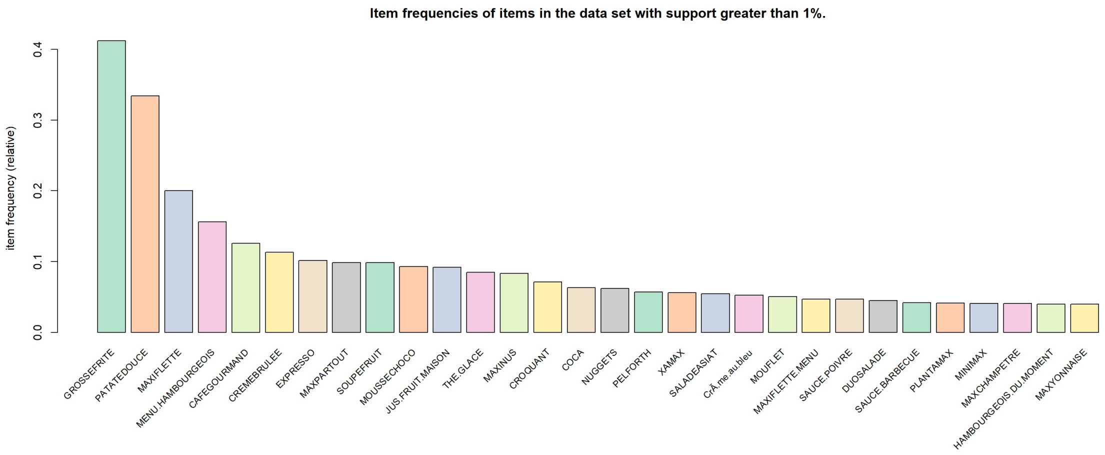
```
summary(trans_itemlabel1)
```

```
## transactions as itemMatrix in sparse format with
##  9940 rows (elements/itemsets/transactions) and
##  125 columns (items) and a density of 0.03076217
##
## most frequent items:
##       GROSSEFRITE       PATATEDOUCE        MAXIFLETTE MENU.HAMBOURGEOIS
##              4096              3324              1990              1552
##      CAFEGOURMAND           (Other)
##              1252             26008
##
## element (itemset/transaction) length distribution:
## sizes
##    1    2    3    4    5    6    7    8    9   10   11   12   13   14   15
##  630 1345 2284 2634 1864  668  245  119   84   33   13    6    6    3    1
##   16   17
##    2    3
##
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   1.000   3.000   4.000   3.845   5.000  17.000
##
## includes extended item information - examples:
##             labels
## 1 ABATILLES.PLATES
## 2    ABATILLES.RED
## 3         AVECESAR
```

The Top 5 Items sold in transactions as GROSSEFRITE,ATATEDOUCE, MAXIFLETTE, MENU.HAMBOURGEOIS and CAFEGOURMAND

We’re going to make an Item Frequency Histogram for TOP30 Items whose supports are greater than 1%

```
#library(arules)
#library(arulesViz)
#library(RColorBrewer)
itemFrequencyPlot(trans_itemlabel1, topN=30,support = 0.01, cex.names=0.8,col=brewer.pal(8,'Pastel2'),main='Ite
m frequencies of items in the data set with support greater than 1%.')
```



See: (https://www.analyticsvidhya.com/blog/2017/08/mining-frequent-items-usingapriori- (https://www.analyticsvidhya.com/blog/2017/08/mining-frequent-items-usingapriori-) algorithm/?share=reddit&nb=1)

Next, we call the function apriori() to find all rules (the default association type for apriori()) with a minimum support of 1% and a confidence of 0.6.

```
rules1 <- apriori(trans_itemlabel1, parameter = list(support = 0.01, confidence = 0.3,minlen=2,maxlen=6))
```

```
## Apriori
##
## Parameter specification:
##  confidence minval smax arem  aval originalSupport maxtime support minlen
##         0.3    0.1    1 none FALSE            TRUE       5    0.01      2
##  maxlen target   ext
##       6  rules FALSE
##
## Algorithmic control:
##  filter tree heap memopt load sort verbose
##     0.1 TRUE TRUE  FALSE TRUE    2    TRUE
##
## Absolute minimum support count: 99
##
## set item appearances ...[0 item(s)] done [0.00s].
## set transactions ...[125 item(s), 9940 transaction(s)] done [0.00s].
## sorting and recoding items ... [68 item(s)] done [0.00s].
## creating transaction tree ... done [0.00s].
## checking subsets of size 1 2 3 4 done [0.00s].
## writing ... [148 rule(s)] done [0.00s].
## creating S4 object  ... done [0.00s].
```

```
summary(rules1)
```

```
## set of 148 rules
##
## rule length distribution (lhs + rhs):sizes
##  2  3  4
## 85 59  4
##
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   2.000   2.000   2.000   2.453   3.000   4.000
##
## summary of quality measures:
##     support          confidence          lift
##  Min.   :0.01016   Min.   :0.3003   Min.   : 0.8684
##  1st Qu.:0.01363   1st Qu.:0.4095   1st Qu.: 1.2311
##  Median :0.01866   Median :0.4809   Median : 1.5210
##  Mean   :0.02331   Mean   :0.5478   Mean   : 3.4348
##  3rd Qu.:0.02508   3rd Qu.:0.6042   3rd Qu.: 5.7414
##  Max.   :0.10201   Max.   :1.0000   Max.   :25.2412
##
## mining info:
##              data ntransactions support confidence
##  trans_itemlabel1          9940    0.01        0.3
```

With min support =0.01, min lenght rule = 2 and max lenght rule =6 , we have obtained a set of 151 rules

Sorting TopN=30 rules by ‘lift’ metric

```
top.lift <- sort(rules1, decreasing = TRUE, na.last = NA, by = "lift")
inspect(top.lift[1:10])
```

```
##      lhs                     rhs                  support confidence      lift
## [1]  {EXPRESSO,
##       GROSSEFRITE,
##       MENU.HAMBOURGEOIS} => {CAFEGOURMANDMENU} 0.01106640  0.9090909 25.241239
## [2]  {EXPRESSO,
##       MENU.HAMBOURGEOIS} => {CAFEGOURMANDMENU} 0.02012072  0.8928571 24.790503
## [3]  {GROSSEFRITE,
##       MINIMAX}           => {MOUFLET}          0.02474849  0.7299703 14.396637
## [4]  {GROSSEFRITE,
##       MOUFLET}           => {MINIMAX}          0.02474849  0.4900398 11.909526
## [5]  {MINIMAX}           => {MOUFLET}          0.02474849  0.6014670 11.862266
## [6]  {MOUFLET}           => {MINIMAX}          0.02474849  0.4880952 11.862266
## [7]  {GROSSEFRITE,
##       NUGGETS}           => {MOUFLET}          0.02384306  0.5550351 10.946526
## [8]  {GROSSEFRITE,
##       OASIS}             => {MOUFLET}          0.01287726  0.4942085  9.746890
## [9]  {CREMEBRULEE,
##       MENU.HAMBOURGEOIS} => {MAXIFLETTE.MENU}  0.01287726  0.3605634  7.625532
## [10] {MOUFLET}           => {NUGGETS}          0.02394366  0.4722222  7.583019
```

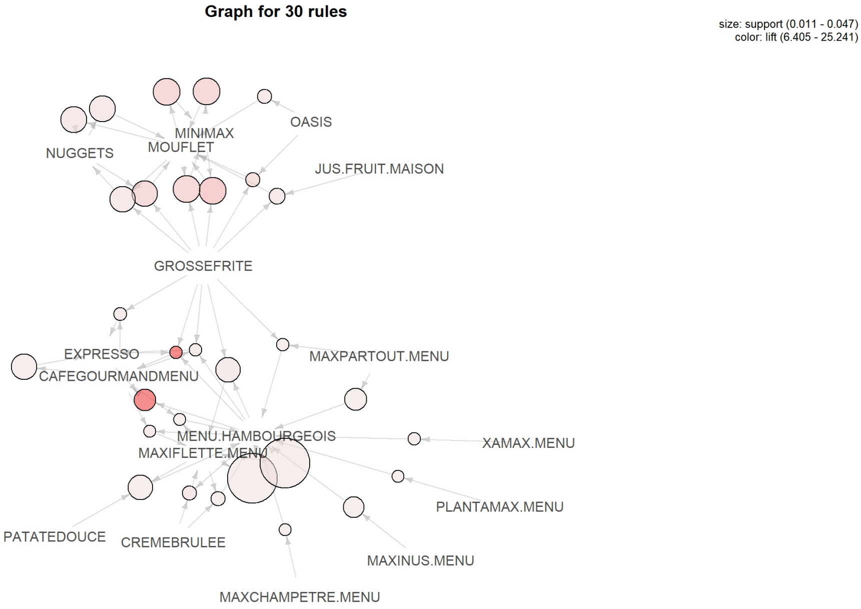
4.2) Graphical Representation

Moving forward in the visualisation, we can use a graph to highlight the support and lifts of various items in our repository but mostly to see which product is associated with which one in the sales environment.

The size of graph nodes is based on support levels and the colour on lift ratios. The incoming lines show the Antecedants or the LHS and the RHS is represented by names of items.

```
plot(top.lift[1:30],
method = "graph",
control = list(type = "items"))
```

```
## Warning: Unknown control parameters: type
## Available control parameters (with default values):
## main  =  Graph for 30 rules
## nodeColors    =  c("#66CC6680", "#9999CC80")
## nodeCol   =  c("#EE0000FF", "#EE0303FF", "#EE0606FF", "#EE0909FF", "#EE0C0CFF", "#EE0F0FFF", "#EE1212FF", "#
EE1515FF", "#EE1818FF", "#EE1B1BFF", "#EE1E1EFF", "#EE2222FF", "#EE2525FF", "#EE2828FF", "#EE2B2BFF", "#EE2E2EF
F", "#EE3131FF", "#EE3434FF", "#EE3737FF", "#EE3A3AFF", "#EE3D3DFF", "#EE4040FF", "#EE4444FF", "#EE4747FF", "#E
E4A4AFF", "#EE4D4DFF", "#EE5050FF", "#EE5353FF", "#EE5656FF", "#EE5959FF", "#EE5C5CFF", "#EE5F5FFF", "#EE6262FF
", "#EE6565FF", "#EE6969FF", "#EE6C6CFF", "#EE6F6FFF", "#EE7272FF", "#EE7575FF",  "#EE7878FF", "#EE7B7BFF", "#E
E7E7EFF", "#EE8181FF", "#EE8484FF", "#EE8888FF", "#EE8B8BFF", "#EE8E8EFF", "#EE9191FF", "#EE9494FF", "#EE9797FF
", "#EE9999FF", "#EE9B9BFF", "#EE9D9DFF", "#EE9F9FFF", "#EEA0A0FF", "#EEA2A2FF", "#EEA4A4FF", "#EEA5A5FF", "#EE
A7A7FF", "#EEA9A9FF", "#EEABABFF", "#EEACACFF", "#EEAEAEFF", "#EEB0B0FF", "#EEB1B1FF", "#EEB3B3FF", "#EEB5B5FF"
, "#EEB7B7FF", "#EEB8B8FF", "#EEBABAFF", "#EEBCBCFF", "#EEBDBDFF", "#EEBFBFFF", "#EEC1C1FF", "#EEC3C3FF", "#EEC
4C4FF", "#EEC6C6FF", "#EEC8C8FF",  "#EEC9C9FF", "#EECBCBFF", "#EECDCDFF", "#EECFCFFF", "#EED0D0FF", "#EED2D2FF"
, "#EED4D4FF", "#EED5D5FF", "#EED7D7FF", "#EED9D9FF", "#EEDBDBFF", "#EEDCDCFF", "#EEDEDEFF", "#EEE0E0FF", "#EEE
1E1FF", "#EEE3E3FF", "#EEE5E5FF", "#EEE7E7FF", "#EEE8E8FF", "#EEEAEAFF", "#EEECECFF", "#EEEEEEFF")
## edgeCol   =  c("#474747FF", "#494949FF", "#4B4B4BFF", "#4D4D4DFF", "#4F4F4FFF", "#515151FF", "#535353FF", "#
555555FF", "#575757FF", "#595959FF", "#5B5B5BFF", "#5E5E5EFF", "#606060FF", "#626262FF", "#646464FF", "#666666F
F", "#686868FF", "#6A6A6AFF", "#6C6C6CFF", "#6E6E6EFF", "#707070FF", "#727272FF", "#747474FF", "#767676FF", "#7
87878FF", "#7A7A7AFF", "#7C7C7CFF", "#7E7E7EFF", "#808080FF", "#828282FF", "#848484FF", "#868686FF", "#888888FF
", "#8A8A8AFF", "#8C8C8CFF", "#8D8D8DFF", "#8F8F8FFF", "#919191FF", "#939393FF",  "#959595FF", "#979797FF", "#9
99999FF", "#9A9A9AFF", "#9C9C9CFF", "#9E9E9EFF", "#A0A0A0FF", "#A2A2A2FF", "#A3A3A3FF", "#A5A5A5FF", "#A7A7A7FF
", "#A9A9A9FF", "#AAAAAAFF", "#ACACACFF", "#AEAEAEFF", "#AFAFAFFF", "#B1B1B1FF", "#B3B3B3FF", "#B4B4B4FF", "#B6
B6B6FF", "#B7B7B7FF", "#B9B9B9FF", "#BBBBBBFF", "#BCBCBCFF", "#BEBEBEFF", "#BFBFBFFF", "#C1C1C1FF", "#C2C2C2FF"
, "#C3C3C4FF", "#C5C5C5FF", "#C6C6C6FF", "#C8C8C8FF", "#C9C9C9FF", "#CACACAFF", "#CCCCCCFF", "#CDCDCDFF", "#CEC
ECEFF", "#CFCFCFFF", "#D1D1D1FF",  "#D2D2D2FF", "#D3D3D3FF", "#D4D4D4FF", "#D5D5D5FF", "#D6D6D6FF", "#D7D7D7FF"
, "#D8D8D8FF", "#D9D9D9FF", "#DADADAFF", "#DBDBDBFF", "#DCDCDCFF", "#DDDDDDFF", "#DEDEDEFF", "#DEDEDEFF", "#DFD
FDFFF", "#E0E0E0FF", "#E0E0E0FF", "#E1E1E1FF", "#E1E1E1FF", "#E2E2E2FF", "#E2E2E2FF", "#E2E2E2FF")
## alpha     =  0.5
## cex   =  1
## itemLabels    =  TRUE
## labelCol  =  #000000B2
## measureLabels     =  FALSE
## precision     =  3
## layout    =  NULL
## layoutParams  =  list()
## arrowSize     =  0.5
## engine    =  igraph
## plot  =  TRUE
## plot_options  =  list()
## max   =  100
## verbose   =  FALSE
```



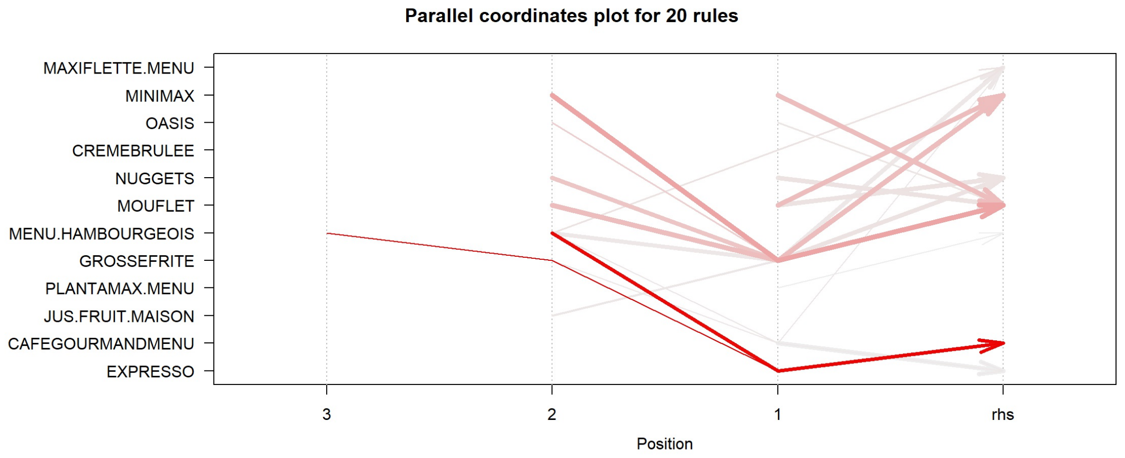
The above graph shows us that most of our transactions were consolidated around “Grossefrite”, âMenu HamBourgeoisâ, " Maxiflette Menu“, and”Mouflet“.

We also see that all Expresso and Cafe Gourmand are very strongly associated so we must place these together.

4.3) Individual Rule Representation

The next plot oôers us a parallel coordinate system ofvisualisation. It would help us clearly see that which produc tsalong with which ones, result in what kinds of sales. As mentioned above, the RHS is the Consequent or the item we propose the customer will buy; the positions are in the LHS where 2 is the most recent addition to our basket and 1 is the item we previously had

```
plot(top.lift[1:20],
method = "paracoord",
control = list(reorder = TRUE))
```

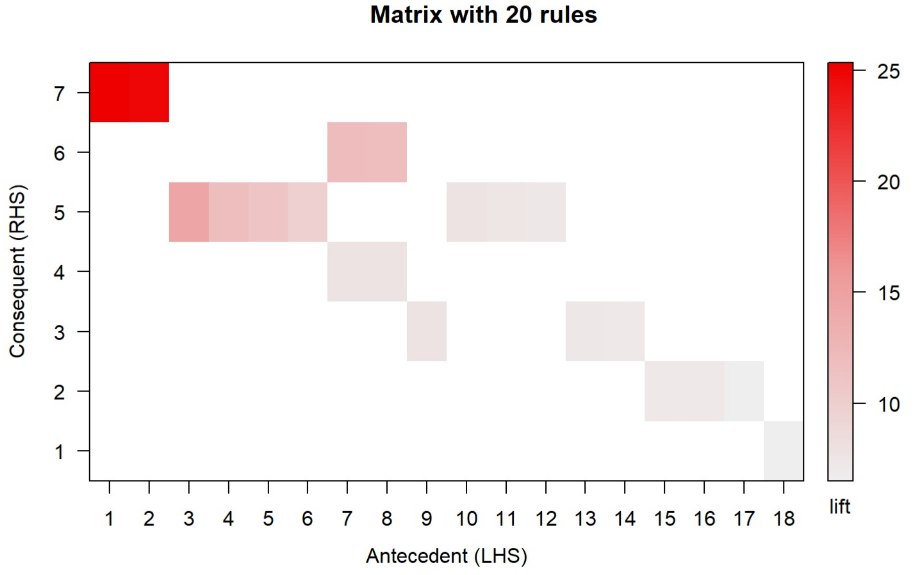


The topmost rule shows us that when the customer have NUGGETS and Jus de Fruit Maison in his shopping cart, He is highly likely to buy Maxiflette Menu to go along with those as well.

If we need to get a matrix representation, an alternate code option would be:

```
plot(top.lift[1:20],
method = "matrix",
control = list(reorder = TRUE))
```

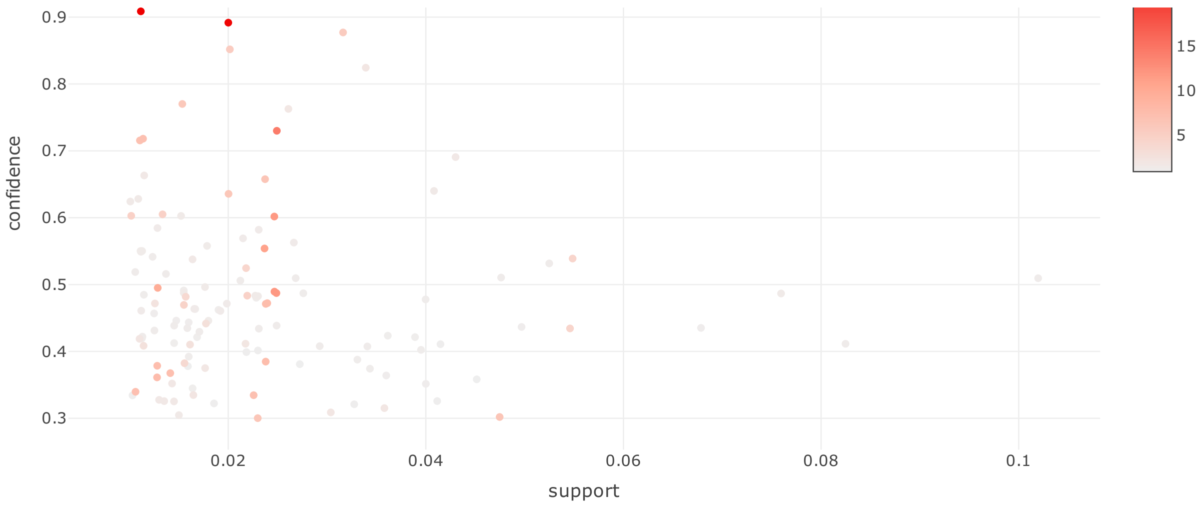
```
## Itemsets in Antecedent (LHS)
##  [1] "{EXPRESSO,GROSSEFRITE,MENU.HAMBOURGEOIS}"
##  [2] "{EXPRESSO,MENU.HAMBOURGEOIS}"
##  [3] "{GROSSEFRITE,MINIMAX}"
##  [4] "{MINIMAX}"
##  [5] "{GROSSEFRITE,NUGGETS}"
##  [6] "{GROSSEFRITE,OASIS}"
##  [7] "{GROSSEFRITE,MOUFLET}"
##  [8] "{MOUFLET}"
##  [9] "{CREMEBRULEE,MENU.HAMBOURGEOIS}"
## [10] "{NUGGETS}"
## [11] "{OASIS}"
## [12] "{GROSSEFRITE,JUS.FRUIT.MAISON}"
## [13] "{CAFEGOURMANDMENU,MENU.HAMBOURGEOIS}"
## [14] "{GROSSEFRITE,MENU.HAMBOURGEOIS}"
## [15] "{CAFEGOURMANDMENU,GROSSEFRITE}"
## [16] "{CAFEGOURMANDMENU,GROSSEFRITE,MENU.HAMBOURGEOIS}"
## [17] "{CAFEGOURMANDMENU}"
## [18] "{PLANTAMAX.MENU}"
## Itemsets in Consequent (RHS)
## [1] "{MENU.HAMBOURGEOIS}" "{EXPRESSO}"          "{MAXIFLETTE.MENU}"
## [4] "{NUGGETS}"           "{MOUFLET}"           "{MINIMAX}"
## [7] "{CAFEGOURMANDMENU}"
```



4.4) Interactive Scatterplot

These plots show us each and every rule visualised into a form of a scatterplot. The conôdence levels are plotted on the Y axis and Support levels on the X axis for each rule. We can hover over them in our interactive plot to see the rule.

```
arulesViz::plotly_arules(top.lift)
```



If we want to get a data frame describing the top20 rules by lift metric, the following code will allow us to do that

```
top20_df=as(top.lift,"data.frame")
top20_df[c(1:10),1:4]
```

```
##                                                              rules
## 148 {EXPRESSO,GROSSEFRITE,MENU.HAMBOURGEOIS} => {CAFEGOURMANDMENU}
## 97              {EXPRESSO,MENU.HAMBOURGEOIS} => {CAFEGOURMANDMENU}
## 91                              {GROSSEFRITE,MINIMAX} => {MOUFLET}
## 92                              {GROSSEFRITE,MOUFLET} => {MINIMAX}
## 31                                          {MINIMAX} => {MOUFLET}
## 32                                          {MOUFLET} => {MINIMAX}
## 118                             {GROSSEFRITE,NUGGETS} => {MOUFLET}
## 89                                {GROSSEFRITE,OASIS} => {MOUFLET}
## 107           {CREMEBRULEE,MENU.HAMBOURGEOIS} => {MAXIFLETTE.MENU}
## 55                                          {MOUFLET} => {NUGGETS}
##        support confidence      lift
## 148 0.01106640  0.9090909 25.241239
## 97  0.02012072  0.8928571 24.790503
## 91  0.02474849  0.7299703 14.396637
## 92  0.02474849  0.4900398 11.909526
## 31  0.02474849  0.6014670 11.862266
## 32  0.02474849  0.4880952 11.862266
## 118 0.02384306  0.5550351 10.946526
## 89  0.01287726  0.4942085  9.746890
## 107 0.01287726  0.3605634  7.625532
## 55  0.02394366  0.4722222  7.583019
```

As typical for association rule mining, the number of rules found is huge.

To analyze these rules, for example, subset() can be used to produce separate subsets of rules for each item which resulted form a given product in which the manager is interested

For instance, if we want to know which are products to produce ‘Jus fruit maison’ in the right-hand-side of the rule.

The following code allows to do this requiring at the same time that the lift measure exceeds 1

```
rulesSoupfruit1<- subset(rules1, subset = rhs %in% "SOUPEFRUIT" & lift > 0.1)
rulesSoupfruit1
```

```
## set of 0 rules
```

```
inspect(rulesSoupfruit1, n = 10, by = "confidence")
```

END OF WORK

Optional: eXPORTING DATA SETS TO csv Files

```
#write.csv(itembinary_matrix,file="itembinaryID.csv", row.names=FALSE) # say "temp.csv" is your text file
#write.csv(itemlabels_matrix,file="itemlabels_matrix.csv", row.names=FALSE)
#write.csv(outcomes,file="itemquanti.csv", row.names=FALSE)
#write.csv(User,file="User.csv", row.names=FALSE)
#write.csv(items_matrix,file="items_matrix.csv",row.names = FALSE)
#write.csv(It_nbdiner,file="items_nbdiner.csv",row.names = FALSE)
#write.csv(It_netprice,file="items_netprice.csv",row.names = FALSE)
#write.csv(IT_return_cust,file="items_return_cust.csv",row.names = FALSE)
#write.csv(z,file="seen_data.csv",row.names = FALSE)
```