Graphic Analysis of Lottery Data

coop711

2015년 3월 14일

Lottery Data Input

- Pick-it 1976 New Jersey Lottery Game

```
lottery<-read.table("lottery.txt",header=T)
str(lottery)
```

```
## 'data.frame':    254 obs. of  2 variables:
##  $ lottery.number: int  810 156 140 542 507 972 431 981 865 499 ...
##  $ lottery.payoff: num  190 120 286 184 384 ...
```

```
head(lottery)
```

```
##   lottery.number lottery.payoff
## 1            810          190.0
## 2            156          120.5
## 3            140          285.5
## 4            542          184.0
## 5            507          384.5
## 6            972          324.5
```

- 기초통계량을 계산하고 lottery.number의 경우 이론값과 비교. 이론값이라 함은?

```
summary(lottery)
```

```
##  lottery.number  lottery.payoff
##  Min.   :  0.0   Min.   : 83.0
##  1st Qu.:230.0   1st Qu.:194.2
##  Median :440.5   Median :270.2
##  Mean   :472.2   Mean   :290.4
##  3rd Qu.:734.5   3rd Qu.:364.0
##  Max.   :999.0   Max.   :869.5
```

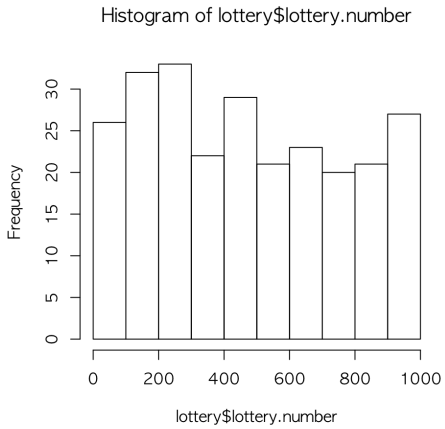
```
apply(lottery, 2, sd)
```

```
## lottery.number lottery.payoff
##       294.4773       128.8884
```

Graphic Analysis on lottery.number

- 당첨번호는 0(사실상 000)에서 999 사이에 254회 추출한 랜덤표본으로 볼 수 있음. 이를 확인하기 위하여 히스토그램 작성.

```
h10<-hist(lottery$lottery.number)
```



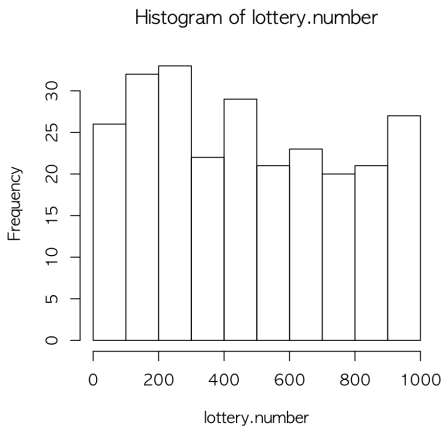
- 메인 타이틀과 x축의 좌표이름, y축의 좌표이름의 디폴트값이 어떻게 주어지는지 살피고, 히스토그램 작성에 계산된 값들 확인(특히 $breaks, $counts, $density 유의)

```
h10
```

```
## $breaks
##  [1]    0  100  200  300  400  500  600  700  800  900 1000
##
## $counts
##  [1] 26 32 33 22 29 21 23 20 21 27
##
## $density
##  [1] 0.0010236220 0.0012598425 0.0012992126 0.0008661417 0.0011417323
##  [6] 0.0008267717 0.0009055118 0.0007874016 0.0008267717 0.0010629921
##
## $mids
##  [1]  50 150 250 350 450 550 650 750 850 950
##
## $xname
## [1] "lottery$lottery.number"
##
## $equidist
## [1] TRUE
##
## attr(,"class")
## [1] "histogram"
```

- 각 계급의 경계선에 있는 관찰값들을 어떻게 처리하는 지 몇 가지 조건을 바꿔가면서 관찰. right=F로 인하여 $counts가 어떻게 변하였는가? attach()의 역할은 무엇인가?. (작업을 끝내기 전에 반드시 detach()할 것). 실제 취하는 값을 1000에서 999로 바꿨을 때, include.lowest=F로 했을 때 어떤 일이 일어나는지 확인하고 이유를 생각해 볼 것.

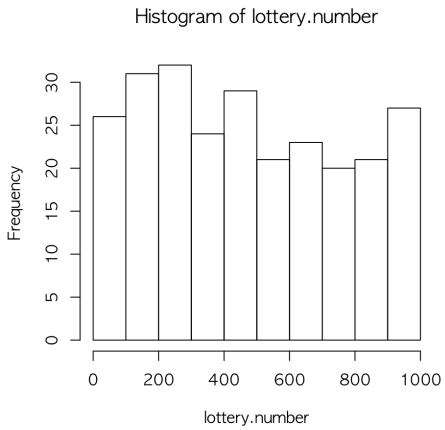
```
attach(lottery)
h10.2<-hist(lottery.number, breaks=seq(0,1000,by=100),include.lowest=T)
```




```
h10.2
```

```
## $breaks
##  [1]    0  100  200  300  400  500  600  700  800  900 1000
##
## $counts
##  [1] 26 32 33 22 29 21 23 20 21 27
##
## $density
##  [1] 0.0010236220 0.0012598425 0.0012992126 0.0008661417 0.0011417323
##  [6] 0.0008267717 0.0009055118 0.0007874016 0.0008267717 0.0010629921
##
## $mids
##  [1]  50 150 250 350 450 550 650 750 850 950
##
## $xname
## [1] "lottery.number"
##
## $equidist
## [1] TRUE
##
## attr(,"class")
## [1] "histogram"
```

```
h10.3<-hist(lottery.number, breaks=seq(0,1000,by=100),right=F)
```

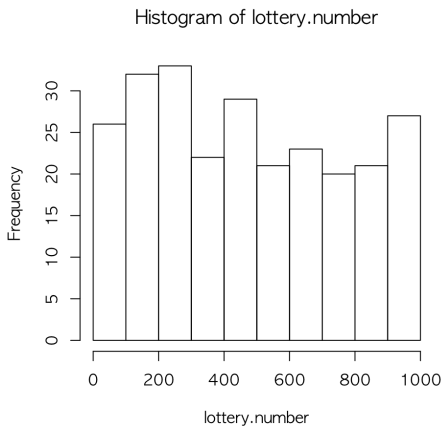



```
h10.3
```

```
## $breaks
##  [1]    0  100  200  300  400  500  600  700  800  900 1000
##
## $counts
##  [1] 26 31 32 24 29 21 23 20 21 27
##
## $density
##  [1] 0.0010236220 0.0012204724 0.0012598425 0.0009448819 0.0011417323
##  [6] 0.0008267717 0.0009055118 0.0007874016 0.0008267717 0.0010629921
##
## $mids
##  [1]  50 150 250 350 450 550 650 750 850 950
##
## $xname
## [1] "lottery.number"
##
## $equidist
## [1] TRUE
##
## attr(,"class")
## [1] "histogram"
```

- breaks 대신 nclass=10 을 사용하였을 때 결과 비교.

```
h10.4<-hist(lottery.number, nclass=10)
```

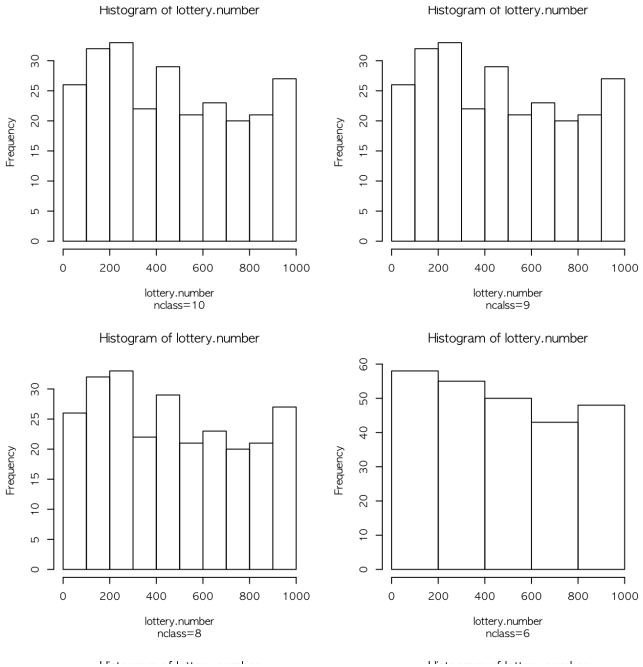


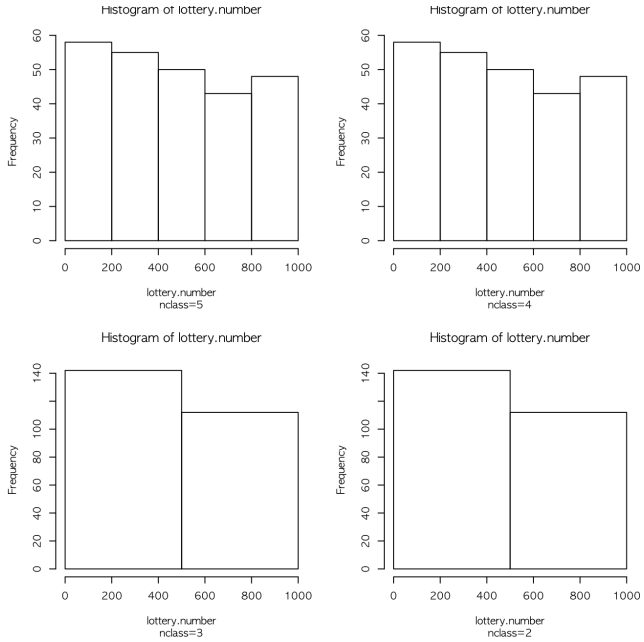

```
h10.4
```

```
## $breaks
##  [1]    0  100  200  300  400  500  600  700  800  900 1000
##
## $counts
##  [1] 26 32 33 22 29 21 23 20 21 27
##
## $density
##  [1] 0.0010236220 0.0012598425 0.0012992126 0.0008661417 0.0011417323
##  [6] 0.0008267717 0.0009055118 0.0007874016 0.0008267717 0.0010629921
##
## $mids
##  [1]  50 150 250 350 450 550 650 750 850 950
##
## $xname
## [1] "lottery.number"
##
## $equidist
## [1] TRUE
##
## attr(,"class")
## [1] "histogram"
```

- 다양한 nclass 값에 대하여 히스토그램 작성. nclass로 요구했을 때 제대로 잘 작동하는가 확인.

```
opar<-par(no.readonly=TRUE)
par(mfrow=c(4,2))
hist(lottery.number, nclass=10, sub="nclass=10")
hist(lottery.number, nclass=9, sub="ncalss=9")
hist(lottery.number, nclass=8, sub="nclass=8")
hist(lottery.number, nclass=6, sub="nclass=6")
hist(lottery.number, nclass=5, sub="nclass=5")
hist(lottery.number, nclass=4, sub="nclass=4")
hist(lottery.number, nclass=3, sub="nclass=3")
hist(lottery.number, nclass=2, sub="nclass=2")
```



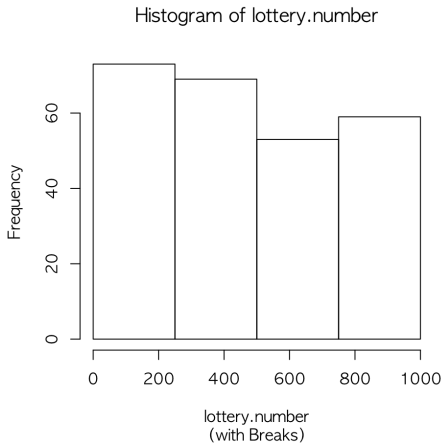


- nclass=9, 8은 모두 nclass=10과 같고, nclass=6은 nclass=5와 같으며, nclass=4, nclass=3 인 경우도 주문과 다르게 나온 점에 유의하고 일부 계산값 확인. argument 중에 sub="nclass=3" 을 놓아 둔채 plot=F 를 하면 어떻게 되는지 시험해 보시오.

```
par(mfrow=c(1,2))
h4<-hist(lottery.number, nclass=4, plot=F)
h4
```

```
## $breaks
## [1]    0  200  400  600  800 1000
##
## $counts
## [1] 58 55 50 43 48
##
## $density
## [1] 0.0011417323 0.0010826772 0.0009842520 0.0008464567 0.0009448819
##
## $mids
## [1] 100 300 500 700 900
##
## $xname
## [1] "lottery.number"
##
## $equidist
## [1] TRUE
##
## attr(,"class")
## [1] "histogram"
```

```
h3<-hist(lottery.number, nclass=3, plot=F)
h3
```




```
h4.breaks
```

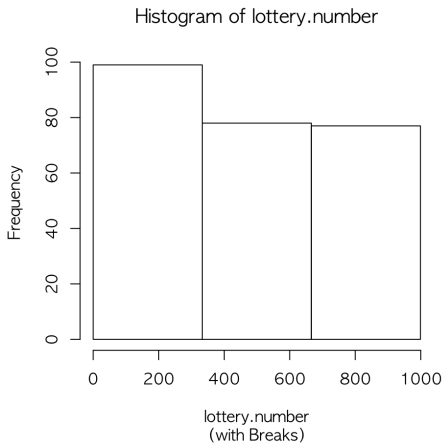
```
## $breaks
## [1]    0  500 1000
##
## $counts
## [1] 142 112
##
## $density
## [1] 0.0011181102 0.0008818898
##
## $mids
## [1] 250 750
##
## $xname
## [1] "lottery.number"
##
## $equidist
## [1] TRUE
##
## attr(,"class")
## [1] "histogram"
```

- nclass=4, nclass=3 을 그리려면 breaks 조정. breaks 가 보다 확실한 방법.

```
h4.breaks<-hist(lottery.number, breaks=seq(0,1000, by=250), sub="(with Breaks)")
```

```
## $breaks
## [1]    0  250  500  750 1000
##
## $counts
## [1] 73 69 53 59
##
## $density
## [1] 0.0011496063 0.0010866142 0.0008346457 0.0009291339
##
## $mids
## [1] 125 375 625 875
##
## $xname
## [1] "lottery.number"
##
## $equidist
## [1] TRUE
##
## attr(,"class")
## [1] "histogram"
```

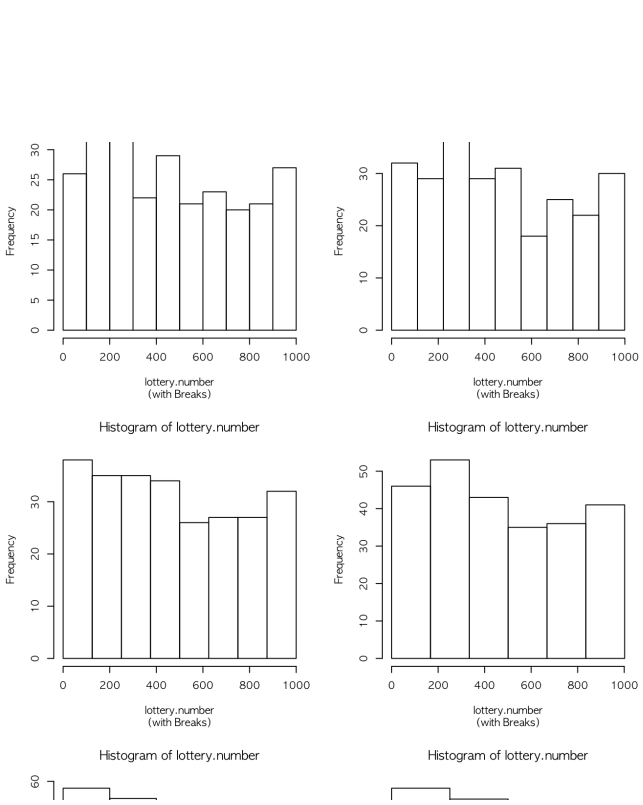
```
h3.breaks<-hist(lottery.number, breaks=seq(0,999, by=333), sub="(with Breaks)")
```

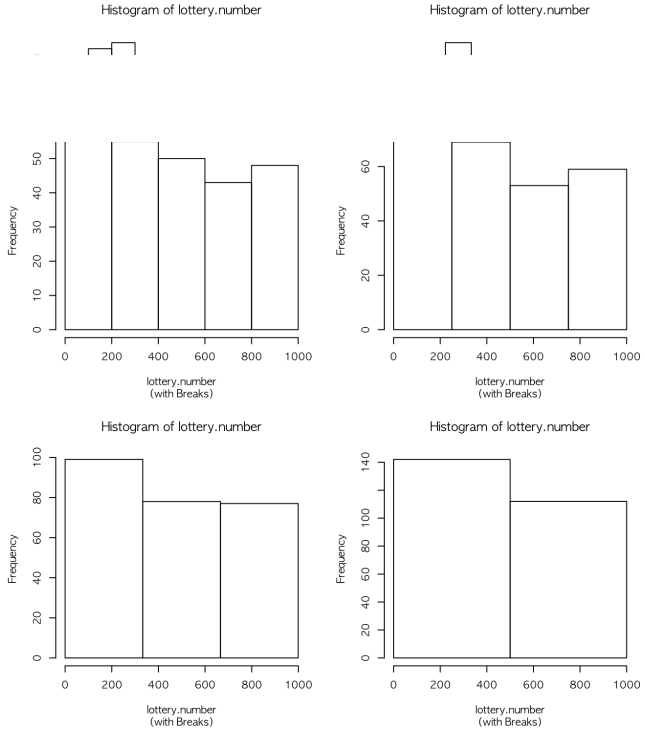



```
h3.breaks
```

```
## $breaks
## [1]   0 333 666 999
##
## $counts
## [1] 99 78 77
##
## $density
## [1] 0.0011704618 0.0009221820 0.0009103592
##
## $mids
## [1] 166.5 499.5 832.5
##
## $xname
## [1] "lottery.number"
##
## $equidist
## [1] TRUE
##
## attr(,"class")
## [1] "histogram"
```

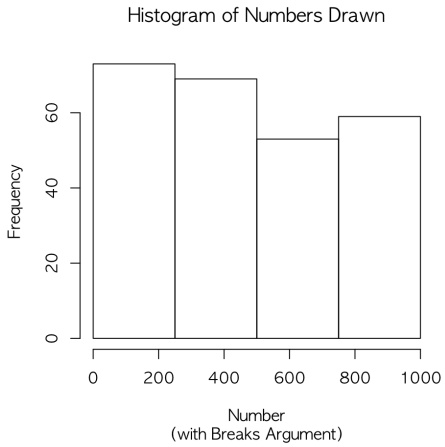
```
opar<-par(no.readonly=TRUE)
par(mfrow=c(4,2))
hist(lottery.number, breaks=seq(0,1000, by=100), sub="(with Breaks)")
hist(lottery.number, breaks=seq(0,999, by=111), sub="(with Breaks)")
hist(lottery.number, breaks=seq(0,1000, by=125), sub="(with Breaks)")
hist(lottery.number, breaks=seq(0,1002, by=167), sub="(with Breaks)")
hist(lottery.number, breaks=seq(0,1000, by=200), sub="(with Breaks)")
hist(lottery.number, breaks=seq(0,1000, by=250), sub="(with Breaks)")
hist(lottery.number, breaks=seq(0,999, by=333), sub="(with Breaks)")
hist(lottery.number, breaks=seq(0,1000, by=500), sub="(with Breaks)")
```





- 히스토그램의 정보를 보다 알기 쉽게 타이틀과 좌표명을 손보려면 ann=F 사용. 다른 히스토그램들에도 적용해 볼 것.

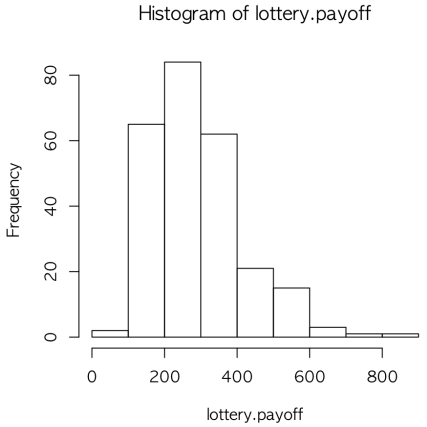
```
par(mfrow=c(1,1))
hist(lottery.number, breaks=seq(0,1000,by=250),ann=F)
title(main="Histogram of Numbers Drawn", sub="(with Breaks Argument)", xlab="Number", ylab="Frequency")
```



Distribution of lottery.payoff

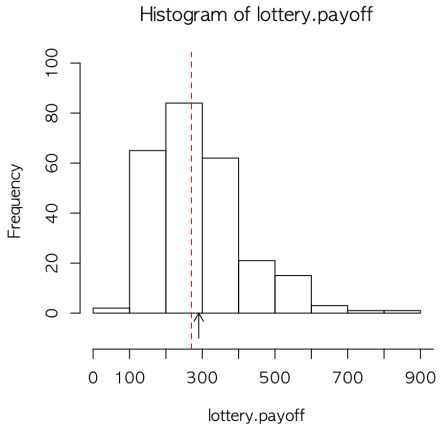
- 이제 당첨번호와 당첨금액과의 관계를 살피기 전에 잠깐 당첨번호의 분포를 살펴보면

```
hist(lottery.payoff)
```



- 평균과 중앙값을 계산하여 화살표와 점선으로 표시하면 다음과 같이 할 수 있는데, 어느 것이 평균이고, 어느 것이 중앙값인가?

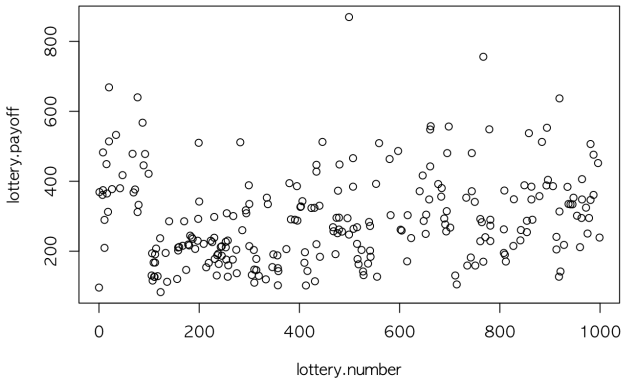
```
mean.payoff<-mean(lottery.payoff)
med.payoff<-median(lottery.payoff)
hist(lottery.payoff,axes=F,ylim=c(-10,100))
axis(side=1,at=seq(0,1000,by=100),labels=paste(seq(0,1000,by=100)))
arrows(x0=mean.payoff,y0=-10, x1=mean.payoff, y1=0, length=0.1, code=2)
abline(v=med.payoff,lty=2,col="red")
axis(side=2,at=seq(0,100,by=20),labels=paste(seq(0,100,by=20)))
```



The Relationship between lottery.number and lottery.payoff

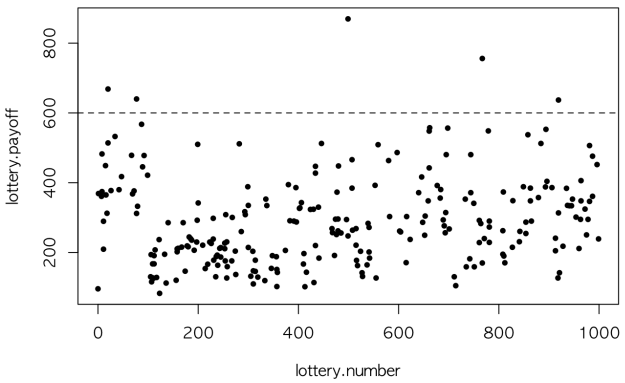
- 이제 두 변수의 산점도를 그려보자.

```
plot(lottery.number, lottery.payoff)
```



- 점의 모양을 바꾸고, 당첨금액이 600불 이상인 당첨번호들을 찾기 위하여 identify() 함수를 이용하면 마우스로 직접 찾을 수 있으나 r markdown 에서는 작동하지 않음.

```
plot(lottery.number, lottery.payoff,pch=20)
abline(h=600,lty=2)
identify(lottery.number, lottery.payoff, n=5, labels=paste(lottery.number))
```




```
## integer(0)
```

- which() 함수와 subscripting([ ]) 을 이용하여 찾아보면

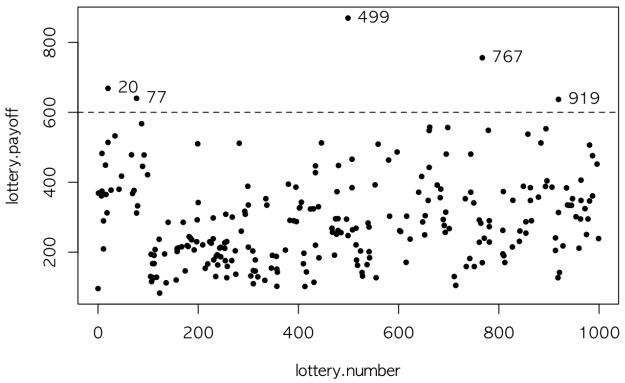
```
high.payoff<-which(lottery.payoff>=600)
high.payoff
```

```
## [1]  10  11  95 107 215
```

```
lottery.number[high.payoff]
```

```
## [1] 499  20  77 767 919
```

```
plot(lottery.number, lottery.payoff,pch=20)
abline(h=600,lty=2)
text(x=lottery.number[high.payoff],y=lottery.payoff[high.payoff],labels=lottery.number[high.payoff],pos=4)
```



- 당첨금액 상위 10위까지의 당첨번호를 살펴보면

```
o.payoff<-order(lottery.payoff,decreasing=TRUE)
lottery.payoff[o.payoff][1:10]
```

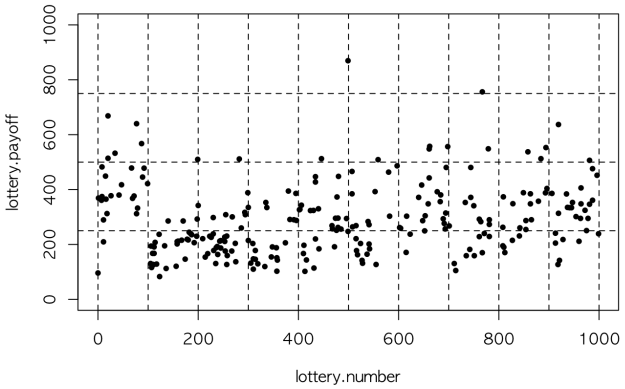
```
##  [1] 869.5 756.0 668.5 640.0 637.0 567.5 557.5 556.5 553.0 548.5
```

```
lottery.number[o.payoff][1:10]
```

```
##  [1] 499 767  20  77 919  87 662 698 894 779
```

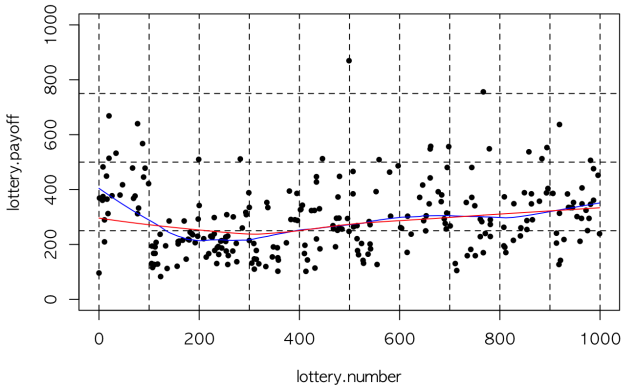
- 당첨번호와 당첨금액의 관계를 살피기 위하여 y축의 범위를 조정하고, 격자를 설치하면

```
plot(lottery.number, lottery.payoff,pch=20, ylim=c(0,1000))
abline(h=c(250,500,750),lty=2)
abline(v=seq(0,1000,by=100),lty=2)
```



- 흐름을 파악하기 위하여 local smoother 를 적용.

```
plot(lottery.number, lottery.payoff,pch=20, ylim=c(0,1000))
abline(h=c(250,500,750),lty=2)
abline(v=seq(0,1000,by=100),lty=2)
lines(lowess(lottery.number,lottery.payoff, f=1/3),col="blue")
lines(lowess(lottery.number,lottery.payoff, f=2/3),col="red")
```



- 이제 당첨금액이 높은 당첨번호들은 숫자가 중복되는 경우가 많고, 당첨번호가 0에서 100 이하인 경우에 당첨금액이 높은지 생각해 봅시다.

```
par(opar)
save(file="lottery.RData",list=ls())
```

```
savehistory("lottery.Rhistory")
```