Wuhan University of technology
School of computer science and technology

WEB DATA MANAGEMENT
07/07/2015
LAB REPORT

Submitted by: OUMOUSS EL MEHDI
Student No: 2014Y90100063
TEL No: 13125029535
Email: oumoussmehdi@hotmail.com
Submitted to: Prof. Dr. Huazhu Song

DATA MINING LAB REPORT 1

(Wuhan University of Technology, Spring 2015, School of Computer Science and Technology)

DATA MINING – PREPROCESSING   APRIL 22ND, 2015

Data preprocessing describes any type of processing performed on raw data to prepare it for another processing procedure. Commonly used as a preliminary data mining practice, data preprocessing transforms the data into a format that will be more easily and effectively processed for the purpose of the user. There are a number of different methods used for preprocessing, including: sampling, which selects a representative subset from a large population of data; transformation, which manipulates raw data to produce a single input; denoising, which removes noise from data; normalization, which organizes data for more efficient access; and feature extraction, which pulls out specified data that is significant in some particular context.

***Note**: It would help for you to read through the contexts on preprocessing, understand its basic conceptions and operations, and solve some applications with it.*

PROBLEM STATEMENT

Please learn R by yourself, and make practice with R to finish the following tasks:

(1) Data **cleaning**, which removes noise and correct inconsistencies
(2) Data **integration**, which merges multiple data sources into a data store, i.e. data warehouse
(3) Data **reduction**, which reduces data size by i.e. aggregating, redundancy reduction, clustering, etc
(4) Data **transformation**, which scales data into a smaller range (i.e. normalization to 0 to 1)

Note: beside R, you could choose python, matlab or java, either of them is ok.

THE MAIN ALOGRITHMS

First of all, we should import data using read.csv function. After this, we should clean data, by removing noise and missing data and also correcting erroneous entries.
The second step, would be to apply several methods on our data, as clustering and regression, add to this corresponding plotting for visualization. Add to this we should calculate covariance and correlation too.
Finally, we will apply a transformation for the data which will be in a form of normalization 0-1 range.

DATA AND ANALYSIS

Note: results are shown all along this process.

heightWeight.txt:

	HeightInches	WeightPounds
1	65.78	112.99
2	71.52	136.49
3	69.40	153.03
4	68.22	142.34
5	67.79	144.30
6	68.70	123.30
7	69.80	141.49

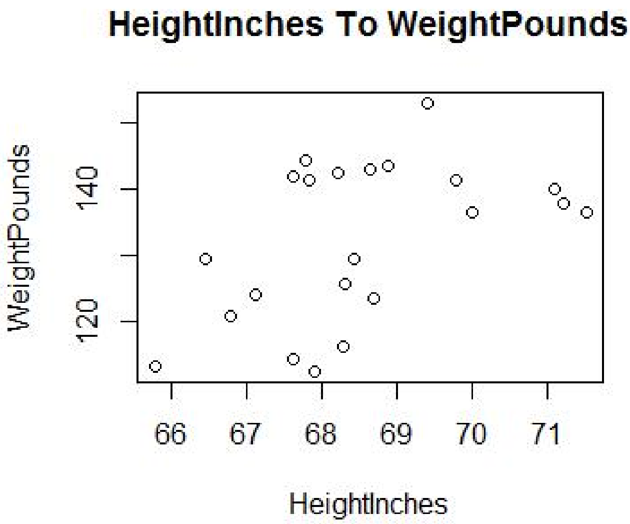
	HeightInches	WeightPounds
8	70.01	136.46
9	67.90	-112.37
10	66.78	120.67
11	66.49	NA
12	67.62	114.14
13	68.30	125.61
14	NA	122.46
15	68.28	116.09
16	71.09	140.00
17	66.46	129.50
18	68.65	142.97
19	71.23	137.90
20	-67.13	124.04
21	67.83	141.28
22	68.88	143.54
23	63.48	NA
24	68.42	129.50
25	67.63	141.85

```
> library(gdata)
> data=read.csv("heightWeight.txt",header=TRUE,sep=",")

#remove duplicated rows
> data=unique(data)

#remove rows with missing values
> data <- na.omit(data)
> rownames(data) <- NULL

#convert negative values in data$Height and data$Weight to positive ones
> data[,1] <- with(data, ifelse(data[,1] <0, abs(data[,1]),data[,1]))
> data[,2] <- with(data, ifelse(data[,2] <0, abs(data[,2]),data[,2]))

> View(data)
```

	HeightInches	WeightPounds
1	65.78	112.99
2	71.52	136.49
3	69.40	153.03
4	68.22	142.34
5	67.79	144.30
6	68.70	123.30
7	69.80	141.49
8	70.01	136.46
9	67.90	112.37
10	66.78	120.67

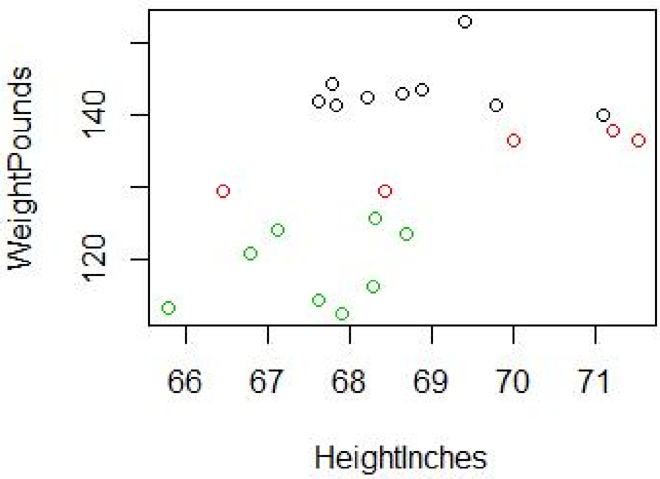
	HeightInches	WeightPounds
11	67.62	114.14
12	68.30	125.61
13	68.28	116.09
14	71.09	140.00
15	66.46	129.50
16	68.65	142.97
17	71.23	137.90
18	67.13	124.04
19	67.83	141.28
20	68.88	143.54
21	68.42	129.50
22	67.63	141.85

```
> plot(data[,1],data[,2],
xlab=colnames(data)[1],ylab=colnames(data)[2],
main=paste(colnames(data)[1],"To",colnames(data)[2]))
```




```
#clustering: kmeans
> km <- kmeans(data,3,15)
> print(km)
K-means clustering with 3 clusters of sizes 9, 5, 8

Cluster means:
  HeightInches WeightPounds
1     68.81000     143.4222
2     69.52800     133.9700
3     67.56125     118.6513

Clustering vector:
 [1] 3 2 1 1 1 3 1 2 3 3 3 3 3 1 2 1 2 3 1 1 2 1

Within cluster sum of squares by cluster:
[1] 127.27396  85.69468 208.01078
 (between_SS / total_SS =  86.2 %)
```

```
Available components:

[1] "cluster"      "centers"      "totss"        "withinss"
[5] "tot.withinss" "betweenss"    "size"         "iter"
[9] "ifault"

> # plot clusters
> plot(data, col = km$cluster)
```

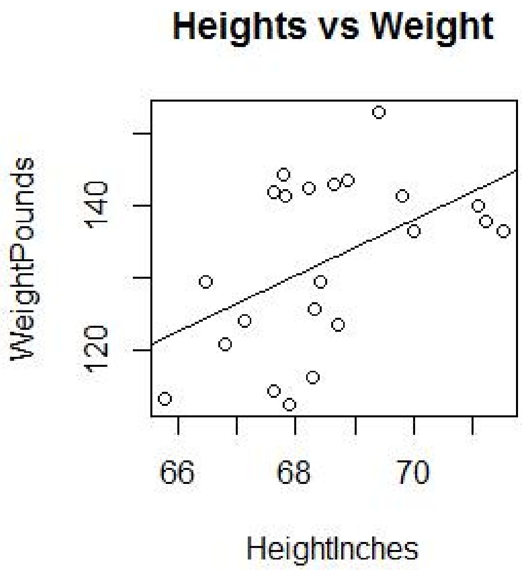



```
# get cluster means
> aggregate(data,by=list(km$cluster),FUN=mean)
  Group.1 HeightInches WeightPounds
1       1     67.56125     118.6513
2       2     68.81000     143.4222
3       3     69.52800     133.9700

# append cluster assignment
> datakm <- data.frame(data, km$cluster)
> view(datakm)
```

	HeightInches	WeightPounds	km.cluster
1	65.78	112.99	1
2	71.52	136.49	3
3	69.40	153.03	2
4	68.22	142.34	2
5	67.79	144.30	2
6	68.70	123.30	1
7	69.80	141.49	2
8	70.01	136.46	3
9	67.90	112.37	1
10	66.78	120.67	1

	HeightInches	WeightPounds	km.cluster
11	67.62	114.14	1
12	68.30	125.61	1
13	68.28	116.09	1
14	71.09	140.00	2
15	66.46	129.50	3
16	68.65	142.97	2
17	71.23	137.90	3
18	67.13	124.04	1
19	67.83	141.28	2
20	68.88	143.54	2
21	68.42	129.50	3
22	67.63	141.85	2

```
#Linear model:

> lm <-lm(WeightPounds ~ HeightInches, datakm)
> abline(lm)
```



```
#Covariance and Correlation:

> cov(datakm)
             HeightInches WeightPounds km.cluster
HeightInches    2.2701515     8.848663  0.6051082
WeightPounds    8.8486632   143.050386  5.5923377
km.cluster      0.6051082     5.592338  0.5995671
> cor(datakm)
             HeightInches WeightPounds km.cluster
HeightInches    1.0000000    0.4910274  0.5186648
WeightPounds    0.4910274    1.0000000  0.6038513
km.cluster      0.5186648    0.6038513  1.0000000

> cov(datakm$HeightInches,datakm$WeightPounds)
```

```
[1] 8.848663
> cor(datakm$HeightInches,datakm$WeightPounds)
[1] 0.4910274
```

#Summary:

```
> summary(datakm)
  HeightInches    WeightPounds      km.cluster
 Min.   :65.78   Min.   :112.4   Min.   :1.000
 1st Qu.:67.67   1st Qu.:123.5   1st Qu.:1.000
 Median :68.29   Median :136.5   Median :2.000
 Mean   :68.52   Mean   :132.3   Mean   :1.864
 3rd Qu.:69.27   3rd Qu.:141.8   3rd Qu.:2.000
 Max.   :71.52   Max.   :153.0   Max.   :3.000
```

#Data normalization:

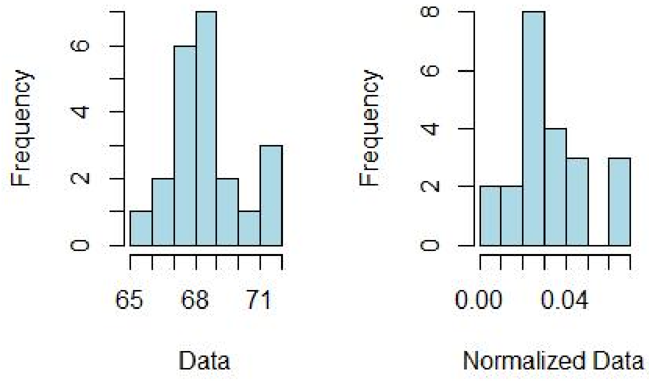
```
#our Data
d = data
#Normalized Data
n = (d-min(d))/(max(d)-min(d))
View(n)
```

	HeightInches	WeightPounds
1	0.000000000	0.5410888
2	0.065787966	0.8104298
3	0.041489971	1.0000000
4	0.027965616	0.8774785
5	0.023037249	0.8999427
6	0.033467049	0.6592550
7	0.046074499	0.8677364
8	0.048481375	0.8100860
9	0.024297994	0.5339828
10	0.011461318	0.6291117
11	0.021088825	0.5542693
12	0.028882521	0.6857307
13	0.028653295	0.5766189
14	0.060859599	0.8506590
15	0.007793696	0.7303152
16	0.032893983	0.8846991
17	0.062464183	0.8265903
18	0.015472779	0.6677364
19	0.023495702	0.8653295
20	0.035530086	0.8912321
21	0.030257880	0.7303152
22	0.021203438	0.8718625

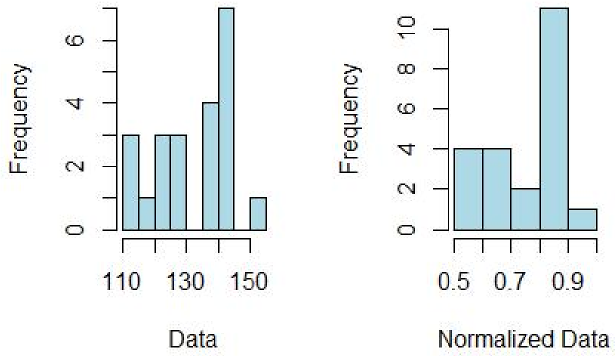
#Histogram of example data and normalized data

Histogram for data$HeightInches

```
hist(d[,1],xlab="Data",col="lightblue",main="")
hist(n[,1],xlab="Normalized Data",col="lightblue",main="")
```




```
                                   # Histogram for data$WeightPounds
hist(d[,2],xlab="Data",col="lightblue",main="")
hist(n[,2],xlab="Normalized Data",col="lightblue",main="")
```



Conclusions and reflection

Preprocessing is based on different methods including: sampling, transformation, denoising and normalization. All this techniques are gathered in order to have a trust worthy and clean data on which we can base our study and analyzing when doing data mining in a research or commercial envirement.

Reference

[1] Data Mining class material (ppts)
[2] Jiawei Han, et. al. *DATA MINING Concepts and Techniques*
[3] An Introduction to R, http://cran.r-project.org/doc/contrib/usingR.pdf
[4] A Community Site for R, http://www.inside-r.org
[5] Edwin de Jonge, Mark van der Loo, *An introduction to data cleaning with R*
[6] Stackoverflow, http://stackoverflow.com

Program source code

(You may used the attached source code to label and to refer to in your report.)

DATA MINING LAB REPORT 2

(Wuhan University of Technology, Spring 2015, School of Computer Science and Technology)

DATA MINING – APRIORI ALGORITHM   MAY 13TH , 2015

Apriori pruning principle: If there is any itemset which is infrequent, its superset should not be generated/tested!
Method:

(1) Initially, scan DB once to get frequent 1-itemset
(2) Generate length (k+1) candidate itemsets from length k frequent itemsets
(3) Test the candidates against DB
(4) Terminate when no frequent or candidate set can be generated

***Note**: please consider the advantage and disadvantages about this algorithm*

PROBLEM STATEMENT

Please coding the apriori algorithm.

THE MAIN ALOGRITHMS

Apriori (T, ε)

$L_1 \leftarrow \{$ *large 1-itemsets that appear in more than* ε *transactions* $\}$

$k \leftarrow 2$

while $L_{k-1} \neq \varnothing$

$C_k \leftarrow$ Generate(L_{k-1})

for transactions $t \in T$

$C_t \leftarrow$ Subset(C_k, t)

for candidates $c \in C_t$

$\mathrm{count}[c] \leftarrow \mathrm{count}[c] + 1$

$L_k \leftarrow \{c \in C_k | \ \mathrm{count}[c] \geq \varepsilon\}$

$k \leftarrow k + 1$

return $\bigcup_k L_k$

DATA, ANALYSIS AND RESULTS

Basket.txt:

Banana,Jus,Bread
Jus,Bread
Banana,Jus,Chips
Banana,Jus,Bread,Chips
Banana
Jus

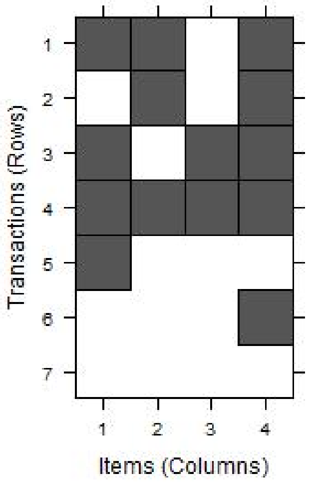
```
> library("arules");
> library("arulesViz");
```

```
> tr<-read.transactions("Basket.txt",format="basket",sep=",")
```

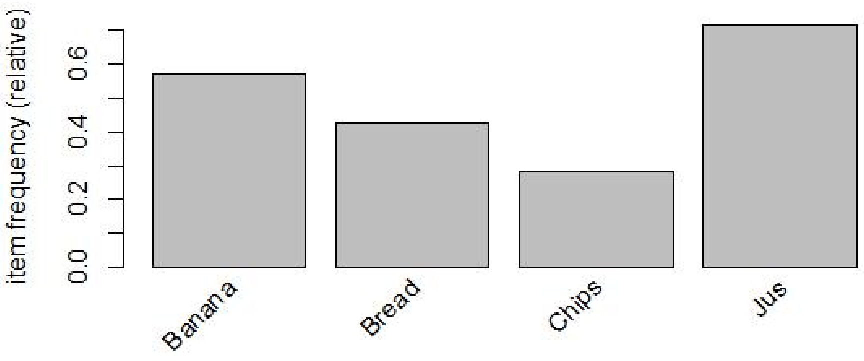
```
> inspect(tr)
```

```
  items
1 {Banana,
   Bread,
   Jus}
2 {Bread,
   Jus}
3 {Banana,
   Chips,
   Jus}
4 {Banana,
   Bread,
   Chips,
   Jus}
5 {Banana}
6 {Jus}
7 {}
```

```
> image(tr)
```




```
> itemFrequencyPlot(tr, support = 0.1)
```



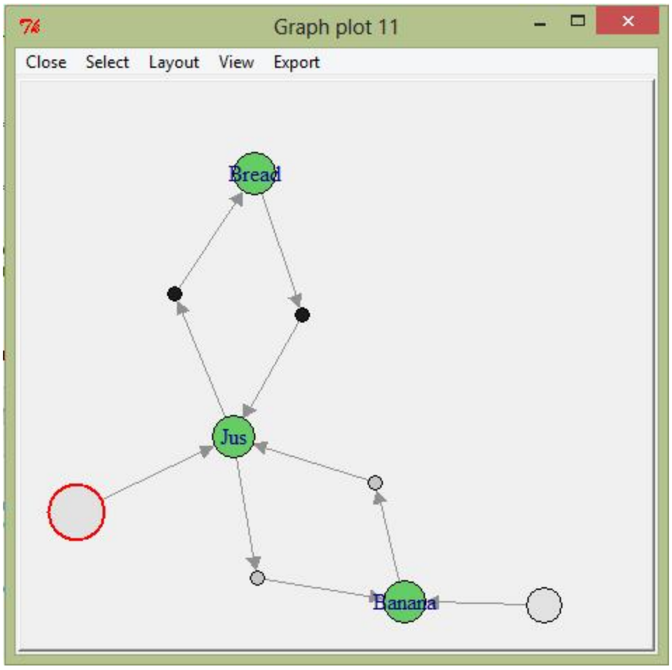

```
> length(tr)
[1] 7
```

```
# Mine association rules.
> rules <- apriori(tr, parameter= list(supp=0.5, conf=0.5))
```

```
Parameter specification:
 confidence minval smax arem  aval originalSupport support minlen maxlen target   ext
        0.5    0.1    1 none FALSE            TRUE     0.5      1     10  rules FALSE
```

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

apriori - find association rules with the apriori algorithm
version 4.21 (2004.05.09)        (c) 1996-2004   Christian Borgelt
set item appearances ...[0 item(s)] done [0.00s].
set transactions ...[4 item(s), 7 transaction(s)] done [0.00s].
sorting and recoding items ... [2 item(s)] done [0.00s].
creating transaction tree ... done [0.00s].
checking subsets of size 1 2 done [0.00s].
writing ... [2 rule(s)] done [0.00s].
creating S4 object  ... done [0.00s].
> inspect(rules)
  lhs    rhs        support confidence lift
1 {}  => {Banana} 0.5714286  0.5714286    1
2 {}  => {Jus}    0.7142857  0.7142857    1
> rules <- apriori(tr, parameter= list(supp=0.4, conf=0.5))

Parameter specification:
 confidence minval smax arem  aval originalSupport support minlen maxlen target   ext
        0.5    0.1    1 none FALSE            TRUE     0.4      1     10  rules FALSE

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

apriori - find association rules with the apriori algorithm
version 4.21 (2004.05.09)        (c) 1996-2004   Christian Borgelt
set item appearances ...[0 item(s)] done [0.00s].
set transactions ...[4 item(s), 7 transaction(s)] done [0.00s].
sorting and recoding items ... [3 item(s)] done [0.00s].
creating transaction tree ... done [0.00s].
checking subsets of size 1 2 done [0.00s].
writing ... [6 rule(s)] done [0.00s].
creating S4 object  ... done [0.00s].

# Check the generated rules using inspect
> inspect(rules)
  lhs         rhs        support confidence lift
1 {}       => {Banana} 0.5714286  0.5714286 1.00
2 {}       => {Jus}    0.7142857  0.7142857 1.00
3 {Bread}  => {Jus}    0.4285714  1.0000000 1.40
4 {Jus}    => {Bread}  0.4285714  0.6000000 1.40
5 {Banana} => {Jus}    0.4285714  0.7500000 1.05
6 {Jus}    => {Banana} 0.4285714  0.6000000 1.05

#If huge number of rules are generated specific rules can read using index
> inspect(rules[1]);

  lhs    rhs        support confidence lift
1 {}  => {Banana} 0.5714286  0.5714286    1


> summary(rules)
set of 6 rules

rule length distribution (lhs + rhs):sizes
1 2 
2 4 

   Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
  1.000   1.250   2.000   1.667   2.000   2.000 

summary of quality measures:
    support         confidence          lift      
 Min.   :0.4286   Min.   :0.5714   Min.   :1.000  
 1st Qu.:0.4286   1st Qu.:0.6000   1st Qu.:1.012  
 Median :0.4286   Median :0.6571   Median :1.050  
 Mean   :0.5000   Mean   :0.7060   Mean   :1.150  
 3rd Qu.:0.5357   3rd Qu.:0.7411   3rd Qu.:1.312  
 Max.   :0.7143   Max.   :1.0000   Max.   :1.400  

mining info:
 data ntransactions support confidence
   tr             7     0.4        0.5
```

```
> interestMeasure(rules, c("support", "chiSquare", "confidence", "conviction",
+ "cosine", "coverage", "leverage", "lift", "oddsRatio"), tr)

    support chiSquared confidence conviction    cosine  coverage   leverage lift     oddsRatio
1 0.5714286         NA  0.5714286   1.000000 0.7559289 1.0000000 0.00000000 1.00            NA
2 0.7142857         NA  0.7142857   1.000000 0.8451543 1.0000000 0.00000000 1.00            NA
3 0.4285714 2.10000000  1.0000000         NA 0.7745967 0.4285714 0.12244898 1.40            NA
4 0.4285714 2.10000000  0.6000000   1.428571 0.7745967 0.7142857 0.12244898 1.40 -6.755399e+15
5 0.4285714 0.05833333  0.7500000   1.142857 0.6708204 0.5714286 0.02040816 1.05  1.500000e+00
6 0.4285714 0.05833333  0.6000000   1.071429 0.6708204 0.7142857 0.02040816 1.05  1.500000e+00

> plot(rules,method="graph",interactive=TRUE)
```



Conclusions and Reflection

Apriori is an algorithm for frequent item set mining and association rule learning. The frequent item sets determined by Apriori can be used to determine association rules which highlight general trends in the database. This is why, It is often used by grocery stores, retailers, and anyone with a large transactional databases.
Association rules use the R *arules* library.
The *arulesViz* add additional features for graphing and plotting the rules.

Reference

[1] Data Mining Algorithms In R/Frequent Pattern Mining/The Apriori Algorithm, https://en.wikibooks.org/wiki/Data_Mining_Algorithms_In_R/Frequent_Pattern_Mining/The_Apriori_Algorithm
[2] The Apriori Algorithm ... How The Apriori Algorithm Works, https://www.youtube.com/watch?v=Hk1zFOMLTrw
[3] Package 'arules', http://cran.r-project.org/web/packages/arules/arules.pdf
[4] Market Basket Analysis with R, http://www.salemmarafi.com/code/market-basket-analysis-with-r/comment-page-1/
[5] Association Rules and Market Basket Analysis with R, http://blog.revolutionanalytics.com/2015/04/association-rules-and-market-basket-analysis-with-r.html

Program Source Code

(You may used the attached source code to label and to refer to in your report.)

DATA MINING LAB REPORT 3

(Wuhan University of Technology, Spring 2015, School of Computer Science and Technology)

DATA MINING – CLUSTERING ALGORITHM   MAY 27TH , 2015

The main alogrithms

Procedure of K-means Algorithm:

(1) Distribute all objects to K number of different cluster at random;
(2) Calculate the mean value of each cluster, and use this mean value to represent the cluster;
(3) Re-distribute the objects to the closest cluster according to its distance to the cluster center;
(4) Update the mean value of the cluster. That is to say, calculate the mean value of the objects in each cluster;
(5) Calculate the criterion function E, until the criterion function converges.

Note*: please consider the advantage and disadvantages about this algorithm*

Problem statement

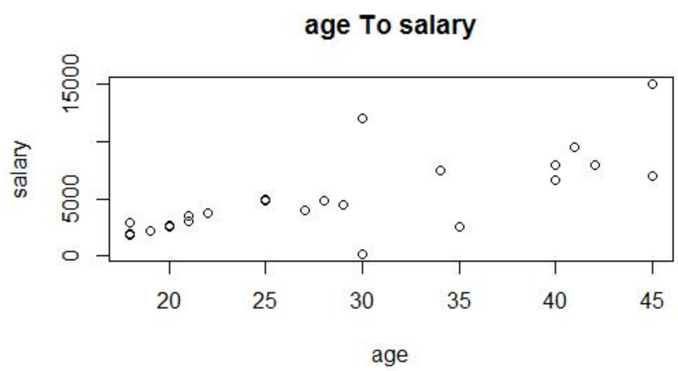
Please coding the K-means algorithm.

Data, Analysis and Results

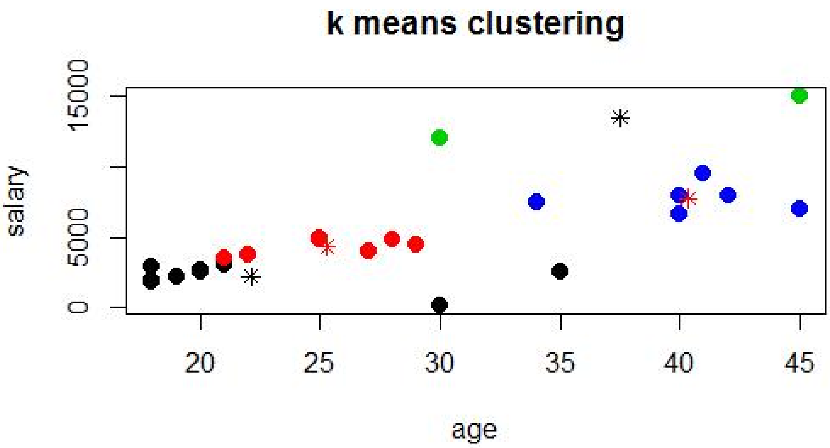
AgeToSalary.txt:

	age	salary
1	18	1800
2	18	2000
3	18	2900
4	19	2200
5	20	2500
6	20	2700
7	21	3000
8	21	3500
9	22	3800
10	25	4800
11	25	5000
12	27	4000
13	28	4800
14	29	4500
15	30	200
16	30	12000
17	34	7500
18	35	2600
19	40	6600
20	40	8000
21	41	9500
22	42	8000
23	45	15000
24	45	7000

```
> library(gdata)
> data=read.csv("AgeToSalary.txt",header=TRUE,sep=",")
> plot(data[,1],data[,2],xlab=colnames(data)[1] ,ylab=colnames(data)[2],main=paste(c
olnames(data)[1],"To",colnames(data)[2]))
```



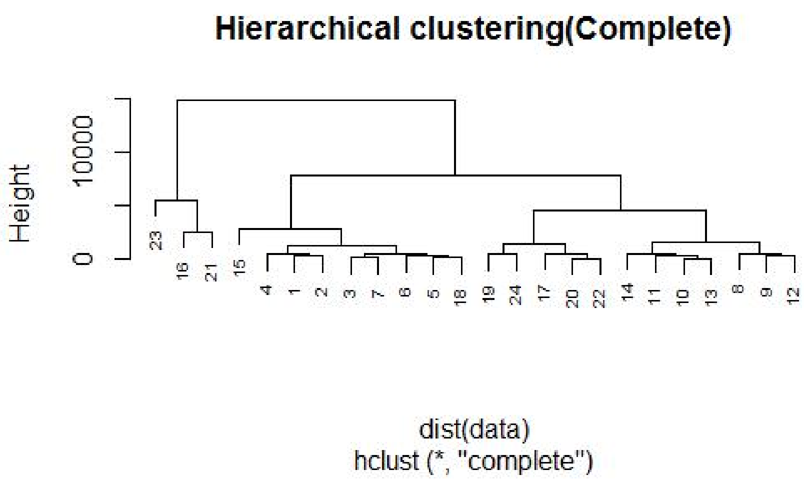
```
# run K-Means
> km <- kmeans(data, 4, 15)
# print components of km
> print(km)
K-means clustering with 4 clusters of sizes 9, 7, 2, 6

Cluster means:
       age    salary
1 22.11111  2211.111
2 25.28571  4342.857
3 37.50000 13500.000
4 40.33333  7766.667

Clustering vector:
 [1] 1 1 1 1 1 1 1 2 2 2 2 2 2 2 1 3 4 1 4 4 4 4 3 4

Within cluster sum of squares by cluster:
[1] 5829188 1997196 4500113 5133399
 (between_SS / total_SS =  93.8 %)

Available components:

[1] "cluster"      "centers"      "totss"        "withinss"
[5] "tot.withinss" "betweenss"    "size"         "iter"
[9] "ifault"
> km$cluster
 [1] 1 1 1 1 1 1 1 2 2 2 2 2 2 2 1 3 4 1 4 4 4 4 3 4
> km$centers
       age    salary
1 22.11111  2211.111
2 25.28571  4342.857
3 37.50000 13500.000
4 40.33333  7766.667
# plot clusters
> plot(data, col = km$cluster,main="k means clustering",pch=20,cex=2)
# plot centers
> points(km$centers, col = 1:2, pch = 8)
```



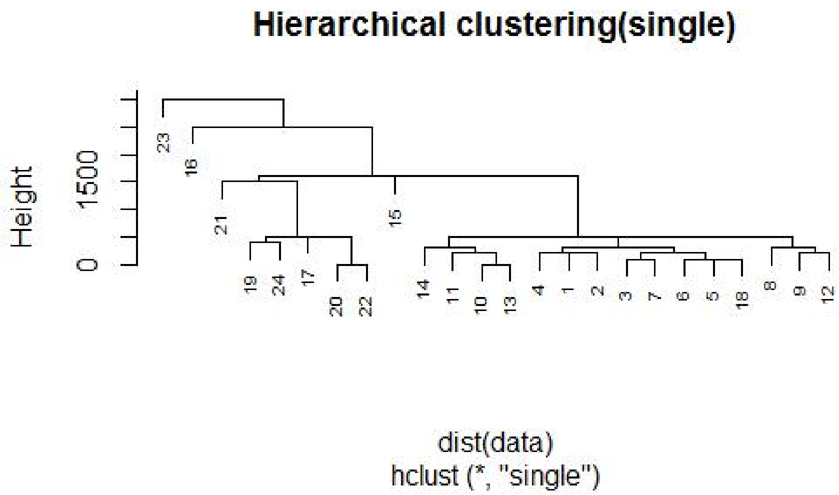
```
> # append cluster assignment
> datakm <- data.frame(data, km$cluster)
> View(datakm)
```

	age	salary	km.cluster
1	18	1800	1
2	18	2000	1
3	18	2900	1
4	19	2200	1
5	20	2500	1
6	20	2700	1
7	21	3000	1
8	21	3500	2
9	22	3800	2
10	25	4800	2
11	25	5000	2
12	27	4000	2
13	28	4800	2
14	29	4500	2
15	30	200	1
16	30	12000	3
17	34	7500	4
18	35	2600	1
19	40	6600	4
20	40	8000	4
21	41	9500	4
22	42	8000	4
23	45	15000	3
24	45	7000	4

```
#Hierarchical clustering:

> hc.complete<-hclust(dist(data),method="complete")
> hc.single<-hclust(dist(data),method="single")

> plot(hc.complete,main="Hierarchical clustering(Complete)",cex=0.6)
```



```
> plot(hc.single,main="Hierarchical clustering(Complete)",cex=0.6)
```



Conclusions and Reflection

K-means is one of the simplest unsupervised learning algorithms which is popular for cluster analysis in data mining. This made k-means one of the standards algorithms when it comes to clustering and it is widely used.

Reference

[1] K-Means Clustering, https://stat.ethz.ch/R-manual/R-devel/library/stats/html/kmeans.html

[2]Cluster Analysis, http://www.statmethods.net/advstats/cluster.html

[3] k-means Clustering, http://www.rdatamining.com/examples/kmeans-clustering

[4] K Means/ Hierarchical Clustering in R, https://www.youtube.com/watch?v=M9jb6KrBlPc

[5] How to Perform K-Means Clustering in R Statistical Computing, https://www.youtube.com/watch?v=sAtnX3UJyN0

Program Source Code

(You may used the attached source code to label and to refer to in your report.)