로또 데이터 분석

빅데이터 경영 MBA
U2016054 이병준

1 Lotte DATA

Library 호출 및 Data Read

Load Library & Data

In [70]:
```
library(arules)
library(reshape2)
```

In [2]:
```
lotte <- read.csv("lotte.csv")
lotte <- lotte[,-8]
```

In [3]:
```
colnames(lotte) <- c("seq","N1","N2","N3","N4","N5","N6")
```

회차 별 뽑힌 숫자.

- 보너스 숫자 제외

In [4]:
```
head(lotte[order(lotte$seq, decreasing = F),])
```

	seq	N1	N2	N3	N4	N5	N6
730	1	10	23	29	33	37	40
729	2	9	13	21	25	32	42
728	3	11	16	19	21	27	31
727	4	14	27	30	31	40	42
726	5	16	24	29	40	41	42
725	6	14	15	26	27	40	42

2 Data Reshaping

Transacition Data 형성을 위해 Reshape
- Melt 활용. (Seq 기준.)

Data Reshaping

- *Transacition Data* 형성을 위해 *Reshape* (*Melt*를 *seq*를 기준으로 수행)

In [5]:
```
melt_lotte <- melt(lotte, id="seq") # seq(회차)를 기준으로 데이터 Melt
```

In [6]:
```
melt_lotte[melt_lotte$seq ==730,] # 확인.
```

	seq	variable	value
1	730	N1	4
731	730	N2	10
1461	730	N3	14
2191	730	N4	15
2921	730	N5	18
3651	730	N6	22

Pick data from DF

In [7]:
```
data <- melt_lotte[,c(1,3)] # seq, value
```

In [8]:
```
head(data[order(data$seq,decreasing = T),])
```

	seq	value
1	730	4
731	730	10
1461	730	14
2191	730	15
2921	730	18
3651	730	22

3 - SplitData with Seq number
- Make Transactions

Value(회차별 나온 숫자) 를 Seq를 기준으로 Split

Split Data with seq number ¶

- *Value*를 *Seq*로 *Split*한다.

In [9]:
```
head(split(data$value, data$seq))
```

```
$`1`
    10  23  29  33  37  40
$`2`
    9  13  21  25  32  42
$`3`
    11  16  19  21  27  31
$`4`
    14  27  30  31  40  42
$`5`
    16  24  29  40  41  42
$`6`
    14  15  26  27  40  42
```

Make Transactions

In [10]:
```
trans <- as(split(data$value, data$seq), "transactions") #transactions
trans
```

```
transactions in sparse format with
 730 transactions (rows) and
 45 items (columns)
```

Inspect a transaction

- 각 회차별 나온 숫자를 Transaction으로 변형

In [11]:
```
inspect(trans[1:10])
```

```
     items                transactionID
[1]  {10,23,29,33,37,40}  1
[2]  {9,13,21,25,32,42}   2
[3]  {11,16,19,21,27,31}  3
[4]  {14,27,30,31,40,42}  4
[5]  {16,24,29,40,41,42}  5
[6]  {14,15,26,27,40,42}  6
[7]  {2,9,16,25,26,40}    7
[8]  {8,19,25,34,37,39}   8
[9]  {2,4,16,17,36,39}    9
[10] {9,25,30,33,41,44}   10
```

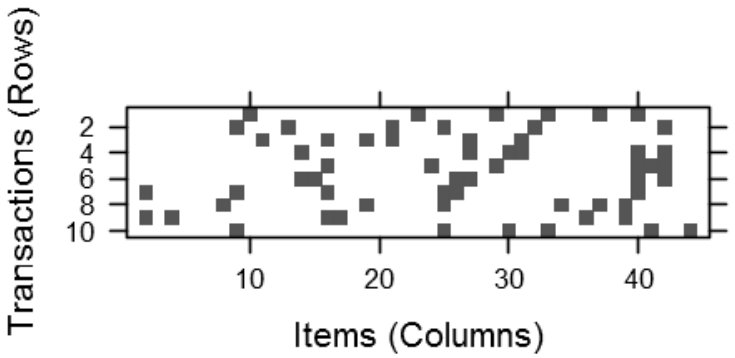
4 Inspect Data

- Image
- Check Frequency
- Posibility of items

Image of Transaction

- 1 ~ 45 까지 숫자 중 각 회차별 새당 되는 숫자에 색이 칠해진다.

In [12]:
```
options(repr.plot.width=4,repr.plot.height=2)
image(trans[1:10])
```



Check Frequency of items

- 각 회차에 나온 개별의 숫자들의 빈발 정도를 확인 (absolute : Counts of values)

In [13]:
```
t(itemFrequency(trans, type="absolute"))
```

1	2	3	4	5	6	7	8	9	10	...	36	37	38	39	40	41	42	43	44	45
109	95	96	105	102	91	100	104	72	99	...	96	108	91	99	114	86	87	101	100	98

Check Posibility of items

- 각 회차에 나온 개별 숫자들의 나온 확률을 확인

In [14]:
```
t(round(itemFrequency(trans)[order(itemFrequency(trans), decreasing = TRUE)],2))
```

20	40	34	27	1	37	4	14	17	8	...	21	23	16	30	42	41	28	32	22	9
0.16	0.16	0.15	0.15	0.15	0.15	0.14	0.14	0.14	0.14	...	0.12	0.12	0.12	0.12	0.12	0.12	0.12	0.12	0.11	0.1

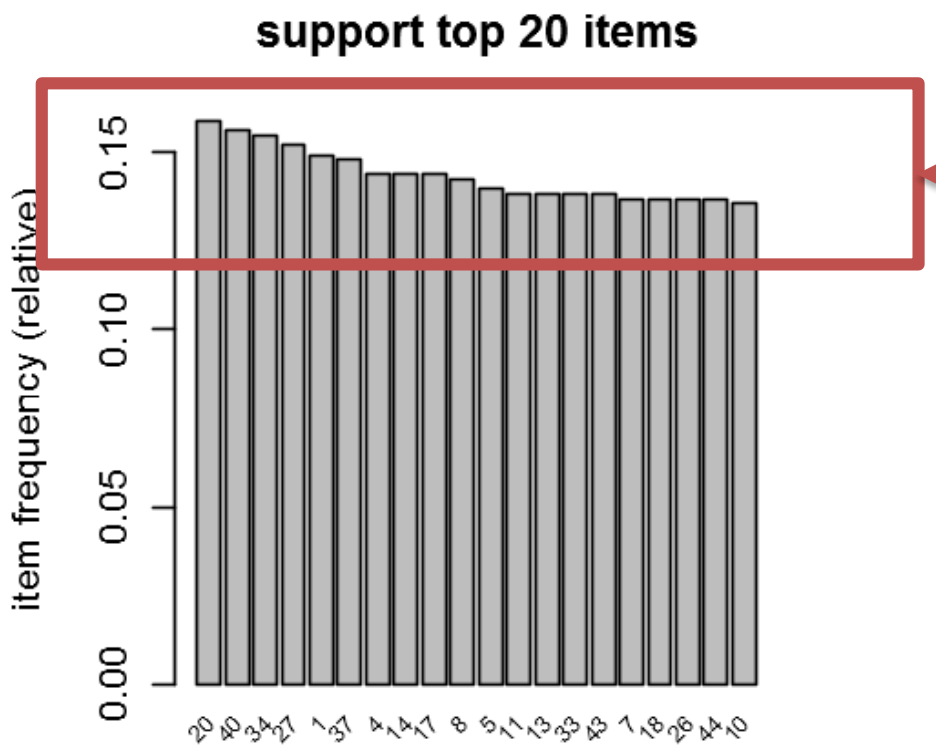
4 Inspect Data

- Image
- Check Frequency
- Posibility of items

Plotting items with support

- 상위 20개 지지도를 가진 Items를 Plotting

In [15]:
```
options(repr.plot.width=4,repr.plot.height=4)
itemFrequencyPlot(trans, topN = 20, main = "support top 20 items",cex.names=0.6)
```



확률적으로 차이가 나지 않는다.

5 Making Rules

Making rules with transaction data, Lotte

- 최소 지지도를 넘는 빈발 집합을 출력.
- Transaction ID 는 필요 없으므로 제외

In [16]:
```
rules <- apriori(trans[,-2],parameter = list(support=0.005,target="frequent itemsets"))
```

```
Apriori

Parameter specification:
 confidence minval smax arem  aval originalSupport maxtime support minlen
         NA    0.1    1 none FALSE            TRUE       5   0.005      1
 maxlen            target   ext
     10 frequent itemsets FALSE

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

Absolute minimum support count: 3

set item appearances ...[0 item(s)] done [0.00s].
set transactions ...[44 item(s), 730 transaction(s)] done [0.00s].
sorting and recoding items ... [44 item(s)] done [0.00s].
creating transaction tree ... done [0.00s].
checking subsets of size 1 2 3 done [0.00s].
writing ... [1241 set(s)] done [0.00s].
creating S4 object  ... done [0.00s].
```

5 Making Rules

```
In [17]: summary(rules)
```

```
set of 1241 itemsets

most frequent items:
     40      20      27       7      35 (Other)
     76      74      70      68      68    2338

element (itemset/transaction) length distribution:sizes
  1   2   3
 44 941 256

   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  1.000   2.000   2.000   2.171   2.000   3.000

summary of quality measures:
    support
 Min.   :0.005479
 1st Qu.:0.008219
 Median :0.013699
 Mean   :0.017469
 3rd Qu.:0.017808
 Max.   :0.158904

includes transaction ID lists: FALSE

mining info:
        data ntransactions support confidence
 trans[, -2]          730   0.005          1
```

{1개짜리}	{2개짜리}	{3개짜리}
44개	941개	256개

5 Making Rules 2

– Get 4 set Rules

4개 이상의 조합을 얻기 위한 지지도 하향 조정

In [45]:
```
rules2 <- apriori(trans[,-2],parameter = list(support=0.0005,target="frequent itemsets"))
summary(rules2)
```

```
Apriori

Parameter specification:
 confidence minval smax arem  aval originalSupport maxtime support minlen
         NA    0.1    1 none FALSE            TRUE       5   5e-04      1
 maxlen            target   ext
     10 frequent itemsets FALSE

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

Absolute minimum support count: 0

set item appearances ...[0 item(s)] done [0.00s].
set transactions ...[44 item(s), 730 transaction(s)] done [0.00s].
sorting and recoding items ... [44 item(s)] done [0.00s].
creating transaction tree ... done [0.00s].
checking subsets of size 1 2 3 4 5 6 done [0.01s].
writing ... [23745 set(s)] done [0.01s].
creating S4 object  ... done [0.01s].

set of 23745 itemsets

most frequent items:
     40      20      27      34      37 (Other)
   2360    2358    2275    2269    2261   78058

element (itemset/transaction) length distribution:sizes
   1    2    3    4    5    6
  44  946 8545 9675 3900  635

   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  1.000   3.000   4.000   3.773   4.000   6.000

summary of quality measures:
    support
 Min.   :0.001370
 1st Qu.:0.001370
 Median :0.001370
 Mean   :0.002478
 3rd Qu.:0.001370
 Max.   :0.158904

includes transaction ID lists: FALSE

mining info:
        data ntransactions support confidence
 trans[, -2]           730   5e-04          1
```

6 Inspect Rules

- 1 Set
- 2 Set
- 3 Set

Top 10 of the Support

```
In [19]: inspect(sort(rules, by = "support")[1:10])
```

```
     items support
[1]  {20}  0.1589041
[2]  {40}  0.1561644
[3]  {34}  0.1547945
[4]  {27}  0.1520548
[5]  {1}   0.1493151
[6]  {37}  0.1479452
[7]  {17}  0.1438356
[8]  {4}   0.1438356
```

2 Set of Rules

```
In [73]: inspect(sort(rules, by = "support")[43:50])
```

```
     items   support
[1]  {22}    0.11232877
[2]  {9}     0.09863014
[3]  {20,35} 0.02876712
[4]  {3,20}  0.02876712
[5]  {31,34} 0.02876712
[6]  {8,39}  0.02876712
[7]  {33,40} 0.02876712
[8]  {27,40} 0.02876712
```

Find 3 Set of Rules

- {19,25,28} 0.006849315 is Maximum support 3 Set

```
In [20]: inspect(sort(rules[rules@quality$support >= 0.005 & rules@quality$support <= 0.00685], by = "support")[25:40])
```

```
     items       support
[1]  {24,43}     0.006849315
[2]  {31,39}     0.006849315
[3]  {19,25,28}  0.006849315
[4]  {15,28,34}  0.006849315
[5]  {4,28,40}   0.006849315
[6]  {10,16,41}  0.006849315
[7]  {34,42,45}  0.006849315
[8]  {5,18,42}   0.006849315
```

7 로또 선택 전략

로또 선택 전략

여러가지 번호를 선택하는 방향이 있을거라고 판단 된다.

**지지도 => 그 숫자 또는 숫자의 집합이 나올 확률 **

1. 개별의 숫자 6개 (지지도 상위 6개) 선택
2. 1 1 1 3 / 1 1 3 1 / 1 3 1 1 / 3 1 1 1
3. 1 1 2 2 / 1 2 1 2 / 1 2 2 1 / 2 1 2 1 / 2 1 1 2 / 2 2 1 1 과 같은 순으로 선택

1. 개별 지지도 상위 6개 선택

- 20,40,34,27,1,37
  - 확률 0.00001290....

In [44]:
```
# 개별 확률의 곱으로 전체 확률을 표현.
prob <- 1
for(i in 1:6){
    #print(paste(df[i,1]," ",df[i,2]))
    #print(df[i,1])
    prob <- prob * df[i,2]
    print(paste("cusum : ",prob))
}
print(prob)
```

```
[1] "cusum :  0.158904109589041"
[1] "cusum :  0.0248151623193845"
[1] "cusum :  0.00384125115354866"
[1] "cusum :  0.000584080654854644"
[1] "cusum :  8.72120429851455e-05"
[1] "cusum :  1.29026036197202e-05"
[1] 1.29026e-05
```

2. 개별 항목 3개, 3개 집합 1개 선택

- 최상위 개별 선택 3개 항목
  - {20} 0.1589041
  - {40} 0.1561644
  - {34} 0.1547945
- 3개 항목 최상위 위의 숫자를 제외한
  - {19,25,28} 0.006849315
- 20,40,34,19,25,28
  - 확률 : 0.0000263

3. 3개의 집단 2개 선택

- {19,25,28} 0.006849315
- {10,16,41} 0.006849315
- 확률 : 0.000049

4 + 2 Set 조합

- {3,9,22,42} 0.001369863
- {20,35} 0.02876712
- 확률 : 0.000042

8 결론

로또 선택을 위한 데이터 분석 적용

◆ 데이터 분석을 통한 로또 선택 과연 ?

1. 개별 항목이 나올 확률은 비슷하게 나온다.
   - 즉, 특정 숫자가 많이 나오지는 않는다.
2. 개별 항목이 높게 나타나더라도 조합이 된다면 2,3개 이상의 조합으로 숫자 선택하는 것이 높은 확률이 된다.
   - 앞서 본 것과 마찬가지로 3x3 조합이 가장 높다.

로또

수치와 데이터를 통한 로또 구매는 어리석은 일이며, 데이터를 아는 사람이라면 하지 않는 것이 정답이라고 생각됩니다. 운과 행운, 즐거움이 목적이라면 재미삼아 즐기시길 바랍니다.

감사합니다.