SMM636 Machine Learning (PRD2 A 2019/20)

R exercises 10: Unsupervised learning (Part I)

DR. Rui Zhu, DR. Feng Zhou

2020-02-27

In R exercise 10, you will know

- How to apply market basket analysis
- How to apply model-based clustering

Don't forget to change your working directory!

1 Market basket analysis

We take the Online-Retail.xlsx data as an example https://archive.ics.uci.edu/ml/datasets/Online%20Retail. Read the data and reformat some variables.

This is a transnational data set which contains all the transactions occurring between 01/12/2010 and 09/12/2011 for a UK-based and registered non-store online retail.The company mainly sells unique all-occasion gifts. Many customers of the company are wholesalers.

```
#install.packages("readxl")
library(readxl)
library(plyr)
library(dplyr)
```

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

```
## The following objects are masked from 'package:plyr':
##
##     arrange, count, desc, failwith, id, mutate, rename, summarise,
##     summarize
```

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

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

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

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

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

```
retail=read_excel('Online_retail.xlsx')
### remove missing values
retail=retail[complete.cases(retail), ]
################################################
### transform data to proper formats
retail$InvoiceNo=gsub("C5", "5", retail$InvoiceNo)
retail$InvoiceNo=as.numeric(as.character(retail$InvoiceNo))
retail$Description=gsub(",", " ", retail$Description)
retail$Description = as.factor(retail$Description)
retail$Country = as.factor(retail$Country)
retail$Date=as.Date(retail$InvoiceDate)
retail$Time=format(retail$InvoiceDate,"%H:%M:%S")
```

Transform the data to the transaction object that should be used in the `apriori` function in the `arules` package.

```
############################################################
### transform the data from data frame to transactions
transaction <- ddply(retail,c("InvoiceNo","Date"),
                     function(df1)paste(df1$Description,
                                        collapse = ","))
transaction=transaction[,3]
write.table(transaction,"transactions.csv", quote = FALSE,
          row.names = FALSE,col.names = FALSE)
################################################
### read the transaction data for association rules analysis
transc = read.transactions('transactions.csv', header=FALSE,
                           format = 'basket', sep=",",
                           quote="",rm.duplicates = TRUE)
```

```
## distribution of transactions with duplicates:
## items
##    1    2    3    4    5    6    7    8    9   10   11   12   13   14   15
## 1090  505  287  204  131  113   93   59   45   45   34   24   26   15    9
##   16   17   18   19   20   21   22   23   24   25   26   27   28   29   30
##    7   18    8    8    2    1    4    3    4    4    6    2    2    3    2
##   32   33   34   36   37   42   46   47   50
##    1    1    1    1    2    1    1    1    1
```

The we could get the summary of the transaction object.

```
summary(transc)
```

```
## transactions as itemMatrix in sparse format with
##  22190 rows (elements/itemsets/transactions) and
##  3882 columns (items) and a density of 0.004602435
##
## most frequent items:
```

```
## WHITE HANGING HEART T-LIGHT HOLDER           REGENCY CAKESTAND 3 TIER
##                                2013                               1884
##            JUMBO BAG RED RETROSPOT                       PARTY BUNTING
##                               1643                               1399
##      ASSORTED COLOUR BIRD ORNAMENT                             (Other)
##                               1385                             388137
##
## element (itemset/transaction) length distribution:
## sizes
##    1    2    3    4    5    6    7    8    9   10   11   12   13   14   15
## 3435 1455 1018  773  765  654  633  618  619  548  552  501  503  524  548
##   16   17   18   19   20   21   22   23   24   25   26   27   28   29   30
##  550  462  437  484  433  394  337  346  303  244  255  238  238  268  221
##   31   32   33   34   35   36   37   38   39   40   41   42   43   44   45
##  196  172  159  169  134  120  125  118  131  116  118   97   93   94   91
##   46   47   48   49   50   51   52   53   54   55   56   57   58   59   60
##   84   76   78   78   79   52   58   68   66   64   45   58   42   32   54
##   61   62   63   64   65   66   67   68   69   70   71   72   73   74   75
##   36   25   40   36   32   36   30   35   24   30   30   20   22   26   24
##   76   77   78   79   80   81   82   83   84   85   86   87   88   89   90
##   22   17   19   11   13   18   19   15   21   15   13    9   13   11    9
##   91   92   93   94   95   96   97   98   99  100  101  102  103  104  105
##    9   15   12    7    5    8    8   12    5   11    8    3    6    7    2
##  106  107  108  109  110  111  112  113  114  115  116  117  118  119  120
##    3    6    4    2    4    4    3    3    6    6    8    3    4    5    5
##  121  122  123  124  125  126  127  128  129  130  131  132  134  135  136
##    5    7    3    4    3    2    5    1    1    2    3    2    2    2    3
##  137  138  139  140  141  142  143  144  145  146  148  149  150  151  153
##    1    2    1    1    4    1    1    1    2    2    1    3    1    1    1
##  154  156  157  163  165  169  171  175  176  178  179  180  181  183  187
##    1    1    1    1    2    2    1    1    2    1    1    1    1    1    1
##  192  193  195  202  204  208  210  219  227  249  259  262  270  280  333
##    1    1    1    1    1    1    1    1    1    1    1    1    1    1    1
##  347  352  363  375  386  419  434  439  525  529  541
##    1    1    1    1    1    1    1    1    1    1    1
##
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    1.00    3.00   12.00   17.87   24.00  541.00
##
## includes extended item information - examples:
##                       labels
## 1     10 COLOUR SPACEBOY PEN
## 2 12 COLOURED PARTY BALLOONS
## 3  12 DAISY PEGS IN WOOD BOX
```

We could also get a plot of the most frequently bought items.

```
### get the plot of the most frequently bought items
itemFrequencyPlot(transc, topN=20, type='absolute')
```

Now we could use the `apriori` function to find the association rules. Note that you could change the parameters based on your needs.

```
### generate assosication rules
association.rules <- apriori(transc, parameter = list(supp=0.01, conf=0.8,maxlen=10))
```

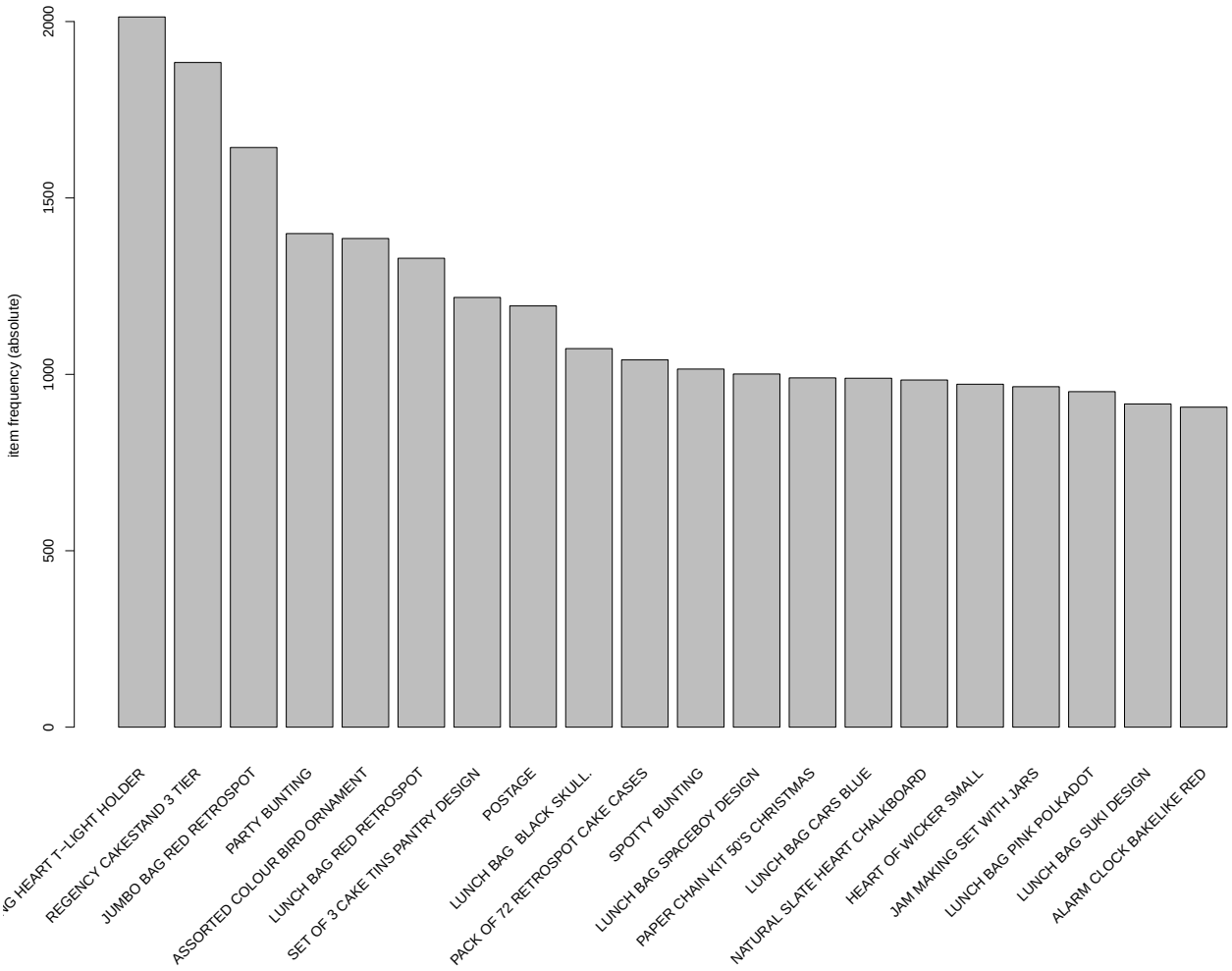

Figure 1: The most frequently bought items.

```
## Apriori
##
## Parameter specification:
##  confidence minval smax arem  aval originalSupport maxtime support minlen
##         0.8    0.1    1 none FALSE            TRUE       5    0.01      1
##  maxlen target   ext
##      10  rules FALSE
##
## Algorithmic control:
##  filter tree heap memopt load sort verbose
##     0.1 TRUE TRUE  FALSE TRUE    2    TRUE
##
## Absolute minimum support count: 221
##
## set item appearances ...[0 item(s)] done [0.00s].
## set transactions ...[3882 item(s), 22190 transaction(s)] done [0.16s].
## sorting and recoding items ... [518 item(s)] done [0.01s].
## creating transaction tree ... done [0.01s].
## checking subsets of size 1 2 3 4 done [0.01s].
## writing ... [10 rule(s)] done [0.00s].
## creating S4 object  ... done [0.01s].
```

```
summary(association.rules)
```

```
## set of 10 rules
##
## rule length distribution (lhs + rhs):sizes
## 2 3 4
## 3 5 2
##
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    2.00    2.25    3.00    2.90    3.00    4.00
##
## summary of quality measures:
##     support          confidence          lift           count
##  Min.   :0.01023   Min.   :0.8195   Min.   :21.90   Min.   :227.0
##  1st Qu.:0.01069   1st Qu.:0.8318   1st Qu.:22.75   1st Qu.:237.2
##  Median :0.01154   Median :0.8466   Median :26.33   Median :256.0
##  Mean   :0.01276   Mean   :0.8517   Mean   :32.30   Mean   :283.2
##  3rd Qu.:0.01377   3rd Qu.:0.8744   3rd Qu.:38.01   3rd Qu.:305.5
##  Max.   :0.01789   Max.   :0.8893   Max.   :56.54   Max.   :397.0
##
## mining info:
##    data ntransactions support confidence
##  transc         22190    0.01        0.8
```

View the summary of the rules.

```
summary(association.rules)
```

```
## set of 10 rules
##
## rule length distribution (lhs + rhs):sizes
## 2 3 4
## 3 5 2
##
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    2.00    2.25    3.00    2.90    3.00    4.00
##
## summary of quality measures:
##     support          confidence          lift           count
##  Min.   :0.01023   Min.   :0.8195   Min.   :21.90   Min.   :227.0
##  1st Qu.:0.01069   1st Qu.:0.8318   1st Qu.:22.75   1st Qu.:237.2
##  Median :0.01154   Median :0.8466   Median :26.33   Median :256.0
##  Mean   :0.01276   Mean   :0.8517   Mean   :32.30   Mean   :283.2
##  3rd Qu.:0.01377   3rd Qu.:0.8744   3rd Qu.:38.01   3rd Qu.:305.5
##  Max.   :0.01789   Max.   :0.8893   Max.   :56.54   Max.   :397.0
##
## mining info:
##    data ntransactions support confidence
##  transc         22190    0.01        0.8
```

We could also look at the first 10 association rules.

```
look at the first five rules
> inspect(association.rules[1:5])
   lhs                                    rhs
[1] {SET/6 RED SPOTTY PAPER CUPS}        => {SET/6 RED SPOTTY PAPER PLATES}
[2] {REGENCY TEA PLATE GREEN}            => {REGENCY TEA PLATE ROSES}
[3] {WOODEN TREE CHRISTMAS SCANDINAVIAN} => {WOODEN STAR CHRISTMAS SCANDINAVIAN}
[4] {GREEN REGENCY TEACUP AND SAUCER,
     PINK REGENCY TEACUP AND SAUCER}     => {ROSES REGENCY TEACUP AND SAUCER}
[5] {PINK REGENCY TEACUP AND SAUCER,
     ROSES REGENCY TEACUP AND SAUCER}    => {GREEN REGENCY TEACUP AND SAUCER}
```

```
inspect(association.rules[1:5])
```

```
##     lhs                                    rhs                                    support confiden
## [1] {SET/6 RED SPOTTY PAPER CUPS}        => {SET/6 RED SPOTTY PAPER PLATES}      0.01063542  0.82807
## [2] {REGENCY TEA PLATE GREEN}            => {REGENCY TEA PLATE ROSES}            0.01045516  0.84363
## [3] {WOODEN TREE CHRISTMAS SCANDINAVIAN} => {WOODEN STAR CHRISTMAS SCANDINAVIAN} 0.01022983  0.81949
## [4] {GREEN REGENCY TEACUP AND SAUCER,
##      PINK REGENCY TEACUP AND SAUCER}     => {ROSES REGENCY TEACUP AND SAUCER}    0.01789094  0.84288
## [5] {PINK REGENCY TEACUP AND SAUCER,
##      ROSES REGENCY TEACUP AND SAUCER}    => {GREEN REGENCY TEACUP AND SAUCER}    0.01789094  0.88026
```

Suppose we want to know what customers buy before buying WOODEN STAR CHRISTMAS SCANDINAVIAN

```
### what customers buy before buying WOODEN STAR CHRISTMAS SCANDINAVIAN
woodenR.association.rules=apriori(transc, parameter = list(supp=0.01, conf=0.8),
appearance = list(default="lhs",rhs="WOODEN STAR CHRISTMAS SCANDINAVIAN"))
```

```
## Apriori
##
## Parameter specification:
##  confidence minval smax arem  aval originalSupport maxtime support minlen
##         0.8    0.1    1 none FALSE            TRUE       5    0.01      1
##  maxlen target   ext
##      10  rules FALSE
##
```

```
## Algorithmic control:
##  filter tree heap memopt load sort verbose
##     0.1 TRUE TRUE  FALSE TRUE    2    TRUE
##
## Absolute minimum support count: 221
##
## set item appearances ...[1 item(s)] done [0.00s].
## set transactions ...[3882 item(s), 22190 transaction(s)] done [0.13s].
## sorting and recoding items ... [518 item(s)] done [0.01s].
## creating transaction tree ... done [0.01s].
## checking subsets of size 1 2 3 4 done [0.01s].
## writing ... [1 rule(s)] done [0.00s].
## creating S4 object  ... done [0.01s].
```

We can have a look at the rules. I deleted the last few columns due to page limit.

```
> inspect(woodenR.association.rules)
    lhs                                       rhs
[1] {WOODEN TREE CHRISTMAS SCANDINAVIAN} => {WOODEN STAR CHRISTMAS SCANDINAVIAN}
```

```
inspect(woodenR.association.rules)
```

```
##     lhs                                       rhs                                       support confiden
## [1] {WOODEN TREE CHRISTMAS SCANDINAVIAN} => {WOODEN STAR CHRISTMAS SCANDINAVIAN} 0.01022983  0.81949
```

We can also know what customers also bought if they bought SET/6 RED SPOTTY PAPER CUPS first

```
### customers bought SET/6 RED SPOTTY PAPER CUPS first also bought
redL.association.rules=apriori(transc, parameter = list(supp=0.01, conf=0.7),
appearance = list(default="rhs",lhs="SET/6 RED SPOTTY PAPER CUPS"))
```

```
## Apriori
##
## Parameter specification:
##  confidence minval smax arem  aval originalSupport maxtime support minlen
##         0.7    0.1    1 none FALSE            TRUE       5    0.01      1
##  maxlen target   ext
##      10  rules FALSE
##
## Algorithmic control:
##  filter tree heap memopt load sort verbose
##     0.1 TRUE TRUE  FALSE TRUE    2    TRUE
##
## Absolute minimum support count: 221
##
## set item appearances ...[1 item(s)] done [0.00s].
## set transactions ...[3882 item(s), 22190 transaction(s)] done [0.13s].
## sorting and recoding items ... [518 item(s)] done [0.01s].
## creating transaction tree ... done [0.01s].
## checking subsets of size 1 2 done [0.00s].
## writing ... [1 rule(s)] done [0.00s].
## creating S4 object  ... done [0.01s].
```

```
> inspect(redL.association.rules)
    lhs                              rhs
```

```
[1] {SET/6 RED SPOTTY PAPER CUPS} => {SET/6 RED SPOTTY PAPER PLATES}
```

```
inspect(redL.association.rules)
```

```
##     lhs                                rhs                                support confidence     lift c
## [1] {SET/6 RED SPOTTY PAPER CUPS} => {SET/6 RED SPOTTY PAPER PLATES} 0.01063542  0.8280702 56.53808
```

We can also get some nice graphs to visualise the rules. Here is an example of an interactive plot. You'll need the `arulesViz` package.

```
### get interactive plots
rules=head(association.rules, n = 10, by = "confidence")
plot(rules, method = "graph",  engine = "htmlwidget")
```

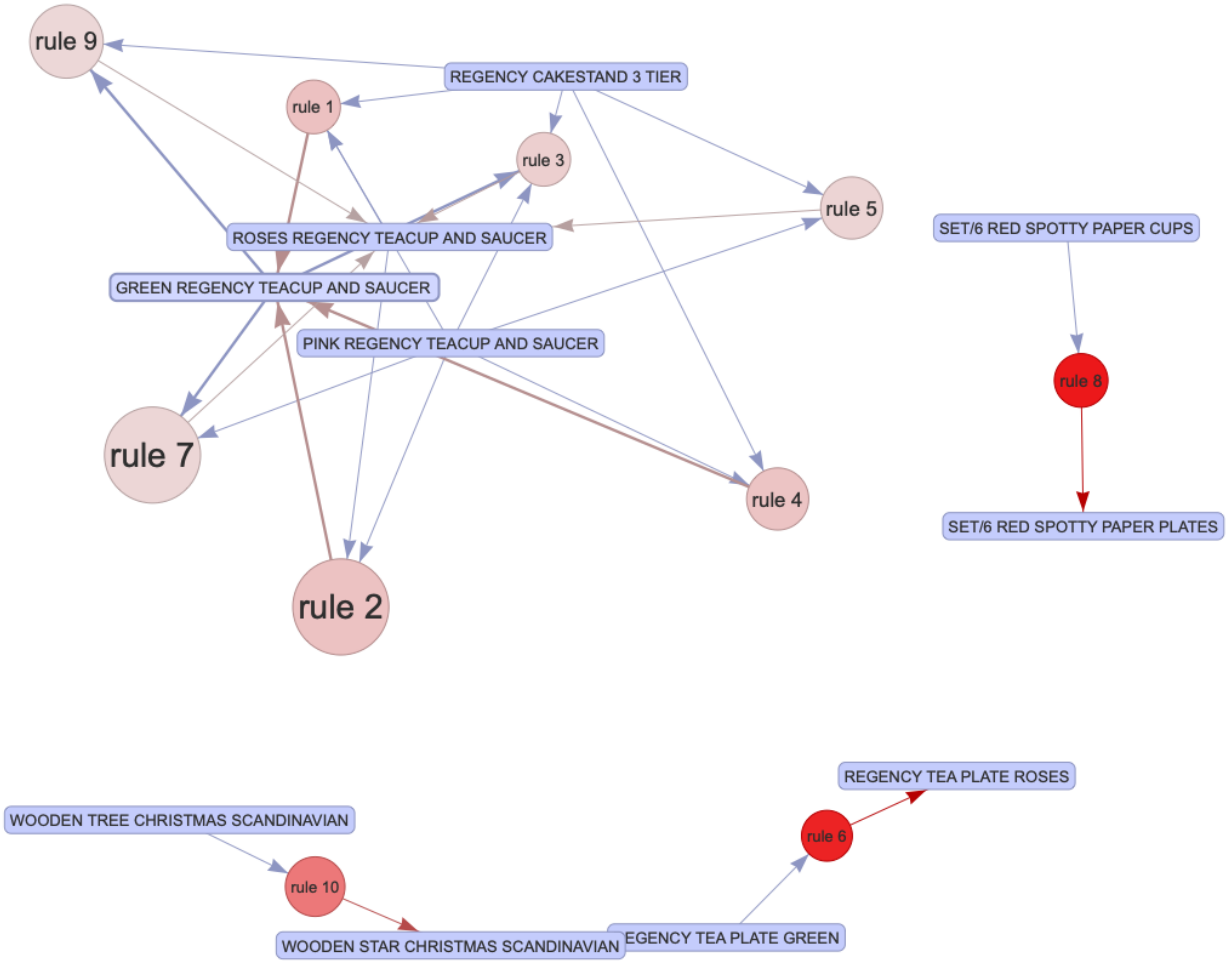

Figure 2: The interactive plot of the top ten association rules.

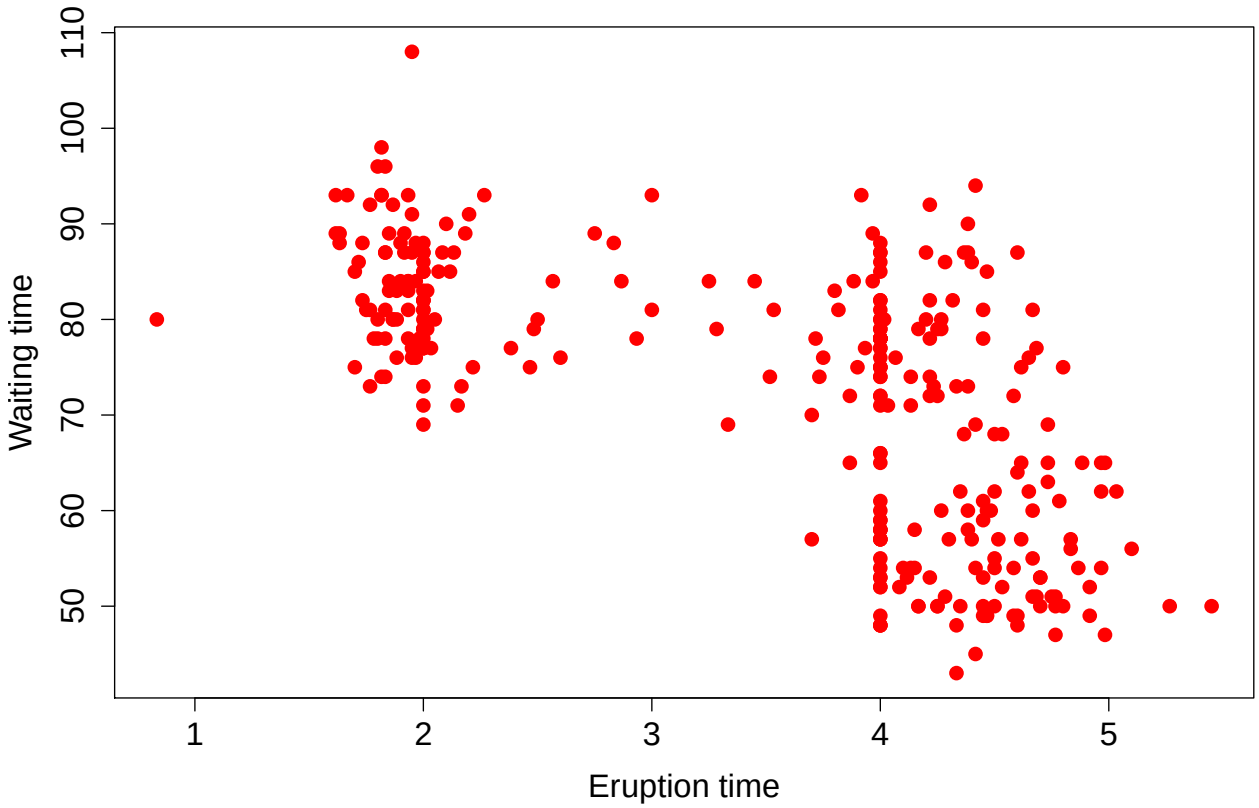

Figure 3: Model-based clustering.

2 Model-based clustering

The model-based clustering method can be performed by using the `Mclust` function in the `mclust` package. Besides the documentation, there's a nice tour of the package in https://cran.r-project.org/web/packages/mclust/vignettes/mclust.html, which includes some other functions that are useful in the package.

Here we focus on the model-based clustering and take the old faithful data as an example.

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

```
##
## Attaching package: 'MASS'
```

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

```
?geyser
str(geyser)
```

```
## 'data.frame':    299 obs. of  2 variables:
##  $ waiting : num  80 71 57 80 75 77 60 86 77 56 ...
##  $ duration: num  4.02 2.15 4 4 4 ...
```

```
plot(geyser[,2:1], col = "red", pch = 16,
     ylab = "Waiting time", xlab = "Eruption time",
     cex=1.5,cex.axis=1.5,cex.lab=1.5)
```

To use the model-based clustering method (mixture of Gaussians), we need to install the package first and then simply use the `Mclust` function. You could explore more of the input options in the documentation https://cran.r-project.org/web/packages/mclust/mclust.pdf.

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

```
## Package 'mclust' version 5.4.3
## Type 'citation("mclust")' for citing this R package in publications.
```

```
mod1=Mclust(geyser)
```

By using the `summary` function, we could obtain the summary of the model. In the `Mclust` function, the best model is chosen based on the BIC criterion: the larger the BIC, the better the model fitting. You need to note that the BIC used in this package is the negative of the usual usage in regression methods. So sometimes you can see 'the smaller the BIC, the better the model fitting' in other packages.

The `mclust` package also provides several choices for the covariance matrix: `E' for equal covariance,V'` for variable covariance and 'I' for coordinate axes. They use three letters to set the covariance matrices: the first refers to volume, the second to shape and the third to orientation. For example, in the following results, the best model has covariance matrices of VVI, meaning that the covariance matrices have varying volume and shape, and orientation equal to coordinate axes.

```
summary(mod1,parameters = TRUE)
```

```
## ----------------------------------------------------
## Gaussian finite mixture model fitted by EM algorithm
## ----------------------------------------------------
##
## Mclust VVI (diagonal, varying volume and shape) model with 4 components:
##
##  log-likelihood   n df       BIC       ICL
##       -1330.13 299 19 -2768.568 -2798.746
##
## Clustering table:
##  1  2  3  4
## 90 17 98 94
##
## Mixing probabilities:
##          1          2          3          4
## 0.29283469 0.07968364 0.33392296 0.29355871
##
## Means:
##               [,1]      [,2]      [,3]      [,4]
## waiting  78.245444 81.693185 55.045717 83.495263
## duration  4.124353  2.585291  4.443712  1.918517
##
## Variances:
## [,,1]
##           waiting   duration
## waiting  41.31804 0.00000000
## duration  0.00000 0.07151279
## [,,2]
##          waiting duration
## waiting  33.8808  0.00000
## duration  0.0000  0.44011
## [,,3]
```

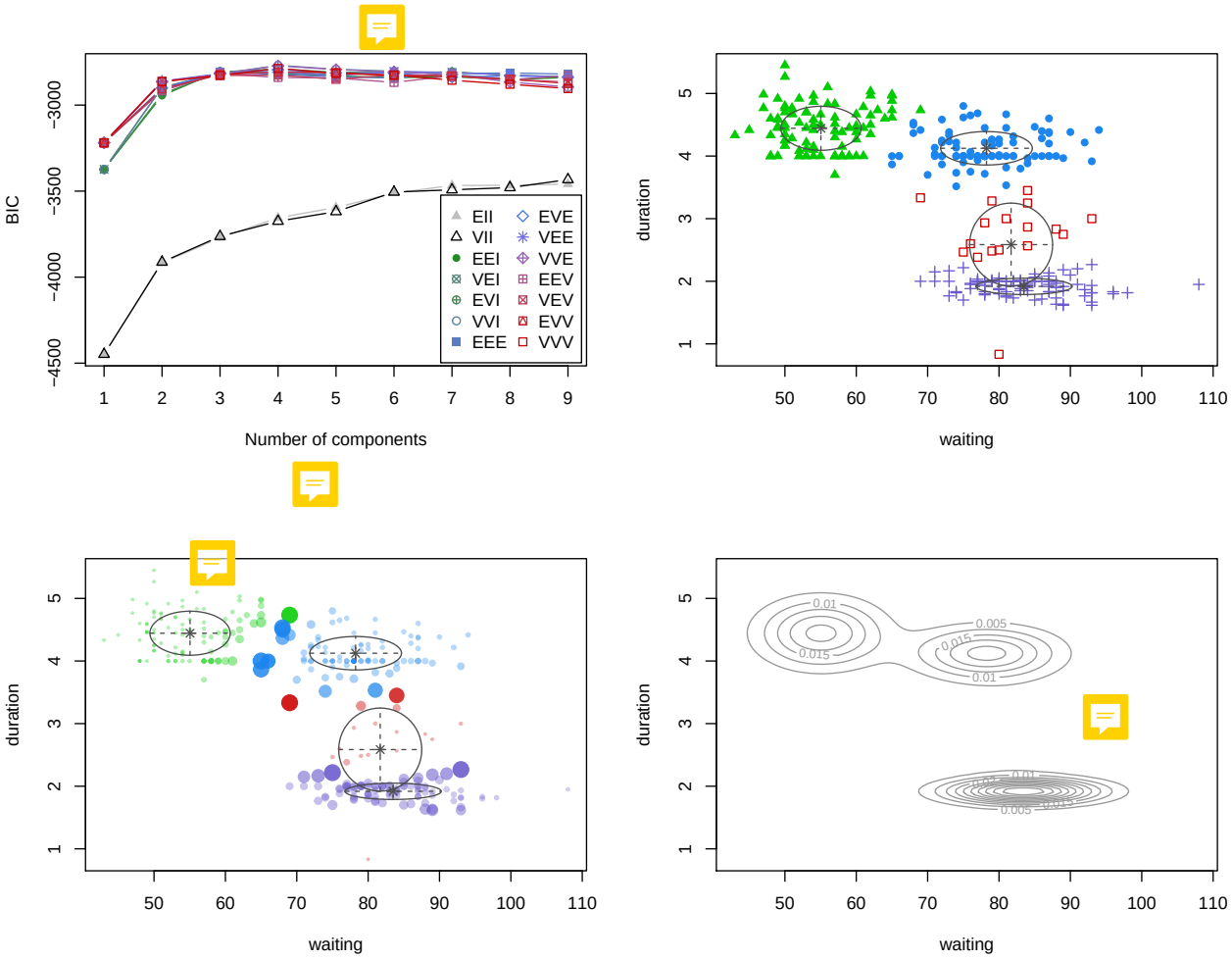

Figure 4: Four plots show you the BICs, the clustering results, the uncertainties of the clustering results and the estimated density of the data.

```
##           waiting  duration
## waiting  31.65528 0.0000000
## duration  0.00000 0.1239616
## [,,4]
##           waiting   duration
## waiting  45.44063 0.00000000
## duration  0.00000 0.01695964
```

It's also convenient to visualise the clustering results. Just use the `plot` function to choose which plot to show, or include which plot in the input option. The four plots show you the BICs, the clustering results, the uncertainties of the clustering results and the estimated density of the data.

```
#plot(mod1)
par(mfrow = c(2, 2))
plot(mod1,what="BIC")
plot(mod1,what="classification")
plot(mod1,what="uncertainty")
plot(mod1,what="density")
```